由 Menhir[20](Ocaml 的一种常用的语法分析器生成器)自动生成的,词法分析程序是由面向 Ocaml 的词法分析器构造工具 Ocamllex 生成的.词法和语法分析相关代码中,除去来自 CompCert 的 validator(Coq 代码 3 138 行)以及 Coq 的库代码,包含 Coq 代码(Tree.v,Tokenizer.v,Extraction.v)522 行、Ocaml 代码(.ml)676 行、lexer 描述文件(.mll)183 行以及 parser 描述文件(.vy)617 行.

5.2.2 静态类型检查的正确性

静态类型检查过程对 P3 抽象语法树实施类型检查,其正确性可从两个方面描述:(1) 通过类型检查程序的 AST 一定是类型良好的;(2) 类型良好的 AST 一定能通过类型检查(即类型检查过程能正常结束).假设类型检查过程实现为函数 *typecheck_prog*;基于类型规则(见第 3 节)可定义 AST 形式的 P3 规范,*p* 是类型良好的,用 *wt_spec*(*p*)表示.这两个性质可以表示如下.

(1) $\forall p.typecheck\_prog(p)=OK(p)\rightarrow wt\_spec(p)$;

(2) $\forall p.wt\_spec(p)\rightarrow typecheck\_prog(p)=OK(p)$.

其中,$typecheck_prog(p)=OK(p)$表示函数 *typecheck_prog* 作用于 P3 规范 *p* 可正常终止,且不对 *p* 进行任何修改(返回 *p* 本身).

5.2.3 产生汇编的正确性

生成 P3 汇编语言是 P3C 编译器的核心工作,它将容易理解和使用的 P3 语言翻译成更加接近目标文件格式的一种中间语言.因此,保证翻译前后两种语言语义的一致性,将是保证编译过程正确性的关键.形式化验证是确保编译过程正确性的必要途径,主要有两种方法:(1) 构造并证明编译过程本身的正确性,类似于 CompCert 编译器[19]的构造方法;(2) 采用翻译确认[12]的思路,构造一个 validator 检查翻译前后的程序/规范在语义上的一致性,同时确保该确认程序的正确性(最好能够给出证明).P3C 编译器选择的是后一种方法(参考图 10),即构造从 P3 AST 到 P3 ASM 的一个确认程序,并证明其正确性.这一选择的出发点主要有:

1) 因需要软硬件协同设计,翻译程序需要尽早实现;而同时,其正确性证明的难度和工作量很大,时间上不能承受.即使能够按期实现,但协同设计的许多不确定性,意味着翻译过程的修改比较频繁.然而证明过程无法重用,导致整个工作成本大增.
2) 从我们的问题出发,设计从 P3 AST 到 P3 ASM 的确认程序的方案是相对容易的,而且其正确性证明比起直接验证翻译过程更加容易.
3) P3 AST 到 P3 ASM 的定义(语法、静态和动态语义)已经明确,比如本文前面几节的 P3 类型系统和操作语义等内容.P3 ASM 的语法和语义定义相对简单一些(限于篇幅本文不予介绍),而这些内容不再变化,因此构造一个确认程序并证明其正确性具有更好的可重用性.

确认程序也可以借助于被认为可信的工具(如模型检验器和自动求解器)实现.比如,Lopes 等人基于 Z3 求解器的原型工具[9]完全可以作为确认程序,用于确认翻译前后的某种语义一致性;Kheradmand 等人基于 KEQ[11]提出了一种 P4 编译器的翻译确认实现方案[10].然而,一般不会尝试对此类工具进行正确性证明,一方面难度很大,另一方面,人们对此类工具已足够信任.与此不同,P3C 中确认程序将进行正确性证明.

确认程序的正确性可描述为[21]:$\forall S,C.Validate(S,C)=\text{true}\Rightarrow S\approx C$,其中,*S* 和 *C* 分别为任一具体的 P3 AST 和 P3 ASM,$Validate(S,C)$成真表示确认成功,$S\approx C$ 表示 *S* 和 *C* 语义上是一致的.

所有自定义寄存器的访问区间不重叠(参见第 2.4.2 节及第 3.4.1 节)的特性可简化确认程序正确性的证明.

确认程序是编译器的一部分,将会附加于翻译过程之后.包含确认程序的翻译过程可描述如下[21].

$Comp'(S)$=match $Comp(S)$ with

$|Error\rightarrow Error$

$|OK(C)\rightarrow$if $Validate(S,C)$ then $OK(C)$ else $Error$

6 总结及未来工作展望

本文的主要贡献是设计了一种面向安全网络的可重构数据包解析器的专用硬件配置描述语言 P3,并提出

其可信实现方案.基于对可重构硬件基本需求的充分理解,经过软硬件设计人员的反复沟通与协同设计,最终明确了如文中展示的 P3 语言核心特性(第 2 节)及其编译结构(第 5 节).P3 语言及其编译器 P3C 的设计侧重于可信需求,因此本文重点描述了该语言的静态语义(第 3 节的类型系统)以及动态语义(第 4 节的操作语义)定义.

P3 语言语法的完整定义参见设计文档[17],该文档中也包含了 P3 编译器 P3C 设计中的中间语言(AST 和汇编语言)和目标语言(硬件配置文件描述语言)的完整定义,其他内容也会随着项目进展补充完善.限于篇幅,关于中间语言和目标语言,本文仅通过样例予以说明,参见第 5.1.3 节和第 5.1.4 节.

本文的核心内容是 P3 语言的设计,对于编译器 P3C 仅介绍其设计框架(参见第 5 节),不包括其实现.基于图 10 所示的编译器结构,P3C 各个部分的实现工作正在进展中.当前已完整实现部分及其各模块功能如图 14 所示,代码可下载[22],其中包括词法和语法分析(含语法分析的确认程序及其正确性验证)的完整实现(参见第 5.1.1 节和第 5.2.1 节),下载后可安装运行.

