A Fu y Abstract May estn e antcs or Concurrent b ects

A an Je rey
C I De au nvers.ty
C ca o IL A
a e rey cs epau e u

Ju an at e
C G nvers.ty o ussex
Br ton K
u anr co s susx ac u

ctober

Abstract

s paper prov es a u y abstract se antcs or a varant o t e concurrent ob ect ca cu us e e ne ay testn or concurrent ob ect co ponents an t en c aracterse t usn a trace se antcs nspre by ML nteracton a ra s e a n resu t o t s paper s to s ow t at t e trace se antcs s u y abstract or ay testn 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 atn eatures o ob ect an ua es suc as encapsu ate state subtypn 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 operatona be avour 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 antcs 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 antcs or concurrent utab e ob ects

In t s paper we present t e rst u y abstract testn se antcs or a varant o Gor on an Han n s concurrent ob ect ca cu us w t out subtypn e ac o subtypn 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 subtypn a so s se antcs 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

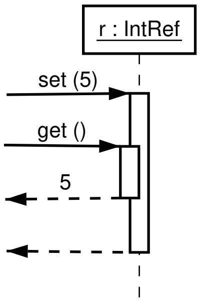
Interaction diagrams

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 nvers 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 inco in or out oin essa e ca s or atc in out oin or inco in returns

- Messa es are ecorate wit t rea i enti ers

- Messa es ay inc u e res na es

e ave on y use a very s a subset o sequence ia ra s w ic in turn is a very s a subset o ML but in t is paper we wi s ow t at t is s a subset is very expressive an in particu ar provi es a u y abstract se antics

. The object calculus

e ob ect ca cu us is a ini a an ua e or o e in ob ect base pro ra in Aba i an Car e i provi e a type syste an operationa se antics or a variety o ob ect ca cu i an prove type sa ety or t e Gor on an Han in ave since exten e t is an ua e to inc u e concurrent eatures

In t is paper we s a investi ate a variant o Gor on an Han in s concurrent ob ect ca cu us w ic inc 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 ity can create new ob ects an t rea s an can iscover t eir own t rea na e

- An operationa se antics base on t e π ca cu us an a si p e type syste

- A trace se antics as iscusse in ection

e are not consi erin 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 recursive types ob ect c onin an ob ect oc in is is ust or si p icity an we o not see any tec nica prob e s wit incorporatin t ese eatures into our an ua e

In anot er stran o researc Di B asio an Fis er a so e ne a ca cu us or o e in i perative concurrent ob ect base syste s As wit Aba i an Car e i s ob ect ca cu us an its various extensions t e e p asis in Di B asio an Fis er s wor is a ain on type syste s an sa ety properties or t e

. Full abstraction

e prob e o u abstraction was rst intro uce by Mi ner an investi ate in ept by ot in Fu abstraction was rst propose or variants o t e λ ca cu us but as since been investi ate or process a ebras t e π ca cu us t e ν ca cu us Concurrent ML an t e i utab e ob ect ca cu us

We can then define the *may testing preorder* as $C \sqsubseteq_{\mathrm{may}} C$ whenever

for any appropriately typed C:
if $C \parallel C$ is successful then $C \parallel C$ is successful

Unfortunately, although it is very simple to define and is quite intuitive, may testing is often very difficult to reason about directly because of the quantification over any appropriately typed C. In practice, we require a proof technique which we can use to show results about may testing.

One approach is to use a *trace* semantics given by defining possible executions of components $C \xRightarrow{s} C'$ where s is a sequence of messages. We then write Traces(C) for the set of all traces of C. We say that:

- Traces are *sound* for may testing when Traces$(C) \subseteq$ Traces(C) implies $C \sqsubseteq_{\mathrm{may}} C$.
- Traces are *complete* for may testing when $C \sqsubseteq_{\mathrm{may}} C$ implies Traces$(C) \subseteq$ Traces(C).
- Traces are *fully absdwhereTj11.993tTJ/R1.991Tf1.19Tdfor-2may-22.9test-21.9ing-2.whenTJsi.9911.so/R11.9*

Co ponents	C	$=$	$\,.\,\|\,C$ ▯ $C\,\|\,\nu(n\ \ T\ \ .C\,\|\,n\ O\ \ \|\,n$▯$t$▯
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	$=$	$v\,\|$ stop $\|$ let $x\ \ T = e$ 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 operat ona se ant cs uc s p er but we can treat any express on as a t rea by η convert n t e $\equiv$ let $x = e$ in x

For t e re a n er o t s sect on we w prov e an n or a escr pt on o t e syntax

A co ponent C s a co ect on o na e ob ects n O an t rea s n t For exa p e one poss b e co ponent cons st n o an nte er re erence p an a t rea n w c ncre ents t e re erence s

$$p \text{ contents} =$$
$$n \text{ let } x = p.\text{contents in } p.\text{contents} = x$$

e a so use t e ν

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 judgement is $\Delta \vdash C : \Theta$ which is read 'as the component C uses names Δ and defines names Θ'. For example, if we define $C_1(v)$, C_2 and IntRef as:

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

$$\frac{}{\Delta \,\square\, , \, (} \qquad \frac{, \Delta, n \;\; T \,\square\, O \;\; T}{\Delta \,\square\, n \, O \;\; (n \;\; T} \qquad \frac{, \Delta, n \;\; \text{thread} \,\square\, t \;\; \text{none}}{\Delta \,\square\, n \square t \square \;\; (n \;\; \text{thread}}$$

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

F ure u es or u e ent $\Delta \,\square C \;\; \Theta$

$$\frac{\Gamma, \Delta \,\square M \;\; T.l \qquad \cdots \qquad \Gamma, \Delta \,\square M_k \;\; T.l_k}{\Gamma, \Delta \,\square\; l \;\; = M \;\; , \ldots , l_k = M_k \;\; T}$$

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

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

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

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

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

$$\frac{\Gamma, \Delta \,\square O \;\; T}{\Gamma, \Delta \,\square\, \text{new } O \;\; T} \qquad \frac{\Gamma, \Delta \,\square t \;\; T}{\Gamma, \Delta \,\square\, \text{new} \square t \square \;\; \text{thread}} \qquad \frac{}{\Gamma, \Delta \,\square\, \text{currentthread} \;\; \text{thread}}$$

$$\frac{\Gamma, \Delta \,\square e \;\; T \qquad \Gamma, x \;\; T \;\; , \Delta \,\square t \;\; T}{\Gamma, \Delta \,\square\, \text{let } x \;\; T \;\; = e \text{ in } t \;\; T} \qquad \frac{}{\Gamma, \Delta \,\square\, \text{stop} \;\; T} \qquad \frac{}{\Gamma, x \;\; T, \Gamma', \Delta \,\square x \;\; T} \qquad \frac{}{\Gamma, \Delta, n \;\; T, \Delta' \,\square n \;\; T}$$

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

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

Whenever Δ ⊢ C : Θ contains a subexpression of [redacted] n appears in Θ.

