A Fu y Abstract May est n e ant cs or Concurrent b ects

A an Je rey — C I De au n vers ty — C ca o IL A — a e rey cs epau e u

Ju an at e — C G n vers ty o ussex — Br ton K — u anr co s susx ac u

ctober

Abstract

s paper prov es a u y abstract se ant cs or a var ant o t e concurrent ob ect ca cu us e e ne ay test n or concurrent ob ect co ponents an t en c aracter se t us n a trace se ant cs nsp re by ML nteract on a ra s e a n resu t o t s paper s to s ow t at t e trace se ant cs s u y abstract or ay test n s s t e rst suc resu t or a concurrent ob ect an ua e

Introduction

Aba an Car e s ob ect ca cu us s a n a an ua e or nvest at n eatures o ob ect an ua es suc as encapsu ate state subtyp n an se var ab es Gor on an Han n a e concurrent eatures to t e ob ect ca cu us to pro uce t e concurrent ob ect ca cu us

r or wor on t e ob ect ca cu us as concentrate on t e operat ona be av our o ob ect syste s an type syste s w c prov e type sa ety uarantees e c osest paper to ours s Gor on an ees s u y abstract se ant cs or t e utab e s n e t rea e ob ect ca cu us ere as been no wor on prov n u y abstract se ant cs or concurrent utab e ob ects

In t s paper we present t e rst u y abstract test n se ant cs or a var ant o Gor on an Han n s concurrent ob ect ca cu us w t out subtyp n e ac o subtyp n ere a or s a s p er presentat on o t e abe e trans t ons an traces but we ant c pate t at t e proo tec n ques use ere are robust enou to cater or subtyp n a so s se ant cs was nsp re by ML nteract on a ra s w c are a co on too or v sua s n nteract ons w t ob ect syste s

Interaction diagrams

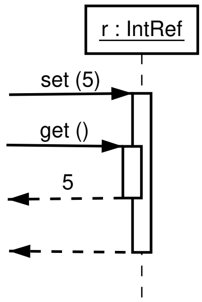
Interact on a ra s n part cu ar sequence a ra s were eve ope by Jacobson an are now part o t e n e Mo e n Lan ua e stan ar Interact on a ra s recor t e essa es sent between ob ects o a co ponent n an ob ect syste ese essa es nc u e et o ca s

esearc part a y supporte by t e u e Foun at on n vers ty o ussex tec n ca report

and returns. Interaction diagrams include other forms of message but we will not use these in this paper.

A simple interaction with an integer reference object r of type `IntRef`

Sequence diagrams can be used for multithreaded applications, for example:



Here two threads independently call methods of the object r, creating a race condition. In our textual representation we give the threads names and we decorate each message with the thread responsible for the message.

thread1 call r.set(

thread2 call r.get

- Messa es are nco n or out o n essa e ca s or atc n out o n or nco n returns

- Messa es are ecorate w t t rea ent ers

- Messa es ay nc u e res na es

e ave on y use a very s a subset o sequence a ra s w c n turn s a very s a subset o ML but n t s paper we w s ow t at t s s a subset s very express ve an n part cu ar prov es a u y abstract se ant cs

. The object calculus

e ob ect ca cu us s a n a an ua e or o e n ob ect base pro ra n Aba an Car e prov e a type syste an operat ona se ant cs or a var ety o ob ect ca cu an prove type sa ety or t e Gor on an Han n ave s nce exten e t s an ua e to nc u e concurrent eatures

In t s paper we s a nvest ate a var ant o Gor on an Han n s concurrent ob ect ca cu us w c nc u es

- A eap o na e ob ects an t rea s

- rea s can ca or up ate ob ect et o s can co pare ob ect or t rea na es or equa ty can create new ob ects an t rea s an can scover t e r own t rea na e

- An operat ona se ant cs base on t e π ca cu us an a s p e type syste

- A trace se ant cs as scusse n ect on

e are not cons er n any o t e ore a vance eatures o t e ob ect ca cu us or t e concurrent ob ect ca cu us suc as recurs ve types ob ect c on n an ob ect oc n s s ust or s p c ty an we o not see any tec n ca prob e s w t ncorporat n t ese eatures nto our an ua e

In anot er stran o researc D B as o an F s er a so es ne a ca cu us or o e n perat ve concurrent ob ect base syste s As w t Aba an Car e s ob ect ca cu us an ts var ous extens ons t e e p as s n D B as o an F s er s wor s a a n on type syste s an sa ety propert es or t e

. Full abstraction

e prob e o u abstract on was rst ntro uce by M ner an nvest ate n ept by ot n Fu abstract on was rst propose or var ants o t e λ ca cu us but as s nce been nvest ate or process a ebras t e π ca cu us t e ν ca cu us Concurrent ML an t e utab e ob ect ca cu us e can then e ne the *may testing preorder* as $C \sqsubseteq_{ay} C$ whenever

or any appropr ate y type C
C C s success u then C C s success u

n ortunate y a thou t s very s p e to e ne an s qu te ntu t ve ay test n s o ten very cu t to reason about rect y because o the quant cat on over any appropr ate y type C In pract ce we requ re a proo tec n que wh c we can use to show resu ts about ay test n

ne approac s to use a *trace* se ant cs ven by e n n poss b e execut ons o co ponents $C \stackrel{s}{\Longrightarrow} C'$ where s s a sequence o essa es e then wr te Traces (C or the set o a traces o C e say that

- races are *sound* or ay test n when Traces (C Traces (C p es $C \sqsubseteq_{ay} C$

- races are *complete* or ay test n when $C \sqsubseteq_{ay} C$ p es Traces (C Traces (C

- races are *fully absdwhereTj11.993tTJ/R1.991Tf1.19Tdfor-2may-22.9test-21.9ing-2.whenTJsi.9911.so/R11.9*

Co ponents	C	$=$	$\mid C \; C \mid \nu(n \;\; T \;\; .C \mid n \; O \;\; \mid n\;t$
b ects	O	$=$	$l = M, \ldots, l = M$
Met o s	M	$=$	$\varsigma(n \;\; T \;\; .\lambda(x \;\; T, \ldots, x \;\; T \;\; .\;t$
rea s	t	$=$	$v \mid \mathrm{stop} \mid \mathrm{let}\; x \;\; T = e \;\mathrm{in}\; t$

e n t on o e s f as zero ar u ent et o s

- A e ec arat on $f = v$ n an ob ect s syntax su ar or a et o ec arat on $f = \varsigma(n$ T $.\lambda($ $.v$
- A e type f T n an ob ect type s syntax su ar or a et o type f $(\rightarrow T$
- A e access express on $v.f$ s syntax su ar or a et o ca $v.f($
- A e up ate express on $n.f = v$ s syntax su ar or a et o up ate $n.f$ $(\varsigma(p$ T $.\lambda($ $.$ v

In a t on we ave restr cte any subexpress ons o an express on to be va ues rat er t an u express ons or exa p e n a et o ca $v.l(\vec{v}$ we requ re t e ob ect an t e ar u ents to be va ues rat er t an express ons $e.l(\vec{e}$ s a es t e operat ona se ant cs uc eas er to e ne an oes not restr ct t e express v ty o t e an ua e or exa p e we can e ne $(e.l(\vec{e}$ $\equiv$ (let $x = e$ in let $\vec{x} = \vec{e}$ in $x.l(\vec{x}$ ar y t e st nct on between t rea s an express ons a es t e

A thread (t) consists of a stack of let expressions terminated either by a return value

$$(\text{let } x_1 : T_1 = e_1 \text{ in } \cdots \text{let } x_n : T_n = e_n \text{ in } v)$$

or by a deadlocked stop thread

$$(\text{let } x_1 : T_1 = e_1 \text{ in } \cdots \text{let } x_n : T_n = e_n \text{ in stop})$$

Each expression is either itself a thread, or

- an if expression if $v_1 = v_2$ then e_1 else e_2
- a method call $v.l(\vec{v})$
- a method update $n.l \Leftarrow M$ on a named object
- a new object new O
- a new thread new(t), or
- the current thread name currentthread.

Each value is simply a name or a variable and we defer the discussion of types until Section .

