Software guards our secrets

Holistic Specifications

Sophia Drossopoulou, James Noble
Julian Mackay, Susan Eisenbach
Toby Murray, Mark S Milller

I AM the Bank Account

- Open World
- Third Party Code
- Future Modifications
- Trust
- Risk

Robustness

Traditional

Holistic

Robustness

Traditional

closed world

Holistic

open world

Robustness

Traditional

closed world

pre- and post conditions

Holistic

open world

temporal conditions

Robustness

Traditional

closed world

pre- and post conditions

sufficient conditions

Holistic

open world

temporal conditions

necessary conditions

Robustness

Traditional

closed world

pre- and post conditions

sufficient conditions

individual functions explict

Holistic

open world

temporal conditions

necessary conditions

emergent behaviour explicit

Robustness

Traditional

closed world

pre- and post conditions

sufficient conditions

individual functions explict

emergent behaviour implicit

Holistic

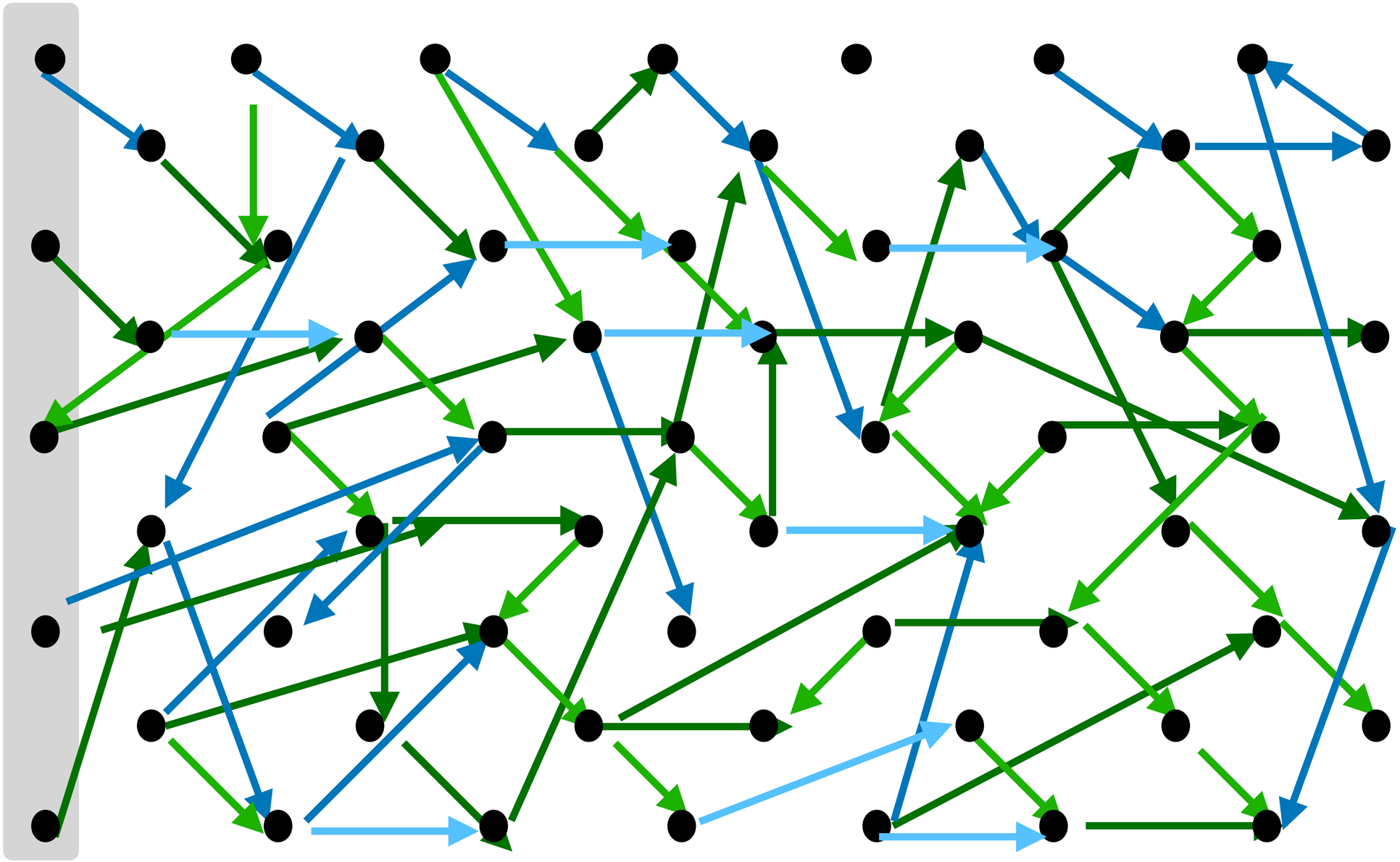
open world

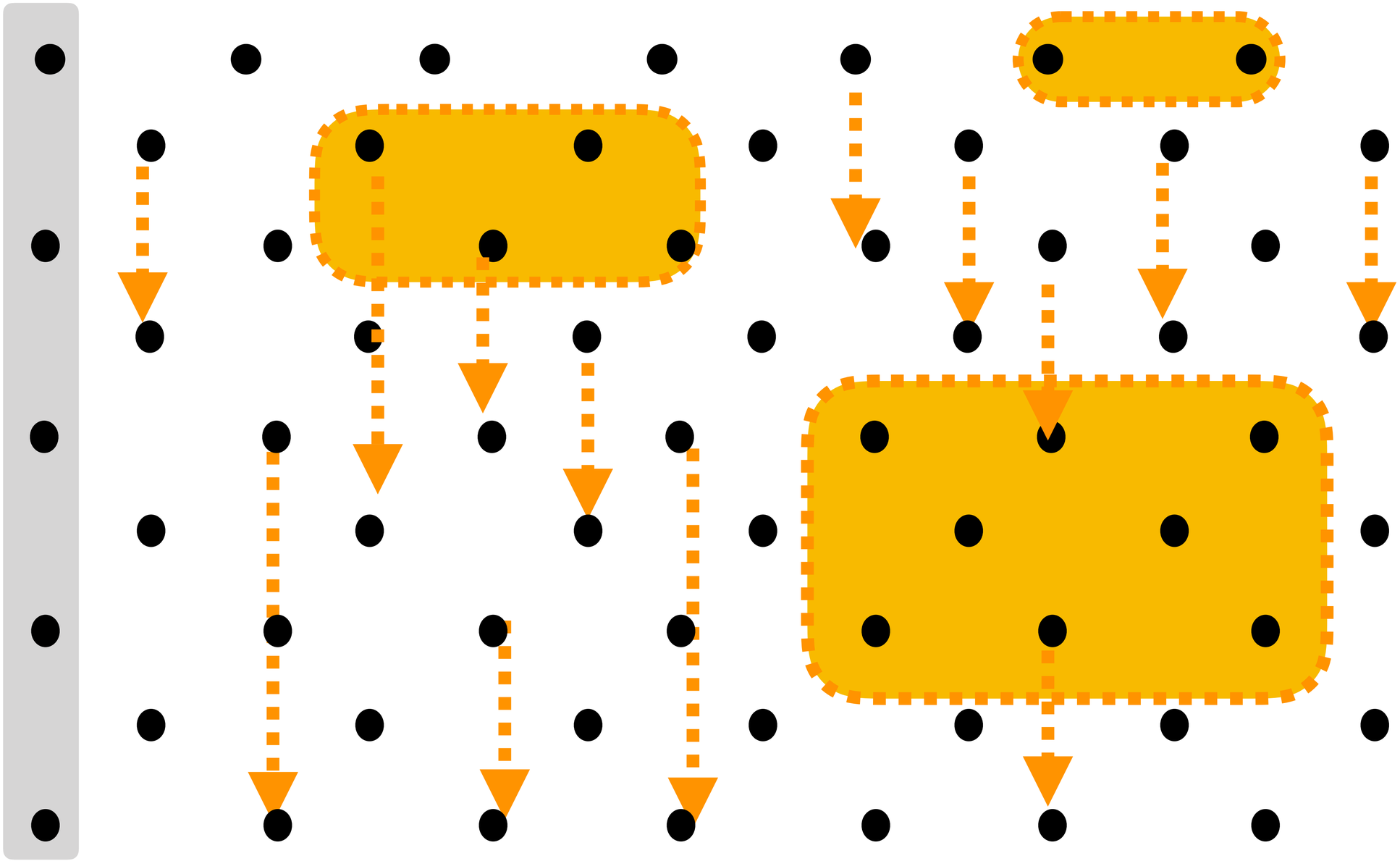
temporal conditions

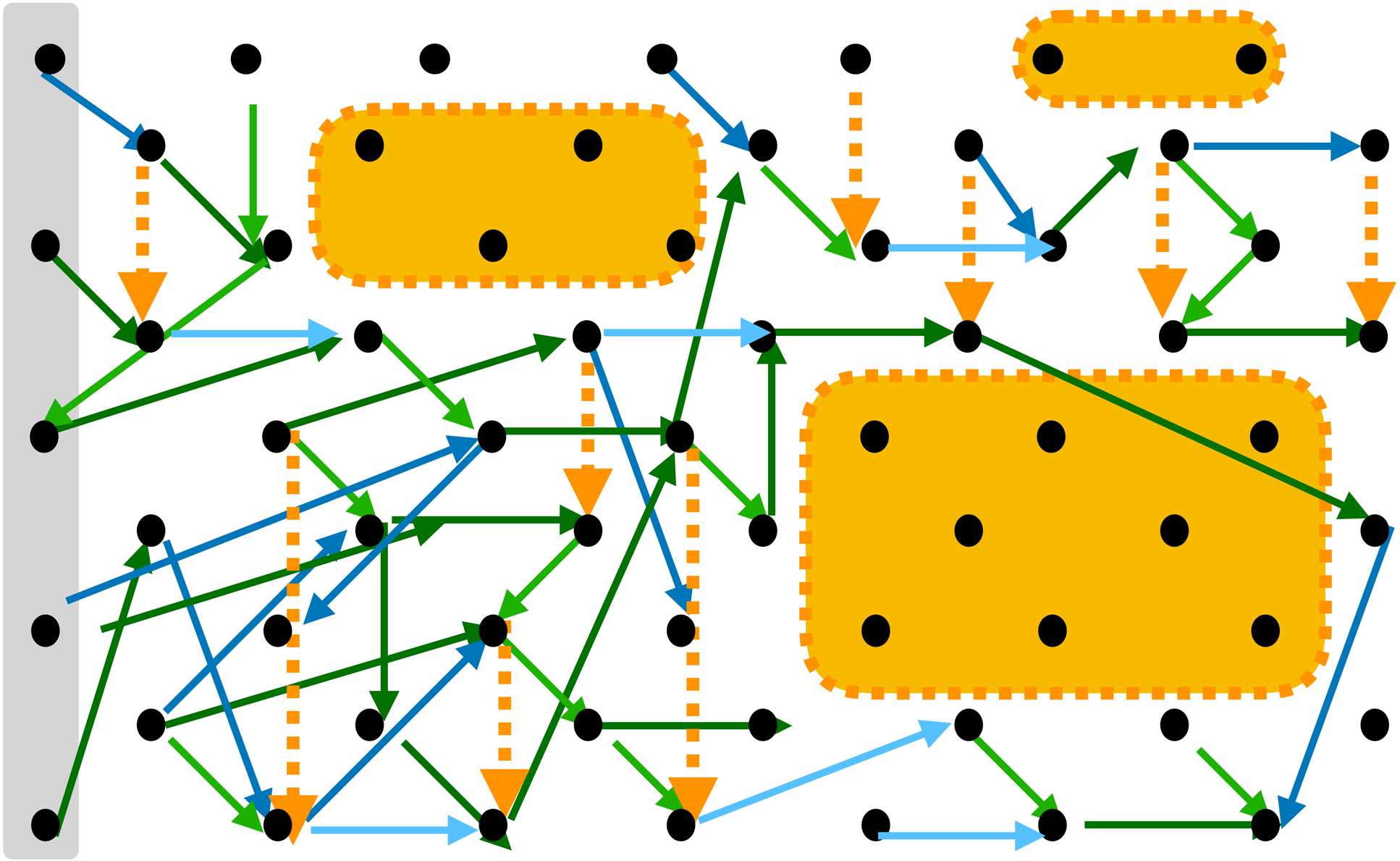
necessary conditions

emergent behaviour explicit

individual functions implicit







Chainmail

Swapsies on the Internet

First Steps towards Reasoning about Risk and Trust in an Open World

Sophia Drossopoulou[1], James Noble[2], Mark S. Miller[3]
[1]Imperial College London, [2]Victoria University Wellington, [3]Google Inc.

How to Break the Bank: Semantics of Capability Policies

Sophia Drossopoulou[1] and James Noble[2]

[1] Imperial College London s.drossopoulou@imperial.ac.uk
[2] Victoria University of Wellington kjx@ecs.vuw.ac.nz

Holistic Specifications for Robust Programs

Sophia Drossopoulou[1[0000−1111−2222−3333]], James Noble[2[0000−1111−2222−3333]], Julian Mackay[2[0000−1111−2222−3333]], and Susan Eisenbach[1[0000−1111−2222−3333]]

[1] Imperial College London {scd,susan}@springer.com

Chainmail Assertions

$$
\begin{aligned}
A \quad ::= \; & \texttt{e} \mid \texttt{e} = \texttt{e} \mid \texttt{e} : \texttt{ClassId} \mid \texttt{e} \in \texttt{S} \mid \\
& A \to A \mid A \wedge A \mid A \vee A \mid \neg A \mid \\
& \forall \texttt{x}.A \mid \forall \texttt{S} : SET.A \mid \exists \texttt{x}.A \mid \exists \texttt{S} : SET.A \mid \\
