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

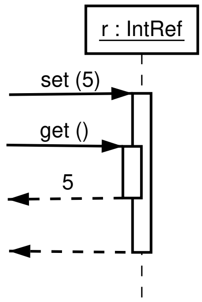
Interact on a ra s n part cu ar sequence a ra s were eve ope by Jacobson an are now part o t e n e Mo e n Lan ua e stan ar Interact on a ra s recor t e essa es sent between ob ects o a co ponent n an ob ect syste ese essa es nc u e et o ca s

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

an returns nteraction a ra s nc u e other or s o essa e but we w not use these n th s paper

A s p e nteraction w th an nte er re erence ob ect r o 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(5)
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 esi 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 e can t en e ne t e *may testing preorder* as $C \sqsubseteq_{\mathrm{ay}} C$ whenever

$$\text{or any appropr ate y type } C$$
$$C \;\; C \text{ s success u t en } C \;\; C \text{ s success u}$$

n ortunate y a t ou 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 t e quant cat on over any appropr ate y type C In pract ce we requ re a proo tec n que w c we can use to s ow 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 $C \stackrel{s}{\Longrightarrow} C'$ w ere s s a sequence o essa es e t en wr te Traces$(C$ or t e set o a traces o C e say t at

- races are *sound* or ay test n w en
  Traces$(C \;\; \text{Traces}(C$ p es $C \sqsubseteq_{\mathrm{ay}} C$
- races are *complete* or ay test n w en
  $C \sqsubseteq_{\mathrm{ay}} C$ p es Traces$(C \;\; \text{Traces}(C$
- races are *fully absdwhereTj11.993tTJ/R1.991Tf1.19Tdfor-2may-22.9test-21.9ing-2.whenTJsi.9911.so/R11.9*

Co ponents	C	$=$	$\cdot \mid C$ [illegible] $C \mid \nu(n \;\; T \;\; . C \mid n\, O \;\mid n$[illegible]$t$[illegible]
b ects	O	$=$	$l = M, \ldots, l = M$
Met o s	M	$=$	$\varsigma(n \;\; T \;\; . \lambda(x \;\; T, \ldots, x \;\; T \;\; .$[illegible]$t$[illegible]
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 $f\ (\ \to T$
- 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 judgement is $\Delta \vdash C : \Theta$ which is read as 'the component C uses names Δ and defines names Θ'. For example, we define $C_1(v)$ and C_2 and IntRef as:

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

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

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

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

$$\frac{\Gamma,\Delta \vdash v\ \ T \quad \Gamma,\Delta \vdash v\ \ T \qquad \Gamma,\Delta \vdash e\ \ T \quad \Gamma,\Delta \vdash e\ \ T}{\Gamma,\Delta \vdash \text{if } v\ \ = v\ \ \text{ then } e\ \ \text{ else } e\ \ T}$$

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

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

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

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

Whenever $\Delta \vdash C : \Theta$ contains a subexpression of [redacted] n appears in Θ.

[redacted]his is intended to capture the common software engin[redacted]s[redacted]hou[redacted]ld not export mutable fields, instead they should export su[redacted]e the configurations C and C' above are write-closed since th[redacted]own component which writes directly to p.contents is not w[redacted]

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

[redacted]or the remainder of the paper we will require compo[redacted] open in a fully abstract semantics much simpler since [redacted]rectly.

[redacted] Dynamic semantics

[redacted]he dynamic semantics for our concurrent object calcu[redacted]n in[redacted] define three relations between components:

- $\equiv$ structural congruence represents the least con[redacted]n co[redacted] axioms in Figure [redacted]

[redacted]

