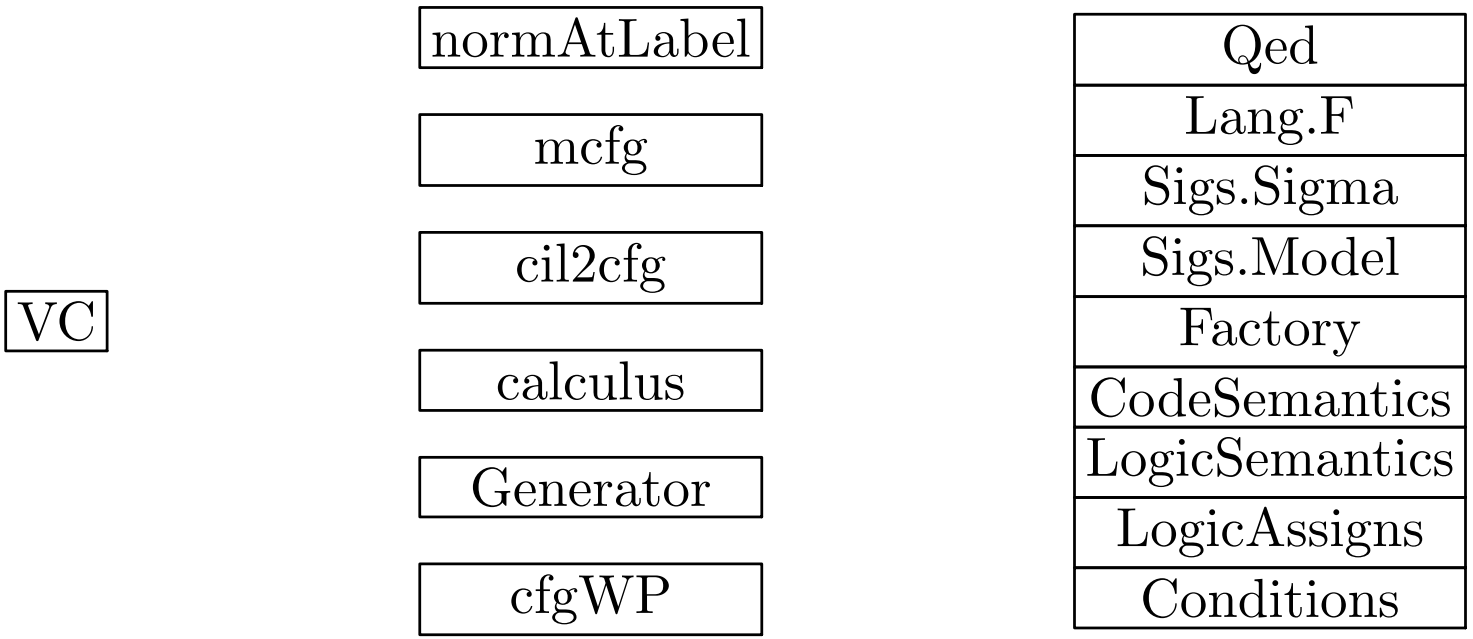
CFG meets Formulas

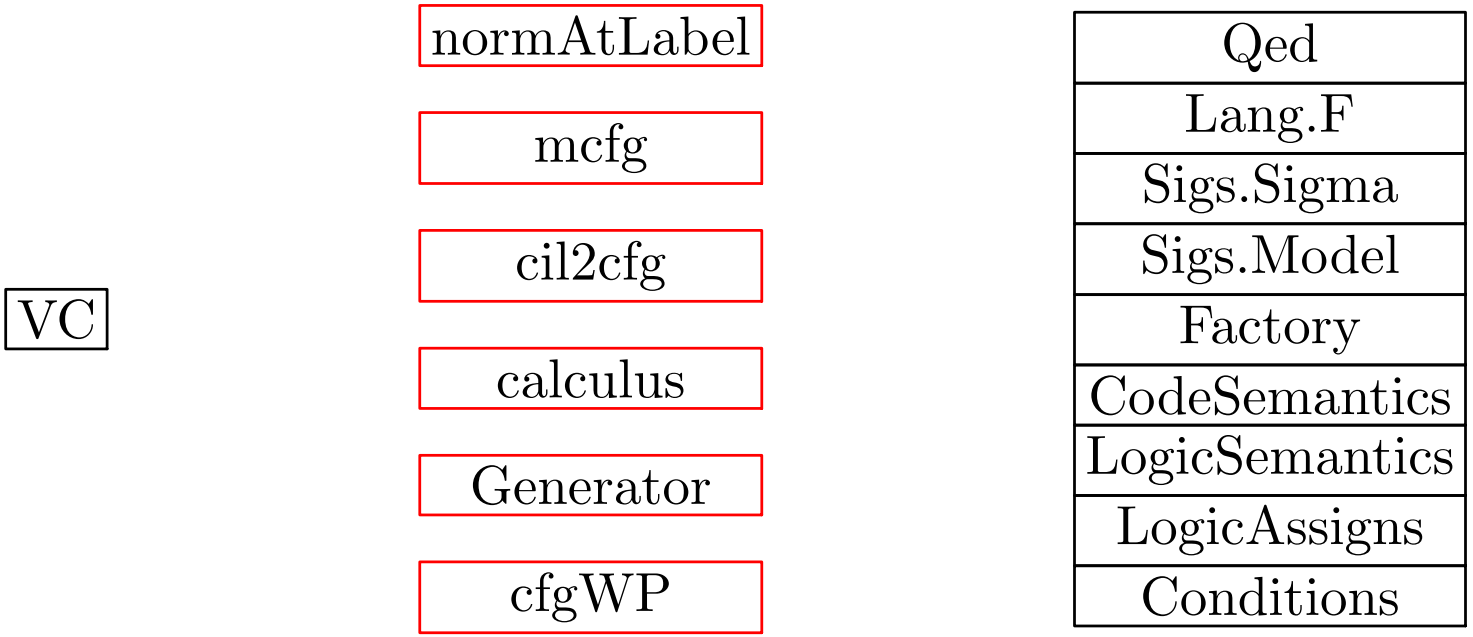
| F.Bobot, L. Correnson, Q. Bouillaguet, L. Blatter, V. Perrelle