& \langle\, x \text{ access } y \,\rangle \mid \text{changes} \langle\, e \,\rangle \mid \langle\, \texttt{x} \text{ calls } \texttt{x.m(}\, \texttt{x}^* \texttt{)} \,\rangle \\
& \text{next} \langle\, A \,\rangle \mid \text{will} \langle\, A \,\rangle \mid \text{prev} \langle\, A \,\rangle \mid \text{was} \langle\, A \,\rangle \mid \\
& \langle\, \texttt{S} \text{ in } A \,\rangle \mid \text{external} \langle\, \texttt{x} \,\rangle \mid \texttt{x} \; \textbf{\textit{obeys}} \; \texttt{S}
\end{aligned}
$$

permission authority control

temporal

space viewpoint trust

ERC 20

Ethereum contract — manages tokens for clients

transfer(dst, m) — transfer **m** tokens from ***sender*** to **dst**

approve(pxy, m) — let **pxy** spend **m** tokens on behalf of ***sender***

transferFrom(src, dst, m) — transfer **m** tokens from **src** to **dst**

balance — number of tokens held by each client

Hoare Logic

precondition

$\{C\}$

postcondition

sufficient

effect

ERC 20 - transfer

$$e : \mathtt{ERC20} \land p, p'' : \mathtt{Object} \land m, m', m'' : \mathtt{Nat} \land e.\mathtt{balance}(p) = m + m' \land e.\mathtt{balance}(p'') = m'' \land \mathtt{this} = p \; \{\, e.\mathtt{transfer}(p'', m') \,\} \; e.\mathtt{balance}(p) = m \land e.\mathtt{balance}(p'') = m'' + m'$$

sufficient effect

$$e : \mathtt{ERC20} \land p, p' : \mathtt{Object} \land m, m', m'' : \mathtt{Nat} \land e.\mathtt{balance}(p) = m \land m < m' \; \{\, e.\mathtt{transfer}(p', m') \,\} \; e.\mathtt{balance}(p) = m$$

insufficient no effect

ERC 20 - delegated authority

$$e : \mathtt{ERC20} \wedge p, p', p'' : \mathtt{Object} \wedge m, m', m'', m''' : \mathtt{Nat} \wedge$$
$$e.\mathtt{balance}(p) = m + m' \wedge e.\mathtt{allowed}(p, p') = m''' + m' \wedge$$
$$e.\mathtt{balance}(p'') = m'' \wedge \mathtt{this} = p'$$
$$\{\ e.\mathtt{transferFrom}(p', p'', m')\ \}$$
$$e.\mathtt{balance}(p) = m \wedge e.\mathtt{balance}(p'') = m'' + m' \wedge e.\mathtt{allowed}(p, p') = m'''$$

$$e : \mathtt{ERC20} \wedge p, p' : \mathtt{Object} \wedge m, m', m'' : \mathtt{Nat} \wedge \mathtt{this} = p' \wedge$$
$$(e.\mathtt{balance}(p) = m \wedge m < m'' \vee e.\mathtt{allowed}(p, p') = m' \wedge m' < m'')$$
$$\{\ e.\mathtt{transferFrom}(p, p'', m'')\ \}$$
$$e.\mathtt{balance}(p) = m \wedge e.\mathtt{allowed}(p, p') = m'$$

ERC 20

Who stole super-client? take 5% from each my money?

