Small Specifications for Tree Update

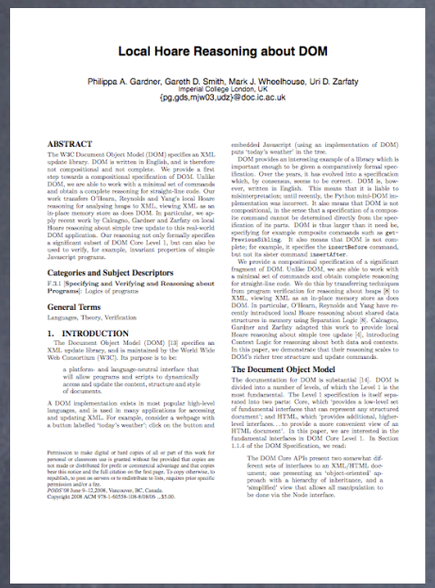
WSFM'09

Mark Wheelhouse
Philippa Gardner

Imperial College London

with thanks to Thomas Dinsdale-Young

Introduction



Local Hoare Reasoning about DOM

Philippa A. Gardner, Gareth D. Smith, Mark J. Wheelhouse, Uri D. Zarfaty
Imperial College London, UK
{pg,gds,mjw03,udz}@doc.ic.ac.uk

ABSTRACT

The W3C Document Object Model (DOM) specifies an XML update library. DOM is written in English, and is therefore not compositional and not complete. We provide a first step towards a compositional specification of DOM. Unlike DOM, we are able to work with a minimal set of commands and obtain a complete reasoning for straight-line code. Our work transfers O'Hearn, Reynolds and Yang's local Hoare reasoning for analysing heaps to XML, viewing XML as an in-place memory store as does DOM. In particular, we apply recent work by Calcagno, Gardner and Zarfaty on local Hoare reasoning about simple tree update to this real-world DOM application. Our reasoning not only formally specifies a significant subset of DOM Core Level 1, but can also be used to verify, for example, invariant properties of simple Javascript programs.

Categories and Subject Descriptors

F.3.1 [**Specifying and Verifying and Reasoning about Programs**]: Logics of programs

General Terms

Languages, Theory, Verification

1. INTRODUCTION

The Document Object Model (DOM) [13] specifies an XML update library, and is maintained by the World Wide Web Consortium (W3C). Its purpose is to be:

> a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents.

A DOM implementation exists in most popular high-level languages, and is used in many applications for accessing and updating XML. For example, consider a webpage with a button labelled 'today's weather'; click on the button and embedded Javascript (using an implementation of DOM) puts 'today's weather' in the tree.

DOM provides an interesting example of a library which is important enough to be given a comparatively formal specification. Over the years, it has evolved into a specification which, by consensus, seems to be correct. DOM is, however, written in English. This means that it is liable to misinterpretation; until recently, the Python mini-DOM implementation was incorrect. It also means that DOM is not compositional, in the sense that a specification of a composite command cannot be determined directly from the specification of its parts. DOM is thus larger than it need be, specifying for example composite commands such as `getPreviousSibling`. It also means that DOM is not complete; for example, it specifies the `insertBefore` command, but not its sister command `insertAfter`.

We provide a compositional specification of a significant fragment of DOM. Unlike DOM, we are able to work with a minimal set of commands and obtain complete reasoning for straight-line code. We do this by transferring techniques from program verification for reasoning about heaps [8] to XML, viewing XML as an in-place memory store as does DOM. In particular, O'Hearn, Reynolds and Yang have recently introduced local Hoare reasoning about shared data structures in memory using Separation Logic [8]. Calcagno, Gardner and Zarfaty adapted this work to provide local Hoare reasoning about simple tree update [5], introducing Context Logic for reasoning about both data and contexts. In this paper, we demonstrate that their reasoning scales to DOM's richer tree structure and update commands.

The Document Object Model

The documentation for DOM is substantial [14]. DOM is divided into a number of levels, of which the Level 1 is the most fundamental. The Level 1 specification is itself separated into two parts: Core, which 'provides a low-level set of fundamental interfaces that can represent any structured document'; and HTML, which 'provides additional, higher-level interfaces... to provide a more convenient view of an HTML document'. In this paper, we are interested in the fundamental interfaces in DOM Core Level 1. In Section 1.1.4 of the DOM Specification, we read:

> The DOM Core APIs present two somewhat different sets of interfaces to an XML/HTML document; one presenting an 'object-oriented' approach with a hierarchy of inheritance, and a 'simplified' view that allows all manipulation to be done via the Node interface.

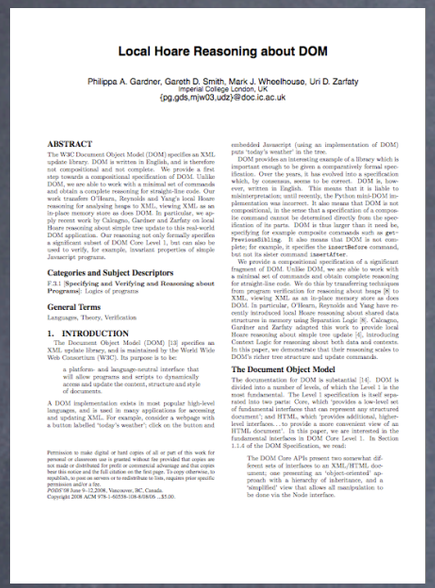
PODS'08 June 9–12,2008, Vancouver, BC, Canada.

Local Hoare Reasoning About DOM

Philippa Gardner, Gareth Smith,
Mark Wheelhose, Uri Zarfaty

(PODS 2008)

Introduction



Local Hoare Reasoning about DOM

Philippa A. Gardner, Gareth D. Smith, Mark J. Wheelhouse, Uri D. Zarfaty
Imperial College London, UK
{pg,gds,mjw03,udz}@doc.ic.ac.uk

ABSTRACT

The W3C Document Object Model (DOM) specifies an XML update library. DOM is written in English, and is therefore not compositional and not complete. We provide a first step towards a compositional specification of DOM. Unlike DOM, we are able to work with a minimal set of commands and obtain a complete reasoning for straight-line code. Our work transfers O'Hearn, Reynolds and Yang's local Hoare reasoning for analysing heaps to XML, viewing XML as an in-place ternary store as does DOM. In particular, we apply recent work by Calcagno, Gardner and Zarfaty on local Hoare reasoning about simple tree update to this real-world DOM application. Our reasoning not only formally specifies a significant subset of DOM Core Level 1, but can also be used to verify, for example, invariant properties of simple Javascript programs.

Categories and Subject Descriptors

F.3.1 [**Specifying and Verifying and Reasoning about Programs**]: Logics of programs

General Terms

Languages, Theory, Verification

1. INTRODUCTION

The Document Object Model (DOM) [13] specifies an XML update library, and is maintained by the World Wide Web Consortium (W3C). Its purpose is to be:

> a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents.

A DOM implementation exists in most popular high-level languages, and is used in many applications for accessing and updating XML. For example, consider a webpage with a button labelled 'today's weather'; click on the button and embedded Javascript (using an implementation of DOM) puts 'today's weather' in the tree.

DOM provides an interesting example of a library which is important enough to be given a comparatively formal specification. Over the years, it has evolved into a specification which, by consensus, seems to be correct. DOM is, however, written in English. This means that it is liable to misinterpretation; until recently, the Python mini-DOM implementation was incorrect. It also means that DOM is not compositional, in the sense that a specification of a composite command cannot be determined directly from the specification of its parts. DOM is thus larger than it need be, specifying for example composite commands such as `getPreviousSibling`. It also means that DOM is not complete; for example, it specifies the `insertBefore` command, but not its sister command `insertAfter`.

We provide a compositional specification of a significant fragment of DOM. Unlike DOM, we are able to work with a minimal set of commands and obtain complete reasoning for straight-line code. We do this by transferring techniques from program verification for reasoning about heaps [8] to XML, viewing XML as an in-place ternary store as does DOM. In particular, O'Hearn, Reynolds and Yang have recently introduced local Hoare reasoning about shared data structures in memory using Separation Logic [8]. Calcagno, Gardner and Zarfaty adapted this work to provide local Hoare reasoning about simple tree update [5], introducing Context Logic for reasoning about both data and contexts. In this paper, we demonstrate that their reasoning scales to DOM's richer tree structure and update commands.

The Document Object Model

The documentation for DOM is substantial [14]. DOM is divided into a number of levels, of which the Level 1 is the most fundamental. The Level 1 specification is itself separated into two parts: Core, which 'provides a low-level set of fundamental interfaces that can represent any structured document'; and HTML, which 'provides additional, higher-level interfaces... to provide a more convenient view of an HTML document'. In this paper, we are interested in the fundamental interfaces in DOM Core Level 1. In Section 1.1.4 of the DOM Specification, we read:

> The DOM Core APIs present two somewhat different sets of interfaces to an XML/HTML document; one presenting an 'object-oriented' approach with a hierarchy of inheritance, and a 'simplified' view that allows all manipulation to be done via the Node interface.

PODS'08 June 9–12,2008, Vancouver, BC, Canada.

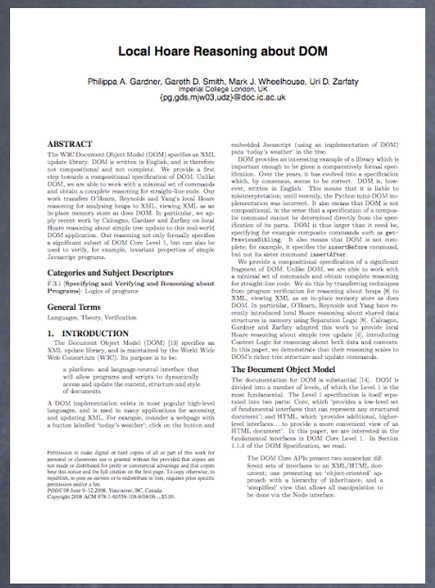
Local Hoare Reasoning About DOM

Philippa Gardner, Gareth Smith,
Mark Wheelhose, Uri Zarfaty

(PODS 2008)

Context Logic

Introduction



Local Hoare Reasoning about DOM

Philippa A. Gardner, Gareth D. Smith, Mark J. Wheelhouse, Uri D. Zarfaty
Imperial College London, UK
{pg,gds,mjw03,udz}@doc.ic.ac.uk

ABSTRACT

The W3C Document Object Model (DOM) specifies an XML update library. DOM is written in English, and is therefore not compositional and not complete. We provide a first step towards a compositional specification of DOM. Unlike DOM, we are able to work with a minimal set of commands and obtain a complete reasoning for straight-line code. Our work transfers O'Hearn, Reynolds and Yang's local Hoare reasoning for analysing heaps to XML, viewing XML as an in-place memory store as does DOM. In particular, we apply recent work by Calcagno, Gardner and Zarfaty on local Hoare reasoning about simple tree update to this real-world DOM application. Our reasoning not only formally specifies a significant subset of DOM Core Level 1, but can also be used to verify, for example, invariant properties of simple Javascript programs.

Categories and Subject Descriptors

F.3.1 [**Specifying and Verifying and Reasoning about Programs**]: Logics of programs

General Terms

Languages, Theory, Verification

1. INTRODUCTION

The Document Object Model (DOM) [13] specifies an XML update library, and is maintained by the World Wide Web Consortium (W3C). Its purpose is to be:

> a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents.

A DOM implementation exists in most popular high-level languages, and is used in many applications for accessing and updating XML. For example, consider a webpage with a button labelled 'today's weather'; click on the button and embedded Javascript (using an implementation of DOM) puts 'today's weather' in the tree.

DOM provides an interesting example of a library which is important enough to be given a comparatively formal specification. Over the years, it has evolved into a specification which, by consensus, seems to be correct. DOM is, however, written in English. This means that it is liable to misinterpretation; until recently, the Python mini-DOM implementation was incorrect. It also means that DOM is not compositional, in the sense that a specification of a composite command cannot be determined directly from the specification of its parts. DOM is thus larger than it need be, specifying for example composite commands such as `getPreviousSibling`. It also means that DOM is not complete; for example, it specifies the `insertBefore` command, but not its sister command `insertAfter`.

We provide a compositional specification of a significant fragment of DOM. Unlike DOM, we are able to work with a minimal set of commands and obtain complete reasoning for straight-line code. We do this by transferring techniques from program verification for reasoning about heaps [8] to XML, viewing XML as an in-place memory store as does DOM. In particular, O'Hearn, Reynolds and Yang have recently introduced local Hoare reasoning about shared data structures in memory using Separation Logic [8]. Calcagno, Gardner and Zarfaty adapted this work to provide local Hoare reasoning about simple tree update [5], introducing Context Logic for reasoning about both data and contexts. In this paper, we demonstrate that their reasoning scales to DOM's richer tree structure and update commands.

The Document Object Model

The documentation for DOM is substantial [14]. DOM is divided into a number of levels, of which the Level 1 is the most fundamental. The Level 1 specification is itself separated into two parts: Core, which 'provides a low-level set of fundamental interfaces that can represent any structured document'; and HTML, which 'provides additional, higher-level interfaces... to provide a more convenient view of an HTML document'. In this paper, we are interested in the fundamental interfaces in DOM Core Level 1. In Section 1.1.4 of the DOM Specification, we read:

> The DOM Core APIs present two somewhat different sets of interfaces to an XML/HTML document; one presenting an 'object-oriented' approach with a hierarchy of inheritance, and a 'simplified' view that allows all manipulation to be done via the Node interface.

PODS'08 June 9–12,2008, Vancouver, BC, Canada.

Local Hoare Reasoning About DOM

Philippa Gardner, Gareth Smith,
Mark Wheelhose, Uri Zarfaty

(PODS 2008)

Context Logic

Segment Logic

Trees on the Web

XML

```
<head>
  <title> My Webpage <\title>
<\head>

</body>
  <heading> Segment Logic <\heading>
  <p>
    This site will tell you everything you need to
    know about Segment Logic. We'll show how
    it relates to Context Logic and Separation
    Logic and explain the choices we have made.
  </p>
  <br>
  <a href="www.CL.com"> Context Logic </a><br>
  <a href="www.SL.com"> Separation Logic <\a><br>
  .
  .
  .

</body>
```

Trees on the Web

XML

```
<head>
  <title> My Webpage <\title>
<\head>

</body>
  <heading> Segment Logic <\heading>
  <p>
    This site will tell you everything you need to
    know about Segment Logic. We'll show how
    it relates to Context Logic and Separation
    Logic and explain the choices we have made.
  </p>
  <br>
  <a href="www.CL.com"> Context Logic </a><br>
  <a href="www.SL.com"> Separation Logic <\a><br>
  .
  .
  .

</body>
```

Trees on the Web

XML

```
<head>
  <title> My Webpage <\title>
<\head>

</body>
  <heading> Segment Logic <\heading>
  <p>
    This site will tell you everything you need to
    know about Segment Logic. We'll show how
    it relates to Context Logic and Separation
    Logic and explain the choices we have made.
  </p>
  <br>
  <a href="www.CL.com"> Context Logic </a><br>
  <a href="www.SL.com"> Separation Logic <\a><br>
  .
  .
  .

</body>
```

Trees on the Web

XML

```
<head>
  <title> My Webpage <\title>
<\head>

</body>
  <heading> Segment Logic <\heading>
  <p>
    This site will tell you everything you need to
    know about Segment Logic. We'll show how
    it relates to Context Logic and Separation
    Logic and explain the choices we have made.
  </p>
  <br>
  <a href="www.CL.com"> Context Logic </a><br>
  <a href="www.SL.com"> Separation Logic <\a><br>
  .
  .
  .

</body>
```

Tree Structure

tree $t ::= \emptyset \mid n[t] \mid t \otimes t$

Unique node identifiers n

$\otimes$ associative with unit $\emptyset$

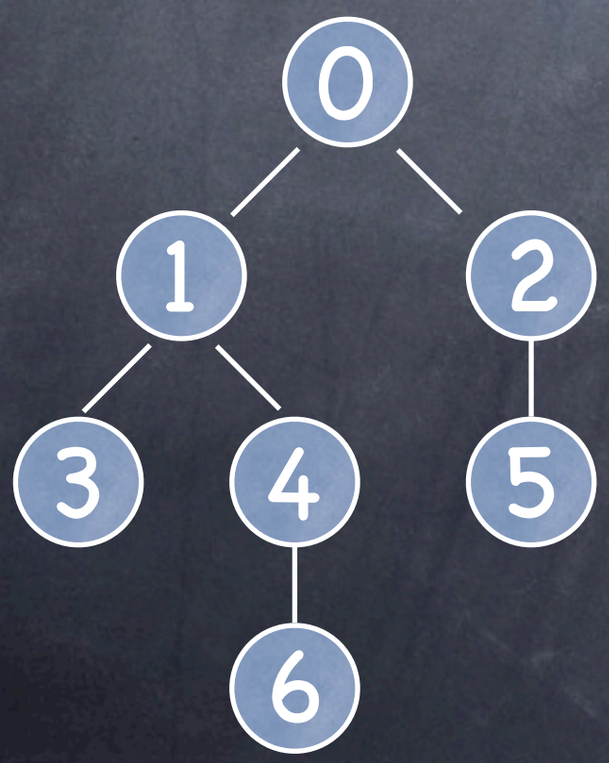
Tree Structure

tree t ::= ∅ | n[t] | t ⊗ t

Unique node identifiers n

⊗ associative with unit ∅

0[1[3[∅]⊗ 4[6[∅]]]⊗ 2[5[∅]]]



