Verifying the Rope Data Structure of Xi-Editor
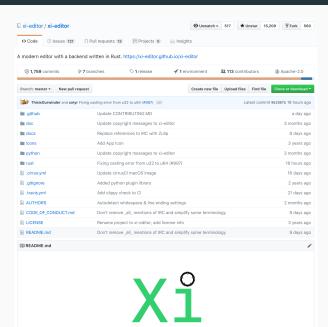
Anna Scholtz

27. November 2018

Outline

Motivation

Goal and Motivation

- **Motivation:** Need for good text editors
  - Xi-Editor: combines newer concepts for storing text to allow very high performance
  - Core idea: modified rope data structure for storing text as well as edit history
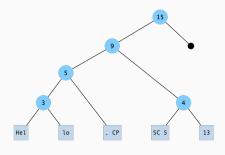- **Goal:** Verification of the rope data structure in xi-editor

Background

Xi-Editor

4

Rope Data Structures
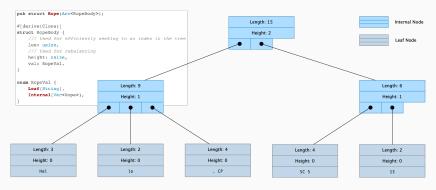
- Binary Tree
- Only leaves contain text
- Operations: Index, Split, Concatenate, Report, Insert, Delete

Hello, CPSC 513

- B-Tree
- Operations: Concatenate, Slice, Insert, SliceToString

Hello, CPSC 513

Verifying Ropes

Verifying Ropes

- **Goal:**
  - Define properties of ropes
  - Verify that after applying operations, these properties still hold
  - Verify that this also works for distributed ropes (collaborative editing)

- Approach: first verify standard rope data structure (simpler), then verify rope data structure of xi-editor

Dafny

- Language and verifier
- Supports verification through pre-conditions, post-conditions, loop invariants, ...
- Dafny programs are translated into Boogie 2 which is used to generate first-order verification conditions that are passed to the SMT solver Z3

```dafny
method Add(x: int, y: int) returns (r: int)
  requires 0 <= x && 0 <= y
  ensures r == 2*x + y
{
  r := x;
  var n := y;
  while n != 0
    invariant r == x+y-n && 0 <= n
  {
    r := r + 1;
    n := n - 1;
  }
}
```

Properties Standard Rope

1. Every node has at most two children
2. Only leaves contain data
3. Weight values of non-leaf nodes is the length of all children in the left subtree
4. Weight values of leaf nodes is the length of the stored text

Standard Rope Verification

Rope Data Structure:

```
datatype Node = Leaf(value: string) | InternalNode(left: Rope?, right: Rope?)

class Rope {
  ghost var Repr: set<object>    // contains all elements that are in the tree

  var len: int    // length
  var val: Node   // node value
```

Validity Conditions:

```
predicate Valid()
  reads this, Repr
{
  this in Repr &&
  (
    match this.val
    case Leaf(v) => true
    case InternalNode(left, right) =>
      (left != null ==>
        left in this.Repr && this.Repr >= left.Repr && this !in left.Repr && left.Valid()
      ) &&
      (right != null ==>
        right in this.Repr && this.Repr >= right.Repr && this !in right.Repr && right.Valid()
      )
  )
}
```

Validity Conditions:

```
predicate ValidLen()
  requires Valid()
  reads this, Repr
{
  match this.val
  case Leaf(v) => this.len == |v|
  case InternalNode(left, right) =>
    (left != null ==> this.len == left.Len() && left.ValidLen()) &&
    (left == null ==> this.len == 0) &&
    (right != null ==> right.ValidLen())
}
```

Implemented Methods:

```
method Index(i: int) returns (charAtIndex: string)
  requires Valid()
  requires ValidLen()
  ensures i >= 0 && this.Len() > i ==> charAtIndex != ""
  decreases Repr
{

method Insert(i: int, s: string) returns (newRope: Rope?)
  requires Valid()
  requires ValidLen()
  ensures Valid()
  ensures ValidLen()
  ensures newRope != null ==> newRope.Valid()
  ensures newRope != null ==> newRope.ValidLen()
  ensures i < 0 || i >= this.Len() <==> newRope == null
{

method Concat(rope: Rope) returns (concatenatedRope: Rope)
  requires Valid()
  requires ValidLen()
  requires rope.Valid()
  requires rope.ValidLen()
  ensures concatenatedRope.Valid()
  ensures concatenatedRope.ValidLen()
{
```

