AARHUS
UNIVERSITY

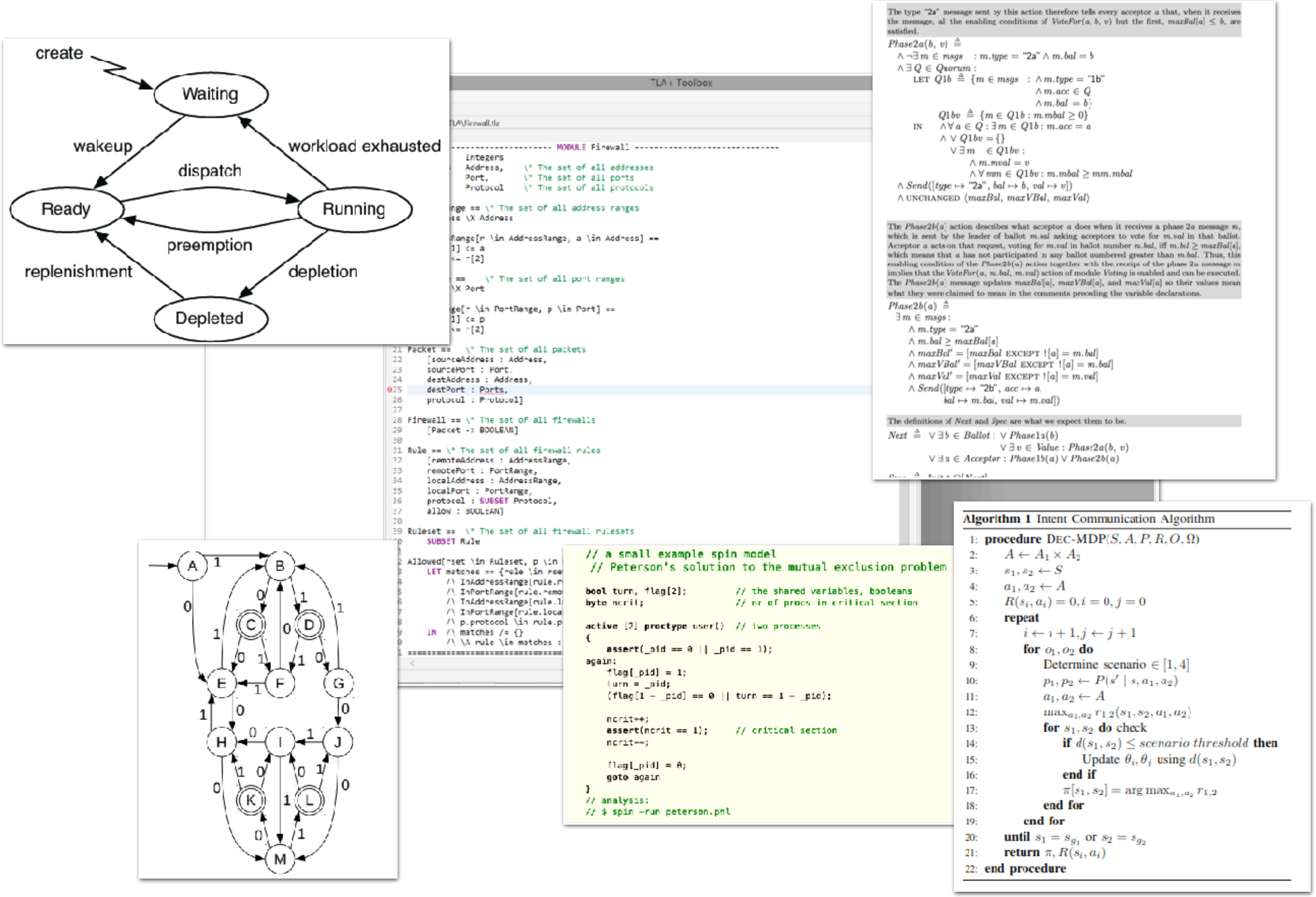
CENTER FOR BASIC RESEARCH IN
PROGRAM VERIFICATION

Higher-Order Separation Logic for Distributed Systems and Security

Simon Oddershede Gregersen

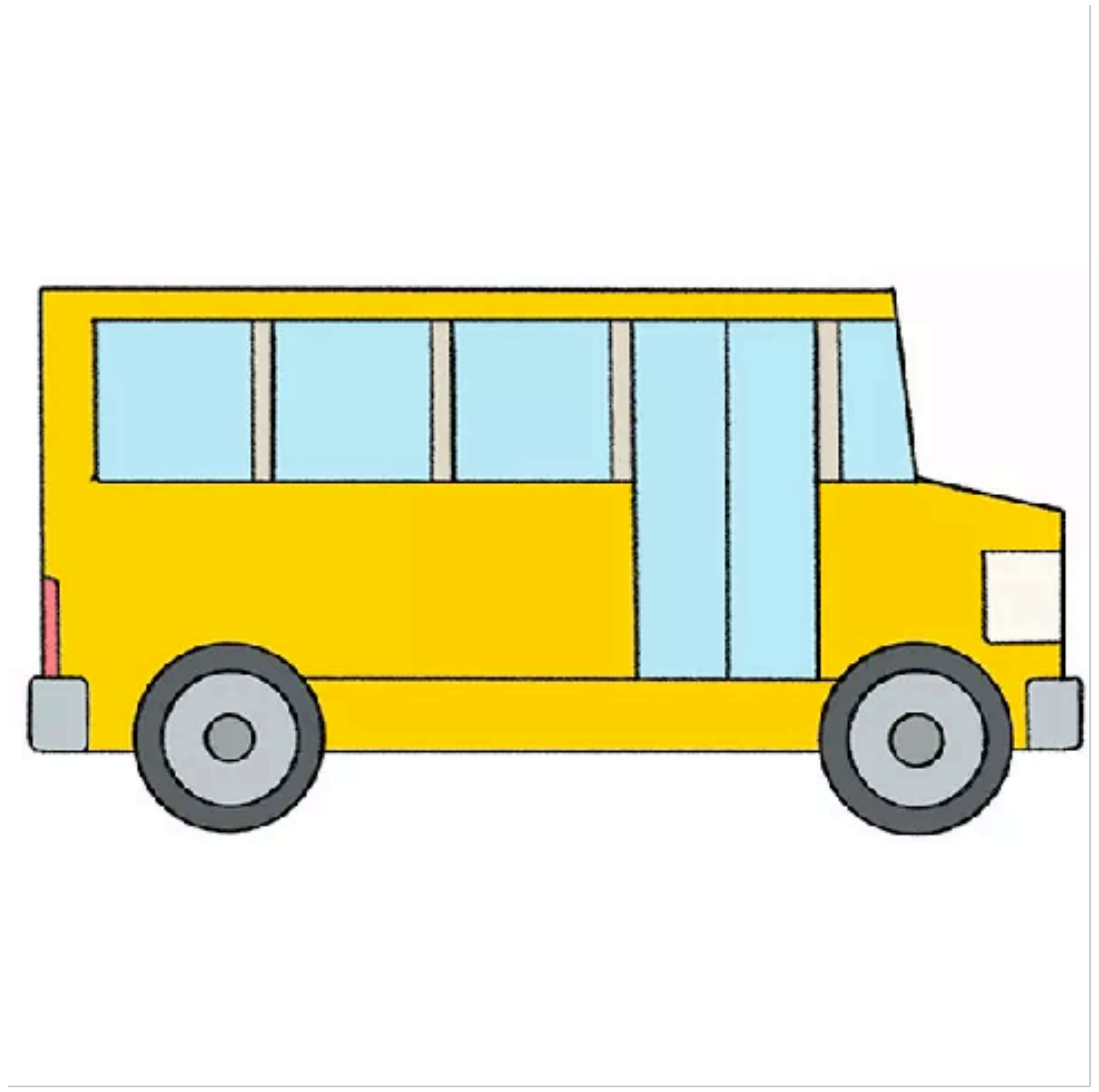
PhD dissertation defence
17 March, 2023

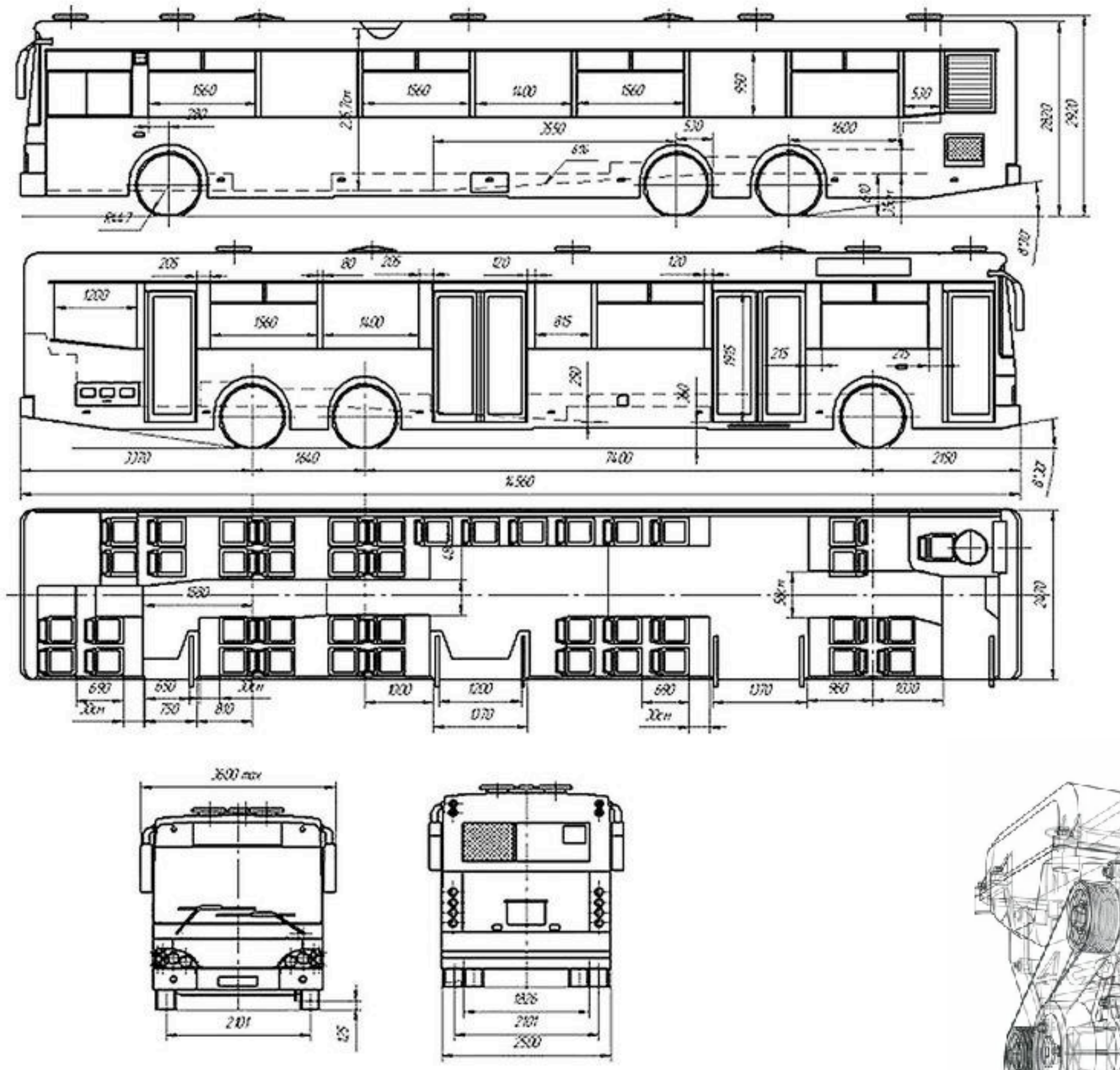
Programs.

