Hyperkernel：操作系统内核的按钮验证

这篇论文描述了一种设计、实现和验证操作系统内核功能的验证方法，大概的方向是给出一些函数或者调用的 specification，即函数具体完成哪些工作，有什么功能，对于系统状态做了什么修改等等。进而通过形式化证明，得到一个“正确的”实现。此前的工作主要在文件系统方面：因为文件系统需要保证实现的正确性来保证在 crash 之后，数据能够恢复到一个预期的状态，或者借此来提高函数调用的可扩展性。近几年，Frans 组更加集中在如何给出一个完备的 specification 和实现一个正确且并发度高的文件系统上。这一系列的工作的核心就是给出一个 specification 并且借助这个 specification 给出正确性证明和实现，这篇论文便是这个思路在 OS kernel 的实现。

Hyperkernel 实现和正式验证操作系统内核功能正确性的方法，具有高度验证自动化和低证明负担。面临三个挑战：界面设计、内核代码中的虚拟内存管理、用 c 编写，会使形式化推理变复杂。它引入三个关键思想来实现证明自动化：限定了内核接口以避免无限循环或递归；分离内核和用户地址空间以简化虚拟内存的推理；在 LLVM 中间表示级别执行验证以避免建模复杂的 C 语言。

论文通过两种规范来描述“正确的”内核接口：状态机规范提出了一个内核调用和中断处理函数的状态机，以及这个规范的“正确的”C 实现；声明式规范用具体的语言描述规范，提炼这两种规范。第一种包含更多的细节，而第二种在更高的层面上描述大概的作用，便于人类理解。

论文的主要工作主要集中在如何缩小系统并修改系统的接口上，可能有很多的时候都是在证明 LLVM 中间代码和接口给出的规范时，遇到了求解器不可解或者实现困难的问题，退回来修改系统本身，使得工作的目标从证明系统的正确性，并借此实现一个系统，变成简化的教学操作系统。

最后论文使用 Z3 SMT 解算器验证了 Hyperkernel 的实施，共检查了 50 个系统调用和其他陷阱处理程序。表明可以避免类似于 xv6 中发现的错误，以及该验证可以在低证明负担下实现，以较小的代价得到内核正确性的证明，也就是所谓的 push-button。论文提供了：一个低开销的方法来构造一个验证过的操作系统内核、一种便于 SMT 求解的接口设计方法、一个有合适的性能的操作系统（Hyperkernel）的实现。

这篇论文相对以往工作有很大不同，同文件系统和一般的 UNIX-like 系统调用不同，操作系统有着更多更加复杂的函数和数据结构，以及更加复杂的层次关系。同时，路径数目更大，不便于 Z3 求解。论文中提出的方法，实际上是针对系统这个目标进行简化和修改。一方面，把证明的内容从 C 代码变为 llvm 中间代码，使得证明的语义更加规范。另一方面，限制了系统的特性，进行证明的是一个单处理器关中断且简化虚拟内存实现的操作系统。最后，还修改了一般接口实现，要求接口实现添加约束，事实上使得证明的代码实现变得简单了。

总的来说，这篇论文提拱了两个主要的贡献：提供经过验证的操作系统内核的按钮式方法以及适用于 SMT 解决方案的内核接口设计。