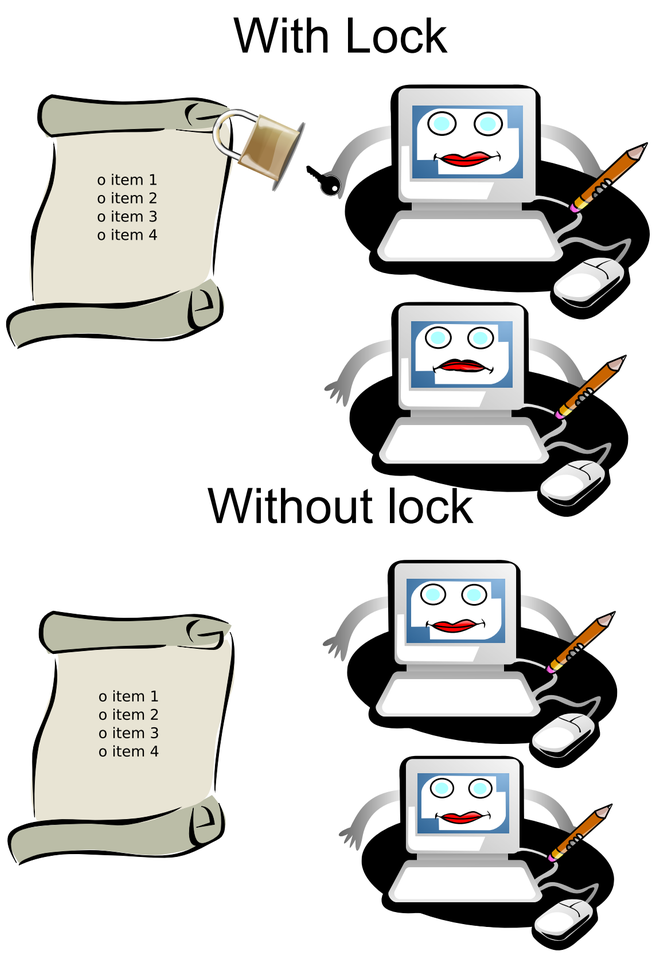
Provably correct lock-free data structures

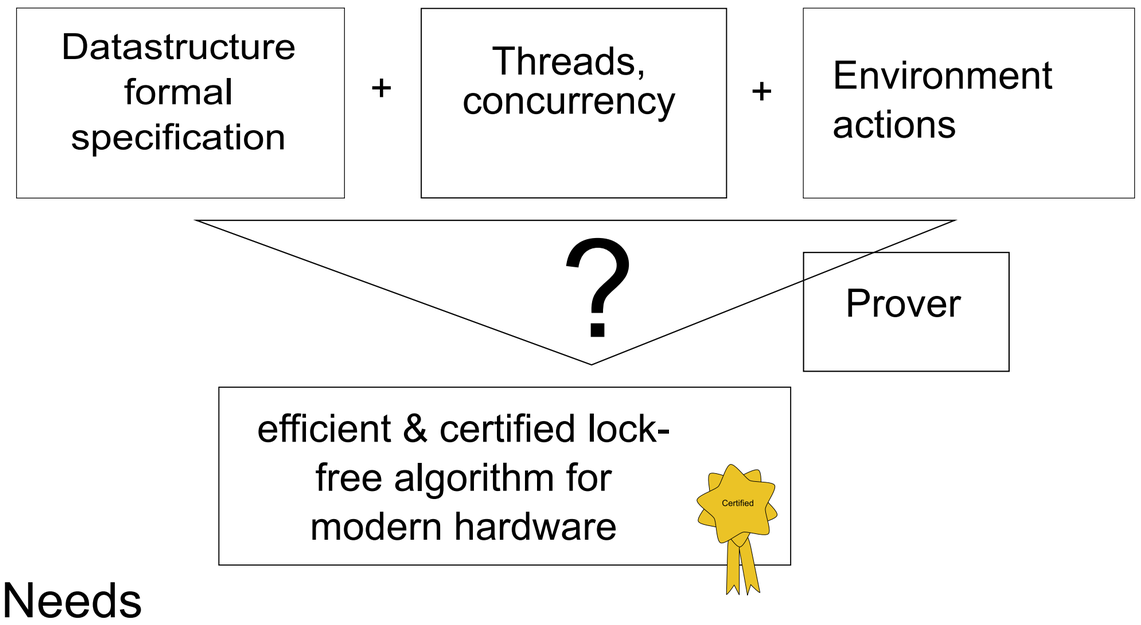
Loïc Fejoz INRIA/MSR/Nancy-University Stephan Merz INRIA Tim Harris MSR

1. Context

Systems and applications ask for highly concurrent actions. Classical solutions like mutual exclusion are not any longer efficient and cause problems like deadlocks, starvations, race conditions.
Moreover, it is difficult to reason about concurrency that is why we need formal proofs of those algorithms.



