法律合约与智能合约一致性综述 *

孟 博，刘 琴，王德军，王潇潇，郑绪睿

(中南民族大学 计算机科学学院，武汉 470074)

摘 要：智能合约是区块链的核心组成部分。随着区块链技术的不断发展，智能合约的法律问题也引起越来越多的关注，法律合约与智能合约的一致性是智能合约的关键属性。从计算机科学的角度，对法律合约与智能合约的一致性研究现状进行了总结和分析。首先，分别对法律合约描述语言和智能合约开发语言进行了分类和分析，总结了每种语言的特点；其次，根据合同自动化执行的三个发展阶段，对法律合约和智能合约的一致性内涵进行了分析和讨论；然后，分别基于形式化模型和合约模板，总结了由法律合约生成智能合约代码的方法和关键技术，并对其进行了讨论和评价；最后，归纳总结全文，对未来法律合约与智能合约的一致性研究方向进行了展望。

关键词：形式化模型；合约模板；特定领域；通用领域；代码生成

中图分类号：TP391 doi: 10.19734/j.issn.1001-3695.2019.12.0652

Review on conformance between legal contract and smart contract

Meng Bo, Liu Qin, Wang Dejun, Wang Xiaoxiao, Zheng Xurui

(*College of Computer Science*, *South-Central University for Nationalities*, *Wuhan* 470074, *China*)

Abstract: Smart contract is the key of the blockchain. With the continuous development of block chain technology, the legal issues of smart contract have attracted more and more attention, and the conformance between legal contract and smart contract is the core property of smart contract. From the view of computer science, this paper summarizes and analyzes the up-to-date development of the conformance between smart contract and legal contract. Firstly, it classifies and summarizes the language of legal contract and smart contract respectively, and presents the advantages and disadvantages of each language. Secondly, it discusses the legal connotation of smart contract according to the three development stages. Then, it summarizes the main research methods of generating smart contract code from legal contracts based on formal transformation and contract template, and analyzes their conformance. Finally, this paper makes a look at the future research directions of the conformance between legal contract and smart contract.

Key words: formal model; contract template; specific field; general field; code generation

0 引言

智能合约是分布式账本技术的关键和重要组成部分，随着分布式账本技术的快速发展，特别是区块链大规模的部署和应用，智能合约受到了人们的重点关注。1995 年，Nick Szabo[1]首次提出智能合约的概念并给出了明确的定义。他认为智能合约不仅是以数字化形式定义的承诺，而且是由计算机执行合约条款的交易协议。区块链技术的出现赋予了智能合约新的含义。区块链是分布式账本技术的典型应用，能够对交易双方的交易数据进行有效、可验证和永久性的记录，具有去中心化、可编程、集体维护和安全可信等特点。区块链智能合约是一种能够自执行、自强制、自验证和自约束其指令执行的计算机协议，应该具有一致性、自强制性、可验证性、可接入等属性[2]，其中一致性是智能合约最关键的属性，它描述了法律合约与智能合约是否一致。因为，智能合约使用计算机程序代码表示法律合约条款，是法律合约的一种载体。智能合约根据其所表示的法律合约进行自执行和自强制约束，故其一致性决定了智能合约的实际应用价值。

从计算法律学的角度，2016 年，赵精武等[3]对法律合约的本体进行分析，但是并没有关注一致性；2017 年，周雪峰等[4]从法律规则的代码化和代码规则的法律控制两个方面，对一致性进行了初步的探讨。此外，近几年，对法律合约与智能合约的一致性研究取得较大进展，特别是在生成与法律合约一致的智能合约代码方面：基于形式化模型生成智能合约代码，2016 年，Frantz 等[5]提出一种把 ADICO 结构表示的法律合约半自动化转换成智能合约 Solidity 代码框架的方法；2017 年，Mavridou A 等[6]提出了一个可将法律合约的 FSM 模型自动转换为智能合约的 Solidity 代码框架；同年，Luciano García-Bañuelos 等[7]提出一种把法律合约的 BPMN 模型映射为智能合约 Solidity 代码的方法；2018 年，Choudhury 等[8]基于法律合约的本体和语义规则提出了一个自动生成智能合约代码的框架；2020 年，Zupan 等[9]提出了一种将法律合约的 Petri 网生成智能合约的方法和可视化的原型开发工具；基于合约模板生成智能合约代码，2016 年，D. Clack 等[10~12]基于 Ricardian Contract 三要素，设计了具有法律效力的智能合约模板，建立法律合约的操作参数与标准化代码映射关系，实现智能合约代码的生成；CommonAccord 项目[13]开发了一个基于 Ricardian Contract 的智能合约模板系统，自动生成智能合约代码；G.Dos Reis 等[14]根据法律合约内容设计了 C++程序语言模板；OpenLaw[15]开发了一个使用特殊标记语言的法律合约模板库，用于调用区块链平台的智能合约代码。

因此，有必要从计算机科学的角度，对法律合约和智能合约一致性的最新研究成果进行全面的总结和分析。本文首先分别对法律合约描述语言和智能合约开发语言进行了分类

收稿日期：2019-12-09；**修回日期：**2020-02-21 **基金项目：**湖北省自然科学基金资助项目(2018ADC150)；中央高校基本科研业务费专项基金资助项目(CZT20013，QSZ17007)

作者简介：孟博(1974-)，男，教授，博士，主要研究方向为网络空间安全、区块链(mengscuec@gmail.com)；刘琴(1990-)，女，湖北黄冈人，硕士研究生，主要研究方向为信息安全、智能合约(liuqin8725@163.com).

和分析，然后对法律合约和智能合约的一致性内涵进行了分析和讨论，接着总结并讨论了现阶段由法律合约生成智能合约代码的方法和关键技术，最后对未来法律合约与智能合约的一致性研究方向进行了展望。

1 法律合约描述语言和智能合约开发语言

1.1 法律合约描述语言

法律合约不仅是具有法律约束力，而且是可由法庭强制执行的协议或文件[16, 17]。法律合约可以在特定领域内规范合约主体行为，涵盖义务和权利，如金融领域的 ISDA 协议。法律合约描述语言可以分为：面向通用领域的和面向特定领域的语言。法律合约通用领域描述语言对各种领域的法律法规进行描述，生成法律合约，如 Mercury[18]、FCL[19, 20]、Legalese 项目的 L4 语言[21]等。法律合约特定领域描述语言对专用领域的法律合约进行描述，生成法律合约，如面向商业领域[21~26]、面向金融领域 [27, 28]、面向媒体领域的 MPEG-22 CEL[29]等。此外，还有法律合约本体语言，分为通用本体语言和领域本体语言[30]。通用本体语言包括:LLD, NOR, LFU, FBO, LRI-Core, IKF-IF-LEX 等[31]，能捕捉法律中的代理、角色、意图、文档、权利、责任等概念。领域本体语言如 MPEG 媒体契约本体(MCO)[29]，用于描述处理有关多媒体资产和知识产权内容的权利；金融领域本体 FIRO[32]，FIBO[33]，用于帮助实现对金融领域合约的监管等。

1.1.1 法律合约通用领域描述语言

法律合约通用领域描述语言可以对各种领域的法律法规进行描述，生成法律合约。2013 年，OMG 组织发布了用于描述法律规则的 SBVR[34]语义规范，但不能描述法律规则的逻辑独特性，即法律规则定义的条件与特定事件相关性。2016 年，Al Khalil 等[35]利用 FIRO 框架实现 SBVR 到 OWL 的映射，实现了法律规则的逻辑表达。但是这种逻辑表达过于冗长和复杂，无法提取逻辑公式，即无法确定事件或操作与规则语句的相关性。同年，Al Khalil 等[18]通过引入动词概念“条件”来丰富 SBVR 的语义，提出了一种以 XML 语言表示的法律中间语言 Mercury，它能够以可读的方式翻译法规并进行合规性检查，简化了法律合约的逻辑表达，但 Mercury 也仅局限于规则表示，无法基于规则实现推理。

