Flexichain An editable sequence and its gap-buffer implementation

Robert Strandh, Tim Moore, Matthieu Villeneuve

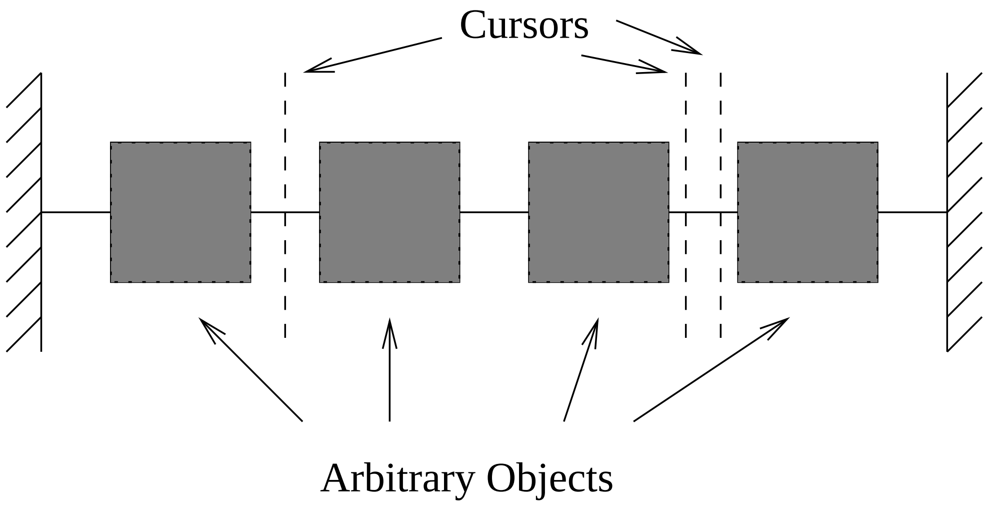
strandh@labri.fr, moore@labri.fr, matthieu.villeneuve@free.fr

Laboratoire Bordelais de Recherche en Informatique

(LaBRI)

API Model

An editable sequence of objects containing an arbitrary number of cursors.



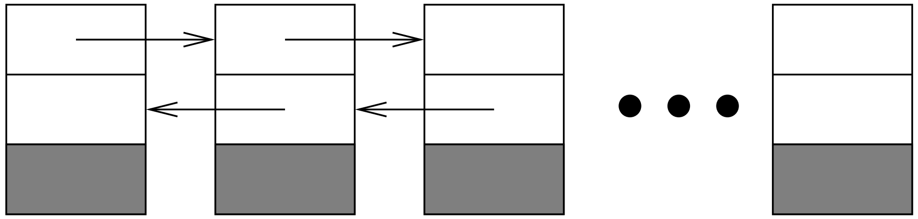
Uses of API

- Emacs-like text editors for the entire buffer
- Emacs-like text editors for one line of a buffer
- Emacs-like text editors for the sequence of lines
- Gsharp (at 4 different levels)

Possible representations

- Doubly linked list
- Gap buffer

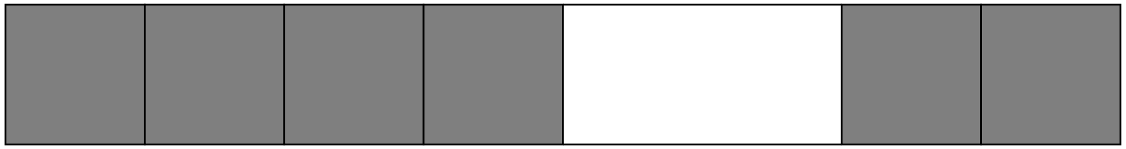
Doubly linked list



Advantages: fast, easy to implement

Inconveniences: no direct access, high memory overhead

Gap buffer



Advantages: direct access, low memory overhead

Inconveniences: slow worst-case, hard to implement

Previous work

- Multics Emacs: a doubly-linked list of lines, each line a vector (special instructions)
- GNU Emacs: entire buffer is a gap buffer
- Hemlock: doubly-linked list of lines, the open line is a gap buffer
- Goatee (McCLIM): doubly-linked list of lines, each line is a gap buffer
- Gsharp: currently uses singly-linked lists

Purpose of Flexichain

In order of priorities:

1. Replace specialized, slow code in Gsharp
2. Replace line implementation in Goatee
3. Replace list of lines in Goatee
4. Perhaps use in portable Hemlock

Two layers: Flexichain and cursorchain

The *Flexichain* layer uses *positions* to access, insert, and delete elements.

The *Cursorchain* layer uses *cursors*.

The two are compatible.

