High Level Program Reasoning

Wessex Theory Seminar - Feb 2010

Mark Wheelhouse
Imperial College London

Introduction

Program Verification:

Introduction

Program Verification:

- Separation Logic
  - C Programs
  - Device Drivers

Introduction

Program Verification:

- Separation Logic

  C Programs

  Device Drivers

- Context Logic

  Tree Update

  DOM Specification

Introduction

Program Verification:

- Separation Logic

  C Programs

  Device Drivers

- Context Logic

  Tree Update

  DOM Specification

- Segment Logic

  Fine-grained Update

  Concurrent Update

Introduction

Program Verification:

- Separation Logic

  C Programs

  Device Drivers

Low Level

- Context Logic

  Tree Update

  DOM Specification

- Segment Logic

  Fine-grained Update

  Concurrent Update

Introduction

Program Verification:

- Separation Logic
  - C Programs
  - Device Drivers
- Context Logic
  - Tree Update
  - DOM Specification
- Segment Logic
  - Fine-grained Update
  - Concurrent Update

Programming Language

skip

x := Exp

$\mathbb{C}$;$\mathbb{C}$

if (B) then { $\mathbb{C}$ } else { $\mathbb{C}$ }

while (B) do { $\mathbb{C}$ }

$\mathbb{C}_{Basic}$

Local Hoare Reasoning

Partial Correctness: $\{ P \} \; \mathbb{C} \; \{ Q \}$

Local Hoare Reasoning

Partial Correctness: $\{ P \} \ \mathbb{C} \ \{ Q \}$

Precondition

Local Hoare Reasoning

Partial Correctness: $\{ P \} \; \mathbb{C} \; \{ Q \}$

Precondition → P

Program → $\mathbb{C}$

Local Hoare Reasoning

Partial Correctness: $\{ P \} \; \mathbb{C} \; \{ Q \}$

Precondition — Program — Postcondition

Separation Logic

heap $h : \mathbb{N}^{+} \xrightarrow[\text{fin}]{} \mathrm{Val}$

Separation Logic

heap $h : \mathbb{N}^{+} \longrightarrow_{\text{fin}} \text{Val}$

emp	empty heap
$x \mapsto y$	heap of exactly one cell
$P * Q$	separating conjunction (disjoint union)

Separation Logic

heap $h : \mathbb{N}^{+} \xrightarrow[\text{fin}]{} \mathrm{Val}$

emp	empty heap
$x \mapsto y$	heap of exactly one cell
$P * Q$	separating conjunction (disjoint union)

$x \mapsto y * y \mapsto z * w \mapsto \emptyset$

Low-Level Reasoning

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

$\{ x \mapsto v \}$
[x] := E
$\{ x \mapsto E[v/x] \}$

Low-Level Reasoning

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

$\{\ x \mapsto v\ \}$
[x] := E
$\{\ x \mapsto E[v/x]\ \}$

Small Axioms

Separation Frame Rule

$$\frac{\{\, P \,\}\ \ \mathbb{C}\ \ \{\, Q \,\}}{\{\, R * P \,\}\ \mathbb{C}\ \{\, R * Q \,\}}$$

Separation Frame Rule

$$\frac{\{\ P\ \}\ \ \mathbb{C}\ \ \{\ Q\ \}}{\{\ R * P\ \}\ \mathbb{C}\ \{\ R * Q\ \}}$$

h

Separation Frame Rule

$$\frac{\{\ P\ \}\quad \mathbb{C}\quad \{\ Q\ \}}{\{\ R * P\ \}\ \mathbb{C}\ \{\ R * Q\ \}}$$

R * P

Separation Frame Rule

$$\frac{\{\, P \,\}\ \ \mathbb{C}\ \ \{\, Q \,\}}{\{\, R * P \,\}\ \mathbb{C}\ \{\, R * Q \,\}}$$

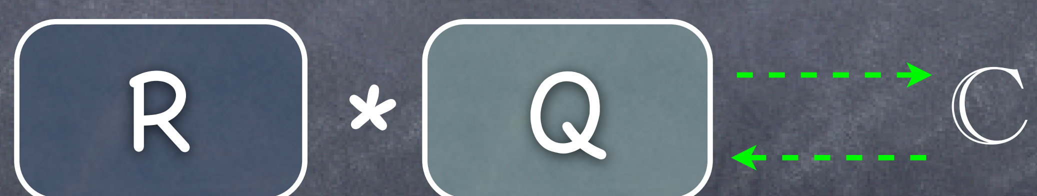
R * P

Separation Frame Rule

$$\frac{\{\, P \,\}\ \ \mathbb{C}\ \ \{\, Q \,\}}{\{\, R * P \,\}\ \mathbb{C}\ \{\, R * Q \,\}}$$

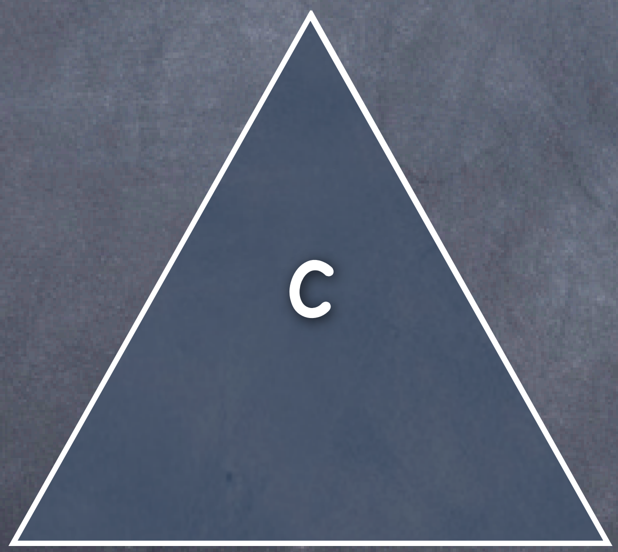
Separation Frame Rule

$$\frac{\{\, P \,\} \quad \mathbb{C} \quad \{\, Q \,\}}{\{\, R * P \,\}\ \mathbb{C}\ \{\, R * Q \,\}}$$

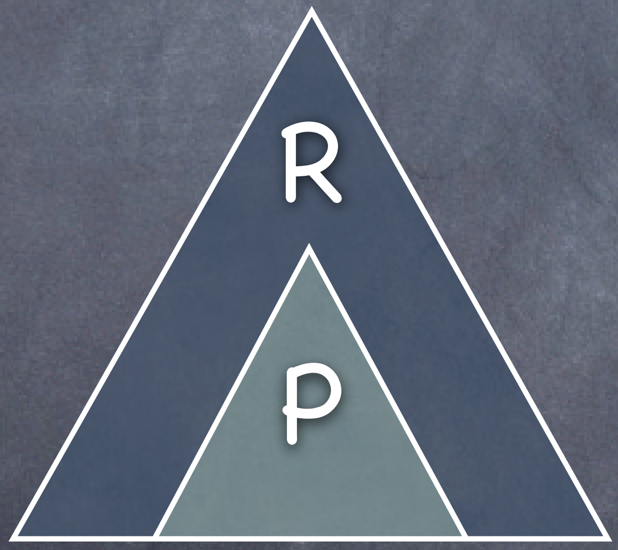


Context Logic in a Nutshell

Context Logic in a Nutshell



Context Logic in a Nutshell



Context Logic in a Nutshell

Context Logic in a Nutshell

Context Logic in a Nutshell

Context Logic in a Nutshell

Context Logic in a Nutshell

Concurrent Programs

⋮

$\mathbb{C} \mid\mid \mathbb{C}$

res r in { $\mathbb{C}$ }

with r when B do { $\mathbb{C}$ }

Conditional Critical Region!!!