2016 年，Frantz 等[5]基于一个能够捕捉法律制度基本特征的语法[36]，将人类可读的制度规则、规章和法律的可解释合同分解为符合 ADICO 结构的语句。它通过使用逻辑连接词建立语句之间的连接，把法律合约映射为智能合约代码结构，但是该语言只能捕获法律的基本制度特征，不能处理较复杂语句之间的逻辑关系。同年，Hu[19]等提出了一个基于简单类型理论的语义框架来描述合同，它是 Governatori[20]等提出的形式化合同语言 FCL(formal contract language，FCL)的拓展。FCL 的本质是基于义务和禁止的一元道义逻辑操作符，由表示合同的状态变量和事件的形式化符号组成。Hu 等强调 FCL 是双方基于时间的承诺，能够避免出现结果和预期不一致的情况，并且能够形式化其他基于时间履约的法律合同。同时，开发了一个面向 FCL 的可以解释合同履行或违反的含义的推理系统。该系统可以推理出合同是否随着时间的推移而履行或违约，验证了 FCL 强大的建模能力。但是 FCL 不能确定合同中的协议是否与合同的相关法律法规相冲突，同时也没有提供 FCL 或推理系统的完整实现。

2018 年，由 LEGALESE.COM 机构发起的 Legalese 项目[21]，从法律合约的基本原理出发，计划提出一种面向程序员的描述法律合约的特定领域语言 L4，用于表示法律合约的特性、语义、道义和逻辑。斯坦福大学组织[37]致力于提出一种合同解释语言，实现计算机可执行的代码合同和法律合约术语的统一表达。

1.1.2 法律合约特定领域描述语言

法律合约特定领域描述语言可以对专用领域的法律合约进行描述，生成法律合约。目前法律合约的特定领域语言主要包含面向商业领域的合同语言、面向金融领域的合同语言和面向媒体领域的合约语言。

1)面向商业领域的合同语言

商业领域合同主要用于企业内部的业务管理、企业之间的业务交互的商业合同，可以帮助企业实现业务流程的管理与监督。面向商业领域的合同语言主要描述商业领域合同的异常行为[26]、业务流程[22, 23]、业务监视[24]、服务管理[25]等方面。

在描述合约的异常行为方面，2003 年，Grosof 等[26]提出了基于 SweetDeal 规则的合约语言来处理商业代理合同的复杂异常行为，比如延迟交付或不付款等情况。

在描述合约的业务流程方面，2004 年，IBM 公司[22]首先提出了可用于构建和相互交互的业务合同的 BPMN 1.0 规范。它提供了一组面向业务用户的符号，帮助不同用户设计、管理和实现业务流程。2006 年，Governatori 等人[23] 将基于 BPMN 模型描述的业务流程与基于逻辑形式语言 FCL 描述的业务合同进行行为比较，验证了 BPMN 模型与业务合同的一致性。

在描述合约的业务监控方面，为表达和监控业务合同中陈述的行为条件，2004 年，Linington 等[24]基于社区模型，设计了面向企业开发了商业合同语言(business contract language，BCL)，用于描述业务组织中复杂的结构和交互，监控企业内贸易伙伴的行为及其服务。

在描述合约的服务管理方面，2017 年，Griffo 等[25]基于 UFO-S 服务核心本体和 UFO-L 法律核心本体，提出了面向企业的服务合约本体，对不能表达服务关系中权利和义务的 ArchiMate 语言进行建模，用于表示企业内商业环境中服务合约的相关法律关系。

2)面向金融领域的合同语言

金融合同是合约当事人之间用于规定未来资产转移的协议。最初面向金融领域的合同是在 1990 年代由 Grigg 提出的 Ricardian Contract[38, 39]，本质上是一种定义两方或多方之间交互的条款和条件的数字文档。Ricardian Contract 制定了一种规范化的合约标准，对发行人和持有人的权利和责任描述，并且通过应用基于哈希值的安全标志，避免合约被单方面窜改或撕毁，保护了较弱一方的权利和权益。2015 年，Patrick Bahr 等[28]提出了一种可适用于金融业务多方的合同语言，该语言本质是一组基于时间的符号，并使用一个框架对合同进行风险分析和管理，降低金融合同构建的风险，但是不能处理永久金融合同，如无期债券。2017 年，Flood 等人[27]使用确定性状态机(deterministic finite automaton，DFA)的状态转换网络的可视化图、5 元组元素的矩阵和正则表达式等三种标准形式，来描述金融领域法律合同文本中不同的事件序列触发交易关系中特定的状态转换序列的规则，不仅可以确定合约内部的一致性、验证特定事件字母表的完整性，还克服了用自然语言表示的法律逻辑时的测试难、操作难等问题。但 DFA 的表达能力有限，一些小而重要的事件可能难以表达。同年，Al Khalil 等[40, 41]提出一种利用金融领域本体 FIRO 和 FIBO 将法律合约表示为人机可读的受控的法律自然语言，进而转换为智能合约的模型的方法，以提高智能合约的法律可信度。

1.1.3 分析和比较

对于法律合约与智能合约的一致性研究而言，法律合约的语言研究是基础。表 1 从语言的领域、功能特点、表达能力和局限性等方面总结了主要的法律合约描述语言，并进行了分析和比较。

由表 1 分析可知：法律合约描述语言发展已经比较成熟，目前已有多个面向通过领域或者特定领域的法律合约描述语言，由于智能合约是一种计算机程序，对合约条款的执行本质是程序代码的执行，程序代码的执行过程代表了法律条款的执行过程，所以，对于描述法律逻辑规则上表达能力较强的法律合约语言，如 DFA、BPMN、FCL、MPEG-22 CEL 和相关本体等可描述法律合约逻辑的语言，更适合分析和验证其与智能合约的一致性。

表 1　法律合约描述语言分析

Tab. 1　Analysis of legal contract description language

分类	领域	法律合约语言	功能特点	表达能力	局限性
面向通用领域	通用领域	Mercury[18]	法律规则逻辑表达式简单，可用于构建法律知识库	中	不具备基于规则的推理能力
		FCL[19, 20]	使用基于时间的逻辑操作符表达合约的底层逻辑，支持表达多种类型的合约；可利用工具检查合同行为是否被执行、是否履行或者违反等情况	强	合约细节表达存在不足
		ADICO 结构[5]	能够捕捉合约制度基本特征，表达简单的逻辑关系	中	对较复杂语句之间的逻辑关系表达存在局限性
		L4[21]	可以表达法律合约的特性、语义、道义和逻辑	强	缺乏基于时间的合约逻辑表达
		LLD 等通用核心本体[31]	能捕捉法律中的角色、意图、权利、责任等概念	强	概念之间的关系较宽泛
		BPMN[22]	采用形式化符号建模合约的业务流程和合约逻辑关系	强	对概念之间的关系表达存在局限性
	商业领域	BCL[24]	监控企业内贸易伙伴对合同的履行行为及其服务	强	不适合建模合约逻辑推理过程
		服务合约本体[25]	将法律责任类型分为两种:1)合法权利(权利、许可、权力、豁免); 2)法律负担/缺乏(责任、无权、服从和违反)；建模企业商业环境中服务合约相关法律关系	弱	只能表达合同中部分元素间法律关系
		基于 SweetDeal 本体[26]	建模商业合同中的复杂异常行为	中	只能表达合同中部分元素间法律关系
面向特定领域		Ricardian Contract[38, 39]	全面详细地描述法律法规	中	不能表达合约内的逻辑推理
	金融领域	DFA[27]	描述法律合同文本的状态转换序列的规则	强	对合约的表达能力有限；对于概念之间的关系表达较弱
		FIBO 本体[33]	形式化、标准化的金融法律概念模型；支持分类、查询和推理等；对法律条文的语义分析、语义推理	强	局限于对金融领域概念及关系的表达
		受控的法律自然语言[40, 41]	信任度高、兼容性和扩张性好、人机可读	弱	无具体实现
	媒体领域	MPEG-22 CEL[29]	一种面向数字媒体领域的 xml 语言，以道义结构表达合约核心元素；定义了扩展机制，可增加新的元素	强	局限于数字媒体领域

