High and Low Level Local Reasoning About Programs That Alter Data Structures

Mark Wheelhouse

This talk possible thanks to the work of:

Tony Hoare
Peter O'Hearn, John Reynolds, Hongseok Yang,
Philippa Gardner, Cristiano Calcagno, Uri Zarfaty

Reasoning About Programs - Why Bother?

Testing:

t=1	t=2	t=3	t=4	...
x=1	x=5	x=10	x=17	...

Reasoning About Programs - Why Bother?

Testing:

t=1	t=2	t=3	t=4	...
x=1	x=5	x=10	x=17	...

Reasoning About Programs - Why Bother?

Testing:

t=1	t=2	t=3	t=4	...	t=256
x=1	x=5	x=10	x=17	...	x= overflow!

Reasoning About Programs - Why Bother?

Testing:

t=1	t=2	t=3	t=4	...	t=256
x=1	x=5	x=10	x=17	...	x= overflow!

Moral of the Story

Test if failure cheep

Prove if safety critical

Hoare Reasoning

HR

Hoare Triples

Fault-avoiding partial correctness interpretation:

$$\{ P \} \; C \; \{ Q \}$$

Hoare Triples

Fault-avoiding partial correctness interpretation:

$$\{ P \}\ C\ \{ Q \}$$

pre-condition

Hoare Triples

Fault-avoiding partial correctness interpretation:

$$\{ P \}\ C\ \{ Q \}$$

pre-condition post-condition

Hoare Triples

Fault-avoiding partial correctness interpretation:

$$\{P\}\ C\ \{Q\}$$

pre-condition — post-condition

$$\{P\}\ C\ \{Q\} \Leftrightarrow \begin{array}{l} \forall s.\ s \models P \Rightarrow C,s \not\rightsquigarrow \text{fault} \\ \qquad \wedge\ \forall s'.\ C,s \rightsquigarrow s' \Rightarrow s' \models Q \end{array}$$

HR

Hoare Triple Examples

$\{ s(x) = 3 \}\ \ x := x+1\ \ \{ s(x) = 4 \}$

$\{ s(x) = 3 \}\ \ x := x+1\ \ \{ s(x) > 3 \}$

HR

Hoare Triple Examples

$$\{\ s(x) = 3\ \}\quad x := x+1\quad \{\ s(x) = 4\ \}$$

$$\{\ s(x) = 3\ \}\quad x := x+1\quad \{\ s(x) > 3\ \}$$

$$\{\ s(x) = v\ \}\quad x := x+1\quad \{\ s(x) = v+1\ \}$$

HR

Hoare Triple Examples

$\{ s(x) = 3 \}\ \ x := x+1\ \ \{ s(x) = 4 \}$

$\{ s(x) = 3 \}\ \ x := x+1\ \ \{ s(x) > 3 \}$

$\{ s(x) = v \}\ \ x := x+1\ \ \{ s(x) = v+1 \}$

Constancy:

$\{ s(x)=3 \wedge s(y)=3 \}\ \ x := x+1\ \ \{ s(x)=4 \wedge s(y)=3 \}$

HR

Hoare Triple Examples

$\{ s(x) = 3 \}\ \ x := x+1\ \ \{ s(x) = 4 \}$

$\{ s(x) = 3 \}\ \ x := x+1\ \ \{ s(x) > 3 \}$

$\{ s(x) = v \}\ \ x := x+1\ \ \{ s(x) = v+1 \}$

Constancy:

$\{ s(x)=3 \wedge s(y)=3 \}\ \ x := x+1\ \ \{ s(x)=4 \wedge s(y)=3 \}$

But in a heap world...

$\{ x \mapsto 3 \wedge y \mapsto 3 \}\ \ [x] := [x]+1\ \ \{ x \mapsto 4 \wedge y \mapsto ? \}$

Separation Logic

SL

Local Reasoning Viewpoint

"To understand how a program works, it should be possible for reasoning and specification to be confined to the cells that the program actually accesses. The value of any other cell will automatically remain unchanged."

- Peter O'Hearn

The Heap Model

The heap h is a finite partial function that maps heap locations (N) to integers (Z):

$$h : N \xrightarrow{fin} Z$$

SL

The Heap Model

The heap h is a finite partial function that maps heap locations (N) to integers (Z):

$$h : N \xrightarrow{fin} Z$$

$(10 \mapsto 1)$:

SL

The Heap Model

The heap h is a finite partial function that maps heap locations (N) to integers (Z):

$$h : N \xrightarrow{fin} Z$$

$(10 \mapsto 1)$:

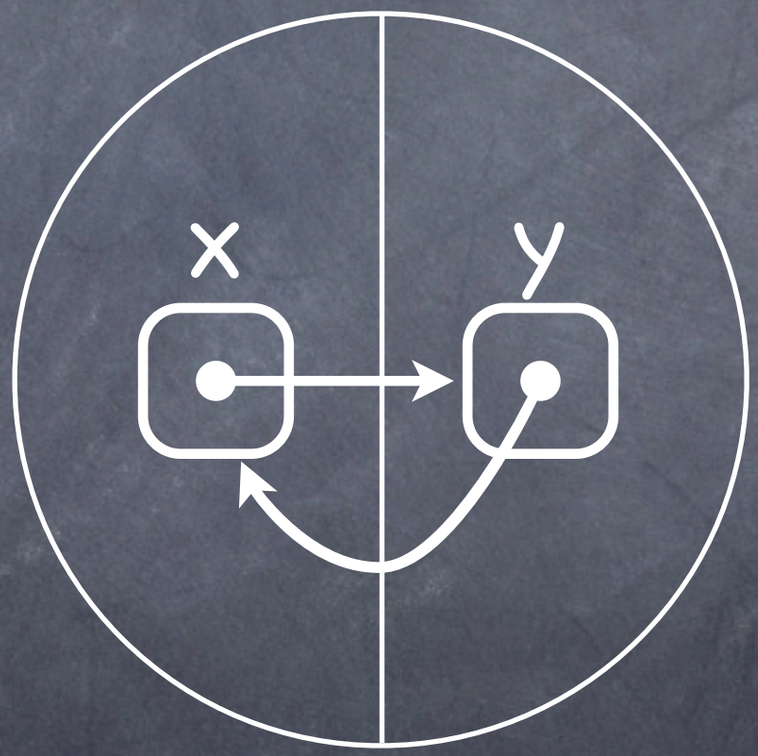
$(1 \mapsto 4)$:

