从模型到可执行程序的形式化验证
可信编译器l2c

康烁

浙江迪捷软件

*本报告内容大部分来自王生原老师的l2c的相关材料

什么是形式化?

✧ 什么是形式化

用非形式化的方式描述：就是一个可辨认的字符集，外加一套既定的规则和公理，来进行严格推理的方法。

自然数的形式化

✧皮亚诺公理：五条公理的非形式化描述

1. 0是自然数；
2. 每一个确定的自然数a，都具有确定的后继数a‘，a’也是自然数
3. 0不是任何自然数的后继数；
4. 不同的自然数有不同的后继数，如果自然数b、c的后继数都是自然数a，那么b=c；（避免了环的出现）
5. 设S⊆N，且满足2个条件（i）0∈S；（ii）如果n∈S，那么n‘∈S。则S是包含全体自然数的集合，即S=N。(这条公理也叫归纳公理，保证了数学归纳法的正确性)

形式化的意义

- 形式化证明
  - 通过形式化方法来证明任意情况下软件是正确的。
- 传统软工的测试
  - 通过各种测试手段来验证大部分情况软件是正确的。

形式化的历史渊源

- ✧亚里士多德（公元前384～前322）
  - ✧推理应该理性化:所有人都会死 & 柏拉图是人 -> 柏拉图会死
- ✧莱布尼兹和布尔（公元17世纪）
  - ✧推理应该数学化：动物 + 理智 = 人 -> 人 – 理智 =动物
- ✧弗雷德（公元19世纪中期）
  - ✧命题逻辑：雪的颜色是白的；
  - ✧谓词逻辑：f(x) = x是白的；
- ✧ 康托，罗素（公元19世纪末期）
  - ✧集合论，数学原理，罗素悖论
- ✧希尔伯特和哥德尔（公元19世纪初期）

形式化的历史渊源（续）

- ✧邱奇和图灵（1930-1940）
  - ✧希尔伯特的第十题：丢番图问题
  - ✧lambda算子和图灵机
- ✧Curry-Howard同构（1933-1968）
  - ✧命题即类型，证明即程序。
- ✧图灵提出程序正确性（1949）
- ✧很多人开始构造程序的证明工具（1950-2000年）
  - ✧ 麦卡锡、Dijkstra、Hoare等
- ✧seL4和CompCert的成功（2000年以后）
  - ✧以Coq和isabell为代表的定理证明工具
  - ✧ 鸿蒙操作系统

如何形式化验证编译器

- 典型的程序验证策略
  - 逻辑与类型的对应（Curry-Howard 对应）
    - Program Specification

      类型 <=> 逻辑公式
    - Program

      项 <=> 证明
    - Program Verification

      类型检查 <=> 证明推导
    - Programing Methodology

      编程技术 <=> 证明方法

为什么需要形式化编译器

- 传统的方法不能彻底解决“误编译”问题
  - Csmith 工具测试 C 编译器

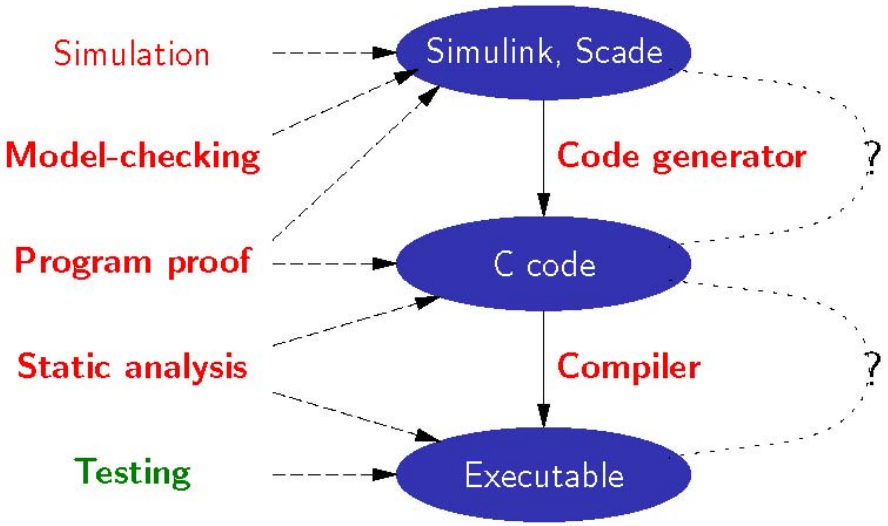
> **Abstract.** Compilers should be correct. To improve the quality of C compilers, we created **Csmith, a randomized test-case generation tool**, and spent three years using it to find compiler bugs. During this period we **reported more than 325 previously unknown bugs** to compiler developers. Every compiler we tested was found to crash and also to silently generate wrong code when presented with valid input. （摘自 PLDI 2011 论文）