1.2　智能合约开发语言

智能合约是指能够自动执行合约条款的计算机程序，是以代码的形式存在于各平台或应用程序中。在区块链出现以前，缺乏安全可靠的执行环境，智能合约没有进行大规模的应用。但是，伴随着区块链技术的发展，智能合约及其开发语言也得到了快速发展，如，Ivy 语言[42]是一种应用于 Bitcoin 平台的高级编程语言，解决了低级语言 Bitcoin 的难编写、难理解等问题；Go 语言[43]是一种安全性较高、可快速编译的语言，因与 Hyperledger Fabric 区块链的兼容性最好，已成为 Hyperledger 项目[44]中智能合约主要开发语言；Solidity 语言[45]是为解决 Bitcoin 的局限性、专为开发智能合约的一种面向合约的高级编程语言，具有高灵活性和高拓展性，主要应用于 Ethereum[46]中；Move 语言[47]，具有比 Solidity 语言更高的安全性，是 Libra 项目[48]采用的智能合约语言。其他智能合约开发语言，如 c++、java、python 等，由于其发展成熟，功能相对稳定，也已成为部分区块链项目的首选智能合约语言，如 Javascript 语言被应用于 Lisk 项目[49]，c、C++、Java 、Python 等多种语言可以应用于 Neo 区块链项目[50]。下面主要介绍目前主流的智能合约开发语言，分析总结了每个语言的特点，最后给出了研究法律合约与智能合约一致性的语言选择建议。

1.2.1 Go 语言

Go 是一种静态强类型、编译型和并发型的编程语言，语法类似于 C 语言，具有比典型的面向对象语言更轻量级、可快速编译等特点，结合了解释语言的灵活性、动态类型语言的开发效率和静态类型的安全性，目前也已广泛应用于区块链智能合约的开发：以太坊的 geth 客户端是基于 Go 语言编写的，用户可通过 geth 实现与以太坊智能合约的交互；在 Fabric 中，由于其 Docker 容器本身是用 Go 语言开发的，使 Go 语言成为 Fabric 区块链中兼容性最好、稳定性最强的智能合约开发语言。但由于 Go 语言缺少开发框架，对开发者不够友好，增加了合约的开发难度。尽管如此，Go 语言由于其强大的功能和高安全性，依然是未来智能合约的主要开发语言之一。

1.2.2 Solidity 语言

Solidity 语言是专为实现智能合约而设计的面向合约的高级编程语言，是目前以太坊中智能合约的主要开发语言，语法类似于 JavaScript，具有易编写、适用面广等优点。智能合约最终被编码为所有客户端中嵌入的以太坊虚拟机(EVM)的字节码，并以 EVM 字节码的形式运行在区块链平台中，EVM 提供了合约间相互调用的能力，增加了智能合约的灵活性。同时，Solidity 语言有强大的异常回滚处理机制，即合约在运行过程中一旦出现异常，所有的执行都将会被回撤，避免了数据不一致，保证了合约执行的原子性。但是，Solidity 语言开发的安全性较弱。首先，由于基于以太坊的项目大多是闭源的，很多可能引发账户资金安全问题的智能合约在部署之前难以被发现。其次，Solidity 语言本身也存在一些容易引起漏洞的特点，比如 Solidity 的多指针指向同一资源的特点可能带来安全漏洞。同时，以太坊中的智能合约的高灵活性也给智能合约的未来执行带来了高不确定性，这些都成为

以太坊智能合约向未来继续发展的核心阻碍。

1.2.3 Move 语言

2019 年，Facebook 推出了区块链数字货币项目 Libra[48]，并为此专门设计了智能合约 Move 语言，类似编程语言 Rust。Libra 是以区块链为基础、有真实资产担保、有独立协会治理的全球货币，未来可应用于更广阔的金融领域，对智能合约的安全性要求更高。在智能合约的编写上，Move 语言具有比 Solidity 语言更高的安全性。首先，由于 Move 语言采用的是安全性更高的是静态类型系统，能够保证很多编程的低级错误都可以在编译时而不是运行期间发现。其次，Move 语言在其数字资产的操作机制上，数字资产只能被消耗，即从一个账户转移到另一个账户，不允许复制资源，可防止资产意外重复或丢失的情况，同时也杜绝了以太坊合约中出现的各种合约记账的错误与漏洞。最后，Move 是一种可以直接在 Move 虚拟机中运行的字节码语言，拒绝执行未通过字节码验证的程序，可避免出现类似 Solidity 语言的安全漏洞。

1.2.4 分析和比较

表 2 从语言类型、适用平台、开发工具和语言的优缺点等方面总结目前主流的智能合约开发语言，并进行了分析和比较。

表 2　不同区块链平台的智能合约开发语言分析

Tab. 2　Analysis of smart contract development languages for different blockchain platforms

智能合约开发语言	语言类型	适用区块链平台	开发工具	优点	缺点
Ivy[42]	命令式	Bitcoin	Ivy Playground	相比 Bitcoin Script 而言，编写合约更容易；安全性高	只能编写用户间转账交易
Go[43]	命令式/声明式	Fabric/Neo	LiteIDE/ VSCode 等	安全性较高；快速编译；轻量级；灵活性高；语言发展成熟	相较于 Solidity 语言，开发难度较高；
Solidity[45]	命令式/声明式	Ethereum	Remix/JIDE 等	代码简单易编写；合约间可相互调用，灵活性高，适用面广；	安全性一般；不确定性高；
Move[47]	命令式/声明式	Libra	ChainIDE/VSCode 等	安全性高；成本低；可实现自定义交易	已发布 2019 年 9 月 26 日修订版，仍在发展中
其他 C/C++/ java/JavaScript /Python	命令式/声明式	Fabric /Neo(全部语言)，Corda(java)、Lisk(JavaScript)	VSCode/Ontology Wasm/ SmartX 等	语言发展相对成熟；对开发者更友好	支持的区块链平台有限

由表 2 分析可知：a)Go 语言编写方式更灵活，Solidity 适用面最广，Move 语言安全性最高，可根据不同的需求来选择不同的语言编写智能合约；b)由于命令式语言能够直接声明为履行合约而执行的计算操作，因此大部分的智能合约开发语言是命令式语言。但 Governatori 等[51]等对采用命令式语言和声明式语言开发智能合约，从法律效力、法律解释和合同生命周期等方面对其表达能力进行分析后指出，在适应法律合约的基本元素方面，基于逻辑的声明式语言更具优势，不仅能够去除法律合约的实际条款与其执行之间的解释差距，还可以更好地处理合同生命周期的七个阶段，如 Go 语言、Solidity 语言、Move 语言、Python 等基于逻辑的声明式语言，适合智能合约开发。

