A semantics for concurrent permission logic

Stephen Brookes
CMU

Cambridge, March 2006

Traditional logic

Owicki/Gries '76

$\Gamma \vdash \{p\}\ c\ \{q\}$

- Resource-sensitive partial correctness
  - Γ specifies *resources* r_i, *protection lists* X_i, and *invariants* R_i
  - p, q describe *unprotected* variables
- Static constraints guarantee race-freedom

Parallel rule

Owicki/Gries

$$\frac{\Gamma \vdash \{p_1\}\ c_1\ \{q_1\} \qquad \Gamma \vdash \{p_2\}\ c_2\ \{q_2\}}{\Gamma \vdash \{p_1 \wedge p_2\}\ c_1 || c_2\ \{q_1 \wedge q_2\}}$$

provided

$$\mathrm{free}(p_1, q_1) \cap \mathrm{writes}(c_2) = \varnothing$$

$$\mathrm{free}(p_2, q_2) \cap \mathrm{writes}(c_1) = \varnothing$$

$$\mathrm{free}(c_1) \cap \mathrm{writes}(c_2) \subseteq \mathrm{owned}(\Gamma)$$

$$\mathrm{free}(c_2) \cap \mathrm{writes}(c_1) \subseteq \mathrm{owned}(\Gamma)$$

critical variables are protected

Resource rules

Owicki/Gries

$$\frac{\Gamma \vdash \{(p \wedge R) \wedge b\}\ c\ \{q \wedge R\}}{\Gamma, r(X){:}R \vdash \{p\}\ \textbf{with}\ r\ \textbf{when}\ b\ \textbf{do}\ c\ \{q\}}$$

$$\frac{\Gamma, r(X){:}R \vdash \{p\}\ c\ \{q\}}{\Gamma \vdash \{p \wedge R\}\ \textbf{resource}\ r\ \textbf{in}\ c\ \{q \wedge R\}}$$

(subject to static constraints)

Validity
Definition

$\Gamma \vdash \{p\}c\{q\}$ is *valid* iff...

Every finite computation of c
in an environment that respects Γ,
from a state satisfying $p \wedge R_1 \wedge \ldots \wedge R_n$,
respects Γ, is race-free,
and ends in a state satisfying $q \wedge R_1 \wedge \ldots \wedge R_n$

(state = store)

Soundness

- Owicki-Gries logic is sound,
  for simple shared-memory programs
- Every provable program is race-free

Problem

- Owicki-Gries logic is *unsound for pointer programs*

$$\frac{\vdash \{[x]=0\}\ [x]{:=}1\ \{[x]=1\} \qquad \vdash \{[y]=0\}\ [y]{:=}1\ \{[y]=1\}}{\vdash \{[x]=0 \wedge [y]=0\}\ [x]{:=}1 \parallel [y]{:=}1\ \{[x]=1 \wedge [y]=1\}}$$

valid premisses, invalid conclusion

- Static constraints cannot prevent *pointer races*

Concurrent separation logic

O'Hearn '02
Brookes '04

- Combine Owicki-Gries with *separation logic*
  - Let resource invariants be *precise* formulas
  - Static constraints ensure race-freedom for *variables*
  - Use $\star$ to enforce mutual exclusion for *heap*

$$(s,h) \vDash \varphi_1 \star \varphi_2$$
$$\text{iff } \exists h_1 \perp h_2.\ h=h_1 \cup h_2 \ \&$$
$$(s,h_1) \vDash \varphi_1 \ \& \ (s,h_2) \vDash \varphi_2$$

Parallel rule

O'Hearn '02

$$\frac{\Gamma \vdash \{p_1\}\ c_1\ \{q_1\} \qquad \Gamma \vdash \{p_2\}\ c_2\ \{q_2\}}{\Gamma \vdash \{p_1 \star p_2\}\ c_1 \| c_2\ \{q_1 \star q_2\}}$$

