M2 internship proposal

The internship will hold at Laboratoire d'Informatique et Système (LIS), in the research group DALGO : Distributed Algorithms, https://www.lis-lab.fr/dalgo/. It will be co-advised by

- Hagit Attiya, Technion, Israel
- Alessia Milani, LIS, Aix-Marseille Université, France. If you are interested, contact alessia.milani@lis.fr

1 Context

A key component in the design of concurrent applications are shared objects providing higher-level semantics for communication among processes. For example, a shared queue to which processes can concurrently enqueue and dequeue, allows them to share tasks, and similarly a shared stack. Shared objects are implemented from more basic primitives supported by the multiprocessing architecture, e.g., reads, writes, swap, and compare&swap.

Maurice Herlihy [1] defined a hierarchy that classifies common objects type according to their ability to solve consensus, which is a fundamental problem in distributed computing.

More formally, the consensus number of an object type T is the greatest integer n such that consensus can be implemented in a system of n processes with atomic read/write registers and objects of type T. No type can implement a type with a higher consensus number. For example, registers, which has consensus number 1, cannot implement queue, which has consensus number 2. However, the consensus hierarchy does not let us determine the structure of the "can implement" relation for types with the same consensus number.

There is an algorithm showing that stacks (which has consensus number 2) can be wait-free implementable from any type with consensus number 2 for any number of processes [2]. However, the question whether this kind of implementation exists for queues has been open for many years and has received a considerable amount of attention. The aim of this internship is to solve simpler problems (detailed in the next section) that can provide insight on the difficulty of solving this long-standing open question (see [4]).

Objective: Initially, the intern has to design an algorithm to implement a 2-window-register object [3], using swap objects and an algorithm to implement a swap object using 2-window registers. Both algorithms should be wait-free, meaning that a correct process has to finish its operations despite the possible failure of all other processes in the system.

The next step would be to understand the relation between 2-window registers and queues and stacks.

References

[1] Maurice Herlihy. 1991. Wait-free synchronization. ACM Trans. Program. Lang. Syst. 13, 1 (Jan. 1991), 124–149. https://doi.org/10.1145/114005.102808

[2] Yehuda Afek, Eli Gafni, Adam Morrison: Common2 extended to stacks and unbounded concurrency. Distributed Comput. 20(4): 239-252 (2007)

[3] Achour Mostéfaoui, Matthieu Perrin, Michel Raynal: A Simple Object that Spans the Whole Consensus Hierarchy. Parallel Process. Lett. 28(2): 1850006:1-1850006:9 (2018)

[4] Hagit Attiya, Armando Castañeda, Danny Hendler: Nontrivial and universal helping for wait-free queues and stacks. J. Parallel Distributed Comput. 121: 1-14 (2018)