2. Wanted: a method for developping and verifying algorithm for lock-free datastructures



Needs

compositionality, i.e. reason locally;
linearizability, i.e. act as if sequential;
automatic proving;
liveness, i.e. no blocking;
self-composable.

3. State-of-the-art methods

Assume-Guarantee
(Assumption-Commitment and Rely-Guarantee)

Work like Hoare triples but add concurrency and compositionality by assuming some properties from environment's actions and ensuring some properties.

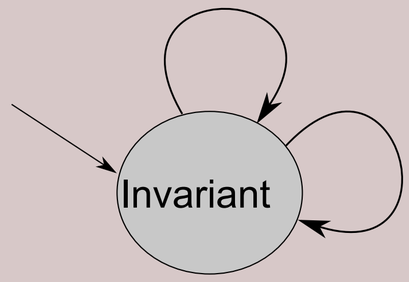
Typical tuple is:
<rely, guarantee>{precondition}Program{postcondition}
like:
<y'=y, y∈ℕ⇒y'∈ℕ>{y=0}y:=10{y'=10}

Parallel composition axiom:

$$\frac{\langle r1,g1\rangle\{\Phi 1\}P1\{\psi 1\}\ \langle r2,g2\rangle\{\Phi 2\}P2\{\psi 2\}}{\langle r1 \wedge r2,\ g1 \vee g2\rangle\{\Phi 1 \wedge \Phi 2\}P1 \parallel P2\{\psi 1 \wedge \psi 2\}}$$

Inductive Invariants

all events of the systems must preserve an invariant predicate.



Refinement

We formalize abstract operations as atomic and then derive specifications by adding details until concrete program. It ensures correctness by construction and eases proofs.

Other methods

Separation logic
Ownership
Atomicity type and effect systems
Software Transactional Memory
Predicate diagrams

4. Plan

Adapt Assume-Guarantee to well-established refinement methods like B and TLA+.

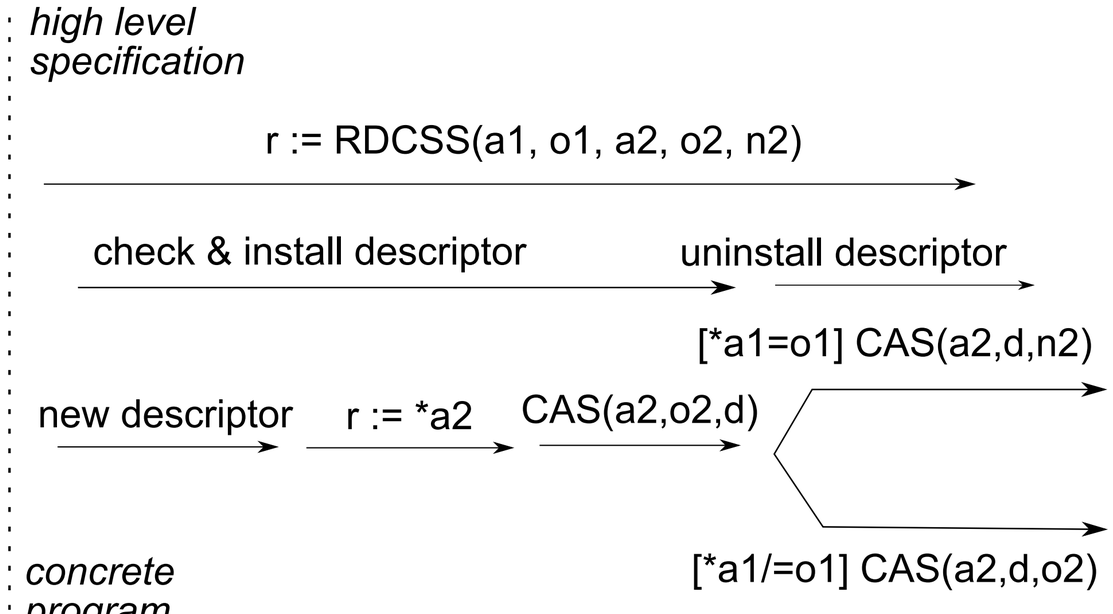
Use automatic theorem-provers.

Prove currently used algorithms.

Develop a methodology to formalize common patterns for such algorithms.

5. Preliminary experiments: RDCSS

RDCSS is a subpart of Multiple-Compare-And-Swap. CAS is the operation that atomically changes the value of a slot if it is equal to a given value. It is present on modern hardware and is known to be a universal operator for concurrency.
RDCSS needs two pointers, two old values and one new value. It changes the first pointer value if the current value of pointers are equal to given old values. The trick in the implementation is to use a descriptor so that other threads can finish operations.



We have formalized this algorithm in B and in TLA+. Abstract models have been checked but current tools and languages do not allow us to verify the implementation.