SL

Separation Logic

* Connective

$$x \mapsto y \ * \ y \mapsto x$$

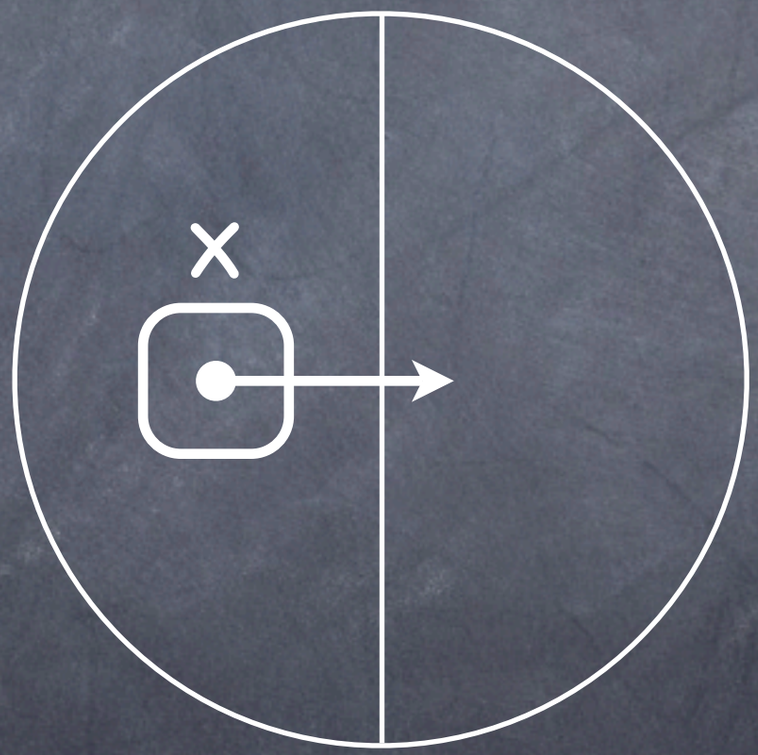


SL

Separation Logic

* Connective

$x \mapsto y$ *



Separation Logic

* Connective

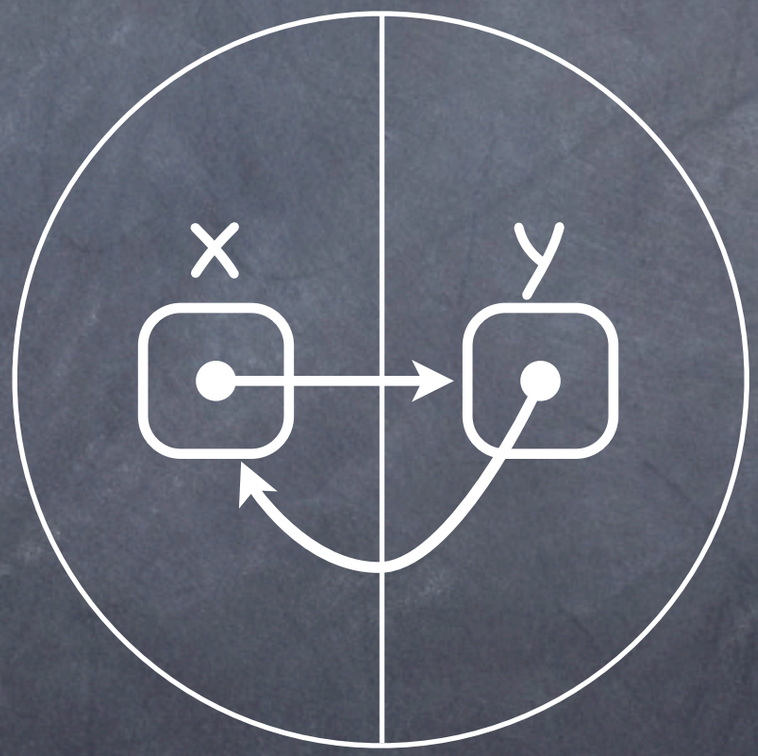
* $y \mapsto x$

SL

Separation Logic

* Connective

$$x \mapsto y \ * \ y \mapsto x$$

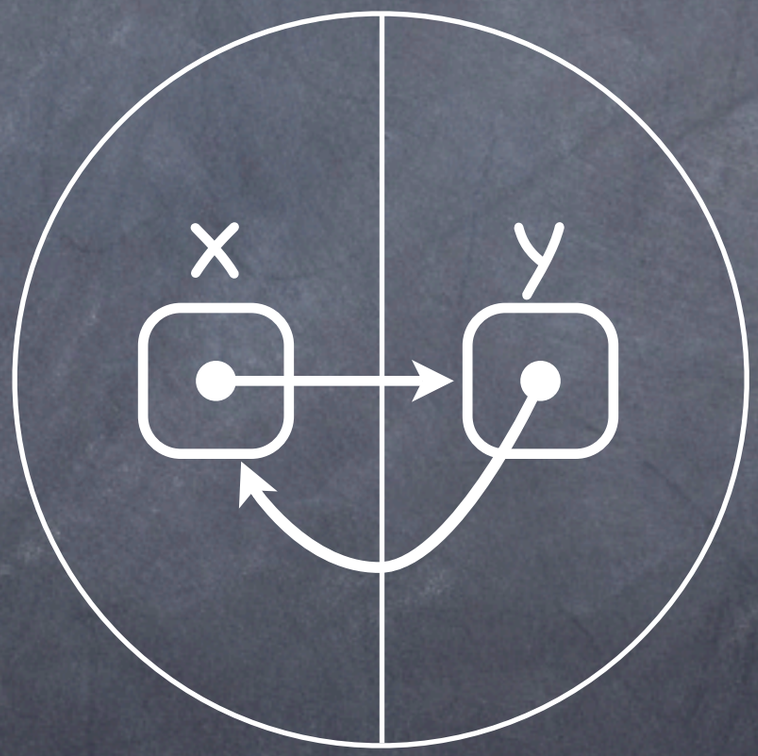


SL

Separation Logic

* Connective

$$x \mapsto y * y \mapsto x$$



x=10 10 42

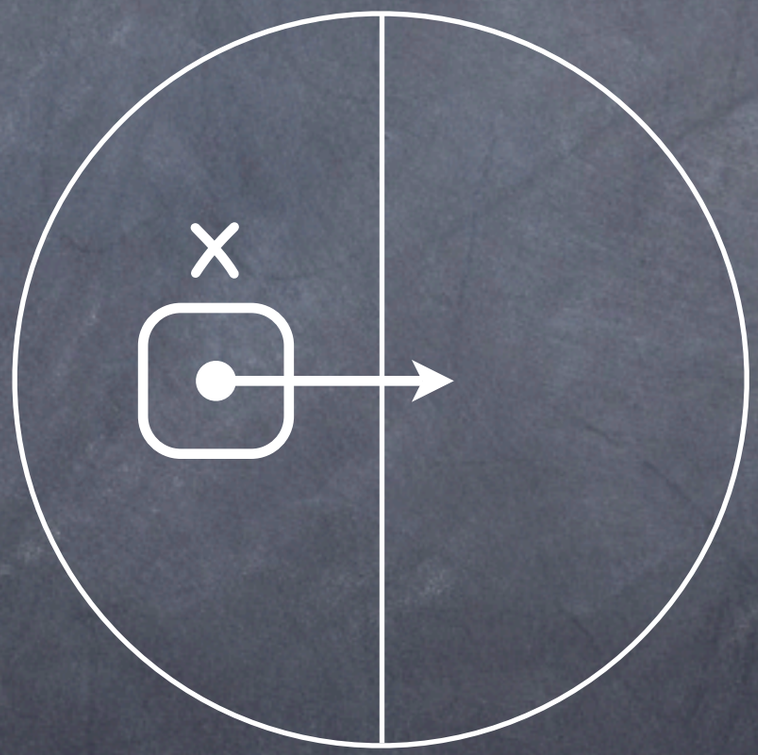
y=42 42 10

SL

Separation Logic

* Connective

$x \mapsto y$ *



x=10 10

y=42 42

SL

Separation Logic

* Connective

* $y \mapsto x$

x=10

y=42

42

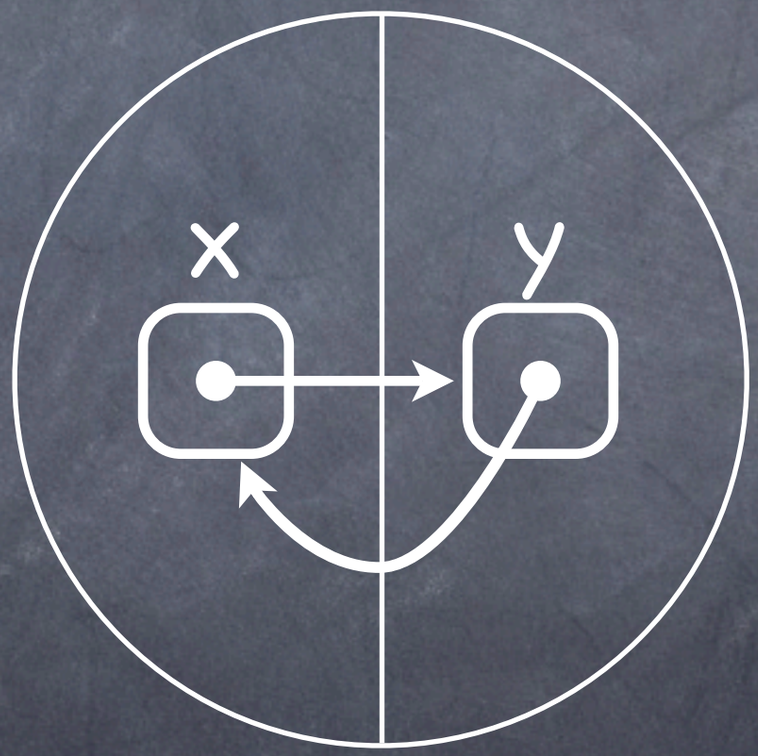
10

SL

Separation Logic

* Connective

$$x \mapsto y * y \mapsto x$$



x=10 10 42

y=42 42 10

The Logic

Add to Classical Logic:

emp the heap is empty

$x \mapsto y$ the heap has exactly one cell x holding y

$A * B$ the heap that can be split so that A is true of one partition and B of the other

