Optimising Faceted Secure Multi-Execution

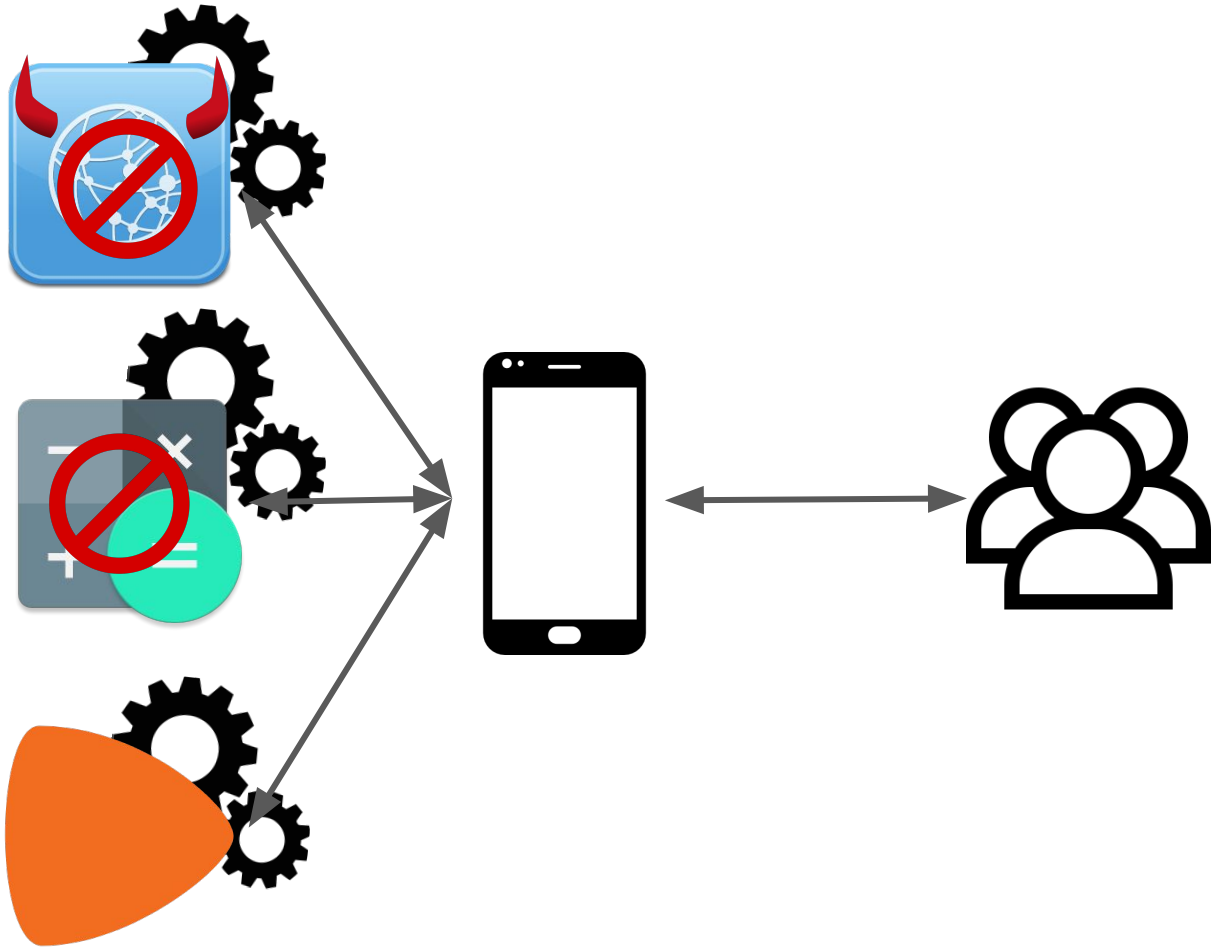
Maximilian Algehed[1], Alejandro Russo[1], Cormac Flanagan[2]

CSF 2019

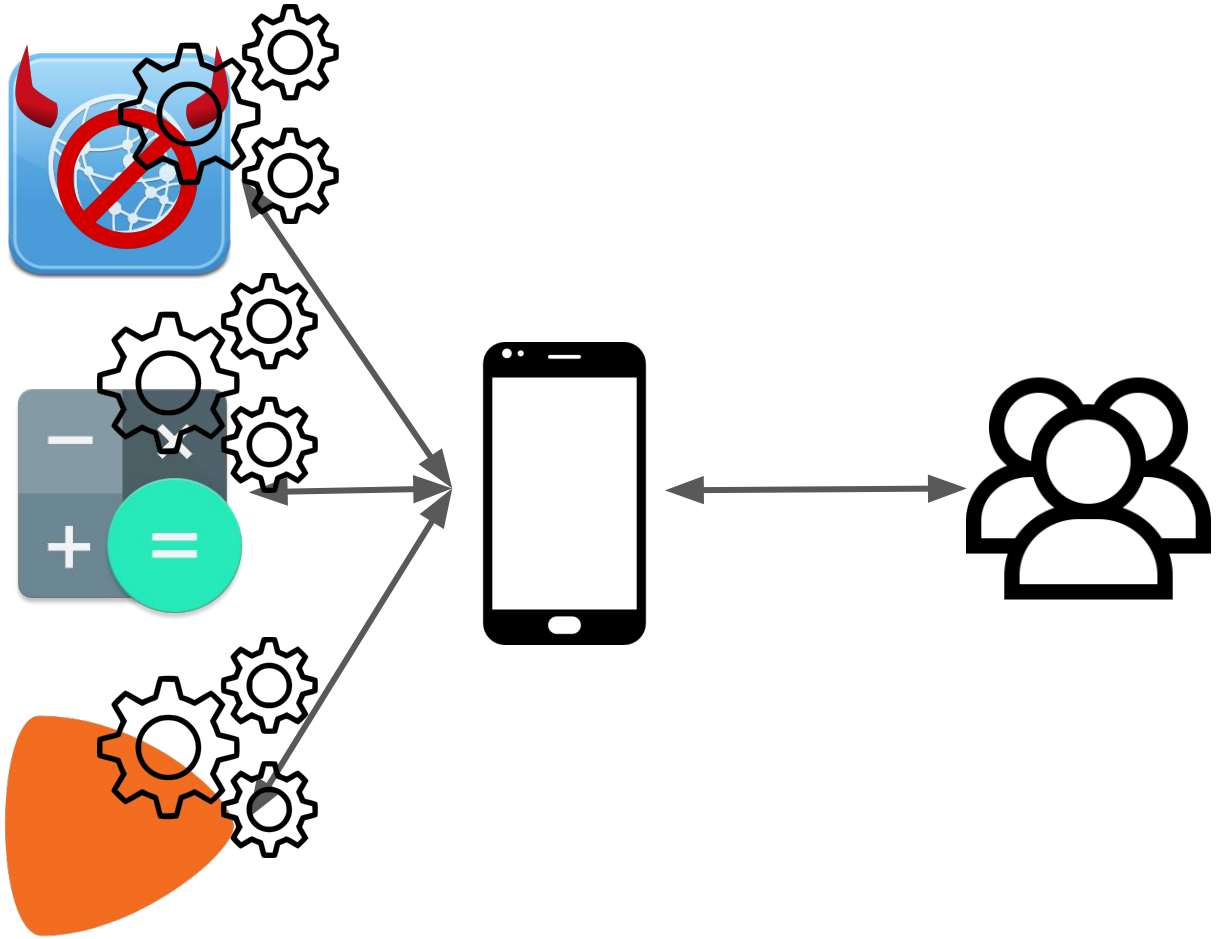
1. Chalmers

2. UCSC

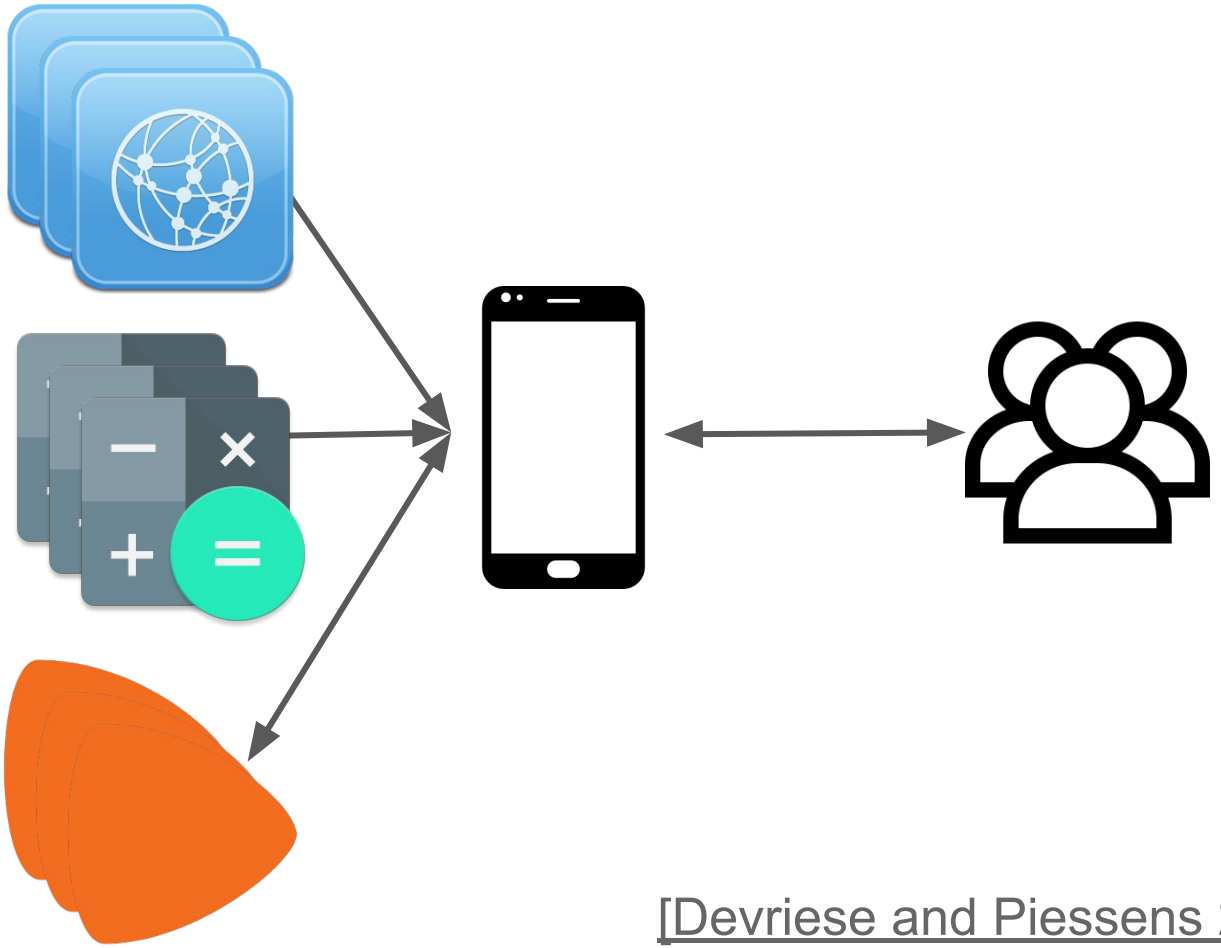
Dynamic IFC



Transparency



Multi-Execution



[Devriese and Piessens 2010]
[Austin and Flanagan 2012]