Static semantics

The static semantics for our concurrent object calculus is given in Figures . Most of the rules are straightforward adaptations of those given by Abadi and Cardelli . The main judgment is $\Delta \vdash C : \Theta$ which is read as the component C uses names Δ and defines names Θ. For example, we define $C_1(v)$, C_2 and IntRef as

$$
\begin{aligned}
& C_1(v) \equiv p \mapsto [\\
& \quad \text{contents} = v,\\
& \quad \text{set} = \varsigma(\text{this} : \text{IntRef}) . \lambda(x : \text{Int}) . (\text{this.contents} \Leftarrow x, x),\\
& \quad \text{get} = \varsigma(\text{this} : \text{IntRef}) . \lambda() . (\text{this.contents})\\
&]\\
\\
& C_2 \equiv n \mapsto (\\
& \quad \text{let } x = p.\text{get}() \text{ in } p.\text{set}(x + 1), \text{stop}\\
&)\\
\\
& \text{IntRef} \equiv [\\
& \quad \text{contents} : \text{Int}
\end{aligned}
$$

$$\frac{}{\Delta \vdash \cdot\ (} \qquad \frac{,\Delta, n\ \ T \vdash O\ \ T}{\Delta \vdash n\ O\ \ (n\ \ T} \qquad \frac{,\Delta, n\ \ \mathrm{thread} \vdash t\ \ \mathrm{none}}{\Delta \vdash n \ t\ \ (n\ \ \mathrm{thread}}$$

$$\frac{\Delta, \Theta\ \ \vdash C\ \ \Theta \qquad \Delta, \Theta\ \ \vdash C\ \ \Theta}{\Delta \vdash (C\ \ \vdash C\ \ (\Theta\ \ , \Theta} \qquad \frac{\Delta \vdash C\ \ \Theta, n\ \ T}{\Delta \vdash \nu(n\ \ T\ . C\ \ \Theta}$$

F ure u es or u e ent $\Delta \vdash C\ \ \Theta$

$$\frac{\Gamma, \Delta \vdash M\ \ T.l \quad \cdots \quad \Gamma, \Delta \vdash M_k\ \ T.l_k}{\Gamma, \Delta \vdash\ \ l\ = M\ , \ldots, l_k = M_k\ \ T}$$

F ure u e or u e ent $\Gamma, \Delta \vdash O\ \ T$ w en $T = l\ \ L\ , \ldots, l_k\ \ L_k$

$$\frac{\Gamma, x\ \ T\ , \ldots, x_k\ \ T_k, \Delta, n\ \ T \vdash t\ \ U}{\Gamma, \Delta \vdash \varsigma(n\ \ T\ \ . \lambda(x\ \ T\ , \ldots, x_k\ \ T_k\ \ t\ \ T.l}$$

F ure u e or u e ent $\Gamma, \Delta \vdash M\ \ T.l$ w en $T = \ \ldots, l\ \ (T\ , \ldots, T_k \to U, \ldots$ an $T.l$ s t e recor l se ecte ro T

$$\frac{\begin{array}{c}\Gamma, \Delta \vdash v\ \ T \qquad \Gamma, \Delta \vdash v\ \ T \\ \Gamma, \Delta \vdash e\ \ T \qquad \Gamma, \Delta \vdash e\ \ T\end{array}}{\Gamma, \Delta \vdash \mathrm{if}\ v\ = v\ \ \mathrm{then}\ e\ \ \mathrm{else}\ e\ \ T}$$

$$\frac{\begin{array}{c}\Gamma, \Delta \vdash v\ \ \ \ldots, l\ \ (T\ , \ldots, T_k \to T, \ldots \\ \Gamma, \Delta \vdash v\ \ T \quad \cdots \quad \Gamma, \Delta \vdash v_k\ \ T_k\end{array}}{\Gamma, \Delta \vdash v.l(v\ , \ldots, v_k\ \ T} \qquad \frac{\Gamma, \Delta \vdash n\ \ T \qquad \Gamma, \Delta \vdash M\ \ T.l}{\Gamma, \Delta \vdash n.l \vdash M\ \ T}$$

$$\frac{\Gamma, \Delta \vdash O\ \ T}{\Gamma, \Delta \vdash \mathrm{new}\ O\ \ T} \qquad \frac{\Gamma, \Delta \vdash t\ \ T}{\Gamma, \Delta \vdash \mathrm{new}\ t\ \ \mathrm{thread}} \qquad \frac{}{\Gamma, \Delta \vdash \mathrm{currentthread}\ \ \mathrm{thread}}$$

$$\frac{\Gamma, \Delta \vdash e\ \ T \qquad \Gamma, x\ \ T\ , \Delta \vdash t\ \ T}{\Gamma, \Delta \vdash \mathrm{let}\ x\ \ T\ = e\ \mathrm{in}\ t\ \ T} \qquad \frac{}{\Gamma, \Delta \vdash \mathrm{stop}\ \ T} \qquad \frac{}{\Gamma, x\ \ T, \Gamma', \Delta \vdash x\ \ T} \qquad \frac{}{\Gamma, \Delta, n\ \ T, \Delta' \vdash n\ \ T}$$

F ure u es or u e ent $\Gamma, \Delta \vdash e\ \ T$

ar ab e contexts $\Gamma\ = x\ \ T, \ldots, x\ \ T$ a e contexts $\Delta, \Theta, \Sigma, \Phi\ = n\ \ T, \ldots, n\ \ T$

In var ab e contexts var ab es ust be un que an are v ewe up to reor er n
In na e contexts na es ust be un que types ust not be none an are v ewe up to reor er n

F ure yntax o na e an var ab e contexts

enever Δ C Θ conta.ns a subexpress.on o [redacted] n
appears .n Θ

.s .s .nten e to capture t e co on so tware en .n [redacted] s ou
[redacted]t export utab e e s .nstea t ey s ou export su. [redacted] e t e
[redacted]n urat.ons C an C above are wr.te c ose s.nce t [redacted] ow.n
[redacted] ponent w .c wr.tes .rect y to p.contents .s not w [redacted]

$$C \equiv n \text{let } x = p.\text{contents in } p.\text{co} \ldots$$

[redacted]or t e re a.n er o t e paper we w. requ.re co po [redacted]
[redacted] op.n a u y abstract se ant.cs uc s. p er s.nc [redacted]
[redacted].rect y

[redacted] Dynamic semantics

[redacted] e yna .c se ant.cs or our concurrent ob ect ca cu [redacted] n .n [redacted]
e e ne t ree re at.ons between co ponents

- $\equiv$ structura con ruence represents t e east con [redacted] n co [redacted]
  ax.o s .n F. ure

[redacted]

t

$$C \equiv C \quad (C \; C \quad C \equiv C \; (C \; C \qquad C \; C \equiv C \; C$$

$$C \; \nu(n \; T \; . C \equiv \nu(n \; T \; . (C \; C \qquad \nu(n \; T \; . \nu(n \; T \; . C \equiv \nu(n \; T \; . \nu(n \; T \; . C$$

F ure Ax o s or structura con ruence w ere n s not ree n C

$$n\, \text{let } x \; T = v \text{ in } t \xrightarrow{\beta} n\, t \; v/x$$

$$n\, \text{let } x \; T = (\text{let } x \; T = e \text{ in } e \text{ in } t \xrightarrow{\beta} n\, \text{let } x \; T = e \text{ in } (\text{let } x \; T = e \text{ in } t$$

$$n\, \text{let } x \; T = (\text{if } v = v \text{ then } e \text{ else } e \text{ in } t \xrightarrow{\beta} n\, \text{let } x \; T = e \text{ in } t$$

$$n\, \text{let } x \; T = (\text{if } v = v \text{ then } e \text{ else } e \text{ in } t \xrightarrow{\beta} n\, \text{let } x \; T = e \text{ in } t \qquad v \; v$$

$$n\, \text{let } x \; T = \text{new } O \text{ in } t \xrightarrow{\beta} \nu(p \; T \; . (p \; O \; n\, \text{let } x \; T = p \text{ in } t \qquad 2\; else$$

$\text{let } x \; T =$

. Testing preorder

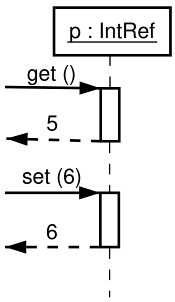
e w now e ne t e test n se ant cs or our concurrent ob ect ca cu us e w o t s by e n n a not on o *barb*

$(\Delta, n$ thread □ C

then where $C(v$ is the new object..., we have

$$
\begin{aligned}
&(\cdot\, C \,(\; \Theta \\
&\xrightarrow{\nu(n\ \mathsf{thread}\ .n\ \mathsf{call}\ p.\mathsf{get}(\ )?} \\
&(\cdot\,(C\,(\quad n\ \mathrm{let}\ x = p.\mathrm{get}(\ \ \mathrm{in\ return}\ x \quad \Theta' \\
&\Box \\
&(\cdot\,(C\,(\quad n\ \mathrm{return} \quad \Theta' \\
&\xrightarrow{n\ \mathsf{return}\ } \\
&(\cdot\,(C\,(\quad n\ \mathrm{block} \quad \Theta' \\
&\xrightarrow{n\ \mathsf{call}\ p.\mathsf{set}(\ )?} \\
&(\cdot\,(C\,(\quad n\ \mathrm{let}\ x = p.\mathrm{set}(\ \ \mathrm{in\ return}\ x \quad \Theta' \\
&\Box \\
&(\cdot\,(C\,(\quad n\ \mathrm{return} \quad \Theta' \\
&\xrightarrow{n\ \mathsf{return}\ } \\
&(\cdot\,(C\,(\quad n\ \mathrm{block} \quad \Theta'
\end{aligned}
$$

which corresponds to the interaction diagram



For any component $(\Delta \cdot C \;\; \Theta$ we define its traces to be

$$\mathrm{Traces}(\Delta \cdot C \;\; \Theta) = \{ s \mid (\Delta \cdot C \;\; \Theta \stackrel{s}{\Longrightarrow} (\Delta' \cdot C' \; \ldots$$

t e base an ua e w c wou ave been reac e a t e co ponent an test actua y nteracte
s operat on o er n s e ne be ow

. The merge operator

De ne t e part a *merge* operator $C \bowtie C$ on co ponents as t e sy etr c operator e ne up to $\equiv$ w ere

$$
\begin{aligned}
\bowtie C &= C \\
(\nu(p\ T\ .C\ \bowtie C &= \nu(p\ T\ .(C \bowtie C \\
(p\ O\ C\ \bowtie C &= p\ O\ (C \bowtie C \\
(p\ t\ C\ \bowtie C &= p\ t\ (C \bowtie C \\
(n\ t\ C \bowtie (n\ t\ C &= n\ t \bowtie t\ (C \bowtie C
\end{aligned}
$$

w en n dom $(C\ ,C$ an p fn $(C$

e over oa notat on an e ne t e part a er e operator $t \bowtie t$ on t rea s as t e sy etr c operator w ere

$$
\begin{aligned}
(\text{let } x\ T = \text{block in } t \bowtie \text{stop} &= \text{stop} \\
(\text{let } x\ T = \text{block in } t \bowtie (\text{let } y\ U = \text{return}(v\ T \text{ in } t &= (\text{let } y\ U = \text{block in } t \bowtie (t\ v/x \\
(\text{let } x\ T = \text{block in } t \bowtie (\text{let } y\ U = e \text{ in } t &= \text{let } y\ U = e \text{ in } ((\text{let } x\ T = \text{block in } t \bowtie t
\end{aligned}
$$

w en e s b oc return ree an y fv $(t$

Lemma . *If* Δ $(C\ C\ \Theta$ *then* $(C \bowtie C \equiv (C\ C$.