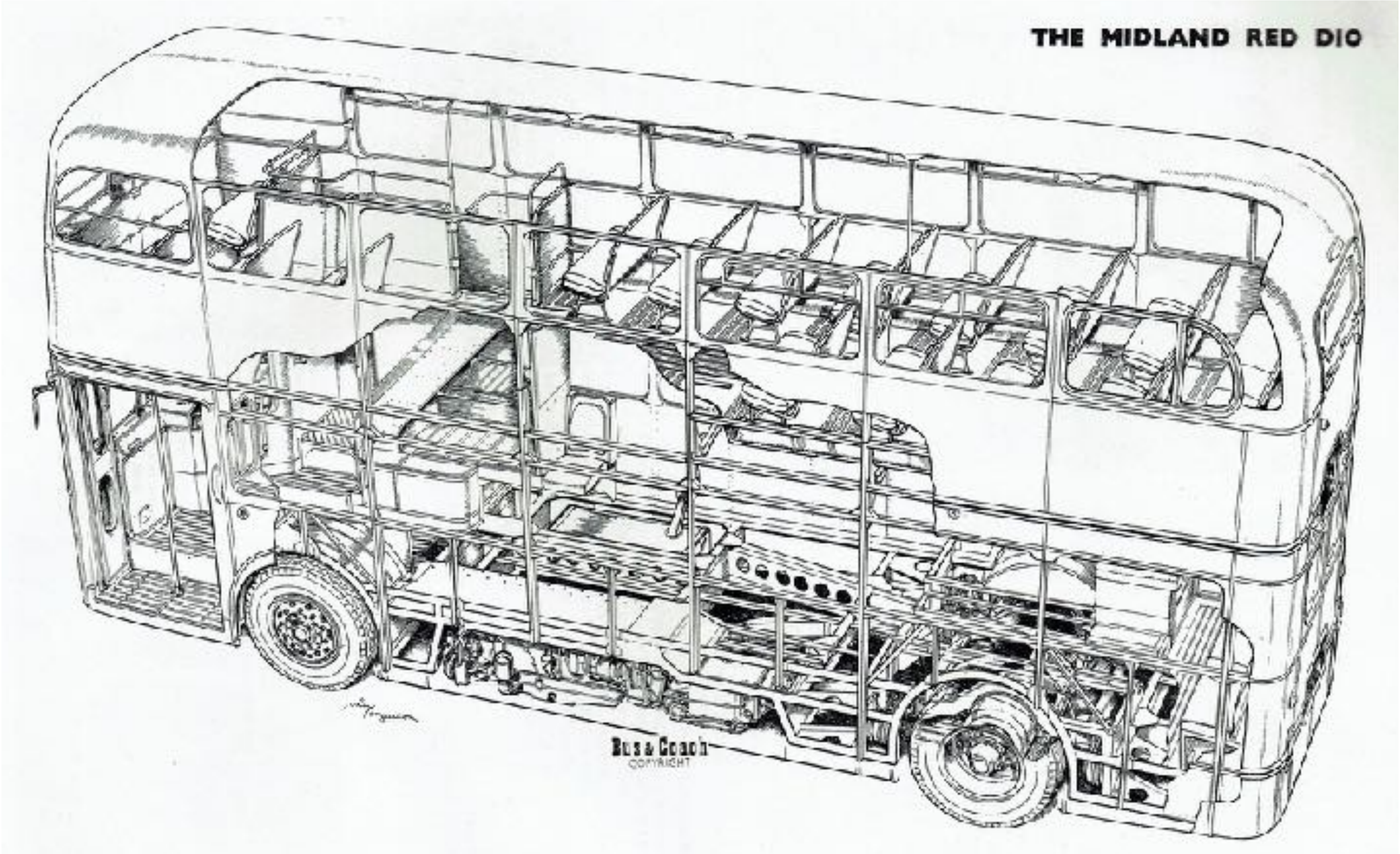

create
Waiting
wakeup
workload exhausted
dispatch
Ready
Running
preemption
replenishment
depletion
Depleted
TLA+ Toolbox
MODULE Firewall
A
B
C
D
E
F
G
H
I
J
K
L
M
// a small example spin model
// Peterson's solution to the mutual exclusion problem
bool turn, flag[2]; // the shared variables, booleans
byte ncrit; // nr of procs in critical section
active [2] proctype user() // two processes
{
assert(_pid == 0 || _pid == 1);
again:
flag[_pid] = 1;
turn = _pid;
(flag[1 - _pid] == 0 || turn == 1 - _pid);
ncrit++;
assert(ncrit == 1); // critical section
ncrit--;
flag[_pid] = 0;
goto again
}
// analysis:
// $ spin -run peterson.pml
Algorithm 1 Intent Communication Algorithm
procedure DEC-MDP(S, A, P, R, O, Ω)
A ← A1 × A2
s1, s2 ← S
a1, a2 ← A
R(si, ai) = 0, i = 0, j = 0
repeat
i ← i + 1, j ← j + 1
for o1, o2 do
Determine scenario ∈ [1, 4]
p1, p2 ← P(s′ | s, a1, a2)
a1, a2 ← A
max r1 2(s1, s2, a1, a2)
for s1, s2 do check
if d(s1, s2) ≤ scenario threshold then
Update θi, θj using d(s1, s2)
end if
π[s1, s2] = arg max r1,2
end for
end for
until s1 = sg1 or s2 = sg2
return π, R(si, ai)
end procedure

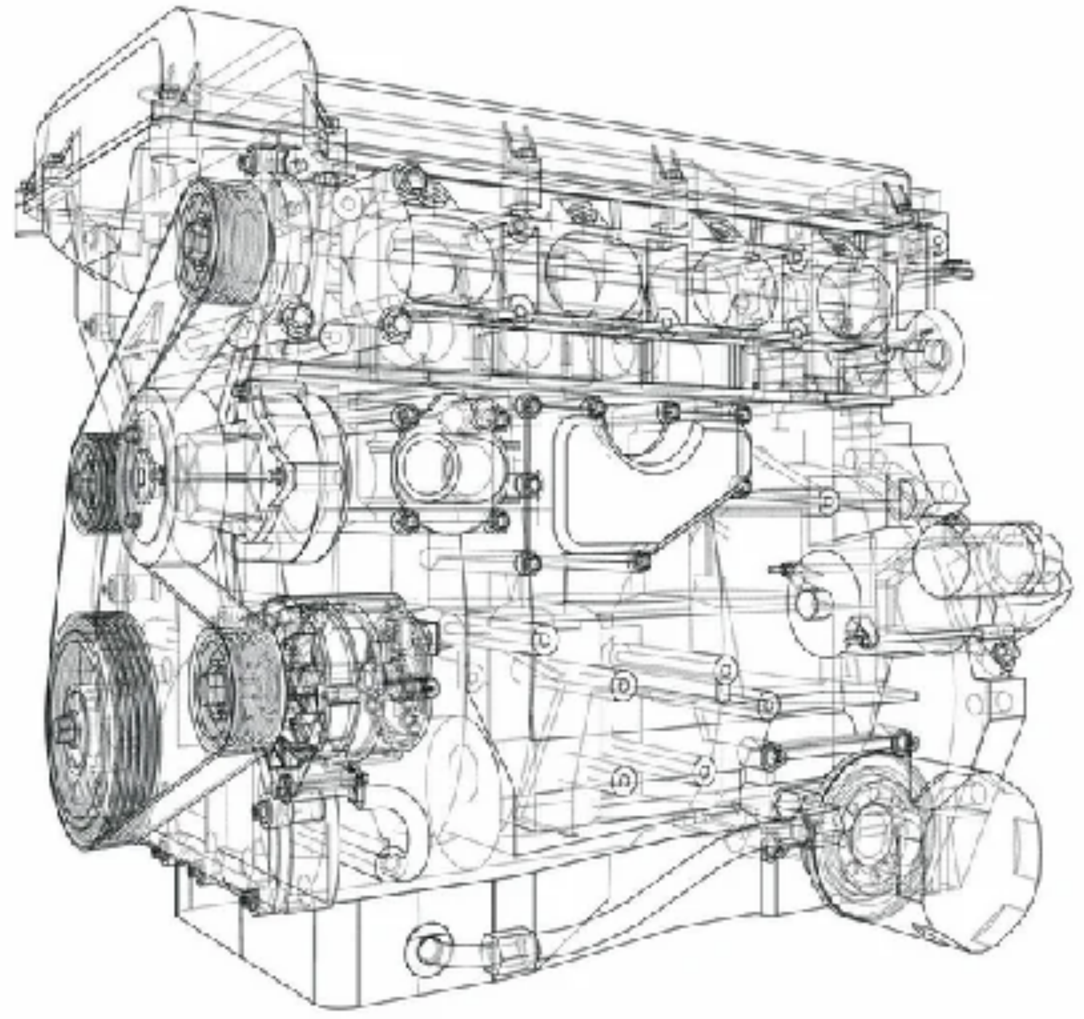
“A computer program is a sequence or set of instructions in a programming language for a computer to execute.”

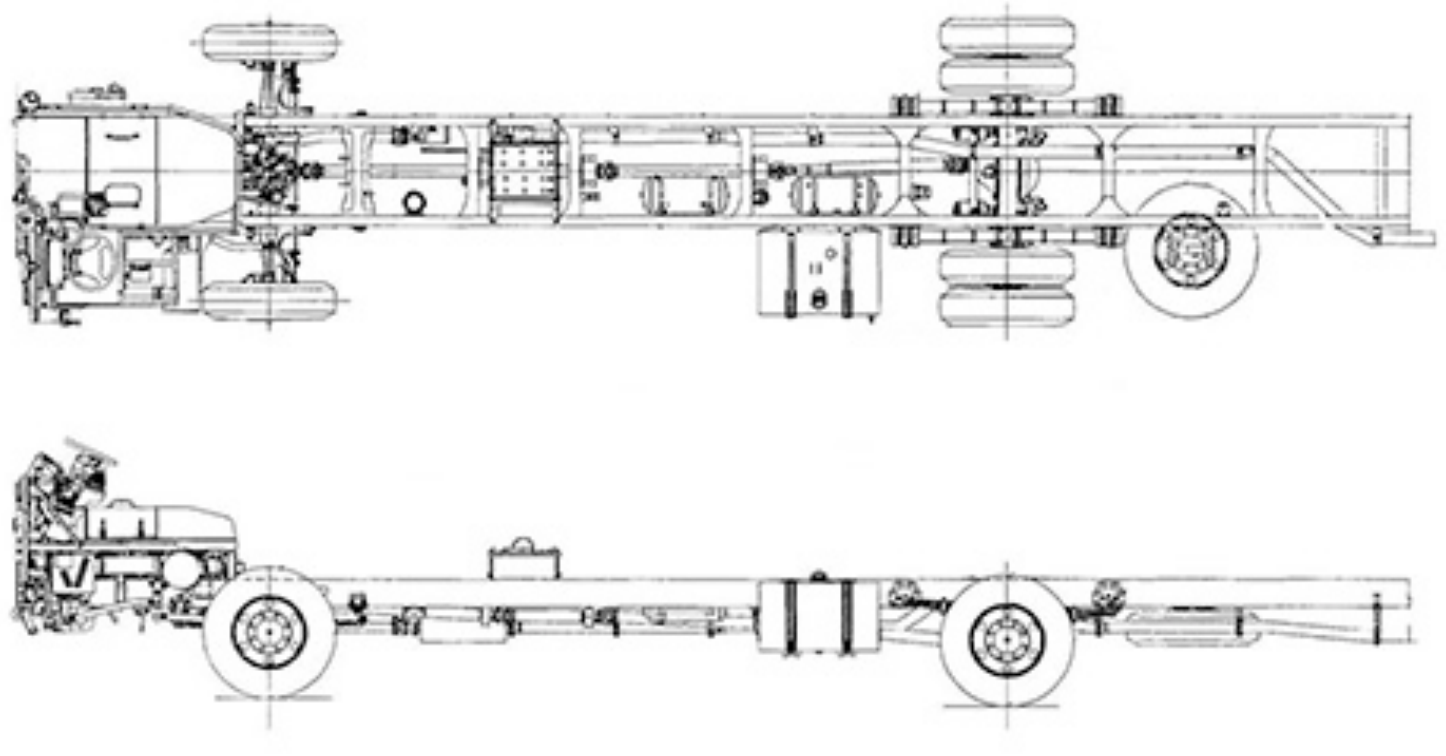

Snedsted Turistbusser
DANMARK
Snedsted
Turistbusser
Knud E. Gregersen · 97 93 40 57
VOLVO
XK 91 370
9700



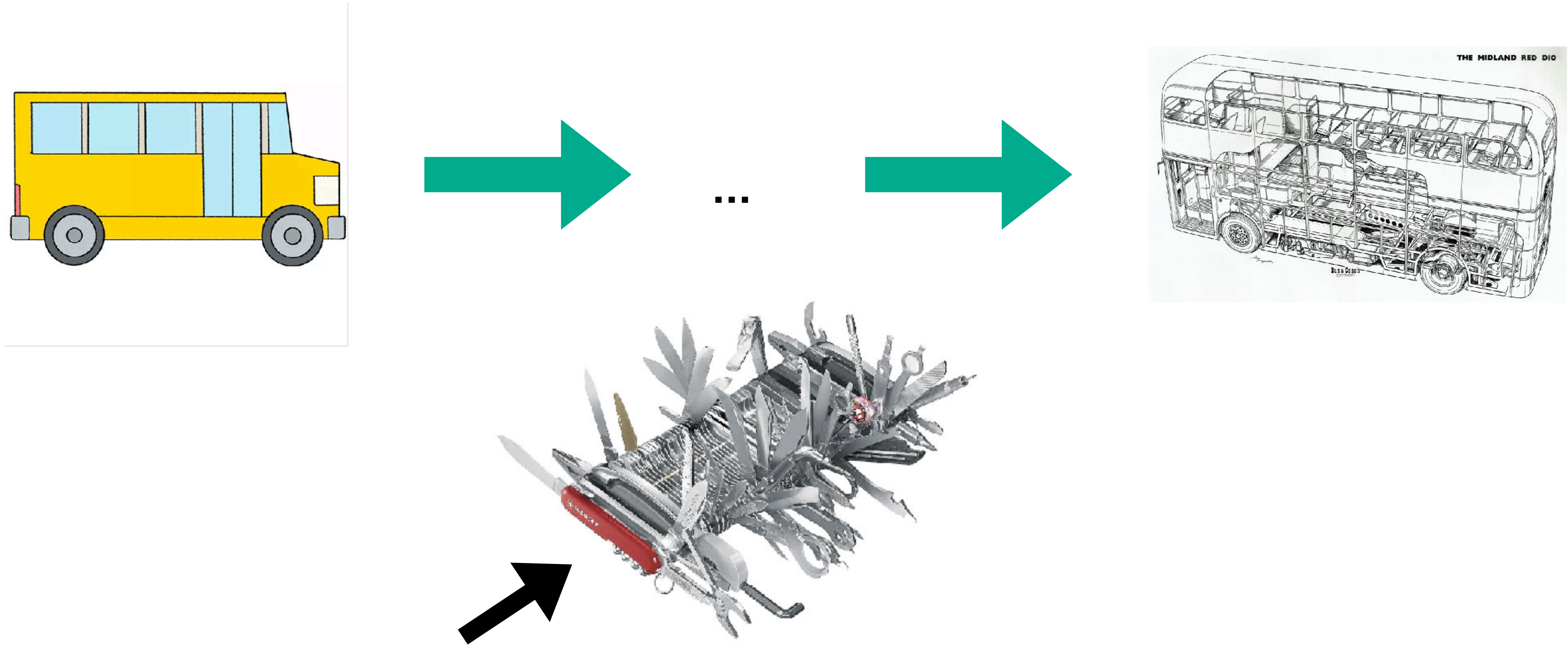



THE MIDLAND RED D10
Bus & Coach
COPYRIGHT





In a nutshell



This dissertation

… for computer programs, not busses!