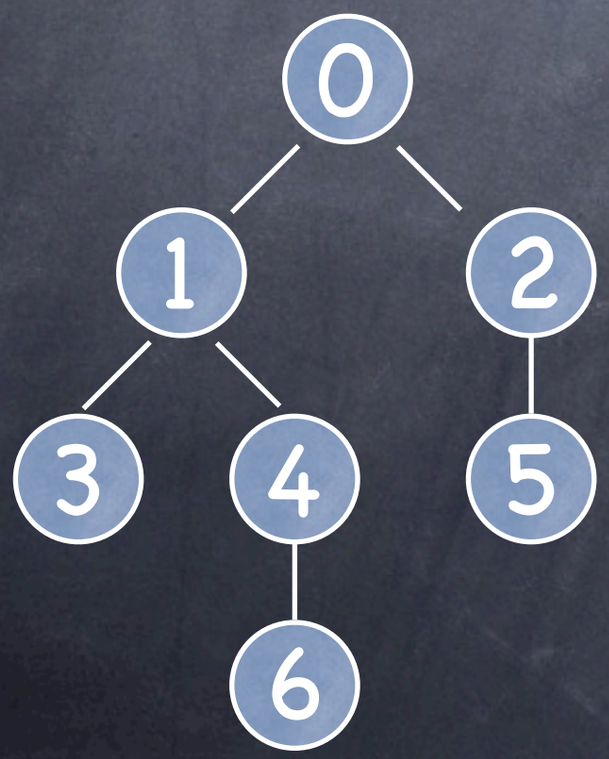
Tree Structure

tree $t ::= \emptyset \mid n[t] \mid t \otimes t$

Unique node identifiers n

$\otimes$ associative with unit $\emptyset$

$0[\ 1[3 \otimes 4[6]] \otimes 2[5]\ ]$



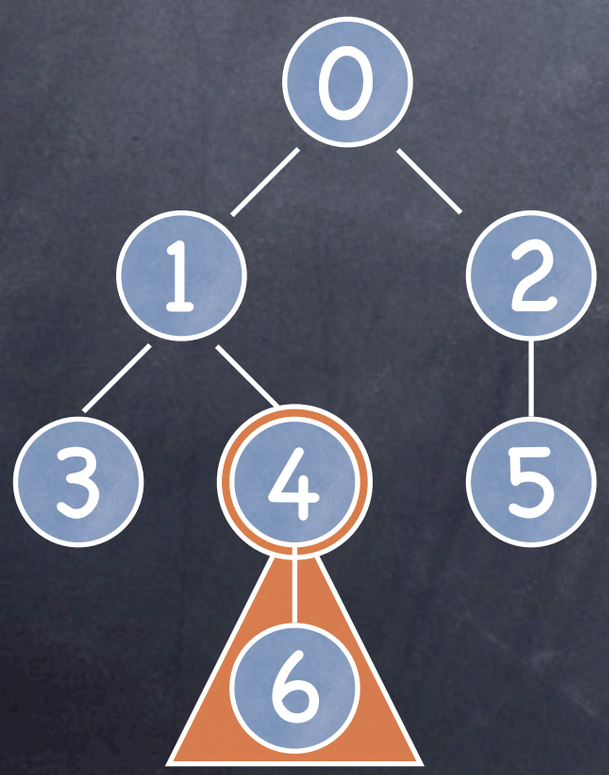
Tree Structure

tree t ::= ∅ | n[t] | t ⊗ t

Unique node identifiers n

⊗ associative with unit ∅

0[1[3 ⊗ 4[6]] ⊗ 2[5]]



delete(4)

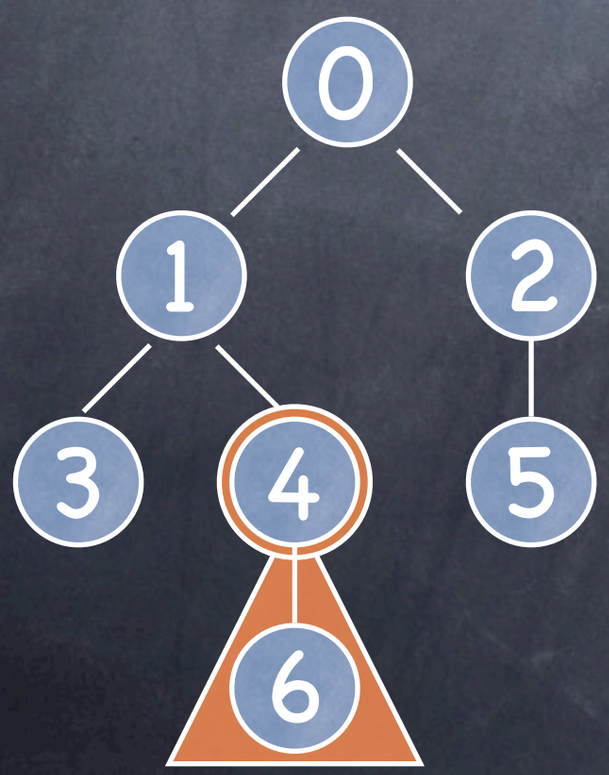
Tree Structure

tree t ::= ∅ | n[t] | t ⊗ t

Unique node identifiers n

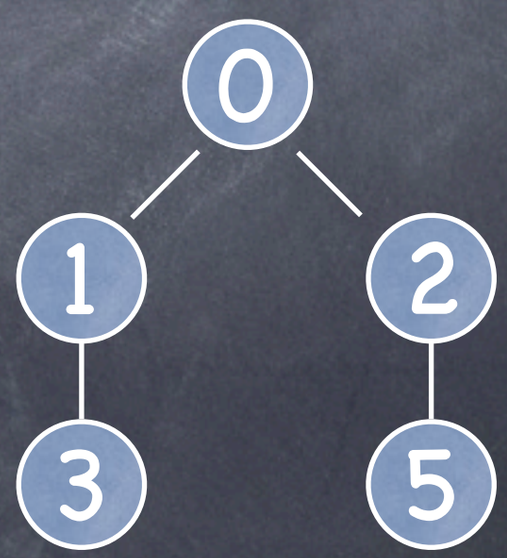
⊗ associative with unit ∅

0[1[3 ⊗ 4[6]] ⊗ 2[5]]



delete(4)

0[1[3] ⊗ 2[5]]



Contexts

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

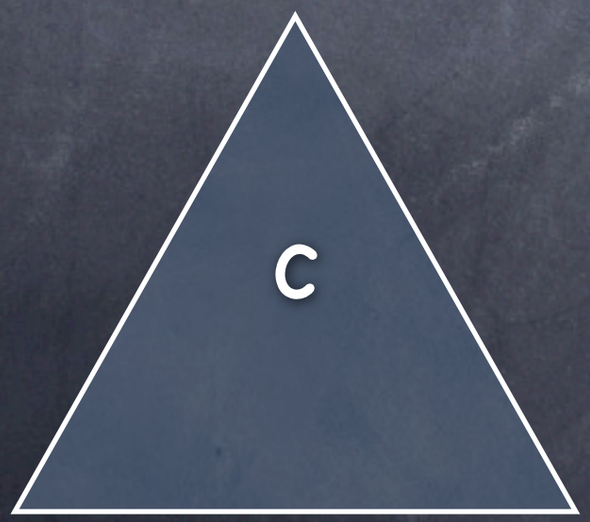
Unique node identifiers n
$\otimes$ associative with unit $\emptyset$
Unique hole labels x

Contexts

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

Unique node identifiers n
⊗ associative with unit ∅
Unique hole labels x

Context Application



Contexts

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

Unique node identifiers n
⊗ associative with unit ∅
Unique hole labels x

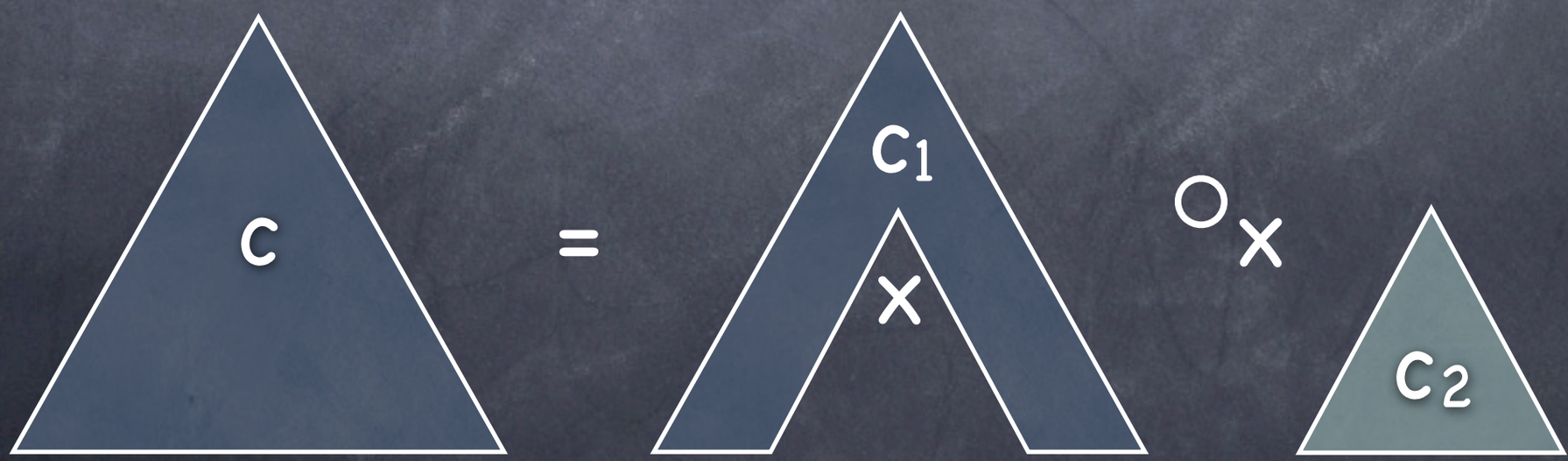
Context Application

Contexts

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

Unique node identifiers n
$\otimes$ associative with unit $\emptyset$
Unique hole labels x

Context Application



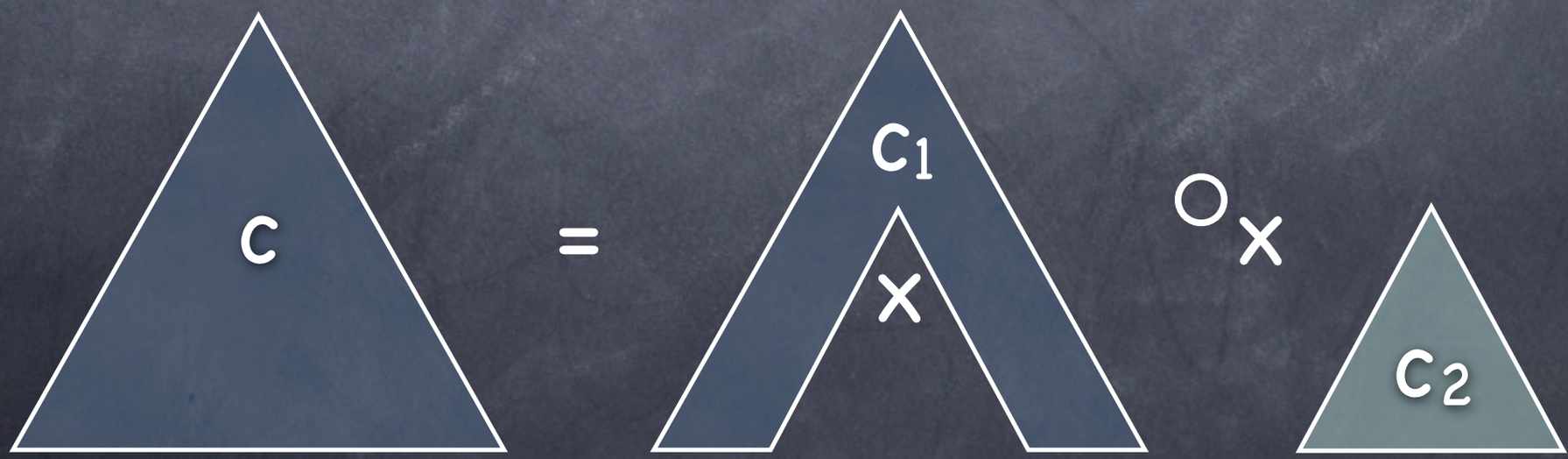
Contexts

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

Unique node identifiers n
$\otimes$ associative with unit $\emptyset$
Unique hole labels x

Context Application

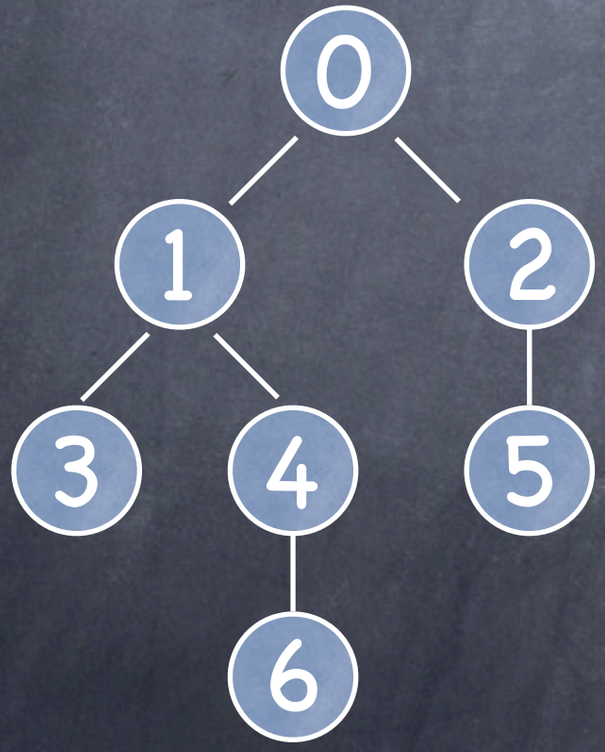
$\lambda x\ (c_1[x])(c_2)$



Local Update

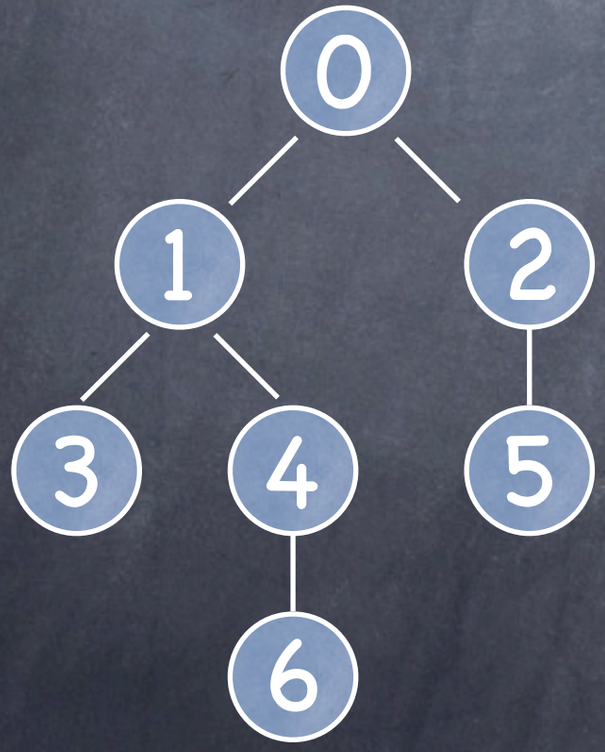
Local Update

$0[1[3 \otimes 4[6]] \otimes 2[5]]$



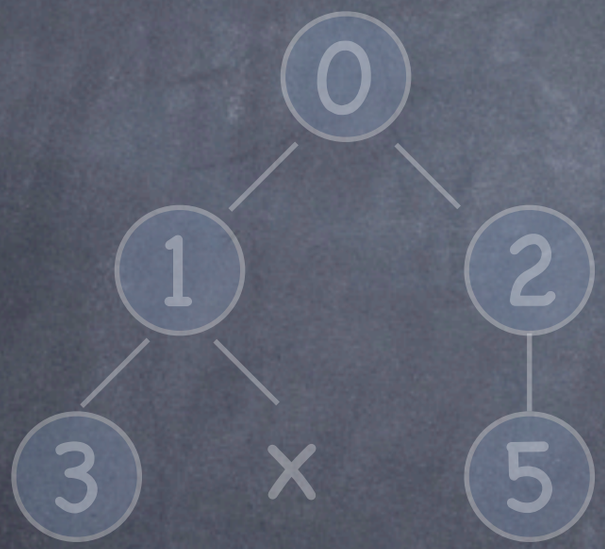
Local Update

$$0[1[3 \otimes 4[6]] \otimes 2[5]] \quad = \quad 0[1[3 \otimes x] \otimes 2[5]] \circ_x 4[6]$$