Proof An n uct on on t e e n t on o $C \bowtie C$ □

Lemma . *If* $C \bowtie C \equiv C$ *and* $C \downarrow_b$ *then* $C\downarrow_b$.

Proof An n uct on on t e e n t on o $C \bowtie C$ □

. Trace composition and decomposition

G ven a trace s we wr te s or t e co p e entary trace

$\varepsilon = \varepsilon\ C$ An vBI IMtrue H B C ID EIe been

Proof G.ven .n Appen .x A

Corollary . *For any components* $(\Delta,\Phi \, C \; \Theta,\Sigma$ *and* $(\Theta,\Phi \, C \; \Delta,\Sigma$ *such that* $C \; ⩕ \; C \; \equiv C$ *and* C_b *then there exists some trace* s *such that* $(\Delta,\Phi \, C \; \Theta,\Sigma \stackrel{s}{\Longrightarrow} (\Delta',\Phi \, C' \; \Theta',\Sigma'$ *and* $(\Theta,\Phi \, C \; \Delta,\Sigma \stackrel{s}{\Longrightarrow} (\Theta',\Phi \, C' \; \Delta',\Sigma'$ *where either* $C'\downarrow_b$ *or* $C'\downarrow_b$.

Proof e now t at C_b w .c te s us t at $C \; C''$ or so e C'' suc t at $C''\downarrow_b$ e use ropos. t.on art to obta.n a trace s suc t at

$$(\Delta,\Phi \, C \; \Theta,\Sigma \stackrel{s}{\Longrightarrow} (\Delta'',\Phi \, C'' \; \Theta'',\Sigma''$$
$$(\Theta,\Phi \, C \; \Delta,\Sigma \stackrel{s}{\Longrightarrow} (\Theta'',\Phi \, C'' \; \Delta'',\Sigma''$$

w ere $\nu(\Delta'',\Theta'',\Sigma'' \setminus \Delta,\Theta,\Sigma \; .(C'' ⩕ C'' \equiv C''$. G.ven t at $C''\downarrow_b$ we now t at $(C'' ⩕ C'' \downarrow_b$ a so By t e e n.t.on o ⩕ we see t at one o t e o ow.n or t e.r sy etr.c counterparts ust o

- $C''\downarrow_b$ an we are one or
- $C'' \equiv \nu(\Delta \; .(n t \; C'''$ an $C'' \equiv \nu(\Delta \; .(n t \; C'''$ w ere $n t ⩕ t \downarrow_b$ e now procee by .n uct.on on t e e n.t.on o $t ⩕ t$ to s ow t at or a suc C'' an C'' we can n s w ere

$$(\Delta'',\Phi \, C'' \; \Theta'',\Sigma'' \stackrel{s}{\Longrightarrow} (\Delta',\Phi \, C' \; \Theta',\Sigma'$$
$$(\Theta'',\Phi \, C'' \; \Delta'',\Sigma'' \stackrel{s}{\Longrightarrow} (\Theta',\Phi \, C' \; \Delta',\Sigma'$$

an e.t er $C'\downarrow_b$ or $C'\downarrow_b$ ere are two cases up to sy etry o ⩕

- I $t = \text{let } x \; T = \text{block in } t'$ an $t = \text{let } y \; U = b.\text{succ}($ in t' t en $C''\downarrow_b$
- I $t = \text{let } x \; T = \text{block in } t'$ an $t = \text{let } y \; U = \text{return}(v \; T$ in t' t en we ave

$$(\Delta'',\Phi \, C'' \; \Theta'',\Sigma'' \xrightarrow{\nu(\Delta' \; .n \text{return } v?} (\Delta'',\Delta',\Phi \; \nu(\Delta \; .(n t' \; v/x \; C''' \; \Theta',\Sigma'$$
$$(\Theta'',\Phi \, C'' \; \Delta'',\Sigma'' \xrightarrow{\nu(\Delta' \; .n \text{return } v} (\Theta'',\Phi \; \nu(\Delta'' \; .(n \text{let } y \; U = \text{block in } t' \; C''' \; \Delta'',\Delta',\Sigma''$$

w ere $\Delta = (\Delta',\Delta''$ an oreover

$$n t ⩕ t \equiv n (\text{let } y \; U = \text{block in } t' ⩕ t \; v/x \downarrow_b$$

so by .n uct.ve ypot es.s

$$(\Delta'',\Phi \, C'' \; \Theta'',\Sigma'' \xrightarrow{\nu(\Delta' \; .n \text{return } v?} \stackrel{s}{\Longrightarrow} (\Delta',\Phi \, C' \; \Theta',\Sigma'$$
$$(\Theta'',\Phi \, C'' \; \Delta'',\Sigma'' \xrightarrow{\nu(\Delta' \; .n \text{return } v} \stackrel{s}{\Longrightarrow} (\Theta',\Phi \, C' \; \Delta',\Sigma'$$

an e.t er $C'\downarrow_b$ or $C'\downarrow_b$ as requ.re

□

.5 Proof of soundness

Theorem . (Soundness of traces for may testing) *If* Traces$(\Delta \triangleright C\ \Theta) \subseteq$ Traces$(\Delta \triangleright C\ \Theta)$ *then* $\Delta \models C \sqsubseteq_{may} C\ \Theta$

Proof uppose that Traces$(\Delta \triangleright C\ \Theta) \subseteq$ Traces$(\Delta \triangleright C\ \Theta)$ an that we ave $(\Theta, b$ barb $\triangleright C_{\bullet}\ \Delta$ such that $(C\ C_{\bullet})_b$, we ust show that $(C\ C_{\bullet})_b$ a so
ow since $(C\ C_{\bullet})_b$ we can use Coro ary to et

$$(\Delta, b\ \text{barb} \triangleright C\ \Theta \stackrel{s}{\Longrightarrow} (\Delta', b\ \text{barb} \triangleright C'\ \Theta', \Sigma')$$

$$(\Theta, b\ \text{barb} \triangleright C_{\bullet}\ \Delta \stackrel{s}{\Longrightarrow} (\Theta', b\ \text{barb} \triangleright C'_{\bullet}\ \Delta', \Sigma')$$

$\overline{\Delta \square \varepsilon \;\; \mathrm{trace}\, \Theta}$

n

- I n □ threads $(s$ then n s ba ance n s
- I n s ba ance n s an s then n s ba ance n s s
- I n s ba ance n s then n s ba ance n $\nu(\Delta\ .n$□call $p.l(\vec{n}$ □? $s\,\nu(\Theta\ .n$□return v□
- I n s ba ance n s then n s ba ance n $\nu(\Theta\ .n$□call $p.l(\vec{n}$ □ $s\,\nu(\Delta\ .n$□return v□?

De ne pop $n(s$ as

- I n s ba ance n s then pop $n(s$ $=$ □
- I n s ba ance n s an $a =$

Proof Easy in uction on s. $\square$

Lemma .