This dissertation

Features

- Distribution
- Information-flow control types
- Randomization

Properties

- Safety
- Simulation
- (Liveness)
- Noninterference
- Contextual equivalence

This dissertation

Aneris: A Mechanized Logic for Modular Distribution **st**r Safety @ ESOP '20
Morten Krogh-Jespersen, Amin Timany, M... , S... *regersen*, Lars Birkedal

Distributed Causal Memory: Modular Sp Distribution **ca** Safety **Distributed Separation Logic** @ POPL '21
Léon Gondelman, *Simon Oddershede Gre* ... Am ... dal

Mechanized Logical Relations for Termin Security types **nir** Noninterference @ POPL '21
Simon Oddershede Gregersen, Johan Bay, ... irk

Trillium: History-Sensitive Refinement in Distribution Safety Simulation Liveness
Amin Timany, *Simon Oddershede Gregers* ... Léo ... Nie

Asynchronous Probabilistic Couplings in Randomization **ati** Ctx. equiv. [Manuscript]
Simon Oddershede Gregersen, Alejandro ... ase ... rotti, Lars Birkedal

Thesis statement:

Higher-order separation logic is all you need!

This dissertation

Aneris: A Mechanized Logic for Modular Reasoning about Distributed Systems @ ESOP '20
Morten Krogh-Jespersen, Amin Timany, Marit Edna Ohlenbusch, *Simon Oddershede Gregersen*, Lars Birkedal

Distributed Causal Memory: Modular Specification and Verification in Higher-Order Distributed Separation Logic @ POPL '21
Léon Gondelman, *Simon Oddershede Gregersen*, Abel Nieto, Amin Timany, Lars Birkedal

Mechanized Logical Relations for Termination-Insensitive Noninterference @ POPL '21
Simon Oddershede Gregersen, Johan Bay, Amin Timany, Lars Birkedal

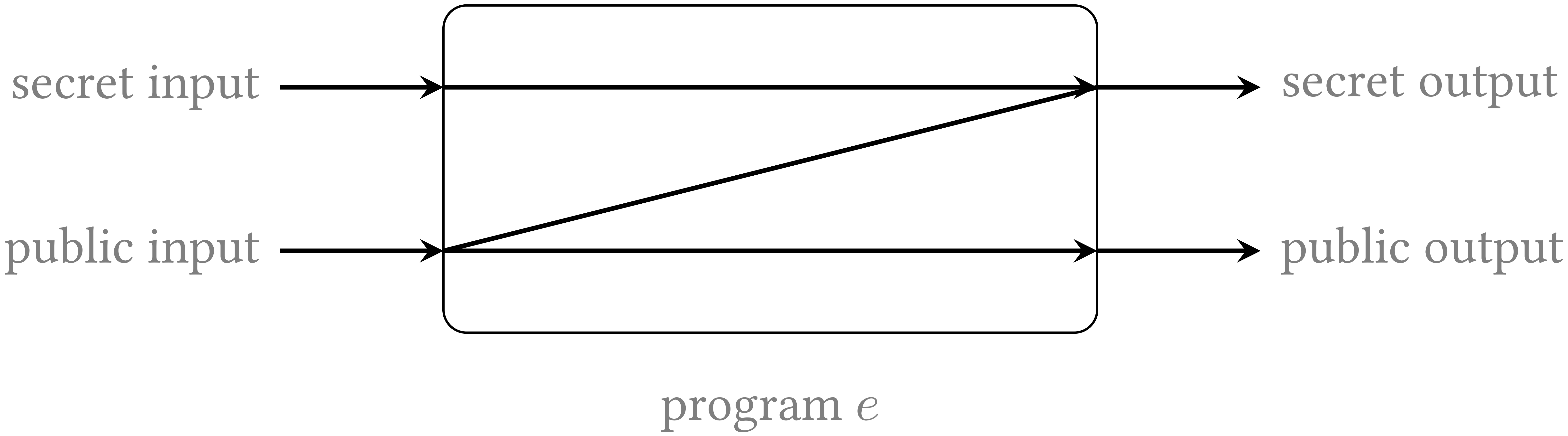
Trillium: History-Sensitive Refinement in Separation Logic [Manuscript]
Amin Timany, *Simon Oddershede Gregersen*, Léo Stefanesco, Léon Gondelman, Abel Nieto, Lars Birkedal

Asynchronous Probabilistic Couplings in Higher-Order Separation Logic [Manuscript]
Simon Oddershede Gregersen, Alejandro Aguirre, Philipp G. Haselwarter, Joseph Tassarotti, Lars Birkedal

Mechanized Logical Relations for Termination-Insensitive Noninterference

joint work with Johan Bay, Amin Timany, and Lars Birkedal

The prevailing basic semantic notion of secure information flow is noninterference.



Program e satisfies termination-insensitive noninterference, abbrv. $\mathsf{TINI}(e)$, when

$$e[v_1/x] \Downarrow o_1 \quad \text{and} \quad e[v_2/x] \Downarrow o_2 \quad \text{implies} \quad o_1 \simeq o_2$$

for all secrets v_1 and v_2.

The problem

Information-flow control enforcement often comes as a static type system:

$$\Gamma \vdash e : t^{\ell} \qquad \text{implies} \qquad \mathsf{TINI}(e)$$

To really be useful, it must support the same features as modern languages:

- higher types
- reference types
- higher-order state
- …

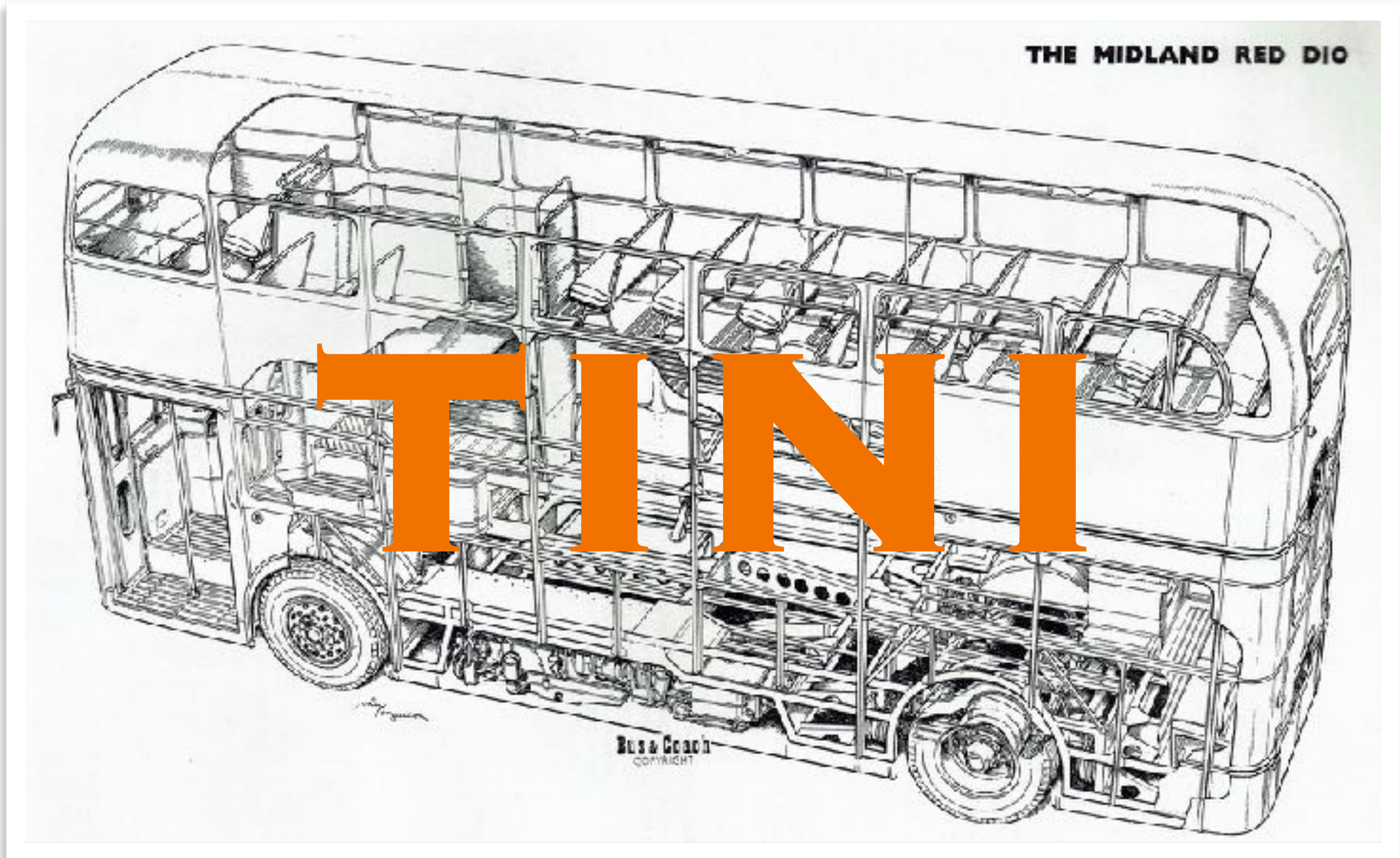
The difficulty of proving the system sound increases, however.

This work

higher-order state

impredicative polymorphism

label polymorphism



existential types

IFC types

recursive types

- shows that such a rich type system satisfies TINI
- with full mechanization of all results in Coq
- using a semantic model

Compositional integration of syntactically well-typed and ill-typed components:

$$\Gamma, x : \tau_2 \vdash e_1 : \tau_1 \quad \text{and} \quad e_2 \in [\![\tau_2]\!] \quad \text{implies} \quad \mathsf{TINI}(e_1[e_2/x])$$

Types

$$
\begin{aligned}
\tau &::= t^{\ell} \\
t &::= \mathbb{B} \mid \mathbb{N} \mid \tau \times \tau \mid \tau + \tau \mid \\
&\qquad \tau \xrightarrow{\ell} \tau \mid \mathsf{ref}(\tau) \mid \alpha \mid \forall_{\ell}\, \alpha.\, \tau \mid \forall\!\!\!/_{\ell}\, \kappa.\, \tau \mid \exists \alpha.\, \tau \mid \mu\, \alpha.\, \tau \\
\ell &::= \kappa \mid l \in \mathcal{L} \mid \ell \sqcup \ell
\end{aligned}
$$

For this presentation we consider $\mathcal{L} = \{\bot, \top\}$ where $\bot \sqsubseteq \top, \top \not\sqsubseteq \bot$

Consider `if` $secret$ `then` $f\ ()$ — if f has public side effects, $secret$ is leaked

Typing judgment

Term-level context

Type-level context

$$\Xi \mid \Psi \mid \Gamma \vdash_{pc} e : \tau$$

Label context

“Program counter” label

Type system

T-IF

$$\frac{\Xi \mid \Psi \mid \Gamma \vdash_{pc} e : \mathbb{B}^{\ell} \qquad \forall i \in \{1, 2\} \,.\, \Xi \mid \Psi \mid \Gamma \vdash_{pc \sqcup \ell} e_i : \tau \qquad \Psi \vdash \tau \searrow \ell}{\Xi \mid \Psi \mid \Gamma \vdash_{pc} \texttt{if}\ e\ \texttt{then}\ e_1\ \texttt{else}\ e_2 : \tau}$$

T-STORE

$$\frac{\Xi \mid \Psi \mid \Gamma \vdash_{pc} e_1 : \mathsf{ref}(\tau)^{\ell} \qquad \Xi \mid \Psi \mid \Gamma \vdash_{pc} e_2 : \tau \qquad \Psi \vdash \tau \searrow pc \sqcup \ell}{\Xi \mid \Psi \mid \Gamma \vdash_{pc} e_1 \mathrel{\texttt{:=}} e_2 : 1^{\perp}}$$

Theorem (Termination-Insensitive Noninterference)

If

$$x : \mathbb{B}^{\top} \vdash_{\perp} e : \mathbb{B}^{\perp}, \qquad \vdash_{\perp} v_1 : \mathbb{B}^{\top}, \qquad \text{and} \qquad \vdash_{\perp} v_2 : \mathbb{B}^{\top}$$

then

$$(\emptyset, e[v_1/x]) \to^* (h_1, v_1') \quad \text{and} \quad (\emptyset, e[v_2/x]) \to^* (h_2, v_2') \quad \text{implies} \quad v_1' = v_2'.$$

Our approach

We set up a binary logical relation

$$\Xi \mid \Psi \mid \Gamma \vDash e_1 \approx e_2 : \tau$$

such that

$$\begin{array}{rcl} \Xi \mid \Psi \mid \Gamma \vdash_{pc} e : \tau & \Rightarrow & \Xi \mid \Psi \mid \Gamma \vDash e \approx e : \tau \\ \Xi \mid \Psi \mid \Gamma \vDash e \approx e : \tau & \Rightarrow & \text{TINI}(e) \end{array}$$

However, this requires manipulating and defining a complex semantic model.

