

A Heap of Problems

<http://wiki.heap-of-problems.org>

There are a number of good separation logic / shape analysis tools:

- Smallfoot
- SLayer
- Space Invader
- ...

However, full functional verification remains a challenge. Even tiny examples like the reversal of a single linked list, are still not trivial.

A Heap of Problems tries to help answering this challenge by collecting benchmark examples. Besides the problems the proofs done using different tools are published as well. This is done in the hope that this set of examples will simplify communication between different developers and enable them to compare their tools and approaches more easily. *A Heap of Problems* is realised as a Wiki. Everyone is welcome to add examples, proofs, tool-descriptions or just join the discussion.

```
List-Reverse
/* list(i_DATA,i) */
list_reverse(i;) [list(i)] {
  local p, x;
  p = NULL;
  /*exists il_DATA i2_DATA.
  i_DATA = APPEND il_DATA i2_DATA &
  list(i2_DATA,i) * list(reverse(il_DATA),p)*/
  while (i != NULL) [list(i) * list(p)] {
    x = i->tl;
    i->tl = p;
    p = i;
    i = x;
  }
  i = p;
} [list(i)]
/* list(reverse(i_DATA),i) */
```

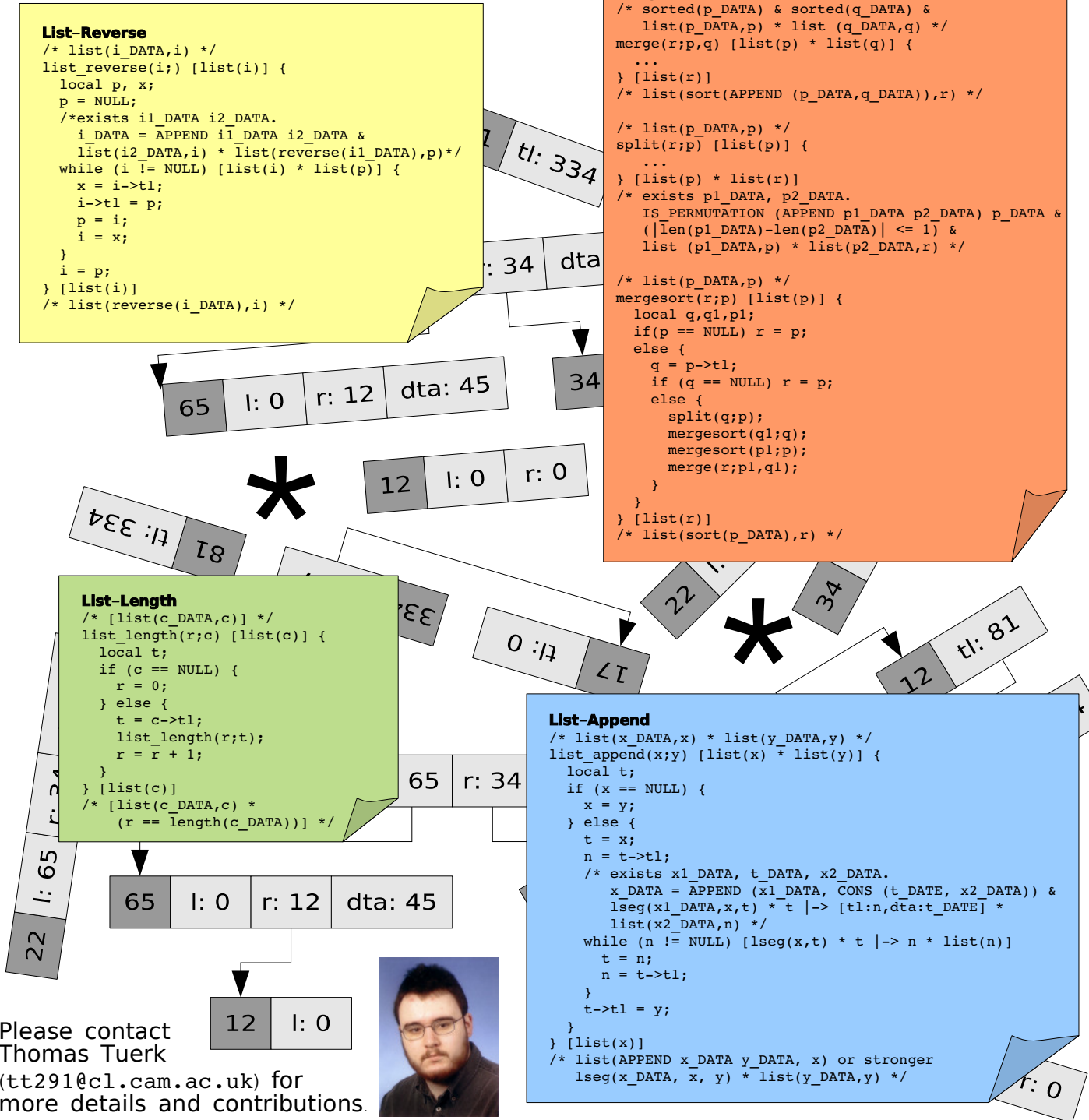
```
Mergesort
/* sorted(p_DATA) & sorted(q_DATA) &
  list(p_DATA,p) * list(q_DATA,q) */
merge(r;p,q) [list(p) * list(q)] {
  ...
} [list(r)]
/* list(sort(APPEND (p_DATA,q_DATA)),r) */

/* list(p_DATA,p) */
split(r;p) [list(p)] {
  ...
} [list(p) * list(r)]
/* exists p1_DATA, p2_DATA.
  IS_PERMUTATION (APPEND p1_DATA p2_DATA) p_DATA &
  (|len(p1_DATA)-len(p2_DATA)| <= 1) &
  list(p1_DATA,p) * list(p2_DATA,r) */

/* list(p_DATA,p) */
mergesort(r;p) [list(p)] {
  local q,q1,p1;
  if(p == NULL) r = p;
  else {
    q = p->tl;
    if (q == NULL) r = p;
    else {
      split(q;p);
      mergesort(q1;q);
      mergesort(p1;p);
      merge(r;p1,q1);
    }
  }
} [list(r)]
/* list(sort(p_DATA),r) */
```

```
List-Length
/* [list(c_DATA,c)] */
list_length(r;c) [list(c)] {
  local t;
  if (c == NULL) {
    r = 0;
  } else {
    t = c->tl;
    list_length(r;t);
    r = r + 1;
  }
} [list(c)]
/* [list(c_DATA,c) *
  (r == length(c_DATA))] */
```

```
List-Append
/* list(x_DATA,x) * list(y_DATA,y) */
list_append(x;y) [list(x) * list(y)] {
  local t;
  if (x == NULL) {
    x = y;
  } else {
    t = x;
    n = t->tl;
    /* exists x1_DATA, t_DATA, x2_DATA.
    x_DATA = APPEND (x1_DATA, CONS (t_DATE, x2_DATA)) &
    lseg(x1_DATA,x,t) * t |-> [tl:n,dta:t_DATE] *
    list(x2_DATA,n) */
    while (n != NULL) [lseg(x,t) * t |-> n * list(n)]
      t = n;
      n = t->tl;
    }
    t->tl = y;
  }
} [list(x)]
/* list(APPEND x_DATA y_DATA, x) or stronger
  lseg(x_DATA, x, y) * list(y_DATA,y) */
```



Please contact
Thomas Tuerk
(tt291@cl.cam.ac.uk) for
more details and contributions.

