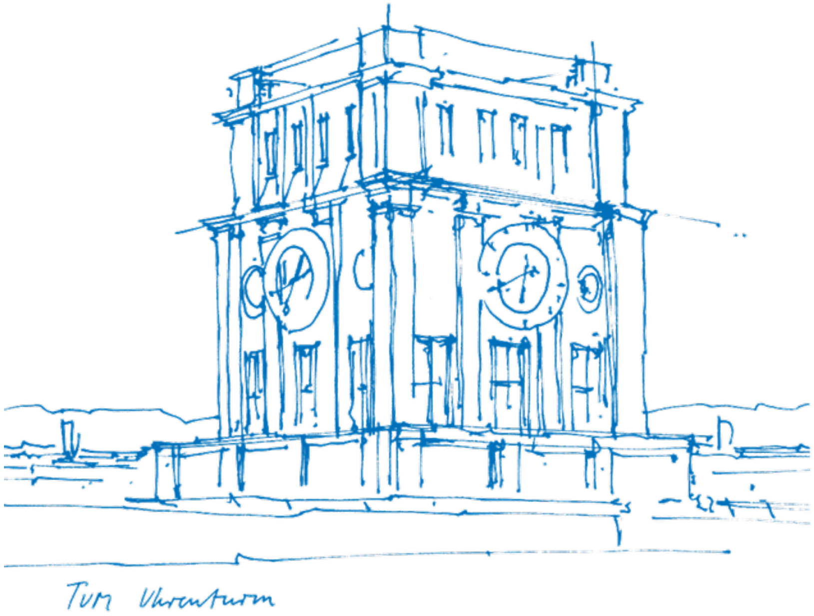
Amortized Time Complexity of Union-Find in Isabelle/HOL

Colloquium to the Bachelor's Thesis

Adrián Löwenberg Casas
Technische Universität München
Department of Informatics
Chair for Logic and Verification
Supervisor: Prof. Tobias Nipkow, Ph.D.
Advisor: Maximilian P.L. Haslbeck, M.Sc.
Munich, October 16, 2019



Overview

1. Union-Find
1.1 Path Compression and Union by Rank
1.2 Design Choices
2. History of the Proof
3. Timeline of the Project
4. Overview of the Isabelle Proof
5. Inverses of Functions $f : \mathbb{N} \to \mathbb{N}$
6. The Ackermann Function and its Inverse
7. Separation Logic with Time Credits
7.1 Amortized Analysis with Time Credits
8. The Potential Function Φ
9. Results of the Thesis
10. Conclusions

Union-Find

- Models a partial equivalence relation over a finite domain
- Implemented by disjoint set forests
- The graph structure is represented by an array
- Supports **Union** and **Find** operations

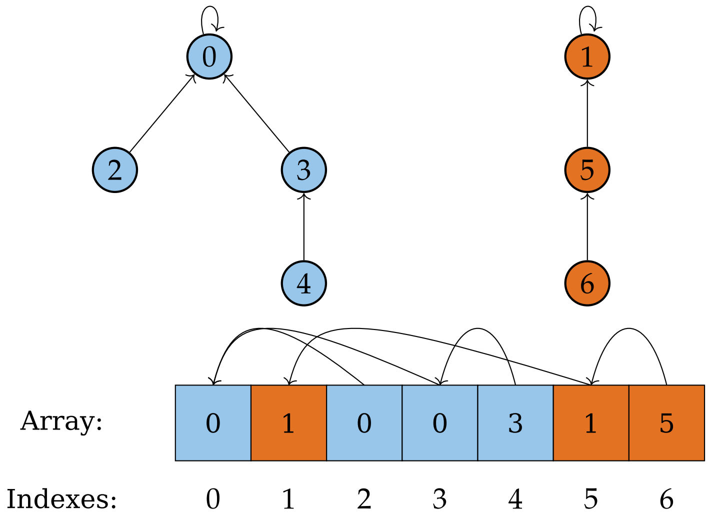


Figure: Two equivalence classes

Path Compression and Union by Rank

- As with many tree-based data structures, trees should be kept flat
- This is done in two ways:
  1. **Path Compression**
  2. **Union by Rank**

