合同文本置标语言 CTML：一种面向智能法律合约的法律信息规范化提取方法

范雨晴[1)]，王　迪[2)]，林鸿杰[1)]，陈　娥[1)]✉，何　啸[1)]，朱　岩[1)]

1) 北京科技大学计算机与通信工程学院，北京 100083　2) 浙江大学控制科学与工程学院，杭州 310027

✉ 通信作者，E-mail: chene@ustb.edu.cn

摘　要　智能合约在法律层面的关注度不断提高，如何将现实法律合同转化为智能合约程序，保证法律元素提取和程序转换的规范化已经成为当前的研究热点。据此，本文从合约模板化和语义规范化的角度出发，提出一种合同文本置标语言（CTML），通过对合同中语法、结构、词汇的内容进行标注，实现合同要素的提取与转化。首先，构建合同元模型并建立“要素-属性-成分”的三层语义结构与数源标记表示，基于元模型设计面向合同文本的置标语言语法规则，通过 CTML 完成法律信息规范化提取，形成标注合同；其次，通过递归抽象语法树（AST）并建立映射关系设计由标注合同到智能法律合约的转换规则，完善法律合同到智能合约可执行代码的转化链条。进而，以保理合同为例，展示了合同文本置标语言的语义提取和代码生成有效性，为普通法律合同转化为智能合约提供了一种技术方法。

关键词　法律信息学；智能法律合约；置标语言；面向领域语言；SPESC；

分类号　TP311.5, DF59

Contract Text Markup Language (CTML): A Regularization Method of Extracting Legal Elements Oriented in Smart Contracts

FAN Yuqing[1)], *WANG Di*[2)], *LIN Hongjie*[1)], *CHEN E*[1)]✉, *HE Xiao*[1)], *ZHU Yan*[1)]

1) School of Computer and Communication Engineering, University of Science and Technology Beijing, Beijing 100083, China

2) College of Control Science and Engineering, Zhejiang University, Hangchow 310027, China

✉ Corresponding author, E-mail: chene@ustb.edu.cn

ABSTRACT　The attention to smart contracts at the legal level is increasing. Considering that contract text is written in natural language, computers cannot process it directly, thus the problem of accurate understanding contract content and meaning representation still needs to be solved. This problem further leads to a lack of regularization in the generation process of smart contract programs, as well as lacks of legal recognition and legal effectiveness. Therefore, it is necessary to develop a new approach of converting real-life legal contracts into smart contract programs and ensuring the regularization of legal element extraction and program conversion. In this paper we point out that the above conversion process has experienced three evolution stages (including manual processing, smart contracts, and smart legal contracts), and is being at the legal code stage. From the perspective of contract template and semantic regularization, we propose a Contract Text Markup Language (CTML), which is a normative computer processing language for expressing meaning in legal contracts. By annotating the content and meaning representation of the syntax, structure, and vocabulary in the contract using CTML, a method for regulating the content and meaning representation of legal contract text is established to achieve the extraction and conversion of contract elements. Firstly, a contract meta-model of CTML, including a three-layer of "element-property-component" semantic structure and metadata markup representation is established, by which the contract text information is gradually refined in order from "large to small" and "coarse to fine" to build the corresponding relationship from real-life contracts to smart legal contracts. Furthermore, the syntax of CTML is designed, then the legal elements can be extracted and regularized to form an annotated contract using CTML. Secondly, we design specific conversion rules from CTML to SPESC (as a smart legal contract

language) to generate smart legal contract programs by recursively Abstracting Syntax Tree (AST) and establishing a mapping relationship. These rules help users write contracts, improves the efficiency of conversing contract text to executable code, and ensures that the writing of smart legal contracts is based on solid grounds, thereby improving the conversion chain from legal contracts to executable smart contracts. In addition, taking a factoring contract as an example, we illustrate the details of semantic extraction and code generation. In this way, the contract semantic extraction is clearer, the conversion is more normative, and the code development is more effective, and therefore the proposed CTML provides an alternative regularization method to generate smart legal contracts.

KEY WORDS Legal Informatics; Smart Legal Contract; Markup Language; Domain Specific Language; SPESC;

1. 引言

智能合约是一类部署在区块链上并在满足预定条件时可自动执行并存证的计算机程序，可用以实现价值交换等应用，已成为新一代区块链的核心技术之一[1]。作为智能合约在现实世界的对应物，合同是民事主体之间设立、变更、终止民事法律关系的协议，是维护市场经济秩序和保障当事人合法权益的重要手段[2]。然而，由于合同文本采用自然语言撰写，目前仍然不能由计算机直接进行处理，缺少准确理解合同内容和意思表示的有效方法，导致所生成的智能合约程序规范化缺失，也缺少法律认可和相应的法律效力。因此，有必要探索一种直接且规范性的法律要素提取方法，显性化展现法律合同文本到智能合约程序的对应关系[3]。

1.1 相关工作

文献[4] 认为智能合约能够通过代码来表现、确认和促进合同条款的自动执行，然而由于将自然语言转化为智能合约需要考虑很多因素，如合同文法结构、法言法语的法律意义等，故合同意思表示的确定将不可避免地面临解释问题[5]。自然语言存在二义性，导致自然语言的含义可能会在某些内涵之间来回波动[6]。相较而言，程序语言的词汇和语法规范是完备的，故相对于自然语言更少产生歧义[7]。因此，如何准确体现原始法律合同的意思表示是值得探讨的问题。

对于上述问题，学者已经做了大量的研究工作。2017 年周学峰等人[8] 认为实现法律代码化，即从法律合同文本生成智能合约代码，需要把法律合同文本的内在逻辑通过人为或计算机推理，再将其转换为计算机语言程序代码，然后使用转换后的程序代码执行该合同，最终代替或帮助法官对案件进行审判。同年，赵精武等人[9]简单讨论了现实法律合同与代码程序的法律一致性，并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。2018 年陈吉栋[10] 进一步从“要约-承诺”角度证明了智能合约是一种书面法律行为。2021 年刘琴[11] 等人从计算机科学角度分析法律合约与智能合约一致性。以上工作为支持法律要素提取的合同元模型提供了设计思路。

除了上述智能合约语义理解和法律信息规范化提取工作之外，国内外近年来也开展了有关现实合同在智能合约的应用研究项目。例如，开源软件 Accord Project[12] 提出了一种合同设计模型，模型包括文本-模型-逻辑三部分，通过绑定数据层提取模板实例化后条款中的数据信息，用于逻辑代码程序的数据输入；OpenLaw 项目[13] 依赖于标记语言将自然语言表示的法律协议转换成在模板中定义的相关变量和逻辑，形成一个机器可读的法律协议模板库。但两个项目的共性问题在于标记不涉及合同语义，无法表示合同文本的整体架构，加之自然语言与程序代码间缺少过渡，导致合同模板和智能合约代码的意思表达是否一致存在争议。

SPESC 项目[14] 是以智能法律合约（Smart Legal Contract, SLC）为目标的代码法律化研究项目，它通过智能法律合约语言（SLCL）的研制提供了一种准确理解合同内容和意思表示的有效方法。该语言作为一种高级智能合约语言，是介于现实法律合同和智能合约程序之间的一种语言，通过易读且规范化的语法，帮助不同领域人员进行沟通[15] ，从而实现法律合同与网络空间的程序代码相衔接。它既可保证智能法律合约既有现实

合同的法律特征和易理解性，又有计算机程序代码的规范性，促进计算机、法律等专业人员的跨领域协作，打破了智能合约与非程序人员之间的领域壁垒，能更好地满足法律合同的自动化、形式化转化[16]。

尽管上述理论研究和项目实践进行了有益探索，但是目前仍然缺少解决法律合同中语义的提取与转换问题的有效方法，同时也没有一种对提取信息进行易于计算机处理的法律元素及其特征予以规范化表示的方法。