This is intended to capture the common software engineering [redacted] should not export mutable fields; instead they should export su[redacted] the configurations C and C above are write close since th[redacted]own in component which writes directly to p.contents is not w[redacted]

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

For the remainder of the paper we will require compo[redacted] the opin a fully abstract semantics much simpler since [redacted] directly.

Dynamic semantics

The dynamic semantics for our concurrent object calcu[redacted] We define three relations between components.

- ≡ structural congruence represents the least con[redacted] axioms in Figure [redacted]

t

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

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

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

n

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

n let $x \quad T = ($let $x \quad T = e$ in e in $t \quad \xrightarrow{\beta} \quad n$ let $x \quad T = e$ in (let $x \quad T = e$ in t

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

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

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

let $x \quad T =$

. Testing preorder

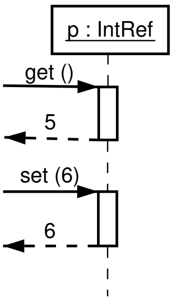
e w now e ne t e test n se ant cs or our concurrent ob ect ca cu us e w o t s by e n n a not on o *barb*

$(\Delta, n$ thread □ C

then where C (v is the one in section we have

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

which corresponds to the interaction diagram



For any component $(\Delta □ C \quad \Theta$ we define its traces to be

$$\mathrm{Traces}(\Delta □ C \quad \Theta \;=\; \{ s \mid (\Delta □ C \quad \Theta \overset{s}{\Longrightarrow}□ \; (\Delta^{□} □ C^{□}$$

the base an ua e which wou have been reache a the co ponent an test actua y interacte . his operation o er in is e ne be~~ow~~

. The merge operator

De ne the partia *merge* operator $C \curlywedge C$ on co ponents as the sy etric operator e ne up to $\equiv$ w~~here~~

$$
\begin{array}{rcl}
\curlywedge C & = & C \\
(\nu(p\ \ T\ .C\ \ \curlywedge C & = & \nu(p\ \ T\ .(C\ \curlywedge C \\
(p\ O\ \ \ C\ \ \curlywedge C & = & p\ O\ \ (C\ \curlywedge C \\
(p t\ \ C\ \ \curlywedge C & = & p t\ \ (C\ \curlywedge C \\
(n t\ \ \ C\ \ \curlywedge (n t\ \ \ C & = & n t\ \ \curlywedge t\ \ \ (C\ \curlywedge C
\end{array}
$$

when n dom $(C\ ,C$ an p fn $(C$.

e over oa notation an e ne the partia er e operator $t\ \curlywedge t$ on threa s as the sy etric operator w~~here~~

$$
\begin{array}{rcl}
(\text{l~~et~~}\ x\ \ T = \text{block in}\ t\ \ \curlywedge \text{stop} & = & \text{stop} \\
(\text{l~~et~~}\ x\ \ T = \text{block in}\ t\ \ \curlywedge (\text{l~~et~~}\ y\ \ U = \text{return}(v\ \ T\ \ \text{in}\ t & = & (\text{l~~et~~}\ y\ \ U = \text{block in}\ t\ \ \curlywedge (t\ \ v/x \\
(\text{l~~et~~}\ x\ \ T = \text{block in}\ t\ \ \curlywedge (\text{l~~et~~}\ y\ \ U = e\ \text{in}\ t & = & \text{l~~et~~}\ y\ \ U = e\ \text{in}\ ((\text{l~~et~~}\ x\ \ T = \text{block in}\ t\ \ \curlywedge t
\end{array}
$$

when e is b oc . return ree an y fv $(t$.

Lemma . *If* Δ $(C$ ~~C~~ Θ *then* $(C\ \curlywedge C\ \ \equiv (C\ \ C$.

Proof An in uction on the e nition o $C\ \curlywedge C$. □

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

Proof An in uction on the e nition o $C\ \curlywedge C$. □

. Trace composition and decomposition

Given a trace s we write s or the 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* $(\Delta,\Phi \Box C\ \Theta,\Sigma$ *and* $(\Theta,\Phi \Box C\ \Delta,\Sigma$ *such that* $C\ /\!\!\backslash\ C\ \equiv C$ *and* $C\Box_b$ *then there exists some trace s such that* $(\Delta,\Phi \Box C\ \Theta,\Sigma \overset{s}{\Longrightarrow}\Box (\Delta',\Phi \Box C'\ \Theta',\Sigma'$ *and* $(\Theta,\Phi \Box C\ \Delta,\Sigma \overset{s}{\Longrightarrow}\Box (\Theta',\Phi \Box C'\ \Delta',\Sigma'$ *where either* $C'\downarrow_b$ *or* $C'\downarrow_b$*.*

Proof e now t at $C\Box_b$ w .c te s us t at $C \Box C''$ 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

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

w ere $\nu(\Delta'',\Theta'',\Sigma'' \setminus \Delta,\Theta,\Sigma\ .(C''\ /\!\!\backslash\ C''\ \equiv C''$. G.ven t at $C''\downarrow_b$ we now t at $(C''\ /\!\!\backslash\ C''\ \downarrow_b$ a so By t e e n.t.on o $/\!\!\backslash$ 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
- $C'' \equiv \nu(\Delta\ .(n\Box t\ \Box\ \Box C'''$ an $C'' \equiv \nu(\Delta\ .(n\Box t\ \Box\ \Box C'''$ w ere $n\Box t\ /\!\!\backslash\ t\ \Box\downarrow_b$ e now procee by .n uct.on on t e e n.t.on o $t\ /\!\!\backslash\ t$ to s ow t at or a suc C'' an C'' we can n s w ere

