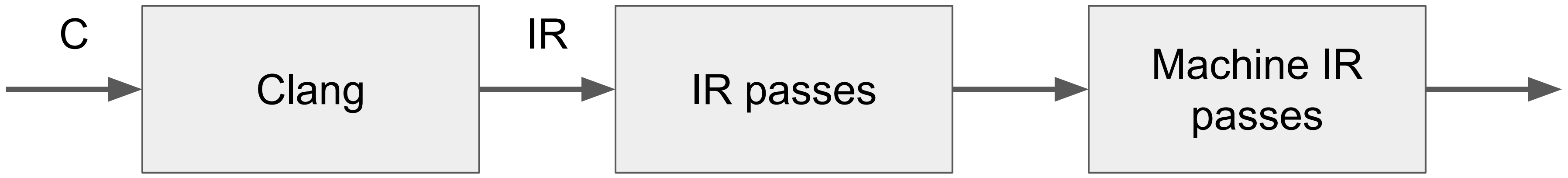
EuroLLVM 2016

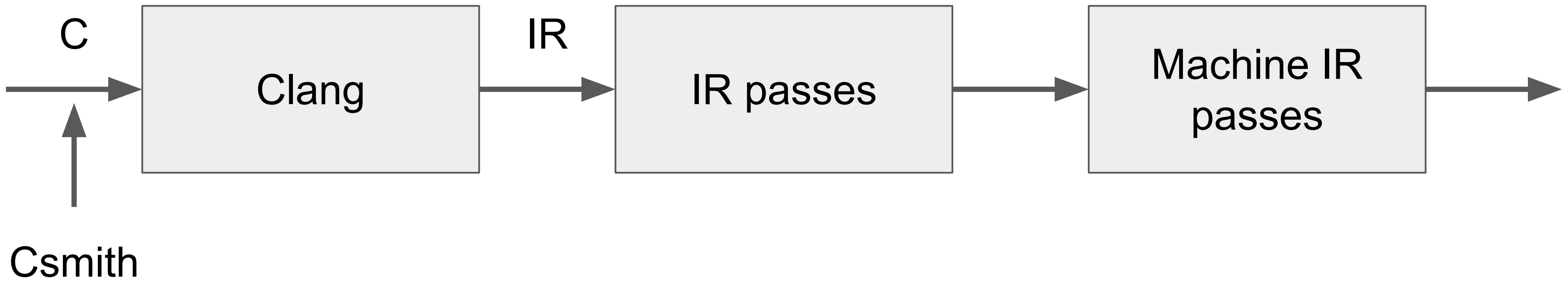
Random Testing of the LLVM Code Generator