The Flexichain layer

```
insert<* chain element position
insert>* chain element position
delete* chain position
element* chain position
(setf element*) position element chain
```

Stack and queue operations

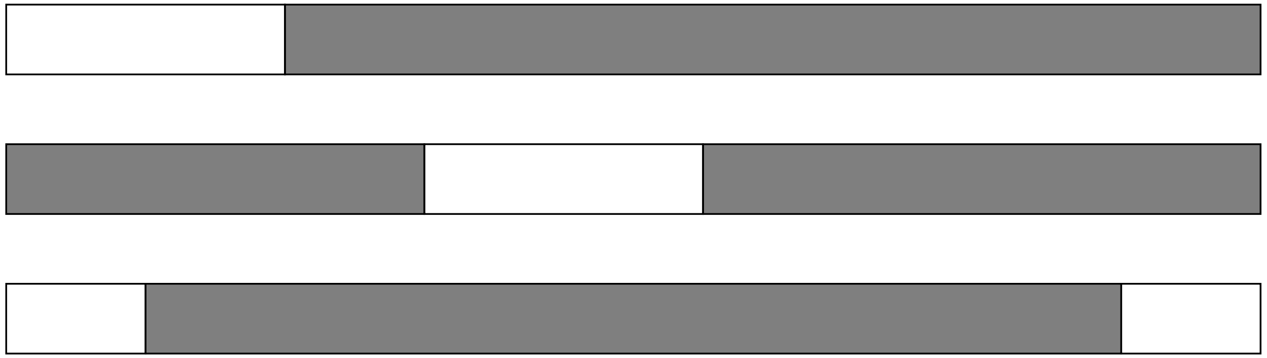
```
push-start chain element
push-end chain element
pop-start chain
pop-end chain
rotate chain &optional (n 1)
```

Implementation of Flexichain layer

- Use a gap buffer
- Consider the buffer as circular (avoids bad worst cases)
- Expand and shrink factors
- Moving the gap is messy

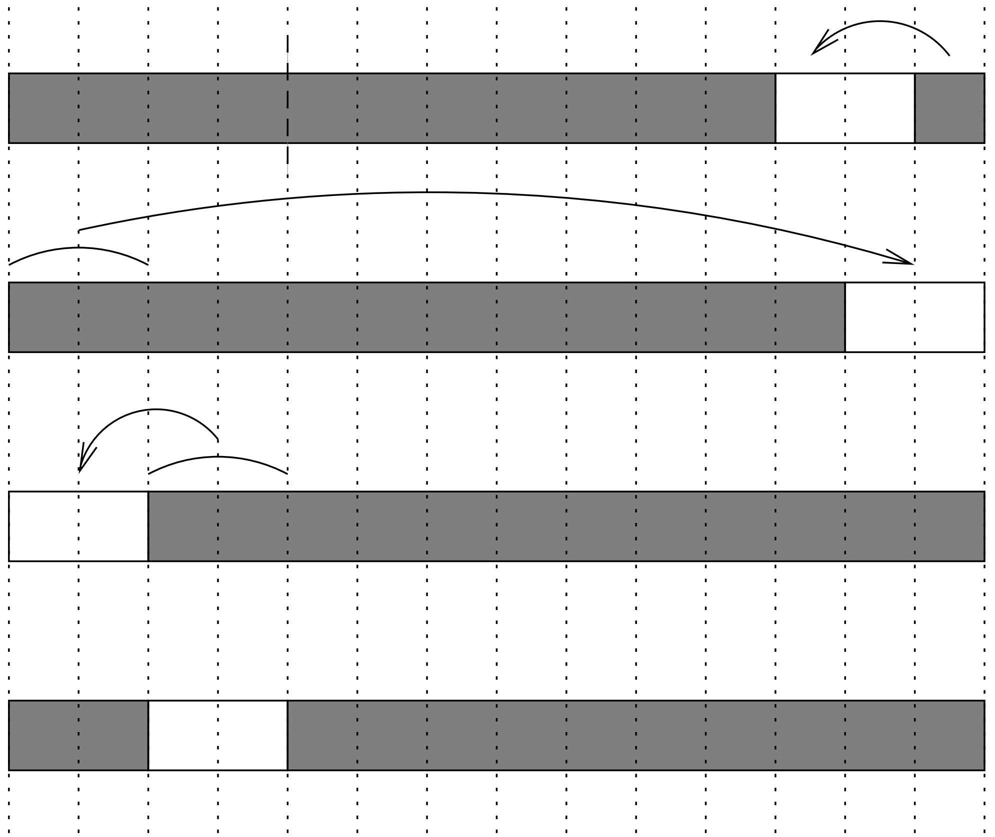
Moving the gap

Three situations exist:



Move left or right according to the number of elements that need to be moved in each case.

Moving the gap



The Cursorchain layer

```
at-beginning-p cursor
at-end-p cursor
move< cursor
move> cursor
insert< object cursor
insert> object cursor
delete< cursor &optional (n 1)
delete> cursor &optional (n 1)
element< cursor
element> cursor
```

The Cursorchain layer

- `cursorchain` is a subclass of `flexichain`
- cursors store physical position rather than logical positions in order to avoid updating all cursors at every operation
- to avoid memory leaks, we use weak references to store cursors

The Cursorchain layer

- we use an internal protocol for resizing and for moving the gap
- cursor update are done by `:before` and `:after` methods of internal protocol

State of implementation

Flexichain is finished.

Cursorchain will be finished this summer (we all have daytime jobs).

That’s all

Thank You!

Questions?