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

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

  - 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

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

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

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

so by .n uct.ve ypot es.s

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

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

. Proof of soundness

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

Proof uppose that Traces$(\Delta$ □C Θ □ Traces$(\Delta$ □C Θ an that we have $(\Theta, b$ barb □$C_{\bullet}$ Δ 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

$$(\Delta, b \quad \text{barb} \; \square C \quad \Theta \overset{s}{\Longrightarrow} \square \; (\Delta', b \quad \text{barb} \; \square C' \quad \Theta', \Sigma'$$

$$(\Theta, b \quad \text{barb} \; \square C_{\bullet} \quad \Delta \overset{s}{\Longrightarrow} \square \; (\Theta', b \quad \text{barb} \; \square C'_{\bullet} \quad \Delta', \Sigma'$$

$$\overline{\Delta \square \varepsilon \;\; \text{trace}\ \Theta}$$

n

- If n □ threads $(s$ then n is balanced in s
- If n is balanced in s and s then n is balanced in s s
- If n is balanced in s then n is balanced in $\nu(\Delta\ .n$ call $p.l(\vec{n}$ $s\,\nu(\Theta\ .n$ return v
- If n is balanced in s then n is balanced in $\nu(\Theta\ .n$ call $p.l(\vec{n}$ $s\,\nu(\Delta\ .n$ return v

Define pop $n(s)$ as

- If n is balanced in s then pop $n(s) =$
- If n is balanced in s and $a =$

Proof Easy induction on s. □

Lemma .

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

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

Proof We prove these properties simultaneously by an induction on the length of s. We only show the argument for part as part can be shown in a similar manner. By analysis of the rules of the ts we have

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

Now partition s into s s pick in s

Case $s = s'\nu(\Delta').n\langle\mathsf{call}\,p.l(\vec{v})\rangle?$. We know that

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

so we have that either

$$C' \equiv \nu(\Delta').\nu(\Delta'').n\langle\mathsf{let}\ x : T = \mathsf{block\ in}\ t\rangle \mid C''$$

or $n \notin \Delta,\Delta(s')$ and n is a fresh thread to s'. We can apply the inductive hypothesis to s' to see that $\Delta \vdash s'$ trace Θ and we consider $\mathrm{pop}_n(s')$. If $n \notin \Delta,\Delta(s')$ and n is fresh thread to s' then $\mathrm{pop}_n(s')$ is necessarily ε. Otherwise we know that $C' \equiv \nu(\Delta').\nu(\Delta'').n\langle\mathsf{let}\ x : T = \mathsf{block\ in}\ t\rangle \mid C''$ and therefore the last action which could have occurred at n must have been an output, that is $\mathrm{pop}_n(s') = \gamma!$. In both cases we see that

n is input enabled in $\Delta \vdash s'$ trace Θ

We know that $(\Delta,\Delta(s') \vdash C' \vdash \Theta,\Theta(s')) \xrightarrow{\nu(\Delta').n\langle\mathsf{call}\,p.l(\vec{v})\rangle?}$ and we know that the side conditions on the transition rule for $\nu(\Delta').\gamma?$ actions guarantees that

$\mathrm{dom}(\Delta') \subseteq \mathrm{fn}(\vec{v})$

We also know that the side conditions on rule for call input actions guarantees that

$$;\Delta,\Delta(s');\Theta,\Theta(s');\Delta' \vdash p.l(\vec{v}) : T \text{ and } p \in \Theta,\Theta(s')$$

We use this to see that

$;\Theta,\Theta(s');\Delta' \vdash p : [\ldots l : (\vec{T}) \to T \ldots]$

and

$;\Delta,\Delta(s');\Theta,\Theta(s');\Delta' \vdash \vec{v} : \vec{T}$

Lastly it is easy to see that

$;\Delta,\Delta(s');\Theta,\Theta(s');\Delta' \vdash n : \mathsf{thread}$

We collect the state ments to gether to see that they form the hypotheses of the type rule which allows us to conclude

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

as required.

Case $s = s'\nu(\Theta').n\langle\mathsf{call}\,p.l(\vec{v})\rangle!$: Similar to previous case.

Case $s = s'\nu(\Theta').n\langle\mathsf{return}\,v\rangle!$: We know that

$$(\Delta \vdash C$$

$\Delta \; \square s'$ trace Θ

an we not.ce t at because C .s b oc return ree we can app y Le a to et

$$s' = s \;\; \nu(\Delta' \;\; . n\square \mathrm{call}\, p.l(\vec{v} \;\square ? \; s$$

w ere n .s ba ance .n s G.ven t .s we see t at

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

ence

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

A a.n t e s. e con .t.ons on t e trans.t.on ru e or $\nu(\Theta' \;\; . \gamma$ uarantee t at

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

e a so now by an t e act t at pre xes o we type traces are a so we type t at

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

an we see t at t .s ust ave been .n erre us.n a ypot es.s

$$,\Theta,\Theta(s \;\; \square p \;\; \dots l \;\; (\vec{U} \;\; \to U' \dots$$

w .c by wea en.n .ves us

$$,\Theta,\Theta(s' \;\; \square p \;\; \dots l \;\; (\vec{U} \;\; \to U' \dots$$

Last y because

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

an

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

we see t at

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

o by Le a to et er w.t t e typ.n s. e con .t.ons or ca .nput trans.t.ons we ave t at $U = U'$ an so

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

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

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

as requ.re

Case $s = s' \;\nu(\Delta' \;\; . n\square \mathrm{return}\, v\square ?$ . . ar to prev.ous case □

. Information order on traces

e n or at on preor er on traces $\Delta \,\square\, r \,\square\, s$ trace Θ s enerate by ax o s w ere n eac case we requ re bot s es o t e nequat on to be we type traces

$$\Delta \,\square\, s \,\square\, sr \quad \text{trace } \Theta$$
$$\Delta \,\square\, s\gamma ? \,\square\, s \quad \text{trace } \Theta$$
$$\Delta \,\square\, s\gamma \; ?\gamma \quad r \,\square\, s\gamma \quad \gamma \; ?r \quad \text{trace } \Theta$$
$$\Delta \,\square\, s\nu(\Delta \quad .\gamma \; ?\gamma \; ?r \,\square\, s\nu(\Delta \quad .\gamma \; ?\gamma \; ?r \quad \text{trace } \Theta$$
$$\Delta \,\square\, s\nu(\Theta \quad .\gamma \quad \gamma \quad r \,\square\, s\nu(\Theta \quad .\gamma \quad \gamma \quad r \quad \text{trace } \Theta$$

Lemma . **(Information Order Duality)** *If* $\Delta \,\square\, r\gamma \;\; \square\, s\gamma$ trace Θ *and* fn $(\gamma \;\cap \Theta(r \;\; = \emptyset$ *and* $\gamma \;\; \square\, s, r$ *then* $\Theta \,\square\, s \,\square\, r$ trace Δ.