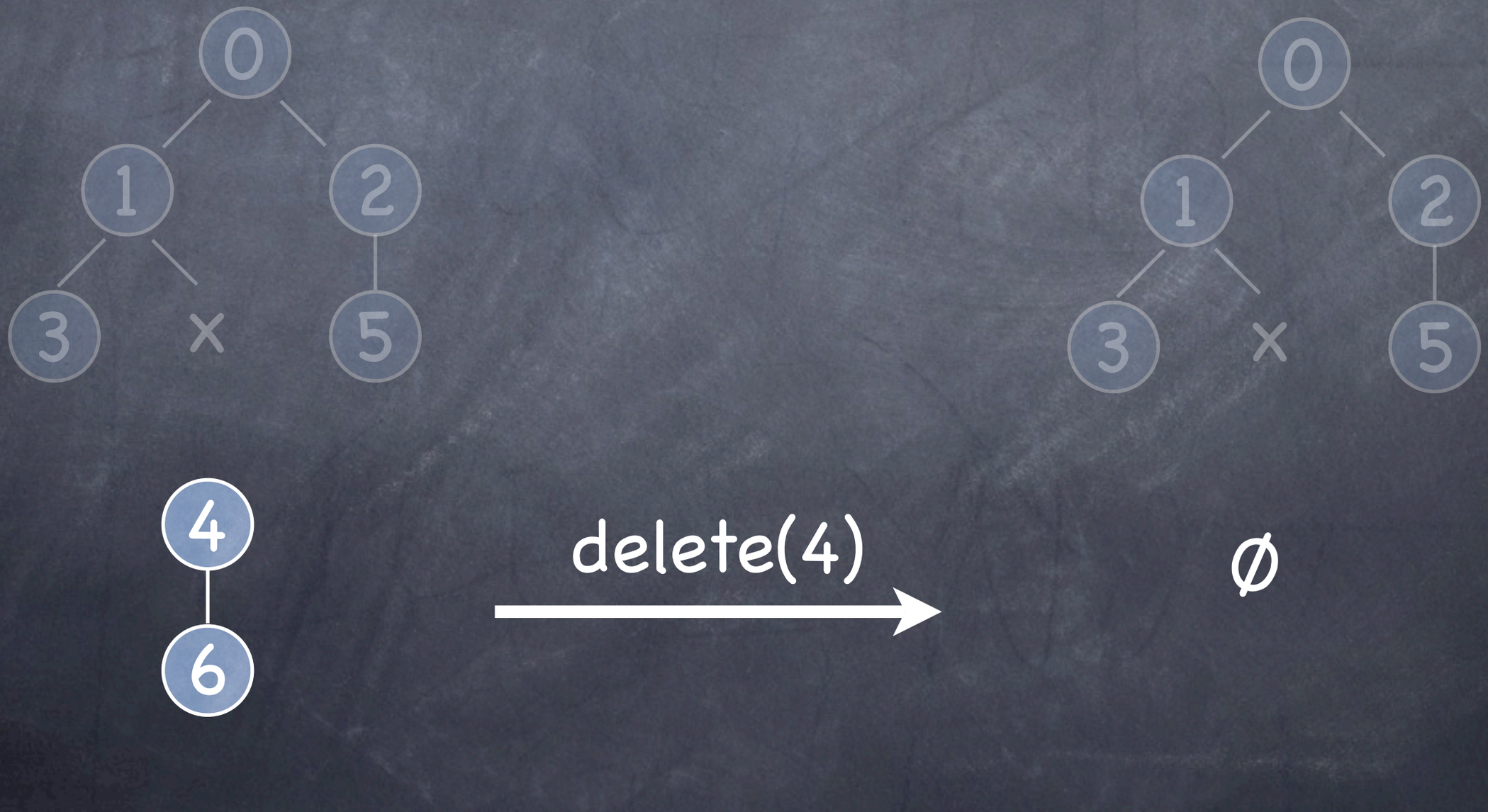


Local Update

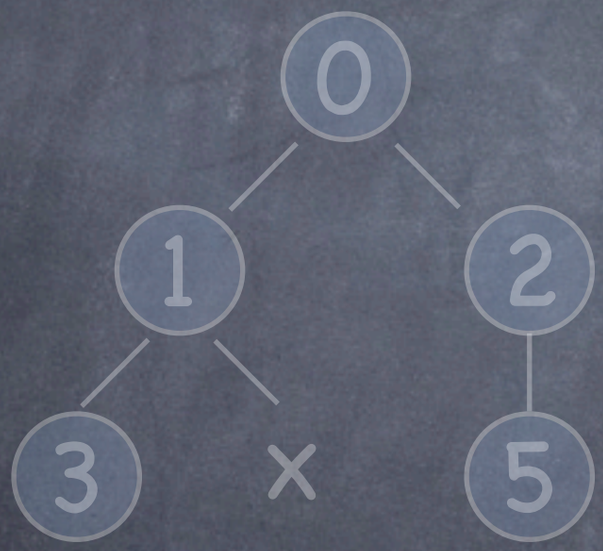
$$0[1[3 \otimes 4[6]] \otimes 2[5]] \quad = \quad 0[1[3 \otimes x] \otimes 2[5]] \circ_x 4[6]$$



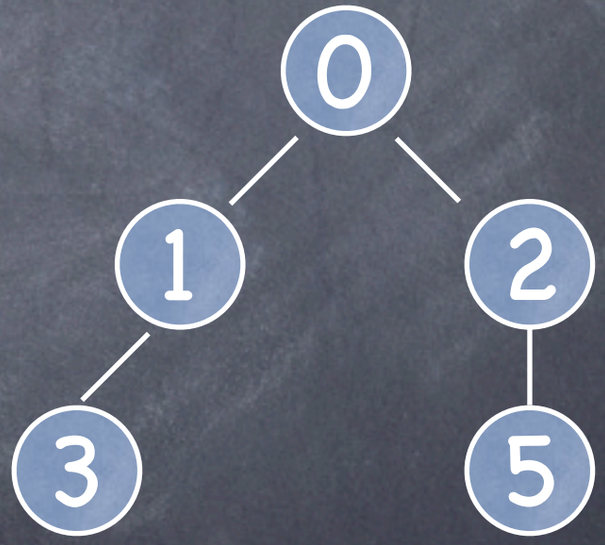
Local Update



Local Update



delete(4)



Local Reasoning - delete

{ n[t] } delete(n) { ∅ }

Local Reasoning - delete

Small Axiom { n[t] } delete(n) { ∅ }

Local Reasoning - delete

Small Axiom { n[t] } delete(n) { ∅ }

{ 4[6] } delete(4) { ∅ }

Local Reasoning - delete

Small Axiom $\{ n[t] \}\ \mathrm{delete}(n)\ \{ \emptyset \}$

$\{ 4[6] \}\ \mathrm{delete}(4)\ \{ \emptyset \}$

$$\frac{\{ n[t] \}\ \mathrm{delete}(n)\ \{ \emptyset \}}{\{ P \circledast n[t] \}\ \mathrm{delete}(n)\ \{ P \circledast \emptyset \}}$$

Local Reasoning - delete

Small Axiom

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

$$\{\ 4[6]\ \}\ \text{delete}(4)\ \{\ \emptyset\ \}$$

Frame

$$\frac{\{\ n[t]\ \}\ \text{delete}(n)\ \{\ \emptyset\ \}}{\{\ P \circledast n[t]\ \}\ \text{delete}(n)\ \{\ P \circledast \emptyset\ \}}$$

Local Reasoning - delete

Small Axiom

$$\{\ n[t]\ \}\ \mathrm{delete}(n)\ \{\ \emptyset\ \}$$

$$\{\ 4[6]\ \}\ \mathrm{delete}(4)\ \{\ \emptyset\ \}$$

Frame

$$\frac{\{\ n[t]\ \}\ \mathrm{delete}(n)\ \{\ \emptyset\ \}}{\{\ P \circ_x n[t]\ \}\ \mathrm{delete}(n)\ \{\ P \circ_x \emptyset\ \}}$$

$$\frac{\{\ 4[6]\ \}\ \mathrm{delete}(4)\ \{\ \emptyset\ \}}{\{\ 0[1[3 \otimes x] \otimes 2[5]] \circ_x 4[6]\ \}\ \mathrm{delete}(4)\ \{\ 0[1[3 \otimes x] \otimes 2[5]] \circ_x \emptyset\ \}}$$

Single-point Update

delete(n)

write(n, data)

n′ := getParent(n)

n′ := getLeftSibling(n)

n′ := getLastChild(n)

insertNodeAbove(m)

d := readNode(n)

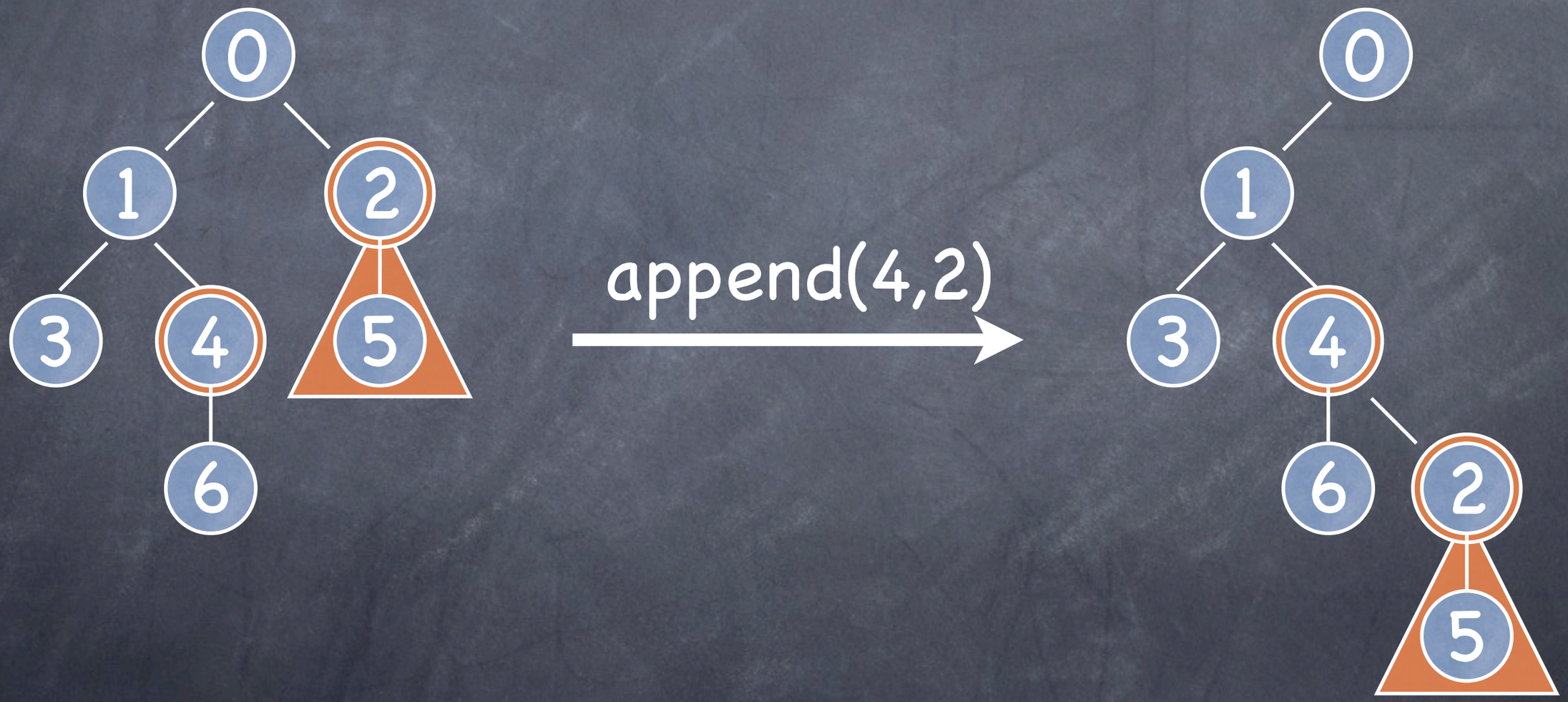
deleteNode(n)

Multi-point Update

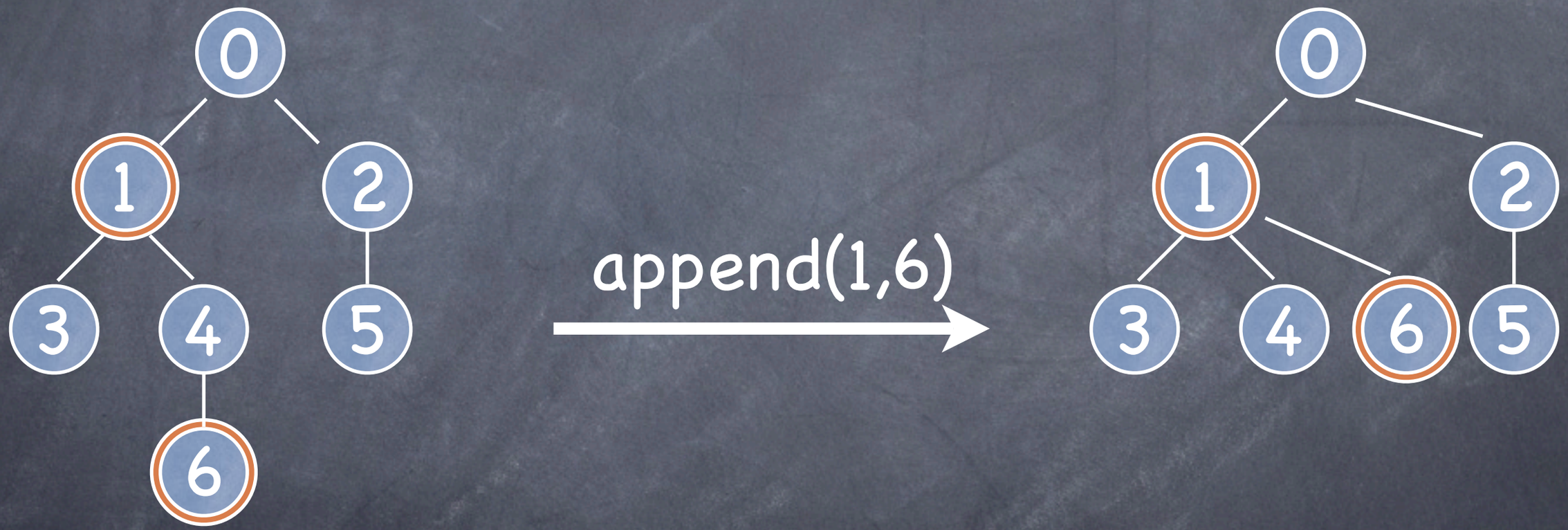
append(n,m)

"append subtree at m to children of n"