1.2 挑战与解决思路

在已有研究中，合约模板化和语义规范化为合同文本到智能合约程序的转化提供了可行技术框架，特别是以智能法律合约作为现实合同和智能合约语言之间的桥梁，对法律领域的行业应用推进具有重大意义。然而，目前的转化过程依然缺乏有效方法建立智能法律合约和法律合同之间的语义对应关系，导致现实合同到智能合约的转化框架无法发挥应有的作用。

针对上述问题，本文拟采用面向合同文本的置标语言实现法律合同到智能合约程序之间的语义转换。置标语言是一种将文本以及文本相关信息结合起来，展现出关于文档结构、特征、词汇意义及意思表示、以及数据处理细节的计算机可处理文字编码。合同文本作为一种符合法律的文档，尽管具有法律效力，但由于合同文本所使用的自然语言客观上存在着表达上二义性，易于引起不同法律主体理解上的歧义性，而采用合同文本置标的方式，有利于法律界人士对合同语义进行解释、实现规范性的二义性消除和语义表示的优势。

以支付合同为例，下文的条款用于来描述债务人希望从债权人那里获得的金额。

> 双方同意通过签订新协议，将数额_____，按照下面规定的条款和条件写入结构性支付协议，以保证债务金额。

在这个简单的例子中，该条款包括三种法律元素：第一种是债务金额，用下划线表示；第二种是当事人的实际行为，被描述为 "签订新的协议"；第三种是其余部分，是双方对该行为的意向表达。通过使用类似 HTML 的标记语言，上述示例条款可以被标记如下：

```
<<genTerm@term1>>双方同意通过
    <<property action@afford #duty=must #party=parties>>
    签订
    <</property>>新协议，将数额<{payment %Money}>，按照规定的条款和条件写入结构性支付协议，以保证债务金额。
<</genTerm>>
```

通过使用 CTML 语言标注，语句中具有法律特征的元素得以凸显，如在该标记结果中，合同文本被封装为一个名为 term1 的条款。而其内部参数（见红色字体）被标记为一个变量，名为 payment，类型为 Money。此外，实际行为被标记为一个法律属性，它包含一组属性，例如，行动名称、职责和当事人（见蓝色字体）。该条款的其余部分将被保留不变。同时，对其他法律元素或因素的注释和提取也允许在不同的应用中进行。需要强调的是，本文研究的“规范性”是指在由现实合同到智能法律合约这一过程中，信息抽取后的表示依旧统一，且内容理解一致无歧义。虽然以以太坊为首的区块链平台相继推出了不同标准（如以太坊改进协议 EIP、代币标准 ERC 等），但这些标准仅服务于智能合约如何在项目的不同实现中保持统一的兼容性，与本文研究的“规范性”无本质联系，二者互不冲突。合同标记也有利于实现合同语义到智能合约程序的自动转化，提高智能合约开发效率。

回顾合同转化代码的历程，已经经历了由人工处理到智能合约、再到智能法律合约三个阶段的变化过程。随着合同本文置标语言（Contract Text Markup Language, CTML）的设计与开发，有利于促使合同转化代码过

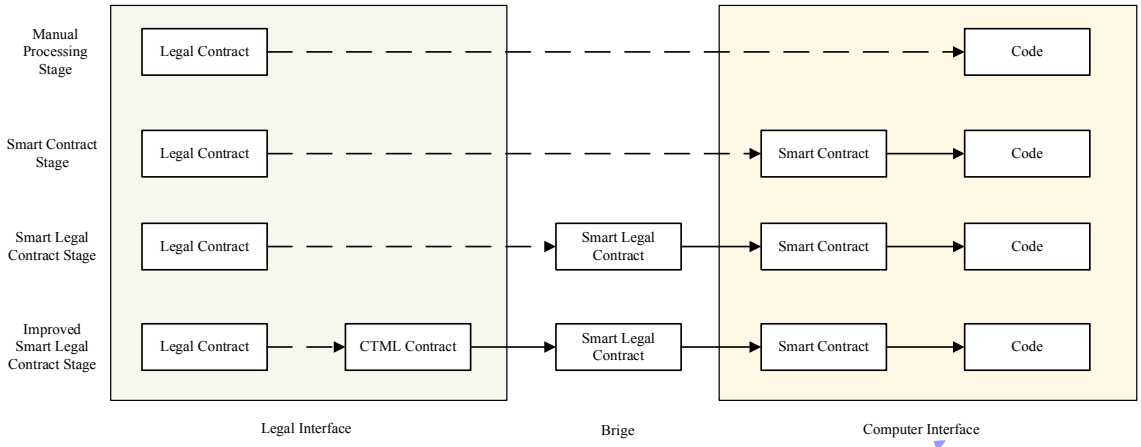

图 1 研究路线图
Fig.1 Roadmap for research

程更加完善。如图 1 所示，合同转向代码过程可概括如下：

1) **人工处理阶段**：人工将合同直接转化为可执行程序或代码，尽管两者有映射关系，但是没有转化工具，也不能验证转化的正确性，无法表明合同中意思表示是否与代码意思表示具有等价性；
2) **智能合约阶段**：人工将合同转化或直接编写智能合约程序，进而通过智能合约程序转化为可执行程序或代码，以辅助完成合同的成立与合同生效期间的执行同步。有些合同其实并不需要转化为智能合约形式，故此后阶段的合同一般特质有自动执行需要的合同。直接编写智能合约程序不利于用户理解程序的意思表示，故采用合同转化依然是本阶段的主流方式。目前已有智能合约被应用于司法判决的案例，如西湖法院的金融借贷纠纷案，通过司法链技术为法院执行减负增效。智能合约程序的出现实现了合约自动化执行的重大突破，但存在可读性差、生产效率低、难以实现合同与智能合约间高效转化等问题，阻碍了智能合约的法律化进程；
3) **智能法律合约阶段**：以智能法律合约语言为标志，智能法律合约的易读性解决了智能合约难以学习与阅读的问题，在合同和智能合约之间建立桥梁，解决了两种语言语法跨越大的问题，但智能法律合约依然无法明确与合同的联系，这影响了智能法律合约的进一步发展；
4) **完善智能法律合约阶段**：以自然语言为载体的法律文本合同通过置标语言标注后生成 CTML 合同，再由转化为智能法律合约语言撰写的智能法律合约，最后经编译后生成智能合约可执行代码，从而实现了由法律文本合同到智能合约代码生成的完整且规范化流程，保证转化后的程序与所标记的法律合同具有同等法律效力[4]。

合同文本置标语言的研制也有助于构建面向法律文本合同的智能法律合约系统。它通过从 CTML 标注中提取合同中法律信息，并映射为智能法律合约语言程序，帮助使用者进行合约编写。所生成的智能法律合约来源有据可依，形成 CTML 标注文档到智能法律合约再到可执行程序代码、完整且规范的智能合约开发流程，也构成了法律可信性链条。

本文组织如下：第二章对 CTML 模型架构进行说明，介绍如何使用 CTML 语言完成法律元素的提取；第三章展示了 CTML 的语法规则表示方法；第四章给出了 CTML 转换到智能法律合约的转换规则，并给出了由法律合同到智能合约的转化框架。第五章通过引入保理合同实例对 CTML 执行过程进行进一步解释。最后，第六章总结本文成果，并进行展望。

2. 系统架构

本文通过设计合同文本置标语言 CTML，以此建立一种将法律合同文本中内容和意思表示进行规范化的方法。该语言定义如下：

定义 1（合同文本置标语言）合同文本置标语言是一种创建法律合同或对法律合同进行意思表示的规范性计算机处理语言。CTML 能够对文档中语法、结构、词汇的内容进行标注，便于更加准确和一致性的提取法律合同文本。

据此，CTML 将法律合同文本到智能合约程序的转化过程拆解为合同中语义的提取与转换两方面：

1) **语义提取**：是指从合同文本中提取内容及其含义中的关键要素，包括当事人、标的、价款或报酬、履行期限、地点和方式、违约责任、争议等法律构成要素及其性质和属性（如数量、质量等），为特定的合同后期处理和应用奠定基础。
2) **语言转换**：是指转化前的法律合同与转化后的智能法律合约之间存在语义对应关系的有效佐证，为实现合同文本与智能法律合约间自动转化提供正则文法，有助于提高开发效率和转换准确性。

