Composite pattern - partial correctness proof

1 Proof

```
class Composite {
  private Composite left, right, parent;
  private int count;

  public Composite
```

$\multimap$ this@$\frac{1}{2}$ parent() $\otimes$ this@$\frac{1}{2}$ left(`null`, 0) $\otimes$ this@$\frac{1}{2}$ right(`null`, 0)

```
  {
    this.count = 1;
```

{ this.count $\rightarrow$ 1 }

```
    this.left = null;
```

{ this.left $\rightarrow$ `null` $\otimes$ this.count $\rightarrow$ 1 }

```
    this.right = null;
```

{ this.right $\rightarrow$ `null` $\otimes$ this.left $\rightarrow$ `null` $\otimes$ this.count $\rightarrow$ 1 }

```
    this.parent = null;
```

{ this.parent $\rightarrow$ `null` $\otimes$ this.right $\rightarrow$ `null` $\otimes$ this.left $\rightarrow$ `null` $\otimes$ this.count $\rightarrow$ 1 }

pack this@1 right(null, 0)

{ this.parent $\rightarrow$ `null` $\otimes$ this.left $\rightarrow$ `null` $\otimes$ this.count $\rightarrow$ 1 $\otimes$ this@1 right(`null`, 0) }

pack this@1 left(null, 0)

{ this.parent $\rightarrow$ `null` $\otimes$ this.count $\rightarrow$ 1 $\otimes$ this@1 right(`null`, 0) $\otimes$ this@1 left(`null`, 0) }

split 1 into two halfs for left, right

pack this@1 count(0)

{ this.parent $\rightarrow$ `null` $\otimes$ this@$\frac{1}{2}$ right(`null`, 0) $\otimes$ this@$\frac{1}{2}$ left(`null`, 0) $\otimes$ this@1 count(0) }

split 1 into two halfs for count

pack this@1 parent()

{ this@$\frac{1}{2}$ right(`null`, 0) $\otimes$ this@$\frac{1}{2}$ left(`null`, 0) $\otimes$ this@1 parent() }

split 1 into halfs

{ this@$\frac{1}{2}$ right(`null`, 0) $\otimes$ this@$\frac{1}{2}$ left(`null`, 0) $\otimes$ this@$\frac{1}{2}$ parent() $\otimes$ this@$\frac{1}{2}$ parent() }

we only need one half of parent in the post-condition

{ this@$\frac{1}{2}$ right(`null`, 0) $\otimes$ this@$\frac{1}{2}$ left(`null`, 0) $\otimes$ this@$\frac{1}{2}$ parent() }

{ QED }
}

```
private void updateCountRec ()
```

$\exists$ k1, opp, lcc, k, ol, lc, or, rc.

(unpacked(this@ k1 parent()) $\otimes$
this.parent $\rightarrow$ opp $\otimes$ opp $\neq$ this $\otimes$
$\Big($(opp $\neq$ null $\multimap$ opp@k parent() $\otimes$
(opp@$\frac{1}{2}$ left(this, lcc) $\oplus$ opp@$\frac{1}{2}$ right(this, lcc))) $\oplus$
(opp = null $\multimap$ this@$\frac{1}{2}$ count(lcc)) $\Big)$ $\otimes$

unpacked(this@$\frac{1}{2}$ count(lcc)) $\otimes$
this.count $\rightarrow$ lcc $\otimes$ lcc = lc + rc + 1 $\otimes$
this@$\frac{1}{2}$ left(ol, lc) $\otimes$ this@$\frac{1}{2}$ right(or, rc)

$\multimap$ $\exists$ k1.this@$k1$ parent())

{

```
if (this.parent != null)
```

this can be the right child of opp or the left child.
For Boogie we need to consider both paths.

{ $\exists$ k1,lcc,opp, k.
unpacked(this@k1 parent()) $\otimes$
this.parent $\rightarrow$ opp $\otimes$ opp $\neq$ this $\otimes$
opp@k parent() $\otimes$
(opp@$\frac{1}{2}$ left(this, lcc) $\oplus$ opp@$\frac{1}{2}$ right(this, lcc)) $\otimes$

unpacked(this@$\frac{1}{2}$ count(lcc)) $\otimes$
$\exists$ ol, lc, or, rc. this.count $\rightarrow$ lcc $\otimes$ lcc = lc + rc + 1 $\otimes$
this@$\frac{1}{2}$ left(ol, lc) $\otimes$ this@$\frac{1}{2}$ right(or, rc) }

split the fraction k of opp in parent and unpack opp from parent()

{ $\exists$ k1,lcc,opp, k.
unpacked(this@k1 parent()) $\otimes$
this.parent $\rightarrow$ opp $\otimes$ opp $\neq$ this $\otimes$
unpacked(opp@$\frac{k}{2}$ parent()) $\otimes$ $\exists$ oppp, lccc, kk. opp.parent $\rightarrow$ oppp
$\otimes$ opp $\neq$ oppp $\otimes$ opp@$\frac{1}{2}$ count(lccc) $\otimes$
$\Big($(oppp $\neq$ `null` $\multimap$ oppp@kk parent() $\otimes$ (oppp@$\frac{1}{2}$ left(opp, lccc)
$\oplus$ oppp@$\frac{1}{2}$ right(opp, lccc))) $\oplus$
(oppp = null $\multimap$ opp@$\frac{1}{2}$ count(lccc))$\Big)$ $\otimes$

(opp@$\frac{1}{2}$ left(this, lcc) $\oplus$ opp@$\frac{1}{2}$ right(this, lcc)) $\otimes$
$\otimes$ opp@$\frac{k}{2}$ parent() $\otimes$

unpacked(this@$\frac{1}{2}$ count(lcc)) $\otimes$
$\exists$ ol, lc, or rc. this.count $\rightarrow$ lcc $\otimes$
lcc = lc + rc + 1 $\otimes$
this@$\frac{1}{2}$ left(ol, lc) $\otimes$ this@$\frac{1}{2}$ right(or, rc)

unpack opp from $\frac{1}{2}$ count(lccc)
{$\exists$ k1, opp, lcc, k.
unpacked(this@k1 parent()) $\otimes$
this.parent $\rightarrow$ opp $\otimes$ opp $\neq$ this $\otimes$
unpacked(opp@$\frac{k}{2}$ parent()) $\otimes$ $\exists$ oppp, lccc, kk. opp.parent $\rightarrow$ oppp
$\otimes$ opp $\neq$ oppp $\otimes$

unpacked(opp@$\frac{1}{2}$ count(lccc)) $\otimes$ $\exists$ oll, orr, llc, rrc. opp.count $\rightarrow$
lccc $\otimes$ lccc = llc + rrc + 1 $\otimes$ opp@$\frac{1}{2}$ left(oll, llc) $\otimes$ opp@$\frac{1}{2}$
right(orr, rrc) $\otimes$

$\Big($(oppp $\neq$ `null` $\multimap$ oppp@kk parent() $\otimes$ (oppp@$\frac{1}{2}$ left(opp, lccc)
$\oplus$ oppp@$\frac{1}{2}$ right(opp, lccc))) $\oplus$
(oppp = null $\multimap$ opp@$\frac{1}{2}$ count(lccc))$\Big)$ $\otimes$
(opp@$\frac{1}{2}$ left(this, lcc) $\oplus$ opp@$\frac{1}{2}$ right(this, lcc)) $\otimes$
$\otimes$ opp@$\frac{k}{2}$ parent() $\otimes$

unpacked(this@$\frac{1}{2}$ count(lcc)) $\otimes$
$\exists$ ol, lc, or, rc.
this.count $\rightarrow$ lcc $\otimes$
lcc = lc + rc + 1 $\otimes$
this@$\frac{1}{2}$ right(or, rc) $\otimes$this@$\frac{1}{2}$ left(ol, lc) }

this is either the right or left child of opp. We analyze both cases.
1.In the first case we assume it's the right child.
instantiate orr = this, rrc = lcc; merge both $\frac{1}{2}$ to opp in right
{$\exists$ k1, opp, lcc, k.
unpacked(this@k1 parent()) $\otimes$
this.parent $\rightarrow$ opp $\otimes$ opp $\neq$ this $\otimes$
unpacked(opp@$\frac{k}{2}$ parent()) $\otimes$ $\exists$ oppp, lccc, kk. opp.parent $\rightarrow$ oppp
$\otimes$ opp $\neq$ oppp $\otimes$

unpacked(opp@$\frac{1}{2}$ count(lccc)) $\otimes$ $\exists$ oll, llc. opp.count $\rightarrow$ lccc $\otimes$
lccc = llc + lcc + 1 $\otimes$ opp@$\frac{1}{2}$ left(oll, llc) $\otimes$