Multi-Execution

```
h := readSecret();
l := 0;
if h then:
  l := 1;
writePublic(l);
```

Multiple Facets

```
h := readSecret();

l := 0;

if h then:
   l := 1;

writePublic(l);
```

x2

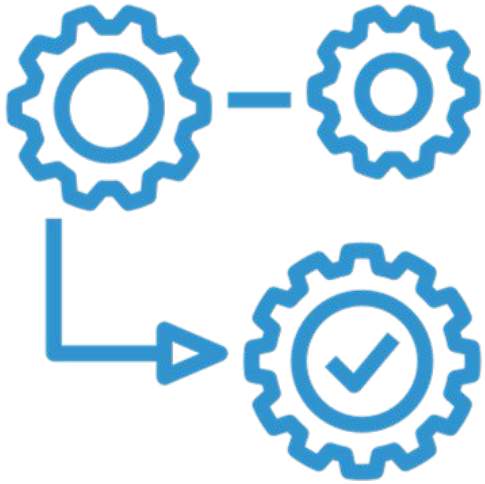
```
[h = < H ? s : 0 >]

[h = < H ? s : 0 >,
 l = 0]

[h = < H ? s : 0 >,
 l = < H ? 1 : 0 >]
```

```
publicView(< H ? 1 : 0 >) = 0
```

[Austin and Flanagan 2012]



Multef

[Schmitz et al. 2018]

Optimisation Strategies

Multef

Data-oriented

Compute-oriented

All contributions mechanised in Agda

Reduce needless duplication in data storage

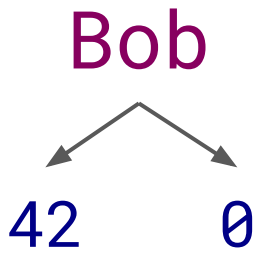
Remove unnecessary duplicated computation

Data-oriented Optimisation

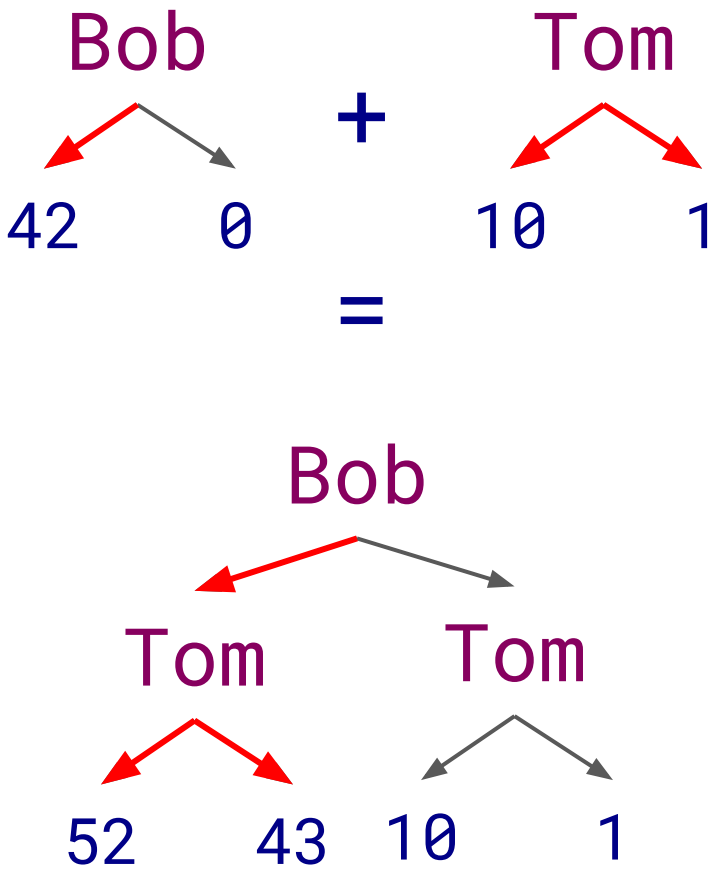
data Faceted a

```
bob, tom : Faceted Int
bob = < Bob ? 42 : 0 >
tom = 0
```

bob =



[Austin and Flanagan 2012] (Multiple Facets)



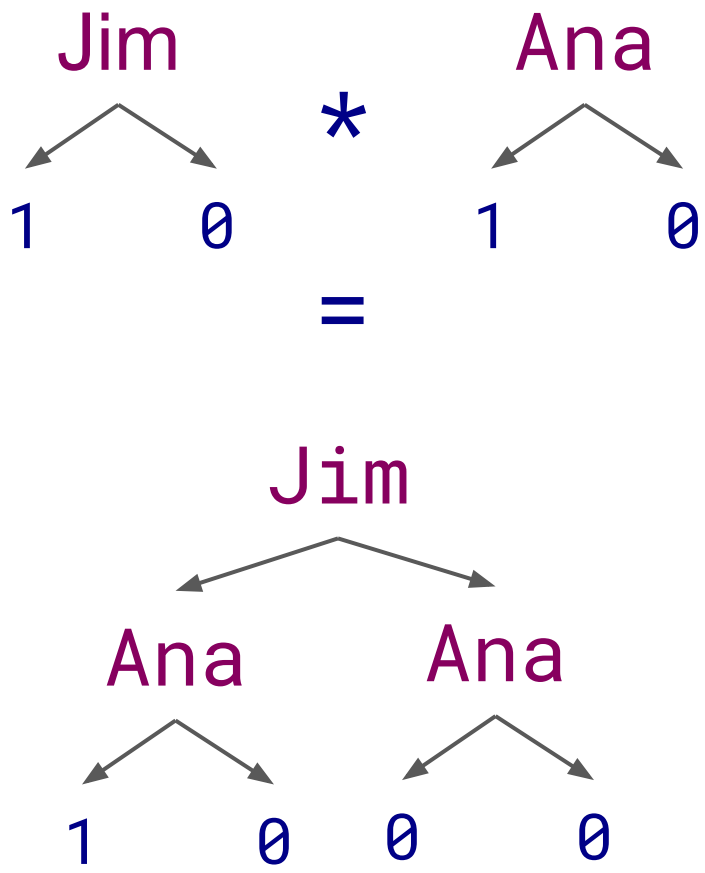
[Austin and Flanagan 2012] (Multiple Facets)

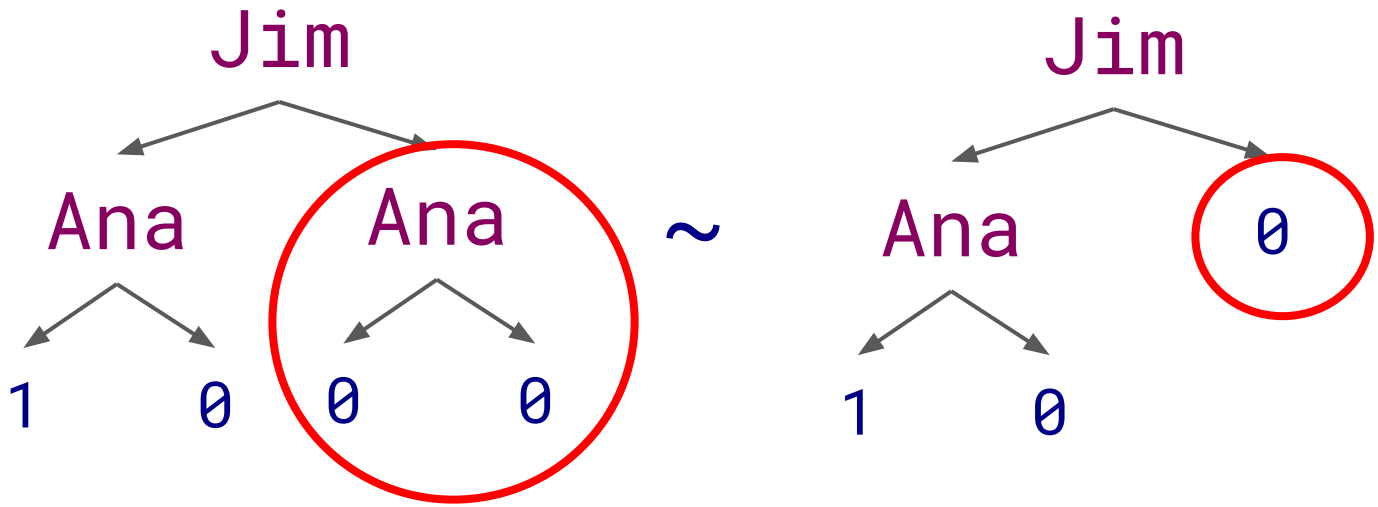
What could possibly go wrong?
(live demo)

How does shrinking work?