Logic Examples

$x \mapsto y,z$

emp $* \; x \mapsto y$

$x \mapsto y \; * \; x \mapsto y$

$x \mapsto y \; \wedge \; x \mapsto y$

$x \mapsto y \; * \; x \mapsto z$

emp $*$ **emp**

$x \mapsto y \; \wedge \; y \mapsto z$

$x \mapsto y \; * \; y \mapsto z$

Logic Examples

$x \mapsto y,z \quad \equiv \quad x \mapsto y * x+1 \mapsto z$

emp $* \; x \mapsto y$

$x \mapsto y * x \mapsto y$

$x \mapsto y \wedge x \mapsto y$

$x \mapsto y * x \mapsto z$

emp $*$ **emp**

$x \mapsto y \wedge y \mapsto z$

$x \mapsto y * y \mapsto z$

Logic Examples

$x \mapsto y,z \equiv x \mapsto y * x+1 \mapsto z$

$\mathbf{emp} * x \mapsto y \equiv x \mapsto y$

$x \mapsto y * x \mapsto y$

$x \mapsto y \wedge x \mapsto y$

$x \mapsto y * x \mapsto z$

$\mathbf{emp} * \mathbf{emp}$

$x \mapsto y \wedge y \mapsto z$

$x \mapsto y * y \mapsto z$

Logic Examples

$x \mapsto y,z \equiv x \mapsto y * x+1 \mapsto z$

$\mathbf{emp} * x \mapsto y \equiv x \mapsto y$

$x \mapsto y * x \mapsto y \equiv$ false!