Lubin!! 27/10/2018

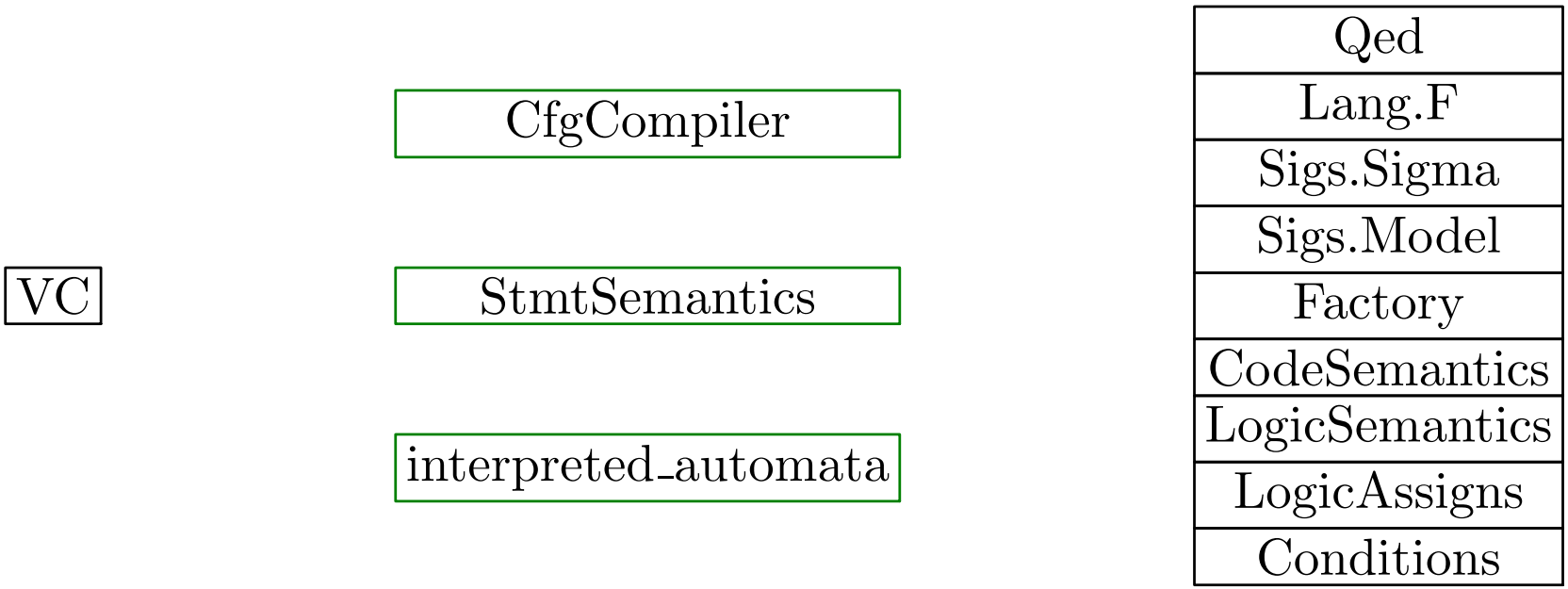
Architecture



Architecture



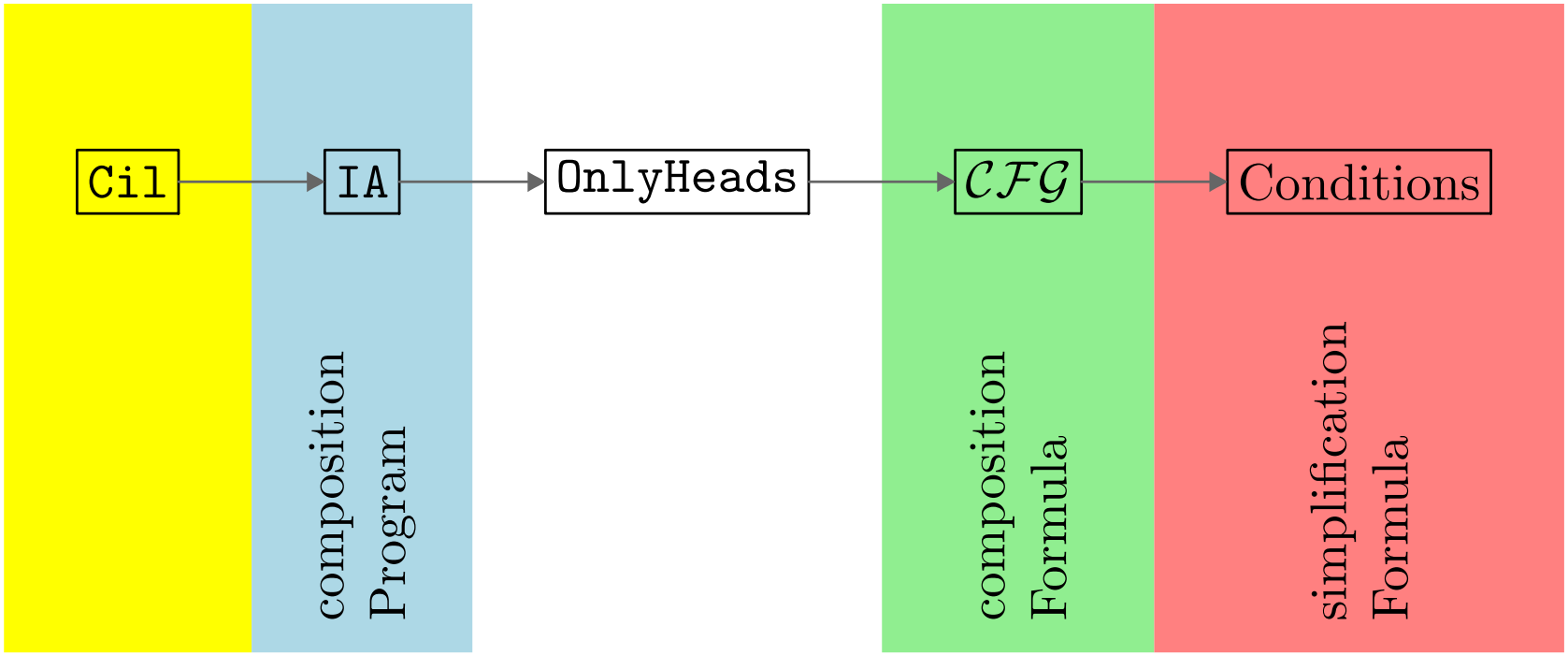
Architecture



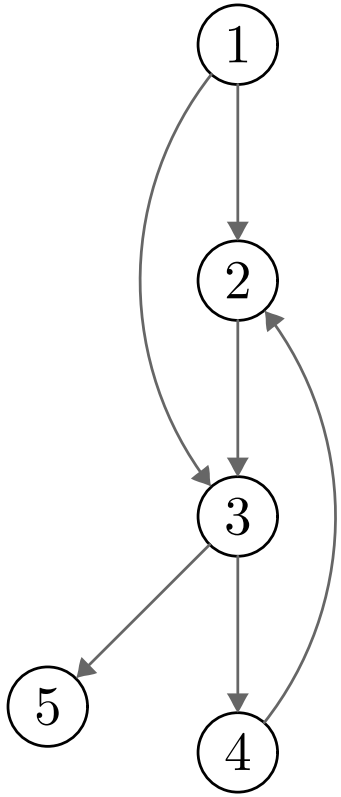
Why?

- Monolithic transformation
- No inlining
- No loop unrolling

Pipeline

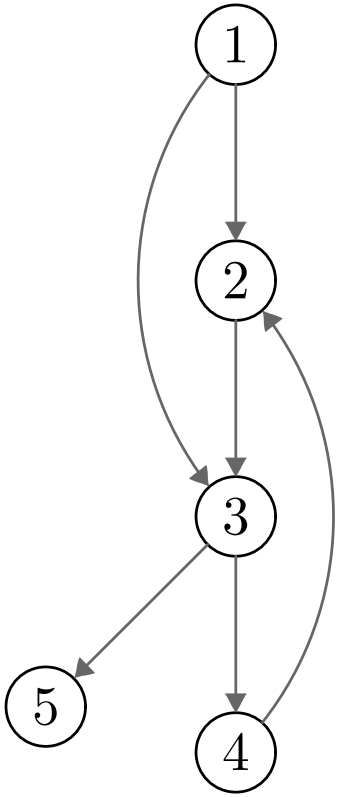

Cil
IA
OnlyHeads
$\mathcal{CFG}$
Conditions
composition
Program
composition
Formula
simplification
Formula

