A semantics for concurrent permission logic

Stephen Brookes CMU

Cambridge, March 2006

Traditional logic



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

Resource-sensitive partial correctness

 F 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

$\Gamma \vdash \{p_1\} c_1 \{q_1\} \quad \Gamma \vdash \{p_2\} c_2 \{q_2\}$

 $\Gamma \vdash \{p_1 \land p_2\} c_1 || c_2 \{q_1 \land q_2\}$

provided $free(p_{1},q_{1}) \cap writes(c_{2}) = \emptyset$ $free(p_{2},q_{2}) \cap writes(c_{1}) = \emptyset$ $free(c_{1}) \cap writes(c_{2}) \subseteq owned(\Gamma)$ $free(c_{2}) \cap writes(c_{1}) \subseteq owned(\Gamma)$

Resource rules



$\Gamma \vdash \{(p \land R) \land b\} c \{q \land R\}$

 Γ , r(X):R \vdash {p} with r when b do c {q}

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

(subject to static constraints)



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

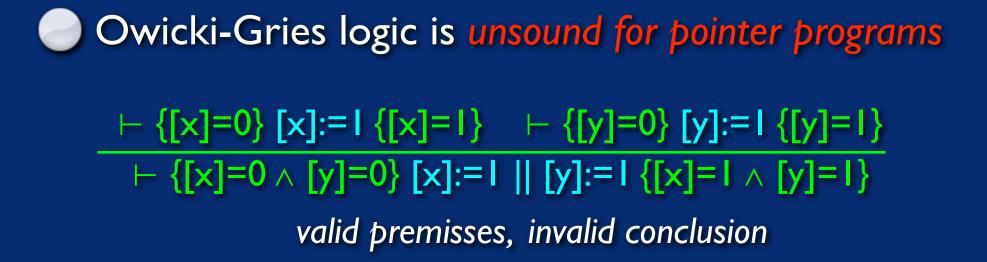
Every finite computation of c in an environment that respects Γ, from a state satisfying p∧R1∧...∧Rn, respects Γ, is race-free, and ends in a state satisfying q∧R1∧...∧Rn

Soundness

Owicki-Gries logic is sound, for simple shared-memory programs

Every provable program is race-free









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

 $(s,h) \vDash \varphi_1 \bigstar \varphi_2$ 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

$\Gamma \vdash \{p_1\} c_1 \{q_1\} \quad \Gamma \vdash \{p_2\} c_2 \{q_2\}$

 $\Gamma \vdash \{p_1 \not \approx p_2\} c_1 \|c_2 \{q_1 \not \approx q_2\}$



provided $free(p_1,q_1) \cap writes(c_2) = \emptyset$ $free(p_2,q_2) \cap writes(c_1) = \emptyset$ $free(c_1) \cap writes(c_2) \subseteq owned(\Gamma)$ $free(c_2) \cap writes(c_1) \subseteq owned(\Gamma)$

Resource rules



$\Gamma \vdash \{(p \bigstar R) \land b\} c \{q \bigstar R\}$ $\overline{\Gamma, r(X): R \vdash \{p\} \text{ with } r \text{ when } b \text{ do } c \{q\}}$



 $\Gamma, r(X): \mathbb{R} \vdash \{p\} \in \{q\}$ $\Gamma \vdash \{p \mathsf{K}\} \text{ resource } r \text{ in } c \{q \mathsf{K}\}$



(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*R *..*R_n, respects Γ, is race-free, and ends in a state satisfying q*R *...*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





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

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

valid but not provable

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

valid, provable



Concurrent separation logic treats store and heap differently

Store handled in side conditions
heap managed in logic, with \bigstar

 $z \mapsto 0 \star z \mapsto 0 = false$

Concurrent permission logic

Parkinson, Bornat, Calcagno '06

to

Blend Owicki-Gries with permission logic Treat store and heap identically Augment state with permissions Use a more permissive form of \star allow concurrent reads but not writes ... no side conditions! ... no protection lists!