Bevin Hansson, M. Sc.
Researcher at SICS Swedish ICT

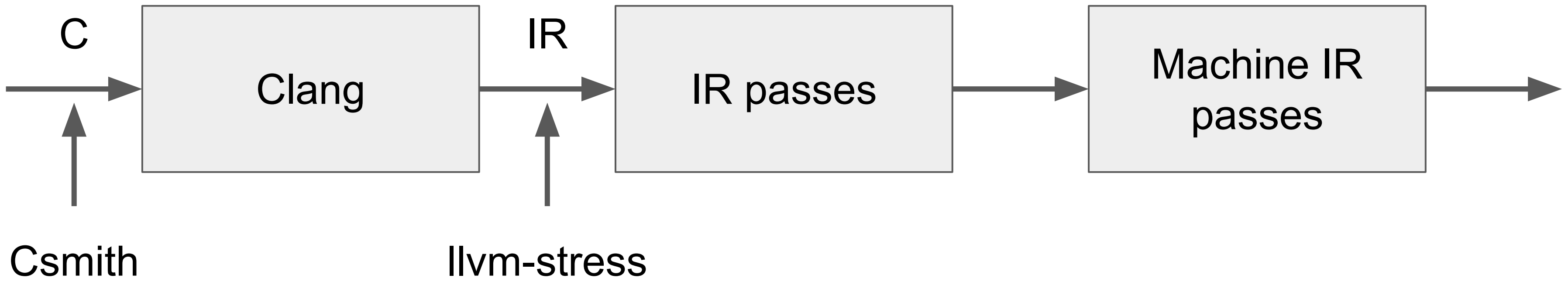
Random testing is useful



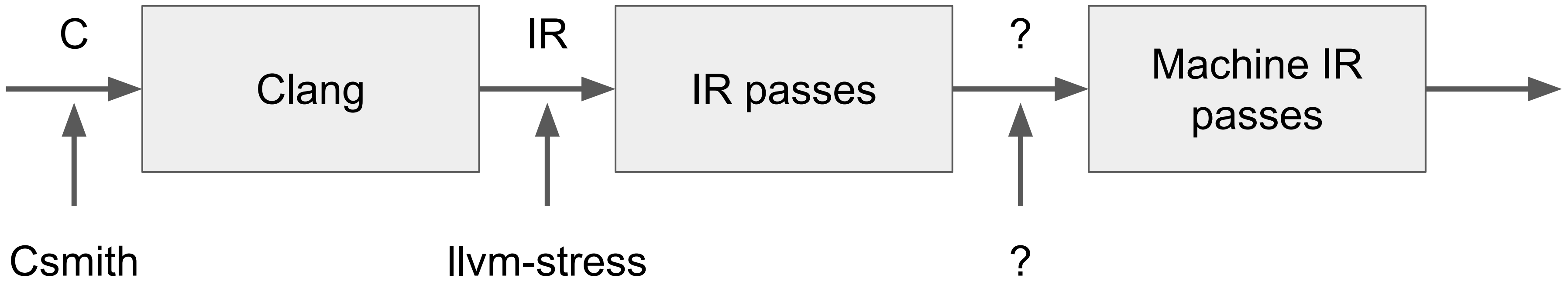
Random testing is useful



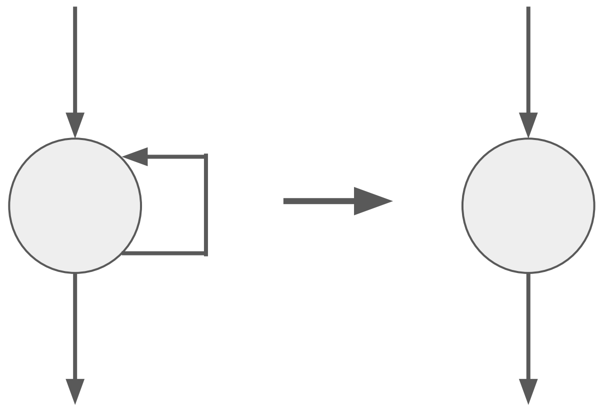