$\Big($(oppp $\neq$ `null` $\multimap$ oppp@kk parent() $\otimes$ (oppp@$\frac{1}{2}$ left(opp, lccc)

$\oplus$ oppp@$\frac{1}{2}$ right(opp, lccc))) $\oplus$
(oppp = null $\multimap$ opp@$\frac{1}{2}$ count(lccc))) $\otimes$
$\otimes$ opp@$\frac{k}{2}$ parent() $\otimes$

opp@1 right(this, lcc) $\otimes$

unpacked(this@$\frac{1}{2}$ count(lcc)) $\otimes$
$\exists$ ol, lc, or, rc.
this.count $\rightarrow$ lcc $\otimes$
lcc = lc + rc + 1 $\otimes$
this@$\frac{1}{2}$ right(or, rc) $\otimes$this@$\frac{1}{2}$ left(ol, lc) }

unpack opp from right(this, lcc)
{$\exists$ k1, opp, lcc, k.
unpacked(this@k1 parent()) $\otimes$
this.parent $\rightarrow$ opp $\otimes$ opp $\neq$ this $\otimes$
unpacked(opp@$\frac{k}{2}$ parent()) $\otimes$ $\exists$ oppp, lccc, kk. opp.parent $\rightarrow$ oppp $\otimes$ opp $\neq$ oppp $\otimes$

unpacked(opp@$\frac{1}{2}$ count(lccc)) $\otimes$ $\exists$ oll, llc. opp.count $\rightarrow$ lccc $\otimes$
lccc = llc + lcc + 1 $\otimes$ opp@$\frac{1}{2}$ left(oll, llc) $\otimes$

((oppp $\neq$ `null` $\multimap$ oppp@kk parent() $\otimes$ (oppp@$\frac{1}{2}$ left(opp, lccc) $\oplus$
oppp@$\frac{1}{2}$ right(opp, lccc))) $\oplus$
(oppp = null $\multimap$ opp@$\frac{1}{2}$ count(lccc))) $\otimes$
$\otimes$ opp@$\frac{k}{2}$ parent() $\otimes$

unpacked(opp@1 right(this, lcc)) $\otimes$
opp.right $\rightarrow$ this $\otimes$ this@$\frac{1}{2}$ count(lcc) $\otimes$

unpacked(this@$\frac{1}{2}$ count(lcc)) $\otimes$
$\exists$ ol, lc, or, rc.
this.count $\rightarrow$ lcc $\otimes$
lcc = lc + rc + 1 $\otimes$
this@$\frac{1}{2}$ right(or, rc) $\otimes$this@$\frac{1}{2}$ left(ol, lc) }

pack this @ $\frac{1}{2}$ count(lcc), add it to the other half
then unpack the count predicate
{$\exists$ k1, opp, lcc, k.
unpacked(this@k1 parent()) $\otimes$
this.parent $\rightarrow$ opp $\otimes$ opp $\neq$ this $\otimes$
unpacked(opp@$\frac{k}{2}$ parent()) $\otimes$ $\exists$ oppp, lccc, kk. opp.parent $\rightarrow$ oppp

$\otimes\ \mathsf{opp} \neq \mathsf{oppp}\ \otimes$

$\text{unpacked}(\text{opp}@\frac{1}{2}\ \text{count}(\text{lccc})) \otimes \exists\ \text{oll, llc.}\ \mathsf{opp.count} \to \mathsf{lccc} \otimes$
$\mathsf{lccc} = \mathsf{llc} + \mathsf{lcc} + 1 \otimes \text{opp}@\frac{1}{2}\ \text{left}(\text{oll, llc}) \otimes$

$\Big((\mathsf{oppp} \neq \texttt{null} \multimap \text{oppp}@\text{kk parent}() \otimes (\text{oppp}@\frac{1}{2}\ \text{left}(\text{opp, lccc}) \oplus$
$\text{oppp}@\frac{1}{2}\ \text{right}(\text{opp, lccc}))) \oplus$
$(\mathsf{oppp} = \mathsf{null} \multimap \text{opp}@\frac{1}{2}\ \text{count}(\text{lccc}))\Big) \otimes$
$\otimes\ \ \text{opp}@\frac{k}{2}\ \text{parent}() \otimes$

$\text{unpacked}(\text{opp}@1\ \text{right}(\text{this, lcc})) \otimes$
$\mathsf{opp.right} \to \mathsf{this} \otimes$

$\text{unpacked}(\text{this}@1\ \text{count}(\text{lcc})) \otimes$
$\exists\ \mathsf{ol, lc, or, rc.}$
$\mathsf{this.count} \to \mathsf{lcc} \otimes$
$\mathsf{lcc} = \mathsf{lc} + \mathsf{rc} + 1 \otimes$
$\text{this}@\frac{1}{2}\ \text{right}(\text{or, rc}) \otimes \text{this}@\frac{1}{2}\ \text{left}(\mathsf{ol, lc})\ \}$

```
this.updateCount();
```

$\{\exists\ \mathsf{k1, opp, lcc, k.}$
$\text{unpacked}(\text{this}@\text{k1 parent}()) \otimes$
$\mathsf{this.parent} \to \mathsf{opp} \otimes \mathsf{opp} \neq \mathsf{this} \otimes$
$\text{unpacked}(\text{opp}@\frac{k}{2}\ \text{parent}()) \otimes \exists\ \mathsf{oppp, lccc, kk.\ opp.parent} \to$
$\mathsf{oppp} \otimes \mathsf{opp} \neq \mathsf{oppp} \otimes$

$\text{unpacked}(\text{opp}@\frac{1}{2}\ \text{count}(\text{lccc})) \otimes \exists\ \text{oll, llc.}\ \mathsf{opp.count} \to \mathsf{lccc} \otimes$
$\mathsf{lccc} = \mathsf{llc} + \mathsf{lcc} + 1 \otimes \text{opp}@\frac{1}{2}\ \text{left}(\text{oll, llc}) \otimes$

$\Big((\mathsf{oppp} \neq \texttt{null} \multimap \text{oppp}@\text{kk parent}() \otimes (\text{oppp}@\frac{1}{2}\ \text{left}(\text{opp, lccc})$
$\oplus\ \text{oppp}@\frac{1}{2}\ \text{right}(\text{opp, lccc}))) \oplus$
$(\mathsf{oppp} = \mathsf{null} \multimap \text{opp}@\frac{1}{2}\ \text{count}(\text{lccc}))\Big) \otimes$
$\otimes\ \ \text{opp}@\frac{k}{2}\ \text{parent}() \otimes$

$\text{unpacked}(\text{opp}@1\ \text{right}(\text{this, lcc})) \otimes$
$\mathsf{opp.right} \to \mathsf{this} \otimes$

$\text{this}@1\ \text{count}(\text{lcc})\ \}$

pack opp in right(this, lcc)
$\{\exists\ \mathsf{k1, opp, lcc, k.}$

$\text{unpacked}(\text{this}@\text{k1 parent}()) \otimes$
$\mathsf{this.parent} \rightarrow \mathsf{opp} \otimes \mathsf{opp} \neq \mathsf{this} \otimes$
$\text{unpacked}(\text{opp}@\frac{k}{2}\ \text{parent}()) \otimes \exists\ \mathsf{oppp,\ lccc,\ kk.\ opp.parent} \rightarrow \mathsf{oppp}$
$\otimes\ \mathsf{opp} \neq \mathsf{oppp} \otimes$