$x \mapsto y \wedge x \mapsto y$

$x \mapsto y * x \mapsto z$

$\mathbf{emp} * \mathbf{emp}$

$x \mapsto y \wedge y \mapsto z$

$x \mapsto y * y \mapsto z$

Logic Examples

$x \mapsto y,z \equiv x \mapsto y * x+1 \mapsto z$

$\mathbf{emp} * x \mapsto y \equiv x \mapsto y$

$x \mapsto y * x \mapsto y \equiv$ false!

$x \mapsto y \wedge x \mapsto y$ well defined formula

$x \mapsto y * x \mapsto z$

$\mathbf{emp} * \mathbf{emp}$

$x \mapsto y \wedge y \mapsto z$

$x \mapsto y * y \mapsto z$

Logic Examples

$x \mapsto y,z \equiv x \mapsto y * x+1 \mapsto z$

$\mathbf{emp} * x \mapsto y \equiv x \mapsto y$

$x \mapsto y * x \mapsto y \equiv$ false!

$x \mapsto y \wedge x \mapsto y$ well defined formula

$x \mapsto y * x \mapsto z \equiv$ false!

$\mathbf{emp} * \mathbf{emp}$

$x \mapsto y \wedge y \mapsto z$

$x \mapsto y * y \mapsto z$

Logic Examples

$x \mapsto y,z \equiv x \mapsto y * x+1 \mapsto z$

$\mathbf{emp} * x \mapsto y \equiv x \mapsto y$

$x \mapsto y * x \mapsto y \equiv$ false!

$x \mapsto y \wedge x \mapsto y$ well defined formula

$x \mapsto y * x \mapsto z \equiv$ false!

$\mathbf{emp} * \mathbf{emp} \equiv \mathbf{emp}$

$x \mapsto y \wedge y \mapsto z$

$x \mapsto y * y \mapsto z$

Logic Examples

$x \mapsto y,z \equiv x \mapsto y * x+1 \mapsto z$

$\mathbf{emp} * x \mapsto y \equiv x \mapsto y$

$x \mapsto y * x \mapsto y \equiv$ false!

$x \mapsto y \wedge x \mapsto y$ well defined formula

$x \mapsto y * x \mapsto z \equiv$ false!

$\mathbf{emp} * \mathbf{emp} \equiv \mathbf{emp}$

$x \mapsto y \wedge y \mapsto z \equiv$ false!

$x \mapsto y * y \mapsto z$

Logic Examples

$x \mapsto y,z \equiv x \mapsto y * x+1 \mapsto z$

$\mathbf{emp} * x \mapsto y \equiv x \mapsto y$

$x \mapsto y * x \mapsto y \equiv$ false!

$x \mapsto y \wedge x \mapsto y$ well defined formula

$x \mapsto y * x \mapsto z \equiv$ false!

$\mathbf{emp} * \mathbf{emp} \equiv \mathbf{emp}$

$x \mapsto y \wedge y \mapsto z \equiv$ false!

$x \mapsto y * y \mapsto z$ well defined formula

SL

In-place Reasoning

$$\{\ x \mapsto - \ \}\ [x]:=4\ \{\ x \mapsto 4\ \}$$

$$\{\ x \mapsto - \ \}\ \mathrm{dispose}(x)\ \{\ \mathbf{emp}\ \}$$

$$\{\ \mathbf{emp}\ \}\ x = \mathrm{cons}(a,b)\ \{\ x \mapsto a,b\ \}$$

The Frame Rule

Frame: $$\frac{\{ P \}\ C\ \{ Q \}}{\{ R * P \}\ C\ \{ R * Q \}}$$

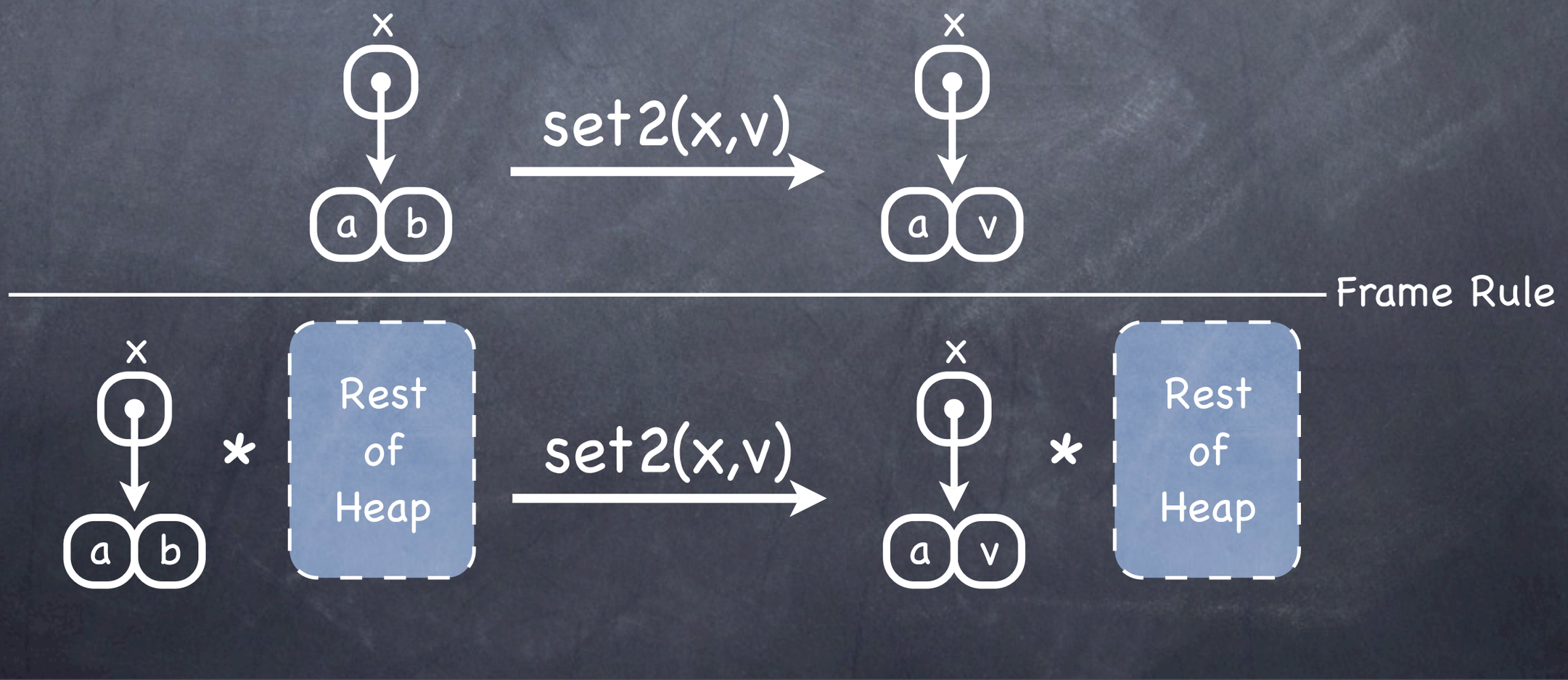
where variables in C do not clash with those in R