Random testing is useful

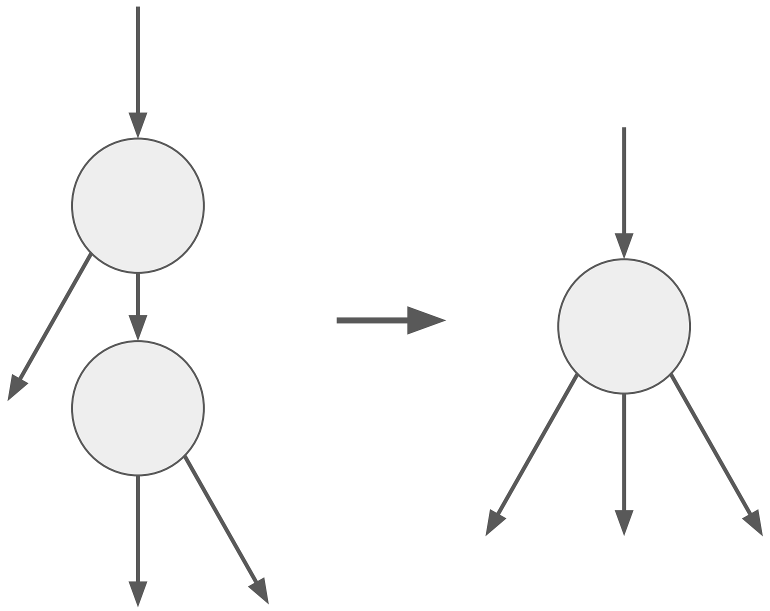


Random testing is useful

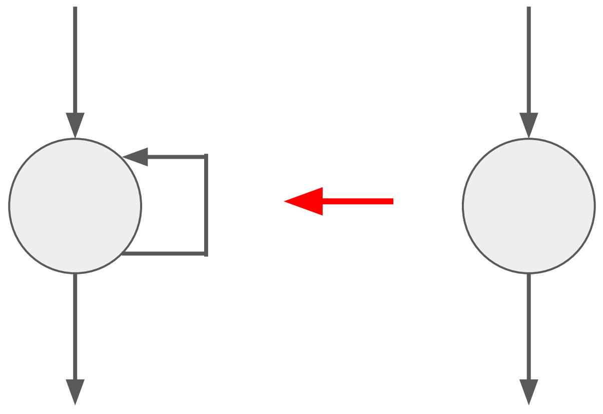


Reducibility

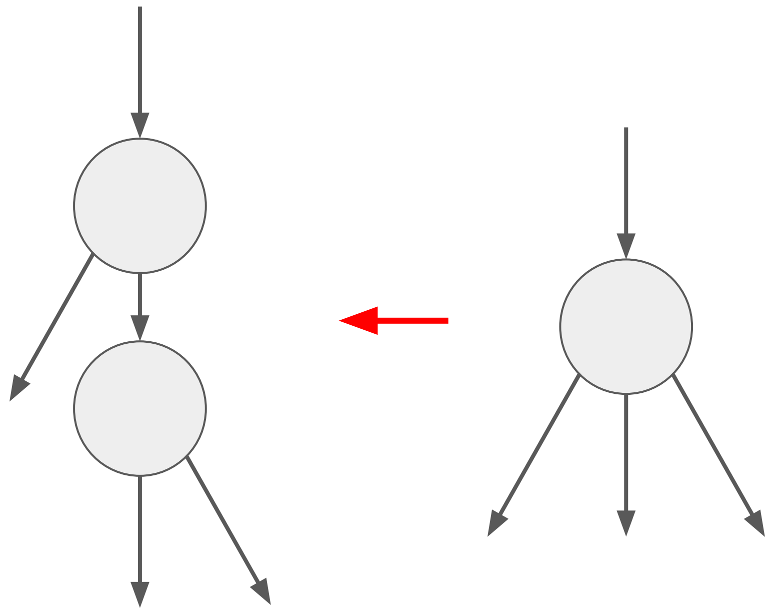

T1


T2