ERC 20 — holistic transfer

effect

$$
\begin{array}{l}
\forall \mathtt{e} : \mathtt{ERC20}.\forall \mathtt{p} : \mathtt{Object}.\forall \mathtt{m}, \mathtt{m}' : \mathtt{Nat}. \\
\quad [\ \mathtt{e.balance(p)} = \mathtt{m} + \mathtt{m}' \ \wedge\ \mathsf{next}\langle \mathtt{e.balance(p)} = \mathtt{m}' \rangle \\
\qquad \longrightarrow \\
\qquad \exists \mathtt{p}', \mathtt{p}'' : \mathtt{Object}. \\
\qquad [\ \langle\, \mathtt{p}\ \mathsf{calls}\ \mathtt{e.transfer(\,p', m)}\ \rangle\ \vee \\
\qquad\quad \mathtt{e.allowed(p, p'')} \geq \mathtt{m} \wedge \langle\, \mathtt{p''}\ \mathsf{calls}\ \mathtt{e.transferFrom(\,p', m)}\ \rangle\ ] \\
\quad]
\end{array}
$$

necessary

ERC 20 — delegated authority

$$
\begin{array}{l}
\forall \mathtt{e} : \mathtt{ERC20}.\forall \mathtt{p}, \mathtt{p}' : \mathtt{Object}.\forall \mathtt{m} : \mathtt{Nat}. \\
\quad [\ \mathtt{e.allowed}(\mathtt{p}, \mathtt{p}') = \mathtt{m} \\
\qquad \longrightarrow \\
\qquad \mathcal{P}rev\langle\ \langle\ \mathtt{p}\ \textsf{calls}\ \mathtt{e.approve}(\ \mathtt{p}', \mathtt{m})\ \rangle \\
\qquad\qquad \vee \\
\qquad\qquad \mathtt{e.allowed}(\mathtt{p}, \mathtt{p}') = \mathtt{m} \wedge \\
\qquad\qquad \neg(\langle\ \mathtt{p}'\ \textsf{calls}\ \mathtt{e.transferFrom}(\ \mathtt{p}, \_)\ \rangle \vee \langle\ \mathtt{p}\ \textsf{calls}\ \mathtt{e.approve}(\ \mathtt{p}, \_)\ \rangle) \\
\qquad\qquad \vee \\
\qquad\qquad \exists \mathtt{p}'' : \mathtt{Object}.\exists \mathtt{m}' : \mathtt{Nat}. \\
\qquad\qquad [\mathtt{e.allowed}(\mathtt{p}, \mathtt{p}') = \mathtt{m} + \mathtt{m}' \wedge \langle\ \mathtt{p}'\ \textsf{calls}\ \mathtt{e.transferFrom}(\ \mathtt{p}'', \mathtt{m}')\ \rangle] \\
\qquad\quad \rangle \\
\quad]
\end{array}
$$

effect

necessary

ERC 20

Separate Concerns:

Transfer

Delegation

Necessary conditions permit effects

DAO

Ethereum contract — "

d.j n(m — *nder* sf n in DA

d.r bay — *s der* w nt he n ney ba k

r.send(m) — *sender* t rs **m**

d.balance(client) — n b f to s he by each nt

Repay

19 Methods

Attack

DAO — holistic balance

necessary

$$\forall d : \mathtt{DAO}.\forall p.\forall : m : \mathtt{Nat}.$$
$$[\ \mathtt{d.Balance(p)} = m \longrightarrow [\ \mathsf{prev}\langle\langle\, p\ \mathsf{calls}\ \mathtt{d.repay(\_)}\ \rangle\rangle \wedge m = 0 \vee$$
$$\mathsf{prev}\langle\langle\, p\ \mathsf{calls}\ \mathtt{d.join(m)}\ \rangle\rangle \vee$$
$$\ldots\ ]$$
$$]$$

effect

DAO — holistic repayments

covers balances

$$\forall d : \mathtt{DAO}.\forall p : \mathtt{Any}.\ \Sigma\, d.\mathtt{balances}(p) \leq d.\mathtt{ether}$$

pays when asked

$$\forall d : \mathtt{DAO}.\forall p : \mathtt{Any}.\forall m : \mathtt{Nat}.$$
$$[\ d.\mathtt{balance}(p) = m \wedge \langle\, p\ \mathsf{calls}\ d.\mathtt{repay}(\_)\ \rangle$$
$$\longrightarrow\ \ \mathsf{will}\langle\, \langle\, d\ \mathsf{calls}\ p.\mathtt{send}(m)\ \rangle\, \rangle\ ]$$
$$\longrightarrow\ \ p.\mathtt{funds} = \mathsf{prev}\langle p.\mathtt{funds}\rangle + m$$

best effort

$$\forall d : \mathtt{DAO}.\forall p : \mathtt{Any}.\forall m : \mathtt{Nat}.$$
$$\langle\, d\ \mathsf{calls}\ p.\mathtt{send}(m)\ \rangle\ \longrightarrow\ d.\mathtt{ether} \geq m$$

eventual payments

DAO

Payment and repayment

Current balance vs pending repayment

Best effort vs ideal effect

Where is the trust boundary?

Digested week
Boris Johnson

Sketch writing feels impossible in the age of Dom and Dommer

John Crace

As satire and reality collide I find it difficult to see the funny side of anything going on in Westminster

@JohnJCrace
Fri 6 Sep 2019 12.30 BST

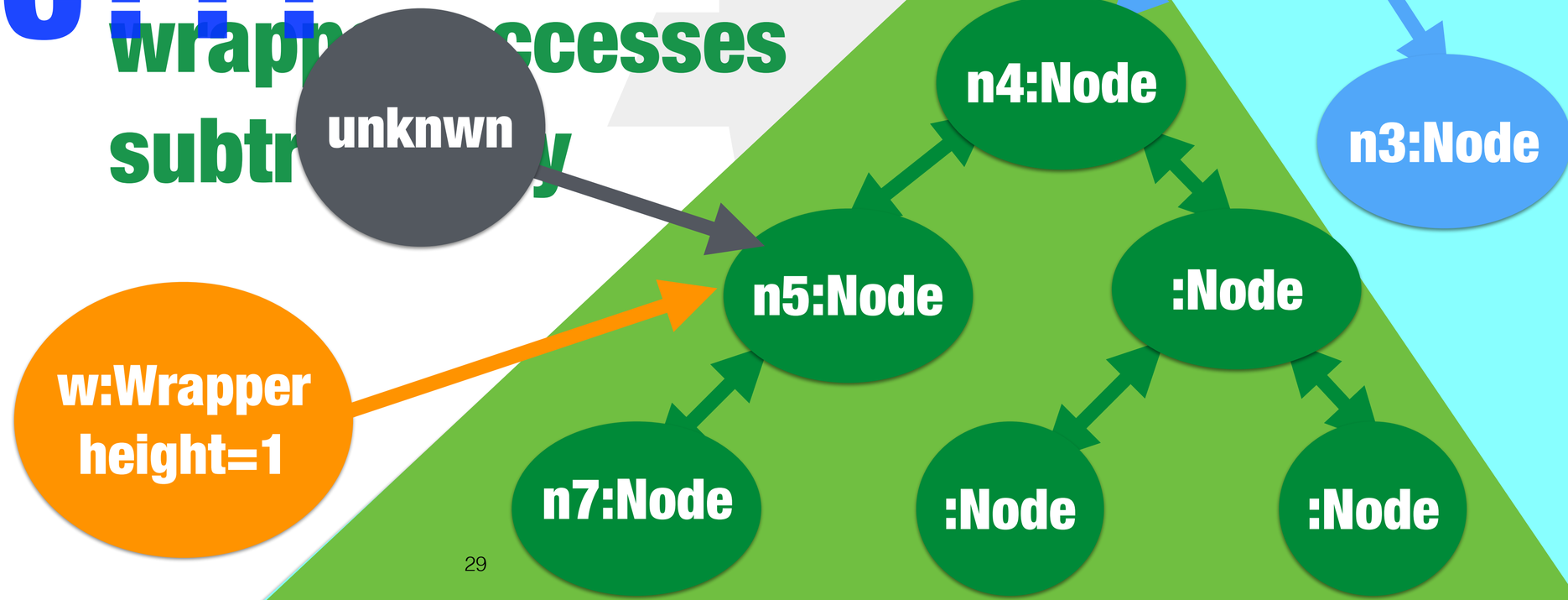
65 1,015

Classic DOM

classical spec???