$\star$ for $\wedge$

provided

$$\mathrm{free}(p_1, q_1) \cap \mathrm{writes}(c_2) = \varnothing$$

$$\mathrm{free}(p_2, q_2) \cap \mathrm{writes}(c_1) = \varnothing$$

same as before

$$\mathrm{free}(c_1) \cap \mathrm{writes}(c_2) \subseteq \mathrm{owned}(\Gamma)$$

$$\mathrm{free}(c_2) \cap \mathrm{writes}(c_1) \subseteq \mathrm{owned}(\Gamma)$$

Resource rules

O'Hearn '02

$$\frac{\Gamma \vdash \{(p \star R) \wedge b\}\ c\ \{q \star R\}}{\Gamma, r(X){:}R \vdash \{p\}\ \textbf{with}\ r\ \textbf{when}\ b\ \textbf{do}\ c\ \{q\}}$$

$\star$ for $\wedge$

$$\frac{\Gamma, r(X){:}R \vdash \{p\}\ c\ \{q\}}{\Gamma \vdash \{p \star R\}\ \textbf{resource}\ r\ \textbf{in}\ c\ \{q \star R\}}$$

$\star$ for $\wedge$

(subject to static constraints)

Validity

$\Gamma \vdash \{p\}c\{q\}$ is *valid* if:

Every finite computation of c
in an environment that respects Γ,
from a state satisfying $p \star R_1 \star \ldots \star R_n$,
respects Γ, is race-free,
and ends in a state satisfying $q \star R_1 \star \ldots \star R_n$

Can be formalized using
action trace semantics

(state = store + heap)

Ownership transfer

- The logic allows proofs in which heap *ownership* transfers between processes and resources

  - for each available resource, invariant holds separately
  - when *acquiring* a resource, process *claims* ownership of protected variables + sub-heap
  - when *releasing* a resource, process must guarantee that invariant holds separately, and *cedes* ownership

Soundness

Brookes '04

Every provable formula is valid

- Based on *action trace* semantics
  - formalizes notion of *validity*
  - supports rigorous account of ownership transfer

precision plays a crucial role in the soundness proof

Problems

- Concurrent separation logic is too rigid
- Cannot handle concurrent reads of heap cells

$\vdash \{z \mapsto 0\}\ x{:=}[z]\ ||\ y{:=}[z]\ \{z \mapsto 0 \wedge x{=}y{=}0\}$

valid but not provable

$\vdash \{z = 0\}\ x{:=}z\ ||\ y{:=}z\ \{z = 0 \wedge x{=}y{=}0\}$

valid, provable

Reason

- Concurrent separation logic treats store and heap differently
  - store handled in side conditions
  - heap managed in logic, with ★

$$z \mapsto 0 \star z \mapsto 0 \ = \text{false}$$

Concurrent permission logic

Parkinson, Bornat, Calcagno '06

- Blend Owicki-Gries with *permission logic*
- Treat store and heap identically
- Augment state with permissions
- Use a more permissive form of $\star$ to allow concurrent reads but not writes

... no side conditions!
... no protection lists!

Parallel rule

PBC '06

$$\frac{\Gamma \vdash \{p_1\}\ c_1\ \{q_1\} \qquad \Gamma \vdash \{p_2\}\ c_2\ \{q_2\}}{\Gamma \vdash \{p_1 \star p_2\}\ c_1 \| c_2\ \{q_1 \star q_2\}}$$

as before

Where's the side condition?

Resource rules

PBC '06

$$\frac{\Gamma \vdash \{(p \star R) \wedge b\}\ c\ \{q \star R\}}{\Gamma, r{:}R \vdash \{p\}\ \textbf{with}\ r\ \textbf{when}\ b\ \textbf{do}\ c\ \{q\}}$$

as before

$$\frac{\Gamma, r{:}R \vdash \{p\}\ c\ \{q\}}{\Gamma \vdash \{p \star R\}\ \textbf{resource}\ r\ \textbf{in}\ c\ \{q \star R\}}$$

(no need for static constraints)

Validity

$\Gamma \vdash \{p\}c\{q\}$ is *valid* if:

Every finite computation of c
in an environment that respects Γ,
from a state satisfying $p \star R_1 \star \ldots \star R_n$,
respects Γ, is race-free,
and ends in a state satisfying $q \star R_1 \star \ldots \star R_n$

Can also be formalized with action trace semantics

(state = store + heap, with permissions)

Permission transfer

- The logic allows proofs in which *permissions* transfer implicitly between processes and resources
  - for each available resource, invariant holds separately
  - when *acquiring* a resource, process *claims* permissions
  - when *releasing* a resource, process must guarantee that invariant holds separately, and *cedes* permissions

Summary of talk

- Concurrent permission logic is *sound*
- Can use *action trace* semantics
- Soundness proof generalizes earlier proof for concurrent separation logic
- Crucial role of *precision*

Actions

heap actions can be incorporated too

δ	idle
$i{=}v$	read
$i{:=}v$	write
$try(r)$, $acq(r)$, $rel(r)$	resource actions
abort	error

Semantics

- A command denotes a set of *action traces*

$$[\![c]\!] \subseteq Tr$$

- Defined by structural induction on c

$$[\![c_1 ; c_2]\!] = \{ \alpha_1 \, \alpha_2 \mid \alpha_1 \in [\![c_1]\!], \; \alpha_2 \in [\![c_2]\!] \}$$

concatenation

$$[\![c_1 \| c_2]\!] = \bigcup \{ \alpha_1 \| \alpha_2 \mid \alpha_1 \in [\![c_1]\!], \; \alpha_2 \in [\![c_2]\!] \}$$

resource-sensitive, race-detecting,
fair interleaving

Permissions

$(\mathcal{P}, \otimes, \top)$

- partial commutative cancellative semi-group
- $\top \otimes p$ undefined
- $p \otimes p' \neq p$

$\top$ allows read/write

$p \neq \top$ allows read permission

+ other properties,
e.g. *divisibility*
when appropriate

Fractional permissions

- $\mathcal{P} = (0,1]$
- $p \otimes p' = p + p'$ if in $(0,1]$
- $\top = 1$

Stacks

$$s : S = \mathrm{Ide} \rightharpoonup_{fin} V \times \mathcal{P}$$

- Map program variables to (v, p) pairs
- $s \star s'$ combines bindings and permissions, when s and s' are *compatible*
- Write $s \ \# \ s'$ when compatible

Stacks

- $s \# s'$ iff $\forall i, v, p, v', p'$.

  if $s(i)=(v, p)$ & $s'(i)=(v', p')$

  then $v=v'$ & $p \# p'$

- $s \star s' =_{def} s \backslash dom(s') \cup s' \backslash dom(s)$

  $\cup \{(i, (v, p \otimes p')) \mid s(i)=(v, p) \;\&\; s'(i)=(v, p')\}$

Logical variables

- Used in the logic to link pre- and post-conditions
- Do not appear in programs

X, Y are logical variables
x, y are program variables

Interpretations

- Map logical variables to logical values
  - integer variables to integers
  - permission variables to permissions

States

state = stack + interpretation

- $\sigma = (s, i)$
- $(s, i) \ \sharp \ (s', i')$ iff $s \ \sharp \ s'$ & $i = i'$
- $(s, i) \star (s', i) = (s \star s', i)$

State formulas

$$
\begin{aligned}
\varphi \;::=\; & \mathbf{emp} \\
\mid\; & \mathrm{Own}_{p}(x) \\
\mid\; & E_1 = E_2 \\
\mid\; & \neg\varphi \\
\mid\; & \varphi_1 \star \varphi_2 \\
\mid\; & \varphi_1 \wedge \varphi_2 \\
& \varphi_1 \Rightarrow \varphi_2 \\
\mid\; & \exists X.\varphi
\end{aligned}
$$