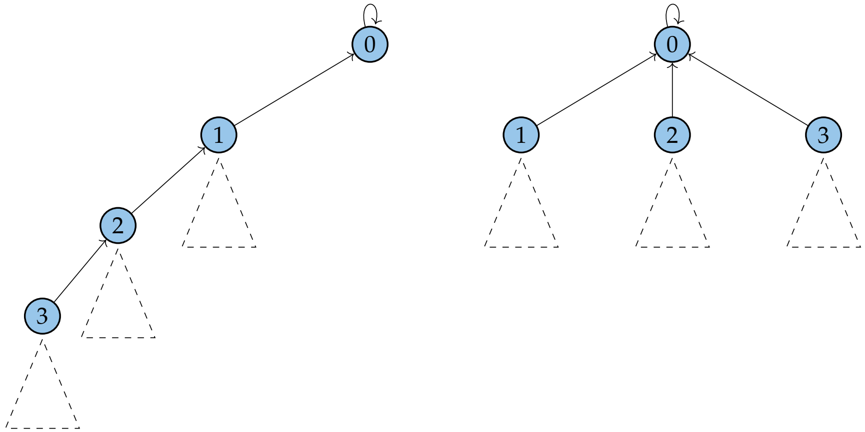


Figure: Illustration of Path Compression

Design Choices

- Done in Imperative/HOL, which can then be exported to several languages
- The arrays used and therefore the domain are static
- We define the operations in Imperative/HOL:
  - **partial_function** (heap) uf_rep_of :: nat array ⇒ nat ⇒ nat Heap
  - **partial_function** (heap) uf_compress :: nat ⇒ nat ⇒ nat array ⇒ unit Heap
  - **definition** uf_rep_of_c :: nat array ⇒ nat ⇒ nat Heap
  - **definition** uf_cmp :: uf ⇒ nat ⇒ nat ⇒ bool Heap

```
definition uf_union :: uf ⇒ nat ⇒ nat ⇒ uf Heap where
uf_union u i j ≡ do {
      let (r,p) = u;
      ci ← uf_rep_of_c p i;        ci ← uf_rep_of p i;
      cj ← uf_rep_of_c p j;        cj ← uf_rep_of p j;
                  .
                  .
                  .
```