2　法律合约与智能合约的一致性含义

法律合约与智能合约的一致性的内涵与合同的自动化执行的发展阶段紧密相关。2018 年，Governatori 等[51]认为合同的自动化执行经历了三个阶段：电子合同、智能合约和区块链智能合约。下面从这三个阶段来分析和讨论法律合约与智能合约的一致性。

a)在电子合同阶段，确定了电子合同的法律地位，为智能合约的实现提供了法律基础。电子合同主要描述包含可计算部分(如数据字段、规则等)的合约，而这些可计算部分可依靠第三方软件实现合同的自动化操作，如合同的起草、谈判、监视和执行[51]，Ricardian Contract[38]就是金融领域电子合同的典型应用。我国的新《合同法》中，把合同形式由纸质扩大至电子数据形式，使得电子合同具有和传统合同同等的法律地位和法律效力。随着电子合同的合约规则被应用于跨企业系统之间的计算机过程，产生了自动化的合约，如智能合约。

b)在智能合约阶段，通过将代码与法律融合，实现对法律条款的自执行，并产生法律合约的义务和权利等结果。Nick Szabo 强调智能合约是建立在交易双方已达成承诺协议的信用基础之上[1]，通过代码实现法律合约条款的自执行，不仅能够消除法律文本合同的模糊性，还能增强法律文本合同执行的确定性。在这个阶段的智能合约是一种计算机程序，由基于交易策略的计算机系统执行，不仅执行法律合约的某些操作，而且通过这些操作来产生执行法律的义务和权利带来的结果[52]，如自动售货机就是智能合约的简单应用。但由于缺乏可信任的平台，智能合约的发展受到了制约。

c)在区块链智能合约阶段，依托于可信任的区块链平台，智能合约不仅在技术上实现法律合约条款的自执行与自监督，而且在法律上可作为法律合约的补充或替代，是法律合约在软件中表达和实现[12,53]。2016 年，Savelyev[52]结合区块链资产，认为智能合约处理当事人之间的数字资产流通相关的经济关系。数字区块链资产是合同的标的物，是合同重要构成部分。2017 年，Modi [54]认为智能合约是基于区块链技术对法律合约逻辑的具体体现，它不仅可以存储数据，记录现实法律合约逻辑所需的任何其他信息，而且能够通过代码对某些法律条款预先定义触发条件。当条件满足时，可保证条款的自执行性和自发性。同年，R3 和 Norton Rose Fulbright[55]讨论了在多个主要司法管辖区的法律下，构成具有法律约束力的合同的可行性。2018 年，在 ISDA Linklaters 组织[56]发布的白皮书详细分析了智能合约在合同法的约束下，应用于不同的场景时与法律有效融合的策略，确保智能合约法律效率最大化。2019 年，郭京飞[57]和陈吉栋[58]从法律的要约和承诺的结构方面，对区块链智能合约的法律内涵进行了分析。2020 年，Omololu[59]从法律制定的必要性角度分析了区块链技术与法律之间的关系，认为需要合适的法律制度以处理区块链智能合约等技术在出现安全问题后的法律纠纷。

智能合约的法律内涵是法律合约与智能合约的一致性实现基础。法律合约与智能合约的一致性本质是智能合约代码与法律合约的一致性。2016 年，胡凯等[2]认为可以基于形式化方法分析并验证已有的智能合约代码与法律合约的相关属性和执行力的一致性，但是他们既没有给出应用形式化方法分析相关属性和执行力的具体方法，又没有涉及生成与法律合约一致的智能合约代码方面的工作。2017 年，Kantara[60]从

法律合约的视角，探讨了智能合约的可执行性，建议基于法律合约的语义，把法律合约转换为等价的、可执行的智能合约代码，分析并总结了生成与法律合约一致的智能合约代码的方法和所面临的挑战。同年，周学峰等[4]从新计算法律学层面，强调要对法律进行动态分析，认为法律实现代码化，即从法律合约生成智能合约代码，把法律的内在逻辑推理转换为程序代码，进而实现利用程序代码执行法律合约，最终代替或帮助法官对案件进行审判。同年，赵精武等[3]简单讨论了现实世界的“法律(law)”与代码世界的“法律(code)”一致性，提出了法律合约转换为智能合约代码的框架，即建立符合法律要求的本体库、形式化建模、对法律本体和语义进行验证和修正、通过区块链记录整个过程数据。以上虽然都提出了法律合约和智能合约代码的一致性内涵，但是它们没有关注如何分析已有的智能合约代码与法律合约的一致性及如何生成与法律合约一致的智能合约代码的方面的工作。

作者认为，法律合约和智能合约的一致性关注合约规则与智能合约的逻辑顺序/执行过程一致性，以及执行产生的承诺结果的法律权利和义务关系一致性。

3　一致性智能合约代码生成

一致性智能合约代码生成是指根据法律合约生成一致性的智能合约代码，主要有两种方法：基于形式化模型和基于智能合约模板。下面对这两种方法进行了详细的说明与分析，并对其一致性研究成果进行了评价。

3.1　基于形式化模型生成智能合约代码

基于法律合约的形式化模型生成智能合约代码，是应用法律合约描述语言对法律合约进行形式化建模，然后根据形式化模型和法律合约形式化描述语言与智能合约开发语言的映射关系生成智能合约代码。具体是，首先对自然语言表示的法律合约进行分析，构建法律合约形式化描述语言，并进行形式化建模形成法律合约的形式化模型，然后从模型中抽取出关于法律合约中的实体联系、逻辑规则等权利和义务的信息，建立法律合约形式化语言与智能合约开发语言的映射模型和具体的语句映射关系，最后基于法律合约的形式化模型，生成智能合约代码。主要有基于 ADICO 结构、基于有限状态机、基于 BPMN 模型、基于本体和语义规则、基于 Petri 网模型等五种生成智能合约代码的方法。基于法律合约的形式模型生成智能合约代码如图 1 所示。

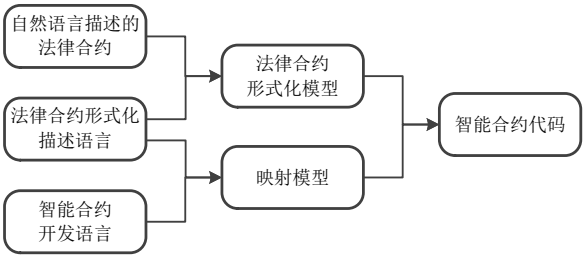

图 1 基于形式化模型生成智能合约代码结构图

Fig. 1　Structure diagram of smart contract code generation based on formal model

3.1.1 基于 ADICO 结构的形式化语言

2016 年，Frantz 等[5]提出了一种将人机可读的合同半自动翻译为 Ethereum 上智能合约的方法。首先提出了一个领域建模语言 DSL(Domain Specific language)：ADICO 结构，然后建立 ADICO 与 Solidity 的语句映射关系，最后根据此映射关系，使用 Scala 工具，将自然语言表示的法律合约半自动翻译为 Ethereum 智能合约 Solidity 代码框架。具体映射规则是：A 表示合约的参与成员，对应 Solidity 代码中参与方的结构成员；D 表示合约的义务、许可或禁止等道义逻辑，对应 Solidity 代码中执行合约时的逻辑判断符号，如“禁止”使用逻辑符号非表示，通常在函数修饰符中；I 表示某个机构声明中控制的动作，更深层次的是对合约的操作行为指定相关的对象和目标，可对应 Solidity 相应的事件函数，也可以对应函数中的结果参数；C 表示合约的动作执行条件，对应 Solidity 代码中合约执行的条件判断函数，如 if 函数，通常在函数修饰符中；O 表示异常结果，对应 Solidity 代码中的异常命令，如 throw。该方法虽然使具有不同技术背景的人编写智能合约更加容易，但是由于生成的只是智能合约代码框架，后期仍然需要大量的手工编写后才能执行。同时，缺乏对 DSL 映射的关系一致性验证。

3.1.2 基于有限状态机的形式化语言

2017 年，Mavrido 等[6]提出了一个基于操作语义的 FSolidM 框架，并以盲拍合同为例，首先通过该框架的图形编辑器编辑合约的 FSM 模型，指定合约的状态、转换、转换条件等，并定义合约的全部输入，然后根据该模型自动生成相应的高安全性的智能合约 Solidity 代码，并可直接部署执行，大大提高了智能合约的开发效率。在代码生成过程中，FSM 模型中的每一个转换都会生成一个 Solidity 函数，其中转换条件和动作构成函数的主体，输入和输出分别对应函数参数和返回值。同时，该框架也提供了一组安全的插件，不仅可以防止某些常见的安全漏洞，而且支持开发复杂智能合约需要的通用设计模式，减少了法律合约执行语义假设和智能合约实际语义之间的差异产生的漏洞，提高了智能合约 Solidity 代码的安全性。FSolidM 框架不仅提供了 FSM 模型到 Solidity 代码的生成机制，而且对 FSM 模型的正确性，以及转换后 Solidity 代码的安全性和语法的正确性都进行了检查。但是，由于 FSM 建模能力有限，需要考虑状态和转换的爆炸问题，适用于简单的智能合约。同时，映射的内容只包含存在状态转换关系的法律合约内容，缺乏对合约中的法律法规的描述，削弱了合约中基于规则的推理功能。最后，FSM 框架虽然提供了图形视图，但图形元素与法律合约的形式化定义之间缺乏必要的一致性验证，建模过程也属于自由编辑状态，对模型的可靠性和一致性保证不足。

