§ 0
Programming is often regarded as problem solving requiring (unstructured) ingenious inventions. At the other end of the spectrum are those that advocate algorithm construction by formal manipulation only, such that machines can do the job.
Although the middle of the road is seldom paved with yellow bricks, we like to lend good things from both. Inventions? Yes, but they should be structured and triggered by the formalism, hence tamed to heuristics. Formal manipulation? Yes, but driven by the heuristics. So we regard programming as a human activity benefitting from suitable modelling and calculation.
To that end one should develop a formalism that reflects the properties of the objects under consideration. It should be concise, with limited manipulative choice. The development strategy should be guided by the derivation techniques, manipulative possibilities and by educated guesses about what may be accomplished. The details are to be fully calculational, and representation should enter the picture at the very last stage.
We shall demonstrate this by a problem thrown at the audience in Marktoberdorf by R.S. Bird. (We return to that in the last section.)
The title above may be misleading: we don't kill rabbits, we try to avoid them. We confess to like rabbits a lot, provided ... they are welltrained enough to pop up at the appropriate stages in the derivations. Beauty loses its splendour if it is all over the place. Therefore we plea for rabbit anti-conception.

§ 1. Example

The problem we use to illustrate some derivation principles with is called:
"the largest rectangle under a histogram"
There are several ways to state it, depending on the chosen formalism and point of view (cf [B], [E]):
Fractions

- transform the following function on lists of naturals

\[ \frac{1}{\text{area}} \ast \frac{\text{segs}}{x} \text{ where } \text{area } x = (\#x) \times \frac{1}{x} \]

to an expression that can be computed in linear time.

- find a linear algorithm that establishes

\[ b = (\max p, q : 0 \leq p \leq q \leq N : (q - p) \ast (\min i : p \leq i < q : X \cdot i)) \]

where \( X \) is an array of naturals.

Instead of introducing heaps (cf [B]) or performing a lot of quantifier juggling (cf [E]), we rephrase the problem in terms of the "relevant concepts": histograms and rectangles.

Let \( I \) be a finite segment of the integers.

A histogram on domain \( I \) is a natural function on \( I \). Histograms on \( I \) are ordered pointwise, i.e.

\[ (0) \quad X \leq Y \equiv (\_ \_ \_ \_ i : i \in I : X \cdot i \leq Y \cdot i) \]

Unless the domain of a histogram is understood, we provide it: \( X[m, n) \) stands for histogram \( X \) on the finite segment \( \{ i \in \text{int} \mid m \leq i < n \} \).

A constant histogram with value \( c \) is denoted by the value in boldface, e.g. \( c[m, n) \).

Histograms can be catenated or restricted:

Let \( X[m, q) \) and \( Y[q, r) \) be histograms and \( m \leq q \leq r \), then

\( X[m, q) \ast Y[q, r) \) is a histogram on \( [m, r) \) (catenation)

for \( m \leq n \leq p \leq r \), \( X[n, p) \) is a histogram (restriction of \( X \) to \( [n, p) \)).

A rectangle on \( [m, q) \) is a histogram of the form (for some \( c \) in \text{nat})

\[ 0[m, n) \ast c[n, p) \ast 0[p, q). \]

\( X \) being a rectangle is expressed by: \( \text{rect} \cdot X \).

Our specification of the problem:

```
[:] N : nat ; X : histogram on [0, N) ;
[: b : nat ;
  S
  { b = MRA \cdot X }
]```