



(51) International Patent Classification:		63/033,063	01 June 2020 (01.06.2020)	US
<i>G06F 16/2455</i> (2019.01)	<i>G06F 16/182</i> (2019.01)	63/034,255	03 June 2020 (03.06.2020)	US
<i>G06F 16/33</i> (2019.01)	<i>G06F 16/23</i> (2019.01)	16/893,290	04 June 2020 (04.06.2020)	US
(21) International Application Number:		16/893,295	04 June 2020 (04.06.2020)	US
PCT/US2020/049777		16/893,299	04 June 2020 (04.06.2020)	US
(22) International Filing Date:		16/893,318	04 June 2020 (04.06.2020)	US
08 September 2020 (08.09.2020)		63/052,329	15 July 2020 (15.07.2020)	US
(25) Filing Language:		63/053,217	17 July 2020 (17.07.2020)	US
English		63/055,783	23 July 2020 (23.07.2020)	US
(26) Publication Language:		63/056,984	27 July 2020 (27.07.2020)	US
English		(71) Applicant: DIGITAL ASSET CAPITAL, INC. [US/US];		
(30) Priority Data:		102 Mission Drive, Gaithersburg, Maryland 20878 (US).		
62/897,240	06 September 2019 (06.09.2019)	US	(72) Inventor: HUNTER, Edward; 102 Mission Drive,	
62/959,377	10 January 2020 (10.01.2020)	US	Gaithersburg, Maryland 20878 (US).	
62/959,418	10 January 2020 (10.01.2020)	US		
62/959,481	10 January 2020 (10.01.2020)	US		
63/020,808	06 May 2020 (06.05.2020)	US		

(54) Title: MODIFICATION OF IN-EXECUTION SMART CONTRACT PROGRAMS

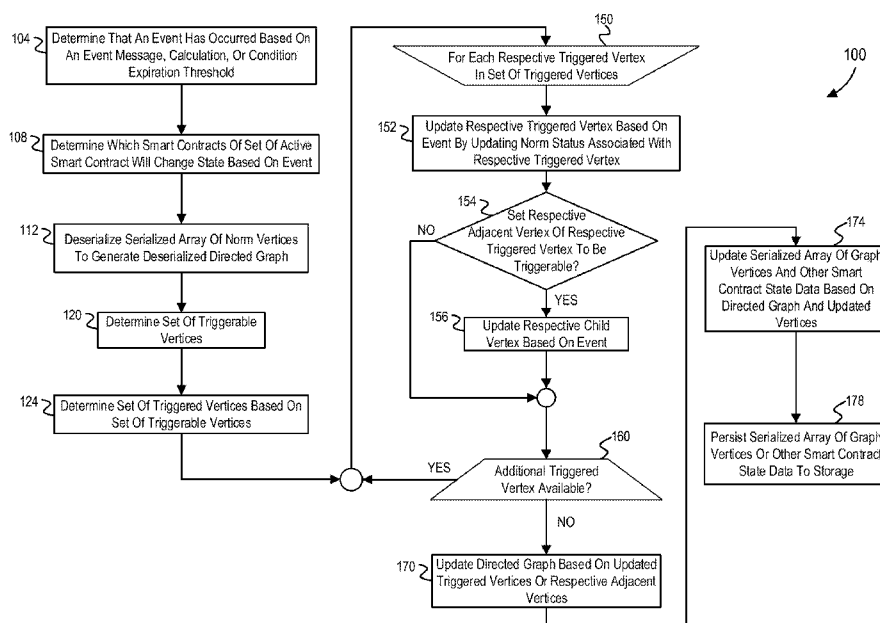


FIG. 1

(57) Abstract: A method includes obtaining program state of a self-executing protocol, wherein the program state includes a set of conditional statements and a directed graph including a set of vertices and a set of directed edges, each respective vertex associated with a respective category label of a set of mutually exclusive categories. The method may include receiving an event message including a set of parameters, selecting a first subset of vertices triggered by the event message and a second subset of vertices based on the first subset of vertices. The method may include determining an aggregated parameter based on a subset of conditional statements, where each respective conditional statement is associated with a respective vertex that is associated with a first category label of the set of mutually exclusive categories. The method may include storing the aggregated parameter in persistent storage.

(74) **Agent:** TUCKER, Joshua; Pillsbury Winthrop Shaw Pittman, LLP, PO Box 10500, McLean, Virginia 22102 (US).

(81) **Designated States** (*unless otherwise indicated, for every kind of national protection available*): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, IT, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW.

(84) **Designated States** (*unless otherwise indicated, for every kind of regional protection available*): ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Published:

— with international search report (Art. 21(3))

MODIFICATION OF IN-EXECUTION SMART CONTRACT PROGRAMS**CROSS-REFERENCE TO RELATED APPLICATIONS****BACKGROUND****1. Field**

[0001] This disclosure relates generally to computer systems and, more particularly, to graph-manipulation based domain-specific execution environments.

2. Background

[0002] Distributed applications operating on a distributed computing platform may be useful in a variety of contexts. Such applications can store program state data on a tamper-evident ledger operating on the distributed computing platform. Use of a tamper-evident ledger or some other data systems distributed over multiple computing devices may increase the security and reliability of distributed applications.

SUMMARY

[0003] The following is a non-exhaustive listing of some aspects of the present techniques. These and other aspects are described in the following disclosure.

[0004] Some include a process that includes obtaining, with a computer system, program state of a smart contract, wherein the program state encodes a directed graph, the directed graph comprising: a set of vertices, and a set of directed edges connecting respective pairs of vertices among the set of vertices, wherein the program state includes a set of conditional statements and a set of entities; obtaining, with the computer system, a request encoding a set of conditional statement parameters corresponding to an amendment to the smart contract; determining, with the computer system, a first subset of vertices in the directed graph, wherein each respective vertex of the first subset causes a state change of the program state in response to a respective conditional statement associated with the respective vertex being satisfied; selecting, with the computer system, a second subset of the first subset based on the set of conditional statement parameters encoded in the request; determining, with the computer system, a set of selected entities based on the second subset; determining, with the computer system, whether a set of criteria associated with the set of selected entities is satisfied; updating,

with the computer system, the set of conditional statement parameters based on the set of conditional statements in response to a determination that the set of criteria associated with the set of selected entities is satisfied; updating, with the computer system, the second subset based on the updated set of conditional statements; and storing, with the computer system, the program state in storage memory after updating the second subset.

[0005] Some aspects include a tangible, non-transitory, machine-readable medium storing instructions that when executed by a data processing apparatus cause the data processing apparatus to perform operations including the above-mentioned process.

[0006] Some aspects include a system, including: one or more processors; and memory storing instructions that when executed by the processors cause the processors to effectuate operations of the above-mentioned process.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] The above-mentioned aspects and other aspects of the present techniques will be better understood when the present application is read in view of the following figures in which like numbers indicate similar or identical elements:

[0008] Figure 1 is a flowchart of an example of a process by which program state data of a program may be deserialized into a directed graph, updated based on an event, and re-serialized, in accordance with some embodiments of the present techniques.

[0009] Figure 2 depicts a data model of program state data, in accordance with some embodiments of the present techniques.

[0010] Figure 3 is flowchart of an example of a process by which a program may simulate outcomes or outcome scores of symbolic AI models, in accordance with some embodiments of the present techniques.

[0011] Figure 4 show a computer system for operating one or more symbolic AI models, in accordance with some embodiments of the present techniques.

[0012] Figure 5 includes a set of directed graphs representing triggered norms and their consequent norms, in accordance with some embodiments of the present techniques.

[0013] Figure 6 includes a set of directed graphs representing possible cancelling relationships and possible permissive relationships between norms, in accordance with some embodiments of the present techniques.

[0014] Figure 7 includes a set of directed graphs representing a set of possible outcome states based on events corresponding to the satisfaction or failure of a set of obligations norms, in accordance with some embodiments of the present techniques.

[0015] Figure 8 includes a set of directed graphs representing a set of possible outcome states after a condition of a second obligations norm of a set of obligations norms is not satisfied, in accordance with some embodiments of the present techniques.

[0016] Figure 9 includes a set of directed graphs representing a set of possible outcome states after a condition of a third obligations norm of a set of obligations norms is not satisfied, in accordance with some embodiments of the present techniques.

[0017] Figure 10 includes a set of directed graphs representing a pair of possible outcome states after a condition of a fourth obligations norm of a set of obligations norms is not satisfied, in accordance with some embodiments of the present techniques.

[0018] Figure 11 is a block diagram illustrating an example of a tamper-evident data store that may be used to render program state tamper-evident and perform the operations in this disclosure, in accordance with some embodiments of the present techniques.

[0019] Figure 12 depicts an example logical and physical architecture of an example of a decentralized computing platform in which a data store of or process of this disclosure may be implemented, in accordance with some embodiments of the present techniques.

[0020] Figure 13 shows an example of a computer system by which the present techniques may be implemented in accordance with some embodiments.

[0021] Figure 14 depicts an example representation of an amendment request modifying a directed graph of a smart contract program, in accordance with some embodiments of the present techniques.

[0022] Figure 15 is a flowchart of a process to modify a program state based on an amendment request, in accordance with some embodiments of the present techniques.

[0023] Figure 16 shows a conceptual diagram representing program states and integrated test conditions determined from program states, in accordance with one or more embodiments.

[0024] Figure 17 shows a flowchart of operations to generate and use integrated test conditions, in accordance with one or more embodiments.

[0025] While the present techniques are susceptible to various modifications and alternative forms, specific embodiments thereof are shown by way of example in the drawings and will herein be described in detail. The drawings may not be to scale. It should be understood, however, that the drawings and detailed description thereto are not intended to limit the present techniques to the particular form disclosed, but to the contrary, the intention is to cover all modifications, equivalents, and alternatives falling within the spirit and scope of the present techniques as defined by the appended claims.

DETAILED DESCRIPTION OF CERTAIN EMBODIMENTS

[0026] To mitigate the problems described herein, the inventors had to both invent solutions and, in some cases just as importantly, recognize problems overlooked (or not yet foreseen) by others in the field of program testing. Indeed, the inventors wish to emphasize the difficulty of recognizing those problems that are nascent and will become much more apparent in the future should trends in industry continue as the inventors expect. Further, because multiple problems are addressed, it should be understood that some embodiments are problem-specific, and not all embodiments address every problem with traditional systems described herein or provide every benefit described herein. That said, improvements that solve various permutations of these problems are described below.

[0027] Technology-based self-executing protocols, such as smart contracts and other programs, allow devices, sensors, and program code have seen increased use in recent years. However, some smart contracts and contract information models often rely on program instructions or industry-specific data structures, which may be difficult to generalize, use for comparison analysis, or re-use in similar contexts due to minor differences in contract details. As a result, uses of smart contracts has not extended into areas that are often the domain of natural language documents. Described herein is a process and related system to construct, interpret, enforce, analyze, and re-use terms for a smart contract in a systematic and unambiguous way across a broad range of applicable fields. In contrast, contracts encoded in natural language text often rely on social, financial, and judicial systems to provide the

resources and mechanisms to construct, interpret, and enforce terms in the contracts. As contract terms increase in number or a situation within which the contract was formed evolves, such a reliance may lead to a lack of enforcement, ambiguity, and wasted resources spent on the re-interpretation or enforcement of contract terms.

[0028] Some embodiments may include smart contracts (or other programs) that include or are otherwise associated with a directed graph representing a state of the smart contract. In some embodiments, vertices of the graph may be associated with (e.g., encode, or otherwise represent) norms (e.g., as norm objects described below) of the smart contract, like formal language statements with a truth condition paired with a conditional statement (sometimes known as a “conditional”) that branches program flow (and changes norm state) responsive to the whether the truth condition is satisfied, for instance, “return a null response if and only if an API request includes a reserved character in a data field.” In some embodiments, norms of a smart contract may represent terms of a contract being represented by the smart contract, legal conditions of the contract, or other verifiable statements. As used herein, a smart contract may be a self-executing protocol executable as a script, an application, or portion of an application on a distributed computing platform, centralized computing system, or single computing device. Furthermore, as used herein, a graph may be referred to as a same graph after the graph is manipulated. For example, if a graph being referred to as a “first graph” is represented by the serialized array `[[1,2], [2,3], [3,4]]` is modified to include the extra vertex and graph edge “[1,5]” and become the modified graph represented by the serialized array `[[1,2], [2,3], [3,4], [1,5]]`, the term “first graph” may be used to refer to the modified graph. Additionally, it should be understood that a data structure need not be labeled in program code as a graph to constitute a graph for the present purposes, as long as that data structure encodes the relationships between values described herein. For example, a graph may be encoded in a key-value store even if source code does not label the key-value store as a graph.

[0029] A self-executing protocol may be a program, like a smart contract. Self-executing protocols may execute responsive to external events, which may include outputs of third-party programs, and human input via a user interface. A self-executing protocol may execute on a computing substrate that involves human intervention to operate, like turning on a computer and launching an event listener.

[0030] A norm of a smart contract may be encoded in various formal languages (like programming languages, such as data structures encoding statements in a domain-specific

programming language) and may include or otherwise be associated with one or more conditional statements, a set of norm conditions of the one or more conditional statements, a set of outcome subroutines of the one or more conditional statements, a norm status, and a set of consequent norms. In some embodiments, satisfying a norm condition may change a norm status and lead to the creation or activation of the consequent norms based on the actions performed by the system when executing the outcome subroutines corresponding to the satisfied norm condition. In some embodiments, a norm may be triggered (i.e. “activated”) when an associated norm condition is satisfied by an event, as further described below. Alternatively, some types of norms may be triggered when a norm condition is not satisfied before a condition expiration threshold is satisfied. As used herein, a triggerable norm (i.e. “active norm”) is a norm having associated norm conditions may be satisfied by an event. In contrast, a norm that is set as not triggerable (i.e. “inactive”) is a norm is not updated even if its corresponding norm conditions are satisfied. As used herein, deactivating a norm may include setting the norm to not be triggerable.

[0031] A smart contract and its norms may incorporate elements of a deontic logic model. A deontic logic model may include a categorization of each of the norms into one of a set of deontic primitive logical categories. A deontic primitive logical category (“logical category”) may include a label such as “right,” “obligation,” or “prohibition.” The logical category may indicate a behavior of the norm when the norm is triggered. In addition, a norm of the smart contract may have an associated norm status such as “true,” “false,” or “unrealized,” where an event may trigger a triggerable norm by satisfying a norm condition (and thus “realizing” the norm). These events may be collected into a knowledge list. The knowledge list may include an associative array of norms, their associated states, an initial norm status during the initial instantiation of the associated smart contract, their norm observation times (e.g., when a norm status was changed, when an event message was received, or the like), or other information associated with the norms. The smart contract may also include a set of consequent actions, where a consequent action may include an association between a triggered norm and any respective consequent norms of the smart contract. As further discussed below, the set of consequent actions may be updated as events occur and the smart contract state is updated, which may result in the formation of a history of previous consequent actions. It should be understood that the term “norm” is used for illustrative purposes and that this term may have different names in other references and contexts. The labeling of norms may also be used for symbolic artificial intelligence (AI) systems. As described further below, the use of these

symbolic AI systems in the context of a smart contract may allow for sophisticated verification and predictive techniques that may be impractical for pure neural network systems which do not use symbolic AI systems. It should be understood that, while the term “logical category” is used in some embodiments, other the terms may be used for categories or types of categories without loss of generality. For example, some embodiments may refer to the use of a “category label” instead of a logical category.

[0032] Some embodiments may store a portion of the smart contract state in a data serialization format (“serialized smart contract state data”). For example, as further described below, some embodiments may store a vertices of a directed graph (or both vertices and edges) in a data serialization format. In response to determining that an event has occurred, some embodiments may deserialize the serialized smart contract state data into a deserialized directed graph. In some embodiments, a vertex (a term used interchangeably with the term node) of the directed graph may be associated with a norm from a set of norms of the smart contract and is described herein as a “norm vertex,” among other terms, where a norm vertex may be connected one or more other norm vertices via graph edges of the directed graph. Some embodiments may then update the directed graph based on a set of consequent norms and their associated consequent norm vertices, where each of the consequent norms are determined based on which norms were triggered by the event and what norm conditions are associated with those active norms. The updated directed graph may then be reserialized to update the smart contract. In some embodiments, a norm vertex may not have any associated conditions. In some embodiments, the amount of memory used to store the serialized smart contract state data may be significantly less than the memory used by deserialized smart contract state data. During or after the operation to update the smart contract, some embodiments may send a message to entities listed in a list of entities (such as an associative array of entities) to inform the entities that the smart contract has been updated, where the smart contract includes or is otherwise associated with the list of entities. Furthermore, it should be understood in this disclosure that a vertex may include (or comprise) a condition by being associated with the condition. For example, a norm vertex may include a first norm condition by including a reference pointer to the first norm condition.

[0033] In some embodiments, generating the smart contract may include using an integrated development environment (IDE) and may include importing libraries of provisions re-used across agreements. Furthermore, some embodiments may generate a smart contract based on

the use of natural language processing (NLP), as further described below. For example, some embodiments may apply NLP operations to convert an existing prose document into a smart contract using operations similar to those described for patent application 63/034,255, titled "Semantic Contract Maps," which is herein incorporated by reference. For example, some embodiments may apply a set of linear combinations of feature observations and cross observations across first order and second orders in feature space to determine a smart contract program or other symbolic AI program. Alternatively, or in addition, some embodiments may include constructing a smart contract from a user interface or text editor without using an existing prose document. In some embodiments, the smart contract may be encoded in various forms, such as source code, bytecode, or machine code encodings. In some embodiments, a smart code may be generated or modified in one type of encoding and be converted to another type of encoding before the smart code is used. For example, a smart contract may be edited in a source code encoding, and the smart contract may be executed by converting the smart contract into a bytecode encoding executing on a distributed computing platform. As used herein, a smart contract may be referred to as a same smart contract between different encodings of the smart contract. For example, a smart contract may be written in source code and then converted to a machine code encoding, may be referred to as a same smart contract.

[0034] Furthermore, as used herein, the sets of items of a smart contract data model may be encoded in various formats. A set of items be encoded in an associative array, a b-tree, a R-tree, a stack, or various other types of data structures. As used herein, the sets of items in the data model may be determined based on their relationships with each other. For example, a set of entities may be encoded as an associative array of entities or may be encoded as an entities b-tree, and elements of a knowledge list may include references to an entity in the set of entities for either type of encoding. In some embodiments, sets of items in their respective data models may be based on the underlying relationships and references between the items in the sets of items, and embodiments should not be construed as limited to specific encoding formats. For example, while some embodiments may refer to an associative array of norms, it should be understood that other embodiments may use a b-tree to represent some or all of the set of norms.

[0035] A smart contract may be stored on different levels of a memory hierarchy. A memory hierarchy of may include (in order of fastest to slowest with respect to memory access speed) processor registers, Level 0 micro operations cache, Level 1 instructions cache, Level 2 shared cache, Level 3 shared cache, Level 4 shared cache, random access memory (RAM), a persistent

flash memory, hard drives, and magnetic tapes. For example, a Level 1 cache of a computing device may be faster than a RAM of the computing device, which in turn may be faster than a persistent flash memory of the computing device. In some embodiments, the memory of a computing device at a first layer of the memory hierarchy may have a lower memory capacity than a memory of the computing device at a slower layer of the memory hierarchy. For example, a Level 0 cache may memory capacity of 6 kibibytes (KiB), whereas a Level 4 cache may have a memory capacity of 128 mebibytes (MiB). In some embodiments, memory may be further distinguished between persistent storage and non-persistent storage (i.e. “non-persistent memory”), where persistent storage is computer memory that may retain the values stored in it without an active power source. For example, persistent storage may include persistent flash memory, hard drives, or magnetic tape, and non-persistent memory may include processor registers, cache memory, or dynamic RAM. In some embodiments, a smart contract may be stored on memory at different levels of the memory hierarchy to increase storage efficiency of the smart contract. For example, serialized smart contract state data of the smart contract may be stored on RAM of a computing device while the deserialized smart contract state data may be stored on a cache of the computing device.

[0036] In some embodiments, the smart contract may update infrequently, such as less than once per hour, less than once day, less than once per month, or the like. The relative infrequency of the updates can mean that the relative computing resources required to deserialize and reserialize data be significantly less than the computing resources required to maintain deserialized data in higher-speed memory. In some embodiments, the dynamic program state By serializing a portion of the smart contract data and persisting the serialized data instead of the corresponding deserialized data to a persistent storage, a computing system may use reduce the memory requirements of storing and executing the smart contract. In addition, the computing system may also increase the number of smart contracts being executed concurrently by a distributed computing platform or single computing device. Furthermore, as used herein, updating a value may include changing the value or generating the value.

[0037] As described herein, some embodiments may store smart contract data in other forms. For example, while some embodiments may temporarily store a directed graph in non-persistent storage, some embodiments may store the directed graph on a persistent storage. In some embodiments, various other types of information such as norm statuses (e.g. “triggered,” “failed,” “satisfied,” etc.) or logical categories (e.g. “rights,” “obligation,” “prohibition,” etc.)

may be included in or otherwise associated with some or all of the vertices of the directed graph. Furthermore, some embodiments may generate visual display representing of the program state data to show the directed graph and its associated statuses, categories, or other information. For example, as further described below, some embodiments may display the directed graph as a hierarchical visual element such as a hierarchy tree in a web application.

[0038] A smart contract may be implemented in various ways. For example, some embodiments may construct, enforce, or terminate the smart contract using a distributed ledger or distributed computing system. Alternatively, some embodiments may implement the smart contract using a request-response system over a public or private internet protocol (IP) network. Use of the methods described herein may increase the efficiency of smart contract enforcement by advancing the state of complex multi-entity agreements in a fast and unambiguous way. Furthermore, implementing and using smart contracts with the embodiments described herein may allow for the comparison, quantification, and reuse of smart contracts in a way that would be inapplicable to custom-coded smart contracts.

[0039] In some embodiments, the smart contract may be stored in a tamper-evident data-store. As discussed below, tamper-evident data stores (e.g., repositories rendering data tamper-evident with one or more tamper-evident data structures) afford desirable properties, including making it relatively easy to detect tampering with entries in the data store and making it relatively difficult or impossible to tailor entries to avoid such detection. Furthermore, various smart contracts may be operating across one or more nodes of the tamper-evident data store, reducing the susceptibility of the smart contract to regional disturbances.

[0040] None of the preceding should be taken to suggest that any technique is disclaimed or that the approaches described herein may not be used in conjunction with other approaches having these or other described disadvantages, for instance, some embodiments may use a custom-written smart-contract that includes one or more of the norms, data structures, or graphs described herein. Or some embodiments may store a directed graph without serialization or deserialization operations. Or some embodiments may be implemented on a centralized server without storing smart contract state data on a distributed computing system such as a decentralized computing system. Further, it should be emphasized that the data structures, concepts, and instructions described herein may bear labels different from those applied here in program code, e.g., a data structure need not be labeled as a “node” or a “graph” in program code to qualify as such, provided that the essential characteristics of such items are embodied.

[0041] In some embodiments, the processes and functionality described herein may be implemented as computer code stored on a tangible, non-transitory, machine-readable medium, such that when instructions of the code are executed by one or more processors, the described functionality may be effectuated. For example, the processes of the figures of this disclosure may be implemented as computer code stored on a non-transitory machine-readable medium. like Instructions may be distributed on multiple physical instances of memory, e.g., in different computing devices, or in a single device or a single physical instance of memory (e.g., non-persistent memory or persistent storage), all consistent with use of the singular term “medium.” In some embodiments, the operations may be executed in a different order from that described, some operations may be executed multiple times per instance of the process’s execution, some operations may be omitted, additional operations may be added, some operations may be executed concurrently and other operations may be executed serially, none of which is to suggest that any other feature described herein is not also amenable to variation.

[0042] Figure 1 is a flowchart of an example of a process by which program state data of a program may be deserialized into a directed graph, updated based on an event, and re-serialized, in accordance with some embodiments of the present techniques. In some embodiments, the process 100, like the other processes and functionality described herein, may be implemented by a system that includes computer code stored on a tangible, non-transitory, machine-readable medium, such that when instructions of the code are executed by one or more processors, the described functionality may be effectuated. Instructions may be distributed on multiple physical instances of memory, e.g., in different computing devices, or in a single device or a single physical instance of memory, all consistent with use of the singular term “medium.” In some embodiments, the operations may be executed in a different order from that described. For example, while the process 100 may be described as performing the operations of block 112 before block 124, the operations of block 124 may be performed before the operations of block 112. As another example, while the process 3600 may be described as performing operations of block 3602 before performing operations of block 3604, the operations of block 3604 may be performed before the operations of block 3602. Some operations may be executed multiple times per instance of the process’s execution, some operations may be omitted, additional operations may be added, some operations may be executed concurrently and other operations may be executed serially, none of which is to suggest that any other feature described herein is not also amenable to variation.

[0043] In some embodiments, the process 100 includes determining that an event has occurred based on an event message, a calculation, or a condition expiration threshold, as indicated by block 104. In some embodiments, the system may determine that an event has occurred after receiving an event message at an API of the system indicating that the event has occurred. As used herein, an event message may be transmitted across or more packets over a wired or wireless connection, where a system may continuously, periodically, or be activated to listen for an event message. In some embodiments, as described further below, an event message may be transmitted over a public or private IP network. Alternatively, or in addition, the event message may be transmitted via the channels of a distributed computing system. For example, the event message may be transmitted from a first node of a distributed computing system (e.g., a blockchain platform) to a second node of the distributed computing system, where the first node and second node may be at different geographic locations (e.g., different nodes executing on different computing devices) or share a same geographic location (e.g., different nodes executing on a same computing device). Furthermore, an event message may be sent by a first smart contract executing on a first computing distributed platform to a second smart contract executing on a same or different distributed computing platform. In some embodiments, determining that an event has occurred does not require verification that the event has occurred. For example, in some embodiments, receiving an event message indicating an event has occurred may be sufficient for the system to determine that the event occurred. Furthermore, in some embodiments, a norm vertex may be triggered based on an event satisfying a subset of its associated norm conditions. Alternatively, a norm vertex may be triggered only after an event satisfies all of its associated norm conditions.

[0044] In some embodiments, the event may include satisfying a condition expiration threshold associated with a triggerable norm vertex (herein “triggerable vertex”) without satisfying a norm condition associated with the triggerable vertex, where a norm condition may be various types of conditions implemented in a computer-readable form to return a value (e.g., “True,” “False,” set of multiple binary values, or the like). For example, a norm condition may include an “if” statement to test whether a payload containing a set of values was delivered to an API of the system by a specific date, where a condition expiration threshold is associated with the norm condition. After the specific date is reached, the system may determine that the condition expiration threshold is satisfied and determine whether the associated norm condition is satisfied. In response to a determination that the norm condition is not satisfied, the system may determine that an event has occurred, where the event indicates that a condition expiration

threshold associated with a triggered norm vertex (herein “triggered vertex”) is satisfied and that an associated norm condition of the triggered vertex is not satisfied. As further stated below, such an event may trigger the associated norm vertex and result in the activation of a set of norms, where the activation of the set of norms may be represented by the generation or association of an adjacent vertex to the triggered vertex, where the adjacent vertex may be updated to be triggerable. As used in this disclosure, it should be understood that satisfying the condition expiration threshold of a triggerable vertex does satisfy a condition associated with the triggerable vertex.

[0045] In some embodiments, the event message may include a publisher identifier to characterize a publisher of the event message. As used herein, a publisher may be an entity and may include various sources of an event message. For example, a publisher may include a publisher in a publisher-subscriber messaging model or a sender of a response or request in a response-request messaging model. In some embodiments, the publisher identifier may be an entity identifier that is a specific name unique to a source of the event message. For example, a publisher identified by the publisher identifier “BLMBRG” may be transmitted in the event message, where “BLMBRG” is unique to a single publisher. Alternatively, or in addition, a publisher identifier may include or be otherwise associated with an identifier corresponding to an entity type that may be assigned to one or more sources of event messages. For example, the publisher identifier may include or otherwise be associated with an entity type such as “TRUSTED-VENDOR,” “ADMIN”, or the like.

[0046] After receiving a publisher identifier, the system may determine whether the publisher identifier is associated with one of a set of authorized publishers with respect to the event indicated by the event message. In some embodiments, the system may refer to a set of authorized publishers corresponding to the event indicated by the event message. For example, the event message may indicate that an event associated with the event message “PAY DELIVERED” has occurred. In response, the system may determine that the event satisfies an condition threshold, where satisfying the condition threshold may include a determination that the event satisfies one or more norm conditions in an associative array of conditions and that the associated publisher is authorized to deliver the message. The associative array of conditions may include a list of norm conditions that, if satisfied, may result in triggering at least one triggerable vertex of the smart contract. For example, the system may determine that the event “PAY DELIVERED” is a direct match with the norm condition “if(PAY

DELIVERED)” of the associative array of conditions. In some embodiments, the system may then refer to the set of authorized publishers associated with the event “PAY DELIVERED.” The system may then determine whether the publisher identifier is in the set of authorized publishers or otherwise associated with the set of authorized publishers, such as by having an entity type representing the set of authorized publishers. In some embodiments, if the system determines that the event message is not authorized, the event message may be rejected as not authorized.

[0047] In some embodiments, the operation to authorize the event may include a operations represented by Statement 1 or Statement 2 below, where “*prop*” may be a string value including an event and “*pub*” may be a string value representing a publisher identifier or entity type. In some embodiments, Statement 1 below may represent an authorization operation that includes the arrival of an event $E[pub]$ from publisher pub . The system may then compare the publisher “ $P[E[pub]]$ ” of the event “ $E[pub]$ ” with each of a set of authorized publishers “ $D[E[prop]][pub]$ ”, where each of the set of authorized publishers is authorized to publish the event “ $E[prop]$ ”. In some embodiments, the set of entities may include or otherwise be associated with the set of authorized publishers. Statement 2 may represent the situation which a plurality of entities may publish a valid event and the systems authorizes a message based on the entity type “ $P[E[pub]][role]$ ” being in the set of authorized publishers “ $D[E[prop]][pub]$,” where the set of authorized publishers “ $D[E[prop]][pub]$ ” may include authorized publisher type:

$$D[E[prop]][pub] == P[E[pub]] \quad (1)$$

.....

$$D[E[prop]][pub] == \quad (2)$$

$$P[E[pub]][role].....$$

[0048] In some embodiments, the set of authorized publishers may include a set of publisher identifiers, and the publisher identifier may in the set of publisher identifiers. For example, if the publisher identifier is “BLMBRG” and the set of authorized publishers include “BLMBRG,” the system may determine that an event message including the publisher identifier “BLMBRG” is authorized. Alternatively, or in addition, the set of authorized publishers may include one or more authorized entity types and a respective publisher may be an authorized publisher if the respective publisher identifier is associated with the authorized

entity type. For example, if the publisher identifier is “BLMBRG,” and if the set of authorized publishers include the entity type “AUTH_PROVIDERS,” and if “BLMBRG” is associated with “AUTH_PROVIDERS” via an associative array, then the system may determine that the publisher identifier is associated with the set of authorized publishers. In response, the system may determine that the event message including the publisher identifier “BLMBRG” is authorized. In some embodiments, the system may determine that one or more events indicated by the event message has occurred only after determining that the event message is authorized.

[0049] In some embodiments, the event message may include a signature value usable by the system to compute a cryptographic hash value. Furthermore, some event messages may include the event payload with the signature value (e.g., via string concatenation) to compute the cryptographic hash value. The system may use various cryptographic hashing algorithms such as SHA-2, Bcrypt, Scrypt, or the like may be used to generate a cryptographic hash value. In some embodiments, the system may use salting operations or peppering operations to increase protection for publisher information. In some embodiments, the system may retrieve a cryptographic certificate based on a publisher identifier as described above and authenticate the event message after determining that on the cryptographic hash value satisfies one or more criteria based on the cryptographic certificate. A cryptographic certificate may include a cryptographic public key used to compare with the cryptographic hash value, as further discussed below. In addition, the cryptographic certificate may also include one or more second cryptographic values indicating a certificate issuer, certificate authority private key, other certificate metadata, or the like.

[0050] In some embodiments, a smart contract may include or be associated with a plurality of cryptographic certificates. The system may determine of which cryptographic certificate to use may be based on a map of entities of the smart contract. In some embodiments, the operation to authenticate the event may include a statement represented by Statement 3 below, where “ v ” may represent a signature verification algorithm, $E[sig]$ may represent a signature value of an event object “ E ,” “ $P[E[pub]]$ ” may represent a data structure that includes the entity that had published the event E , and $P[E[pub]][cert]$ may represent a cryptographic certificate value such as cryptographic public key:

$$v(E[sig], P[E[pub]][cert]) == True \dots\dots\dots (3)$$

[0051] Various signature verification algorithms may be used to authenticate an event message based on a signature value of the event message. For example, the system may determine that the cryptographic hash value is equal to the cryptographic certificate, and, in response, authenticate the event message. In some embodiments, the system may determine that one or more events indicated by the event message has occurred only after authenticating the event message.

[0052] In some embodiments, the system may determine that an event has occurred based on a determination that a condition expiration threshold has been reached. One or more norms represented by norm vertices in the smart contract may include a condition expiration threshold such as an obligation that must be fulfilled by a first date or a right that expires after a second date. For example, a smart contract instance executing on the system may include a set of condition expiration thresholds, where the set of condition expiration thresholds may include specific dates, specific datetimes, durations from a starting point, other measurements of time, other measurements of time intervals, or the like. The system may check the set of condition expiration thresholds to determine if any of the condition expiration thresholds have been satisfied.

[0053] An event message may be transmitted under one of various types of messaging architecture. In some embodiments, the architecture may be based on a representational state transfer (REST) system, where the event message may be a request or response. For example, a system may receive a request that includes the event message, where the request includes a method identifier indicating that the event message is stored in the request. As an example, the system may receive a request that includes a “POST” method indicator, which indicates that data is in the request message. In addition, the request may include a host identifier, where the host identifier indicates a host of the smart contract being executed by the system. For example, the host identifier may indicate a specific computing device, a web address, an IP address, a virtual server executing on a distributed computing platform, a specific node of a decentralized computing system, or the like.

[0054] In some embodiments, the architecture may be based on a publisher-subscriber architecture such as the architecture of the advanced message queuing protocol (AMQP), where the event message may be either a publisher message or subscriber message. For example, using the AMQP, a client publisher application may send an event message over a TCP layer to an AMQP server. The event message may include a routing key, and the AMQP server may

act as a protocol broker that distributes the event message to the system based on the routing key after storing the event message in a queue. In some embodiments, the system may be a subscriber to the client publisher application that sent the event message.

[0055] In some embodiments, the process 100 includes determining which smart contracts of a set of active smart contracts that will change state based on the event, as indicated by block 108. As discussed above, in some embodiments, the system may determine that the event satisfies one or more norm conditions, and, in response, determine that the instance of the smart contract will change state. For example, as further discussed below, the system may determine that the event indicated “PAYLOAD 0105 PROVIDED” satisfies the norm condition represented by the condition “IF DELIVERED(PAYLOAD).” In response, the system may determine that the smart contract will change state. Alternatively, or in addition, as discussed above, the system may determine that the event does not satisfy one or more norm conditions but does satisfy a condition expiration threshold. In response, the system may determine that the instance of the smart contract will change state based on the event not satisfying one or more norm conditions while having satisfied the condition expiration threshold. Furthermore, while this disclosure may recite the specific use of a smart contract program in certain sections, some embodiments may use, modify, or generate other symbolic AI programs in place of a smart contract, where symbolic AI programs are further discussed below.

[0056] In some embodiments, the system may include or otherwise have access to a plurality of smart contracts or smart contract instances. The system may perform a lookup operation to select which of the smart contracts to access in response to determining that an event has occurred. In some operations, the smart contract may compare an event to the associative array of conditions corresponding to each of a set of smart contracts to select of the set of smart contracts should be updated and filter out smart contracts that would not change state based on the event. The system may then update each of the smart contract instances associated with a changed norm status, as discussed further below. Furthermore, the system may then update the respective associative array of conditions corresponding to the set of smart contracts. In some embodiments, an associative array of conditions may include only a subset of norm conditions associated with a smart contract, where each the subset of norm conditions is associated with a triggerable vertex of the smart contract. In some embodiments, the system may first deduplicate the norm conditions before performing a lookup operation to increase performance efficiency. For example, after determining that an event has occurred, some embodiments may

search through a deduplicated array of norm conditions. For each norm condition that the event would trigger, the system may then update the one or more smart contracts associated with the norm condition in the deduplicated array of norm conditions. By selecting smart contracts from a plurality of smart contracts based on an array of norm conditions instead of applying the event to the norm conditions associated with the norm vertices of each of the set of smart contracts, the system may reduce computations required to update a set of smart contracts.

[0057] The smart contract or associated smart contract state data may be stored on various types of computing systems. In some embodiments, the smart contract state data may be stored in a centralized computing system and the associated smart contract may be executed by the centralized computing system. Alternatively, or in addition, the smart contract or associated smart contract state data may be stored on a distributed computing system (like a decentralized computing system) and the associated smart contract may be executed using a decentralized application. For example, the smart contract may be stored on and executed by a Turing-complete decentralized computing system operating on a set of peer nodes, as further described below.

[0058] In some embodiments, the smart contract data may include or be otherwise associated with a set of entities, such as a set of entities encoded as an associative array of entities. The associative array of entities that may include one or more entities that may interact with or view at least a portion of the data associated with the smart contract. In some embodiments, the associative array of entities may include a first associative array, where keys of the first associative array may indicate specific smart contract entities (e.g. data observers, publishers, or the like), and where each of the keys may correspond with a submap containing entity data such as a full legal name, a legal identifier such as a ISIN/CUSIP and an entity type of the entity such as “LENDER,” “BORROWER,” “AGENT,” “REGULATOR,” or the like. In some embodiments, one or more entities of the associative array of entities may include or be associated with a cryptographic certificate such as a cryptographic public key. As described above, the cryptographic certificate may be used to authenticate an event message or other message. By including authorization or authentication operations, the system may reduce the risk that an unauthorized publisher sends an event message or that the event message from a publisher is tampered without the system determining that tampering had occurred. In addition, authorization or authentication operations increase the non-repudiation of event messages,

reducing the risk that a publisher may later disclaim responsibility for transmitting an event message.

[0059] In some embodiments, the smart contract may also include or otherwise be associated a set of conditions, such as a set of conditions encoded as an associative array of conditions. In some embodiments, the associative array of conditions may include a set of norm conditions and associated norm information. In some embodiments, the set of norm conditions may be represented by an associative array, where a respective key of the associative array may be a respective norm condition or norm condition identifier. The corresponding values of the associative array may include a natural language description of the corresponding condition and one or more publisher identifiers allowed to indicate that an event satisfying the respective norm condition has occurred. In some embodiments, the publisher identifier may indicate a specific entity key or an entity type. Furthermore, the smart contract may also include or otherwise be associated with a set of norm vertices or a set of graph edges connecting the vertices, as further described below.

[0060] In some embodiments, the process 100 includes deserializing a serialized array of norm vertices to generate a deserialized directed graph, as indicated by block 112. In some embodiments, the smart contract may include or otherwise be associated with a set of norm vertices encoded as a serialized graph in various data serialization formats, where the smart contract may encode a part or all the norm vertices by encoding the graph edges connecting the norm vertices. The serialized graph may include a representation of an array of subarrays. A data serialization format may include non-hierarchical formats or flat-file formats, and may be stored in a persistent storage. In some embodiments, a serialized array of norm vertices may include numeral values, strings, strings of bytes, or the like. For example, the array of norm vertices (or other data structures in program state) may be stored in a data serialization format such as JSON, XML, YAML, XDR, property list format, HDF, netCDF, or the like. For example, an array may be decomposed into lists or dictionaries in JSON amenable to serialization. Each subarray of an array of subarrays may include a pair of norm vertices representing a directed graph edge. For example, a subarray may include a first value and a second value, where the first value may represent a tail vertex of a directed graph edge, and where the second value may represent a head vertex of the directed graph edge. For example, a subarray may include the value “[1,5]” where the first value “1” represents a tail vertex indicated by the index value “1” and “5” represents a head vertex indicated by the index value

“5.” While in serialized form, the array of norm vertices may reduce memory requirements during data storage operations and bandwidth requirements during data transfer operations.

[0061] In some embodiments, the serialized array of norm vertices may be used to construct an adjacency matrix or an index-free adjacency list to represent a deserialized directed graph during a deserialization operation. In some embodiments, an adjacency matrix or adjacency list may increase efficient graph rendering or computation operations. In some embodiments, the deserialized directed graph may be stored in a faster layer of memory relative to the serialized graph, such as in a non-persistent memory layer. For example, the system may deserialize a serialized array of vertices stored in flash memory to a deserialized directed graph stored in Level 3 cache. In some embodiments, as further described below, instead of forming a directed graph that includes all of the norm vertices included in the serialized array of norm vertices, the system may instead form a directed graph from a subset of the serialized array of norm vertices. As described above, each norm vertex may have an associated norm status indicating whether the norm vertex is triggerable. In response, the system may form a directed graph of the triggerable vertices without rendering or otherwise processing one or more norm vertices not indicated to be triggerable. Using this method, a vertex that is included in the serialized array of vertices may be absent in the directed graph stored in non-persistent memory. By reducing the number of number of vertices in a deserialized directed graph, the efficiency of querying and updating operations of the smart contract may be increased.

[0062] In some embodiments, the system may include an initial set of norm vertices that is distinct from the array of norm vertices. For example, some embodiments may determine that the smart contract had made a first determination that an event had occurred. In some embodiments, the system may search the data associated with the smart contract to find an initial set of norm vertices representing an initial state of the smart contract. The system may then deserialize the initial set of norm vertices when executing the smart contract and perform the operations further described below. The system may then deserialize a different array of norm vertices during subsequent deserialization operations.

[0063] In some embodiments, the process 100 includes determining a set of triggerable vertices based on the directed graph, as indicated by block 120. In some embodiments, the system may determine the set of triggerable vertices based on the directed graph stored in non-persistent memory by searching through the vertices of the directed graph for each of the head vertices of the directed graph and assigning these vertices as a set of head vertices. The system may

then search through the set of head vertices and filter out all head vertices that are also tail vertices of the directed graph, where the remaining vertices may be the set of leaf vertices of the directed graph, where each of the leaf vertices represent a triggerable vertex. Thus, the set of leaf vertices determined may be used as the set of triggerable vertices.

