Dominance is not a Tree

Towards More Precise Dominance Relations

George Stelle Tarun Prabhu Pat McCormick
Daniel Shevitz Joseph Donato

Los Alamos National Laboratory

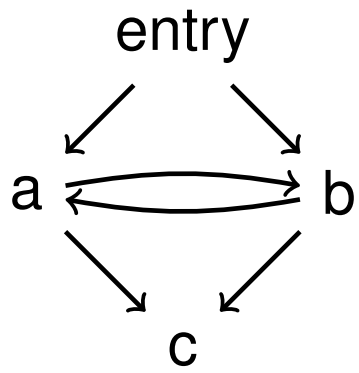
March 2, 2024

Outline

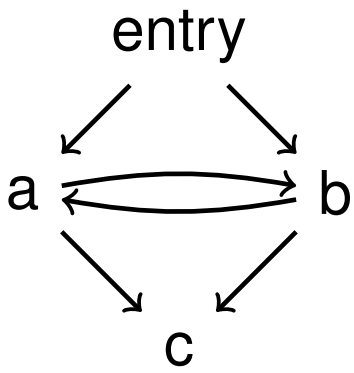
- Dominator Tree
- Non-Tree Dominance
- Dominator Grove
- Empirical Results
- The Good, The Bad, and The Ugly
- Formalization
- Concurrency
- Questions

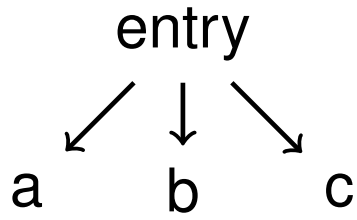
Control Flow Graph (CFG)

```
define f(cond){
  entry:
    br cond, a, b
  a:
    A
    br cond, b, c
  b:
    B
    br cond, c, a
  c:
    C
    ret
}
```



Dominator Tree

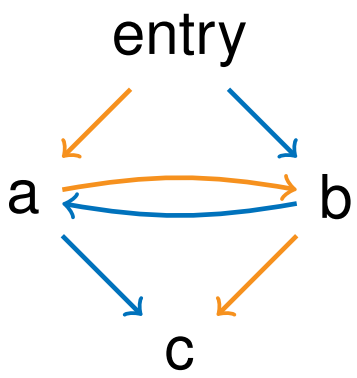




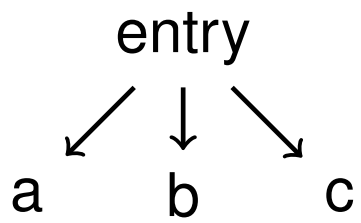
Dominator Tree

```
define f(cond){
  entry:
    br cond, a, b
  a:
    A
    br cond, b, c
  b:
    B
    br cond, c, a
  c:
    C
    ret
}
```

CFG



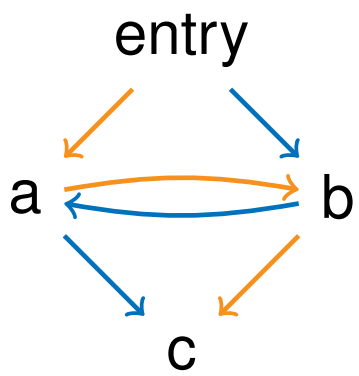
Dominator Tree



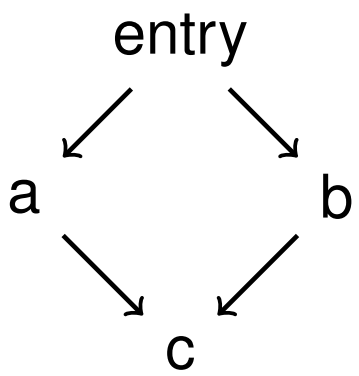
Dominance Partial Order

```
define f(cond){
  entry:
    br cond, a, b
  a:
    A
    br cond, b, c
  b:
    B
    br cond, c, a
  c:
    C
    ret
}
```

