

# Verification of Data Layout Transformations

**Ramon Fernández Mir**

with Arthur Charguéraud

Inria

17/09/2018

# Motivating example

```
typedef struct {  
    // Position  
    float x, y, z;  
    // Other fields  
    float vx, vy, vz, c, m, v;  
} particle;  
  
particle data[NUM_PARTICLES];  
  
for (int i = 0; i < NUM_PARTICLES; i++) {  
    // Some calculation  
}
```

# Motivation

- To have an efficient data structure for stacks, queues, dequeues, sequences, catenable sequences, random access sequences, strings
- Both ephemeral and persistent versions with full persistence
- Convert between ephemeral and persistent versions in constant time

# Demo Coq

```
Lemma foo :  $\forall x, \exists y, x = y$ .
```

```
Proof using.
```

```
  foo_solved.
```

```
Qed.
```

```
Lemma bar :  $\forall x, \exists y, x = y$ .
```

```
Proof using.
```

```
  bar_solved.
```

```
Qed.
```

## Related Work

| Structure    | Memory        | Time        | Limitations  |
|--------------|---------------|-------------|--|
| Arrays       | $1\times$     | $1\times$   | concat/split/resize: $O(n)$                                |
| Vectors      | $2 - 4\times$ | $2\times$   | concat/split: $O(n)$                                       |
| Lists        | $3\times$     | $3\times$   | concat/split/random access $O(n)$                          |
| Finger trees | $> 3\times$   | $> 3\times$ | Not transient  |
| Ropes        | ?             | ?           | More complex access to ends,<br>not automatically balanced |
| Chunked Seq  | $< 1.2\times$ | $< 2\times$ |  |

# Interface

Chunks: fixed capacity arrays in which elements are stored  
 $K$  = size of chunks

| Operation                          | Ephemeral                     | Persistent                    |
|------------------------------------|-------------------------------|-------------------------------|
| push/pop/front/back                | $O(1 + \frac{1}{K} \log_K n)$ | $O(K + \frac{1}{K} \log_K n)$ |
| usual case                         | $O(1)$                        | $O(1)$                        |
| concat/split/get/set               | $O(K \log_K n)$               | $O(K \log_K n)$               |
| iter/fold/...                      | $O(n)$                        | $O(n)$                        |
| Ephemeral $\rightarrow$ Persistent |                               |                               |
| destructive                        | $O(1)$                        |                               |
| nondestructive                     | $O(K)$                        |                               |
| Persistent $\rightarrow$ Ephemeral | $O(K)$                        |                               |

# Tree Structure

[image]

# Sequence Representation - persistent

## Pchunk:

Fixed capacity persistent sequence

Implemented using a view on a shared “support” chunk

```
type 'a pchunk = {  
    support : 'a chunk;  
    mutable view : segment; }
```

```
type segment = int * int
```

The shared chunk is reusable when popping or when pushing past its bounds. Other push cases need copy-on-write.

Pops are always  $O(1)$ , pushes are in amortized  $O(1)$  if iterated.

```
type 'a pseq =  
| Empty  
| Struct of 'a pchunk * ('a pchunk) seq * 'a pchunk
```



# Versions

**Goal:** Work on pchunks with in-place updates in ephemeral sequences

**Solution:** Maintain whether a chunk is shared or uniquely possessed in ephemeral sequences

**Invariants:**

Persistent: all chunks are marked false

Ephemeral: some are false and shared, some are true and were created in this sequence

Ephemeral  $\rightarrow$  persistent = mark all chunks back to false

Version number trick enables this to be done in constant time.

```
type 'a pchunk = {  
  version : version;  
  support : 'a chunk;  
  mutable view : segment; }
```

# Transient Sequences - Types

```
type 'a seq = {  
  mutable version : version;  
  mutable front : 'a chunk;  
  mutable middle : ('a pchunk) pseq;  
  mutable back : 'a chunk;  
}
```

```
type 'a pseq =  
| Empty of 'a  
| Struct of 'a pchunk * ('a pchunk) pseq * 'a pchunk
```

Note: Persistent sequence version number is stored in back chunk.

# Summary and Additional Fields

```
type 'a chunk = {  
  mutable head : int;  
  mutable size : int;  
  mutable data : 'a array;  
  default : 'a; }
```

```
type 'a pchunk = {  
  version : version;  
  support : 'a chunk;  
  mutable view : segment;  
  mutable weight : weight; }
```

```
type 'a seq = {  
  mutable version : version;  
  mutable front : 'a chunk;  
  mutable free_front : ('a chunk) option;  
  mutable middle : ('a pchunk) PWSeq.t;  
  mutable free_back : ('a chunk) option;  
  mutable back : 'a chunk;  
}
```

```
type 'a pseq =  
  | Empty of 'a  
  | Struct of weight * 'a pchunk * ('a pchunk) t * 'a pchunk
```