[0064] Alternatively, in some embodiments, a vertex of the set of norm vertices may include or otherwise be associated with a norm status indicating whether the vertex is triggerable or not. In some embodiments, the system may search through the directed graph for vertices that have an associated norm status indicating that the respective vertex is triggerable. Alternatively, or in addition, the system may search through a list of norm statuses associated with the vertices of the serialized array of norm vertices to determine which of the vertices is triggerable and determine the set of triggerable vertices. For example, in some embodiments, each norm vertex of a smart contract may have an associated norm status indicating whether the vertex is triggerable or not triggerable, where the vertices and their associated statuses may be collected into a map of vertex trigger states. The system may then perform operations to traverse the map of vertex trigger states and determine the set of triggerable vertices by collecting the vertices associated with a norm status indicating that the vertex is triggerable (e.g. with a boolean value, a numeric value, a string, or the like). For example, the system may perform operations represented by Statement 4 below, where G may represent a graph and may be an array of subarrays g , where each subarray g may represent a norm vertex and may include a set of values that include the value assigned to the subarray element $g[4]$, where the subarray element $g[4]$ indicates a norm status, and “Active” indicates that the norm vertex associated with subarray g is triggerable, and A is the set of triggerable vertices:

$$A \leftarrow \{g \in G \mid g[4] = \text{"Active"}\} \dots\dots\dots (4)$$

[0065] In some embodiments, the process 100 includes determining a set of triggered vertices based on the set of triggerable vertices, as indicated by block 124. In some embodiments, the system may compare determine the set of triggered vertices based on which the norm conditions associated with the vertices of the directed graph are satisfied by the event. In some embodiments, a norm condition may directly include satisfying event. For example, a norm condition may include “IF DELIVERED(PAYMENT),” where the function “DELIVERED” returns a boolean value indicating whether a payment represented by the variable “PAYMENT” is true or false. The system may then determine that the norm condition is

satisfied if “DELIVERED(PAYMENT)” returns the boolean value “True.” The system may then add the vertex associated with the norm condition to the set of triggered vertices. For example, the system may perform operations represented by Statement 5 below, where “ A ” is the set of triggerable vertices determined above, and where each subarray “ a ” may represent a triggerable vertex and may include a set of values that include the value assigned to the subarray element $a[1]$, where the subarray element $a[1]$ indicates a condition, and “ U ” is the set of triggered vertices, and “ N ” is an associative array that describes the possible graph nodes that may be triggered, such that, for an event $prop$, $N[prop]$ may return a structure that contains defining details of the vertices associated with the event $prop$:

$$U \leftarrow \{a \in A \mid N[a[1]][prop] = E[prop]\} \dots\dots\dots (5)$$

[0066] In some embodiments, the determination that an event satisfies a norm condition may be based on a categorization of a norm into logical categories. As further described below in Figure 5, logical categories may include values such as a “right,” “obligation,” “prohibition,” “permission,” or the like. In some embodiments, after a determination that an event triggers a norm condition, the generation of consequent norms or norm status changes associated with a triggered vertex may be based on the logical category.

[0067] In some embodiments, a snapshot contract status may be associated with the smart contract and may be used to indicate a general state of the smart contract. The snapshot contract status may indicate whether the obligations of a contract are being fulfilled or if any prohibitions of the contract are being violated. For example, in some embodiments, satisfying an obligation norm condition may result in an increase in the snapshot contract status and triggering a prohibitions norm may result in a negative change to the snapshot contract status.

[0068] In some embodiments, the process 100 includes performing one or more operations indicated by blocks 152, 154, 156, and 160 for each of the respective triggered vertex of the set of triggered vertices, as indicated by block 150. In some embodiments, the process 100 includes updating the respective triggered vertex based on an event by updating a norm status associated with the respective triggered vertex, as indicated by block 152. Updating a respective triggered vertex may include updating one or more norm statuses or other status values associated with the respective triggered vertex. For example, a norm status of the respective triggered vertex may be updated to include one of the strings “SATISFIED,”

“EXERCISED,” “FAILED,” or “CANCELED,” based on the norm conditions associated with the respective triggered vertex having been satisfied, exercised, failed, or canceled, respectively. In some embodiments, the system may update a norm status to indicate that the respective triggered vertex is not triggerable. For example, an obligation norm of a smart contract may be required to be satisfied only once. In response, after determining that the norm condition associated with the obligation has been satisfied by an event, the system may update a first status value associated with the respective triggered vertex to “false,” where the first status value indicates whether the respective triggered vertex is triggerable. In some embodiments, the one or more status values may include a valence value indicating the number of connections from the respective triggered vertex to another vertices, the number of connections to the respective triggered vertex from other vertices, or the like. As further described below, in some embodiments, the valence value or other status value associated with the respective triggered vertex may be updated after performing operations associated with the adjacent vertices of the respective triggered vertex.

[0069] In some embodiments, the process 100 includes determining whether a respective adjacent vertex of the respective triggered vertex should be set to be triggerable, as indicated by block 154. In some embodiments, the respective triggered vertex may include a pointer to or otherwise be associated with a set of adjacent vertices, where each of the set of adjacent vertices represent a norm of the smart contract that are set to occur after the respective triggered vertex is triggered. In some embodiments, the system may determine whether an adjacent vertex of a respective triggered vertex should be set as triggerable based on specific conditions associated with the adjacent vertex. For example, a respective triggered vertex may include program code instructing that a first set of adjacent vertices should be set to be triggerable if a first set of conditions are satisfied and that a second set of adjacent vertices should be set to be triggerable if a second set of conditions are satisfied, where the first set of adjacent vertices are distinct from the second set of adjacent vertices. Alternatively, or in addition, the respective triggered vertex may include program instructing that a third set of adjacent vertices should be set to be triggerable if the first set of conditions are not satisfied but an associated condition expiration threshold is satisfied.

[0070] In some embodiments, the process 100 includes updating the respective adjacent vertex based on the event, as indicated by block 156. Updating the respective adjacent vertex based on the event may include setting one or more norm statuses associated with the adjacent vertex

to indicate that the respective adjacent vertex is triggerable. For example, after a determination that a respective adjacent vertex associated with a permission norm is to be set to be triggerable, a norm status associated with the respective adjacent vertex may be updated to the value “triggerable.”

[0071] In some embodiments, the process 100 includes determining whether any additional triggered vertices are available, as indicated by block 160. In some embodiments, the system may determine that additional triggered vertices are available based on a determination that an iterative loop used to cycle through each the triggered vertices has not reached a termination condition. In response to a determination that additional triggered vertices are available, the process 100 may return to the operations of block 150. Otherwise, operations of the process 100 may proceed to block 164.

[0072] In some embodiments, the process 100 includes updating the directed graph based on the updated triggered vertices or the respective adjacent vertices, as indicated by block 170. In some embodiments, updating the directed graph may include updating an adjacency matrix or adjacency list representing the directed based on each of the triggered vertices or their respective adjacent vertices. In some embodiments, instead of looping through each updated vertex and then updating the directed graph, the system may update the directed graph during or after each update cycle. For example, after updating the respective triggered vertex as described in block 156, the system may update the deserialized directed graph.

[0073] In some embodiments, the process 100 includes updating the serialized array of norm vertices or other smart contract state data based on the directed graph and updated vertices, as indicated by block 174. In some embodiments, updating the serialized array of norm vertices may include serializing the directed graph into a data serialization format, as described above. In some embodiments, the data serialization format may be the same as the data serialization format used when performing operations described for block 112. For example, the system may implement a depth-first search (DFS) over the deserialized directed graph to record distinct edge pairs and update the serialized array of norm vertices by either modifying or replacing the serialized array of norm vertices.

[0074] In some embodiments, the system may update a knowledge set based on the event and smart contract state changes that occurred in response to the event. In some embodiments, the knowledge set may include a set of previous events. The set of previous events may be encoded

as a list of previous events. The list of previous events may include a subarray, where each subarray includes an event identifier of a recorded event or information associated with the recorded event. For example, the list of previous events may include a date and time during which an event occurred, an event identifier, one or more norm conditions satisfied by the event, or the like. In some embodiments, a norm condition may be based on the list of previous events. For example, a norm condition may include a determination of whether an event type had occurred twice within a time duration based on the list of previous events. In some embodiments, the knowledge set may include a set of previously-triggered vertices, where the set of previously-triggered vertices may be encoded as an array of previously-triggered vertices. In some embodiments, the system may further update the knowledge set by updating the array of previously-triggered vertices based on the triggered vertices described above. For example, after updating a respective triggered vertex as described above, the system may update the array of previously-triggered vertices to include the respective triggered vertex. The array of previously-triggered vertices may include a vertex identifier associated with the respective triggered vertex, an event identifier associated with the event that triggered the respective triggered vertex, and a set of values identifying the vertices that are set to be triggerable after triggering the respective triggered vertex.

[0075] In some embodiments, the process 100 includes persisting the updated serialized array of norm vertices or other smart contract data to storage, as indicated by block 178. In some embodiments, persisting the smart contract data to storage may include updating the memory storage in a single computing device or a computing device of a centralized computing system. Alternatively, or in addition, persisting the smart contract data to storage may include storing the smart contract data to a decentralized tamper-evident data store. In some embodiments, by storing the serialized array of norm vertices in a decentralized tamper-evident data store instead of storing a deserialized directed graph in the decentralized tamper-evident data store, the system may increase the efficiency and performance of the data distribution amongst the nodes of the decentralized tamper-evident data store. Furthermore, in some embodiments, triggering a norm vertex may include triggering a smart contract termination action. When a smart contract termination action is triggered, vertices other than the respective triggered vertex may be updated to set the statuses of each vertex of these other vertices as not triggerable, even if these other vertices are not directly connected to the triggered vertex.

[0076] In some embodiments, the system may display a visualization of the smart contract state. For example, the system may display a visualization of smart contract state as a directed graph, such as (though not limited to) those shown in Figures 5-10 below, where the vertices may have different colors based on norm status and/or logical category. Alternatively, or in addition, the system may generate other types of visualizations of the smart contract state. For example, the system may display a pie chart representing of a plurality of smart contract types that indicate which type of the smart contracts have the highest amount of associated cost.

[0077] In some embodiments, the process 100 or other processes described in this disclosure may execute on a decentralized computing platform capable of persisting state to a decentralized tamper-evident data store. Furthermore, in some embodiments, the decentralized computing platform may be capable of executing various programs, such as smart contracts, on the computing platform in a decentralized, verifiable manner. For example, each of a set of peer nodes of the computing platform may perform the same computations, and a consensus may be reached regarding results of the computation. In some embodiments, various consensus algorithms (e.g., Raft, Paxos, Helix, Hotstuff, Practical Byzantine Fault Tolerance, Honey Badger Byzantine Fault Tolerance, or the like) may be implemented to determine states or computation results of the various programs executed on the decentralized computing platform without requiring that any one computing device be a trusted device (e.g., require an assumption that the computing device's computation results are correct). The one or more consensus algorithms used may be selected or altered to impede an entity from modifying, corrupting, or otherwise altering results of the computation by peer nodes not under the entity's control. Examples of a decentralized tamper-evident data store may include Interplanetary File System, Blockstack, Swarm, or the like. Examples of a decentralized computing platform may include Hyperledger (e.g., Sawtooth, Fabric, or Iroha, or the like), Stellar, Ethereum, EOS, Bitcoin, Corda, Libra, NEO, or Openchain.

[0078] Figure 2 depicts a data model of program state data, in accordance with some embodiments of the present techniques. In some embodiments, a smart contract may include or otherwise be associated with program state data such as smart contract state data 200. The smart contract state data 200 includes an associative array of entities 210, an associative array of conditions 220, an associative array of norms 230, a graph list 240, and a knowledge list 250. The associative array of entities 210 may include a set of keys, each key representing an entity capable of interacting with or observing smart contract data. For example, a publisher

providing an event message to the smart contract may be an entity. The corresponding value of a key of the associative array of entities 210 may include a submap that includes values for a name, a legal identifier value (e.g., a ISIN/CUSIP identifier), an entity type for authorization operations, and a public key for authentication operations (e.g., a cryptographic public key). In some embodiments, the name, identifier value, entity type, or public keys may be used in the authorization and authentication operations discussed for block 104.

[0079] The associative array of conditions 220 may include a set of keys, where each key represents an event that may trigger at least one triggerable vertex that would result in a change in norm status, and where a corresponding value of each key includes an events submap. The events submap may include a publisher identifier. As shown by the link 221, the publisher identifier may be used as a reference to the key of the associative array of entities. Alternatively, or in addition, the events submap may include a subject identifier, which may include natural text language to provide a context for the corresponding event.

[0080] The associative array of norms 230 may include a set of keys, where each key may represent a norm of the smart contract, which may be associated with as a norm vertex in a graph, norm conditions and consequent norms. In some embodiments, the consequent norms may themselves be associated with their own norm vertices. Each value corresponding to the norm may include a norms submap that includes one or more norm conditions that may be used to trigger the norm by satisfying a norm condition, or by not satisfying the norm condition after satisfying condition expiration threshold associated with the norm. As shown by the link 231, the norm conditions may include a norm identifier that may be used as a reference to a key of the associative array of conditions 220. The norms submap may also include an entity identifier, where the entity identifier may be used as reference to a key of the associative array of entities 210, as shown by the link 232. The norm may also include a condition expiration threshold, which may be represented by the “expiry” field shown in the associative array of norms 230. As discussed above, some embodiments may result in a norm status change or trigger other updates to a vertex if a norm condition is not satisfied but the condition expiration threshold is satisfied. The norm submap may also include a consequences list, where the consequences list may include set of sublists that includes a tail vertex representing a consequent norm that become triggerable, a head vertex of the new norm (which may be the triggered norm), and a label.

[0081] In some embodiments, a smart contract state may initially construct the graph list 240 in a first iteration based on the associative array of norms 230 and update the graph list 240 based on a previous iteration of the graph list 240. As described above, the graph list may be in a serialized form, such as a serialized array of norm vertices written in the YAML markup language. As discussed above, the graph list 240 may be a list of graph sublists, where each sublist includes a tail vertex value, a head vertex value, a label associated with the graph edge connecting the tail vertex with the head vertex, a group identifier, and a norm status value. In some embodiments, the norm status may include values such as “satisfied,” “exercised,” “failed,” “active,” “terminal,” “canceled,” “triggerable,” or “untriggerable.” In some embodiments, a norm vertex may be associated with more than one norm status. As shown by link 241, a tail vertex of the graph may be linked to a norm in the associative array of norms 230. Similarly, as shown by the links 242-243, the tail and head vertices of the graph list 240 may be associated with a listed tail norm or head norm in the associative array of norms 230 for a respective norm. Furthermore, as shown by the link 244, the group identifier listed in a graph sublist may also be associated with a value in the associative array of norms 230, such as with a key in the associative array of norms 230.

[0082] In some embodiments, a smart contract state may initially construct the knowledge list 250 in a first iteration based on the associative array of norms 230 and update the knowledge list 250 based on smart contract state changes. The knowledge list 250 may be sequentially ordered in time (e.g. a time when a norm status changes, a time when an event is received, or the like). In some embodiments, each entry of the knowledge list 250 may include an identifier “eid,” an event time “etime,” a publisher identifier associated with an event that triggered a norm vertex, the event that triggered the norm vertex. In addition, the knowledge list 250 may include various other data related to the smart contract state change, such as a field “negation” to indicate whether an event is negated, a field “ptime” in ISO8601 format to represent a sub-event time (e.g. for event that require multiple sub-events to trigger a norm vertex), a field “signature” to provide a signature value that allows authentication against the public key held by a publisher for later data authentication operations or data forensics operations. In some embodiments, the knowledge list 250 may include an evidence list, where the evidence list may include a base64 encoded blob, an evidence type containing a string describing the file type of the decoded evidence, and a field for descriptive purposes. In some embodiments, the evidence list may be used for additional safety or verification during transactions.

[0083] As described above, some embodiments may efficiently store or update program state data using a set of serialization or deserialization operations. Some embodiments may assign outcome scores to possible outcomes of an update operation, which may then be used to predict future states of a program. Some embodiments may perform operations, such as those described further below, to predict an outcome score using data encoded in a directed graph with greater efficiency or accuracy.

[0084] GRAPH OUTCOME DETERMINATION IN DOMAIN-SPECIFIC EXECUTION ENVIRONMENT

[0085] In some embodiments, outcomes of symbolic AI models (like the technology-based self-executing protocols discussed in this disclosure, expert systems, and others) may be simulated and characterized in various ways that are useful for understanding complex systems. Examples of symbolic AI systems include systems that may determine a set of outputs from a set of inputs using one or more lookup tables, graphs (e.g. a decision tree), logical systems, or other interpretable AI systems (which may include non-interpretable sub-components or be pipelined with non-interpretable models). The data models, norms, or other elements described in this disclosure constitute an example of a symbolic AI model. Some embodiments may use a symbolic AI model (like a set of smart contracts) in order to predict possible outcomes of the model and determine associated probability distributions for the set of possible outcomes (or various population statistics). Features of a symbolic AI model that incorporates elements of data model described in this disclosure may increase the efficiency of smart contract searches. In addition, the use of logical categories (e.g., “right,” “permission,” “obligation”) describing the relationships between conditional statements (or other logical units) of a smart contract may allow the accurate prediction of (or sampling of) outcomes across a population of differently-structured smart contracts without requiring a time-consuming analysis of each of the contexts of individual smart contracts from the population of differently-structured smart contracts. Furthermore, the operations of a symbolic AI model may be used to predict outcomes (e.g., of a smart contract, or call graph of such smart contracts) and may be tracked to logical units (like conditional statements, such as rules of a smart contract). These predicted outcomes may be explainable to an external observer in the context of the terms of the logical units of symbolic AI models, which may be useful in medical fields, legal fields, robotics, dev ops, financial fields, or other fields of industry or research.

[0086] In some embodiments, the symbolic AI model may include the use of scores for a single smart contract or a plurality of smart contracts, where the score may represent various values, like a range of movement along a degree of freedom of an industrial robot, an amount of computer memory to be allocated, an amount of processing time that a first entity owes a second entity, an amount to be changed between two entities, a total amount stored by an entity, or the like. A symbolic AI model may include scores of different type. Changes in scores of different type may occur concurrently when modeling an interaction between different entities. For example, a first score type may represent an amount of computer memory to be stored within a first duration and a second score type may represent an amount of computer memory to be stored within a second duration that occurs after the first duration. A smart contract may be used to allocate computer memory across two different entities to optimize memory use across the entity domains. Possible outcomes and with respect to memory allocation across the two domains may be simulated. Alternatively, or in addition, exchanges in other computing resources of the same type or different types may be simulated with scores in a symbolic AI model. For example, a symbolic AI model may include a first score and as second score, where the first score may represent an amount of bandwidth available for communication between a first entity or second entity and a third entity, and where the second score may represent an amount of memory available for use by the first or second entity. The outcome of an exchange negotiated via a smart contract between the first and second entity for bandwidth and memory allocation may then be simulated to predict wireless computing resource distribution during operations of a distributed data structure across a wireless network or other computing operations.

[0087] In some embodiments, simulating outcomes of may include processing one or more norm vertices representing one or more norms of a smart contract as described in this disclosure. For example, the symbolic AI model may include an object representing a norm vertex, where the object includes a first score representing an amount owed to a first entity and a second score representing an amount that would be automatically transferred to the first entity (e.g., as a down payment). In some embodiments, the symbolic AI model may incorporate the entirety of a smart contract and its associated data model when performing simulations based on the smart contract. For example, a symbolic AI model may include one or more directed graphs of to represent the state of a data model. Alternatively, or in addition, some embodiments may include more data than the smart contract being simulated or less data than the smart contract be simulated.

[0088] In some embodiments, the symbolic AI system (a term used interchangeably with symbolic AI model) may process the conditional statements (or other logical units) associated with each of the norms of a smart contract to increase simulation efficiency by extracting only quantitative changes and making simplifying assumptions about score changes. For example, a system may collect the norm conditions and associated outcome subroutines associated with each of a set of norm vertices and extract only the changes in an amount of currency owed as a first score and changes in an amount of currency transferred as a second score when incorporating this information into the conditions of the symbolic AI model. In some embodiments, the information reduction may increase computation efficiency by removing information from the analysis of a smart contract determined to be not pertinent to a selected score. Some embodiments simulate outcomes across a plurality of smart contracts using a standardized search and simulation heuristic, and the system described herein may provide a population of scores, where the population of scores may be the plurality of outcome scores determined from a simulation of each of the smart contracts or values computed from the plurality of outcome scores. For example, values determined based on the population of scores may include parameters of a probability distribution of the scores, a total score value, a measure of central tendency (e.g. median score value, mean score value, etc.), or the like.

[0089] In some embodiments, the symbolic AI model may be an un-instantiated smart contract or may be a transformation thereof, e.g., approximating the smart contract. For example, as further described below, the system may instantiate a program instance that includes a symbolic AI model based on a selected smart contract that is not yet instantiated. Alternatively, a symbolic AI model may be determined based on an instantiated smart contract. For example, the system may select an instantiated smart contract with a program state that has already changed from its initial program state in order to determine future possible outcomes in the context of the existing changes. The system may then copy or otherwise use a simulated version of the changed program state when simulating the instantiated smart contract. For example, the system may select an instantiated smart contract for simulation with a symbolic AI system and deserialize a directed graph of the instantiated smart contract. The symbolic AI system may copy the deserialized directed graph to generate a simulation of the directed graph, where the nodes of the simulated directed graph are associated with simplified conditional statements that convert quantifiable changes into scores and are stripped of non-quantifiable changes in comparison to the conditional statements of the smart contract.

[0090] Figure 3 is flowchart of an example of a process by which a program may simulate outcomes or outcome scores of symbolic AI models, in accordance with some embodiments of the present techniques. In some embodiments, a process 300 includes selecting a set of smart contracts (or other symbolic AI models) based on a search parameter, as indicated by block 304. In some embodiments, a system may include or otherwise have access to a plurality of smart contracts or smart contract instances, and the system may select a set of smart contracts from the plurality based on a specific search parameter, such as an entity, entity type, event, event type, or keyword. For example, the system may perform a lookup operation to select which of the smart contracts to access based on an event. During the lookup operation, the system may compare an event to the associative arrays of conditions corresponding to each of a plurality of smart contracts and select a set of smart contracts based on which of the smart contracts would change state in response to receiving the event. Some embodiments may crawl a call graph (of calls between smart contracts, or other symbolic AI models) to select additional smart contracts.

[0091] In addition, or alternatively, the system may perform a lookup operation to select which of the smart contracts to access based on an entity or entity type. For example, the system may compare an entity to the associative arrays of entities corresponding to each of a plurality of smart contracts and select a set of smart contracts based on which of the corresponding arrays of entities include the entity. An entity identifier may be in an array of entities or some other set of entities if an entity type associated with the entity identifier is in the array of entities. For example, if the entity “BLMBRG” has an associated entity type of “trusted publisher,” some embodiments may determine that “BLMBRG” is in the set of entities of a smart contract if the entity type “trusted publisher” is listed in the set of entities. Alternatively, some embodiments may require that the exact entity identifier be listed in a set of entities before determining that the entity identifier is in the set of entities. For example, some embodiments may determine that “BLMBRG” is in a set of entities of a smart contract only if “BLMBRG” is one of the elements of the set of entities. Furthermore, in some embodiments, the search may include intermediary entities between two different entities, where intermediary smart contract may be a smart contract (other than the first or second smart contract) that has relationships with both the first and second entities. For example, a search for smart contracts relating a first entity and a second entity may return a set smart contracts that include a first smart contract and a second smart contract, where the array of entities of the first smart contract includes the first entity and an

intermediary entity, and where the array of entities of the second smart contract includes the second entity and the intermediary entity.

[0092] In some embodiments, an intermediary entity for a first entity and a second entity may be found by determining the intersection of entities between a first set of smart contracts associated with the first entity and a second set of smart contracts associated with the second entity. For example, the system may select a first set of smart contracts from a plurality of smart contracts based on which sets of entities associated with plurality of smart contracts include the first entity. Similarly, the system may select a second set of smart contracts from a plurality of smart contracts based on which sets of entities associated with plurality of smart contracts include the second entity. The system may then determine the intersection of entities by searching through the sets of entities of the first and second set of smart contracts to collect the entities that appear in both the first set and second set and determine that these collected entities are intermediary entities. In some embodiments, as further described below, additional methods are possible to determine a set of smart contracts associating a first entity with a second entity in order to quantify a relationship between the first entity and the second entity.

[0093] As discussed in this disclosure, some embodiments may crawl a call graph to select additional smart contracts based on possible relationships between a first entity and a second entity. The call graph may be a privity graph, which may track privity relations between the first entity and entities other than the second entity in order to determine or quantify relations between the first entity and the second entity. if For example, some embodiments may crawl through a privity graph of possible score changes across multiple contracts and determine a quantitative score relationship between a first entity and a second entity based on a first transaction between the first entity and a third entity, a second transaction between the third entity and a fourth entity, a third transaction between the fourth entity and a fifth entity, and a fourth transaction between the fifth entity and the second entity.

[0094] In some embodiments, the process 300 includes performing one or more operations indicated by blocks 312, 316, 320, 324, 328, 336, 340, 344, and 350 for each of the respective smart contracts or other programs of the selected set of smart contracts or other programs, as indicated by block 308. As further discussed below, the one or more outputs from executing each of the smart contracts may be used to determine a population of scores of multiple smart contracts. As used herein, the population of scores of multiple smart contracts may represent one or more population metric values calculated from scores of the smart contract. For

example, the population of scores of multiple smart contracts may include a measure of central tendency, a measure of dispersion, a kurtosis value, a parameter of a statistical distribution, one or more values of histogram, or the like. Furthermore, in some embodiments, the process 300 may include performing one or more operations in parallel using multiple processor cores, where performing multiple operations in parallel may include performing the multiple operations concurrently. For example, some embodiments may perform the operations of the blocks 312, 316, 320, 324, 328, 336, 340, 344, and 350 for a plurality of smart contracts in parallel by using one or more processors for each of the plurality of smart contracts. By performing operations in parallel, computation times may be significantly reduced.

[0095] In some embodiments, the process 300 includes acquiring a set of conditional statements (or other logical units), set of entities, set of indices indexing the conditional statements, or other data associated with the selected smart contract, as indicated by block 312. Each of the set of conditional statements may be associated with an index value and may include or be otherwise associated with a respective set of conditions and a respective set of outcome subroutines, where a computing device may execute the respective set of outcome subroutines in response to an event satisfying the respective set of conditions. In some embodiments, the set of conditional statements may form a network, like a tree structure, with respect to each other. For example, an outcome subroutine of one the conditional statements may include a reference to or otherwise use an index value associated with another conditional statement. In some embodiments, the set of conditional statements and set of indices may be acquired from a data model, where the index values may be or otherwise correspond to the identifiers for norm vertices of a directed graph. For example, the set of conditional statements and set of indices may be acquired from the associative array of norms 230, the associative array of conditions 220, and the graph list 240. Alternatively, the system may acquire the conditional statements and indices from data stored using other data models. For example, the system may acquire the conditional statements from an indexed array of objects, where each object may include a method that can take an event as a parameter, test the event based on a condition of the method, and return a set of values or include a reference to another object of the array. The system may use the indices of the indexed array as the indices of the conditional statements and parse the methods to provide the set of conditional statements.

[0096] In some embodiments, the process 300 includes instantiating or otherwise executing a program instance having program state data that includes a symbolic AI model that includes

values from the data associated with the selected smart contract, as indicated by block 316. In some embodiments, the symbolic AI model may include graph vertices associated with the set of conditional statements described in this disclosure and may also include directed graph edges connecting the graph vertices. In addition, or alternatively, the symbolic AI model may include a set of tables, decision trees, graphs, or logical systems to provide a predicted value as an output based on one or more inputs corresponding to real or simulated events. For example, the system may traverse the directed graph of a symbolic AI model to determine which nodes of the directed graph to visit based on a decision tree of the symbolic AI model. Furthermore, in some embodiments, the symbolic AI system may be re-instantiated or be modified in real-time in response to a particular event message updating a smart contract being simulated. For example, an instantiated smart contract may be executing and concurrently being simulated by a symbolic AI system. In response to the smart contract receiving an event message, the symbolic AI system may determine a new set of events based on the event message and update its own program state such that its new initial state is based on the smart contract program state after the smart contract program state has been updated by the events of the event message.

[0097] In some embodiments, the symbolic AI model may include a graph. In some embodiments, the system may generate a graph list such as the graph list 240 using the methods discussed in this disclosure. In some embodiments, the program instance may be a local version of a selected smart contract and have program state data identical to program state data in the selected smart contract. Alternatively, the program instance may include program data not included in the smart contract or exclude data included in the smart contract. In some embodiments, the graph of the symbolic AI model may include a set of graph vertices and a set of directed graph edges connecting the graph vertices, where each of the graph vertices may be identified by an identifier and corresponds to a conditional statement of a smart contract. In some embodiments, the identifier may be the set of index values associated with the conditional statements of the smart contract. Alternatively, the identifier may be different from the set of index values associated with the conditional statements of the smart contract. For example, the system may choose a set of identifiers that are different from the set of index values to increase system efficiency or reduce memory use.

[0098] In some embodiments, the directed graph edges may be structured to provide directional information about the graph vertices of a symbolic AI model. For example, a directed graph edge may be represented as an array of identifier pairs. The first element of each of the identifier

pairs may be treated as a tail vertex by the symbolic AI system and the second element of the identifier pairs may be treated as a head vertex by the symbolic AI system. In some embodiments, the selected smart contract may already be in the process of being executed and the program state data of the program instance may include the norm statuses and scores of the smart contract state. For example, the program state data may be copied directly from the state data of a selected smart contract, where the changes effected by the outcome subroutines may be treated as scores.

[0099] A smart contract score may represent one of various types of values. For example, a smart contract score may represent a reputation score of an entity in a social network, a cryptocurrency value such as an amount of cryptocurrency, an amount of electrical energy, an amount of computing effort such as Ethereum's Gas, an amount of computing memory, or the like. A smart contract score may represent an objective value associated with an entity, such as an available amount of computing memory associated with the entity. Alternatively, a smart contract score may represent an amount by which a stored value is to be changed, such as a credit amount transferred from a first entity to a second entity.

[0100] In some embodiments, a program state may keep track of a plurality of scores. For example, a vertex of a directed graph of a symbolic AI model may include or otherwise be associated with a first score representing an amount of possessed by a first entity, a second score representing an amount owed to or owed by the first entity, a third score representing an amount possessed by a second entity, and a fourth score representing an amount owed to or owed by the second entity. In some embodiments, a conditional statement may be parsed to determine outcome scores. For example, an outcome subroutine associated with a vertex of a graph of the symbolic AI model may include instructions that a first entity is obligated provide 30 cryptocurrency units to a second entity and that the second entity is obligated to send a message to the first entity with an electronic receipt, and the system may determine that an associated score of the first vertex is equal to 30 and also determine that no score value is needed for the sending of the message. As further discussed below, by keeping track of scores and score changes, entire populations of smart contracts may be analyzed with greater accuracy without requiring a deep understanding of the specific terms or entity behaviors of any specific contract.

[0101] In some embodiments, a symbolic AI model may include statuses corresponding to each of a set of vertices representing the norms of a smart contract. The symbolic AI model

statuses may use the same categories as the norm statuses of a smart contract. Furthermore, the symbolic AI model status for a vertex may be identical to or be otherwise based on the status for the corresponding norm vertex being simulated. For example, if a norm status for a first norm vertex of a smart contract is “triggered – satisfied,” the symbolic AI model status for a first symbolic AI model vertex corresponding to the first norm vertex may also be “triggered – satisfied.” Alternatively, the system may select a different categorical value for a symbolic AI model vertex status that is still based on the corresponding norm status. Similarly, the symbolic AI model may include vertex categories similar to or identical to the logical categories associated with of the set of norm vertices of a smart contract. Furthermore, the symbolic AI model vertex category may be identical to or be otherwise based on the logical for the corresponding norm vertex being simulated. For example, if a logical category for a first norm vertex of a smart contract is “Rights” the symbolic AI model category for a first symbolic AI model vertex (“vertex category”) corresponding to the first norm vertex may also be “Rights.” Alternatively, the system may select a different categorical value for a vertex category that is still based on the corresponding logical category.

[0102] In some embodiments, the instantiated program may be a smart contract that may use or otherwise process events. Alternatively, or in addition, the program instance may be a modeling application and not an instance of the selected smart contract itself. For example, a symbolic AI system may be a modeling application that determines the values of a corresponding symbolic AI model based on the conditional statements of a smart contract without requiring that an event message be sent to an API of the modeling application. In some embodiments, the program instance of the symbolic AI system may change program state without performing one or more operations used by the smart contract that the program instance is based on. For example, the program instance of the symbolic AI system may change its program state data without deserializing serialized smart contract data, even if the smart contract that the program instance is based on includes operations to deserialize serialized smart contract data. In some embodiments, the program state data may be stored using a data model similar to that described in this disclosure for Figure 2. Alternatively, or in addition, the program state data may be stored in various other ways. For example, instead of storing values in separate arrays, the program instance may store the norm conditions, norm outcome actions, and their relationships to each other as part of a same array.

[0103] In some embodiments, the process 300 includes performing one or more iterations of the operations indicated by blocks 320, 324, 328, 332, 336, and 340 for each of the respective smart contracts or other programs of the selected set of smart contracts or other programs, as indicated by block 320. Furthermore, in some embodiments, the process 300 may include performing the one or more iterations in parallel using multiple processor cores. For example, some embodiments may include performing multiple iterations of the operations of the blocks 320, 324, 328, 332, 336, or 340 for multiple iterations in parallel using a plurality of processor cores. By performing the multiple iterations of the operations in parallel, computation times may be significantly reduced.

[0104] In some embodiments, the system may perform one or more iterations of operations to modify the statuses of a first set of vertices and then update the program state data based on the modified statuses in order to acquire a plurality of outcomes. The program state data or a portion of the program state data may be in a same state at the start each iteration, where two states of program state data are identical if both states have the same set of values. For example, if a first state of program state data is [1,2,3], and if a second state of program state data is [1,2,4], and if the program state data is reverted to [1,2,3], the reverted program state data may be described as being in the first state. In some embodiments, the system may execute the smart contract or smart contract simulation for a pre-determined number of iterations. Alternatively, or in addition, as further recited below, the smart contract or smart contract simulation may be repeatedly executed until a set of iteration stopping criteria are achieved. As further discussed below, the plurality of outcomes corresponding to the plurality of iterations may be used to provide one or more multi-iteration scores usable for decision-support systems and for determining multi-protocol scores.

[0105] In some embodiments, the system may modify one or more statuses associated with the vertices of the graph of the symbolic AI model based on a scenario and update the program state data based on the modified statuses, as indicated by block 328. In some embodiments, the scenario may be a set of inputs based on events. For example, a scenario may include simulated events or simulated event messages that may be testable by the conditions of a conditional statement. In response, a first vertex of the program instance may compare the simulated event to a condition and determine that a second vertex of the symbolic AI model of the should be activated. For example, an input may include an event “entity A transmitted data 0x104ABC to entity C,” which may satisfy a condition and change a status associated with a first vertex

associated with the conditional statement to “satisfied.” As discussed below, the system may then update the symbolic AI model based on the status change by activating an adjacent vertex to the first vertex.

[0106] Alternatively, or in addition, an input may include a message to change a program state without including an event that satisfies the norm conditions associated with the norm. For example, the input may include direct instructions interpretable by a symbolic AI system to set a vertex status to indicate that the corresponding vertex is triggered and direct which of a set of outcome subroutines to execute. The system may then update the symbolic AI model by activating one or more adjacent vertices described by the subset of outcome subroutines to execute.

[0107] In some embodiments, the scenario may include a single input. Alternatively, the scenario may include a sequence of inputs. For example, the scenario may include a first event, second event, and third event in sequential order. In some embodiments, the set of events may be generated using a Monte Carlo simulator. Some embodiments may randomly determine subsequent states from an initial state based on one or more probability distributions associated with each state of a set of possible subsequent states with respect to a previous state, where the probability distributions may be based on scores and logical categories associated with the set of possible states. For example, the program state may be in a state where only two subsequent possible states are possible, where the first subsequent possible state includes triggering a rights norm and a second subsequent possible state includes triggering an obligations norm.

[0108] In some embodiments, one or more inputs of a scenario may be determined using a decision tree. In some embodiments, a decision tree may be used to provide a set of decision based on scores, logical categories, statuses, and other factors associated with the active vertices of a simulated smart contract state. For example, a symbolic AI system may determine that the two possible states for a smart contract may result from either exercising a first rights norm or exercising of a second rights norm. A decision tree may be used to compare the logical categories, the scores associated with each norm, and the other information related to the active norms to determine which rights norm an entity would be most likely to exercise. In some embodiments, the symbolic AI system may compare a first score associated with a possible state represented by a first tree node with a second score of a different possible state represented by a second tree node. In response to the first score being greater than the second score, the symbolic AI system may determine a simulated input that will result in the future state

represented by the first tree node. Furthermore, in some embodiments, the decision tree may incorporate probability distributions or other decision-influencing factors to more accurately simulate real-world scenarios.

[0109] Alternatively, or in addition, some embodiments may include a Monte Carlo Tree Search (MCTS) method to generate a random sequence of events based on a set of possible events and a probability distribution by which the events may occur. The operations of the simulation may be made more efficient by selecting events that known to satisfy at least one condition of the set of conditional statements of the smart contract being simulated. In some embodiments, a symbolic AI system may determine a set of events for a smart contract simulation by determining a first simulated input based on a set of weighting values assigned to vertices of a graph of a symbolic AI model associated with norms of the smart contract. In some embodiments, the system may further determine a simulated input based on a count of the number of iterations of the simulation performed so far.

[0110] The system may then update the symbolic AI model based on the first simulated input, advancing the symbolic AI model to a second state. For example, after changing the status of a first vertex associated with an obligations norm from “unrealized” to “failed,” the symbolic AI model may then activate a first adjacent vertex representing a rights norm and a second adjacent vertex representing an prohibitions norm, where both adjacent vertices are adjacent to the first vertex. The symbolic AI system may then determine a second simulated input, wherein the second simulated input may be selected based on weighting value corresponding to each of the first adjacent vertex and second adjacent vertex, where the weighting value may be a score of the smart contract. For example, the weighting value of the first adjacent vertex may be $2/4$ and the weighting value of the second adjacent vertex may be $1/6$. Some embodiments may then update the symbolic AI model when it is in the second state based on the second simulated input in order to advance the second model to a terminal state, where a terminal state is one that satisfies a terminal state criterion. Once in a terminal state, the symbolic AI system may update the weighting values associated with the symbolic AI model before performing another iteration of the simulation.

[0111] Various terminal state criteria may be used. For example, a terminal state criterion may be that there is no further state change possible. Alternatively, a terminal state criterion may be that the smart contract is cancelled. The system may then update each of the weighting values associated with each of the nodes after reaching a terminal state before proceeding to perform

another iteration. In some embodiments, the symbolic AI system may set a status of a vertex to “failed” to simulate the outcomes of a first entity failing to transfer a score (e.g. failure to pay) a second entity.

[0112] In some embodiments, the determination of an input may be based on the type of conditional statement being triggered. As further discussed below, one or more of the conditional statements may be non-exclusively classified as one or more types of norms. Example of norm types include rights norms, obligations norms, or prohibition norms. As further discussed below, norm types may also include associations as being part of a pattern, such as a permission pattern. For example, a vertex may include or be otherwise associated with the label “consent or request.” By determining activities based on logical categories associated with the conditional statements instead of specific events, predictive modeling may be performed using globalized behavior rules without interpreting each of the globalized behavior rules for each specific contract. For example, a sequence of event may be generated a based on a first probability distribution that approximates an obligation of a first entity as having a 95% chance of being fulfilled and a 5% chance of being denied and a second probability distribution that approximates that a second entity has a 10% chance of cancelling a smart contract before the first entity exercises a right to cure the failure to satisfy the obligation. Using these rules, population scores associated with the population of smart contracts between a first entity and a second entity that consist of obligations norms to pay, rights norms to cure, and rights norms to cancel may be determined without regards to the specific structure of individual smart contracts in the population of smart contracts.

[0113] The system may then update each of the smart contract instances associated with a changed norm status, as discussed further below. Furthermore, the system may then update the respective associative array of conditions corresponding to the set of smart contracts. In some embodiments, an associative array of conditions may include only a subset of norm conditions associated with a smart contract, where each the subset of norm conditions is associated with a triggerable vertex of the smart contract. In some embodiments, the system may first deduplicate the norm conditions before performing a lookup operation to increase performance efficiency. For example, after determining that an event has occurred, some embodiments may search through a deduplicated array of norm conditions. For each norm condition that the event would trigger, the system may then update the one or more smart contracts associated with the norm condition in the deduplicated array of norm conditions.

[0114] Some embodiments may obtain a sequence of inputs instead of a single input. In some embodiments, the system may use a neural network to generate the sequence of inputs. In some embodiments, the neural network may determine a state value s based on the program state data and provide a vector of probabilities associated with for each of a set of possible changes in the program state. The neural network may also determine a state value to estimate the expected value of the program state after system applies the scenario to the program. In some embodiments, the neural network may use a MCTS algorithm to traverse a tree representing possible future states of the smart contract from a root state. The system may determine a next possible state s_{+1} for each state s by selecting a state with a low visit count, high predicted state value, and high probability of selection. The parameters (e.g. weights, biases, etc.) of the neural network making the state value determination may be represented by θ . After each iteration ending in a terminal state, the system may adjust the values θ to increase the accuracy of the neural network's predicted state value in comparison to the actual state value assessed whenever a terminal state is reached. Furthermore, a symbolic AI model may have a total score value, and the system may update the total score value based on the state value.

[0115] In some embodiments, the process 300 includes determining an outcome score based on the updated program state data, as indicated by block 336. In some embodiments, as stated in this disclosure, a set of scores may be associated with one or more of the outcome states. For example, an outcome of a first norm may include a transfer of currency values from a first entity to a second entity. The symbolic AI system may record this score and combine it with other scores in the same iteration in order to determine a net score for that score type. For example, the symbolic AI system may record each currency change based on inputs and outcomes in order to determine a net currency change, where a score of the smart contract may be the net currency change. Alternatively, or in addition, the symbolic AI system may record scores across different iterations to determine a multi-iteration score, as described further below. Example outcome scores may include a net amount of currency exchanged, a net amount of computing resources consumed, a change in the total cryptocurrency balance for an entity, or the like.