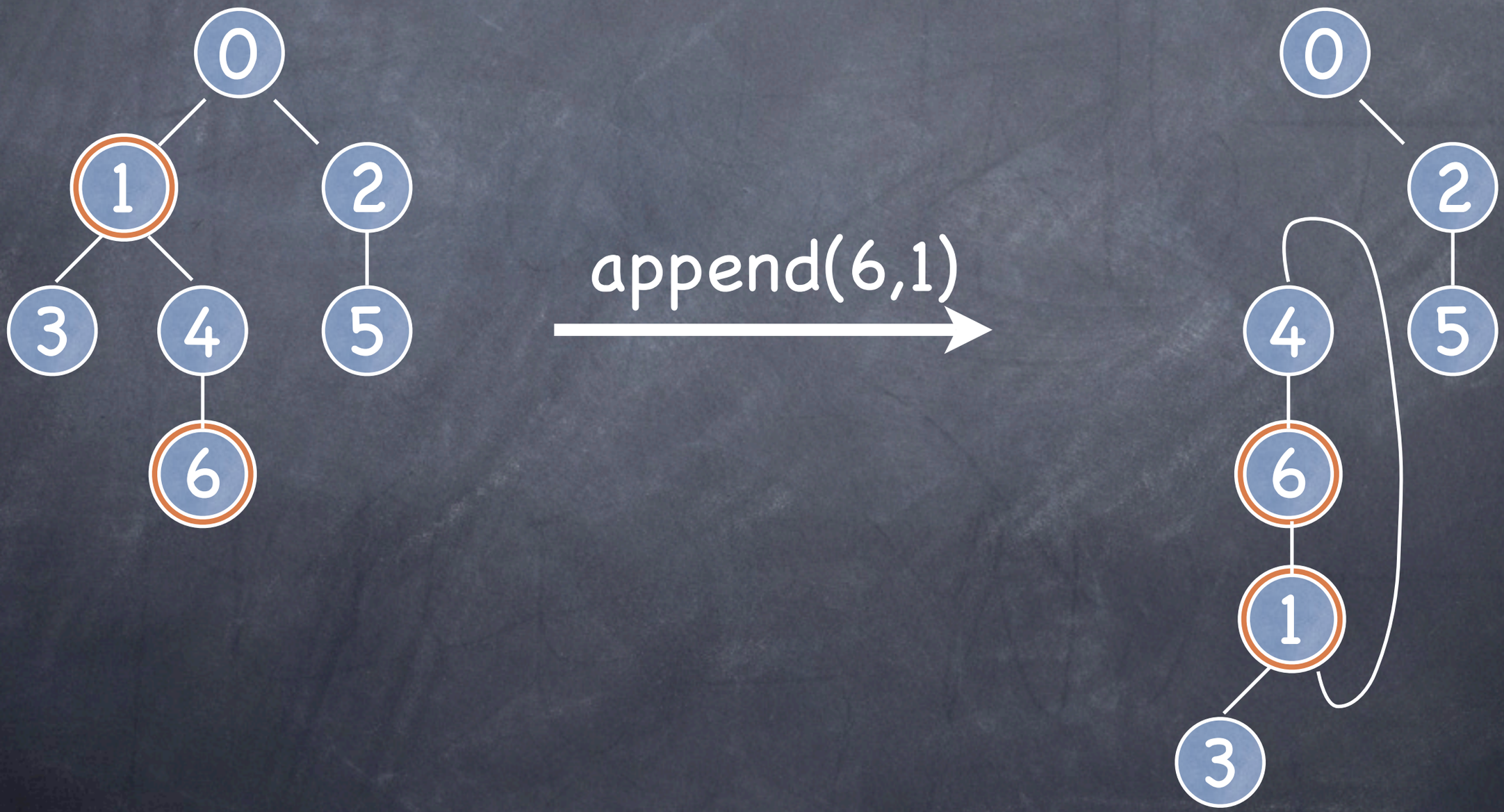
Multi-point Update



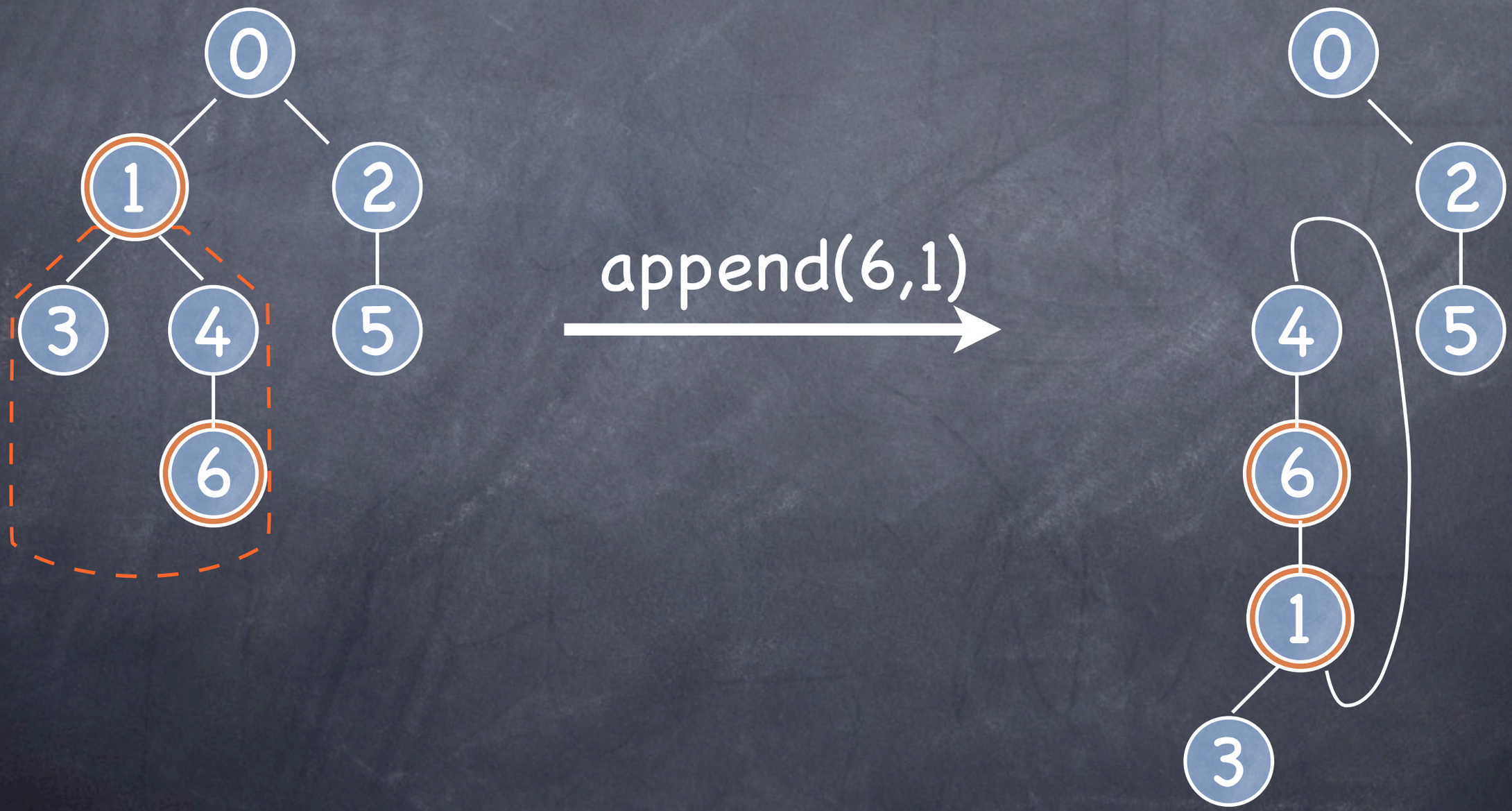
Multi-point Update



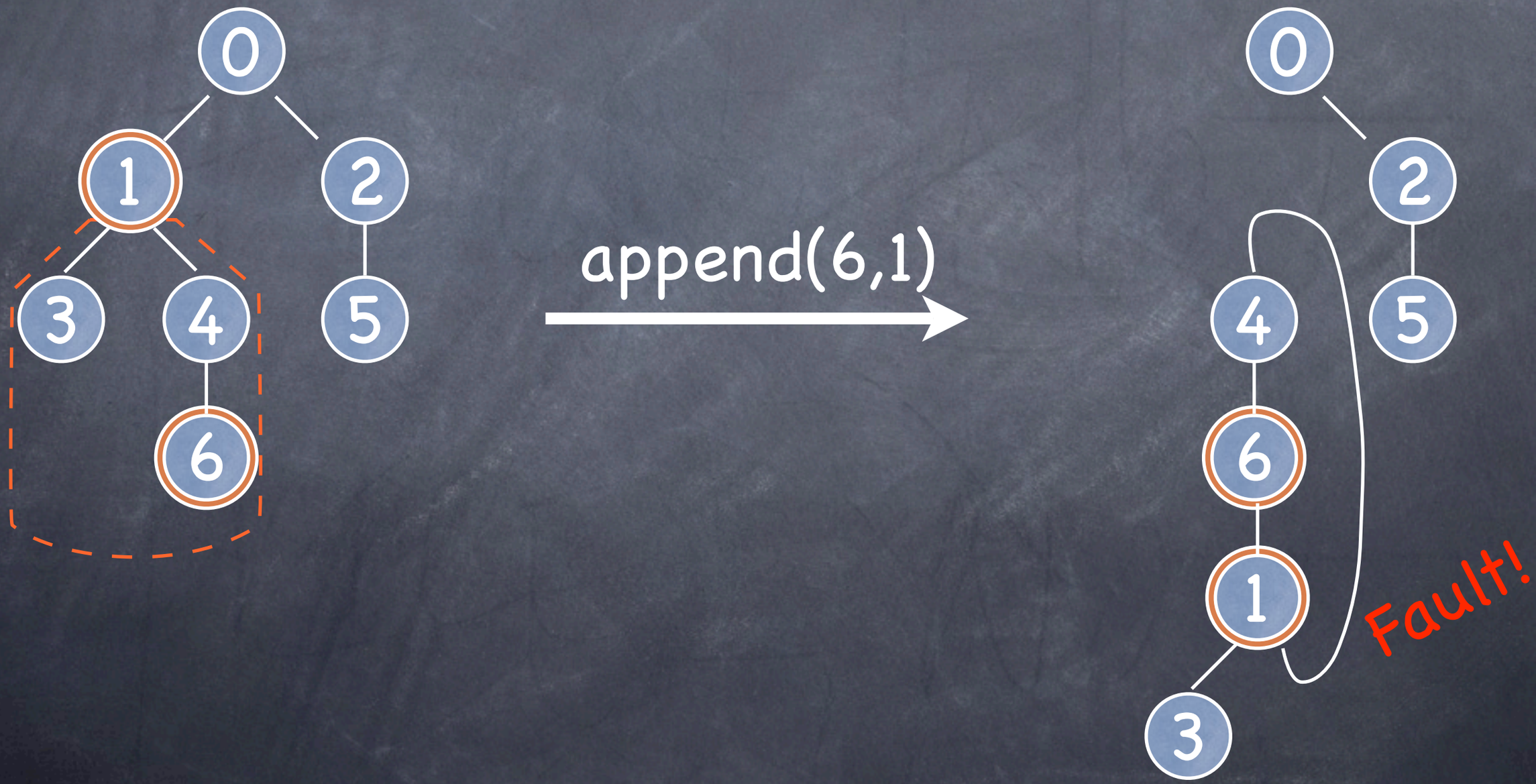
Multi-point Update



Multi-point Update

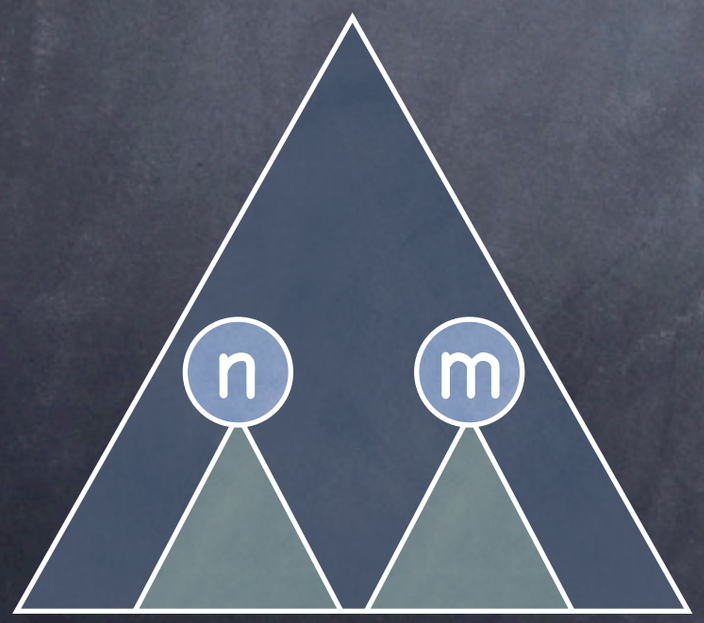
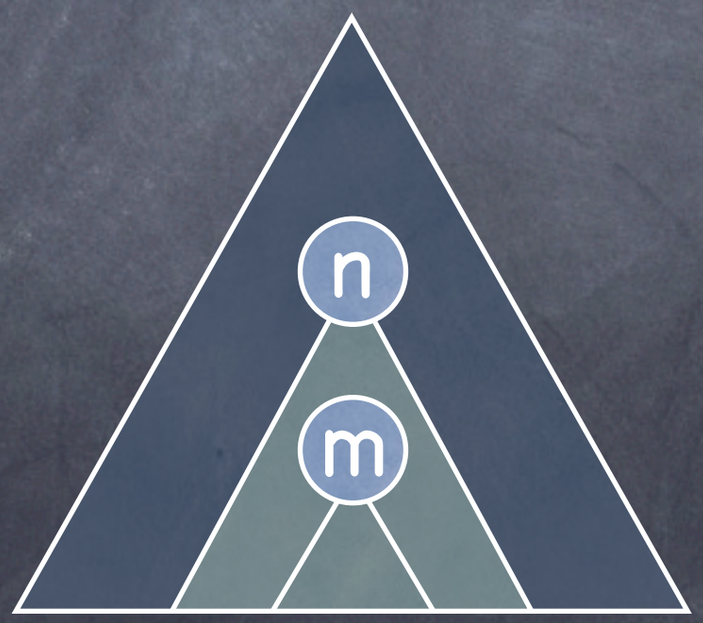
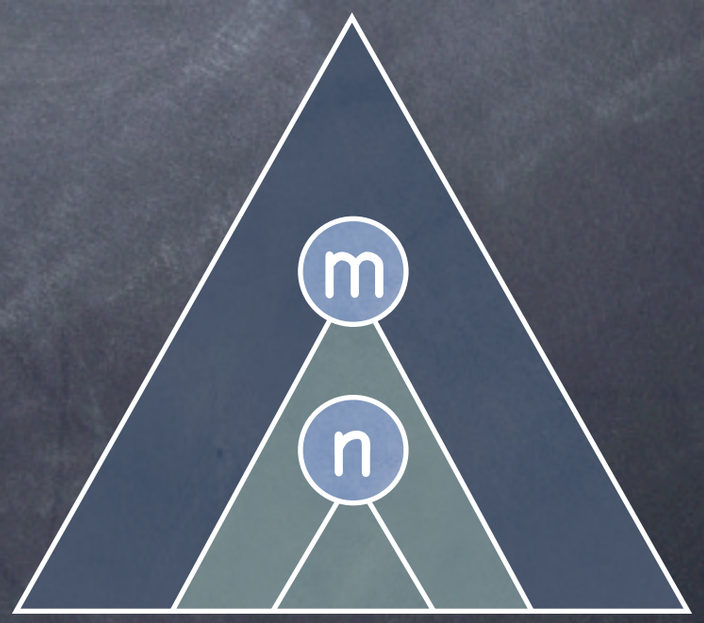


Multi-point Update



3 Cases of append

append(n,m)

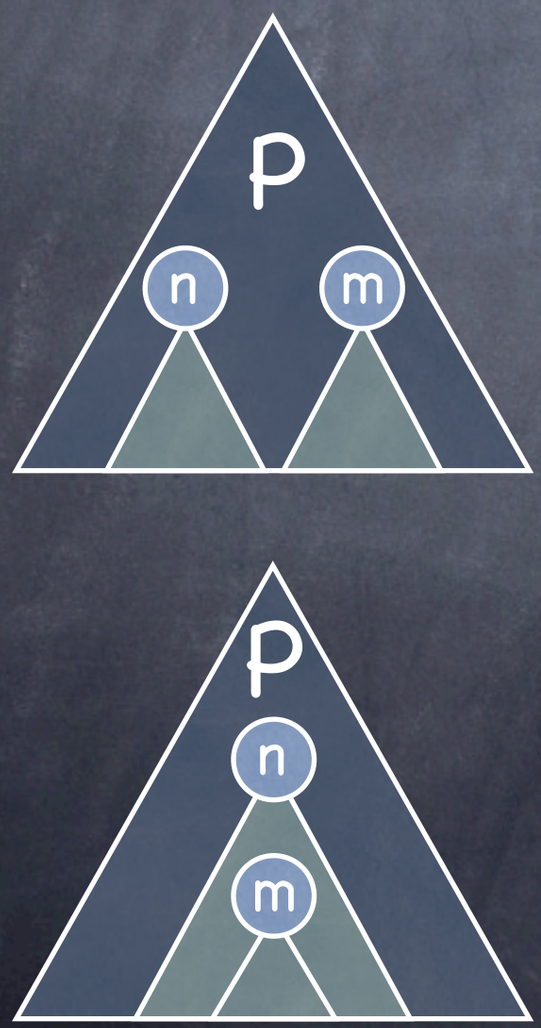
subtrees disjoint	n ancestor of m	m ancestor of n
		
✓	✓	✗

Local Reasoning - append

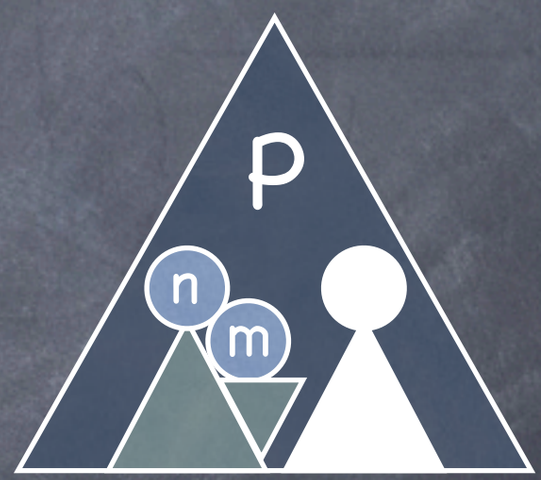
$$\{ (P \circ_x n[c]) \circ_y m[t] \}\ \mathrm{append}(n,m)\ \{ (P \circ_x n[c \otimes m[t]]) \circ_y \emptyset \}$$

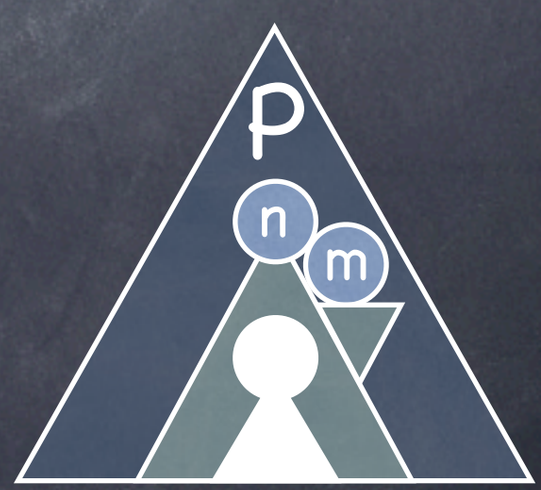
Local Reasoning - append

$\{ (P \circ_x n[c]) \circ_y m[t] \}$ append(n,m) $\{ (P \circ_x n[c \otimes m[t]]) \circ_y \emptyset \}$



append(n,m)