Our approach cont'd

We combat the complexity by defining the relation in Iris:

- Convenient logical connectives for expressing the relation,
- High-level logic to reason within, and
- Coq formalization and the Iris Proof Mode to mechanize our proofs

Not a novel approach, but some novel challenges!

Value relation

$$\llbracket \alpha \rrbracket^{\rho}_{\Theta} \triangleq \pi_1\left(\Theta(\alpha)\right)$$

$$\llbracket 1 \rrbracket^{\rho}_{\Theta}(v, v') \triangleq v = v' = ()$$

$$\llbracket \mathbb{B} \rrbracket^{\rho}_{\Theta}(v, v') \triangleq v = v' \in \{\texttt{true}, \texttt{false}\}$$

$$\llbracket \mathbb{N} \rrbracket^{\rho}_{\Theta}(v, v') \triangleq v = v' \in \mathbb{N}$$

$$\llbracket \tau_1 \times \tau_2 \rrbracket^{\rho}_{\Theta}(v, v') \triangleq \exists v_1, v_2, v_1', v_2'.\, v = (v_1, v_2) * v' = (v_1', v_2') * \llbracket \tau_1 \rrbracket^{\rho}_{\Theta}(v_1, v_1') * \llbracket \tau_2 \rrbracket^{\rho}_{\Theta}(v_2, v_2')$$

$$\llbracket \tau_1 + \tau_2 \rrbracket^{\rho}_{\Theta}(v, v') \triangleq \bigvee_{i \in \{1,2\}} \exists w, w'.\, v = \texttt{inj}_i\, w * v' = \texttt{inj}_i\, w' * \llbracket \tau_i \rrbracket^{\rho}_{\Theta}(w, w')$$

$$\begin{aligned}
\llbracket \tau_1 \xrightarrow{\ell_e} \tau_2 \rrbracket^{\rho}_{\Theta}(v, v') \triangleq\ & \Box \left(\forall w, w'.\, \llbracket \tau_1 \rrbracket^{\rho}_{\Theta}(w, w') \mathrel{-\!\!*} \mathcal{E}\llbracket \tau_2 \rrbracket^{\rho}_{\Theta}(v\ w, v'\ w')\right) * \\
& \llbracket \tau_1 \xrightarrow{\ell_e} \tau_2 \rrbracket^{\rho}_{\Theta_L}(v) * \llbracket \tau_1 \xrightarrow{\ell_e} \tau_2 \rrbracket^{\rho}_{\Theta_R}(v')
\end{aligned}$$

$$\begin{aligned}
\llbracket \forall_{\ell_e}\, \alpha.\, \tau \rrbracket^{\rho}_{\Theta}(v, v') \triangleq\ & \Box \big(\forall \mathit{\Phi} : \mathit{Rel}.\, \forall \mathit{\Phi}_L, \mathit{\Phi}_R : \mathit{Pred}. \\
& \quad \Box \left(\forall v, v'.\, \mathit{\Phi}(v, v') \mathrel{-\!\!*} \mathit{\Phi}_L(v) * \mathit{\Phi}_R(v')\right) \mathrel{-\!\!*} \mathcal{E}\llbracket \tau \rrbracket^{\rho}_{\Theta, \alpha \mapsto (\mathit{\Phi}, \mathit{\Phi}_L, \mathit{\Phi}_R)}(v\ _, v'\ _)\big) * \\
& \llbracket \forall_{\ell_e}\, \alpha.\, \tau \rrbracket^{\rho}_{\Theta_L}(v) * \llbracket \forall_{\ell_e}\, \alpha.\, \tau \rrbracket^{\rho}_{\Theta_R}(v')
\end{aligned}$$

$$\llbracket \forall\!\!\!/_{\ell_e}\, \kappa.\, \tau \rrbracket^{\rho}_{\Theta}(v, v') \triangleq \Box \left(\forall l \in \mathcal{L}.\, \mathcal{E}\llbracket \tau \rrbracket^{\rho, \kappa \mapsto l}_{\Theta}(v\ \_, v'\ \_)\right) * \llbracket \forall\!\!\!/_{\ell_e}\, \kappa.\, \tau \rrbracket^{\rho}_{\Theta_L}(v) * \llbracket \forall\!\!\!/_{\ell_e}\, \kappa.\, \tau \rrbracket^{\rho}_{\Theta_R}(v')$$

$$\begin{aligned}
\llbracket \exists \alpha.\, \tau \rrbracket^{\rho}_{\Theta}(v, v') \triangleq\ & \Box \big(\exists \mathit{\Phi} : \mathit{Rel}.\, \exists \mathit{\Phi}_L, \mathit{\Phi}_R : \mathit{Pred}. \\
& \quad \Box \left(\forall v, v'.\, \mathit{\Phi}(v, v') \mathrel{-\!\!*} \mathit{\Phi}_L(v) * \mathit{\Phi}_R(v')\right) * \\
& \quad \exists w, w'.\, v = \texttt{pack}\, w * v' = \texttt{pack}\, w' * \llbracket \tau \rrbracket^{\rho}_{\Theta, \alpha \mapsto (\mathit{\Phi}, \mathit{\Phi}_L, \mathit{\Phi}_R)}(w, w')\big)
\end{aligned}$$

$$\begin{aligned}
\llbracket \mu\, \alpha.\, \tau \rrbracket^{\rho}_{\Theta} \triangleq\ & \mu\, \mathit{\Phi} : \mathit{Rel}.\, \lambda(v, v').\exists w, w'.\, v = \texttt{fold}\, w * v' = \texttt{fold}\, w' * \\
& \triangleright \llbracket \tau \rrbracket^{\rho}_{\Theta, \alpha \mapsto (\mathit{\Phi}, \llbracket \mu\, \alpha.\, \tau \rrbracket^{\rho}_{\Theta_L}, \llbracket \mu\, \alpha.\, \tau \rrbracket^{\rho}_{\Theta_R})}(w, w')
\end{aligned}$$

$$\llbracket \mathsf{ref}(\tau) \rrbracket^{\rho}_{\Theta}(v, v') \triangleq \exists \ell, \ell'.\, v = \ell * v' = \ell' * \boxed{\exists w, w'.\, \ell \mapsto_L w * \ell' \mapsto_R w' * \llbracket \tau \rrbracket^{\rho}_{\Theta}(w, w')}^{\mathcal{N}_{root}.(\ell, \ell')}$$

$$\llbracket t^{\ell} \rrbracket^{\rho}_{\Theta}(v, v') \triangleq \begin{cases} \llbracket t \rrbracket^{\rho}_{\Theta}(v, v') & \text{if } \llbracket \ell \rrbracket_{\rho} \sqsubseteq \zeta \\ \llbracket t \rrbracket^{\rho}_{\Theta_L}(v) * \llbracket t \rrbracket^{\rho}_{\Theta_R}(v') & \text{if } \llbracket \ell \rrbracket_{\rho} \not\sqsubseteq \zeta \end{cases}$$

Expression relation

$$\mathcal{E}\llbracket \tau \rrbracket^{\rho}_{\Theta}(e, e') \triangleq \mathsf{mwp}\ e \sim e'\ \{\llbracket \tau \rrbracket^{\rho}_{\Theta}\}$$

Environment relation

$$\mathcal{G}\llbracket \cdot \rrbracket^{\rho}_{\Theta}(\epsilon, \epsilon) \triangleq \mathsf{True}$$

$$\mathcal{G}\llbracket \Gamma, x : \tau \rrbracket^{\rho}_{\Theta}(\vec{v}w, \vec{v'}w') \triangleq \mathcal{G}\llbracket \Gamma \rrbracket^{\rho}_{\Theta}(\vec{v}, \vec{v'}) * \llbracket \tau \rrbracket^{\rho}_{\Theta}(w, w')$$

Semantic typing judgment

$$\mathit{Coh}(\Theta) \triangleq \mathop{\Huge *}_{(\mathit{\Phi}, \mathit{\Phi}_L, \mathit{\Phi}_R) \in \Theta} \Box \left(\forall v, v'.\, \mathit{\Phi}(v, v') \mathrel{-\!\!*} \mathit{\Phi}_L(v) * \mathit{\Phi}_R(v')\right)$$

$$\Xi \mid \Psi \mid \Gamma \vDash e \approx_{\zeta} e' : \tau \triangleq \Box \begin{pmatrix} \forall \Theta, \rho, \vec{v}, \vec{v'}.\, \mathsf{dom}(\Xi) \subseteq \mathsf{dom}(\Theta) * \mathsf{dom}(\Psi) \subseteq \mathsf{dom}(\rho) \mathrel{-\!\!*} \\ \mathit{Coh}(\Theta) * \mathcal{G}\llbracket \Gamma \rrbracket^{\rho}_{\Theta}(\vec{v}, \vec{v'}) \mathrel{-\!\!*} \mathcal{E}\llbracket \tau \rrbracket^{\rho}_{\Theta}(e[\vec{v}/\vec{x}], e'[\vec{v'}/\vec{x}]) \end{pmatrix}$$

Value relation

$$\llbracket \alpha \rrbracket^{\rho}_{\Delta} \triangleq \Delta(\alpha)$$

$$\llbracket 1 \rrbracket^{\rho}_{\Delta}(v) \triangleq v = ()$$

$$\llbracket \mathbb{B} \rrbracket^{\rho}_{\Delta}(v) \triangleq v \in \{\texttt{true}, \texttt{false}\}$$

$$\llbracket \mathbb{N} \rrbracket^{\rho}_{\Delta}(v) \triangleq v \in \mathbb{N}$$

$$\llbracket \tau_1 \times \tau_2 \rrbracket^{\rho}_{\Delta}(v) \triangleq \exists v_1, v_2.\, v = (v_1, v_2) * \llbracket \tau_1 \rrbracket^{\rho}_{\Delta}(v_1) * \llbracket \tau_2 \rrbracket^{\rho}_{\Delta}(v_2)$$

$$\llbracket \tau_1 + \tau_2 \rrbracket^{\rho}_{\Delta}(v) \triangleq \bigvee_{i \in \{1,2\}} \exists w.\, v = \texttt{inj}_i\, w * \llbracket \tau_i \rrbracket^{\rho}_{\Delta}(w)$$

$$\llbracket \tau_1 \xrightarrow{\ell_e} \tau_2 \rrbracket^{\rho}_{\Delta}(v) \triangleq \Box \left(\forall w.\, \llbracket \tau_1 \rrbracket^{\rho}_{\Delta}(w) \mathrel{-\!\!*} \mathcal{E}_{\ell_e}\llbracket \tau_2 \rrbracket^{\rho}_{\Delta}(v\ w)\right)$$

$$\llbracket \forall_{\ell_e}\, \alpha.\, \tau \rrbracket^{\rho}_{\Delta}(v) \triangleq \Box \left(\forall f : \mathit{Pred}.\, \mathcal{E}_{\ell_e}\llbracket \tau \rrbracket^{\rho}_{\Delta, \alpha \mapsto f}(v\ \_)\right)$$

$$\llbracket \forall\!\!\!/_{\ell_e}\, \kappa.\, \tau \rrbracket^{\rho}_{\Delta}(v) \triangleq \Box \left(\forall l \in \mathcal{L}.\, \mathcal{E}_{\ell_e}\llbracket \tau \rrbracket^{\rho, \kappa \mapsto l}_{\Delta}(v\ \_)\right)$$

$$\llbracket \exists \alpha.\, \tau \rrbracket^{\rho}_{\Delta}(v) \triangleq \Box \left(\exists \mathit{\Phi} : \mathit{Pred}.\, \exists w.\, v = \texttt{pack}\, w * \llbracket \tau \rrbracket^{\rho}_{\Delta, \alpha \mapsto \mathit{\Phi}}(w)\right)$$

$$\llbracket \mu\, \alpha.\, \tau \rrbracket^{\rho}_{\Delta} \triangleq \mu\, \mathit{\Phi} : \mathit{Pred}.\, \lambda v.\exists w.\, v = \texttt{fold}\, w * \triangleright \llbracket \tau \rrbracket^{\rho}_{\Delta, \alpha \mapsto f}(w)$$

$$\llbracket \mathsf{ref}(t^{\ell}) \rrbracket^{\rho}_{\Delta}(v) \triangleq \exists \ell, \mathcal{N}.\, v = \ell * \mathcal{R}(\Delta, \rho, \ell, \ell, \mathcal{N})$$

$$\mathcal{R}(\Delta, \rho, \ell, \ell, \mathcal{N}) \triangleq \begin{cases} \begin{aligned} & \Box \forall \mathcal{E}.\, \mathcal{N} \subseteq \mathcal{E} \Rightarrow \\ & \quad \left({}_{\mathcal{E}}{\Rrightarrow}_{\mathcal{E} \setminus \mathcal{N}} \triangleright \begin{pmatrix} \exists w.\, \ell \mapsto_i w * \llbracket \tau \rrbracket^{\rho}_{\Delta}(w) * \\ \left((\triangleright \ell \mapsto_i w * \llbracket \tau \rrbracket^{\rho}_{\Delta}(w)) \mathrel{-\!\!*} {}_{\mathcal{E} \setminus \mathcal{N}}{\Rrightarrow}_{\mathcal{E}} \mathsf{True} \right) \end{pmatrix} \right) \end{aligned} & \text{if } \llbracket \ell \rrbracket_{\rho} \sqsubseteq \zeta \\[2ex] \begin{aligned} & \Box \forall \mathcal{E}.\, \mathcal{N} \subseteq \mathcal{E} \Rightarrow \\ & \quad \left({}_{\mathcal{E}}{\Rrightarrow}_{\mathcal{E} \setminus \mathcal{N}} \triangleright \begin{pmatrix} \exists w.\, \ell \mapsto_i w * \llbracket \tau \rrbracket^{\rho}_{\Delta}(w) * \\ \left((\triangleright \exists w'.\, \ell \mapsto_i w' * \llbracket \tau \rrbracket^{\rho}_{\Delta}(w')) \mathrel{-\!\!*} {}_{\mathcal{E} \setminus \mathcal{N}}{\Rrightarrow}_{\mathcal{E}} \mathsf{True} \right) \end{pmatrix} \right) \end{aligned} & \text{if } \llbracket \ell \rrbracket_{\rho} \not\sqsubseteq \zeta \end{cases}$$

$$\llbracket t^{\ell} \rrbracket^{\rho}_{\Delta}(v) \triangleq \llbracket t \rrbracket^{\rho}_{\Delta}(v)$$