为什么需要形式化编译器

✧ 形式化验证编译器的好处

– 经过形式化验证的编译器具语义保持性

- ✓ 能够将源程序的行为和性质保持到所生成的目标程序
- ✓ 源程序级的验证工作不必要在目标程序重复（特别适合于和静态分析工具搭档）



典型的安全攸关的软件开发流程

（图片源于 X.Leroy POPL2011 特邀报告）

为什么要形式化编译器

✧ 形式化验证过的编译器有很好的口碑

– 例　Csmith 工具测试 CompCert 编译器

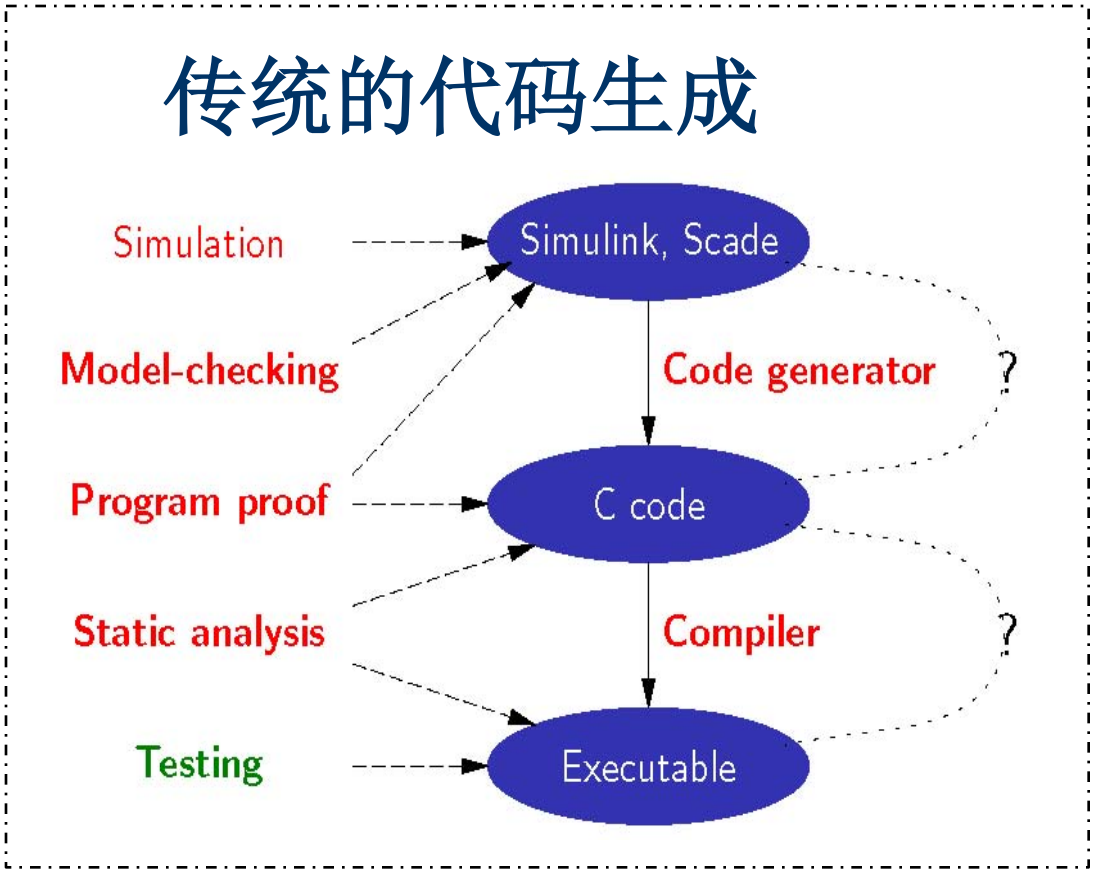
The striking thing about our CompCert results is that the middle end bugs we found in all other compilers are absent. As of early 2011, the under-development version of **CompCert is the only compiler we have tested for which Csmith cannot find wrong-code errors**. This is not for lack of trying: we have devoted about six CPU-years to the task.