Proof e wr te $\Delta \,\square\, r \,\square^{n}\, s$ trace Θ $\Delta \,\square\, r \,\square\, s$ trace Θ can be er ve us n n nstances o

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

Proof ow t at t e o own a ra s can be co p ete w en thread $(\gamma \; \boxminus$ thread $(\gamma$

.

```
Comp $(\Delta$ □ $s$ trace $\Theta$ $= \nu(\Theta(s$ ,ref Ref, state$_\varepsilon$ State .(
    ref val $=$ state$_\varepsilon$ □
    state$_\varepsilon$ State$(\Delta$ □ $\varepsilon \leq s$ trace $\Theta$ □
    $\prod\{p\ l_i =$ ref.val.inCall$_{p.l_i\ L_i}$ $\mid i =\ \ldots n$ $\mid p\ l_i\ L_i \mid i =\ \ldots n$ □ $\Theta, \Theta(s\ \}$ □
    $\prod\{n$□ref.val.out$_{\mathsf{none}}($ □ $\mid n$ thread □ $\Theta, \Theta(s\ \}$

Ref $=$ val State
State $=$ out$_T$ ( $\to T$, inReturn$_T$ $(T \to T$, inCall$_{p.l\ L}$ $L$
State$(\Delta$ □ $r \leq s$ trace $\Theta$ $=($
    out$_T$ $=$ Out$_T(\Delta$ □ $r \leq s$ trace $\Theta$ ,
    inReturn$_T$ $=$ InReturn$_T(\Delta$ □ $r \leq s$ trace $\Theta$ ,
    inCall$_{p.l\ L}$ $=$ InCall$_{p.l\ L}(\Delta$ □ $r \leq s$ trace $\Theta$

Out$_T(\Delta$ □ $r \leq s$ trace $\Theta$ $= \lambda($ .(
    when $ra \leq s$ an $a = \nu(\Theta$ . $n$□call $p.l(\vec{v}$ □ an $,\Delta,\Theta,\Delta(r\ ,\Theta(r\ ,\Theta$ □ $p.l(\vec{v}\ U$
        if currentthread $= n$ then
            ref.val $=$ new State$(\Delta$ □ $ra \leq s$ trace $\Theta$ ,
            ref.val.inReturn$_U(p.l(\vec{v}$ ,
            ref.val.out$_T($
    when $ra \leq s$ an $a = \nu(\Theta$ . $n$□return $v$□ an $,\Delta,\Theta,\Delta(r\ ,\Theta(r\ ,\Theta$ □ $v\ T$
        if currentthread $= n$ then
            ref.val $=$ new State$(\Delta$ □ $ra \leq s$ trace $\Theta$ ,
            $v$
    otherwise
        stop

InReturn$_T(\Delta$ □ $r \leq s$ trace $\Theta$ $= \lambda(x\ T$ .(
    when $ra \leq s$ an $a = \nu(\Delta$ . $n$□return $v$□? an $,\Delta,\Theta,\Delta(r\ ,\Theta(r\ ,\Delta$ □ $v\ T$
        if $\Delta,\Theta,\Delta(r\ ,\Theta(r$ □ (currentthread, $x$ $= \nu(\Delta$ . $(n, v$ then
            ref.val $=$ new State$(\Delta$ □ $ra \leq s$ trace $\Theta$ ,
            $v$
    otherwise
        stop

InCall$_{p.l\ (\vec{T}\ \to T}(\Delta$ □ $r \leq s$ trace $\Theta$ $= \lambda(\vec{x}\ \vec{T}$ .(
    when $ra \leq s$ an $a = \nu(\Delta$ . $n$□call $p.l(\vec{v}$ □? an $,\Delta,\Theta,\Delta(r\ ,\Theta(r\ ,\Delta$ □ $\vec{v}\ \vec{T}$
        if $\Delta,\Theta,\Delta(r\ ,\Theta(r$ □ (currentthread, $\vec{x}$ $= \nu(\Delta$ . $(n, \vec{v}$ then
            ref.val $=$ new State$(\Delta$ □ $ra \leq s$ trace $\Theta$ ,
            ref.val.out$_T($
    otherwise
        stop
```

Figure Definition of Comp $(\Delta$ □ s trace Θ

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

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

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

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

. 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 ~~oo~~ Traces Θ (=(((

1. *If* C ⩕ $C \equiv D$ E *then there exist components such that* $C \equiv D$ E *and* $C \equiv D$ E *with* $D \equiv D$ ⩕ D *and* $E \equiv E$ ⩕ E .

2. *If* C ⩕ $C \equiv \nu(\vec{n}\ \vec{T}\ .C$ *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* $C \equiv C$ ⩕ C.

Proof rove by n uct on on t e er vat on o C ⩕ C □

Lemma A. *If* C ⩕ $C \equiv C$ *and* $C \xrightarrow{\beta} C$ b

- **Case** $(\gamma = \nu(\vec{n}\ \ \vec{T}\ \ .\ n\square\text{call}\, p.l(\vec{v}\ \square$ an $\ n \square \Sigma$

  ar to t e prev ous case

- **Case** $(\gamma = \nu(\vec{n}\ \ \vec{T}\ \ .\ n\square\text{return}\, v\square$

  nce $(\Delta, \Phi\ \square C\ \ \Theta, \Sigma\ \ \xrightarrow{\gamma?}\ (\Delta^{\square}, \Phi\ \square C^{\square}\ \ \Theta^{\square}, \Sigma^{\square}$ we ust ave t at

$$\begin{aligned}
C\ \ &\equiv \nu(\vec{p}\ \ \vec{U}\ \ .\ (C^{\square\square}\ \square\ n\square\text{let}\ x\ \ T = \text{block in}\ t\ \square \\
C^{\square}\ &\equiv \nu(\vec{p}\ \ \vec{U}\ \ .\ (C^{\square\square}\ \square\ n\square t\ \ v/x\ \square \\
\Delta^{\square} &= \Delta, \vec{n}\ \ \vec{T} \\
\Theta^{\square} &= \Theta \\
\Sigma^{\square} &= \Sigma
\end{aligned}$$

  nce $(\Theta, \Phi\ \square C\ \ \Delta, \Sigma\ \ \xrightarrow{\gamma}\ (\Theta^{\square}, \Phi\ \square C^{\square}\ \ \Delta^{\square}, \Sigma^{\square}$ we ust ave t at