t

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

Figure Axioms for structural congruence where n is not free in C

$$n\,\mathrm{let}\; x \; T = v \;\mathrm{in}\; t \xrightarrow{\beta} n\; t\; v/x$$
$$n\,\mathrm{let}\; x \; T = (\mathrm{let}\; x \; T = e \;\mathrm{in}\; e \;\mathrm{in}\; t \xrightarrow{\beta} n\,\mathrm{let}\; x \; T = e \;\mathrm{in}\; (\mathrm{let}\; x \; T = e \;\mathrm{in}\; t$$
$$n\,\mathrm{let}\; x \; T = (\mathrm{if}\; v = v \;\mathrm{then}\; e \;\mathrm{else}\; e \;\mathrm{in}\; t \xrightarrow{\beta} n\,\mathrm{let}\; x \; T = e \;\mathrm{in}\; t$$
$$n\,\mathrm{let}\; x \; T = (\mathrm{if}\; v = v \;\mathrm{then}\; e \;\mathrm{else}\; e \;\mathrm{in}\; t \xrightarrow{\beta} n\,\mathrm{let}\; x \; T = e \;\mathrm{in}\; t \qquad v \neq v$$
$$n\,\mathrm{let}\; x \; T = \mathrm{new}\; O \;\mathrm{in}\; t \xrightarrow{\beta} \nu(p \; T \; .(p \; O \; n\,\mathrm{let}\; x \; T = p \;\mathrm{in}\; t \qquad \mathit{else}$$
$$\mathrm{let}\; x \; T =$$

. Testing preorder

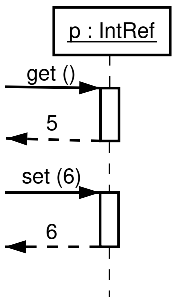
e w now e ne t e test n se ant cs or our concurrent ob ect ca cu us e w o t s by e n n a not on o *barb*

$(\Delta, n$ thread □ C

then where $C(v)$ is defined in section , we have

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

which corresponds to the interaction diagram:



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

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

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 $C \parallel C$ on co ponents as t e sy etr c operator e ne up to $\equiv$ w ere

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

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 $t \parallel t$ on t rea s as t e sy etr c operator w ere

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

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

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

Proof An n uct on on t e e n t on o $C \parallel C$ $\square$

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

Proof An n uct on on t e e n t on o $C \parallel C$ $\square$

. 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. Given in Appendix A. $\square$

Corollary. *For any components* $(\Delta,\Phi \vdash C \dashv \Theta,\Sigma)$ *and* $(\Theta,\Phi \vdash \overline{C} \dashv \Delta,\Sigma)$ *such that* $C \bowtie \overline{C} \equiv C$ *and* $C \Downarrow_b$ *then there exists some trace* s *such that* $(\Delta,\Phi \vdash C \dashv \Theta,\Sigma) \xRightarrow{s} (\Delta',\Phi \vdash C' \dashv \Theta',\Sigma')$ *and* $(\Theta,\Phi \vdash \overline{C} \dashv \Delta,\Sigma) \xRightarrow{\overline{s}} (\Theta',\Phi \vdash \overline{C'} \dashv \Delta',\Sigma')$ *where either* $C' \downarrow_b$ *or* $\overline{C'} \downarrow_b$.

Proof. We know that $C \Downarrow_b$ which tells us that $C \to^* C''$ for some C'' such that $C'' \downarrow_b$. We use Proposition … part … to obtain a trace s such that

$$(\Delta,\Phi \vdash C \dashv \Theta,\Sigma) \xRightarrow{s} (\Delta'',\Phi \vdash C'' \dashv \Theta'',\Sigma'')$$
$$(\Theta,\Phi \vdash \overline{C} \dashv \Delta,\Sigma) \xRightarrow{\overline{s}} (\Theta'',\Phi \vdash \overline{C''} \dashv \Delta'',\Sigma'')$$

where $\nu(\Delta'',\Theta'',\Sigma'' \setminus \Delta,\Theta,\Sigma).(C'' \bowtie \overline{C''}) \equiv C''$. Given that $C'' \downarrow_b$ we know that $(C'' \bowtie \overline{C''}) \downarrow_b$ also. By the definition of $\bowtie$ we see that one of the following or their symmetric counterparts must hold:

- $C'' \downarrow_b$ and we are done, or
- $C'' \equiv \nu(\Delta_1).(n\langle t_1 \rangle \parallel C''')$ and $\overline{C''} \equiv \nu(\Delta_2).(n\langle t_2 \rangle \parallel \overline{C'''})$ where $n\langle t_1 \bowtie t_2 \rangle \downarrow_b$. We now proceed by induction on the definition of $t_1 \bowtie t_2$ to show that, for all such C'' and $\overline{C''}$ we can find s' where

$$(\Delta'',\Phi \vdash C'' \dashv \Theta'',\Sigma'') \xRightarrow{s'} (\Delta',\Phi \vdash C' \dashv \Theta',\Sigma')$$
$$(\Theta'',\Phi \vdash \overline{C''} \dashv \Delta'',\Sigma'') \xRightarrow{\overline{s'}} (\Theta',\Phi \vdash \overline{C'} \dashv \Delta',\Sigma')$$

and either $C' \downarrow_b$ or $\overline{C'} \downarrow_b$. There are two cases (up to symmetry of $\bowtie$):

