Hybrid Programming Language for Cryptography in Rocq

Zhenhao Li, Li Zhou

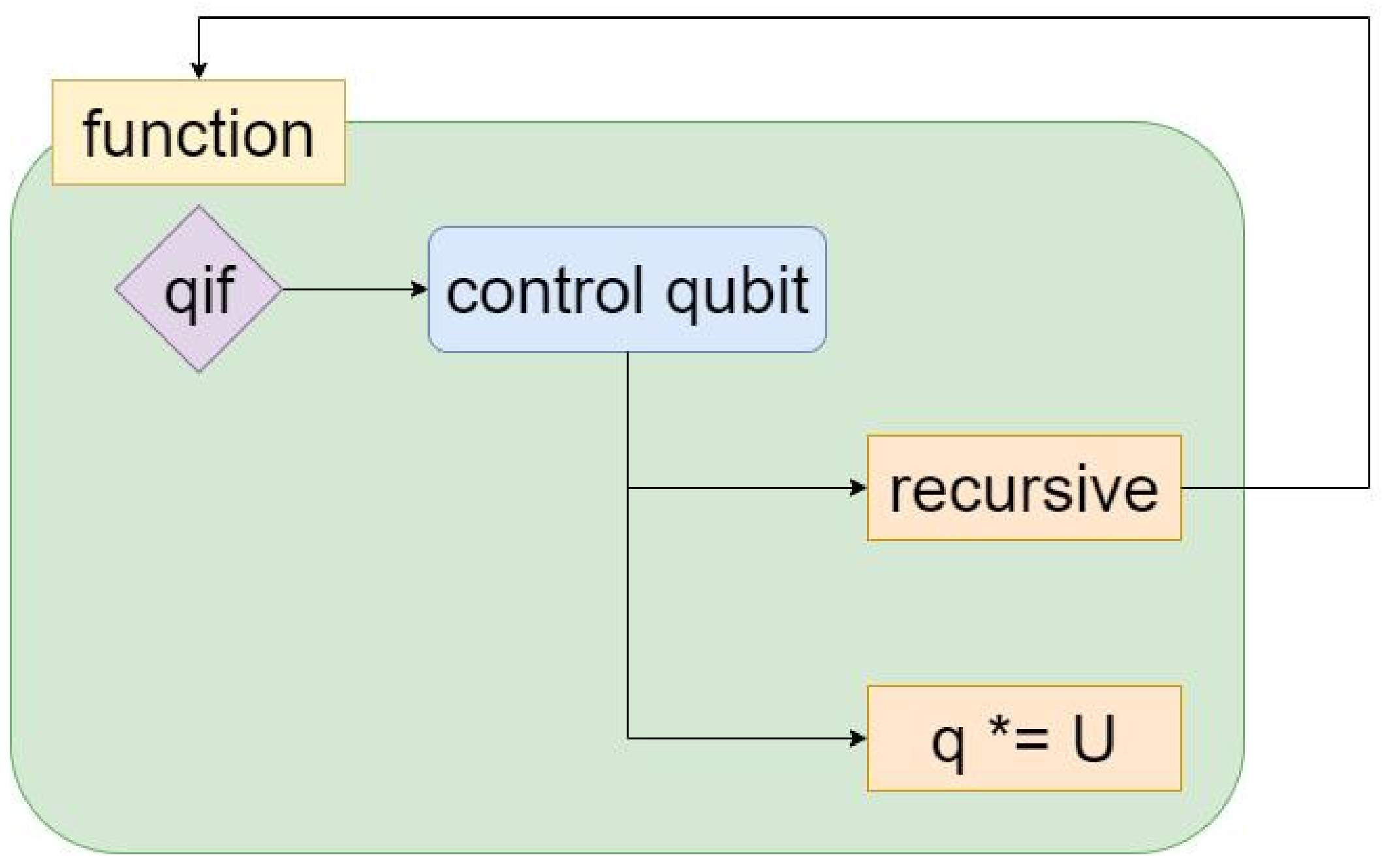
Institute of Software, CAS, China

Background & Motivation

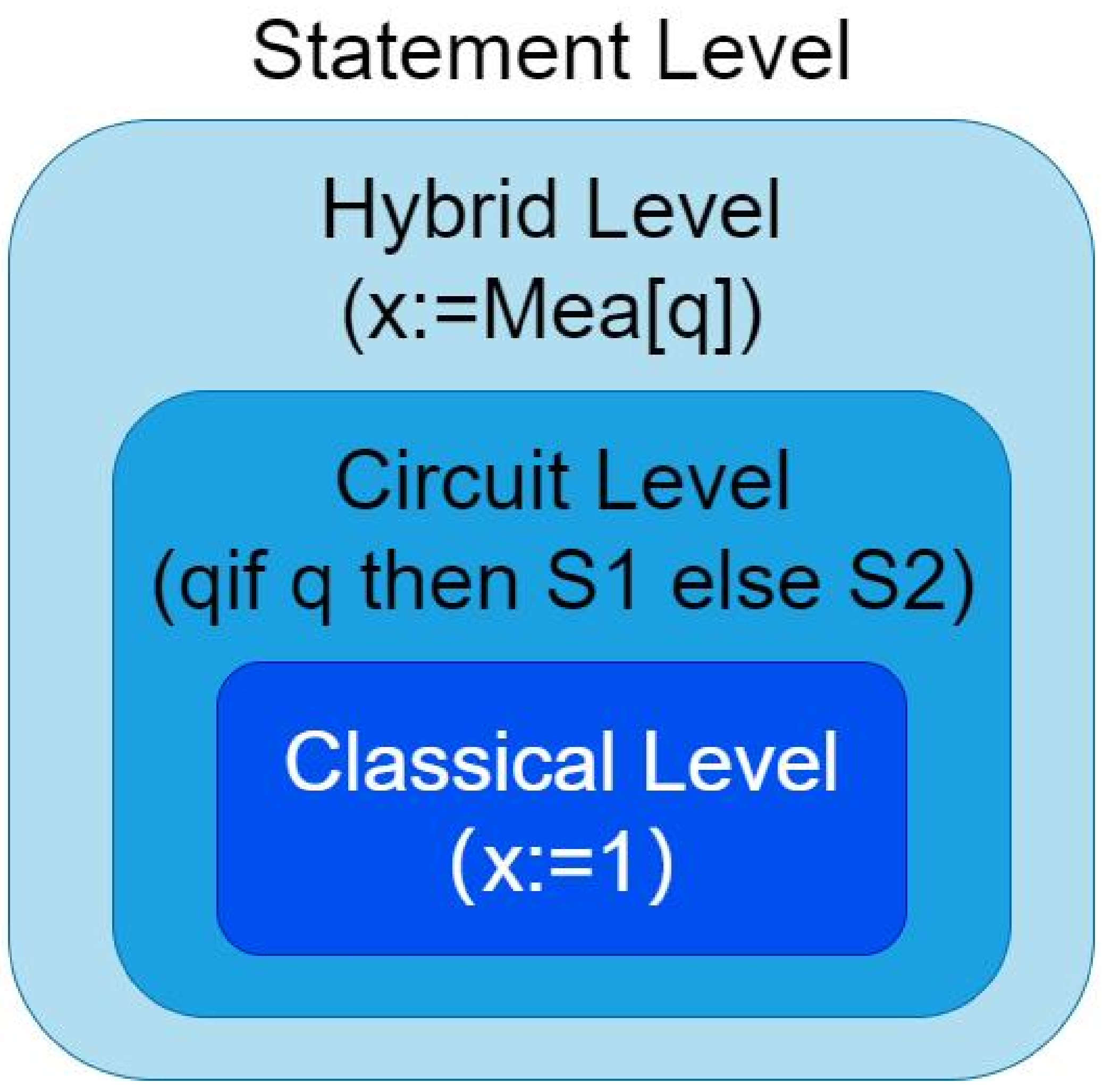
Quantum cryptographic protocols are complex and hard to verify. Existing tools lack support for adversaries and security reasoning, with recent ones offering only limited, non-foundational capabilities.