Parallel rule



$\Gamma \vdash \{p_1\} c_1 \{q_1\} \quad \Gamma \vdash \{p_2\} c_2 \{q_2\}$

 $\Gamma \vdash \{\mathbf{p}_1 \neq \mathbf{p}_2\} \mathbf{c}_1 || \mathbf{c}_2 \{\mathbf{q}_1 \neq \mathbf{q}_2\}$









Γ ⊢ {(**p**≭R)∧b} c {**q**≭R}

as before

[Γ, r:**R** ⊢ {**p**} **with r when b do c** {**q**}

$\Gamma, r: \mathbb{R} \vdash \{p\} \in \{q\}$ $\Gamma \vdash \{p \neq \mathbb{R}\} \text{ resource } r \text{ in } c \{q \neq \mathbb{R}\}$

(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*R₁*..*R_n, respects Γ, is race-free, and ends in a state satisfying q*R₁*..*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





Concurrent permission logic is sound

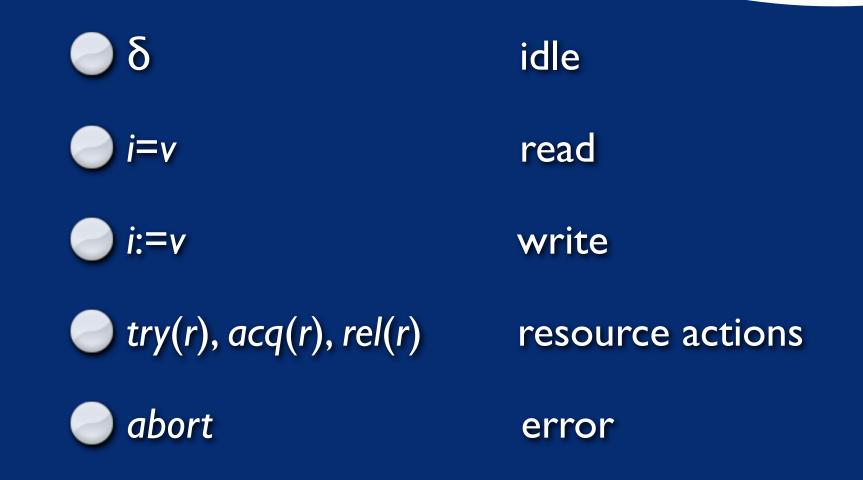
Can use action trace semantics

Soundness proof generalizes earlier proof for concurrent separation logic

Crucial role of precision



heap actions can be incorporated too



Semantics

A command denotes a set of action traces [[c]] ⊆ Tr Defined by structural induction on c [[c1;c2]] = { α₁ α₂ | α₁ ∈ [[c1]], α₂ ∈ [[c2]] }

concatenation

 $\llbracket c_1 \| c_2 \rrbracket = \bigcup \{ \alpha_1 \| \alpha_2 \mid \alpha_1 \in \llbracket c_1 \rrbracket, \alpha_2 \in \llbracket c_2 \rrbracket \}$

resource-sensitive, race-detecting, fair interleaving

Permissions

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

partial commutative cancellative semi-group

 $\top \otimes p$ undefined

T allows read/write $p \neq T$ allows read permission

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

Fractional permissions

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



$s: S = Ide \longrightarrow_{fin} V \times P$

Map program variables to (v, p) pairs

 $s \star s'$ combines bindings and permissions, when s and s' are compatible



 \bigcirc Write s \ddagger s' when compatible



s ♯ s' iff ∀i, v, p, v', p'. if s(i)=(v, p) & s'(i)=(v', p') then v=v' & p # p'

s ★ s' =_{def} s\dom(s') ∪ s'\dom(s) ∪ {(i, (v, $p \otimes p'$)) | 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



state = stack + interpretation

• $\sigma = (s, i)$ • $(s, i) \ddagger (s', i') \text{ iff } s \ddagger s' \& i = i'$

