Overview

This folder contains a formalization (in the Coq proof assistant) of the methodology proposed in the following manuscript.

Ximeng Li, Shanyan Chen, Yong Guan, Qianying Zhang, Guohui Wang, Zhiping Shi: Correctness Proofs for Extended OS Kernels. (under submission)

The manuscript proposes an approach to formally verify an *extended OS kernel* based on existing verification result for the *OS kernel*. This verification result for the OS kernel is obtained using a concurrent separation logic (e.g., CSL-R [1]). The extension of the OS kernel is by introducing new objects (called *service objects*) that augment the functionalities of the objects in the OS kernel (*kernel objects*). For this development, the OS kernel is μC/OS-II, and the kernel objects are the Event Control Blocks (ECBs) for the event objects --- the mutexes, semaphores, and message queues --- of the kernel.

This manuscript is a substantial extension of the conference paper [2].

The formalization in this folder includes

1. formalization of code patterns for the creation, deletion and usage of service objects (in a general sense), and a design of invariant conditions that help ascertain a generic integrity condition about the connection between service objects and kernel objects as established by the execution of the code patterns **(Section 3 of the manuscript)**
2. formalization of an enhancement of the CSL-R verification framework that supports the expression of the following content in the abstract specification of API functions, with a soundness proof **(Section 4 of the manuscript)**
   a) the results of the internal functions that are subsequently used in their callers
   b) the assumptions about the data provided by client applications
3. formalization of a technique that helps verify the global properties of modules in the extended OS kernel, and the formal verification of a global property for semaphores --- the initial value of a semaphore, increased by the number of successful V operations, and decreased by the number of successful P operations, is greater than or equal to the current value of the semaphore **(Section 5 of the manuscript)**

Paper-to-code Correspondence

The following tables shows the correspondence between each specific formalized content and the corresponding formal (mechanized) content.

Formalized content	**Manuscript**	**File**	**Name of Formalization**

Code patterns for creation/deletion/use of service objects	Section 3.3	certiucos/code/obj/service_obj_create.v
		certiucos/code/obj/service_obj_delete.v
		certiucos/code/obj/service_obj_oper.v
			`Definition service_obj_create_impl`
			`Definition service_obj_delete_impl`
			`Definition service_obj_oper_impl`

Requirement 1	Section 3.1	certiucos/spec/os_inv.v	`Definition RH_OBJ_ECB_P`
Requirement 2	Section 3.2	certiucos/spec/os_inv.v	`Definition objref_distinct`
The invariant condition sobj_kobj_aux	Section 3.5	certiucos/spec/os_inv.v	`Definition OBJ_AUX_P`

The invariant condition cre_del_mut_ex	Section 3.5	certiucos/spec/os_inv.v	`Definition OBJ_AUX_P`
			`Definition objcre_nodup`

			`Definition objdel_nodup`

Refinement verification of the service functions (thereby establishing the preservation of the invariant conditions)	Section 3.5	certiucos/proofs/obj/ service_obj_create_proof.v
		certiucos/proofs/obj/ service_obj_delete_proof.v
		certiucos/proofs/obj/ service_obj_oper_proof.v
		certiucos/proofs/obj/ kernel_obj_create_proof.v
		certiucos/proofs/obj/ kernel_obj_delete_proof.v
		certiucos/proofs/obj/ kernel_obj_oper_proof.v
		certiucos/proofs/obj/ get_free_obj_idx_proof.v
		…
			`Lemma service_obj_create_Proof`
			`Lemma service_obj_delete_Proof`
			`Lemma service_obj_oper_Proof`
			`Lemma kernel_obj_create_proof`
			`Lemma kernel_obj_delete_proof`
			`Lemma kernel_obj_oper_proof`
			`Lemma get_free_obj_idx_proof`
			…

Adaption of the framework to express results of internal functions and assumptions (semantic level)	Section 4	framework/model/opsem.v framework/proof/ simulation.v	`Inductive spec_step` (modified wrt. CSL-R) `Inductive hapistep` (constructor `hapienter_step` modified wrt. CSL-R) `CoInductive MethSimMod` (modified wrt. CSL-R) `CoInductive MethSim` (modified wrt. CSL-R) `CoInductive TaskSim` (modified wrt. CSL-R) `CoInductive ProgSim` (modified wrt. CSL-R, corresponding to **Definition 4.1** in the manuscript)
Adaption of the framework to express results of internal functions and assumptions (syntactical level)	Section 4	framework/logic/inferules.v	`Definition BuildPreA'` (modified wrt. CSL-R) `Inductive absinfer` (modified wrt. CSL-R) `Inductive absinferfull` (modified wrt. CSL-R) `Inductive InfRules` (addition of `absabort_rule` wrt. CSL-R)