CFG



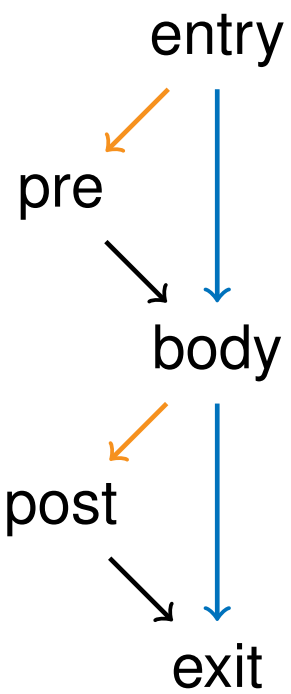
Dominance



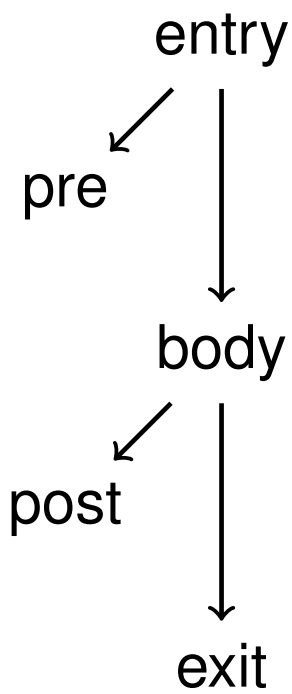
Dominator Tree

```
define f(cond){
  entry:
    br cond, pre, body
  pre:
    PRE
    br body
  body:
    BODY
    br cond, post, exit
  post:
    POST
    br exit
  exit:
    ret
}
```

CFG



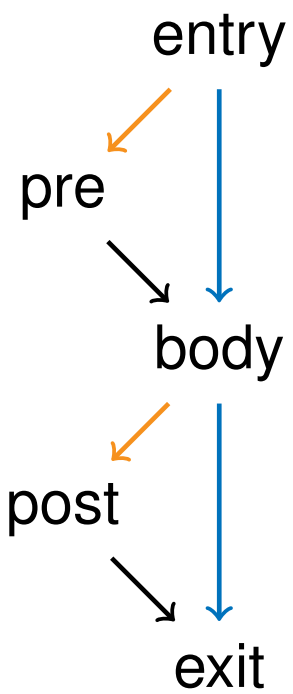
Dominator Tree



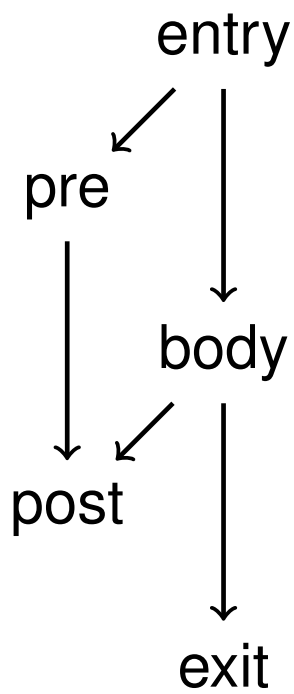
Dominance Partial Order

```
define f(cond){
  entry:
    br cond, pre, body
  pre:
    PRE
    br body
  body:
    BODY
    br cond, post, exit
  post:
    POST
    br exit
  exit:
    ret
}
```

CFG



Dominance



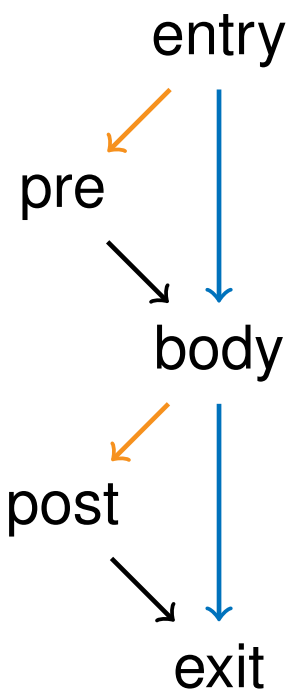
Dominator Grove

Idea: Use shared SSA condition variables to do case analysis-based dominance queries