$$t \sim t'$$

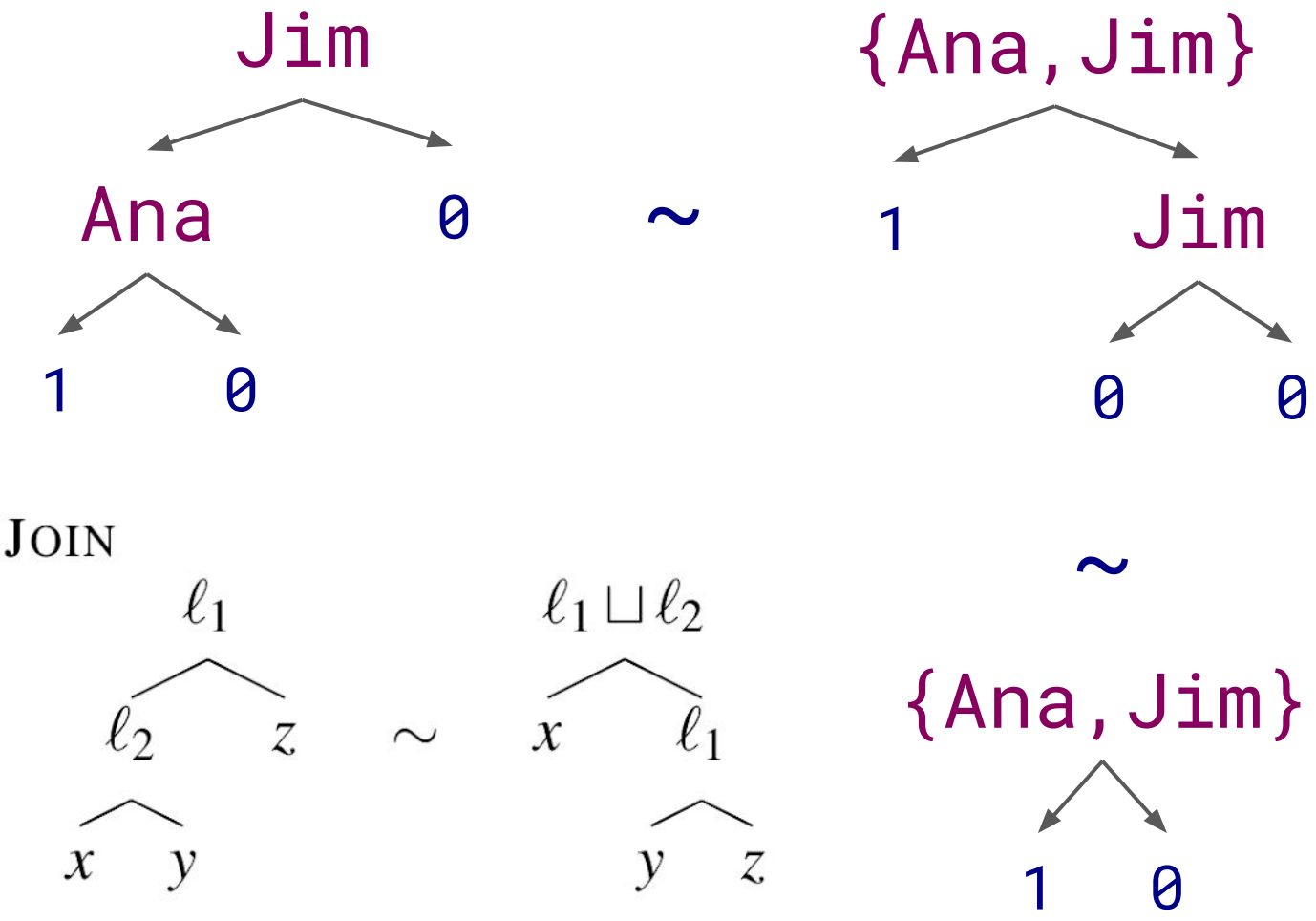
t and t’ look the same to all observers

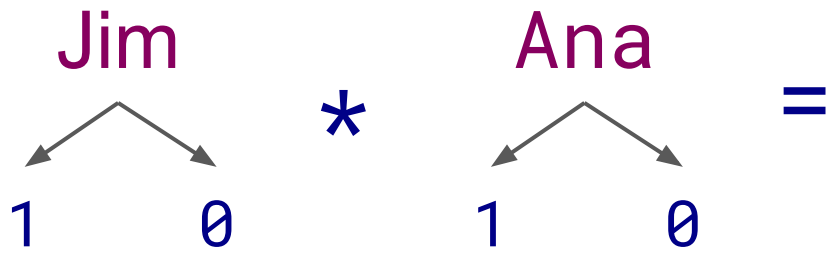

Jim
Ana
*
1
0
1
0
=
Jim
Ana
Ana
1
0
0
0

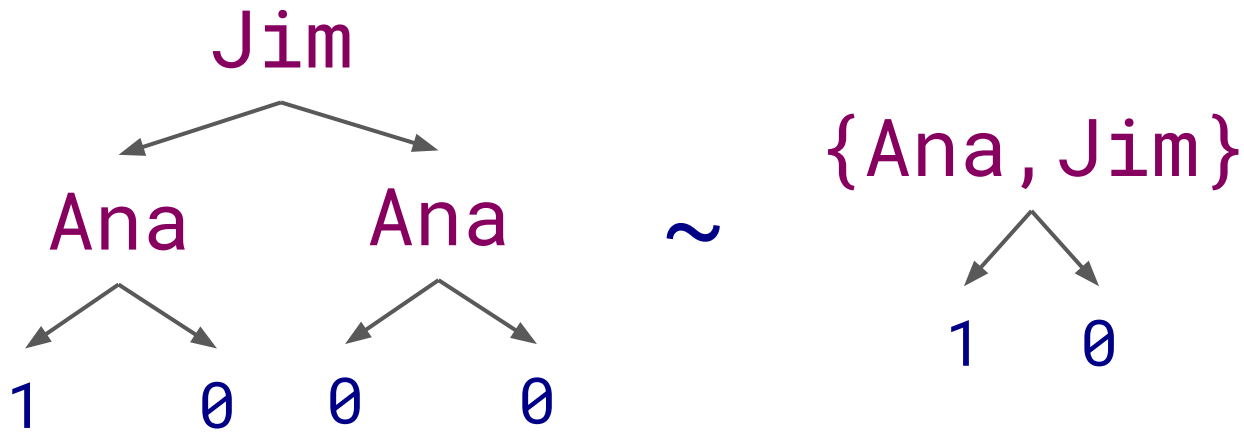


CHOICE IRRELEVANCE

$$\begin{array}{ccc} \ell & & \\ \swarrow\ \searrow & \sim & x \\ x \quad x & & \end{array}$$

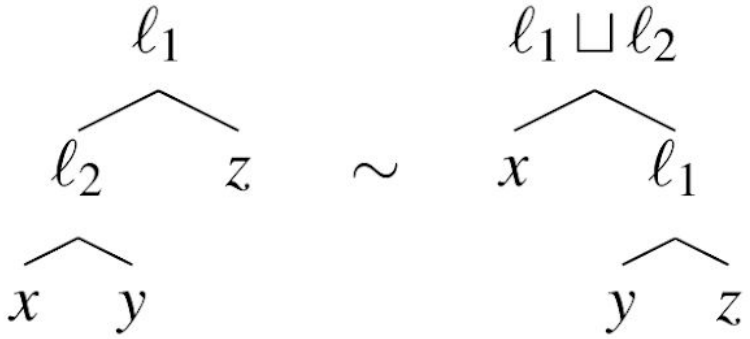

Jim
Ana
0
1
0
~
{Ana,Jim}
1
Jim
0
0
JOIN
ℓ_1
ℓ_2
z
x
y
~
$\ell_1 \sqcup \ell_2$
x
ℓ_1
y
z
~
{Ana,Jim}
1
0