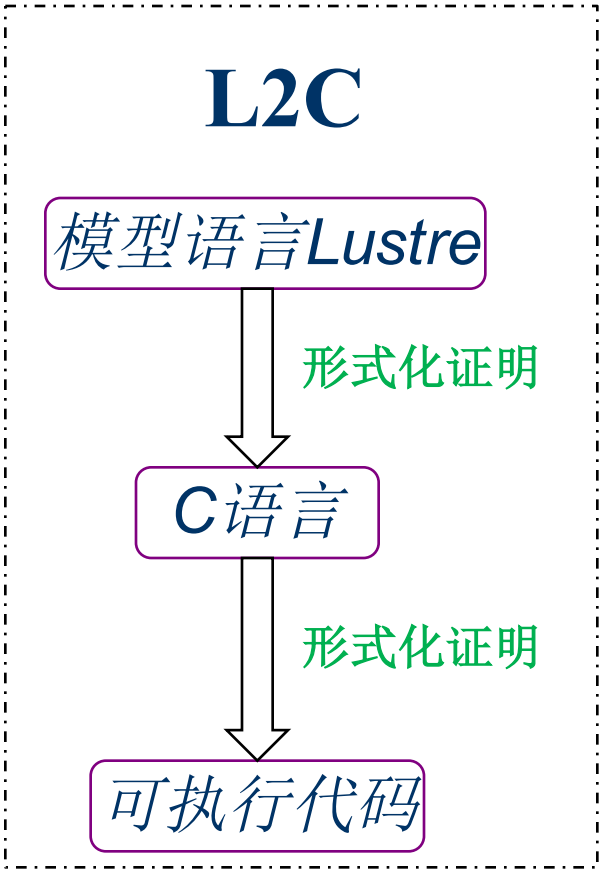
（摘自 PLDI 2011 论文）

– CompCert 的代表性论文的作者Xavier Leroy获得了2016年度的十年来最有影响 POPL论文奖（Most Influential POPL Paper Award）

