区块链上实现法律效力的自我意识智能合约管理
第一版
（草案）

摘要:本白皮书旨在为传统合约因不具备计算机可读性而带来的问题提供解决方案。传统合约制订步骤复杂，不与 ICT 系统相连，执行困难，出现争议时跟踪合约执行情况的效率就会大大降低。自我意识合约和传统合约有相似的法律效力，但后者是一种计算机可读合约，以区块链技术为支持。鉴于区块链技术已经提供了去信任化的环境，自我意识合约不再需要合约方相互信任，相关事件一旦保存便不可更改。但是，当前的计算机可读合约（如智能合约）缺乏适用于执行合约时的义务建构，也无法理解不断变化的法律关系。所以，在智能合约中用日常行为包装法律义务变得十分重要。针对这一点，本白皮书提出了 Agrello 框架，实现以区块链为驱动、以智能代理为支持的自我意识合约，进而为去中心化的点对点经济带来了可能。预先存在的研究结果和示范原型体现了 Agrello 框架的可行性。

关键词：自我意识；多智能代理；区块链；智能合约；去中心化；点对点；电子治理；人类可读

第一章 引言

人们对于传统合约的理解是“合约双方交换具备法律强制执行效力的承诺的行为”。合约成立的重要前提是合约双方自愿达成共识，这一点最常见的书面合同就可以佐证。在大多数商业案例中，传统合约的作用是定义合约双方并明确双方承诺。履行承诺的行为一旦开始，合约状态就会发生改变。这种传统的合约制定和管理方式还有一个问题，不规范限制了人工跟踪合约状态的能力。人们对传统合约状态缺乏总体认识，双方关系就容易产生争议，由此导致的高昂冲突解决方案甚至可能会摧毁整个合约。而且，执行这类传统合约要么复杂费时，要么根本无法实施，当然前提是在国际背景下。

《迈向基于数据意识过程的共享账本业务协同语言》（*Towards a Shared Ledger Business Collaboration Language Based on Data-Aware Processes*）的作者认为，共享区块链技术不仅可以实现可靠性高的业务协作，还能为智能合约提供集共享、信任、保护隐私和不可更改等属性为一体的数据存储库。实现数据意识过程的业务部件是共享区块链技术的基础，而区块链可以使业务协作语言发展到以太坊编程语言 Solidity 的级别。在《用区块链监督和执行不可信业务过程》（*Untrusted Business Process Monitoring and Execution Using Blockchain*）中，作者将一项协作过程的运行环境映射到智能合约的脚本语言中。这一方法解决了协作过程中的信任问题，不再需要第三方实体对事件进行

监督。至此，区块链实现了去信任过程协作。将协作过程映射到区块链中还可以监督协作的执行情况以及审计相关事件。《用于构建区块链系统的基于逻辑的智能合约评估》（*Evaluation of Logic-Based Smart Contracts for Blockchain Systems*）对不同的智能合约语言进行了比较。虽然使用流程语言是当前的一贯做法，但同样也可以选择基于逻辑的语言。区块链与基于逻辑的语言结合的关键所在是算法必须高效。时间管理是智能合约的一大重要元素。区块链产生间隔时间(blocktime)是指验证一个新区块所需要的时间，不同于人类时间和计算时间。区块链产生间隔时间为用智能合约实现对未来事件进行稳健编制带来了可能。《修改和撤销智能合约》（*Setting Standards for Altering and Undoing Smart Contracts*）的作者强调要让合约方都可以修改或撤销合同。由于后者在智能合约中失败，现在需要制定出一套修改和撤销智能合同的新规范。

可以看出，区块链技术是部分智能合约存在的基石。但是，要将智能合约发展为自我意识合约，目前还缺乏具体实施框架。自我意识合约可以收集内部上下文和外部上下文的状态和进度信息，并在充当法律部件时推断合约方的行为。而且，当前的最新技术并不认可“这类自我意识合约仅限于满足制定法律合约这一个需求”的说法。针对这一技术空白，本白皮书提出了“如何让具备人类可读性的自我意识合约拥有法律可行性？”这个问题。为降低该问题的复杂程度并提取出关注点，我们又提出了三个子问题，分别为：“合约具有自我意识的条件是什么？”、“怎样才能使自我意识合约具备人类可操控性？”以及“为实现法律可行性，哪些条件可以确保合约不可更改？”。

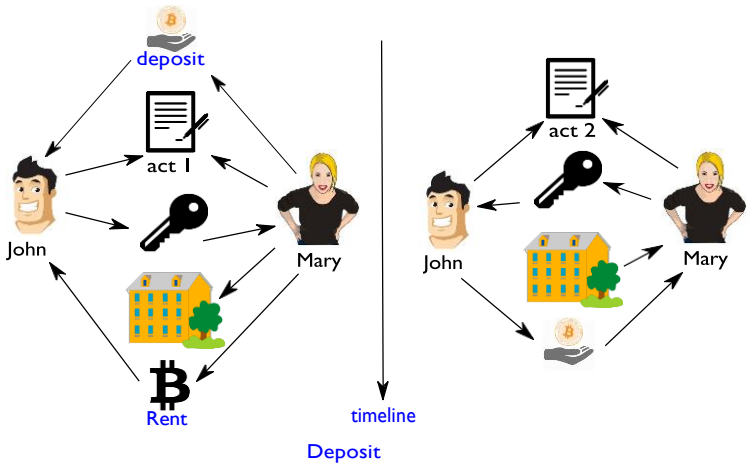
本白皮书结构如下：第二章呈现了一个自我意识合约管理的运行环境，并例举了相关文献，为接下来的章节进行铺垫；第三章强调了自我意识合约中的基本内容与需要监督的业务过程映射之间的重要关系；第四章讨论了人类参与自我意识合约生命周期中的重要意义；第五章探讨了适当整合目前哪些现有的区块链技术可以实现对合约要素的信任管理；第六章运用概念验证原型，评估了第二章中的案例结果；第七章对全书进行了总结，并探讨了自我意识合约的局限性、未解决问题和未来工作。

第二章 运行环境和背景文献

本章第一节将现实生活中的租房合约设置为运行环境；第二节介绍相关文献，供读者了解接下来的章节。在本文中，我们用“获益人”代表“债权人”，用“义务人”代表“债务人”。

第一节 运行环境

图 1 描述了出租方和承租方签订租房合约的过程。出租方是房屋所有人或其代表人（代表人有权代表产权人签订租赁合约）。我们称出租方为“John”。本场景中，房屋属于法律上的不动产。不动产包括土地和土地着定物，如房屋、公寓楼、公寓间、车库、停车场或用于出租的棚屋。房屋还可以属于动产，如房车和工具等。承租方是寻找出租房屋进行长期或短期居住的人。我们称承租方为“Mary”。



(a) 合同签订 (b)合同到期和终止

图 1.（a）展现的是非正式签订协议。John 向出租方支付押金后，双方在租房合约中签名（协议一）。John 将钥匙交给 Mary，Mary 每月向 John 支付一次房租。（b）展现的是非正式到期与终止协议。双方签订房屋收回协议（协议二），协议可能会要求租户维持房屋状态，须与合约签订时的房屋状态一致。Mary 交还钥匙并搬出，同时 John 退还押金。

租房合同的生命周期分为以下几个阶段：准备、协商、执行、回滚和到期。图 3 中，Mary 寻找合适房源租住 12 个月的需求触发该租房合同进入准备阶段。在该阶段中，John 和 Mary 需要确定具体规范要求。在此之后，合约进入协商阶段。Mary 需要了解房屋本身和房屋产权所有人的相关信息，同时 John 也在了解 Mary 的信息。合约必须包含双方的姓名、身份证号码、地址等各种数据。房屋的状况也必须在合约中写明。合约目的由房屋的功能决定，如房子的地理位置（地址）、面积（平方米）和使用用途（居住、仓储或办公）。合约条款一般由 John 负责插入。但在协商阶段，条款可以根据 Mary 的需求进行修改。比如，John 需要确定他是想长期出租还是短期出租。