3.1.3 基于 BPMN 模型的形式化语言

2017 年，García-Bañuelos 等[7]基于区块链技术，首先给出 BPMN 的核心子集 core BPMN，使用预先定义的基于状态和函数转换的操作语义变换规则，以贷款申请业务为例，将 core BPMN 表示的贷款申请流程转换成 Petri 网，同时通过数据流分析方法以确定约束每个任务执行条件的位置。然后，简化 Petri 网，以消除不可见的转换和错误的地方，并使用基于数据流分析提取的元数据对简化后的小型 Petri 网进行注释。最后，将小型 Petri 网表示的业务流程生成以太坊中可执行的智能合约 Solidity 代码。该方法的特点是通过把流程模型的当前状态转换为一个特殊的数据结构，再进行空间优化，从而减少执行智能合约时的 gas 消耗。但该方法在生成的代码与法律合约的一致性表示方面存在不足。首先，它侧重于对 Petri 网中的控制流关系和条件生成代码，忽略了协作中的各方如何关联流程实例和访问控制问题，表达内容有限；其次，它侧重于对 BPMN 的“核心子集”生成代码，不包括计时器事件、子流程和边界事件，但这些内容都是法律合约的关键内容。

3.1.4 基于本体和语义规则的形式化语言

2018 年，Choudhury 等[8]通过使用本体和语义规则对以交易为中心的特定领域知识进行编码，提出了一个自动生成智能合约的框架。首先设计特定领域的本体和语义规则，分别表示底层的系统和约束，形成基于知识的框架；然后基于此设计一个可包含本体所有可能的表现形式的智能合约模板，

并将合约模板转换为一个抽象语法树(AST)；接着在本体实例化后，通过特定的程序遍历 AST，将本体所派生的属性和关系的约束插入到 AST 中；最后基于修改后的 AST 生成智能合约代码，且该代码封装了所有可能发生的事务的规则和流程。该方法的核心点在于需要领域专家来设计本体和语义规则，需要法律人员和程序人员的共同参与，同时，该方法原则上可支持不同的区块链平台和不同的智能合约开发语言。但是该方法对法律合约模板与 AST 之间的转换、AST 与智能合约代码之间的转换都缺乏一致性验证。

3.1.5 基于 Petri 网模型的形式化语言

2020 年，Zupan 等[9]提出了一种基于 Petri 网的智能合约生成方法和可视化的原型开发工具。具体是，首先使用 Petri 网构建合约的工作流模型，然后通过不同的引擎工具完成模型中各工作流的验证以及各功能到智能合约代码的转换工作，最后在系统外通过对其进行人工干预，包括开发人员继续完善生成的智能合约代码和审计专家对代码的审查，最终形成可直接部署执行的智能合约代码。该工具不仅支持 Petri 网模型的可视化创建、导入和导出，增加了操作上的便捷性，同时严格的模型验证机制保证了生成的合约代码的安全性，且专为独立于平台的区块链系统而设计，原则上支持所有的智能合约开发语言.但是该方法只能生成智能合约框架，同时该模型在与法律合约的一致性表达方面也存在不足，仅能粗略地表示法律合约的逻辑内容，对于合约逻辑的细节表达后期仍需要人工干预。

总体来看，基于法律合约形式化模型生成智能合约代码，普遍存在 2 个问题：

a)缺乏对映射模型或者相应的语义转换关系的一致性验证；

b)模型映射的结果局限于某种特殊语言，可移植性弱。

3.2　基于合约模板生成智能合约代码

基于合约模板生成智能合约代码，首先基于多个法律合约形成法律合约模板，模板实例化后可得到具体的法律合约，模板代码化形成智能合约代码框架，然后通过将法律合约的参数传递给智能合约代码框架，生成智能合约代码。通过合约模板将智能合约与现实世界的法律合同建立了联系，在简化智能合约代码生成的步骤的同时，实现法律合约与智能合约代码的一致性。合约模板主要包含：基于 Ricardian Contract 的合约模板[12~15]、基于 C++语言的合约模板[14]和基于法律标记语言的合约模板[15]。基于智能合约模板实现法律合约和智能合约的一致性原理结构图如图 2 所示。

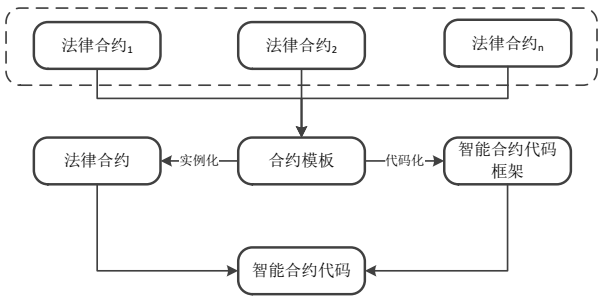

图 2　基于合约模板生成一致性的智能合约代码的原理

Fig. 2　Schematic diagram of smart contract code generating consistency based on contract template

3.2.1 基于 Ricardian Contract 的合约模板

Ricardian Contract[38, 39]应用数字签名的三要素<P;C;M>表示 prose，code 和 parameters，其中 P 是法律条款，使用指称语义捕获合同的法律意义，C 是平台上的特定代码，M 是 P 和 C 中的参数映射，即键值对，该参数建立了代码和法律合约之间的联系。基于 Ricardian Contract 的法律合约模板，首先对标准机构创建的法律文件形成模板化，再将模板中的相关操作参数传递给智能合约代码，以支持智能合约代码的合法执行。尽管 Ricardian Contract 最初是为发行金融工具而提出的，但它的思想对现在仍然有借鉴意义，可以为智能合约模板创建提供基础，且已被证明可应用于分布式分类账和智能合约等新型经济环境。

2016 年 8 月，D. Clack 等人[10~12]在 Ricardian Contract 的三要素基础上，提出基于智能合约模板生成与法律合约一致的智能合约代码的方法，如图 3 所示。

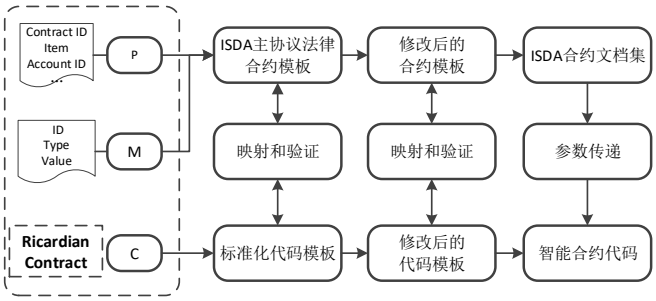

图 3　Clack 等人的基于 Ricardian Contract 的智能合约模板结构图

Fig. 3　Clack et al. 's structure diagram of smart contract template based on ricardian contract

首先，基于 Ricardian Contract 中的 prose、parameters 两个要素形成合约模板，并应用于 ISDA 主协议法律合约中，形成 ISDA 主协议法律合约模板，同时对 Ricardian Contract 中 code 要素形成合约的标准化代码模板。其次，依据 ISDA 主协议法律合约模板和标准化代码模板，跟据实际法律合约要求对其模板进行不断完善，并对两种模板之间的映射关系进行验证，逐步形成实例化后 ISDA 协议法律合约文档集和智能合约代码模板。最后，将 ISDA 协议法律合约文档集中的具体参数传递给智能合约代码模板，生成最终的智能合约代码。Clack 等认为，可通过基于智能合约模板实现法律合约和智能合约代码的生成过程的一致性，来保证智能合约代码遵循法律合约，即保证法律合约和智能合约代码的一致性。