- If $t_1 = \text{let } x : T = \text{block in } t'$ and $t_2 = \text{let } y : U = b.\text{succ}()$ in t'' then $C'' \downarrow_b$.
- If $t_1 = \text{let } x : T = \text{block in } t'$ and $t_2 = \text{let } y : U = \text{return}(v : T)$ in t'' then we have

$$(\Delta'',\Phi \vdash C'' \dashv \Theta'',\Sigma'') \xrightarrow{\nu(\Delta').n\langle\text{return } v\rangle?} (\Delta'',\Delta',\Phi \vdash \nu(\Delta_1).(n\langle t'[v/x]\rangle \parallel C''') \dashv \Theta',\Sigma')$$
$$(\Theta'',\Phi \vdash \overline{C''} \dashv \Delta'',\Sigma'') \xrightarrow{\nu(\Delta').n\langle\text{return } v\rangle!} (\Theta'',\Phi \vdash \nu(\Delta_2').(n\langle\text{let } y : U = \text{block in } t''\rangle \parallel \overline{C'''}) \dashv \Delta'',\Delta',\Sigma'')$$

where $\Delta_2 = (\Delta',\Delta_2')$ and moreover

$$n\langle t_1 \bowtie t_2 \rangle \equiv n\langle(\text{let } y : U = \text{block in } t'') \bowtie t'[v/x]\rangle \downarrow_b$$

so by inductive hypothesis

$$(\Delta'',\Phi \vdash C'' \dashv \Theta'',\Sigma'') \xrightarrow{\nu(\Delta').n\langle\text{return } v\rangle?} \xRightarrow{s} (\Delta',\Phi \vdash C' \dashv \Theta',\Sigma')$$
$$(\Theta'',\Phi \vdash \overline{C''} \dashv \Delta'',\Sigma'') \xrightarrow{\nu(\Delta').n\langle\text{return } v\rangle!} \xRightarrow{\overline{s}} (\Theta',\Phi \vdash \overline{C'} \dashv \Delta',\Sigma')$$

and either $C' \downarrow_b$ or $\overline{C'} \downarrow_b$ as required. $\square$

Proof of soundness

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

Proof uppose that Traces(Δ C Θ Traces(Δ C Θ an that we ave (Θ, 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

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

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

$\overline{\Delta \square \varepsilon \quad \text{trace}\, \Theta}$

n

- If $n \notin \mathrm{threads}(s)$ 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 $\nu(\Delta).n\langle \mathrm{call}\, p.l(\vec{n})\rangle?\, s\, \nu(\Theta).n\langle \mathrm{return}\, v\rangle!$
- If n is balanced in s then n is balanced in $\nu(\Theta).n\langle \mathrm{call}\, p.l(\vec{n})\rangle!\, s\, \nu(\Delta).n\langle \mathrm{return}\, v\rangle?$

Define $\mathrm{pop}\,n(s)$ as

- If n is balanced in s then $\mathrm{pop}\,n(s) = \bot$
- If n is balanced in s and $a =$

Proof Easy in uction on s. □

Lemma .

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

Proof e prove t ese propert es s u taneous y by an n uct on on t e en t o s. e on y s ow t e ar u ent or art as art can be s own n a s ar anner. By ana ys s o t e ru es o t e ts we ave

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

ow part t on s nto s s p c n s

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

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

so we have that either

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

or $n \notin \Delta, \Delta(s')$ an n is a fres thread to s'. e can app y the in uctive hypotheses to s' to see that $\Delta \vdash s'$ trace Θ an we cons er pop $n(s')$. $n \in \Delta, \Delta(s')$ an n is fres thread to s' then pop $n(s')$ is necessar y therwise we now that $C' \equiv \nu(\Delta'' . \nu(\Delta''' . n\text{let}\, x \; T =$ block in $t) \mid C''$ an therefore the ast action which cou have occurre at n ust have been an output that is pop $n(s') = \gamma$ In both cases we see that