(s, i) \star (s', i) = (s \star s', i)

$\varphi ::= emp$ $| Own_{p}(x)$ $| E_{1}=E_{2}$ $| \neg \varphi$ $| \varphi_{1} \neq \varphi_{2}$ $| \varphi_{1} \land \varphi_{2}$ $| \varphi_{1} \Rightarrow \varphi_{2}$ $| \exists X.\varphi$

State formulas

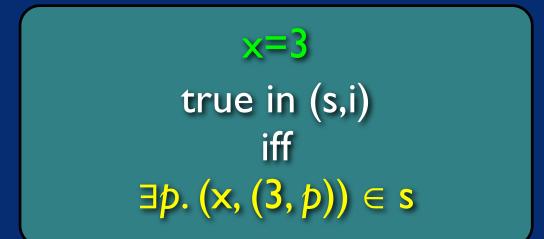
Satisfaction

$(s,i) \models emp \text{ iff } s=\{\}$ $(s,i) \models Own_{p}(x) \text{ iff } \exists v. s = \{(x, (v, |p|i))\}$ $\sigma \models \varphi_1 \star \varphi_2$ iff $\exists \sigma_1, \sigma_2, \sigma = \sigma_1 \star \sigma_2 \& \sigma_1 \models \varphi_1 \& \sigma_2 \models \varphi_2$ $\sigma \models E_1 = E_2$ iff $|\mathsf{E}_1|\sigma = |\mathsf{E}_2|\sigma \& \mathsf{free}(\mathsf{E}_1,\mathsf{E}_2) \subseteq \mathsf{dom}(\sigma)$

Examples

$Own_p(\mathbf{x}) * Own_q(\mathbf{x})$

true in (s,i) iff **p#q & ∃v. s={(x, (v, |p⊗q|i))}**





9 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 $\sigma_1 \models 9$

emp, $Own_p(x)$ are precise

if ϑ_1, ϑ_2 are precise, so are $\vartheta_1 \star \vartheta_2$, $(B \land \vartheta_1) \lor (\neg B \land \vartheta_2)$

Ownership claims

Formulas of the form

$Own_{P_i}(x_1) \star ... \star Own_{P_k}(x_k)$

(always precise!)

Program formulas

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

 \bigcirc Γ of form $r_1: \vartheta_1, ..., r_k: \vartheta_k$

 $\bigcirc 9_1, ..., 9_k$ precise

─ r₁,..., r_k distinct

 $\bigcirc \Phi$, Ψ arbitrary state formulas

no static constraints

no protection lists



$\Gamma \vdash_{vr} \{\phi\} \text{ skip } \{\phi\}$

no static constraint

ASSIGNMENT

not the usual substitution rule!

$\Gamma \vdash_{vr} \{Own_{\top}(x) \bigstar O \land X=e\} x:=e \{Own_{\top}(x) \bigstar O \land x=X\}$

note how permission constraints are expressed for e, x

• ranges over ownership claims

SEQUENCING

$\Gamma \vdash_{vr} \{\phi\} c_1 \{\psi\} \quad \Gamma \vdash_{vr} \{\psi\} c_2 \{\xi\}$

$\Gamma \vdash_{vr} \{\phi\} c_1; c_2 \{\xi\}$

as before



$\Gamma \vdash_{vr} \{\phi_1\} c_1 \{\psi_1\} \quad \Gamma \vdash_{vr} \{\phi_2\} c_2 \{\psi_2\}$

 $\Gamma \vdash_{vr} \{ \phi_1 \star \phi_2 \} c_1 \| c_2 \{ \psi_1 \star \psi_2 \}$

no static constraints

IF and WHILE

$\varphi \Rightarrow b=b \quad \Gamma \vdash_{vr} \{\varphi \land b\} c_1 \{\psi\} \quad \Gamma \vdash_{vr} \{\varphi \land \neg b\} c_2 \{\psi\}$