SL

The Frame Rule

Frame: $$\frac{\{ P \}\ C\ \{ Q \}}{\{ R * P \}\ C\ \{ R * Q \}}$$

where variables in C do not clash with those in R



Frame Rule Examples

Small Axiom: { x↦- } [x]:=4 { x↦4 }

Small Axiom: { x↦- } dispose(x) { **emp** }

Small Axiom: { **emp** } x = cons(a,b) { x↦a,b }

Frame Rule Examples

Small Axiom: $\{ x \mapsto - \}\ [x]:=4\ \{ x \mapsto 4 \}$

Framed: $\{ (x \mapsto -) * P \}\ [x]:=4\ \{ (x \mapsto 4) * P \}$

Small Axiom: $\{ x \mapsto - \}$ dispose(x) $\{$ **emp** $\}$

Small Axiom: $\{$ **emp** $\}$ x = cons(a,b) $\{ x \mapsto a,b \}$

Frame Rule Examples

Small Axiom: $\{ x \mapsto - \}\ [x]:=4\ \{ x \mapsto 4 \}$

Framed: $\{ (x \mapsto -) * P \}\ [x]:=4\ \{ (x \mapsto 4) * P \}$

Small Axiom: $\{ x \mapsto - \}$ dispose(x) $\{$ **emp** $\}$

Framed: $\{ (x \mapsto -) * P \}$ dispose(x) $\{ P \}$

Small Axiom: $\{$ **emp** $\}$ x = cons(a,b) $\{ x \mapsto a,b \}$

Frame Rule Examples

Small Axiom: $\{ x \mapsto - \}\ [x]:=4\ \{ x \mapsto 4 \}$

Framed: $\{ (x \mapsto -) * P \}\ [x]:=4\ \{ (x \mapsto 4) * P \}$

Small Axiom: $\{ x \mapsto - \}$ dispose(x) $\{$ **emp** $\}$

Framed: $\{ (x \mapsto -) * P \}$ dispose(x) $\{ P \}$

Small Axiom: $\{$ **emp** $\}$ x = cons(a,b) $\{ x \mapsto a,b \}$

Framed: $\{ P \}$ x = cons(a,b) $\{ (x \mapsto a,b) * P \}$

Inductive Definitions

$$\mathsf{tree}(T) \Leftrightarrow \text{if } (T = \mathbf{nil}) \text{ then } \mathbf{emp} \text{ else } \exists x,y.\ (T \mapsto l{:}x,\ r{:}y) * \mathsf{tree}(x) * \mathsf{tree}(y)$$

Inductive Definitions

$$\text{tree}(T) \Leftrightarrow \text{if } (T = \textbf{nil}) \text{ then } \textbf{emp} \text{ else } \exists x,y.\ (T \mapsto l{:}x,\ r{:}y) * \text{tree}(x) * \text{tree}(y)$$

```
DispTree(p) = i := p.left;
              j := p.right;
              dispose(p);
              DispTree(i);
              DispTree(j)
```

Inductive Definitions

$$\text{tree}(T) \Leftrightarrow \text{if } (T = \textbf{nil}) \text{ then } \textbf{emp} \text{ else } \exists x,y.\ (T \mapsto l{:}x,\ r{:}y) * \text{tree}(x) * \text{tree}(y)$$

```
DispTree(p) = i := p.left;
              j := p.right;
              dispose(p);
              DispTree(i);
              DispTree(j)
```

Spec:

$$\{\ \text{tree}(p)\ \}\ \text{DispTree}(p)\ \{\ \text{emp}\ \}$$

SL

Extended In-place Reasoning

i := p.left; j := p.right

dispose(p)

SL

Extended In-place Reasoning

{ tree(p) ∧ p ≠ **nil**}

i := p.left; j := p.right

dispose(p)

SL

Extended In-place Reasoning

{ tree(p) ∧ p ≠ **nil**}

{ (p ↦ l:x, r:y) ∗ tree(x) ∗ tree(y) }

i := p.left; j := p.right

dispose(p)

SL

Extended In-place Reasoning

{ tree(p) ∧ p ≠ **nil**}

{ (p↦l:x, r:y) ∗ tree(x) ∗ tree(y) }

i := p.left; j := p.right

{ (p↦l:i, r:j) ∗ tree(i) ∗ tree(j) }

dispose(p)

SL

Extended In-place Reasoning

$\{\ \mathsf{tree}(p) \wedge p \neq \mathbf{nil}\}$

$\{\ (p \mapsto l{:}x,\ r{:}y) * \mathsf{tree}(x) * \mathsf{tree}(y)\ \}$

```
  i := p.left; j := p.right
```

$\{\ (p \mapsto l{:}i,\ r{:}j) * \mathsf{tree}(i) * \mathsf{tree}(j)\ \}$

```
  dispose(p)
```