  -

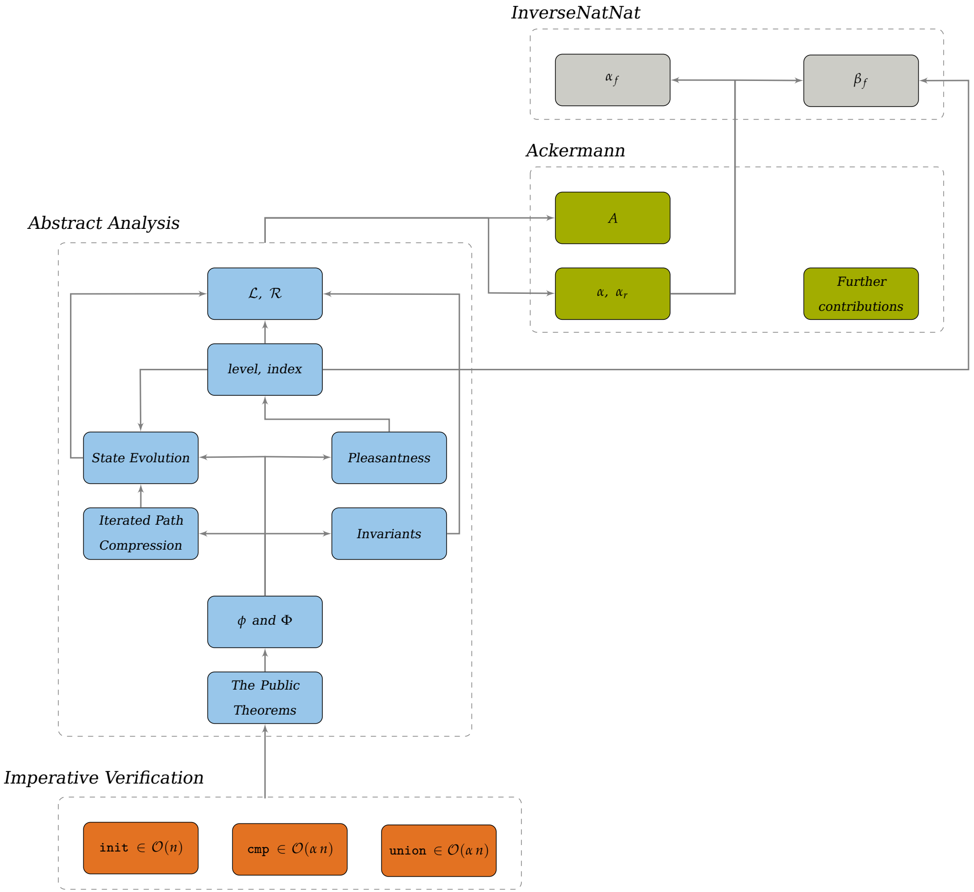
History of the Proof

- Several, slightly different results about the amortized time complexity of the operations of Union-Find
- The first result involving α by Tarjan in 1975. The proof was simplified until the version in CLRS
- In 1989 Fredman and Saks prove (in some sense) the optimality of the result
- The latest result by Alstrup et al. in 2014 tightens the bound
- In 2017, Charguéraud and Pottier formalize the result in Coq
- Now, we present this formalization in Isabelle/HOL
- In all these proofs, but especially since Alstrup et al., most work is needed for the abstract analysis

Timeline of the Project

- **May** — Background work: Familiarisation with Sepreftime and the paper by Charguéraud and Pottier.
- **May 15th** — Official thesis registration.
- **June** — Cleanup of the previous work by Lammich and Haslbeck.
- **June 5th** — Armaël Guéneau visits TUM.
- **June** — **Bottom-up**: Ackermann and InverseNatNat theories and important definitions.
- **July** — First abstract proofs: rank, level, index.
- **July 16th** — **Top-down**: Sanity check of definitions done, skipping the rest and proof of the Hoare-Triples.
- **August** — Exam phase.
- **September** — Parallel writing of the thesis, proof gap-filling and correction of definitions.
- **September 8th** — Complete sound theory.
- **Semptember 13th** — Submission of the thesis.

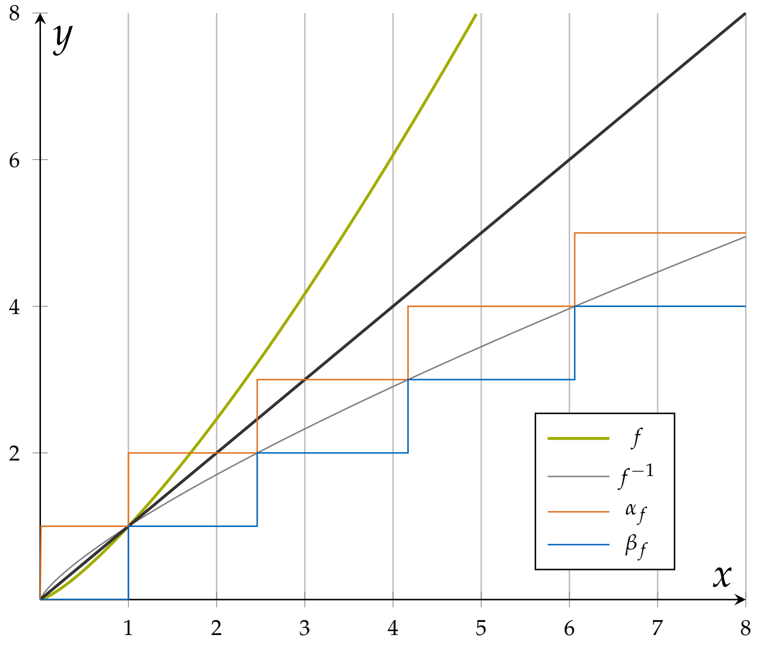
Overview



InverseNatNat
α_f
β_f
Ackermann
A
$\alpha,\ \alpha_r$
Further contributions
Abstract Analysis
$\mathcal{L},\ \mathcal{R}$
level, index
State Evolution
Pleasantness
Iterated Path Compression
Invariants
ϕ and Φ
The Public Theorems
Imperative Verification
init $\in \mathcal{O}(n)$
cmp $\in \mathcal{O}(\alpha\, n)$
union $\in \mathcal{O}(\alpha\, n)$