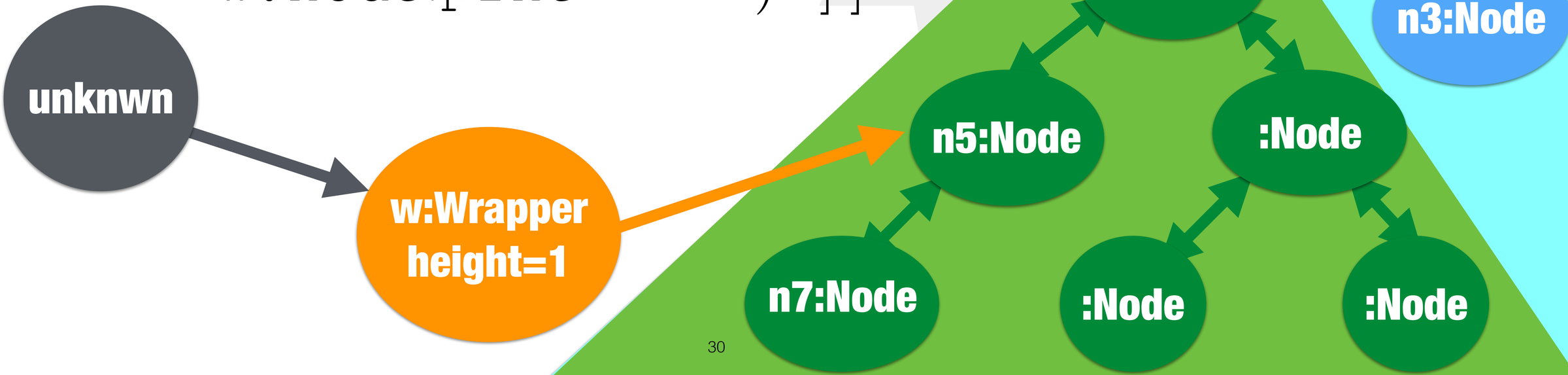
any node leaks whole tree

wrapp ccesses subtr y



29

$$
\begin{array}{l}
\forall \mathtt{S} : \mathtt{Set}.\forall \mathtt{nd} : \mathtt{Node}.\forall \mathtt{o} : \mathtt{Object}. \\
[\ \langle\, \textsf{will}\langle\, \textsf{changes}\langle\, \mathtt{nd.property} \,\rangle\,\rangle\, \textsf{in}\, \mathtt{S} \,\rangle \\
\quad \longrightarrow \\
\quad \exists \mathtt{o}.[\, \mathtt{o} \in \mathtt{S} \ \wedge\ \neg(\mathtt{o} : \mathtt{Node}) \ \wedge\ \neg(\mathtt{o} : \mathtt{Wrapper}) \ \wedge \\
\qquad [\, \exists \mathtt{nd}' : \mathtt{Node}.\langle\, \mathtt{o}\ \textsf{access}\ \mathtt{nd}' \,\rangle \ \vee \\
\qquad\quad \exists \mathtt{w} : \mathtt{Wrapper}.\exists k{:}\mathbb{N}.(\, \langle\, \mathtt{o}\ \textsf{access}\ \mathtt{w} \,\rangle \\
\qquad\qquad \wedge\ \mathtt{nd.prnt}^{k} \\
\qquad\qquad\quad = \mathtt{w.node.prnt}^{\mathtt{w.hgt}}) \ ]] \\
]
\end{array}
$$

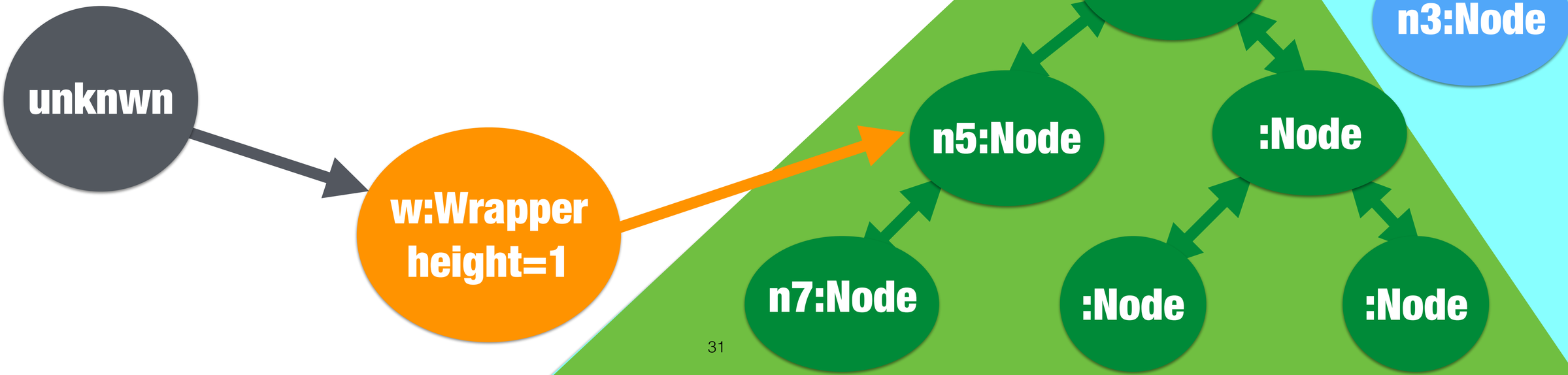


30

```
method usingWrappers(unknwn)
     {
    n1=Node(null,"fixed");
    n2=Node(n1,"robust");
    n3=Node(n2,"const");
    n4=Node(n3,"volatile");
    n5=Node(n4,"variable");
    n6=Node(n5,"ethereal");
    w=Wrapper(n5,1);
```

```
unknwn.untrusted(w);

assert n2.property=="
    robust"
...
```