CommonAccord 项目[13]旨在基于 Ricardian Contract 方法，通过编纂和自动化法律文件来创建全球法律交易代码。该项目计划实现一个基于 Ricardian Contract 的智能合约模板系统。该系统利用一个可将法律合约编码为基于标准文本对象的简单数据模型，将法律合约的可计算部分封装成各类小型的技术对象，以供代码调用，同时代码也能匹配与之对应的法律文档，实现法律合约和智能合约代码的双向映射，实现智能合约代码的生成。用户可在模板系统中搜索相应的法律条款，然后组成一个完整的法律合约。由于每选择一个法律条款都会自动生成相应的代码，在创建整个法律合约的过程中，自动生成对应智能合约代码。同时，可将该模板系统连接到各种软件系统和平台，不仅可应用于分布式平台，未来也可以运用于多平台，与各种软件实现交互，拓展了基于 Ricardian Contract 智能合约模板的适用范围。

但由于法律合约结构的复杂性，基于 Ricardian Contract 实现完善的合约模板难度较大，尤其是对于如何表述法律合约的非操作语义部分、如何处理模糊性法律术语等仍然是目前面临的主要难题，但是它对未来基于 Ricardian Contract 的智能合约模板促进法律合约和智能合约代码的一致性研究提供了一个切实可行的思路。

3.2.2 基于 C++语言的合约模板

2016 年，Dos Rei 等[14]使用 C++语言设计了法律合约开发语言模板。该模板通过一种映射机制，把指定合同中的前提条件、后置条件和断言等规则用 C++语言描述，并制定了相应的转换规则。但是由于 C++不属于目前智能合约的主流

开发语言之一，使该模板的使用范围有限。此外，对于该模板的映射机制也并没有进行有效的验证，且该模板是否能够正确表达法律内涵还未得到证明。

3.2.3 基于法律标记语言的合约模板

OpenLaw 项目[15]旨在实现对智能合约代码实现一致性调用。首先，利用 OpenLaw 特有的标记语言将自然语言表示的法律协议转换成机器可读的对象，并在文档中定义相关的变量和逻辑，形成一个法律协议模板库。然后，利用转换工具将标记语言表示的法律合约模板转换成基于 HTML 文件的模板，用于嵌入到 JAVA 语言或 Scala 语言编写的项目中，实现对智能合约的调用。法律从业人员在使用过程中，可使用标记语言对法律合约模板进行修改，以创建和管理法律文件的执行，同时还可以通过将相应的参数传递给嵌入其中的基于以太坊的智能合约中，保证智能合约是按照预先签署的法律合约指令要求和顺序执行来执行，实现法律合约与智能合约代码执行的过程一致性。但是标记语言只能处理可参数化的法律内容，形成的模板也不能直接转换为智能合约代码，只能通过嵌入合约地址来保证对智能合约代码的调用，不能保证调用的合约代码就是与法律合约的内容是一致的。

对于通过代码实现对区块链智能合约的调用，2019 年，Shi 等[61]提出一种利用智能合约模板实现对区块链智能合约调用的交易模型。该模型以法律合约和智能合约模板为基础，首先由法律专家依据法律合约形成法律合约模板，提取法律合约模板的相关信息形成智能合约模板，智能合约模板包含模板代码、合约参数、智能合同模板 ID 以及其他可变内容等组成，并将两种模板以地址的形式存放于区块链平台；然后，利用第三方信息作为智能合约的触发条件，用户可通过传递不同的参数，从而达到调用不同区块链平台智能合约的效果。该方法可通过地址映射找到智能合约模板对应的法律合约模板信息，但是对于法律合约转换为法律合约模板、法律合约模板转换为智能合约模板两方面的过程并没有进行详细的描述和一致性验证。

目前，基于合约模板实现智能合约代码的生成有以下 3 个方面的问题：

a)应用于不同的平台需要不同的参数。由于存在多个智能合约执行平台，这意味着在不同的平台中使用该模板就需要不同的参数，涉及到的部分技术细节调整可能会超出律师等法律专业人士的能力范围，增加代码生成的难度。

b)对合约模板表达能力的验证难度较大。合约模板是否能完全表达法律信息，尤其是对法律合约的非操作语义部分表示的验证，仍然需要大量的工作。

c)缺乏一致性验证。如何验证通过合约模板生成智能合约代码的一致性，还有大量的工作要开展。

以上两个项目都只是实现了法律合约和区块链智能合约代码的交互，保证了相互之间可调用，并没有对法律合约进行实质性的结构分析和操作分析，形成基于某种开发语言的法律合约语言模板。

3.3 分析与比较

关于法律合约与智能合约的一致性研究成果主要是从法律合约生成智能合约代码，对其从一致性研究方法、法律合约语言和智能合约语言、方法原理、生成代码运行平台和一致性结果进行了分析和总结，见表 3。

表 3　法律合约与智能合约代码的一致性分析

Tab. 3　Consistency analysis of legal contract and smart contract code

研究方法	法律合约语言	智能合约语言	方法原理	生成代码运行平台	一致性结果
形式化模型	ADICO 结构	Solidity	应用模型映射将合同半自动化转换为 Ethereum 上的智能合约 Solidity 代码框架	仅支持运行 EVM 的 Ethereum 平台	由于建模语言不能完全表达法律合约，导致到代码的映射内容不全；缺乏对映射关系的一致性验证；生成的 solidity 代码框架不能直接运行
	BPMN	Solidity	将法律合约的 BPMN 结构转换成 Petri 网，生成 Ethereum 上可执行的智能合约 Solidity 代码	仅支持运行 EVM 的 Ethereum 平台	由于建模语言不能完全表达法律合约，导致到代码的映射内容不全；缺乏对其中的映射关系的一致性验证
	有限状态机	Solidity	利用基于语义的 FSolidM 框架，把法律合约的 FSM 模型自动生成 solidity 代码框架	仅支持运行 EVM 的 Ethereum 平台	缺乏对其中的映射关系的一致性验证；只转换为 Solidity 部分代码，不能直接运行
	本体和语义规则	各平台源语言	使用本体和语义规则对特定领域知识进行编码，通过自动生成智能合约框架来生成智能合约代码	跨平台，跨语言	缺乏对合约模板与 AST 之间的转换、AST 与智能合约代码之间的转换的一致性验证。
	Petri 网	各平台源语言	将法律合约的 Petri 网模型半自动转换为可执行的智能合约代码	跨平台，跨语言	对于合约逻辑的细节表达仍需要人工干预
	BNF-like 的语义	各平台源语言	基于 Ricardian Contract 合约模板实现生成过程的一致性	跨平台，跨语言	模板实现难度较大，项目在进行中
合约模板	自然语言	C++	将法律合约描述为 C++语言模板	跨平台	对映射机制缺乏有效的验证；项目在进行中
	法律标记语言	Solidity	应用标记语言表示法律合约，实现对区块链智能合约代码的调用	仅支持运行 EVM 的 Ethereum 平台	标记语言表示的法律合约内容有限；标记语言和智能合约代码之间的调用缺乏有效地一致性验证

从表 3 中可以总结出，目前根据法律合约生成智能合约代码方法，主要存在以下局限性：生成的智能合约语言大多都相对固定，可移植性较弱；对法律合约与智能合约代码之间的映射机制没有给出具体的一致性分析和验证。