1. *If C is block/return free and* $(\Delta, C, \Theta) \stackrel{s}{\Longrightarrow} \xRightarrow{\nu(\Theta'). n\,\mathsf{return}\, v} $ *then* $s = s \ \nu(\Delta'). n\,\mathsf{call}\, p.l(\vec{v}) \, s$ *where n is balanced in s.*

2. *If C is block/return free and* $(\Delta, C, \Theta) \stackrel{s}{\Longrightarrow} \xRightarrow{\nu(\Delta'). n\,\mathsf{return}\, v} $ *then* $s = s \ \nu(\Theta'). n\,\mathsf{call}\, p.l(\vec{v}) \ s$ *where n is balanced in s.*

Proof e prove these properties si u taneous y by an in uction on the en th of s. e on y show the ar u ent for art as art can be shown in a si i ar anner. By ana ysis of the ru es of the ts we have

$$(\Delta, C, \Theta) \stackrel{s}{\Longrightarrow} (\Delta'', C'', n\,\mathsf{let}\ x : T = \mathsf{return}\ (v : U)\ \mathsf{in}\ t, \Theta'') \xrightarrow{\nu(\Theta'). n\,\mathsf{return}\, v}$$

ow partition s into s s pic in s

Case $s = s' \nu(\Delta' . n\,\mathsf{call}\, p.l(\vec{v}\,)?$ e now that

$$(\Delta \; C \quad \Theta \quad \overset{s'}{\Longrightarrow} \; (\Delta, \Delta(s' \; C' \quad \Theta, \Theta(s' \quad \xrightarrow{\nu(\Delta' . n\,\mathsf{call}\, p.l(\vec{v}\,)?}$$

so we have that eit er

$$C' \equiv \nu(\Delta' . \nu(\Delta'' . n\,\mathsf{let}\, x \quad T = \mathsf{block}\ \mathsf{in}\ t \; C''$$

or n $\Delta, \Delta(s'$ an n s a res t rea to s' e can app y t e n uct ve ypot es s to s' to see that Δ s' trace Θ an we cons er pop $n(s'$ n $\Delta, \Delta(s'$ an n s res t rea to s' then pop $n(s'$ s necessar y t erw se we now that $C' \equiv \nu(\Delta' . \nu(\Delta'' . n\,\mathsf{let}\, x \quad T = \mathsf{block}\ \mathsf{in}\ t \; C''$ an t ere ore t e ast act on w c cou ave occurre at n ust ave been an output that s pop $n(s' = \gamma$ In both cases we see that

n s nput enab e n Δ s' trace Θ

e now that $(\Delta, \Delta(s' \; C' \quad \Theta, \Theta(s' \xrightarrow{\nu(\Delta' . n\,\mathsf{call}\, p.l(\vec{v}\,)?}$ an we now that t e s e con t ons on t e trans t on ru e or $\nu(\Delta' . \gamma?$ act ons uarantees that

dom $(\Delta'$ fn $(\vec{v}$

e a so now that t e s e con t ons on ru e or ca nput act ons uarantees that

$$,\Delta, \Delta(s' \, ,\Theta, \Theta(s' \, ,\Delta' \; p.l(\vec{v} \quad T \text{ an } \; p \; \Theta, \Theta(s'$$

e use t s to see that

$,\Theta, \Theta(s' \, ,\Delta' \; p \quad \ldots l \; (\vec{T} \; \to T$

an

$,\Delta, \Delta(s' \, ,\Theta, \Theta(s' \, ,\Delta' \; \vec{v} \quad \vec{T}$

Last y t s easy to see that

$,\Delta, \Delta(s' \, ,\Theta, \Theta(s' \, ,\Delta' \; n$ thread

e co ect t e state ents to et er to see that t ey or t e ypot eses o t e type ru e w c a ows us to conc u e

$$\Delta \; s' \nu(\Delta' . n\,\mathsf{call}\, p.l(\vec{v}\,)? \text{ trace } \Theta$$

as requ re

Case $s = s' \nu(\Theta' . n\,\mathsf{call}\, p.l(\vec{v}\,)$ ar to prev ous case

Case $s = s' \nu(\Theta' . n\,\mathsf{return}\, v$ e now that

$$(\Delta \; C$$

$\Delta \vdash s'$ trace Θ

an we notice that because C is b oc return ree we can app y Le a to et

$$s' = s \ \nu(\Delta' \ .\ n \ \mathrm{call}\ p.l(\vec{v}\ ) \ s$$

where n is ba ance in s Given this we see that

$$\mathrm{pop}\, n\,(s \ \nu(\Delta' \ .\ n \ \mathrm{call}\ p.l(\vec{v}\ ) \ s \ = \nu(\Delta' \ .\ n \ \mathrm{call}\ p.l(\vec{v}\ )$$

hence

$$\mathrm{pop}\, n\,(s' \ = \nu(\Delta' \ .\ n \ \mathrm{call}\ p.l(\vec{v}\ )$$

A ain the si e con itions on the transition ru e or $\nu(\Theta' \ .\ \gamma$ uarantee that

$$\mathrm{dom}\,(\Theta' \ \ \mathrm{fn}\,(v$$

e a so now by an the act that pre xes o we type traces are a so we type that

$$\Delta \vdash s \ \nu(\Delta' \ .\ n \ \mathrm{call}\ p.l(\vec{v}\ ) \ \text{trace}\ \Theta$$

an we see that this ust have been in erre usin a hypothesis

$$,\Theta,\Theta(s \ \ p \ \ .\ l \ (\vec{U} \ \to U' \ldots$$

which by wea enin ives us

$$,\Theta,\Theta(s' \ \ p \ \ .\ l \ (\vec{U} \ \to U' \ldots$$

Last y because

$$(\Delta,\Delta(s' \ \ C' \ \ \Theta,\Theta(s'$$

an

$$C' \equiv C' \ n \ \mathrm{let}\ x \ \ T = \mathrm{return}\,(v \ \ U \ \ \mathrm{in}\ t$$

we see that

$$,\Delta,\Delta(s' \ ,\Theta,\Theta(s' \ ,\Theta' \ \ v \ \ U$$

o by Le a to ether with the typin si e con itions or ca input transitions we have that $U = U'$ an so

$$,\Delta,\Delta(s \ ,\Theta,\Theta(s \ ,\Theta' \ \ v \ \ U$$

e co ect the state ents to ether to see that they or the hypotheses o the type ru e which a ows us to conc u e

$$\Delta \vdash s' \ \nu(\Theta' \ .\ n \ \mathrm{return}\ v \ \ \text{trace}\ \Theta$$

as require

Case $s = s' \ \nu(\Delta' \ .\ n \ \mathrm{return}\ v\ )$ i i ar to previous case $\square$

. Information order on traces

The information preorder on traces $\Delta \vdash r \sqsubseteq s : \mathrm{trace}\ \Theta$ is generated by axioms where in each case we require both sides of the inequation to be well typed traces.

$$
\begin{array}{c}
\Delta \vdash s \sqsubseteq s\,r : \mathrm{trace}\ \Theta \\
\Delta \vdash s\gamma? \sqsubseteq s : \mathrm{trace}\ \Theta \\
\Delta \vdash s\gamma\,?\gamma\ \ r \sqsubseteq s\gamma\ \ \gamma\,?r : \mathrm{trace}\ \Theta \\
\Delta \vdash s\nu(\Delta\ \ .\gamma\ ?\gamma\ ?r \sqsubseteq s\nu(\Delta\ \ .\gamma\ ?\gamma\ ?r : \mathrm{trace}\ \Theta \\
\Delta \vdash s\nu(\Theta\ \ .\gamma\ \ \gamma\ \ r \sqsubseteq s\nu(\Theta\ \ .\gamma\ \ \gamma\ \ r : \mathrm{trace}\ \Theta
\end{array}
$$

Lemma . **(Information Order Duality)** *If* $\Delta \vdash r\gamma \sqsubseteq s\gamma : \mathrm{trace}\ \Theta$ *and* $\mathrm{fn}(\gamma\ \cap \Theta(r\ \ = \emptyset$ *and* $\gamma \notin s, r$ *then* $\Theta \vdash s \sqsubseteq r : \mathrm{trace}\ \Delta$.

Proof We write $\Delta \vdash r \sqsubseteq^{n} s : \mathrm{trace}\ \Theta$ if $\Delta \vdash r \sqsubseteq s : \mathrm{trace}\ \Theta$ can be derived using n instances of

Proposition . **(Information Order Closure)** *If* $(\Delta \; C \quad \Theta \stackrel{s}{\Longrightarrow}$ *and* $\Delta \; r \; s$ trace Θ *then* $(\Delta \; C \quad \Theta \stackrel{r}{\Longrightarrow}$.

Proof ow t at t e o ow n a ra s can be co p ete w en thread (γ ⊟ thread (γ

.

$\mathrm{Comp}(\Delta \; s \;\; \mathrm{trace}\,\Theta \;\; = \nu(\Theta(s \;\; , \mathrm{ref} \;\; \mathrm{Ref}, \mathrm{state}_\varepsilon \;\; \mathrm{State} \;\; .($

$\quad \mathrm{ref}\ \mathrm{val} = \mathrm{state}_\varepsilon$

$\quad \mathrm{state}_\varepsilon\ \mathrm{State}(\Delta \;\; \varepsilon \leq s \;\; \mathrm{trace}\,\Theta$

$\quad \prod\{p\ l_i = \mathrm{ref.val.inCall}_{p.l_i\ L_i} \mid i = \;\; \ldots n \mid p \;\; l_i \;\; L_i \mid i = \;\; \ldots n \;\; \Theta, \Theta(s \;\; \}$

$\quad \prod\{n\ \mathrm{ref.val.out}_{\mathsf{none}}(\;\; \mid n \;\; \mathrm{thread} \;\; \Theta, \Theta(s \;\; \}$

$\mathrm{Ref} = \;\; \mathrm{val} \;\; \mathrm{State}$

$\mathrm{State} = \;\; \mathrm{out}_T \;\; (\;\; \to T, \mathrm{inReturn}_T \;\; (T \;\; \to T, \mathrm{inCall}_{p.l\ L} \;\; L$

$\mathrm{State}(\Delta \;\; r \leq s \;\; \mathrm{trace}\,\Theta \;\; = ($

$\quad \mathrm{out}_T = \mathrm{Out}_T(\Delta \;\; r \leq s \;\; \mathrm{trace}\,\Theta \;\; ,$

$\quad \mathrm{inReturn}_T = \mathrm{InReturn}_T(\Delta \;\; r \leq s \;\; \mathrm{trace}\,\Theta \;\; ,$