Inverse reducibility transform generation

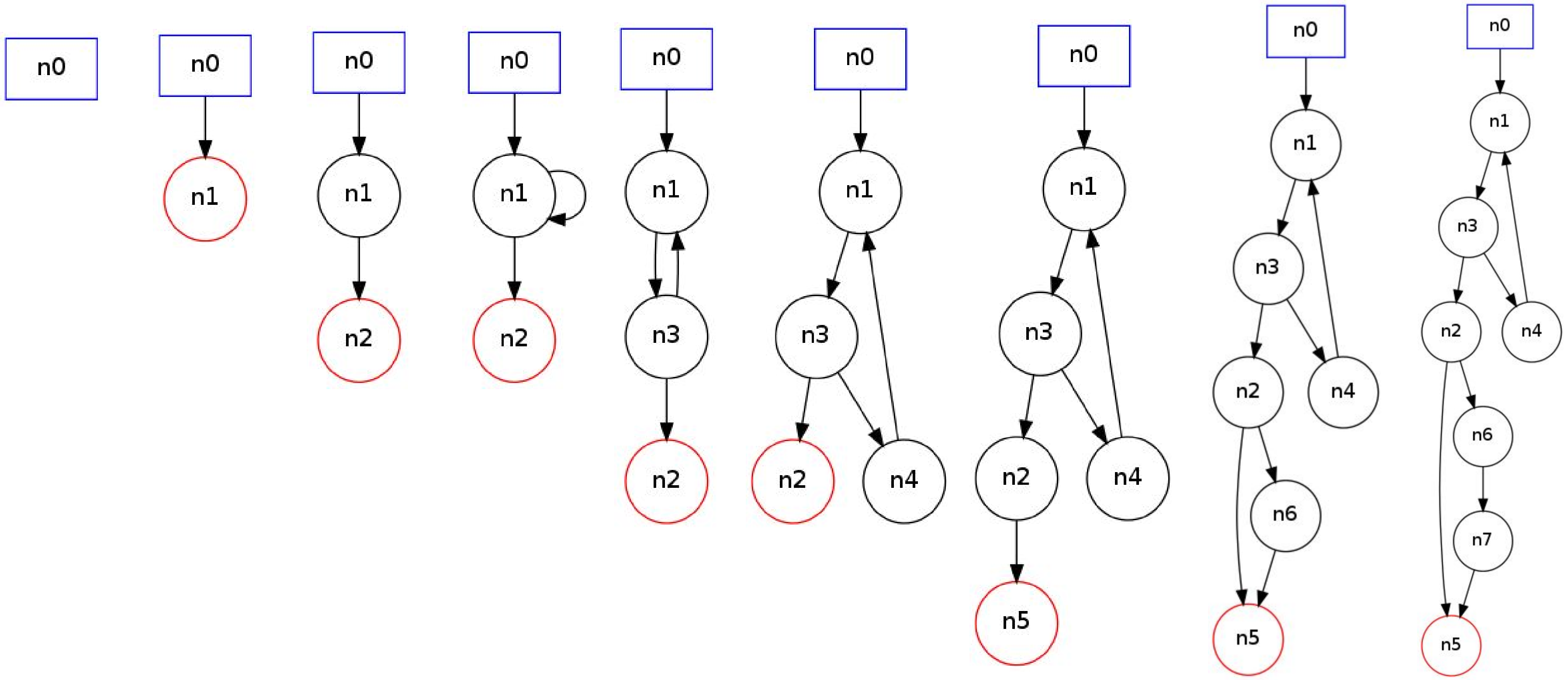


T1

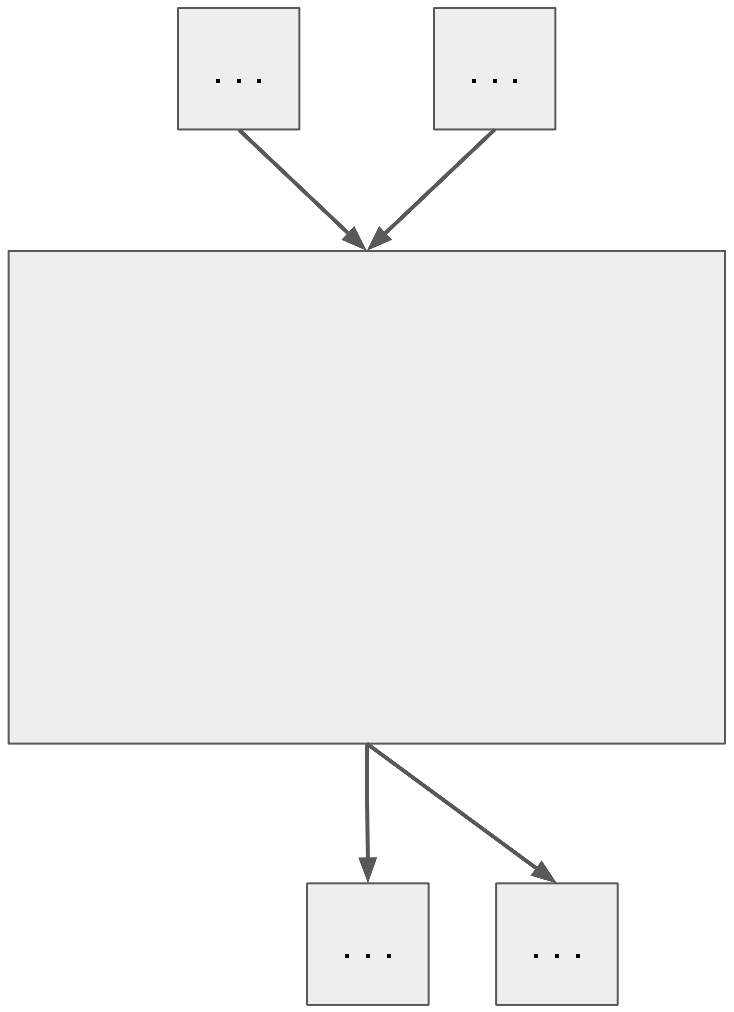


T2

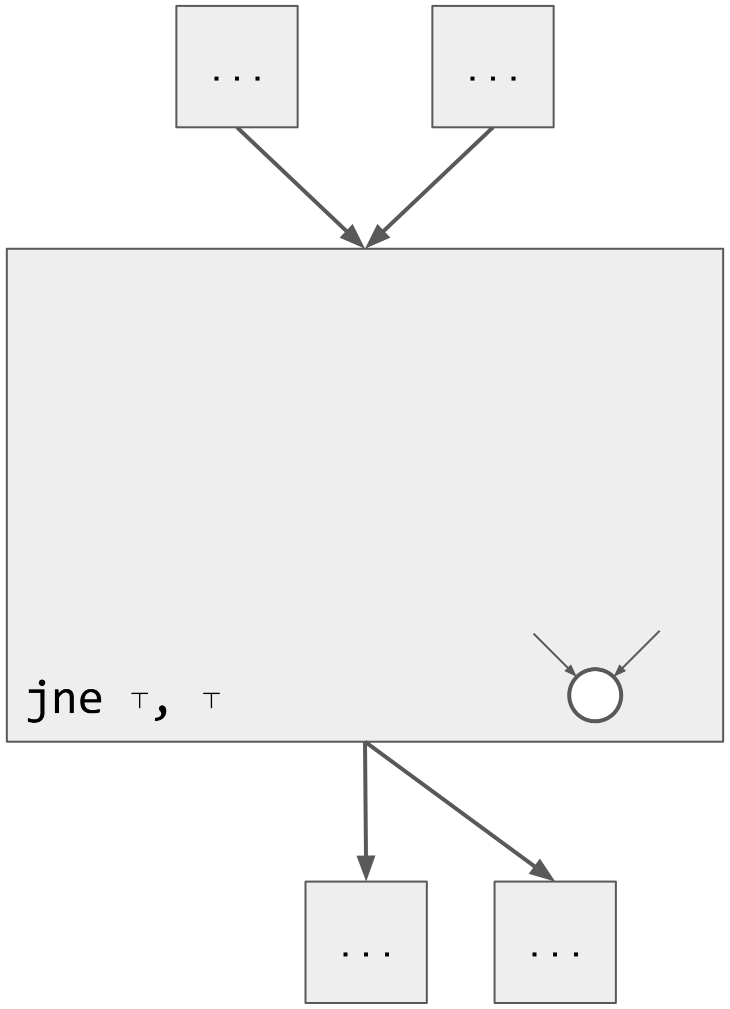
Inverse reducibility transform generation