Satisfaction

$(s,i) \vDash \mathbf{emp}$ iff $s=\{\ \}$

$(s,i) \vDash \mathrm{Own}_p(x)$ iff $\exists v.\ s=\{(x, (v, |p|i))\}$

$\sigma \vDash \varphi_1 \star \varphi_2$ iff

$\exists \sigma_1,\sigma_2.\ \sigma = \sigma_1 \star \sigma_2\ \&\ \sigma_1 \vDash \varphi_1\ \&\ \sigma_2 \vDash \varphi_2$

$\sigma \vDash E_1 = E_2$ iff

$|E_1|\sigma = |E_2|\sigma\ \&\ \mathrm{free}(E_1,E_2) \subseteq \mathrm{dom}(\sigma)$

Examples

$\mathrm{Own}_p(x) \star \mathrm{Own}_q(x)$

true in (s,i)

iff

$p \# q$ & $\exists v.\ s = \{(x, (v, |p \otimes q| i))\}$

$x=3$

true in (s,i)

iff

$\exists p.\ (x, (3, p)) \in s$

Precision

ϑ is *precise* iff for all σ there is at most one pair (σ_1,σ_2) such that $\sigma = \sigma_1 \star \sigma_2$ and $\sigma_1 \vDash \vartheta$

emp, $\mathrm{Own}_p(x)$ are precise

if ϑ_1, ϑ_2 are precise, so are

$$\vartheta_1 \star \vartheta_2 , \quad (B \wedge \vartheta_1) \vee (\neg B \wedge \vartheta_2)$$

Ownership claims

- Formulas of the form

$$\mathrm{Own}_{p_1}(x_1) \star \ldots \star \mathrm{Own}_{p_k}(x_k)$$

(always precise!)

Program formulas

$\Gamma \vdash_{vr} \{\Phi\}c\{\Psi\}$

no protection lists

- Γ of form $r_1 : \vartheta_1 , \ldots, r_k : \vartheta_k$
- $\vartheta_1 , \ldots, \vartheta_k$ precise
- $r_1 , \ldots, r_k$ distinct
- Φ , Ψ arbitrary state formulas

no static constraints

SKIP

$$\frac{}{\Gamma \vdash_{vr} \{\varphi\}\ \mathbf{skip}\ \{\varphi\}}$$

no static constraint

ASSIGNMENT

not the usual substitution rule!

$$\Gamma \vdash_{vr} \{\mathrm{Own}_{\top}(x) \star O \wedge X{=}e\}\ x{:=}e\ \{\mathrm{Own}_{\top}(x) \star O \wedge x{=}X\}$$

note how
permission constraints
are expressed for e, x

O ranges over *ownership claims*

SEQUENCING

$$\frac{\Gamma \vdash_{vr} \{\varphi\}\ c_1\ \{\psi\} \qquad \Gamma \vdash_{vr} \{\psi\}\ c_2\ \{\xi\}}{\Gamma \vdash_{vr} \{\varphi\}\ c_1;c_2\ \{\xi\}}$$

as before

PARALLEL

$$\frac{\Gamma \vdash_{vr} \{\varphi_1\}\ c_1\ \{\psi_1\} \qquad \Gamma \vdash_{vr} \{\varphi_2\}\ c_2\ \{\psi_2\}}{\Gamma \vdash_{vr} \{\varphi_1 \star \varphi_2\}\ c_1 \| c_2\ \{\psi_1 \star \psi_2\}}$$

no static constraints

IF and WHILE

$$\frac{\phi \Rightarrow b{=}b \quad \Gamma \vdash_{vr} \{\phi \wedge b\}\ c_1\ \{\psi\} \quad \Gamma \vdash_{vr} \{\phi \wedge \neg b\}\ c_2\ \{\psi\}}{\Gamma \vdash_{vr} \{\phi\}\ \textbf{if}\ b\ \textbf{then}\ c_1\ \textbf{else}\ c_2\ \{\psi\}}$$

$$\frac{\phi \Rightarrow b{=}b \quad \Gamma \vdash_{vr} \{\phi \wedge b\}\ c\ \{\phi\}}{\Gamma \vdash_{vr} \{\phi\}\ \textbf{while}\ b\ \textbf{do}\ c\ \{\phi \wedge \neg b\}}$$