$\{\ \mathbf{emp} * \mathsf{tree}(i) * \mathsf{tree}(j)\ \}$

SL

Extended In-place Reasoning

{ tree(p) ∧ p ≠ **nil**}

{ (p ↦ l:x, r:y) ∗ tree(x) ∗ tree(y) }

i := p.left; j := p.right

{ (p ↦ l:i, r:j) ∗ tree(i) ∗ tree(j) }

dispose(p)

{ **emp** ∗ tree(i) ∗ tree(j) }

{ tree(i) ∗ tree(j) }

SL

Extended In-place Reasoning

{ tree(i) * tree(j) }
DispTree(i)
DispTree(j)

SL

Extended In-place Reasoning

{ tree(i) * tree(j) }

DispTree(i)

{ **emp** * tree(j) }

DispTree(j)

SL

Extended In-place Reasoning

{ tree(i) * tree(j) }

DispTree(i)

{ **emp** * tree(j) }

DispTree(j)

{ **emp** * **emp** }

SL

Extended In-place Reasoning

{ tree(i) * tree(j) }

DispTree(i)

{ **emp** * tree(j) }

DispTree(j)

{ **emp** * **emp** }

{ **emp** }

Separation Logic Main Points

- * lets you do in-place reasoning
- * interacts well with inductive definitions
- powerful way to avoid writing frame axioms
- pre/post of specification tied to footprint (describe local operations)

Context Logic

CL

The Tree Context Model

tree t ::= ∅ | n[t] | t⊗t

CL

The Tree Context Model

tree t ::= ∅ | n[t] | t⊗t

tree context c ::= _ | n[c] | c⊗t | t⊗c

(single holed)

CL

The Tree Context Model

tree t ::= ∅ | n[t] | t⊗t

tree context c ::= _ | n[c] | c⊗t | t⊗c

(single holed)

unique node identifiers n

CL

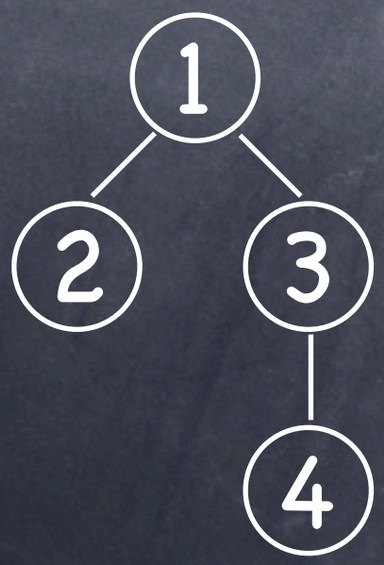
The Tree Context Model

tree t ::= ∅ | n[t] | t⊗t

tree context c ::= _ | n[c] | c⊗t | t⊗c

(single holed)

unique node identifiers n



= 1[2[∅]⊗3[4[∅]]]

CL

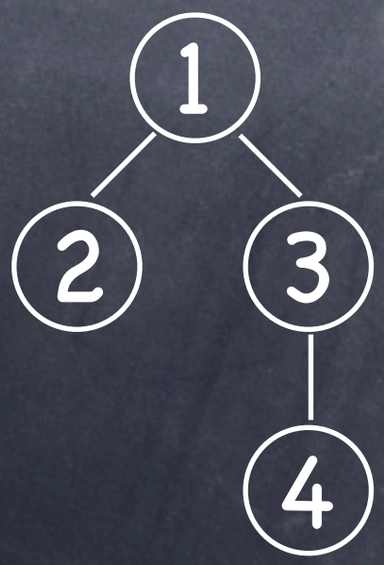
The Tree Context Model

tree t ::= ∅ | n[t] | t⊗t

tree context c ::= _ | n[c] | c⊗t | t⊗c

(single holed)