[0116] The process 300 may execute a number of iterations of smart contract state change simulations to determine possible outcomes and outcome scores. In some embodiments, there may be one or more criteria to determine if an additional iteration is needed, as indicated by block 340. In some embodiments, the one or more criteria may include whether or not a pre-

determined number of iterations of simulations have been executed. For example, some embodiments may determine that additional iterations are needed if the total number of executed iterations is less than an iteration threshold, where the iteration threshold may be greater than five iterations, greater than ten iterations, greater than 100 iterations, greater than 1000 iteration, greater than one million iterations, greater than one billion iterations, or the like. Alternatively, or in addition, the one or more criteria may include determining whether a specific outcome occurs. For example, the one or more criteria may include determining whether the outcome score is less than zero after a terminal state is reached. If the additional iterations are needed, operations of the process 300 may return to block 320. Otherwise, operations of the process 300 may proceed to block 344.

[0117] In some embodiments, the process 300 includes determining a multi-iteration score based on the outcome scores of executed iterations, as indicated by block 344. The multi-iteration score may be one of various types of scores and may include values such as a net change in score across multiple iterations, a probability distribution parameter, a measure of central tendency across multiple iterations, a measure of dispersion, or a measure of kurtosis. For example, the system may use a first outcome score from a first iteration, a second outcome score from a second iteration, or additional outcome scores from additional iterations to determine an average outcome score. The system may determine additional multi-iteration scores in the form of probability distribution parameters to determine a probability distribution. As used herein, a measure of kurtosis value may be correlated with a ratio of a first value and a second value, wherein the first value is based on a measure of central tendency, and wherein the second value is based on a measure of dispersion. For example, the measure of kurtosis may equal to μ^4/σ^4 , where μ may be a fourth central moment of a probability distribution and σ may be a standard deviation of the probability distribution.

[0118] In some embodiments, the multi-iteration score may be used to provide one or more predictions using Bayesian inference methods. In some embodiments, the multi-iteration score may be used to generate a probability distribution for the probability that a particular event or event type occurred based on a score, such as a change in currency value or an amount of computing resources consumed. For example, the system may calculate a mean average cryptocurrency amount determined across multiple iterations as a first multi-iteration score and a standard deviation of the cryptocurrency amount as the second multi-iteration score while

tracking the number of payment delays associated with the respective cryptocurrency amounts. The system may then use the first and second multi-iteration scores to generate a gaussian distribution, where the system may use the gaussian distribution to perform Bayesian inferences in order to determine a probability that a payment delay occurred after obtaining the value of a new cryptocurrency amount.

[0119] In some embodiments, the multi-iteration score may be a weight, bias, or other parameter of a neural network. For example, some embodiments may use a set of multi-iteration scores as weights of a neural network, where the training inputs of the neural network may be outcome scores and the training outputs of the neural network may be events, indicators representing activated outcome subroutines, or activated patterns. Once trained, the neural network may determine the probability of events, triggered conditional statements, or triggered patterns based on observed scores. In some embodiments, the parameters of the neural network may be transferred to other neural networks for further training. For example, a first neural network may be trained using the outcome scores as inputs and sets of events as outputs, and the weights and biases of the training may be transmitted to a second neural network for further training. The second neural network may then be used to indicate whether a particular event had a sufficiently high possibility of occurring based on a score or score change. In addition, the multi-iteration score may include outputs of a convolutional neural network, which may be used to determine behavior patterns across multiple smart contracts.

[0120] In some embodiments, the symbolic AI system may use a fuzzy logic method to predict the occurrence of an event based on the outcomes of a smart contract. A fuzzy logic method may include fuzzifying inputs by using one or more membership functions to determine a set of scalar values for each of a set of inputs, where the set of scalar values indicate the degree of membership of the inputs of a set of labels for each of the inputs of a smart contract being simulated by a symbolic AI system. For example, the system may use a membership function to determine a percentage values between 0% and 100% for a set of labels such as “profitable,” “risky,” or the like. The percentage values may indicate, for each of the smart contracts, a degree of membership to each of the labels. The symbolic AI system may then determine an fuzzified outcome score based on the set of fuzzified data by first using a set of rules in combination with an inference engine to determines the degree of match associated with the fuzzy input and determine which of the set of rules to implement. As used herein, an inference engine may be a system that applies a set of pre-defined rules. For example, an inference engine

may include a set of “if-then” expressions that provided responses to particular inputs. By using the inference engine in combination with the set of rules, the fuzzified outcome score may provide an indication of a broader label for the smart contract, such as “unconventional,” “risk too high,” or the like. In some embodiments, the symbolic AI system may defuzzify the fuzzified outcome score using various methods such as centroid of area method, bisector of area method, mean of maximum method, or the like. The defuzzifying process may result in a defuzzified outcome score that may also be used to determine a label.

[0121] In some embodiments, each of the scenarios may have an associated scenario weight, where the associated scenario may be a numeric value representing a normalized or nonnormalized probability of occurrence. For example, a smart contract may be processed based one of three possible scenarios, where the first scenario may have a weighting value equal to 0.5, the second scenario may have a weighting value equal to 0.35, and the third scenario may have a weighting value equal to 0.15. The system may use the associated scenario weights when determining a multi-iteration score. For example, if the first, second, and third scenarios results in allocating, respectively, 100, -10, or -100 computing resource units to a first entity, the system may determine that the expectation resource units allocated to the first entity is equal to 31.5 computing resource units and use the expectation resource units as the allocation value. While the above described using a scalar value as a weighting value, some embodiments may instead use a probability distribution as an associated scenario weight for each of the scenarios and determine the weighting value.

[0122] In some embodiments, the system may determine if data from an additional smart contract is to be processed, as indicated by block 350. As discussed in this disclosure, the process 300 may execute a number of simulations of different smart contracts to simulate possible outcomes and score changes. In some embodiments, each of the set of selected smart contracts may be simulated using a symbolic AI simulator. Furthermore, each of the set smart contracts may use the same set of weights/probability values to determine unique scenarios. For example, using the same set of weights corresponding to different combinations of available vertices, the system may determine a first scenario for a first symbolic AI model and a second scenario for a second symbolic AI model, where the first and second symbolic AI models have directed graphs that are different from each other. In some embodiments, the same weights may be used because the plurality of symbolic AI models may include vertices based on the same set of statuses and same set of logical categories. If the additional iterations are

needed, operations of the process 300 may return to block 308. Otherwise, operations of the process 300 may proceed to block 354.

[0123] In some embodiments, the process 300 includes determining a multi-protocol score based on the outcome scores across multiple smart contracts, as indicated by block 354. A multi-protocol score may be any score that is determined based on a plurality of outcomes from simulating different smart contracts, where the plurality of outcomes may include either or both multi-iteration scores or scores determined after a single iteration. In some embodiments, the multi-protocol score may be determined by determining a population of scores associated with a given entity. For example, a population of scores may be a population of expected income across a population of 500 instantiated smart contracts. The multi-protocol score may be a total income value, average income value, kurtosis income value, or the like.

[0124] In some embodiments, one or more methods to determine a multi-iteration score may be used to determine a multi-protocol score. For example, use of fuzzy logic, Bayesian inference, or neural networks may be used to predict multi-protocol scores. For example, some embodiments may use a first set of multi-iteration scores from a plurality of smart contract simulations as inputs and a second set of multi-iteration scores from the same plurality of smart contract simulations as outputs when training a neural network, where a set of multi-protocol score may be one or more the parameters of the trained neural network. For example, some embodiments may include a neural network trained to predict the probability that a specific type of smart contract was used based on multi-iteration scores such as an average payment duration and an average payment amount.

[0125] In some embodiments, multiple multi-protocol scores may be used to determine risk between a first entity and a second entity. For example, operations of the process 300 may be performed to determine a list of smart contracts shared by a first entity and a second entity and predict possible risks to the first entity in scenarios resulting from the incapability of the second entity to fulfill one or more norms in the list of smart contracts. In some embodiments, the risk posed to a first entity by a second entity may include considerations for intermediate relationships. For example, a first entity may be owed multiple amounts from a plurality of entities other than a second entity, and a second entity may owe multiple amounts to the plurality of entities. In some embodiments, a risk associated with the total amount of a score value to be collected by the first entity from the plurality of entities may be assessed based on the risk of the second entity failing to fulfill one or more obligations to transfer score values to

one or more of the plurality of entities. While the relationship between the first entity and the second entity may be difficult to determine using conventional smart contract systems if no explicit privacy relations are listed in the smart contracts, the symbolic AI models described in this disclosure allow these relationships to be determined by searching through entity lists or crawling through one or more privacy graphs.

[0126] Figures 4 – 9 below show a set of directed graphs that represent examples of program state of a smart contract or a simulation of a smart contract. Each vertex of the directed graph may represent conditional statements that encode or are otherwise associated with norm conditions and outcome subroutines that may be executed when a norm condition is satisfied. Each directed graph edge of the directed graph may represent a relationship between different conditional statements. For example, the tail vertex of a directed graph edge may represent a norm vertex that, if triggered, will activate the respective head vertex of the directed graph edge. As used in this disclosure, the direction of a directed graph edge points from the tail vertex of the directed graph edge to the head vertex of the directed graph edge. Furthermore, the direction of the of the directed graph edge may indicate that the respective head vertex to which the directed graph edge points is made triggerable based on a triggering of the respective tail vertex. In some embodiments, a norm vertex may be triggered if the trigger direction is the same as the directed graph edge direction for each directed graph edge. In some embodiments, the direction of a directed graph edge associated with norm condition may be used to categorize a norm or norm vertex.

[0127] Figure 4 shows a computer system for operating one or more symbolic AI models, in accordance with some embodiments of the present techniques. As shown in FIG. 4, a system 400 may include computer system 402, first entity system 404, second entity system 406 or other components. The computer system 402 may include a processor 412 and a local memory 416, or other components. Each of the first entity system 404 or second entity system 406 may include any type of mobile computing device, fixed computing device, or other electronic device. In some embodiments, the first entity system 404 may perform transactions with the second entity system 406 by sending messages via the network 450 to the computer system 402. In some embodiments, the computer system 402 may execute one or more applications using one or more symbolic AI models with a processor 412. In addition, the computer system 402 may be used to perform one or more of the operations described in this disclosure for the process 100 or the process 300. Parameters, variables, and other values used by a symbolic AI

model or provided by the symbolic AI model may be retrieved or stored in the local memory 416. In some embodiments, parameters, variables, or other values used or provided by the computer system 402, entity systems 404 – 406, or other systems may be sent to or retrieved from the remote data storage 444 via the network 450.

[0128] Figure 5 includes a set of directed graphs representing triggered norms and their consequent norms, in accordance with some embodiments of the present techniques. The table 500 shows various triggered vertices and their respective consequent vertices in the form of directed graphs. In some embodiments, the system may include categories for norms of a smart contract based on a deontic logic model, where the categories may include obligation norms, rights norms, or prohibition norms. In addition to various contract-specific ramifications of these categories, norms within each category may share a common set of traits with respect to their transiency and possible outcomes. As shown in table 500, the relationship between a triggered norm and its consequent norms may be represented as a directed graph, where each of the norms may be represented by a vertex of the directed graph and where each triggering event may be used to as a label associated with a graph edge.

[0129] Box 510 includes a directed graph representing a smart contract state (or simulation of the smart contract state) after an event satisfying a norm condition of the obligation norm represented by the norm vertex 511. As shown in box 510, after a determination that the norm condition P associated with the norm vertex 511 is satisfied by an event (indicated by the directed graph edge 512), the system may generate an adjacent vertex 513 indicating that the norm vertex 511 is satisfied, where a norm status of the adjacent vertex 513 may be set as “terminal” to indicate that the adjacent vertex is terminal. In some embodiments, a determination that the state of the smart contract or simulation thereof is terminal may be made if a vertex of the smart contract or simulation thereof indicated to be terminal. In some embodiments, instead of generating the adjacent vertex 513, the system may update a norm status associated with the norm vertex 511 to indicate that the norm vertex 511 is satisfied. For example, the system may update a norm vertex associated with the obligation norm by setting a norm status associated with the norm vertex to “satisfied,” “terminal,” or some other indicator that the obligation norm has been satisfied by an event. In some embodiments, updating a norm vertex associated with the obligation norm may be represented by the statement $6, \text{ where } \overset{P}{\rightarrow}$ represents a result of the norm condition associated with the obligation norm “OP” being

satisfied, and S represents the generation of a norm vertex indicating that the conditions of the obligation norm have been satisfied:

$$\text{In } OP \xrightarrow{P} S \dots\dots\dots (6)$$

[0130] As shown in box 520, a norm condition P may end up not satisfying a norm condition associated with the norm vertex 521 after satisfying a condition expiration threshold, where the norm vertex 521 is associated with an obligation norm. In response, the system may update the norm vertex 521 by setting a norm status associated with the norm vertex 521 to “failed” or some other indicator that the norm condition associated with the norm vertex 521 has been not satisfied. For example, an event may indicate that a condition expiration threshold has been satisfied without an obligation norm condition being satisfied. In response, the system may generate or otherwise set as triggerable the set of consequent norms associated with adjacent vertices 523, where the relationship between a failure to satisfy a norm condition P of the norm vertex 521 and the adjacent vertices 523 is indicated by the directed graph edges 522. In some embodiments, the generation of the adjacent vertices may be represented by the statement 7, where $\neg^P \rightarrow$ indicates that the instructions to the right of the symbol $\neg^P \rightarrow$ are to be performed if a norm condition “OP” is not satisfied, and the instructions represented by the symbolic combination $\Lambda_i X_i Q_i$ represents the generation or activation of the consequent norms that result from the failure of OP:

$$\begin{array}{l} OP \qquad \qquad \qquad (7) \\ \neg^P \rightarrow \Lambda_i X_i Q_i \dots\dots\dots \end{array}$$

[0131] In some embodiments, in response to an event satisfying a norm condition of a rights norm, the system may update a norm vertex associated with the rights norm by setting a norm status associated with the norm vertex to “exercised” or some other indicator that the rights norm has been triggered based on an event. For example, as shown in box 530, in response to an event satisfying a norm condition associated a rights norm represented by the norm vertex 531, the system may update a norm vertex associated with the norm vertex 531 by setting a norm status associated with the norm vertex 531 to “exercised” or some other indicator that the rights norm has been exercised. In response, the system may generate or otherwise set as triggerable the set of consequent norms associated with adjacent vertices 533, where the relationship between satisfying a norm condition P associated with the norm vertex 531 and

the set of consequent norms associated with adjacent vertices 533 is indicated by the directed graph edges 532. Furthermore, in some embodiments, a rights norm may be contrasted with an obligation norm by allowing a rights norm to remain triggerable after triggering. This may be implemented by further generating or otherwise setting as triggerable the rights norm associated with the rights norm vertex 534. In some embodiments, a rights norm may expire after use. For example, some embodiments may not generate the rights norm vertex 534 after triggering the norm vertex 531. In some embodiments, the operation described above may be represented by statement 8 below, where the result of triggering a rights norm RP_1 by satisfying the norm condition P may result in a conjunction of newly-triggerable consequent norms $\Lambda_i X_i Q_i$ and a rights norm RP_2 that is identical to the rights norm RP_1 , where Λ represents a mathematical conjunctive operation:

$$RP_1 \xrightarrow{P} \Lambda_i X_i Q_i \wedge RP_2 \dots \dots \dots (8)$$

[0132] In some embodiments, in response to an event satisfying the norm condition of a “prohibition” norm, the system may update a norm vertex associated with the “prohibition” norm by setting a norm status associated with the norm vertex to “violated” or some other indicator that the “prohibitions” norm has been triggered based on an event. For example, as shown in box 550, an event may satisfy a norm condition P associated with the prohibition norm represented by a norm vertex 551. In response, the system may update the norm vertex 551 by setting a norm status associated with the norm vertex 551 to “violated” or some other indicator that the associated prohibitions norm condition has been satisfied. In response, the system may generate or otherwise set as triggerable the set of consequent norms associated with adjacent vertices 553, where the relationship between satisfying a norm condition P associated with the norm vertex 551 and the set of consequent norms associated with adjacent vertices 553 is indicated by the directed graph edges 552. Furthermore, in some embodiments, a prohibitions norm may be contrasted with an obligation norm by allowing a prohibitions norm to survive triggering. In addition, in some embodiments, triggering a prohibitions norm may result in the system decreasing a value representing the state of the smart contract. This may be implemented by further generating or otherwise setting as triggerable the prohibitions norm associated with the prohibitions norm vertex 554 after triggering the norm vertex 551. In some embodiments, the operation described above may be represented by statement 9 below, where the result of triggering a prohibition norm PP_1 by satisfying the norm condition P may result in a conjunction of newly-triggerable consequent norms $\Lambda_i X_i Q_i$ and a prohibition norm

PP_2 that is identical to the prohibition norm PP_1 , where Λ represents a mathematical conjunctive operation:

$$PP_1 \xrightarrow{P} \Lambda_i X_i Q_i \Lambda PP_2 \dots \dots \dots (9)$$

[0133] Figure 6 includes a set of directed graphs representing possible cancelling relationships and possible permissive relationships between norms, in accordance with some embodiments of the present techniques. The table 600 includes a left column that includes a directed graph 610 representing an initial state and a directed graph 620 that represents a first possible outcome state of the initial state and a directed graph 630 that represents a second possible outcome state of the initial state. In some embodiments, a norm condition may be a cancellation condition, where satisfying a cancellation condition results in the cancellation of one or more norms. Cancelling a norm may include deactivating a norm, deleting the norm, deleting graph edges to the norm, or otherwise or otherwise setting the norm as not triggerable. For example, an obligations norm may include a cancellation outcome subroutine, where triggering the obligations norm may result in the cancellation of one or more norms adjacent to the obligations norm. In some embodiments, the effect of satisfying a cancellation norm may be represented by statement 10 below, where XP may represent an obligations norm, $\xrightarrow{P, \neg P}$ may indicate that the event which triggers the norm XP occurs when the norm condition P is either satisfied or failed, $\Lambda_i X_i Q_i$ represents the set of consequent norms that are set to be triggerable based on the event triggering XP , and $X_j U_j$ may represent the set of consequent norms that cancelled based on event triggering XP :

$$XP \xrightarrow{P, \neg P} \Lambda_i X_i Q_i \Lambda \neg X_j U_j \dots \dots \dots (10)$$

[0134] As shown by statement 10 above, one or more norms may be cancelled. In some embodiments, a cancellation may be implemented as an inactive graph edge between the norm XP and the norms $X_j U_j$, where the graph edge representing the conditional relationship between the norm XP and the norms $X_j U_j$ are directed towards the norm XP . In some embodiments, the cancellation of a norm may be implemented by setting an indicator to indicate that a norm or condition associated with the cancelled norm is no longer triggerable.

[0135] The directed graph 610 may represent a state of a start contract and may include the first vertex 611, second vertex 613, third vertex 617, and fourth vertex 619, each of which are

associated with a norm of a smart contract. The directed graph 610 also depicts a mutual cancellation relationship between the norm associated with the second vertex 613 and the third vertex 617 represented by the XQ1-XQ2 graph edge 614, where a mutual cancellation relationship of a pair of norm vertices may include a cancellation of one norm vertex of the pair upon triggering of the other norm vertex of the pair. The directed graph 610 also depicts a unidirectional cancellation relationship between the norm associated with the fourth vertex 619 and the third vertex 617 as represented by the XP2-XQ2 graph edge 618. In some embodiments, satisfying or otherwise triggering the norm associated with the third vertex 617 may instantiate the RZ-XQ2 graph edge 618 and cancel the fourth vertex 619. In some embodiments, each of vertices and graph edges shown in Figure 5 may be represented using a protocol simulation program. For example, the first vertex 611 may be modeled in a simulation program and may be associated with a conditional statement of a smart contract.

[0136] In some embodiments, the state represented by the directed graph 610 may advance to the state represented by the directed graph 620. The state represented by the directed graph 620 may be achieved by triggering the norm associated with the second vertex 613, which may result in the cancellation of the norm associated with the third vertex 617. Furthermore, as illustrated by the directed graph 620, triggering the norm associated with the second vertex 613 may also result in the activation of fifth vertex 621 and sixth vertex 623. In addition, triggering the norm associated with the third vertex 617 may result in the cancellation of the norm associated with the fourth vertex 619. Furthermore, as illustrated by the directed graph 630, triggering the norm associated with the third vertex 617 may also result in the activation of a seventh vertex 631 and eighth vertex 633. Each of these triggering behaviors may be implemented directly by a smart contract.

[0137] In some embodiments, the triggering relationship described in this disclosure may be modeled using a symbolic AI system that may keep track of any scores associated with events that trigger the norms and the outcomes of triggering the norms. For example, a first probability value may be assigned to the state represented by the directed graph 620 and a second probability value may be assigned to the state represented by the directed graph 630 during a simulation of the smart contract. The symbolic AI system may use the first and second probability values to advance the state represented by either the directed graph 620 or the directed graph 630 over multiple iterations to compute a multi-iteration score using the methods described in this disclosure. For example, if the first probability value is 20% and the second

probability value is 80%, and a first score represented by the directed graph 620 is equal to 100 cryptocurrency units and a second score represented by the directed graph 630 is equal to 1000 cryptocurrency units, a multi-iteration score may be equal to 820 cryptocurrency units.

[0138] The right column of table 600 includes a directed graph 650, which may represent an initial state of a smart contract (or simulation thereof). The right column of table 600 also includes a directed graph 660 that represents a first possible outcome state of the initial state and a directed graph 670 that represents a subsequent possible outcome state of the first possible outcome state. The initial state represented by the directed graph 650 may include a permissive condition of a permission norm, where satisfying a permissive condition may result in the activation of one or more norms. For example, after being activated, a rights norm RP may include a set of permissions $\{RV_k\}$ that are triggered after satisfying an norm condition associated with the rights norm RP , where the rights norm RP may also be described as a permission norm. Triggering the set of permissions $\{RV_k\}$ may either set the norm XP to be triggerable or otherwise prevent an outcome subroutine of the norm XP from being executed until the set of permissions $\{RV_k\}$ are triggered. This relationship may be represented by statement 11 below, where XP may represent an obligations norm, RV_k represents the permissions that must be triggered before XP may be triggered, $\xrightarrow{P, \neg P}$ may indicate that the event which triggers the norm XP occurs when the norm condition P is either satisfied or failed, $\Lambda_i X_i Q_i$ represents the set of consequent norms that are set to be triggerable based on the event triggering XP after the permissions RV_k are triggered, and $X_j U_j$ may represent the set of consequent norms that cancelled based on event triggering XP after the permissions RV_k are triggered:

$$XP|RV_k \xrightarrow{P, \neg P} \Lambda_i X_i Q_i \wedge \Lambda_j \neg X_j U_j \dots \dots \dots (11)$$

[0139] As shown by statement 11 above, XP may be set to be triggerable upon triggering of the permission RV_k . Triggering XP after the permissions RV_k are triggered results in activation of the consequent norms $\Lambda_i X_i Q_i$ and cancels the norms $X_j U_j$. In some embodiments, the conditions needed to trigger permissions may be activated in conjunction with rights norms dependent on the permissions, and thus XP and RV_k may be activated as a result of triggering the same triggered norm. In some embodiments, permission behavior may be performed by a smart contract or a simulation thereof by modifying a first status of a first vertex and a second status of a second vertex to indicate that the first and second vertices are triggered, where the

first vertex may represent a first rights norm such as XP and the second vertex may represent a permission norm such as a norm having outcome permissions RV_k . The smart contract, or a simulation thereof, may trigger a third vertex that is adjacent to the first vertex and the second vertex such as a vertex in $\Lambda_i X_i Q_i$ in response to the first and second statuses now being triggered.

[0140] The directed graph 650 may include a first vertex 651, second vertex 653, third vertex 657, and fourth vertex 659. The directed graph 650 also depicts a mutual cancellation relationship between the norm associated with the second vertex 653 and the third vertex 657 represented by the XQ1-XQ2 graph edge 654. The directed graph 650 also depicts a permission relationship between the norm associated with the fourth vertex 659 and the third vertex 657 as represented by the RZ-XQ2 graph edge 658, where the fourth vertex 659 may include or otherwise be associated with permission conditions that must be satisfied in order to trigger the third vertex 657. In some embodiments, satisfying or otherwise triggering the norm associated with the fourth vertex 659 may instantiate the RZ-XQ2 graph edge 658 and allow the outcome subroutines of the third vertex 657 to be executed.

[0141] In some embodiments, the program state represented by the directed graph 650 may produce an outcome state represented by the directed graph 660. The outcome state represented by the directed graph 660 may be achieved by satisfying a norm condition associated with the fourth vertex 659. In some embodiments, after the XQ1-XQ2 graph edge 654 becomes instantiated, an event satisfying a norm condition associated with the third vertex 657 may result in the program state represented by the directed graph 670. The directed graph 670 may represent a program state where the norm associated with the third vertex 657 is triggered, resulting in the activation of additional norms associated with the fifth vertex 671 and sixth vertex 673.

[0142] In some embodiments, a symbolic AI system may be used to generate a scenario that includes a sequence of inputs having a first input and a second input. The first input may advance the state represented by the directed graph 650 to the state represented by the directed graph 660 and the second input may advance the state represented by the directed graph 660 to the state represented by the directed graph 670. The sequence of inputs may be determined using any of the methods described in this disclosure. For example, the sequence of inputs may be determined using a Monte Carlo method, a neural network, or the like.

[0143] Figure 7 includes a set of directed graphs representing a set of possible outcome states based on events corresponding to the satisfaction or failure of a set of obligations norms, in accordance with some embodiments of the present techniques. The set of directed graphs 710 includes a set of three vertices 711 – 713, each representing an obligation norm to perform a set of related tasks. In some embodiments, the obligation norm may represent an obligation to transmit digital assets, deliver a data payload, or perform a computation. For example, the obligation norm represented by the first vertex 711 may be associated with an obligation for a first entity to transmit a down payment to a second entity, where a determination that the down payment occurred may be based on an event message sent by the second entity confirming that payment was delivered. The obligation norm represented by the second vertex 712 may be associated with an obligation for the second entity to deliver an asset to the first entity, where a determination that the asset was delivered may be based on an event message sent by the second entity confirming that the asset was delivered. The obligation norm represented by the third vertex 713 may be associated with an obligation for the first entity to pay a balance value to the second entity.

[0144] The set of directed graphs 720 may represent a first outcome state that may result from the program state represented by the set of directed graphs 710, where each of the obligation norms represented by the three vertices 711 – 713 are satisfied. In some embodiments, a smart contract simulation system such as a symbolic AI system may assign a probability value to the possibility the state represented by the set of directed graphs 710 is advanced to the outcome state represented by the set of directed graphs 720. For example, a symbolic AI system may assign a probability for the outcome state represented by the set of directed graphs 720 to be equal to 82% when starting from the state represented by the set of directed graphs 710. The symbolic AI system may then perform a set of simulations based on this probability value using a Monte Carlo simulator.

[0145] The set of directed graphs 730 may represent a second outcome state that may result from the program state represented by the set of directed graphs 710, where the first obligation is not satisfied and the time has exceeded a condition expiration threshold associated with the first vertex 711. As shown in the set of directed graphs 730, a failure to meet the first obligation represented by the first vertex 711 may result in a system generating or otherwise activating norms associated with a fourth vertex 721 and a fifth vertex 722. In some embodiments, the norm associated with the fourth vertex 721 may represent a first entity's right to cure the

payment failure and the norm associated with the fifth vertex 722 may represent a second entity's right to terminate the smart contract. The bidirectional graph edge 723 indicates that triggering one of the pair of vertices 721 – 722 will cancel or otherwise render as inactive the other of the pair of vertices 721, which may indicate that curing a failed obligation and terminating the smart contract may be mutually exclusive outcomes. In some embodiments, a symbolic AI system (or other modeling system) may assign a probability value to the possibility the state represented by the set of directed graphs 710 is advanced to the outcome state represented by the set of directed graphs 720. For example, the symbolic AI system may assign a probability for the outcome state represented by the set of directed graphs 720 to be equal to 6% when performing a simulation based on the smart contract program state represented by the set of directed graphs 710.

[0146] In some embodiments, the state represented by the set of directed graphs 730 may be advanced to the state represented by a set of directed graphs 740. In some embodiments, the state represented by a set of directed graphs 740 may be an outcome state after the norm associated with the fourth vertex 721 is triggered. As shown in the set of directed graphs 740, triggering the norm associated with the fourth vertex 721 may result in cancelling the norm associated with fifth vertex 722. In some embodiments, a symbolic AI system may use probability value representing the probability of the state represented by the set of directed graphs 730 advancing to the state represented by a set of directed graphs 740. For example, a symbolic AI system may use 50% as the probability that the state represented by the set of directed graphs 730 advances to the state represented by a set of directed graphs 740. If the probability of the state represented by the set of directed graphs 720 advancing to the state represented by a set of directed graphs 730 is equal to 6%, this would mean that the probability of the state represented by the set of directed graphs 710 advancing to the state represented by a set of directed graphs 740 is equal to 3% by applying the multiplication rule for the probability of independent events.

[0147] In some embodiments, the state represented by the set of directed graphs 730 may be advanced to the state represented by a set of directed graphs 750. In some embodiments, the state represented by a set of directed graphs 750 may be an outcome state after the norm associated with the fifth vertex 722 is triggered. As shown in the set of directed graphs 750, triggering the norm associated with the fifth vertex 722 may result in cancelling the norm associated with second vertex 712, third vertex 713, and fourth vertex 721. In some

embodiments, a symbolic AI system may assign a probability value to the possibility of a smart contract state being in the outcome state represented by the set of directed graphs 750 when starting from the program state represented by the set of directed graphs 730. In some embodiments, the probability values associated with each state may be updated after each iteration in a set of simulated iterations using one or more of the methods in this disclosure. For example, some embodiments may apply a MCTS method to explore the program states represented by the sets of directed graphs 710, 720, 730, and 740 across multiple iterations while keeping track of scores for each iteration in order to determine outcome scores for each iteration and multi-iteration scores.

[0148] Figure 8 includes a set of directed graphs representing a set of possible outcome states after a condition of a second obligations norm of a set of obligations norms is not satisfied, in accordance with some embodiments of the present techniques. In some embodiments, the set of directed graphs 810 may represent an initial state of a smart contract. Alternatively, the set of directed graphs 810 may represent an outcome state. For example, the program state represented by the set of directed graphs 810 may be an outcome state of the program state represented by the set of directed graphs 710, with an associated occurrence probability equal to 6%. The set of directed graphs 810 may represent a failure to satisfy a norm condition associated with the second vertex 812. In some embodiments, the second vertex 812 may represent an obligation norm indicating an obligation for a second entity to deliver an asset, such as a schematic, to the first entity.

[0149] In some embodiments, the state represented by the set of directed graphs 810 may be advanced to the state represented by a set of directed graphs 820. In some embodiments, the state represented by a set of directed graphs 820 may be an outcome state after the norm associated with the fifth vertex 822 is triggered. As shown in the set of directed graphs 820, triggering the norm associated with the fifth vertex 822 may result in cancelling the norm associated with sixth vertex 823. In some embodiments, the fifth vertex 822 may represent a first entity's right to terminate the order and obtain a refund. This outcome may be represented by the eighth vertex 831, which may represent an obligation norm indicating that the second entity has an obligation to pay the first entity, and that this obligation may either be satisfied or failed, as indicated by vertices 841 and 842, respectively.

[0150] In some embodiments, the state represented by the set of directed graphs 810 may be advanced to the state represented by a set of directed graphs 830. In some embodiments, the

state represented by a set of directed graphs 820 may be an outcome state after the norm associated with the sixth vertex 823 is triggered. As shown in the set of directed graphs 830, triggering the norm associated with the sixth vertex 823 may result in cancelling the norm associated with sixth vertex 823. In some embodiments, the sixth vertex 823 may represent a first entity's right to cure the failure to satisfy the norm represented by the second vertex 812. This outcome may be represented by the ninth vertex 832, which may represent an obligation norm indicating that the second entity has an obligation to deliver an asset to the first entity, and that this obligation may either be satisfied or failed, as indicated by vertices 843 and 844, respectively.

[0151] In some embodiments, a symbolic AI system may assign a probability value to the possibility of a smart contract state being in the outcome state represented by the set of directed graphs 820 or set of directed graphs 830 when starting from the program state represented by the set of directed graphs 810. For example, a symbolic AI system may determine that the probability that the outcome state represented by the set of directed graphs 820 is equal to 40%. Similarly, the symbolic AI system may determine that the probability that the outcome state represented by the set of directed graphs 830 is equal to 60%. In some embodiments, the symbolic AI system may use a Bayesian inference to determine if an obligation norm was failed based on a probability distribution computed from the scores associated with program states such as those states represented by the sets of directed graphs 820 or 830. For example, the symbolic AI system may acquire a new score value and, based on the score value, predict whether an obligation represented by the second vertex 812 was failed.

[0152] Figure 9 includes a set of directed graphs representing a set of possible outcome states after a condition of a third obligations norm of a set of obligations norms is not satisfied, in accordance with some embodiments of the present techniques. In some embodiments, the set of directed graphs 910 may represent an initial state of a smart contract. Alternatively, the set of directed graphs 910 may represent an outcome state. For example, the program state represented by the set of directed graphs 910 may be an outcome state of the program state represented by the set of directed graphs 810, with an associated occurrence probability equal to 6%. The set of directed graphs 910 may represent a failure to satisfy a norm condition associated with the third vertex 913. In some embodiments, the third vertex 913 may represent an obligation norm indicating an obligation for a first entity to pay a balance value to the second entity. Triggering the norm associated with third vertex 913 by failing to satisfy an associated

obligation condition may result in activating norms associated with a sixth vertex 923 and a seventh vertex 924. In some embodiments, the norm associated with the sixth vertex 923 may represent a first entity's right to cure the payment failure and the norm associated with the seventh vertex 924 may represent a second entity's right to declare a breach and flag the first entity for further action (e.g. initiate arbitration, incur a reputation score decrease, or the like).

[0153] In some embodiments, the state represented by the set of directed graphs 910 may be advanced to the state represented by a set of directed graphs 920. In some embodiments, the state represented by a set of directed graphs 920 may be an outcome state after the norm associated with the sixth vertex 923 is triggered. In some embodiments, the norm associated with the sixth vertex 923 may represent a first entity's right to cure the payment failure, and thus triggering the rights norm associated with the sixth vertex 923 may represent a first entity's right to cure the failure. As indicated by the satisfaction vertex 931, curing the payment failure may end all outstanding obligations of the smart contract.

[0154] In some embodiments, the state represented by the set of directed graphs 910 may be advanced to the state represented by a set of directed graphs 930. In some embodiments, the state represented by a set of directed graphs 930 may be an outcome state after the norm associated with the seventh vertex 924 is triggered. In some embodiments, the norm associated with the seventh vertex 924 may represent a second entity's right to declare a breach, and thus triggering the rights norm associated with the seventh vertex 924 may represent a second entity's declaration of contract breach. This may result in the activation of the failure vertex 932, which may include outcome subroutines that sends a message indicating that the smart contract is in breach to a third party or sends instructions to an API of another application.

[0155] Figure 10 includes a set of directed graphs representing a pair of possible outcome states after a condition of a fourth obligations norm of a set of obligations norms is not satisfied, in accordance with some embodiments of the present techniques. Figure 10 includes a directed graph 1010 representing a first program state of a smart contract or a symbolic AI simulation thereof. The program state represented by the directed graph 1010 may be changed to the program state represented by a directed graph 1020. Alternatively, the program state represented by the directed graph 1010 may be changed to the program state represented by a directed graph 1030. The directed graph 1010 includes a first vertex 1011 that may represent an obligations norm. In some embodiments, the first vertex 1011 may represent an obligation norm reflecting an obligation to pay by the time a condition expiration threshold is satisfied. If

the obligation to pay is failed, the obligation norm associated with the first vertex 1011 may be triggered and the rights norms associated with the second vertex 1012 and the third vertex 1013 may be activated. The second vertex 1012 may represent a rights norm to cure the failure to satisfy the obligations norm represented by the first vertex 1011, and the third vertex 1013 may represent a rights norm to accelerate the payments the smart contract. The directed graph 1010 also includes a pair of vertices 1014-1015 representing future obligations to pay, where exercising the rights norm represented by the third vertex 1013 may cancel the future obligations to pay.

[0156] In some embodiments, the state represented by the directed graph 1010 may be advanced to the state represented by the directed graph 1020. In some embodiments, the state represented by the directed graph 1020 may be an outcome state after the norm associated with the second vertex 1012 is triggered. In some embodiments, the norm associated with the second vertex 1012 may represent a right to cure the failure to satisfy the norm condition associated with the first vertex 1011. As indicated by the directed graph 1020, exercising the rights norm associated with the second vertex 1012 may satisfy the norm and activate the vertex 1023, which may indicate that the rights norm associated with the second vertex 1012 has been satisfied.

[0157] In some embodiments, the state represented by the directed graph 1010 may be advanced to the state represented by the directed graph 1030. In some embodiments, the state represented by the directed graph 1030 may be an outcome state after the norm associated with the third vertex 1013 is triggered. In some embodiments, the rights norm associated with the third vertex 1013 may represent a right to accelerate payment. Triggering the rights norm associated with the third vertex 1013 may cancel the rights norm associated with the second vertex 1012. In addition, triggering the rights norm associated with the third vertex 1013 may also cancel the obligation norms associated with the vertices 1014-1015. Triggering the rights norm associated with the third vertex 1013 may cause the system to activate a new obligation norm associated with the fourth vertex 1031. In some embodiments, the new obligation norm may include norm conditions to determine whether a first entity transmits a payment amount to the second entity. For example, the new obligation norm may determine whether the first entity transmitted the entirety of a principal payment of a loan to the second entity. The obligation norm associated with the fourth vertex 1031 may be associated to a satisfaction norm represented by a fifth vertex 1041 or a failure norm represented by a sixth vertex 1042.

[0158] In some embodiments, advancement of the state represented by the directed graph 1010 to the state represented by the directed graph 1020 or the state represented by the directed graph 1030 may be simulated using a symbolic AI system. For example, the state represented by the directed graph 1010 may be copied into a symbolic AI model, where both the conditional statements associated with the nodes and of the directed graph the edges connecting the nodes of the directed graph may be copied. A symbolic AI system may then simulate state changes using the symbolic AI model to determine an expected value for a smart contract that has already reached the state represented by the directed graph 1010, where the expected value may be a multi-iteration score.

[0159] In some embodiments, each of the smart contracts represented by the directed graphs 610, 650, 710, and 1010 may be analyzed using a symbolic AI system to determine one or more multi-protocol scores. For example, each of the smart contracts represented by the directed graphs 610, 650, 710, and 1010 may be analyzed to produce multi-iteration scores such as average scores for each smart contract and a kurtosis value of expected scores. In some embodiments, the analysis may use the same rules to govern the behavior entities in the smart contract by basing the rules on logic types and vertex statuses instead of the contexts of specific agreements. For example, each smart contract simulation may be simulated with a set of rules that include a rule that the probability that a rights norm to cure is triggered instead of a rights norm to accelerate being triggered is equal to 90%. The multi-iteration scores may then be further analyzed to determine a multi-protocol score. For example, based on a multi-iteration score representing a risk score associated with each of the smart contracts, the total exposed risk of a first entity with respect to a second entity may be determined, where the total exposed risk may be a multi-protocol score.

[0160] Figure 11 is a block diagram illustrating an example of a tamper-evident data store that may be used to render program state tamper-evident and perform the operations in this disclosure, in accordance with some embodiments of the present techniques. In some embodiments, the tamper-evident data store may be a distributed ledger, such as a blockchain (or other distributed ledger) of one of the blockchain-based computing platforms described in this disclosure. Figure 11 depicts two blocks in a blockchain, and also depicts tries of cryptographic hash pointers having root hashes stored in the two blocks. The illustrated arrows may represent pointers (e.g., cryptographic hash). For example, the arrow 1103 may represent a pointer from a later block to block 1104 that joins the two blocks together. In some embodiments, blocks may be

consecutive. Alternatively, the data from the use of a smart contract may skip several blocks between uses of the smart contract. As shown in Figure 11, a tamper-evident data store 1102 may include a linked list of blocks that includes the block 1104 and other blocks, where the linked list of blocks may be connected by cryptographic hash pointers.

[0161] In some embodiments, a directed acyclic graph of cryptographic hash pointers may be used to represent the tamper-evident data store 1102. Some or all of the nodes of the directed acyclic graph may be used to form a skip list or linked list, such as the node corresponding to or otherwise representing as block 1104. In some embodiments, each block represented by a node of this list may include multiple values as content. For example, each respective block may include a timestamp of creation 1106, a cryptographic hash of content of the previous node pointed to by an edge connecting those nodes 1108, a state root value 1110 for a trie of cryptographic hash values that may be referred to as a state trie 1118, a cryptographic hash 1112 that is a root value of a receipt trie 1124 of cryptographic hash values referred to as a receipt trie, and a cryptographic hash value 1114 that is a root value of a trie of cryptographic hash values referred to as a transaction trie 1122. In some embodiments, the block 1104 may be connected to a plurality of tries (e.g., three or more tries) via cryptographic hash pointers. For example, the block 1104 may be connected to Merkle roots (or other roots) of the plurality of tries of cryptographic hash values.

[0162] In some embodiments, the state trie 1118 may include multiple levels of cryptographic hash pointers that expand from a root to leaf nodes through 2 or more (e.g. 3, 11, 5, 6, etc.) hierarchical levels of branching. In some embodiments, an account address of a smart contract or instance of invocation thereof may correspond to a leaf nodes, where the smart contract may be an instance of the smart contract described in one or more operations of one or more processes described in this disclosure. In some embodiments, leaf nodes or paths to the leaf nodes of the state trie 1118 may include the fields in the account object 1126. The address may be a smart contract address or instance of invocation of the smart contract, the nonce value may be a count of the times that the smart contract was invoked, the code hash value may be or otherwise include a cryptographic hash of a bytecode representation of the smart contract 1130, the storage hash may be a root (e.g. Merkle root) of a trie of cryptographic hash pointers 1120. In some embodiments, the trie of cryptographic hash pointers 1120 may store key-value pairs encoding a transient program state of the smart contract that changes or is not needed between invocations of the smart contract. In some embodiments, the fields of the account object 1126

may include a predecessor pointer that points to a previous entry of an earlier state trie corresponding to a previous invocation of the smart contract and associated information or hashes.

[0163] Figure 12 depicts an example logical and physical architecture of an example of a decentralized computing platform in which a data store of or process of this disclosure may be implemented, in accordance with some embodiments of the present techniques. In some embodiments, there may be no centralized authority in full control of a decentralized computing platform 1200. The decentralized computing platform 1200 may be executed by a plurality of different peer computing nodes 1202 via the ad hoc cooperation of the peer computing nodes 1202. In some embodiments, the plurality of different peer computing nodes 1202 may execute on a single computing device, such as on different virtual machines or containers of a single computing device. Alternatively, or in addition, the plurality of different computing nodes 1202 may execute on a plurality of different computing devices, where each computing device may execute one or more of the peer computing nodes 1202. In some embodiments, the decentralized computing platform 1200 may be a permissionless computing platform (e.g., a public computing platform), where a permissionless computing platform allows one or more various entities having access to the program code of the peer node of the permissionless computing platform to participate by using the peer node.