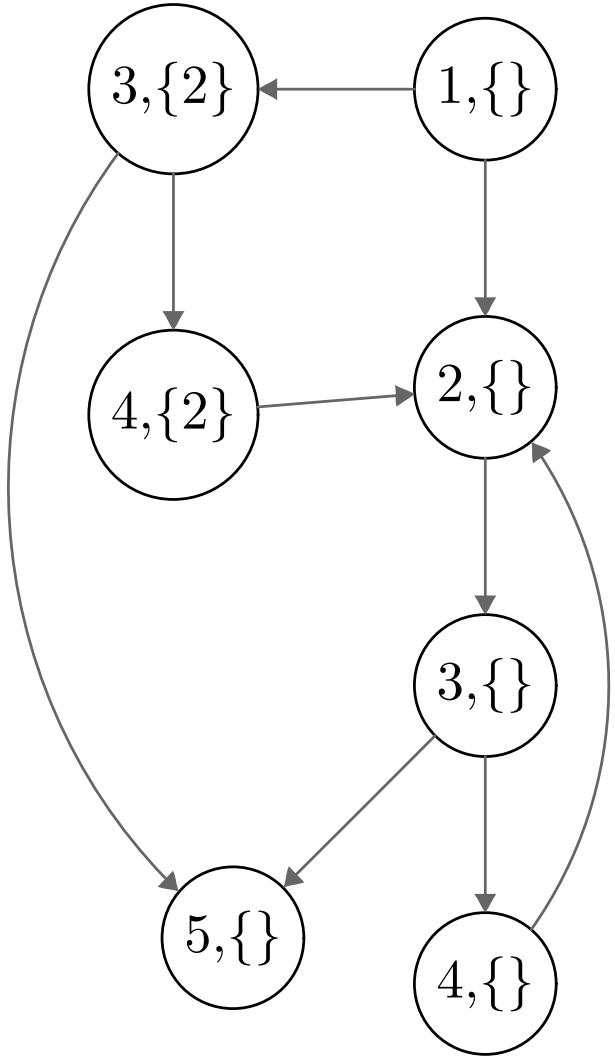
OnlyHeads



(1 (2 3 4) 5)

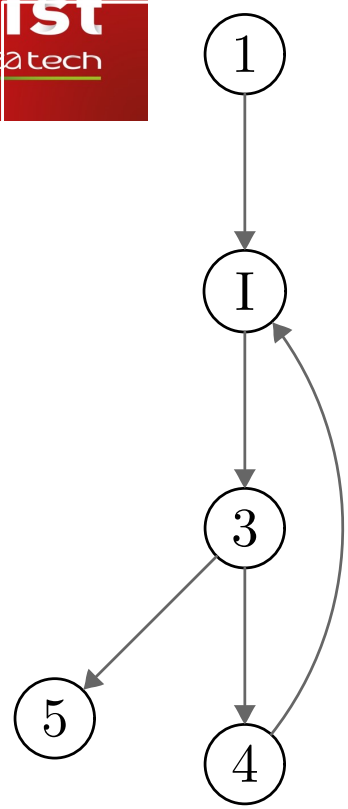
OnlyHeads



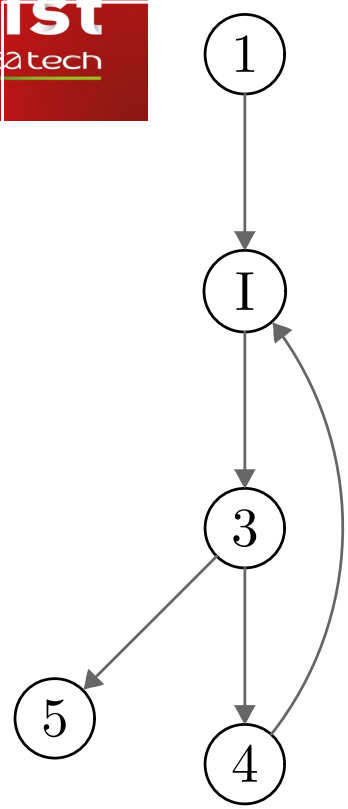


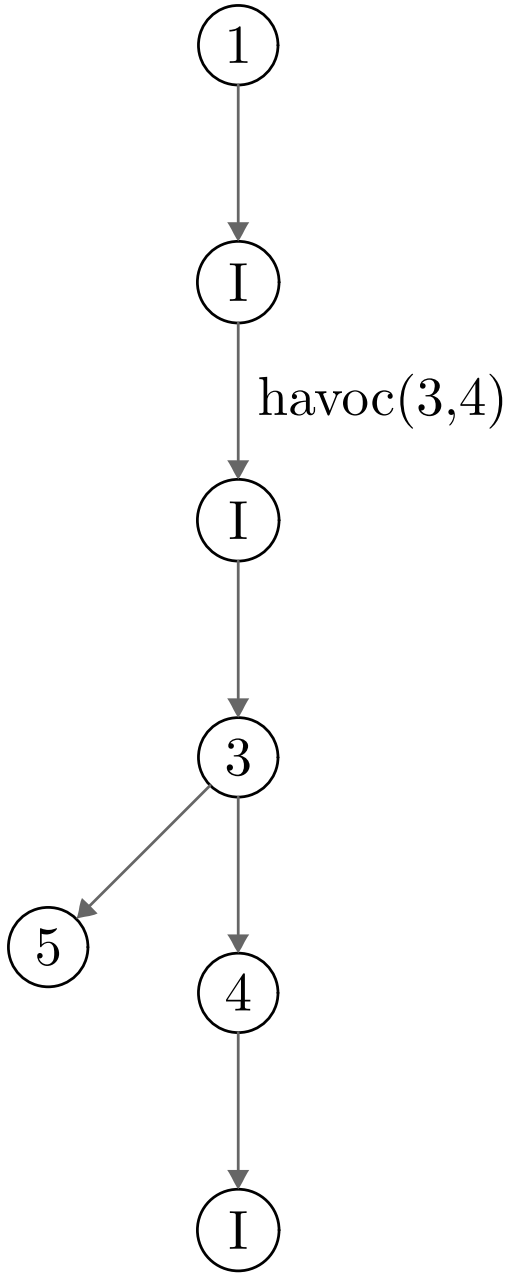
(1 (2 3 4) 5)