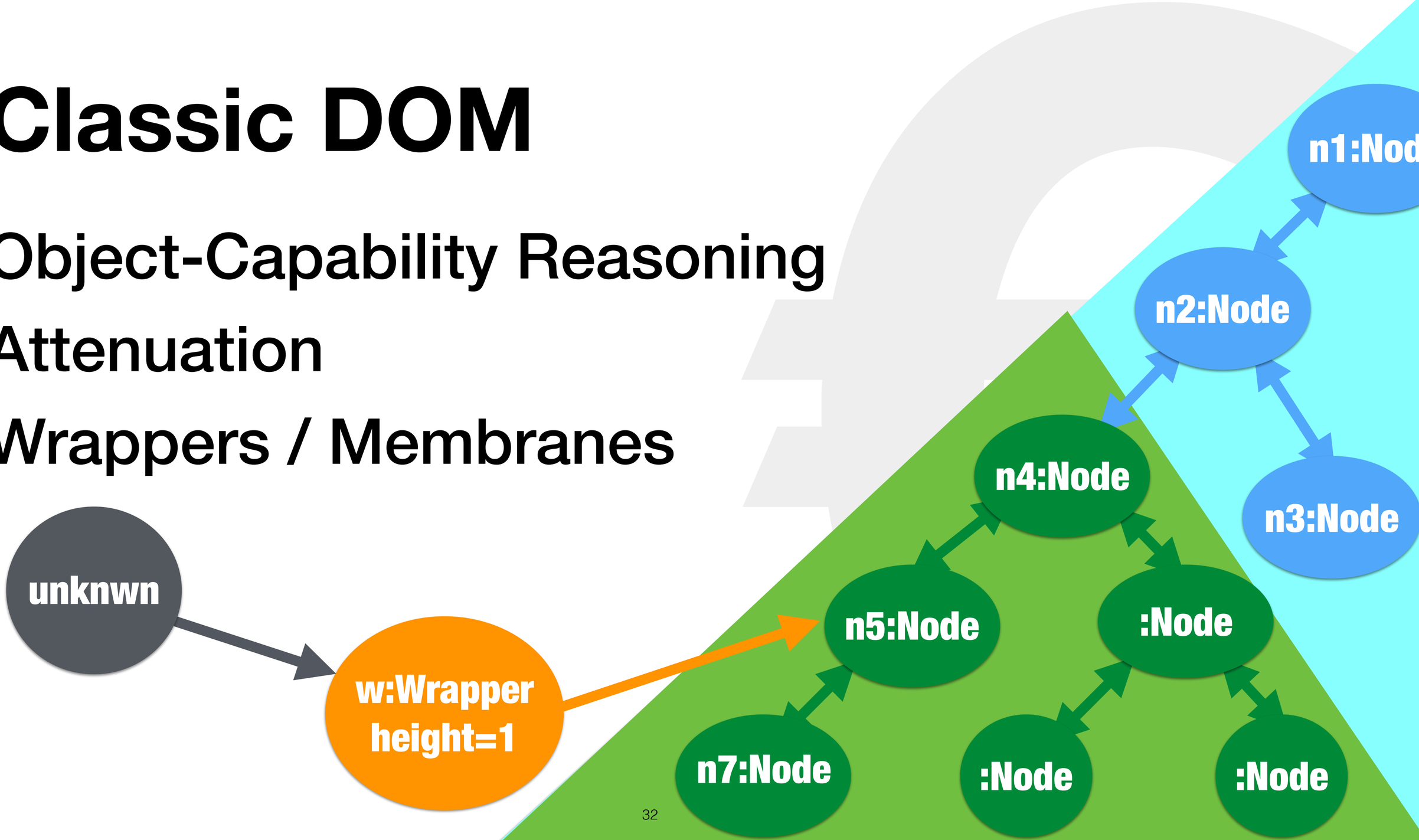


Classic DOM

Object-Capability Reasoning

Attenuation

Wrappers / Membranes



32

Chainmail Semantics

Underlying OO language model

Kripke worlds

Two-module execution

Assertions

L_{oo} syntax

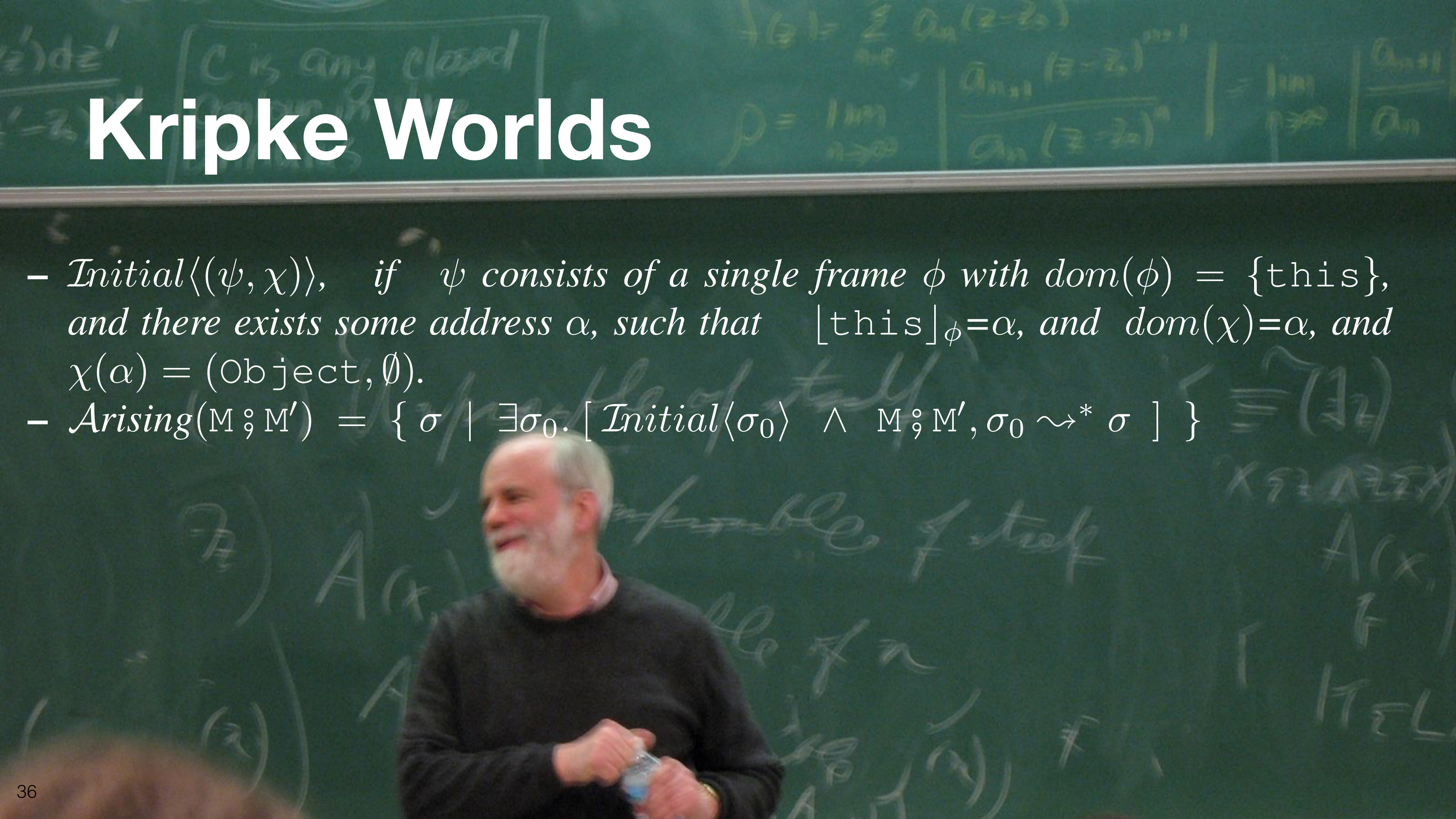
$$
\begin{array}{lll}
\mathit{ClassDescr} & ::= & \texttt{class}\ \mathit{ClassId}\ \{\ \ (\ \mathit{FieldDecl}\ )^*\ \ (\ \mathit{MethDecl}\ )^*\ \ (\ \mathit{GhosDecl}\ )^*\ \ \} \\
\mathit{FieldDecl} & ::= & \texttt{field f} \\
\mathit{MethDecl} & ::= & \texttt{method m(}\ \texttt{x}^*\texttt{)}\ \{\ \ \mathit{Stmts}\ \ \} \\
\mathit{Stmts} & ::= & \mathit{Stmt}\ \mid\ \mathit{Stmt}\ \texttt{;}\ \mathit{Stmts} \\
\mathit{Stmt} & ::= & \texttt{x.f := x}\ \mid\ \texttt{x := x.f}\ \mid\ \texttt{x := x.m(}\ \texttt{x}^*\texttt{)}\ \mid\ \texttt{x := new C(}\ \texttt{x}^*\texttt{)}\ \mid\ \texttt{return x} \\
\mathit{GhostDecl} & ::= & \texttt{ghost f(}\ \texttt{x}^*\ \texttt{)}\ \{\ \texttt{e}\ \} \\
\texttt{e} & ::= & \texttt{true}\ \mid\ \texttt{false}\ \mid\ \texttt{null}\ \mid\ \texttt{x}\ \mid\ \texttt{e=e}\ \mid\ \texttt{if e then e else e}\ \mid\ \texttt{e.f(}\ \texttt{e}^*\ \texttt{)} \\
\texttt{x, f, m} & ::= & \texttt{Identifier}
\end{array}
$$