$\text{unpacked}(\text{opp}@\frac{1}{2}\ \text{count}(\text{lccc})) \otimes \exists\ \text{oll, llc.}\ \mathsf{opp.count} \rightarrow \mathsf{lccc} \otimes$
$\mathsf{lccc} = \mathsf{llc} + \mathsf{lcc} + 1 \otimes \text{opp}@\frac{1}{2}\ \text{left}(\text{oll, llc}) \otimes$

$\Big((\mathsf{oppp} \neq \texttt{null} \multimap \text{oppp}@\text{kk parent}() \otimes (\text{oppp}@\frac{1}{2}\ \text{left}(\text{opp, lccc})$
$\oplus\ \text{oppp}@\frac{1}{2}\ \text{right}(\text{opp, lccc}))\big) \oplus$
$(\mathsf{oppp} = \mathsf{null} \multimap \text{opp}@\frac{1}{2}\ \text{count}(\text{lccc}))\Big) \otimes$
$\otimes\ \ \text{opp}@\frac{k}{2}\ \text{parent}() \otimes$

$\text{opp}@1\ \text{right}(\text{this, lcc}) \otimes$

$\text{this}@\frac{1}{2}\ \text{count}(\text{lcc})\ \}$

```
this.parent.updateCountRec();
```

$\{\exists\ \mathsf{k1,\ opp,\ lcc,\ k,\ k3.}$
$\text{unpacked}(\text{this}@\text{k1 parent}()) \otimes$
$\mathsf{this.parent} \rightarrow \mathsf{opp} \otimes \mathsf{opp} \neq \mathsf{this} \otimes$
$\text{unpacked}(\text{opp}@\frac{k}{2}\ \text{parent}()) \otimes \exists\ \mathsf{oppp,\ lccc,\ kk.\ opp.parent} \rightarrow \mathsf{oppp} \otimes \mathsf{opp}$
$\neq \mathsf{oppp} \otimes$

$\Big((\mathsf{oppp} \neq \texttt{null} \multimap \text{oppp}@\text{kk parent}() \otimes (\text{oppp}@\frac{1}{2}\ \text{left}(\text{opp, lccc}) \oplus$
$\text{oppp}@\frac{1}{2}\ \text{right}(\text{opp, lccc}))\big) \oplus$
$(\mathsf{oppp} = \mathsf{null} \multimap \text{opp}@\frac{1}{2}\ \text{count}(\text{lccc}))\Big) \otimes$

$\text{opp}@\frac{1}{2}\ \text{right}(\text{this, lcc}) \otimes$

$\text{this}@\frac{1}{2}\ \text{count}(\text{lcc}) \otimes$
$\text{opp}@\text{k3 parent}()\ \}$

pack opp in parent()
$\{\exists\ \mathsf{k1,\ opp,\ lcc,\ k,\ k3.}$
$\text{unpacked}(\text{this}@\text{k1 parent}()) \otimes$
$\mathsf{this.parent} \rightarrow \mathsf{opp} \otimes \mathsf{opp} \neq \mathsf{this} \otimes$
$\text{unpacked}(\text{opp}@\frac{k}{2}\ \text{parent}()) \otimes \exists\ \mathsf{oppp,\ lccc,\ kk.\ opp.parent} \rightarrow \mathsf{oppp} \otimes \mathsf{opp}$
$\neq \mathsf{oppp} \otimes$

$\Big((\mathsf{oppp} \neq \texttt{null} \multimap \text{oppp}@\text{kk parent}() \otimes (\text{oppp}@\frac{1}{2}\ \text{left}(\text{opp, lccc}) \oplus$

$\text{oppp@}\frac{1}{2}\ \text{right(opp, lccc)))}\ \oplus$
(oppp = null $\multimap$ $\text{opp@}\frac{1}{2}\ \text{count(lccc)}$)$\Big)$ $\otimes$

$\text{opp@}\frac{1}{2}\ \text{right(this, lcc)}\ \otimes$

$\text{this@}\frac{1}{2}\ \text{count(lcc)}\ \otimes$
$\text{opp@k3 parent()}\ \}$

pack this in parent(), assuming opp is not null.
It's not since we just called updateCountRec on it, we are on that branch.
$\{\exists\ \text{k1. this@k1 parent()}\}$
QED

2.In the second case we assume it's the left child.
instantiate oll = this, llc = lcc; merge both $\frac{1}{2}$ to opp in right
$\{\exists$ k1, opp, lcc, k.
$\text{unpacked(this@k1 parent())}\ \otimes$
this.parent $\rightarrow$ opp $\otimes$ opp $\neq$ this $\otimes$
$\text{unpacked(opp@}\frac{k}{2}\ \text{parent())}\ \otimes\ \exists$ oppp, lccc, kk. opp.parent $\rightarrow$ oppp
$\otimes$ opp $\neq$ oppp $\otimes$

$\text{unpacked(opp@}\frac{1}{2}\ \text{count(lccc))}\ \otimes\ \exists\ \text{orr, rrc.}$ opp.count $\rightarrow$ lccc $\otimes$
lccc = lcc + rrc + 1 $\otimes$ $\text{opp@}\frac{1}{2}\ \text{right(orr, rrc)}\ \otimes$

$\Big($(oppp $\neq$ `null` $\multimap$ $\text{oppp@kk parent()}\ \otimes\ (\text{oppp@}\frac{1}{2}\ \text{left(opp, lccc)}$
$\oplus\ \text{oppp@}\frac{1}{2}\ \text{right(opp, lccc)))}\ \oplus$
(oppp = null $\multimap$ $\text{opp@}\frac{1}{2}\ \text{count(lccc)}$)$\Big)$ $\otimes$
$\otimes\ \ \text{opp@}\frac{k}{2}\ \text{parent()}\ \otimes$

opp@1 left(this, lcc$)\ \otimes$

$\text{unpacked(this@}\frac{1}{2}\ \text{count(lcc))}\ \otimes$
$\exists$ ol, lc, or, rc.
this.count $\rightarrow$ lcc $\otimes$
lcc = lc + rc + 1 $\otimes$
$\text{this@}\frac{1}{2}\ \text{right(or, rc)}\ \otimes\text{this@}\frac{1}{2}\ \text{left(}ol, lc)\ \}$

unpack opp from left(this, lcc)
$\{\exists$ k1, opp, lcc, k.
$\text{unpacked(this@k1 parent())}\ \otimes$
this.parent $\rightarrow$ opp $\otimes$ opp $\neq$ this $\otimes$
$\text{unpacked(opp@}\frac{k}{2}\ \text{parent())}\ \otimes\ \exists$ oppp, lccc, kk. opp.parent $\rightarrow$ oppp

$\otimes$ opp $\neq$ oppp $\otimes$

$\mathrm{unpacked}(\mathrm{opp}@\frac{1}{2}\ \mathrm{count}(\mathrm{lccc})) \otimes \exists\ \mathrm{orr, rrc.}$ opp.count $\rightarrow$ lccc $\otimes$
lccc = rrc + lcc + 1 $\otimes$ $\mathrm{opp}@\frac{1}{2}\ \mathrm{right}(\mathrm{orr, rrc}) \otimes$

$\Big($(oppp $\neq$ `null` $\multimap$ $\mathrm{oppp}@\mathrm{kk\ parent}() \otimes (\mathrm{oppp}@\frac{1}{2}\ \mathrm{left}(\mathrm{opp, lccc})$
$\oplus\ \mathrm{oppp}@\frac{1}{2}\ \mathrm{right}(\mathrm{opp, lccc})))\ \oplus$
(oppp = null $\multimap$ $\mathrm{opp}@\frac{1}{2}\ \mathrm{count}(\mathrm{lccc}))\Big) \otimes$
$\otimes\ \ \mathrm{opp}@\frac{k}{2}\ \mathrm{parent}() \otimes$

$\mathrm{unpacked}(\mathrm{opp}@1\ \mathrm{left}(\mathrm{this, lcc})) \otimes$
opp.left $\rightarrow$ this $\otimes$ $\mathrm{this}@\frac{1}{2}\ \mathrm{count}(\mathrm{lcc}) \otimes$