4　结束语

法律合约与智能合约一致性保障了智能合约在当前的法律框架内的自动执行。随着智能合约的应用越来越广泛，人

们更加关注法律合约与智能合约的一致性。本文从计算机科学的角度，首先分别对法律合约描述语言的表达能力和智能合约的开发语言的特点进行了总结分析，然后根据合约自动化发展的三个阶段，总结了法律合约与智能合约的一致性内涵；最后，从形式化转换和合约模板两个方面，对将法律合约转换成智能合约代码的研究成果进行了一致性分析。作者认为法律合约与智能合约一致性实现方法包含两方面的内容，一是基于法律合约生成一致性的智能合约代码，二是验证已有的智能合约代码与法律合约的一致性。针对目前的研究现状，对于法律合约与智能合约的一致性研究未来可从以下 2 个方面考虑：

1)映射模型的一致性分析

目前，基于法律合约的形式化模型，生成相应的智能合约代码的方法，一方面，生成的代码不完善，或者是不可直接执行的代码框架，另一方面，研究者鲜少验证它们之间映射关系的一致性。曹流等[62]提出了一种基于 OCL(一种描述约束的形式化语言)的体系结构，用于分析软件建模领域的一致性，来保证最终的模型符合用户所定义的一致性约束。但是该方法不适合法律合约与智能合约代码映射模型的一致性验证。随着操作语义保留的转换、互摸拟技术、可信编译器技术的发展，为法律合约与智能合约代码映射模型的一致性验证提供了新的途径。

2)已存在的智能合约代码与法律合约的一致性分析

当下，对从法律合约的形式化模型，生成相应的智能合约代码的方法研究较多。但是对已存在的智能合约代码与法律合约的一致性分析，较少关注。而对后者，有三个方法：

a)可以应用模型抽取方法得到智能合约代码的形式化模型和法律合约的形式化模型，然后分别分析抽取的两个形式化模型的正确性，进而建立这两个模型的映射关系，最后分析映射关系的一致性，得到法律合约与智能合约的一致性分析结果；

b)可以借鉴网络协议一致性研究方法[63,64]，对其进行一致性分析。基于黑盒测试，根据输入输出测试集，利用一致性测试工具对测试集进行测试，通过控制输入、监控输出来测试并评价一致性。基于白盒测试，根据法律合约的一致性需求文档，应用抽象语法树、控制流程图分析、符号执行技术[65~67]、深度学习等对智能合约代码的执行过程进行一致性分析。

参考文献：

[1] Nick Szabo, Smart Contract: Building Blocks for Digital Markets. [EB/OL]. (1996) [2019-10-20]. http://www.fon.hum.uva.nl/rob/Courses/InformationInSpeech/CDROM/Literature/LOTwinterschool2006/szabo.best. vwh. net/smart. contracts. html.

[2] 赵精武, 丁海俊. 论代码的可规制性: 计算法律学基础与新发展 [J]. 网络法律评论, 2016, 19 (01): 97-112. (Zhao Jingwu, Ding Haijun. On the regulability of code: foundation and new development of computational law [J]. Internet Law Review, 2016, 19 (01): 97-112.)

[3] 胡凯, 白晓敏, 高灵超, 等. 智能合约的形式化验证方法 [J]. 信息安全研究, 2016, 2 (12): 1080-1089. (Hu Kai, Bai Xiaomin, Gao Lingchao, *et al.* Formal verification method of smart contract [J]. Journal of Information Security Research, 2016, 2 (12): 1080-1089.)

[4] 周学峰, 赵梓皓. 解析计算法律学 [J/OL]. 中国计算机学会通讯, 2017, 2017 (5) . [2019-10-20]. https://www. ccf. org. cn/c/2017-05-15/594989. shtml.

[5] Frantz C K, Nowostawski M. From institutions to code: Towards automated generation of smart contracts [C]// Proc of the 1th IEEE International Workshops on Foundations and Applications of Self* Systems (FAS* W) . IEEE, 2016: 210-215.

[6] Mavridou A, Laszka A. Designing secure ethereum smart contracts: A finite state machine based approach [C]// Proc of International Conference on Financial Cryptography and Data Security. Springer, Berlin, Heidelberg, 2018: 523-540.

[7] García-Bañuelos L, Ponomarev A, Dumas M, *et al.* Optimized execution of business processes on blockchain [C]// Proc of International Conference on Business Process Management. Springer, Cham, 2017: 130-146.

[8] Choudhury O, Rudolph N, Sylla I, *et al.* Auto-Generation of Smart Contracts from Domain-Specific Ontologies and Semantic Rules [C]// Proc of IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData) , Halifax, NS, Canada, 2018: 963-970.

[9] Zupan N, Kasinathan P, Cuellar J, *et al.* Secure Smart Contract Generation Based on Petri Nets [M]// Blockchain Technology for Industry 4. 0. Singapore: Springer, 2020: 73-98.

[10] Clack C D, Bakshi V A, Braine L. Smart contract templates: foundations, design landscape and research directions [J]. arXiv preprint arXiv: 1608. 00771, 2016.

[11] Clack C D, Bakshi V A, Braine L. Smart Contract Templates: essential requirements and design options [J]. arXiv preprint arXiv: 1612.04496, 2016.

[12] Clack C D. Smart Contract Templates: Legal semantics and code validation [J]. Journal of Digital Banking, 2018, 2 (4): 338-352.

[13] CommonAccord [EB/OL]. [2019-10-20]. http://www.commonaccord.org/.

[14] Dos Reis G, Garcia J D, Lakos J, *et al.* A CONTRACT DESIGN [EB/OL]. (2016-05-28) [2019-10-20]. http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2016/p0380r0. pdf.

[15] openlaw [EB/OL]. (2019-04-04) [2019-10-21]. https://docs. openlaw. io.

[16] Daskalopulu A K. Logic-based tools for the analysis and representation of legal contracts [D]. London: University of London, 1999.

[17] McKendrick E. Contract law: text, cases, and materials [M]. UK: Oxford University Press, 2014.

[18] Ceci M, Al Khalil F, O'Brien L. Making Sense of Regulations with SBVR [C]// RuleML (Supplement) . 2016.

[19] Farmer W M, Hu Q. A formal language for writing contracts [C]// Proc of the 17th IEEE International Conference on Information Reuse and Integration (IRI) . IEEE, 2016: 134-141.

[20] Governatori G, Milosevic Z. A formal analysis of a business contract language [J]. International Journal of Cooperative Information Systems, 2006, 15 (04): 659-685.

[21] Solftware is eating law [EB/OL]. (2017) [2019-10-21]. https://legalese. com/aboutus. html.

[22] Business Process Model and Notation (BPMN) , version 2. 0 [EB/OL]. (2016-07-10) [2019-10-5]. http://www. omg. org/spec/BPMN/2. 0.

[23] Governatori G, Milosevic Z, Sadiq S. Compliance checking between business processes and business contracts [C]// Proc of the 10th IEEE International Enterprise Distributed Object Computing Conference, Proceedings. IEEE, 2006: 221-232.

[24] Linington P F, Milosevic Z, Cole J, *et al.* A unified behavioural model and a contract language for extended enterprise [J]. Data & Knowledge Engineering, 2004, 51 (1): 5-29.

[25] Griffo C, Almeida J P A, Guizzardi G, *et al.* From an ontology of service contracts to contract modeling in enterprise architecture [C]// Proc of the 21th IEEE International Enterprise Distributed Object Computing Conference (EDOC) . IEEE, 2017: 40-49.

[26] Grosof B N, Poon T C. SweetDeal: representing agent contracts with exceptions using XML rules, ontologies, and process descriptions [C]// Proc of the 12th international conference on World Wide Web. ACM, 2003: 340-349.

[27] Flood M D, Goodenough O R. Contract as automaton: the computational representation of financial agreements [J]. Office of Financial Research Working Paper, 2015 (15-04) .

[28] Bahr P, Berthold J, Elsman M. Certified symbolic management of financial multi-party contracts [J]. ACM SIGPLAN Notices, 2015, 50 (9): 315-327.