Loop

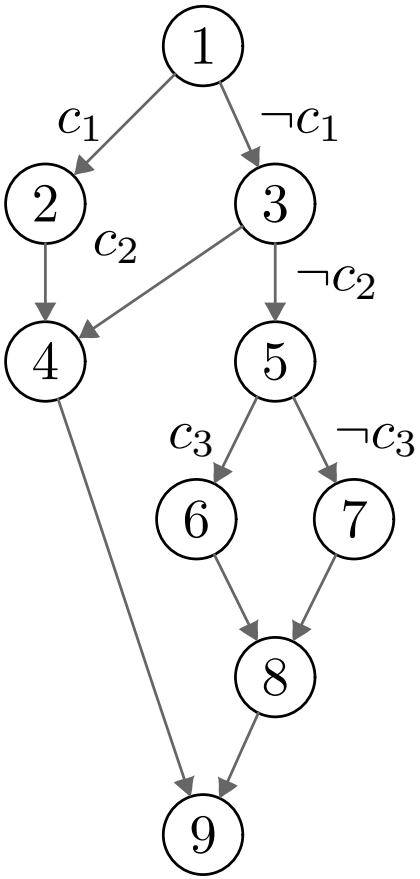


Loop

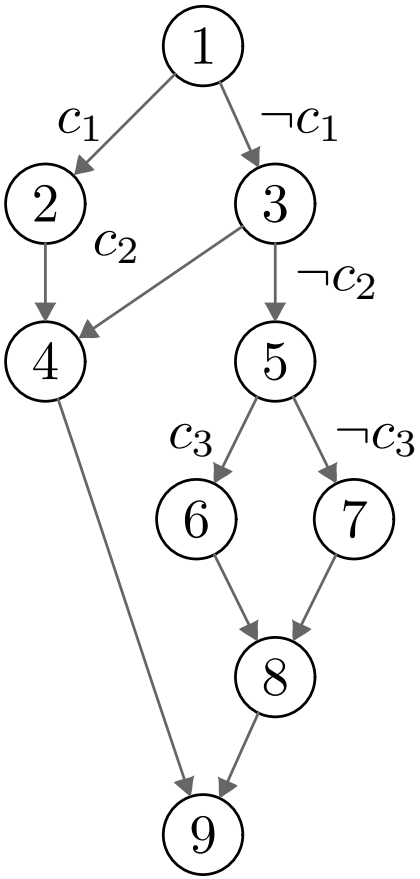


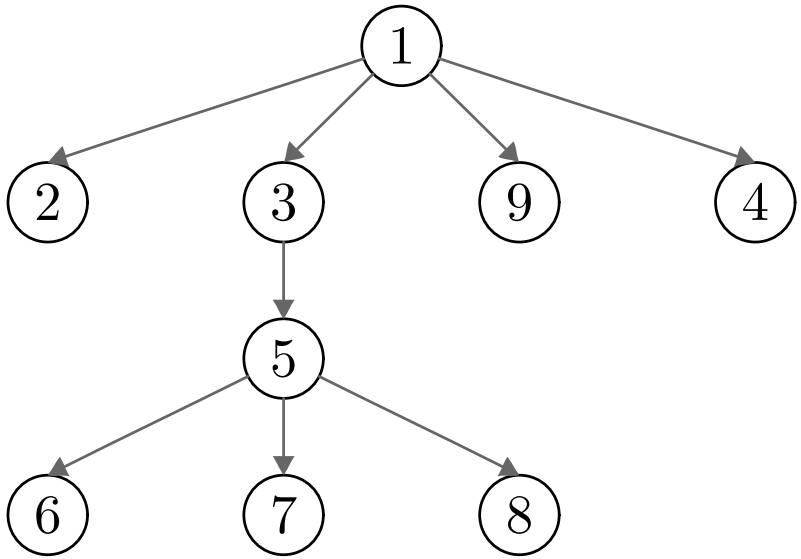


Reconstruction du sequent

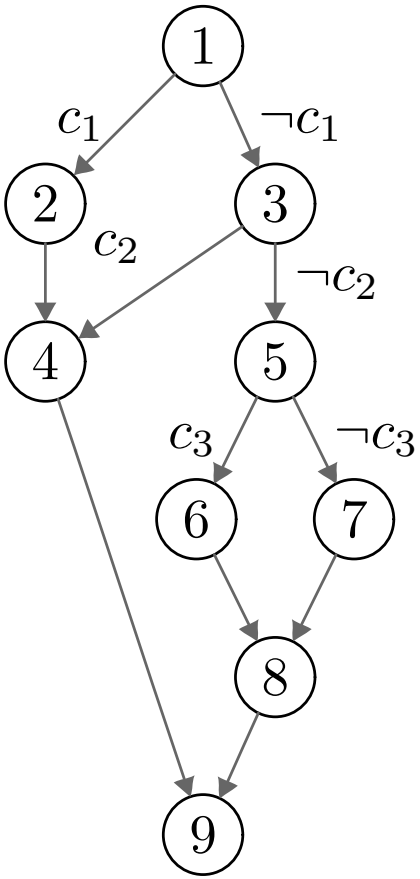


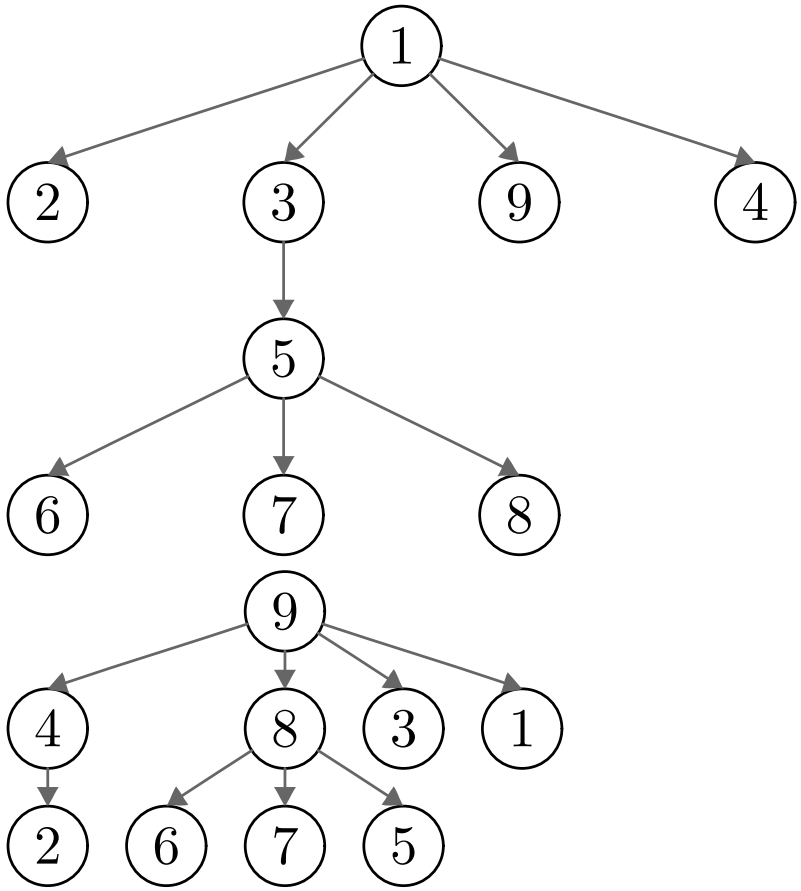
Reconstruction du sequent



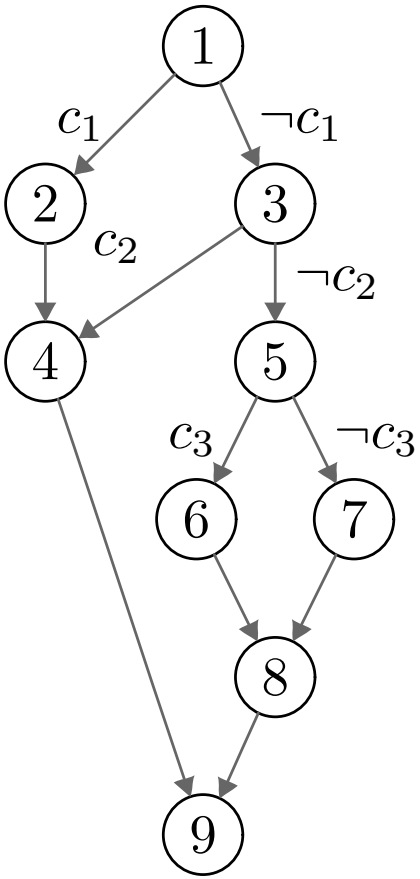


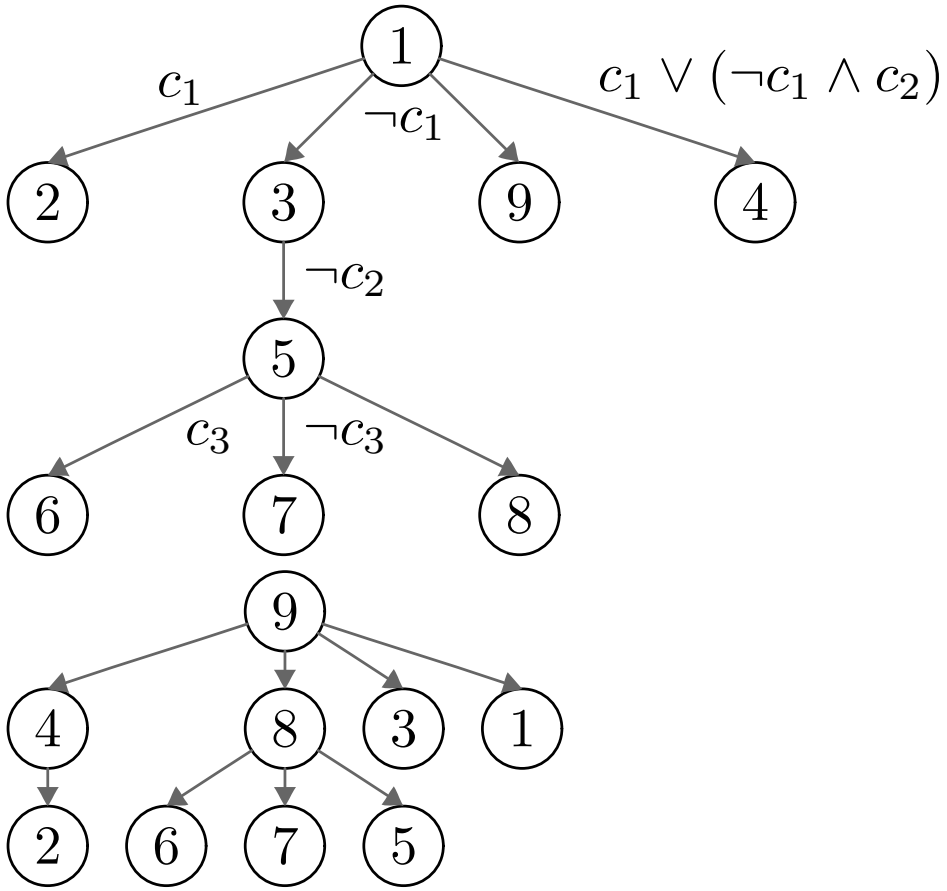
Reconstruction du sequent





Reconstruction du sequent





Sigma

Chunks defined by memory model:

$\mathcal{M}$ is all the memory

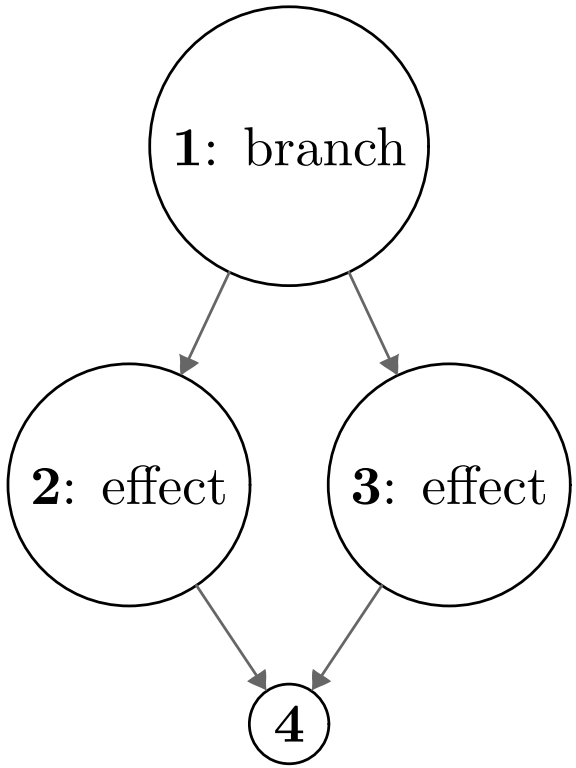
$\mathcal{V}$ is the set of variables QED

$\mathcal{M} = \uplus_{c \in \texttt{Chunks}} c$