$\mathrm{unpacked}(\mathrm{this}@\frac{1}{2}\ \mathrm{count}(\mathrm{lcc})) \otimes$
$\exists$ ol, lc, or, rc.
this.count $\rightarrow$ lcc $\otimes$
lcc = lc + rc + 1 $\otimes$
$\mathrm{this}@\frac{1}{2}\ \mathrm{right}(\mathrm{or, rc}) \otimes \mathrm{this}@\frac{1}{2}\ \mathrm{left}(ol, lc)$ }

pack this @ $\frac{1}{2}$ count(lcc), add it to the other half
then unpack the count predicate
{$\exists$ k1, opp, lcc, k.
$\mathrm{unpacked}(\mathrm{this}@\mathrm{k1\ parent}()) \otimes$
this.parent $\rightarrow$ opp $\otimes$ opp $\neq$ this $\otimes$
$\mathrm{unpacked}(\mathrm{opp}@\frac{k}{2}\ \mathrm{parent}()) \otimes \exists$ oppp, lccc, kk. opp.parent $\rightarrow$ oppp
$\otimes$ opp $\neq$ oppp $\otimes$

$\mathrm{unpacked}(\mathrm{opp}@\frac{1}{2}\ \mathrm{count}(\mathrm{lccc})) \otimes \exists\ \mathrm{orr, rrc.}$ opp.count $\rightarrow$ lccc $\otimes$
lccc = rrc + lcc + 1 $\otimes$ $\mathrm{opp}@\frac{1}{2}\ \mathrm{right}(\mathrm{orr, rrc}) \otimes$

$\Big($(oppp $\neq$ `null` $\multimap$ $\mathrm{oppp}@\mathrm{kk\ parent}() \otimes (\mathrm{oppp}@\frac{1}{2}\ \mathrm{left}(\mathrm{opp, lccc})$
$\oplus\ \mathrm{oppp}@\frac{1}{2}\ \mathrm{right}(\mathrm{opp, lccc})))\ \oplus$
(oppp = null $\multimap$ $\mathrm{opp}@\frac{1}{2}\ \mathrm{count}(\mathrm{lccc}))\Big) \otimes$
$\otimes\ \ \mathrm{opp}@\frac{k}{2}\ \mathrm{parent}() \otimes$

$\mathrm{unpacked}(\mathrm{opp}@1\ \mathrm{left}(\mathrm{this, lcc})) \otimes$
opp.left $\rightarrow$ this $\otimes$

$\mathrm{unpacked}(\mathrm{this}@1\ \mathrm{count}(\mathrm{lcc})) \otimes$
$\exists$ ol, lc, or, rc.
this.count $\rightarrow$ lcc $\otimes$
lcc = lc + rc + 1 $\otimes$

$\text{this}@\frac{1}{2}\ \text{right}(\text{or}, \text{rc}) \otimes \text{this}@\frac{1}{2}\ \text{left}(\mathsf{ol}, \mathsf{lc})\ \}$

```
this.updateCount();
```

$\{\exists\ \mathsf{k1, opp, lcc, k.}$

$\text{unpacked}(\text{this}@\text{k1 parent}()) \otimes$
$\mathsf{this.parent} \rightarrow \mathsf{opp} \otimes \mathsf{opp} \neq \mathsf{this} \otimes$
$\text{unpacked}(\text{opp}@\frac{k}{2}\ \text{parent}()) \otimes \exists\ \mathsf{oppp, lccc, kk.\ opp.parent} \rightarrow \mathsf{oppp} \otimes \mathsf{opp} \neq \mathsf{oppp} \otimes$

$\text{unpacked}(\text{opp}@\frac{1}{2}\ \text{count}(\text{lccc})) \otimes \exists\ \text{orr, rrc.}\ \mathsf{opp.count} \rightarrow \mathsf{lccc} \otimes$
$\mathsf{lccc} = \mathsf{rrc} + \mathsf{lcc} + 1 \otimes \text{opp}@\frac{1}{2}\ \text{right}(\text{orr}, \text{rrc}) \otimes$

$\Big(\big(\mathsf{oppp} \neq \texttt{null} \multimap \text{oppp}@\text{kk parent}() \otimes (\text{oppp}@\frac{1}{2}\ \text{left}(\text{opp}, \text{lccc})$
$\oplus\ \text{oppp}@\frac{1}{2}\ \text{right}(\text{opp}, \text{lccc}))\big) \oplus$
$(\mathsf{oppp} = \mathsf{null} \multimap \text{opp}@\frac{1}{2}\ \text{count}(\text{lccc}))\Big) \otimes$
$\otimes\ \ \text{opp}@\frac{k}{2}\ \text{parent}() \otimes$

$\text{unpacked}(\text{opp}@1\ \text{left}(\text{this}, \text{lcc})) \otimes$
$\mathsf{opp.left} \rightarrow \mathsf{this} \otimes$

$\text{this}@1\ \text{count}(\text{lcc})\ \}$

pack opp in left(this, lcc)

$\{\exists\ \mathsf{k1, opp, lcc, k.}$

$\text{unpacked}(\text{this}@\text{k1 parent}()) \otimes$
$\mathsf{this.parent} \rightarrow \mathsf{opp} \otimes \mathsf{opp} \neq \mathsf{this} \otimes$
$\text{unpacked}(\text{opp}@\frac{k}{2}\ \text{parent}()) \otimes \exists\ \mathsf{oppp, lccc, kk.\ opp.parent} \rightarrow \mathsf{oppp} \otimes \mathsf{opp} \neq \mathsf{oppp} \otimes$

$\text{unpacked}(\text{opp}@\frac{1}{2}\ \text{count}(\text{lccc})) \otimes \exists\ \text{orr, rrc.}\ \mathsf{opp.count} \rightarrow \mathsf{lccc} \otimes$
$\mathsf{lccc} = \mathsf{rrc} + \mathsf{lcc} + 1 \otimes \text{opp}@\frac{1}{2}\ \text{right}(\text{orr}, \text{rrc}) \otimes$

$\Big(\big(\mathsf{oppp} \neq \texttt{null} \multimap \text{oppp}@\text{kk parent}() \otimes (\text{oppp}@\frac{1}{2}\ \text{left}(\text{opp}, \text{lccc})$
$\oplus\ \text{oppp}@\frac{1}{2}\ \text{right}(\text{opp}, \text{lccc}))\big) \oplus$
$(\mathsf{oppp} = \mathsf{null} \multimap \text{opp}@\frac{1}{2}\ \text{count}(\text{lccc}))\Big) \otimes$
$\otimes\ \ \text{opp}@\frac{k}{2}\ \text{parent}() \otimes$

$\text{opp}@1\ \text{left}(\text{this}, \text{lcc}) \otimes$

$\text{this}@\frac{1}{2}\ \text{count}(\text{lcc})\ \}$

```
this.parent.updateCountRec();
```

$\{\exists$ k1, opp, lcc, k, k3.

unpacked(this@k1 parent()) $\otimes$

this.parent $\rightarrow$ opp $\otimes$ opp $\neq$ this $\otimes$

unpacked(opp@$\frac{k}{2}$ parent()) $\otimes$ $\exists$ oppp, lccc, kk. opp.parent $\rightarrow$ oppp $\otimes$ opp $\neq$ oppp $\otimes$

$\Big($(oppp $\neq$ `null` $\multimap$ oppp@kk parent() $\otimes$ (oppp@$\frac{1}{2}$ left(opp, lccc) $\oplus$ oppp@$\frac{1}{2}$ right(opp, lccc))) $\oplus$

(oppp = null $\multimap$ opp@$\frac{1}{2}$ count(lccc))$\Big)$ $\otimes$

opp@$\frac{1}{2}$ left(this, lcc) $\otimes$

this@$\frac{1}{2}$ count(lcc) $\otimes$

opp@k3 parent()$\}$

pack this in parent(), assuming opp is not null.

It's not since we just called updateCountRec on it, we are on that branch.