找出租房的传统做法是承租方在浏览网上租房广告或雇佣房屋中介与出租方就合约条款进行协商。这说明 Mary 必须花时间在搜索租房信息、打电话给房东和看房上，虽然房屋中介会帮你省去这些麻烦，但会收取一定的劳务费。在合约协商阶段时，John 会预先定义基本条款（出租对象、出租时长、租金和承租方信用评级），并在此基础上寻找和他的设定最相符的租客。

如果 Mary 与 John 的要求匹配，他们就有可能签订合约。如果 Mary 拒绝签订，说明她不赞成合约中的部分条款，这时 John 就要修改他的租客搜索标准，寻找更符合他要求的租客。如果合约签订成功，那么双方已经表达出缔结合约的意愿(如签字确认)，而第三方无权介入更改。

对房屋状况进行记录标志合约进入执行阶段。在传统意义上，这份记录是使用权转让协议，包括房屋的使用情况、水电气暖表的当前读数以及交给承租方的钥匙数量。John 要将公寓钥匙交给 Mary，Mary 必须为此支付押金。在此之后，Mary 有义务每月准时向 John 支付房租，并妥善使用该公寓。如果 Mary 未能及时交付房租，那 John 有权向其收取滞纳金。在合约有效期内，如果 John 无法继续将公寓出租给 Mary，则合约回滚，John 必须退还 Mary 预先支付的房租。

无论是合约期满还是提前终止，公寓都会回到 John 手中。合约到期时，John 希望此时的公寓状况和他转交给 Mary 时的一样。财产转让程序与以上流程相类似，水电气表的状态也必须进行记录。

第二节 相关文献

第三章 合约的自我意识

当前智能合约的通用编程语言是 Solidity，非专业人士根本无法理解。这样

的智能合约与第三方中心化机构相比没有任何优势，缺乏法律层面的适用性、实用性和可表达性。比如，Solidity 就不包含合约双方权利与义务的语言建构。

人们签订传统合约的目的是明确各方权利与义务，建立合约关系并管理各方行为。随着信息物理系统的出现，智能合约需要具备推断权利与义务的能力，而 BDI 智能代理为之带来了可能。于是，基于各种社会技术应用场景，我们设计出了大量智能合约，让人们以协作的方式解决问题。传统合约下，律师必须不断检查合约状态，防止错过合约期限或监督是否有人违反义务。而在具有自我意识的智能合约中，传统合约与律师合并成为软件智能代理，由计算机可读的合约义务逻辑操作。更准确地说，这样的智能代理具备推断能力（如推断是否错过履行义务的最终期限）。鉴于我们将这类义务智能代理视为智能合约，我们得出结论：智能合约具备推断自身行为的能力。

本章结构如下：第一节讨论本体论与合约义务属性；第二节正式地展示出智能代理的义务处理过程；第三节解释 BDI 智能代理在智能合约管理中的使用情况。

第一节 合约内容

设计自我意识合约，首先要保证合约的重要元素可以在合约执行阶段提供元数据。之后，元数据可通过信息系统运用在许多方面）。但最为重要是，要确保起辅助作用的智能代理能够自动操作和管理合约的执行流程。

正如前文所说，智能合约的核心部分是权利与义务，而这一部分需要针对计算机可读性进行优化。接着，我们会介绍什么是权利与义务以及它们的重要性。最后，我们会解释在保证非专业人士有能力通过智能合约判断权利与义务的情况下，实现计算机可读的权利与义务。

图 4 是 Agrello 框架本体论的层级表。其中的次层级关系包含了合约的所有基本要素。比如，我们通过增加**金钱义务**和**非金钱义务**这两个次层级完善了对义务的类型，以此表明某些补救措施只适用于非金钱义务（如修理或替换），某些只适用于金钱义务（如滞纳金）。这里还包括人的次层级，如**义务人**（必须履行义务的人）、**获益人**（因别人履行义务而从中获益的人）以及视情况选择的**第三方**（也可因别人履行义务而从中获益的人，如租房合约中的水电气提供方）。

[10] Agrello-OWL: http://tinyurl.com/lkkapvg

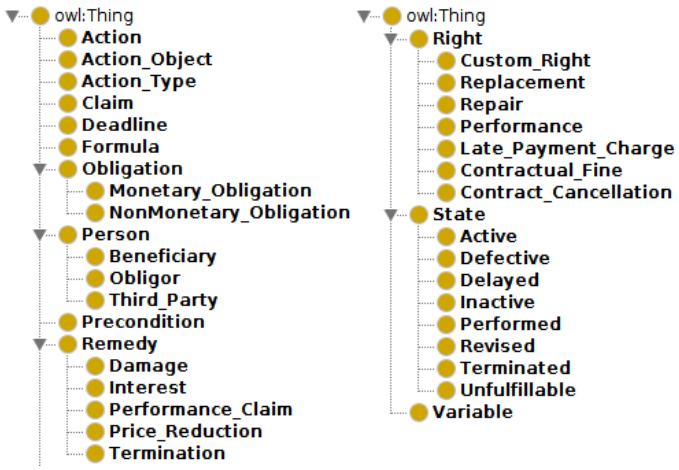


图 4. Agrello 框架本体论层级表。

图 4 中，设置**补救措施**次层级的目的在于消除因违约而造成的负面后果。通过实行补救措施，获益人可以获得补偿，不必受到义务未正确履行而带来的影响。比如，如果承租方逾期未付房租，则出租方有权向其收取滞纳**利息**。

权利次层级直接反映了获益人握有哪些权利，所以十分重要。比如，承租方损坏家具，出租方有权要求其进行修理或替换。**状态**次层级反映了合约生命周期中**义务**的执行情况。

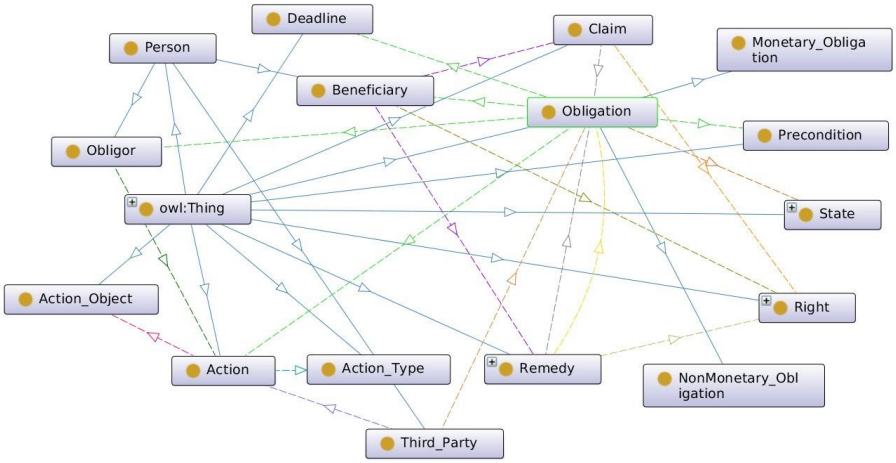


图 5. 义务本体论图。

在主层级方面，我们着重强调权利与义务。图 5 展示的是**义务**层级的主要本体关系。图 5 列出了**义务人**、**获益人**、**先决条件**、**行为**和**截止日期**，而且还展示出了**补救措施**基于**义务**，**第三方**履行**义务**，**权利**创造**义务**，**义务跟随**合约生命周期**状态**变化。