Disjoint Concurrency at the Low-Level

$$\frac{\{ P_1 \}\ \mathbb{C}_1\ \{ Q_1 \} \quad \{ P_2 \}\ \mathbb{C}_2\ \{ Q_2 \}}{\{ P_1 * P_2 \}\ \mathbb{C}_1 \mid\mid \mathbb{C}_2\ \{ Q_1 * Q_2 \}}\ \text{PAR}$$

Disjoint Concurrency Example

$$[x] := 5 \parallel \text{dispose}(y)$$

Disjoint Concurrency Example

$$\{\ x \mapsto - * y \mapsto -\ \}$$

$$[x] := 5 \quad \| \quad \text{dispose}(y)$$

Disjoint Concurrency Example

$$\{ x \mapsto - * y \mapsto - \}$$

$\{ x \mapsto - \}$	$\Vert$	$\{ y \mapsto - \}$
[x] := 5		dispose(y)

Disjoint Concurrency Example

$$\{ x \mapsto - * y \mapsto - \}$$

$\{ x \mapsto - \}$	$\|$	$\{ y \mapsto - \}$
[x] := 5	$\|$	dispose(y)
$\{ x \mapsto 5 \}$	$\|$	$\{ emp \}$

Disjoint Concurrency Example

$$\{\, x \mapsto - * y \mapsto - \,\}$$

$\{\, x \mapsto - \,\}$	$\|$	$\{\, y \mapsto - \,\}$
[x] := 5	$\|$	dispose(y)
$\{\, x \mapsto 5 \,\}$	$\|$	$\{\, emp \,\}$

$$\{\, x \mapsto 5 * emp \,\}$$

Disjoint Concurrency Example

$$\{ x \mapsto - * y \mapsto - \}$$

$\{ x \mapsto - \}$	$\Vert$	$\{ y \mapsto - \}$
[x] := 5	$\Vert$	dispose(y)
$\{ x \mapsto 5 \}$	$\Vert$	$\{ emp \}$

$$\{ x \mapsto 5 * emp \}$$

$$\{ x \mapsto 5 \}$$

Complex Concurrency at the Low-Level

$$\frac{\{\ P\ \}\ \mathbb{C}\ \{\ Q\ \}}{\{\ RI_r * P\ \}\ \text{res } r \text{ in } \{\ \mathbb{C}\ \}\ \{\ RI_r * Q\ \}}\ \text{RES}$$

$$\frac{\{\ RI_r * P \wedge B\ \}\ \mathbb{C}\ \{\ RI_r * Q\ \}}{\{\ P\ \}\ \text{with } r \text{ when } B \text{ do } \{\ \mathbb{C}\ \}\ \{\ Q\ \}}\ \text{CCR}$$

Complex Concurrency Example

```
with r when x=0 do {        ||   with r when x=1 do {
  c := cons(); x=1          ||     dispose(c); x=0
}                           ||   }
```

Complex Concurrency Example

$RI_r = (x=0 \wedge emp) \vee (x=1 \wedge c \mapsto -)$

```
with r when x=0 do {          ||   with r when x=1 do {
  c := cons(); x=1            ||     dispose(c); x=0
}                             ||   }
```

Complex Concurrency Example

$RI_r = (x{=}0 \wedge \mathrm{emp}) \vee (x{=}1 \wedge c \mapsto -)$

{ emp }

with r when x=0 do {	with r when x=1 do {
c := cons(); x=1	dispose(c); x=0
}	}

Complex Concurrency Example

$RI_r = (x=0 \wedge emp) \vee (x=1 \wedge c \mapsto -)$

{ emp }
with r when x=0 do {
  { $RI_r \wedge x=0$ }
  c := cons(); x=1
}

||

with r when x=1 do {

  dispose(c); x=0
}

Complex Concurrency Example

$RI_r = (x{=}0 \wedge emp) \vee (x{=}1 \wedge c \mapsto -)$

```
{ emp }
with r when x=0 do {
  { RIr ∧ x=0 }
  c := cons(); x=1
  { RIr ∧ x=1 }
}
```

||

```
with r when x=1 do {

   dispose(c); x=0

}
```

Complex Concurrency Example

$RI_r = (x{=}0 \wedge emp) \vee (x{=}1 \wedge c \mapsto -)$

{ emp }	‖	
with r when x=0 do {	‖	with r when x=1 do {
{ $RI_r \wedge x{=}0$ }	‖	
c := cons(); x=1	‖	dispose(c); x=0
{ $RI_r \wedge x{=}1$ }	‖	
}	‖	}
{ emp }	‖	

Complex Concurrency Example

$RI_r = (x=0 \wedge emp) \vee (x=1 \wedge c \mapsto -)$

```
{ emp }
with r when x=0 do {
  { RIr ∧ x=0 }
  c := cons(); x=1
  { RIr ∧ x=1 }
}
{ emp }
```

||

```
{ emp }
with r when x=1 do {
  { RIr ∧ x=1 }
  dispose(c); x=0
  { RIr ∧ x=0 }
}
{ emp }
```

Problems at the High-Level

{ n[t] }	‖	{ m[t′] }
deleteTree(n)	‖	deleteTree(m)
{ ∅ }	‖	{ ∅ }

Problems at the High-Level

$\{\ n[t]\ \text{-?-}\ m[t']\ \}$

$\{\ n[t]\ \}$	‖	$\{\ m[t']\ \}$
deleteTree(n)	‖	deleteTree(m)
$\{\ \emptyset\ \}$	‖	$\{\ \emptyset\ \}$

Problems at the High-Level

{ n[t] -?- m[t'] }

{ n[t] } deleteTree(n) { ∅ }	‖	{ m[t'] } deleteTree(m) { ∅ }

- \- but what if not siblings?