[0164] In some embodiments, the decentralized computing platform 1200 may be private, which may allow a peer computing node of the decentralized computing platform 1200 to authenticate itself to the other computing nodes of the decentralized computing platform 1200 by sending a value based on a private cryptographic key, where the private cryptographic key may be associated with a permissioned tenant of the decentralized computing platform 1200. While Figure 12 shows five peer computing nodes, commercial embodiments may include more computing nodes. For example, the decentralized computing platform 1200 may include more than 10, more than 100, or more than 1000 peer computing nodes. In some embodiments, the decentralized computing platform 1200 may include a plurality of tenants having authentication credentials, wherein a tenant having authentication credentials may allow authorization of its corresponding peer nodes for participation in the decentralized platform 1200. For example, the plurality of tenants may include than 2, more than 12, more than 10, more than 120, more than 100, or more than 1000 tenants. In some embodiments, the peer computing nodes 1202 may be co-located on a single on-premise location (e.g., being executed

on a single computing device or at a single data center). Alternatively, the peer computing nodes 1202 may be geographically distributed. For example, the peer computing nodes 1202 may be executing on devices at different data centers or on devices at different sub-locations of an on-premise location. In some embodiments, distinct subsets of the peer nodes 1202 may have distinct permissions and roles. In some cases, some of the peer nodes 1202 may operate to perform the deserialization operations, graph update operations, or reserialization operations as described in this disclosure.

[0165] Figure 13 shows an example of a computer system by which the present techniques may be implemented in accordance with some embodiments. Various portions of systems and methods described herein, may include or be executed on one or more computer systems similar to computer system 1300. Further, processes (such as those described for Figures 1, 3, or other Figures of this disclosure) and modules described herein may be executed by one or more processing systems similar to that of computer system 1300.

[0166] Computer system 1300 may include one or more processors (e.g., processors 1310a-1310n) coupled to System memory 1320, an input/output I/O device interface 1330, and a network interface 1340 via an input/output (I/O) interface 1350. A processor may include a single processor or a plurality of processors (e.g., distributed processors). A processor may be any suitable processor capable of executing or otherwise performing instructions. A processor may include a central processing unit (CPU) that carries out program instructions to perform the arithmetical, logical, and input/output operations of computer system 1300. A processor may execute code (e.g., processor firmware, a protocol stack, a database management system, an operating system, or a combination thereof) that creates an execution environment for program instructions. A processor may include a programmable processor. A processor may include general or special purpose microprocessors. A processor may include one or more microcontrollers. A processor may receive instructions and data from a memory (e.g., System memory 1320). Computer system 1300 may be a uni-processor system including one processor (e.g., processor 1310a), or a multi-processor system including any number of suitable processors (e.g., 1310a-1310n). Multiple processors may be employed to provide for parallel or sequential execution of one or more portions of the techniques described herein. Processes, such as logic flows, described herein may be performed by one or more programmable processors executing one or more computer programs to perform functions by operating on input data and generating corresponding output. Processes described herein may be performed

by, and apparatus can also be implemented as, special purpose logic circuitry, e.g., an FPGA (field programmable gate array) or an ASIC (application specific integrated circuit). Computer system 1300 may include a plurality of computing devices (e.g., distributed computer systems) to implement various processing functions.

[0167] I/O device interface 1330 may provide an interface for connection of one or more I/O devices 1360 to computer system 1300. I/O devices may include devices that receive input (e.g., from a user) or output information (e.g., to a user). I/O devices 1360 may include, for example, graphical user interface presented on displays (e.g., a cathode ray tube (CRT) or liquid crystal display (LCD) monitor), pointing devices (e.g., a computer mouse or trackball), keyboards, keypads, touchpads, scanning devices, voice recognition devices, gesture recognition devices, printers, audio speakers, microphones, cameras, or the like. I/O devices 1360 may be connected to computer system 1300 through a wired or wireless connection. I/O devices 1360 may be connected to computer system 1300 from a remote location. I/O devices 1360 located on remote computer system, for example, may be connected to computer system 1300 via a network and network interface 1340.

[0168] Network interface 1340 may include a network adapter that provides for connection of computer system 1300 to a network. Network interface 1340 may facilitate data exchange between computer system 1300 and other devices connected to the network. Network interface 1340 may support wired or wireless communication. The network may include an electronic communication network, such as the Internet, a local area network (LAN), a wide area network (WAN), a cellular communications network, or the like.

[0169] System memory 1320 may be configured to store program instructions 1324 or data 1315. Program instructions 1324 may be executable by a processor (e.g., one or more of processors 1310a-1310n) to implement one or more embodiments of the present techniques. Program instructions 1324 may include modules of computer program instructions for implementing one or more techniques described herein with regard to various processing modules. Program instructions may include a computer program (which in certain forms is known as a program, software, software application, script, or code). A computer program may be written in a programming language, including compiled or interpreted languages, or declarative or procedural languages. A computer program may include a unit suitable for use in a computing environment, including as a stand-alone program, a module, a component, or a subroutine. A computer program may or may not correspond to a file in a file system. A

program may be stored in a portion of a file that holds other programs or data (e.g., one or more scripts stored in a markup language document), in a single file dedicated to the program in question, or in multiple coordinated files (e.g., files that store one or more modules, sub programs, or portions of code). A computer program may be deployed to be executed on one or more computer processors located locally at one site or distributed across multiple remote sites and interconnected by a communication network.

[0170] System memory 1320 may include a tangible program carrier having program instructions stored thereon. A tangible program carrier may include a non-transitory, computer-readable storage medium. A non-transitory, computer-readable storage medium may include a machine readable storage device, a machine readable storage substrate, a memory device, or any combination thereof. Non-transitory, computer-readable storage medium may include non-volatile memory (e.g., flash memory, ROM, PROM, EPROM, EEPROM memory), volatile memory (e.g., random access memory (RAM), static random access memory (SRAM), synchronous dynamic RAM (SDRAM)), bulk storage memory (e.g., CD-ROM and/or DVD-ROM, hard-drives), or the like. System memory 1320 may include a non-transitory, computer-readable storage medium that may have program instructions stored thereon that are executable by a computer processor (e.g., one or more of processors 1310a-1310n) to cause the subject matter and the functional operations described herein. A memory (e.g., System memory 1320) may include a single memory device and/or a plurality of memory devices (e.g., distributed memory devices). Instructions or other program code to provide the functionality described herein may be stored on a tangible, non-transitory, computer-readable media. In some cases, the entire set of instructions may be stored concurrently on the media, or in some cases, different parts of the instructions may be stored on the same media at different times.

[0171] I/O interface 1350 may be configured to coordinate I/O traffic between processors 1310a-1310n, System memory 1320, network interface 1340, I/O devices 1360, and/or other peripheral devices. I/O interface 1350 may perform protocol, timing, or other data transformations to convert data signals from one component (e.g., System memory 1320) into a format suitable for use by another component (e.g., processors 1310a-1310n). I/O interface 1350 may include support for devices attached through various types of peripheral buses, such as a variant of the Peripheral Component Interconnect (PCI) bus standard or the Universal Serial Bus (USB) standard.

[0172] Embodiments of the techniques described herein may be implemented using a single instance of computer system 1300 or multiple computer systems 1300 configured to host different portions or instances of embodiments. Multiple computer systems 1300 may provide for parallel or sequential processing/execution of one or more portions of the techniques described herein.

[0173] Those skilled in the art will appreciate that computer system 1300 is merely illustrative and is not intended to limit the scope of the techniques described herein. Computer system 1300 may include any combination of devices or software that may perform or otherwise provide for the performance of the techniques described herein. For example, computer system 1300 may include or be a combination of a cloud-computing system, a data center, a server rack, a server, a virtual server, a desktop computer, a laptop computer, a tablet computer, a server device, a client device, a mobile telephone, a personal digital assistant (PDA), a mobile audio or video player, a game console, a vehicle-mounted computer, or a GPS device, or the like. Computer system 1300 may also be connected to other devices that are not illustrated, or may operate as a stand-alone system. In addition, the functionality provided by the illustrated components may in some embodiments be combined in fewer components or distributed in additional components. Similarly, in some embodiments, the functionality of some of the illustrated components may not be provided or other additional functionality may be available.

[0174] Those skilled in the art will also appreciate that while various items are illustrated as being stored in memory or on storage while being used, these items or portions of them may be transferred between memory and other storage devices for purposes of memory management and data integrity. Alternatively, in other embodiments some or all of the software components may execute in memory on another device and communicate with the illustrated computer system via inter-computer communication. Some or all of the system components or data structures may also be stored (e.g., as instructions or structured data) on a computer-accessible medium or a portable article to be read by an appropriate drive, various examples of which are described in this disclosure. In some embodiments, instructions stored on a computer-accessible medium separate from computer system 1300 may be transmitted to computer system 1300 via transmission media or signals such as electrical, electromagnetic, or digital signals, conveyed via a communication medium such as a network or a wireless link. Various embodiments may further include receiving, sending, or storing instructions or data implemented in accordance with the foregoing description upon a computer-accessible

medium. Accordingly, the present techniques may be practiced with other computer system configurations.

[0175] In some embodiments, additional operations may be performed to determine outcome scores, determine counterparty actions, update a directed graph, or retrieve data from a directed graph. Some embodiments may perform such operations or other operations using methods or systems described in the co-pending PCT application bearing attorney docket number “053173-0515078” titled “GRAPH-MANIPULATION BASED DOMAIN-SPECIFIC EXECUTION ENVIRONMENT,” PCT application bearing attorney docket number “053173-0515079” titled “GRAPH OUTCOME DETERMINATION IN DOMAIN-SPECIFIC EXECUTION ENVIRONMENT,” and PCT application bearing attorney docket number 053173-0515081 titled “GRAPH EVOLUTION AND OUTCOME DETERMINATION FOR GRAPH-DEFINED PROGRAM STATES,” which were filed on 2020-09-08 and assigned to the applicant, “Digital Asset Capital, Inc.,” and which are herein incorporated by reference. Some embodiments may further perform operations such as scoring entities, using hybrid systems to efficiently query data, determine outcome data based on an event with respect to multiple directed graphs. Some embodiments may perform such operations or other operations using methods or systems described in the co-pending US patent application bearing attorney docket number “053173-0515218” titled “EVENT-BASED ENTITY SCORING IN DISTRIBUTED SYSTEMS,” US patent application bearing attorney docket number “053173-0515223” titled “CONFIDENTIAL GOVERNANCE VERIFICATION FOR GRAPH-BASED SYSTEM,” US patent application bearing attorney docket number “053173-0515224” titled “HYBRID DECENTRALIZED COMPUTING ENVIRONMENT FOR GRAPH-BASED EXECUTION ENVIRONMENT,” and US patent application bearing attorney docket number 053173-0515226 titled “MULTIGRAPH VERIFICATION,” which were filed on 2020-09-08, and are assigned to the applicant, “Digital Asset Capital, Inc.,” and which are herein incorporated by reference. Some embodiments may perform operations such as dimensionally reducing graph data, querying a data structure to obtain data associated with a directed graph, perform transfer learning operations, or efficiently notify entities. Some embodiments may perform such operations or other operations using methods or systems described in the co-pending US patent application bearing attorney docket number “053173-0508429” titled “GRAPH-BASED PROGRAM STATE NOTIFICATION,” US patent application bearing attorney docket number “053173-0508434” titled “DIMENSIONAL REDUCTION OF CATEGORIZED DIRECTED GRAPHS,” US patent application bearing attorney docket number “053173-

0508433” titled “QUERYING GRAPH-BASED MODELS,” and US patent application bearing attorney docket number “053173-0508438” titled “ADAPTIVE PARAMETER TRANSFER FOR LEARNING MODELS,” which were filed on 2020-09-08, and are assigned to the applicant, “Digital Asset Capital, Inc.,” and which are herein incorporated by reference.

[0176] As described above, some embodiments may predict an outcome score based on program state data. Some embodiments may provide mechanisms to update the state in-execution program in response to one or more predicted outcomes. Some embodiments may perform operations, such as those described further below, to update or otherwise modify an in-execution smart contract program.

[0177] MODIFICATION OF IN-EXECUTION SMART CONTRACT PROGRAMS

[0178] Modifying the functions of a smart contract program or other symbolic AI program after the program has begun executing may pose challenges in distributed computing environments. For example, a set of criteria of a distributed computing platform and the time required to transfer the data needed to implement a modification in each local memory of a distributed computing platform may significantly increase the cost of operating a smart contract program on a distributed computing platform. While such computational costs may be advantageous by increasing the security of a transaction and making tampering attempts evident, they may inhibit the responsiveness of smart contract programs or other symbolic AI programs operating on a distributed computing platform. For example, some distributed computing platforms may require more than one minute, more than five minutes, or more than ten minutes to verify and distribute an update to a smart contract program. The concurrent execution of more than 10, more than 100, more than 1000, or more than 100,000 operations to amend smart contract programs on a distributed computing platform may cause network or computing performance losses. Such network or computing performance losses may reduce the reliability of smart contract programs or make them less responsive to future events. Operations or related systems that reduce the cost of amending a smart contract program may increase the responsiveness of a smart contract program to future events or verifiable changes.

[0179] Some embodiments may obtain an amendment request and extract one or more values from the amendment request usable to update a smart contract program state or other symbolic AI program state. The values extracted from the amendment request may include a set of conditional statement parameters, a set of entity identifiers, a set of conditional statement

identifiers, or the like. The values extracted from the amendment request may be used to select or update a set of target norm vertices of a smart contract program directed graph. Some embodiments may determine, based on the amendment request, one or more types operations to perform such as updating a set of conditional statements, updating a set of norm vertices, updating a set of entities, or the like, where updating a set may include generating, modifying, or deleting an element of the set.

[0180] Some embodiments may simulate the updating of a directed graph based on one or more conditional statement identifiers encoded in the amendment request and modify an amendment request based on the simulation. For example, some embodiments may simulate a state change caused by an amendment request and then determine an outcome program state based on the simulated program state change. Based on a comparison of an outcome program state value with a threshold (e.g., a threshold provided by a verification agent), some embodiments may prevent the amendment request from changing the program state or modify instructions of the amendment request to satisfy the threshold.

[0181] Some embodiments may use a simulation of a state change caused by an amendment request to determine whether a modified smart contract program state would satisfy a set of criteria of one or more of the entities of a smart contract program, where the directed graph of the smart contract program is stored in a deserialized form on a first computing device. In response to a determination that the set of criteria is satisfied, some embodiments may modify the smart contract program state based on the amendment request and serialize the deserialized directed graph into a serialized array for distribution to other computing devices of a distributed computing platform. By deserializing and reserializing a directed graph based on an amendment request, some embodiments may reduce the memory needs of each computing device of the distributed computing platform. Furthermore, such operations may allow some embodiments may reduce the load on a network used to operate a distributed computing platform. However, while some embodiments may perform one or more of the above-recited operations, these operations are not necessary for some embodiments, and some embodiments may forego such operations to reduce processor use or to enjoy other advantages.

[0182] Some embodiments may select a set of entities based on the set of target norm vertices and determine whether a set of criteria of each selected entity of the set of selected entities is satisfied. Some embodiments may use the modified smart contract program state described above to determine whether the set of criteria are satisfied. Some embodiments may send a

message to each respective entity of the set of selected entities, where the message may indicate that the respective entity is a participant of a target norm vertex. Alternatively, or in addition, some embodiments may include a requirement that a confirmation message authenticating the acceptance of the amendment request from each respective entity of the set of selected entities is obtained.

[0183] Some embodiments may update a smart contract program state in response to determining that the set of criteria are satisfied, such as by updating a set of conditional statements, updating their associated target norm vertices, updating their associated entities, or the like. Some amendment requests may further cause some embodiments to associate a newly-generated directed graph portion or an existing directed graph portion to one or more target norm vertices. Some embodiments may assign priority category values to one or more norm vertices, where the priority category values may be used to determine an order of vertex activation or vertex triggering in the case of a single event triggering multiple norm vertices.

[0184] Figure 14 depicts an example representation of an amendment request modifying a directed graph of a smart contract program, in accordance with some embodiments of the present techniques. The directed graphs shown in Figure 14 are visualized in the form of boxes and edges. However, other representations are possible, and the directed graphs be stored in computer memory in various formats, such as arrays, matrices, data objects, or the like. The dashed box encloses a directed graph 2010 associated with a smart contract program state having a first norm vertex 2012 that may be categorized as an obligation norm. A conditional statement of the first norm vertex 2012 may include a condition that is satisfied if a first entity used to fill the entity field P1 allocates a first quantity equal to 100 units to a second entity used to fill the entity field P2. In some embodiments, the conditional statement of the first norm vertex may encode the first entity having the entity identifier “Ent1” and the second entity having the entity identifier “Ent2” based on the entity identifiers filling their respective entity fields of the conditional statement. Alternatively, or in addition, some embodiments may encode the first and second entity by hardcoding their respective entity identifiers into them. Additionally, some embodiments may determine that a norm vertex is associated with a set of entities encoded in a conditional statement associated with first norm vertex 2012. For example, some embodiments may determine that the first norm vertex 2012 is associated with the first entity and second entity because their corresponding entity identifiers are encoded in a conditional statement that is associated with first norm vertex 2012.

[0185] As indicated by the first directed graph edge 2013, failing to satisfy the conditional statement of the first norm vertex 2012 may result in the activation of the second norm vertex 2014, which may be categorized as a rights norm. The rights norm represented by the second norm vertex 2014 may represent a right of the first entity “Ent1” to allocate 105 units to the second entity “Ent2.” In some embodiments, while not shown, satisfaction of the first norm vertex 2012 or second norm vertex 2014 may cause the activation of a norm vertex having no further child vertices that indicate that the obligation is fulfilled. Alternatively, or in addition, some embodiments may update a label to indicate that the norm vertex satisfied and has no additional child vertices. As indicated by the second directed graph edge 2015, failing to satisfy the conditional statement of the first norm vertex 2012 may also result in the activation of a third norm vertex 2016 categorized a rights norm. The third norm vertex 2016 may represent a right of the second entity “Ent2” to force the allocation of an amount represented by the value “TTL” by the first entity “Ent1,” where “TTL” may be variable number dependent on a remaining amount. For example, the third norm vertex 2016 may represent a right of the second entity “Ent2” to force the allocation of 100 GB of a non-duplicable resource by the first entity “Ent1.”

[0186] The smart contract program state associated with the directed graph 2010 may be modified by the amendment request 2030. The amendment request 2030 may include various parameters. For example, the amendment request 2030 may include a date on which the amendment is to take effect, a conditional statement identifier indicating the conditional statement that the amendment request 2030 is to modify or replace, or a pair of modifying conditional statements. The first modifying conditional statement of the pair of modifying conditional statements may indicate that the first entity “Ent1” is to transfer 10 units to a third entity “Ent3.” A second modifying conditional statement of the pair of modifying conditional statements may indicate that the first entity is to transfer 89 units to the second entity “Ent2.” As discussed further below, some embodiments may extract conditional statement parameters such as the entity identifiers, the quantitative amounts, the date, and the conditional statement identifier.

[0187] The directed graph 2040 may be an outcome directed graph after a state change to the program state of the directed graph 2010, where the state change is caused by the amendment request 2030. The directed graph 2040 includes a first directed graph portion 2050 and a second directed graph portion 2060, where the first directed graph portion 2050 and the second directed

graph portion 2060 are disconnected from each other. The first directed graph portion 2050 includes the first norm vertex 2012, where the entity identifier “Ent3” replaces the entity identifier “Ent2” for the entity field “P2.” In some embodiments, conditional statements of adjacent norm vertices may be affected based on the amendment request, even if the adjacent norm vertex is not directly referenced by an amendment request or does not use a conditional statement identified by an amendment request. For example, the second norm vertex 2014 may be updated such that the quantity “105” is changed to the quantity “94” based on the quantity of the rights norm the set as equal to the sum of the first quantity and an additional five units. Additionally, the third norm vertex 2016 may be updated such that the quantity “TTL” is changed to the quantity “(TTL-10).” Furthermore, some embodiments may include a score change, transfer of scores, or allocation of resources as a result of modifying a smart contract program based on the amendment. For example, an amendment request may cause some embodiments to cause the first entity to directly allocate an hour of processor time to a fourth entity.

[0188] The directed graph 2040 includes the second directed graph portion 2060, where the second directed graph portion may include a fourth norm vertex 2061, a fifth norm vertex 2063, a sixth norm vertex 2065, a seventh norm vertex 2067, an eighth norm vertex 2069, and a ninth norm vertex 2071. A conditional statement of the fourth norm vertex 2061 may include a condition that is satisfied if a first entity used to fill the entity field P1 allocates a first quantity 10 units to a second entity used to fill the entity field P2, where the first entity has the entity identifier “Ent1,” and the second entity has the entity identifier “Ent3.” As indicated by the third directed graph edge 2062, failing to satisfy the conditional statement of the fourth norm vertex 2061 may result in the activation of the fifth norm vertex 2063. The rights norm represented by the fifth norm vertex 2063 may represent a right of the first entity “Ent1” to allocate 11 units to the second entity “Ent3,” which may be interpreted as curing a failure to satisfy the fourth norm vertex 2061. As indicated by the second directed graph edge 2064, failing to satisfy the conditional statement of the fourth norm vertex 2061 may also result in a third rights norm represented by the sixth norm vertex 2065. The third rights norm represented by the sixth norm vertex 2065 may represent a right of the second entity “Ent3” to force the allocation of an amount represented by the variable $TTL/10$ by the first entity “Ent1.” Furthermore, as indicated by the directed graph edge 2072, failing to satisfy the conditional statement of the sixth norm vertex 2065 may result in the activation of the seventh norm vertex

2073. An obligations norm represented by the seventh norm vertex 2073 may represent an obligation of the second entity “Ent2” to allocate eight units to the third entity “Ent3.”

[0189] In some embodiments, the process 2100 of Figure 15, like the other processes and functionality described herein, may be implemented as computer code stored on a tangible, non-transitory, machine-readable medium, such that when instructions of the code are executed by one or more processors, the described functionality may be effectuated. Instructions may be distributed on multiple physical instances of memory, e.g., in different computing devices, or in a single device or a single physical instance of memory (e.g., non-persistent memory or persistent storage), all consistent with use of the singular term “medium.” In some embodiments, the operations may be executed in a different order from that described, some operations may be executed multiple times per instance of the process’s execution, some operations may be omitted, additional operations may be added, some operations may be executed concurrently and other operations may be executed serially, none of which is to suggest that any other feature described herein is not also amenable to variation.

[0190] Figure 15 is a flowchart of a process to modify a program state based on an amendment request, in accordance with some embodiments of the present techniques. In some embodiments, the process 2100 may include obtaining a smart contract program or other symbolic AI program encoding an associated directed graph, as indicated by block 2104. Obtaining a smart contract program state encoding an associated directed graph may include loading data from, copying a version of, or otherwise accessing a smart contract program or other symbolic AI program. In some embodiments, the smart contract program state may be active and in the process of being executed by a computing system and include data stored in persistent memory or non-persistent memory. For example, some embodiments may obtain the smart contract program state by executing the smart contract program state and retrieving a program state encoding the associated directed graph of the smart contract program state. Alternatively, or in addition, the smart contract program state may be archived or otherwise stored in a persistent memory of a computing system. The smart contract program state may be obtained from a smart contract program state, a predicted future state, a simulated state, or the like.

[0191] In some embodiments, as described above, the smart contract program state may encode a directed graph in the form of a serialized array of norm vertices and its corresponding set of graph edges. For example, the smart contract program state may include a serialized array of

norm vertices “[1, 4, 7],” where each number of the numbers indicates a norm vertex, and a corresponding set of directed graph edges “[1,4], [4,7],” where each subarray indicates a directed graph edge. Alternatively, or in addition, the smart contract program state may include a plurality of serialized arrays of norm vertices and their corresponding edges. For example, the smart contract program state may include a first serialized array of norm vertices “[1, 4, 7]” associated with a corresponding serialized array of directed graph edges [[1, 7], [7,4]]. The smart contract program state may include a second serialized array of norm vertices “[2, 5, 7, 4, 41]” and their corresponding directed graph edges [[2,5], [7,5], [5,4], [4,41]]. As discussed further below, storing portions of a graph in separate serialized arrays of norm vertices may contribute to increasing memory efficiency or update efficiency when executing a smart contract program state.

[0192] In some embodiments, the directed graph may be disconnected, having two or more unconnected portions. For example, the smart contract program state may include a first serialized array of norm vertices “[1, 4, 7]” associated with a corresponding serialized array of directed graph edges [[1, 7],[7,4]]. The smart contract program state may also include a second serialized array of norm vertices “[2, 5, 7, 4, 41]” and their corresponding directed graph edges [[2,5], [4,5], [4,41]], where the graph portion formed by the first serialized array of norm vertices is disconnected from the graph portion formed by the second serialized array of norm vertices.

[0193] In some embodiments, the process 2100 may include obtaining an amendment request encoding a set of conditional statement parameters, as indicated by block 2108. The amendment request may be obtained at an API of a smart contract program, API of application in communication with the smart contract program, an API of a distributed computing platform, an API of a computing system executing the smart contract program, or the like. Some embodiments may obtain the amendment request from data entered by a user at a graphical user interface. Alternatively, or in addition, some embodiments may obtain an amendment request that was machine-generated or updated using a machine-learning system. The set of conditional statement parameters may include various types of information, such as a date of enforcement, a quantitative value, a categorical value, an entity identifier or other identifier, or the like. For example, an amendment request corresponding to the natural language instructions “entity 12591xc3 is obligated to allocate 300 GB to entity 27831t6” may include the quantity parameter “300” and the entity identifiers “12591xc3” and “27831t6.”

[0194] The set of conditional statement parameters may include some or all of a conditional statement written in a computer-readable programming language. For example, the set of conditional statement parameters may include the conditional statement “if (ENTITY == “entity1x1”): SENDRESOURCE(100, “entity1x1”, “entity1x2”)”. Alternatively, or in addition, the set of conditional statement parameters may include a set of parameter values used to fill a field a conditional statement. For example, the set of conditional statement parameters may include a conditional statement identifier “cond_state10105421” with the associated parameters [“entity1x1”; “entity1x2”; 194]. The conditional statement identifier “cond_state10105421” may identify a conditional statement that may be represented in the form “if RCVD_FUNCTION(ARG1, ARG2, ARG3).” The function “RCVD_FUNCTION” may be a function that accepts the three parameters ARG1, ARG2, and ARG3 and returns the boolean “true” if ARG1 and ARG2 are entities and ARG1 has received the amount ARG3 from the entity identified by ARG2. The associated parameters may indicate that “entity1x1” is to be used in place of ARG1, that “entity1x2” is to be used in place of ARG2, and that “194” is to be used in place of ARG3.

[0195] In some embodiments, the amendment request may include specific instructions to generate a new directed graph portion. For example, the amendment request may include the instructions “Generate_Vertex(1525, 215, “satisfied,” 15216),” which may cause a smart contract program to generate a norm vertex identified as “1525.” A graph edge may associate the newly-generated norm vertex to the norm vertex identified as “215” and be activated based on the satisfaction of the norm vertex identified as “215.” The final parameter of the function “Generate_Vertex” may indicate that the newly-generated norm vertex may be associated with a conditional statement identified as “15216.” Alternatively, or in addition, some embodiments may determine that an amendment request includes instructions or values satisfiable by the generation of a new directed graph portion and, in response, generate the new graph portion. For example, some embodiments may determine that the instructions include “if vert_failed(vert[1653]): cond_state[15216],” where the instructions may cause a smart contract program to determine that a norm vertex associated with the conditional statement “15216” is in existence.

[0196] In some embodiments, the process 2100 may include determining a set of target norm vertices of the smart contract program state or other symbolic AI system based on the amendment request, as indicated by block 2116. In some embodiments, the amendment request

may include direct references to one or more norm vertices of a directed graph of a smart contract program. For example, the event request may include a reference to the norm vertex “OP111-1,” where the value “OP111-1” is a norm vertex identifier for a norm vertex in a smart contract graph. Alternatively, the amendment request may include a series of the old parameters to identify one or more smart contract graph norm vertices. For example, some embodiments may receive an amendment request that includes a set of norm vertex-identifying parameters, such as a parameter that specifies a transaction date, a set of affected entities, or a transaction amount. Some embodiments may extract the conditional statement parameters or other norm vertex-identifying parameter from the amendment request, where a norm vertex-identifying parameter may be any value that can be used to identify a set of norm vertices. For example, a norm vertex-identifying parameter may include a norm vertex identifier, a category type specific to a set of norm vertices, a time or time interval that can be used to isolate a set of norm vertices having fulfillment deadlines due after the time or within the time interval, or the like. Each parameter of the set of norm vertex-identifying parameters may then be used by a computing system to determine one or more norm vertices or conditional statements. For example, an amendment request specifying a modification for all transactions associated with a conditional statement identifier after a specified date may result in the extraction of the conditional statement identifier and the specified date as norm vertex-identifying parameters.

[0197] Some embodiments may determine a set of active norm vertices of a directed graph of a smart contract program or other symbolic AI program. Some embodiments may search through the set of active norm vertices instead of searching through all norm vertices of the directed graph. In some embodiments, an active norm vertex may be a norm vertex having an associated conditional statement that may be triggered, whereupon the triggering of the associated conditional statement causes the activation of another norm vertex to take place or may otherwise cause a state change to occur (e.g., update a label indicating that an obligation norm is satisfied). For example, some embodiments may determine that a first, second, and third norm vertex are each active norm vertices of a set of active norm vertices. Some embodiments may then search through the set of active norm vertices that satisfy a set of norm vertex-identifying parameters to determine a target norm vertex. For example, an amendment request may include norm vertex-identifying parameters specifying a change to terms associated with an allocation of 100 units by a first entity for a second entity. Some embodiments may extract the first entity identifier, the second entity identifier, and the allocation of 100 units to form a set of norm vertex-identifying parameters and search through

a set of active norm vertices to find which norm vertices satisfy the set norm vertex-identifying parameters. This is not to suggest that all embodiments may restrict the search to a set of active norm vertices, and some embodiments may search through other norm vertices of a directed graph or all norm vertices of a directed graph for their own benefit(s), such as for ensuring a completeness of the search or for providing data for a study of historical performance.

[0198] Some embodiments may obtain amendment requests that may modify one or more conditional statements and search for a list of affected conditional statements, which may then be used to select a target norm vertex. For example, some embodiments may include an amendment request that includes the computer-interpretable code ‘if entity[2123].sending() == true, Replace(2123, 254121),’ which may be converted from the natural language statement, “if the entity having the entity identifier ‘2123’ is sending a resource, replace it with the entity having the entity identifier ‘254121.’” Some embodiments may then search through a set of conditional statements to determine which of the set of conditional statements are affected by the amendment request based on which entities are part of a condition of the conditional statement. For example, a first conditional statement may include the condition ‘if entity[2123].sending() == true: message(entity[254121], “SENT”).’ Some embodiments may determine that the first conditional statement is an affected conditional statement and then determine a set of target norm vertices that includes, uses, or is otherwise associated with the first conditional statement.

[0199] In some embodiments, the process 2100 may include determining a set of selected entities based on the set of target vertices, as indicated by block 2120. As discussed above, the set of target vertices may be associated with a set of entities via the set of conditional statements encoding one or more of the set of entities. This set of entities may be used as a set of selected entities, where the set of selected entities may indicate entities that are directly affected by the amendment request. By determining which entities are directly affected by an amendment request, some embodiments may more effectively message the set of affected entities without overloading a messaging system by notifying only entities that are likely to have an interest in the amendment request. For example, a smart contract program may include a list of 100 entities, amongst which 25 are determined as being affected by an amendment request based on the entities associated with vertices affected by the amendment request. These 25 entities may then be sent a first message indicating that they are affected by the amendment request, while the other 75 entities are not sent any messages or sent a message different from the first

message. Furthermore, some embodiments may require that an amendment request be confirmed by the set of selected entities instead of requiring that confirmation be provided by all entities of a smart contract program. By reducing the number of required confirmation messages, some embodiments may significantly reduce the time needed to amend a smart contract program having more than ten entities, more than twenty entities, more than two hundred entities, or the like by requiring a confirmation message from only a subset of the entities of the smart contract program.

[0200] In some embodiments, the process 2100 may include simulating a modification of the smart contract program state or other symbolic AI program state based on the amendment request, as indicated by block 2124. Some embodiments may simulate the modification of an application program state based on the amendment request by generating a version the smart contract program state and changing the version based on the amendment request without requiring the distribution of the changed version to other computing devices. Simulating a modification of the smart contract program state may include generating a new graph structure that is different from the graph structure of an unmodified smart contract program state with respect to the number of norm vertices, the number of edges, or the set of logical categories associated with each of the norm vertices. For example, a first graph structure of unmodified smart contract program state may include three active obligation norm vertices. After simulating a modification based on the amendment request, the graph structure of the modified smart contract program state may be changed such that the logical category of one norm vertex is changed from being an obligation norm to a prohibition norm. This may cause the simulated modification of the graph structure to include two active obligation norm vertices and one active prohibition norm vertex. Simulating a modification of the smart contract program state may include generating a version of a set of conditional statements and updating the version of the set of conditional statements based on the set of conditional statement parameters.

[0201] As further discussed in this disclosure, a simulated modification based on an amendment request may be used to determine whether an outcome state caused by the amendment request satisfies a set of requirements of the smart contract program. For example, as further illustrated below, an amendment request to a smart contract program state may cause a first entity to be obligated to allocate five terabytes of memory to a second entity. Some embodiments may simulate an implementation of the amendment request and determine that the simulated outcome state violates a third entity's requirement. For example, the third entity's

requirement may be that the second entity is prohibited from reserving memory allocated by the first entity, where third entity's requirement may be implemented as a prohibition norm or may be implemented in another form (e.g., program-wide rule) encoded in the smart contract program. In response to this simulated violation, some embodiments may prevent the amendment request from modifying the smart contract program state or may modify the instructions of the amendment request (e.g., by changing the source entity for the allocated memory).

[0202] Some embodiments may simulate the modification of an application program state without simulating every modification caused by the amendment request. For example, a first amendment request may cause the modification of a program state to change the conditional statement and logical category of a first norm vertex. The change may cause an obligation norm of a first entity to allocate a first amount to a rights norm triggerable by a second entity to request a second amount from the first entity after an interval of time encoded in the amendment request. The computing system may simulate the modification of the application program state by changing the logical category associated with the first norm vertex and any associated graph structure changes without changing the first amount to the second amount or include the encoded interval of time. Based on a determination that a simulated modified graph structure is not identical to or otherwise different from a graph structure before a smart contract modification, some embodiments may generate a new norm vertex for a directed graph of a smart contract program.

[0203] In some embodiments, a representation of the simulated modification of the application program state based on the amendment request may be sent to one or more verification agents. A verification agent may include a third-party entity, an automated testing system, or another system. For example, some embodiments may simulate modification of a first smart contract program and send the graph structure, their corresponding conditional statements, or other related data to a verification agent for display a graphical user interface. Alternatively, some embodiments may simulate the modification of the application program state without any messages. The verification agent may send a message via an API, where the message may confirm that the simulated modification is acceptable, reject the simulated modification or may include a second simulated modification of the smart contract program state. As further described below, some embodiments may require a message confirming the authorization of the simulated modification before proceeding to use the simulated modifications for additional

analysis. Alternatively, some embodiments may use the simulated modifications for further analysis without requiring a confirmation message.

[0204] Some embodiments may simulate the occurrence of a set of simulated events or a sequence of simulated events for a simulated modified smart contract program. For example, some embodiments may simulate the occurrence of a sequence of simulated events indicated to have occurred based on an associated set of occurrence times, where the associated set of occurrence times indicate the occurrence of a first simulated event of the sequence of simulated events on a first day and the occurrence of a second simulated event of the sequence of simulated events on a second day. Some embodiments may simulate the occurrence of a plurality of simulated events or a plurality of sequences of simulated events to determine a set of outcome scores corresponding to a set of simulated modified smart contract program states. For example, some embodiments may set the average amount of computing memory allocated over one month as an outcome score and determine a set of outcome program states across three equally-likely sequences of events based on five different simulated program states.

[0205] Some embodiments may determine a respective set of outcome program states for each respective simulated program state of the five different simulated program states, where each respective simulated program state is a result of modifying the program state based on a respective amendment request of the plurality of amendment requests. Some embodiments may then determine a respective set of outcome scores for each respective set of outcome program states. Some embodiments may then select an amendment request from the plurality of amendment requests based on the set of outcome scores, where the set of outcome scores may include each of the respective set of outcome scores. For example, some embodiments may select an amendment request based on which results in a maximum outcome score of the set of outcome scores. Some embodiments may perform one or more of the simulation operations described above using a single computing device or subset of computing devices of a distributed computing platform instead of using each computing device of the distributed computing platform to simulate a modification. By performing the simulation on a one computing device or a small number computing devices, some embodiments may reduce the computational load on the distributed computing platform and reduce the network traffic used to operate the distributed computing platform.

[0206] In some embodiments, the process 2100 may include determining whether the amendment request satisfies the set of criteria of the set of selected entities, as indicated by

block 2128. Determining whether the amendment request satisfy the set of criteria of the set of selected entities may include determining whether the amendment request satisfies a set of governing conditional statements associated with the smart contract program. For example, a governing set of conditional statements may prohibit transactions with entities of a first entity type, and the amendment request may change a participating entity to an entity of the first entity type. In response, some embodiments may determine that the amendment request does not satisfy the set of criteria of the set of selected entities. In some embodiments, each entity of the set of selected entities may have a different set of criteria that must be satisfied in order to provide a respective confirmation message.

[0207] In some embodiments, the set of confirmation messages may include one or more authentication frameworks to authenticate a confirmation message. For example, the set of confirmation messages may include a set of passkey values. For example, each respective message of a set of confirmation messages may include a respective passkey value of the set of passkey values, where each message of the set of confirmation messages may be associated with a respective entity of the set of selected entities. For example, an entity may send a confirmation message, including a human-entered or machine-provided passkey value. Some embodiments may compare a respective passkey value with a respective stored passkey value to determine whether the respective passkey value matches with the respective stored passkey value. In some embodiments, some embodiments, the passkey value may be encrypted and compared to a set of encrypted passkey values to determine a match. Alternatively, or in addition, a passkey value may be decrypted and compared to a set of decrypted passkey values to determine a match. Based on the entity-sent passkey value matching a stored passkey value in a set of stored passkey value, some embodiments determine that a criterion associated with one or more of the set of selected entities is satisfied. While the above describes an implementation of one type of authentication framework, some embodiments may use one of various other types of authentication frameworks when sending confirmation messages or other messages. For example, some embodiments may implement a Public Key Infrastructure (PKI) framework, such as that described in “Introduction to public key technology and the federal PKI infrastructure” (Kuhn, D. Richard, et al., National Inst of Standards and Technology Gaithersburg MD, 2001), which is hereby incorporated by reference. Some embodiments may use various data transport protocols when implementing the authentication framework, such as secure socket layer (SSL) or transport layer security (TLS).

[0208] Various types of criteria may be used to determine whether to modify a smart contract program state based on the amendment request. In some embodiments, determining whether a criterion is satisfied may include determining whether an entity of a smart contract program is one of a set of prohibited entities or one of a set of prohibited entity types. For example, the entity “entity1” may have the entity type “x1x1,” and determining whether the set of criteria is satisfied may include determining that entities of the entity type “x1x1” are entities of a prohibited entity type. In response, some embodiments may determine that the set of criteria is not satisfied by the entity “entity1.”

[0209] In some embodiments, determining whether the set of criteria is satisfied may include determining whether a non-duplicable asset is concurrently transferred or allocated to different entities based on a single event or sequence of events. A non-duplicable asset may include an amount of computing time on a specific computing resource during a specific time interval. For example, a non-duplicable asset may include an allocated utilization time between 04:00 and 06:00 on a specified set of processor cores. The transfer or allocation of a non-duplicable resource to multiple resources may be detected as a conflict, and some embodiments may include a verification mechanism to prevent the conflict. For example, some embodiments may determine a simulated contract program state based on a simulation of the modification of a smart contract program state based on an amendment request. Some embodiments may then simulate how the simulated smart contract program state responds to a sequence of events and determine that a first entity is allocating control of a specific computing resource to a second entity based on an event, and that the first entity will also be caused to allocate control of the specific computing resource to a third entity based on the event. Some embodiments may determine this concurrent allocation and, in response, prevent the amend request from being implemented, send a message indicating that the amendment may cause a conflict, or the like.

[0210] In some embodiments, the set of criteria may include determining that an entity, set of entities, or the entity type is required for transactions of a specified transaction type or all transaction types. For example, some embodiments may include a criterion that each entity of a smart contract program is indicated as verified based on a verification field being populated with the value “verified.” As another example, the entity “entity2” may have the entity type “x2x2” and a possible criterion may be that all entities of a smart contract program state be of the entity type “x2x2.” In response, some embodiments may determine that the entity “entity2” satisfies the criterion.

[0211] Some embodiments may perform one or more of the determination operations described above using a subset of computing devices of a distributed computing platform instead of using each computing device of the distributed computing platform to determine whether a set of criteria are satisfied. For example, some embodiments may store a version of each of a set of criteria of a set of entities at a storage memory. Some embodiments may then determine whether an amendment request satisfies the set of entities using a computing device that includes or is otherwise capable of accessing the storage memory. After determining that the set of criteria is satisfied, some embodiments may send the amendment request or parameters stored in the amendment request to other computing devices of the distributed computing platform. By restricting the determination operation to one computing device or a small number of computing devices, some embodiments may reduce the overall computational load on the distributed computing platform and reduce the network traffic used to operate the distributed computing platform. In some embodiments, if the amendment request satisfies the set of criteria of the set of selected entities, operations of the process 2100 may proceed to block 2140. Otherwise, operations of the process 2100 may proceed to block 2132.

[0212] In some embodiments, the process 2100 may include updating an amendment request, as indicated by block 2132. In some embodiments, the amendment request may be updated in response to a failure to satisfy the set of criteria of the selected entities. In some embodiments, the amendment request being generated may be generated in a parameter space that allows the amendment request to have multiple possibilities. In some embodiments, some of these possibilities of the amendment request may satisfy the set of criteria of the selected entities while other possibilities of the amendment request may not satisfy the set of criteria of the selected entities. For example, a first version of the request may change a first quantity from the value “100” to the value “300” and a first criterion of one of the selected entities may require that the value of the first quantity be less than the value “200.” In response, after determining that the amendment request failed the first criterion, some embodiments may update the amendment request to change the first quantity from the value “300” to the value “150” using one or more of various types of optimization methods or machine-learning methods.