We design a hybrid language combining classical and quantum features to support complex protocols.
Unified syntax for hybrid programming: unified syntax with three level statements, quantum control flow, quantum recursive procedure
Module system for unspecified programs: module system for describing unspecified or adversarial components

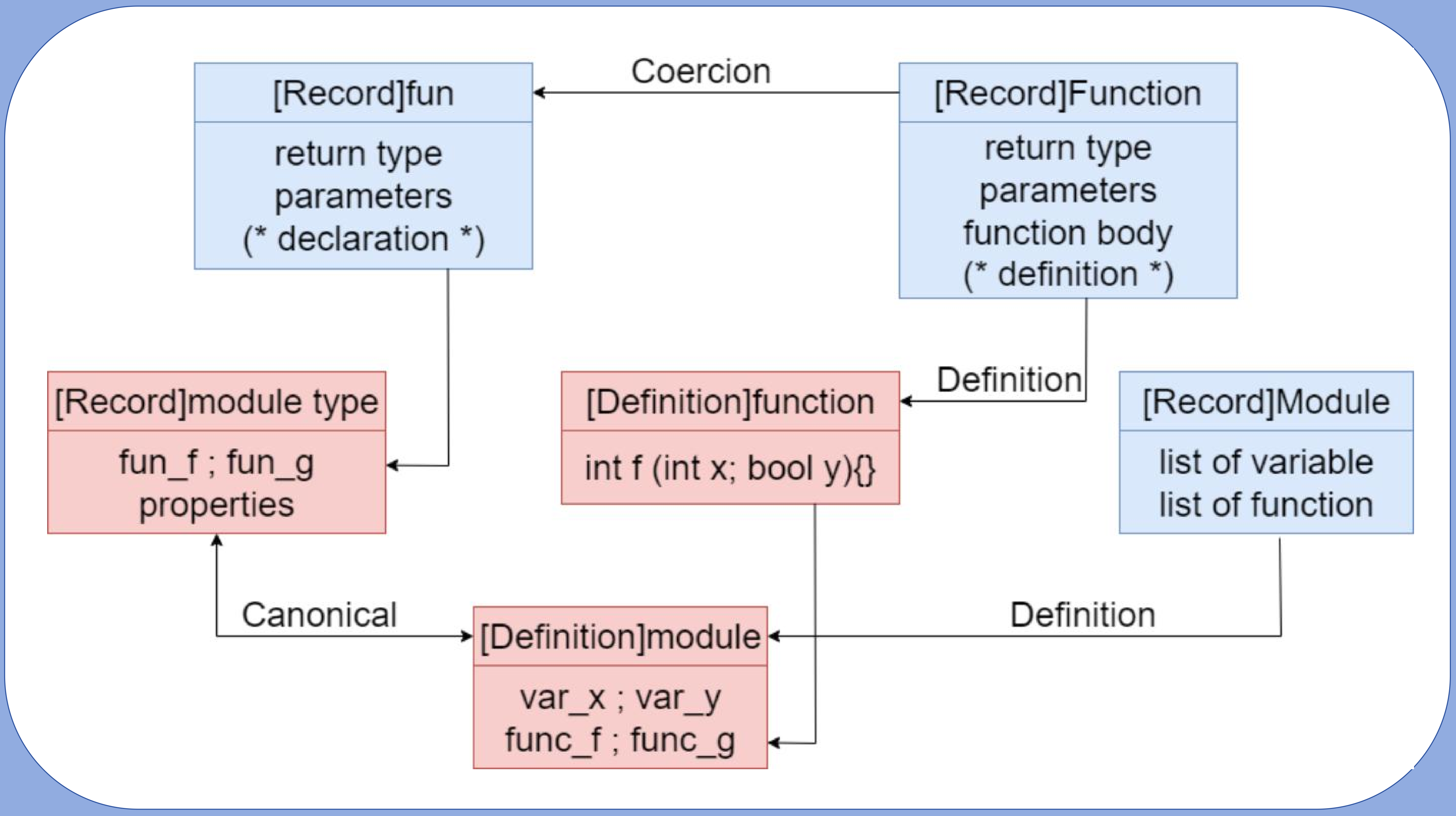
Unified Syntax Hybrid Programming

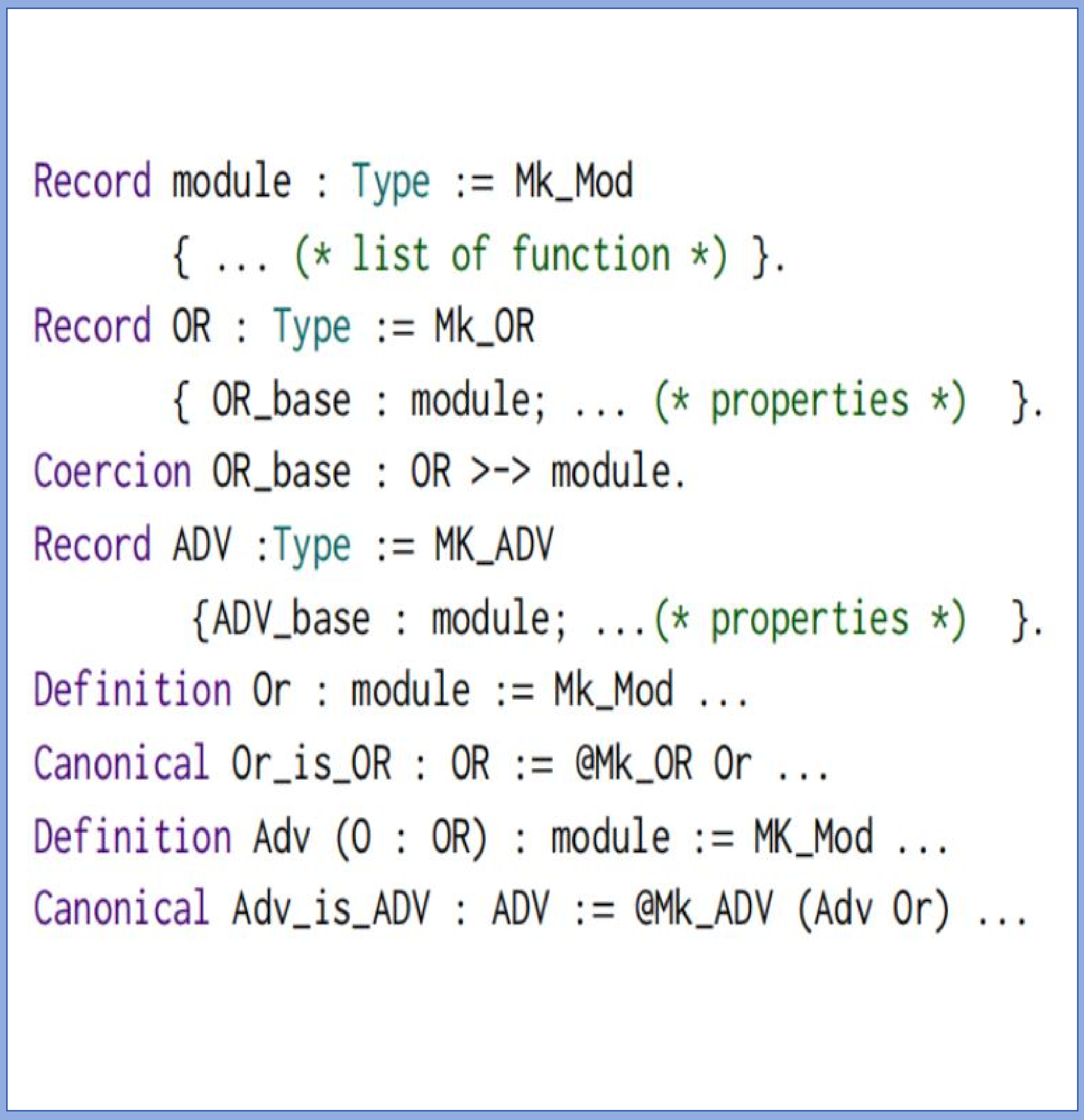


Quantum Control Flow & Quantum Recursive



Module System





```
Record module : Type := Mk_Mod
       { ... (* list of function *) }.
Record OR : Type := Mk_OR
       { OR_base : module; ... (* properties *)  }.
Coercion OR_base : OR >-> module.
Record ADV :Type := MK_ADV
        {ADV_base : module; ...(* properties *)  }.
Definition Or : module := Mk_Mod ...
Canonical Or_is_OR : OR := @Mk_OR Or ...
Definition Adv (O : OR) : module := MK_Mod ...
Canonical Adv_is_ADV : ADV := @Mk_ADV (Adv Or) ...
```

Other Components

We formalized syntax and type checking in Rocq; other components includes:

- ◆ **Plugin support**: rocq-elpi, high-level commands
- ◆ **Semantics**: statements, module system
- ◆ **Verification**: quantum Hoare logic
- ◆ **Rewriting**: rewrite-based reasoning
- ◆ **Applications**: one-way-to-hiding lemma