Inverses of Functions $f : \mathbb{N} \to \mathbb{N}$

- Defined in InverseNatNat.thy
- Upper inverse α_f and Lower inverse β_f
- Lemmas to change the proof obligations between inverses and the original function
- Used for Inverse Ackermann Function, but also the “index” of a node



The Ackermann Function and its Inverse

- There is no single Ackermann Function "A" or inverse Ackermann Function "α"
- All definitions share the property of growing faster than any primitive recursive function
- α grows **very** slowly
- We use the definition of A and α by Tarjan:

Definition
Ackermann Function

$$A\,0\,x = x + 1$$
$$A\,(k+1)\,x = (A\,k)^{(x+1)}\,x$$

Definition
Inverse Ackermann Function

$$\alpha\,n = \min\{k \mid A\,k\,1 \geq n\}$$
$$\alpha_r\,n = 1 + \min\{k \mid A\,k\,r \geq (n+1)\}$$

- In Isabelle/HOL, the definitions are expressed slightly differently and make use of InverseNatNat.thy

The Ackermann Function and its Inverse

Lemma

observable_universe_α

Assume: $n \leq 10^{80}$

$$\alpha\, n \leq 4 \tag{1}$$

- 10^{80} is an estimate of the number of atoms in the universe

Lemma

α_n_0_α_logn

Assume: $16 \leq n$

$$\alpha\, n \leq 1 + (\alpha\, (\log n)) \tag{2}$$

- α and $\alpha(\log n)$ are asymptotically equivalent
- True for every primitive recursive strictly monotonic function

Separation Logic with Time Credits

- Logic to reason about mutable resources in a heap. Enables Hoare-Triple definition
- It also allows for “pure” assertions, predicates independent of the heap content
- Time Credits can also appear in assertions and are required to execute atomic operations
- The components of the logic are:
  - $\uparrow(P)$
  - true and false
  - $p \mapsto_a xs$
  - $P_1 * P_2$
  - $\exists_A x.\, P$
- The Framework by Haslbeck implementing this for Isabelle/HOL, as well as this thesis as a usage example will be published in the AFP.

Amortized Analysis with Time Credits

- We define a potential function Φ that measures the “entropy” of the data structure
- The idea is to “hide” Φ credits in the assertion defining the data structure
- If we are able to prove a Hoare-Triple of the form:

$$\langle \text{invar}(\mathcal{D}) * \$(\Phi(\mathcal{D})) * \underbrace{\$(f(\mathcal{D}))}_{\text{Advertised Cost}} \rangle \quad \texttt{op}(\mathcal{D}) \quad \langle \text{invar}(\mathcal{D}') * \$(\Phi(\mathcal{D}')) \rangle$$

  for any Φ and $f \in \mathcal{O}(g)$, we can conclude that the operation `op` has an amortized cost in $\mathcal{O}(g)$
- Remaining questions:
  - What Φ do you choose? How does Φ evolve?
  - How do you prove the functional correctness?

    $\Longrightarrow$ Mathematical analysis of the data structure

The Potential Function Φ

Definition

Potential for a single node

$$\phi\,\mathcal{L}\,\mathcal{R}\,i := \begin{cases} \alpha_r\,(\mathcal{R}_r\,i)\cdot(1+(\mathcal{R}_r\,i)) & \text{if}\,\mathcal{L}!i = i \\ (\alpha_r\,(\mathcal{R}_r\,i) - \text{level}_{\mathcal{L},\mathcal{R}}\,i)\cdot\mathcal{R}_r\,i - \text{index}_{\mathcal{L},\mathcal{R}}\,i + 1 & \text{if}\,\alpha_r\,(\mathcal{R}_r\,i) = \alpha_r\,(\mathcal{R}_r\,(\mathcal{L}!i)) \\ 0 & \text{otherwise} \end{cases}$$

Definition

Potential of the data structure

$$\Phi\,\mathcal{L}\,\mathcal{R} := \sum_{i=0}^{|\mathcal{L}|-1} \phi\,\mathcal{L}\,\mathcal{R}\,i \tag{3}$$