[0213] Some embodiments may send a message to one or more of the set of selected entities in response to failing to satisfy the set of criteria of the selected entities. For example, some embodiments may send a message to all of the set of selected entities in response to failing the

set of criteria. Some embodiments may send a message indicating that the set of criteria has been failed without updating the amendment request. Additionally, or alternatively, some embodiments may send a message to an entity or other agent that sent the amendment request indicating that the amendment request on a program state resulting from the amendment request has failed the set of criteria.

[0214] In some embodiments, the process 2100 may update a set of conditional statements based on the set of conditional statement parameters, as indicated by block 2140. In some embodiments, determining the updated set of conditional statements may include replacing one or more conditional statements with a new conditional statement determined from the set of conditional statement parameters. For example, a norm vertex may be associated with a first conditional statement, where the first conditional statement is indicated to be replaced by a second conditional statement encoded in an amendment request. In some embodiments, a conditional statement that is to be replaced or otherwise unused may be marked as deprecated. For example, some embodiments may change a usage indicator associated with a conditional statement to indicate that the conditional statement is deprecated based on an amendment request indicating that the conditional statement should be removed from use in the smart contract program. Alternatively, instead of deprecating a conditional statement stored in a set of conditional statements, some embodiments may determine delete the conditional statements.

[0215] In some embodiments, updating the set of conditional statements may include filling out, replacing, or otherwise using one or more conditional statement parameters to populate fields of the set of conditional statements. For example, the computing system may obtain a first plurality of entity identifiers, and a user may use the entity identifiers fill out the function that uses the first plurality of entity identifiers. By filling out a field of an existing conditional statement instead of replacing the condition statement, some embodiments may increase the speed by which a smart contract program state may be distributed on a distributed computing platform. However, while the above describes filling, replacing, or otherwise using one or more conditional statement parameters to populate fields of the set of conditional statements, some embodiments may forego such operations and update the set of conditional statements using other methods.

[0216] In some embodiments, updating the set of conditional statements may include adding one or more conditional statements to the set of conditional statements. Some embodiments may determine an updated set of conditional statements indexed by conditional statement

identifiers by adding a new conditional statement that may have its own associated conditional statement identifier. For example, an amendment request may include an instruction to add a specific conditional statement to a set of conditional statements of a smart contract program. In response to obtaining the amendment request, some embodiments may include instructions to add a conditional statement identifier “x1x1” to a conditional statement identifier index.

[0217] In some embodiments, updating the set of conditional statements may include determining whether a conditional statement (or an associated norm vertex) was triggered by a past event. An outcome of the conditional statement may include a transaction between a first entity and a second entity. For example, after determining that a past event had triggered a norm vertex based on an amendment request modifying the norm vertex or an associated conditional statement, some embodiments may determine a first score value associated with a transaction caused by the outcome. Some embodiments may determine a second score value encoded in the amendment request and determine a difference between the first score value and the second score value. Some embodiments may then initiate a transaction between the first entity and the second entity based on the score difference. By using a score differences to account for differences between amendment requests and past events, some embodiments may include mechanisms to retroactively apply an amendment request.

[0218] In some embodiments, updating the set of conditional statements, set of target vertices, or other values of a program state operating on a distributed computing platform may include sending a set of values to each computing device of the distributed computing platform. In some embodiments, to determine the validity of a distributed value, the smart contract program may use one or more consensus algorithms. For example, to reach a consensus on the validity of a set of conditional statements, some embodiments may use a consensus Paxos algorithm, a Raft algorithm, HotStuff, or the like. Furthermore, some embodiments may centralize one or more of the operations described above at a single computing device or a subset of computing devices and then send a processed set of values to other computing devices of the distributed computing platform. For example, some embodiments may simulate modifications of a smart contract program using a first computing device and determine an updated amendment request based on the simulated modifications before sending the updated amendment request to other computing devices. Additionally, some embodiments may store or otherwise have access to a set of criteria of each entity of a set of selected entities to determine whether the set of criteria is satisfied. By centralizing operations at a single computing device or a subset of computing

devices, some embodiments may reduce the overall computational cost of amending a smart contract program or other symbolic AI program.

[0219] In some embodiments, the process 2100 may include updating the set of target norm vertices or values associated with the set of target norm vertices based on the amendment request, as indicated by block 2144. Updating the set of target norm vertices may include updating a field of the target norm vertex for a conditional statement identifier of the target norm vertex with a new conditional statement identifier. Updating the field may associate the target norm vertex with the new conditional statement having the new conditional statement identifier. For example, a first target norm vertex may have or otherwise be associated with the conditional statement identifier “4457” and updated to have instead or otherwise be associated with the conditional statement identifier “9941.” After the update, a first event satisfying the conditional statement having the identifier “9941” may trigger the first target norm vertex, whereas a second event satisfying the conditional statement having the identifier “4457” does not trigger first target norm vertex.

[0220] Some embodiments may update the smart contract program state by deserializing a serialized array representing a directed graph stored in persistent memory. The deserialized directed graph may be stored in non-persistent memory for fast processing or operations. Some embodiments may then add, modify, or remove a norm vertex or edge of the directed graph stored in the non-persistent memory and then reserialize the directed graph. For example, some embodiments may include a first serialized array “[1 3 5]” having an associated serialized array of edges “[1,3], [3,5]” stored in a persistent memory of a computing system. Some embodiments may deserialize the serialized array into an adjacency matrix form stored in the non-persistent memory and add a norm vertex having a norm vertex identifier “7” and edge directing from the norm vertex “5” to the norm vertex “7.” Some embodiments may reserialize the deserialized directed graph to determine the updated serialized array “[1 3 5 7]” having an associated serialized array of edges “[1,3], [3,5], [5,7].” The deserialization and reserialization of directed graph data may result in increased storage memory use efficiency, or network performance efficiency (e.g., in the case of the smart contract being implemented on a distributed computing platform). However, while the above suggests some embodiments may implement a deserialization/reserialization operation, such operations are not necessary. Some embodiments may forego such operations and use other methods to benefit from

increased computational performance efficiencies, network performance efficiencies, or the like.

[0221] In some embodiments, the process 2100 may include updating a directed graph of a smart contract program with an additional set of norm vertices, as indicated by block 2148. Some embodiments may associate an additional set of norm vertices with a directed graph of the smart contract program in response to an amendment request causing the creation of the additional set of norm vertices. For example, some embodiments may determine that an amendment request includes instructions or values that causes the generation of a new norm vertex in the directed graph based on a determination that a set of conditional statements parameters in the amendment request is unrelated to an existing norm vertex of the directed graph. Some embodiments may then generate a norm vertex by creating a new conditional statement and associating the new conditional statement with a newly created norm vertex having associated directed graph edge. In some embodiments, the amendment request may cause the creation of a plurality of norm vertices and their corresponding directed graph edges.

[0222] In some embodiments, the generated set of norm vertices may be serialized into a serialized array in a persistent memory. In some embodiments, the generated set of norm vertices may be deserialized into a deserialized directed graph stored on a non-persistent memory, where the deserialized directed graph may include an adjacency matrix or adjacency list. Some embodiments may generate an edge that connects one or more of the set of norm vertices with one or more norm vertices of the set of target norm vertices determined above. In some embodiments, one or more of the set of generated norm vertices may have an associated priority category value used to determine a sequence by which different norm vertices are triggered. In some embodiments, the implementation of an execution sequence based on a set of associated priority category values may be used to reduce the risk of contradictions or logical errors.

[0223] Some embodiments may associate a target norm vertex with another portion of an existing directed graph based on an amendment request. The other portion of the existing directed graph may be part of the smart contract program state. Alternatively, the other portion of the existing directed graph may be part of a different smart contract program state. For example, an amendment request for a first smart contract program may include a program identifier of a second smart contract program and a vertex identifier of a first norm vertex of

the second smart contract program. Some embodiments may associate a target norm vertex with the first norm vertex based on the program identifier and the vertex identifier.

[0224] Various formats may be used to indicate this cross-program relationship or the order by which active norm vertices are triggered across different smart contract programs. For example, some embodiments may add a new directed graph edge to a set of directed graph edges, where the new directed graph edge may point from the target norm vertex to the norm vertex of the other smart contract program. Some embodiments may account for potential confusion by having the directed graph vertex point to a dummy norm vertex, where the dummy norm vertex includes values identifying the norm vertex of the other smart contract program. Some embodiments may assign a priority category value to each respective norm vertex or respective smart contract program and refer to the respective priority category values to determine an order by which norm vertices are triggered in response to an event that triggers multiple norm vertices.

[0225] In some embodiments, the process 2100 may include storing the smart contract program state or other symbolic AI program state in a persistent memory, as indicated by block 2152. After updating the program state of a smart contract program or other symbolic AI model, some embodiments may then store the smart contract program in a persistent storage memory of one or more computing devices. In addition, some embodiments may store the amendment request, rejected amendment requests, or other data related to an amendment request in a same storage memory.

[0226] As described above, some embodiments may modify or otherwise update an in-execution smart contract program. Additionally, some embodiments may perform verification operations based on multiple vertices of one or more directed graphs, where each respective directed graph may encode its own respective codified agreement. Some embodiments may perform operations, such as those described further below, to decrease computational resource requirements when determining the effects that one or more events may have on a set of vertices.

[0227] MULTIGRAPH VERIFICATION

[0228] Testing the integrity of applications encoding a set of graph structures face unique challenges. Data may grow exponentially when used to represent a program that includes a basic set of conditional statements capable of activating one another. Such permutations may

grow into more than thousands, more than hundreds of thousands, or more than millions of possible states, which increases the risk that one or more of those possible states may incur the risk of a failure or unintended consequence. While some unit testing operations may account for such failures by using a known set of inputs and expected outcomes, such operations become more difficult for a program that may be executed concurrently with other programs, each of which may interfere with each other. The concurrently-executing programs may include over ten, over a hundred, or over a thousand different active conditional statements, which of which may be triggered an entity action. Such operations may increase in difficulty in cases where one or more conditions are designed to be hidden from unpermitted entities, which result from instructions to obfuscate one or more conditional statements encoded by a concurrently-executed program.

[0229] Some embodiments may provide a method of testing whether a program may satisfy or fail the conditions of other programs by determining a set of integrated test conditions from multiple conditional statements of a program having multiple participating entities. Some embodiments may use properties of a directed graph encoded in the data of a smart contract program or other multi-entity program to determine the set of integrated test conditions. The conditional statements used to generate an integrated test condition may be collected based on a shared category or resource type, where the amounts encoded in the conditional statements may be used to determine a threshold of the integrated test condition.

[0230] Some embodiments may test an integrated test condition of a first program with a simulated event occurring in the context of a second program and its corresponding second program state, where the second program may be selected based on a shared associated entity with the first program. The values of the simulated event may be determined based on resource amounts, resource types, or other parameters encoded in a conditional statement of the second program. The integrated test condition may be tested by either directly simulating the occurrence of the same event in the context of the integrated test condition or by testing an outcome caused by the event. Some embodiments may then store the test result or send a message to one or more entities that is predicted to be affected based on the test result.

[0231] By using different conditional statements associated with different vertices of a directed graph of a smart contract, the number of tests that other programs must use may be dramatically reduced. For example, some embodiments may use two or more conditional statements of a smart contract program (or other program) to generate a single integrated test condition, which

can reduce the number of verification checks by a factor of ten or more. In addition, some embodiments may use operations described in this disclosure to determine whether a type of conditional statement would be triggered by an event without revealing what specific conditional statements or entities would be affected by the event. Such operations may reduce the risk of non-permitted entities from obtaining information other smart contract programs or otherwise protect the privacy of entities participating in a set of smart contract programs or other symbolic AI programs. It should be understood that, while some embodiments may be described as gaining a specific benefit or performing a specific operation described in this disclosure, not all embodiments described in this disclosure must perform that specific operation or provide that specific benefit, and may perform a set of operations that do not include that specific operation for its own corresponding set of benefits.

[0232] Figure 16 shows a conceptual diagram representing program states and integrated test conditions determined from program states, in accordance with one or more embodiments. As shown in the conceptual diagram 3000, a box 3002 includes a graph having five vertices, where the graph in the box 3002 may be stored in program state as a set of vertex identifiers, associated edges, or the like in tables, arrays, other data structures, or the like. The box 3002 includes a first set of vertices enclosed by dotted line box 3010, where the dotted line box 3010 encloses the obligation norm vertices 3011 - 3013. The box 3002 includes a second set of vertices enclosed by the dotted line box 3020, where the dotted line box 3010 encloses the prohibition norm vertices 3021-3022.

[0233] Some embodiments may generate a first integrated test condition 3031 or second integrated test condition 3032 from parameters of the conditional statements of the norm vertices 3011-3013 or the norm vertices 3021-3022. A conditional statement parameter may include a threshold, a resource type, a resource amount, a category, or the like. Some embodiments may generate the first integrated test condition 3031 using parameters of the conditional statements associated with the set of norm vertices enclosed by the dotted line box 3010. For example, the first obligation norm vertex 3011 may be associated with a first conditional statement specifying that a minimum of 37 units of a resource type “X” must be transferred to satisfy the first conditional statement. Similarly, the second obligation norm vertex 3012 may be associated with a second conditional statement specifying that a total of 59 units of a resource type “X” must be transferred to satisfy the second conditional statement. Additionally, the third obligation norm vertex 3013 may be associated with a third conditional

statement specifying that a total of 45 units of a resource type “X” must be transferred to satisfy the third conditional statement. Some embodiments may collect the thresholds of each of a first set of conditional statements into the set of respective thresholds [“37”, “59”, “45”] based on the conditional statements (or their associated vertices) sharing a category label “obligation,” sharing a resource type “X,” sharing a “greater than or equal to” condition, or the like. Some embodiments may include instructions to select the maximum value “59” of the set [“37”, “59”, “45”] based on each of the conditional statements sharing a “greater than or equal to” condition. Some embodiments may generate the first integrated test condition 3031 by combining the shared relationship condition “greater than or equal to” with the maximum value “59.” Some embodiments may also set the outcome fulfillment state (sometimes called “satisfaction state”) of the integrated test condition based on the shared category label “obligation,” where the label “obligation” may indicate that satisfying the integrated test condition may cause some embodiments to update_the integrated test condition as “fulfilled” (or sometimes called “satisfied”).

[0234] Some embodiments may generate a second integrated test condition 3032 using parameters of the conditional statements associated with the set of norm vertices enclosed by the dotted line box 3020. The first prohibition vertex 3021 may be associated with a fourth conditional statement specifying that the number of units of resource type “Y” must be less than or equal to 157 units to avoid triggering the fourth conditional statement. Similarly, the second prohibition norm vertex 3022 may be associated with a fifth conditional statement specifying that the number of units of resource type “Y” must be less than or equal to 157 units to avoid triggering the fourth conditional statement. In some embodiments, triggering a conditional statement associated with a prohibition norm vertex may cause some embodiments to indicate that the prohibition norm vertex has been failed. Some embodiments may collect the thresholds of each of a second set of conditional statements into the set of respective thresholds [“157”, “327”] based on the conditional statements (or their associated vertices) sharing a category label “prohibition,” sharing a resource type “Y,” sharing a “less than or equal to” condition, or the like. Some embodiments may include instructions to select the minimum value “157” of the set [“157”, “327”] based on each of the conditional statements sharing a “less than or equal to” condition. Some embodiments may then use this minimum value to generate the second integrated test condition 3032 by combining the shared relationship condition “less than or equal to” with the minimum value “157” and setting the outcome based on the shared category “prohibition.”

[0235] The graph vertex enclosed by the box 3040 may be associated with a first entity associated with an entity profile 3090. The entity of the entity profile 3090 may also be associated with a smart contract associated with a directed graph represented by the diagram in the box 3040. As discussed in this disclosure, a set of simulated events may be generated based on the conditional statement of the box 3040. Some embodiments may generate a first event 3042, second event 3050, or third event 3054 based on data associated with the conditional statement associated with the vertex 3041 enclosed in the box 3040. For example, an event may include a message to be received by an application program interface (API). A first event 3042 may encode data indicating that the first entity “Ent1” has transferred 36 units to a second entity “Ent2” at a time “t1.” Additionally, or alternatively, some embodiments may generate events based on an entity role, a value associated with the entity, or the like. For example, some embodiments may generate an event based on data from the entity profile 3090, where the event may indicate that the parameter “Free Allocation Units” has been updated from 1091 units to a new value from 6192.

[0236] In some embodiments, multiple simulated events may be generated from a conditional statement, where the simulated events may be modified with respect to each other to include values greater than or less than a parameter of the conditional statement. Some embodiments may generate multiple simulated events to test expected fulfillment states, expected violation states, expected non--responsiveness to certain events, or the like. For example, some embodiments may generate the first event 3042 and the second event 3052 based on the same vertex 3041. Some embodiments may determine whether the first event 3042 satisfies a first conditional statement that is expected to be satisfied by the first event 3042 and whether the first event 3042 satisfies a second conditional statement that is expected to be failed by the first event 3042. By generating a plurality of independent simulated events, some embodiments may concurrently test whether outcomes of the simulated events actually result in the fulfillment of conditional statements that are expected to be fulfilled or violation of conditional statements that are expected to be violated.

[0237] In some embodiments, a sequence of simulated events may be generated from a conditional statement. For example, some embodiments may generate a sequence of events that includes the second event 3052 and a third event 3054. In some embodiments, the sequence of events may be simulated as occurring at different times in a simulated environment. For example, some embodiments may simulate the occurrence of the second event 3052 at a time

t₂ and the occurrence of a third event 3054 at a time t₃. As further discussed below, the sequence of simulated events may be generated with associated time values, causing the sequence of simulated events to occur as a time-series in simulated time. In some embodiments, the values of one or more parameters of the sequence of simulated events may change over time based on a pre-set pattern. Such patterns may trend towards a constant value for a parameter of a simulated event. For example, some embodiments may simulate the allocation of 40 units. Alternatively, or in addition, some patterns may trend towards an oscillating set of values for a parameter over time, a monotonically increasing set of values for a parameter over time, a monotonically decreasing set of values for a parameter over time, a random or pseudo-random set of values for a parameter over time, or the like.

[0238] In some embodiments, the various operations of the processes described in this disclosure may be executed in a different order, operations may be omitted, operations may be replicated, additional operations may be included, some operations may be performed concurrently, some operations may be performed sequentially, and multiple instances of a process may be executed concurrently, none of which is to suggest that any other description herein is limited to the arrangement described. In some embodiments, the operations of a process described in this disclosure may be effectuated by executing program code stored in one or more instances of a machine-readable non-transitory medium, which in some cases may include storing different subsets of the instructions on different physical embodiments of the medium and executing those different subsets with different processors, an arrangement that is consistent with use of the singular term "medium" herein.

[0239] Figure 17 shows a flowchart of operations to generate and use integrated test conditions, in accordance with one or more embodiments. In some embodiments, the process 3100 may include obtaining a set of conditional statements associated with the first directed graph, as indicated by block 3104. The first directed graph may include a directed graph of a smart contract program stored in program state or otherwise may be associated with a smart contract program or another symbolic AI program having multiple participants. Some embodiments may obtain the set of conditional statements by searching through the vertices of the directed graph and extracting the set of conditional statements associated with the vertices. Some embodiments may apply one or more criteria to the set of vertices of the graph during the search or during another operation described in this disclosure, such as some embodiments may search only for active vertices of a directed graph. For example, some embodiments may search

through a directed graph of a program to obtain 20 active vertices, where each of the vertices may be associated with a respective conditional statement of a set of 20 conditional statements. An active vertex may be associated with a fulfilled or violated state in response to an event satisfying or failing a conditional statement of the active vertex.

[0240] As described in this disclosure, the fulfillment state associated with a vertex or its associated conditional statement may depend on the category type or whether a condition of the conditional statement has been fulfilled. Some embodiments may set the fulfillment state of a graph vertex or its associated conditional statement to indicate fulfillment if a condition of the conditional statement is satisfied. Alternatively, or in addition, some embodiments may set the fulfillment state of a graph vertex or its associated conditional statement to indicate violation if a condition of the conditional statement is satisfied. While the terms “fulfilled” and “violated” are used in this disclosure to describe whether a graph vertex is violated, some embodiments may perform other activities.

[0241] Alternatively, or in addition, some embodiments may search through a list of conditional statements encoded in a data structure listing conditional statements. For example, some embodiments may obtain a set of conditional statements by searching through a program state of an application to obtain conditional statements linked to an associative array of conditional statements such as the associative array of conditions 220 of figure 2. Alternatively, or in addition, some embodiments may search through a data structure that includes both conditional statements used by and not used by a vertex of the directed graph to collect a set of conditional statements. By searching through a list of conditional statements directly, some embodiments may reduce the number of searches to be performed when executing one or more operations of the process 300.

[0242] Some embodiments may further obtain multiple sets of conditional statements, where each of the multiple sets of conditional statements may be associated with different sets of smart contract data. A set of smart contract data may include a set of values of a smart contract stored in program state, data for an in-execution smart contract, a simulated smart contract, or historical smart contract data. For example, some embodiments may obtain a first set of conditional statements from an in-execution smart contract program and obtain a second set of conditional statements from a partial simulation of a smart contract program.

[0243] In some embodiments, the first directed graph or its associated set of conditional statements may be associated with a natural language document, where different vertices of the first directed graph may be linked to different text sections of the natural language document. For example, a directed graph may include four vertices associated with four conditional statements, where the first vertex is associated with a first text section, the second and third vertices are be associated with a second text section, and a fourth vertex is associated with a third and fourth text section of a document (e.g., a set of government regulations, a company policy, or the like). In some embodiments, each of the associated text sections may be obtained based on a set of NLP operations. For example, some embodiments may associate a text section used to determine a conditional statement with the vertex that encodes the conditional statement. Alternatively, or in addition, some embodiments may include a user interface that allows the direct association of a text section, text section position, or another indicator of a text section with a conditional statement or an associated vertex. Some embodiments may use these associations to indicate that sections of a natural language document have been fulfilled or violated by an event.

[0244] In some embodiments, the process 3100 may include determining a subset of conditional statements based on a shared category used by the set of conditional statements, as indicated by block 3108. Determining the subset of conditional statements may include collecting identifiers of conditional statements or associated graph vertices of a smart contract encoded in program state, where each of the conditional statements may share one or more associated category values. For example, some embodiments may determine a first subset of conditional statements based on each conditional statement of the first subset of conditional statements being labeled with the category label “prohibition.” Some embodiments may further determine the first subset of conditional statements based on a shared resource type or shared range encoded in the first subset of conditional statements. For example, some embodiments may select each conditional statement in a set of conditional statements based on which of the statements was listed as a prohibition on an allocation of 3 TB or more of computer memory.

[0245] Some embodiments may collect the subset of conditional statements based on a set of shared categories, parameter values, or other conditional statement properties. A shared category may be selected from one or more sets of categories listed above, such as category labels from a set of mutually exclusive categories. The labels for the mutually exclusive categories may indicate whether meeting the criteria of a conditional statement results in a

fulfillment state being set to “fulfilled” or “violated” of the category. For example, a vertex or its corresponding conditional statement labeled as a “prohibition” may cause or otherwise indicate that the vertex or associated conditional statement is violated if the conditional statement is satisfied.

[0246] In some embodiments, the subset of conditional statements may be selected based on a shared category label. For example, a subset of conditional statements may share the category label “prohibition,” and some embodiments may select conditional statements to form the subset of conditional statements based on the shared category label. Alternatively, or in addition, some embodiments may select conditional statements based on a shared resource type, an amount of the shared resource type, or the like. For example, some embodiments may select conditional statements when determining the subset of conditional statements based on each of them testing whether some amount of the resource type “computer memory” was allocated, whether or a threshold amount of digital currency was transferred, or the like.

[0247] Some embodiments may select the subset of conditional statements based on a combination of criteria. For example, some embodiments may determine a subset of conditional statements based on each of the conditional statements being labeled as a “prohibition” and including a condition that an amount of the shared resource type “cryptocurrency” being transferred is less than a set of thresholds. By using mixed criteria, some embodiments may increase the utility or applicability of an integrated test condition determined using the subset of conditional statements.

[0248] In some embodiments, the process 3100 may include generating an integrated test condition based on the subset of conditional statements, as indicated by block 3116. An integrated test condition may indicate whether it has been fulfilled or violated in response to an event. For example, if an integrated test condition includes the condition “if $X > 20$: return True,” some embodiments may update a fulfillment state of an integrated test condition to “fulfilled” in response to an event indicating a value for “X” that is greater than twenty. As discussed further below, determining that an integrated test condition has been fulfilled or violated may cause some embodiments to determine that at least one of the subset of conditional statements used to generate the integrated test condition has been fulfilled or violated.

[0249] An integrated test condition may be generated based on a shared resource type, shared category, other shared property of the subset of conditional statements, or the like. The integrated test condition may be used to simplify or reduce the number of conditions that must be tested to verify that a portion of a smart contract will not be violated. For example, instead of testing each of the subset of conditional statements, some embodiments may test the integrated test condition generated from the subset of conditional statements. Such operations may increase computing performance by reducing the total number of conditional statements that an application must test without reducing confidence in the test result. Furthermore, the generation of the integrated test condition may be standardized by taking advantage of shared category labels or resource types, which may reduce the time required to generate the testing conditions.

[0250] Some embodiments may determine one or more parameters of an integrated test condition based on the values of a set of thresholds associated with the subset of conditional statements determined above. For example, some embodiments may determine a subset of conditional statements associated with a set of values representing the resource type amounts 150, 141, and 987 to represent prohibitions on transferring amounts of cryptocurrencies greater than 150 cryptocurrencies, 141 cryptocurrencies, and 987 cryptocurrencies, respectively. Some embodiments may generate an integrated test condition having a test condition threshold that is based on the set of values, such as basing a parameter of the integrated test condition on a maximum, minimum, measure of central tendency (e.g., mean average, median, mode), a percentage thereof, or the like. For example, some embodiments may generate an integrated test condition having the test condition threshold 141 based on 141 being the minimum value of the set of values [141, 150, 987]. Such an integrated test condition may return a value indicating a violation in response to receiving an event indicating that a cryptocurrency transfer amount exceeds 141 cryptocurrencies. Alternatively, or in addition, some embodiments may use a percentage of the amount. For example, instead of using 141 as the test condition threshold, some embodiments may use 126.9 as the test condition threshold, where the test condition threshold is pre-set to be 90% of the minimum value.

[0251] In some embodiments, the integrated test condition may be associated with a date or time. For example, if the integrated test condition is determined from ten conditional statements, each of which is associated with a date, some embodiments may use an extremum of the set of dates (e.g., the earliest date or the latest date) when generating the integrated test

condition. Some embodiments may select whether to use an earliest or latest date based on indicated failure state associated with one or more elements of the subset of conditional statements. For example, some embodiments may determine that a prohibition on an amount to be transferred is effective until a date threshold. The date threshold may be used in the conditional statement in conjunction with a “less than” operator function, a specialized function, or the like. In response, some embodiments may select the earliest date of the set of dates.

[0252] Some embodiments may generate multiple integrated test conditions from one subset of the conditional statements. For example, a first, second, and third conditional statement may be labeled as “violated” in response to a first entity having less 500 units of a digital asset, having less than 1000 units of the digital asset, or having less than 5000 units of the digital asset on different dates, respectively. Some embodiments may generate a first integrated test condition indicating violation of the integrated test condition in response to the first entity having less than 500 units of the digital currency during any time after the earliest of the different dates. Some embodiments may also generate a second integrated test condition indicating violation of the integrated test condition in response to the first entity having less than 5000 units of the digital currency during any time after the latest of the different dates.

[0253] In some embodiments, the process 3100 may include obtaining a second directed graph, as indicated by block 3120. In some embodiments, the second directed graph may be associated with a second program, where a version of the second directed graph may be stored in program state. Additionally, the second program may include a first entity as a participant, where the first entity is also associated with the first directed graph as a participant or potential participant. An entity listed as a participant of a program may participate in a transaction with another entity of the program, communicate with another entity of the program, change a value of the program stored in program state, or the like. For example, a first entity may be a participant in a smart contract program if a transaction between the first and second entities causes a change in the ledger value of the smart contract program stored in program state.

[0254] In some embodiments, the second directed graph may be a portion of a larger directed graph that includes the first directed graph and the second directed graph. For example, a third directed graph may include a first graph portion and a second graph portion. The first graph portion may have five vertices, four of which are connected by three directed edges. The second graph portion may have six vertices connected to each other by five directed edges. As

discussed elsewhere in this disclosure, each of the vertices and edges may be stored in program state as arrays, table values, records, or other data structures, where each respective may be associated with a respective conditional statement. Some embodiments may partition the third directed graph into the first graph portion and second graph portion. Some embodiments may then use the first graph portion as a first directed graph discussed above for block 3104 and use the second graph portion as a second directed graph.

[0255] In some embodiments, the second directed graph may include an updated directed graph that is updated with respect to a previous version of the directed graph. Some embodiments may execute an amendment to a program or simulate the execution of an amendment to the program that causes an update to a previous version of a directed graph of the program stored in program state. The updated graph may then be used as a second directed graph. For example, some embodiments may apply an amendment that changes the conditions and outcomes of a smart contract. Applying the amendment may cause some embodiments to update a directed graph of the smart contract to add a child vertex to the directed graph, where the child vertex may be associated with a parent vertex of the directed graph via an additional directed edge added by the amendment. As further described below, in some embodiments, the parent vertex may encode or otherwise be associated with a conditional statement being used to simulate an event. Alternatively, or in addition, some embodiments may apply an amendment or a smart contract that changes a parameter associated with a first vertex of the directed graph. The amended directed graph, or a copy of the amended directed graph, may then be used as the second directed graph. As further discussed in this disclosure, by simulating changes to a directed graph or their corresponding vertices, some embodiments may test various possible changes to reduce the risk of amending a program such that it conflicts with one or more other programs.

[0256] In some embodiments, the process 3100 may include determining a simulated event based on the conditional statement of the second directed graph, as indicated by block 3124. Some embodiments may simulate the occurrence of an event by generating event messages, where the event messages may be received by an API smart contract program or computing platform on which the smart contract program is executed. For example, some embodiments may simulate the transfer of 500 resource units (or some other numeric value representing resource units) from a first entity to a second entity by sending a message to an API of observer node of a computing platform in a development environment. Alternatively, or in addition,

some embodiments may simulate an event by changing program state values to emulate what those values would be had the event occurred. For example, some embodiments may simulate a reduction in a sentiment score by reducing the sentiment score in a simulated environment.

[0257] Some embodiments may determine what event to simulate based on a conditional statement. For example, a set of conditional statements of the second directed graph may include a first conditional statement to indicate fulfillment if a first set of conditions are satisfied. Some embodiments may determine the parameters of a first simulated event based on the first set of conditions, such as simulating events based on a maximum or minimum value(s) encoded in a conditional statement. For example, a first conditional statement may be indicated as fulfilled if a determination is made that a first entity has a total score that is greater than a threshold. Some embodiments may use this first conditional statement to simulate an event by generating a simulated event message indicating that the total score of the first entity is greater than the threshold value. Alternatively, or in addition, a second conditional statement may cause some embodiments to indicate that the first conditional statement is violated if the total score is less than the threshold after a time limit (e.g., a time of day, a duration, a date, or the like) is reached. Some embodiments may use this second conditional statement to simulate a second event by generating a simulated event message indicating that the total score of the first entity is less than the threshold value at a simulated time after the time limit.

[0258] Some embodiments may simulate events that are associated with a change in an environmental variable, such as by changing the environmental variable in a simulation environment. Various conditional statements related to an entity may be based on an environmental variable. For example, some embodiments may test a conditional statement that causes a first entity to re-allocate or deactivate a computing resource in response to a temperature exceeding a temperature threshold. Some embodiments may simulate an event indicating that a server temperature has exceeded a temperature threshold based on the conditional statement. As further described below, some embodiments may then determine whether the temperature change or other environmental variable change causes a fulfillment or violation of an integrated test condition.

[0259] In some embodiments, the simulated event may be based on a conditional statement of a parent vertex updated by an amendment to add a child vertex to the second directed graph. For example, as discussed above, some embodiments may amending a condition of a smart contract, which include modifying a directed graph of the smart contract with an additional

vertex and additional directed edge leading away from a parent vertex and changing the conditional statement associated with the parent vertex. By simulating possible outcomes based on events generated from parameters of the updated conditional statement of the parent vertex, some embodiments may be able to efficiently test that outcome states of the amended directed graph by providing tests that are most likely to reveal possible issues with an updated directed graph with respect to the directed graph(s) of other programs.

[0260] Some embodiments may generate a set of simulated events indicated to have occurred in the same execution of a simulation. The set of simulated events may be simulated to occur concurrently, in sequence, or some combination thereof with respect to each other. For example, some embodiments may generate a sequence of six simulated events, where each simulated event indicates that a first entity has transferred a score value to a second entity at the start of each month for six months. By generating sequences of simulated events, an integrated test condition based on accumulated scores (e.g., an accumulated amount of computing resource, an accumulated amount of a digital asset, or the like) may be tested. In some embodiments, a sequence of simulated events may be determined based on a pre-determined pattern, such as an oscillating pattern, a monotonically increasing pattern, a monotonically decreasing pattern, or a constant value. For example, some embodiments may generate a sequence of events indicating that a first entity is to allocate 100 GB of memory, 200 GB of memory, and 300 GB of memory to a second entity at the start of each month based on a monotonically increasing pattern. Alternatively, or in addition, some embodiments may generate multiple sets of simulated events for different sets of integrated test conditions. For example, a first set of integrated test conditions may be associated with a first smart contract program, and a second set of integrated test conditions may be associated with a second smart contract program. Some embodiments may use different parameters or patterns to generate different respective sets of simulated events for the first and second sets of integrated test conditions.

[0261] Some embodiments may generate a sequence of events based on conditional statements of vertices that are activated from previous events in the sequence of events. For example, during a simulation of a sequence of events, a first event may trigger a first vertex of a first directed graph, causing a second vertex of the first directed graph to activate. Some embodiments may generate a second event for the same simulation based on a parameter of the conditional statement of the second vertex. In some embodiments, the second event may then

trigger a conditional statement of the first program or an integrated test condition of a second program, which may indicate that the second event triggers one or more conditional statements of the second program.

[0262] In some embodiments, the process 3100 may include determining whether the simulated event triggers the set of integrated test conditions, as indicated by block 3140. As discussed above, an integrated test condition may include or be based on properties or parameters of one or more conditional statements. Alternatively, or in addition, some embodiments may determine that a simulated event has triggered the integrated test conditions if an outcome state caused by the simulated event triggers the integrated test condition.

[0263] Some embodiments may determine that an integrated test condition has been triggered based on a determination that a fulfillment state of the integrated test condition has been changed from indicating no fulfillment to a different value. For example, some embodiments may determine that, in response to an event or an outcome of the event, an integrated test condition has been fulfilled or violated. For example, a state of an integrated test condition may be updated to indicate fulfillment if an entity score is greater than a first threshold, be updated to indicate violation if the entity score is less than the second threshold, or remain triggerable if the entity score is between the first threshold and the second threshold. Some embodiments may simulate an event that causes the entity score to fall below the second threshold, and test the entity score with the integrated test condition to receive a value indicating that the integrated test condition has been violated. In response, some embodiments may determine that the integrated test condition has been triggered.

[0264] As disclosed above, some embodiments may apply a simulated event in real-time or over a simulated period of time when testing a second directed graph. Some embodiments may simulate the occurrence of an event that is indicated to have occur over an hour, over a day, over a week, over a month, or over year longer than a present time or a past time. Some embodiments may determine whether a time value of a simulated event, such as a timestamp encoded with the simulated event, satisfies a time threshold of the integrated test condition. For example, some embodiments may simulate the occurrence of a sequence of events, where each simulated event indicates the transfer of an amount of a digital currency from a first entity to a second entity at the end of every month starting from the year 2000 to the year 2100. Some embodiments may then determine whether this transfer schedule satisfies a first time threshold of an integrated test condition. For example, if the integrated test condition requires that a

transfer not occur on a specific day or not occur within a time interval that covers the first day of a month, some embodiments may determine that the event has triggered the corresponding integrated test condition.

[0265] Alternative to or in addition to determining whether a simulated event triggers an integrated test condition generated from conditional statements associated with vertices of a directed graph, some embodiments may determine whether a simulated event triggers other types of test conditions. For example, some embodiments may determine whether a simulated event satisfies an alert condition indicating that conflicting requirements are made of an entity. Some embodiments may determine whether a simulated outcome state caused by the simulated event indicates the activation of a conditional statement (e.g., via activating a graph vertex associated with the second conditional statement) that is labeled as a prohibition. In some embodiments, the activated conditional statement may prohibit an activity that would have been required to satisfy an obligation norm or otherwise listed as a required activity by an entity. For example, some embodiments may simulate the occurrence of a first event indicating the allocation of a first computing resource (e.g., an amount of computing time on a server having a set of tensor processing units) from a first entity to a second entity. This first event may activate a prohibition norm associated with a conditional statement that prohibits the first entity from allocating additional computing resources to the second entity. If the first entity is obligated to provide additional computing resources to the second entity due to a second conditional statement of a pre-existing norm or newly-activated norm categorized as an obligation, some embodiments detect these conflicting requirements by simulating the occurrence of a second event performed by the first entity to satisfy this second conditional statement. Once a determination is made that the second event triggers the prohibition norm, some embodiments may determine that test condition has been triggered and indicate or send a message regarding the violation of the test condition, as further described below.

[0266] If a determination is made that the simulated event has triggered an integrated test condition, operations of the process 3100 may proceed to operations described for block 3144. Otherwise, operations of the process 3100 may proceed to operations described for block 3160.

[0267] In some embodiments, the process 3100 may include indicating vertices of the first directed graph associated with a triggered integrated test condition, as indicated by block 3144. As discussed above, an integrated test condition may be combined from a plurality of conditional statements of a first directed graph, where each respective conditional statement of

the plurality of conditional statements may be associated with a respective vertex of the first directed graph. For example, an integrated test condition may be determined based on a first, second, and third conditional statement, each of which is associated with a respective first, second, and third vertex of the first directed graph. The integrated test condition may include a pointer to or otherwise be associated with each of the three respective vertices. After a determination that an integrated test condition has been triggered, some embodiments may use the association between the integrated test condition and each of the respective vertices to generate an indicator. The indicator may be used to indicate that the simulated event triggered at least one of the three respective vertices.

[0268] Some embodiments may be able to include program code to determine which conditional statement associated with a triggered integrated test condition was satisfied or failed. For example, an integrated test condition may specify that a first conditional statement used to generate the integrated test condition was failed based on the specified conditional statement being used to generate a parameter used by the integrated test condition. Some embodiments may then store a value identifying the specified test condition that was failed, generate an indicator for the specified test condition, or the like.

[0269] As discussed above, some embodiments may store associations between vertices and text sections of a natural language document. Some embodiments may use these associations to determine a set of possible sections of the natural language document triggered by an event based on the triggering of a corresponding integrated test condition. For example, some embodiments may associate a first vertex with a first clause in a contract document written in natural language form. A conditional statement of the first vertex may be used to generate an integrated test condition, and upon a determination that the integrated test condition has been failed by an event, some embodiments may generate an indicator indicating the first clause of the natural language document. The indicator may be stored in a record of a database or other data structure or may be used to visually present the clause on a graphical display screen.

[0270] Some embodiments may determine an entity predicted to be affected by a possible amendment based on the entities associated with the simulation-triggered vertices. Some embodiments may then send a message to the possibly-affected entity, the message indicating that the integrated test condition with the simulated event has been satisfied. For example, some embodiments may determine that a triggered integrated test condition is associated with five graph vertices via their associated conditional statements, determine that a first vertex of the

five graph vertices is actually triggered, and determine the set of entities associated with the first vertex. Some embodiments may then send a message via to an email, account, or API of one or more entities of the set of entities associated with the first vertex.

[0271] In some embodiments, the process 3100 may include storing a result of testing the integrated test condition with the simulated event, as indicated by block 3160. For example, some embodiments may store an identifier for an integrated test condition, a simulated API message sent in a simulation environment, a value representing a fulfillment or violation state of the integrated test condition, or the like. Some embodiments may store the testing result on a distributed ledger or other distributed data storage distributed across multiple persistent memory storage devices, such as one described in this disclosure (e.g., IPFS, Swarm, or the like). By using distributed data storage, some embodiments may provide a means of securely sharing simulation results. This secure sharing may be advantageous in cases where multiple entities may be simulating similar amendments to a smart contract but have different permission authorizations for viewing the conditional statements of smart contracts. By determining and storing the result of one or more operations described in this disclosure, some embodiments allow a first entity to test proposed amendments for a first smart contract with respect to conditions of other smart contract programs. Furthermore, by using integrated test conditions, some embodiments allow such testing even if the first entity does not have access to read or otherwise view the conditional statements of the other programs.

[0272] As described above, some embodiments may generate a set of integrated test conditions to reduce an effective number of computations to determine whether an event will trigger a set of vertices. Additionally, some embodiments may reduce an effective amount of data to analyze when categorizing or comparing directed graphs using dimensional reduction operation. Some embodiments may perform operations, such as those described further below, to more efficiently analyze one or more types of directed graphs.

[0273] DIMENSIONAL REDUCTION OF CATEGORIZED DIRECTED GRAPHS

[0274] Attempts to analyze or optimize real-world programs used to formalize, automate, or enforce repeated transactions between multiple entities may quickly suffer from the curse of dimensionality. Such programs are often challenged with large datasets having multiple (over 5, over 10, over 100) variables and more than 20, more than 50, more than 1000, or more than 1,000,000 individual records due to both broadly-applicable challenges and domain-specific

challenges. Broadly used operations such as processing a large volume of data, determining similarities between different programs, and implementing meaningful visualization of relatively small sections of the data may be exacerbated by data characteristics in specific domains. In the case of smart contracts or other symbolic AI programs, domain-specific issues may include adapting operations to account for the various types of features that can be obtained from data stored in program state, relationships between different conditional statements, and differing priorities of different data consumers. The variety of data types, complex nature of relationships between terms or entities of a smart contract, and different priorities of different visualization audiences may challenge attempts to determine the most significant variables of a smart contract, compare the similarity of different contracts, and visualization attempts.