$\quad \mathrm{inCall}_{p.l\ L} = \mathrm{InCall}_{p.l\ L}(\Delta \;\; r \leq s \;\; \mathrm{trace}\,\Theta$

$\mathrm{Out}_T(\Delta \;\; r \leq s \;\; \mathrm{trace}\,\Theta \;\; = \lambda(\;\; .($

$\quad$ when $ra \leq s$ and $a = \nu(\Theta' \;\; .n\ \mathrm{call}\ p.l(\vec{v} \;\;$ and $\;\; ,\Delta,\Theta,\Delta(r \;\; ,\Theta(r \;\; ,\Theta' \;\; p.l(\vec{v} \;\; U$

$\qquad$ if currentthread $= n$ then

$\qquad\quad \mathrm{ref.val} = \mathrm{new\ State}(\Delta \;\; ra \leq s \;\; \mathrm{trace}\,\Theta \;\; ,$

$\qquad\quad \mathrm{ref.val.inReturn}_U(p.l(\vec{v} \;\; ,$

$\qquad\quad \mathrm{ref.val.out}_T($

$\quad$ when $ra \leq s$ and $a = \nu(\Theta' \;\; .n\ \mathrm{return}\ v$ and $\;\; ,\Delta,\Theta,\Delta(r \;\; ,\Theta(r \;\; ,\Theta' \;\; v \;\; T$

$\qquad$ if currentthread $= n$ then

$\qquad\quad \mathrm{ref.val} = \mathrm{new\ State}(\Delta \;\; ra \leq s \;\; \mathrm{trace}\,\Theta \;\; ,$

$\qquad\quad v$

$\quad$ otherwise

$\qquad$ stop

$\mathrm{InReturn}_T(\Delta \;\; r \leq s \;\; \mathrm{trace}\,\Theta \;\; = \lambda(x \;\; T \;\; .($

$\quad$ when $ra \leq s$ and $a = \nu(\Delta' \;\; .n\ \mathrm{return}\ v?$ and $\;\; ,\Delta,\Theta,\Delta(r \;\; ,\Theta(r \;\; ,\Delta' \;\; v \;\; T$

$\qquad$ if $\Delta,\Theta,\Delta(r \;\; ,\Theta(r \;\; (\mathrm{currentthread}, x \;\; = \nu(\Delta' \;\; .(n, v \;\;$ then

$\qquad\quad \mathrm{ref.val} = \mathrm{new\ State}(\Delta \;\; ra \leq s \;\; \mathrm{trace}\,\Theta \;\; ,$

$\qquad\quad v$

$\quad$ otherwise

$\qquad$ stop

$\mathrm{InCall}_{p.l\ (\vec{T} \;\; \to T}(\Delta \;\; r \leq s \;\; \mathrm{trace}\,\Theta \;\; = \lambda(\vec{x} \;\; \vec{T} \;\; .($

$\quad$ when $ra \leq s$ and $a = \nu(\Delta' \;\; .n\ \mathrm{call}\ p.l(\vec{v} \;\; ?$ and $\;\; ,\Delta,\Theta,\Delta(r \;\; ,\Theta(r \;\; ,\Delta' \;\; \vec{v} \;\; \vec{T}$

$\qquad$ if $\Delta,\Theta,\Delta(r \;\; ,\Theta(r \;\; (\mathrm{currentthread}, \vec{x} \;\; = \nu(\Delta' \;\; .(n, \vec{v} \;\;$ then

$\qquad\quad \mathrm{ref.val} = \mathrm{new\ State}(\Delta \;\; ra \leq s \;\; \mathrm{trace}\,\Theta \;\; ,$

$\qquad\quad \mathrm{ref.val.out}_T($

$\quad$ otherwise

$\qquad$ stop

Figure Definition of $\mathrm{Comp}(\Delta \;\; s \;\; \mathrm{trace}\,\Theta$

$$\text{if } \Delta \square (\quad = \nu(\quad . (\quad \text{then } t \quad = \quad t$$

$$\text{if } \Delta \square (v, \vec{v} \quad = \nu(p \quad U, \vec{n} \quad \vec{T} \quad . (p, \vec{p} \quad \text{then } t \quad = \quad \text{if } v \square \Delta^{-} \quad (U \quad \text{then}$$

$$(\text{if } \Delta, p \quad U \square (\vec{v} \quad = \nu(\vec{n} \quad \vec{T} \quad . (\vec{p} \quad \text{then } t \quad v/p \quad \text{else stop}$$

$$\text{if } \Delta \square (v, \vec{}$$

. Proof of completeness

Theorem . . (Completeness of traces for may testing) *If* $\Delta \models C \sqsubseteq_{may} C$ Θ *then* Traces(Δ C Θ Traces(Δ C Θ .

Proof Choose any trace s Δ Traces (=(((

1. *If* $C \;\wedge\!\!\wedge\; C \;\equiv D \,\square\, E$ *then there exist components such that* $C \;\equiv D \;\square E$ *and* $C \;\equiv D \;\square E$ *with* $D \equiv D \;\wedge\!\!\wedge D$ *and* $E \equiv E \;\wedge\!\!\wedge E$ *.*

2. *If* $C \;\wedge\!\!\wedge C \;\equiv \nu(\vec{n} \; \vec{T} \;.C$ *then there exist components such that* $C \;\equiv \nu(\vec{n} \; \vec{T} \;.C'$ *and* $C \;\equiv \nu(\vec{n} \; \vec{T} \;.C'$ *with* $(\vec{n} \; \vec{T} \; = (\vec{n} \; \vec{T}, \vec{n} \; \vec{T}$ *and* $C' \equiv C' \wedge\!\!\wedge C'$.

Proof rove by .n uct.on on t e er.vat.on o $C \wedge\!\!\wedge C$ $\square$

Lemma A. *If* $C \;\wedge\!\!\wedge C \;\equiv C$ *and* $C \xrightarrow{\beta} C'$b

- **Case** $(\gamma = \nu(\vec{n} \; \vec{T}) . n\,\mathrm{call}\; p.l(\vec{v})$ an $n \in \Sigma$

  ar to the previous case

- **Case** $(\gamma = \nu(\vec{n} \; \vec{T}) . n\,\mathrm{return}\; v)$

  nce $(\Delta, \Phi \vdash C \quad \Theta, \Sigma) \xrightarrow{\gamma?} (\Delta', \Phi \vdash C' \quad \Theta', \Sigma')$ we ust ave at

$$\begin{aligned}
C &\equiv \nu(\vec{p} \; \vec{U}) . (C'' \mid n\,\mathrm{let}\; x \; T = \mathrm{block\;in}\; t) \\
C' &\equiv \nu(\vec{p} \; \vec{U}) . (C'' \mid n\, t \; v/x) \\
\Delta' &= \Delta, \vec{n} \; \vec{T} \\
\Theta' &= \Theta \\
\Sigma' &= \Sigma
\end{aligned}$$

  nce $(\Theta, \Phi \vdash C \quad \Delta, \Sigma) \xrightarrow{\gamma} (\Theta', \Phi \vdash C' \quad \Delta', \Sigma')$ we ust ave at

$$\begin{aligned}
C &\equiv \nu(\vec{n} \; \vec{T}) . \nu(\vec{p} \; \vec{U}) . (C'' \mid n\,\mathrm{let}\; y \; U = \mathrm{return}(v \; T) \;\mathrm{in}\; t) \\
C' &\equiv \nu(\vec{p} \; \vec{U}) . (C'' \mid n\,\mathrm{let}\; y \; U = \mathrm{block\;in}\; t)
\end{aligned}$$

  e then show that

$$C \parallel C \equiv \nu(\vec{n} \; \vec{T}) . \nu(\vec{p} \; \vec{U}) . \nu(\vec{p} \; \vec{U}) . ((C'' \parallel C'') \mid n(\mathrm{let}\; y \; U = \mathrm{block\;in}\; t) \parallel (t \; v/x))$$

  an that

$$C' \parallel C' \equiv \nu(\vec{p} \; \vec{U}) . \nu(\vec{p} \; \vec{U}) . ((C'' \parallel C'') \mid n(\mathrm{let}\; y \; U = \mathrm{block\;in}\; t) \parallel (t \; v/x))$$

  an so

$$\nu(\Delta', \Theta', \Sigma' \setminus \Delta, \Theta, \Sigma) . (C' \parallel C') \equiv C$$

  as require □

Co position o ows by n uction on the er vation o $(\Delta, \Phi \vdash C \quad \Theta, \Sigma) \overset{s}{\Longrightarrow} (\Delta', \Phi \vdash C' \quad \Theta', \Sigma')$ an $(\Theta, \Phi \vdash C \quad \Delta, \Sigma) \overset{s}{\Longrightarrow} (\Theta', \Phi \vdash C' \quad \Delta', \Sigma')$ a n use o Le as A A an A

A. Decomposition

e show three e as ro which Deco position o ows

Lemma A. *For any* $\Delta, \Phi \vdash C \quad \Theta, \Sigma$ *and* $\Theta, \Phi \vdash C \quad \Delta, \Sigma$ *if* $(C \parallel C) \equiv \nu(\vec{n} \; \vec{T}) . (C \mid n\,\mathrm{let}\; x \; T = e \;\mathrm{in}\; t)$ *then either we have:*

$$(\Delta, \Phi \vdash C \quad \Theta, \Sigma) \overset{s}{\Longrightarrow} (\Delta', \Phi \vdash \nu(\vec{n} \; \vec{T}) . (C' \mid n\,\mathrm{let}\; x \; T = e \;\mathrm{in}\; t) \quad \Theta', \Sigma')$$
$$(\Theta, \Phi \vdash C \quad \Delta, \Sigma) \overset{s}{\Longrightarrow} (\Theta', \Phi \vdash C' \quad \Delta', \Sigma')$$

where:

$$\nu(\Delta', \Theta', \Sigma' \setminus \Delta, \Theta, \Sigma) . \nu(\vec{n} \; \vec{T}) . (C' \mid n\, t) \parallel C' \equiv \nu(\vec{n} \; \vec{T}) . (C \mid n\, t)$$

or symmetrically, swapping the roles of C and C .

