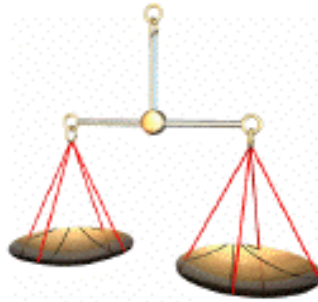


## A trade-off for deterministic dictionaries



Rasmus Pagh



Department of Computer Science  
University of Aarhus

SWAT 2000, Bergen, July 5, 2000

## The dictionary problem

Store a set  $S \subseteq U$  under the following operations:

- Insertion of an element, with satellite information.
- [Deletion of an element.]
- Query for membership, and retrieval of any satellite information.

Complexity expressed in terms of  $n = |S|$ .

### Model of computation:

- Unit cost RAM with word size  $w$  and a standard instruction set.
- $U = \{0, 1\}^w$ .
- Free access to  $O(1)$  words computed at “compile time”.

Space constraint: Use  $O(n)$  words.

## Great expectations

Using *randomization*, very good expected bounds can be achieved.

There is a dictionary with:

- worst-case *constant* query time, and
- worst-case *constant* update time with probability  $1 - 1/n^{\omega(1)}$ .

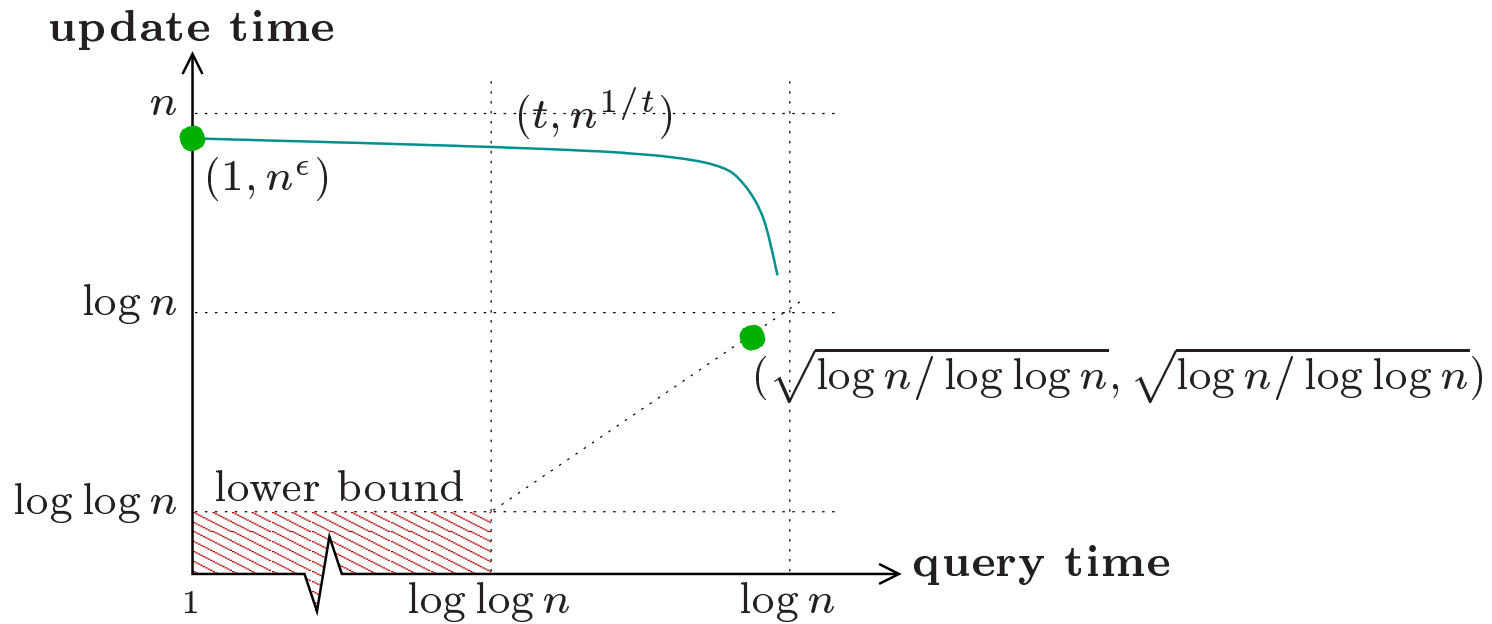
Shown in the literature for  $w = O(\log n)$  [Willard '00].

Can be extended using [Dietzfelbinger & Meyer auf der Heide '90].

**Main question:** What can be done

- if there is no source of random bits, or
- if a small chance of failure is unacceptable?

# Previous deterministic dictionaries



... and the result of this talk.