$\mathbf{L_{oo}}$ runtime

- *Continuations are either statements (as defined in Definition 12) or a marker,* `x:=` $\bullet$*, for a nested call followed by statements to be executed once the call returns.*
  Continuation $::=$ *Stmts* $\mid$ `x:=` $\bullet$ `;` *Stmts*
- *Frames,* ϕ*, consist of a code stub and a mapping from identifiers to values:*
  $\phi \in$ *CodeStub* $\times$ `Ident` $\rightarrow Value$*,*
- *Stacks,* ψ*, are sequences of frames,* $\psi ::= \phi \mid \phi \cdot \psi$*.*
- *Objects consist of a class identifier, and a partial mapping from field identifier to values:*
  $Object =$ `ClassID` $\times$ (`FieldId` $\rightarrow Value$)*.*
- *Heaps,* χ*, are mappings from addresses to objects:* $\chi \in$ `Addr` $\rightarrow Object$*.*
- *Runtime configurations,* σ*, are pairs of stacks and heaps,* $\sigma ::= (\psi, \chi)$*.*

Kripke Worlds

- $\mathcal{I}nitial\langle(\psi, \chi)\rangle$, *if* ψ *consists of a single frame* ϕ *with* $dom(\phi) = \{\texttt{this}\}$, *and there exists some address* α, *such that* $\lfloor\texttt{this}\rfloor_\phi = \alpha$, *and* $dom(\chi) = \alpha$, *and* $\chi(\alpha) = (\texttt{Object}, \emptyset)$.
- $\mathcal{A}rising(\texttt{M;M}') = \{\, \sigma \mid \exists \sigma_0.\, [\mathcal{I}nitial\langle\sigma_0\rangle \wedge \texttt{M;M}', \sigma_0 \rightsquigarrow^* \sigma\ ]\ \}$



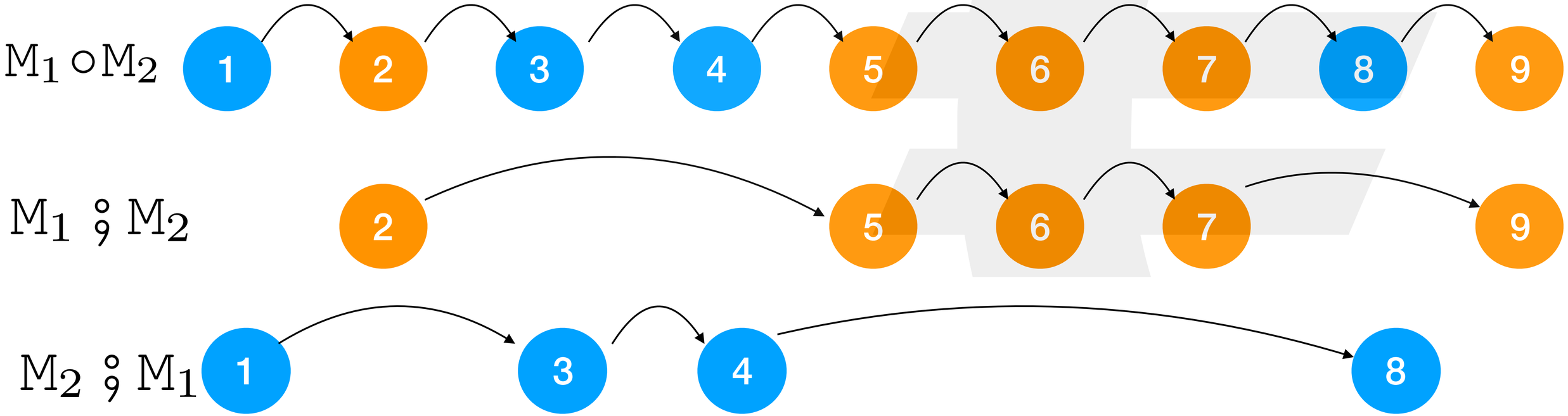
Two-module execution

- $\mathtt{M} \fatsemi \mathtt{M}', \sigma \rightsquigarrow \sigma'$ *if* *there exist* $n \geq 2$ *and runtime configurations* σ_1, ... σ_n, *such that*
  - $\sigma = \sigma_1$, *and* $\sigma_n = \sigma'$.
  - $\mathtt{M} \circ \mathtt{M}', \sigma_i \rightsquigarrow \sigma'_{i+1}$, *for* $1 \leq i \leq n-1$
  - $\mathcal{C}lass(\mathtt{this})_{\sigma} \notin dom(\mathtt{M})$, *and* $\mathcal{C}lass(\mathtt{this})_{\sigma'} \notin dom(\mathtt{M})$,
  - $\mathcal{C}lass(\mathtt{this})_{\sigma_i} \in dom(\mathtt{M})$, *for* $2 \leq i \leq n-2$

M — internal module — Internal method calls are abstracted away

M' - external module

Two-module execution



Internal $\wr$ External — Internal method calls are abstracted away

Assertions

$$
\begin{aligned}
A \quad ::= \; & \texttt{e} \mid \texttt{e} = \texttt{e} \mid \texttt{e} : \texttt{ClassId} \mid \texttt{e} \in \texttt{S} \mid \\
& A \to A \mid A \wedge A \mid A \vee A \mid \neg A \mid \\
& \forall \texttt{x}.A \mid \forall \texttt{S} : SET.A \mid \exists \texttt{x}.A \mid \exists \texttt{S} : SET.A \mid \\
& \langle\, x \;\mathsf{access}\; y \,\rangle \mid \mathsf{changes}\langle\, e \,\rangle \mid \langle\, \texttt{x} \;\mathsf{calls}\; \texttt{x.m(}\, \texttt{x}^* \texttt{)} \,\rangle \\
& \mathsf{next}\langle\, A \,\rangle \mid \mathsf{will}\langle\, A \,\rangle \mid \mathsf{prev}\langle\, A \,\rangle \mid \mathsf{was}\langle\, A \,\rangle \mid \\
& \langle\, \texttt{S} \;\mathsf{in}\; A \,\rangle \mid \mathsf{external}\langle\, \texttt{x} \,\rangle \mid \texttt{x} \;\boldsymbol{obeys}\; \texttt{S}
\end{aligned}
$$

permission authority control

temporal

space viewpoint trust

Permission

- $\mathtt{M}\,;\,\mathtt{M}', \sigma \models \langle\, \mathtt{x}\ \mathsf{access}\ \mathtt{y}\,\rangle$ *if* $\lfloor \mathtt{x} \rfloor_\sigma$ *and* $\lfloor \mathtt{y} \rfloor_\sigma$ *are defined, and*
  - $\lfloor \mathtt{x} \rfloor_\sigma = \lfloor \mathtt{y} \rfloor_\sigma$, *or*
  - $\lfloor \mathtt{x.f} \rfloor_\sigma = \lfloor \mathtt{y} \rfloor_\sigma$, *for some field* $\mathtt{f}$, *or*
  - $\lfloor \mathtt{x} \rfloor_\sigma = \lfloor \mathtt{this} \rfloor_\sigma$ *and* $\lfloor \mathtt{y} \rfloor_\sigma = \lfloor \mathtt{z} \rfloor_\sigma$, *for some variable* $\mathtt{z}$ *and* $\mathtt{z}$ *appears in* $\sigma.\mathtt{contn}$.