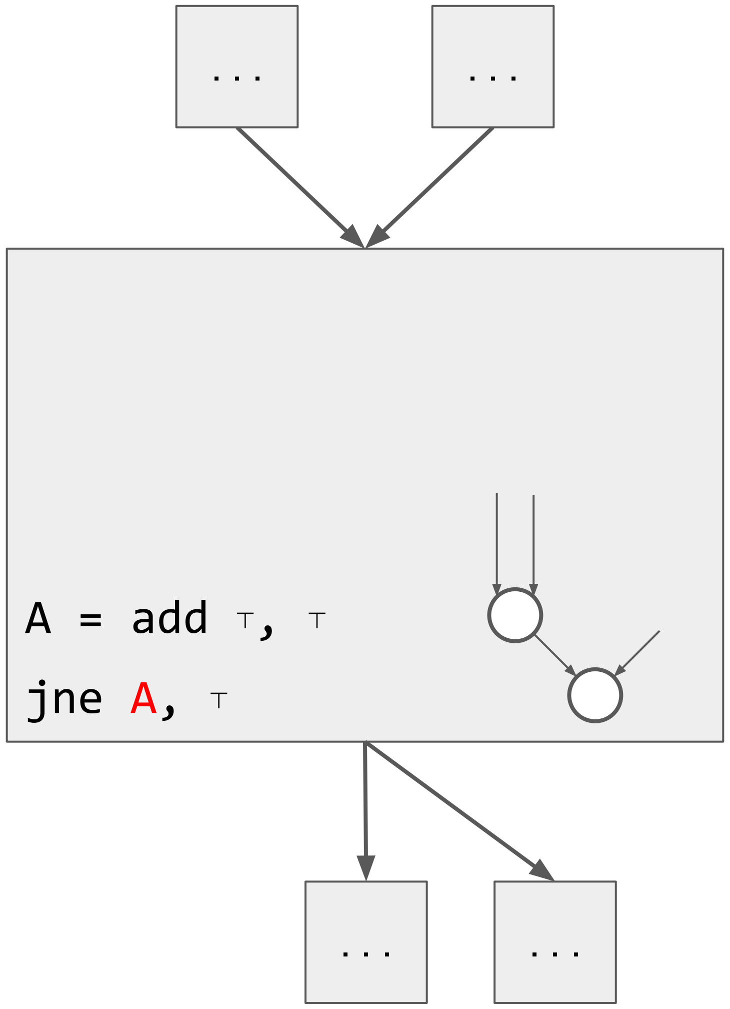
Fixedpoint bottom-up data flow generation



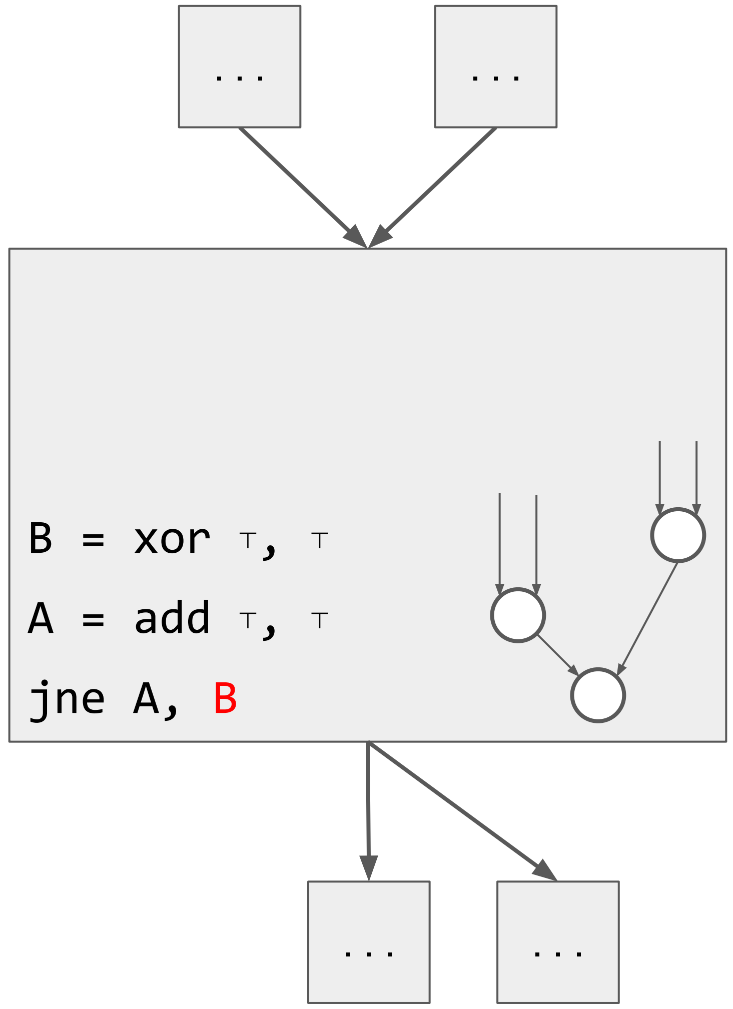
Fixedpoint bottom-up data flow generation