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

e now that $(\Delta, \Delta(s') \vdash C' \; \Theta, \Theta(s') \xrightarrow{\nu(\Delta' . n\text{call}\, p.l(\vec{v}))?}$ an we now that the s e con itions on the trans tion ru e or $\nu(\Delta' . \gamma)?$ actions uarantees that

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

e a so now that the s e con itions on ru e or ca input actions uarantees that

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

e use this to see that

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

an

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

Last y it is easy to see that

$,\Delta, \Delta(s') ,\Theta, \Theta(s') ,\Delta' \vdash n$ thread

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

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

as require

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

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

$$(\Delta \vdash C$$

$\Delta \; s$ trace Θ

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

$$s = s \; \nu(\Delta \; . n \,\text{call}\, p.l(\vec{v} \; s$$

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

$$\text{pop}\, n(s \; \nu(\Delta \; . n \,\text{call}\, p.l(\vec{v} \; s \; = \nu(\Delta \; . n \,\text{call}\, p.l(\vec{v}$$

ence

$\text{pop}\, n(s = \nu(\Delta \; . n \,\text{call}\, p.l(\vec{v}$

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

$\text{dom}\,(\Theta \; \text{fn}\,(v$

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

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

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

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

w .c by wea en.n .ves us

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

Last y because

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

an

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

we see t at

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

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

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

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

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

as requ.re

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

. Information order on traces

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

$$\Delta \vdash s \sqsubseteq s\, r : \text{trace}\ \Theta$$
$$\Delta \vdash s\gamma ? \sqsubseteq s : \text{trace}\ \Theta$$
$$\Delta \vdash s\gamma ?\gamma\ r \sqsubseteq s\gamma\ \gamma ?r : \text{trace}\ \Theta$$
$$\Delta \vdash s\nu(\Delta .\gamma ?\gamma ?r \sqsubseteq s\nu(\Delta .\gamma ?\gamma ?r : \text{trace}\ \Theta$$
$$\Delta \vdash s\nu(\Theta .\gamma\ \gamma\ r \sqsubseteq s\nu(\Theta .\gamma\ \gamma\ r : \text{trace}\ \Theta$$

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

Proof We write $\Delta \vdash r \sqsubseteq^n s : \text{trace}\ \Theta$ if $\Delta \vdash r \sqsubseteq s : \text{trace}\ \Theta$

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

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

.

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

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

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

$$\mathrm{State}(\Delta \; r \leq s \quad \mathrm{trace}\;\Theta \;\; = (
\\ \quad \mathrm{out}_T = \mathrm{Out}_T(\Delta \; r \leq s \quad \mathrm{trace}\;\Theta \;\; ,
\\ \quad \mathrm{inReturn}_T = \mathrm{InReturn}_T(\Delta \; r \leq s \quad \mathrm{trace}\;\Theta \;\; ,
\\ \quad \mathrm{inCall}_{p.l\,L} = \mathrm{InCall}_{p.l\,L}(\Delta \; r \leq s \quad \mathrm{trace}\;\Theta$$

$$\mathrm{Out}_T(\Delta \; r \leq s \quad \mathrm{trace}\;\Theta \;\; = \lambda(\;\; . (
\\ \quad \text{when } ra \leq s \text{ and } \;\; a = \nu(\Theta' \;\; . n\;\mathrm{call}\, p.l(\vec{v} \;\; \text{ and } \;\; , \Delta, \Theta, \Delta(r \;\; , \Theta(r \;\; , \Theta' \;\; p.l(\vec{v} \;\; U
\\ \qquad \text{if currentthread} = n \text{ then}
\\ \qquad\quad \mathrm{ref.val} \;\; = \text{new State}(\Delta \; ra \leq s \quad \mathrm{trace}\;\Theta \;\; ,
\\ \qquad\quad \mathrm{ref.val.inReturn}_U(p.l(\vec{v} \;\; ,
\\ \qquad\quad \mathrm{ref.val.out}_T(
\\ \quad \text{when } ra \leq s \text{ and } \;\; a = \nu(\Theta' \;\; . n\;\mathrm{return}\, v \;\; \text{ and } \;\; , \Delta, \Theta, \Delta(r \;\; , \Theta(r \;\; , \Theta' \;\; v \;\; T
\\ \qquad \text{if currentthread} = n \text{ then}
\\ \qquad\quad \mathrm{ref.val} \;\; = \text{new State}(\Delta \; ra \leq s \quad \mathrm{trace}\;\Theta \;\; ,
\\ \qquad\quad v
\\ \quad \text{otherwise}
\\ \qquad \text{stop}$$