[0275] Some embodiments may use a directed graph encoding elements of a smart contracts or other symbolic AI programs to determine a set of features and vertices to prioritize or display. Some embodiments may determine a set of features for a set of vertices of the directed graph, where the features and their corresponding feature values for a vertex may be used to characterize the vertex, its relationships with other vertices in the directed graph, or the like. Some embodiments may include determining multiple candidate subsets of features or performing other feature selection operation to determine a prioritized subset of features from the set of features. Some embodiments may perform a feature extraction operation that accounts for the prioritized subset of features to determine a prioritized subset of vertices, where such operations may include using a neural network, applying principal component analysis (PCA), or the like. Some embodiments may then visually indicate the prioritized subset of features or the prioritized subset of vertices. Some embodiments may apply dimension reduction results to compare different graphs to determine which elements of a program state may be common across multiple smart contract programs and, as a result, also determine which set of vertices are uncommon.

[0276] By visualizing smart contract programs or other symbolic AI programs encoding transactions between multiple entities as directed graphs and indicating outcome-relevant vertices, some embodiments may increase the interpretability of directed graphs. Furthermore, some embodiments may use category labels associated with a vertex or other structural associated with a conditional statement to increase the efficiency and accuracy of analysis results. For example, as further discussed below, a feature of a vertex may include the category

label “right,” which is selected from a set of mutually exclusive categories that may include the category labels “right,” “obligation,” or “prohibition.” Such a feature may also be used to accelerate further feature processing by being used as an indicator of a pre-established outcome that occurs upon satisfaction or failure of a conditional statement or of structural elements. For example, an indicator may indicate structural elements such as a condition of a vertex being associated with one entity and one score threshold, an outcome of the vertex being associated with two entities and two transaction scores, an outcome of the vertex causing an anomalous program exit for all entities, or the like). By indicating features that most affect a directed graph, indicating similarities between different graphs, or otherwise indicating high-effect parameters and vertices, some embodiments may provide a tool that can reduce issues that are encountered when dealing with a high dimensional parameter space. Such issue reduction may in turn increase efficiency and accuracy when generating, updating, or simulating a program that is modeled by directed graph.

[0277] In some embodiments, the present techniques may be implemented using data visualization packages implemented in code such as Python code, Javascript code, or the like. In some embodiments, code implementing the present techniques may be implemented using one or more statistical programming languages such as R and data visualization packages for R, such as Ggplot2. Some embodiments may use Python and data visualization libraries for Python, such as Matplotlib or Seaborn. Some embodiments may use data visualization tools or platforms, such as Gephi™, Neo4j™, Tableau™, Wolfram Mathematica™, Matlab™, or the like. Some embodiments may use web-compatible tools to display data, such as the Javascript library D3.js, and permit the initiation or modification of one or more operations described in this disclosure via a user interface (UI) sent over the Internet. Some embodiments may integrate different data visualization tools. For example, some embodiments may implement one or more operations described in this disclosure using the Python Bokeh library, which may allow Python to use one or more elements of Tableau™ to visualize data. Some embodiments may perform just in time compilation of program code and may parse the code to an abstract syntax tree, which may then be transformed into a bytecode representation that is then compiled into machine code (e.g., native machine code of the computer executing the text viewing application, or machine code of a virtual machine).

[0278] In block diagrams, illustrated components are depicted as discrete functional blocks, but embodiments are not limited to systems in which the functionality described herein is

organized as illustrated. The functionality provided by each of the components may be provided by software or hardware modules that are differently organized than is presently depicted, for example such software or hardware may be intermingled, conjoined, replicated, broken up, distributed (e.g. within a data center or geographically), or otherwise differently organized. The functionality described herein may be provided by one or more processors of one or more computers executing code stored on a tangible, non-transitory, machine readable medium. In some cases, notwithstanding use of the singular term "medium," the instructions may be distributed on different storage devices associated with different computing devices, for instance, with each computing device having a different subset of the instructions, an implementation consistent with usage of the singular term "medium" herein. In some cases, third party content delivery networks may host some or all of the information conveyed over networks, in which case, to the extent information (e.g., content) is said to be supplied or otherwise provided, the information may be provided by sending instructions to retrieve that information from a content delivery network.

[0279] The reader should appreciate that the present application describes several independently useful techniques. Rather than separating those techniques into multiple isolated patent applications, applicants have grouped these techniques into a single document because their related subject matter lends itself to economies in the application process. But the distinct advantages and aspects of such techniques should not be conflated. In some cases, embodiments address all of the deficiencies noted herein, but it should be understood that the techniques are independently useful, and some embodiments address only a subset of such problems or offer other, unmentioned benefits that will be apparent to those of skill in the art reviewing the present disclosure. Due to costs constraints, some techniques disclosed herein may not be presently claimed and may be claimed in later filings, such as continuation applications or by amending the present claims. Similarly, due to space constraints, neither the Abstract nor the Summary of the Invention sections of the present document should be taken as containing a comprehensive listing of all such techniques or all aspects of such techniques.

[0280] It should be understood that the description and the drawings are not intended to limit the present techniques to the particular form disclosed, but to the contrary, the intention is to cover all modifications, equivalents, and alternatives falling within the spirit and scope of the present techniques as defined by the appended claims. Further modifications and alternative embodiments of various aspects of the techniques will be apparent to those skilled in the art in

view of this description. Accordingly, this description and the drawings are to be construed as illustrative only and are for the purpose of teaching those skilled in the art the general manner of carrying out the present techniques. It is to be understood that the forms of the present techniques shown and described herein are to be taken as examples of embodiments. Elements and materials may be substituted for those illustrated and described herein, parts and processes may be reversed or omitted, and certain features of the present techniques may be utilized independently, all as would be apparent to one skilled in the art after having the benefit of this description of the present techniques. Changes may be made in the elements described herein without departing from the spirit and scope of the present techniques as described in the following claims. Headings used herein are for organizational purposes only and are not meant to be used to limit the scope of the description.

[0281] As used throughout this application, the word “may” is used in a permissive sense (i.e., meaning having the potential to), rather than the mandatory sense (i.e., meaning must). The words “include”, “including”, and “includes” and the like mean including, but not limited to. As used throughout this application, the singular forms “a,” “an,” and “the” include plural referents unless the content explicitly indicates otherwise. Thus, for example, reference to “an element” or “a element” includes a combination of two or more elements, notwithstanding use of other terms and phrases for one or more elements, such as “one or more.” The term “or” is, unless indicated otherwise, non-exclusive, i.e., encompassing both “and” and “or.” Terms describing conditional relationships, e.g., “in response to X, Y,” “upon X, Y,” “if X, Y,” “when X, Y,” and the like, encompass causal relationships in which the antecedent is a necessary causal condition, the antecedent is a sufficient causal condition, or the antecedent is a contributory causal condition of the consequent, e.g., “state X occurs upon condition Y obtaining” is generic to “X occurs solely upon Y” and “X occurs upon Y and Z.” Such conditional relationships are not limited to consequences that instantly follow the antecedent obtaining, as some consequences may be delayed, and in conditional statements, antecedents are connected to their consequents, e.g., the antecedent is relevant to the likelihood of the consequent occurring. Statements in which a plurality of attributes or functions are mapped to a plurality of objects (e.g., one or more processors performing steps A, B, C, and D) encompasses both all such attributes or functions being mapped to all such objects and subsets of the attributes or functions being mapped to subsets of the attributes or functions (e.g., both all processors each performing steps A-D, and a case in which processor 1 performs step A, processor 2 performs step B and part of step C, and processor 3 performs part of step C and

step D), unless otherwise indicated. Similarly, reference to “a computer system” performing step A and “the computer system” performing step B can include the same computing device within the computer system performing both steps or different computing devices within the computer system performing steps A and B. Further, unless otherwise indicated, statements that one value or action is “based on” another condition or value encompass both instances in which the condition or value is the sole factor and instances in which the condition or value is one factor among a plurality of factors. Unless otherwise indicated, statements that “each” instance of some collection have some property should not be read to exclude cases where some otherwise identical or similar members of a larger collection do not have the property, i.e., each does not necessarily mean each and every. Limitations as to sequence of recited steps should not be read into the claims unless explicitly specified, e.g., with explicit language like “after performing X, performing Y,” in contrast to statements that might be improperly argued to imply sequence limitations, like “performing X on items, performing Y on the X’ed items,” used for purposes of making claims more readable rather than specifying sequence. Statements referring to “at least Z of A, B, and C,” and the like (e.g., “at least Z of A, B, or C”), refer to at least Z of the listed categories (A, B, and C) and do not require at least Z units in each category. Unless specifically stated otherwise, as apparent from the discussion, it is appreciated that throughout this specification discussions utilizing terms such as “processing,” “computing,” “calculating,” “determining” or the like refer to actions or processes of a specific apparatus, such as a special purpose computer or a similar special purpose electronic processing/computing device. Features described with reference to geometric constructs, like “parallel,” “perpendicular/orthogonal,” “square,” “cylindrical,” and the like, should be construed as encompassing items that substantially embody the properties of the geometric construct, e.g., reference to “parallel” surfaces encompasses substantially parallel surfaces. The permitted range of deviation from Platonic ideals of these geometric constructs is to be determined with reference to ranges in the specification, and where such ranges are not stated, with reference to industry norms in the field of use, and where such ranges are not defined, with reference to industry norms in the field of manufacturing of the designated feature, and where such ranges are not defined, features substantially embodying a geometric construct should be construed to include those features within 15% of the defining attributes of that geometric construct. The term “set” may indicate a single item or a plurality of items, e.g., “set of widgets” may indicate only one widget or may indicate multiple widgets. The terms “first,” “second,” “third,” “given” and so on, if used in the claims, are used to distinguish or otherwise identify, and not to show a sequential or numerical limitation. As is the case in ordinary usage

in the field, data structures and formats described with reference to uses salient to a human need not be presented in a human-intelligible format to constitute the described data structure or format, e.g., text need not be rendered or even encoded in Unicode or ASCII to constitute text; images, maps, and data-visualizations need not be displayed or decoded to constitute images, maps, and data-visualizations, respectively; speech, music, and other audio need not be emitted through a speaker or decoded to constitute speech, music, or other audio, respectively. Computer implemented instructions, commands, and the like are not limited to executable code and can be implemented in the form of data that causes functionality to be invoked, e.g., in the form of arguments of a function or API call.

[0282] In this patent, to the extent any U.S. patents, U.S. patent applications, or other materials (e.g., articles) have been incorporated by reference, the text of such materials is only incorporated by reference to the extent that no conflict exists between such material and the statements and drawings set forth herein. In the event of such conflict, the text of the present document governs, and terms in this document should not be given a narrower reading in virtue of the way in which those terms are used in other materials incorporated by reference.

[0283] The present techniques will be better understood with reference to the following enumerated embodiments:

A-1. A tangible, non-transitory, machine-readable medium storing instructions that when executed by one or more processors effectuate operations comprising: determining, with a computer system, that an event has occurred; selecting, with the computer system, a self-executing protocol among a plurality of self-executing protocols based on the event, wherein: the self-executing protocol comprises a set of conditions, a set of entities, a set of vertices, and a set of directed graph edges connecting the set of vertices, the set of vertices comprise different respective subsets of the conditions, the set of entities are encoded in an associative array, the set of conditions are encoded in an associative array, the set of vertices are encoded as a serialized array of vertices, wherein the serialized array of vertices is in a serialized data format in persistent storage, selecting is based on whether the event satisfies any of the set of conditions; deserializing, with the computer system, the serialized array of vertices to generate a directed graph in a non-persistent memory, wherein the directed graph encodes the set of conditions, set of vertices, set of entities, and set of directed edges; determining, with the computer system, a set of triggerable vertices from the vertices of the directed graph in the non-persistent memory; determining, with the computer system, a set of triggered vertices from the

set of triggerable vertices based on which of the set of triggerable vertices are associated with the set of conditions satisfied by the event; updating, with the computer system, the directed graph in the non-persistent memory based on the set of triggered vertices, wherein updating the directed graph comprises, for each respective triggered vertex of the set of triggered vertices: updating a first value associated with the respective triggered vertex based on the event, where the first value indicates whether the respective triggered vertex is triggerable; updating a respective adjacent vertex to indicate that the respective adjacent vertex is triggerable, wherein the respective adjacent vertex is associated with a directed graph edge of the respective triggered vertex; updating, with the computer system, the serialized array of vertices by serializing the directed graph in the non-persistent memory after updating the directed graph in the non-persistent memory based on the set of triggered vertices; and persisting, with the computer system, the serialized array of vertices to the persistent storage after the serialized array of vertices is updated by serialization.

A-2. The medium of embodiment A-1, wherein: a first vertex in the set of vertices is indicated to not be triggerable by a first set of values, wherein each of the first set of values indicate whether a vertex in the set of vertices is triggerable; and the directed graph in the non-persistent memory does not include the first vertex of the serialized array of vertices.

A-3. The medium of any of embodiments A-1 to A-2, wherein the serialized array of vertices comprises an array of subarrays, wherein each subarray comprises a head vertex of a directed graph edge of the set of directed graph edges, a tail vertex of the directed graph edge, a label associated with the directed graph edge, and a valence value indicating a number of other edges associated with the directed graph edge.

A-4. The medium of any of embodiments A-1 to A-3, wherein determining that an event occurred further comprises: receiving an event message from a publisher, wherein the publisher is identified by a publisher identifier; determining whether the publisher is associated with one of a set of authorized publishers based on the publisher identifier; and authorizing the event message based on a determination that the publisher identifier is associated with one of the set of authorized publishers.

A-5. The medium of any of embodiments A-1 to A-4, wherein the operations further comprise: receiving an event message from a publisher, wherein the event message is associated with a signature value and a publisher identifier; retrieving a cryptographic certificate based on the publisher identifier; computing a cryptographic hash value based on the signature value; and

authenticating the event message based on the cryptographic hash value and the cryptographic certificate.

A-6. The medium of any of embodiments A-1 to A-5, wherein determining the set of triggered vertices comprises: determining a first set of vertices in the directed graph in the non-persistent memory, wherein each respective vertex of the first set of vertices is indicated as a head vertex by one of the set of directed graph edges; and determining the set of triggerable vertices based on the first set of vertices by filtering out a set of tail vertices from the first set of vertices, wherein each of the set of tail vertices is indicated as a tail vertex by one of the set of directed graph edges.

A-7. The medium of any of embodiments A-1 to A-6, wherein the serialized array of vertices is stored in a tamper-evident data store being executed by a set of peer nodes, wherein the tamper-evident data store comprises a directed acyclic graph of cryptographic hash pointers, and wherein deserializing the serialized array of vertices comprises using a first node of the set of peer nodes to deserialize the serialized array of vertices, and wherein the operations further comprising transmitting the serialized array of vertices from the first node to another node of the set of peer nodes after updating the serialized array of vertices.

A-8. The medium of any of embodiments A-1 to A-7, the operations further comprising receiving an event message, wherein receiving the event message comprises receiving a request that comprises the event message, and wherein the request comprises a method identifier and a host identifier, wherein the method identifier indicates that the request comprises an amount of data to modify data stored by the system, and wherein the host identifier indicates a host of the self-executing protocol.

A-9. The medium of any of embodiments A-1 to A-8, the operations further comprising receiving an event message, wherein the event message comprises a routing key, and wherein a data broker stores the event message in a queue, and wherein a protocol broker transmits the event message to an API associated with the self-executing protocol based on the routing key.

A-10. The medium of any of embodiments A-1 to A-9, wherein determining the set of triggered vertices comprises determining the set of triggered vertices based on a second set of values, wherein each of the second set of values is associated with one of a set of vertices of the directed graph in the non-persistent memory, and wherein one of the second set of values indicate that one of the set of vertices of the directed graph in the non-persistent memory is triggerable.

A-11. The medium of any of embodiments A-1 to A-10, wherein determining that the event has occurred comprises determining that a condition expiration threshold has been satisfied, and wherein the condition expiration threshold is associated with a first condition of a first triggerable vertex, and wherein the event does not satisfy the first condition.

A-12. The medium of any of embodiments A-1 to A-11, the operations further comprising updating an array of previously-triggered vertices based on a vertex identifier associated with the respective triggered vertex.

A-13. The medium of any of embodiments A-1 to A-12, the operations further comprising generating an initial directed graph based on an initial set of vertices, wherein the initial set of vertices is different from the serialized array of vertices.

A-14. The medium of any of embodiments A-1 to A-13, wherein a vertex of the directed graph stored in the non-persistent memory comprises a condition of the set of conditions.

A-15. The medium of any of embodiments A-1 to A-14, the operations further comprising updating a third set of values associated with the serialized array of vertices, wherein the third set of values indicate that the respective triggered vertex is not triggerable.

A-16. The medium of any of embodiments A-1 to A-15, wherein updating the respective adjacent vertex comprises setting a plurality of statuses associated with a plurality of vertices other than the respective triggered vertex as not triggerable.

A-17. The medium of any of embodiments A-1 to A-16, wherein updating the first value comprises updating the first value to indicate that the respective triggered vertex remains triggerable after updating the serialized array of vertices.

A-18. The medium of embodiment A-17, wherein updating the respective adjacent vertex comprises decreasing a second value, wherein the second value indicates a state of the self-executing protocol.

A-19. The medium of any of embodiments A-1 to A-18, the operations further comprising updating a set of previous events based on the event, wherein the set of previous events comprises a plurality of previous events that caused a state change in the self-executing protocol, wherein the set of previous events comprises a time during which the event occurred.

A-20. A tangible, non-transitory, machine-readable medium storing instructions that when executed by one or more processors effectuate operations comprising: determining, with a computer system, that an event has occurred; selecting, with the computer system, a self-

executing protocol among a plurality of self-executing protocols based on the event, wherein: the self-executing protocol comprises a set of conditions, a set of entities, a set of vertices, and a set of directed graph edges connecting the set of vertices, the set of vertices comprise different respective subsets of the conditions, the set of entities are encoded in an associative array, the set of conditions are encoded in an associative array, the set of vertices are encoded as a serialized array of vertices, wherein the serialized array of vertices is in a serialized data format in persistent storage, selecting is based on whether the event satisfies any of the set of conditions; deserializing, with the computer system, the serialized array of vertices to generate a directed graph in a non-persistent memory, wherein the directed graph encodes the set of conditions, set of vertices, set of entities, and set of directed edges; determining, with the computer system, a set of triggerable vertices from the vertices of the directed graph in the non-persistent memory; determining, with the computer system, a set of triggered vertices from the set of triggerable vertices based on which of the set of triggerable vertices are associated with the set of conditions satisfied by the event; updating, with the computer system, the directed graph in the non-persistent memory based on the set of triggered vertices, wherein updating the directed graph comprises, for each respective triggered vertex of the set of triggered vertices: updating a first value associated with the respective triggered vertex based on the event, where the first value indicates whether the respective triggered vertex is triggerable; updating a respective adjacent vertex to indicate that the respective adjacent vertex is triggerable, wherein the respective adjacent vertex is associated with a directed graph edge of the respective triggered vertex; updating, with the computer system, the serialized array of vertices by serializing the directed graph in the non-persistent memory after updating the directed graph in the non-persistent memory based on the set of triggered vertices; and persisting, with the computer system, the serialized array of vertices to the persistent storage after the serialized array of vertices is updated by serialization.

A-21. A method to perform the operations of any of the embodiments A-1 to A-19.

A-22. A system, comprising: one or more processors; and memory storing instructions that when executed by the processors cause the processors to effectuate operations comprising: the operations of any one of embodiments A-1 to A-19.

B-1. A tangible, non-transitory, machine-readable medium storing instructions that when executed by one or more processors effectuate operations comprising: obtaining, with a computer system, a set of conditional statements, wherein: a conditional statement of the set of conditional statements is associated with an outcome subroutine that specifies operations in

each of one or more branches of the conditional statement, a set of index values index the set of conditional statements, and a first outcome subroutine of a first conditional statement of the set of conditional statements uses a first index value of the set of index values, wherein the first index value is associated with a second conditional statement of the set of conditional statements; executing, with the computer system, a program instance of an application based on the set of conditional statements, wherein program state data of the program instance comprises: a set of vertices and a set of directed graph edges, wherein each of the set of vertices comprises a identifier value and is associated with one of the set of conditional statements, and wherein each of the set of directed graph edges associates a pair of the set of vertices and a direction from a tail vertex of the pair to a head vertex of the pair, a set of statuses, wherein each of the set of statuses is associated with one of the set of vertices, a set of vertex categories, wherein each of the set of vertex categories is a category value and is associated with a respective vertex of the set of vertices and is determined based a respective conditional statement of the respective vertex, and a set of scores, wherein each respective score of the set of scores is associated with a respective vertex and is based a respective conditional statement of the respective vertex; updating, with the computer system, the program state data based on a set of inputs comprising a first input, wherein updating the program state data comprises: modifying a status of a first vertex of the set of vertices based on the first input, updating a vertex adjacent to the first vertex; and determining, with the computer system, an outcome score based on the set of scores after updating the program state data.

B-2. The medium of embodiment B-1, wherein the status is a first status, and wherein updating the program state data comprises updating the program state data based on the first status, and wherein the operations further comprise: modifying a second status of a second vertex of the set of vertices based on a second input; updating a third vertex adjacent to the second vertex, wherein determining the outcome score comprises determining the outcome score after updating the third vertex.

B-3. The medium of embodiment B-2, wherein the operations further comprise determining the first input based on a probability value associated with one of the set of vertex categories.

B-4. The medium of any of embodiments B-2 to B-3, wherein the outcome score is a first outcome score, and wherein the program state data is in a first state before modifying the program state data, and wherein the operations further comprise: updating a neural network parameter after updating the third vertex based on the first outcome score, wherein the neural network parameter comprises a set of probability values assigned to each of a subset of vertices

of the set of vertices; determining a third input based on the neural network parameter; updating the program state data that is in the first state based on the third input; and determining a second outcome score after updating the program state data based on the third input.

B-5. The medium of any of embodiments B-1 to B-4, wherein executing the program instance comprises executing the program instance during a first iteration, and wherein the set of inputs is a first set of inputs, and wherein the outcome score is a first outcome score, and wherein the program state data is in a first state before modifying the program state data, and wherein the operations further comprise: executing the program instance during a second iteration by updating the program state data based on a second set of inputs, wherein the program state data is in the first state before updating the program state data based on the second set of inputs; determining a second outcome score based on the second set of inputs; and determining a multi-iteration score based on the first outcome score and the second outcome score.

B-6. The medium of embodiment B-5, wherein the operations further comprise: acquiring a third score; and determining a possible event based the third score using a probability distribution, wherein the probability distribution is based on the multi-iteration score.

B-7. The medium of embodiment B-6, wherein determining the possible event comprises using a neural network that is trained using inputs based on the first outcome score and the second outcome score, and wherein the neural network is trained using a training output based on the first set of inputs and the second set of inputs.

B-8. The medium of any of embodiments B-5 to B-7, wherein: the first set of inputs is associated with a first weighting value; the second set of inputs is associated with a second weighting value; and determining the multi-iteration score is based on the first weighting value and the second weighting value.

B-9. The medium of any of embodiments B-5 to B-8, the operations further comprising determining a probability distribution function based on the multi-iteration score.

B-10. The medium of any of embodiments B-1 to B-9, wherein modifying the status of the first vertex comprises determining a set of events, wherein each of the set of events satisfies a condition of the set of conditional statements.

B-11. The medium of any of embodiments B-1 to B-10, wherein acquiring the set of conditional statements comprises: acquiring an event; for a respective self-executing protocol of a plurality of self-executing protocols, determining whether the event satisfies a condition associated with the respective self-executing protocol; and acquiring the set of conditional statements

associated with the respective self-executing protocol in response to the event satisfying the condition associated with the respective self-executing protocol.

B-12. The medium of any of embodiments B-1 to B-11, wherein acquiring the set of conditional statements comprises: acquiring an entity identifier; for a respective self-executing protocol of a plurality of self-executing protocols, determining whether the entity identifier is in a respective set of entities associated with the respective self-executing protocol; and acquiring the set of conditional statements associated with the respective self-executing protocol in response to the entity identifier being in the respective set of entities associated with the respective self-executing protocol.

B-13. The medium of any of embodiments B-1 to B-12, the operations further comprising: acquiring a first entity identifier and a second entity identifier; selecting a first set of self-executing protocols from a plurality of self-executing protocols, wherein each of the first set of self-executing protocols comprises a first set of entities that comprises the first entity identifier; determining a second set of self-executing protocols from the plurality of self-executing protocols, wherein each of the second set of self-executing protocols comprises a second set of entities that comprises the second entity identifier; and determining a set of intermediary entities, wherein each of the set of intermediary entities is in a set of entities of the first set of self-executing protocols, and wherein each of the set of intermediary entities is in a set of entities of the second set of self-executing protocols.

B-14. The medium of any of embodiments B-1 to B-13, wherein modifying the status of the first vertex comprises setting a first status to indicate that a first entity fails to transfer a score to a second entity.

B-15. The medium of any of embodiments B-1 to B-14, the operations further comprising: detecting a pattern based on a plurality of the set of vertices and a plurality of the set of directed graph edges; and sending a message indicating that the pattern is detected.

B-16. The medium of any of embodiments B-1 to B-15, the operations further comprising determining a measure of central tendency based on the outcome score.

B-17. The medium of any of embodiments B-1 to B-16, the operations further comprising determining a kurtosis value based on the outcome score, wherein the kurtosis value correlates with a ratio of a first value and a second value, wherein the first value is based on a measure of central tendency, and wherein the second value is based on a measure of dispersion.

B-18. The medium of any of embodiments B-1 to B-17, the operations further comprising: acquiring an event message via an application protocol interface; determining a first set of events based on the event message, wherein the set of inputs does not include the first set of events; and updating the program state data based on the first set of events, wherein the program state data is updated based on the set of inputs after the program state data is updated with the first set of events.

B-19. The medium of any of embodiments B-1 to B-18, the operations further comprising: modifying a first status of a first vertex of the set of vertices to indicate that the first vertex is triggered; modifying a second status of a second vertex of the set of vertices to indicate that the second vertex is triggered; and in response to the first status and the second status being modified to indicate they are triggered, triggering a third vertex that is adjacent to the first vertex and the second vertex.

B-20. A method comprising: acquiring a set of conditional statements, wherein: a conditional statement of the set of conditional statements is associated with an outcome subroutine and an index value of a set of index values, and a first outcome subroutine of a first conditional statement of the set of conditional statements uses a first index value of the set of index values, wherein the first index value is associated with a second conditional statement of the set of conditional statements; executing a program instance of an application based on the set of conditional statements, wherein program state data of the program instance comprises: a set of vertices and a set of directed graph edges, wherein each of the set of vertices comprises a identifier value and is associated with one of the set of conditional statements, and wherein each of the set of directed graph edges associates a pair of the set of vertices and a direction from a tail vertex of the pair to a head vertex of the pair, a set of statuses, wherein each of the set of statuses is associated with one of the set of vertices, and a set of vertex categories, wherein each of the set of vertex categories is a category value and is associated with a respective vertex of the set of vertices and is determined based a respective conditional statement of the respective vertex, a set of scores, wherein each respective score of the set of scores is associated with a respective vertex and is based a respective conditional statement of the respective vertex; updating the program state data based on a set of inputs comprising a first input, wherein updating the program state data comprises: modifying a status of a first vertex of the set of vertices based on the first input, updating a vertex adjacent to the first vertex; and determining an outcome score based on the set of scores after updating the program state data.

B-21. A tangible, non-transitory, machine-readable medium storing instructions that when executed by one or more processors effectuate operations comprising: obtaining, with one or more processors, identifiers of a plurality of entities; obtaining, with one or more processors, a plurality of symbolic artificial intelligence (AI) models, wherein: each of the plurality of symbolic AI models is configured to produce outputs responsive to inputs based on events caused by at least one of the plurality of entities, at least some of the plurality of entities are associated with outputs of respective symbolic AI models, and at least some of the plurality of entities have respective scores corresponding to the respective outputs of the symbolic AI models; obtaining, with one or more processors, a plurality of scenarios, wherein: each scenario comprises simulated inputs corresponding to one or more simulated events, and at least some scenarios comprise a plurality of simulated inputs; determining, with one or more processors, a population of scores of a given entity among the plurality of entities, wherein respective members of the population of scores correspond to respective outputs of the plurality of symbolic AI models, and wherein the respective outputs correspond to respective scenarios among the plurality of scenarios; and storing, with one or more processors, the population of scores in memory.

B-22. The medium of embodiment B-21, wherein at least one of the plurality of symbolic AI models comprises: a set of vertices and a set of directed graph edges, wherein each of the set of vertices comprises a identifier value and is associated with one of a set of conditional statements, and wherein each of the set of directed graph edges associates a pair of the set of vertices and a direction from a tail vertex of the pair to a head vertex of the pair; a set of statuses, wherein each of the set of statuses is associated with one of the set of vertices; a set of vertex categories, wherein each of the set of vertex categories is a category value and is associated with a respective vertex of the set of vertices and is determined based a respective conditional statement of the respective vertex; and a set of scores, wherein each respective score of the set of scores is associated with a respective vertex and is based a respective conditional statement of the respective vertex.

B-23. The medium of any of embodiments B-21 to B-22, wherein obtaining the plurality of scenarios comprises: determining a first simulated input for a first model of the plurality of symbolic AI models based on a multi-iteration score associated with the first model, wherein the first model is in a first state before updating the first model based on the first simulated input; update the first model based on the first simulated input to advance the first model to a second state, wherein the second state is different from the first state; determine a second input,

wherein the second input may be selected based on scores associated with each of a set of possible states associated with the first state; update the first model when it is in the second state based on the second input to advance the second model to a third state, wherein the third state is different from the first state and the second state, and wherein the third state satisfies a terminal state criterion, and wherein a terminal state value is associated with the third state; and update the score associated with the first model based on the terminal state value; and determining a scenario of the plurality of scenarios based on the score.

B-24. The medium of embodiment B-23, wherein determining a first set of simulated inputs comprises determining the first set of inputs based on a first term and a second term, wherein the first term is based on a count of simulations executed that started from the first state and the second term is based on a score value associated with the third state.

B-25. The medium of any of embodiments B-21 to B-24, wherein determining the population of scores comprises using a convolutional neural network to determine a respective score based on values in a respective model of the symbolic AI models.

B-26. The medium of any of embodiments B-21 to B-25, the operations further comprising: fuzzifying the population of scores to provide a set of fuzzified inputs, wherein fuzzifying the outputs comprises using a membership function to determine a degree of membership, and wherein the fuzzified inputs comprises the degree of membership; determine a fuzzified outcome score based on the degree of membership using an inference engine, wherein the inference comprises a set of executable rules that may be matched to the fuzzified inputs; and determine a label associated with a smart contract based on the fuzzified outcome score.

B-27. The medium of any of embodiments B-21 to B-26, wherein obtaining the plurality of scenarios comprises: determining a first scenario for a first symbolic AI model of the plurality of AI models based on a first set of weights corresponding to each of a set of categories, wherein the first symbolic AI model comprises a first plurality of the set of categories; and determining a second scenario for a second symbolic AI model of the plurality of AI models based on the first set of weights, wherein the second symbolic AI model comprises a second plurality of the set of categories.

B-28. The medium of any of embodiments B-21 to B-27, wherein determining the simulated input comprises using a decision tree, wherein the decision tree comprises a first tree node and a second tree node, and wherein the first tree node is associated with a first score, and wherein the first tree node is associated with a second score and wherein the operations further

comprise: determining whether the first score is greater than a second score; and in response to the first score being greater than the second score, determining the simulated input based on a value associated with the first tree node.

B-29. The medium of any of embodiments B-21 to B-28, the operations further comprising updating a set of parameters of a neural network based on the population of scores, wherein the neural network provides a weighting value associated with a decision to cancel a self-executing protocol.

B-30. The medium of embodiment B-29, wherein determining the population of scores of a given entity among the plurality of entities comprises determining a sum of the scores.

B-31. A method to perform any of the operations of embodiments B-21 to B-30.

B-32. A method to perform any of the operations of embodiments B-1 to B-19.

B-33. A system, comprising: one or more processors; and memory storing instructions that when executed by the processors cause the processors to effectuate operations comprising: the operations of any one of embodiments B-1 to B-19.

B-34. A system, comprising: one or more processors; and memory storing instructions that when executed by the processors cause the processors to effectuate operations comprising: the operations of any one of embodiments B-21 to B-30.

C-1. A tangible, non-transitory, machine-readable medium storing instructions that, when executed by a computer system, effectuate operations comprising: obtaining, with a computer system, program state of a smart contract, wherein the program state encodes a directed graph, the directed graph comprising: a set of vertices, and a set of directed edges connecting respective pairs of vertices among the set of vertices, wherein the program state includes a set of conditional statements and a set of entities; obtaining, with the computer system, a request encoding a set of conditional statement parameters corresponding to an amendment to the smart contract; determining, with the computer system, a first subset of vertices in the directed graph, wherein each respective vertex of the first subset causes a state change of the program state in response to a respective conditional statement associated with the respective vertex being satisfied; selecting, with the computer system, a second subset of the first subset based on the set of conditional statement parameters encoded in the request; determining, with the computer system, a set of selected entities based on the second subset; determining, with the computer system, whether a set of criteria associated with the set of selected entities is satisfied; updating, with the computer system, the set of conditional statement parameters based on the set of

conditional statements in response to a determination that the set of criteria associated with the set of selected entities is satisfied; updating, with the computer system, the second subset based on the updated set of conditional statements; and storing, with the computer system, the program state in storage memory after updating the second subset.

C-2. The medium of embodiment C-1, wherein: the vertices are norm vertices; the first subset is a set of active vertices; the second subset is a set of target vertices; the set of entities include parties to the smart contract; and updating the set of conditional statements comprises: determining an affected conditional statement based on the request, wherein the set of target vertices comprises a reference that is associated with the affected conditional statement; setting an indicator associated with the affected conditional statement to indicate that the affected conditional statement is deprecated; generating a new conditional statement based on the set of conditional statement parameters; and setting the reference to be associated with the new conditional statement.

C-3. The medium of any of embodiments C-1 to C-2, wherein: the directed graph of the program state is stored in persistent memory and comprises a first serialized array of vertices and a second serialized array of vertices; a target vertex of the set of target vertices is a vertex of the first serialized array of vertices and not a vertex of the second serialized array of vertices; and updating the set of target vertices comprises: deserializing the first serialized array of vertices to generate a first deserialized directed graph in a non-persistent memory, wherein the first serialized array of vertices comprises an identifier of the target vertex, and wherein the first deserialized directed graph comprises an adjacency matrix or an adjacency list, and wherein the second serialized array of vertices is not concurrently deserialized, and serializing the first deserialized directed graph in the non-persistent memory to determine an updated first serialized array of vertices, and storing the updated first serialized array of vertices in the persistent memory.

C-4. The medium of any of embodiments C-1 to C-3, wherein the request comprises an entity identifier, and wherein the operations further comprises adding the entity identifier to a set of entities associated with the program state.

C-5. The medium of any of embodiments C-1 to C-4, wherein: the request comprises a first entity identifier associated with a first entity; a first conditional statement of the set of conditional statements comprises a condition that a second entity allocate a resource to a third entity, wherein a second entity identifier is associated with the second entity; and the operations

further comprises: determining an entity field used by the first conditional statement, wherein the entity field comprises the second entity identifier, and updating the entity field to comprise the first entity identifier.

C-6. The medium of any of embodiments C-1 to C-5, wherein updating the set of target vertices comprises deleting a first directed edge from the set of directed edges, wherein the first directed edge associates a target vertex of the set of target vertices with a second vertex of the directed graph.

C-7. The medium of any of embodiments C-1 to C-6, the operations further comprising: obtaining a set of confirmation messages from each entity of the set of selected entities, wherein the set of confirmation messages comprises a set of passkey values, and wherein each respective passkey value of the set of passkey values is associated with a respective entity of the set of selected entities; and wherein determining that the set of criteria associated with the set of selected entities is satisfied comprises determining that each respective passkey value of the set of passkey values matches with a respective stored passkey value of a set of stored passkey values.

C-8. The medium of any of embodiments C-1 to C-7, wherein the operations further comprise: determining a first graph structure based on the request, wherein the first graph structure comprises the set of directed edges and a set of logical categories associated with each vertex of the first graph structure; determining whether the first graph structure is identical to a second graph structure of the directed graph; and in response to a determination that the first graph structure is different from the second graph structure with respect to a number of vertices or number of edges, adding a new vertex to the set of vertices based on the request in persistent memory.

C-9. The medium of any of embodiments C-1 to C-8, the operations further comprising: determining a simulated modified program state, wherein determining the simulated modified program state comprises updating a version of the set of conditional statements, and wherein the simulated modified program state comprises the version of the set of conditional statements; determining a set of simulated events, wherein the set of simulated events are determined occur in sequence based on an associated set of occurrence times; and determining a set of outcome program states based on the simulated modified program state and the set of simulated events.

C-10. The medium of any of embodiments C-1 to C-9, wherein the request is a first request of a plurality of requests, the operations further comprising: determining a set of simulated events;

determining a set of outcome scores, wherein determining the set of outcome scores comprises, for each respective request of the plurality of requests: determining a respective simulated program state, wherein determining the respective simulated program state comprises updating a version of the set of conditional statements of the program state, and wherein the respective simulated program state comprises the version of the set of conditional statements, determining a respective set of outcome program states based on the respective simulated program state and the set of simulated events, and determining a respective set of outcome scores based on the respective set of outcome program states, wherein the respective set of outcome scores are part of the set of outcome scores; and selecting the first request from the plurality of requests based on the set of outcome scores.

C-11. The medium of any of embodiments C-1 to C-10, wherein the request comprises data encoding a second directed graph, and wherein updating the set of target vertices comprises updating a target vertex of the set of target vertices based on the second directed graph, and wherein an event that triggers a condition of the target vertex causes a second vertex of the second directed graph be set as triggerable.

C-12. The medium of any of embodiments C-1 to C-11, wherein the request comprises an identifier associated with a second directed graph of a second smart contract program, and wherein a target vertex of the set of target vertices is associated with a first priority category value, and wherein the operations further comprise: updating a set of directed edges based on a edge associating the target vertex with a second vertex of the second directed graph; and assigning a second priority category value to the second vertex, wherein: the first priority category value is different from the second priority category value, a first event triggers both the target vertex and the second vertex, and the order by which the target vertex and the second vertex is triggered in response to the first event is determined based on the first priority category value and the second priority category value.

C-13. The medium of any of embodiments C-1 to C-12, wherein: a target vertex of the set of target vertices is associated with a first priority category value; the request comprises a second priority category value; and updating the target vertex comprises changing the first priority category value to the second priority category value.

C-14. The medium of any of embodiments C-1 to C-13, wherein the request is a first request, and wherein the operations further comprise: determining that a previous request failed to satisfy the set of criteria associated with the set of selected entities is satisfied; and determining

the first request based on the previous request by modifying a conditional statement parameter encoded in the previous request.

C-15. The medium of any of embodiments C-1 to C-14, wherein the request comprises a set of entity identifiers; determining whether the set of criteria is satisfied comprises determining whether the set of entity identifiers comprises any entities of set of prohibited entities or entities of a set of prohibited entity types.

C-16. The medium of any of embodiments C-1 to C-15, the operations further comprising: obtaining a previous request; determining a simulated modified program state, wherein determining the simulated modified program state comprises updating a version of the set of conditional statements based on the previous request, and wherein the simulated modified program state comprises the version of the set of conditional statements; determining a simulated event; determining that the simulated event causes an allocation of a non-duplicable asset from a first entity to a second entity based on the simulated event and an allocation of the non-duplicable asset from the first entity to a third entity based on the simulated event; and generating a message indicating that the previous request may cause a contradiction in response to a determination that the simulated event causes the allocation of the non-duplicable asset from the first entity to the second entity and the third entity based on the simulated event.

C-17. The medium of any of embodiments C-1 to C-16, wherein determining whether the set of criteria associated with the set of selected entities is satisfied the operations further comprising: storing the set of criteria of the set of selected entities at a storage memory accessible to a first computing device of a distributed computing platform; determining whether the set of criteria associated with the set of selected entities is satisfied using the first computing device; and in response to a determination that the set of criteria associated with the set of selected entities is satisfied using the first computing device, modifying a version of the program state based on the request using a second computing device of the distributed computing platform.

C-18. The medium of any of embodiments C-1 to C-17, wherein: updating the set of conditional statements comprises adding a new conditional statement to the set of conditional statements, wherein the new conditional statement is indexed by a conditional statement identifier; and updating the set of target vertices comprises associating a target vertex of the set of target vertices with the conditional statement identifier.

C-19. The medium of any of embodiments C-1 to C-18, wherein the request comprises a first amount, the operations further comprise: determining whether a target vertex of the set of target vertices is associated with a conditional statement that was triggered by a past event, wherein an outcome of the conditional statement is associated with a transaction between a first entity and a second entity, and wherein the transaction is associated with a score equal to a second amount; determining a difference between the first amount and the second amount; and initiating a transaction between the first entity and the second entity based on the difference.

C-20. The medium of any of embodiments C-1 to C-19, the operations comprising steps for amending the smart contract while the smart contract is active.

C-21. The medium of any of embodiments C-1 to C-20, the operations comprising: steps for amending the smart contract in a manner that affects only a subset of parties to the smart contract.

C-22. A system comprising: one or more processors; and memory storing instructions that, when executed by at least one of the one or more processors, causes at least one of the one or more processors to effectuate any of the operations of embodiments C-1 to C-21.

C-23. A method to perform the operations of any of the embodiments C-1 to C-21.

D-1. A tangible, non-transitory, machine-readable medium storing instructions that when executed by one or more processors effectuate operations comprising: obtaining, with a computer system, a first directed graph of a first program state of a symbolic artificial intelligence (AI) model, wherein: the first directed graph comprises a first set of vertices and a set of directed edges, the first directed graph encodes a set of conditions and is associated with a first entity and a second entity, the conditions being conditional statements, each respective vertex of the first set of vertices is associated with a status among a set of types of status, wherein the set of types of status comprises a first status indicating that the respective vertex is satisfied, a second status indicating that the respective vertex is failed, and a third status indicating that the respective vertex is not satisfied but satisfiable, and each respective vertex of the first set of vertices is categorized in a vertex category among a set of vertex categories, the set of vertex categories comprising a first category, and the first set of vertices comprises a first vertex and a second vertex, wherein the first vertex is categorized as being of the first category, and wherein: a directed edge associates with first vertex with the second vertex, and the first vertex is associated with a first condition satisfiable by a first event caused by the first entity, and wherein a change of status of the first vertex from the third status causes the status

of the second vertex to be changed to the third status; simulating, with the computer system, evolving program state of the symbolic AI model from the first program state by evaluating conditions of the set of conditions of the first directed graph to form a second directed graph, wherein: the second directed graph comprises a second set of vertices, and wherein a third vertex of the second set of vertices is associated with a second condition satisfiable by a second event caused by the second entity, simulating comprises determining a set of action values of the first entity based on the second directed graph by: determining a set of reward values based on a second set of conditions associated with the second directed graph, wherein each of the set of reward values is associated with a vertex of the second directed graph, and determining the set of action values based on the set of reward values and a set of paths starting from the first vertex to a terminal vertex; and determining and storing in memory, with the computer system, an outcome program state based on the set of action values, wherein the outcome program state is different from the first program state.