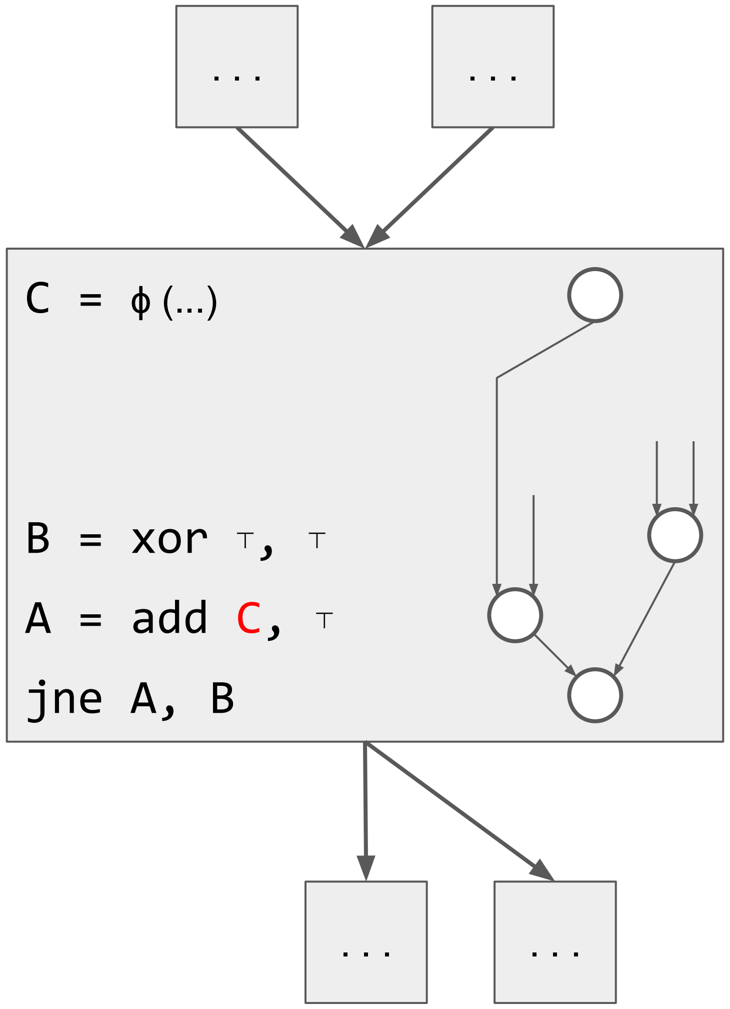
Fixedpoint bottom-up data flow generation