相比直接人工将传统合同转化为法律化代码，CTML 标注的优势与价值可总结为以下几点：

1) **依据清晰**：通过“标注”这一行为，使得后续智能法律合约的生成能够做到有据可依；
2) **转化规范**：使用 CTML 对合同标注后，其从标注合同至智能法律合约代码的过程由代码生成器完成，通过确定合同要素与智能法律合约元素间的对应关系，实现转化的规范性；
3) **减少代码工作**：智能法律合约基于标注合同自动生成，减少代码的学习负担。

2.1 建立原则与目标

本文的研究目标为设计合同文本置标语言语法规范，建立智能法律合约和法律合同之间的语义关系。该目标可进一步被细分为法律合同意思的准确表示和法律合同意思表示的计算机理解两个阶段。对此，本文针对上述两大阶段提出解决方案：

- 法律合同意思的准确表示：通过对法律元素进行元模型解决建模，设计标记语言，实现对文本内容中法律元素信息的判断与定位，提取文本中法律元素；
- 法律合同意思表示的计算机理解：将提取出的法律元素定义模块描述方法和数据处理方法，由映射生成智能法律合约代码，将法律元素由标记合同形式过渡到智能法律合约形式。

由此，本文实现合同文本置标语言并提出面向法律文本合同的智能法律合约系统架构，完成合同到代码的可信链条，简化智能法律合约的编写过程，促进协作式智能法律合约开发。

面向法律文本合同的智能法律合约系统使用标记语言，对法律合同中的法律元素从粗到细地加以形式化表示。作为连接现实合同和智能法律合约的主要手段，在设计的过程中主要遵守以下几条原则：

- 标记语言能完整提取合同的意思表示，完成现实合同到智能法律合约的连接工作；
- 规范智能法律合约的编写，提高合同文本到可执行代码的转化效率；
- 生成更系统的智能法律合约程序框架。

2.2 CTML 系统框架

CTML 系统框架如图 2 所示，CTML 支持文本合同到智能合约代码生成的完整且规范化流程，其流程如下：

1) 支持基于文本合同的智能合约开发与部署，步骤包括：
   a) 文本合同采用 CTML 标注后生成 CTML 合同和交换标记数据表（Exchange Markup Datasheet, EMD）；
   b) 通过 CTML 到 SLCL 的词汇映射和转化规则生成 SLC 程序；

c) SLC 程序经编译后与 EMD 链接生成智能合约可执行代码；

d) 智能合约代码部署至智能合约平台，实现合约部署。

2) CTML 合同、EMD、SLC 程序整合至合约服务器，宜于合同相关方与合约服务器间实时交互及合约服务器与智能合约平台间通信，实现合同协商、订立与执行。

EMD 是指以键-值对形式描述 CTML 合同中数源标记的数据表，用于对客户操作数源标记的交互过程进行确认、特指或限制。上述 CTML 系统能够实现法律一致性和消除二义性的依据在于：

- 架构上：CTML 标注现实合同形成 CTML 标注合同，并通过 CTML 到 SLCL 的转化规则自动生成智能法律合约，通过文献[17] 所述的智能法律合约语言的可执行合约转换方法生成相同意义的智能合约，现实合同-CTML 标注合同-智能法律合约-智能合约可执行代码的链条完整，合同模块保持一致。
- 分析上，CTML 标注合同通过穷举搜索分析生成唯一推导树（语法分析树），根据该推导树映射规则生成的智能法律合约唯一。

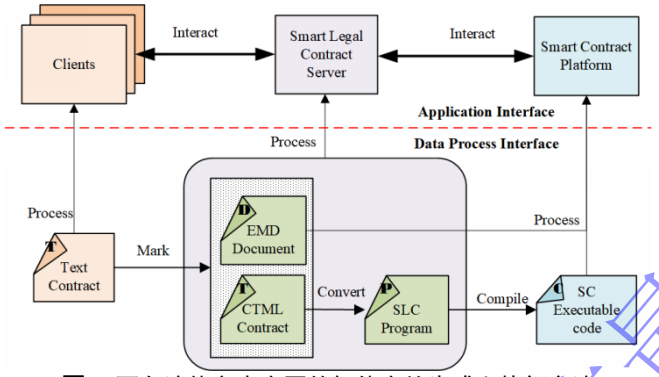

图 2 面向法律文本合同的智能合约生成和执行方法

Fig.2 Smart contract generation and execution methods for legal contracts

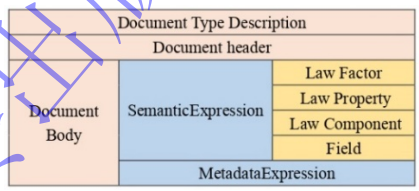

图 3 CTML 语法整体结构图

Fig.3 The overall structure of CTML syntax

3. CTML 语法设计

CTML 的语法规则使用 Xtext 实现。Xtext 是基于扩展巴科斯范式规则（EBNF，Extended Backus－Naur Form）的用于开发领域专用语言 DSL 的开源软件框架[18] [19] [20] 。选用 Xtext 作为 CTML 语言实现的工具，是因为该方法具有以下优点：

- Xtext 能够生成该语言模型对应的抽象语法树（Abstract Syntax Tree，AST）的类模型[21]，该语法树可以有效辨析实例的元素层次结构，利于实例的架构分析；
- Xtext 的“Validation Package”模块[22] 可以指定特定领域语言的验证规则，实现对领域语言的约束；并为文本具体语法提供自动补全、语法着色、重命名重构、大纲视图和代码格式，以正确缩进文档。
- Xtext 的“Code Generation”模块用于实现元模型之间的实体映射，可将当前领域的本体实例转化为可执行的语义相同的其他本体示例，该部分将在第五章进行进一步讨论；
- Xtext 支持语言服务器协议[23] (Language Server Protocol，LSP)，该协议是一种开放的、基于 JSON-RPC 的协议，将服务器和开发工具的通信进行协议标准化，从而使面向领域语言支持多种平台，提高语言的扩展能力和使用范围。

3.1 CTML 语法整体架构

根据 CTML 的建立规则及领域语言的设计方法， CTML 整体架构根据文本合同的元素进行整体性划分，将其分为文档类型说明、文档头和文档主体三大部分，如图 3 所示：

文档类型说明：用于声明文件类型（DOCTYPE），在文档的序列化形式中，它表现为符合特定语法的简

短标记字符串。此处仅将该声明用于模式选择使用。

文档头：用于声明 CTML 文档必要信息，如声明合同文本语言语种。

文档主体：文档主要内容，在原有合同的基础上用 CTML 标记进行标识，故主体部分包括原合同文本（Text）和标记。标记依据功能需求又划分为语义标记 SemanticExpression 和数源标记 MetadataExpression。

语义标记 SemanticExpression，用于指示现实法律合同中的法律元素和意思表示，通过语义标记为智能法律合约提供法律依据；语义标记的层级划分如图 4 所示。根据《民法典》相关内容，结合先前在智能合约的规范化表示的相关工作[16] [24] 实现文本合同的抽象表示，