L2C做了什么事情

✧ 从模型到可执行代码的全流程验证

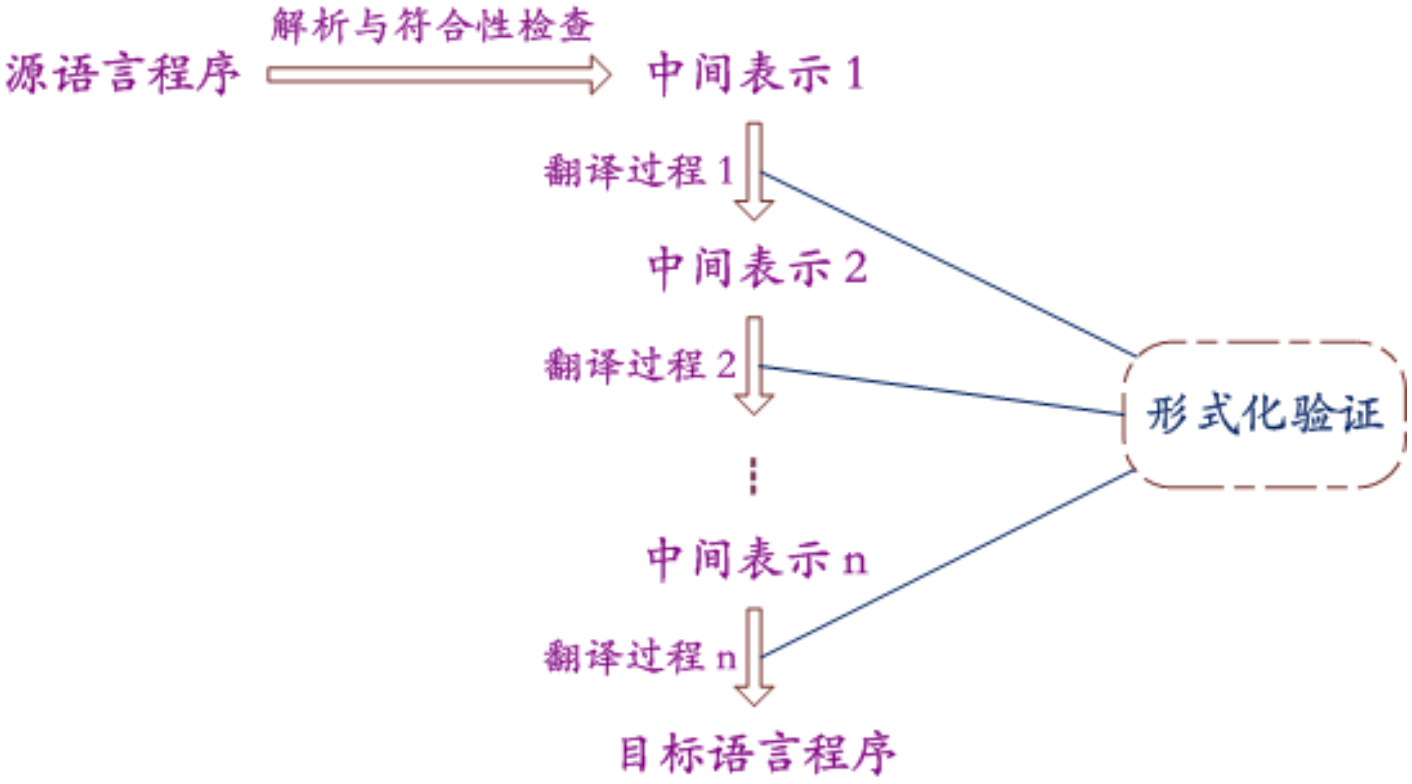




L2C的原理和实现

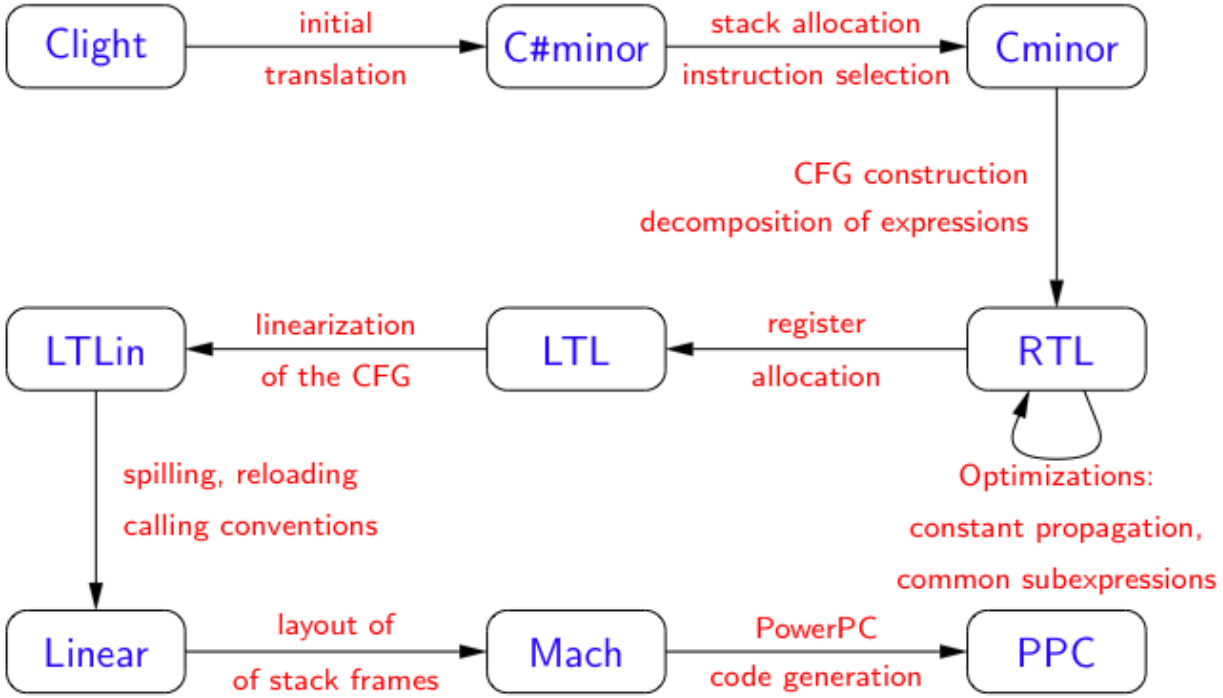
✧ 实现思路

– 通过定理证明方法对各个编译过程本身进行验证



L2C的原理和实现

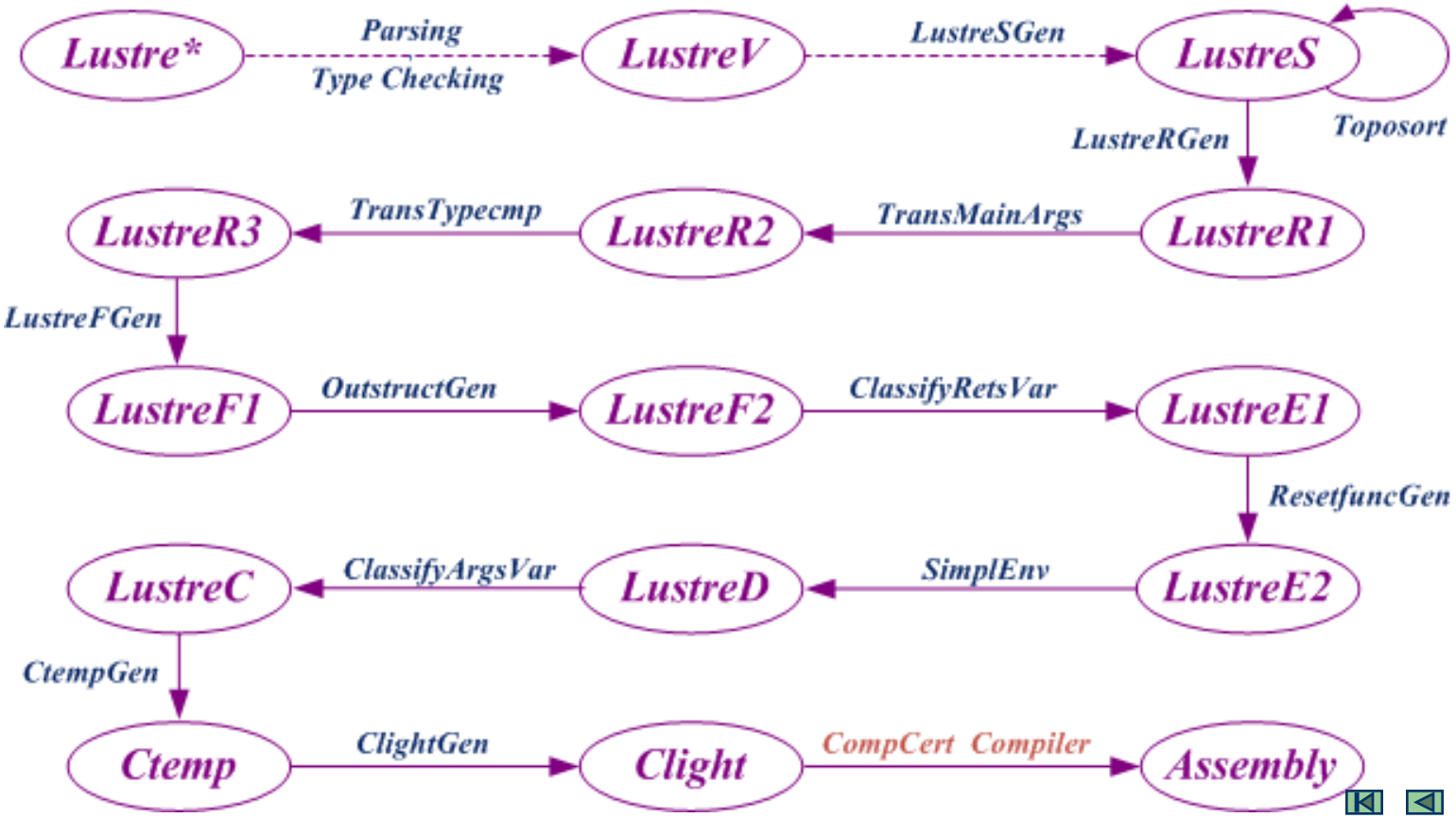
- 通过定理证明方法对编译过程本身进行验证
  - 例：*CompCert* 编译器经过验证的各遍翻译过程



（图片源于 CompCert 项目）

L2C的原理和实现

- 通过定理证明方法对编译过程本身进行验证