Fixedpoint bottom-up data flow generation

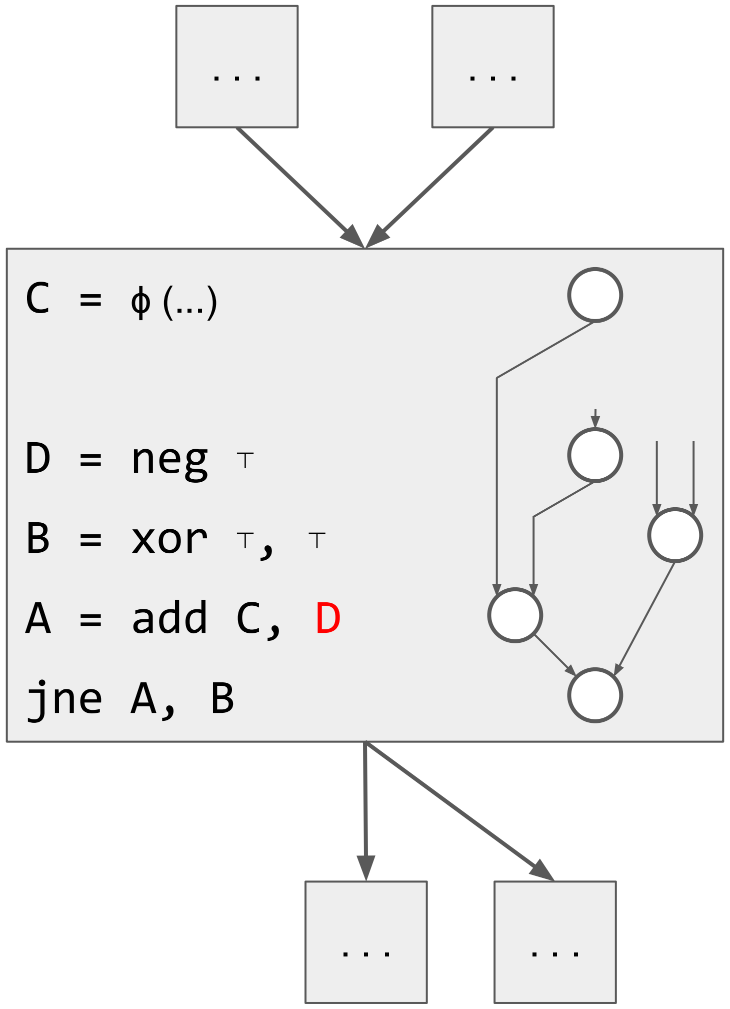


Fixedpoint bottom-up data flow generation



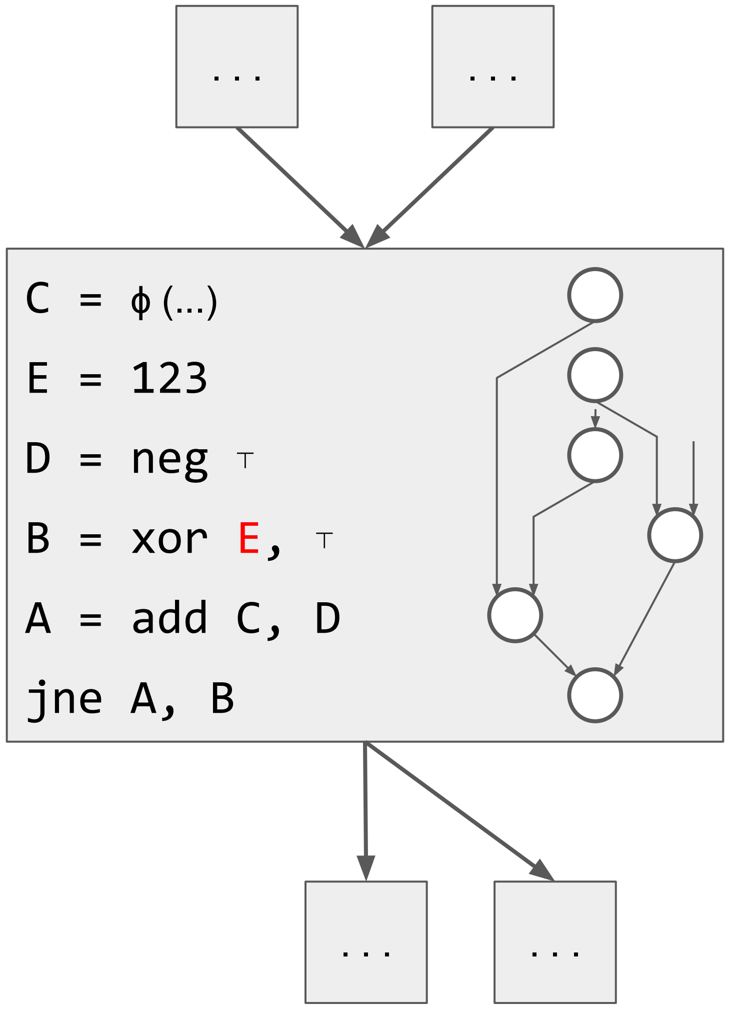
```
C = ϕ(…)



B = xor ⊤, ⊤
A = add C, ⊤
jne A, B
```