- $\{0,\ldots,|\mathcal{L}|-1\}$ is in this case the domain of our equivalence relation

Results of the Thesis

The $\mathrm{is\_uf}$ Assertion takes a relation and two arrays as arguments and ensures:

- The well-formedness of the disjoint set forest and the ranks
- That the relation modeled by the array is the one given
- That the necessary potential is stored as Time Credits

Definition

$$
\begin{aligned}
\mathrm{is\_uf}\,\mathcal{X}\,(s,p) :=\; & \exists_A \mathcal{L}\,\mathcal{R}.\, p \mapsto_a \mathcal{L} * s \mapsto_a \mathcal{R} * \\
& \uparrow (\mathrm{ufa\_}\alpha_{\mathcal{L}} = \mathcal{X} \wedge \mathrm{invar\_rank}\,\mathcal{L}\,\mathcal{R}) * \\
& \$(4 \cdot \Phi\,\mathcal{L}\,\mathcal{R})
\end{aligned} \tag{4}
$$

Results of the Thesis

Lemma

$$\text{uf\_cmp\_time} \in \mathcal{O}(\alpha_r\, n)$$

Theorem

$$\langle \text{is\_uf}\, \mathcal{X}\, u * \$(\text{uf\_cmp\_time}\, |\text{Dom}\, \mathcal{X}|) \rangle$$
$$\texttt{uf\_cmp}\, u\, i\, j$$
$$\langle \text{is\_uf}\, \mathcal{X}\, u * \uparrow (r \leftrightarrow (i, j) \in \mathcal{X}) \rangle_t$$

Lemma

$$\text{uf\_union\_time} \in \mathcal{O}(\alpha_r\, n)$$

Theorem

Assumes: $i, j \in \text{Dom}\, \mathcal{R}$

$$\langle \text{is\_uf}\, \mathcal{X}\, u * \$(\text{uf\_union\_time}\, |\text{Dom}\, \mathcal{X}|) \rangle$$
$$\texttt{uf\_union}\, u\, i\, j$$
$$\langle \text{is\_uf}\, (\text{per\_union}\, \mathcal{X}\, i\, j) \rangle_t$$

- `uf_union` corresponds to the **Union** and `uf_cmp` to the **Find** operation
- These theorems follow from abstract results about disjoint set forests
- The whole proof is about 5KLoc long in Isabelle/HOL

Conclusions

- Formalization of the state-of-the-art result about Union-Find
- Comprehensive theory about Ackermann, including quantitative and asymptotic results
- The proofs of the imperative programs are still too long:
  - The automation does not deal well with arithmetic
  - It also instantiates existentials too aggressively
  - Repetitive proofs generated by branching $\Longrightarrow$ optimization possible
- Formal verification of non-trivial results about runtime, not only correctness are possible
- Some overhead, but this helps to reveal hidden assumptions

Thank You for listening!

Any questions?

Important Definitions

Definition
invar_rank

$$
\begin{aligned}
\text{invar\_rank}\,\mathcal{L}\,\mathcal{R} :=& \text{ufa\_invar}\,\mathcal{L} \land & (5)\\
& |\mathcal{L}| = |\mathcal{R}| \land & (6)\\
& (\forall (i, j) \in \text{ufa\_}\beta\text{\_start}_{\mathcal{L}}.\, \mathcal{R}!i < \mathcal{R}!j) & (7)\\
& (\forall i < |\mathcal{L}|.\, \mathcal{L}!i = i \longrightarrow 2^{\mathcal{R}!i} \le |\text{descendants}_{\mathcal{L}}\, i|) & (8)
\end{aligned}
$$

Definition
ufa_invar

$$\text{ufa\_invar}\,\mathcal{L} := \forall i < |\mathcal{L}|.\, i \in \text{Dom}\,\text{rep\_of}_{\mathcal{L}} \land \mathcal{L}!i < |\mathcal{L}| \tag{9}$$

Definition
ufa_β_start

$$\text{ufa\_}\beta\text{\_start}_{\mathcal{L}} := \{(x, y) \mid x < |\mathcal{L}| \land y < |\mathcal{L}| \land x \neq y \land \mathcal{L}!x = y\} \tag{10}$$