$\sigma : \texttt{Chunks} \mapsto \mathcal{V}$

Cfg

```
L:
 if(x > 0) x = x + 1;
 else x = x + 2;
/*@ assert \at(L,x) < x; */
```



- **1**: branch$(\mathbf{2}, \mathbf{3}, x_1 > 0, \{x \mapsto x_1\})$
- **2**: effect$(x_2 = x_1 + 1, \{\mathbf{Pre}, \{x \mapsto x_1\}; \mathbf{Post}, \{x \mapsto x_2\}\}$
- **3**: effect$(0 = -x_2 + x_1 + 2, \{\mathbf{Pre}, \{x \mapsto x_1\}; \mathbf{Post}, \{x \mapsto x_2\}\}$
- goal$(x_1 < x_2, \{\mathbf{1}, \{x \mapsto x_1\}; \mathbf{4}, \{x \mapsto x_2\}\}$
- assume$(x_1 < x_2, \{\mathbf{1}, \{x \mapsto x_1\}; \mathbf{4}, \{x \mapsto x_2\}\}$

Sigmas and WhyML records

- Sigmas similar in use to WhyML records
- Remove one indirection
- Contains directly the variables (no `at`)
- Could share variables

CFG

Interpretation as formulas: Node $\mapsto$ Sigma$^{\perp}$,

`concat: cfg -> cfg -> cfg`

- nop(1)

CFG

Interpretation as formulas: Node $\mapsto$ Sigma$^{\perp}$,

`concat: cfg -> cfg -> cfg`

- nop(1)
- goto(1,2): $\sigma_1 \neq \perp \implies \sigma_2 \neq \perp$

CFG

Interpretation as formulas: Node $\mapsto$ Sigma$^{\perp}$,

`concat: cfg -> cfg -> cfg`

- nop(1)
- goto(1,2): $\sigma_1 \neq \perp \implies \sigma_2 \neq \perp$
- guard(1,b,2): $\sigma_1 \neq \perp \implies b \implies \sigma_2 \neq \perp$