图 5 中，**先决条件**是指是形成一项**义务**必须满足的条件。**行为**是义务人未实现**获益人**利益必须进行的任务，如“支付房租”。**行为**有两个属性——**行为种类**（如“付”）和**行为对象**（“租金”）。**截止日期**是指**义务**必须在何时完成。

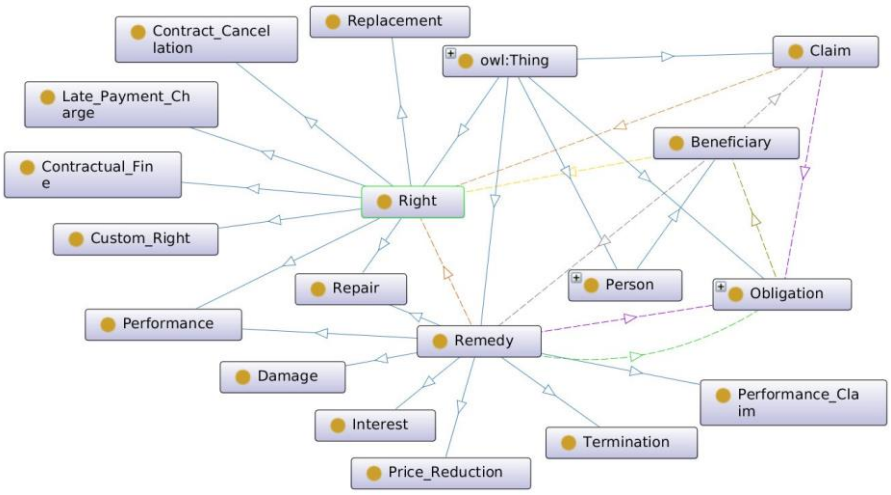


Fig. 6. Right ontology graph.

图 6. 权利本体论图

图 6 展示了与权利层级相关的静态关系。与图 2 不同的是获益人拥有合约中规定的权利，并在对方违约的情况下，有权要求使用补救措施。如果承租方损坏了屋内的某样东西，则出租方有权要求其进行修理或重新购买并赔偿。

第二节 义务处理

在合约的生命周期内，义务贯穿各处理阶段。根据图 4 的本体论层级表，这些阶段可以分为**未激活**、**激活**、**执行**、**延迟**、**瑕疵**和**终止**。除此以外，还存在**修改**和**未完成**阶段，但这不是自动化义务处理的重点。更准确地说，我们将讨论义务处理的以下阶段：

- 未激活：智能代理还未接收到义务，如义务的先决条件还未达到。
- 激活：智能代理接收到义务，如义务的先决条件已经达到。这就可以推断出，义务人必须在截止日期之前完成相关行为。
- 执行：义务人已经完成某行为。
- 延迟：义务人未在规定期限内完成某行为。延迟状态意味着，义务中的行为对象数量还未递交给获益人，或是递交给获益人的行为对象数量不足够。
- 瑕疵：义务的行为对象有瑕疵。
- 终止：义务的履行可以在重大违约或达成共识的情况下终止，不会再继续履行。

根据图 7 的 CPN 模型，当义务处于**延迟**或**瑕疵**阶段时，合约智能代理就会启动，推断违约可能性，告知另一方他们有权采用补救措施或其他冲突解决办法。在**延迟**阶段，义务的行为对象在截止日期之前还未被递交，或递交数量不足够。比如，租金未付或租金少付。

图 7 中，**瑕疵**阶段可以通过金钱义务和非金钱义务区分。金钱义务包括的是金钱行为，非金钱义务包括的是非金钱行为。比如，支付租金是金钱义务，转让公寓的使用权是非金钱义务。只有非金钱义务才可以进入义务的**瑕疵**阶段，因为它的条件是与合约相比，行为对象未达到规定的质量要求。比如，承租方归还公寓的使用权时，房屋状况与出租方要求得不一致。支付租金就与之相反，因为这不是质量要求，而是数量要求。即便义务已经进入执行阶段，但如果后期发现瑕疵，还是会被改为**瑕疵**阶段。

图 7 中，在义务的**延迟**和**瑕疵**阶段，获益人都有权要求采用补救措施。**延迟**阶段的补救措施激活了要求对方支付滞纳金的金钱义务和支付违约金的非金钱义务。而**瑕疵**阶段只能激活非金钱义务，获益人可以要求义务人进行修理或替换，同时也可以要求其进行赔偿。

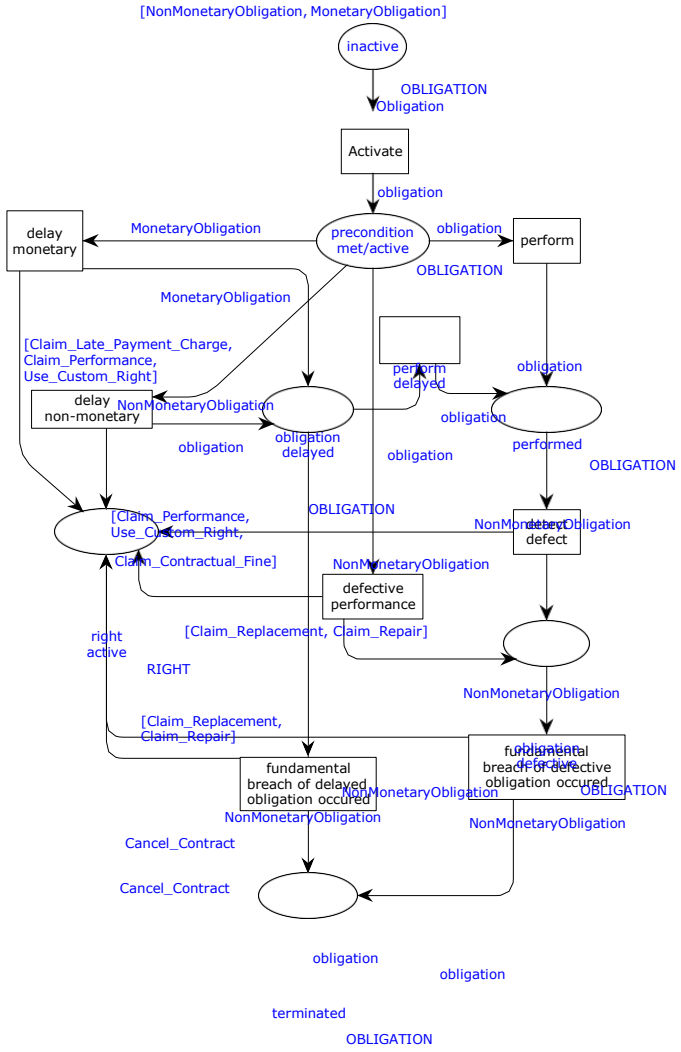


图 7. 义务的交易处理。

如果补救措施也无法帮助获益人实现要求对方履行义务的目的，这时对方即构成重大违约，义务会进入**终止**阶段。终止阶段激活了获益人取消合约的权利。义务也可以在双方达成共识的情况下进入**终止**阶段。

第三节 互动型合约智能代理

我们将第二节中的租房运行环境用 UML 顺序图进行了优化，描绘了各智能代理之间的互动协议。图 8 是对图 1（a）优化过之后得到的顺序图，描述的是租房合约的签订，图 9 对图 1（b）优化过之后得到的顺序图，描述的是租房合约的终止。