$\{\exists$ k1. this@k1 parent()$\}$

QED

else

$\{$ unpacked(this@k1 parent()) $\otimes$ $\exists$ opp, lcc. unpacked(this@$\frac{1}{2}$ count(lcc)) $\otimes$

$\exists$ ol, lc. this@$\frac{1}{2}$ left(l, lc) $\otimes$

$\exists$ or, rc, lc1. this.count $\rightarrow$ lcc $\otimes$

lcc = lc1 + rc + 1 $\otimes$

this@$\frac{1}{2}$ right(or, rc) $\otimes$

this.parent $\rightarrow$ opp $\otimes$

opp $\neq$ this $\otimes$ opp = null $\otimes$ this@$\frac{1}{2}$ count(lcc) $\}$

merge this, $\frac{1}{2}$ in count(lcc) from packed and unpacked

$\{$ unpacked(this@k1 parent()) $\otimes$ $\exists$ opp, lcc. unpacked(this@1 count(lcc)) $\otimes$

$\exists$ ol, lc. this@$\frac{1}{2}$ left(l, lc) $\otimes$

$\exists$ or, rc, lc1. this.count $\rightarrow$ lcc $\otimes$

lcc = lc1 + rc + 1 $\otimes$

this@$\frac{1}{2}$ right(or, rc) $\otimes$

this.parent $\rightarrow$ opp $\otimes$

opp $\neq$ this $\otimes$ opp = null $\}$

```
        this.updateCount();


{ unpacked(this@k1 parent()) ⊗ ∃ opp, lcc. this@1 count(lcc) ⊗
    this.parent → opp ⊗
    opp ≠ this ⊗ opp = null }

        split count in half and pack this in parent
        {∃ k1. this@k1 parent() }
        QED
  }

  private void updateCount()
  ∃ c, c1, c2, ol, or. unpacked(this@1 count(c)) ⊗
      this.count → c ⊗ c = c1 + c2 + 1 ⊗
      this@1/2 left(ol, c1) ⊗this@1/2 right(or, c2)
      ⊸ ∃ c. this@1 count(c)
    {
     int newc = 1;
     unpack this @1/2left(ol,c1)
     { newc = 1 ⊗
         unpacked(this@1 count(c)) ⊗
         this.count → c ⊗ c = c1 + c2 + 1 ⊗
         unpacked(this@1/2 left(ol, c1)) ⊗
         this.left → ol ⊗ (ol = null ⊸ c1 = 0) ⊗
         (ol ≠ null ⊸ ol@1/2 count(c1))  ⊗
         this@1/2 right(or, c2) }
     if (this.left != null)
       { newc = 1 ⊗
           unpacked(this@1 count(c)) ⊗
           this.count → c ⊗ c = c1 + c2 + 1 ⊗
           unpacked(this@1/2 left(ol, c1)) ⊗
           this.left → ol ⊗
           ol@1/2 count(c1)  ⊗
           this@1/2 right(or, c2) }
       unpack ol in 1/2 count(c1)
       { newc = 1 ⊗
           unpacked(this@1 count(c)) ⊗
           this.count → c ⊗ c = c1 + c2 + 1 ⊗
           unpacked(this@1/2 left(ol, c1)) ⊗
           this.left → ol ⊗
           unpacked(ol@1/2 count(c1)) ⊗
           ∃ lol,lor,llc,lrc. ol.count → c1 ⊗
           c1 = llc + lrc + 1 ⊗
           ol@1/2 left(lol, llc) ⊗
```

ol@$\frac{1}{2}$ right(lor, lrc) $\otimes$
this@$\frac{1}{2}$ right(or, c2) $\otimes$}
`newc = newc + left.count;`
{ newc $=$ 1 $+$ c1 $\otimes$
unpacked(this@1 count(c)) $\otimes$
this.count $\rightarrow$ c $\otimes$ c $=$ c1 $+$ c2 $+$ 1 $\otimes$
unpacked(this@$\frac{1}{2}$ left(ol, c1)) $\otimes$
this.left $\rightarrow$ ol $\otimes$
unpacked(ol@$\frac{1}{2}$ count(c1)) $\otimes$
$\exists$ lol,lor,llc,lrc. ol.count $\rightarrow$ c1 $\otimes$
c1 $=$ llc $+$ lrc $+$ 1 $\otimes$
ol@$\frac{1}{2}$ left(lol, llc) $\otimes$ ol@$\frac{1}{2}$ right(lor, lrc) $\otimes$
this@$\frac{1}{2}$ right(or, c2) }
pack ol in count(c1)
{ newc $=$ 1 $+$ lc $\otimes$
unpacked(this@1 count(c)) $\otimes$
this.count $\rightarrow$ c $\otimes$ c $=$ c1 $+$ c2 $+$ 1 $\otimes$
unpacked(this@$\frac{1}{2}$ left(ol, c1)) $\otimes$
this.left $\rightarrow$ ol $\otimes$
ol@$\frac{1}{2}$ count(c1) $\otimes$
this@$\frac{1}{2}$ right(or, c2)}
pack this in left(ol, c1)
{ newc $=$ 1 $+$ c1 $\otimes$
unpacked(this@1 count(c)) $\otimes$
this.count $\rightarrow$ c $\otimes$ c $=$ c1 $+$ c2 $+$ 1 $\otimes$
this@$\frac{1}{2}$ left(ol, c1) $\otimes$
this@$\frac{1}{2}$ right(or, c2) }
unpack this in $\frac{1}{2}$ right(or, c2)
{ newc $=$ 1 $+$ c1 $\otimes$
unpacked(this@1 count(c)) $\otimes$
this.count $\rightarrow$ c $\otimes$ c $=$ c1 $+$ c2 $+$ 1 $\otimes$
this@$\frac{1}{2}$ left(ol, c1) $\otimes$
unpacked(this@$\frac{1}{2}$ right(or, c2)) $\otimes$
this.right $\rightarrow$ or $\otimes$
((or $\neq$ null $\multimap$ or@$\frac{1}{2}$ count(c2)) $\oplus$
(or = null $\multimap$ c2 = 0)) }
if (**this**.right != **null**)
{ newc $=$ 1 $+$ c1 $\otimes$
unpacked(this@1 count(c)) $\otimes$
this.count $\rightarrow$ c $\otimes$ c $=$ c1 $+$ c2 $+$ 1 $\otimes$
this@$\frac{1}{2}$ left(ol, c1) $\otimes$
unpacked(this@$\frac{1}{2}$ right(or, c2)) $\otimes$
this.right $\rightarrow$ or $\otimes$
or@$\frac{1}{2}$ count(c2) }
unpack or in $\frac{1}{2}$ count(c2)

```
  { newc = 1 + c1 ⊗
      unpacked(this@1 count(c)) ⊗
      this.count → c ⊗ c = c1 + c2 + 1 ⊗
      this@1/2 left(ol, c1) ⊗
      unpacked(this@1/2 right(or, c2)) ⊗
      this.right → or ⊗
      unpacked(or@1/2 count(c2)) ⊗
      ∃ rol,ror,rlc,rrc. or.count → c2 ⊗
      c2 = rlc + rrc + 1 ⊗
      or@1/2 left(rol, rlc) ⊗
      or@1/2 right(ror, rrc) }
  newc = newc + right.count;
  { newc = 1 + c1 + c2 ⊗
      unpacked(this@1 count(c)) ⊗
      this.count → c ⊗ c = c1 + c2 + 1 ⊗
      this@1/2 left(ol, c1) ⊗
      unpacked(this@1/2 right(or, c2)) ⊗
      this.right → or ⊗
      unpacked(or@1/2 count(c2)) ⊗
      ∃ rol,ror,rlc,rrc. or.count → c2 ⊗
      c2 = rlc + rrc + 1 ⊗
      or@1/2 left(rol, rlc) ⊗
      or@1/2 right(ror, rrc) }
  pack or in count
  { newc = 1 + c1 + c2 ⊗
      unpacked(this@ 1 count(c)) ⊗
      this.count → c ⊗ c = c1 + c2 + 1 ⊗
      this@1/2 left(ol, c1) ⊗
      unpacked(this@1/2 right(or, c2)) ⊗
      this.right → or ⊗
      or@1/2 count(c2) }
  pack this in right
  { newc = 1 + c1 + c2 ⊗
      unpacked(this@1 count(c)) ⊗
      this.count → c ⊗ c = c1 + c2 + 1 ⊗
      this@1/2 left(ol, c1) ⊗
      this@1/2 right(or, c2) }
this.count = newc;
{ newc = 1 + c1 + c2 ⊗
    unpacked(this@1 count(c)) ⊗
    this.count → c ⊗ c = newc ⊗
    this@1/2 left(ol, c1) ⊗
    this@1/2 right(or, c2) }
pack this in count(newc)
{ this@1 count(newc) }
QED
```