 $\Gamma \vdash_{vr} \{ \phi \}$ if b then c_1 else $c_2 \{ \psi \}$

 $\varphi \Rightarrow b=b \quad \Gamma \vdash_{vr} \{\varphi \land b\} c \{\varphi\}$

 $\Gamma \vdash_{vr} \{ \phi \}$ while b do c $\{ \phi \land \neg b \}$

extra premiss ensures permission for b



$\phi \star \theta \Rightarrow b = b$ $\Gamma \vdash_{vr} \{(\phi \star \theta) \land b\} c \{\psi \star \theta\}$

Γ , r:θ $\vdash_{vr} {\phi}$ with r when b do c { ψ }

extra premiss implies permission for b



$\Gamma, r:θ ⊢_{vr} {φ} c {ψ}$ $F, r:θ ⊢_{vr} {φ} c {ψ}$ $F ⊢_{vr} {φ*θ} resource r in c {ψ*θ}$

as before

CHANGE of BOUND RESOURCE

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

$\Gamma \vdash_{vr} \{ \phi \}$ resource r in c $\{ \psi \}$

provided r' not free in c



$\Gamma \vdash_{vr} \{ Own_{\top}(x') \neq \varphi \} [x'/x]c \{ Own_{\top}(x') \neq \psi \}$ $\Gamma \vdash_{vr} \{ \varphi \} \text{ local } x \text{ in } c \{ \psi \}$

provided x' not free in Γ , ϕ , ψ , c



Γ⊢_{vr} {φ} c {ψ} Γ⊢_{vr} {φ*θ} c {ψ*θ}

no static constraints



$\Gamma \vdash_{vr} {\phi} c {\psi}$ $\Gamma \vdash_{vr} {\exists X. φ} c {\exists X. ψ}$

X a logical variable

CONSEQUENCE

$\begin{array}{ll} \phi' \Rightarrow \phi & \Gamma \vdash_{vr} \{\phi\} \ \mathsf{c} \{\psi\} & \psi \Rightarrow \psi' & \Gamma \Leftrightarrow \Gamma' \\ \\ & \Gamma' & \vdash_{vr} \{\phi'\} \ \mathsf{c} \{\psi'\} \end{array}$

as before

AUXILIARY VARIABLES

$\frac{\Gamma \vdash_{vr} \{\phi \bigstar \mathsf{Own}_{\top}(A)\} \mathsf{c} \{\psi \bigstar \mathsf{Own}_{\top}(A)\}}{\Gamma \vdash_{vr} \{\phi\} \mathsf{c} \land A \{\psi\}}$

provided A auxiliary for c and no variable in A is free in Γ, ϕ, ψ

A DERIVED RULE

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

if x not free in e

where Φ is $Own_{\top}(x) * Own_{P_{1}}(x_{1}) * \dots * Own_{P_{k}}(x_{k})$ and free(e) = {x_{1}, ..., x_{k}}



concurrent reads

$$\vdash_{vr} \{ Own_{\top}(x) * Own_{\top}(y) * Own_{q}(z) \}$$

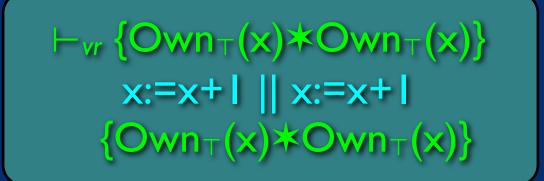
$$x:=z \parallel y:=z$$

$$\{ Own_{\top}(x) * Own_{\top}(y) * Own_{q}(z) \land x=y=z \}$$

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



race condition



valid, provable

vacuous



Let $p_1 \otimes q_1 = p_2 \otimes q_2 = \top$ $\Gamma = r: Own_{\top}(x) * Own_{p_1}(x_1) * Own_{p_2}(x_2) \land x = x_1 + x_2$