数源标记 MetadataExpression，用于标记合同模板或仍未将全部内容填写完整的合同时，对带填写部分进行指示与定位。

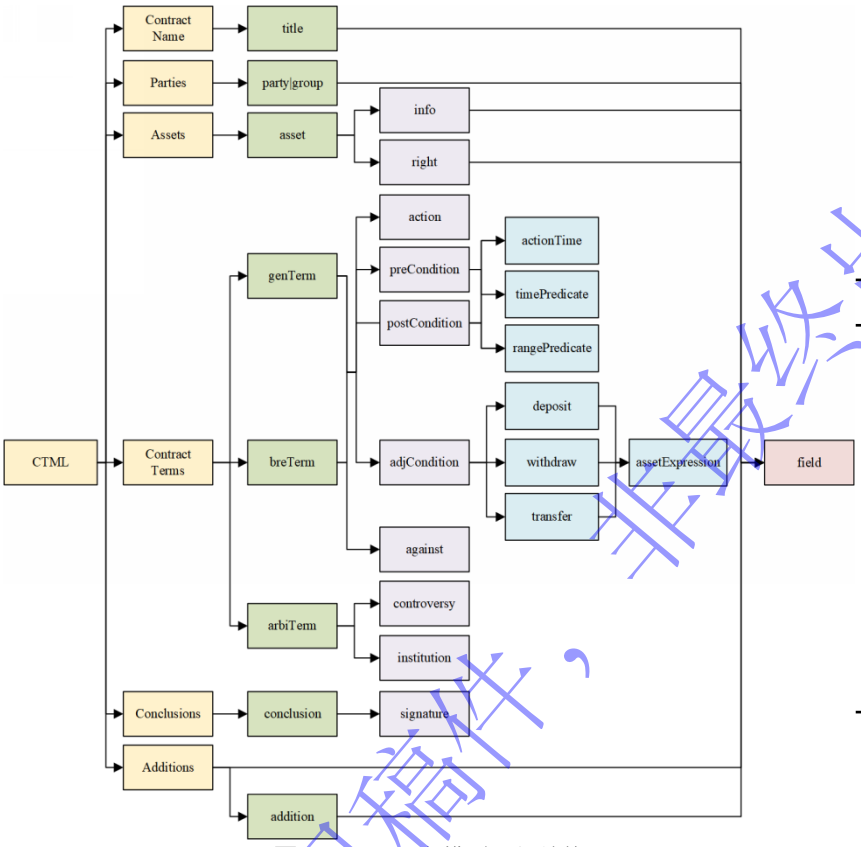

图 4 CTML 语义模型层级结构

Fig4 The layer structure of CTML Semantic Model

表 1 CTML 通用符号及说明

Table 1 The common symbol and description of CTML

Symbols	Description
@	Marking prefix
|	Element or
[]	Choosing keyword
#	Optional Indicator
%	Type Indicator
+	Zero or more
.	Hierarchical Connectors
<<>> <</>>	Semantic Markup
<{ }>	Metadata Markup

CTML 适用于“嵌套结构”和“通用符号”两项通用规则，为定义 CTML 语法提供理解性支持。

嵌套结构是指由要素标识、属性、成分构成的标识或名称通过层级关系连接符“.”加以标识的元素间包含关系。需要注意的是，由于合同文本存在无序性，一般不具备元模型一般清晰的架构，存在法律要素和法律要素相关法律属性异位的问题，故语法规则允许法律属性在文档主体中以通过层级关系连接符“.”指出对应要素标识的方式与法律要素同级存在。

依据第三章的合同元模型进行语法设计，元模型中的每个元素均有对应的一到多条语法规则。语言设计所采用符号及说明如表 1 所示：

3.2 CTML 语义标记

语义标记使用嵌套结构的文法进行法律合同的语义信息标注，其语义的提取与顶层框架的制定标准以《民法典》的相关内容为主导，同时以要素提取的相关文献（如文献[25] [26] 关注于要素分类，文献[27] [28]

关注于建立规则识别模型）中的抽取结果为辅完成，形成具备层级结构、信息逻辑清晰的“合同范例”，进而转化为智能法律合约程序与通用性智能合约程序及代码。

语义标记中的标记信息置于双尖括号内部，并将前后部语义标记置于被标注文本外侧，即<<标记信息>>文本<</标记信息>>，其中标记信息指文本中提取得到的需要标注的法律信息，且该法律信息可用于后续代码的解析转化。语义标记定义如下：

定义 2（语义标记）语义标记**用于**对**文档中法律构成要素、法律属性、法律成分和域**进行标**注**，格式如下**：

SemanticExpression::=

<<**factor|property|component|field** parameterList>> text <</**factor|property|component|field**>>

其中，

factor、property、component、field **分**别为**法律要素、属性、成分、域的保留字**，parameterList **表示参数列表**，text **表示被**标**注文本。**

参数列表依据标记**种**类**具有不同定**义：

定义 2.1 法律要素（简**称要素** factor）**指用于构成合同、具有独立性的法律要素，包括**：**合同**标题、**当事人**、标**的、条款、合同**订**立、附加信息等文本信息。**

定义 2.2 法律属性（简**称属性** property）**指法律构成要素中的基本性**质，**包括**资产**的属性与**权**属、合同条款的**行为、**前置条件、伴随条件、后置条件、**违**反、争**议、**机构以及合同**订**立中的**签名等文本信息。**

定义 2.3 法律成分（简**称成分** component）指**法律属性表达所涉及到的限定信息，包括**时间**表达中的行**为时间、时间谓词、边**界**谓词，资产**操作中的存入**动**作、取回**动**作、转**移**动**作以及**资产**表达式等文本信息。**

定义 2.4 域（field）**代表文本合同中的合同基本信息。**

此处语义标记定义的是此类标记的普遍格式，不直接进行文本标注，而是以此标记格式为基础，进一步定义了法律要素标注（Law Factor Mark , LFM）、法律属性标注（Law Property Mark, LPM）、法律成分标注（Law Component Mark, LCM）以及辅助性标注的方法，其中辅助性标注包括域标注（Field Mark, FM），这些定义的标记才会被用于具体文本的标注过程。且在使用这四类标注时，采用层级标注方法。层级标注结构依据合同文本信息间的“包含关系”将文本分成不同的层次，从而使得用户可以依照“从大到小”、“从粗到细”的顺序对文本进行逐层细化。详细的标注方法和语法定义可见表 2 中的语义标记部分。

实现 CTML 语义标记的通用规则时，需要讨论以下情况，如图 5 所示：

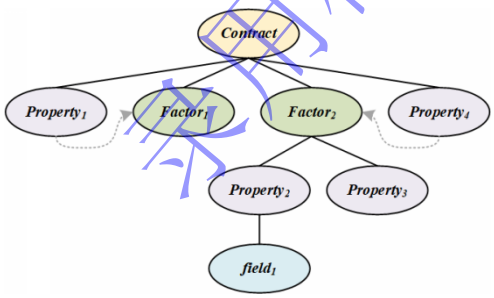

图 5 CTML 语法树示例图

Fig.5 Example diagram of a CTML syntax tree

Algorithm 1 getScope

Input: field, context, reference
Output: A list of Property

```
1:  function GETSCOPE(field, context, reference)
2:      factor ← getFactor(field.factorID)
3:      if factor is null or factor.eIsProxy is null then
4:          return super.getScope(context, reference)
5:      else
6:          propertyList ← extractProperty(context, factor.name)
7:          for property ∈ getProperties(context) do
8:              if property.name is not null then
9:                  if property.factorID == factor.name then
10:                     addElement(property, propertyList)
11:                 end if
12:             end if
13:         end for
14:         return propertyList
15:     end if
16: end function
```

图 6 交叉引用重定义函数算法

Fig.6 algorithm for function overriding getScope

1) 示例图中，域 $field_1$ 为 $Property_2$ 的子节点，若 $field_1$ 需要使用嵌套规则标识的所属的要素、属性关系时，其法律属性的交叉引用默认只能检索到该分支上的相关实例，即只能检索到 $Property_2$ 和 $Property_3$，而无法检索到 $Property_1$ 和 $Property_4$，该结果无法满足语法规则实现需求；$Property_1$ 和 $Property_4$ 被称为属性的非嵌套形

式；

2) 示例图中，$Property_1$ 已标识其包含于要素 $Factor_1$，若域 $field_1$ 标识其包含于 $Factor_2$，则域 $field_1$ 的交叉引用不应检索到属性 $Property_1$，因该属性与所包含的要素 $Factor_2$ 无包含关系。