$$\begin{aligned}
C\ \ &\equiv \nu(\vec{n}\ \ \vec{T}\ \ .\ \nu(\vec{p}\ \ \vec{U}\ \ .\ (C^{\square\square}\ \square\ n\square\text{let}\ y\ \ U = \text{return}(v\ \ T\ \ \text{in}\ t\ \square \\
C^{\square}\ &\equiv \nu(\vec{p}\ \ \vec{U}\ \ .\ (C^{\square\square}\ \square\ n\square\text{let}\ y\ \ U = \text{block in}\ t\ \square
\end{aligned}$$

  e t en s ow t at

$$C\ \ /\!\!\backslash\!\!\backslash\, C\ \ \equiv \nu(\vec{n}\ \ \vec{T}\ \ .\ \nu(\vec{p}\ \ \vec{U}\ \ .\ \nu(\vec{p}\ \ \vec{U}\ \ .\ ((C^{\square\square}\ /\!\!\backslash\!\!\backslash\, C^{\square\square}\ \ \square\ n\square(\text{let}\ y\ \ U = \text{block in}\ t\ \ /\!\!\backslash\!\!\backslash\ (t\ \ v/x\ \ \square$$

  an t at

$$C^{\square}\ /\!\!\backslash\!\!\backslash\, C^{\square}\ \equiv \nu(\vec{p}\ \ \vec{U}\ \ .\ \nu(\vec{p}\ \ \vec{U}\ \ .\ ((C^{\square\square}\ /\!\!\backslash\!\!\backslash\, C^{\square\square}\ \ \square\ n\square(\text{let}\ y\ \ U = \text{block in}\ t\ \ /\!\!\backslash\!\!\backslash\ (t\ \ v/x\ \ \square$$

  an so

$$\nu(\Delta^{\square}, \Theta^{\square}, \Sigma^{\square} \setminus \Delta, \Theta, \Sigma\ \ .\ (C^{\square}\ /\!\!\backslash\!\!\backslash\, C^{\square}\ \ \equiv C$$

  as requ re □

Co pos t on o ows by n uct on on t e er vat on o $(\Delta, \Phi\ \square C\ \ \Theta, \Sigma\ \ \overset{s}{\Longrightarrow}\ (\Delta^{\square}, \Phi\ \square C^{\square}\ \ \Theta^{\square}, \Sigma^{\square}$ an $(\Theta, \Phi\ \square C\ \ \Delta, \Sigma\ \ \overset{s}{\Longrightarrow}\ (\Theta^{\square}, \Phi\ \square C^{\square}\ \ \Delta^{\square}, \Sigma^{\square}$ a n use o Le as A A an A

A. Decomposition

e s ow t ree e as ro w c Deco pos t on o ows

Lemma A. *For any* $\Delta, \Phi\ \square C\ \ \Theta, \Sigma$ *and* $\Theta, \Phi\ \square C\ \ \Delta, \Sigma$ *if* $(C\ \ /\!\!\backslash\!\!\backslash\, C\ \ \equiv \nu(\vec{n}\ \ \vec{T}\ \ .\ (C\ \square\ n\square\text{let}\ x\ \ T = e\ \text{in}\ t\square$ *then either we have:*

$$\begin{aligned}
&(\Delta, \Phi\ \square C\ \ \Theta, \Sigma\ \ \overset{s}{\Longrightarrow}\ (\Delta^{\square}, \Phi\ \square\ \nu(\vec{n}\ \ \vec{T}\ \ .\ (C^{\square}\ \square\ n\square\text{let}\ x\ \ T = e\ \text{in}\ t\ \square\ \ \Theta^{\square}, \Sigma^{\square} \\
&(\Theta, \Phi\ \square C\ \ \Delta, \Sigma\ \ \overset{s}{\Longrightarrow}\ (\Theta^{\square}, \Phi\ \square C^{\square}\ \ \Delta^{\square}, \Sigma^{\square}
\end{aligned}$$

where:

$$\nu(\Delta^{\square}, \Theta^{\square}, \Sigma^{\square} \setminus \Delta, \Theta, \Sigma\ \ .\ \nu(\vec{n}\ \ \vec{T}\ \ .\ (C^{\square}\ \square\ n\square t\ \square\ \ /\!\!\backslash\!\!\backslash\, C^{\square}\ \equiv \nu(\vec{n}\ \ \vec{T}\ \ .\ (C\ \square\ n\square t\square$$

or symmetrically, swapping the roles of C and C .

Proof: An induction on the derivation of

$$(C \mathbin{/\!\!\backslash\!\backslash} C) \equiv \nu(\vec{n} : \vec{T}).(C \parallel n\langle \text{let } x : T = e \text{ in } t\rangle)$$

The interesting case is when

$$\begin{aligned} C &\equiv n\langle \text{let } x : T = \text{block in } t\rangle \\ C &\equiv n\langle \text{let } x : T = \text{return}(v : T) \text{ in } t\rangle \end{aligned}$$

and

$$n\langle t[v/x]\rangle \mathbin{/\!\!\backslash\!\backslash} n\langle \text{let } x : T = \text{block in } t\rangle \equiv \nu(\vec{n} : \vec{T}).(C \parallel n\langle \text{let } x : T = e \text{ in } t\rangle)$$

so by definition of the ts and by induction we have

$$\begin{aligned} &(\Delta, \Phi \vdash C : \Theta, \Sigma) \xrightarrow{n\langle\text{return } v\rangle?} (\Delta, \Phi \vdash n\langle t[v/x]\rangle : \Theta, \Sigma) \\ &(\Delta, \Phi \vdash n\langle t[v/x]\rangle : \Theta, \Sigma) \overset{s}{\Longrightarrow} (\Delta', \Phi \vdash \nu(\vec{n} : \vec{T}).(C' \parallel n\langle \text{let } x : T = e \text{ in } t\rangle) : \Theta', \Sigma') \end{aligned}$$

and

$$\begin{aligned} &(\Delta, \Phi \vdash C : \Theta, \Sigma) \xrightarrow{n\langle\text{return } v\rangle!} (\Theta, \Phi \vdash n\langle \text{let } x : T = \text{block in } t\rangle : \Delta, \Sigma) \\ &(\Theta, \Phi \vdash n\langle \text{let } x : T = \text{block in } t\rangle : \Delta, \Sigma) \overset{s}{\Longrightarrow} (\Theta', \Phi \vdash C' : \Delta', \Sigma') \end{aligned}$$

where

$$\nu((\Delta', \Theta', \Sigma') \setminus (\Delta, \Theta, \Sigma)).\nu(\vec{n} : \vec{T}).(C' \parallel n\langle t\rangle) \mathbin{/\!\!\backslash\!\backslash} C' \equiv \nu(\vec{n} : \vec{T}).(C \parallel n\langle t\rangle)$$

or symmetrically as required. □

Lemma A. *If* $C \mathbin{/\!\!\backslash\!\backslash} C \equiv C$ *and* $C \xrightarrow{\beta}$ 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