**Query time:**  $O((\log \log n)^2 / \log \log \log n)$ .

**Update time:**  $O(\log^2 n)$ . [Amortized – can be made worst-case]

# The ingredients

## 1. Very fast (vEB tree-descendant) dictionary for short words

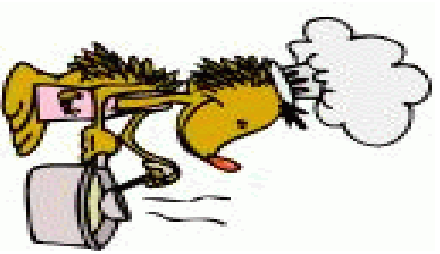
- Time/operation  $O((\log \log n)^2 / \log \log \log n)$ , for  $w = (\log n)^{O(1)}$ .  
[Beame & Fich '99] + [Andersson & Thorup '00]

## 2. Universe reduction tools

- A function  $h : \{0, 1\}^w \rightarrow \{0, 1\}^{O(\log n)}$  that is 1-to-1 on  $S$  and constant-time evaluable, “translates” the problem to a smaller universe.

Such a function can be found in time  $O(n \log n)$ .

[Miltersen '98] + [Hagerup '99] + [Pagh '00]



- Supports predecessor (and longest common prefix) queries.
- If words are not short, many  $h$ 's can be evaluated simultaneously in constant time.

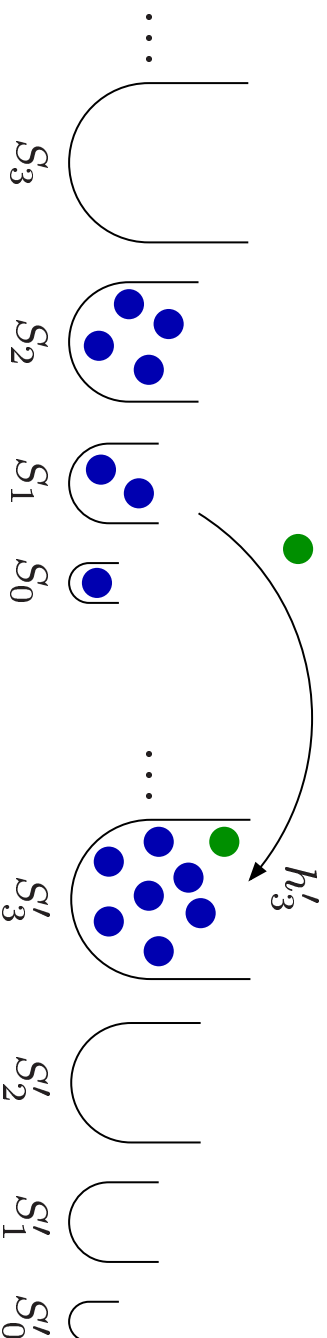
# Dynamic universe reduction

**Idea:** Map to strings over  $\Sigma = \{0, 1\}^{O(\log n)}$  of length  $l = O(\log n)$ ,

$$h : x \mapsto h_{l-1}(x) h_{l-2}(x) \dots h_0(x)$$

where  $h_i$  is 1-to-1 on  $S_i$ ,  $S = \cup_i S_i$ , and  $|S_i| \in \{0, 2^i\}$ .

Maintaining  $h$  in  $O(\log^2 n)$  amortized time/insertion:



**Problem 1:** How do we evaluate  $h$  quickly?

**Problem 2:**  $h(S)$  is “highly non-constant” over time.

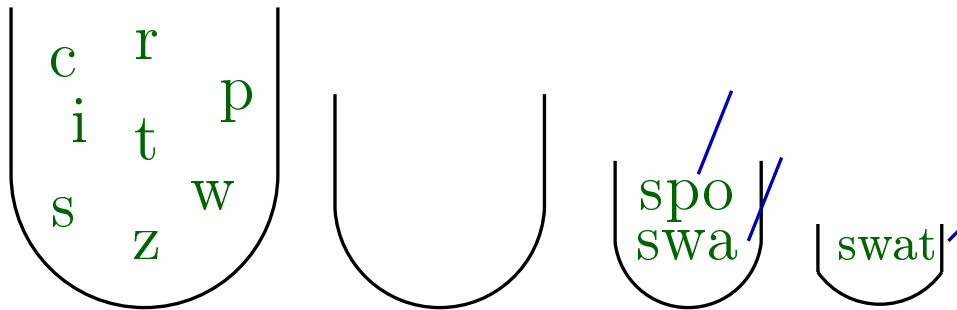
## Dealing with changes – the idea

The key: For  $x \in S_i$ , use the key

$$k_x = h_{l-1}(x) h_{l-2}(x) \dots h_i(x)$$

The key is fixed, unique to  $x$ , and a prefix of  $h(x)$ .

Example:



Query for  $x$ :

- Search for  $k$  that is prefix of  $h(x)$  (if none exists, then  $x \notin S$ ).

$$\begin{array}{l} \{x_s, x_{\text{spo}}\} \\ \{x_s, x_{\text{swa}}\} \\ \{x_s, x_{\text{swa}}, x_{\text{swat}}\} \end{array}$$

the longest

- Search for  $x$  in  $k$ 's attached list,  $\{x \in S \mid k_x \text{ is a prefix of } k\}$ .

## Wrapping up

### The actual details ...

- Predecessor (binary longest common prefix) data structure.
- Maintaining the associated lists at no additional overhead.
- Linear space?

### Open questions:

- **Can Sundar's lower bound be matched (up to a polynomial)?**
  - Using dynamic universe reduction with exponential search trees is a promising approach.
  - Near-linear-time computation of reduction function wanted!
- Is there a “smoother” range of trade-offs than presently known?