Problems at the High-Level

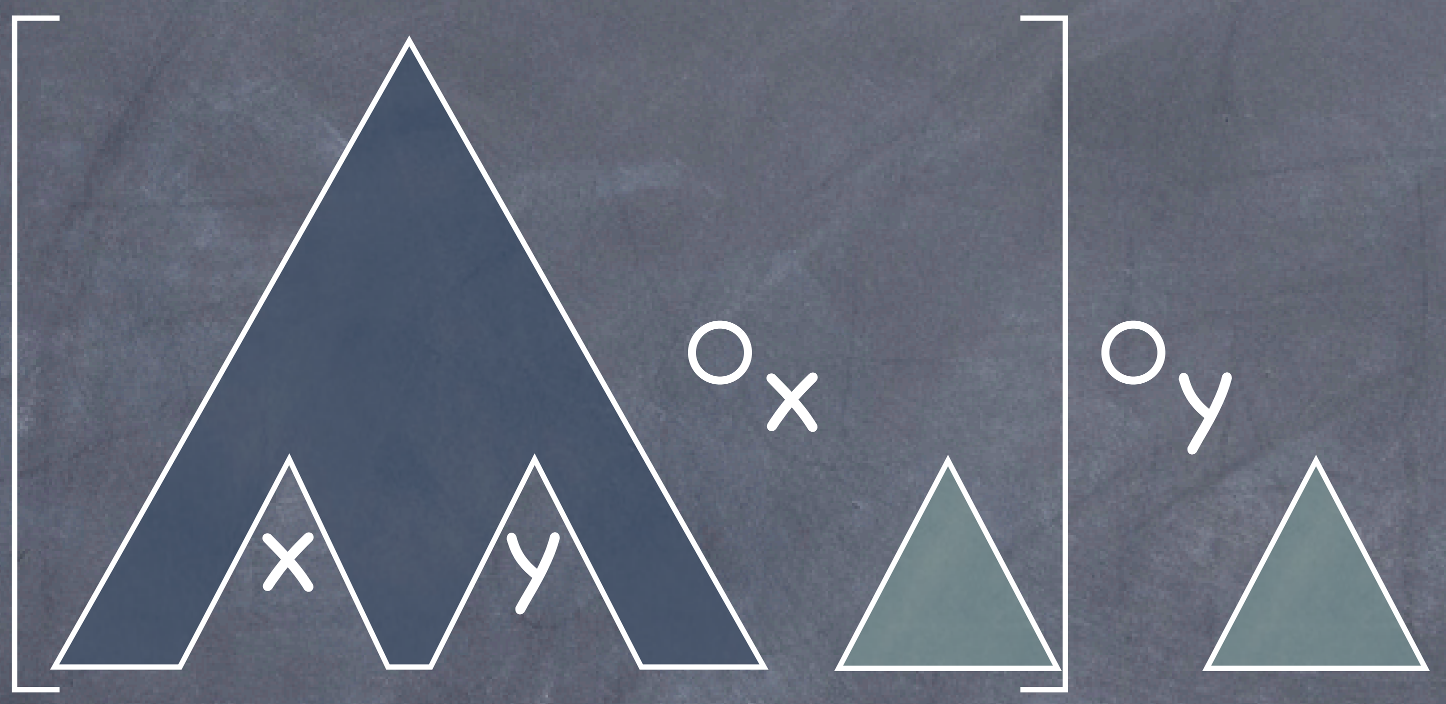
{ n[t] -?- m[t′] }

{ n[t] }	‖	{ m[t′] }
deleteTree(n)	‖	deleteTree(m)
{ ∅ }	‖	{ ∅ }

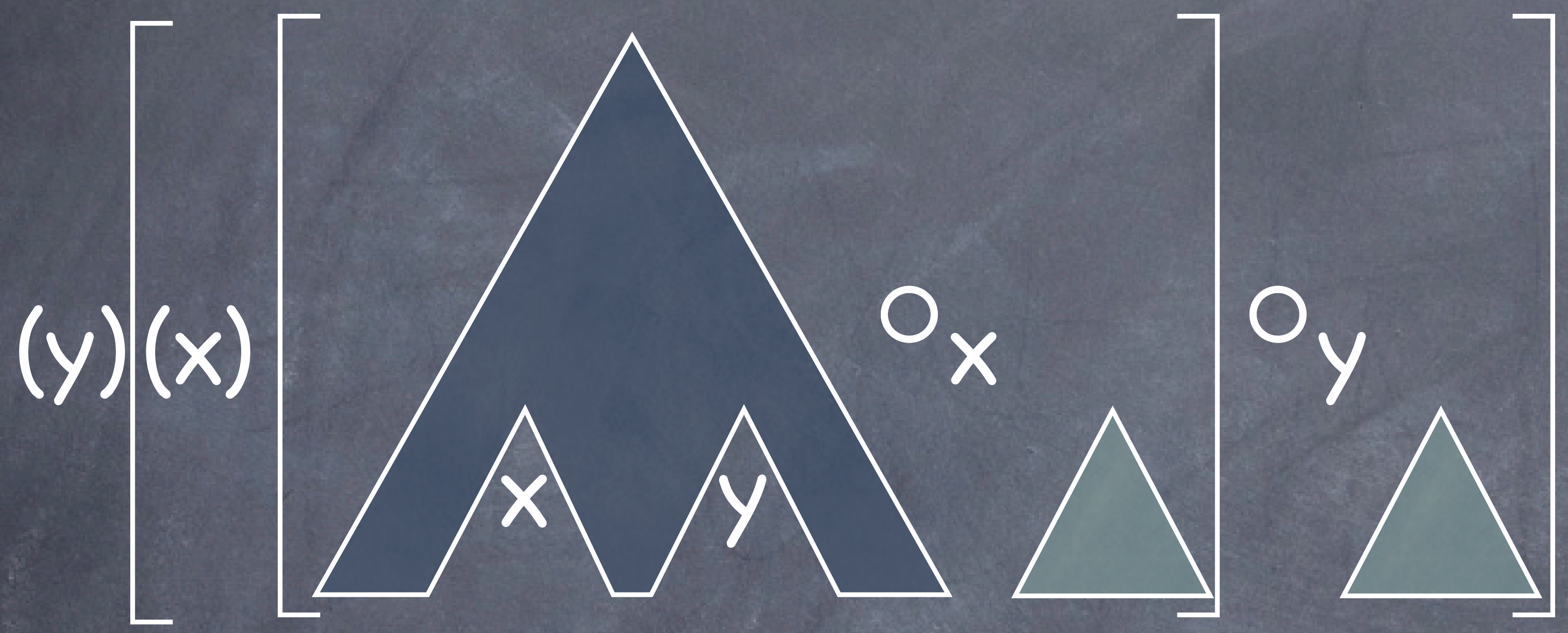
- $\otimes$ - but what if not siblings?
- $\circ_x$ - but neither is a context.