Expression relation

$$\mathcal{E}_{pc}\llbracket \tau \rrbracket^{\rho}_{\Delta}(e) \triangleq \llbracket pc \rrbracket_{\rho} \not\sqsubseteq \zeta \Rightarrow \mathsf{mwp}^{\mathcal{M} \Rrightarrow\triangleright}\ e\ \{\llbracket \tau \rrbracket^{\rho}_{\Delta}\}$$

Environment relation

$$\mathcal{G}\llbracket \cdot \rrbracket^{\rho}_{\Delta}(\epsilon) \triangleq \mathsf{True}$$

$$\mathcal{G}\llbracket \Gamma, x : \tau \rrbracket^{\rho}_{\Delta}(\vec{v}w) \triangleq \mathcal{G}\llbracket \Gamma \rrbracket^{\rho}_{\Delta}(\vec{v}) * \llbracket \tau \rrbracket^{\rho}_{\Delta}(w)$$

Semantic typing judgment

$$\Xi \mid \Psi \mid \Gamma \vDash_{pc} e : \tau \triangleq \Box \begin{pmatrix} \forall \Delta, \rho, \vec{v}.\, \mathsf{dom}(\Xi) \subseteq \mathsf{dom}(\Delta) * \mathsf{dom}(\Psi) \subseteq \mathsf{dom}(\rho) \mathrel{-\!\!*} \\ \mathcal{G}\llbracket \Gamma \rrbracket^{\rho}_{\Delta}(\vec{v}) \mathrel{-\!\!*} \mathcal{E}_{pc}\llbracket \tau \rrbracket^{\rho}_{\Delta}(e[\vec{v}/\vec{x}]) \end{pmatrix}$$

Challenge #1

Existing encodings of "logical" logical relations are termination sensitive:

$$e_1 \to^* v_1 \quad \Rightarrow \quad e_2 \to^* v_2 \quad \wedge \quad v_1 \approx v_2.$$

However, we need a terminiation *insensitive* notion:

$$e_1 \to^* v_1 \quad \wedge \quad e_2 \to^* v_2 \quad \Rightarrow \quad v_1 \approx v_2.$$

Solution: a new modal weakest precondition theory mwp $e \sim e' \; \{Q\}$

Challenge #2

$$\textsc{T-If}\quad \frac{\Xi \mid \Psi \mid \Gamma \vdash_{pc} e : \mathbb{B}^{\ell} \qquad \forall i \in \{1,2\} \,.\, \Xi \mid \Psi \mid \Gamma \vdash_{pc \sqcup \ell} e_i : \tau \qquad \Psi \vdash \tau \searrow \ell}{\Xi \mid \Psi \mid \Gamma \vdash_{pc} \texttt{if}\, e\, \texttt{then}\, e_1\, \texttt{else}\, e_2 : \tau}$$

As part of our proofs, we have to show, e.g.,

$$\vDash \quad \texttt{if}\, v\, \texttt{then}\, e_1\, \texttt{else}\, e_2 \quad \approx \quad \texttt{if}\, v'\, \texttt{then}\, e_1\, \texttt{else}\, e_2 \quad : \quad t^{\top}$$

where $\vDash v \approx v' : \mathbb{B}^{\top}$, meaning $v, v' \in \{\texttt{true}, \texttt{false}\}$. This means we have to prove, e.g.,

$$\vDash \quad e_1 \quad \approx \quad e_2 \quad : \quad t^{\top}$$

Luckily, we don't really need to care about return values, only side-effects!

Challenge #2

Solution:

- A **binary relation** for relating terms that are "publicly equivalent"
- A **unary relation** for characterising terms that do not have public side-effects

$$\llbracket t^{\ell} \rrbracket^{\rho}_{\Theta}(v, v') \triangleq \begin{cases} \llbracket t \rrbracket^{\rho}_{\Theta}(v, v') & \text{if } \llbracket \ell \rrbracket_{\rho} = \bot \\ \llbracket t \rrbracket^{\rho}_{\Theta_L}(v) * \llbracket t \rrbracket^{\rho}_{\Theta_R}(v') & \text{othw.} \end{cases}$$

Needs two instantiation of the MWP theory: $\mathsf{mwp}\ e\ \{Q\}$ and $\mathsf{mwp}\ e \sim e'\ \{Q\}$ and a logical way of encoding a "subsumption" property.

Asynchronous Probabilistic Couplings in Higher-Order Separation Logic

joint work with Alejandro Aguirre, Philipp G. Haselwarter, Joseph Tassarotti, and Lars Birkedal

Setting the stage

- Distributed applications often communicate over an untrusted network.
- Randomization is a crucial ingredient in cryptographic protocols.
- Security is often phrased as an indistinguishability of two probabilistic programs.

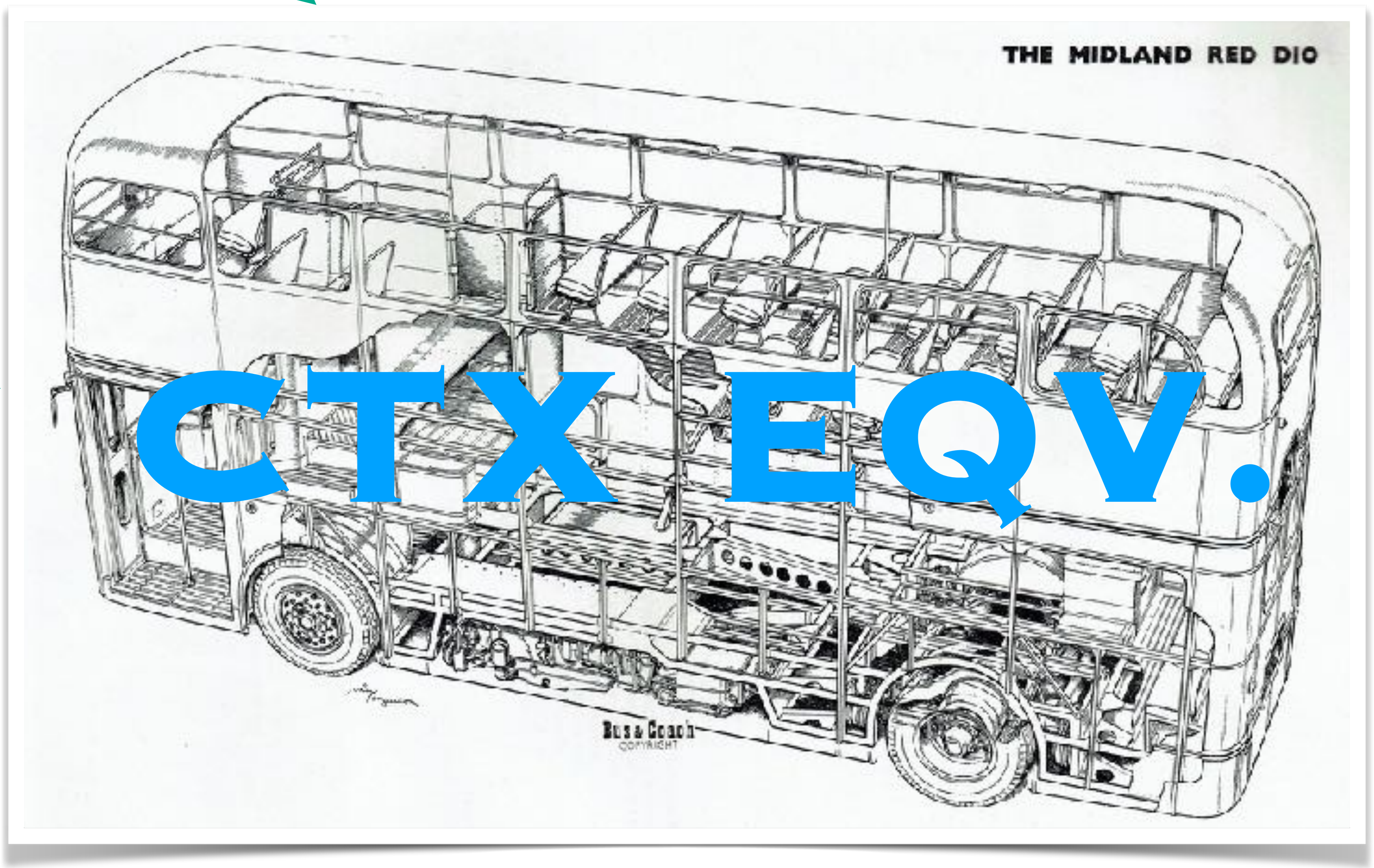
Goal: a relational program logic for an expressive language with coin flips for proving contextual equivalences.

`flip`

higher-order state

probabilistic choice

impredicative polymorphism



existential types

recursive types

Complications

Programs evaluate to distributions over values, not just values.

What do we do about those?

Many probabilistic relational Hoare logics (pRHLs) make use of probabilistic couplings:

$$\mu_1 \sim \mu_2 : R$$

If $R \triangleq (=)$ then $\mu_1 = \mu_2$.

$$\Theta \mid \Gamma \vdash e_1 \simeq_{\text{ctx}} e_2 : \tau \triangleq \forall \tau', \mathcal{C} : (\Theta \mid \Gamma \vdash \tau) \Rightarrow (\emptyset \mid \emptyset \vdash \tau'), \sigma.$$
$$\text{exec}_{\Downarrow}(\mathcal{C}[\,e_1\,], \sigma) = \text{exec}_{\Downarrow}(\mathcal{C}[\,e_2\,], \sigma)$$

Couplings in pRHLs

In pRHLs, couplings manifest as coupling rules:

PRHL-COUPLE

$$\frac{f \text{ bijection}}{\{\mathsf{True}\}\ \texttt{flip} \sim \texttt{flip}\ \{v_1, v_2. \exists b : \mathbb{B}.\ v_1 = b \wedge v_2 = f(b)\}}$$

E.g., for One-Time Pad:

$$\begin{array}{lcl} \texttt{let}\ k = \texttt{flip in} & & \\ & \sim & \texttt{flip} \\ k \otimes m & & \end{array}$$

Pick $f(b) = \text{if } m \text{ then } \neg b \text{ else } b$

Couplings in pRHLs cont'd

However, the approach requires you to synchronize the probabilistic choices.

This is not always possible.

$$\begin{array}{l}\texttt{let}\ b = \texttt{flip in} \\ \lambda\_.\ b\end{array} \quad \overset{?}{\sim} \quad \begin{array}{l}\texttt{let}\ r = \texttt{ref}(\mathrm{None})\ \texttt{in} \\ \lambda\_.\ \texttt{match}\ !\,r\ \texttt{with} \\ \qquad \mathrm{Some}\ (b) \Rightarrow b \\ \qquad \mid \mathrm{None} \Rightarrow \texttt{let}\ b = \texttt{flip in} \\ \qquad\qquad\qquad\quad r \texttt{ := } \mathrm{Some}\ (b); \\ \qquad\qquad\qquad\quad b \\ \quad \texttt{end}\end{array}$$

This work

- A higher-order probabilistic relational separation logic, “Clutch”, for proving contextual equivalence of probabilistic programs with higher-order references, impredicative polymorphism, and recursive types.
- A proof method for asynchronous couplings that allows us to reason about sampling as if it was state.
- Full mechanization of all results in Coq.

Key ideas of Clutch

A (separation logic) refinement judgment

$$\Delta \vDash e_1 \precsim e_2 : \tau$$

"e_1 refines e_2 at type τ"

$$\Theta \mid \Gamma \vdash e_1 \precsim_{\mathrm{ctx}} e_2 : \tau \triangleq \forall \tau', (\mathcal{C} : (\Theta \mid \Gamma \vdash \tau) \Rightarrow (\emptyset \mid \emptyset \vdash \tau')), \sigma. \\ \mathrm{exec}_{\Downarrow}(\mathcal{C}[\, e_1 \,], \sigma) \leq \mathrm{exec}_{\Downarrow}(\mathcal{C}[\, e_2 \,], \sigma)$$

REL-PURE-L

$$\frac{e_1 \overset{\text{pure}}{\rightsquigarrow} e_1' \qquad \Delta \vDash K[\, e_1' \,] \precsim e_2 : \tau}{\Delta \vDash K[\, e_1 \,] \precsim e_2 : \tau}$$

REL-LOAD-L

$$\frac{\ell \mapsto v \qquad \ell \mapsto v \twoheadrightarrow \Delta \vDash K[\, v \,] \precsim e_2 : \tau}{\Delta \vDash K[\, !\ell \,] \precsim e_2 : \tau}$$

REL-STORE-R

$$\frac{\ell \mapsto_{\mathsf{s}} v \qquad \ell \mapsto_{\mathsf{s}} w \twoheadrightarrow \Delta \vDash e_1 \precsim K[\, () \,] : \tau}{\Delta \vDash e_1 \precsim K[\, \ell := w \,] : \tau}$$

REL-COUPLE-FLIPS

$$\frac{f \text{ bijection} \qquad \forall b.\ \Delta \vDash K[\, b \,] \precsim K'[\, f(b) \,] : \tau}{\Delta \vDash K[\, \texttt{flip} \,] \precsim K'[\, \texttt{flip} \,] : \tau}$$

Asynchronous couplings

To support asynchronous couplings, we introduce presampling tapes.