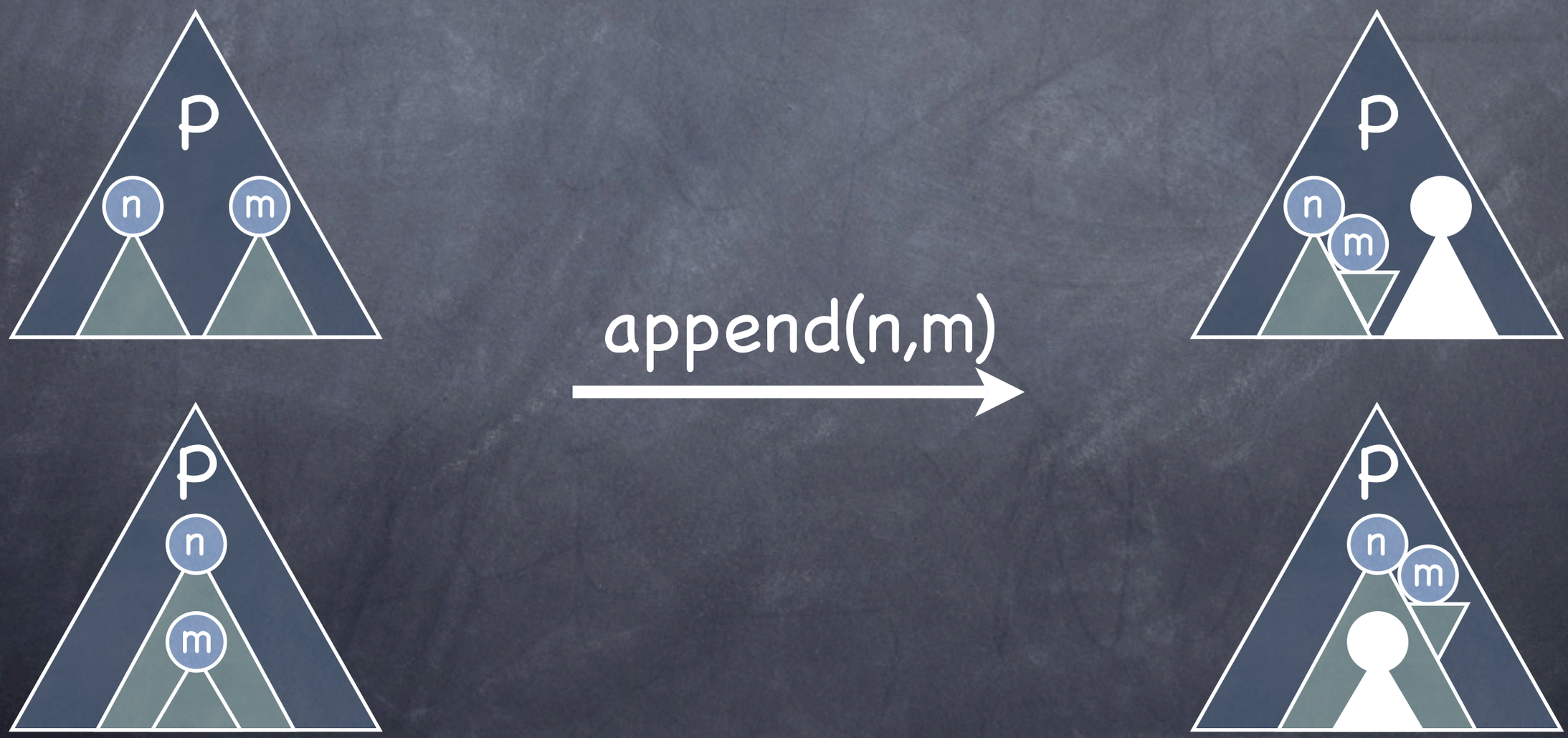




Local Reasoning - append

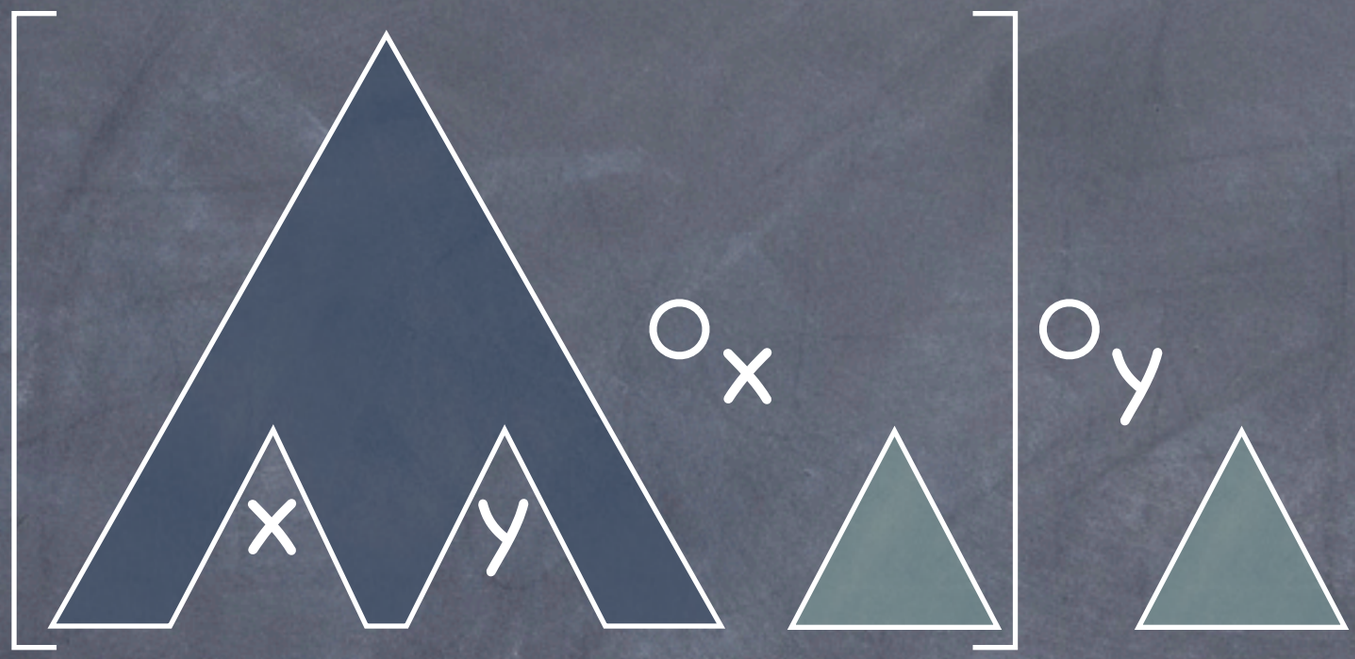
Axiom

$\{ (P \circ_x n[c]) \circ_y m[t] \}$ append(n,m) $\{ (P \circ_x n[c \otimes m[t]]) \circ_y \emptyset \}$