[29] Rodríguez E, Delgado J, Boch L, *et al.* Media contract formalization using a standardized contract expression language [J]. IEEE multimedia, 2014, 22 (2): 64-74.

[30] 赵忠君. 国外法律本体研究综述 [J]. 情报科学, 2012, 30 (01): 149-154. (Zhao Zhongjun. A Summary of Study on Legal Ontology Abroad [J]. Information Science, 2012, 30 (01): 149-154)

[31] Benjamins V R, Contreras J, Casanovas P, *et al.* Ontologies of professional legal knowledge as the basis for intelligent it support for judges [J]. Artificial Intelligence and Law, 2006, 12 (4): 359-378.

[32] Espinoza A, Abi-Lahoud E, Butler T. Ontology-driven financial regulatory change management: an iterative development process [C]// Proc of the 2nd Semantic Web and Linked Open Data workshop (SW-LOD) . Anais. 2014.

[33] Financial Industry Business Ontology [EB/OL]. (2016) [2019-09-27]. http://ww w. edmcouncil. org/financialbusiness/.

[34] Semantics of Business Vocabulary and Business Rules (SBVR) Version 1. 2 [EB/OL]. (2013) [2019-10-15]. http://www. omg. org/spec/SBVR/1. 2/PDF.

[35] Al Khalil F, Ceci M, Yapa K, *et al.* SBVR to OWL 2 Mapping in the Domain of Legal Rules [C]// Proc of International Symposium on Rules and Rule Markup Languages for the Semantic Web. Springer, Cham, 2016: 258-266.

[36] Crawford S E S, Ostrom E. A grammar of institutions [J]. American Political Science Review, 1995, 89 (3): 582-600.

[37] CodeX [EB/OL]. [2019-10-15]. https://law. stanford. edu/codex-the-stanford-center-for-legal-informatics/.

[38] Grigg I. The Ricardian Contract [C/OL]/ Proc of the First IEEE International Workshop on Electronic Contracting. IE, 2004. [2019-10-21]. http://iang. org/papers/ricardian_contract. html.

[39] Ricardian contracts [EB/OL]. [2019-10-21]. http://www.webfunds.org/guide/ricardian. html.

[40] Al Khalil F, Butler T, O'Brien L, *et al.* Trust in smart contracts is a process, as well [C]// Proc of International Conference on Financial Cryptography and Data Security. Springer, Cham, 2017: 510-519.

[41] Al Khalil F, Ceci M, O'Brien L, *et al.* A solution for the problems of translation and transparency in smart contracts [J/OL]. Government Risk and Compliance Technology Centre, 2017. (2017-8-31) . http://www.grctc.com/wp-content/uploads/2017/06/GRCTC-Smart-Contracts-White-Paper-2017.pdf.

[42] Ivy [EB/OL]. [2019-10-21]. https://chain.com/docs/1.2/ivy-playground/ install

[43] Go [EB/OL]. [2019-10-21]. https://golang. google. cn/doc/.

[44] Hyperledger-fabric [EB/OL]. [2019-10-21]. https://hyperledger-fabric. readthedocs. io/en/release-1. 1/chaincode. html.

[45] Solidity [EB/OL]. [2019-10-21]. https://media.readthedocs.org/pdf/solidity/develop/solidity. pdf.

[46] Ethereum [EB/OL]. [2019-10-21]. https://github.com/ethereum/wiki/wiki/White-Paper.

[47] Libra [EB/OL]. [2019-10-21]. https://libra. org/zh-CN/white-paper/.

[48] Move [EB/OL]. [2019-10-21]. https://developers.libra.org/docs/move-paper.

[49] Lisk [EB/OL]. [2019-10-21]. https://lisk. io/.

[50] Neo [EB/OL]. [2019-10-21]. https://neo. org/.

[51] Governatori G, Idelberger F, Milosevic Z, *et al.* On legal contracts, imperative and declarative smart contracts, and blockchain systems [J]. Artificial Intelligence and Law, 2018, 26 (4): 377-409.

[52] Savelyev A. Contract law 2. 0: 'Smart'contracts as the beginning of the end of classic contract law [J]. Information & Communications Technology Law, 2017, 26 (2): 116-134.

[53] Stark J. Making sense of blockchain smart contracts [EB/OL]. (2016-06-07) [2019-10-30]. https://www. coindesk. com/making-sense-smart-contracts

[54] Modi R. Solidity Programming Essentials: A beginner's guide to build smart contracts for Ethereum and blockchain [M]. Packt Publishing Ltd, 2018.

[55] Murphy S, Cooper C. Can smart contracts be legally binding contracts [J/OL]. RE, 2016, 11 (17) . http://www. nortonrosefulbright. com/knowledge/publications/144559/cansmart-contracts-be-legally-binding-contracts.

[56] Hansen D, Rosini L, Reyes C. More legal aspects of smart contract applications [EB/OL]. (2018-05-29) [2019-11-10]. https://www.perkinscoie.com/images/content/1/9/v5/199672/2018-More-Legal-Aspects-of-Smart-Contract-Applications-White-Pa. pdf.

[57] 郭少飞. 区块链智能合约的合同法分析 [J]. 东方法学, 2019 (03): 4-17. (Guo Shaofei. Contract law analysis of smart contract on blockchain [J]. Oriental Law, 2019 (03): 4-17.)

[58] 陈吉栋. 智能合约的法律构造 [J]. 东方法学, 2019 (03): 18-29. (Chen Jidong. Legal structure of smart contract [J]. Oriental Law, 2019 (03): 18-29.)

[59] Omololu A A. Legal Ramifications of Blockchain Technology [M]// Decentralised Internet of Things. Cham: Springer, 2020: 217-230.

[60] Hardjono T, Maler E. Report from the Blockchain and Smart Contracts Discussion Group to the Kantara Initiative [EB/OL]. (2018) [2019-10-21]. http://hardjono. mit. edu/sites/default/files/documents/Report-from-the-Blockchain-and-Smart-Contracts-Discussion-Group-to-the-K. pdf.

[61] Shi Y, Lu Z, Tao R, *et al.* A Trading Model Based on Legal Contracts Using Smart Contract Templates [C]// Proc of International Conference on Blockchain and Trustworthy Systems. Springer, Singapore, 2019: 446-460.

[62] 曹流, 曹春. 一种基于 OCL 的体系结构一致性验证环境 [J]. 计算机科学, 2012, 39 (S3): 409-414. (Cao Liu, Cao Chun. Validation Environment of Software Architecture Based on OCL [J]. Computer Science, 2012, 39 (S3): 409-414.)

[63] 朱雪峰, 许建军, 邹彪, 等. 网络协议一致性测试研究综述 [J]. 计算机科学, 2009, 36 (12): 5-7+36. (Zhu Xuefeng, Xu Jianjun, Zhou Biao, *et al.* Network Protocol Conformance Testing: An Overview [J]. Computer Science, 2009, 36 (12): 5-7+36.)

[64] 李强, 余祥, 齐建业, 等. 协议一致性测试研究进展 [J]. 西南科技大学学报, 2013, 28 (04): 85-92. (Li Qiang, Yu Xiang, Qi Jianye, *et al.* The Headway of Protocol Conformance Testing [J]. Journal of Southwest University of Science and Technology, 2013, 28 (04): 85-92.)

[65] Luu L, Chu D H, Olickel H, *et al.* Making smart contracts smarter [C]// Proc of the 2016 ACM SIGSAC conference on computer and communications security. ACM, 2016: 254-269.

[66] Krupp J, Rossow C. Teether: Gnawing at ethereum to automatically exploit smart contracts [C]// Proc of the 27th{USENIX}Security Symposium ({USENIX}Security 18) . 2018: 1317-1333.

[67] Kalra S, Goel S, Dhawan M, *et al.* ZEUS: Analyzing Safety of Smart Contracts [C]// Proc of NDSS. 2018.