Asynchronous couplings

To support asynchronous couplings, we introduce presampling tapes.

Operationally, we extend the state of program execution with a “heap of tapes” onto which we can presample bits.

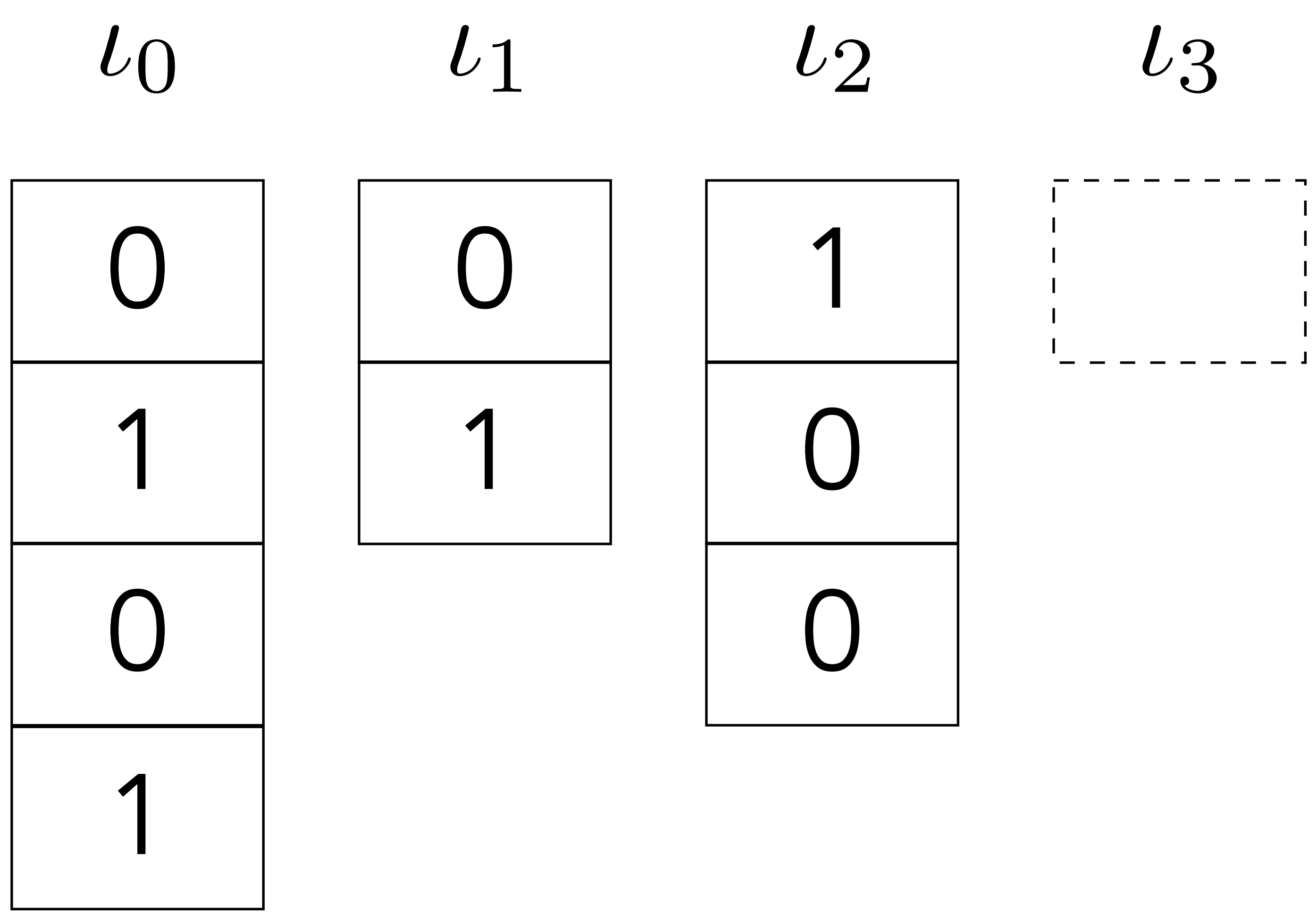
ι_0	ι_1	ι_2
0	0	1
1	1	0
0		0
1		

Asynchronous couplings

To support asynchronous couplings, we introduce presampling tapes.

Operationally, we extend the state of program execution with a "heap of tapes" onto which we can presample bits.

`tape` $\rightarrow$ ι_3

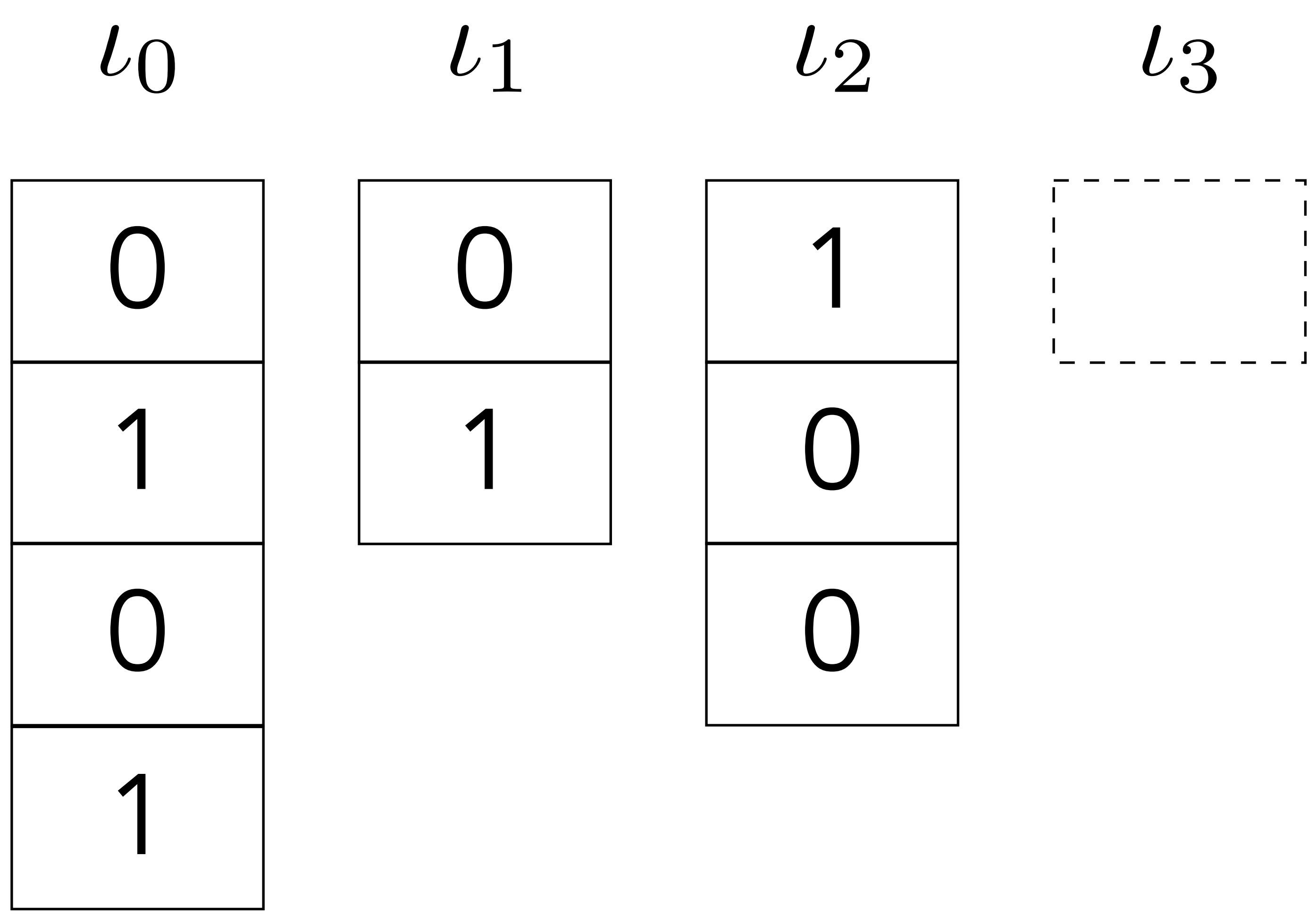


Asynchronous couplings

To support asynchronous couplings, we introduce presampling tapes.

Operationally, we extend the state of program execution with a “heap of tapes” onto which we can presample bits.

$$\texttt{flip}(\iota_3) \quad \to_{\frac{1}{2}} \quad b$$

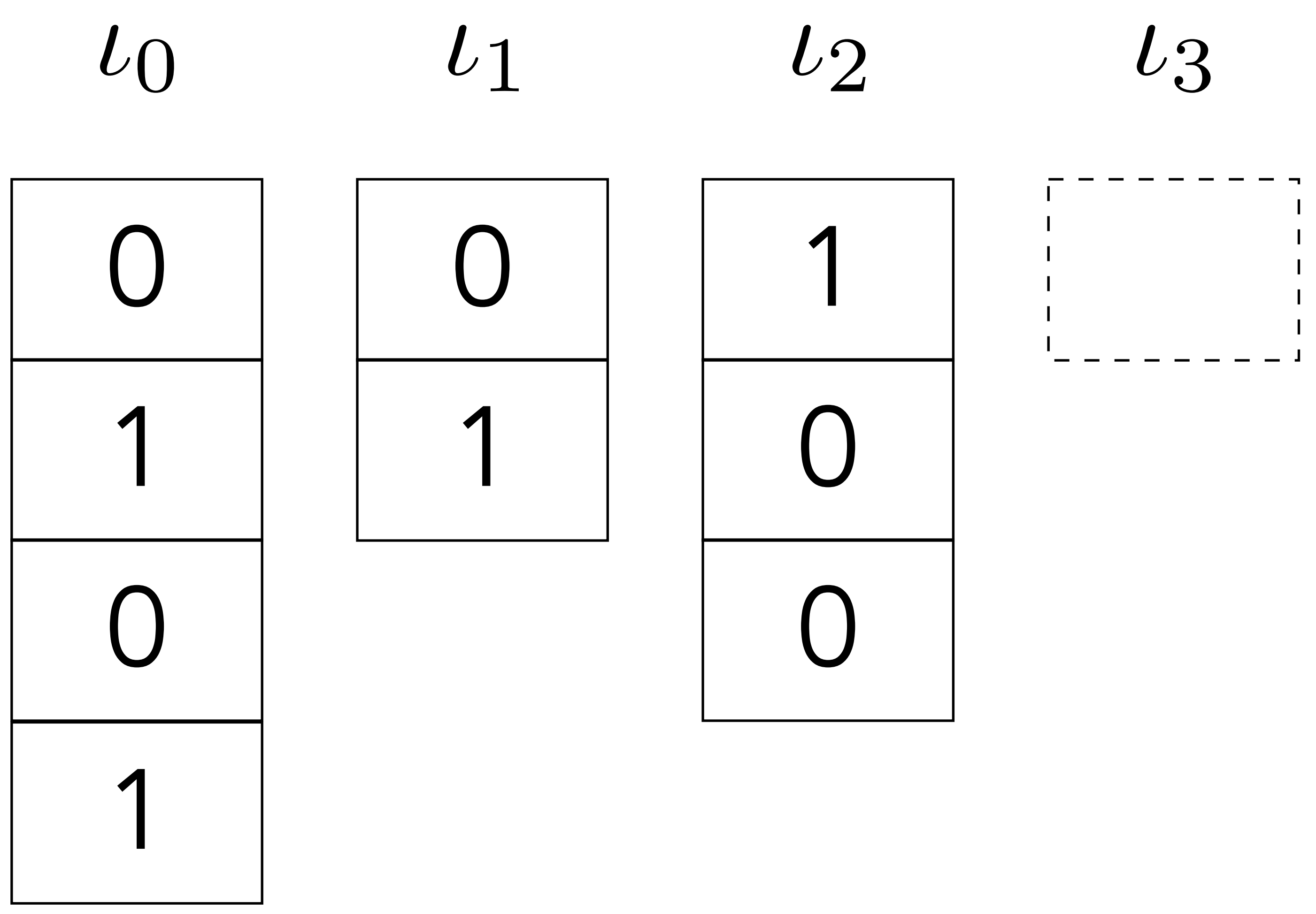


Asynchronous couplings

To support asynchronous couplings, we introduce presampling tapes.

Operationally, we extend the state of program execution with a "heap of tapes" onto which we can presample bits.