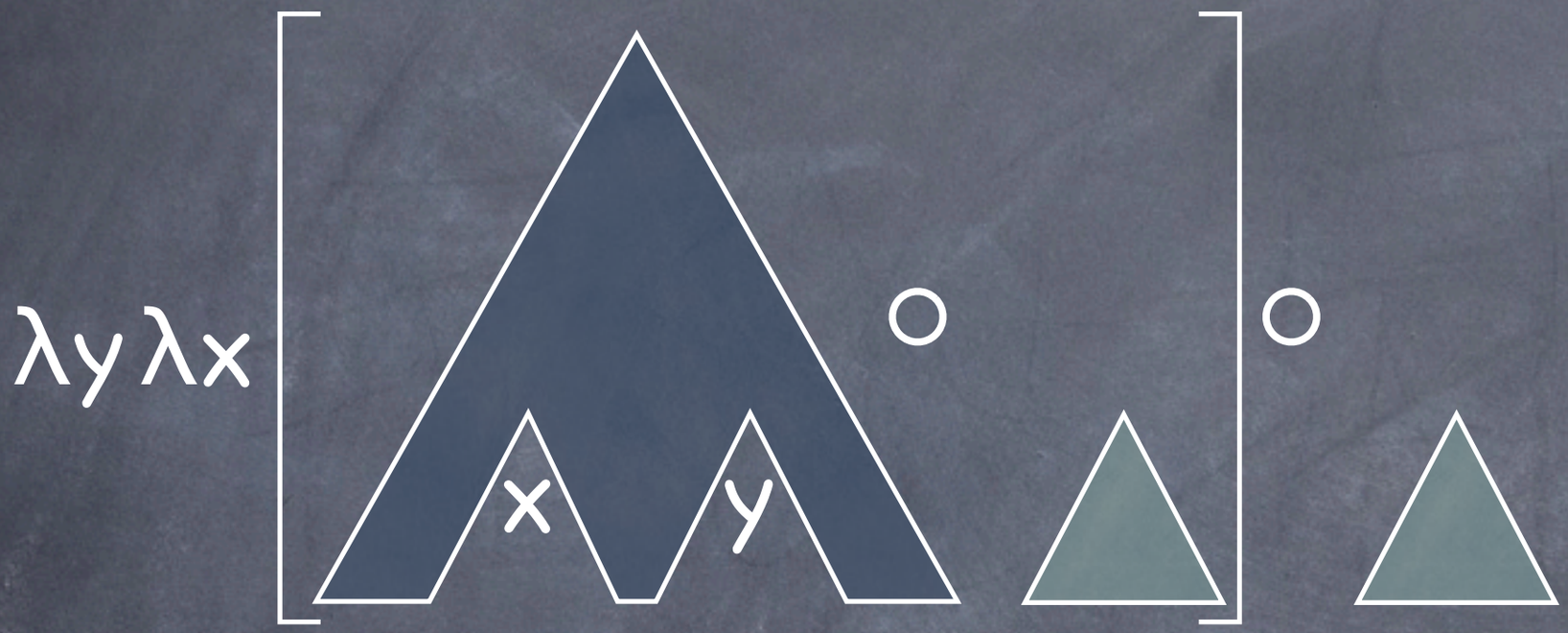


Segment Idea

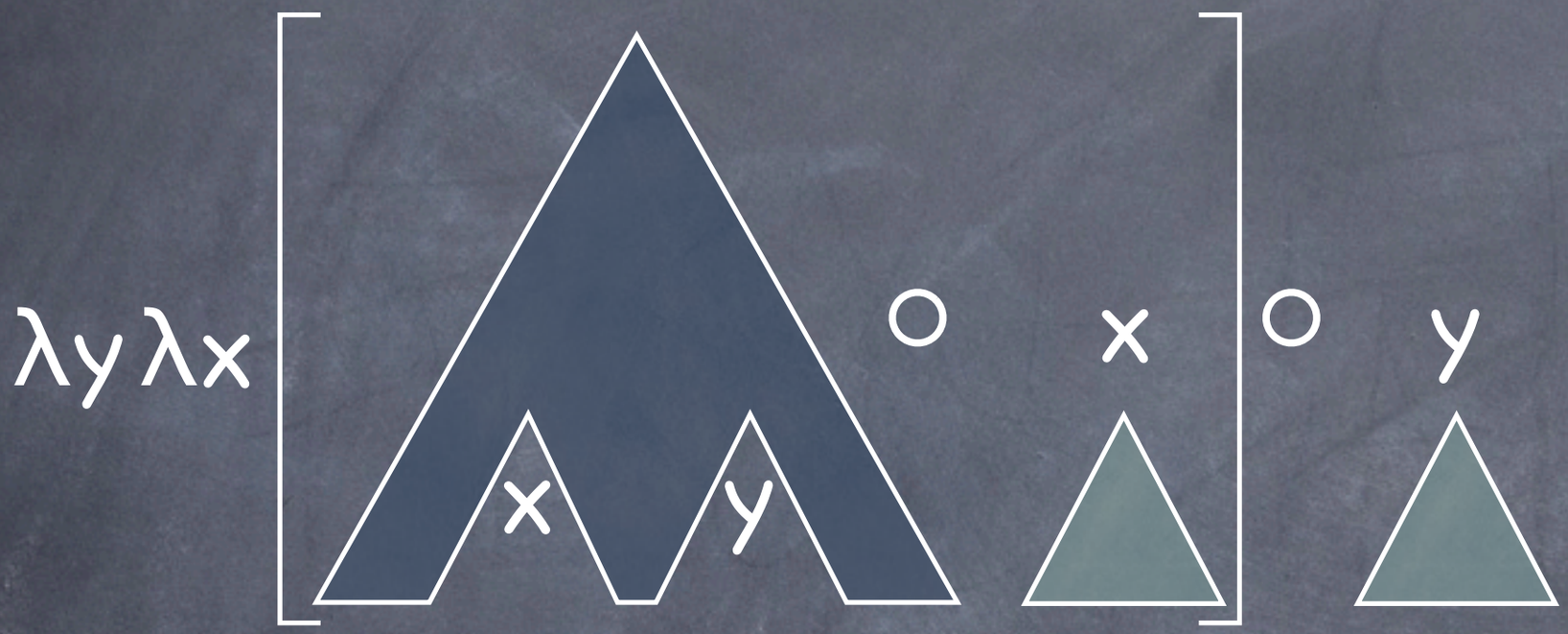
Segment Idea



Segment Idea

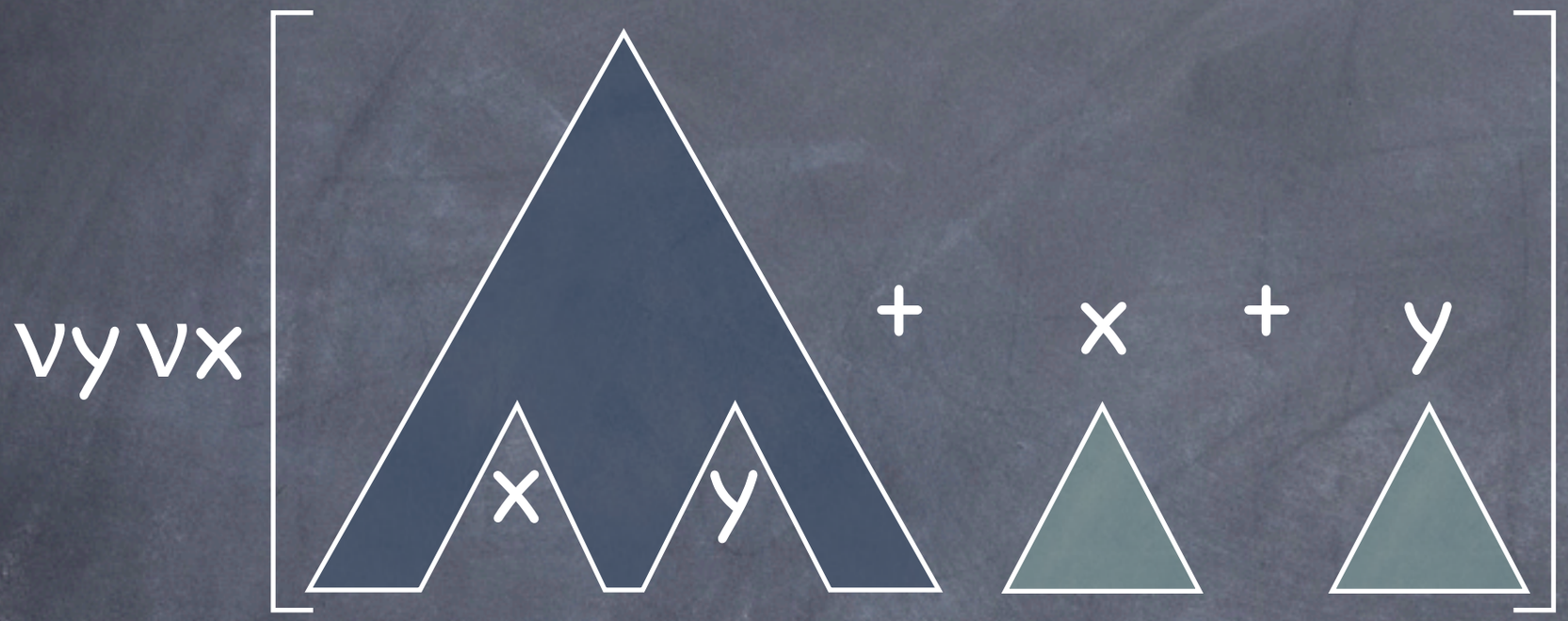


Segment Idea

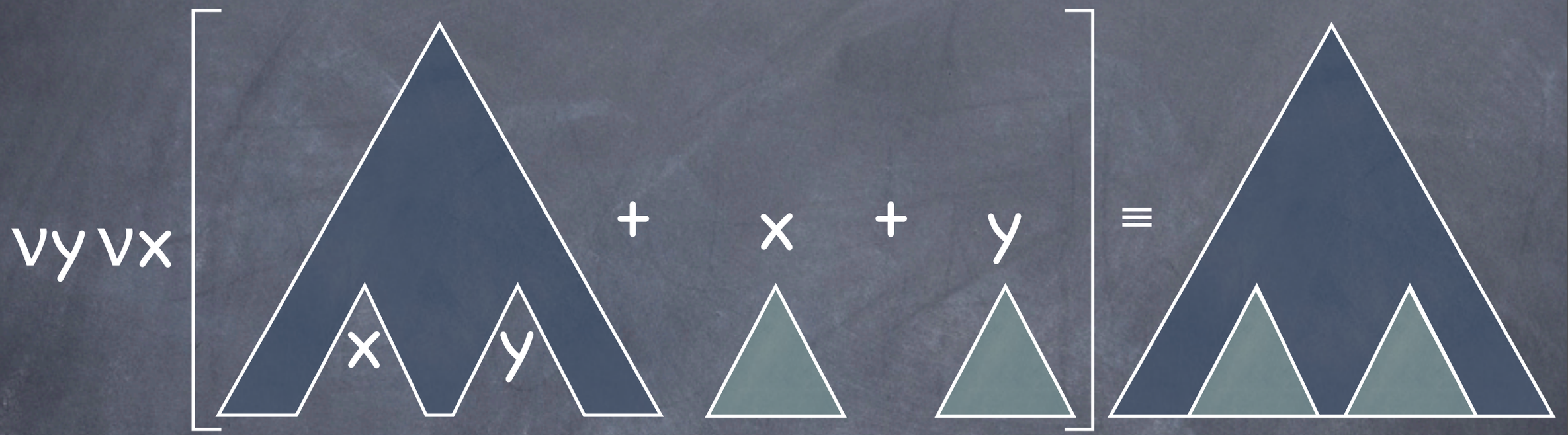


Segment Idea

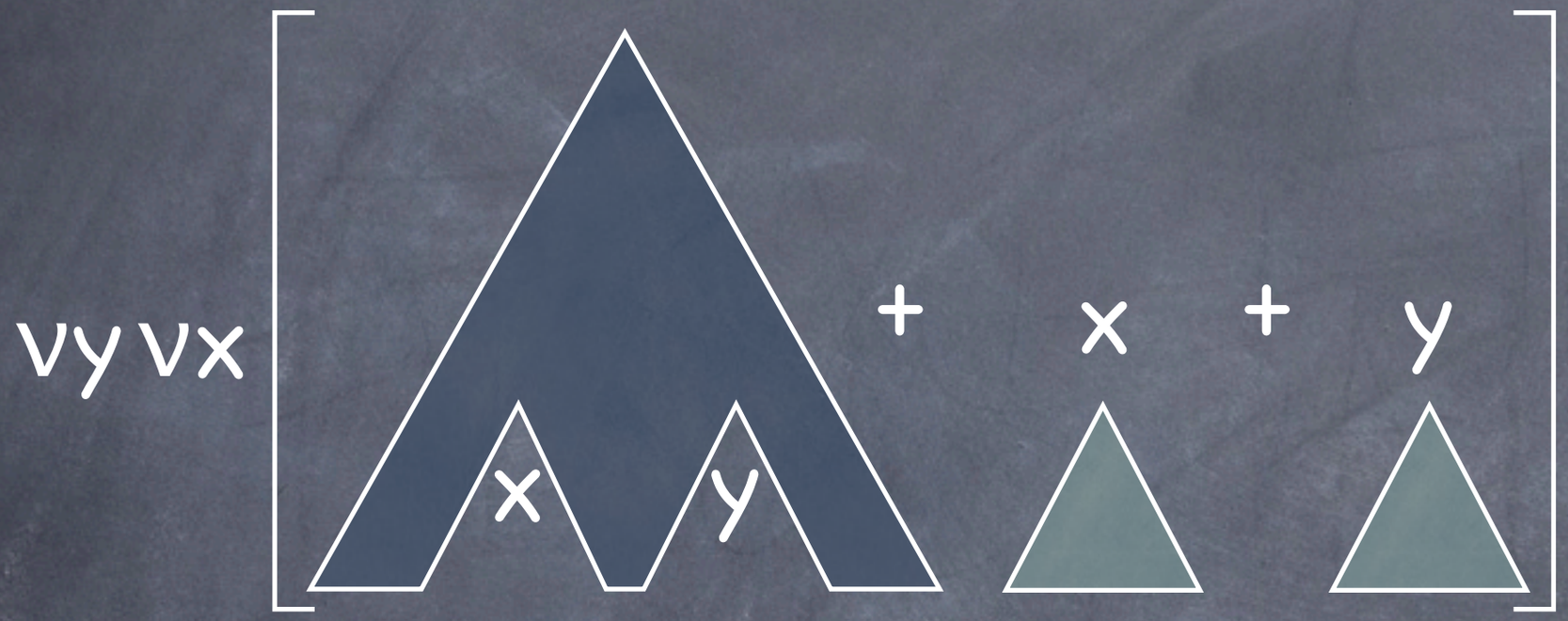
Segment Idea



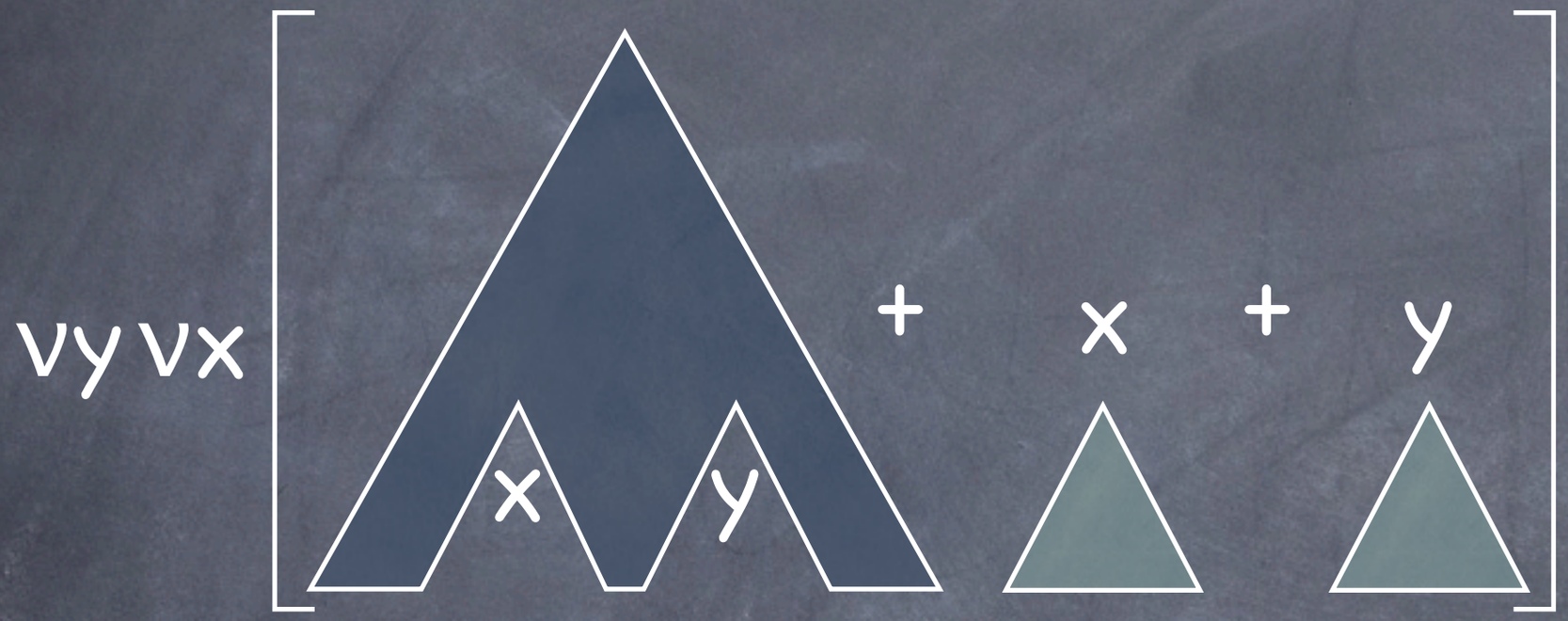
Segment Idea



Segment Idea

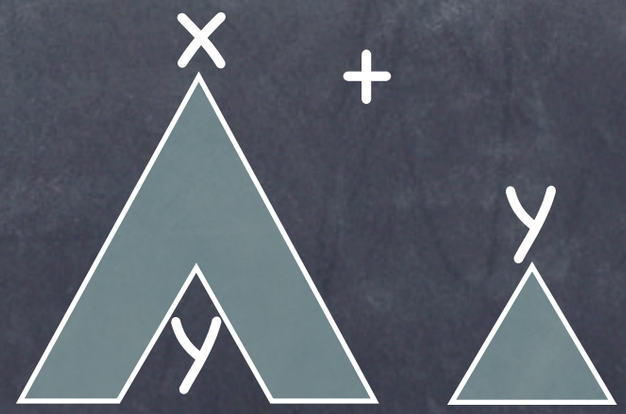


Segment Idea



Segment Idea

x + y



Tree Segments

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

tree segment $s ::= \emptyset_s \mid x{\leftarrow}c \mid s + s \mid (\nu x)(s)$

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!

Tree Segments

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

tree segment $s ::= \emptyset_s \mid x{\leftarrow}c \mid s + s \mid (\nu x)(s)$

Unique node identifiers n

Unique free hole addresses $x{\leftarrow}$

Unique free hole labels x

+ associative & commutative with unit $\emptyset_s$

$x{\leftarrow}y + y{\leftarrow}x$

$\otimes$ associative with unit $\emptyset$

& no cycles!

Tree Segments

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

tree segment $s ::= \emptyset_s \mid x{\leftarrow}c \mid s + s \mid (\nu x)(s)$

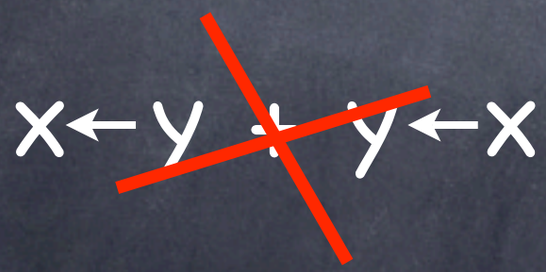
Unique node identifiers n

Unique free hole addresses $x{\leftarrow}$

Unique free hole labels x

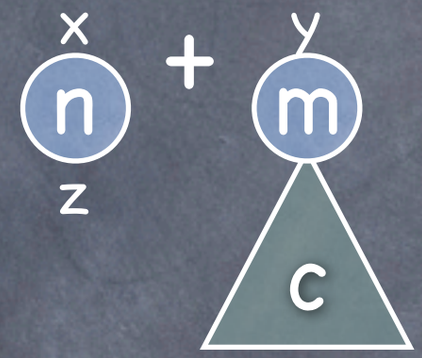
+ associative & commutative with unit $\emptyset_s$

$\otimes$ associative with unit $\emptyset$

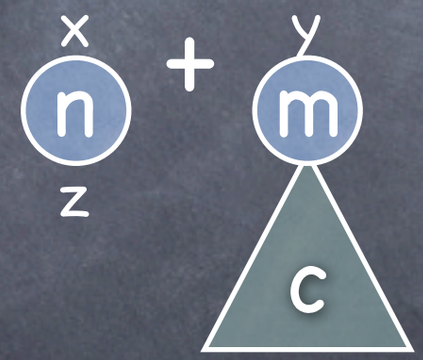


& no cycles!