}

```
public void setLeft (Composite l)
```

this $\neq$ l $\otimes$
$\exists k1, k2.(\text{this@k1 parent()} \otimes \text{l@k2 parent()} \otimes$
$\text{this@}\frac{1}{2}\text{ left(}\texttt{null}\text{, 0)} \multimap$
$\exists$ k, k2.this@k parent() $\otimes$ l@k2 parent())

{

unpack l from parent

{ unpacked(l@k2 parent()) $\otimes$ $\exists$ op, lc, k,k1,k3. l.parent $\rightarrow$ op $\otimes$
op $\neq$ l $\otimes$ l@$\frac{1}{2}$ count(lc) $\otimes$
$\Big($(op $\neq$ null $\multimap$
op@k3 parent() $\otimes$
(op@$\frac{1}{2}$ left(l, lc) $\oplus$
op@$\frac{1}{2}$ right(l, lc))) $\oplus$
(op = null $\multimap$ l@$\frac{1}{2}$ count(lc)) $\Big)$ $\otimes$ this $\neq$ l $\otimes$
this@k1 parent() $\otimes$ this@$\frac{1}{2}$ left($\texttt{null}$, 0) }

We want to show that op is null.
Let's assume that op is not null.
Then there must exist o1$\neq$o2 such that o1.left=this and o2.left=this.
or such that o1.left=this and o2.right=this, etc. (the reasoning is the same)
It means that o1 and o2 both are the parents of this.
Then the count predicate of o1 contains half permission to this,
the count predicate of o2 contains half permission to this,
and the parent predicate of this also contains half permission to this.
Thus the sum of the permissions is more than 1, but that is a contradiction.
Thus op must be null.

{ unpacked(l@k2 parent()) $\otimes$ $\exists$ lc, k1. l.parent $\rightarrow$ $\texttt{null}$ $\otimes$
$\texttt{null}$ $\neq$ l $\otimes$ l@$\frac{1}{2}$ count(lc) $\otimes$
l@$\frac{1}{2}$ count(lc) $\otimes$ this $\neq$ l $\otimes$
this@k1 parent() $\otimes$ this@$\frac{1}{2}$ left($\texttt{null}$, 0) }

```
l.parent = this;
```

assignment rule

{ unpacked(l@k2 parent()) $\otimes$ $\exists$ lc. l.parent $\rightarrow$ this $\otimes$
$\texttt{null}$ $\neq$ l $\otimes$ l@$\frac{1}{2}$ count(lc) $\otimes$
l@$\frac{1}{2}$ count(lc) $\otimes$ this $\neq$ l $\otimes$
this@k1 parent() $\otimes$ this@$\frac{1}{2}$ left($\texttt{null}$, 0) }

unpack this from parent

{ unpacked(l@k2 parent()) $\otimes$ unpacked(this@k1 parent()) $\otimes$ $\exists$
opp, lcc, k,k4. this.parent $\rightarrow$ opp $\otimes$
opp $\neq$ this $\otimes$ this@$\frac{1}{2}$ count(lcc) $\otimes$
$\Big($(opp $\neq$ null $\multimap$
opp@k4 parent() $\otimes$

$(\mathrm{opp}@\frac{1}{2}\ \mathrm{left}(\mathsf{this},\ \mathsf{lcc})\ \oplus$
$\mathrm{opp}@\frac{1}{2}\ \mathrm{right}(\mathsf{this},\ \mathsf{lcc})))\ \oplus$
$(\mathsf{opp} = \mathsf{null} \multimap \mathrm{this}@\frac{1}{2}\ \mathrm{count}(\mathsf{lcc}))\ \Big)\ \otimes$
$\exists\ \mathsf{lc}.\ \mathsf{l.parent} \rightarrow \mathsf{this}\ \otimes$
$\mathtt{null} \neq \mathsf{l} \otimes \mathrm{l}@\frac{1}{2}\ \mathrm{count}(\mathsf{lc})\ \otimes$
$\mathrm{l}@\frac{1}{2}\ \mathrm{count}(\mathsf{lc}) \otimes \mathsf{this} \neq \mathsf{l}\ \otimes$
$\mathrm{this}@\frac{1}{2}\ \mathrm{left}(\mathtt{null},\ 0)\ \}$

unpack this from $\frac{1}{2}$ count(lcc)

$\{\ \mathrm{unpacked}(\mathrm{l}@\mathrm{k2}\ \mathrm{parent}()) \otimes \mathrm{unpacked}(\mathrm{this}@\mathrm{k1}\ \mathrm{parent}()) \otimes \exists\ \mathsf{opp},\ \mathsf{lcc},\ \mathsf{k}.$
$\mathrm{unpacked}(\mathrm{this}@\frac{1}{2}\ \mathrm{count}(\mathrm{lcc})) \otimes \exists\ \mathsf{ol},\ \mathsf{llc},\ \mathsf{or},\ \mathsf{rc}.\ \mathsf{this.count} \rightarrow \mathsf{lcc}\ \otimes$
$\mathsf{lcc} = \mathsf{llc} + \mathsf{rc} + 1\ \otimes$
$\mathrm{this}@\frac{1}{2}\ \mathrm{left}(\mathrm{ol},\ \mathrm{llc})\ \otimes$
$\mathrm{this}@\frac{1}{2}\ \mathrm{right}(\mathrm{or},\ \mathrm{rc})\ \otimes$

$\mathsf{this.parent} \rightarrow \mathsf{opp}\ \otimes$
$\mathsf{opp} \neq \mathsf{this}\ \otimes$
$\Big((\mathsf{opp} \neq \mathsf{null} \multimap$
$\mathrm{opp}@\frac{k}{2}\ \mathrm{parent}()\ \otimes$
$(\mathrm{opp}@\frac{1}{2}\ \mathrm{left}(\mathsf{this},\ \mathsf{lcc})\ \oplus$
$\mathrm{opp}@\frac{1}{2}\ \mathrm{right}(\mathsf{this},\ \mathsf{lcc})))\ \oplus$
$(\mathsf{opp} = \mathsf{null} \multimap \mathrm{this}@\frac{1}{2}\ \mathrm{count}(\mathsf{lcc}))\ \Big)\ \otimes$

$\exists\ \mathsf{lc}.\ \mathsf{l.parent} \rightarrow \mathsf{this}\ \otimes$
$\mathtt{null} \neq \mathsf{l} \otimes \mathrm{l}@\frac{1}{2}\ \mathrm{count}(\mathsf{lc})\ \otimes$
$\mathrm{l}@\frac{1}{2}\ \mathrm{count}(\mathsf{lc}) \otimes \mathsf{this} \neq \mathsf{l}\ \otimes$
$\mathrm{this}@\frac{1}{2}\ \mathrm{left}(\mathtt{null},\ 0)\ \}$

existentialize ol with null and llc with 0 (to unify left permissions)