Proof An in uction on the erivation of

$$(C \parallel C \equiv \nu(\vec{n} \; \vec{T} \; . (C \; n\text{let } x \; T = e \text{ in } t$$

he interesting case is when

$$C \equiv n\text{let } x \; T = \text{block in } t$$
$$C \equiv n\text{let } x \; T = \text{return}(v \; T \; \text{in } t$$

an

$$n t \; v/x \parallel n\text{let } x \; T = \text{block in } t \equiv \nu(\vec{n} \; \vec{T} \; . (C \; n\text{let } x \; T = e \text{ in } t$$

so by e inition of the ts an by in uction we have

$$(\Delta, \Phi \; C \; \Theta, \Sigma \xrightarrow{n \text{return } v?} (\Delta, \Phi \; n t \; v/x \; \Theta, \Sigma$$
$$(\Delta, \Phi \; n t \; v/x \; \Theta, \Sigma \stackrel{s}{\Longrightarrow} (\Delta', \Phi \; \nu(\vec{n} \; \vec{T} \; . (C' \; n\text{let } x \; T = e \text{ in } t \; \Theta', \Sigma'$$

an

$$(\Delta, \Phi \; C \; \Theta, \Sigma \xrightarrow{n \text{return } v} (\Theta, \Phi \; n\text{let } x \; T = \text{block in } t \; \Delta, \Sigma$$
$$(\Theta, \Phi \; n\text{let } x \; T = \text{block in } t \; \Delta, \Sigma \stackrel{s}{\Longrightarrow} (\Theta', \Phi \; C' \; \Delta', \Sigma'$$

where

$$\nu(\Delta', \Theta', \Sigma' \setminus \Delta, \Theta, \Sigma \; . \nu(\vec{n} \; \vec{T} \; . (C' \; n t \parallel C' \equiv \nu(\vec{n} \; \vec{T} \; . (C \; n t$$

or sy etrica y as require □

Lemma A. *If* $C \parallel C \equiv C$ *and* $C -^{\beta 1}$

an so we use t e ax o to et

$$(\Delta, \Phi \; C \quad \Theta, \Sigma \overset{s}{\Longrightarrow} (\Delta', \Phi \; C' \quad \Theta', \Sigma'$$

w ere we e ne

$$C' \equiv \nu(\vec{n} \quad \vec{T}, \vec{n}' \quad \vec{T}' \; .\, (C'' \; E \; n \; \text{let} \; \vec{x} \quad \vec{T} = \vec{e} \text{ in } t$$

- **Case** $(p \, \square \, \mathrm{dom}\,(C^{\square\square}$ $, n \, \square \, \mathrm{dom}\,(C^{\square\square}$

  e ust ave t at

$$C^{\square\square} \equiv \nu(\vec{p} \quad \vec{U} \quad . (C^{\square\square\square} \, \square \, p \; O \; \square \, n \square \mathrm{let}\; y \quad U = \text{block in } t \; \square$$

Moreover s nce C s wr te c ose we ust ave t at t e ax o s

$$p \; O \; \square \, n \square \mathrm{let}\; x \quad T = p.l(\vec{v} \;\; \text{in } t \square \xrightarrow{\tau} p \; O \; \square \, n \square \mathrm{let}\; x \quad T = O.l(p \;\; (\vec{v} \;\; \text{in } t \square$$

n w c case

$$(\Delta, \Phi \, \square \, C \quad \Theta, \Sigma \;\; \xRightarrow{s \, \nu(\vec{n}^{\square} \; \vec{T}^{\square} \; . n \square \mathrm{call}\; p.l(\vec{v} \; \square} \square \;\; (\Delta, \Phi \, \square \, C^{\square} \quad \Theta^{\square}, \vec{n}^{\square} \quad \vec{T}^{\square}, \Sigma^{\square}$$

w ere we e ne

$$C^{\square} \equiv \nu(\vec{n}^{\square\square} \quad \vec{T}^{\square\square} \quad . (C^{\square\square} \, \square \, n \square \mathrm{let}\; x \quad T = \text{block in } t \; \square$$

an we part t on $\{\vec{n} \quad \vec{T}\}$ nto $\{\vec{n}^{\square} \quad \vec{T}^{\square}, \vec{n}^{\square\square} \quad \vec{T}^{\square\square}\}$ suc t at $\{\vec{n}^{\square}\} \, \square \, \mathrm{fn}\,(p.l(\vec{v}$ an $\{\vec{n}^{\square\square}\} \cap \mathrm{fn}\,(p.l(\vec{v} \quad = \emptyset$

  e a so ave

$$(\Delta, \Phi \, \square \, C \quad \Theta, \Sigma \;\; \xRightarrow{s \, \nu(\vec{n}^{\square} \; \vec{T}^{\square} \; . n \square \mathrm{call}\; p.l(}$$

B. $\dagger$ Technical preliminaries

In a co ponent $\nu(\Delta \ .(p\ O \ C$

A *component for* $\Delta ; r \leq s$: trace Θ respecting for $\Delta ; q ; r \leq s$: trace Θ is one of the form:

$$\nu(\Theta(s) \setminus \Theta(q)) . \nu(\mathit{ref} : \mathrm{Ref}) . \nu(\mathrm{state}_{r'} : \mathrm{State}\,|\,\Delta ; r' \leq r : \mathrm{trace}\,\Theta) . ($$
$$\mathrm{ref.val} = \mathrm{state}_r \;|$$
$$\prod\{\mathrm{state}_{r'} : \mathrm{State}(\Delta ; r' \leq s : \mathrm{trace}\,\Theta) \,|\, \Delta ; r' \leq r : \mathrm{trace}\,\Theta\} \;|$$
$$\prod\{p.l_i = \mathrm{ref.val.inCall}_{p.l_i, L_i} \,|\, i = 1 \ldots n\} \;\text{where}\; p : \{l_i : L_i \,|\, i = 1 \ldots n\} \in \Theta, \Theta(s) \;|$$
$$\prod\{n[t_n] \,|\, n : \mathrm{thread} \in \Theta, \Theta(s)\} \;|$$
$$\prod\{n[t_n] \,|\, n : \mathrm{thread} \in \Delta, \Delta(s) \text{ and } n \notin \mathrm{threads}(q)\})$$

where t_n is a thread at n for $\Delta ; r \leq s$: trace Θ respecting for $\Delta ; q ; r \leq s$: trace Θ.

A *thread at* n *for* $\Delta ; r \leq s$: trace Θ is one of the following:

- let $x : T = \mathrm{ref.val.out}_T()$ in t
  where n is output enabled in $\Delta ; r$: trace Θ and t is a return$(x : T)$ thread at n for $\Delta ; r \leq s$: trace Θ.
- let $x : T = \mathrm{block}$ in t
  where n is input enabled in $\Delta ; r$: trace Θ and t is a return$(x : T)$ thread at n for $\Delta ; r \leq s$: trace Θ.

A return$(v : T)$ *thread at* n *for* $\Delta ; r \leq s$: trace Θ is one of the following:

- v
  where n is balanced in r.
- $\mathrm{ref.val.inReturn}_T(v), t$
  where $r = r' a r''$, $a = \nu(\Theta') . n[\mathrm{call}\; p.l(\vec{v})]$, n is balanced in r''
  and t is a thread at n for $\Delta ; r' \leq s$: trace Θ.
- let $y : U = \mathrm{return}(v : T)$ in t
  where $r = r' a r''$, $a = \nu(\Theta') . n[\mathrm{call}\; p.l(\vec{v})]?$, n is balanced in r''
  and t is a return$(y : U)$ thread at n for $\Delta ; r' \leq s$: trace Θ.