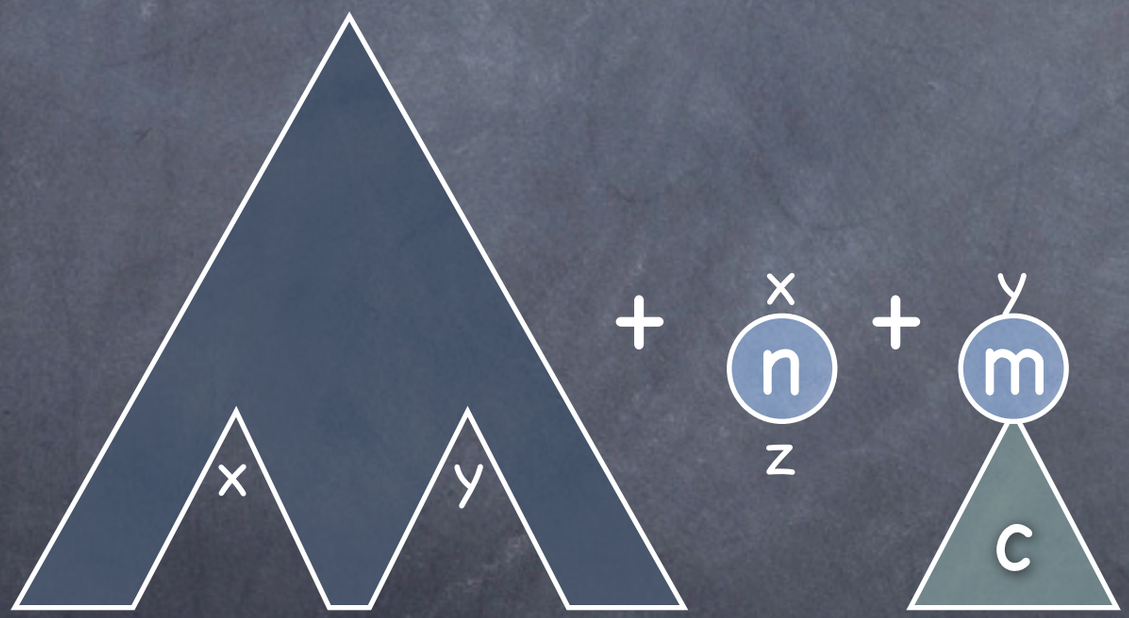
append footprint



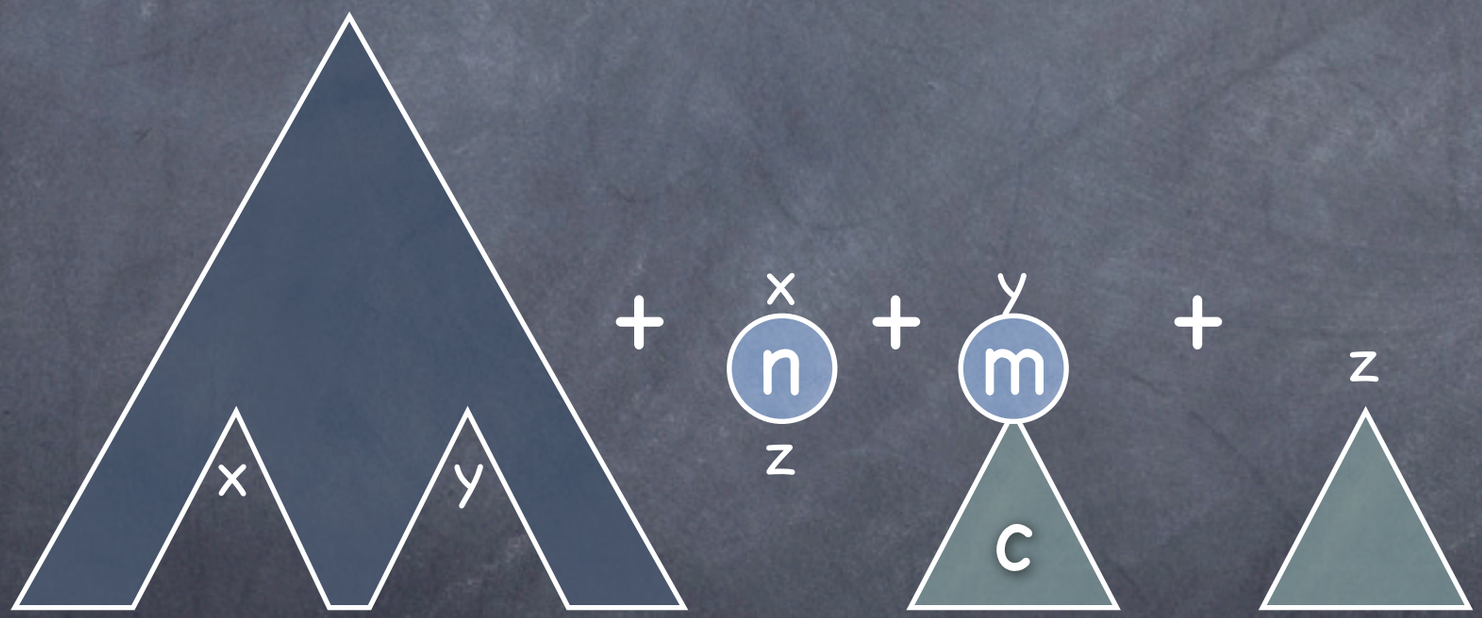
Case 1 - disjoint trees



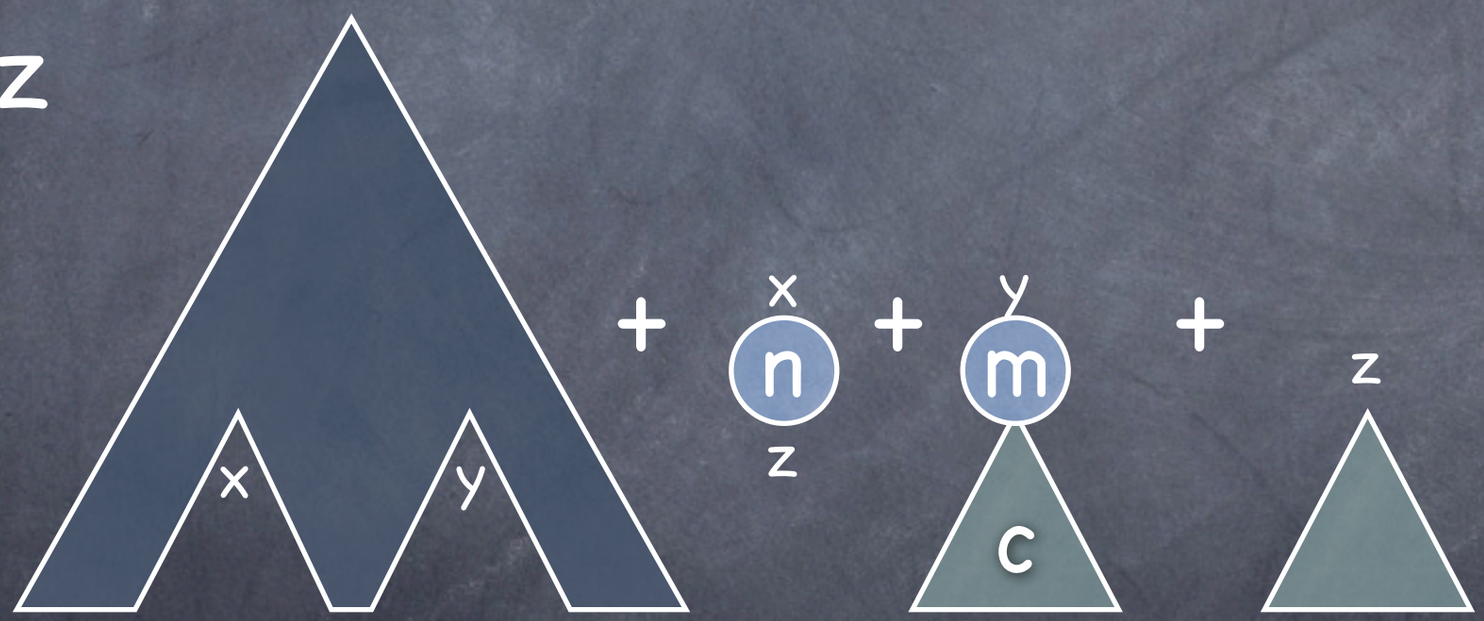
Case 1 - disjoint trees



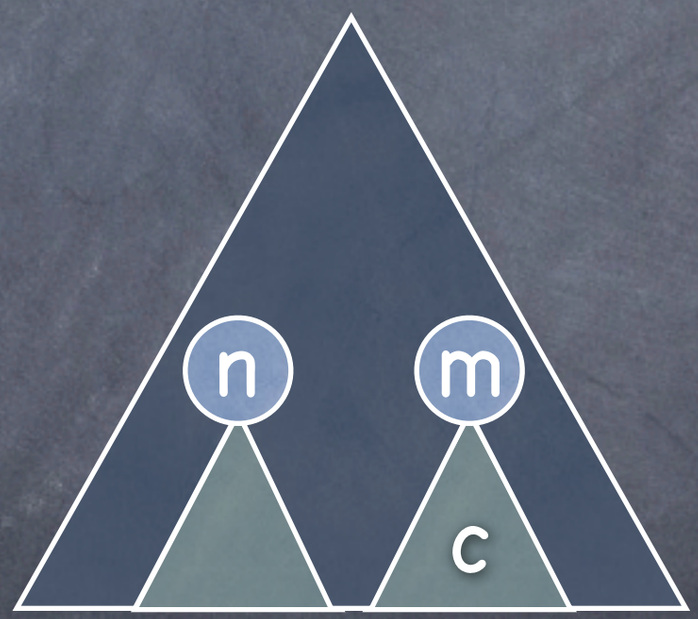
Case 1 - disjoint trees



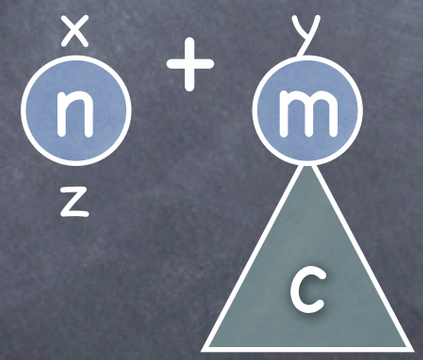
Case 1 - disjoint trees



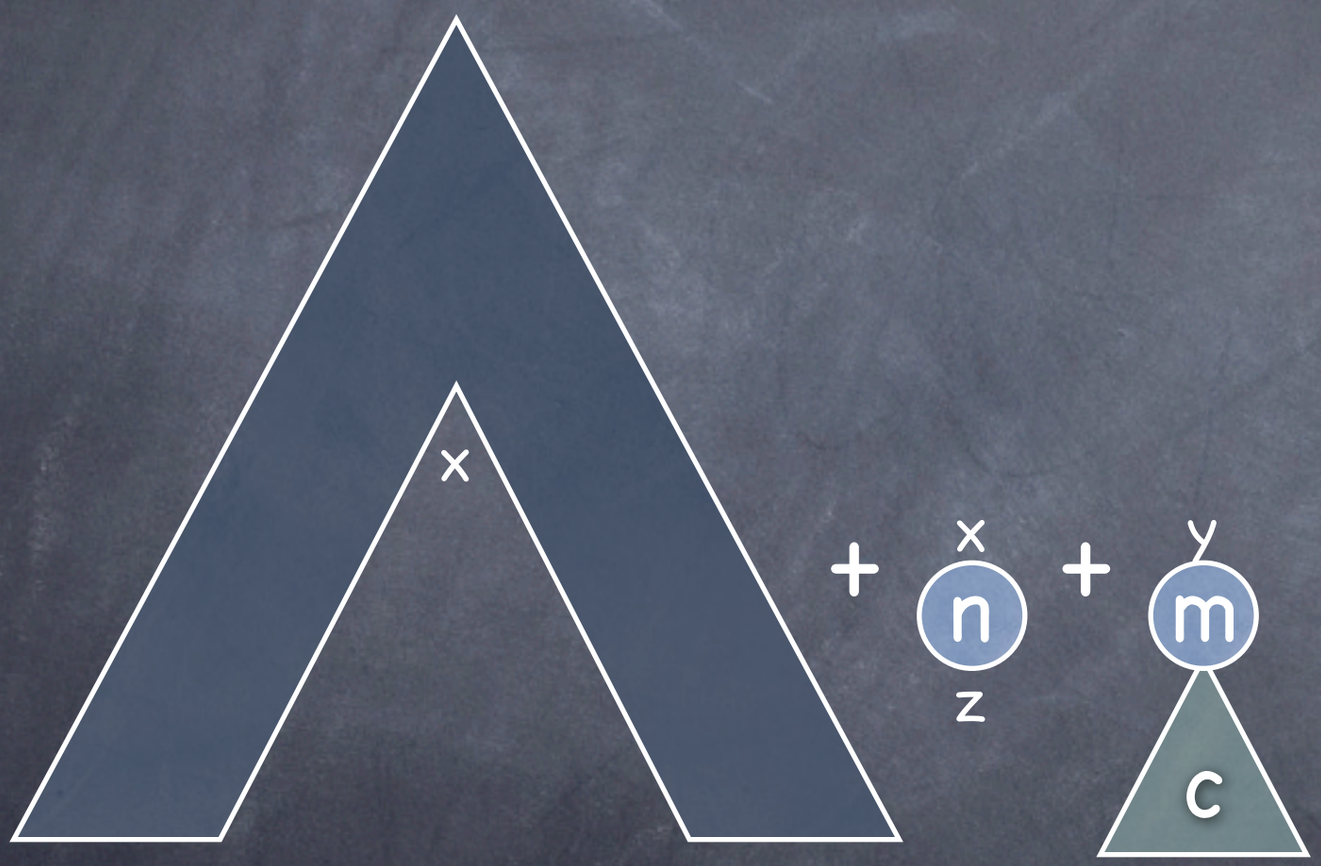
Case 1 - disjoint trees



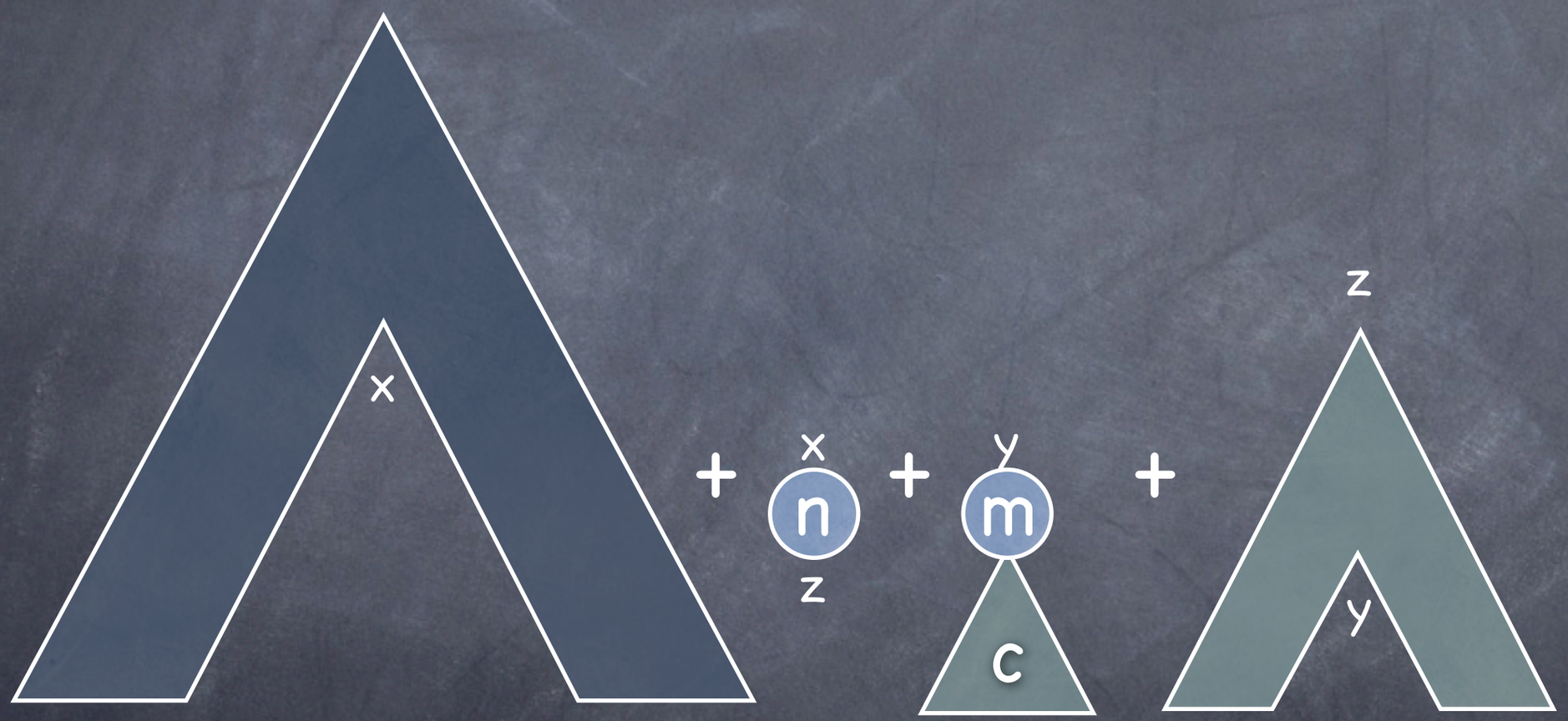
Case 2 - m under n



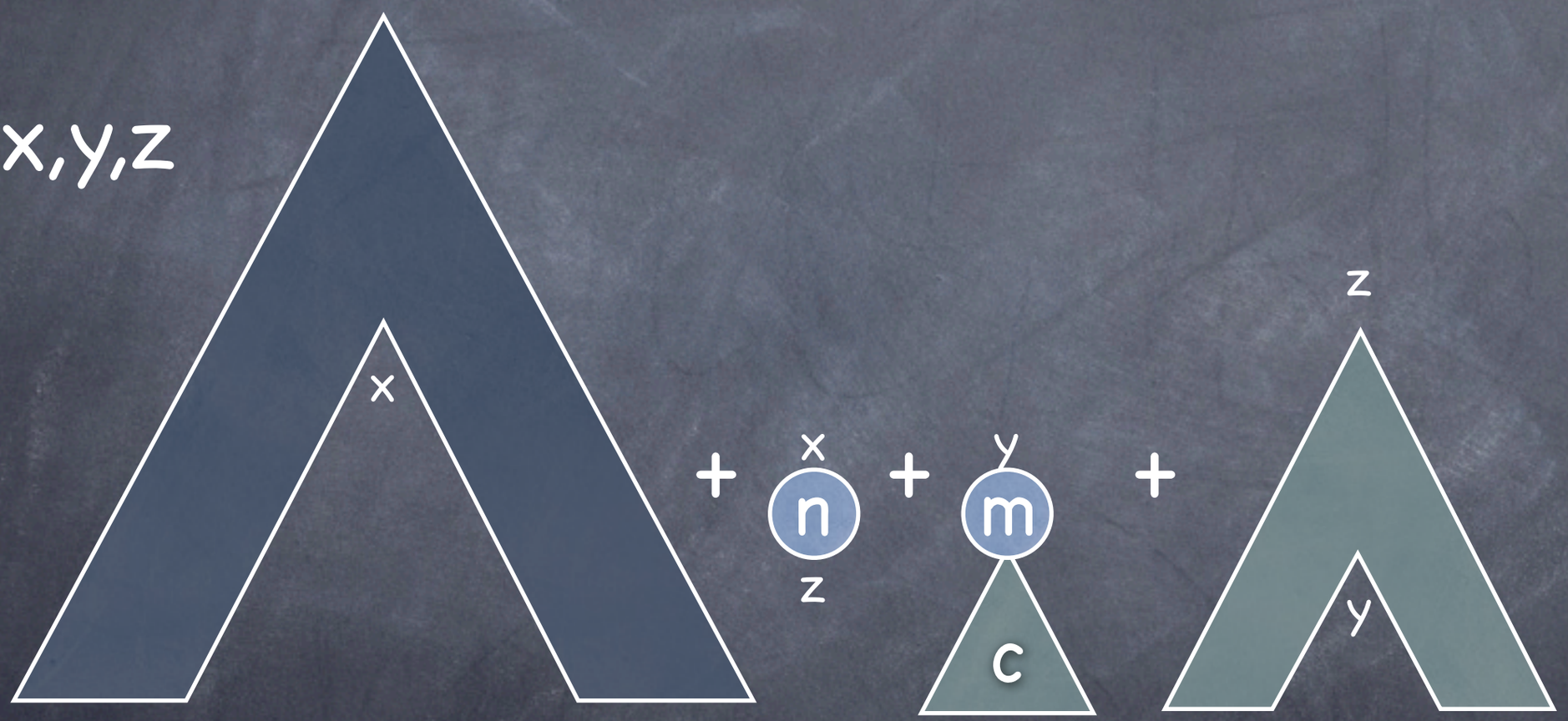
Case 2 - m under n



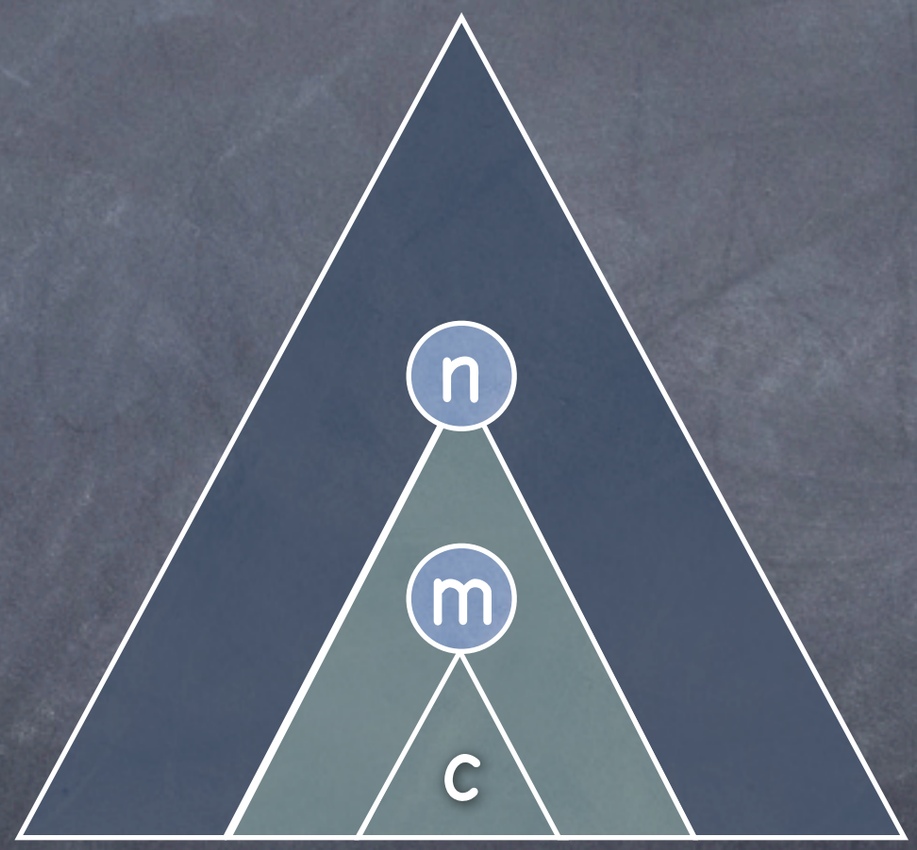
Case 2 - m under n



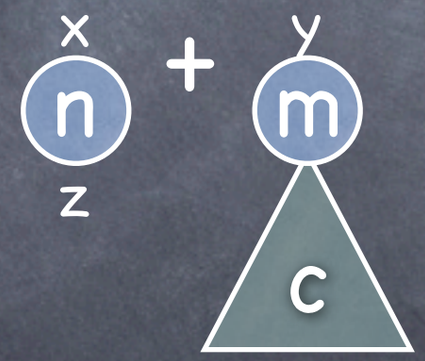
Case 2 - m under n



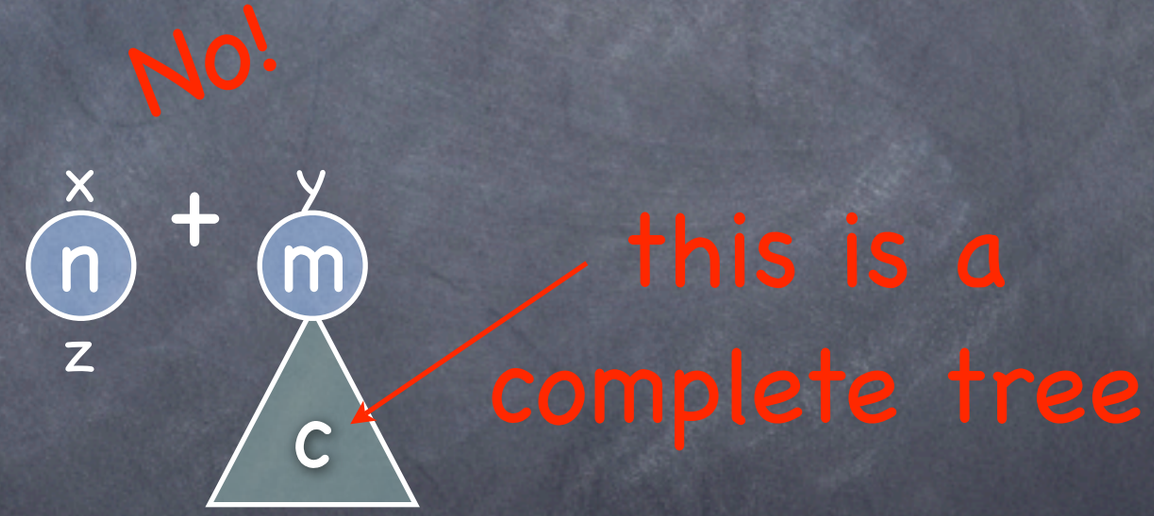
Case 2 - m under n



Case 3 - n under m?