Control

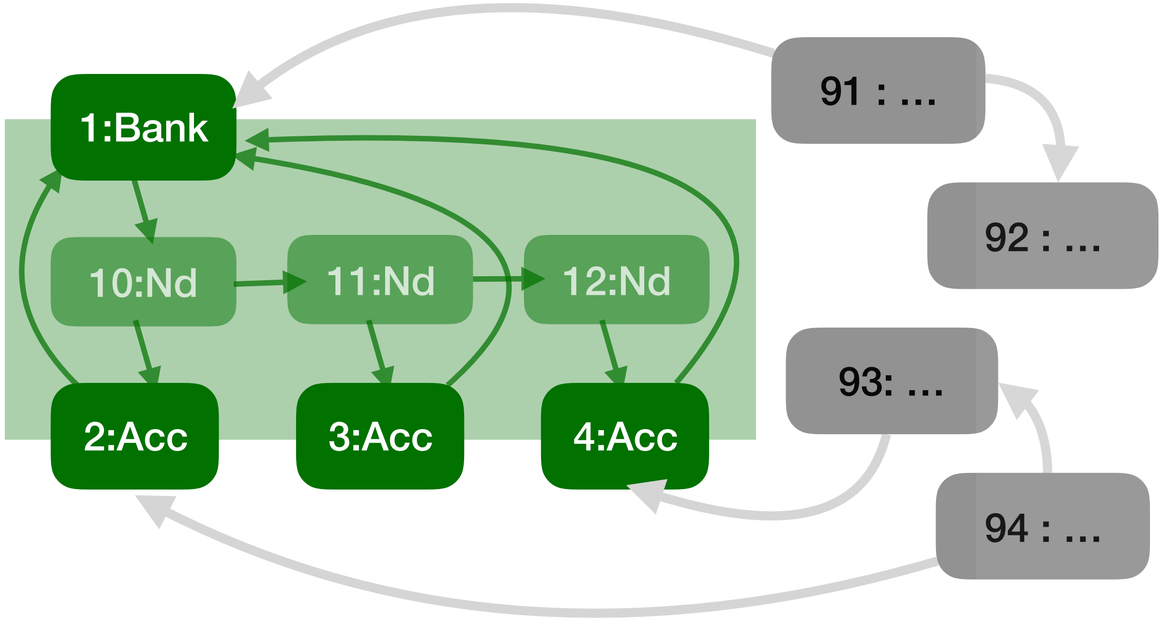
- $\mathtt{M} \,;\, \mathtt{M}', \sigma \models \langle\, \mathtt{x}\ \mathsf{calls}\ \mathtt{y.m}(\,\mathtt{z}_1, \ldots \mathtt{z}_n)\,\rangle$ *if* $\lfloor \mathtt{x} \rfloor_\sigma$, $\lfloor \mathtt{y} \rfloor_\sigma$, $\lfloor \mathtt{z}_1 \rfloor_\sigma$, ... $\lfloor \mathtt{z}_n \rfloor_\sigma$ *are defined, and*
  - $\lfloor \mathtt{this} \rfloor_\sigma = \lfloor \mathtt{x} \rfloor_\sigma$, *and*
  - $\sigma.\mathtt{contn} = \mathtt{u.m}(\mathtt{v}_1, .. \mathtt{v}_n); \_$, *for some* $\mathtt{u}, \mathtt{v}_1, \ldots \mathtt{v}_n$, *and*
  - $\lfloor \mathtt{y} \rfloor_\sigma = \lfloor \mathtt{u} \rfloor_\sigma$, *and* $\lfloor \mathtt{z}_i \rfloor_\sigma = \lfloor \mathtt{v}_i \rfloor_\sigma$, *for all* i.

Viewpoint

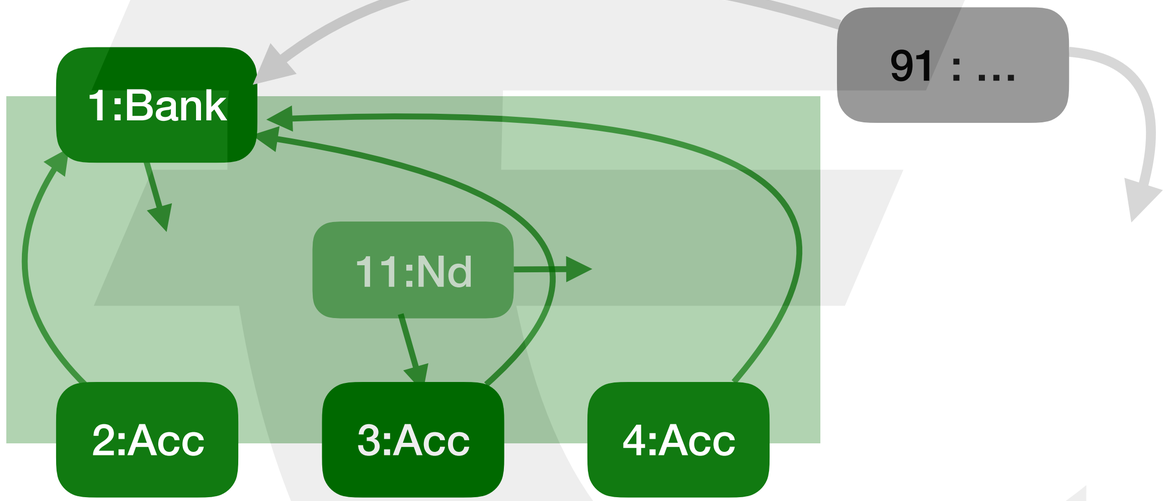
- $\mathtt{M} \fatsemi \mathtt{M}', \sigma \models \mathsf{external}\langle\, \mathtt{x} \,\rangle \quad \textit{if} \quad \lfloor \mathtt{x} \rfloor_\sigma \textit{ is defined and } \mathcal{C}lass(\lfloor \mathtt{x} \rfloor_\sigma)_\sigma \notin dom(\mathtt{M})$
- $\mathtt{M} \fatsemi \mathtt{M}', \sigma \models \mathsf{internal}\langle\, \mathtt{x} \,\rangle \quad \textit{if} \quad \lfloor \mathtt{x} \rfloor_\sigma \textit{ is defined and } \mathcal{C}lass(\lfloor \mathtt{x} \rfloor_\sigma)_\sigma \in dom(\mathtt{M})$

Space

Restriction



Restrict heap to 1,2,3,4, 11, 91



Space

Restriction:

- $\sigma{\downarrow_{\mathtt{S}}} \triangleq (\psi, \chi'), \quad \textit{if} \quad \sigma{=}(\psi, \chi), \textit{ and } dom(\chi') = \lfloor \mathtt{S} \rfloor_\sigma, \textit{and} \forall \alpha \in dom(\chi').\chi(\alpha) = \chi'(\alpha)$

Space:

- $\mathtt{M} ; \mathtt{M}', \sigma \models \langle\, A \,\mathsf{in}\, \mathtt{S} \,\rangle \quad \textit{if} \quad \mathtt{M} ; \mathtt{M}', \sigma{\downarrow_{\mathtt{S}}} \models A.$

Time

Temporal Operators:

- $\mathtt{M} \fatsemi \mathtt{M}', \sigma \models \mathsf{prev}\langle\, A \,\rangle \quad \textit{if} \quad \forall \sigma_1, \sigma_2.[\ \mathcal{I}\textit{nitial}\langle \sigma_1 \rangle \wedge \mathtt{M} \fatsemi \mathtt{M}', \sigma_1 \rightsquigarrow^* \sigma_2 \wedge \mathtt{M} \fatsemi \mathtt{M}', \sigma_2 \rightsquigarrow \sigma \longrightarrow \mathtt{M} \fatsemi \mathtt{M}', \sigma \triangleleft \sigma_2 \models A\ ]$
- $\mathtt{M} \fatsemi \mathtt{M}', \sigma \models \mathsf{was}\langle\, A \,\rangle \quad \textit{if} \quad \forall \sigma_1.[\ \mathcal{I}\textit{nitial}\langle \sigma_1 \rangle \wedge \mathtt{M} \fatsemi \mathtt{M}', \sigma_1 \rightsquigarrow^* \sigma \longrightarrow (\ \exists \sigma_2.\mathtt{M} \fatsemi \mathtt{M}', \sigma_1 \rightsquigarrow^* \sigma_2 \wedge \mathtt{M} \fatsemi \mathtt{M}', \sigma_2 \rightsquigarrow^* \sigma \wedge \mathtt{M} \fatsemi \mathtt{M}', \sigma \triangleleft \sigma_2 \models A\ )]$

- $\mathtt{M} \fatsemi \mathtt{M}', \sigma \models \mathsf{next}\langle\, A \,\rangle \quad \textit{if} \quad \exists \sigma'.[\ \mathtt{M} \fatsemi \mathtt{M}', \phi \rightsquigarrow \sigma' \wedge \mathtt{M} \fatsemi \mathtt{M}', \sigma \triangleleft \sigma' \models A\ ],$ *and where* ϕ *is so that* $\sigma = (\phi \cdot \_, \_)$.
- $\mathtt{M} \fatsemi \mathtt{M}', \sigma \models \mathsf{will}\langle\, A \,\rangle \quad \textit{if} \quad \exists \sigma'.[\ \mathtt{M} \fatsemi \mathtt{M}', \phi \rightsquigarrow^* \sigma' \wedge \mathtt{M} \fatsemi \mathtt{M}', \sigma \triangleleft \sigma' \models A\ ],$ *and where* ϕ *is so that* $\sigma = (\phi \cdot \_, \_)$.