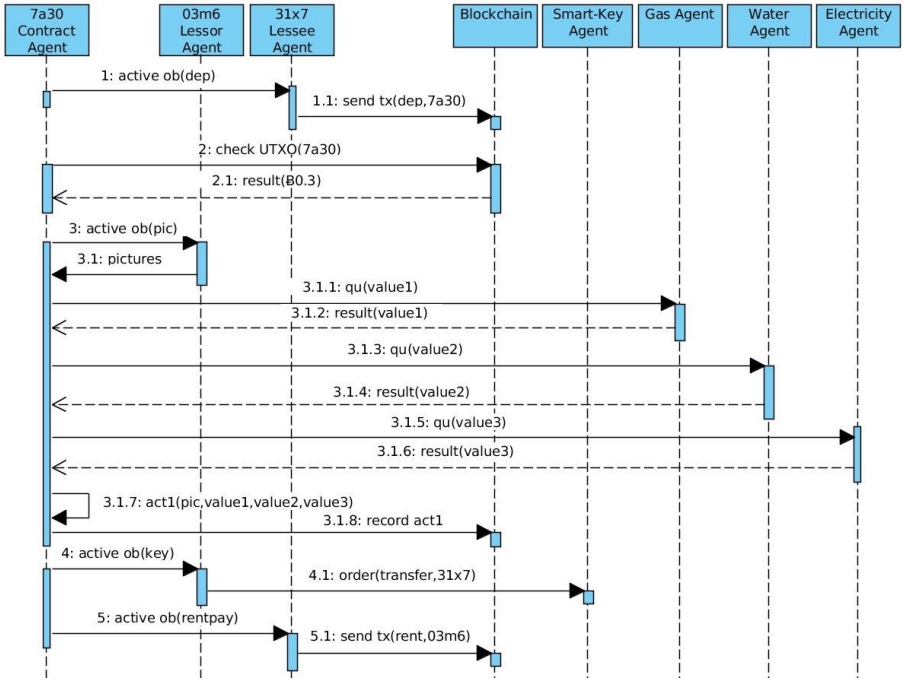


图 8. 合约智能代理的签订协议。

图 8 中，为实现可读性，我们设置了一个四字符的虚拟公钥。左侧的三个实体分别代表合约智能代理、出租方智能代理和承租方智能代理。图中的第四个实体代表区块链，所有事件都会被记录在上面。我们假设一个智慧型家庭的场景。该公寓由四个智能代理管理，其中一个管理智能钥匙，另外三个分别管理水电气智能仪表。

图 8 的起点是合约智能代理向承租方智能代理发送信息，告知其义务 *ob(dep)*被激活，这代表承租方要支付押金。于是，承租方智能代理向区块链发送 *tx(dep, 7a30)*消息，执行押金支付行为（如押金由合约智能代理保管）。请注意，押金一般直接进入出租方账户，且只可用于修理公寓损坏后的修理。如果出租方独占该笔资金，会引发各种问题。即便公寓未被损坏，偶尔也会出现不退还押金的情况。但如果押金是由合约智能代理保管，各合约方则必须对押金的处理达成共识。

图 8 中的第二条消息是由合约智能代理发送给区块链的，通知区块链检查 *UTXO(7a30)*，确认押金已转入合约智能代理在区块链上的钱包。区块链会进行确认回复 *result(B- 0.3)*，表示押金已到达合约智能代理在区块链上的公钥地址。

要形成图 1 中协议一转让协议，必须收集以下几类信息。一，合约智能代理向出租方智能代理发送义务激活消息，请求访问公寓情况的图片。出租方智能代理发送图片进行回复。接着，合约智能代理会将 value-query 消息 *qu(value)*分别发送给水电气智能代理，水电气智能代理会回复 *result(value)*消息，其中写明目前智能仪表上的读数。合约智能代理会用这些读数和公寓照片生成协议一并保存在区块链上。

二，合约智能代理向出租方智能代理发送义务激活消息 *ob(key)*后，出租方智能代理向智能钥匙智能代理发送命令消息 *order(transfer,31x7)*，如承租方现

已可以使用该公寓智能钥匙。请注意，通过使用区块链来分配智能钥匙，说明可以将一项任务分配给好几个出租方已经知晓其身份的人。最后，合约智能代理向承租方智能代理发送义务激活消息 *ob(rentpay)*，承租方智能代理会向区块链发送交易 *tx(rent,03m6)*，如第一个月的房租接收人是出租方。

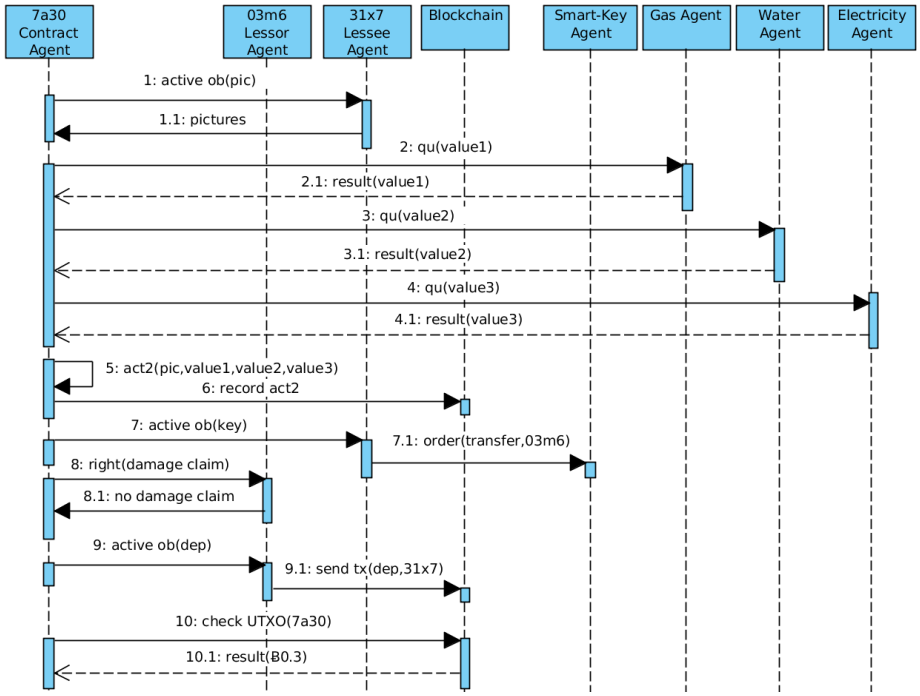


图 9. 合约智能代理的终止协议。

图 9 的租房合约终止协议的起点是合约智能代理向承租方智能代理发送义务激活 *ob(pic)*消息，承租方智能代理会发送当前公寓状况的图片进行回复。然后，合约智能代理会向水电气智能代理发送 *qu(value)*消息，获取智能仪表的当前读数。水电气智能代理会回复*result(value)*消息，发送各智能仪表上的读数。假设出租方对传送回来的公寓状况照片表示接受，合约智能代理就会执行命令 *act2(pic,value1,value2,value3)*，并将协议二保存在区块链中。接着，合约智能代理会向承租方智能代理发送义务激活信息 *ob(key)*，说明此时承租方要将智能钥匙交还给出租方。于是，承租方智能代理向智能钥匙智能代理发送 *order(transfer,03m6)*消息。

合约智能代理向出租方智能代理发送 *right(damage claim)*消息，告知出租方有权进行最后一次验房，检查房屋是否需要进行损坏赔偿。假设图 9 中的房屋没有损坏，合约智能代理会向出租方智能代理发送义务激活信息 *ob(dep)*，要求将押金归还给承租方。于是，出租方智能代理向区块链发送交易信息 *tx(dep,31x7)*。最后，合约智能代理向区块链发送检查命令 *UTXO(7a30)*，区块链发送 *result(B- 0.3)*消息进行回复，如押金已交还给承租方。