$\texttt{flip}(\iota_2)$

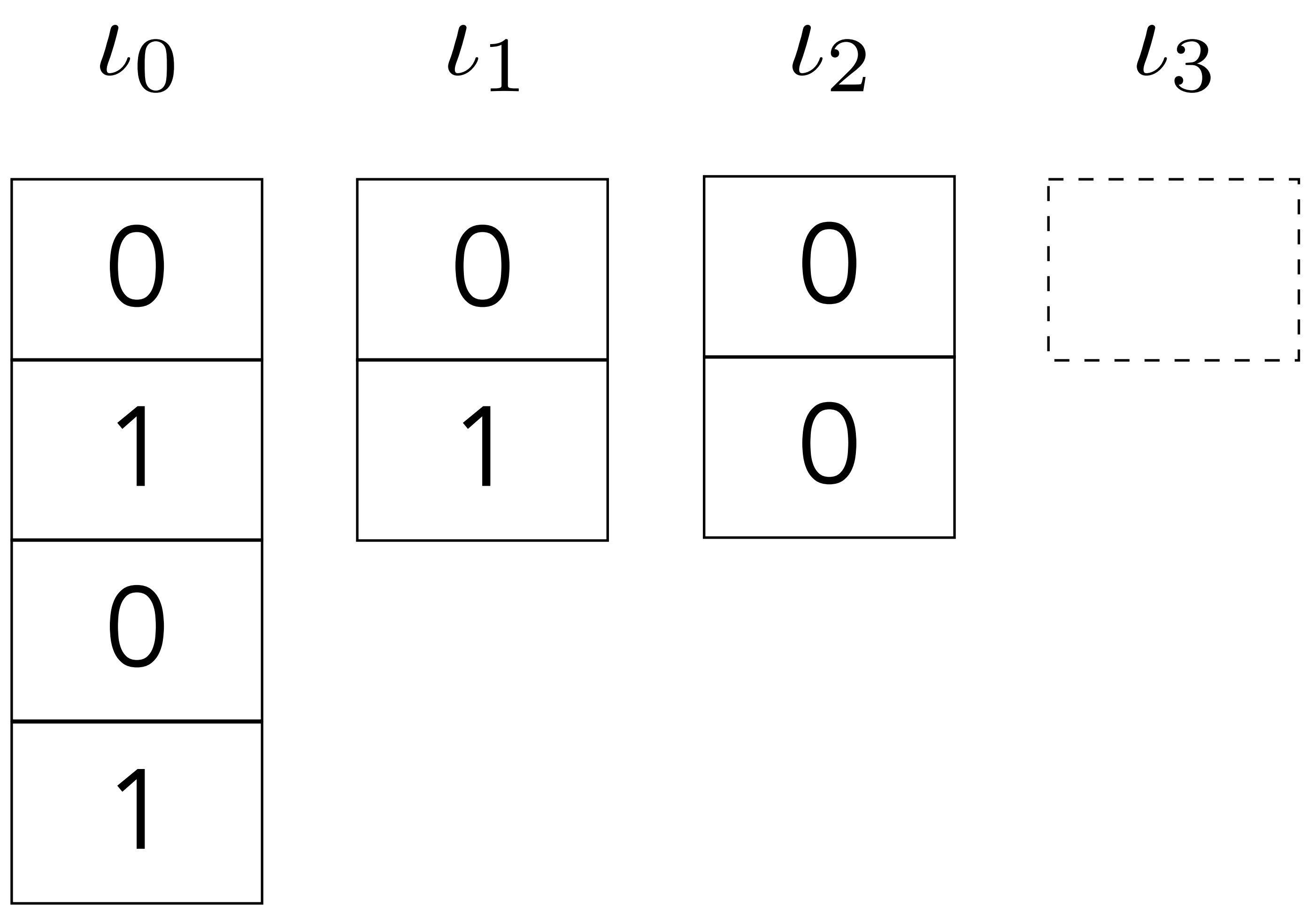


Asynchronous couplings

To support asynchronous couplings, we introduce presampling tapes.

Operationally, we extend the state of program execution with a "heap of tapes" onto which we can presample bits.

$$\texttt{flip}(\iota_2) \quad \to^1 \quad 1$$



Asynchronous couplings

👻

But no language primitives add values to the tapes!

Instead, presampling steps will be ghost operations purely used in the logic.

— in fact, they can be entirely erased!

Asynchronous couplings cont'd

Tapes are "just" state so we introduce a separation logic connective

$$\iota \hookrightarrow \vec{b}$$

that denotes ownership of a tape and its contents.

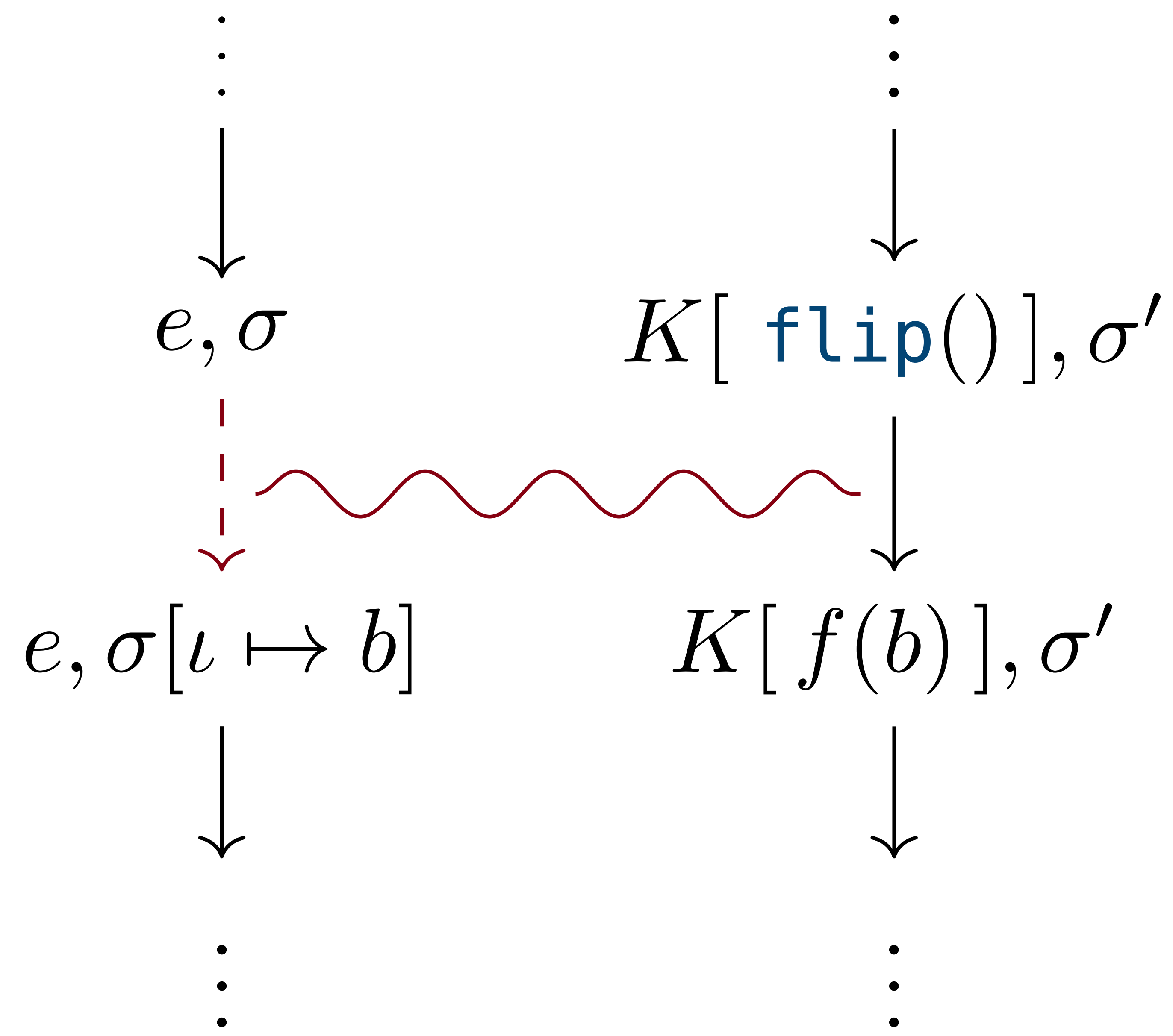
$$\textsc{rel-alloc-tape-l}\quad \frac{\forall \iota.\, \iota \hookrightarrow \epsilon \mathrel{-\!\!*} \Delta \vDash K[\,\iota\,] \precsim e : \tau}{\Delta \vDash K[\,\texttt{tape}\,] \precsim e : \tau}$$

$$\textsc{rel-flip-tape-l}\quad \frac{\iota \hookrightarrow b \cdot \vec{b} \qquad \iota \hookrightarrow \vec{b} \mathrel{-\!\!*} \Delta \vDash K[\,b\,] \precsim e_2 : \tau}{\Delta \vDash K[\,\texttt{flip}(\iota)\,] \precsim e_2 : \tau}$$

Asynchronous couplings cont'd

REL-COUPLE-TAPE-L

$$\frac{f \text{ bijection} \qquad \iota \hookrightarrow \vec{b} \qquad \forall b.\, \iota \hookrightarrow \vec{b} \cdot b \twoheadrightarrow\!\!* \; \Delta \vDash e \precsim K'[\, f(b)\,] : \tau}{\Delta \vDash e \precsim K'[\, \texttt{flip}()\,] : \tau}$$



Motivating example

$$\begin{array}{l}
\texttt{let}\ r = \texttt{ref}(\mathsf{None})\ \texttt{in} \\
\lambda_.\ \texttt{match}\ !r\ \texttt{with} \\
\quad \mathsf{Some}\ (b) \Rightarrow b \\
\quad \mid \mathsf{None} \Rightarrow \texttt{let}\ b = \texttt{flip}\ \texttt{in} \\
\qquad\qquad r := \mathsf{Some}\ (b); \\
\qquad\qquad b \\
\quad \texttt{end}
\end{array}
\quad \precsim_{\text{ctx}} \quad
\begin{array}{l}
\texttt{let}\ b = \texttt{flip}\ \texttt{in} \\
\lambda_.\ b
\end{array}$$

Motivating example

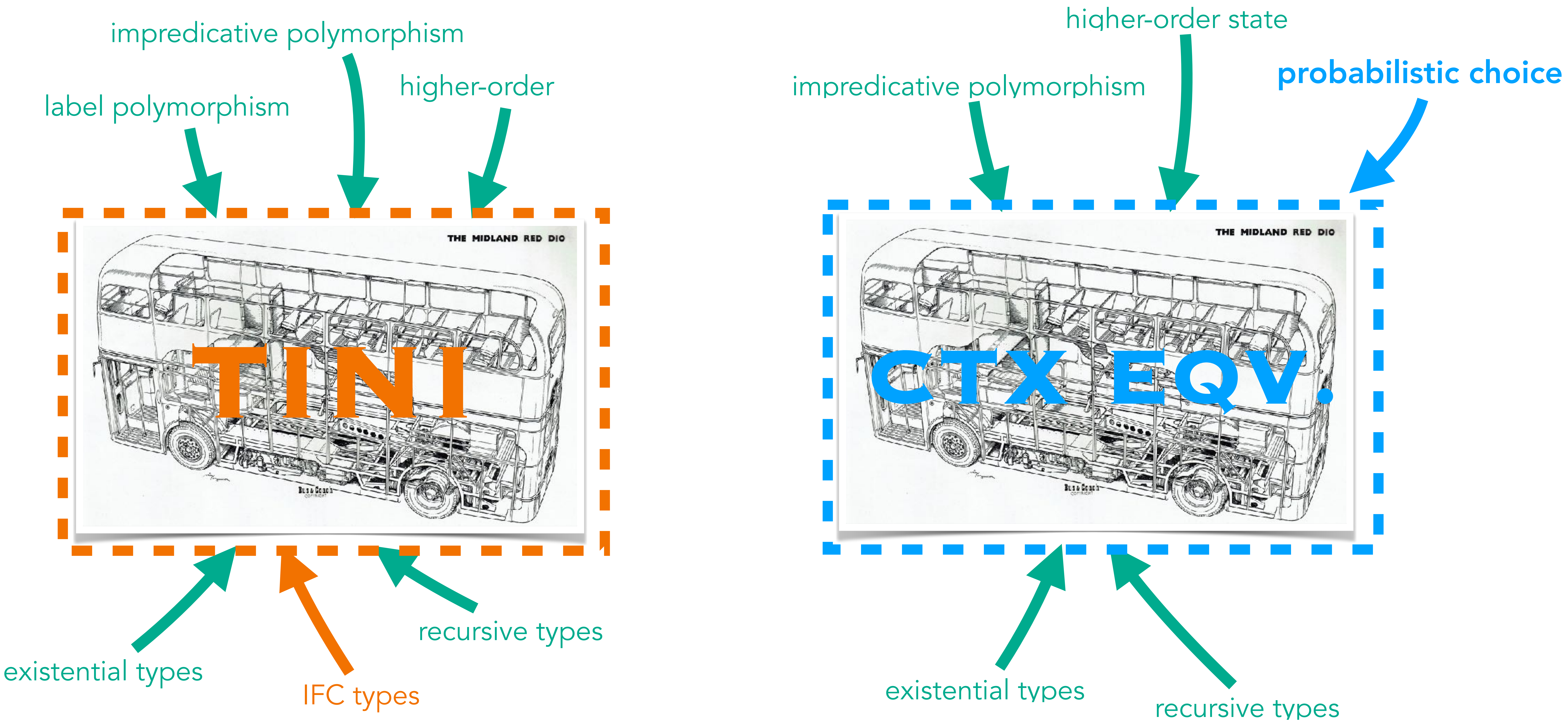
$$\begin{array}{l}\texttt{let}\ r = \texttt{ref}(\mathsf{None})\ \texttt{in}\\ \lambda\_.\ \texttt{match}\ !\,r\ \texttt{with}\\ \quad \mathsf{Some}\ (b) \Rightarrow b\\ \quad \mid \mathsf{None} \Rightarrow \texttt{let}\ b = \texttt{flip}\ \texttt{in}\\ \qquad\qquad r := \mathsf{Some}\ (b);\\ \qquad\qquad b\\ \quad \texttt{end}\end{array} \precsim_{\text{ctx}} \begin{array}{l}\texttt{let}\ \iota = \texttt{tape}\ \texttt{in}\\ \texttt{let}\ r = \texttt{ref}(\mathsf{None})\ \texttt{in}\\ \lambda\_.\ \texttt{match}\ !\,r\ \texttt{with}\\ \quad \mathsf{Some}\ (b) \Rightarrow b\\ \quad \mid \mathsf{None} \Rightarrow \texttt{let}\ b = \texttt{flip}(\iota)\ \texttt{in}\\ \qquad\qquad r := \mathsf{Some}\ (b);\\ \qquad\qquad b\\ \quad \texttt{end}\end{array} \precsim_{\text{ctx}} \begin{array}{l}\texttt{let}\ b = \texttt{flip}\ \texttt{in}\\ \lambda\_.\ b\end{array}$$