Segment Idea

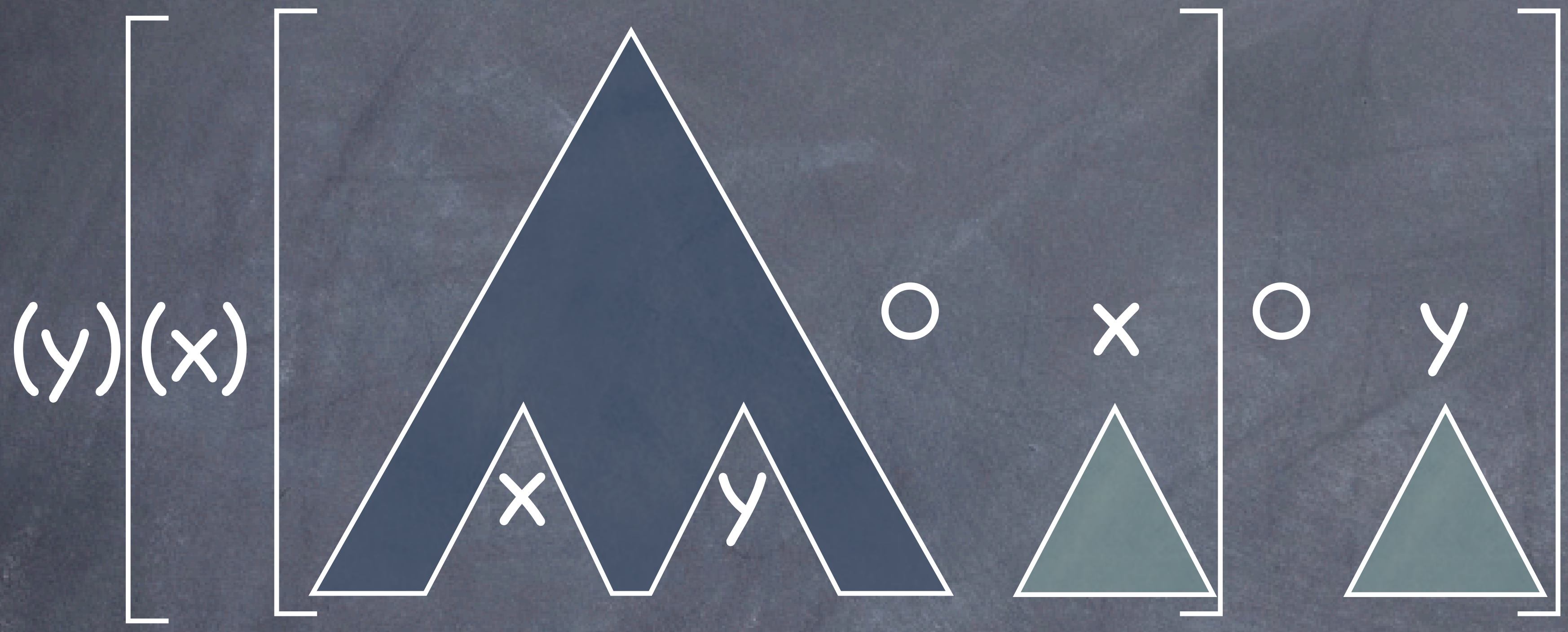
Segment Idea



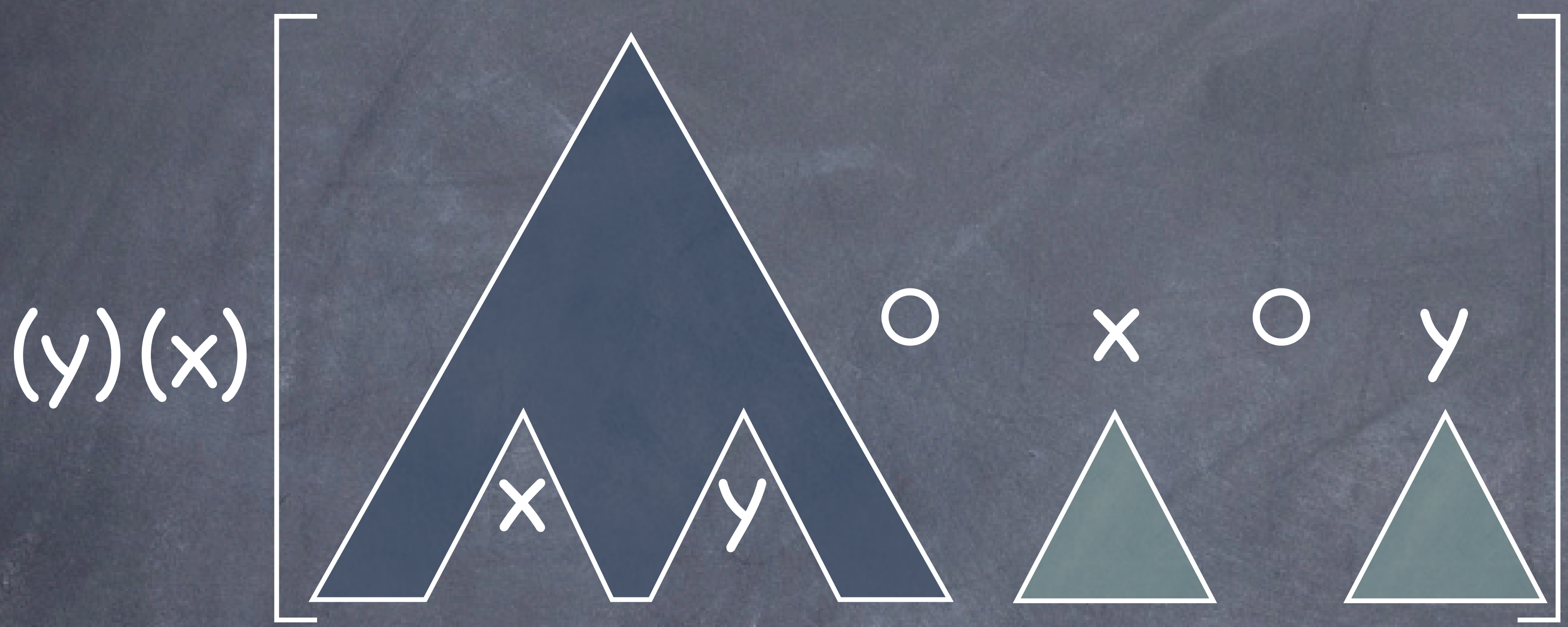
Segment Idea



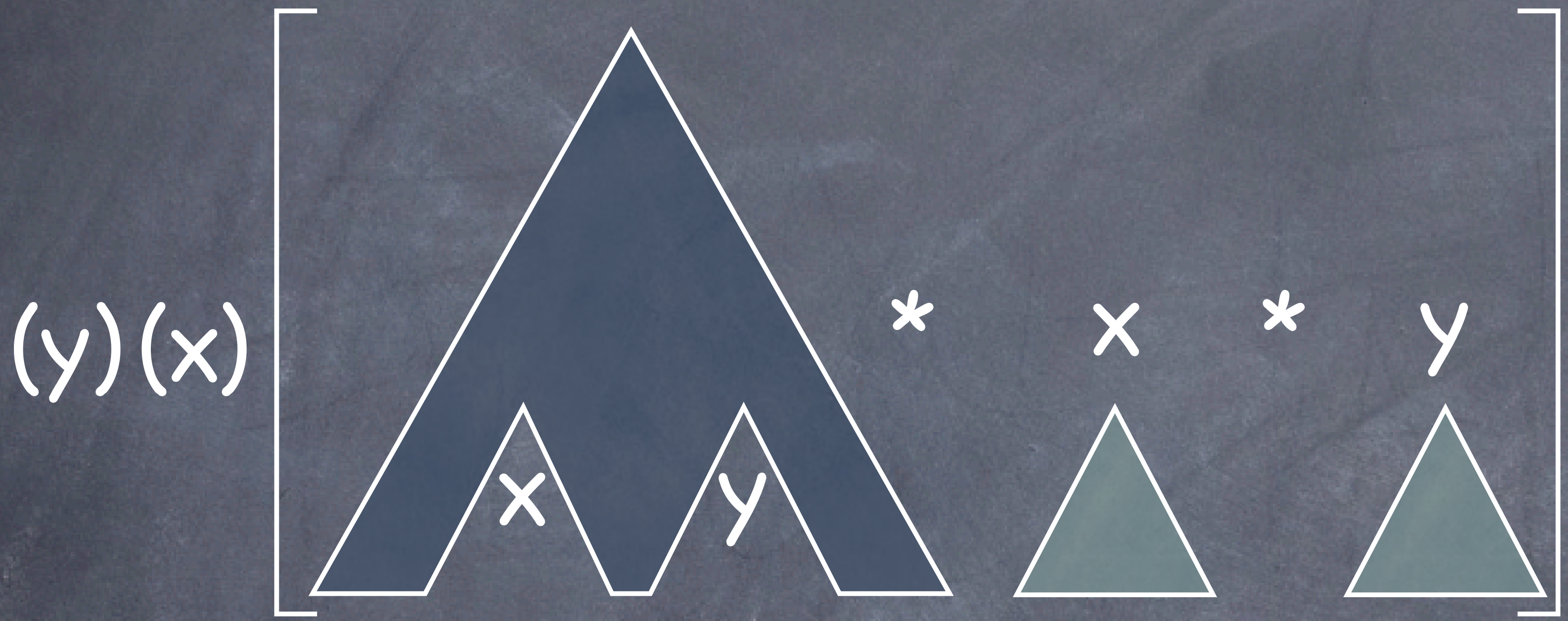
Segment Idea



Segment Idea



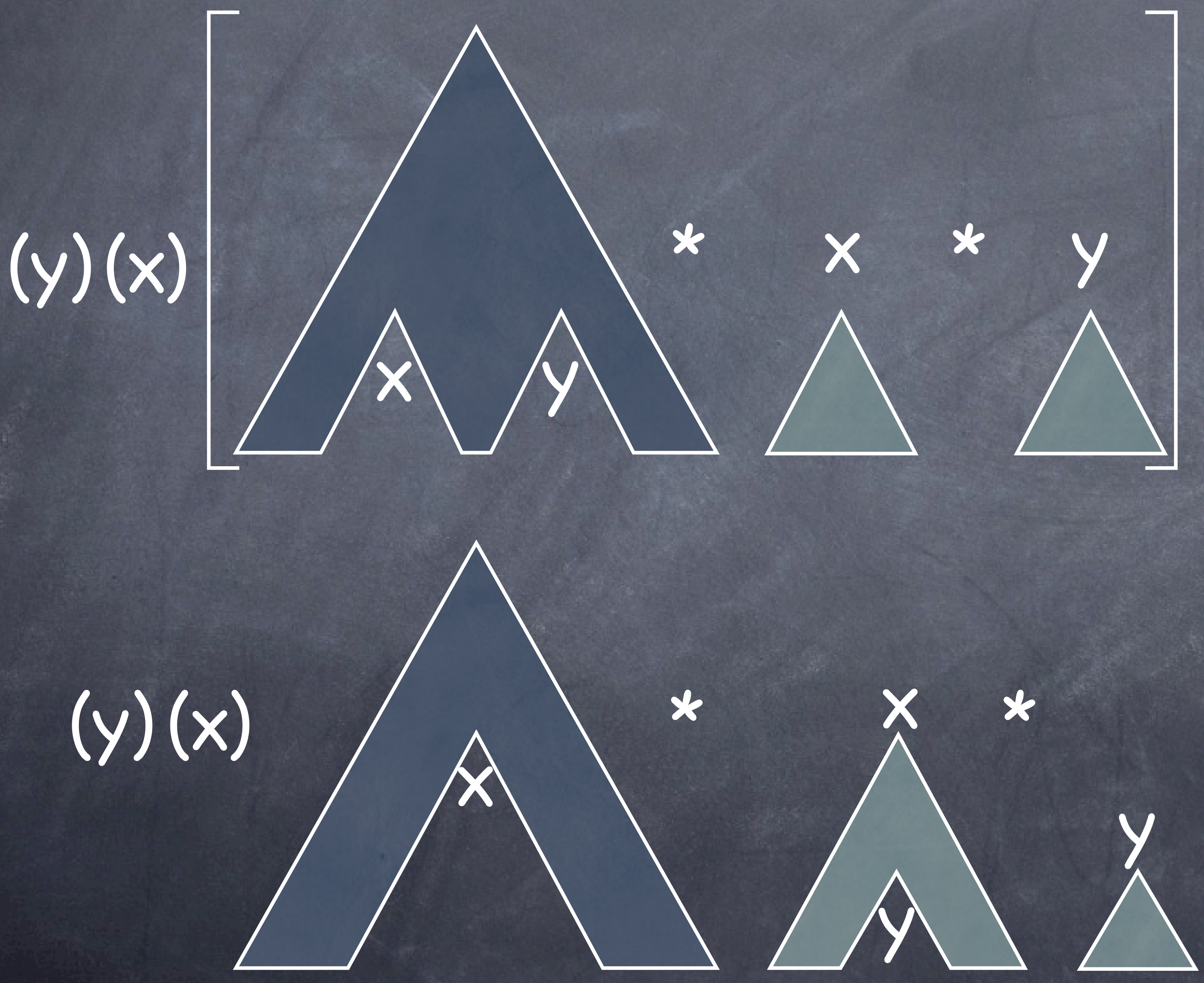
Segment Idea



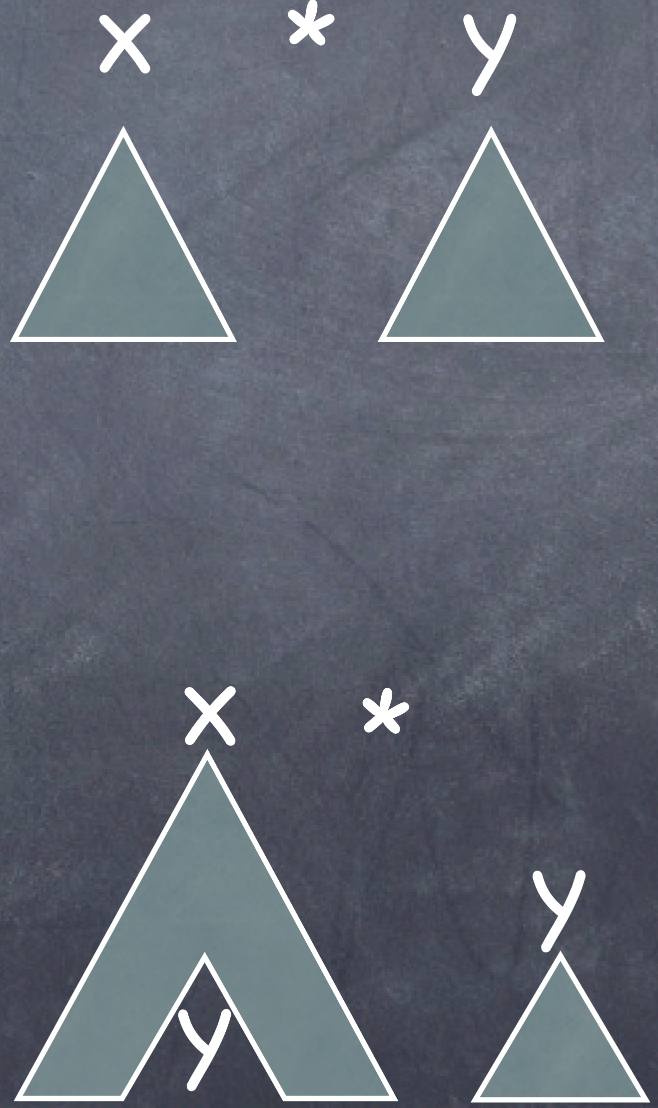
Segment Idea

Segment Idea

Segment Idea



Segment Idea



Tree Segments

tree context $c ::= \emptyset \mid x \mid n[c] \mid c \otimes c$

tree segment $s ::= \emptyset_S \mid x \leftarrow c \mid s + s \mid (x)(s)$

Unique node identifiers n

Unique free hole addresses $x\leftarrow$

Unique free hole labels x

$+$ associative & commutative with unit $\emptyset_S$

$\otimes$ associative with unit $\emptyset$

& no cycles!

Important Formula

Adjoints important for Weakest Preconditions

Structural formulae +

$P * Q$	separating conjunction
$P \mathrel{-\!\!*} Q$	separation right adjoint
$\alpha ® P$	revelation
$\alpha \mathrel{-\!\!®} Q$	revelation right adjoint
$\reflectbox{N}\alpha.P$	fresh label
$H\alpha.P$	derived label hiding

Fine-grained High-Level Reasoning

$\{ \alpha \leftarrow n[t] \}$
deleteTree(n)
$\{ \alpha \leftarrow \emptyset \}$

$\{ \alpha \leftarrow n[\gamma] * \beta \leftarrow m[t] \}$
append(n,m)
$\{ \alpha \leftarrow n[\gamma \otimes m[t]] * \beta \leftarrow \emptyset \}$

Fine-grained High-Level Reasoning

$\{ \alpha \leftarrow n[t] \}$
deleteTree(n)
$\{ \alpha \leftarrow \emptyset \}$

$\{ \alpha \leftarrow n[\gamma] * \beta \leftarrow m[t] \}$
append(n,m)
$\{ \alpha \leftarrow n[\gamma \otimes m[t]] * \beta \leftarrow \emptyset \}$