unique node identifiers n

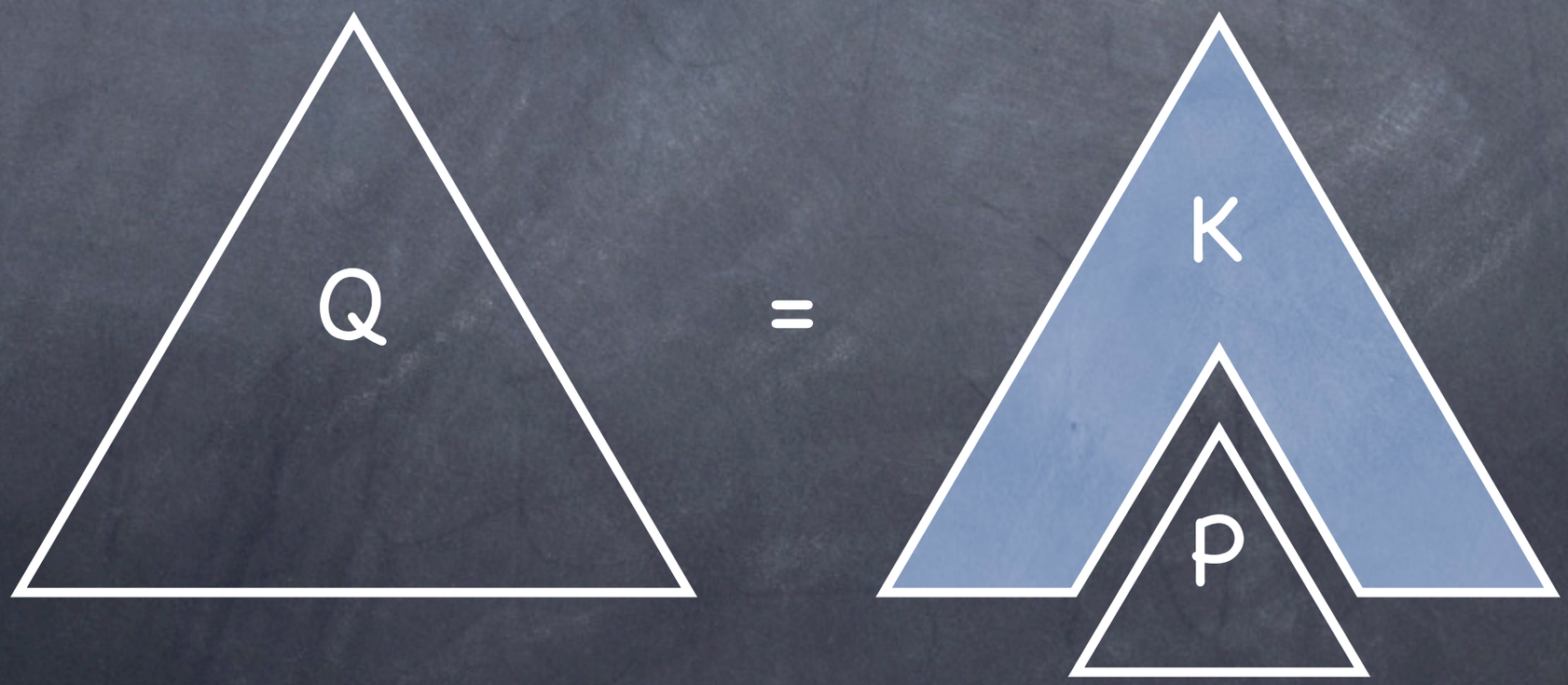


= 1[2⊗3[4]]

CL

Context Application

$$Q = K \circ P$$



Application Example

$$1[\ 2 \otimes 3[\ \_\ ]\ ] \circ 4 \;=\; 1[\ 2 \otimes 3[4]\ ]$$

CL

Application Example

$$1[\ 2 \otimes 3[\ \_\ ]\ ] \circ 4 \ = \ 1[\ 2 \otimes 3[4]\ ]$$

CL

The Logic

Add to Classical Logic:

$\emptyset$	the empty tree
$n[P]$	the node n with P true of the subtree
$P \otimes P$	parallel composition of trees
$\_$	context hole
$K \circ P$	context application

Logic Examples

$$\emptyset \otimes P$$

$$n[P] \otimes n[P]$$

$$n[P] \wedge n[P]$$

$$n[\ \_\ ] \circ \emptyset$$

$$n[\ \_\ ] \circ n$$

$$m \circ n[\ \_\ ]$$

$$n[\ \_\ ] \circ m[\ \_\ ] \circ p$$

Logic Examples

$$\emptyset \otimes P \equiv P$$

$$n[P] \otimes n[P]$$

$$n[P] \wedge n[P]$$

$$n[\ \_\ ] \circ \emptyset$$

$$n[\ \_\ ] \circ n$$

$$m \circ n[\ \_\ ]$$

$$n[\ \_\ ] \circ m[\ \_\ ] \circ p$$

Logic Examples

$$\emptyset \otimes P \equiv P$$

$$n[P] \otimes n[P] \equiv \text{false!}$$

$$n[P] \wedge n[P]$$

$$n[\ \_\ ] \circ \emptyset$$

$$n[\ \_\ ] \circ n$$

$$m \circ n[\ \_\ ]$$

$$n[\ \_\ ] \circ m[\ \_\ ] \circ p$$

CL

Logic Examples

$$\emptyset \otimes P \equiv P$$

$$n[P] \otimes n[P] \equiv \text{false!}$$

$$n[P] \wedge n[P] \quad \text{well defined}$$

$$n[\ \_\ ] \circ \emptyset$$

$$n[\ \_\ ] \circ n$$

$$m \circ n[\ \_\ ]$$

$$n[\ \_\ ] \circ m[\ \_\ ] \circ p$$

Logic Examples

$$\emptyset \otimes P \equiv P$$

$$n[P] \otimes n[P] \equiv \text{false!}$$

$$n[P] \wedge n[P] \quad \text{well defined}$$

$$n[\ \_\ ] \circ \emptyset \equiv n$$

$$n[\ \_\ ] \circ n$$

$$m \circ n[\ \_\ ]$$

$$n[\ \_\ ] \circ m[\ \_\ ] \circ p$$

Logic Examples

$$\emptyset \otimes P \equiv P$$

$$n[P] \otimes n[P] \equiv \text{false!}$$

$$n[P] \wedge n[P] \quad \text{well defined}$$

$$n[\ \_\ ] \circ \emptyset \equiv n$$

$$n[\ \_\ ] \circ n \equiv \text{false!}$$

$$m \circ n[\ \_\ ]$$

$$n[\ \_\ ] \circ m[\ \_\ ] \circ p$$

Logic Examples

$$\emptyset \otimes P \equiv P$$

$$n[P] \otimes n[P] \equiv \text{false!}$$

$$n[P] \wedge n[P] \quad \text{well defined}$$

$$n[\ \_\ ] \circ \emptyset \equiv n$$

$$n[\ \_\ ] \circ n \equiv \text{false!}$$

$$m \circ n[\ \_\ ] \equiv \text{false!}$$

$$n[\ \_\ ] \circ m[\ \_\ ] \circ p$$

Logic Examples

$$\emptyset \otimes P \equiv P$$

$$n[P] \otimes n[P] \equiv \text{false!}$$

$$n[P] \wedge n[P] \quad \text{well defined}$$

$$n[\ \_\ ] \circ \emptyset \equiv n$$

$$n[\ \_\ ] \circ n \equiv \text{false!}$$

$$m \circ n[\ \_\ ] \equiv \text{false!}$$

$$n[\ \_\ ] \circ m[\ \_\ ] \circ p \equiv n[m[p]]$$

In-place Reasoning

$$\{\ n[\ t \otimes m[t']\ ]\ \}$$
$$n' := \text{getLastChild}(n)$$
$$\{\ n[\ t \otimes m[t']\ ]\ \wedge\ (n'=m)\ \}$$