$$\begin{split} \Gamma \vdash_{vr} \{ Own_{q1}(x_1) * Own_{q2}(x_2) \} \\ with r do (x:=x+1; x_1:=x_1+1) \\ || with r do (x:=x+1; x_2:=x_2+1) \\ \{ Own_{q1}(x_1) * Own_{q2}(x_2) \} \end{split}$$

using PAR, REGION



 $\vdash_{\text{W}} \{ Own_{T}(x,x_{1},x_{2}) \land x \equiv x_{1} + x_{2} \}$ **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) || **with** r **do** (x:=x+1; x_{2}:=x_{2}+1) {Own_{T}(x,x_{1},x_{2}) \land x \equiv x_{1} + x_{2} }

by **RESOURCE** rule



Fyr {(Own⊤(x) ∧ x=0)*Own⊤(x1,x2))}
x1:=0; x2:=0;
resource r in
with r do (x:=x+1; x1:=x1+1)
|| with r do (x:=x+1; x2:=x2+1)
{(Own⊤(x) ∧ x=2)*Own⊤(x1,x2)}

by seq rule and consequence



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



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

interactive computation in environment respecting Γ

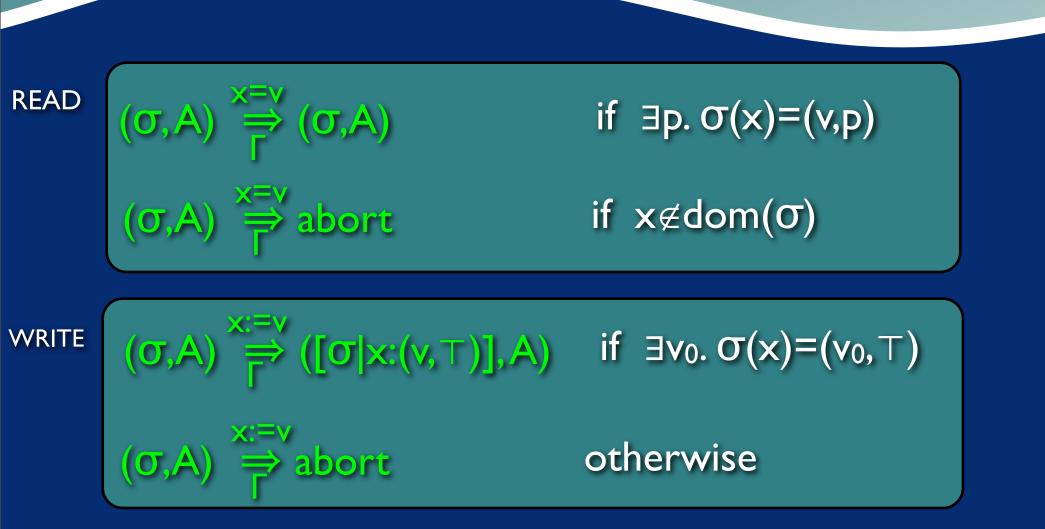
For all $\alpha \in [[c]], \forall \sigma, \sigma'$. if $\sigma \models \Phi$ and $\sigma \stackrel{\alpha}{\Rightarrow} \sigma'$ then $\sigma' \models \Psi$

 $(\sigma, A) \stackrel{\alpha}{\Rightarrow} (\sigma', A')$

When a process with resources A, in "local" state σ , can do α

 \triangleright Assumes environment that respects Γ

Causes abort if α exceeds permissions, breaks an invariant, or produces runtime error