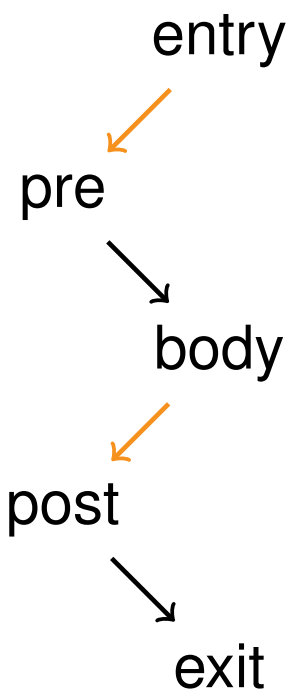
```
dominates(a, b, f){
  cvs <- findSharedConditionVariables(f)
  copses <- [[dominatorTree(f [ cv = true ]),
              dominatorTree(f [ cv = false ])]
              for cv in cvs]
  return any([all([t.dominates(a, b) for t in copse])
              for copse in copses])
}
```

Dominator Grove Example

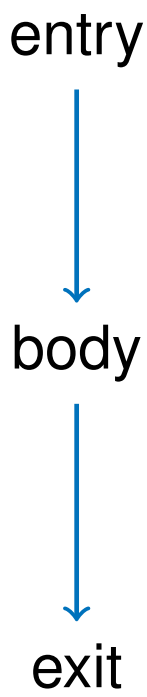
CFG



Cond=True



Cond=False



Empirical Observations

How often do non-tree domination relations occur in practice?

`llvm-test-suite` (imprecise measurements)

- $\approx$ 15% of LLVM modules
- $\approx$ 0.15% of total calls to `dominates`

The Good: licm-control-flow-hoisting

Input

```
define f(cond){
  entry:
    br loop
  loop:
    br cond, if,
        then
  if:
    inv = ...
    call f(inv)
    br then
  then:
    ...
    br ..., loop
}
```

DomTree

```
define f(cond){
  entry:
    inv = ...
    br loop
  loop:
    br cond, if,
        then
  if:
    call f(inv)
    br then
  then:
    ...
    br ..., loop
}
```

DomGrove

```
define f(cond){
  entry:
    br cond, pIf, loop
  pIf:
    inv = ...
    br loop
  loop:
    br cond, if, then
  if:
    call f(inv)
    br then
  then:
    ...
    br ..., loop
}
```

The Bad

- DomGrove updates/invalidation: Need changes on potentially all terminator instruction updates
- Unclear performance impact
- Iterated dominance frontier
- Transforms

The Ugly

Implicit assumptions of dominance tree structure

Example: Region Analysis

```
return (DT->dominates(entry, BB)
        !(DT->dominates(exit, BB)
          DT->dominates(entry, exit)));
```

Formalization

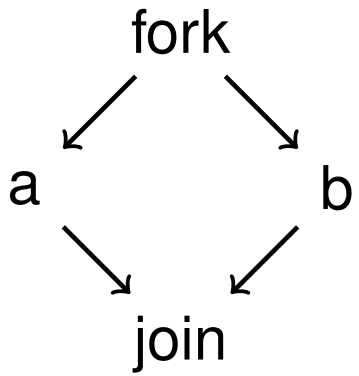
Valid Paths

$$\text{LLVM} \subseteq \text{Conditional CFG} \subseteq \text{CFG}$$

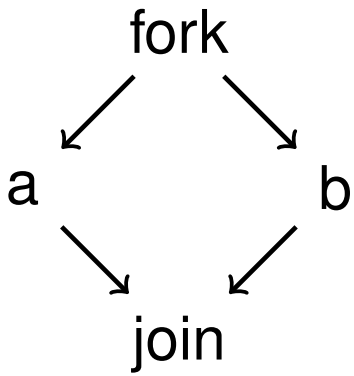
Dominance Relation

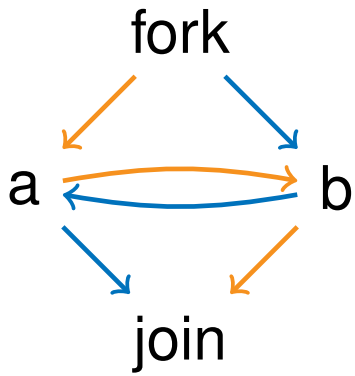
$$\text{CFG} \subseteq \text{Conditional CFG} \subseteq \text{LLVM}$$

Concurrency



Concurrency





Questions?

Answers?