$$\{\ n[t]\ \}\ \text{insertBefore}(n,X)\ \{\ X \otimes n[t]\ \}$$

$$\{\ n[t]\ \}\ \text{disposeTree}(n)\ \{\ \emptyset\ \}$$

The Frame Rule

Frame: $$\frac{\{ P \}\ C\ \{ Q \}}{\{ K \circ P \}\ C\ \{ K \circ Q \}}$$

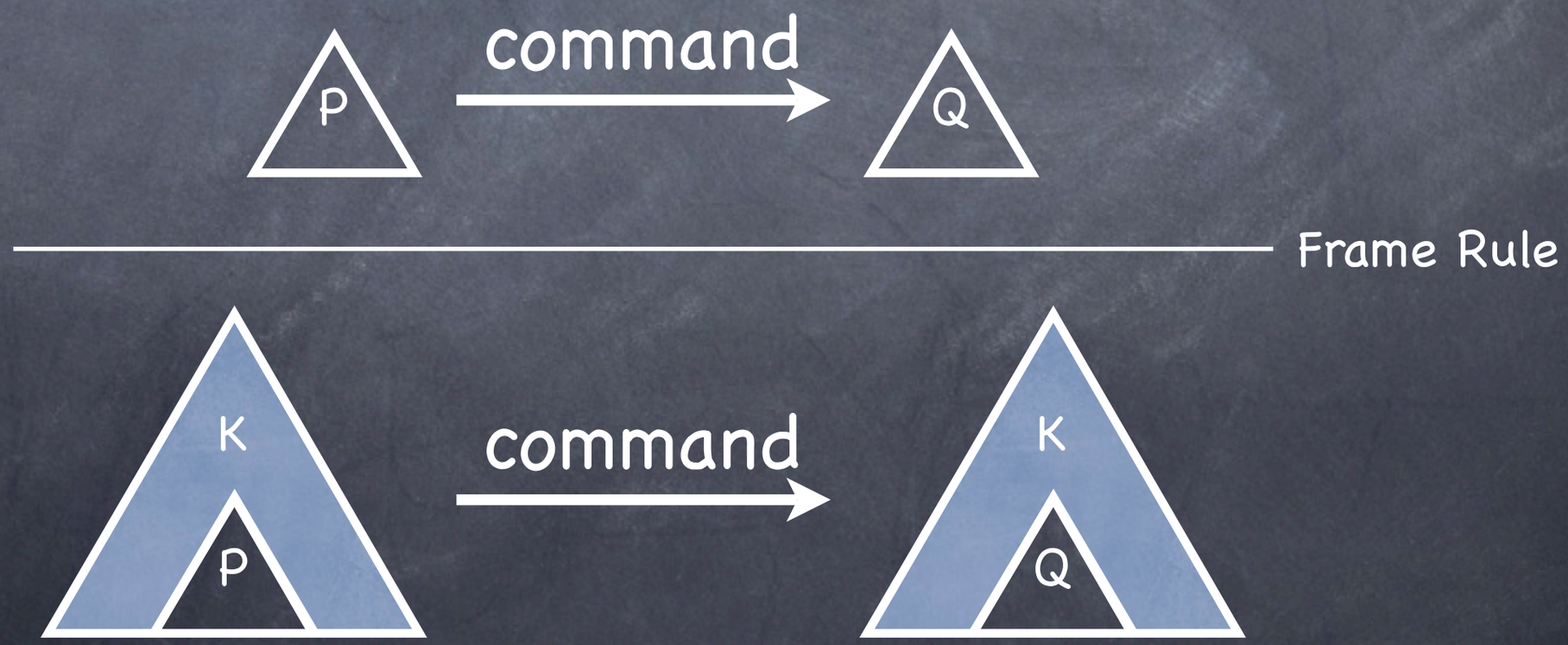
where variables in C do not clash with those in K

CL

The Frame Rule

Frame: $$\frac{\{ P \}\ C\ \{ Q \}}{\{ K \circ P \}\ C\ \{ K \circ Q \}}$$

where variables in C do not clash with those in K



Frame Rule Examples

$$\{\ K \circ n[\ t \otimes m[t']\ ]\ \}$$
$$n' := \text{getLastChild}(n)$$
$$\{\ K \circ n[\ t \otimes m[t']\ ]\ \wedge\ (n'=m)\ \}$$

$$\{\ K \circ n[t]\ \}\ \text{insertBefore}(n,X)\ \{\ K \circ (X \otimes n[t])\ \}$$

$$\{\ K \circ n[t]\ \}\ \text{disposeTree}(n)\ \{\ K \circ \emptyset\ \}$$

CL

Extended In-place Reasoning

replaceTree(n,X) = insertBefore(n,X);
disposeTree(n)

Spec:

{ n[t] } replaceTree(n,X) { X }

CL

Extended In-place Reasoning

insertBefore(n,X)

disposeTree(n)

CL

Extended In-place Reasoning

{ n[t] }

insertBefore(n,X)

disposeTree(n)

CL

Extended In-place Reasoning

{ n[t] }

insertBefore(n,X)

{ X⊗n[t] }

disposeTree(n)

CL

Extended In-place Reasoning

{ n[t] }

insertBefore(n,X)

{ X⊗n[t] }

{ (X⊗_)∘ n[t] }

disposeTree(n)

CL

Extended In-place Reasoning

{ n[t] }

insertBefore(n,X)

{ X⊗n[t] }

{ (X⊗_)∘ n[t] }

disposeTree(n)

{ (X⊗_)∘ ø }

CL

Extended In-place Reasoning

{ n[t] }

insertBefore(n,X)

{ X⊗n[t] }

{ (X⊗_)∘ n[t] }

disposeTree(n)

{ (X⊗_)∘ ø }

{ X⊗ø }

CL

Extended In-place Reasoning

{ n[t] }

insertBefore(n,X)

{ X⊗n[t] }

{ (X⊗_)∘ n[t] }

disposeTree(n)

{ (X⊗_)∘ ø }

{ X⊗ø }

{ X }

CL

Context Logic Main Points

- o lets you do in-place reasoning
- provides a high level of abstraction
- not tied to one data structure (trees just an example)
- also lets you avoid writing frame axioms
- pre/post tied to footprint size

Separation Logic
The Future

- Concurrency and True Concurrency
- Rely/Guarantee style reasoning
- More Tools, improvements to current tools (Slayer, Space Invader, ...)
- jStar - verification for Java

CL

Context Logic
The Future

- Multi-holed Context Logic
- Concurrency
- Smaller Axioms / Fine-grained splittings
- Automated Verification / Proof Assistant
- Complex Data Stores (e.g. B*Trees)
- Links between High/Low level reasoning

What all this means...

What all this means...