$\{\ \mathrm{unpacked}(\mathrm{l}@\mathrm{k2}\ \mathrm{parent}()) \otimes \mathrm{unpacked}(\mathrm{this}@\mathrm{k1}\ \mathrm{parent}()) \otimes \exists\ \mathsf{opp},\ \mathsf{lcc},\ \mathsf{k}.$
$\mathrm{unpacked}(\mathrm{this}@\frac{1}{2}\ \mathrm{count}(\mathrm{lcc})) \otimes \exists\ \mathsf{or},\ \mathsf{rc}.\ \mathsf{this.count} \rightarrow \mathsf{lcc}\ \otimes$
$\mathsf{lcc} = 0 + \mathsf{rc} + 1\ \otimes$
$\mathrm{this}@\frac{1}{2}\ \mathrm{left}(\mathtt{null},\ 0)\ \otimes$
$\mathrm{this}@\frac{1}{2}\ \mathrm{right}(\mathrm{or},\ \mathrm{rc})\ \otimes$
$\mathsf{this.parent} \rightarrow \mathsf{opp}\ \otimes$

$\mathsf{opp} \neq \mathsf{this}\ \otimes$
$\Big((\mathsf{opp} \neq \mathsf{null} \multimap$
$\mathrm{opp}@\mathrm{k4}\ \mathrm{parent}()\ \otimes$
$(\mathrm{opp}@\frac{1}{2}\ \mathrm{left}(\mathsf{this},\ \mathsf{lcc})\ \oplus$
$\mathrm{opp}@\frac{1}{2}\ \mathrm{right}(\mathsf{this},\ \mathsf{lcc})))\ \oplus$
$(\mathsf{opp} = \mathsf{null} \multimap \mathrm{this}@\frac{1}{2}\ \mathrm{count}(\mathsf{lcc}))\ \Big)\ \otimes$

$\exists\ \mathsf{lc}.\ \mathsf{l.parent} \rightarrow \mathsf{this}\ \otimes$
$\mathtt{null} \neq \mathsf{l} \otimes \mathrm{l}@\frac{1}{2}\ \mathrm{count}(\mathsf{lc})\ \otimes$

$\mathrm{l@}\frac{1}{2}$ count(lc) $\otimes$ this $\neq$ l $\otimes$
$\mathrm{this@}\frac{1}{2}$ left(`null`, 0) }

merge the half fractions to left and unpack this in left(null, 0)

{ unpacked(l@k2 parent()) $\otimes$ unpacked(this@k1 parent()) $\otimes$ $\exists$ opp, lcc, k.
unpacked($\mathrm{this@}\frac{1}{2}$ count(lcc)) $\otimes$ unpacked(this@1 left(`null`, 0)) $\otimes$
this.left $\rightarrow$ `null` $\otimes$ $\exists$ or, rc. this.count $\rightarrow$ lcc $\otimes$
lcc = 0 + rc + 1 $\otimes$
$\mathrm{this@}\frac{1}{2}$ right(or, rc) $\otimes$
this.parent $\rightarrow$ opp $\otimes$
opp $\neq$ this $\otimes$
((opp $\neq$ null $\multimap$
opp@k4 parent() $\otimes$
($\mathrm{opp@}\frac{1}{2}$ left(this, lcc) $\oplus$
$\mathrm{opp@}\frac{1}{2}$ right(this, lcc))) $\oplus$
(opp = null $\multimap$ $\mathrm{this@}\frac{1}{2}$ count(lcc))) $\otimes$
$\exists$ lc. l.parent $\rightarrow$ this $\otimes$
`null` $\neq$ l $\otimes$ $\mathrm{l@}\frac{1}{2}$ count(lc) $\otimes$
$\mathrm{l@}\frac{1}{2}$ count(lc) $\otimes$ this $\neq$ l }

```
this.left = l;
```

assignment

{ unpacked(l@k2 parent()) $\otimes$ unpacked(this@k1 parent()) $\otimes$ $\exists$ opp, lcc, k.
unpacked($\mathrm{this@}\frac{1}{2}$ count(lcc)) $\otimes$ unpacked(this@1 left(`null`, 0)) $\otimes$
this.left $\rightarrow$ l $\otimes$ $\exists$ or, rc. this.count $\rightarrow$ lcc $\otimes$
lcc = lc + rc + 1 $\otimes$
$\mathrm{this@}\frac{1}{2}$ right(or, rc) $\otimes$
this.parent $\rightarrow$ opp $\otimes$
opp $\neq$ this $\otimes$
((opp $\neq$ null $\multimap$
opp@k4 parent() $\otimes$
($\mathrm{opp@}\frac{1}{2}$ left(this, lcc) $\oplus$
$\mathrm{opp@}\frac{1}{2}$ right(this, lcc))) $\oplus$
(opp = null $\multimap$ $\mathrm{this@}\frac{1}{2}$ count(lcc))) $\otimes$
$\exists$ lc. l.parent $\rightarrow$ this $\otimes$
`null` $\neq$ l $\otimes$ $\mathrm{l@}\frac{1}{2}$ count(lc) $\otimes$
$\mathrm{l@}\frac{1}{2}$ count(lc) $\otimes$ this $\neq$ l }

pack this in left(l, lc)

{ unpacked(l@k2 parent()) $\otimes$ unpacked(this@k1 parent()) $\otimes$ $\exists$
opp, lcc, k. unpacked($\mathrm{this@}\frac{1}{2}$ count(lcc)) $\otimes$
$\exists$ lc. this@1 left(l, lc) $\otimes$
$\exists$ or, rc. this.count $\rightarrow$ lcc $\otimes$
lcc = lc + rc + 1 $\otimes$
$\mathrm{this@}\frac{1}{2}$ right(or, rc) $\otimes$
this.parent $\rightarrow$ opp $\otimes$

```
        opp ≠ this ⊗
        ((opp ≠ null ⊸
        opp@k4 parent() ⊗
        (opp@1/2 left(this, lcc) ⊕
        opp@1/2 right(this, lcc))) ⊕
        (opp = null ⊸ this@1/2 count(lcc)) ) ⊗
        l.parent → this ⊗
        null ≠ l ⊗
        l@1/2 count(lc) ⊗ this ≠ l }
      this.updateCountRec();
{ ∃ k1, k2, lc. unpacked(l@k2 parent()) ⊗
    this@1/2 left(l, lc) ⊗
    l.parent → this ⊗
    null ≠ l ⊗
    this@k1 parent()
    l@1/2 count(lc) ⊗ this ≠ l }
    pack l in parent()
     {∃ k1, k2. l@k2 parent() ⊗ this@k1 parent()}
     QED
  }
}
```

2 Predicates

$$\text{predicate } \mathit{left}\ (\mathsf{Composite\ ol, int\ lc}) \equiv \mathsf{this.left} \to \mathsf{ol} \otimes \big((\mathsf{ol} \neq \mathsf{null} \multimap \mathsf{ol}@\frac{1}{2}\ \mathit{count}(\mathsf{lc})) \oplus (\mathsf{ol} = \mathsf{null} \multimap \mathsf{lc} = 0)\big)$$

$$\text{predicate } \mathit{right}\ (\mathsf{Composite\ or, int\ rc}) \equiv \mathsf{this.right} \to \mathsf{or} \otimes \big((\mathsf{or} \neq \mathsf{null} \multimap \mathsf{or}@\frac{1}{2}\ \mathit{count}(\mathsf{rc})) \oplus (\mathsf{or} = \mathsf{null} \multimap \mathsf{rc} = 0)\big)$$

$$\text{predicate } \mathit{count}\ (\mathsf{int\ c}) \equiv \exists \mathsf{ol, or, lc, rc.\ this.count} \to \mathsf{c} \otimes \mathsf{c} = \mathsf{lc} + \mathsf{rc} + 1 \otimes \mathsf{this}@\frac{1}{2}\ \mathit{left}(\mathsf{ol, lc}) \otimes \mathit{this}@\frac{1}{2}\ \mathit{right}(\mathsf{or, rc})$$

$$\text{predicate } \mathit{parent}\ () \equiv \exists \text{op, c, k. this.parent} \rightarrow \text{op} \otimes$$
$$\text{op} \neq \text{this} \otimes \text{this}@\frac{1}{2}\ \mathit{count}(\text{c}) \otimes$$
$$\Big((\text{op} \neq \text{null} \multimap \text{op}@\text{k}\ \mathit{parent}() \otimes$$
$$(\text{op}@\frac{1}{2}\ \mathit{left}(\text{this, c}) \oplus \text{op}@\frac{1}{2}\ \mathit{right}(\text{this, c}))\Big)\oplus$$
$$(\text{op} = \text{null} \multimap \mathit{this}@\frac{1}{2}\ \mathit{count}(\text{c}))\Big)$$