Jim
*
Ana
=
1
0
1
0


Jim
Ana
Ana
~
{Ana,Jim}
1
0
0
0
1
0

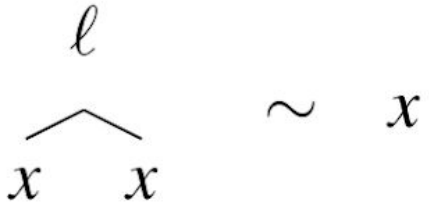
Multef

JOIN



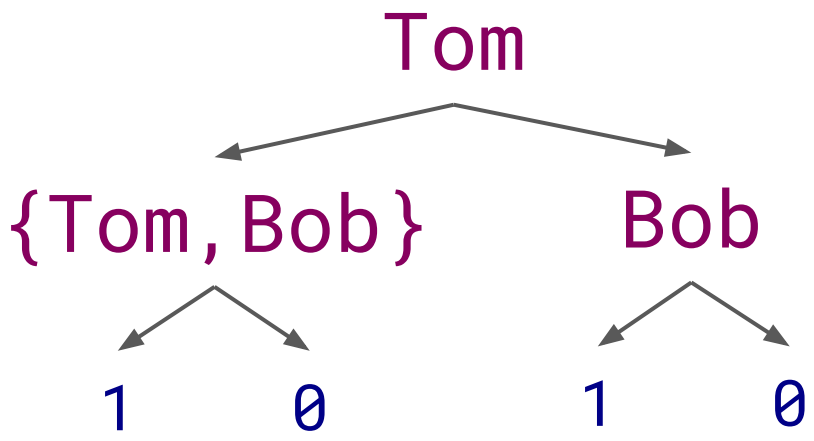
Change Stuff

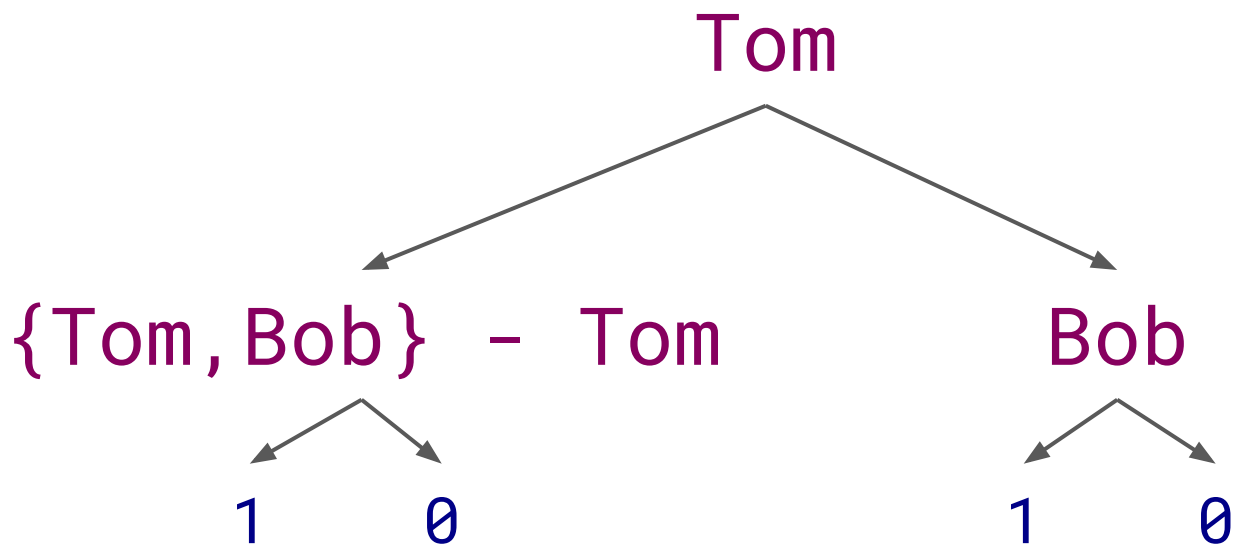
CHOICE IRRELEVANCE

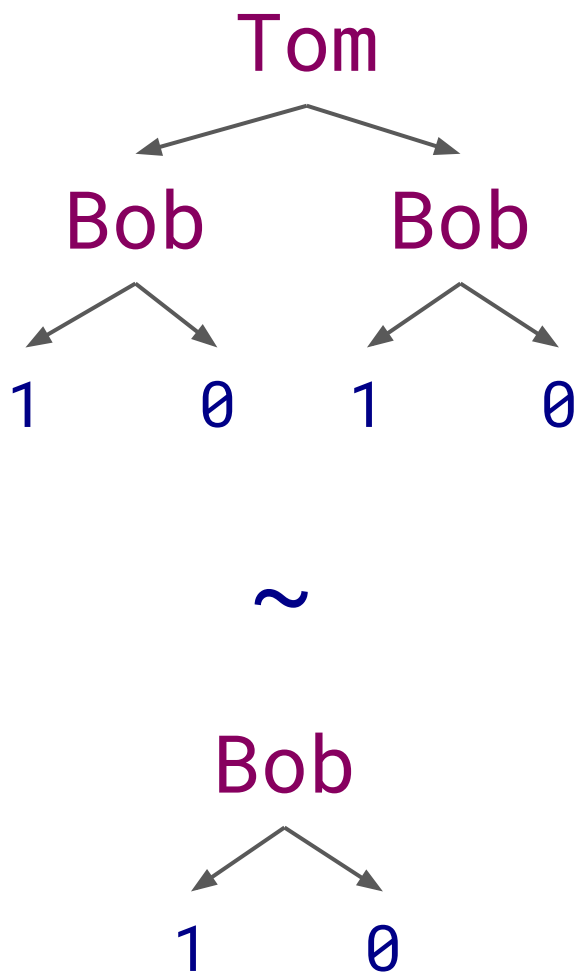


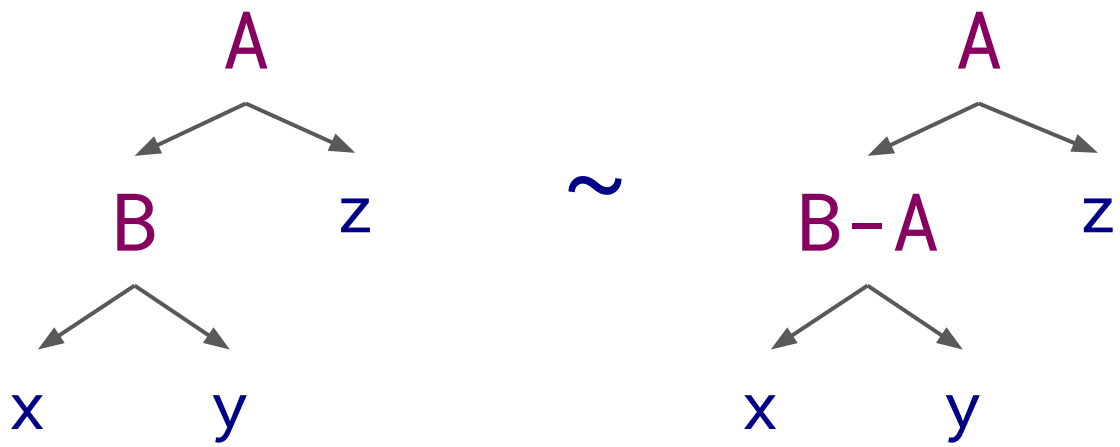
Remove Stuff

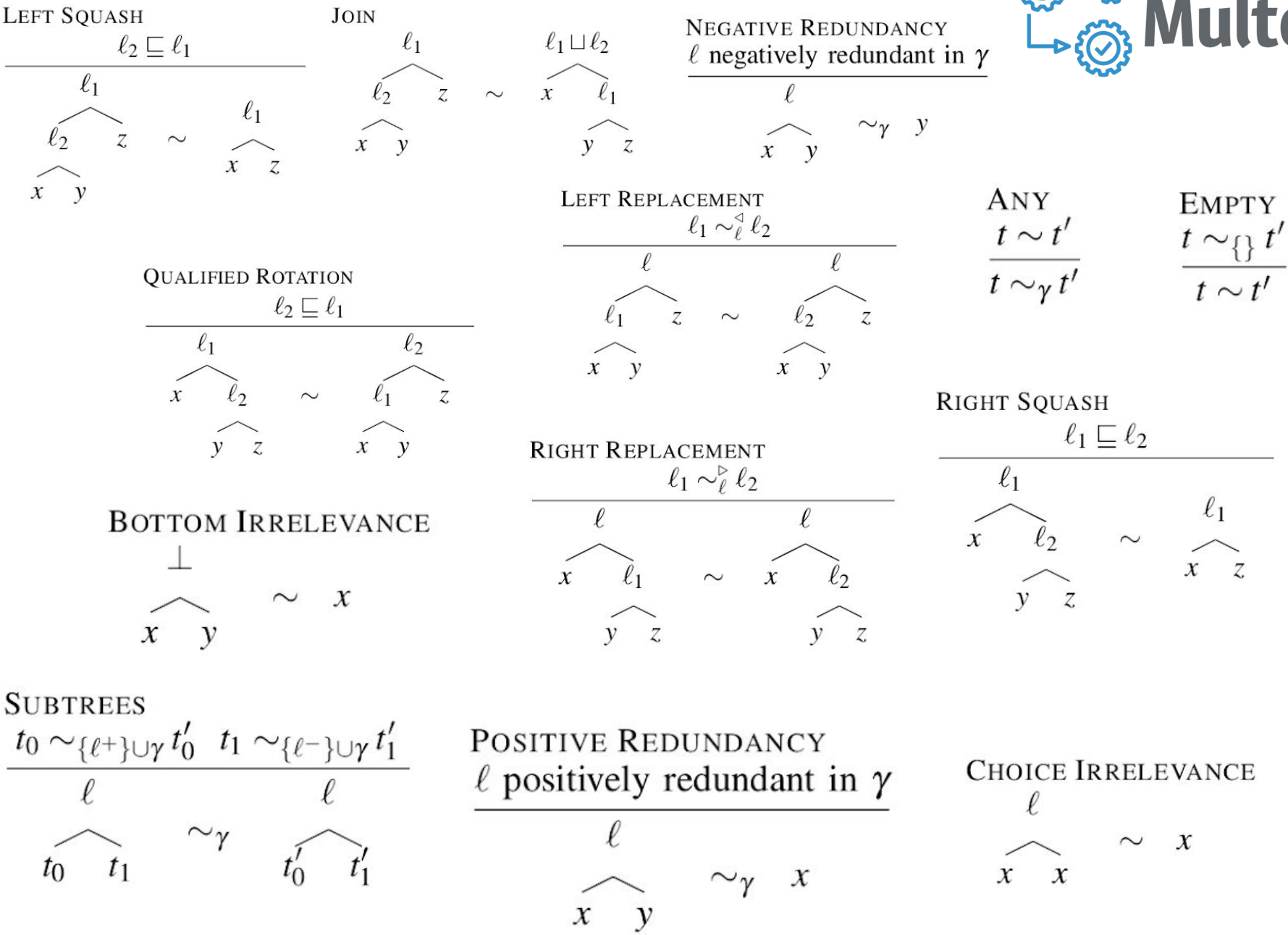
Two rules? Is that enough?


Tom
{Tom,Bob}
Bob
1
0
1
0

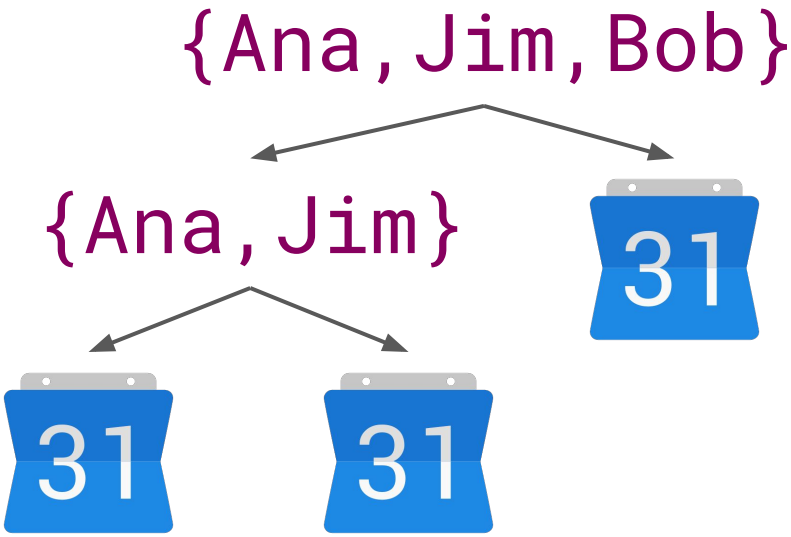

Tom
{Tom,Bob} - Tom
Bob
1
0
1
0


Tom
Bob
Bob
1
0
1
0
~
Bob
1
0


A
B
z
x
y
~
A
B−A
z
x
y


LEFT SQUASH
$\ell_2 \sqsubseteq \ell_1$
ℓ_1 ℓ_2 z $\sim$ ℓ_1 x y x z
JOIN
ℓ_1 ℓ_2 z x y $\sim$ $\ell_1 \sqcup \ell_2$ x ℓ_1 y z
NEGATIVE REDUNDANCY
ℓ negatively redundant in γ
ℓ x y $\sim_\gamma$ y
LEFT REPLACEMENT
$\ell_1 \sim_\ell^\triangleleft \ell_2$
ℓ ℓ_1 z x y $\sim$ ℓ ℓ_2 z x y
ANY
$t \sim t'$ / $t \sim_\gamma t'$
EMPTY
$t \sim_{\{\}} t'$ / $t \sim t'$
QUALIFIED ROTATION
$\ell_2 \sqsubseteq \ell_1$
ℓ_1 x ℓ_2 y z $\sim$ ℓ_2 ℓ_1 z x y
RIGHT REPLACEMENT
$\ell_1 \sim_\ell^\triangleright \ell_2$
ℓ x ℓ_1 y z $\sim$ ℓ x ℓ_2 y z
RIGHT SQUASH
$\ell_1 \sqsubseteq \ell_2$
ℓ_1 x ℓ_2 y z $\sim$ ℓ_1 x z
BOTTOM IRRELEVANCE
$\bot$ x y $\sim$ x
SUBTREES
$t_0 \sim_{\{\ell^+\} \cup \gamma} t_0'$ $t_1 \sim_{\{\ell^-\} \cup \gamma} t_1'$
ℓ t_0 t_1 $\sim_\gamma$ ℓ t_0' t_1'
POSITIVE REDUNDANCY
ℓ positively redundant in γ
ℓ x y $\sim_\gamma$ x
CHOICE IRRELEVANCE
ℓ x x $\sim$ x

Multef



[Yang et al. 2016]

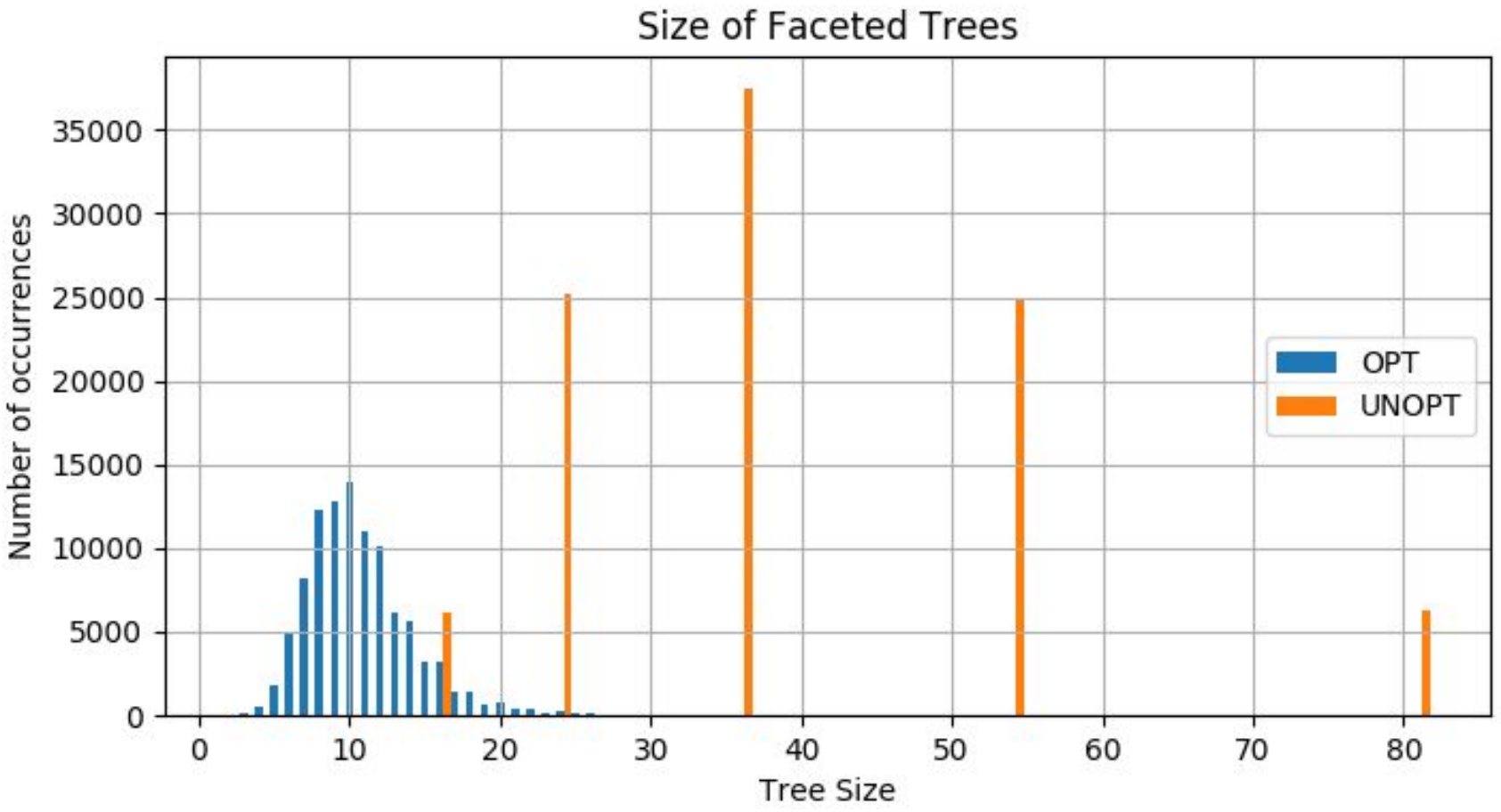
Multef

a) 31 + 31 + 31 + 31

b) overlap(31, 31, 31, 31)