Case 3 - n under m?



Even More Local Reasoning

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

Small Axiom

Frame Rules

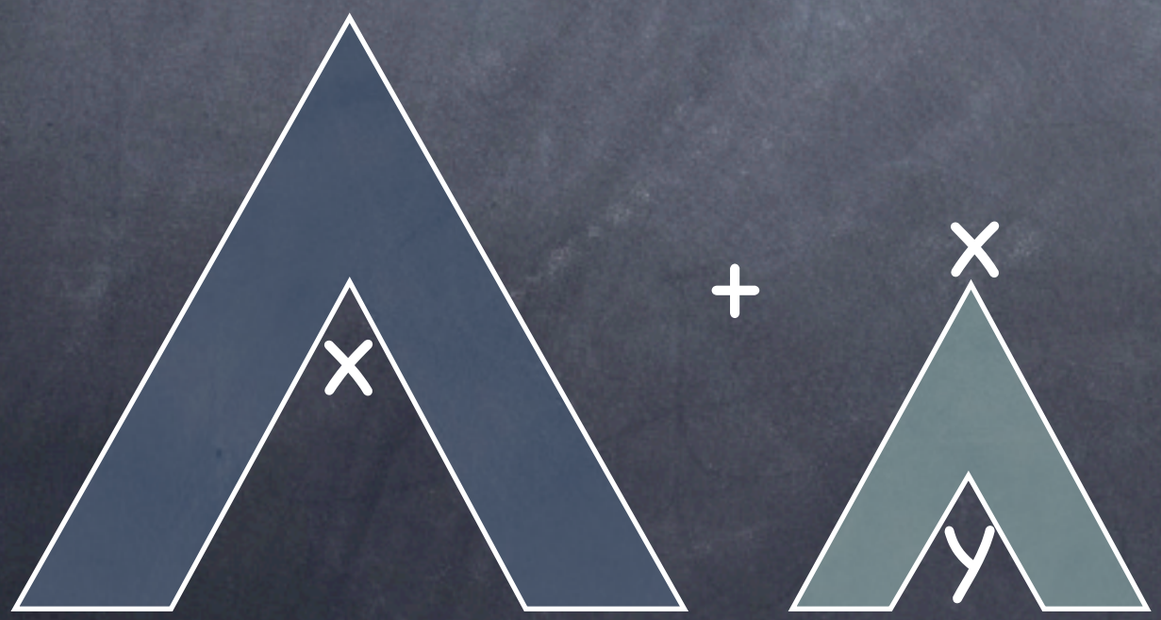
$$\frac{\{ P \}\ C\ \{ Q \}}{\{ R * P \}\ C\ \{ R * Q \}} \qquad \frac{\{ P \}\ C\ \{ Q \}}{\{ H\alpha.P \}\ C\ \{ H\alpha.Q \}}$$

Frame Rules

$$\frac{\{ P \}\ C\ \{ Q \}}{\{ R * P \}\ C\ \{ R * Q \}} \qquad \frac{\{ P \}\ C\ \{ Q \}}{\{ H\alpha.P \}\ C\ \{ H\alpha.Q \}}$$

Frame Rules

$$\frac{\{ P \}\ C\ \{ Q \}}{\{ R * P \}\ C\ \{ R * Q \}} \qquad \frac{\{ P \}\ C\ \{ Q \}}{\{ H\alpha.P \}\ C\ \{ H\alpha.Q \}}$$



Frame Rules

$$\frac{\{ P \}\ C\ \{ Q \}}{\{ R * P \}\ C\ \{ R * Q \}} \qquad \frac{\{ P \}\ C\ \{ Q \}}{\{ H\alpha.P \}\ C\ \{ H\alpha.Q \}}$$

Frame Rules

$$\frac{\{ P \}\ C\ \{ Q \}}{\{ R * P \}\ C\ \{ R * Q \}} \qquad \frac{\{ P \}\ C\ \{ Q \}}{\{ H\alpha.P \}\ C\ \{ H\alpha.Q \}}$$

Frame Rules

$$\frac{\{ P \}\ C\ \{ Q \}}{\{ R * P \}\ C\ \{ R * Q \}} \qquad \frac{\{ P \}\ C\ \{ Q \}}{\{ H\alpha.P \}\ C\ \{ H\alpha.Q \}}$$

Frame Rules

$$\frac{\{ P \}\ C\ \{ Q \}}{\{ R * P \}\ C\ \{ R * Q \}} \qquad \frac{\{ P \}\ C\ \{ Q \}}{\{ H\alpha.P \}\ C\ \{ H\alpha.Q \}}$$

Important Formulae

$P * Q$ separating conjunction

$H\alpha.P$ label hiding

Important Formulae

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

Weakest Preconditions

$$\{\ \exists c.H\alpha.(\ (\alpha \leftarrow \emptyset \mathrel{-\!\!*} (\alpha \mathrel{-\!\!\circledR} P)) * \alpha \leftarrow n[tree(c)]\ )\ \}$$

delete(n)

$$\{\ P\ \}$$

Weakest Preconditions

$$\{\ \exists c.H\alpha.(\ (\alpha \leftarrow \emptyset \mathrel{-\!\!*} (\alpha \mathrel{-\!\!\circledast} P)) * \alpha \leftarrow n[tree(c)]\ )\ \}$$

$$\mathrel{\uparrow}\alpha.\alpha\circledast$$

$$\mathrm{delete}(n)$$

$$\{\ P\ \}$$

Weakest Preconditions

$$\{\ \exists c.H\alpha.(\ (\alpha \leftarrow \emptyset \mathrel{-\!\!*} (\alpha \mathrel{-\!\!\circledast} P)) * \alpha \leftarrow n[tree(c)]\ )\ \}$$

$$\underline{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}$$

$$\{\ P\ \}$$

delete(n)

$И\alpha.\alpha\circledast$

Weakest Preconditions

$$\{\ \exists c.H\alpha.(\ (\alpha \leftarrow \emptyset \mathrel{-\!\!*} (\alpha \mathrel{-\!\!\circledast} P)) * \alpha \leftarrow n[tree(c)]\ )\ \}$$

$$\underline{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}$$

$$delete(n)$$

$$\{\ P\ \}$$

$$И\alpha.\alpha\circledast$$

Weakest Preconditions

$$\{\ \exists c.H\alpha.(\ (\alpha \leftarrow \emptyset \mathrel{-\!\!*} (\alpha \mathrel{-\!\!\circledR} P)) * \alpha \leftarrow n[tree(c)]\ )\ \}$$

$$\underline{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}$$

$$delete(n)$$

$$\{\ P\ \}$$

$И\alpha.\alpha\circledR$

$\alpha = x$

Weakest Preconditions

$$\{\ \exists c.H\alpha.(\ (\alpha\leftarrow\emptyset \mathrel{-\!\!*} (\alpha \mathrel{-\!\!\circledR} P)) * \alpha\leftarrow n[tree(c)]\ )\ \}$$

$$\underline{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}$$

$$delete(n)$$

$$\{\ P\ \}$$

$И\alpha.\alpha\circledR$

$\alpha=x$

νx

Weakest Preconditions

$$\{\ \exists c.H\alpha.(\ (\alpha \leftarrow \emptyset \mathrel{-\!\!*} (\alpha \mathrel{-\!\!\circledast} P)) * \alpha \leftarrow n[tree(c)]\ )\ \}$$

$$\underline{\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad}$$

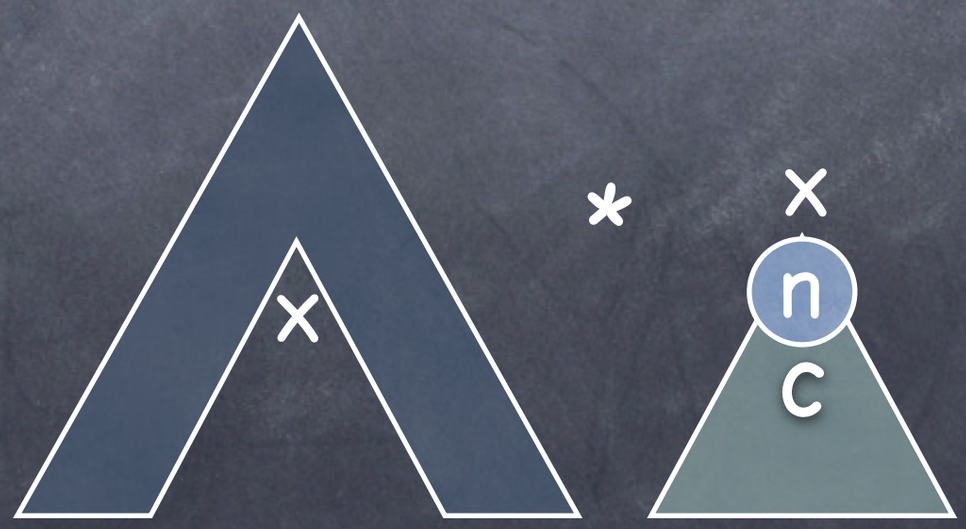
$$delete(n)$$

$$\{\ P\ \}$$

$H\alpha.\alpha\circledast$

$\alpha = x$

νx



Weakest Preconditions

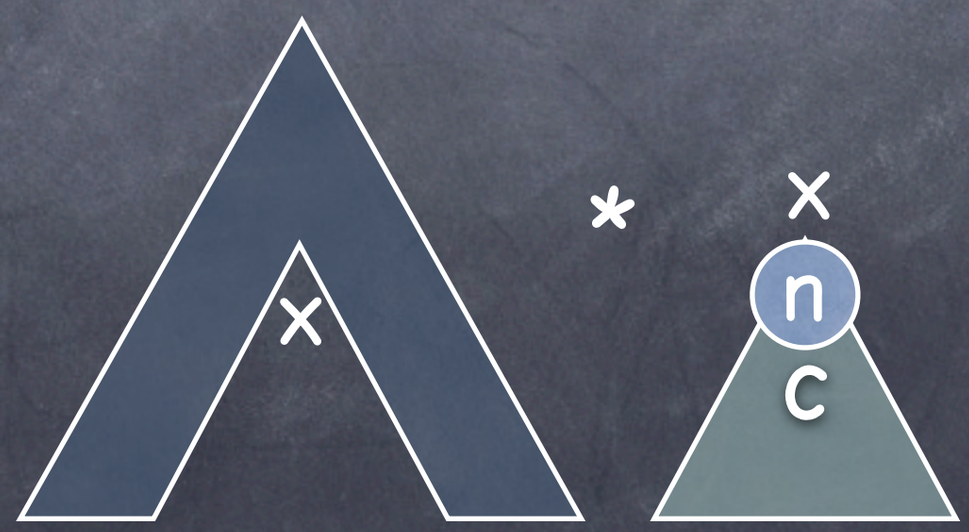
$$\{\ \exists c.H\alpha.(\ \underline{(\alpha \leftarrow \emptyset \mathrel{-\!\!*} (\alpha \mathrel{-\!\!\circledR} P)) * \alpha \leftarrow n[tree(c)]}\ )\ \}$$

$$\frac{}{delete(n)}$$

$$\{\ P\ \}$$

$H\alpha.\alpha\circledR$

$\alpha = x$



Weakest Preconditions

$$\{\ \exists c.H\alpha.(\ (\alpha \leftarrow \emptyset \mathrel{-\!\!*} (\alpha \mathrel{-\!\!\circledR} P)) * \alpha \leftarrow n[tree(c)]\ )\ \}$$

$$\{\ P\ \}$$

delete(n)

$H\alpha.\alpha\circledR$

$\alpha = x$

Weakest Preconditions

$$\{\ \exists c.H\alpha.(\ (\alpha \leftarrow \emptyset \mathrel{-\!\!*} (\alpha \mathrel{-\!\!\circledR} P)) * \alpha \leftarrow n[tree(c)]\ )\ \}$$

$$\underline{\alpha \mathrel{-\!\!\circledR} P}$$

$$\underline{\quad delete(n) \quad}$$

$$\{\ P\ \}$$

$$И\alpha.\alpha\circledR$$

$\alpha = x$

$$* \quad x \mapsto \emptyset$$

Weakest Preconditions

$$\{\ \exists c.H\alpha.(\ (\alpha \leftarrow \emptyset \mathrel{-\!\!*} (\alpha \mathrel{-\!\!\circledR} P)) * \alpha \leftarrow n[tree(c)]\ )\ \}$$

$$\underline{\alpha \mathrel{-\!\!\circledR} P}$$

$$\underline{\quad delete(n) \quad}$$

$$\{\ P\ \}$$

$$И\alpha.\alpha\circledR$$

$$\alpha = x$$

$$\nu x$$

$$*$$

$$x$$

$$\emptyset$$

Weakest Preconditions

$$\{\ \exists c.H\alpha.(\ (\alpha \leftarrow \emptyset \twoheadrightarrow (\alpha \multimap \circledR P)) * \alpha \leftarrow n[tree(c)]\ )\ \}$$

$$\text{delete}(n)$$

$$\{\ P\ \}$$

$И\alpha.\alpha\circledR$

$\alpha = x$

Weakest Preconditions

{ ∃c.Hα.((α←∅ -* (α —®P)) * α←n[tree(c)]) }

delete(n)

{ P }

Иα.α®

α=x

Summary

- Segment Logic enables reasoning about disjoint tree update
- Segment Logic allows us to write Small Axioms for DOM commands
- Segment Logic can be applied to more than just trees - e.g. lists, sets, ...

Future Work

Future Work

- We believe Segment Logic is important for reasoning about concurrent tree update

Future Work

- We believe Segment Logic is important for reasoning about concurrent tree update

delete(n) || delete(m)

Future Work

- We believe Segment Logic is important for reasoning about concurrent tree update

$$\{\ \alpha{\leftarrow}n[tree(c_1)] * \beta{\leftarrow}m[tree(c_2)]\ \}$$

delete(n) || delete(m)

Future Work

- We believe Segment Logic is important for reasoning about concurrent tree update

$$\{\ \alpha{\leftarrow}n[tree(c_1)] * \beta{\leftarrow}m[tree(c_2)]\ \}$$

$$\{\ \alpha{\leftarrow}n[tree(c_1)]\ \} \quad \{\ \beta{\leftarrow}m[tree(c_2)]\ \}$$

$$\text{delete}(n) \quad || \quad \text{delete}(m)$$

Future Work

- We believe Segment Logic is important for reasoning about concurrent tree update

$$\{\ \alpha{\leftarrow}n[tree(c_1)] * \beta{\leftarrow}m[tree(c_2)]\ \}$$

$$\{\ \alpha{\leftarrow}n[tree(c_1)]\ \} \qquad \{\ \beta{\leftarrow}m[tree(c_2)]\ \}$$

$$\text{delete}(n) \quad \| \quad \text{delete}(m)$$

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

Future Work

- We believe Segment Logic is important for reasoning about concurrent tree update

$$\{\ \alpha{\leftarrow}n[tree(c_1)] * \beta{\leftarrow}m[tree(c_2)]\ \}$$

$$\{\ \alpha{\leftarrow}n[tree(c_1)]\ \} \qquad \{\ \beta{\leftarrow}m[tree(c_2)]\ \}$$

$$\text{delete}(n) \quad \| \quad \text{delete}(m)$$

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

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

Future Work

- We believe Segment Logic is important for reasoning about concurrent tree update

$$\{\ \alpha{\leftarrow}n[tree(c_1)] * \beta{\leftarrow}m[tree(c_2)]\ \}$$

$$\{\ \alpha{\leftarrow}n[tree(c_1)]\ \} \qquad \{\ \beta{\leftarrow}m[tree(c_2)]\ \}$$

$$\text{delete(n)} \quad \| \quad \text{delete(m)}$$

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

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

- Next Steps: reasoning about more interesting concurrency - e.g. locks, shared data, ...

Thanks for Listening

Any Questions ?

The Full Paper Can Be Found Online At:

http://www.doc.ic.ac.uk/~mjw03/PersonalWebpage/pdfs/moveFull.pdf