Fixedpoint bottom-up data flow generation



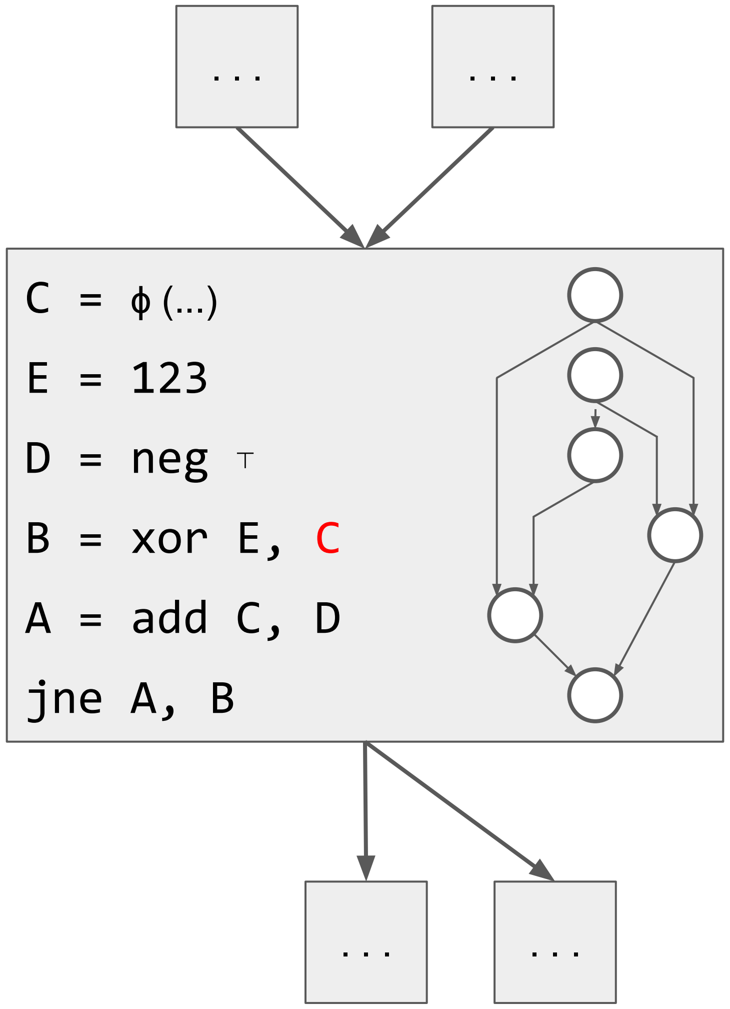
```
C = ϕ(…)


D = neg ⊤
B = xor ⊤, ⊤
A = add C, D
jne A, B
```

Fixedpoint bottom-up data flow generation



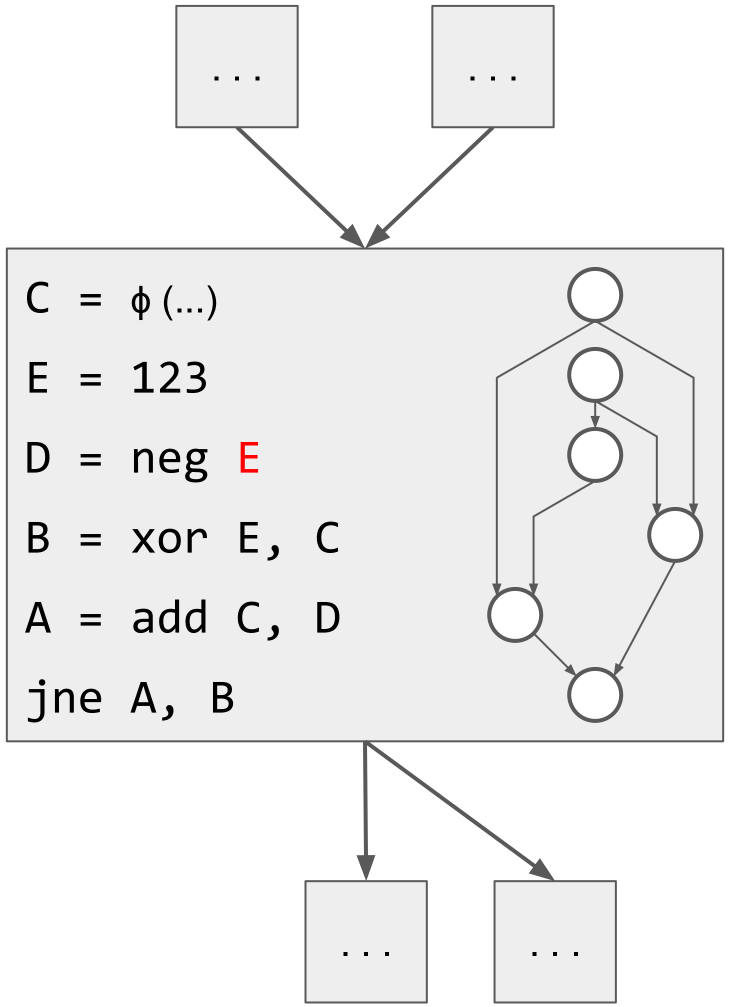
```
C = ϕ(...)
E = 123
D = neg ⊤
B = xor E, ⊤
A = add C, D
jne A, B
```

Fixedpoint bottom-up data flow generation



```
C = ϕ(...)
E = 123
D = neg ⊤
B = xor E, C
A = add C, D
jne A, B
```

Fixedpoint bottom-up data flow generation



```
C = ϕ(...)
E = 123
D = neg E
B = xor E, C
A = add C, D
jne A, B
```

Conclusion

- A textual representation of Machine IR was designed
- A Hexagon implementation of random generation was made
- Testing discovered some assertion failures, but no bugs
- Perhaps the new textual MIR implementation could yield better results
- Or: redesign llvm-stress with these methods in mind