故需要对交叉引用范围（Scope）进行重定义，对法律属性 Property 的检索结果加以强化和约束。该重定义函数需要经历以下流程：

1) 获取被引用的 Factor 实例；

2) 遍历 CTML 抽象语法树，提取指向该 Factor 实体的法律属性，具体包括该要素实体内部嵌套形式的属性，和非嵌套形式指向该要素实例且实例化唯一标识符 name 的属性；

3) 返回符合条件的属性集合。

属性 Property 的可选用范围依据所选要素实例转化为该要素实例所包含的属性，伪代码如图 6 所示。

表 2 CTML 标记语法定义表

Table 2 The syntax definition table of CTML

Markup		Syntax definition	
Document		**<!DOCTYPE ctml> <ctml lang=zh_cn> <<factor title**@printerDeal**>>** text **<</factor>> </ctml>**	
Semantic Expression	**Law Factor**	要素形式化表示 <<要素 要素类别@要素标识 (#特征=特征值)+>> 文本 <</要素>> 要素类别::={标题，当事人|群体，资产，一般条款|违约条款|仲裁条款，合同订立，附加信息}	***factorExpression::=*** **<<factor** factorSet@factorID (#attribute=value)+**>>** text **<</factor>>** ***factorSet ::=*** {title, party|group, asset, genTerm|breTerm|arbiTerm, conclusion, addition}
	Law Property	法律属性形式化表示 <<属性 [要素标识.]属性类别 [@属性标识] (#特征=特征值)+>> 文本 <</属性>> 属性类别::={信息,权属,行为,前置条件,伴随条件,后置条件,违反,争议,机构}	***propertyExpression::=*** **<<property** [factorID.]propertySet[@propertyID] (#attribute=value)+**>>** text **<</property>>** ***propertySet ::=*** {info, right, action, preCondition, adjCondition, postcondition, against, controversy, institution, signature}
	Law Component	法律成分形式化表示 <<成分 (#特征=特征值)+>> 文本 <</成分>> 成分::={行为时间，时间谓词，边界谓词，存入动作|取回动作|转移动作，资产表达式}	***componentExpression::=*** **<<component (#attribute=**value)+**>>** text **<</component >>** ***component::=***{actionTime, timePredicate, rangePredicate,deposit|withdraw|transfer, assetExpression}
	Field	域信息形式化表示 <<域 [要素标识.属性集]@域标识 [%类型] [#域值=取值]>>文本<</域>>	***fieldExpression::=*** **<<field** [factorID.propertySet]@fieldID **[%**type] [**#quantity=**value]**>>**text**<</field>>**
Metadata Expression		数源标记 <{[要素标识]@交互数据标识 [%类型] (#选择方式=备选数据)+}>	***metadataExpression::=*** <{[factorID]@exchangedDataID [**%**type] (#option=value)+}>

3.3 CTML 数源标记

数源指合同中待协商内容，其标记称为数源标记，仅作为占位存在。数源标记用于标记合同模板或仍未将全部内容填写完整的合同时，对带填写部分进行指示与定位。

数源标记标注因合同当事人或标的物等不同而带来的合同个性化文本元素（被称为交互数据 exchangedData），数源标记所对应的文本常是文档中基本且不可分割词汇，用于合约当事人对合同内容进行宣称、填入或选择，标记者根据合同交互要求可以在文档中任何地方使用数源标记。数源标记定义如下：

定义 3(数源标记)所述数源标记**采用如下格式**:

metadataExpression::= <{[factorID]@exchangedDataID [**%**type] (#option=value)+}>

其中,

factorID **表示交互数据所属**层**次中最外**层**的要素**标识,

exchangedDataID **表示**该**交互数据的唯一**标识,

option **表示交互数据的**选择**方式**, **可包括**单选 singleOption、**多**选 multiOption、**外部**输**入** import、**触**发 trigger、**分配** allocate **等**, **用集合** optionSet **表示**, **即**:

optionSet::=

{singleOption|multiOption,import,trigger, allocate}

其中,

value 表示合同文本在此处的备选数据或取值,

type 表示交互数据的类型, 可分为数据类型 dataType 和权属类型 rightType,

外部输入 import 表示可接收用户传入数据、

触发 trigger 表示可接收外部事件、

分配 allocate 表示可接收用户定义的复杂类型的数据。

通过对合同文本置标语言 CTML 中文本元素区分语义标记和数源标记，由前者表示法律合同的语义信息并形成“合同范例”，借助标记语言编辑器生成的抽象语法树，实现对合同整体框架的提取；后者定位待标注信息，并提取信息构成交换标记数据表 EMD，从而实现代码与数据分离，便于智能合约通用开发与个性化应用、以及计算机高效处理与样式个性化显示。

4. 代码生成

CTML 语言目前只是一种标记语言，其执行需要完成 CTML 到智能法律合约语言的映射。映射是通过模型转换提供的，即将 CTML 元模型实体映射到现有智能法律合约的元模型实体中，得到具有定义良好、可理解且可执行的智能法律合约。

```
Contract returns Contract:
  'contract'
  name=ID
  '{'
        parties+=Party+
        assets+=Asset*
        fields+=Field*
        additions+=ComplexType*
        (terms+=Term '.')*
        (conclusion += Conclusion)*
'}';
```

图 7 SPESC 的 Contract 语法

Fig.7 Contract syntax for SPESC

Algorithm 2 fieldProcess

Input: contract, EcoreUtil2

Output: A list of field

```
1:  function FIELDPROCESS(contract, EcoreUtil2)
2:      root ← EcoreUtil2.getRootContainer(contract)
3:      fieldList ← EcoreUtil2.getAllContentsOfType(root, Field)
4:      for field ∈ fieldList do
5:          if field.factorID is null then
6:              if field.propertyID is null then
7:                  fieldList.remove(field)
8:              end if
9:          end if
10:     end for
11:     return fieldList
12:
13: end function
```

图 8 域标记预处理算法

Fig.8 algorithm for pre-processing field

这一步在领域特定语言中通常交给代码生成器（Generator）完成，代码生成使用 Xtend 进行开发。Xtend 源自 Xtext，是一门用于定义语言和编辑器的技术[29] 。Xtend 的语法和语义起源于 Java 编程语言，但专注于更简洁的语法。运行时，Xtend 程序会被编译为 Java 代码，从而与所有现有的 Java 库无缝集成。

本研究选用智能法律合约语言 SPESC 作为代码生成的目标语言。SPESC 语言语法定义如图 7 所示，其合同框架是以当事人、标的、域属性、附加信息、条款、订立签名的顺序进行合同规范的。上述模型在 CTML 中均能找到对应的要素与之对应，故在代码生成器的设计时，应以要素及其相关信息为单位，依照要素类型顺序进行映射，并将映射结果写入生成的 SPESC 文档中。

CTML 代码生成器的工作流程简单描述如下：

1) 预处理，提前提取域标记和非嵌套属性集，对 AST 进行变换；
2) 根据 AST 递归到特定类型要素实体，并根据该实体筛选信息，包括预处理时的域和属性，及该实体的嵌套实体元素；
3) 依据映射关系生成对应要素实体字段；
4) 遍历要素实体所含信息，依据映射关系生成对应属性、成分或域字段，转到步骤 2)。

代码生成器具体实现过程如下：

4.1 预处理

预处理的必要性在于：解析器传入抽象语法生成树 AST 时，该生成树为实例文档中的实体语义模型，但并没有按照法律合同元素的层级关系进行排列，故需要对生成树上法律合同元素进行整合与再排序。实体 AST 和法律合同层次模型实体的差别主要体现在：

- 域标记实体可以存在于任意地方
- 非嵌套形式的属性实体在 AST 中与要素实体同属在根节点

故需要提前分类提取 AST 中的所有域标记和非嵌套形式的属性实体，以保证后续映射时要素信息完整。

1）域标记：

域标记预处理从标注好的合同文本中的提取指定包含关系的域标记。在实现域标记的预处理提取时，可以利用 EMF 模型元模型层（EcoreUtil2）提供的函数 getAllContentsOfType()[30]，该函数可通过指定 AST 根节点和 Xtext 转化后元素的 java 类，提取树中所有该类型的元素实体。