Figure: Definition of a component for $\Delta ; r \leq s$: trace Θ and for $\Delta ; q ; r \leq s$: trace Θ

A *thread at n for* $\Delta \,\square\, q \,\square\, r \leq s$ trace Θ is one of the following:

- stop
- a thread at n for $\Delta \,\square\, r \leq s$ trace Θ
  where $\mathrm{proj}\, n\, (q) = \mathrm{proj}\, n\, (r)$
- let $x : T = p.l(\vec{v})$ in t
  where $\mathrm{proj}\, n\, (qa) = \mathrm{proj}\, n\, (r)$, $a = \nu(\Theta').n\,\square\,\mathrm{call}\; p.l(\vec{v})\,\square$ and t is a return$(x : T)$ thread at n for $\Delta \,\square\, r \leq s$ trace Θ'
- let $x : T = \mathrm{return}(v : U)$ in t
  where $\mathrm{proj}\, n\, (qa) = \mathrm{proj}\, n\, (r)$, $a = \nu(\Theta').n\,\square\,\mathrm{return}\; v\,\square$ and t is a return$(x : T)$ thread at n for $\Delta \,\square\, r \leq s$ trace Θ'
- let $y : U = \mathrm{ref.val.inCall}_{p.l\ L}(\vec{v})$ in let $x : T = \mathrm{return}(y : U)$ in t
  where $\mathrm{proj}\, n\, (q) = \mathrm{proj}\, n\, (ra)$, $a = \nu(\Delta').n\,\square\,\mathrm{call}\; p.l(\vec{v})\,\square?$ and t is a return$(x : T)$ thread at n for $\Delta \,\square\, r \leq s$ trace Θ'
- t
  where $\mathrm{proj}\, n\, (q) = \mathrm{proj}\, n\, (ra)$, $a = \nu(\Delta').n\,\square\,\mathrm{return}\; v\,\square?$ and t is a return$(y : T)$ thread at n for $\Delta \,\square\, r \leq s$ trace Θ, for some T
- $\mathrm{ref.val} = \mathrm{new\ State}(\Delta \,\square\, ra \leq s$ trace $\Theta)$, t
  where $\mathrm{proj}\, n\, (q) = \mathrm{proj}\, n\, (ra)$ and t is a thread at n for $\Delta \,\square\, ra \leq s$ trace Θ
- t
  where $n\,\square\, t\,\square \xrightarrow{\beta} n\,\square\, t'\,\square$ and t' is a thread at n for $\Delta \,\square\, q \,\square\, r \leq s$ trace Θ

Figure: Definition of a thread for $\Delta \,\square\, q \,\square\, r \leq s$ trace Θ

Proof. An inspection of the definition of $\mathrm{Comp}(\Delta \vdash s : \mathrm{trace}\ \Theta)$. □

Lemma B. *If* $\Delta \vdash ra \lesssim s : \mathrm{trace}\ \Theta$ *and* $(\Delta' \vdash C : \Theta')$ *is a component for* $\Delta \vdash r \lesssim s : \mathrm{trace}\ \Theta$ *then* $(\Delta' \vdash C : \Theta') \stackrel{a}{\Longrightarrow} (\Delta'' \vdash C' : \Theta'')$ *where* C' *is a component for* $\Delta \vdash ra \lesssim s : \mathrm{trace}\ \Theta$.

Proof. By considering the definition of $\Delta \vdash r : \mathrm{trace}\ \Theta$ we see that the following cases are exhaustive.

- **Case** $a = \nu(\Theta''').n.\mathrm{return}\ v$ and $C \equiv \nu(\Theta''').(C \mid \mathrm{ref}.\mathrm{val} = \mathrm{state}_r \mid n.\mathrm{let}\ y : U = \mathrm{ref}.\mathrm{val}.\mathrm{out}_U()\ \mathrm{in}\ \mathrm{let}\ x : T = \mathrm{return}(y : U)\ \mathrm{in}\ t)$.

  We have

  $$\begin{array}{ll} & (\Delta' \vdash C : \Theta') \\ \stackrel{\tau}{\dashrightarrow} & (\Delta' \vdash \nu(\Theta''').(C \mid \mathrm{ref}.\mathrm{val} = \mathrm{state}_r \mid \\ & \quad n.\mathrm{let}\ y : U = \mathrm{state}_r.\mathrm{out}_U()\ \mathrm{in}\ \mathrm{let}\ x : T = \mathrm{return}(y : U)\ \mathrm{in}\ t) : \Theta') \\ \stackrel{\beta}{\dashrightarrow}{}^{*} & (\Delta' \vdash \nu(\Theta''').(C \mid \mathrm{ref}.\mathrm{val} = \mathrm{state}_r \mid \\ & \quad n.(\mathrm{ref}.\mathrm{val} = \mathrm{new}\ \mathrm{State}(\Delta \vdash ra \lesssim s : \mathrm{trace}\ \Theta), \mathrm{let}\ y : U = v\ \mathrm{in}\ \mathrm{let}\ x : T = \mathrm{return}(y : U)\ \mathrm{in}\ t)) : \Theta') \\ \stackrel{\tau}{\dashrightarrow} & (\Delta' \vdash \nu(\Theta''', \mathrm{state}_{ra} : \mathrm{State}).(C \mid \mathrm{ref}.\mathrm{val} = \mathrm{state}_{ra} \mid \mathrm{state}_{ra} = \mathrm{State}(\Delta \vdash ra \lesssim s : \mathrm{trace}\ \Theta) \mid \\ & \quad n.\mathrm{let}\ y : U = v\ \mathrm{in}\ \mathrm{let}\ x : T = \mathrm{return}(y : U)\ \mathrm{in}\ t) : \Theta') \\ \stackrel{\beta}{\dashrightarrow}{}^{*} & (\Delta' \vdash \nu(\Theta''', \mathrm{state}_{ra} : \mathrm{State}).(C \mid \mathrm{ref}.\mathrm{val} = \mathrm{state}_{ra} \mid \mathrm{state}_{ra} = \mathrm{State}(\Delta \vdash ra \lesssim s : \mathrm{trace}\ \Theta) \mid \\ & \quad n.\mathrm{let}\ x : T = \mathrm{return}(v : U)\ \mathrm{in}\ t) : \Theta') \\ \stackrel{a}{\dashrightarrow} & (\Delta' \vdash \nu(\mathrm{state}_{ra} : \mathrm{State}).(C \mid \mathrm{ref}.\mathrm{val} = \mathrm{state}_{ra} \mid \mathrm{state}_{ra} = \mathrm{State}(\Delta \vdash ra \lesssim s : \mathrm{trace}\ \Theta) \mid \\ & \quad n.\mathrm{let}\ x : T = \mathrm{block}\ \mathrm{in}\ t) : \Theta', \Theta''') \end{array}$$

  which is a component for $\Delta \vdash ra \lesssim s : \mathrm{trace}\ \Theta$ as required.

- **Case** $a = \nu(\Theta''').n.\mathrm{call}\ p.l(\vec{v})$ and $C \equiv \nu(\Theta''').(C \mid \mathrm{ref}.\mathrm{val} = \mathrm{state}_r \mid n.\mathrm{let}\ y : U = \mathrm{ref}.\mathrm{val}.\mathrm{out}_U()\ \mathrm{in}\ t)$.

  We have

  $$\begin{array}{ll} & (\Delta' \vdash C : \Theta') \\ \stackrel{\tau}{\dashrightarrow} & (\Delta' \vdash \nu(\Theta''').(C \mid \mathrm{ref}.\mathrm{val} = \mathrm{state}_r \mid \\ & \quad n.\mathrm{let}\ y : U = \mathrm{state}_r.\mathrm{out}_U()\ \mathrm{in}\ t) : \Theta') \\ \stackrel{\beta}{\dashrightarrow}{}^{*} & (\Delta' \vdash \nu(\Theta''').(C \mid \mathrm{ref}.\mathrm{val} = \mathrm{state}_r \mid \\ & \quad n.(\mathrm{ref}.\mathrm{val} = \mathrm{new}\ \mathrm{State}(\Delta \vdash ra \lesssim s : \mathrm{trace}\ \Theta), \\ & \qquad \mathrm{let}\ x : T = p.l(\vec{v})\ \mathrm{in}\ \mathrm{ref}.\mathrm{val}.\mathrm{inReturn}_T(x), \mathrm{let}\ y : U = \mathrm{ref}.\mathrm{val}.\mathrm{out}_U()\ \mathrm{in}\ t)) : \Theta') \\ \stackrel{\tau}{\dashrightarrow} & (\Delta' \vdash \nu(\Theta''', \mathrm{state}_{ra} : \mathrm{State}).(C \mid \mathrm{ref}.\mathrm{val} = \mathrm{state}_{ra} \mid \mathrm{state}_{ra} = \mathrm{State}(\Delta \vdash ra \lesssim s : \mathrm{trace}\ \Theta) \mid \\ & \quad n.\mathrm{let}\ x : T = p.l(\vec{v})\ \mathrm{in}\ \mathrm{ref}.\mathrm{val}.\mathrm{inReturn}_T(x), \mathrm{let}\ y : U = \mathrm{ref}.\mathrm{val}.\mathrm{out}_U()\ \mathrm{in}\ t) : \Theta') \\ \stackrel{a}{\dashrightarrow} & (\Delta' \vdash \nu(\mathrm{state}_{ra} : \mathrm{State}).(C \mid \mathrm{ref}.\mathrm{val} = \mathrm{state}_{ra} \mid \mathrm{state}_{ra} = \mathrm{State}(\Delta \vdash ra \lesssim s : \mathrm{trace}\ \Theta) \mid \\ & \quad n.\mathrm{let}\ x : T = \mathrm{block}\ \mathrm{in}\ \mathrm{ref}.\mathrm{val}.\mathrm{inReturn}_T(x), \mathrm{let}\ y : U = \mathrm{ref}.\mathrm{val}.\mathrm{out}_U()\ \mathrm{in}\ t) : \Theta', \Theta''') \end{array}$$

  which is a component for $\Delta \vdash ra \lesssim s : \mathrm{trace}\ \Theta$ as required.