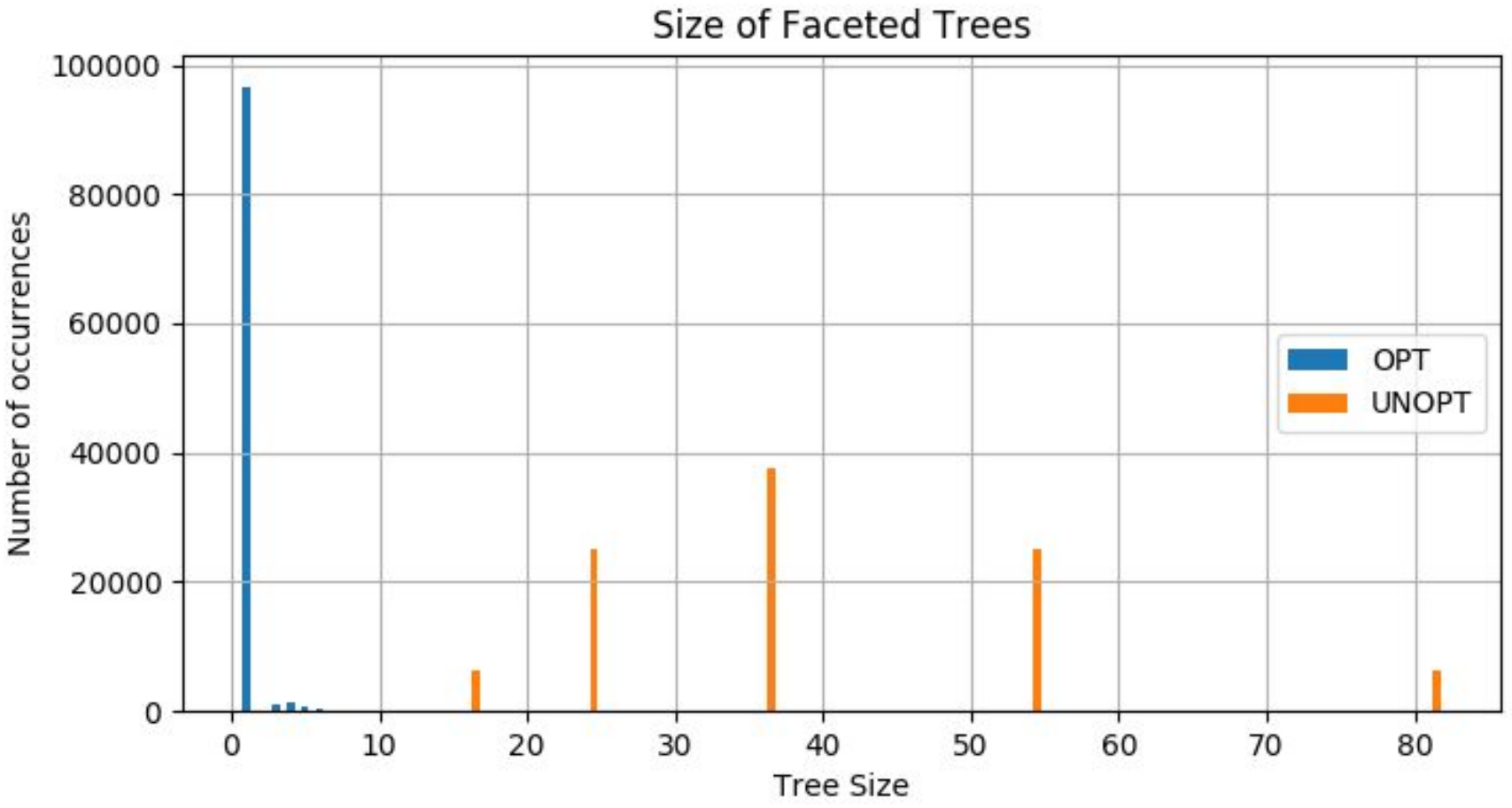
a) 31 + 31 + 31 + 31

Multef



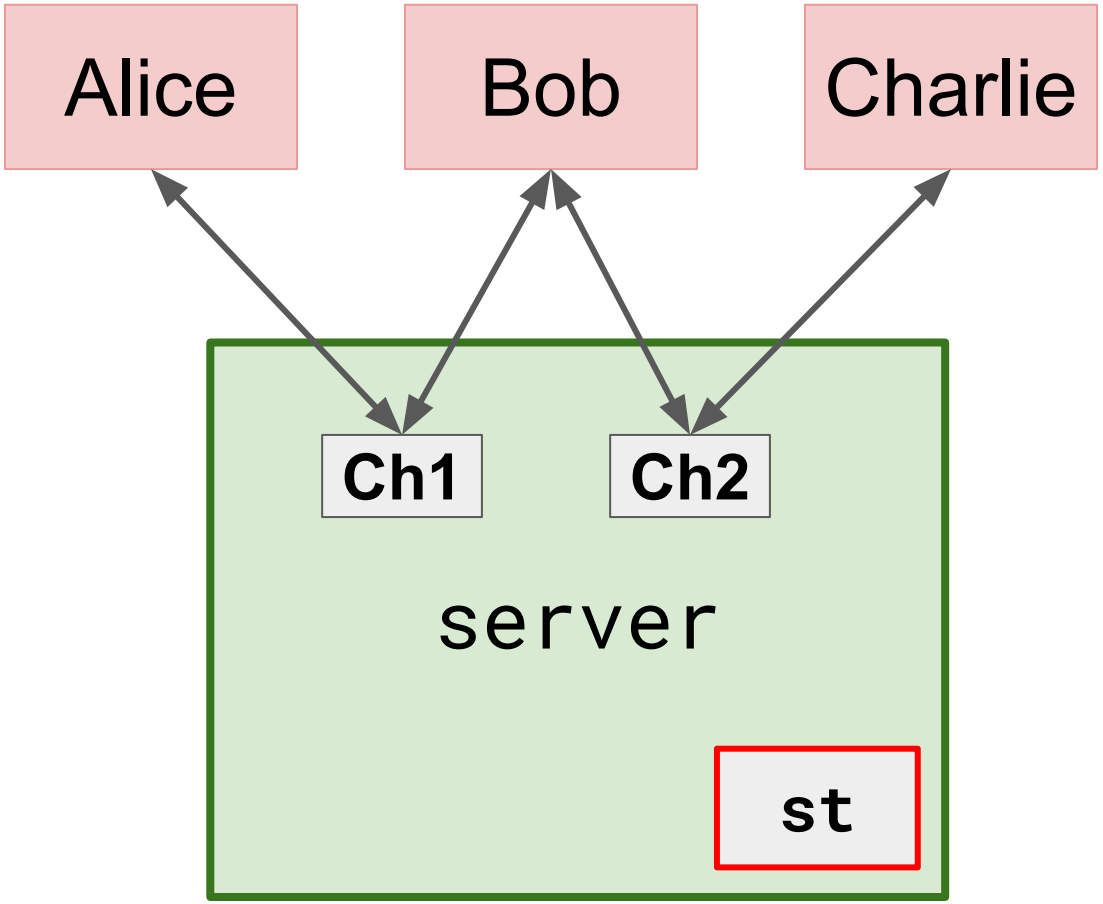
b) overlap(31 , 31 , 31 , 31)