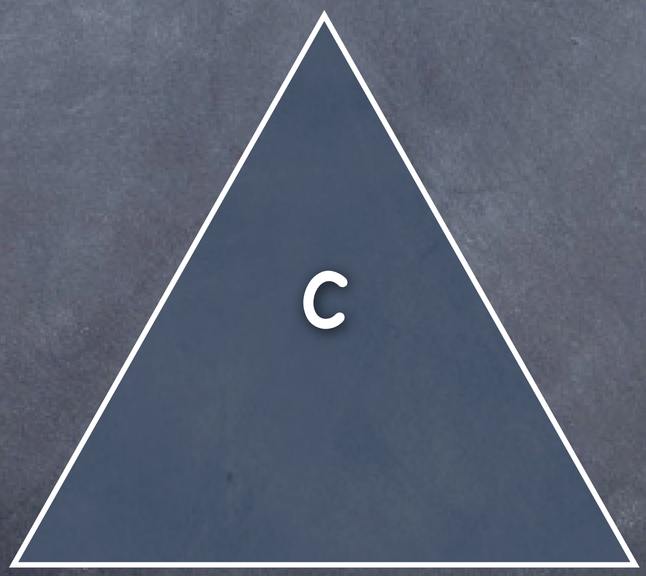
Small Axioms

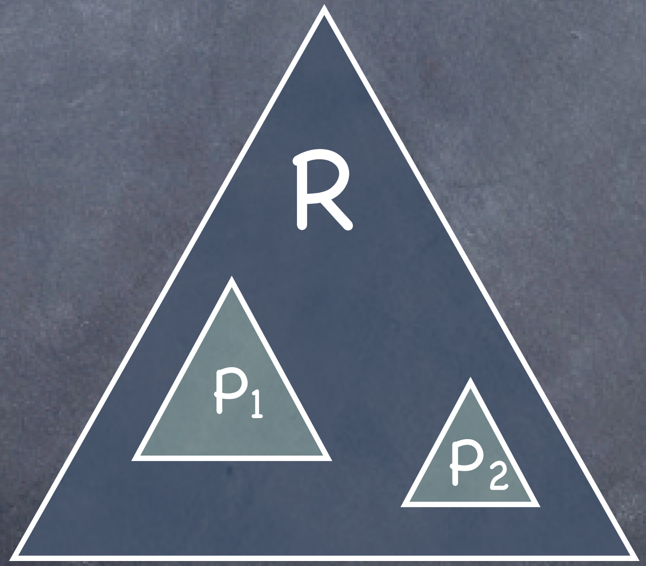
Hoare Rules

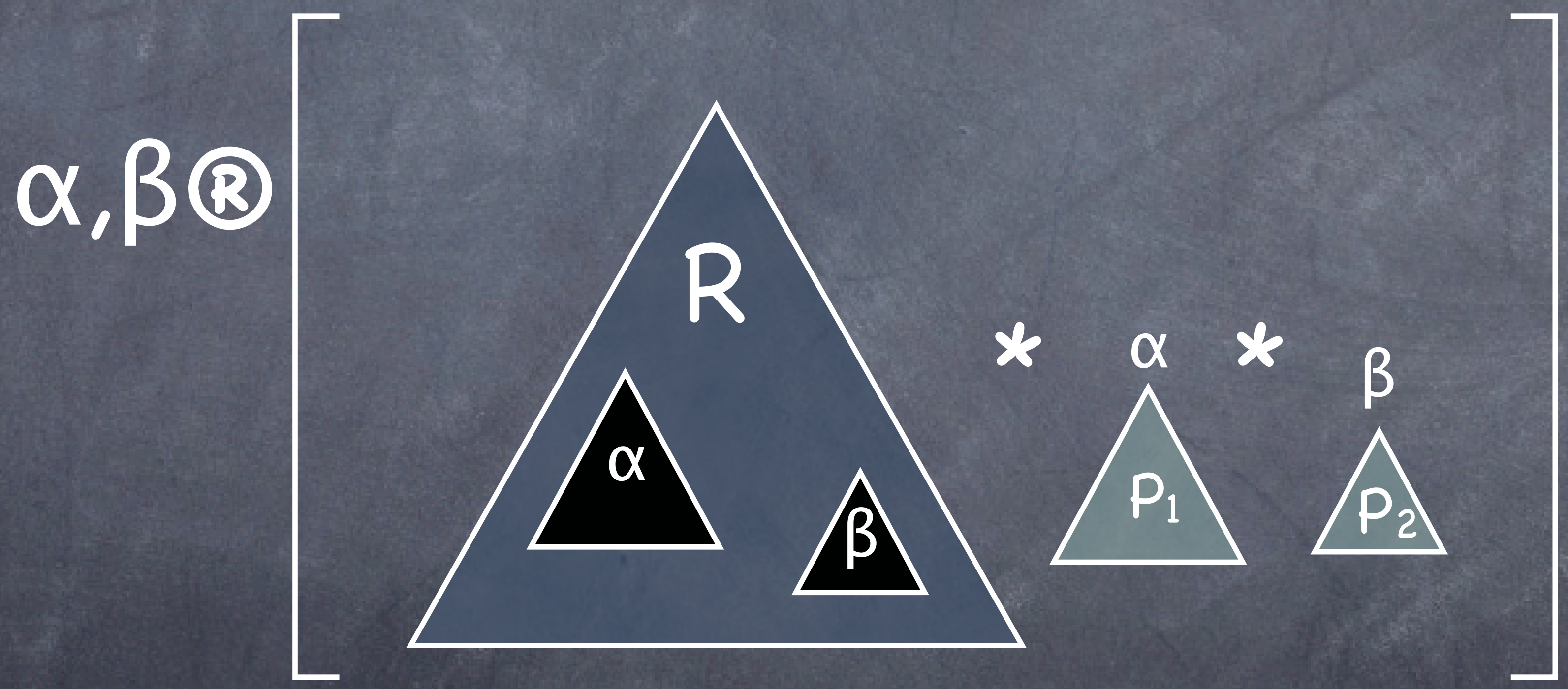
$$\frac{\{ P \} \quad \mathbb{C} \quad \{ Q \}}{\{ R * P \} \; \mathbb{C} \; \{ R * Q \}}$$

$$\frac{\{ P \} \quad \mathbb{C} \quad \{ Q \}}{\{ \text{И}\alpha.P \} \; \mathbb{C} \; \{ \text{И}\alpha.Q \}}$$

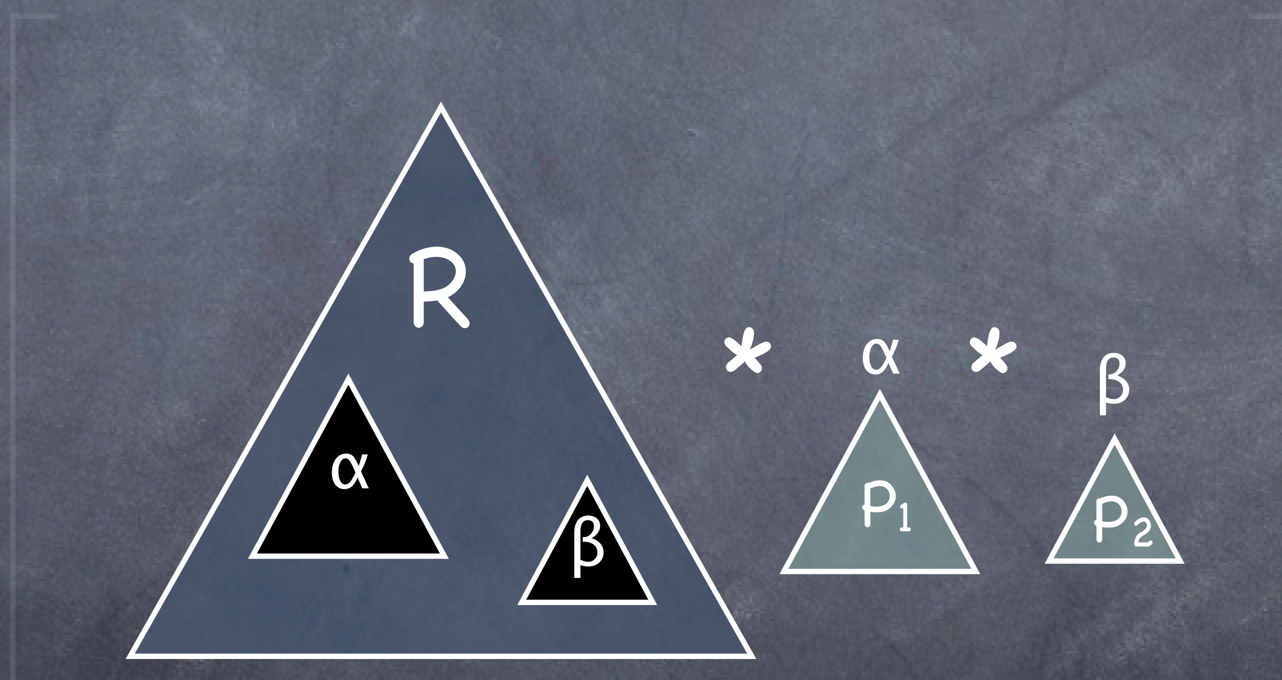
$$\frac{\{ P \} \quad \mathbb{C} \quad \{ Q \}}{\{ \alpha \text{®} P \} \; \mathbb{C} \; \{ \alpha \text{®} Q \}}$$