第四章 自我意识合约的可管理性

Agrello 框架旨在提高信息和价值传输的效率。对于后者，在租房合约这样的运行环境里需要进行大量的编排工作。我们必须认识到，合约的生命周期必须符合合约的制订、实施、回滚和终止。因此，本章第一节描述了自我意识合约的生命周期；第二节强调 BDI 智能代理在合约生命周期的参与情况；第三节讨论人类与自我意识合约的互动方式。

第一节 自我意识合约的生命周期

图 10 中，我们再次用 BPMN 法表现了智能合约的生命周期。该合约生命周期的起点是各方想要制订点对点合约进行协作。第一个阶段是准备合约模板（包含服务类型和相关智能代理的角色）。我们假设有一个数据库，里面保存了许多第三方制订的租房合约模版，并且已经设定了预定义参数，比如租金的价格范围。其次，服务类型由提供服务的特定智能代理确定。在本白皮书的运行环境中，智能代理的角色有出租方智能代理、承租方智能代理、智能钥匙、水电气智能代理及区块链。

相较于传统方法，使用智能合约租房有优势。使用传统方式租房，水电气提供方通常由出租方决定，承租方没有选择的权利。但在智能合约中，出租方会根据承租方的要求更换智能代理和服务类型，让自己更有竞争力。比如，承租方有环保意识，对目前的燃气智能代理不满意，那出租方就可以寻找环保的燃气提供方进行替换。这样，租房合约的准备阶段中的一点竞争，却提高了承租方找到最佳服务提供商的可能性。

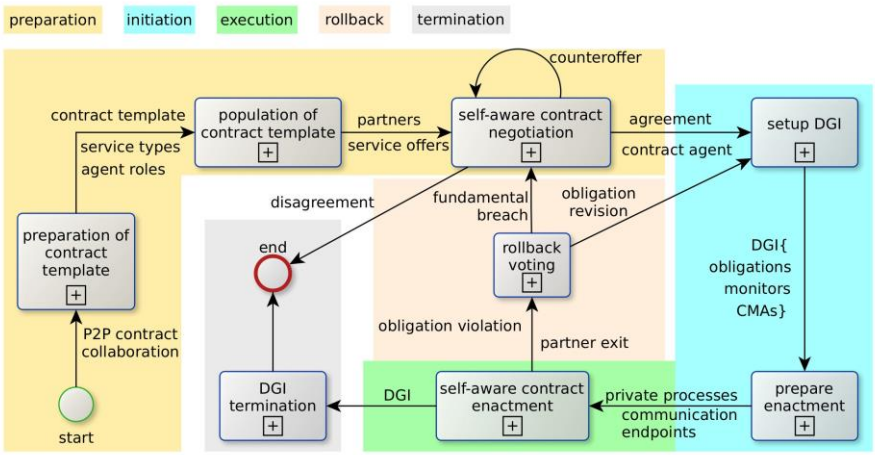


图 10. Agrello 自我意识合约的生命周期。

图 10 中，智能代理会在协商阶段确定各自的价格并组装协议合约。各智能代理会收到一份合约副本用来协商。协商结果一般有三种：1）承租方智能代理不接受该合约中的月租金，进行还价。这意味着所有智能代理将收到一份更新后的合约副本，用于下一轮协商；2）有智能代理不接受该合约，协商失败。这种情况下，合约的生命周期终止；3）所有智能代理接受该协议，自我意识智能合约智能代理由此产生。

合约智能代理产成后，合约进入签订阶段。但在接下来的执行阶段，我们还需要建立一个去中心化的管理基础设施（DGI），用于分配合约智能代理对其他各类智能代理的义务集合。此外，我们还在合约中设置了监督者的角色和合约监督智能代理（CMA），用于监督各项义务是否得到履行。如果出现违约现象，合约监督智能代理要向合约智能代理汇报，并终止执行接下来的步骤。为租房合约进入执行阶段做准备是说在技术层面上为所有参与协作的智能代理建立单独通道和可以清晰地展示出异构数据对的通信端点。

在合约的执行阶段，承租方每月向区块链支付房租。如果缴费期限即将到期，但承租方还未支付，那么先前建立的合约监督智能代理就会对其进行监督。

若义务未履行，投票程序就会启动，来决定这一情况是属于重大违约还是需要对义务进行修改。在租房合约的情境下，违约就是指承租方迟交房租或拒绝支付房租。如果房租中还包含每月的水电费，那么水电气智能代理也要和出租方一起参与投票。那么在这种情况下，各方的投票比重就由它们的应收费用占房租的比重决定。

如果承租方拒绝支付房租，那么投票结果将会是重大违约。该合约会进入重新协商过程，以调查清楚究竟是哪些问题导致承租方拒绝支付房租。虽然重新协商会让合约暂停并很有可能回滚，但合约内容在该阶段内不会更改。如果重新协商失败，该租房合约则完全终止，参与在内的各智能代理也会随之终止。如果承租方只是逾期未缴纳房租，投票结果很可能是要去承租方补交租金并支付滞纳金。如果导致这一结果的原因是承租方的个人情况与制订该合约时的情况不一致（如收到薪水的时间推迟），那我们就要对支付义务进行修改。在这种情况下，我们会在合约回滚到签订阶段的过程中对承租方义务进行调整，该合约就可以继续生效。在合约的完全终止阶段，去中心化的管理基础设施会被撤销，所有的智能代理同样将全部停止协作。

第二节 BDI 智能代理的参与

在自我意识租房合约中，一共有八个智能代理，每个智能代理都有不同的任务。我们将逐一介绍它们的职责、限制因素以及分别在智能合约的哪个生命阶段出现。

合约智能代理类似于房屋中介。合约的准备阶段即将结束时，如果其他智能代理对当前合约达成共识，合约智能代理就会产成。它负责推断 DGI，出租方智能代理和承租方智能代理由此可获得本地义务集合。合约智能代理还要协调各合约监督智能代理。如果有本地义务集合需要完成，后者代表合约智能代理进行监督。除此以外，合约智能代理还要在合约的订立阶段担负起众多职责。在合约执行阶段，合约智能代理要听从合约监督智能代理的意见。如果房租未及时交付，合约智能代理就要触发投票程序，出现之前介绍过的合约回滚情况。合约智能代理还负责接收来自出租方或承租方终止合约的请求，并解散去中心化的管理基础设施。合约智能代理的基本限制因素是模版中的合约各方和相应智能代理角色必须在准备阶段就预先设定。各智能代理只有达成共识，合约智能代理才会为 DGI 的建立而实例化。按要求传送用于生成协议一的数据（如房屋照片和水电气智能仪表上的读数）这一点也很重要。合约智能代理的另一个限制因素是在合约回滚阶段，所有智能代理都必须参与投票。

在合约的准备阶段，**出租方智能代理和承租方智能代理**都需要在合约模版中确定相应的角色和服务类型。它们必须要参与合约的协商过程，进行还盘，表达意见，最终达成共识。它们还需要在合约的签订阶段和其它智能代理开展合作，收集生成协议一和协议二的数据（如图 8 和图 9 所示）。出租方智能代理向承租方转让智能钥匙。在合约的执行阶段，承租方智能代理每月在区块链上支付一次房租。如果承租方未能履行支付房租的义务，承租方智能代理有义务配合倒退程序，要么同意履行一项新义务，使该合约继续成立，要么在构成重大违约的情况下支付赔偿金。如果合约智能代理收到合约监督智能代理的报告，被告知有违约现象，则出租方智能代理有义务配合执行回滚投票程序。