$$\mathrm{InReturn}_T(\Delta \; r \leq s \quad \mathrm{trace}\;\Theta \;\; = \lambda(x \;\; T \;\; . (
\\ \quad \text{when } ra \leq s \text{ and } \;\; a = \nu(\Delta' \;\; . n\;\mathrm{return}\, v ? \text{ and } \;\; , \Delta, \Theta, \Delta(r \;\; , \Theta(r \;\; , \Delta' \;\; v \;\; T
\\ \qquad \text{if } \Delta, \Theta, \Delta(r \;\; , \Theta(r \;\; (\text{currentthread}, x \;\; = \nu(\Delta' \;\; . (n, v \;\; \text{then}
\\ \qquad\quad \mathrm{ref.val} \;\; = \text{new State}(\Delta \; ra \leq s \quad \mathrm{trace}\;\Theta \;\; ,
\\ \qquad\quad v
\\ \quad \text{otherwise}
\\ \qquad \text{stop}$$

$$\mathrm{InCall}_{p.l\,(\vec{T} \;\to T}(\Delta \; r \leq s \quad \mathrm{trace}\;\Theta \;\; = \lambda(\vec{x} \;\; \vec{T} \;\; . (
\\ \quad \text{when } ra \leq s \text{ and } \;\; a = \nu(\Delta' \;\; . n\;\mathrm{call}\, p.l(\vec{v} \;\; ? \text{ and } \;\; , \Delta, \Theta, \Delta(r \;\; , \Theta(r \;\; , \Delta' \;\; \vec{v} \;\; \vec{T}
\\ \qquad \text{if } \Delta, \Theta, \Delta(r \;\; , \Theta(r \;\; (\text{currentthread}, \vec{x} \;\; = \nu(\Delta' \;\; . (n, \vec{v} \;\; \text{then}
\\ \qquad\quad \mathrm{ref.val} \;\; = \text{new State}(\Delta \; ra \leq s \quad \mathrm{trace}\;\Theta \;\; ,
\\ \qquad\quad \mathrm{ref.val.out}_T(
\\ \quad \text{otherwise}
\\ \qquad \text{stop}$$

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

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

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

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

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

. 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 Θ (=(((

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

2. *If* $C \parallel C \equiv \nu(\vec{n} \; \vec{T}).C$ *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* $C' \equiv C' \parallel C'$.

Proof Proved by induction on the derivation of $C \parallel C$. □

Lemma A. *If* $C \parallel C \equiv C$ *and* $C \xrightarrow{\beta} C'$

- **Case** $(\gamma = \nu(\vec{n} : \vec{T}).n\langle \mathrm{call}\ p.l(\vec{v})\rangle$ and $n \notin \Sigma)$.

  Similar to the previous case.

- **Case** $(\gamma = \nu(\vec{n} : \vec{T}).n\langle \mathrm{return}\ v\rangle)$.

  Since $(\Delta, \Phi \vdash C : \Theta, \Sigma) \xrightarrow{\gamma'} (\Delta', \Phi \vdash C' : \Theta', \Sigma')$ we must have that

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

  Since $(\Theta, \Phi \vdash C : \Delta, \Sigma) \xrightarrow{\gamma} (\Theta', \Phi \vdash C' : \Delta', \Sigma')$ we must have that

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

  We then show that

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

  and that

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

  and also

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

  as required. $\square$

Composition follows by induction on the derivation of $(\Delta, \Phi \vdash C : \Theta, \Sigma) \stackrel{s}{\Longrightarrow} (\Delta', \Phi \vdash C' : \Theta', \Sigma')$ and $(\Theta, \Phi \vdash C : \Delta, \Sigma) \stackrel{s}{\Longrightarrow} (\Theta', \Phi \vdash C' : \Delta', \Sigma')$, and a final use of Lemmas A., A. and A..

A. Decomposition

We show three lemmas, from which Decomposition follows.