C


R
P1
P2

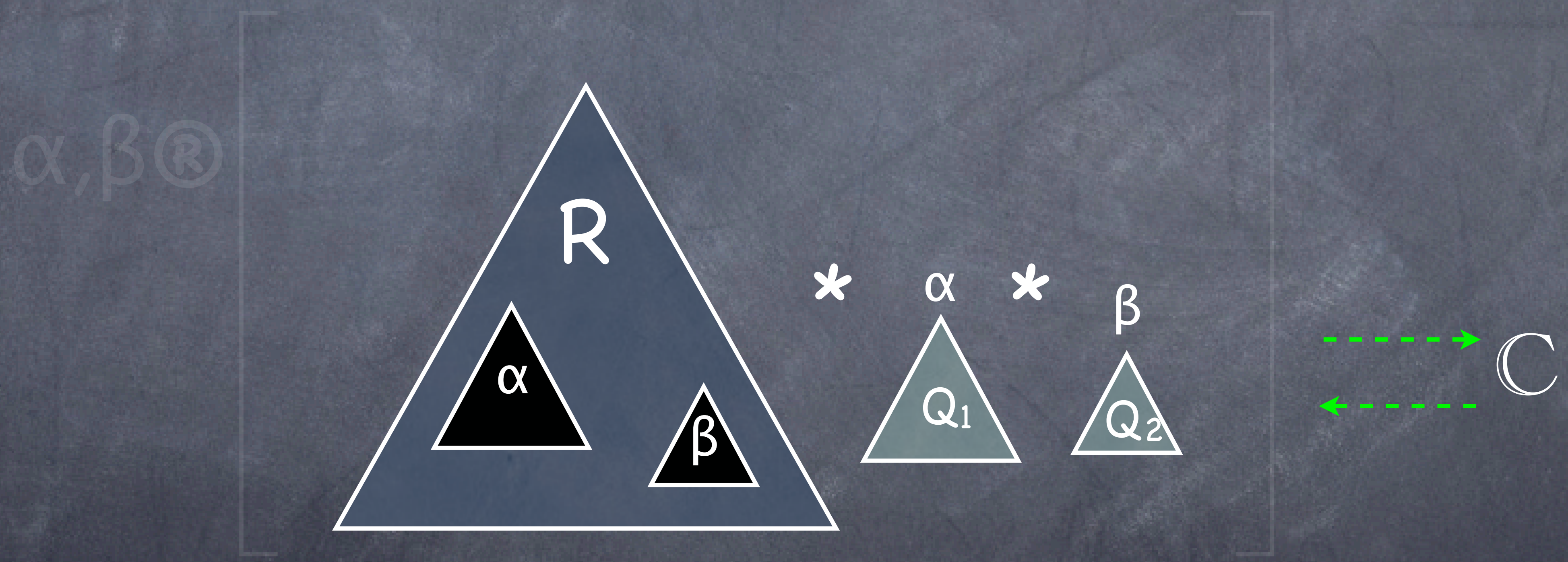

α,β®
R
α
β
*
α
P1
*
β
P2

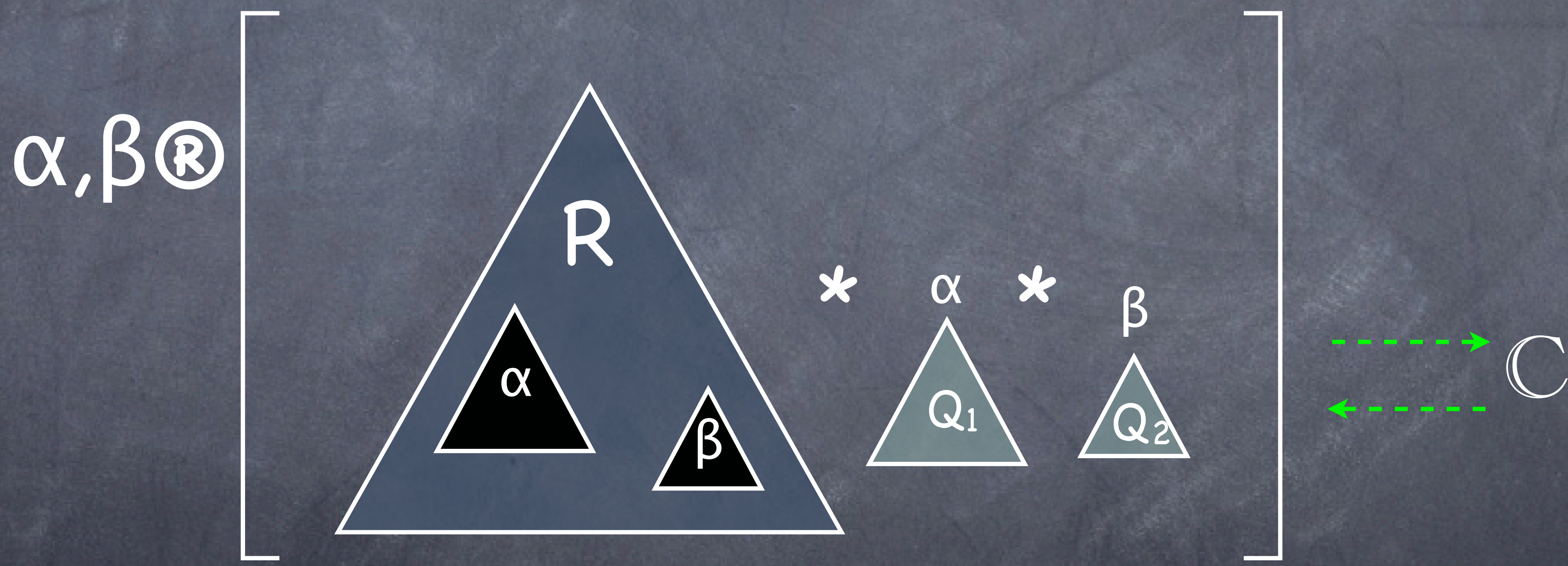
α,β®




α,β®
R
α
β
*
α
*
β
P1
P2

α,β®
R
α
β
*
α
*
β
Q1
Q2
ℂ


α,β®
R
α
β
*
α
*
β
Q1
Q2
ℂ


α,β®
R
α
β
*
α
*
β
Q1
Q2
ℂ


R
Q1
Q2
ℂ

Back to Disjoint Concurrency

$$\frac{\{ P_1 \}\ \mathbb{C}_1\ \{ Q_1 \} \quad \{ P_2 \}\ \mathbb{C}_2\ \{ Q_2 \}}{\{ P_1 * P_2 \}\ \mathbb{C}_1 \mid\mid \mathbb{C}_2\ \{ Q_1 * Q_2 \}}\ \text{PAR}$$

Disjoint Concurrency Example

deleteTree(n) || deleteTree(m)

Disjoint Concurrency Example

$\{ \alpha \leftarrow n[t] * \beta \leftarrow m[t'] \}$

deleteTree(n) || deleteTree(m)

Disjoint Concurrency Example

$\{ \alpha \leftarrow n[t] * \beta \leftarrow m[t'] \}$

$\{ \alpha \leftarrow n[t] \}$	$\|\|$	$\{ \beta \leftarrow m[t'] \}$
deleteTree(n)		deleteTree(m)

Disjoint Concurrency Example

$$\{\ \alpha \leftarrow n[t] * \beta \leftarrow m[t'] \ \}$$

$\{\ \alpha \leftarrow n[t]\ \}$	$\Vert$	$\{\ \beta \leftarrow m[t']\ \}$
deleteTree(n)	$\Vert$	deleteTree(m)
$\{\ \alpha \leftarrow \emptyset\ \}$	$\Vert$	$\{\ \beta \leftarrow \emptyset\ \}$