when acquiring r, assume invariant holds, claim extra state

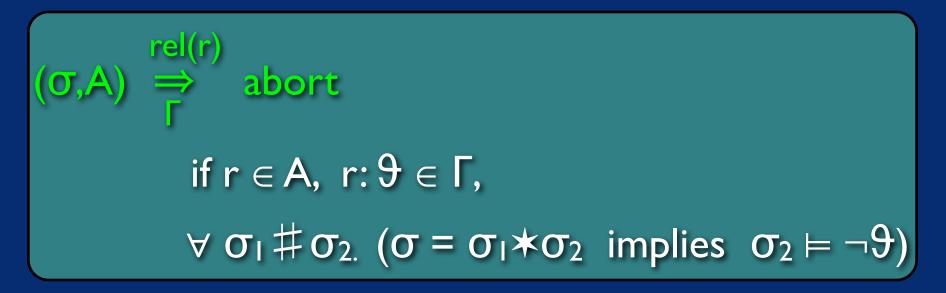
ACQUIRE

$\begin{array}{c} \overset{\text{acq(r)}}{(\sigma,A)} \xrightarrow{\sigma} (\sigma \ast \sigma', A \cup \{r\}) \\ \text{if } r \not\in A, r: \vartheta \in \Gamma, \sigma \ddagger \sigma', \sigma' \models \vartheta \end{array}$

when releasing r, ensure invariant holds, relinquish claim

RELEASE







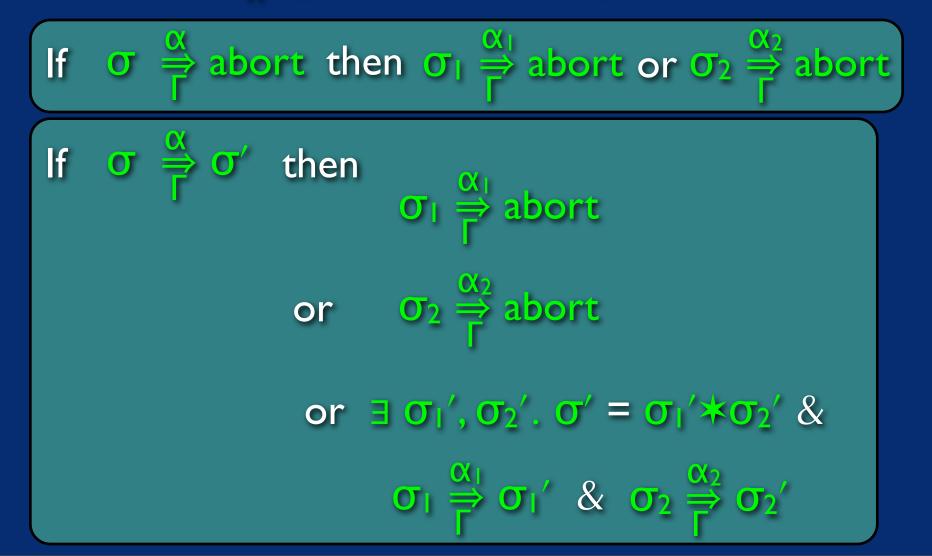
Every provable formula is valid

Each inference rule preserves validity

Key lemma: parallel decomposition

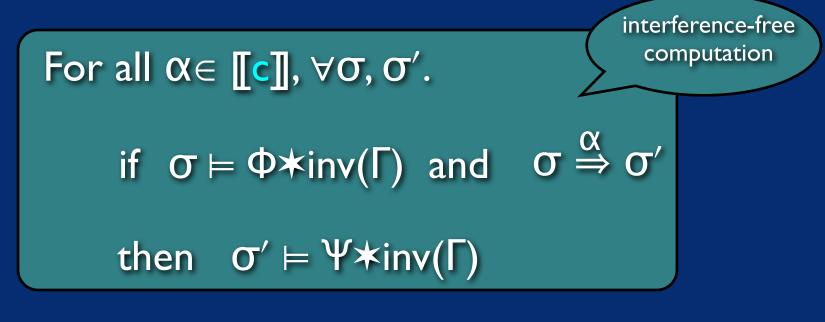
Parallel decomposition

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



Race-freedom

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



... NO RACES



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



IT IS EASIER TO GET FORGIVENESS THAN IT IS TO GET PERMISSION!

403 - Forbidden -

You are not authorized to view this page

You do not have permission to view this directory or page using the credentials you supplied.