承租方智能代理的限制因素是在合约签订阶段支付押金的能力和在合约执行阶段支付租金的能力以及在合约倒退阶段对房屋造成的损坏和支付的赔偿。在合约终止阶段，承租方智能代理的限制因素是房屋情况的照片应与其搬入时所拍摄的照片一致。如果房屋情况变差，承租方必须向出租方的区块链地址支付赔偿金。在合约签订阶段，出租方智能代理的限制因素是向合约智能代理传送房屋的照片和未在可接受的时间范围向承租方发送智能钥匙。在合约终止阶段，出租方智能代理必须在可接受的时间范围内检查房屋的损坏情况。

在合约的准备阶段，**水电气等公共事业智能代理**需要完成各自在合约模版中的任务，并在合约的签订阶段汇报智能仪表的读数。假设水电费是房租的一部分，如承租方未支付租金，则水电气智能代理必须参与合约的倒退投票程序。在合约终止阶段，水电气智能代理必须再次发送智能仪表的读数以确定协议二（图 9）。水电气智能代理的限制因素条件是无法在规定时间内完成各自任务。

智能钥匙代理在合约签订阶段移交给承租方，在合约终止阶段被交还给出租方。除此以外，智能钥匙智能代理在自我意识合约的整个生命周期（见图 10）中无其他职责。唯一的限制因素条件是立即回应用户更改的命令。最后，我们要强调是区块链不是智能代理，只是合约生命周期中的一个不可更改的用于记录数据的账本。

第三节 人类互动的方式

图 11 的组织模式展示了人类在租房合约运行环境中的参与情况。组织模式面向智能代理建模符号的一部分，指示了人类智能代理和 BDI 智能代理之间的关系。箭头用以明确关系类型。下图中，我们用“控制”来描述各智能代理之间的从属关系，用“等于”描述平等角色之间的关系，用“顺从”来描述人类智能代理之间的关系。

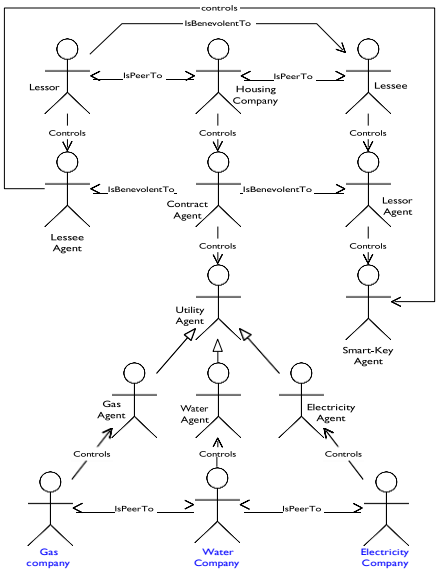


图 11. 租房合约的组织模式。

图 11 中，人类成功参与智能合约是在 BDI 智能代理的协助下实现的。举个例子，在该租房合约中，承租方智能代理承租方（人类），燃气智能代理代表燃气公司。这些类似于人类代理的 BDI 智能代理在合约智能代理的管理下进行协作，而合约智能代理又由物业公司管理。智能钥匙代理被出租方智能代理和承租方智能代理轮流管理。

对于人类和对应的智能代理之间的互动，我们认为可以通过移动设备进行实现。Qtum 系统是目前适用基于智能合约交易的唯一平台，可以在移动设备上运行轻钱包。这是因为 Qtum 平台创新地将比特币的特点和以太坊虚拟机合二为一，取长补短。尽管以太坊的账户模式类似于银行账户，Qtum 采用的则是 UTXO 协议和比特币简单支付验证结合的方式。因此，比特币不具备图灵完备智能合约语言的缺点被消除，能够适应以太坊虚拟机交易模式并适用 Solidity 语言。因此，Qtum 提供生产大量只能管理哈希头的轻钱包，同时兼顾以 Solidaty 语言写成的智能合约，使用与移动设备相连的轻钱包中的 Qtum 版本以太币和 Gas。因此，图 11 中的人类智能代理有望于通过自我意识合约在移动设备上使用这类轻钱包。

第五章 合约的信任要素

合约的不可更改性和法律可行性存在这样几个问题：不可更改不仅意味着要保存计算机可读的智能合约，还要保存会影响合约不可更改性的事件（如付款）。在传统合约中，这些事件包括发票、电邮或出租方和承租方之间的通话记录。只要保持这些事件不变，就可以避免合约双方出现各执一词的情况。让智能代理监督合约的执行情况也是一个前提条件。有一个单独的合同智能代理与事件存储库连接，专门负责处理有问题的合约。它不停地使用相关事件，处理判断合同中的义务问题。除了保证价值不可变更，合约智能代理还要保证这些事件的时间戳不可变更。

本章结构如下：第一节列举了自我意识合约的基本要素；第二节讲解了执行这类合约的技术细节；第三节描述了合约执行的其他信任要素；第四节解释了上下文可信度。

第一节 构成信任的元素

利用区块链技术执行自我意识智能合约，需要满足以下要素：

身份：合约双方的身份必须明确。出租方希望能将房子租给回应他们报价的人。假设有一个信用评级良好的人回应了出租方的报价，但在签名时使用的身份无法核实，那么这个租客可能会让另一个信用评级不够的人搬进来，比如他的朋友或者家人。

签名：传统合约要求手写签名。为了能让自我意识合约有同等或更高的法律效力，数字签名或加密签名必不可少。

事件：正如上文所说，要自动执行合约必须获取与合约义务相关的外部事件。

如果合同智能代理要了解这个月的房租是否已缴纳，那它要么是从外部被告知，要么是从向保存着各类事件的区块链询问获得，或者是从该区块链相连的其他区块链处间接获得。不同类型的事件不必保存在同一个区块链中，比如获得智能仪表的传感数据或访问同一个智能锁都可以在不同的区块链上进行。

时间戳：这类自我意识合约会通过考量外部事件来处理各种义务(如支付义务)。为了推断出延迟时间和截止日期，合约智能代理处理各种项目时都需要打上时间戳。

合约源代码：源代码是一种用计算机可读的正式符号写成的合约义务。为保证合约不可更改，合约的源代码和相应的哈希值必须保留在区块链上。如果有一方声称因源代码更改而导致智能代理执行合约的过程中出现瑕疵，必须要使用原始代码的副本才能解决问题。只要其中有一方可以提供出带有相同哈希值的源代码，这一方法就足以解决问题。

第二节 合约执行

执行合约时必须要考虑法律可行性，要考虑很多方面。如运行该合约的独立节点的数量以及链上执行还是链下执行。

与以太坊平台上的智能合约不同，自我意识合约不需要外部事件，也不需要将义务执行的进度保存在区块链上。对自我意识合约而言，将合约方之间的交易记录保存在区块链上就已足够。而且，自我意识合约当前的执行状态随时可以在区块链上调取。

处理房租支付义务（图 12）需要设定好截止日期的逻辑。义务（*l.1-4*）规定，承租方（义务人）必须每月向出租方（获益人）支付租金。存储在智能代理信念库中的时间代币用来触发对循环的 o6 义务的处理过程。在面向智能代理的方法中，对应代码会以数据库或模块的形式提供（图 13 为一段代码摘录）。在 *l.1-9* 中，设计在每个月的第一天将房租支付义务实例化。在 *l.12-17 中*，为获得下个月需要的数据，该设计就会被使用。*l.20-30* 的设计调用了 1-9 行中的一行，目的是在智能代理内部创建事件，使该智能代理可在每月的第一日触发创建具体义务。因此，该事件会再次触发设计 l.1-9。如果该事件发生，类似于图 12 l.6 的一段期限就会被添加进信念库中。