- **Case** $a = \nu(\Delta''').n.\mathrm{return}\ v?$ and $C \equiv (C \mid \mathrm{ref}.\mathrm{val} = \mathrm{state}_r \mid n.\mathrm{let}\ x : T = \mathrm{block}\ \mathrm{in}\ \mathrm{ref}.\mathrm{val}.\mathrm{inReturn}_T(x), t)$

e ave

$(\Delta \vdash C \quad \Theta)$

$\xrightarrow{a} (\Delta, \Delta''' \vdash C$ ref val $=$ state$_r$ $\|$
n let x $T = v$ in ref.val.inReturn$_T(x$ $, t)$ $\Theta)$

$\xrightarrow{\beta}$ $(\Delta, \Delta''' \vdash C$ ref val $=$ state$_r$ $\|$
n ref.val.inReturn$_T(v$ $, t)$ $\Theta)$

$\xrightarrow{\tau} (\Delta, \Delta''' \vdash C$ ref val $=$ state$_r$ $\|$
n state$_r$.inReturn$_T(v$ $, t)$ $\Theta)$

$\xrightarrow{\beta}$ $(\Delta, \Delta''' \vdash C$ ref val $=$ state$_r$ $\|$
n ref.val $=$ new State$(\Delta \vdash ra \leq s$ trace Θ $, t)$ $\Theta)$

$\xrightarrow{\tau} (\Delta, \Delta''' \vdash C\ \nu($state$_{ra}$ State .ref val $=$ state$_{ra}$ $\|$ state$_{ra}$ State$(\Delta \vdash ra \leq s$ trace Θ $\|$
$n\ t)$ $\Theta)$

w c s a co ponent or $\Delta \vdash ra \leq s$ trace Θ as requ re

Case $a = \nu(\Delta''' . n$ call $p.l(\vec{v})$? an $C \equiv C$ ref val $=$ state$_r$ $\|$ n let x $T =$ block in t

e ave

$(\Delta \vdash C \quad \Theta)$

$\xrightarrow{a} (\Delta, \Delta''' \vdash C$ ref val $=$ state$_r$ $\|$
n let y $U = p.l(\vec{v}$ in let x $T =$ return$(y$ U in $t)$ $\Theta)$

$\xrightarrow{\beta}$ $(\Delta, \Delta''' \vdash C$ ref val $=$ state$_r$ $\|$
n let y $U =$ ref.val.inCall$_{p.l\ L}(\vec{v}$ in let x $T =$ return$(y$ U in $t)$ $\Theta)$

$\xrightarrow{\tau} (\Delta, \Delta''' \vdash C$ ref val $=$ state$_r$ $\|$
n let y $U =$ state$_r$.inCall$_{p.l\ L}(\vec{v}$ in let x $T =$ return$(y$ U in $t)$ $\Theta)$

$\xrightarrow{\beta}$ $(\Delta, \Delta''' \vdash C$ ref val $=$ state$_r$ $\|$
n ref.val $=$ new State$(\Delta \vdash ra \leq s$ trace Θ ,
let y $U =$ ref.val.out

for $\Delta \vdash q \sqcap r \leq s$ trace Θ in Figures and with the intended meaning that a component for $\Delta \vdash q \sqcap r \leq s$ trace Θ has performed the trace q and this is related to some prefix of s. Note that as prefix ordering $\leq$ on traces is contained in $\sqsubseteq$

- **Case** $C \equiv C$ ref val $=$ state$_r$ n ref val $=$ new State(Δ $ra \leq s$ trace Θ $, t$

$$\xrightarrow{\tau} \nu(\text{state}^{T}_{ra}\ \text{State}\ .C\ \text{ref val} = \text{state}_{ra}\ \text{state}_{ra}\ \text{State}(\Delta\ ra \leq s\ \text{trace}\ \Theta\ \ n\ t \equiv C'$$

w ere t s a t rea at n or Δ $ra \leq s$ trace Θ

By e n t on C' s a co ponent or Δ q $ra \leq s$ trace Θ

- **Case** $C \equiv C$ n let x $T =$ ref.val.out$_T$(in t $\xrightarrow{\tau}$ C n let x $T =$ state$_r$.out$_T$(in t $\equiv C'$
w ere proj n (q $=$ proj n (r n s output enab e n Δ r trace Θ an t s a return (x T t rea at n or Δ $r \leq s$ trace Θ

I Δ $ra \leq s$ trace Θ an $a = \nu(\Theta'$ $.n$ call $p.l(\vec{v}$ t en

$$C' \xrightarrow{\beta} \ C\ n\ \text{ref val} = \text{new State}(\Delta\ ra \leq s\ \text{trace}\ \Theta\ ,$$
$$\text{ref.val.inReturn}_U(p.l(\vec{v}\ ,\text{let}\ x\ T = \text{ref.val.out}_T(\ \text{in}\ t$$

w c s a co ponent or Δ q $r \leq s$ trace Θ as requ re

I Δ $ra \leq s$ trace Θ an $a = \nu(\Theta'$ $.n$ return v t en we ust ave t at $r = r$ $\nu(\Theta'$ $.$ n call $p.l(\vec{v}$? r w ere n s ba ance n r

- **Case** $(\Delta' \square C\ n\square \text{let } x\ \ T = \text{block in } t\square\ \ \Theta' \xrightarrow{\nu(\Delta''' \ .n\square\text{call } p.l(\vec{v}\ \square ?} (\Delta', \Delta''' \square C\ n\square \text{let } y\ \ U = p.l(\vec{v}\ \text{ in let } x\ \ T = \text{return}(y\ \ U\ \text{ in } t\square\ \ \Theta'$
w ere proj n $(q\ = \text{proj } n\ (r\ \ n$ s nput enab e n $\Delta \square r$ trace Θ an t s a return$(x\ T$ t rea at n or $\Delta \square r \leq s$ trace Θ
e ave

$$C' \xrightarrow{\beta} \square\ C\ n\square \text{let } y\ \ U = \text{ref.val.inCall}_{p.l\ L}(\vec{v}\ \text{ in let } x\ \ T = \text{return}(y\ \ U\ \text{ in } t\square$$

w c s a co ponent or $\Delta \square qa \square r \leq s$ trace Θ as requ re

- **Case** $(\Delta' \square C\ n\square \text{let } x\ \ T = \text{block in } t\square\ \ \Theta' \xrightarrow{\nu(\Delta''' \ .n\square\text{return } v\square ?} (\Delta', \Delta''' \square C\ n\square \text{let } x\ \ T = v \text{ in } t\square\ \ \Theta'$
w ere proj n $(q\ = \text{proj } n\ (r\ \ n$ s nput enab e n $\Delta \square r$ trace Θ an t s a return$(x\ T$ t rea at n or $\Delta \square r \leq s$ trace Θ
e ave

$$C' \xrightarrow{\beta} \square\ C\ n\square t\ v/x\ \square$$

w c s a co ponent or $\Delta \square qa \square r \leq s$ trace Θ as requ re

- **Case** $(\Delta' \square \nu(\Theta''' \ .C\ n\square \text{let } x\ \ T = p.l(\vec{v}\ \text{ in } t\square\ \ \Theta' \xrightarrow{\nu(\Theta''' \ .n\square\text{call } p.l(\vec{v}\ \square} (\Delta' \square C\ n\square \text{let } x\ \ T = \text{block in } t\square\ \ \Theta', \Theta'''$
w ere proj n $(qa\ = \text{proj } n\ (r$ an t s a return$(x\ T$ t rea at n or $\Delta \square r \leq s$ trace Θ
e ave C' s a co ponent or $\Delta \square qa \square r \leq s$ trace Θ as requ re

- **Case** $(\Delta' \square \nu(\Theta''' \ .C\ n\square \text{let } x\ \ T = \text{return}(v\ \ U\ \text{ in } t\square\ \ \Theta' \xrightarrow{\nu(\Theta''' \ .n\square\text{return } v\square} (\Delta' \square C\ n\square \text{let } x\ \ T = \text{block in } t\square\ \ \Theta', \Theta'''$
w ere proj n $(qa\ = \text{proj } n\ (r$ an t s a return$(x\ T$ t rea at n or $\Delta \square r \leq s$ trace Θ
e ave C' s a co ponent or $\Delta \square qa \square r \leq s$ trace Θ as requ re □

e on y a o e nab ty now o ows by n uct on on Le as B B an B w t Le a B as t e base case a n appropr ate use o Coro ary B

References

M Aba an L Car e *A Theory Of Objcets* pr n er er a

Abra s y Ja a eesan an r a t Fu t J absn tract oe nn orn

M. ner Fu y abstract se ant.cs o type λ ca cu . *Theoret. Comput. Sci.*

M. ner *Communicating and Mobile Systems* Ca br. e n.vers.ty ress

M. ner J arrow an D a er A ca cu us o ob. e processes *Inform. and Comput.*

M. ner an D an .or . Barbe b.s. u at.on In *Proc. Int. Colloq. Automata, Languages and Programming* vo u e o *Lecture Notes in Computer Science* pr.n er er a

J H Morr.s La b a ca cu us o e s o pro ra .n an ua es D.ssertat.on M I

B .erce an D an .or . yp.n an subtyp.n or ob. e processes *Mathematical Structures in Computer Science*

A M .tts an I D B tar bservab e propert.es o . er or er unct.ons t at yna .ca y create oca na es or at s new? In *Proc. MFCS 93* pa es pr.n er er a L C

G ot .n LCF cons. ere as a pro ra .n an ua e *Theoret. Comput. Sci.*