  - *L2C* 可信编译器/代码生成器的各遍翻译过程



L2C的未来

✧ 安全攸关领域的嵌入式软件设计与开发未来

– 日益增加的规模和复杂性

随着各种设备的智能化和复杂化，安全攸关领域（航空、航天 、核电、轨道交通、医疗及军事等）的软件代码规模越来越大，这些领域的核心或嵌入式设备中的软件代码一旦出错（失效），会造成人民生命财产损失巨大。

– 趋势：基于模型的系统工程（MBSE）或者MBD

传统的靠程序员手工编码进行研制的方式早已不能满足安全需要。

目前基于模型的系统工程的研发（在智能制造领域称为基于模型的开发和设计）已日益趋向于主流，特别是面向安全攸关领域的高安全性嵌入式软件的设计与开发

空客A380 的多数代码由 Scade 工具自动生成

– 未来的无人驾驶、物联网和人工智能等

L2C的应用成果和现状

- ✧ L2C的意义
  - ✧第一个把模型语言到二进制代码做全过程形式化的工具
  - ✧法国M. Pouzet教授的项目组的VŽlus项目，PLDI2017
- ✧应用成果

  相关软件和技术已应用于某核电公司的的可信代码生成器（之前用Scade工具）
- ✧开源版本

  清华王生原老师团队

  项目地址：https://github.com/l2ctsinghua/l2c
- ✧商业版本

  浙江迪捷软件科技有限公司负责商业化，迪捷软件公司是专注于关键领域的软件工具提供商，有skyeye,l2c等产品，用于防务，核电和轨交等领域。

欢迎大家参与L2C！

谢谢！

联系：13651119140
ksh@skyeye.org