3 Code and specifications (minimal proof)

```
class Composite {
  private Composite left , right , parent;
  private int count;

  public Composite
  ⊸ this@½ parent() ⊗ this@½ left(null, 0) ⊗ this@½ right(null, 0)
   {
    this.count = 1;
    this.left = null;
    this.right = null;
    this.parent = null;
    pack this@1 right(null, 0)
    pack this@1 left(null, 0)
    split 1 into two halfs for left, right
    pack this@1 count(0)
    split 1 into two halfs for count
    pack this@1 parent()
    split 1 into halfs
    we only need one half of parent in the post-condition
    { QED }
   }

  private void updateCountRec()
∃ k1, opp, lcc, k, ol, lc, or, rc.
    (unpacked(this@ k1 parent()) ⊗
    this.parent → opp ⊗ opp ≠ this ⊗
    ((opp ≠ null ⊸ opp@k parent() ⊗
    (opp@½ left(this, lcc) ⊕ opp@½ right(this, lcc))) ⊕
    (opp = null ⊸ this@½ count(lcc)) ) ⊗

    unpacked(this@½ count(lcc)) ⊗
```

this.count $\rightarrow$ lcc $\otimes$ lcc = lc + rc + 1 $\otimes$
this@$\frac{1}{2}$ left(ol, lc) $\otimes$ this@$\frac{1}{2}$ right(or, rc)

$\multimap$ $\exists$ k1.this@k1 parent())
{

```
if (this.parent != null)
```

this can be the right child of opp or the left child.
For Boogie we need to consider both paths.
split the fraction k of opp in parent and unpack opp from parent()
unpack opp from $\frac{1}{2}$ count(lccc)
this is either the right or left child of opp. We analyze both cases.
1.In the first case we assume it's the right child.
instantiate orr = this, rrc = lcc; merge both $\frac{1}{2}$ to opp in right
unpack opp from right(this, lcc)
pack this @ $\frac{1}{2}$ count(lcc), add it to the other half
then unpack the count predicate

```
this.updateCount();
```

pack opp in right(this, lcc)

```
this.parent.updateCountRec();
```

pack opp in parent()
pack this in parent(), assuming opp is not null.
It's not since we just called updateCountRec on it, we are on that branch.
{$\exists$ k1. this@k1 parent()}
QED

2.In the second case we assume it's the left child.
instantiate oll = this, llc = lcc; merge both $\frac{1}{2}$ to opp in right
unpack opp from left(this, lcc)
pack this @ $\frac{1}{2}$ count(lcc), add it to the other half
then unpack the count predicate

```
this.updateCount();
```

pack opp in left(this, lcc)

```
this.parent.updateCountRec();
```

pack this in parent(), assuming opp is not null.
It's not since we just called updateCountRec on it, we are on that branch.
{$\exists$ k1. this@k1 parent()}
QED

```
else
```

merge this, $\frac{1}{2}$ in count(lcc) from packed and unpacked

```
this.updateCount();
```

split count in half and pack this in parent
{$\exists$ k1. this@k1 parent()}

QED
}

```
private void updateCount ()
```

$\exists$ c, c1, c2, ol, or. unpacked(this@1 count(c)) $\otimes$

this.count $\to$ c $\otimes$ $c = c1 + c2 + 1$ $\otimes$

this@$\frac{1}{2}$ left(ol, c1) $\otimes$this@$\frac{1}{2}$ right(or, c2)

$\multimap$ $\exists$ c. this@1 count(c)

```
{
  int newc = 1;
```

unpack left, the other case is analogue

```
  if (this.left != null)
```

unpack ol in $\frac{1}{2}$ count(lc)

```
    newc = newc + left.count;
```

pack ol in count(lc)

pack this in left(ol, lc)

merge branches with existential ol and lc

unpack this in $\frac{1}{2}$ right(or, rc)

```
  if (this.right != null)
```

unpack or in $\frac{1}{2}$ count(rc)

```
    newc = newc + right.count;
```

pack or in count

pack this in right

merge branches with existential or and rc

```
  this.count = newc;
```

pack this in count(newc)

QED

}

```
public void setLeft (Composite l)
```

this $\neq$ l $\otimes$

$\exists k1, k2.$(this@k1 parent() $\otimes$ l@k2 parent() $\otimes$

this@$\frac{1}{2}$ left(`null`, 0) $\multimap$

$\exists$ k, k2.this@k parent() $\otimes$ l@k2 parent())

{

unpack l from parent

if op is not null, then we get to a contradiction. So op must be null.

```
  l.parent = this;
```

assignment rule

unpack this from parent

unpack this from $\frac{1}{2}$ count(lcc)

existentialize ol with null and llc with 0 (to unify left permissions)

merge the half fractions to left and unpack this in left(null, 0)

```
  this.left = l;
```

assignment

pack this in left(l, lc)

```
    this.updateCountRec();
    pack l in parent()
    QED
  }
}
```

4 Code and Specifications (no proof)

```
class Composite {
  private Composite left, right, parent;
  private int count;

  public Composite
  ⊸ this@1/2 parent() ⊗ this@1/2 left(null, 0) ⊗ this@1/2 right(null, 0)
   {
    this.count = 1;
    this.left = null;
    this.right = null;
    this.parent = null;
   }

  private void updateCountRec()
 ∃ k1, opp, lcc, k, ol, lc, or, rc.
     (unpacked(this@ k1 parent()) ⊗
     this.parent → opp ⊗ opp ≠ this ⊗
     ((opp ≠ null ⊸ opp@k parent() ⊗
     (opp@1/2 left(this, lcc) ⊕ opp@1/2 right(this, lcc))) ⊕
     (opp = null ⊸ this@1/2 count(lcc)) ) ⊗

     unpacked(this@1/2 count(lcc)) ⊗
     this.count → lcc ⊗ lcc = lc + rc + 1 ⊗
     this@1/2 left(ol, lc) ⊗ this@1/2 right(or, rc)

     ⊸ ∃ k1.this@k1 parent())
   {
    if (this.parent != null)
      this.updateCount();
      this.parent.updateCountRec();
    else
      this.updateCount();
  }

  private void updateCount ()
 ∃ c, c1, c2, ol, or. unpacked(this@1 count(c)) ⊗
     this.count → c ⊗ c = c1 + c2 + 1 ⊗
```

```
    this@½ left(ol, c1) ⊗this@½ right(or, c2)
    ⊸ ∃ c. this@1 count(c)
   {
    int newc = 1;
    if (this.left == null)
    else
      newc = newc + left.count;
    if (this.right == null)
    else
      newc = newc + right.count;
    this.count = newc;
   }

  public void setLeft (Composite l)
   this ≠ l ⊗
        ∃k1, k2.(this@k1 parent() ⊗ l@k2 parent() ⊗
        this@½ left(null, 0) ⊸
        ∃ k, k2.this@k parent() ⊗ l@k2 parent())
   {
    l.parent = this;
    this.left = l;
    this.updateCountRec();
  }
}
```

5 Code

```
class Composite {
  private Composite left, right, parent;
  private int count;

 public Composite()
    {
        this.count = 1;
        this.left = null;
        this.right = null;
        this.right = null;
    }

  private void updateCountRec()
   {
    if (this.parent != null)
      this.updateCount();
      this.parent.updateCountRec();
    else
      this.updateCount();
```

```
  }

  private void updateCount ()
   {
     int newc = 1;
     if (this.left != null)
         newc = newc + left.count;
     if (this.right != null)
         newc = newc + right.count;
     this.count = newc;
   }

  public void setLeft (Composite l)
   {
    l.parent = this;
    this.left = l;
    this.updateCountRec();
  }

  public void setRight (Composite r)
   {
    r.parent = this;
    this.right = r;
    this.updateCountRec();
  }
}
```