由于嵌套方式存在的域标记本身就可以作为要素的实体信息被获取，故需要去除掉嵌套形式的域标记以免造成重复，嵌套的表现形式为不指定上级要素和属性。预处理伪代码如图 8 所示：

2）非嵌套形式属性：

非嵌套形式的属性实体可以直接通过 AST 根节点获取属性实体。由于属性 Property 为抽象类，故需要进一步细化属性类型，提取不同属性类的数组。本部分提取方法与图 6 相同。

4.2 映射关系

CTML 代码生成器的工作步骤 3) 包括以下两部分内容：

- 依据映射关系生成对应要素实体字段
- 遍历要素实体所含信息，依据映射关系生成对应属性、成分或域实体字段

现举例对该步骤流程进行说明：

对被标注文本中出现的所述简单要素对应的要素标识 factorID 和特征 attribute，依据转化关系中的元素映射关系，将要素标识 factorID 和特征 attribute 写回到智能法律合约 SPESC 代码中，再将映射后智能法律合约语言 SPESC 代码添加到智能法律合约程序中，从而完成以要素标识 factorID 为名称的要素对象建立；

以标题 title 转换处理为例，标题要素对应智能法律合约 SPESC 语言中标题语法如下：

Title ::= **contract** Cname (: **serialNumber** Chash)?

其中，提取标注中的标题标识 titleID 和特征 serialNumber 对应特征值 value，分别作为智能法律合约语言 SLCL 的标题语法中对应的合约标题 Cname 和合约序号 Chash，即生成智能法律合约代码为“**contract** titleID (: **serialNumber** value)?”。

下面对 CTML 代码生成器的工作步骤 4)进行映射过程说明，以域信息为例：域 field 信息转化处理是指提取域标记中信息，并依据智能法律合约语言 SLCL 中属性域 field 语法：

field ::= attribute : (constant | type)

进行词汇映射，并将映射后语法作为智能法律合约代码，其中，提取标注中的域标识 fieldID 作为上述语法中 attribute，取值 value 或类型 type 作为上述语法中 constant 或 type，所提取的要素标识 factorID 及属性类别 propertySet 用于确定所生成智能法律合约代码“fieldID : (value|type) ”在目标智能法律合约中所处要素及其属性中的位置，其中，含有属性域的要素包括标题 title、当事人 party 或 group、附加信息 addition、资产 asset。

5. 应用实例

在现实合约层，《民法典》整理了生活中常见的法律文本合同，包含买卖合同、租赁合同、赠与合同等大类合同。根据《民法典》第七百六十一条 “保理合同是应收账款债权人将现有的或者将有的应收账款转让给保理人，保理人提供资金融通、应收账款管理或者催收、应收账款债务人付款担保等服务的合同”，其文本元素包括合约名称、当事人、标的声明、条款，能够完整表述 CTML 不同层次标注要求。考虑到目前合同文本置标语言的研究尚处于初期阶段，且智能合约主要应用于经济交易的场景，智能法律合约也大多基于该类应用场景进行设计，故本章将主要围绕有名合同中的保理合同展示 CTML 的语义提取和代码生成有效性，合同来源于行业协会（如中国服务贸易协会）推荐的商业合同样例节选。

《应收账款转让通知书》

编号：

致：_____（下称“贵方”）

自：_____（下称“我方”）

我方因经营发展的需要，已申请将下述应收账款及所从属的一切从权利和权益转让给_____（下称“商业保理人”），兹此通知贵方：

1、如下第____项应收账款及从属的一切从权利和权益转已转让给商业保理人：

（1）我方与贵方之间_____项下的全部应收账款。

（2）我方对贵方下表所列的全部应收账款：

序号	应收账款债务人	基础教育合同及编号	单据名称	单据号码	应收账款金额	应收账款到期日

2、 请贵方于应收账款到期日将上述应收账款金额足额支付到如下账户（开户人：_____；开户银行：_____；账号：_____）。我方确认，贵方仅应向该指定账户支付上述应收账款、且只有向该指定账户支付上述应收账款才能解除贵方的付款义务。除商业保理人书面通知（无论单方还是联合我方共同书面通知）贵方外，该收款账户不得变更或撤销。

……

7、如无问题，请贵方于收到本通知书后 日内在本通知书的回执上签章后发送给商业保理人（地址：_____；邮编：_____；收件人：_____；电话：_____）。

非常感谢贵方的理解和支持！

通知人： 应收账款债权人 （签章）

法定代表人或授权代理人（签章）：

签署日期： 年 月 日

通知人： 商业保理人（签章）

法定代表人或授权代理人（签章）：

签署日期： 年 月 日

```
<!DOCTYPE ctml>
    <ctml lang=zh_cn, link="exchangesheet.emd">
    <<factor title@assignNotification>> 应收账款转让通知书 <</factor>>
        <<field@contractNumber>>编号：<{@contractNumber}><</field>>
    致：<<factor party@partyA>><<field partyA@name>><{partyA@name}><</field>>
    <</factor>>（下称“贵方”）
    自：<<factor party@partyB>><<field partyB@name>><{partyB@name}><</field>>
    <</factor>>（下称“我方”）
        我方因经营发展的需要，已申请将下述应收账款及所从属的一切从权利和权益转让给
        <<field@factoringName>><{@factoringName}>（下称“商业保理人”），<</field>>
    兹此通知贵方：
  1、如下应收账款及从属的一切从权利和权益转已转让给商业保理人：
  （1）我方与<{partyB@name}>之间<<field@assetRange>><{@assetRange}><</field>>项下的的全部应收账款以及该应收账款上所从属的一切从权利和权益。
  （2）我方对贵方下表所列的全部应收账款：
    <<factor asset@relatedInformation>>
    <<property info>>
        <<field@assetNumber>>序号：<{@assetNumber}><</field>>
        <<field@assetBuyer>>应收账款债务人：<{@assetBuyer}><</field>>
        <<field@assetBaseContract>>基础交易合同及编号：<{@assetBaseContract}><</field>>
        <<field@documentName>>单据名称：<{@documentName}><</field>>
        <<field@documentNumber>>单据号码：<{@documentNumber}><</field>>
        <<field@accountAmount %Money>>应收账款金额：<{@accountAmount}><</field>>
        <<field@accountDue %Date>>应收账款到期日：<{@accountDue}><</field>>
    <</property>>
    <</factor>>
    2、 请贵方于应收账款到期日将上述应收账款金额足额支付到如下账户（
        <<field@accountHolder>>开户人：<{@accountHolder}>；<</field>>
        <<field@accountBank>>开户银行：<{@accountBank}>；<</field>>
        <<field@accountAccount>>账号：<{@accountAccount}><</field>>）。我方确认，贵方仅应向该指定账户支付上述应收账款、且只有向该指定账户支付上述应收账款才能解除贵方的付款义务。除商业保理人书面通知（无论单方还是联合我方共同书面通知）贵方外，该收款账户不得变更或撤销。
    <<factor genTerm@affordTerm>> 7、如无问题，请贵方于
        <<property preCondition@affordTerm>>
            <<rangePredicate #within=true #boundary=setDate #prep=after #baseTime=receiveLetter>>
        收到本通知书后<<field@setDate>><{@setDate %Time}><</field>>日内
            <</rangePredicate>>
        <</property>>在本通知书的回执上签章后
        <<property action@sendLetter #party=partyA #duty=must>>发送给商业保理人<</property>>
    <</factor>>（
    <<field@factoringAddress>>地址：<{@factoringAddress}>；<</field>>
    <<field@factoringPostalCode>>邮编：<{@factoringPostalCode}>；<</field>>
    <<field@factoringRecipient>>收件人：<{@factoringRecipient}>；<</field>>
    <<field@factoringPhone>>电话：<{@factoringPhone}>）。<</field>>
    非常感谢贵方的理解和支持！
    <<factor conclusion@conclution1>>
        <<property signature#party=partyA>><<field partyA@partyANotification>>
        通知人：应收账款债权人（签章）<</field>><<field partyA@signingPartyA>>
        法定代表人或授权代理人（签字）：<{partyA@signingPartyA}><</field>>
        <<field partyA@partyASignDate %Date>>
        签署日期：<{partyA@partyASignDate}><</field>>
        <</property>>
        <<property signature#party=partyB>><<field partyB@partyBNotification>>
        通知人：商业保理人（签章）<</field>><<field partyB@signingPartyB>>
        法定代表人或授权代理人（签字）：<{partyB@signingPartyB}><</field>>
        <<field partyB@partyBSignDate %Date>>
        签署日期：<{partyB@partyBSignDate}><</field>>
        <</property>>
    <</factor>>
</ctml>
```