Motivating example

$$\frac{\textsc{rel-couple-tape-l}\quad f \text{ bijection} \qquad \iota \hookrightarrow \vec{b} \qquad \forall b.\, \iota \hookrightarrow \vec{b} \cdot b \twoheadrightarrow\!\!* \; \Delta \vDash e \precsim K'[\,f(b)\,] : \tau}{\Delta \vDash e \precsim K'[\,\texttt{flip}()\,] : \tau}$$

$$\begin{array}{l}
\texttt{let}\ r = \texttt{ref}(\mathsf{None})\ \texttt{in} \\
\lambda_.\ \texttt{match}\ !\,r\ \texttt{with} \\
\quad\ \mathsf{Some}\ (b) \Rightarrow b \\
\quad \mid \mathsf{None} \quad\ \Rightarrow \texttt{let}\ b = \texttt{flip}\ \texttt{in} \\
\qquad\qquad\qquad\quad r := \mathsf{Some}\ (b); \\
\qquad\qquad\qquad\quad b \\
\quad \texttt{end}
\end{array}
\precsim_{\text{ctx}}
\begin{array}{l}
\texttt{let}\ \iota = \texttt{tape}\ \texttt{in} \\
\texttt{let}\ r = \texttt{ref}(\mathsf{None})\ \texttt{in} \\
\lambda_.\ \texttt{match}\ !\,r\ \texttt{with} \\
\quad\ \mathsf{Some}\ (b) \Rightarrow b \\
\quad \mid \mathsf{None} \quad\ \Rightarrow \texttt{let}\ b = \texttt{flip}(\iota)\ \texttt{in} \\
\qquad\qquad\qquad\quad r := \mathsf{Some}\ (b); \\
\qquad\qquad\qquad\quad b \\
\quad \texttt{end}
\end{array}
\precsim_{\text{ctx}}
\begin{array}{l}
\texttt{let}\ b = \texttt{flip}\ \texttt{in} \\
\lambda_.\ b
\end{array}$$

Conclusion



impredicative polymorphism
label polymorphism
higher-order
THE MIDLAND RED D10
TINI
existential types
IFC types
recursive types
higher-order state
impredicative polymorphism
probabilistic choice
THE MIDLAND RED D10
CTX EQV.
existential types
recursive types

indeed

Higher-order separation logic is all you need!

SOLIDUM PETIT IN PROFUNDIS · UNIVERSITAS ARHUSIENSIS ·

AARHUS UNIVERSITY

CENTER FOR BASIC RESEARCH IN PROGRAM VERIFICATION

Thank you!

Simon Oddershede Gregersen

PhD dissertation defence

17 March, 2023

(Very) simple examples

- **Multiplying by zero**

$$\lambda\, v.\, v * 0$$

cannot be syntactically typed at $\mathbb{N}^{\top} \to \mathbb{N}^{\perp}$.

- **Temporary explicit "leaks"**

$$\texttt{let}\, x = \texttt{!}\, l\ \texttt{in}\, l\ \texttt{:=}\ \texttt{!}\, h;\ \ldots;\ l\ \texttt{:=}\ x$$

is not syntactically well-typed if h contains sensitive information.

$$\frac{\iota = \mathrm{fresh}(\sigma)}{\texttt{tape}, \sigma \to^{1} \iota, \sigma[\iota \mapsto \epsilon]} \qquad \frac{\sigma(\iota) = \epsilon \qquad b \in \{\texttt{true}, \texttt{false}\}}{\texttt{flip}(\iota), \sigma \to^{1/2} b, \sigma} \qquad \frac{}{\texttt{flip}(\iota), \sigma[\iota \mapsto b \cdot \vec{b}] \to^{1} b, \sigma[\iota \mapsto \vec{b}]}$$

Definition (Coupling). Let $\mu_1 \in \mathcal{D}(A)$, $\mu_2 \in \mathcal{D}(B)$. A sub-distribution $\mu \in \mathcal{D}(A \times B)$ is a *coupling* of μ_1 and μ_2 if

1. $\forall a.\ \sum_{b \in B} \mu(a, b) = \mu_1(a)$
2. $\forall b.\ \sum_{a \in A} \mu(a, b) = \mu_2(b)$

Given relation $R : A \times B$ we say μ is an R-coupling if furthermore $\mathrm{supp}(\mu) \subseteq R$. We write $\mu_1 \sim \mu_2 : R$ if there exists an R-coupling of μ_1 and μ_2.

Lemma (Composition of couplings). *Let $R : A \times B$, $S : A' \times B'$, $\mu_1 \in \mathcal{D}(A)$, $\mu_2 \in \mathcal{D}(B)$, $f_1 : A \to \mathcal{D}(A')$, and $f_2 : B \to \mathcal{D}(B')$.*

1. *If $(a, b) \in R$ then $\mathrm{ret}(a) \sim \mathrm{ret}(b) : R$.*
2. *If $\forall (a, b) \in R.\ f_1(a) \sim f_2(b) : S$ and $\mu_1 \sim \mu_2 : R$ then $\mu_1 \gg\!= f_1 \sim \mu_2 \gg\!= f_2 : S$*

Definition (Refinement Coupling). Let $\mu_1 \in \mathcal{D}(A)$, $\mu_2 \in \mathcal{D}(B)$. A sub-distribution $\mu \in \mathcal{D}(A \times B)$ is a *refinement coupling* of μ_1 and μ_2 if

1. $\forall a.\ \sum_{b \in B} \mu(a, b) = \mu_1(a)$
2. $\forall b.\ \sum_{a \in A} \mu(a, b) \leq \mu_2(b)$

Given relation $R : A \times B$ we say μ is an R-refinement-coupling if furthermore $\mathrm{supp}(\mu) \subseteq R$. We write $\mu_1 \lesssim \mu_2 : R$ if there exists an R-refinement-coupling of μ_1 and μ_2.

$$\mathrm{exec}_n(e,\sigma) \triangleq \begin{cases} \mathrm{ret}(e) & \text{if } e \in \mathit{Val} \\ \mathbf{0} & \text{if } e \notin \mathit{Val}, n = 0 \\ \mathrm{step}(e,\sigma) \gg\!= \mathrm{exec}_{(n-1)} & \text{otherwise} \end{cases}$$

$$\mathrm{exec}(\rho)(v) \triangleq \mathsf{lim}_{n\to\infty}\, \mathrm{exec}_n(\rho)(v)$$

$$\mathrm{exec}_{\Downarrow}(\rho) \triangleq \sum_{v} \mathrm{exec}(\rho)(v)$$

Lemma (Erasure). *If* $\sigma_1(\iota) \in \mathsf{dom}(\sigma_1)$ *then*

$$\mathrm{exec}_n(e_1,\sigma_1) \sim (\mathrm{step}_\iota(\sigma_1) \gg\!= \lambda\sigma_2.\ \mathrm{exec}_n(e_1,\sigma_2)) : (=)$$

Theorem (Adequacy). *Let* $\varphi : \mathit{Val} \times \mathit{Val} \to \mathit{Prop}$ *be a predicate in the meta-logic. If*

$$\mathsf{specCtx} * \mathsf{spec}(e') \vdash \mathsf{wp}\ e\ \{v.\exists v'.\ \mathsf{spec}(v') * \varphi(v,v')\}$$

is provable in Clutch then $\forall n.\ \mathrm{exec}_n(e,\sigma) \lesssim \mathrm{exec}(e',\sigma') : \varphi$.

$$\Delta \vDash_{\mathcal{E}} e_1 \precsim e_2 : \tau \triangleq \forall K.\, \mathsf{specCtx} \twoheadrightarrow \mathsf{spec}(K[\, e_2\,]) \twoheadrightarrow \mathsf{naTok}(\mathcal{E}) \twoheadrightarrow$$
$$\mathsf{wp}\ e_1\ \{v_1.\exists v_2.\, \mathsf{spec}(K[\, v_2\,]) * \mathsf{naTok}(\top) * [\![\tau]\!]_{\Delta}(v_1, v_2)\}$$

$$G(\rho) \triangleq \mathsf{specInterp}_{\bullet}(\rho)$$
$$\mathsf{specInv} \triangleq \exists \rho, e, \sigma, n.\, \mathsf{specInterp}_{\circ}(\rho) * \mathsf{spec}_{\bullet}(e) * \mathsf{heaps}(\sigma) * \mathrm{execConf}_n(\rho)(e, \sigma) = 1$$
$$\mathsf{specCtx} \triangleq \boxed{\mathsf{specInv}}^{\mathcal{N}.\mathsf{spec}}$$

$$\begin{aligned}
\mathsf{wp}_{\mathcal{E}}\ e_1\ \{\Phi\} \triangleq\ & (e_1 \in \mathit{Val} \wedge \Rrightarrow_{\mathcal{E}} \Phi(e_1)) \vee \\
& (e_1 \notin \mathit{Val} \wedge \forall \sigma_1, \rho_1. \\
& \quad S(\sigma_1) * G(\rho_1) \twoheadrightarrow {}_{\mathcal{E}}\Rrightarrow_{\emptyset} \\
& \quad \mathsf{execCoupl}(e_1, \sigma_1, \rho_1)(\lambda e_2, \sigma_2, \rho_2. \\
& \qquad \triangleright\, {}_{\emptyset}\Rrightarrow_{\mathcal{E}} S(\sigma_2) * G(\rho_2) * \mathsf{wp}_{\mathcal{E}}\ e_2\ \{\Phi\}))
\end{aligned}$$

$$\begin{aligned}
&\mathsf{execCoupl}(e_1, \sigma_1, e_1', \sigma_1')(Z) \triangleq \mu \Psi : \mathit{Cfg} \times \mathit{Cfg} \to \mathit{iProp}. \\
&\quad (\exists R.\, \mathsf{red}(e_1, \sigma_1) * \\
&\qquad \mathrm{step}(e_1, \sigma_1) \sim \mathrm{step}(e_1', \sigma_1') : R * \\
&\qquad \forall \rho_2, \rho_2'.\, R(\rho_2, \rho_2') \twoheadrightarrow \Rrightarrow_{\emptyset} Z(\rho_2, \rho_2')) \vee \\
&\quad (\exists R.\, \mathsf{red}(e_1, \sigma_1) * \\
&\qquad \mathrm{step}(e_1, \sigma_1) \sim \mathrm{ret}(e_1', \sigma_1') : R * \\
&\qquad \forall \rho_2.\, R(\rho_2, (e_1', \sigma_1')) \twoheadrightarrow \Rrightarrow_{\emptyset} Z(\rho_2, (e_1', \sigma_1'))) \vee \\
&\quad (\exists R, n.\ \mathrm{ret}(e_1, \sigma_1) \sim \mathrm{execConf}_n(e_1', \sigma_1') : R * \\
&\qquad \forall \rho_2'.\, R((e_1, \sigma_1), \rho_2') \twoheadrightarrow \Rrightarrow_{\emptyset} \Psi((e_1, \sigma_1), \rho_2')) \vee \\
&\quad \left(\bigvee_{\iota \in \sigma_1} \begin{array}{l} \exists R.\ \mathrm{step}_{\iota}(\sigma_1) \sim \mathrm{step}(e_1', \sigma_1') : R * \\ \quad \forall \sigma_2, \rho_2'.\, R(\sigma_2, \rho_2') \twoheadrightarrow \Rrightarrow_{\emptyset} \Psi((e_1, \sigma_2), \rho_2') \end{array} \right) \vee \\
&\quad \left(\bigvee_{\iota' \in \sigma_2} \begin{array}{l} \exists R.\ \mathrm{step}(e_1, \sigma_1) \sim \mathrm{step}_{\iota'}(\sigma_1') : R * \\ \quad \forall \rho_2, \sigma_2'.\, R(\rho_2, \sigma_2') \twoheadrightarrow \Rrightarrow_{\emptyset} Z(\rho_2, (e_1', \sigma_2')) \end{array} \right) \vee \\
&\quad \left(\bigvee_{(\iota, \iota') \in \sigma_1 \times \sigma_1'} \begin{array}{r} \exists R.\ \mathrm{step}_{\iota}(\sigma_1) \sim \mathrm{step}_{\iota'}(\sigma_1') : R * \\ \forall \sigma_2, \sigma_2'.\, R(\sigma_2, \sigma_2') \twoheadrightarrow \Rrightarrow_{\emptyset} \\ \Psi((e_1, \sigma_2), (e_1', \sigma_2')) \end{array} \right)
\end{aligned}$$