Mutual exclusion with pointers
Master's thesis

July 10, 2017

supervisor	Martin Steffen et. al.
group	PMA
status	open
type	60 ECTS
recommended background	background in concurrency and formal program verification
study program	computer science

Short description

The task is to develop and prove correct mutual exclusion in a restricted setting, supporting "*pointers only.*"

Problem setting

Mutual exclusion is a classical and important problem in concurrent systems. Many different solutions have been developed, which differ on the assumptions they can rely on and the guarantees they offer. The work is concerned with a rather restricted setting, namely a language that features *only* pointers as data structures. Especially, there are not integers and booleans, and similar data values.

This makes mutual exclusion tricky; conceptually, a way to achive mutual exclusion is that competing activities apply for a "lock" that grants access to the critical region. At the bottom, such a lock is some sort of binary data structure —either the lock is free or taken— and acquiring access involves setting the lock in an atomic manner from free to taken or vice versa. Without booleans, however, one needs to use the pointers to encode such locks, which makes the atomic access to the such-encoded lock tricky itself.

A (partial) solution to that problem is known. The task is to formalize it and to prove it correct. Furthermore, the solution is partial in the sense that certain desirable properties of mutex algorithms are ignored (for instance progress, fairness etc); it concentrates on the basic safety property of mutual exclusion, only. The task is to generalize the solution to better mutex properties.

As a remark: the result is of theoretical importance; to prove semantical problems of certain languages, one needs insight into the expressivity of the language. In this case, the background deals with a "pure" object oriented language, i.e., a language, stripped of all data, but having only object references, and the question underlying the task of above is: does the language get more powerful if having locks in objects or not. If one can encode mutual exlusion/locks with references, only, the locks in the object do not add to the expressivity of the language.