图 9 CTML 标注后的保理合同

Fig.9 A factoring contract marked by CTML

5.1 语义提取

保理合同内容可见图 9 左半部分，对该合同文本进行模块分析，包括合同标题、当事人、附加信息、资产、条款、合同订立六大模块。使用章节 3 所述层级标注方法对范例语义部分进行逐层标注。在标记语义过程中，可对需交互数据进行数源信息标记，以便生成交互式数据表（Exchange Markup Datasheet，EMD）。

数源标记标注过程包含以下步骤：

1) 用户选择需标注交互数据在文本中位置，并指定该交互数据所属类型 type；

2) 用户获得自动生成包含文本所在要素的要素标识 factorID 的交互数据标识 exchangedDataID 供使用修订；

3) 根据交互数据所属类型 type 提示用户选择、确认和设定该交互数据 exchangedData 的选择方式 option，供选择的选择方式包括单选 singleOption、多选 multiOption、外部输入 import、触发 trigger、分配 allocate；

4) 将上述用户指定的交互数据标识 exchangedDataID、交互数据所属类型 type、选择方式 option 按照数源标记 metadataExpression 填入到用户指定位置或替代所选需标注交互数据所在的文本。

依照 CTML 语法规则与标注方法完成对保理合同范例的标注，结果如图 9 右半部分所示。

5.2 代码生成能力

CTML 文档经代码生成器映射后自动生成的 SPESC 代码结果如图 10 所示，合同与智能法律合约映射情况如图 11 所示。从结果上看，将使用合同文本置标语言标注的现实合同通过代码生成器自动转化为智能法律合约 SPESC 后，合同内容提取依据清晰，转化规范，同时法律合同元素排列方式更具有条理性。

```
contract assignNotification:  {

    party partyA {
        name : String
        partyANotification : String
        signingPartyA : String
        partyASignDate : Date
    }
    party partyB {
        name : String
        partyBNotification : String
        signingPartyB : String
        partyBSignDate : Date
    }
    asset relatedInformation {
        info {
            assetNumber : String
            assetBuyer : String
            assetBaseContract : String
            documentName : String
            documentNumber : String
            accountAmount : Money
            accountDue : Date
        }
        right {
        }
    }
    contractNumber : String
    factoringName : String
    assetRange : String
    accountHolder : String
    factoringName : String
    assetRange : String
    accountHolder : String
    accountBank : String
    accountAccount : String
    factoringAddress : String
    factoringPostalCode : String
    factoringRecipient : String
    factoringPhone : String
    setDate : String
    term affordTerm: partyA must sendLetter()
        when within setDate after receiveLetter.

    Contract conclusion:

    {
        Signature of party partyA:
        {       printedName:
                signature:
                date:
        },
        Signature of party partyB:
        {       printedName:
                signature:
                date:
        }
    }
}
```

图 10 CTML 标注合同转化为 SPESC 代码

Fig.10 SPESC code converted by CTML

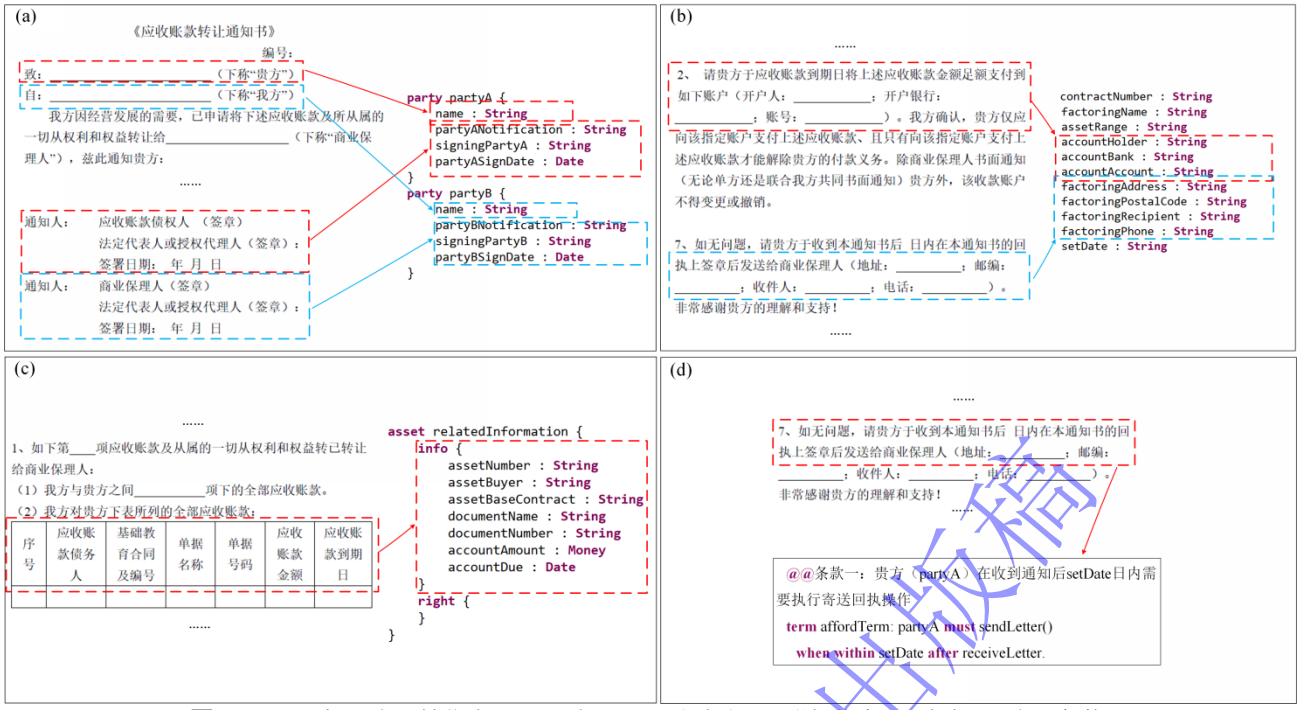

图 11 CTML 标注合同转化为 SPESC 代码 (a) 当事人; (b)附加信息; (c)资产; (d)合约条款;
Fig.11 SPESC code converted by CTML (a) parties; (b)additions; (c)assets; (d)Contract terms;

此外，本研究基于资产证券化 ABS 和供应链融资两个应用场景，分别标注了对应场景中的多份商业保理合同与保兑仓融资合同。通过多个合约组合对场景下复杂业务流程加以标注，验证了方案的可行性，标注结果如表 3 所示：

同时，本研究通过建立线上智能法律合约范例库，集成了包括 CTML 辅助标注，SLC 合约生成等一系列功能：包括合同文本的显示，调用语法解析器自动判断内容，显示解析器可能返回的报错信息与修改提示；CTML 辅助标注通过填入相关语义信息添加 CTML 标记，完成合同法律信息的提取；调用代码生成器，将标注后合同文本转换成对应智能法律合约。

表 3 两种应用场景下 CTML 标注结果
Table 3 CTML annotation results in two scenarios

Contract name	Number of Term	Number of semantic markup	Number of metadata markup
Accounts Receivable Financing Application cum Confirmation	4	37	26
Notification of assignment of accounts receivable	6	36	27
Supply Chain Financing Tri-Party Agreement	9	38	25
Supply Chain Finance Procurement Contracts	8	46	34

