1 Concrete Subtyping

The downcasting operation that would allow us to safely use references that expect a smaller heap in a context where a larger heap is available requires some preliminary work.

First of all let us understand what kind of operation we will perform whenever we try to use a heap h that is too large with respect to the get or set functions of the reference:

1. We downcast the larger heap to an adequately smaller one
2. We perform our computation on the downcast heap, thereby obtaining a new smaller heap
3. We store the new smaller heap in the corresponding locations of the original heap

The three steps listed above are summarized in the following diagram:

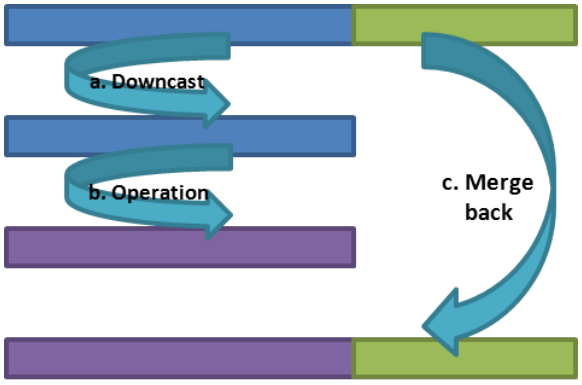

Figure 1: An example of picture.

The downcast operation is implemented easily enough on heaps thanks to the transitivity of the subtyping relationship. We just need to specify that:

New h $\alpha \leq$ h

downcast=**delete**

And the first step of the computation is covered. Now we define the in-place substitution operation with an appropriate predicate:

HList h $\wedge$ HList $h' \wedge h' \leq$ h $\Rightarrow$ InPlaceSubstitute h h'

inPlaceSubstitute : h $\rightarrow h' \rightarrow h'$

This new predicate is instanced inductively on the length of the prefix h:

InPlaceSubstitute Nil h

```
    inPlaceSubstitute Nil h = h
```

and

```
InPlaceSubstitute (h::tl) (h'::tl')
    inPlaceSubstitute (h::tl) (h'::tl') = h::(inPlaceSubstitute tl tl')
```

Thanks to this new operator, which we could consider an upcasting operator of sorts, we can now define the proper downcasting operation for references:

```
HList h, HList h', h ≤ h' ⇒ Reference h' α ≤ Reference α
downcast (Reference get set)=Reference
                    (λh.upcast h (get (downcast h)))
                    (λv.λh.upcast h (set v (downcast h)))
        where upcast h (x,h') = (x, inPlaceSubstitute h' h)
```

This operation invokes the get and set functions of a reference with the downcast (smaller) heap with respect to the input (larger) heap, and then rebuilds a larger heap by stitching together the input heap with the resulting heap from the get or set operation.