现阶段 P3C 项目正在开展的工作包括静态类型检查和产生汇编的模块,同时,硬件方面也在设计更多的使用样例,用于测试和协同设计.未来剩余的工作中,最具挑战性的部分是产生汇编过程的正确性验证(参见第 5.2.3 节),将采用翻译确认的方法实现一个 validator,并证明其正确性.确认程序本身的实现并不复杂,仅与 AST 和汇编代码的结构相关,与产生汇编代码的过程无关.确认程序的正确性证明涉及到 AST 和汇编语言的语义定义,前者参见第 4 节,后者已完成.剩余工作均在 Coq 中完成,有一定的工作量.一些看似简单的工作,在实现时也可能不简单(至少从代码量角度来看).比如,根据本文第 3 节和第 4 节的定义,针对 P3 操作语义,P3 类型系统的可靠性是显然的,在 Coq 实现中,虽然证明思路也是显然的,但给出相应的证明会有不小的工作量.

期待 P3C 项目的开展能促进相关工作的进一步研究,比如实现 P4 语言编译器本身或者翻译确认程序的机器证明(对于 P4 语言编译器的正确性验证,目前已有的工作主要集中在基于模型检验器或自动求解器进行翻译确认).

References:

[1] Broadcom Corporation. Broadcom BCM56850 StrataXGS®Trident II Switching Technology. Broadcom, 2013.

[2] Davie BS, Rekhter Y. MPLS: Technology and Applications. Morgan Kaufmann Publishers Inc., 2000.

[3] Hanks S, Meyer D, Farinacci D, *et al*. Generic Routing Encapsulation (GRE). RFC 2784, 2000.

[4] Mahalingam M, Dutt DG, Duda K, *et al*. Virtual eXtensible local area network (VXLAN): A framework for overlaying virtualized layer 2 networks over layer 3 networks. RFC 7348, 2014.

[5] McKeown N. Software-defined networking. INFOCOM Keynote Talk, 2009,17(2):30−32.

[6] Bosshart P, Gibb G, Kim HS, *et al*. Forwarding metamorphosis: Fast programmable match-action processing in hardware for SDN. ACM SIGCOMM Computer Communication Review, 2013,43(4):99−110.

[7] Bosshart P, Daly D, Gibb G, *et al*. P4: Programming protocol-independent packet processors. ACM SIGCOMM Computer Communication Review, 2014,44(3):87−95.

[8] Liu J, Hallahan W, Schlesinger C, *et al*. P4v: Practical verification for programmable data planes. In: Proc. of the 2018 Conf. of the ACM Special Interest Group on Data Communication. ACM, 2018. 490−503.

[9] Lopes N, Bjørner N, McKeown N, *et al*. Automatically verifying reachability and well-formedness in P4 networks. Technical Report, 2016.

[10] Kheradmand A, Rosu G. P4K: A formal semantics of P4 and applications. arXiv Preprint arXiv:1804.01468, 2018.

[11] The K Framework Development Team. KEQ. 2017. https://github.com/kframework/k/tree/keq

[12] Pnueli A, Siegel M, Singerman E. Translation validation. In: Proc. of the Int'l Conf. on Tools and Algorithms for the Construction and Analysis of Systems. Berlin, Heidelberg: Springer-Verlag, 1998. 151−166.

[13] Benácek P, Pu V, Kubátová H. P4-to-Vhdl: Automatic generation of 100 Gbps packet parsers. In: Proc. of the 2016 IEEE 24th Annual Int'l Symp. on Field-programmable Custom Computing Machines (FCCM). IEEE, 2016. 148−155.

[14] Zhao Y, Yin SJ, Li XY. Reconfigurable unit design in a reconfigurable Ethernet packet parser. Computer Engineering & Science, 2020,42(2):220−228 (in Chinese with English abstract).

[15] P4 Language Consortium. P4 language specification, Version 1.0.4. 2017. https://p4.org/specs/

[16] Peterson LL, Davie BS. Computer Networks: A Systems Approach. Elsevier, 2007.

[17] P3 language specification, draft. 2019. https://github.com/leeehh/P3_language_compiler/raw/master/doc/P3_Compiler.pdf

[18] Jourdan JH, Pottier F, Leroy X. Validating LR (1) parsers. In: Proc. of the European Symp. on Programming. Berlin, Heidelberg: Springer-Verlag, 2012. 397–416.

[19] CompCert Home. 2008. http://compcert.inria.fr/

[20] Pottier F, Régis-Gianas Y. Menhir reference manual. 2018. http://gallium.inria.fr/~fpottier/menhir/manual.pdf

[21] Leroy X. Formal verification of a realistic compiler. Communications of the ACM, 2009,52(7):107–115.

[22] P3 language compiler. 2019. https://github.com/leeehh/P3_language_compiler

附中文参考文献:

[14] 赵宇,殷树娟,李翔宇.一种可重构以太网数据包解析器中可重构单元的设计.计算机工程与科学,2020,42(2):220–228.

李璜华(1996－),男,硕士生,主要研究领域为编译器,形式化验证.

李凌(1995－),男,学士,主要研究领域为编译器,形式化验证.

赵宇(1995－),男,硕士生,主要研究领域为数字集成电路系统与设计.

王生原(1964－),男,博士,副教授,CCF 高级会员,主要研究领域为程序设计语言与系统,编译器设计,形式化方法.

李翔宇(1977－),男,博士,副研究员,博士生导师,CCF 专业会员,主要研究领域为信息安全芯片,硬件安全.