6. 总结与展望

本文通过提出合同文本置标语言 CTML，建立一种将法律合同文本中内容和意思表示进行规范化的方法，允许法律人士通过对文档中语法、结构、词汇的内容进行标注，使其他人员更加准确和一致性的理解法律合同文本。同时，CTML 加入了待输入定位的思想，通过实现合同模型化提高生产效率。此外，本文还提出了一种基于 CTML 的智能法律合约生成方法。对于一个给定的采用 CTML 标注的法律合同，通过合同文本置标语言 CTML 到智能法律合约语言的词汇映射和转化规则，利用该文档中标注的层级标记信息生成智能法律合约程序，帮助使用者进行合约编写，做到智能法律合约的编写有据可依。

法律合同一般存在协商-订立-成立-生效等主要阶段。智能合约之智能的法律含义在于合同的成立、生效与执行同步。CTML 作为语法规范的标记语言，实现了现实合同与智能（法律）合约之间的链接：一方面通

过在合同中对语义元素进行标注，依此确定与提取法律元素；另一方面通过转换规则实现由法律文本合同到智能法律合约生成的规范化流程；同时，智能法律合约生成后 CTML 可以在现实合同中进行用户输入标识，通过编译器提取实例信息，结合交换标记数据表 EMD 为智能合约提供输入，辅助完成合约的协商与订立，提高智能法律合约和合同的可复用性。

CTML 依然存在待解决的问题，如：如何在技术上使智能合约适应不同业务与应用场景，如何支持区块链节点的荣誉、激励等实现机制，如何在司法阶段用智能合约实现自动执行与缺席审判。这些问题的解决都需围绕条款的形式一一表达，将在今后工作中将予以关注并努力解决，从而将 CTML 和智能（法律）合约技术用于司法实践，促进法律与科技深度融合。

参 考 文 献

[1] Wei S, Dai K M. An Analysis of Blockchain Applications in Financial Scenarios and an Exploration of Enterprise Software Architecture of Blockchain as a Service (BaaS). *Journal of Guangdong University of Technology,* 2020, 37 (2):1
（魏生, 戴科冕. 区块链金融场景应用分析及企业级架构探讨. 广东工业大学学报, 2020, 37(2): 1）

[2] Civil Code of the People's Republic of China. *People's Daily,* 2020-06-02(001)
（中华人民共和国民法典. 人民日报, 2020-06-02(001)）

[3] He H W, Yan A, Chen Z H. Survey of Smart Contract Technology and Application Based on Blockchain. *Journal of Computer Research and Development,* 2018, 55(11):2452
（贺海武,延安,陈泽华.基于区块链的智能合约技术与应用综述. 计算机研究与发展, 2018, 55 (11): 2452）

[4] Guo S F. Blockchain Smart Contracts in Contract Law. *Oriental Law*, 2019, 3: 4
（郭少飞. 区块链智能合约的合同法分析. 东方法学, 2019, 3: 4）

[5] Chai Z G. Thoughts on Contract Law of Smart Contracts Under Blockchain. *Social Sciences in Guangdong.* 2019, 4: 236
（柴振国. 区块链下智能合约的合同法思考. 广东社会科学, 2019, 4: 236）

[6] Larenz K. *Methodenlehre der rechtswissenschaft.* Springer-Verlag, 2013

[7] Ogilvie J W L. Defining computer program parts under learned hand's abstractions test in software copyright infringement cases[J]. Michigan Law Review, 1992, 91(3): 526-570

[8] Zhou X F, Zhao Z H. Parsing computational jurisprudence. *Communications of the CCF.* 2017, 25: 43
（周学峰, 赵梓皓. 解析计算法律学. 中国计算机学会通讯, 2017, 25: 43）

[9] Zhao J W, Ding H J. On the Regulability of Codes: Foundations and New Developments in Computational Jurisprudence. *Internet Law Review.* 2016(01): 97
（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. 网络法律评论, 2016 (01): 97）

[10] Chen J D. The Legal Structure of Smart Contract. *Oriental Law.* 2019 (03): 18
（陈吉栋. 智能合约的法律构造. 东方法学, 2019 (03): 18）

[11] Liu Q, Wang D J, Wang X X, et al. Review on conformance between legal contract and smart contract. *Application Research of Computers,* 2021, 38(01): 1
（刘琴, 王德军, 王潇潇, 等. 法律合约与智能合约一致性综述. 计算机应用研究, 2021, 38(01): 1）

[12] Accord [EB/OL]. *IBM* (2021-03-17) [2021-04-15].https://accordproject.org/

[13] Winkels R. The openlaws project: Big open legal data//*Proceedings of the International Legal Informatics Symposium (IRIS 2015)*. Universität Salzburg, 2015: 189-196.

[14] SPESC [EB/OL]. *Smart Legal Contract* (2022-06-06)[2022-12-26]. https://www.smartlegalcontract.cn/

[15] He X, Qin B, Zhu Y, et al. SPESC: A Specification Language for Smart Contracts //2018 *IEEE 42nd Annual computer software and applications conference (COMPSAC).* Tokyo, 2018, 1: 132

[16] University of Science and Technology Beijing, T/CIE 095-2020 *Formal Expression of Blockchain Smart Contract*. Beijing: Chinese Institute of Electronics, 2020

[17] Zhu Y, Qin B H, Chen E, et al. An advanced smart contract conversion and its design and implementation for auction contract. Chin J

Comput, 2021, 44(3): 652
（朱岩, 秦博涵, 陈娥, 等. 一种高级智能合约转化方法及竞买合约设计与实现. 计算机学报, 2021,44(03): 652）
[18] Mernik M. When and how to develop domain-specific languages. *ACM Computing Surveys,* 2005, 37 (4): 316
[19] Eysholdt M, Behrens H. Xtext: implement your language faster than the quick and dirty way// Companion to the 25th annual ACM Sigplan Conference on Object-Oriented Programming, Systems, Languages, and Applications, Splash/oopsla 2010, October 17-21, 2010, Reno/Tahoe, Nevada, Usa. Reno.2010:307
[20] ISO. ISO/IEC 14977: 1996 *Information technology—Syntactic metalanguage—Extended BNF*. London: ISO, 1996
[21] Wile D S. Supporting the DSL spectrum. J*ournal of Computing & Information Technology,* 2001, 9(4): 263
[22] Xtext Language Implementation [EB/OL]. Eclipse (2022-10-27) [2022-12-27]. https://www.eclipse.org/Xtext/documentation/303_runtime_concepts.html
[23] LSP/LSIF[EB/OL]. *Microsoft* (2021-03-11)[2021-03-16]. https://microsoft.github.io/language-server-protocol/
[24] University of Science and Technology Beijing, T/CIE 129-2021 *Blockchain—Smart contract—Contract text markup language* (*CTML*). Beijing: Chinese Institute of Electronics, 2022
[25] Chalkidis I, Androutsopoulos I, Michos A. Extracting contract elements[C]//Proceedings of the 16th edition of the International Conference on Articial Intelligence and Law. 2017: 19-28.
[26] Aejas B, Bouras A, Belhi A, et al. A Review of Contract Entity Extraction[C]//Proceedings of Sixth International Congress on Information and Communication Technology: ICICT 2021, London, Volume 4. Springer Singapore, 2022: 763-771.
[27] Wyner A, Peters W. Towards annotating and extracting textual legal case factors[C]//Proceedings of the Language Resources and Evaluation Conference Workshop on Semantic Processing of Legal Texts, Malta. 2010.
[28] Wyner A, Peters W. On rule extraction from regulations[M]//Legal knowledge and information systems. IOS Press, 2011: 113-122.
[29] Xtend[EB/OL]. *Eclipse* (2022-12-15) [2022-12-29].https://www.eclipse.org/xtend/
[30] Bettini L. *Implementing domain-specific languages with Xtext and Xtend*. 2nd Revised. USA: Packt Publishing Ltd, 2016.