CFG

Interpretation as formulas: Node $\mapsto$ Sigma$^{\perp}$,
`concat: cfg -> cfg -> cfg`

- nop(1)
- goto(1,2): $\sigma_1 \neq \perp \implies \sigma_2 \neq \perp$
- guard(1,b,2): $\sigma_1 \neq \perp \implies b \implies \sigma_2 \neq \perp$
- branch(1,b,2,3): $\sigma_1 \neq \perp \implies ite(\sigma_1 b, \sigma_2 \neq \perp, \sigma_3 \neq \perp)$

CFG

Interpretation as formulas: Node $\mapsto$ Sigma$^{\perp}$,

`concat: cfg -> cfg -> cfg`

- nop(1)
- goto(1,2): $\sigma_1 \neq \perp \implies \sigma_2 \neq \perp$
- guard(1,b,2): $\sigma_1 \neq \perp \implies b \implies \sigma_2 \neq \perp$
- branch(1,b,2,3): $\sigma_1 \neq \perp \implies ite(\sigma_1 b, \sigma_2 \neq \perp, \sigma_3 \neq \perp)$
- either(l): $\vee t_i \ ((b \wedge t_1) \vee (\neg b \wedge t_2))$

CFG

Interpretation as formulas: Node $\mapsto$ Sigma$^{\perp}$,

`concat: cfg -> cfg -> cfg`

- nop(1)
- goto(1,2): $\sigma_1 \neq \perp \implies \sigma_2 \neq \perp$
- guard(1,b,2): $\sigma_1 \neq \perp \implies b \implies \sigma_2 \neq \perp$
- branch(1,b,2,3): $\sigma_1 \neq \perp \implies ite(\sigma_1 b, \sigma_2 \neq \perp, \sigma_3 \neq \perp)$
- either(l): $\vee t_i \ ((b \wedge t_1) \vee (\neg b \wedge t_2))$
- implies(b,t;...): $\bigwedge(b_i \implies t_i) \ ((b \implies t_1) \wedge (\neg b \implies t_2))$

CFG

Interpretation as formulas: Node $\mapsto$ Sigma$^{\perp}$,

`concat: cfg -> cfg -> cfg`

- nop(1)
- goto(1,2): $\sigma_1 \neq \perp \implies \sigma_2 \neq \perp$
- guard(1,b,2): $\sigma_1 \neq \perp \implies b \implies \sigma_2 \neq \perp$
- branch(1,b,2,3): $\sigma_1 \neq \perp \implies ite(\sigma_1 b, \sigma_2 \neq \perp, \sigma_3 \neq \perp)$
- either(l): $\vee t_i \ ((b \wedge t_1) \vee (\neg b \wedge t_2))$
- implies(b,t;...): $\bigwedge(b_i \implies t_i) \ ((b \implies t_1) \wedge (\neg b \implies t_2))$
- havoc

CFG

Interpretation as formulas: Node $\mapsto$ Sigma$^{\perp}$,
`concat: cfg -> cfg -> cfg`

- nop(1)
- goto(1,2): $\sigma_1 \neq \perp \implies \sigma_2 \neq \perp$
- guard(1,b,2): $\sigma_1 \neq \perp \implies b \implies \sigma_2 \neq \perp$
- branch(1,b,2,3): $\sigma_1 \neq \perp \implies ite(\sigma_1 b, \sigma_2 \neq \perp, \sigma_3 \neq \perp)$
- either(l): $\vee t_i \ ((b \wedge t_1) \vee (\neg b \wedge t_2))$
- implies(b,t;...): $\bigwedge(b_i \implies t_i) \ ((b \implies t_1) \wedge (\neg b \implies t_2))$
- havoc
- effect(1,e,2): $\sigma_1 \neq \perp \implies \sigma_1 \sigma_2 e$

CFG

Interpretation as formulas: Node $\mapsto$ Sigma$^{\perp}$,

`concat: cfg -> cfg -> cfg`

- nop(1)
- goto(1,2): $\sigma_1 \neq \perp \implies \sigma_2 \neq \perp$
- guard(1,b,2): $\sigma_1 \neq \perp \implies b \implies \sigma_2 \neq \perp$
- branch(1,b,2,3): $\sigma_1 \neq \perp \implies ite(\sigma_1 b, \sigma_2 \neq \perp, \sigma_3 \neq \perp)$
- either(l): $\vee t_i \ ((b \wedge t_1) \vee (\neg b \wedge t_2))$
- implies(b,t;...): $\bigwedge(b_i \implies t_i) \ ((b \implies t_1) \wedge (\neg b \implies t_2))$
- havoc
- effect(1,e,2): $\sigma_1 \neq \perp \implies \sigma_1 \sigma_2 e$
- loop?

WP will require Why3 next release

- Why3 version 1.1
- no intermediate file
- WP_SHARE reads directly why3 file
- call using Why3 API
- Futur: QED uses why3 types
- Futur: counter-example by multiple prover call?