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

$$
\begin{aligned}
&(\Delta, \Phi \vdash C : \Theta, \Sigma) \stackrel{s}{\Longrightarrow} (\Delta', \Phi \vdash \nu(\vec{n} : \vec{T}).(C' \mid n\langle \mathrm{let}\ x : T = e\ \mathrm{in}\ t\rangle) : \Theta', \Sigma') \\
&(\Theta, \Phi \vdash C : \Delta, \Sigma) \stackrel{s}{\Longrightarrow} (\Theta', \Phi \vdash C' : \Delta', \Sigma')
\end{aligned}
$$

where:

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

or symmetrically, swapping the roles of C *and* C.

Proof An n uct on on t e er vat on o

$$(C \ /\!\!/\!\backslash\ C \quad \equiv \nu(\vec{n} \quad \vec{T} \quad .(C \ n \ \mathrm{let}\ x \quad T = e \ \mathrm{in}\ t$$

e nterest n case s w en

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

an

$$n \ t \quad v/x \ /\!\!/\!\backslash\ n \ \mathrm{let}\ x \quad T = \mathrm{block\ in}\ t \ \equiv \nu(\vec{n} \quad \vec{T} \quad .(C \ n \ \mathrm{let}\ x \quad T = e \ \mathrm{in}\ t$$

so by e n t on o t e ts an by n uct on we ave

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

an

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

w ere

$$\nu(\Delta', \Theta', \Sigma' \setminus \Delta, \Theta, \Sigma \ .\nu(\vec{n} \quad \vec{T} \quad .(C' \ n \ t \ /\!\!/\!\backslash\ C' \ \equiv \nu(\vec{n} \quad \vec{T} \quad .(C \ n \ t$$

or sy etr ca y as requ re □

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

an so we use t e ax o to et

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

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

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

Moreover since C s write c ose we ust ave t at t e ax o s

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

n w c case

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

w ere we e ne

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

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

e a so ave

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

B. Technical preliminaries

In a co ponent ν(Δ . (*p* *O* C

A *component for* $\Delta \vdash r \leq s$ trace Θ resp. for $\Delta \vdash q \vdash r \leq s$ trace Θ is one of the form

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

where t_n is a thread at n for $\Delta \vdash r \leq s$ trace Θ resp. for $\Delta \vdash q \vdash r \leq s$ trace Θ.

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

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

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

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

Figure 9: Definition of a component for $\Delta \vdash r \leq s$ trace Θ and for $\Delta \vdash q \vdash r \leq s$ trace Θ

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

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

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

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

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

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 $C \equiv \nu(\Theta''').(C \mid \mathrm{ref\ val} = \mathrm{state}_r \mid n\langle \mathrm{let}\ (y : U) = \mathrm{ref.val.out}_U()\ \mathrm{in}\ \mathrm{let}\ (x : T) = \mathrm{return}(y : U)\ \mathrm{in}\ t\rangle)$.

  We have

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

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

  $\stackrel{\beta}{\dashrightarrow}{}^{*} (\Delta' \vdash \nu(\Theta''').(C \mid \mathrm{ref\ val} = \mathrm{state}_r \mid$
  $\quad n\langle \mathrm{ref.val} := \mathrm{new\ State}(\Delta \vdash ra \leq s : \mathrm{trace}\ \Theta); \mathrm{let}\ (y : U) = v\ \mathrm{in}\ \mathrm{let}\ (x : T) = \mathrm{return}(y : U)\ \mathrm{in}\ t\rangle) : \Theta')$

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

  $\stackrel{\beta}{\dashrightarrow}{}^{*} (\Delta' \vdash \nu(\Theta''', \mathrm{state}_{ra} : \mathrm{State}).(C \mid \mathrm{ref\ val} = \mathrm{state}_{ra} \mid \mathrm{state}_{ra}\ \mathrm{State}(\Delta \vdash ra \leq s : \mathrm{trace}\ \Theta) \mid$
  $\quad n\langle \mathrm{let}\ (x : T) = \mathrm{return}(v : U)\ \mathrm{in}\ t\rangle) : \Theta')$

  $\stackrel{a}{\dashrightarrow} (\Delta' \vdash \nu(\mathrm{state}_{ra} : \mathrm{State}).(C \mid \mathrm{ref\ val} = \mathrm{state}_{ra} \mid \mathrm{state}_{ra}\ \mathrm{State}(\Delta \vdash ra \leq s : \mathrm{trace}\ \Theta) \mid$
  $\quad n\langle \mathrm{let}\ (x : T) = \mathrm{block}\ \mathrm{in}\ t\rangle) : \Theta', \Theta''')$