Multef



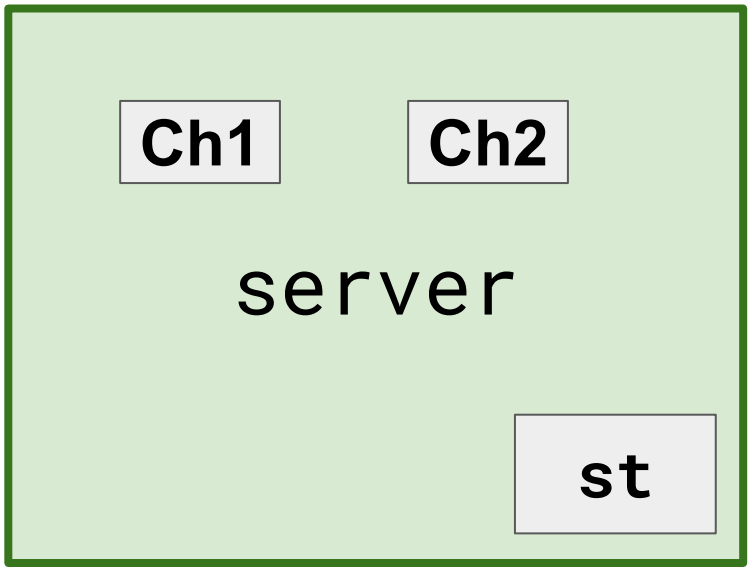
Computation-oriented Optimisation

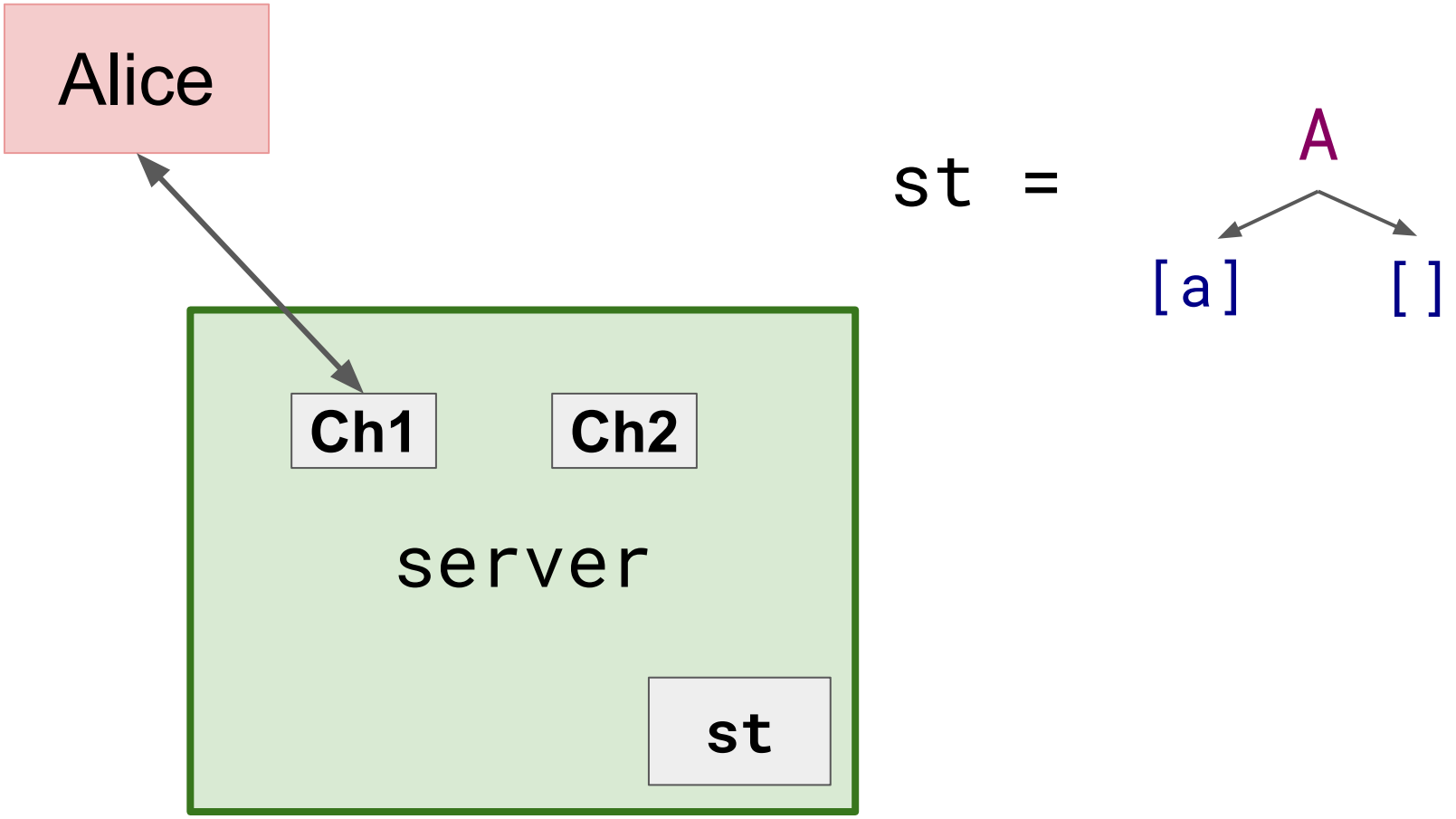
Multef

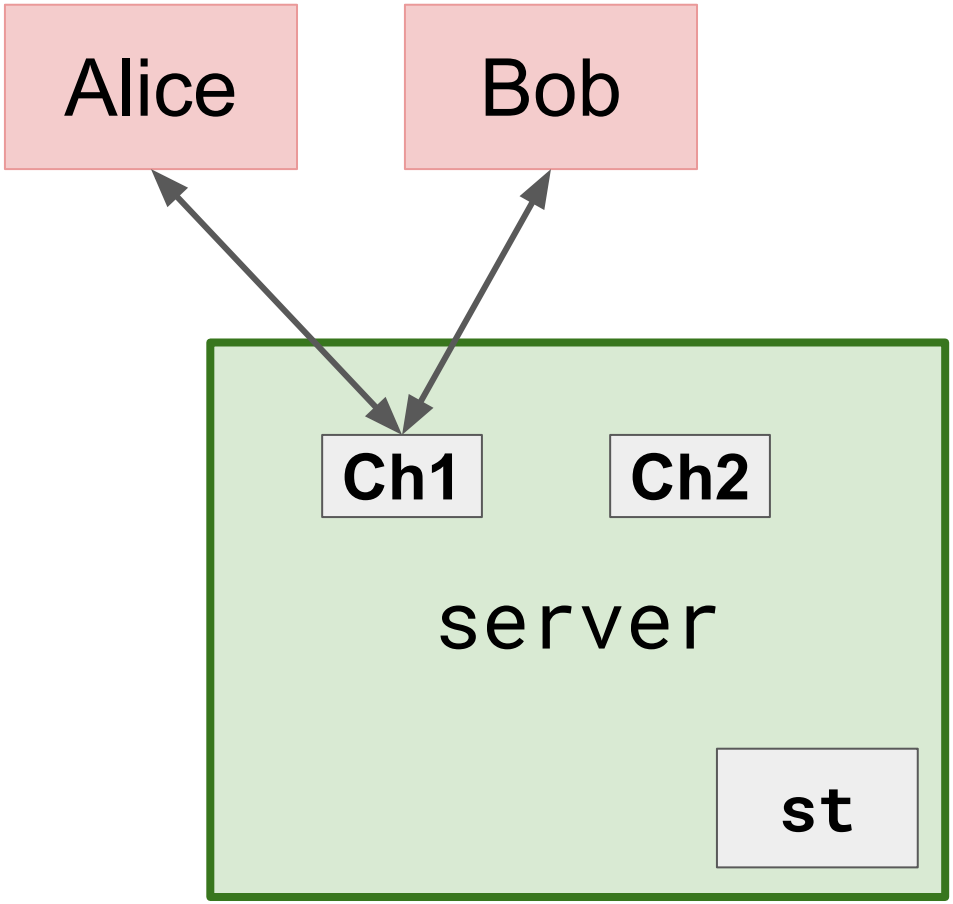


Multef

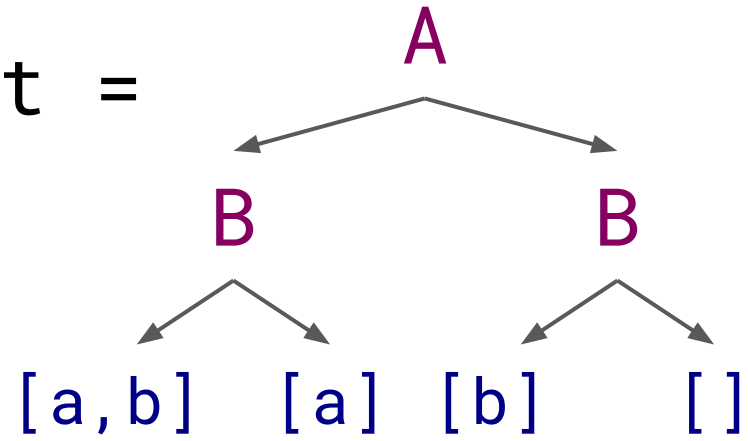
```
st = []
```



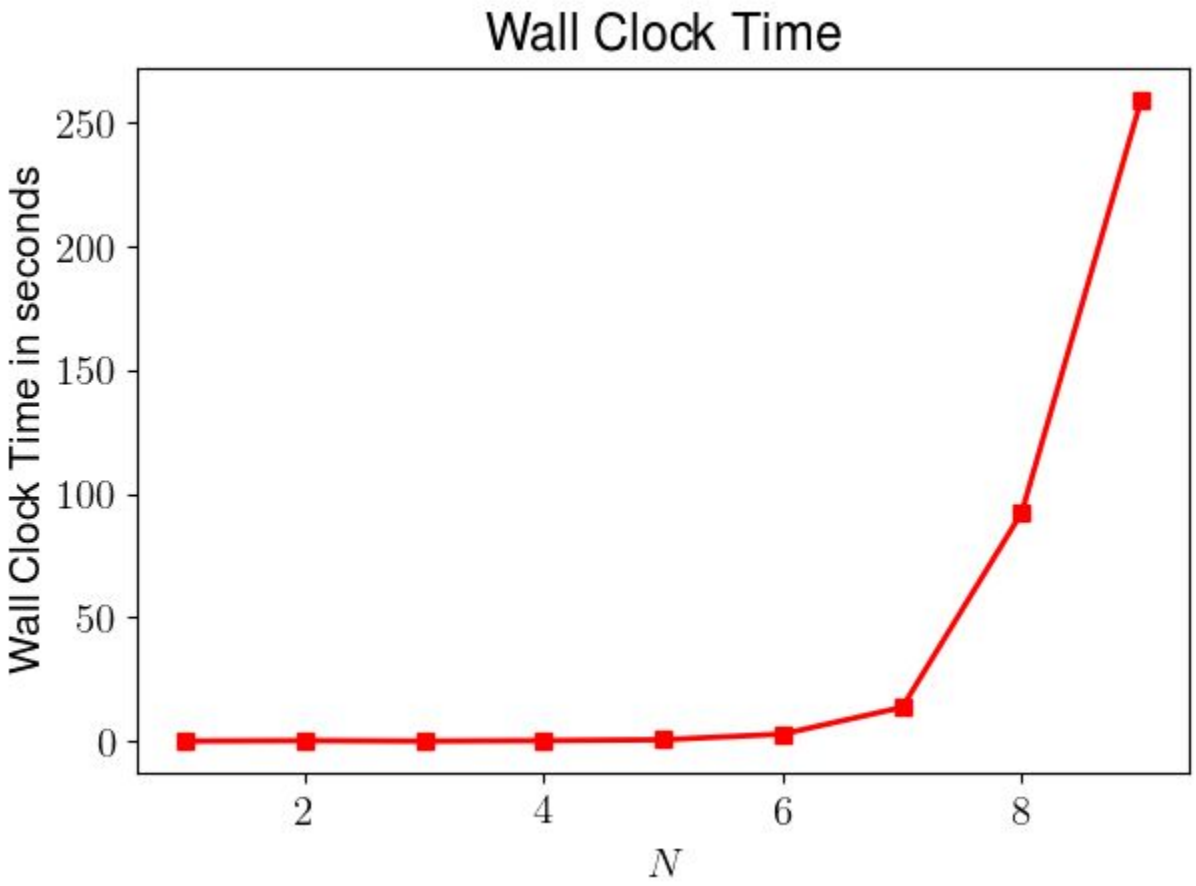

Alice
Ch1
Ch2
server
st
st =
A
[a]
[]