D-2. The medium of embodiment D-1, wherein evolving from the first program state further comprises: determining a set of active vertices of the first set of vertices, wherein each respective active vertex of the set of active vertices may be triggered by an action of the first entity to triggers an adjacent vertex of the respective active vertex; and determining a first set of child vertices, wherein each respective child vertex of the first set of child vertices is adjacent to one respective active vertex of the set of active vertices, wherein the second set of vertices comprises the first set of child vertices.

D-3. The medium of any of embodiments D-1 to D-2, wherein evolving from the first program state further comprises evolving from the first program state to each of a set of terminal program states over a plurality of simulated state evolutions, wherein each respective simulated state evolution determines a respective path through the second directed graph that ends at a terminal program state, and wherein the plurality of simulated state evolutions provide a plurality of paths such that each terminal vertex of the second directed graph is in at least one of the plurality of paths.

D-4. The medium of any of embodiments D-1 to D-3, wherein determining the set of action values further comprises: determining an initial set of actions using a trained neural network based on the first directed graph, wherein each respective action of the initial set of actions is performable by the first entity and is associated with a respective score of an initial set of action values determined by the trained neural network; and iteratively traversing the second directed

graph based on the initial set of actions and the initial set of action values using a tree search operation to determine the set of action values.

D-5. The medium of embodiment D-4, wherein iteratively traversing the second directed graph comprises: determining a first heuristic value based on the first category, wherein the first heuristic value is associated with satisfaction of the first vertex; determining a second heuristic value based on the first category, wherein the second heuristic value is associated with failure of the first vertex; and determining whether the first vertex is satisfied based on the first heuristic value and the second heuristic value, wherein the first heuristic value is associated with a greater probability of selection than the second heuristic value.

D-6. The medium of any of embodiments D-1 to D-5, wherein the outcome program state is a first predicted outcome program state, the operations further comprise: determining a set of outcome program states comprising the first predicted outcome program state; obtaining an event performed by the second entity, wherein the second event causes the program state to change to an actual outcome program state; determining whether an unexpected event threshold is satisfied based on whether the actual outcome program state is not in the set of outcome program states; and sending a message indicating that the unexpected event threshold is satisfied.

D-7. The medium of any of embodiments D-1 to D-6, wherein the operations further comprise: determining a transaction graph based on a set of smart contract programs, wherein a first transaction graph vertex of the transaction graph is associated with the first entity, and wherein a second transaction graph vertex of the transaction graph is associated with the second entity; determining a transaction path between the first entity and the second entity; determining an inter-entity score based on the transaction path; and wherein determining the outcome program state comprises determining the outcome program state based on the inter-entity score.

D-8. The medium of embodiment D-7, wherein the operations further comprise: determining that the transaction path is a cyclical path, wherein the cyclical path comprises a set of transaction graph edges that are connected and begin and end at a same transaction graph vertex; and storing a value indicating that the transaction path is cyclical.

D-9. The medium of any of embodiments D-7 to D-8, wherein determining the transaction graph comprises: traversing a set of directed graphs of the set of smart contract programs to determine a set of score changes between a pair of entities of the transaction graph; and

updating, for each of the set of score changes between the pair of entities, a transaction graph edge associating the pair of entities.

D-10. The medium of any of embodiments D-7 to D-9, wherein the operations further comprise: determining a set of smart contract programs associated with the first transaction graph vertex, wherein each smart contract program of the set of smart contract programs is determined to cause a score change for the first entity based on conditions of the set of smart contract programs; and determining a set of contribution weights, wherein each respective contribution weight of the set of contribution weights is associated with a respective smart contract program of the set of smart contract programs, and wherein each respective contribution weight is correlated with a ratio by which the respective smart contract program contributes to a net score change of the first entity.

D-11. The medium of any of embodiments D-7 to D-10, wherein determining the transaction path between the first entity and the second entity the operations further comprise: obtaining a transaction path threshold; determining whether a set of transaction graph edges of the transaction graph satisfies the transaction path threshold; and responsive to a determination that the set of transaction graph edges satisfies the transaction path threshold, determining the transaction path between the first entity and the second entity based on the set of transaction graph edges.

D-12. The medium of any of embodiments D-1 to D-11, and wherein satisfying a failure threshold of the vertex causes the vertex to be associated with the second status, and wherein the operations further comprise: determining whether the second entity is associated with a third event caused by the second entity that resulted in a fourth vertex being associated with the second status, wherein the fourth vertex is associated with the first category; and responsive to the second entity being associated with the third event caused by the second entity that resulted in the fourth vertex being associated with the second status, reduce a reward value determined from a score transfer from the second entity to the first entity.

D-13. The medium of any of embodiments D-1 to D-12, wherein the operations further comprise categorizing one or more quantitative values of the first program state before determining the set of action values.

D-14. The medium of any of embodiments D-1 to D-13, the operations further comprising determining whether the first vertex should be associated with the second status based on determining whether a time point satisfies a failure time threshold.

D-15. The medium of any of embodiments D-1 to D-14, wherein determining the set of action values of the first entity comprises using an intelligent agent, wherein the intelligent agent comprises: a set of stored parameters; a first routine to update the set of stored parameters one or more times; a second routine to determine action values based on the set of stored parameters and on the first program state.

D-16. The medium of embodiment 15, wherein using the intelligent agent comprises: determining a first path score associated with a first path from the first vertex to a terminal vertex of the second directed graph; determining a first weight based on reaching the first program state from an initial program state; determining an intermediate program state based on an event that changes a status of the first vertex to the first status or the second status; determining a second weight based on reaching a terminal program state from the intermediate program state; determining a counterfactual regret value based on a summation comprising a product of the first path score, the first weight, and the second weight; and determining the set of action values based on the counterfactual regret value.

D-17. The medium of any of embodiments D-1 to D-16, wherein determining the set of reward values based on the second set of conditions comprises: determining a threshold value to satisfy the first vertex based on a condition associated with the first vertex; and determining a first reward value based on the threshold value, wherein the first reward value is associated with the first vertex.

D-18. The medium of any of embodiments D-1 to D-17, wherein obtaining the first directed graph of the first program state further comprises: determining whether the first program state is different from a previous program state; and responsive to a determination that the first program state is different from the previous program state, obtaining the first directed graph of the first program state.

D-19. The medium of any of embodiments D-1 to D-18, wherein determining the second directed graph comprises: varying a first set of threshold values of the set of conditions to determine a set of modified values; obtaining an initial set of events causable by the first entity based on the set of modified values; determining a set of possible events based on the initial set of events causable by the first entity using a first trained neural network, wherein the first trained neural network is trained using a set of self-play operations; and determining the second directed graph based on the set of possible events.

D-20. The medium of any of embodiments D-1 to D-19, wherein evolving from the first program state comprises: obtaining a failure penalty; wherein the second directed graph comprises a first path that includes a vertex having the second status; and modifying an action value associated with the first path based on the failure penalty.

D-21. The medium of any of embodiments D-1 to D-20, wherein the operations further comprise: determining an outcome score associated with the outcome program state; determining whether the outcome score satisfies an outcome score threshold; and responsive to the outcome score satisfying the outcome score threshold, generate an alert indicating that the outcome score threshold is satisfied.

D-22. The medium of any of embodiments D-1 to D-21, wherein determining the action value further comprises determining an action value based on an entity property associated with the first entity or the second entity.

D-23. The medium of any of embodiments D-1 to D-22, the operations further comprising determining the set of paths by performing a set of simulated state evolutions, wherein performing each respective simulated state evolution of the set of simulated state evolutions comprises: determining a set of computed values using a random, pseudorandom, or quasi-random number generation method; and selecting a respective subset of vertices of a respective path based on the set of computed values.

D-24. The medium of embodiment 23, wherein a selectable vertex in the second directed graph is not in any of the subsets of vertices determined from the set of simulated state evolutions.

D-25. The medium of any of embodiments D-1 to D-24, wherein simulating program state evolution comprises simulating program state evolution for at least 10,000 iterations from an initial program state to a terminal program state.

D-26. The medium of any of embodiments D-1 to D-25, wherein a count of vertices in the first directed graph is greater than 1000 vertices.

D-27. The medium of any of embodiments D-1 to D-26, the operations further comprising a means of determining the outcome program state.

D-28. The medium of any of embodiments D-1 to D-27, wherein the operations further comprise: determining a transaction graph based on a set of smart contract programs, wherein a first transaction graph vertex of the transaction graph is associated with the first entity, and wherein a second transaction graph vertex of the transaction graph is associated with the second

entity; determining a transaction path between the first entity and the second entity; determining an inter-entity score based on the transaction path; and wherein determining the outcome program state comprises determining the outcome program state based on the inter-entity score.

D-29. The medium of any of embodiments D-1 to D-28, wherein the operations further comprise: determining a plurality of outcome states; and determining a population score based on the plurality of outcome states.

D-30. A tangible, non-transitory, machine-readable medium storing instructions that when executed by one or more processors effectuate operations comprising: obtaining, with one or more processors, a symbolic artificial intelligence (AI) model, wherein the symbolic AI model is configured to produce an outcome state responsive to an input based on events, wherein at least some of the events are caused by a first entity or a second entity; obtaining, with one or more processors, a first scenario and a second scenario, wherein the first scenario causes the failure of a condition associated with a norm of the symbolic AI model, and wherein the second scenario satisfies the condition associated with the norm of the symbolic AI model; obtaining, with one or more processors, a failure penalty value; determining, with one or more processors, a first outcome state based on the symbolic AI model, the first scenario, and the failure penalty value; determining, with one or more processors, a second outcome state based on the symbolic AI model and the second scenario; determining, with one or more processors, an outcome score based on the first outcome state and the second outcome state; and storing, with one or more processors, the outcome score in memory.

D-31. A method to perform the operations of any of the embodiments D-1 to D-30.

D-32. A system comprising: one or more processors; and memory storing instructions that, when executed by at least one of the one or more processors, causes at least one of the one or more processors to effectuate any of the operations of embodiments D-1 to D-30.

E-1. A tangible, non-transitory, machine-readable medium storing instructions that, when executed by a computing system, effectuate operations comprising: obtaining, with a computing system, program state of a self-executing protocol, wherein the program state encodes: a set of conditional statements; a set of entities, wherein the set of entities comprises a first entity; a directed graph, the directed graph comprising: a set of vertices, wherein each respective vertex of the set of vertices is associated with a respective category label of a set of mutually exclusive categories; a set of directed edges connecting respective pairs of vertices

among the set of vertices; obtaining, with the computing system, an entity profile of the first entity, wherein: the entity profile comprises a first graph portion template, the first graph portion template comprises a first vertex template and an edge template, the first vertex template is associated in memory with a first category label of the set of mutually exclusive category labels, and the edge template specifies an edge direction to or from a vertex matching the first vertex template; determining, with the computing system, whether the first graph portion template matches a graph portion in the directed graph based on a first vertex of the directed graph matching the first vertex template and a first directed edge of the directed graph matching the edge template; determining, with the computing system, an outcome score based on the first graph portion template matching the graph portion in the directed graph; determining, with the computing system, whether the outcome score satisfies an outcome score threshold; and in response to the outcome score satisfying the outcome score threshold, storing, with the computing system, a value indicating that the outcome score satisfies the outcome score threshold.

E-2. The medium of embodiment E-1, wherein: the set of vertices are a set of norm vertices; the first vertex is a first norm vertex; the set of entities include parties to the self-executing protocol; the operations further comprising: obtaining a plurality of self-executing protocol programs comprising a plurality of directed graphs, wherein each respective directed graph of the plurality of directed graphs is associated with a respective set of entities that comprises the first entity; determining the first graph portion template based on the plurality of directed graphs, wherein a second norm vertex of the plurality of directed graphs matches the first norm vertex template of the first graph portion template, and wherein a condition of the second norm vertex is indicated to have been failed by the first entity based on an event message; and determining an outcome determination parameter based on a number of times that the first graph portion template matches with a respective graph portion in the plurality of self-executing protocol programs, wherein determining the outcome score comprises determining the outcome score based on the outcome determination parameter.

E-3. The medium of any of embodiments E-1 to E-2, the operations further comprising: obtaining a plurality of self-executing protocol programs comprising a plurality of directed graphs, wherein each respective self-executing protocol program of the plurality of self-executing protocol programs comprises a respective directed graph of the plurality of directed graphs; determining the first graph portion template based on the plurality of self-executing protocol programs, wherein a second vertex of a second directed graph of the plurality of

directed graphs matches the first vertex template, and wherein a third vertex of the plurality of directed graphs matches a second vertex template, and wherein a condition of the third vertex is indicated as having been satisfied based on an event message; and determining an outcome determination parameter based on a number of times that the first graph portion template matches with a respective graph portion in the plurality of self-executing protocol programs, wherein determining the outcome score comprises determining the outcome score based on the outcome determination parameter.

E-4. The medium of any of embodiments E-1 to E-3, wherein the entity profile is a first entity profile, and wherein the operations further comprise: determining a transaction score based on the directed graph, wherein the transaction score is associated with a transaction between the first entity and a second entity; and updating an association between the first entity profile and a second entity profile based on the transaction score, wherein the second entity profile is associated with the second entity.

E-5. The medium of any of embodiments E-1 to E-4, the operations further comprising: determining whether the first entity has failed a conditional statement associated with a second vertex of the directed graph; and in response to a determination that the first entity has failed the conditional statement, updating an entity score of an entity graph, wherein the entity score is associated with the first entity, and wherein the entity graph comprises a plurality of entity vertices, and wherein each respective entity vertex of the plurality of entity vertices is associated with a respective entity profile.

E-6. The medium of embodiment E-5, wherein the entity graph is stored on a distributed, tamper-evident ledger, and wherein updating the entity score comprises: obtaining an encryption key associated with the first entity; obtaining a previous entity score from the distributed, tamper-evident ledger based on the encryption key; and updating the entity score based on the previous entity score.

E-7. The medium of any of embodiments E-5 to E-6, wherein the entity graph is stored on a distributed, tamper-evident ledger, and wherein the operations further comprise: determining whether the entity score satisfies an entity score threshold of a verification entity; and in response to the entity score satisfying the entity score threshold, storing an indicator that the first entity satisfies the entity score threshold of the verification entity.

E-8. The medium of any of embodiments E-5 to E-7, the operations further comprising: determining whether the entity score satisfies an entity score threshold of a verification entity;

and sending a message to an application program interface, wherein the message indicates that the first entity satisfies the entity score threshold of the verification entity.

E-9. The medium of any of embodiments E-5 to E-8, wherein the entity profile is a first entity profile, and wherein the operations further comprise: determining a second entity score associated with the first entity, wherein the first entity profile does not comprise the second entity score; obtaining a passkey value; and in response to receiving the passkey value, sending a message comprising the second entity score.

E-10. The medium of any of embodiments E-1 to E-9, wherein determining the outcome score comprises determining the outcome score using a neural network based on a feature set, wherein: determining the feature set, wherein determining the feature set comprises determining whether the first graph portion template matches a graph portion in the directed graph; and the neural network is trained on a plurality of directed graphs of a plurality of a self-executing protocol programs, wherein the first graph portion template matches a graph portion of a subset of the plurality of directed graphs.

E-11. The medium of any of embodiments E-1 to E-10, wherein determining the outcome score comprises: generating a set of embeddings based on a set of vertices of the directed graph, wherein each vertex of the set of vertices is associated with an embedding of the set of embeddings, and wherein each embedding comprises a vector; determining a feature set based on the set of embeddings; and determining the outcome score using a neural network based on the feature set.

E-12. The medium of any of embodiments E-1 to E-11, wherein the entity profile is a first entity profile and the outcome score is a first outcome score, and wherein the operations further comprise: obtaining a second entity profile, wherein the second entity profile is associated with a second entity, and wherein the second entity profile comprises the first graph portion template, and wherein a second outcome determination parameter is determined based on the first graph portion template; determining a second outcome score associated with the second entity profile based on the second outcome determination parameter; and selecting the first entity based on the first outcome score and the second outcome score.

E-13. The medium of any of embodiments E-1 to E-12, the operations further comprising: sampling the directed graph to determine a set of subgraphs; determining a vector based on the set of subgraphs using a skip-gram model; and determining the outcome score using a neural network based on the vector.

E-14. The medium of any of embodiments E-1 to E-13, wherein the first graph portion template further comprises a second vertex template, wherein the second vertex template is associated with a second category label of the set of mutually exclusive category labels, and wherein the second category label is different from the first category label.

E-15. The medium of any of embodiments E-1 to E-14, the operations further comprising: updating the entity profile based a history of the first entity; storing the entity profile on a centralized computing platform, wherein the entity profile is associated with an entity identifier; and updating a value associated with the entity identifier, wherein the value is stored on a distributed, tamper-evident ledger operating on a distributed computing platform.

E-16. The medium of any of embodiments E-1 to E-15, wherein the entity profile is a first entity profile, and wherein the operations further comprising: obtaining a second entity profile; determining whether a set of entity similarity criteria is satisfied based on the first entity profile and the second entity profile; and storing value indicating that the first entity profile and the second entity profile satisfy the set of entity similarity criteria.

E-17. The medium of any of embodiments E-1 to E-16, wherein the first graph portion template further comprises a second vertex template, wherein the second vertex template is not connected to the first vertex template in the first graph portion template by any edge templates.

E-18. The medium of any of embodiments E-1 to E-17, wherein the directed graph is a first self-executing protocol directed graph, and wherein the operations further comprise: determining a first transaction amount between the first entity and a second entity based on the first self-executing protocol directed graph; determining a second transaction amount between the second entity and a third entity based on a second self-executing protocol directed graph; updating a first association between the first entity and the second entity of an entity graph based on the first transaction amount; updating a second association between the second entity and the third entity of the entity graph based on the second transaction amount; and determining whether the first entity is associated with the third entity based on the first association, the first transaction amount, the second association, and the second transaction amount.

E-19. The medium of embodiment 18, the operations further comprising: determining whether the first entity has failed a conditional statement associated with the first vertex; in response to a determination that the first entity has failed the conditional statement, updating an entity score is associated with the first entity; and sending a message to the third entity in response to the updating of the entity score associated with the first entity.

E-20. A method to perform the operations of any of the embodiments E-1 to E-19.

E-21. A system comprising: one or more processors; and memory storing instructions that, when executed by at least one of the one or more processors, causes at least one of the one or more processors to effectuate any of the operations of embodiments E-1 to E-19.

F-1. A tangible, non-transitory, machine-readable medium storing instructions that, when executed by a computing system, effectuate operations comprising: obtaining, with a computing system, a set of conditions; obtaining, with the computing system, a first cross-program entity identifier of a first entity, wherein the first cross-program entity identifier is unique amongst a set of cross-program entity identifiers of a decentralized computing platform; obtaining, with the computing system, a set of directed graphs of a set of self-executing protocols comprising a first self-executing protocol and a second self-executing protocol that are executed on the decentralized computing platform, wherein: each respective self-executing protocol of the set of self-executing protocols comprises data of a respective directed graph of the respective self-executing protocol, and the first cross-program entity identifier is associated with a first program-specific entity identifier of the first self-executing protocol and a second program-specific entity identifier of the second self-executing protocol; determining, with the computing system, that the set of conditions is applicable to the first entity based on the first cross-program entity identifier; determining, with the computing system, whether the set of conditions are satisfied based on whether a graph portion associated with the set of directed graphs corresponds to a graph portion template of the set of conditions; and in response to a determination that the graph portion corresponds to the graph portion template, storing, with the computing system, an indication that the first entity violated the set of conditions in a profile of the first entity using the first cross-program entity identifier.

F-2. The medium of embodiment F-1, the operations further comprising: determining a first set of geographic locations associated with the first entity based on the first cross-program entity identifier; and determining whether the first set of geographic locations satisfies a first condition of the set of conditions based on whether the first set of geographic locations is within a geofence indicated by the first condition, wherein the indication indicates that the first entity violated the set of conditions based on whether the first set of geographic locations satisfies the first condition.

F-3. The medium of any of embodiments F-1 to F-2, the operations further comprising determining a second set of counterparty entities based on the set of self-executing protocols,

wherein each counterparty entity of the set of counterparty entities is associated with a transaction with the first entity.

F-4. The medium of any of embodiments F-1 to F-3, wherein obtaining the set of conditions comprises: obtaining a governing document; determining a set of entity categories using a natural language processing model based on governing document; and determining a condition of the set of conditions based on the set of entity categories.

F-5. The medium of any of embodiments F-1 to F-4, the operations further comprising: obtaining a governing document; selecting a section of the governing document based on a text header indicated by a set of text sizes or text spacings; and determining a condition of the set of conditions based on the section of the governing document.

F-6. The medium of any of embodiments F-1 to F-5, the operations further comprising: obtaining a first profile associated with the first cross-program entity identifier; obtaining a natural language document, wherein the natural language document comprises a verifying agent identifier and an entity name associated with the first cross-program entity identifier; using a natural language processing model to parse the natural language document to determine the verifying agent identifier and the entity name; sending a first message comprising the entity name to an application program interface (API) of a third-party entity based on the verifying agent identifier; and obtaining a second message from the third-party entity indicating that the entity name is valid and, in response, setting the first profile associated with the first cross-program entity identifier as a verified profile.

F-7. The medium of any of embodiments F-1 to F-6, the operations further comprising sending a notification message to a second entity indicating that the first entity failed the set of conditions.

F-8. The medium of any of embodiments F-1 to F-7, the operations further comprising: sending a first message comprising data of a pending transaction to a third entity, wherein a participant of the pending transaction is associated with the first cross-program entity identifier; obtaining a second message from the third entity, wherein the second message indicates that the third entity has verified the pending transaction; and in response to receiving the second message, storing a value indicating that the transaction was verified by the third entity on a distributed, tamper-evident data structure.

F-9. The medium of any of embodiments F-1 to F-8, the operations further comprising: determining, after a threshold duration of time after determining whether the set of conditions

are satisfied, whether the set of conditions are satisfied a second time; and in response to a determination that the set of conditions are satisfied, setting a value to indicate that a resource transfer or allocation of a pending transaction is permitted, wherein a participant of the pending transaction is associated with the first cross-program entity identifier.

F-10. The medium of any of embodiments F-1 to F-9, the operations further comprising: determining that a variable of the set of conditions is not stored in data of a smart self-executing protocol; compute a value for the variable using a function encoded in the set of conditions; determining whether a the value satisfies a threshold value of a first condition; and in response to a determination that the value satisfies the threshold value, storing a value indicating that the first entity satisfies the first condition to a persistent storage.

F-11. The medium of any of embodiments F-1 to F-10, the operations further comprising: obtaining an additional governing document; updating the set of conditions based on the additional governing document; and determining whether the updated set of conditions is satisfied.

F-12. The medium of any of embodiments F-1 to F-11, wherein determining whether the set of conditions is satisfied further comprises: determining a first score change of the first self-executing protocol; determining that the first score change is associated with the first entity based on an association between the first program-specific entity identifier and the first cross-program entity identifier; determining a second score change of the second self-executing protocol; determining that the second score change is associated with the first entity based on an association between the second program-specific entity identifier and the first cross-program entity identifier; and determining whether the first entity satisfies the set of conditions based on the first score change and the second score change..

F-13. The medium of any of embodiments F-1 to F-12, the operations further comprising: determine a summation based on the first score change and the second score change, wherein determining whether the set of conditions is satisfied comprises determining whether the summation satisfies a threshold value.

F-14. The medium of any of embodiments F-1 to F-13, wherein a set of entities participating in the first self-executing protocol do not have permission to view the first cross-program entity identifier and the computer system prevents such viewing responsive to the lack of permission.

F-15. The medium of any of embodiments F-1 to F-14, the operations further comprising: determining whether a first value of a transaction satisfies a warning threshold, wherein the

warning threshold is based on a condition of the set of conditions; and sending a message indicating that the warning threshold has been satisfied to the first entity.

F-16. The medium of any of embodiments F-1 to F-15, the operations further comprising: determining a hierarchy of conditions based on a set of precedence values associated with the set of conditions; determining a pair of conflicting conditions based on the set of conditions and a difference in labels between category labels of the set of conditions, wherein each category label of a respective condition of the set of conditions is one of a set of mutually exclusive category labels; and determining an overriding condition based on the hierarchy of governing conditions, wherein the overriding condition is one of the pair of conflicting conditions, and wherein the overriding condition is indicated to take precedence over the other condition of the pair of conflicting conditions.

F-17. The medium of any of embodiments F-1 to F-16, the operations further comprising: determining that a second cross-program entity identifier is associated with the first entity; determining that a condition is associated with the second cross-program entity identifier; generating an association between the first cross-program entity identifier and the second cross-program entity identifier in a database of cross-program entity identifiers; and persisting the database of cross-program entity identifiers to a persistent storage of the computing system.

F-18. The medium of embodiment F-17, the operations further comprising steps for obtaining the set of conditions.

F-19. The medium of any of embodiments F-1 to F-18, the operations further comprising steps for determining whether the set of conditions is violated.

F-20. A method to perform the operations of any of the embodiments F-1 to F-19.

F-21. A system, comprising: one or more processors; and memory storing instructions that when executed by the processors cause the processors to effectuate operations comprising: the operations of any one of embodiments F-1 to F-19.

G-1. A tangible, non-transitory, machine-readable medium storing instructions that, when executed by a computing system, effectuate operations comprising: executing, with one or more processors, an instance of an application, wherein: program state of the instance of the application comprises a set of vertices and a set of directed graph edges, each vertex of the set of vertices comprises an identifier and a category label of a set of mutually exclusive category labels, and each of the set of directed graph edges associates a pair of the set of vertices and a direction from a tail vertex of the pair to a head vertex of the pair; serializing, with one or more

processors, the set of vertices in a serialized data format and storing a record comprising the serialized set of vertices in a first persistent storage of a first computing device of a plurality of computing devices communicatively coupled to each other via a network; distributing, with one or more processors, the serialized set of vertices to a second computing device of the plurality of computing devices; deserializing, with one or more processors, the serialized set of vertices with the second computing device to generate a second instance of a directed graph, wherein the second instance of the directed graph encodes the set of vertices and the set of directed graph edges in non-serialized data format; storing, with one or more processors, the second instance of the directed graph in a second persistent storage, wherein the second persistent storage is a local persistent storage of the second computing device; receiving, with one or more processors, a message encoding a graph portion template, wherein the graph portion template comprises a first vertex template and an edge template, and wherein the first vertex template is associated with a first category label of the set of mutually exclusive category labels, and wherein the edge template specifies an edge direction to or from a vertex matching the first vertex template; querying, with one or more processors, a data structure of the second persistent storage based on the graph portion template to retrieve a subset of vertices encoded in the second instance of the directed graph; computing, with one or more processors, a response value based on the subset of vertices; and sending, with one or more processors, a response comprising the response value from the second computing device to a response destination indicated by the message.

G-2. The medium of embodiment G-1, wherein the computing the response value comprises: determining a set of entities based on the subset of vertices; determining whether a data retrieval criterion of the set of entities is satisfied; and wherein computing the response value comprises computing the response value in response to a determination that the data retrieval criterion is satisfied.

G-3. The medium of any of embodiments G-1 to G-2, wherein the message comprises instructions to update the directed graph, the operations comprising: storing an updated directed graph in the second persistent storage based on the message; determining a first update confirmation value associated with the updated directed graph; receiving a second update confirmation value at the second computing device; determining whether the first update confirmation value satisfies a set of storage update criteria based on the second update confirmation value; and in response to a determination that the first update confirmation value

satisfies the set of storage update criteria, set an indicator to indicate that the updated directed graph is valid in the second persistent storage.

G-4. The medium of any of embodiments G-1 to G-3, the operations further comprising: determining whether a graph portion of the directed graph matches a graph portion template of a library of graph portion templates; and in response to a determination that the graph portion of the directed graph matches the graph portion template, associate an index value with a first vertex of the graph portion, wherein the first vertex is retrievable with the index value.

G-5. The medium of any of embodiments G-1 to G-4, the operations further comprising: selecting a plurality of directed graphs stored in the second persistent storage based on the message; for each respective directed graph of the plurality of directed graphs, determining a respective subset of vertices associated with the respective directed graph; and wherein determining the response value based on the respective subsets of vertices of the plurality of directed graphs.

G-6. The medium of any of embodiments G-1 to G-5, the operations further comprising: detecting a state-updating event associated with an update to program state, wherein the state-updating event causes a change in the set of vertices or a status associated with the set of vertices; updating a historical sequence of records based on the state-updating event, wherein each respective vertex of the historical sequence of records is associated with a respective state-updating event; and storing the historical sequence of records in the second persistent storage, wherein each vertex of the historical sequence of records is associated with a respective version of the directed graph.

G-7. The medium of any of embodiments G-1 to G-6, wherein the graph portion template comprises a plurality of vertex templates and a plurality of directed graph edges associating the plurality of vertex templates with each other.

G-8. The medium of any of embodiments G-1 to G-7, wherein the plurality of computing devices is a first plurality of computing devices, the operations further comprising: selecting a subset of nodes from the plurality of computing devices, where each respective node of the subset of nodes corresponds with a respective computing device of the plurality of computing devices, wherein the subset of nodes is fewer in number than the total number of nodes of the plurality of computing devices; and generating, with at least one the subset of nodes, a block of a sequence of blocks stored in the first persistent storage.

G-9. The medium of any of embodiments G-1 to G-8, wherein the first computing device validates an update to program state based on a consensus voting process involving other computing devices in the plurality of computing devices.

G-10. The medium of any of embodiments G-1 to G-9, wherein the directed graph is a first directed graph, the operation further comprising: determining a predicted directed graph based on the first directed graph; and storing the predicted directed graph in association with the first directed graph.

G-11. The medium of any of embodiments G-1 to G-10, the operations further comprising validating the directed graph at the second computing device based on a signature value provided by the first computing device.

G-12. The medium of any of embodiments G-1 to G-11, the operations further comprising storing a third instance of the directed graph in a third persistent storage, wherein the third persistent storage is a persistent storage of a cloud computing server.

G-13. The medium of any of embodiments G-1 to G-12, wherein the second persistent storage is a persistent storage of a peer-to-peer data-sharing network comprising a second plurality of computing devices, wherein data stored on a first peer of the second plurality of computing devices is distributed to other peers of the second plurality of computing devices.

G-14. The medium of any of embodiments G-1 to G-13, the operations further comprising storing a natural language document in association with the directed graph in the data structure of the second persistent storage.

G-15. The medium of any of embodiments G-1 to G-14, the operations comprising steps for serializing, steps for storing, and steps for deserializing the directed graph.

G-16. The medium of any of embodiments G-1 to G-15, the operations comprising steps for updating program state of the application.

G-17. The medium of any of embodiments G-1 to G-16, the operations further comprising: generating a plurality of previous versions of the directed graph based on a sequence of blocks stored in the first persistent storage; and storing the plurality of previous versions in the second persistent storage.

G-18. The medium of any of embodiments G-1 to G-17, wherein storing the second instance of the directed graph in the second persistent storage further comprises: determining an updated

vertex of the second instance of the directed graph; and storing a score change or a set of entities associated with the updated vertex in the data structure of the second persistent storage.

G-19. The medium of any of embodiments G-1 to G-18, wherein computing the response value based on the subset of vertices comprises: determining a set of conditional statements associated with the subset of vertices; determining a set of scores based on the set of conditional statements; and determining the response value based on a sum of the set of scores.

G-20. A method to perform the operations of any of the embodiments G-1 to G-19.

G-21. A system, comprising: one or more processors; and memory storing instructions that when executed by the processors cause the processors to effectuate operations comprising: the operations of any one of embodiments G-1 to G-19.

H-1. A tangible, non-transitory, machine-readable medium storing instructions that, when executed by a computing system, effectuate operations comprising: obtaining, with the computing system, a first set of conditional statements, wherein each of the first set of conditional statements is associated with a vertex of a first directed graph of a distributed application listing a first entity and second entity as associated with the distributed application; determining, with the computing system, a subset of conditional statements from the first set of conditional statements based on a first category selected from a set of mutually exclusive categories, wherein: triggering each respective conditional statement of the subset of conditional statements causes a state associated with the respective conditional statement to be updated from an initial state to a different state; each of the subset of conditional statements is associated with the first category; and each of the subset of conditional statements is indicated to be triggered based on a first event, wherein the first event comprises a value indicating a resource amount; generating, with the computing system, an integrated test condition based on the subset of conditional statements, wherein the integrated test condition is associated with a shared resource type and a numeric value; obtaining, with the computing system, a second directed graph, wherein the second directed graph is associated with the first entity; determining, with the computing system, a simulated event based on a first conditional statement of the second directed graph; determining, with the computing system, whether the simulated event triggers the integrated test condition; and storing, with the computing system, a result indicating that the simulated event triggers the integrated test condition in response to a determination that the simulated event triggers the integrated test condition.

H-2. The medium of embodiment H-1, wherein determining the simulated event comprises simulating adding an additional vertex and directed edge to the first directed graph, wherein the additional vertex is associated with a parent vertex via the additional directed edge, wherein the parent vertex is associated with the first conditional statement.

H-3. The medium of any of embodiments H-1 to H-2, wherein determining the simulated event comprises simulating a sequence of simulated events, wherein each respective simulated event of the sequence of simulated events is indicated to at a different time.

H-4. The medium of any of embodiments H-1 to H-3, the operations further comprising determining that the simulated event triggers the integrated test condition, wherein the triggering of the integrated test condition causes the integrated test condition to indicate a specified conditional statement of the subset of conditional statements as failed.

H-5. The medium of any of embodiments H-1 to H-4, the operations further comprising sending a message to the first entity in response to a determination that the simulated event triggered the integrated test condition, wherein the message indicates that the simulated event triggered the integrated test condition.

H-6. The medium of any of embodiments H-1 to H-5, wherein: the first directed graph are associated with a first smart contract associated with a first set of entities; each of the first set entities is associated with a conditional statement of one or more vertices of the first directed graph; the second directed graph is associated with a second smart contract associated with a second set of entities; each of the second set entities is associated with a conditional statement of one or more vertices of the second directed graph; and the first set of entities is different from the second set of entities.

H-7. The medium of any of embodiments H-1 to H-6, the operations further comprising partitioning a parent directed graph into the first directed graph and the second directed graph.

H-8. The medium of any of embodiments H-1 to H-7, wherein the first directed graph is associated with a first natural language document, the second directed graph is associated with a second natural language document, and the operations further comprise: visually indicating a first text section of the first natural language document, wherein the first text section is associated with a parent vertex, wherein the parent vertex is associated with the conditional statement; and in response to a determination that the event triggers the second conditional statement of the second directed graph, visually indicating a second section of the

second natural language document, wherein the first text section is associated with the parent vertex.

H-9. The medium of any of embodiments H-1 to H-8, the operations further comprising retrieving the second directed graph based on data associated with the first entity, wherein the event is caused by the first entity.

H-10. The medium of any of embodiments H-1 to H-9, wherein the first directed graph was previously updated from a previous version of the first directed graph, wherein the updated first directed graph comprises an active vertex that was not active in the previous version of the first directed graph, the operations further comprising: determining whether the first directed graph was updated; and in response to a determination that the first directed graph was update.

H-11. The medium of any of embodiments H-1 to H-10, wherein generating the integrated test condition comprises: determining a set of values associated with the shared resource type, wherein each of the set of values is used by at least one of the subset of conditional statements; and setting a test condition threshold as a maximum or minimum value of the set of values, wherein the integrated test condition comprises the test condition threshold.

H-12. The medium of any of embodiments H-1 to H-11, wherein the integrated test condition is a first integrated test condition, and wherein the subset of conditional statements is a second subset of conditional statements, the operations further comprising: obtaining a third set of conditional statements, wherein each conditional statement of the third set of conditional statements is associated with a vertex of a third directed graph; determining a second subset of conditional statements from the third set of conditional statements based on a second category selected from the set of mutually exclusive categories; generating a second integrated test condition based on the second subset of conditional statements, wherein the second integrated test condition; and determining whether the simulated event triggers the second integrated test condition.

H-13. The medium of any of embodiments H-1 to H-12, wherein determining the simulated event comprises determining the simulated event based on an entity identifier associated with the first entity.

H-14. The medium of any of embodiments H-1 to H-13, wherein: the first entity is associated with a first entity role; determining the simulated event comprises determining the simulated

event based on an entity identifier associated with a third entity; and the third entity is not associated with the first entity role.

H-15. The medium of any of embodiments H-1 to H-14, wherein the integrated test condition comprises a time threshold, and wherein determining whether the simulated event satisfies the integrated test condition comprises determining whether a time value of the simulated event satisfies the time threshold.

H-16. The medium of any of embodiments H-1 to H-15, the operations further comprising determining whether a simulated outcome state caused by the simulated event indicates an activation of a second conditional statement, wherein the second conditional statement is triggered by a subsequent event, and wherein the subsequent event satisfies a third conditional statement of the first directed graph or the second directed graph.

H-17. The medium of any of embodiments H-1 to H-16, the operations further comprising: determining whether the first set of conditional statements has been updated; updating the subset of conditional statements based on the updated first set of conditional statements in response to a determination that the first set of conditional statements has been updated; updating the integrated test condition based on the updated subset of conditional statements; and determining whether the simulated event triggers the updated integrated test condition.

H-18. The medium of any of embodiments H-1 to H-17, the operations further comprising steps for determining the simulated event.

H-19. The medium of any of embodiments H-1 to H-18, the operations further comprising steps for determining whether the simulated event triggers the integrated test condition.

H-20. A method to perform the operations of any of the embodiments H-1 to H-19.

H-21. A system, comprising: one or more processors; and memory storing instructions that when executed by the processors cause the processors to effectuate operations comprising: the operations of any one of embodiments H-1 to H-19.

I-1. A tangible, non-transitory, machine-readable medium storing instructions that, when executed by a computing system, effectuate operations comprising: determining, with a computer system, a set of features associated in memory of the computer system with a set of vertices of a first directed graph, wherein a feature of the set of features is associated in memory of the computer system with a category type comprising a set of mutually exclusive categories; obtaining, with the computer system, a set of feature values associated with the set

of vertices, wherein each respective vertex of set of vertices is associated with a respective subset of feature values, wherein: each feature value is associated with a feature of the set of features, and the respective subset of feature values comprise a respective category of the set of mutually exclusive categories; selecting, with the computer system, a first subset of features based on the set of feature values, wherein the selecting comprises: determining a plurality of candidate subsets of features; determining, with the computer system, a plurality of feature subset scores associated with the plurality of candidate subsets of features based on a category label selected from the set of mutually exclusive categories and the set of feature values; and selecting the first subset of features based on the plurality of feature subset scores; performing, with the computer system, a first operation to determine a set of extracted feature values, the first operation comprising: determining a set of input values by increasing a set of feature values associated with the first subset of features with a set of weights; and determining, with the computer system, the set of extracted feature values based on the set of input values, wherein the set of extracted feature values comprises a first multidimensional vector associated with the first directed graph; determining, with the computer system, a metric based on a distance between the first multidimensional vector and a second multidimensional vector of a second directed graph; determining, with the computer system, whether the metric satisfies a first threshold; and storing, with the computer system, the metric in persistent storage.

I-2. The medium of embodiment I-1, wherein determining the plurality of feature subset scores associated with the plurality of candidate subsets comprises: determining a first candidate subset of features, wherein the plurality of candidate subsets comprises the first candidate subset of features; determining a first feature subset score based on the first candidate subset of features using a neural network or decision tree; and selecting the first candidate subset of features as the first subset of features based on the first feature subset score being a maximum or minimum of the plurality of feature subset scores.

I-3. The medium of any of embodiments I-1 to I-2, the operations further comprising: obtaining a set of eigenvectors; and computing the first multidimensional vector based on a first set of feature values for a first vertex and the set of eigenvectors, wherein a sum of the set of eigenvectors when weighted by the first multidimensional vector satisfies a second threshold associated with the first set of feature values.

I-4. The medium of any of embodiments I-1 to I-3, wherein determining an extracted feature score comprises using a neural network that comprises a set of input layers and a set of output layers, wherein a count of the set of input layers is equal to a count of the set of output layers.

I-5. The medium of any of embodiments I-1 to I-4, wherein determining the metric comprises determining a Minkowski distance between the first multidimensional vector and a second multidimensional vector.

I-6. The medium of any of embodiments I-1 to I-5, the operations further comprising determining a first subset of vertices of the set of vertices based on the first subset of features and the set of extracted feature values satisfying a third threshold.

I-7. The medium of embodiment I-6, wherein determining the first subset of vertices comprises obtaining a set of prioritization parameters comprising the third threshold via a user interface element.

I-8. The medium of any of embodiments I-1 to I-7, the operations further comprising: determining whether a graph portion of the first directed graph matches a graph portion template, the graph portion template indicating a first vertex template, a second vertex template, and a directed edge template; generate an indicator associated with the graph portion; and visually indicating the graph portion associated with the graph portion based on the indicator.

I-9. The medium of any of embodiments I-1 to I-8, the operations further comprising increasing a feature value associated with a first feature in response to a determination that a first vertex is associated with a first category label and that a first conditional statement associated with the first vertex is satisfied.

I-10. The medium of any of embodiments I-1 to I-9, the operations further comprising visually indicating a natural language text section associated with a vertex of the first directed graph.

I-11. The medium of embodiment I-10, wherein determining the set of feature values comprises: determining a set of embedding values based on the natural language text section using a neural network; and determining a topic score based on the set of embedding scores, wherein the set of feature values comprises the topic score.

I-12. The medium of any of embodiments I-1 to I-11, the operations further comprising visually indicating a first vertex of the first directed graph with a shape, color, pattern, or animation that is different from a shape, color, pattern, or animation of a second vertex of the first directed graph in a visual display of the first directed graph.

I-13. The medium of any of embodiments I-1 to I-12, the operations further comprising providing a user interface, the user interface comprising a set of shapes representing vertices and a set of lines connecting the set of shapes, wherein each line is associated with an edge.

I-14. The medium of embodiment I-13, the operations further comprising providing a user interface (UI), the UI indicating a first subset of vertices in a different color than a second subset of vertices of the first directed graph.

I-15. The medium of any of embodiments I-1 to I-14, wherein determining the set of extracted feature values comprises: determining a set of adjacency values associated with a first vertex; and determining a matrix multiplication product based on the set of adjacency values and one or more feature values of the set of feature values.

I-16. The medium of any of embodiments I-1 to I-15, the operations further comprising providing a user interface (UI), the UI comprising: a set of identifiers associated with the first subset of features; and a set of UI elements that, after manipulation, causes an update to a feature value of a vertex of the first directed graph.