Disjoint Concurrency Example

$$\{ \alpha \leftarrow n[t] * \beta \leftarrow m[t'] \}$$

$\{ \alpha \leftarrow n[t] \}$	$\Vert$	$\{ \beta \leftarrow m[t'] \}$
deleteTree(n)		deleteTree(m)
$\{ \alpha \leftarrow \emptyset \}$		$\{ \beta \leftarrow \emptyset \}$

$$\{ \alpha \leftarrow \emptyset * \beta \leftarrow \emptyset \}$$

More Than Just Disjoint

$$\frac{\{\ P\ \}\ \mathbb{C}\ \{\ Q\ \}}{\{\ \pi_r®(RI_r * P)\ \}\ \text{res r in}\ \{\ \mathbb{C}\ \}\ \{\ \pi_r®(RI_r * Q)\ \}}\ \text{RES}$$

$$\frac{\{\ \pi_r®(RI_r * P \wedge B)\ \}\ \mathbb{C}\ \{\ \pi_r®(RI_r * Q)\ \}}{\{\ P\ \}\ \text{with r when B do}\ \{\ \mathbb{C}\ \}\ \{\ Q\ \}}\ \text{CCR}$$

Complex Concurrency Example

```
with r do{

    a := getLeft(n)

}

deleteTree(a)
```

||

```
with r do{
    b := getRight(n)
}
deleteTree(b)
```

Complex Concurrency Example

$RI_r = \alpha \leftarrow \beta \otimes n[\gamma] \otimes \delta$

$\pi_r = \beta, \delta$

```
with r do{
    a := getLeft(n)
}
deleteTree(a)
```

||

```
with r do{
    b := getRight(n)
}
deleteTree(b)
```

Complex Concurrency Example

$RI_r = \alpha \leftarrow \beta \otimes n[\gamma] \otimes \delta$
$\pi_r = \beta, \delta$

```
{ β←p[t] }
with r do{

    a := getLeft(n)

}

deleteTree(a)
```

||

```
with r do{
    b := getRight(n)
}
deleteTree(b)
```

Complex Concurrency Example

$RI_r = \alpha \leftarrow \beta \otimes n[\gamma] \otimes \delta$

$\pi_r = \beta, \delta$

$\{ \beta \leftarrow p[t] \}$

with r do{

$\{ \pi_r ® (RI_r * \beta \leftarrow p[t]) \}$

a := getLeft(n)

}

deleteTree(a)

||

with r do{

b := getRight(n)

}

deleteTree(b)

Complex Concurrency Example

$RI_r = \alpha \leftarrow \beta \otimes n[\gamma] \otimes \delta$

$\pi_r = \beta, \delta$

{ $\beta \leftarrow p[t]$ }

with r do{

  { π_r®($RI_r * \beta \leftarrow p[t]$) }

  { δ®($\alpha \leftarrow p[t] \otimes n[\gamma] \otimes \delta$) }

  a := getLeft(n)

}

deleteTree(a)

||

with r do{

  b := getRight(n)

}

deleteTree(b)

Complex Concurrency Example

$RI_r = \alpha \leftarrow \beta \circledast n[\gamma] \circledast \delta$

$\pi_r = \beta, \delta$

$\{ \beta \leftarrow p[t] \}$

with r do{

$\{ \pi_r ® (RI_r * \beta \leftarrow p[t]) \}$

$\{ \delta ® (\alpha \leftarrow p[t] \circledast n[\gamma] \circledast \delta) \}$

a := getLeft(n)

$\{ \delta ® (\alpha \leftarrow p[t] \circledast n[\gamma] \circledast \delta \wedge (a=p)) \}$

}

deleteTree(a)

||

with r do{

b := getRight(n)

}

deleteTree(b)

Complex Concurrency Example

$RI_r = \alpha \leftarrow \beta \circledast n[\gamma] \circledast \delta$

$\pi_r = \beta, \delta$

{ $\beta \leftarrow p[t]$ }

with r do{

{ π_r®$(RI_r * \beta \leftarrow p[t])$ }

{ δ®$(\alpha \leftarrow p[t] \otimes n[\gamma] \otimes \delta)$ }

a := getLeft(n)

{ δ®$(\alpha \leftarrow p[t] \otimes n[\gamma] \otimes \delta \wedge (a=p))$ }

{ π_r®$(RI_r * \beta \leftarrow p[t] \wedge (a=p))$ }

}

deleteTree(a)

||

with r do{

b := getRight(n)

}

deleteTree(b)

Complex Concurrency Example

$RI_r = \alpha \leftarrow \beta \otimes n[\gamma] \otimes \delta$

$\pi_r = \beta, \delta$

$\{ \beta \leftarrow p[t] \}$

with r do{

$\{ \pi_r ® (RI_r * \beta \leftarrow p[t]) \}$

$\{ \delta ® (\alpha \leftarrow p[t] \otimes n[\gamma] \otimes \delta) \}$

a := getLeft(n)

$\{ \delta ® (\alpha \leftarrow p[t] \otimes n[\gamma] \otimes \delta \wedge (a=p)) \}$

$\{ \pi_r ® (RI_r * \beta \leftarrow p[t] \wedge (a=p)) \}$

}

$\{ \beta \leftarrow p[t] \wedge (a=p) \}$

deleteTree(a)

||

with r do{

b := getRight(n)

}

deleteTree(b)

Complex Concurrency Example

$RI_r = \alpha \leftarrow \beta \otimes n[\gamma] \otimes \delta$

$\pi_r = \beta, \delta$

$\{ \beta \leftarrow p[t] \}$

with r do{

$\{ \pi_r ® (RI_r * \beta \leftarrow p[t]) \}$

$\{ \delta ® (\alpha \leftarrow p[t] \otimes n[\gamma] \otimes \delta) \}$

a := getLeft(n)

$\{ \delta ® (\alpha \leftarrow p[t] \otimes n[\gamma] \otimes \delta \wedge (a=p)) \}$

$\{ \pi_r ® (RI_r * \beta \leftarrow p[t] \wedge (a=p)) \}$

}

$\{ \beta \leftarrow p[t] \wedge (a=p) \}$

deleteTree(a)

$\{ \beta \leftarrow \emptyset \}$

||

with r do{

b := getRight(n)

}

deleteTree(b)

Complex Concurrency Example

$RI_r = \alpha \leftarrow \beta \otimes n[\gamma] \otimes \delta$

$\pi_r = \beta, \delta$

$\{ \beta \leftarrow p[t] \}$

with r do{

$\{ \pi_r ® (RI_r * \beta \leftarrow p[t]) \}$

$\{ \delta ® (\alpha \leftarrow p[t] \otimes n[\gamma] \otimes \delta) \}$

a := getLeft(n)

$\{ \delta ® (\alpha \leftarrow p[t] \otimes n[\gamma] \otimes \delta \wedge (a=p)) \}$

$\{ \pi_r ® (RI_r * \beta \leftarrow p[t] \wedge (a=p)) \}$

}

$\{ \beta \leftarrow p[t] \wedge (a=p) \}$

deleteTree(a)

$\{ \beta \leftarrow \emptyset \}$

||

$\{ \delta \leftarrow q[t'] \}$

with r do{

b := getRight(n)

}

deleteTree(b)

$\{ \delta \leftarrow \emptyset \}$

Summary

- Separation Logic allows low-level local reasoning for sequential and concurrent programs.
- Context Logic allows high-level local reasoning for sequential programs.
- Small axioms are necessary for local reasoning about concurrency.
- Segment Logic allows high-level local reasoning about concurrent programs.

Ongoing Research

- Abstract Local Reasoning : translating from specification to implementation.
  (with Philippa Gardner and Thomas Dinsdale-Young - Imperial)

- Concurrent XML Update : designing and specifying a language in the style of DOM.
  (with James Kearney - Imperial)

- Applying Rely-Guarantee and Deny-Guarantee techniques to BTrees.
  (with Pedro da Rocha Pinto and Thomas Dinsdale-Young - Imperial)

Thanks for Listening

Any Questions ?