st =

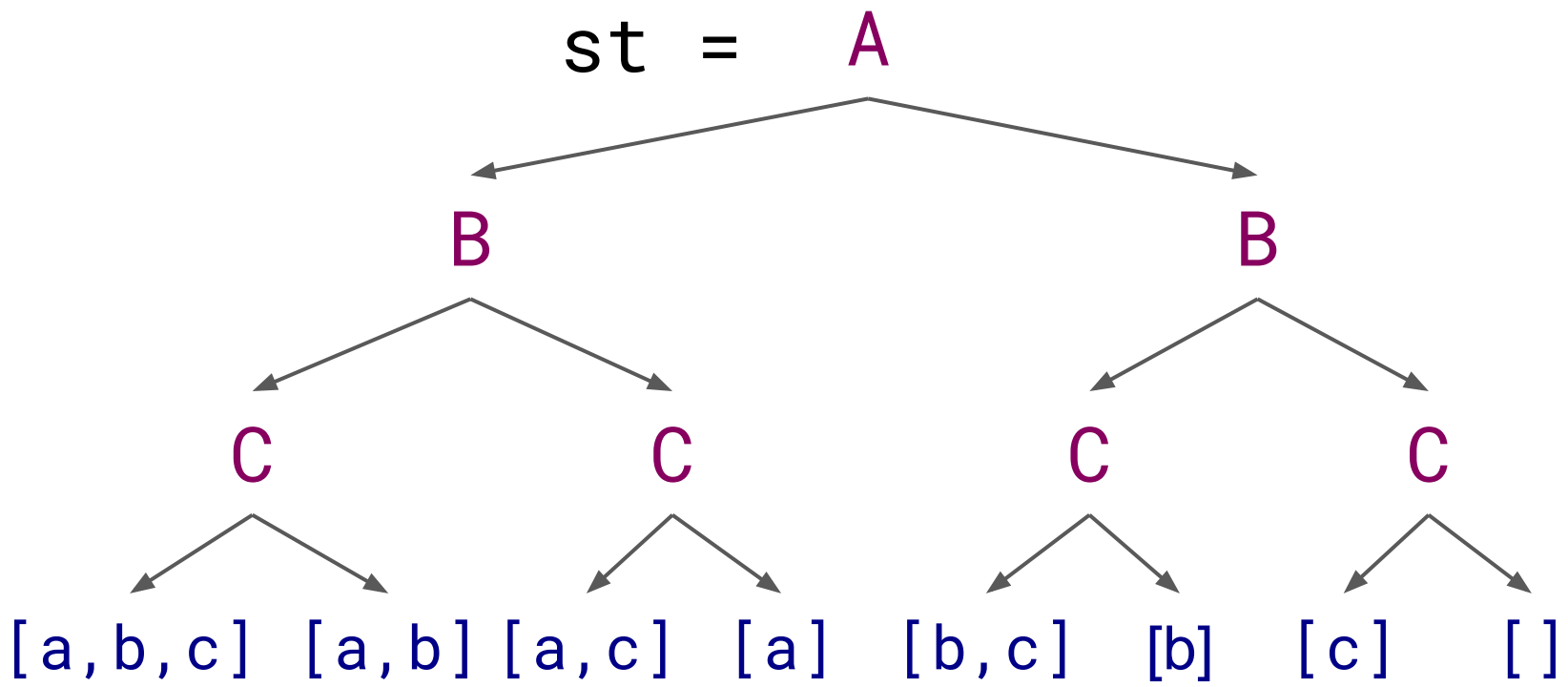


Multef


Wall Clock Time
Wall Clock Time in seconds
250
200
150
100
50
0
2
4
6
8
N

```
Connect “Alice” “Ch1”
Connect “Bob” “Ch1”
Connect “Charlie” “Ch1”
```

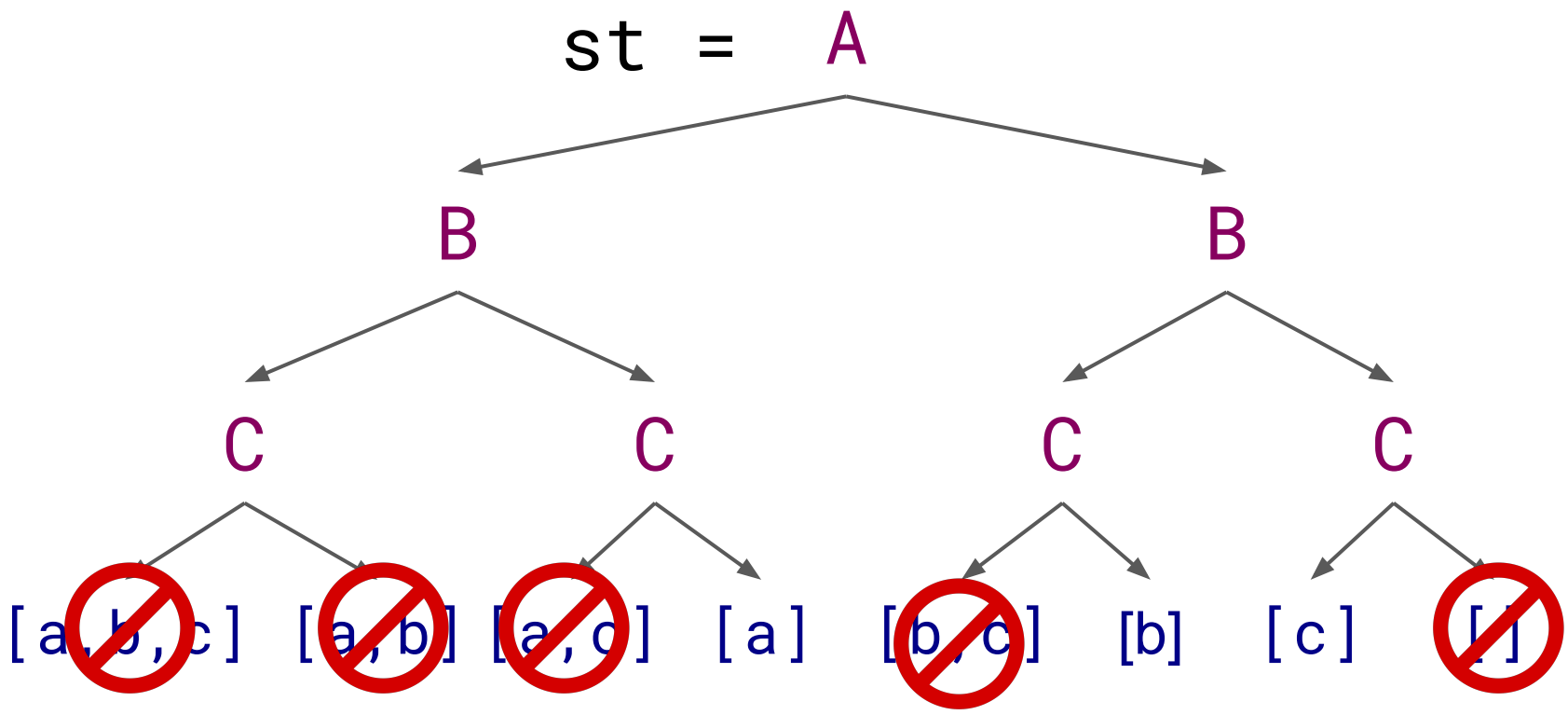
Multef



Write “Alice” “Ch1” “Hi everybody”

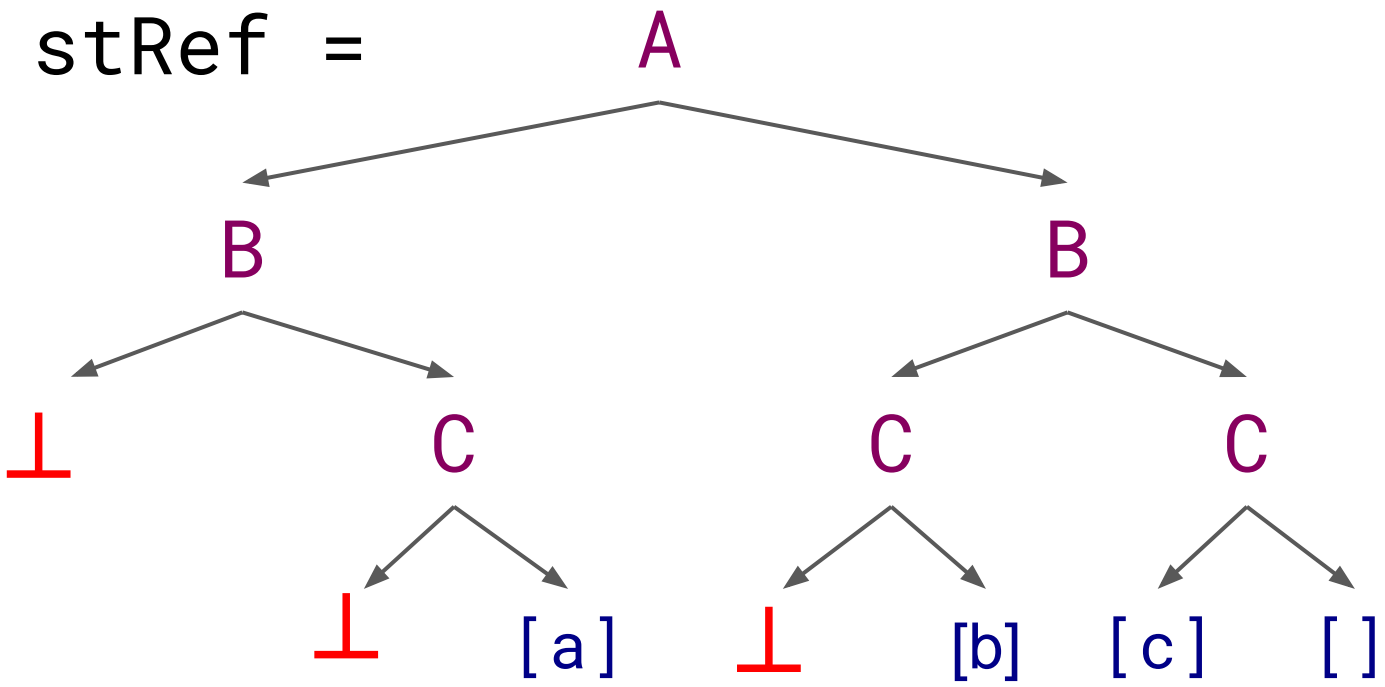
```
foreach client in st.Ch1 do:
  writeSocket client “Alice> Ch1: Hi everybody”
```

```
writeSocket “Alice”   “Alice> Ch1: Hi everybody”
writeSocket “Bob”     “Alice> Ch1: Hi everybody”
writeSocket “Charlie” “Alice> Ch1: Hi everybody”
```


st = A
B
B
C
C
C
C
[a,b,c]
[a,b]
[a,c]
[a]
[b,c]
[b]
[c]
[]

```
st_ <- readFIORef stRef
let st = atMostOne st_
stRef := st
```