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

- **Case** $(p$ □ dom $(C''$ $, n$ □ dom $(C''$

  e ust ave t at

$$C'' \equiv \nu(\vec{p} \quad \vec{U} \quad . (C''' \; \square \; p \; O \; \square \; n \square \text{let } y \quad U = \text{block in } t \; \square$$

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

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

n w c case

$$(\Delta, \Phi \; \square \; C \quad \Theta, \Sigma \; \xRightarrow{s \; \nu(\vec{n}' \; \vec{T}' \;\; .n \square \text{call } p.l(\vec{v} \; \square} \square \; (\Delta, \Phi \; \square \; C' \quad \Theta', \vec{n}' \quad \vec{T}', \Sigma'$$

w ere we e ne

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

an we part t on $\{\vec{n} \quad \vec{T}\}$ nto $\{\vec{n}' \quad \vec{T}', \vec{n}'' \quad \vec{T}''\}$ suc t at $\{\vec{n}'\}$ □ fn $(p.l(\vec{v}$ an $\{\vec{n}''\} \cap$ fn $(p.l(\vec{v} \quad = \emptyset$

  e a so ave

$$(\Delta, \Phi \; \square \; C \quad \Theta, \Sigma \; \xRightarrow{s \; \nu(\vec{n}' \; \vec{T}' \;\; .n \square \text{call } p.l(}$$

B. † Technical preliminaries

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

A *component for* Δ □ r ≤ s trace Θ resp *f* or Δ □ q □ r ≤ s trace Θ is one o*f* the *f* or

$\nu(\Theta(s \setminus \Theta(q$ $.\nu($ref Ref $.\nu($state$_r$ State $|\,\Delta$ □ r' ≤ r trace Θ $.($
ref val $=$ state$_r$ □
$\prod\{$state$_{r'}$ State$(\Delta$ □ r' ≤ s trace Θ $|\,\Delta$ □ r' ≤ r trace $\Theta\}$ □
$\prod\{p\ l_i =$ ref.val.inCall$_{p.l_i\ L_i}$ $|\ i = \ \ldots n$ $|\ p\ \ l_i\ \ L_i\ |\ i = \ \ldots n$ □ $\Theta, \Theta(s\ \}$ □
$\prod\{n$□t_n□$|\ n$ thread □ $\Theta, \Theta(s\ \}$ □
$\prod\{n$□t_n□$|\ n$ thread □ $\Delta, \Delta(s$ an n □ threads $(q\ \}$

where t_n is a thread at n *f* or Δ □ r ≤ s trace Θ resp *f* or Δ □ q □ r ≤ s trace Θ

A *thread at n for* Δ □ r ≤ s trace Θ is one o*f* the *f* o own

- let x $T =$ ref.val.out$_T($ in t
  where n is output enab e in Δ □ r trace Θ an t is a return$(x$ T threa at n *f* or Δ □ r ≤ s trace Θ
- let x $T =$ block in t
  where n is input enab e in Δ □ r trace Θ an t is a return$(x$ T threa at n *f* or Δ □ r ≤ s trace Θ

A return$(v$ T *thread at n for* Δ □ r ≤ s trace Θ is one o*f* the *f* o own

- v
  where n is ba ance in r
- ref.val.inReturn$_T(v$ $,t$
  where $r = r$ $a\ r$ $a = \nu(\Theta'$ $.n$□call $p.l(\vec{v}$ □ n is ba ance in r
  an t is a threa at n *f* or Δ □ r ≤ s trace Θ
- let y $U =$ return$(v$ T in t
  where $r = r$ $a\ r$ $a = \nu(\Theta'$ $.n$□call $p.l(\vec{v}$ □? n is ba ance in r
  an t is a return$(y$ U threa at n *f* or Δ □ r ≤ s trace Θ

Fi ure ● De nition o*f* a co ponent *f* or Δ □ r ≤ s trace Θ an *f* or Δ □ q □ r ≤ s trace Θ

A *thread at n for* Δ □ q □ r ≤ s trace Θ s one o t e o ow n

- stop
- a t rea at n or Δ □ r ≤ s trace Θ
  w ere proj n $(q$ = proj n $(r$
- let x $T = p.l(\vec{v}$ in t
  w ere proj n $(qa$ = proj n $(r$ $a = \nu(\Theta$ $.n$□call $p.l(\vec{v}$ □ an t s a return $(x$ T
  t rea at n or Δ □ r ≤ s trace Θ
- let x T = return $(v$ U in t
  w ere proj n $(qa$ = proj n $(r$ $a = \nu(\Theta$ $.n$□return v□ an t s a return $(x$ T
  t rea at n or Δ □ r ≤ s trace Θ
- let y U = ref.val.inCall$_{p.l\ L}(\vec{v}$ in let x T = return $(y$ U in t
  w ere proj n $(q$ = proj n $(ra$ $a = \nu(\Delta$ $.n$□call $p.l(\vec{v}$ □? an t s a return $(x$ T
  t rea at n or Δ □ r ≤ s trace Θ
- t
  w ere proj n $(q$ = proj n $(ra$ $a = \nu(\Delta$ $.n$□return v□? an t s a return $(v$ T
  t rea at n or Δ □ r ≤ s trace Θ or so e T
- ref.val = new State(Δ □ ra ≤ s trace Θ ,t
  w ere proj n $(q$ = proj n $(ra$ an t s a t rea at n or Δ □ ra ≤ s trace Θ
- t
  w ere n□t□ $\xrightarrow{\beta}$ n□t'□ an t' s a t rea at n or Δ □ q □ r ≤ s trace Θ

F ure De nition o a t rea or Δ □ q □ r ≤ 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* $(\Delta' \vdash C : \Theta') \overset{a}{\Longrightarrow} (\Delta'' \vdash C' : \Theta'')$ *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 $C \equiv \nu(\Theta''').C\ \mathrm{ref\ val} = \mathrm{state}_r;\ n.\mathrm{let}\ y : U = \mathrm{ref.val.out}_U()\ \mathrm{in\ let}\ x : T = \mathrm{return}(y : U)\ \mathrm{in}\ t$.