extra premiss ensures
permission for b

REGION

$$\frac{\varphi \star \theta \Rightarrow b{=}b \qquad \Gamma \vdash_{vr} \{(\varphi \star \theta) \wedge b\}\ c\ \{\psi \star \theta\}}{\Gamma, r{:}\theta \vdash_{vr} \{\varphi\}\ \mathbf{with}\ r\ \mathbf{when}\ b\ \mathbf{do}\ c\ \{\psi\}}$$

extra premiss implies
permission for b

RESOURCE

$$\frac{\Gamma, r{:}\theta \vdash_{vr} \{\varphi\}\ c\ \{\psi\}}{\Gamma \vdash_{vr} \{\varphi \star \theta\}\ \textbf{resource}\ r\ \textbf{in}\ c\ \{\psi \star \theta\}}$$

as before

CHANGE of BOUND RESOURCE

$$\frac{\Gamma \vdash_{vr} \{\phi\}\ \textbf{resource}\ r'\ \textbf{in}\ [r'/r]c\ \{\psi\}}{\Gamma \vdash_{vr} \{\phi\}\ \textbf{resource}\ r\ \textbf{in}\ c\ \{\psi\}}$$

provided r' *not free in* c

LOCAL

$$\frac{\Gamma \vdash_{vr} \{\mathrm{Own}_{\top}(x') \star \phi\}\ [x'/x]c\ \{\mathrm{Own}_{\top}(x') \star \psi\}}{\Gamma \vdash_{vr} \{\phi\}\ \textbf{local}\ x\ \textbf{in}\ c\ \{\psi\}}$$

provided x' not free in Γ, ϕ, ψ, c

FRAME

$$\frac{\Gamma \vdash_{vr} \{\phi\}\ c\ \{\psi\}}{\Gamma \vdash_{vr} \{\phi \star \vartheta\}\ c\ \{\psi \star \vartheta\}}$$

no static constraints

EXISTS

$$\frac{\Gamma \vdash_{vr} \{\varphi\}\ c\ \{\psi\}}{\Gamma \vdash_{vr} \{\exists X.\ \varphi\}\ c\ \{\exists X.\ \psi\}}$$

X *a logical variable*

CONSEQUENCE

$$\frac{\varphi' \Rightarrow \varphi \quad \Gamma \vdash_{vr} \{\varphi\}\ c\ \{\psi\} \quad \psi \Rightarrow \psi' \quad \Gamma \Leftrightarrow \Gamma'}{\Gamma' \vdash_{vr} \{\varphi'\}\ c\ \{\psi'\}}$$

as before

AUXILIARY VARIABLES

$$\frac{\Gamma \vdash_{vr} \{\varphi \star \mathrm{Own}_{\top}(A)\}\ \mathrm{c}\ \{\psi \star \mathrm{Own}_{\top}(A)\}}{\Gamma \vdash_{vr} \{\varphi\}\ \mathrm{c} \backslash A\ \{\psi\}}$$

provided A auxiliary for c

and no variable in A is free in Γ, φ, ψ

A DERIVED RULE

$$\frac{}{\Gamma \vdash_{vr} \{\Phi\}\ x{:=}e\ \{\Phi \wedge x{=}e\}}$$

if x *not free in* e

where Φ is

$$\mathrm{Own}_{\top}(x) \star \mathrm{Own}_{p_1}(x_1) \star \ldots \star \mathrm{Own}_{p_k}(x_k)$$

and $\mathrm{free}(e) = \{x_1, \ldots, x_k\}$

Example

concurrent reads

$$\vdash_{vr} \{\mathrm{Own}_{\top}(x) \star \mathrm{Own}_{\top}(y) \star \mathrm{Own}_{q}(z)\}$$
$$x{:=}z \parallel y{:=}z$$
$$\{\mathrm{Own}_{\top}(x) \star \mathrm{Own}_{\top}(y) \star \mathrm{Own}_{q}(z) \wedge x=y=z\}$$