Properties Xi-Editor Rope

1. Every node has at most *MAX_CHILDREN* children
2. Every non-leaf node, except the root node, has at least *MIN_CHILDREN* child nodes
3. The root has at least two children if it is not a leaf node
4. Only leaves contain data
5. The length of the text stored in leaf nodes is at most *MAX_LEAF*
6. Weight values of non-leaf nodes is the sum of the children's weights
7. Weight values of leaf nodes is the length of the stored text
8. All leaves appear in the same level

Xi-Editor Rope Verification

Rope Data Structure:

```
datatype Node = Leaf(value: string) | InternalNode(children: seq<Rope>)

class Rope {
  ghost var Repr: set<Rope>
  ghost var Content: seq<string>

  var val: Node
  var len: int
  var height: int

  constructor Init()
    ensures Valid()
    ensures ValidLen()
  {
    val := Leaf("");
    len := 0;
    height := 0;
    Repr := {this};
    Content := [""];
  }
}
```

Validity Conditions:

```
predicate Valid()
  reads this, Repr
  requires MAX_LEAF_LEN >= MIN_LEAF_LEN
  requires MIN_CHILDREN <= MAX_CHILDREN && MIN_CHILDREN >= 2
{
  this in Repr &&
  (
    match this.val
    case Leaf(v) => |v| <= MAX_LEAF_LEN && Content == [v] && height == 0
    case InternalNode(children) =>
      height >= 0 &&
      |children| >= 2 &&
      |children| <= MAX_CHILDREN &&
      forall c: Rope :: c in children ==>
        c in Repr && this !in c.Repr && c.Repr < Repr &&
        c.ValidNonRoot() && c.height == height - 1 && |c.Content| <= |Content| &&
        forall cont: string :: cont in c.Content ==> cont in this.Content
  )
}
```

Xi-Editor Rope Verification

Validity Conditions:

```
predicate ValidNonRoot()
  reads this, Repr
  requires MAX_LEAF_LEN >= MIN_LEAF_LEN
  requires MIN_CHILDREN <= MAX_CHILDREN && MIN_CHILDREN >= 2
{
  this in Repr &&
  (
    match this.val
    case Leaf(v) => |v| <= MAX_LEAF_LEN && Content == [v] && height == 0
    case InternalNode(children) =>
      height >= 0 &&
      |children| >= MIN_CHILDREN &&
      |children| <= MAX_CHILDREN &&
      forall c: Rope :: c in children ==>
        c in Repr && this !in c.Repr && c.Repr < Repr && c.ValidNonRoot() &&
        c.height == height - 1 && |c.Content| <= |Content| &&
        forall cont: string :: cont in c.Content ==> cont in this.Content
  )
}
```

Validity Conditions:

```
predicate ValidLen()
  requires Valid()
  reads this, Repr
{
  match this.val
  case Leaf(v) =>
    this.len == |v| && ContentLen(this.Content) == |v| && |Content| == 1
  case InternalNode(children) =>
    this.len >= 0 && forall c: Rope :: c in children ==>
      c.len <= this.len && c.ValidLen()
}
```

Implemented Methods (work in progress):

```
method Index(i: int) returns (charAtIndex: string)
  requires Valid()
  requires ValidLen()
  ensures i < 0 || i >= this.len ==> charAtIndex == ""
  decreases Repr
{
```

```
method SliceToString(i: int, j: int) returns (slice: string)
  requires Valid()
  requires ValidLen()
  ensures i < 0 || i >= this.len || j >= this.len || i > j || j < 0 ==> slice == ""
  decreases Repr
{
```

Future Work

Future Work

- Implement remaining methods for xi-editor rope
- Implement transactions and verify that properties hold in a collaborative environment (out-of-scope)
- Verify other parts of xi-editor (very much out-of-scope)