Adaption of the soundness proof for the CSL-R verification framework	Section 4.4	framework/proof/ soundness.v
		framework/proof/ toptheorem.v
		and the files relied upon by these two files
			`Theorem RuleSound`
			`Lemma SmCTaskSim'`
			`Lemma SmCTaskSim`
			`Lemma tsimtopsim`
			`Theorem toptheorem`
			`Theorem Soundness` (corresponding to **Theorem 4.2** in the manuscript)

Instrumented model of computation (syntax)	Section 5.2	properties/anno_language.v	`Inductive spec_code` (corresponding to the syntax of $\hat{s}$ in the manuscript)
Instrumented model of computation (semantics)	Section 5.2	properties/anno_opsem.v	`Inductive spec_step` (modified to maintain local auxiliary states `LAuxSt`) `Inductive htstep` (modified to maintain local auxiliary states `LAuxSt`) `Inductive hpstep` (modified to maintain mappings from task IDs to local auxiliary states `LAuxStMod.map`)

Syntactical descendant relations for annotated abstract statements	Section 5.2	properties/succ_gen.v	`Inductive has_succ0`
			`Definition has_succ`
			`Lemma step_succ0` (corresponding to **Lemma 5.1**

			in the manuscript) `Lemma succ_step_preserve` (corresponding to **Lemma 5.2** in the manuscript)
Satisfaction of current assertion in given annotated abstract statement by given local auxiliary state	Section 5.2	properties/common_defs.v	`Fixpoint head_asrt_sat` (corresponding to the predicate *asrt_sat* in the manuscript)
Proof method for global properties	Section 5.3	properties/proof_method.v	`Lemma proof_method` (corresponding to **Theorem 5.3** in the manuscript)

Instrumented specifications for the semaphores	Section 5.4	properties/sem_sys.v	`Definition sem_pend`
			`Definition semwait`
			`Definition sem_post`
			`Definition semsignal`

Invariant conditions for verifying the correctness property of semaphores	Section 5.4	properties/sema_invs.v	`Definition cond_pv1`
			`Definition cond_pv0`
			`Definition dtinv`

Verification of correctness property for semaphores	Section 5.4	properties/sema_correct.v	`Theorem sema_correct`

Building and Running the Proof Scripts

This formalization is developed using Coq 8.13.2 (September 2021).
To build the proof scripts, first ensure that the "bin" folder of the Coq installation is in the system path.
A makefile should be generated from _CoqProject with the following command:

```
coq_makefile -f _CoqProject -o Makefile
```

To build the complete development, issue the "`make`" command in the top-level folder.
The enhanced verification framework for CSL-R can be built separately with the following command:

```
make framework/proof/toptheorem.vo
```

The refinement verification result for the extended kernel (mainly encompassing the operations of service objects in a general sense, the functions for creating and deleting tasks, and a simplistic tick handler that realizes preemptive scheduling) can be built separately with the following command:

```
make certiucos/proofs/funcs_refine_spec.vo
```

The verification result for the global correctness of semaphores can be built separately with the following command:

```
make properties/sema_correct.vo
```

After the proof scripts are built, they can be run using Proof General or CoqIDE.

References:

[1] Fengwei Xu, Ming Fu, Xinyu Feng, Xiaoran Zhang, Hui Zhang, and Zhaohui Li. A Practical Verification Framework for Preemptive OS Kernels. In Proceedings of 28th International Conference on Computer Aided Verification (CAV), pp. 59-79, 2016.

[2] Ximeng Li, Shanyan Chen, Yong Guan, Qianying Zhang, Guohui Wang, and Zhiping Shi. Refinement Verification of OS Services based on a Verified Preemptive Microkernel. In Proceedings of 27th International Conference on Fundamental Approaches to Software Engineering (FASE), Held as Part of ETAPS, pp. 188-209, 2024.