need total permission for x,y
\+ any permission for z

Example

race condition

$$\vdash_{vr} \{\mathrm{Own}_{\top}(x) \star \mathrm{Own}_{\top}(x)\}$$
$$x{:=}x{+}1 \;||\; x{:=}x{+}1$$
$$\{\mathrm{Own}_{\top}(x) \star \mathrm{Own}_{\top}(x)\}$$

valid, provable

vacuous

Example

distributed counter

Let $p_1 \otimes q_1 = p_2 \otimes q_2 = \top$

$$\Gamma \;=\; r\!: \mathrm{Own}_{\top}(x) \star \mathrm{Own}_{p1}(x_1) \star \mathrm{Own}_{p2}(x_2) \;\wedge\; x = x_1 + x_2$$

$$\Gamma \vdash_{vr} \{\mathrm{Own}_{q1}(x_1) \star \mathrm{Own}_{q2}(x_2)\}$$

$$\textbf{with } r \textbf{ do } (x := x+1;\; x_1 := x_1+1)$$

$$\|\ \textbf{with } r \textbf{ do } (x := x+1;\; x_2 := x_2+1)$$

$$\{\mathrm{Own}_{q1}(x_1) \star \mathrm{Own}_{q2}(x_2)\}$$

using PAR, REGION

Example

distributed counter

$$\vdash_{vr} \{\mathrm{Own}_{\top}(x,x_1,x_2) \wedge x=x_1+x_2\}$$

resource r **in**

with r **do** $(x:=x+1;\ x_1:=x_1+1)$

|| **with** r **do** $(x:=x+1;\ x_2:=x_2+1)$

$$\{\mathrm{Own}_{\top}(x,x_1,x_2) \wedge x=x_1+x_2\}$$

by RESOURCE rule

Example

distributed counter

$\vdash_{vr} \{(\mathrm{Own}_\top(x) \wedge x=0) \star \mathrm{Own}_\top(x_1,x_2)\}$

$x_1{:=}0;\ x_2{:=}0;$

resource r **in**

with r **do** $(x{:=}x+1;\ x_1{:=}x_1+1)$

|| **with** r **do** $(x{:=}x+1;\ x_2{:=}x_2+1)$

$\{(\mathrm{Own}_\top(x) \wedge x=2) \star \mathrm{Own}_\top(x_1,x_2)\}$

by SEQ rule and CONSEQUENCE

Example

distributed counter

$\vdash_{vr}$ {$\mathrm{Own}_{\top}(x) \wedge x=0$}
resource r **in**
with r **do** x:=x+1
|| **with** r **do** x:=x+1
{$\mathrm{Own}_{\top}(x) \wedge x=2$}

by AUX rule

Intuition

- Rules designed to ensure writes only with *total* permission, reads with *any* permission
- Permissions transfer implicitly on acquiring and releasing resources
- Old side conditions absorbed into the permission calculus

Validity

$\Gamma \vdash_{vr} \{\Phi\}c\{\Psi\}$ is *valid* iff

For all $\alpha \in [\![c]\!]$, $\forall \sigma, \sigma'$.

if $\sigma \vDash \Phi$ and $\sigma \overset{\alpha}{\underset{\Gamma}{\Rightarrow}} \sigma'$

then $\sigma' \vDash \Psi$

interactive computation
in environment respecting Γ

Logical enabling

$$(\sigma, A) \underset{\Gamma}{\overset{\alpha}{\Rightarrow}} (\sigma', A')$$

- When a process with resources A, in "local" state σ, can do α
- Assumes environment that respects Γ
- Causes abort if α exceeds permissions, breaks an invariant, or produces runtime error

Logical enabling

READ

$$(\sigma,A) \overset{x=v}{\underset{\Gamma}{\Rrightarrow}} (\sigma,A) \qquad \text{if } \exists p.\ \sigma(x)=(v,p)$$

$$(\sigma,A) \overset{x=v}{\underset{\Gamma}{\Rrightarrow}} \text{abort} \qquad \text{if } x \notin dom(\sigma)$$