[11] 确切来说，AgentSpeak 是声明式和逻辑式编程语言。Jason 智能代理就是用 AgentSpeak 编程写成的，运行该智能代理意味着源代码必须在 Jason 框架内解读。

图 12 中的这一段逻辑建立在 Jason 智能代理框架构成的声明式和逻辑式编程条件之上。将该智能代理与区块链结合会导致交易时出现一个问题：是在链上运行该合约智能代理还是在链下运行该合约智能代理。

链下：在链下运行自我意识智能合约是务实的做法。Jason 智能代理框架要运行合约，合约就必须保存在它的服务器上。这就要求该智能代理框架提供可以与区块链交流的方式。Jason 框架及其推理循环通过 Java 实现，并考虑到扩展应用到手写代码（如通过数据库与区块链连接）。因此，区块链必须提供更新的应用程序编程借口（API），这也为智能代理和 Java 引入了一些依赖项。至于链下方式，除 API 外，不需要对当前使用的区块链进行任何调整或修改。

```
1 obligation ( o6 , lessee , lessor , date_
2     precondition(year,month,1), pay(rent)
3     )[recurring_deadline(date(year,month
4     ,10)), state(new),recurring].
5
6 time_token_for_recurring_obligation(o6,timeToken(2017, 2, 1)).
```

图 12. 每月支付房租的义务。

链上：与以太坊平台上的智能合约相比，义务合约不需要再将合同状态信息写在区块链上。合约的执行由保存在区块链上的各事件驱动。执行合约需要在机器或硬件上保存合约的执行状态。声明式和逻辑式编程语言（如 AgentSpeak）带来的较高抽象性说明了引入推理循环（如对认知对象的反应）和推理机（如统一期限）的必要性。要在链上运行自我意识合约，必须要在虚拟机上实现这些要素，而这需要开发出新语言。这种方式不需要将 API 引入智能代理。链上运行的另一个优点是压缩和隐藏手写代码，组成一般义务处理过程的微观生命周期（图 13 为一段代码摘录）。

这里有一个关于智能合约中的声明式和命令式编程关系的开放性问题。我们致力于开发的自我意识合约语言基于义务，而且更加静态、更加具有声明性。这就带来了问题：像投票协议或算法这样的逻辑是否也能用这种方式转换？

[12] 这里的“合同”是指包含义务的计算机可读文件。

[13] 解决这个问题的另一个办法是引入一种基于 π 演算的中间语言。

```
1  +time_token_for_recurring_obligation(ObligationName, timeToken(OY,
2  OM, OD))
3      <-
4      + obligation ( ObligationName , Obligor , Beneficiary
5          , date_precondition(OY,OM,OD), Task )[recurring,
6          state(new)];
7      ?obligation(ObligationName, Obligor,
8          Beneficiary, date_precondition(Y,M,D),
9          Task )[recurring]
10
11
12 +?getNextTimeToken(
13     timeToken(TY, TM, TD),
14     date_precondition(year,month,
15     Day), NewDate) : .number(Day)
16     & TM <= 11
17     <-
18     NewDate = date(TY, TM +1, Day).
19
20
21 +!create_event_for_recurring_obligation(ObligationName,
22     date(NY, NM, ND)) : sulfur.date(A,B,C) & a(NY,NM,ND) <=
23     a(A,B,C)
24     <-
25     .concat(
26         "+time_token_for_recurring_obligation
27         (", ObligationName,
28         ",",
29         timeToken(NY, NM, ND
30         ), ")",
```

图 13. 对每月第一天支付房租义务的实例化。

这个问题起始于哪一方运行合约智能代理。一方面，合约细节不被公众知晓；另一方面，区块链原本的目的是分布信息使其不被篡改。有一个解决方案是每个合约方为合约代码的执行设置只读权限，其中只存储了与该合约和合约方相关的交易信息。

第三节 信任事件

自我意识合约执行时不会只涉及类似于支付这样的事件。智能代理不仅要推断义务，还要跟踪细粒度的信息位。如果有某项循环义务（如每月支付一次房租）必须要处理，智能代理必须在每月的第一天创建一项具体的义务，包含相应日期（图 13，第 3 行）。

[14] 循环具体的义务：在租房合同中只有一项义务，即每月支付房租。但是要处理这项义务，我们必须将其分为几部分，以简化智能代理的任务。

智能代理感知其所在环境，检查当前数据，并承认自每月第一天开始，循环义务必须实例化。这样就创造出了合约智能代理需要的其他信息，用来正确处理各种义务。与支付事件、给定签名或价值转移相反，这类信息不需要保存在区块链上，但可以通过重启合约智能代理恢复，如通过第二次实例化来审核合约执行情况。

第四节 上下文可信度

自我意识合约和上下文的集成可以达到不同程度。对于租房合同来说，只有达到相当高的集成度才可以通过读取水电智能仪表上的数字计算出每月的水电费。这在以下方面存在困难：

安全性：物联网设备不断暴露出严重的安全漏洞，如果每月自动交一次公共事业费用可能会给租户带来不便。所以，用来识别和处理错误传感数据的逻辑一定要引入合同智能代理中。这还牵涉到个人隐私，即敏感测量数据的安全传输。

互操作性：整合不同技术领域的技术（如软件智能代理、区块链、智能设备和物联网）要解决一个大难题——数据交换。数据传输与语法互操作性和语义互操作性相关。可靠的数据传输过程还需要可靠的**架构**。如果合同智能代理接收的传感数据来自另一个充当物联网代理并提供统一接口的智能代理处获得，或是通过访问被硬编码进智能代理的智能仪表获得，架构的重要性就会显现。

我们提出的自我意识合约是基于义务的。在本白皮书中，点对点经济是一个高频词，一直贯穿始终。虽然我们提出自我意识合约的本意是要解决点对点运行环境中的义务问题，但商业环境中同样存在义务问题。我们期盼一个稍作改变但核心概念不变的适用于商业环境的智能合约框架。

第六章 可行性评估

第一节 系统结构
第二节 Agrello 语言
第三节 概念证明模型
第四节 讨论

结语

参考文献

1. A.M Antonopoulos. Mastering bitcoins, 2014.
2. Pleszka K. Araszkiewicz, M., editor. *Logic in the Theory and Practice of Lawmaking*. Springer Publishing Company, Incorporated, 1 edition, 2016.
3. R. Baheti and H. Gill. Cyber-physical systems. *The impact of controltechnology*, 12:161–166, 2011.
4. I. Bentov, A. Gabizon, and A. Mizrahi. *Cryptocurrencies Without Proof of Work*, pages 142–157. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.
5. A. Biryukov and D. Khovratovich. Equihash: Asymmetric proof-of-work based on the generalized birthday problem. *Proceedings of NDSS˜aA˘Z´16, 21–24 February 2016, San Diego, CA, USA. ISBN 1-891562-41-X*, 2016.
6. B. Bisping, P.D. Brodmann, T. Jungnickel, C. Rickmann, H. Seidler, A. Sttber,

A. Wilhelm-Weidner, K. Peters, and U. Nestmann. Mechanical verification of a constructive proof for flp. In *International Conference on Interactive Theorem Proving*, pages 107–122. Springer, 2016.
7. R.H. Bordini, J.F. Hu¨bner, and M. Wooldridge. *Programming multi-agent systems in AgentSpeak using Jason*, volume 8. John Wiley & Sons, 2007.
8. O. Bussmann. *The Future of Finance: FinTech, Tech Disruption, and Orchestrating Innovation*, pages 473–486. Springer International Publishing, Cham, 2017.
9. C. Cachin. Architecture of the hyperledger blockchain fabric. In *Workshop on Distributed Cryptocurrencies and Consensus Ledgers*, 2016.