  We have

  $(\Delta' \vdash C : \Theta')$

  $\xrightarrow{\tau} (\Delta' \vdash \nu(\Theta''').C\ \mathrm{ref\ val} = \mathrm{state}_r;$
  $\quad n.\mathrm{let}\ y : U = \mathrm{state}_r.\mathrm{out}_U()\ \mathrm{in\ let}\ x : T = \mathrm{return}(y : U)\ \mathrm{in}\ t : \Theta')$

  $\xrightarrow{\beta} (\Delta' \vdash \nu(\Theta''').C\ \mathrm{ref\ val} = \mathrm{state}_r;$
  $\quad n.\mathrm{ref.val} := \mathrm{new\ State}(\Delta \vdash ra \lesssim s : \mathrm{trace}\ \Theta); \mathrm{let}\ y : U = v\ \mathrm{in\ let}\ x : T = \mathrm{return}(y : U)\ \mathrm{in}\ t : \Theta')$

  $\xrightarrow{\tau} (\Delta' \vdash \nu(\Theta''', \mathrm{state}_{ra} : \mathrm{State}).C\ \mathrm{ref\ val} = \mathrm{state}_{ra};\ \mathrm{state}_{ra}\ \mathrm{State}(\Delta \vdash ra \lesssim s : \mathrm{trace}\ \Theta);$
  $\quad n.\mathrm{let}\ y : U = v\ \mathrm{in\ let}\ x : T = \mathrm{return}(y : U)\ \mathrm{in}\ t : \Theta')$

  $\xrightarrow{\beta} (\Delta' \vdash \nu(\Theta''', \mathrm{state}_{ra} : \mathrm{State}).C\ \mathrm{ref\ val} = \mathrm{state}_{ra};\ \mathrm{state}_{ra}\ \mathrm{State}(\Delta \vdash ra \lesssim s : \mathrm{trace}\ \Theta);$
  $\quad n.\mathrm{let}\ x : T = \mathrm{return}(v : U)\ \mathrm{in}\ t : \Theta')$

  $\xrightarrow{a} (\Delta' \vdash \nu(\mathrm{state}_{ra} : \mathrm{State}).C\ \mathrm{ref\ val} = \mathrm{state}_{ra};\ \mathrm{state}_{ra}\ \mathrm{State}(\Delta \vdash ra \lesssim s : \mathrm{trace}\ \Theta);$
  $\quad n.\mathrm{let}\ x : T = \mathrm{block\ in}\ t : \Theta', \Theta''')$

  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 $C \equiv \nu(\Theta''').C\ \mathrm{ref\ val} = \mathrm{state}_r;\ n.\mathrm{let}\ y : U = \mathrm{ref.val.out}_U()\ \mathrm{in}\ t$.

  We have

  $(\Delta' \vdash C : \Theta')$

  $\xrightarrow{\tau} (\Delta' \vdash \nu(\Theta''').C\ \mathrm{ref\ val} = \mathrm{state}_r;$
  $\quad n.\mathrm{let}\ y : U = \mathrm{state}_r.\mathrm{out}_U()\ \mathrm{in}\ t : \Theta')$

  $\xrightarrow{\beta} (\Delta' \vdash \nu(\Theta''').C\ \mathrm{ref\ val} = \mathrm{state}_r;$
  $\quad n.\mathrm{ref.val} := \mathrm{new\ State}(\Delta \vdash ra \lesssim s : \mathrm{trace}\ \Theta);$
  $\qquad \mathrm{let}\ x : T = p.l(\vec{v})\ \mathrm{in\ ref.val.inReturn}_T(x); \mathrm{let}\ y : U = \mathrm{ref.val.out}_U()\ \mathrm{in}\ t : \Theta')$

  $\xrightarrow{\tau} (\Delta' \vdash \nu(\Theta''', \mathrm{state}_{ra} : \mathrm{State}).C\ \mathrm{ref\ val} = \mathrm{state}_{ra};\ \mathrm{state}_{ra}\ \mathrm{State}(\Delta \vdash ra \lesssim s : \mathrm{trace}\ \Theta);$
  $\quad n.\mathrm{let}\ x : T = p.l(\vec{v})\ \mathrm{in\ ref.val.inReturn}_T(x); \mathrm{let}\ y : U = \mathrm{ref.val.out}_U()\ \mathrm{in}\ t : \Theta')$

  $\xrightarrow{a} (\Delta' \vdash \nu(\mathrm{state}_{ra} : \mathrm{State}).C\ \mathrm{ref\ val} = \mathrm{state}_{ra};\ \mathrm{state}_{ra}\ \mathrm{State}(\Delta \vdash ra \lesssim s : \mathrm{trace}\ \Theta);$
  $\quad n.\mathrm{let}\ x : T = \mathrm{block\ in\ ref.val.inReturn}_T(x); \mathrm{let}\ y : U = \mathrm{ref.val.out}_U()\ \mathrm{in}\ t : \Theta', \Theta''')$

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

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

e ave

$(\Delta \; C \;\; \Theta$

$\xrightarrow{a} (\Delta, \Delta \; C \text{ ref } \text{val} = \text{state}_r$

$\quad n \text{ let } x \;\; T = v \text{ in ref.val.inReturn}_T(x \;\; ,t \;\; \Theta$

$\xrightarrow{\beta} (\Delta, \Delta \; C \text{ ref } \text{val} = \text{state}_r$

$\quad n \text{ ref.val.inReturn}_T(v \;\; ,t \;\; \Theta$

$\xrightarrow{\tau} (\Delta, \Delta \; C \text{ ref } \text{val} = \text{state}_r$

$\quad n \text{ state}_r\text{.inReturn}_T(v \;\; ,t \;\; \Theta$

$\xrightarrow{\beta} (\Delta, \Delta \; C \text{ ref } \text{val} = \text{state}_r$

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

$\xrightarrow{\tau} (\Delta, \Delta \; C \; \nu(\text{state}_{ra} \;\; \text{State} \;\; \text{.ref val} = \text{state}_{ra} \;\; \text{state}_{ra} \text{ State}(\Delta \; ra \leq s \;\; \text{trace } \Theta$