Time

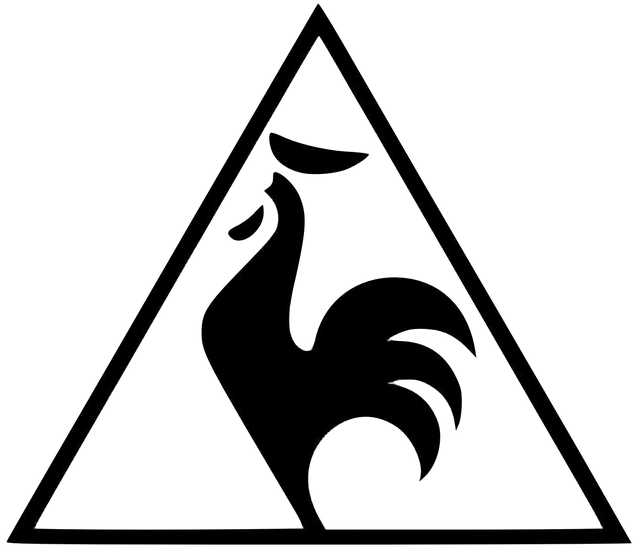
Adaptation: $\sigma \triangleleft \sigma'$

take top frame from σ, and keep those bindings

take heap from σ', rename continuation consistently

– $\sigma \triangleleft \sigma' \triangleq (\phi'' \cdot \psi', \chi')$ *if*

- $\sigma = (\phi \cdot \_, \_)$, $\sigma' = (\phi' \cdot \psi', \chi')$, *and*
- $\phi = (\texttt{contn}, \beta)$, $\phi' = (\texttt{contn}', \beta')$, $\phi'' = (\texttt{contn}'[\texttt{zs}/\texttt{zs}'], \beta[\texttt{zs}' \mapsto \beta'(\texttt{zs})])$, *where*
- $\texttt{zs} = dom(\beta)$, $\texttt{zs}'$ *is a set of variables with the same cardinality as* $\texttt{zs}$, *and*
- *all variables in* $\texttt{zs}'$ *are fresh in* β *and in* β'.



Properties of Linking	(1) moduleLinking_associative (2) moduleLinking_commutative_1 (3) moduleLinking_commutative_2 (4) linking_preserves_reduction
(1) $A \wedge \neg A \equiv \texttt{false}$ (2) $A \vee \neg A \equiv \texttt{true}$ (3) $A \vee A' \equiv A' \wedge A$ (4) $A \wedge A' \equiv A' \wedge A$ (5) $(A \vee A') \vee A'' \equiv A \vee (A' \vee A'')$	(1) sat_and_nsat_equiv_false (2) - (3) and_commutative (4) or_commutative (5) or_associative
(1) $A \wedge \neg A \equiv \texttt{false}$ (2) $A \vee \neg A \equiv \texttt{true}$ (3) $A \vee A' \equiv A' \wedge A$ (4) $A \wedge A' \equiv A' \wedge A$ (5) $(A \vee A') \vee A'' \equiv A \vee (A' \vee A'')$ (6) $(A \vee A') \wedge A'' \equiv (A \vee A'') \wedge (A' \vee A'')$ (7) $(A \wedge A') \vee A'' \equiv (A \wedge A'') \vee (A' \wedge A'')$ (8) $\neg(A \wedge A') \equiv (\neg A \vee \neg A')$ (9) $\neg(A \vee A') \equiv (\neg A \wedge \neg A')$ (10) $\neg(\exists x.A) \equiv \forall x.(\neg A)$ (11) $\neg(\exists S.A) \equiv \forall S.(\neg A)$ (12) $\neg(\forall x.A) \equiv \exists x.(\neg A)$ (13) $\neg(\forall S.A) \equiv \exists S.(\neg A)$	(1) sat_and_nsat_equiv_false (2) - (3) and_commutative (4) or_commutative (5) or_associative (6) and_distributive (7) or_distributive (8) neg_distributive_and (9) neg_distributive_or (10) not_ex_x_all_not (11) not_ex_Σ_all_not (12) not_all_x_ex_not (13) not_all_Σ_ex_not

Grant Matcher

The Left Hand of Equals
Noble, Black, Bruce, Homer, Miller

Grant Matcher

$ $

The Left Hand of Equals
Noble, Black, Bruce, Homer, Miller

Grant Matcher

$ $

C C

The Left Hand of Equals
Noble, Black, Bruce, Homer, Miller

```
method match ( amount : Number,
    aliceAccount : Account, aliceCharity : Charity,
    danaAccount : Account, danaCharity : Charity)
          -> Boolean {
  if ((aliceAccount.balance >= amount) &&
      (danaAccount.balance >= amount) &&
       aliceCharity.validates(danaCharity))

      then {
        aliceCharity.accept(aliceAccount, amount)
        danaCharity.accept(danaAccount, amount)
        return true}
      else {return false}
}
```


The Left Hand of Equals
Noble, Black, Bruce, Homer, Miller

Grant Matcher

```
class fakeCharity → Charity {
    def backPocket : Account = ...
    method validates(other) { true }
    method accept(donation, amount) {
      backPocket.accept(donation, amount) }

}
```

The Left Hand of Equals
Noble, Black, Bruce, Homer, Miller

```
method match ( amount : Number,
    aliceAccount : Account, aliceCharity : Charity,
    danaAccount : Account, danaCharity : Charity)
          -> Boolean {
  if ((aliceAccount.balance >= amount) &&
      (danaAccount.balance >= amount) &&
       aliceCharity.validates(danaCharity))

      then {
        aliceCharity.accept(aliceAccount, amount)
        danaCharity.accept(danaAccount, amount)
        return true}
      else {return false}
}
```


The Left Hand of Equals
Noble, Black, Bruce, Homer, Miller

```
method match ( amount : Number,
    aliceAccount : Account, aliceCharity : Charity,
    danaAccount : Account, danaCharity : Charity)
          -> Boolean {
  if ((aliceAccount.balance >= amount) &&
      (danaAccount.balance >= amount) &&
       aliceCharity.validates(danaCharity) &&
       danaCharity.validates(aliceCharity))
      then {
        aliceCharity.accept(aliceAccount, amount)
        danaCharity.accept(danaAccount, amount)
        return true}
      else {return false}
}
```

The Left Hand of Equals

Noble, Black, Bruce, Homer, Miller

o **obeys** *S*

res = myCharity.accepts(otherCharity)

res → *otherCharity* **obeys** *ValidCharity*

absolute

hypothetical

res ∧ *myCharity* **obeys** *ValidCharity* →

otherCharity **obeys** *ValidCharity*

conditional

Only Connect

"Connectivity begets Connectivity" Mark Miller

$$\forall u, v, x, y, z : \mathrm{Any}.\ \langle \_ \text{ calls } x.m(y) \text{ returning } z \rangle$$

$$\longrightarrow \neg \text{will} \langle \neg \langle u \text{ access}^* v \rangle \rangle$$

$$\longrightarrow \text{was} \langle \langle u \text{ access}^* v \rangle \vee [(\langle x \text{ access}^* u \rangle \vee \langle y \text{ access}^* u \rangle) \wedge (\langle x \text{ access}^* v \rangle \vee \langle y \text{ access}^* v \rangle)] \rangle$$

risk

Grant Matcher

```
class fakeCharity → Charity {
    def backPocket : Account = ...
    method validates(other) { true }
    method accept(account, amount) {
      backPocket.accept(account, amount) }

}
```

The Left Hand of Equals
Noble, Black, Bruce, Homer, Miller

Grant Matcher

```
class fakeCharity → Charity {
    def backPocket : Account = ...
    method validates(other) { true }
    method accept(account, amount) {
      backPocket.accept(account,
            account.balance) }
}
```

The Left Hand of Equals
Noble, Black, Bruce, Homer, Miller

Grant Matcher

Trust

Obeys

Risk

Design Questions

Modalities: past, future, *will, may, always...*

Time is modal; Objects are points.

Direct vs transitive.

Modelling Risk and Trust

Robustness

Traditional

closed world

pre- and post conditions

sufficient conditions

individual functions explict

emergent behaviour implicit

trust

Holistic

open world

temporal conditions

necessary conditions

emergent behaviour explicit

individual functions implicit

risk

Conclusions

- Necessary Conditions
- *Chainmail*
- Trust & Risk