10. C. Dannen. *Solidity Programming*, pages 69–88. Apress, Berkeley, CA, 2017.
11. N. Emmadi and H. Narumanchi. Reinforcing immutability of permissioned blockchains with keyless signatures' infrastructure. In *Proceedings of the 18th In- ternational Conference on Distributed Computing and Networking*, ICDCN '17, pages 46:1–46:6, New York, NY, USA, 2017. ACM.
12. R. Eshuis, A. Norta, O. Kopp, and E. Pitkanen. Service outsourcing with process views. *IEEE Transactions on Services Computing*, 99(PrePrints):1, 2013.
13. R. Eshuis, A. Norta, and R. Roulaux. Evolving process views. *Information and Software Technology*, 80:20 – 35, 2016.
14. Rik Eshuis, Alex Norta, Oliver Kopp, and Esa Pitkanen. Service outsourcing with process views. *IEEE Transactions on Services Computing*, 2014. In press. Preprint at http://is.ieis.tue.nl/staff/heshuis/TSC2014.pdf.
15. B. Glimm, I. Horrocks, B. Motik, G. Stoilos, and Z. Wang. Hermit: An owl 2 reasoner. *Journal of Automated Reasoning*, 53(3):245–269, 2014.
16. P.A. Hamburger. The development of the nineteenth-century consensus theory of contract. *Law and History Review*, 7(2):241–329, 10 2011.
17. R. Hull, V.S. Batra, Y.M. Chen, A. Deutsch, F.F.T. Heath III, and V. Vianu. *Towards a Shared Ledger Business Collaboration Language Based on Data-Aware Processes*, pages 18–36. Springer International Publishing, Cham, 2016.
18. F. Idelberger, G. Governatori, R. Riveret, and G. Sartor. *Evaluation of Logic-Based Smart Contracts for Blockchain Systems*, pages 167–183. Springer International Publishing, Cham, 2016.
19. Kurt Jensen, Lars Michael, Kristensen Lisa Wells, K. Jensen, and L. M. Kristensen. Coloured petri nets and cpn tools for modelling and validation of concurrent sys- tems. In *International Journal on Software Tools for Technology Transfer*, page 2007, 2007.
20. M. K˜olvart, M. Poola, and A. Rull. Smart contracts. In *The Future of Law and eTechnologies*, pages 133–147. Springer, 2016.
21. L. Kutvonen, A. Norta, and S. Ruohomaa. Inter-enterprise business transaction management in open service ecosystems. In *Enterprise Distributed Object Com- puting Conference (EDOC), 2012 IEEE 16th International*, pages 31–40. IEEE, 2012.
22. L. Luu, D.H. Chu, H. Olickel, P. Saxena, and A. Hobor. Making Smart Contracts Smarter. In *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security*, CCS '16, pages 254–269, 2016.
23. B. Marino and A. Juels. *Setting Standards for Altering and Undoing Smart Con- tracts*, pages 151–166. Springer International Publishing, Cham, 2016.
24. D.L. McGuinness, F. Van Harmelen, et al. Owl web ontology language overview. *W3C recommendation*, 10(10):2004, 2004.
25. Business Process Model. Notation (bpmn) version 2.0. *Object Management Group specification*, 2011. http://www.bpmn.org.
26. O. Morten. How firms overcome weak international contract enforcement: repeated interaction, collective punishment and trade finance. *Collective Punishment and Trade Finance (January 22, 2015)*, 2015.
27. M.A. Musen. The prot´eg´e project: A look back and a look forward. *AI matters*, 1(4):4–12, 2015.
28. S. Nakamoto. Bitcoin: A peer-to-peer electronic cash system. *Consulted*, 1(2012):28, 2008.
29. N.C. Narendra, A. Norta, M. Mahunnah, L. Ma, and F.M. Maggi. Sound conflict management and resolution for virtual-enterprise collaborations. *Service Oriented Computing and Applications*, 10(3):233–251, 2016.

30. A. Norta. *Creation of Smart-Contracting Collaborations for Decentralized Autonomous Organizations*, pages 3–17. Springer International Publishing, Cham, 2015.
31. A. Norta. *Establishing Distributed Governance Infrastructures for Enacting Cross-Organization Collaborations*, pages 24–35. Springer Berlin Heidelberg, Berlin, Heidelberg, 2016.
32. A. Norta, P. Grefen, and N.C Narendra. A reference architecture for managing dynamic inter-organizational business processes. *Data & Knowledge Engineering*, 91(0):52 – 89, 2014.
33. A. Norta and L. Kutvonen. A cloud hub for brokering business processes as a service: A "rendezvous" platform that supports semi-automated background checked partner discovery for cross-enterprise collaboration. In *SRII Global Conference (SRII), 2012 Annual*, pages 293–302, July 2012.
34. A. Norta, L. Ma, Y. Duan, A. Rull, M. K˜olvart, and K. Taveter. eContractual choreography-language properties towards cross-organizational business collaboration. *Journal of Internet Services and Applications*, 6(1):1–23, 2015.
35. A. Norta, K. Nyman-Metcalf, A.B. Othman, and A. Rull. âĂIJmy agent will not let me talk to the generalâĂİ: Software agents as a tool against internet scams. In *The Future of Law and eTechnologies*, pages 11–44. Springer, 2016.
36. A. Norta, A. B. Othman, and K. Taveter. Conflict-resolution lifecycles for gov- erned decentralized autonomous organization collaboration. In *Proceedings of the 2015 2Nd International Conference on Electronic Governance and Open Society: Challenges in Eurasia*, EGOSE '15, pages 244–257, New York, NY, USA, 2015. ACM.
37. Aafaf Ouaddah, Anas Abou Elkalam, and Abdellah Ait Ouahman. *Towards a Novel Privacy-Preserving Access Control Model Based on Blockchain Technology in IoT*, pages 523–533. Springer International Publishing, Cham, 2017.
38. R. Ragunathan, I. Lee, L. Sha, and J. Stankovic. Cyber-physical systems: The next computing revolution. In *Proceedings of the 47th Design Automation Conference*, DAC '10, pages 731–736, New York, NY, USA, 2010. ACM.
39. T. Roxenhall and P. Ghauri. Use of the written contract in long-lasting business relationships. *Industrial Marketing Management*, 33(3):261 – 268, 2004.
40. A. Rull, E. T˙aks, and A. Norta. Towards software-agent enhanced privacy protection. In *Regulating eTechnologies in the European Union*, pages 73–94. Springer, 2014.
41. J. Rumbaugh, I. Jacobson, and G. Booch. *Unified Modeling Language Reference Manual, The (2Nd Edition)*. Pearson Higher Education, 2004.
42. L. Sterling and K. Taveter. *The art of agent-oriented modeling*. MIT Press, 2009.
43. M. Swan. *Blockchain Temporality: Smart Contract Time Specifiability with Blocktime*, pages 184–196. Springer International Publishing, Cham, 2016.
44. M. Vukoli´c. The quest for scalable blockchain fabric: Proof-of-work vs. bft repli- cation. In *International Workshop on Open Problems in Network Security*, pages 112–125. Springer, 2015.
45. M. Vukoli´c. *The Quest for Scalable Blockchain Fabric: Proof-of-Work vs. BFT Replication*, pages 112–125. Springer International Publishing, Cham, 2016.
46. I. Weber, X. Xu, R. Riveret, G. Governatori, A. Ponomarev, and J. Mendling. *Untrusted Business Process Monitoring and Execution Using Blockchain*, pages 329–347. Springer International Publishing, Cham, 2016.