$\quad n \; t \;\; \Theta$

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

Case $a = \nu(\Delta \;\; . n \text{ call } p.l(\vec{v}$? an $C \equiv C$ ref val $= \text{state}_r \;\; n \text{ let } x \;\; T = \text{block in } t$

e ave

$(\Delta \; C \;\; \Theta$

$\xrightarrow{a} (\Delta, \Delta \; C \text{ ref } \text{val} = \text{state}_r$

$\quad n \text{ let } y \;\; U = p.l(\vec{v} \;\; \text{in let } x \;\; T = \text{return}(y \;\; U \;\; \text{in } t \;\; \Theta$

$\xrightarrow{\beta} (\Delta, \Delta \; C \text{ ref } \text{val} = \text{state}_r$

$\quad n \text{ let } y \;\; U = \text{ref.val.inCall}_{p.l \; L}(\vec{v} \;\; \text{in let } x \;\; T = \text{return}(y \;\; U \;\; \text{in } t \;\; \Theta$

$\xrightarrow{\tau} (\Delta, \Delta \; C \text{ ref } \text{val} = \text{state}_r$

$\quad n \text{ let } y \;\; U = \text{state}_r\text{.inCall}_{p.l \; L}(\vec{v} \;\; \text{in let } x \;\; T = \text{return}(y \;\; U \;\; \text{in } t \;\; \Theta$

$\xrightarrow{\beta} (\Delta, \Delta \; C \text{ ref } \text{val} = \text{state}_r$

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

$\quad \text{let } y \;\; U = \text{ref.val.out}$

for Δ □ *q* □ *r* ≤ *s* trace Θ n F ures an w t t e nten e ean n t at a co ponent or Δ □ *q* □ *r* ≤ *s* trace Θ as per or e t e trace *q* an t s s □ re ate to so e pre x o *s* ote t at as pre x or er n ≤ on traces s conta ne n □

- **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

$$C□ \xrightarrow{\beta} □ \quad C\ n□\text{ref val} = \text{new State}(\Delta □ ra \leq s \text{ trace } \Theta \;\; ,$$
$$\text{ref.val.inReturn}_U(p.l(\vec{v} \;\; ,\text{let } x \;\; T = \text{ref.val.out}_T(\;\; \text{in } t□$$

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** $(\Delta^{\square} \square C\ n\square\mathrm{let}\ x\ \ T = \mathrm{block\ in}\ t\square\ \ \Theta^{\square} \xrightarrow{\nu(\Delta^{\square\square\square}\ .n\square\mathrm{call}\,p.l(\vec{v}\ \square?}(\Delta^{\square},\Delta^{\square\square\square} \square C\ n\square\mathrm{let}\ y\ \ U = p.l(\vec{v}\ \ \mathrm{in\ let}\ x\ \ T = \mathrm{return}(y\ \ U\ \ \mathrm{in}\ t\square\ \ \Theta^{\square}$
  where $\mathrm{proj}\,n\,(q\ \ = \mathrm{proj}\,n\,(r$ n is input enabled in $\Delta \square r$ trace Θ and t is a $\mathrm{return}(x\ \ T$ thread at n. For $\Delta \square r \leq s$ trace Θ.

  We have

  $$C^{\square} \xrightarrow{\beta}{}^{\square} C\ n\square\mathrm{let}\ y\ \ U = \mathrm{ref.val.inCall}_{p.l\ L}(\vec{v}\ \ \mathrm{in\ let}\ x\ \ T = \mathrm{return}(y\ \ U\ \ \mathrm{in}\ t\square$$

  which is a component for $\Delta \square qa \square r \leq s$ trace Θ as required.
- **Case** $(\Delta^{\square} \square C\ n\square\mathrm{let}\ x\ \ T = \mathrm{block\ in}\ t\square\ \ \Theta^{\square} \xrightarrow{\nu(\Delta^{\square\square\square}\ .n\square\mathrm{return}\ v\square?}(\Delta^{\square},\Delta^{\square\square\square} \square C\ n\square\mathrm{let}\ x\ \ T = v\ \mathrm{in}\ t\square\ \ \Theta^{\square}$
  where $\mathrm{proj}\,n\,(q\ \ = \mathrm{proj}\,n\,(r$ n is input enabled in $\Delta \square r$ trace Θ and t is a $\mathrm{return}(x\ \ T$ thread at n. For $\Delta \square r \leq s$ trace Θ.

  We have

  $$C^{\square} \xrightarrow{\beta}{}^{\square} C\ n\square t\ v/x\ \square$$

  which is a component for $\Delta \square qa \square r \leq s$ trace Θ as required.
- **Case** $(\Delta^{\square} \square \nu(\Theta^{\square\square\square}\ .C\ n\square\mathrm{let}\ x\ \ T = p.l(\vec{v}\ \ \mathrm{in}\ t\square\ \ \Theta^{\square} \xrightarrow{\nu(\Theta^{\square\square\square}\ .n\square\mathrm{call}\,p.l(\vec{v}\ \square}(\Delta^{\square} \square C\ n\square\mathrm{let}\ x\ \ T = \mathrm{block\ in}\ t\square\ \ \Theta^{\square},\Theta^{\square\square\square}$
  where $\mathrm{proj}\,n\,(qa\ \ = \mathrm{proj}\,n\,(r$ and t is a $\mathrm{return}(x\ \ T$ thread at n. For $\Delta \square r \leq s$ trace Θ.
  We have $C^{\square}$ is a component for $\Delta \square qa \square r \leq s$ trace Θ as required.
- **Case** $(\Delta^{\square} \square \nu(\Theta^{\square\square\square}\ .C\ n\square\mathrm{let}\ x\ \ T = \mathrm{return}(v\ \ U\ \ \mathrm{in}\ t\square\ \ \Theta^{\square} \xrightarrow{\nu(\Theta^{\square\square\square}\ .n\square\mathrm{return}\ v\square}(\Delta^{\square} \square C\ n\square\mathrm{let}\ x\ \ T = \mathrm{block\ in}\ t\square\ \ \Theta^{\square},\Theta^{\square\square\square}$
  where $\mathrm{proj}\,n\,(qa\ \ = \mathrm{proj}\,n\,(r$ and t is a $\mathrm{return}(x\ \ T$ thread at n. For $\Delta \square r \leq s$ trace Θ.
  We have $C^{\square}$ is a component for $\Delta \square qa \square r \leq s$ trace Θ as required. □

The only if half of definability now follows by induction on Lemmas B. B. and B. with Lemma B. as the base case and an appropriate use of Corollary 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 . y.p.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.*