WRITE

$$(\sigma,A) \overset{x:=v}{\underset{\Gamma}{\Rrightarrow}} ([\sigma|x:(v,\top)],A) \qquad \text{if } \exists v_0.\ \sigma(x)=(v_0,\top)$$

$$(\sigma,A) \overset{x:=v}{\underset{\Gamma}{\Rrightarrow}} \text{abort} \qquad \text{otherwise}$$

Logical enabling

when acquiring r, assume invariant holds, claim extra state

ACQUIRE

$$(\sigma, A) \overset{acq(r)}{\underset{\Gamma}{\Rightarrow}} (\sigma \star \sigma', A \cup \{r\})$$

$$\text{if } r \notin A,\ r : \vartheta \in \Gamma,\ \sigma \sharp \sigma',\ \sigma' \vDash \vartheta$$

Logical enabling

when releasing r, ensure invariant holds, relinquish claim

RELEASE

$$(\sigma, A) \overset{\mathrm{rel}(r)}{\underset{\Gamma}{\Rightarrow}} (\sigma_1, A - \{r\})$$

if $r \in A$, $r{:}\,\vartheta \in \Gamma$, $\sigma = \sigma_1 \star \sigma_2$, $\sigma_2 \models \vartheta$

$$(\sigma, A) \overset{\mathrm{rel}(r)}{\underset{\Gamma}{\Rightarrow}} \text{abort}$$

if $r \in A$, $r{:}\,\vartheta \in \Gamma$,

$\forall\, \sigma_1 \# \sigma_2.\ (\sigma = \sigma_1 \star \sigma_2 \text{ implies } \sigma_2 \models \neg\vartheta)$

Theorem

Every provable formula is valid

- Each inference rule preserves validity
- Key lemma: parallel decomposition

Parallel decomposition

Let $\alpha \in \alpha_1 || \alpha_2$ and $\sigma = \sigma_1 \star \sigma_2$

If $\sigma \overset{\alpha}{\underset{\Gamma}{\Rightarrow}}$ abort then $\sigma_1 \overset{\alpha_1}{\underset{\Gamma}{\Rightarrow}}$ abort or $\sigma_2 \overset{\alpha_2}{\underset{\Gamma}{\Rightarrow}}$ abort

If $\sigma \overset{\alpha}{\underset{\Gamma}{\Rightarrow}} \sigma'$ then

$\sigma_1 \overset{\alpha_1}{\underset{\Gamma}{\Rightarrow}}$ abort

or $\sigma_2 \overset{\alpha_2}{\underset{\Gamma}{\Rightarrow}}$ abort

or $\exists\, \sigma_1', \sigma_2'.\ \sigma' = \sigma_1' \star \sigma_2'$ &

$\sigma_1 \overset{\alpha_1}{\underset{\Gamma}{\Rightarrow}} \sigma_1'$ & $\sigma_2 \overset{\alpha_2}{\underset{\Gamma}{\Rightarrow}} \sigma_2'$

Race-freedom

Validity of $\Gamma \vdash_{vr} \{\Phi\}c\{\Psi\}$ implies

For all $\alpha \in [[c]], \forall\sigma, \sigma'$.

if $\sigma \models \Phi \star inv(\Gamma)$ and $\sigma \overset{\alpha}{\Rightarrow} \sigma'$

then $\sigma' \models \Psi \star inv(\Gamma)$

interference-free computation

... NO RACES

References

- Brookes '04 *A semantics for concurrent separation logic*
  CONCUR 2004
- O'Hearn '04 *Resources, concurrency, and local reasoning*
  CONCUR 2004
- O'Hearn '02 *Notes on separation logic for shared-variable concurrency*
  Unpublished manuscript
- Reynolds '02 *Separation logic: a logic for shared mutable data structures*
  LICS 2002

Thought for the Day