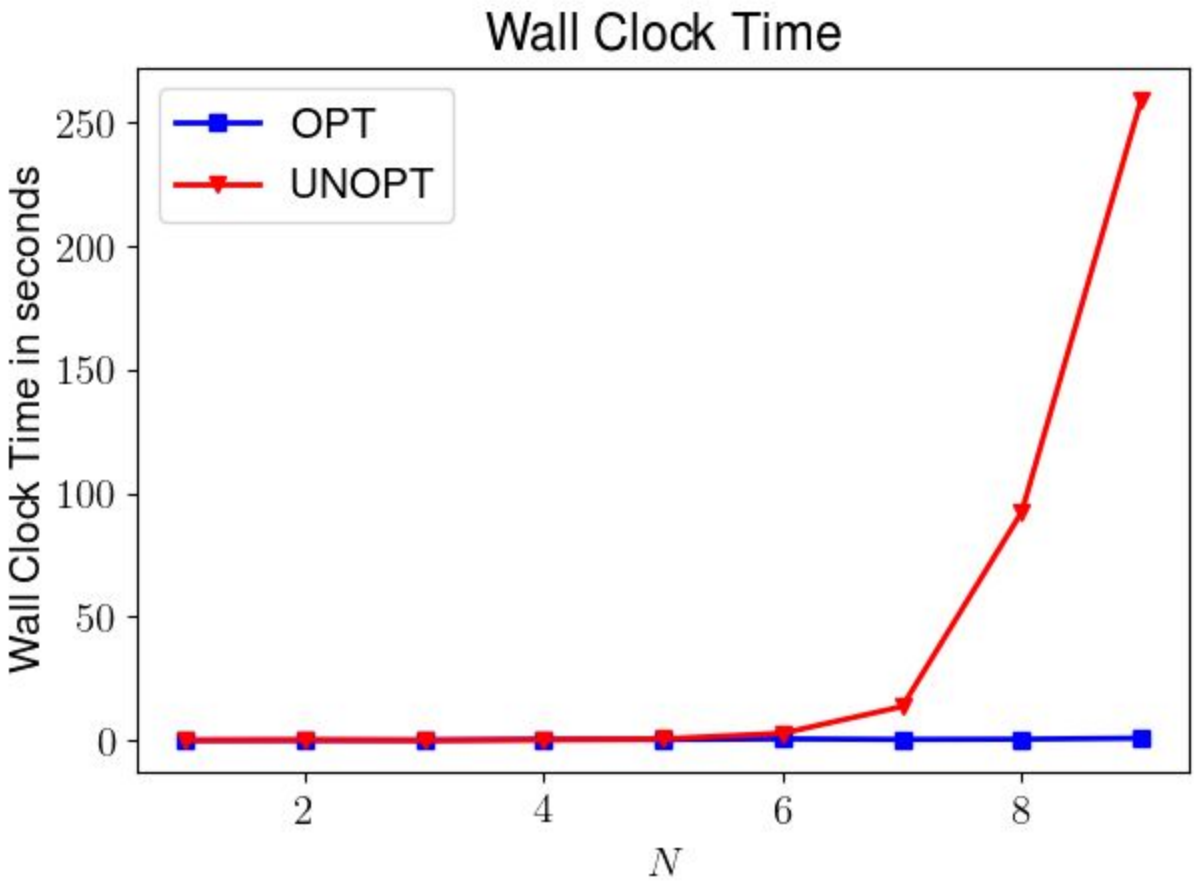
Choice of optimisation depends on application!



Multef

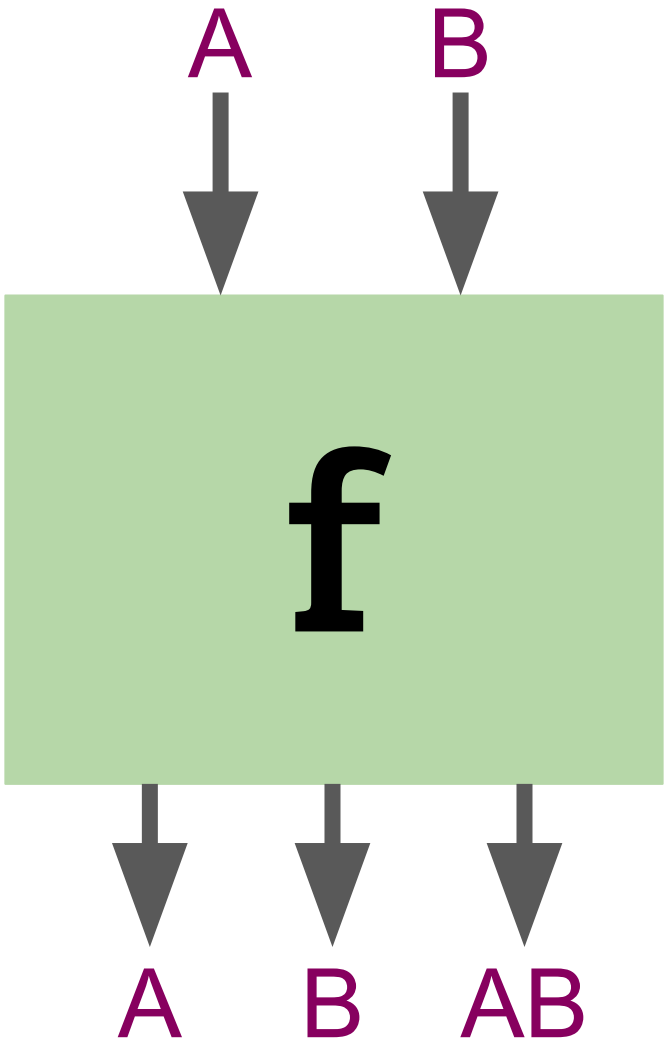
$$O(2^n) \rightarrow O(n^2)$$

Multef

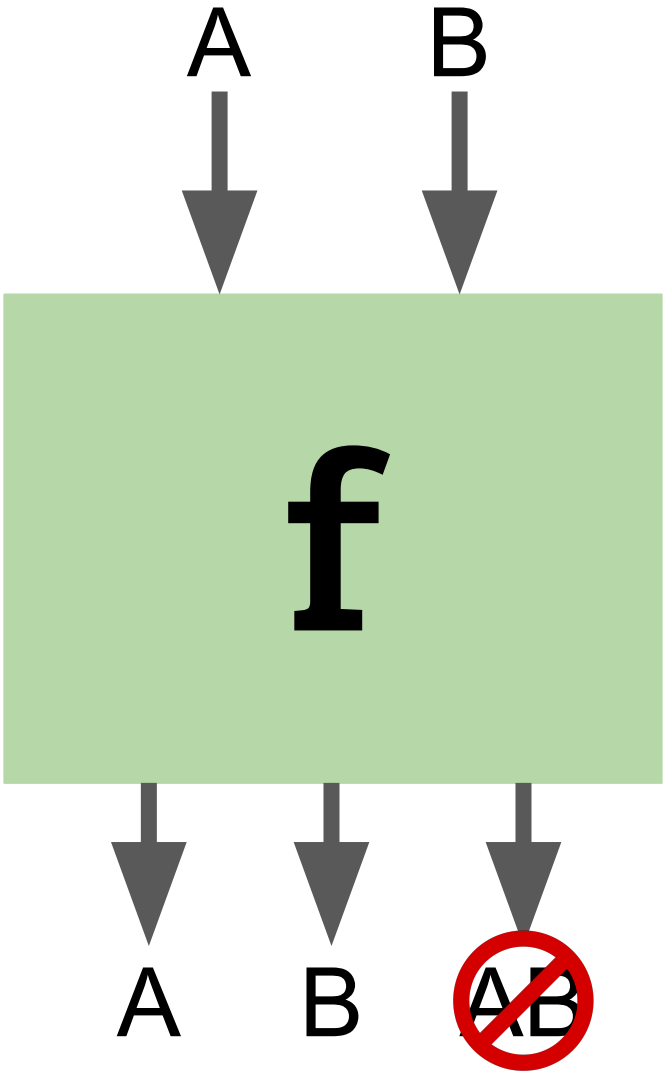

Wall Clock Time
OPT
UNOPT
Wall Clock Time in seconds
250
200
150
100
50
0
2
4
6
8
N

$$e ::= \ell \mid e \vee e \mid e \wedge e \mid \neg e$$

$$f \Downarrow (\text{Alice} \wedge \neg\text{Bob} \vee \neg\text{Alice} \wedge \text{Bob})$$


A
B
f
A
B
AB

Multef



$$\Downarrow (\text{Alice} \wedge \neg\text{Bob} \vee \neg\text{Alice} \wedge \text{Bob})$$

Multef

$$e ::= \ell \mid e \vee e \mid e \wedge e \mid \neg e$$

$$\begin{aligned}
[\![\ell]\!](t_0,t_1) &= \langle \ell \,?\, t_0 : t_1 \rangle \\
[\![e \vee e']\!](t_0,t_1) &= [\![e]\!](t_0, [\![e']\!](t_0,t_1)) \\
[\![e \wedge e']\!](t_0,t_1) &= [\![e]\!]([\![e']\!](t_0,t_1),t_1) \\
[\![\neg e]\!](t_0,t_1) &= [\![e]\!](t_1,t_0)
\end{aligned}$$

RPROJECT

$$t \Downarrow e \longrightarrow [\![e]\!](t,\bot)$$

RPROJECTSTD

$$t \Downarrow e - \texttt{std} \rightarrow t$$

$$f \Downarrow (\text{Alice} \wedge \neg\text{Bob} \vee \neg\text{Alice} \wedge \text{Bob})$$

$$\frac{\ell \models e \quad \ell \succeq t}{\ell \succeq t \Downarrow e} \qquad \frac{\ell' \sqsubseteq \ell \quad \ell \succeq t_0}{\ell \succeq \langle \ell' \,?\, t_0 : t_1 \rangle}$$

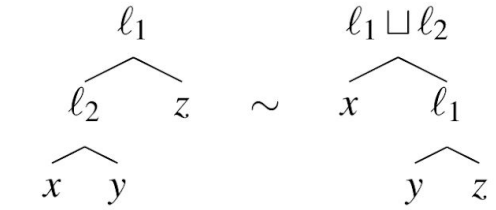
$$\ell \models \ell' \iff \ell' \sqsubseteq \ell$$

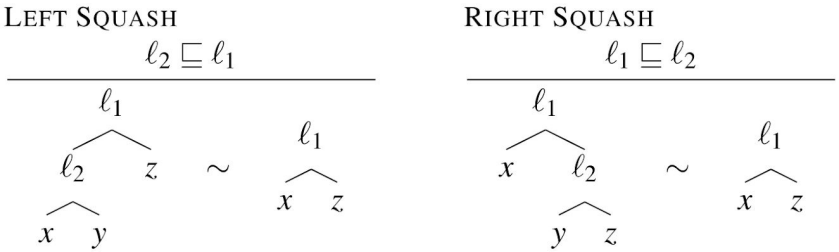
$$\ell \succeq x \qquad \frac{\ell \succeq t}{\ell \succeq \lambda x.\, t} \qquad \frac{\ell \succeq t_0 \quad \ell \succeq t_1}{\ell \succeq t_0\ t_1} \qquad \ell \succeq \texttt{unit}$$

$$\frac{\ell \succeq t}{\ell \succeq \mu x.\, t} \qquad \ell \succeq \bot \qquad \frac{\ell \models e \quad \ell \succeq t}{\ell \succeq t \Downarrow e} \qquad \frac{\ell' \sqsubseteq \ell \quad \ell \succeq t_0}{\ell \succeq \langle \ell' \,?\, t_0 : t_1 \rangle}$$

Theorem 10 (Focused Transparency)

$$\begin{aligned}
\ell \models \ell' &\iff \ell' \sqsubseteq \ell \\
\ell \models e \vee e' &\iff \ell \models e \text{ or } \ell \models e' \\
\ell \models e \wedge e' &\iff \ell \models e \text{ and } \ell \models e' \\
\ell \models \neg e &\iff \ell \not\models e
\end{aligned}$$





$$
\begin{aligned}
[\![\ell]\!](t_0,t_1) &= \langle \ell \,?\, t_0 : t_1 \rangle \\
[\![e \vee e']\!](t_0,t_1) &= [\![e]\!](t_0, [\![e']\!](t_0,t_1)) \\
[\![e \wedge e']\!](t_0,t_1) &= [\![e]\!]([\![e']\!](t_0,t_1),t_1) \\
[\![\neg e]\!](t_0,t_1) &= [\![e]\!](t_1,t_0)
\end{aligned}
$$

RPROJECT

$$t \Downarrow e \longrightarrow [\![e]\!](t,\bot)$$

RPROJECTSTD

$$t \Downarrow e - \texttt{std} \rightarrow t$$

Theorem 10 (Focused Transparency)