Exercise 11.1 Sparse Binary Numbers

Implement operations carry, inc, and add on sparse binary numbers, analogously to the operations link, ins, and meld 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 ≡ strictly_ascending s”
definition α :: “snat ⇒ nat” where “α s = (∑ i← s. 2ˆi)”

fun carry :: “rank ⇒ snat ⇒ snat”
lemma carry_invar[simp]:
lemma carry_α:

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”
  assumes “invar rs_2”
  shows “invar (add rs_1 rs_2)”
lemma add_α[simp]:
  assumes “invar rs_1”
  assumes “invar rs_2”
  shows “α (add rs_1 rs_2) = α rs_1 + α rs_2”

lemma size_snat:
  assumes “invar rs”
  shows “2ˆlength 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 \( t_{\text{add\_bound}} \):  
  fixes \( rs_1 \) \( rs_2 \)  
  defines \( n_1 \equiv \alpha \) \( rs_1 \)  
  defines \( n_2 \equiv \alpha \) \( rs_2 \)  
  assumes INVARS: \("\text{invar} \) \( rs_1 \) \" \(" \text{invar} \) \( rs_2 \) \"  
  shows \(" t_{\text{add} \ rs_1 \ rs_2} \leq 4 * \log 2 \ (n_1 + n_2 + 1) + 2 \)\"

Homework 11.1 Largest Representable Number

Submission until Friday, 14. 7. 2017, 11:59am.

Assume we use numbers \( \{0..<K\} \) to represent the ranks in a sparse binary number. Define \( \text{max\_snat} 
K \) to be the largest representable sparse binary number (its value should be \( 2^K - 1 \)), and prove that your definition is correct.

definition \( \text{max\_snat} :: \"\text{nat} \Rightarrow \text{snat} \)\"

lemma \"\text{invar} \ (\text{max\_snat} 
K) \"  
lemma \( \alpha_{\text{max\_snat}} \:: \" \alpha \ (\text{max\_snat} 
K) = 2^K - 1 \"  
lemma \( \text{max\_snat\_bounded} \:: \" \text{set} \ (\text{max\_snat} 
K) \subseteq \{0..<K\} \"  
lemma \( \text{max\_snat\_max} \):
  assumes \"\text{invar} \) \( rs \) \"  
  assumes \"\text{set} \) \( rs \subseteq \{0..<K\} \"  
  shows \" \alpha \) \( rs \leq \alpha \ (\text{max\_snat} 
K) \"\"

Homework 11.2 Be Original!

Submission until Friday, 28. 7. 2017, 11:59am.

Develop a nice Isabelle formalization yourself!

- This homework is for 3 weeks, and will yield 15 points + 15 bonus points.
- You may develop a formalization from all areas, not only functional data structures.
- Set yourself a time frame and some intermediate/minimal goals. Your formalization needs not be universal and complete after 3 weeks.
- You are welcome to discuss the realizability of your project with the tutor!
- In case you should 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, etc.