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

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

  We have

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

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

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

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

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

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

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

we have

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

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

$\xrightarrow{\beta}{}^{*} (\Delta', \Delta''' \vdash C \parallel \mathrm{ref}\ \mathrm{val} = \mathrm{state}_r \parallel$
$\quad n\langle \mathrm{ref.val.inReturn}_T(v), t\rangle \parallel \Theta')$

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

$\xrightarrow{\beta}{}^{*} (\Delta', \Delta''' \vdash C \parallel \mathrm{ref}\ \mathrm{val} = \mathrm{state}_r \parallel$
$\quad n\langle \mathrm{ref.val} := \mathrm{new}\ \mathrm{State}(\Delta \vdash ra \leq s : \mathrm{trace}\ \Theta), t\rangle \parallel \Theta')$

$\xrightarrow{\tau} (\Delta', \Delta''' \vdash C\ \nu(\mathrm{state}_{ra} : \mathrm{State}).\mathrm{ref}\ \mathrm{val} = \mathrm{state}_{ra} \parallel \mathrm{state}_{ra}\ \mathrm{State}(\Delta \vdash ra \leq s : \mathrm{trace}\ \Theta) \parallel$
$\quad n\langle t\rangle \parallel \Theta')$

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

Case $a = \nu(\Delta''').\, n\langle \mathrm{call}\, p.l(\vec{v})\rangle$? and $C \equiv C\ \mathrm{ref}\ \mathrm{val} = \mathrm{state}_r \parallel n\langle \mathrm{let}\ x : T = \mathrm{block}\ \mathrm{in}\ t\rangle$

we have

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

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

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

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

$\xrightarrow{\beta}{}^{*} (\Delta', \Delta''' \vdash C \parallel \mathrm{ref}\ \mathrm{val} = \mathrm{state}_r \parallel$
$\quad n\langle \mathrm{ref.val} := \mathrm{new}\ \mathrm{State}(\Delta \vdash ra \leq s : \mathrm{trace}\ \Theta),$
$\quad \mathrm{let}\ y : U = \mathrm{ref.val.out}$

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

$$\xrightarrow{\tau} \nu(\text{state}_{ra}\ T\ \text{State}\ .C\ \text{ref val} = \text{state}_{ra}\ \text{state}_{ra}\ \text{State}(\Delta\ ra \leq s\ \text{trace}\ \Theta\ \ n\ t\ \equiv C$$

where t is a thread at n for Δ $ra \leq s$ trace Θ

By definition C is a component for 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 $\equiv C$

where proj n (q $=$ proj n (r n is output enabled in Δ r trace Θ and t is a return (x T thread at n for Δ $r \leq s$ trace Θ

If Δ $ra \leq s$ trace Θ and $a = \nu(\Theta$. $n$call $p.l(\vec{v}$ then

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

which is a component for Δ q $r \leq s$ trace Θ as required

If Δ $ra \leq s$ trace Θ and $a = \nu(\Theta$. $n$return v then we must have that $r = r\ \nu(\Theta$. $n$call $p.l(\vec{v}$? r where n is balanced in r

- **Case** $(\Delta' \ C\ n \text{let } x\ T = \text{block in } t\ \ \Theta' \xrightarrow{\nu(\Delta''' .n\ \text{call } p.l(\vec{v}) ?} (\Delta', \Delta''' \ C\ n \text{let } y\ U = p.l(\vec{v}) \text{ in let } x\ T = \text{return}(y\ U) \text{ in } t\ \ \Theta'$
w ere proj n $(q$ $=$ proj n $(r$ n s nput 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
e ave

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

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

- **Case** $(\Delta' \ C\ n \text{let } x\ T = \text{block in } t\ \ \Theta' \xrightarrow{\nu(\Delta''' .n\ \text{return } v ?} (\Delta', \Delta''' \ C\ n \text{let } x\ T = v \text{ in } t\ \ \Theta'$
w ere proj n $(q$ $=$ proj n $(r$ n s nput 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
e ave

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

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

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

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

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 and 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 abs n 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 proceses *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.*