Technische Universität München
Institut für Informatik
Prof. Tobias Nipkow, Ph.D.
Fabian Huch

SS 2021
2. 8. 2021

# Functional Data Structures

Exercise Sheet 12

## **Exercise 12.1** Sparse Binary Numbers

Implement operations carry, inc, and add on sparse binary numbers, analogously to the operations link, ins, and merge on binomial heaps.

Show that the operations have logarithmic worst-case complexity.

**type_synonym** *rank* = *nat*
**type_synonym** *snat* = *"rank list"*

**abbreviation** *invar* :: *"snat ⇒ bool"* **where** *"invar s ≡ sorted_wrt (<) s"*
**definition** *α* :: *"snat ⇒ nat"* **where** *"α s = sum_list (map ((^) 2) s)"*

**lemmas** [*simp*] = *sorted_wrt_append*

**fun** *carry* :: *"rank ⇒ snat ⇒ snat"*

**lemma** *carry_invar*[*simp*]:
  **assumes** *"invar rs"*
  **shows** *"invar (carry r rs)"*

**lemma** *carry_α*:
  **assumes** *"invar rs"*
    **and** *"∀ r'∈set rs. r≤r'"*
  **shows** *"α (carry r rs) = 2^r + α rs"*

**definition** *inc* :: *"snat ⇒ snat"*

**lemma** *inc_invar*[*simp*]: *"invar rs ⟹ invar (inc rs)"*

**lemma** *inc_α*[*simp*]: *"invar rs ⟹ α (inc rs) = Suc (α rs)"*

**fun** *add* :: *"snat ⇒ snat ⇒ snat"*

**lemma** *add_invar*[*simp*]:
  **assumes** *"invar $rs_1$"*
    **and** *"invar $rs_2$"*
  **shows** *"invar (add $rs_1$ $rs_2$)"*

```
lemma add_α[simp]:
  assumes "invar rs₁"
      and "invar rs₂"
  shows "α (add rs₁ rs₂) = α rs₁ + α rs₂"

thm sorted_wrt_less_sum_mono_lowerbound

lemma size_snat:
  assumes "invar rs"
  shows "2^length rs ≤ α rs + 1"

fun T_carry :: "rank ⇒ snat ⇒ nat"

definition T_inc :: "snat ⇒ nat"

lemma T_inc_bound:
  assumes "invar rs"
  shows "T_inc rs ≤ log 2 (α rs + 1) + 1"

fun T_add :: "snat ⇒ snat ⇒ nat"

lemma T_add_bound:
  fixes rs₁ rs₂
  defines "n₁ ≡ α rs₁"
  defines "n₂ ≡ α rs₂"
  assumes INVARS: "invar rs₁" "invar rs₂"
  shows "T_add rs₁ rs₂ ≤ 4*log 2 (n₁ + n₂ + 1) + 2"
```

## Homework 12 Explicit Priorities

*Submission until Thursday, July 8, 23:59pm.*

Modify the priority queue interface to handle multisets of pairs of data and priority, i.e., the new *mset* function has the signature $mset::'q \Rightarrow ('d \times 'a::linorder)\ multiset$.

Next, implement the new interface using leftist heaps. Implement them with *rank* instead of *min_height*!

Hints:

- Start with content from the existing theories (*HOL−Data_Structures.Priority_Queue_Specs* and *HOL−Data_Structures.Leftist_Heap*), and modify them!
- Be careful to design a good specification for *get_min*!

## **Homework 12.1** Be Original!

*Submission until Thursday, July 8, 23:59pm.*

Develop a nice Isabelle formalisation yourself!

We will reduce regular homework load (the sheets are half the size/points), such that you have a time-frame of 3 weeks with reduced regular homework load. You should have finished your formalization until next week. Submit it either via the Submission system (if it's a single theory file – import *Defs* as first import!) or via email to the tutor.

- The homework will yield 15 points (for minimal solutions). Additionally, up to 15 bonus points may be awarded for particularly nice/original/etc solutions.
- You may develop a formalisation from all areas, not only functional data structures.
- Document your solution, such that it is clear what you have formalised and what your main theorems state!
- Set yourself a time frame and some intermediate/minimal goals. Your formalisation needs not be universal and complete after 3 weeks.
- Should you still need inspiration to find a project: Sparse matrices, skew binary numbers, arbitrary precision arithmetic (on lists of bits), interval data structures (e.g. interval lists), spatial data structures (quad-trees, oct-trees), Fibonacci heaps, prefix tries/arrays and BWT, etc. You can also ask the tutor for possible ideas, and you are encouraged to discuss the realisability of your project with us!