I-17. The medium of embodiment I-16, the operations further comprising determining a limit associated with a first feature of the first subset of features, wherein the UI causes a display of the limit.

I-18. The medium of any of embodiments I-1 to I-17, the operations further comprising steps for determining the metric between the first directed graph and the second directed graph.

I-19. The medium of any of embodiments I-1 to I-18, the operations further comprising steps for determining the first subset of features.

I-20. A method to perform the operations of any of the embodiments I-1 to I-19.

I-21. A system comprising: one or more processors; and memory storing instructions that, when executed by at least one of the one or more processors, causes at least one of the one or more processors to effectuate any of the operations of embodiments I-1 to I-19.

J-1. A tangible, non-transitory, machine-readable medium storing instructions that, when executed by a computer system, effectuate operations comprising: obtaining, with a computer system, program state of a self-executing protocol, wherein the program state comprises: a set of conditional statements; a first identifier of a first entity; and a directed graph, the directed graph comprising a set of vertices and a set of directed edges connecting respective pairs of vertices among the set of vertices, wherein each respective vertex of the set of vertices is associated with a respective category label of a set of mutually exclusive categories; receiving, at an application program interface of the computer system, an event message comprising a set of parameters; selecting, with the computer system, a first subset of vertices triggered by the event message based on the set of parameters; selecting, with the computer system, a second subset of vertices based on the first subset of vertices, wherein the second subset of vertices is associated with the first subset of vertices via the set of directed edges; determining, with the computer system, an aggregated parameter based on a subset of conditional statements, wherein each respective conditional statement of the subset of conditional statements is associated with a respective vertex of the second subset of vertices, and wherein the respective vertex is associated with a first category label of the set of mutually exclusive categories that is associated to each of the other vertices associated with the subset of conditional statements; and storing, with the computer system, the aggregated parameter in memory.

J-2. The medium of embodiment J-1, the operations further comprising: determining whether the event message is valid using a set of validator nodes of a peer-to-peer network, wherein each node of the peer-to-peer network is communicatively coupled to at least one other node of the peer-to-peer network; in response to a determination that the event message is valid, distributing a validation message indicating that the event message is valid; and storing a value based on the event message on a tamper-evident, distributed ledger encoding records of a plurality of previous values in a directed acyclic graph of cryptographic hash pointers, wherein the tamper-evident, distributed ledger is stored on the peer-to-peer network.

J-3. The medium of embodiment J-2, wherein determining the first subset of vertices comprises determining the first subset of vertices at a first node of the peer-to-peer network before the validation message is received by the first node.

J-4. The medium of any of embodiments J-1 to J-3, wherein the program state further comprises a first identifier of a first entity, the operations further comprising: determining whether the event message is valid using a set of validator nodes of a peer-to-peer network; based on a determination that the event message is not valid, sending an issue notification to a node of the peer-to-peer network associated with the first entity, wherein the issue notification comprises an identifier of the event message.

J-5. The medium of any of embodiments J-1 to J-4, wherein the program state further comprises a first identifier of a first entity, the operations further comprising: determining a network path from a first node of a peer-to-peer network to a second node of the peer-to-peer network using a breadth first search, wherein: the first node received the event message before the second node, the second node is associated with the first entity, and the network path comprises a plurality of nodes of the peer-to-peer network; and sending data of the event message to the second node from the first node via the network path.

J-6. The medium of any of embodiments J-1 to J-5, wherein the event message is a first event message, the operations further comprising: receiving a second event message within a duration threshold before or after receiving the first event message; determining whether the second event message causes a vertex of the first subset of vertices to trigger; in response to a determination that the second event message causes the vertex of the first subset of vertices to trigger, obtaining a set of triggering parameters of the second event message, wherein the set of triggering parameters comprise values that satisfy a condition of the vertex; determining whether a first value of the first event message and a second value of the second event message differ with respect to the set of triggering parameters; and based on a determination that the first value matches the second value, updating a parameter associated with the second event message to indicate that the second event message is a duplicate event message.

J-7. The medium of any of embodiments J-1 to J-6, wherein the program state further comprises a first identifier of a first entity and an second identifier of a second entity, the operations further comprising: retrieving a private conditional statement associated with the first entity, wherein the private conditional statement is not stored in program state accessible

to the second entity; and determining whether the private conditional statement is satisfied based on the first subset of vertices or the second subset of vertices.

J-8. The medium of any of embodiments J-1 to J-7, wherein a first stored value of the self-executing protocol is stored on a peer-to-peer network, and wherein a first node of the peer-to-peer network is permitted to access the first stored value of the program state, and wherein a second node of the peer-to-peer network is not permitted to access the first stored value.

J-9. The medium of any of embodiments J-1 to J-8, wherein determining the aggregated parameter comprises: determining that triggering a first vertex of a pair of vertices of the directed graph causes the cancellation of a second vertex of the pair of vertices of the directed graph, wherein the first vertex is associated with a first conditional statement and the second vertex is associated with a second conditional statement; selecting one of the pair parameters, the pair of parameters comprising a first parameter of the first conditional statement and a second parameter of the second conditional statement; and determining the aggregated parameter based on the first parameter.

J-10. The medium of any of embodiments J-1 to J-9, wherein the program state further comprises a first identifier of a first entity, and wherein the first entity is associated with an entity role, the operations further comprising selecting the first entity, wherein selecting the first entity comprises: selecting a vertex of the first subset of vertices based on the set of parameters; and selecting the first entity based on the entity role being associated with the vertex.

J-11. The medium of embodiment J-10, wherein a second entity is associated the entity role, the operations further comprising sending a second message to the second entity based on the second entity being associated with the entity role.

J-12. The medium of any of embodiments J-1 to J-11, wherein the program state further comprises a first identifier of a first entity, the operations further comprising: determining that the first entity is associated with an entity role; in response to a determination that the first entity is associated with the entity role, selecting a previous message from a history of messages based on the entity role; and sending the previous message to the first entity.

J-13. The medium of any of embodiments J-1 to J-12, the operations further comprising providing a user interface (UI), wherein vertices displayed in the UI are colored based on color associations with category labels associated with the vertices, and wherein each

respective category label of the set of mutually exclusive categories is associated with a different color.

J-14. The medium of any of embodiments J-1 to J-13, wherein the program state further comprises a first identifier of a first entity, the operations further comprising: determining whether a first confirmation key associated with a first representative of the first entity is received; determining whether a second confirmation key associated with a second representative the first entity is received; and in response to a determination that the first confirmation key and the second confirmation key is received, storing the first confirmation key and the second confirmation key in data storage in association with a record of a transaction between a pair entities comprising the first entity.

J-15. The medium of any of embodiments J-1 to J-14, wherein the program state further comprises a first identifier of a first entity, the operations further comprising: obtaining a score associated with the first entity, wherein the score is associated with a resource type; and updating the score based on the set of parameters, wherein the set of parameters comprises the resource type.

J-16. The medium of any of embodiments J-1 to J-15, wherein determining the aggregated parameter comprises determining a sum of values, wherein each respective value used to determine the sum of values is encoded in a respective conditional statement of the subset of conditional statements.

J-17. The medium of any of embodiments J-1 to J-16, the operations further comprising providing a user interface (UI), wherein the UI visually indicates the second subset of vertices based on a difference in color, difference in size, or difference in animation between the second subset of vertices and other vertices of the set of vertices.

J-18. The medium of any of embodiments J-1 to J-17, wherein determining the first subset of vertices comprises steps for determining the first subset of vertices.

J-19. The medium of any of embodiments J-1 to J-18, wherein determining the aggregated parameter comprises steps for determining the aggregated parameter.

J-20. A method to perform the operations of any of the embodiments J-1 to J-19.

J-21. A system comprising: one or more processors; and memory storing instructions that, when executed by at least one of the one or more processors, causes at least one of the one or more processors to effectuate any of the operations of embodiments J-1 to J-19.

K-1. A tangible, non-transitory, machine-readable medium storing instructions that, when executed by a computer system, effectuate operations comprising: receiving, with a computer system, a request via an application program interface (API), wherein the request comprises a callback address; determining, with the computer system, a query based on a set of query parameters; determining, with the computer system, a target graph portion template based on the query; searching, with the computer system, a set of directed graphs to determine a set of graph portions based on the query, where each of the set of graph portions match the target graph portion template, and wherein each respective directed graph of the set of directed graphs comprises: a set of vertices, wherein each respective vertex of the set of vertices is associated with a respective category label of a set of mutually exclusive categories, and a set of directed edges connecting respective pairs of vertices among the set of vertices; selecting, with the computer system, a set of event records, wherein each respective event records of the set of event records is indicated to occur before or during a vertex of a respective graph portion matching the target graph portion template; and sending, with the computer system, a value of the set of event records to the callback address.

K-2. The medium of embodiment K-1, wherein the query is associated with a first entity, the operations further comprising determining the set of related entities of the first entity, wherein each respective entity of the set of related entities is indicated to have had a transaction with the first entity.

K-3. The medium of any of embodiments K-1 to K-2, the operations further comprising determining whether the target graph portion template is stored in a library of graph portion templates, wherein: the library of graph portion templates comprises a graph database; a respective record of the graph database is associated with a respective graph portion template, and a respective identifier of the respective record comprises a respective set of vertices and a respective set of edges associating the respective set of vertices.

K-4. The medium of any of embodiments K-1 to K-3, the operations further comprising: determining whether a candidate graph portion matches with a graph portion template; updating a count associated with the graph portion template in a database based a determination that the candidate graph portion satisfies the graph portion template.

K-5. The medium of any of embodiments K-1 to K-4, the operations further comprising: determining whether the target graph portion template matches with a graph portion template stored in a library of graph portion templates; and in response to a determination that the

target graph portion template does not matches the graph portion template stored in the library of graph portion templates, update the library of graph portion templates based on the target graph portion template.

K-6. The medium of any of embodiments K-1 to K-5, wherein the query is a first query, the operations further comprising: determining whether results of a set of previous searches based on a preceding query have been made, where the first query occurs after the preceding query; determining a first search time based on the preceding query, wherein the first search time indicates a time of occurrence for the set of previous searches; and modifying the query based on the first search time.

K-7. The medium of any of embodiments K-1 to K-6, wherein the set of vertices are encoded as a serialized array of vertices, and wherein determining a set of graph portions comprises: deserializing the serialized array of vertices to generate a first directed graph in a non-persistent memory, wherein the first directed graph encodes the set of vertices, set of entities, and set of directed edges; determining a first graph portion based on the first directed graph, wherein selecting the set of graph portions comprises selecting the first graph portion.

K-8. The medium of any of embodiments K-1 to K-7, wherein: a directed graph of the set of directed graphs is stored on a tamper-evident, distributed ledger encoding records of a plurality of previous values in a directed acyclic graph of cryptographic hash pointers, wherein the tamper-evident, distributed ledger is stored on a peer-to-peer network; each record comprises a relational database record, the relational database record comprising a balanced search tree (b-tree); and a set of root values of the b-tree comprise identifiers associated with vertices of the directed graph or graph portions of the directed graph.

K-9. The medium of embodiment K-8, the operations further comprising determining whether the query comprises at least one of a first set of query parameters, wherein searching through the set of directed graphs comprises using the b-tree in response to a determination that the query comprises at least one of the first set of query parameters.

K-10. The medium of any of embodiments K-1 to K-9, the operations further comprising: wherein each respective directed graph of the set of directed graphs is stored on a respective tamper-evident, distributed ledger encoding records of a plurality of previous values in a directed acyclic graph of cryptographic hash pointers, wherein the tamper-evident, distributed ledger is stored on a peer-to-peer network; storing a first version of a first directed graph of the set of directed graphs on a set of data centers, wherein the set of data centers does not use

at least one peer node of the peer-to-peer network; and wherein searching through the set of directed graphs comprises searching through the first version of the first directed graph stored on the set of data centers.

K-11. The medium of embodiment K-10, wherein the set of data centers is a first set of data centers, the operation further comprising: storing a second version of the first directed graph on a second set of data centers, wherein the second set of data centers is different from the first set of data centers; and determining whether the first version of the first directed graph is valid based on the second version of the first directed graph.

K-12. The medium of any of embodiments K-1 to K-11, wherein determining the query comprises: determining whether an entity satisfies an access criteria, wherein the request is received from the entity; and based on a determination that the entity satisfies the access criteria, update the entity.

K-13. The medium of any of embodiments K-1 to K-12, wherein determining the query comprises: determining whether the set of query parameters comprises a command; and incorporating the command into the query.

K-14. The medium of any of embodiments K-1 to K-13, wherein the request is a first request, the operations further comprising: receiving a second request; determining whether the second request is valid; and in response to a determination that the second request is not valid, sending a message indicating that the second request is invalid.

K-15. The medium of any of embodiments K-1 to K-14, the operations further comprising: wherein each respective directed graph of the set of directed graphs is stored on a respective tamper-evident, distributed ledger encoding records of a plurality of previous values in a directed acyclic graph of cryptographic hash pointers, wherein the tamper-evident, distributed ledger is stored on a peer-to-peer network; receiving the request at a first node of the peer-to-peer network; determining whether the request causes a database search; in response to a determination that the request causes a database search, sending the request to a second node; wherein searching through the set of directed graphs comprises performing the database search using the second node.

K-16. The medium of any of embodiments K-1 to K-15, wherein searching through the set of directed graphs comprises searching through a set of binary trees associated with the set of directed graphs.

K-17. The medium of any of embodiments K-1 to K-16, the operations further comprising determining whether the callback address is valid based on a set of permitted addresses.

K-18. The medium of any of embodiments K-1 to K-17, wherein searching through the set of directed graphs comprises steps for searching through the set of directed graphs.

K-19. The medium of any of embodiments K-1 to K-18, wherein determining the query comprises steps for determining the query.

K-20. A method to perform the operations of any of the embodiments K-1 to K-19.

K-21. A system comprising: one or more processors; and memory storing instructions that, when executed by at least one of the one or more processors, causes at least one of the one or more processors to effectuate any of the operations of embodiments K-1 to K-19.

L-1. A tangible, non-transitory, machine-readable medium storing instructions that, when executed by a computer system, effectuate operations comprising: obtaining, with a computer system, a directed graph encoding a symbolic artificial intelligence (AI) model used by a first entity, the directed graph comprising a first set of vertices and a first set of edges associating pairs of vertices of the first set of vertices, wherein: each respective vertex of the first set of vertices is associated with a vertex status and is labeled with a category selected from a set of mutually-exclusive categories, and a vertex of the first set of vertices is associated with a conditional statement that is indicated as triggerable by the first entity; determining, with the computer system, a set of features based on the directed graph, the set of features comprising an identifier of a graph portion template, wherein each respective vertex of the graph portion template of the graph portion template is labeled with a same category from the set of mutually-exclusive categories as a corresponding respective vertex of a graph portion of the directed graph and is associated with a same count of edges; obtaining, with the computer system, a set of model parameter values for a machine learning model based on the graph portion template; and providing, with the computer system, the set of model parameter values and the graph portion templates to the first entity, wherein the set of model parameter values are used by to determine an outcome score based on the directed graph.

L-2. The medium of embodiment L-1, the operations further comprising: obtaining a conditional statement parameter, wherein the conditional statement parameter is used by the conditional statement; wherein obtaining the set of model parameter values comprises selecting a parameter of the set of model parameter values based on the conditional statement parameter.

L-3. The medium of any of embodiments L-1 to L-2, wherein the outcome score is a first outcome score, the operations further comprising: determining a map indicating a first graph portion of the directed graph, wherein the map comprises identifiers for vertices of the directed graph; and determining a second outcome score using the set of model parameters based on the map.

L-4. The medium of any of embodiments L-1 to L-3, wherein the directed graph is a first directed graph, and wherein determining the first graph portion comprises: using a first neural network based on the first directed graph and the set of model parameter values to determine the first outcome score, wherein the machine learning model comprises the first neural network; updating the map multiple times to generate a plurality of simulated directed graphs, wherein a first simulated directed graph matches the first graph portion with respect to vertex categories, and wherein each of the simulated directed graphs is a subgraph of the first directed graph; determining a set of simulated outcome scores with the neural network based on the plurality of simulated directed graphs and the set of model parameter values, wherein the plurality of simulated outcome scores comprises the second outcome score, and wherein using the neural network based on the first simulated directed graph provides the second outcome score; selecting the second outcome score of the set of simulated outcome scores based a difference between the second outcome score and the first outcome score; selecting the first graph portion based on the selection of the second outcome score; and sending an indicator associated with the first graph portion to the first entity.

L-5. The medium of any of embodiments L-1 to L-4, wherein the directed graph is a first directed graph, and wherein the set of features is a first set of features, and wherein determining the first graph portion comprises: using a first neural network based on the first directed graph and the set of model parameter values to determine the first outcome score, wherein the machine learning model comprises the first neural network; updating the map multiple times to generate a plurality of sets of features, wherein each respective set of features is different from the first set of features; determining a set of simulated outcome scores based on the plurality of sets of features using the set of model parameter values and the neural network, wherein the plurality of simulated outcome scores comprises the second outcome score, and wherein using the neural network based on a first set of features provides the second outcome score; selecting the second outcome score of the set of simulated outcome scores based a difference between the second outcome score and the first outcome score; selecting the first

graph portion based on the first graph portion being associated with first set of features; and sending an indicator associated with the first graph portion to the first entity.

L-6. The medium of any of embodiments L-1 to L-5, the operations further comprising: obtaining a plurality of directed graphs; determining multiple sets of features, wherein each respective set of features is determined based on a respective directed graph of the plurality of directed graphs; determining the set of model parameter values by training a version of the machine learning model based on the multiple sets of features and a set of training outputs; storing the set of model parameter values in a record of a database, wherein the record is associated with a set of labels, and wherein a search through the database using the set of labels provides an identifier of the record.

L-7. The medium of any of embodiments L-1 to L-6, wherein determining the multiple sets of features comprises determining a set of values in a non-Euclidean domain based on the multiple sets of directed graphs, and wherein determining the set of values in the non-Euclidean domain comprises determining a matrix inversion of a diagonal of an adjacency matrix of the directed graph.

L-8. The medium of any of embodiments L-1 to L-7, the operations further comprising: obtaining a first transaction score, wherein the first transaction score is associated with a first transaction indicated to have triggered or activated a first vertex of a first directed graph of the plurality of directed graphs; obtaining a second transaction score, wherein the second transaction score is associated with a second transaction indicated to have triggered or activated a second vertex of a second directed graph of the plurality of directed graphs; aggregating the first score and the second score to form an aggregated score, the aggregating comprising a summation operation; and determining the set of model parameter values during a training operation based on the aggregated score.

L-9. The medium of any of embodiments L-1 to L-8, wherein obtaining the plurality of directed graphs comprises obtaining the plurality of directed graphs from a tamper-evident, distributed ledger.

L-10. The medium of any of embodiments L-1 to L-9, the operations further comprising selecting the machine learning model based on the set of features, wherein obtaining the set of model parameter values comprises selecting a model parameter value based on the selected learning model.

L-11. The medium of any of embodiments L-1 to L-10, wherein the set of features is a first set of features, the operations further comprising determining a second set of features based on the selected learning model and the directed graph, wherein obtaining the set of model parameter values comprises selecting a model parameter value based on the second set of features.

L-12. The medium of any of embodiments L-1 to L-11, wherein the first set of features and the second set of features are orthogonal to each other.

L-13. The medium of any of embodiments L-1 to L-12, wherein the set of model parameter values is received at a computer device controlled by the first entity, the operations further comprising: determining the outcome score using the set of model parameter values; and in response to a determination that the outcome score satisfies a warning threshold, sending a message to the first entity associated with the warning threshold.

L-14. The medium of any of embodiments L-1 to L-13, where obtaining the set of model parameter values comprises obtaining the set of model parameter values from a record stored on a cloud computing resource.

L-15. The medium of any of embodiments L-1 to L-14, wherein obtaining the set of model parameter values comprises: determining an entity role associated with the entity; and selecting a model parameter value of the set of model parameter values based on the entity role.

L-16. The medium of any of embodiments L-1 to L-15, wherein the machine learning model comprises a neural network, and wherein the model parameter values comprise weights for neurons of a neural network.

L-17. The medium of any of embodiments L-1 to L-16, wherein the set of model parameter values is a first set of model parameter values, the operations further comprising: obtaining a second set of model parameter values associated with an entity identifier; determining a second outcome score based on the second set of model parameter values; and based on a comparison between the first outcome score and the second outcome score, causing a transaction that updates a score associated with an entity identified by the entity identifier, wherein the score is stored in tamper-evident, distributed ledger.

L-18. The medium of any of embodiments L-1 to L-17, wherein determining the set of model parameter values comprises steps for determining the set of model parameter values.

L-19. The medium of any of embodiments L-1 to L-18, wherein determining the outcome score comprises steps for determining the outcome score.

L-20. A method to perform the operations of any of the embodiments L-1 to L-19.

L-21. A system, comprising: one or more processors; and memory storing instructions that when executed by the processors cause the processors to effectuate operations comprising: the operations of any one of embodiments L-1 to L-19.

L-22. A tangible, non-transitory, machine-readable medium storing instructions that, when executed by a computer system, effectuate operations comprising: obtaining, with a first computer system controlled by a first entity, a first training set, the first training set comprising a plurality of pairs of symbolic AI models and logs of state of the symbolic AI models; training, with the first computer system, a first machine learning model on the first training set, wherein training comprises iteratively adjusting parameters of the first machine learning model based on a first objective function; after training the first machine learning model, providing the first machine learning model to a second computer system controlled by a second entity, wherein the second entity does not have access to at least some of the first training set; obtaining, with the second computer system, a second training set, the second training set comprising a plurality of pairs of symbolic AI models and logs of state of the symbolic AI models, the second training set being different at least in part from the first training set; training, with the second computer system, a second machine learning model that includes the first machine learning model on the second training set, wherein training comprises iteratively adjusting parameters of the second machine learning model based on a second objective function; and after training the second machine learning model, storing, with the second computer system, the second machine learning model in memory.

L-23. The medium of embodiment L-22, wherein the second machine learning model comprises: the first machine learning model as a sub-model having an output; an error-correcting machine learning model as a sub-model having an input based on the output of the first machine learning model, wherein training the second machine learning model comprises iteratively adjusting parameters of the error-correcting machine learning model without changing parameters of the of the first machine learning model while adjusting parameters of the error-correcting machine learning model.

L-24. The medium of any of embodiments L-22 to L-23, wherein: parameters of the second machine learning model are initialized to values of parameters of the first machine learning model before being iteratively adjusted during training of the second machine learning model.

L-25. The medium of any of embodiments L-22 to L-24, wherein the first and second machine learning models are non-symbolic artificial intelligence models comprising steps for machine learning.

L-26. A method to perform the operations of any of the embodiments L-22 to L-25.

L-27. A system, comprising: one or more processors; and memory storing instructions that when executed by the processors cause the processors to effectuate operations comprising: the operations of any one of embodiments L-22 to L-26.

CLAIMS

What is claimed is:

1. A tangible, non-transitory, machine-readable medium storing instructions that, when executed by a computer system, effectuate operations comprising:
 - obtaining, with a computer system, program state of a smart contract, wherein the program state encodes a directed graph, the directed graph comprising:
 - a set of vertices, and
 - a set of directed edges connecting respective pairs of vertices among the set of vertices,
 - wherein the program state includes a set of conditional statements and a set of entities;
 - obtaining, with the computer system, a request encoding a set of conditional statement parameters corresponding to an amendment to the smart contract;
 - determining, with the computer system, a first subset of vertices in the directed graph, wherein each respective vertex of the first subset causes a state change of the program state in response to a respective conditional statement associated with the respective vertex being satisfied;
 - selecting, with the computer system, a second subset of the first subset based on the set of conditional statement parameters encoded in the request;
 - determining, with the computer system, a set of selected entities based on the second subset;
 - determining, with the computer system, whether a set of criteria associated with the set of selected entities is satisfied;
 - updating, with the computer system, the set of conditional statement parameters based on the set of conditional statements in response to a determination that the set of criteria associated with the set of selected entities is satisfied;
 - updating, with the computer system, the second subset based on the updated set of conditional statements; and
 - storing, with the computer system, the program state in storage memory after updating the second subset.

2. The medium of claim 1, wherein:

the vertices are norm vertices;

the first subset is a set of active vertices;

the second subset is a set of target vertices;

the set of entities include parties to the smart contract; and

updating the set of conditional statements comprises:

determining an affected conditional statement based on the request, wherein the set of target vertices comprises a reference that is associated with the affected conditional statement;

setting an indicator associated with the affected conditional statement to indicate that the affected conditional statement is deprecated;

generating a new conditional statement based on the set of conditional statement parameters; and

setting the reference to be associated with the new conditional statement.

3. The medium of claim 1, wherein:

the directed graph of the program state is stored in persistent memory and comprises a first serialized array of vertices and a second serialized array of vertices;

a target vertex of the set of target vertices is a vertex of the first serialized array of vertices and not a vertex of the second serialized array of vertices; and

updating the set of target vertices comprises:

deserializing the first serialized array of vertices to generate a first deserialized directed graph in a non-persistent memory, wherein the first serialized array of vertices comprises an identifier of the target vertex, and wherein the first deserialized directed graph comprises an adjacency matrix or an adjacency list, and wherein the second serialized array of vertices is not concurrently deserialized, and

serializing the first deserialized directed graph in the non-persistent memory to determine an updated first serialized array of vertices, and

storing the updated first serialized array of vertices in the persistent memory.

4. The medium of claim 1, wherein:

the request comprises a first entity identifier associated with a first entity;

a first conditional statement of the set of conditional statements comprises a condition that a second entity allocate a resource to a third entity, wherein a second entity identifier is associated with the second entity; and

the operations further comprises:

determining an entity field used by the first conditional statement, wherein the entity field comprises the second entity identifier, and

updating the entity field to comprise the first entity identifier.

5. The medium of claim 1, wherein updating the set of target vertices comprises deleting a first directed edge from the set of directed edges, wherein the first directed edge associates a target vertex of the set of target vertices with a second vertex of the directed graph.

6. The medium of any of claims 1 to 5, the operations further comprising:

obtaining a set of confirmation messages from each entity of the set of selected entities, wherein the set of confirmation messages comprises a set of passkey values, and wherein each respective passkey value of the set of passkey values is associated with a respective entity of the set of selected entities;

wherein determining that the set of criteria associated with the set of selected entities is satisfied comprises determining that each respective passkey value of the set of passkey values matches with a respective stored passkey value of a set of stored passkey values;

determining a first graph structure based on the request, wherein the first graph structure comprises the set of directed edges and a set of logical categories associated with each vertex of the first graph structure;

determining whether the first graph structure is identical to a second graph structure of the directed graph;

in response to a determination that the first graph structure is different from the second graph structure with respect to a number of vertices or number of edges, adding a new vertex to the set of vertices based on the request in persistent memory;

determining a simulated modified program state, wherein determining the simulated modified program state comprises updating a version of the set of conditional statements, and wherein the simulated modified program state comprises the version of the set of conditional

statements;

determining a set of simulated events, wherein the set of simulated events are determined occur in sequence based on an associated set of occurrence times; and

determining a set of outcome program states based on the simulated modified program state and the set of simulated events.

7. The medium of any of claim 1 to 5, wherein the request is a first request of a plurality of requests, the operations further comprising:

determining a set of simulated events;

determining a set of outcome scores, wherein determining the set of outcome scores comprises, for each respective request of the plurality of requests:

determining a respective simulated program state, wherein determining the respective simulated program state comprises updating a version of the set of conditional statements of the program state, and wherein the respective simulated program state comprises the version of the set of conditional statements,

determining a respective set of outcome program states based on the respective simulated program state and the set of simulated events, and

determining a respective set of outcome scores based on the respective set of outcome program states, wherein the respective set of outcome scores are part of the set of outcome scores; and

selecting the first request from the plurality of requests based on the set of outcome scores.

8. The medium of any of claim 1 to 5, wherein the request comprises an identifier associated with a second directed graph of a second smart contract program, and wherein a target vertex of the set of target vertices is associated with a first priority category value, and wherein the operations further comprise:

updating a set of directed edges based on an edge associating the target vertex with a second vertex of the second directed graph; and

assigning a second priority category value to the second vertex, wherein:

the first priority category value is different from the second priority category value,

a first event triggers both the target vertex and the second vertex, and

the order by which the target vertex and the second vertex is triggered in response to the first event is determined based on the first priority category value and the second priority category value.

9. The medium of any of claim 1 to 8, wherein:

a target vertex of the set of target vertices is associated with a first priority category value;

the request comprises a second priority category value;

updating the target vertex comprises changing the first priority category value to the second priority category value;

the request is a first request; and

the operations further comprise:

determining that a previous request failed to satisfy the set of criteria associated with the set of selected entities is satisfied; and

determining the first request based on the previous request by modifying a conditional statement parameter encoded in the previous request.

10. The medium of any of claim 1 to 9, wherein:

the request comprises a set of entity identifiers;

determining whether the set of criteria is satisfied comprises determining whether the set of entity identifiers comprises any entities of set of prohibited entities or entities of a set of prohibited entity types;

the request comprises an entity identifier; and

the operations further comprises adding the entity identifier to a set of entities associated with the program state.

11. The medium of any of claim 1 to 10, the operations further comprising:

obtaining a previous request;

determining a simulated modified program state, wherein determining the simulated modified program state comprises updating a version of the set of conditional statements based on the previous request, and wherein the simulated modified program state comprises the version of the set of conditional statements;

determining a simulated event;

determining that the simulated event causes an allocation of a non-duplicable asset from a first entity to a second entity based on the simulated event and an allocation of the non-duplicable asset from the first entity to a third entity based on the simulated event;

generating a message indicating that the previous request may cause a contradiction in response to a determination that the simulated event causes the allocation of the non-duplicable asset from the first entity to the second entity and the third entity based on the simulated event; wherein determining whether the set of criteria associated with the set of selected entities is satisfied the operations further comprising:

storing the set of criteria of the set of selected entities at a storage memory accessible to a first computing device of a distributed computing platform;

determining whether the set of criteria associated with the set of selected entities is satisfied using the first computing device; and

in response to a determination that the set of criteria associated with the set of selected entities is satisfied using the first computing device, modifying a version of the program state based on the request using a second computing device of the distributed computing platform.

12. The medium of any of claim 1 to 11, wherein:

the request comprises data encoding a second directed graph;

updating the set of conditional statements comprises adding a new conditional statement to the set of conditional statements, wherein the new conditional statement is indexed by a conditional statement identifier;

updating the set of target vertices comprises:

associating a target vertex of the set of target vertices with the conditional statement identifier, and

updating a target vertex of the set of target vertices based on the second directed graph; and

an event that triggers a condition of the target vertex causes a second vertex of the second directed graph be set as triggerable.

13. The medium of any of claims 1 to 12, wherein the request comprises a first amount, the operations further comprise:

determining whether a target vertex of the set of target vertices is associated with a

conditional statement that was triggered by a past event, wherein an outcome of the conditional statement is associated with a transaction between a first entity and a second entity, and wherein the transaction is associated with a score equal to a second amount;
determining a difference between the first amount and the second amount; and
initiating a transaction between the first entity and the second entity based on the difference.

14. A method, comprising the operations of any of claims 1 to 13.

15. A system, including: one or more processors; and memory storing instructions that when executed by at least some of the processors cause the processors to effectuate the operations of any of claims 1 to 13.

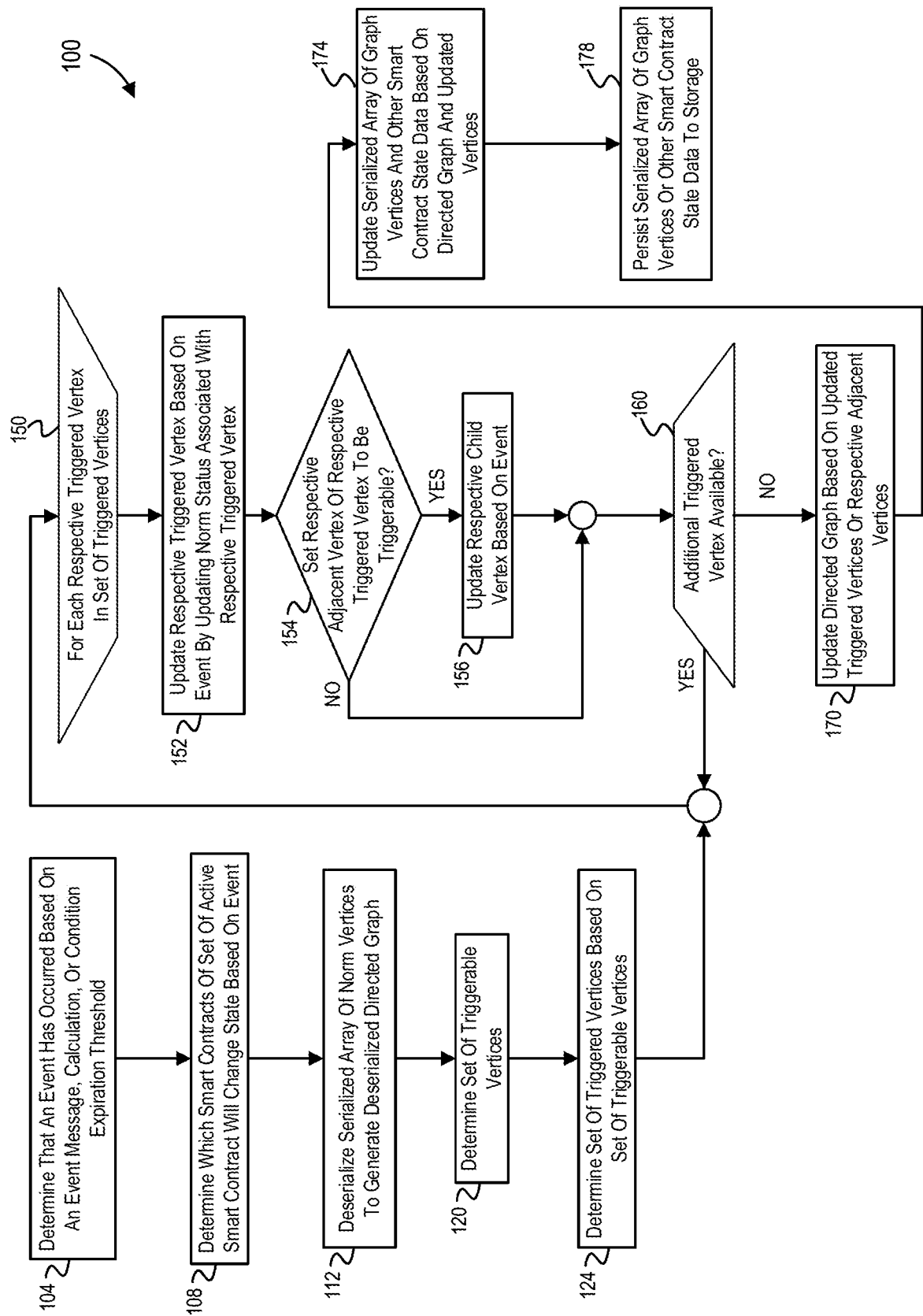


FIG. 1

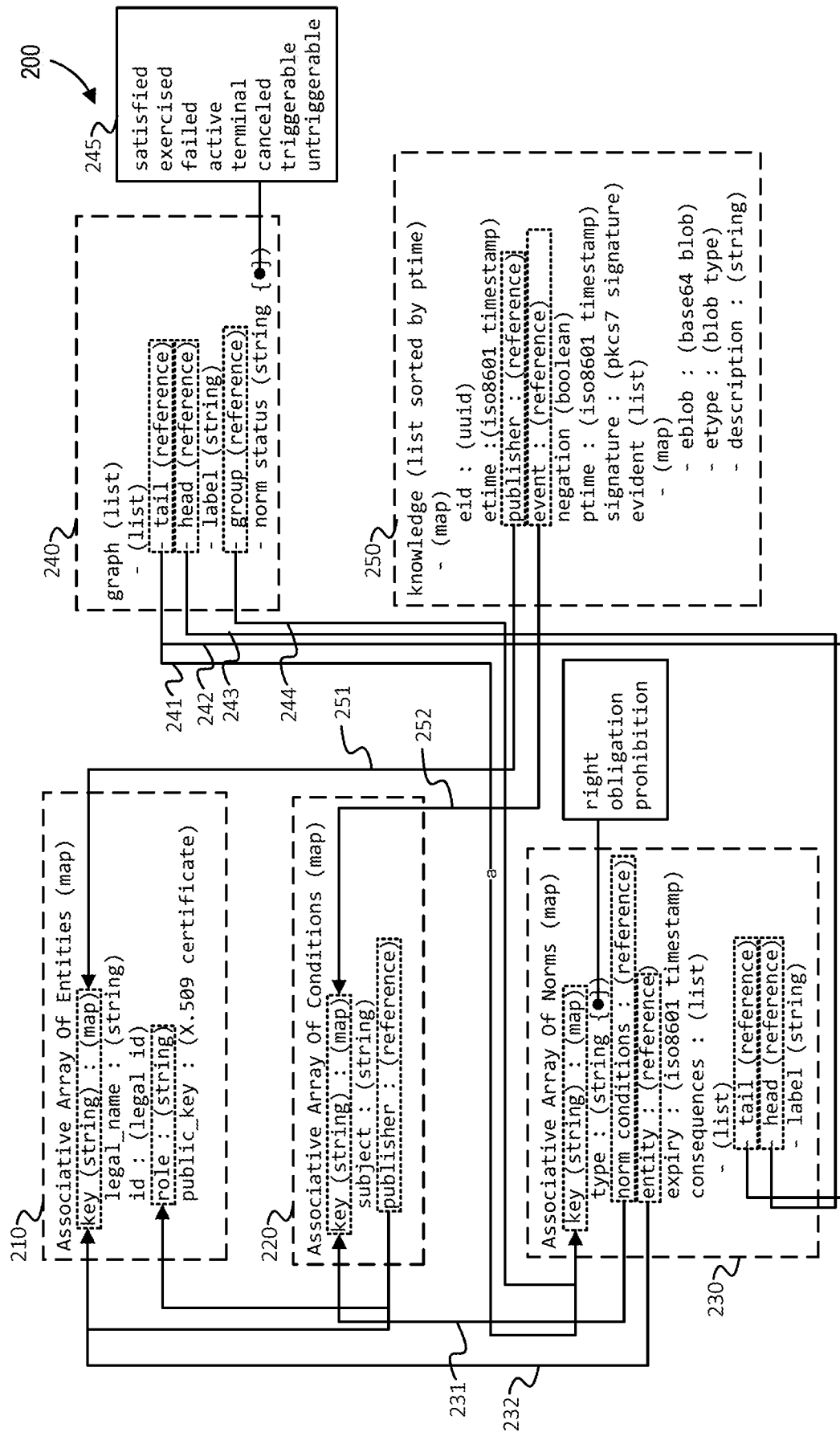


FIG. 2

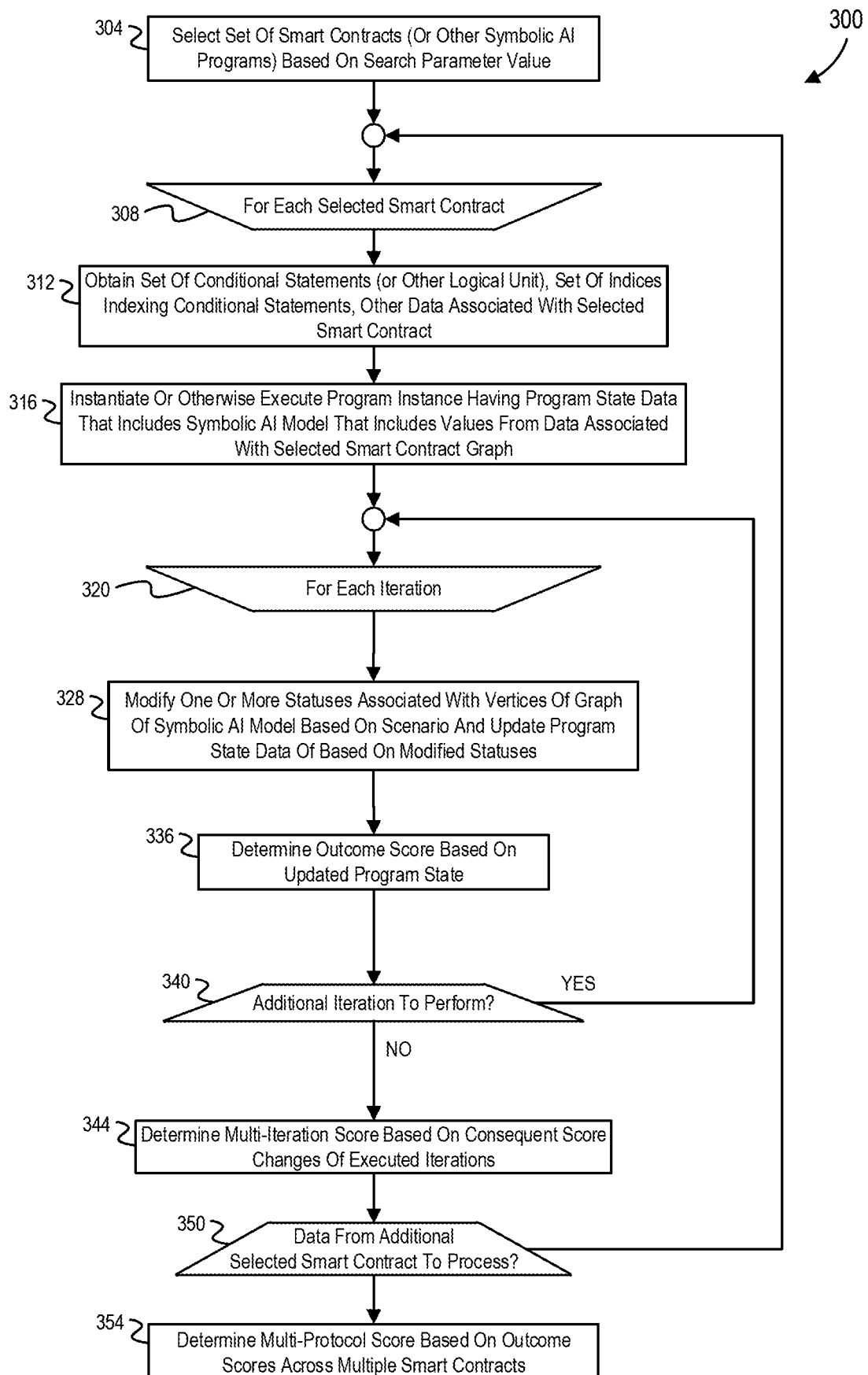


FIG. 3

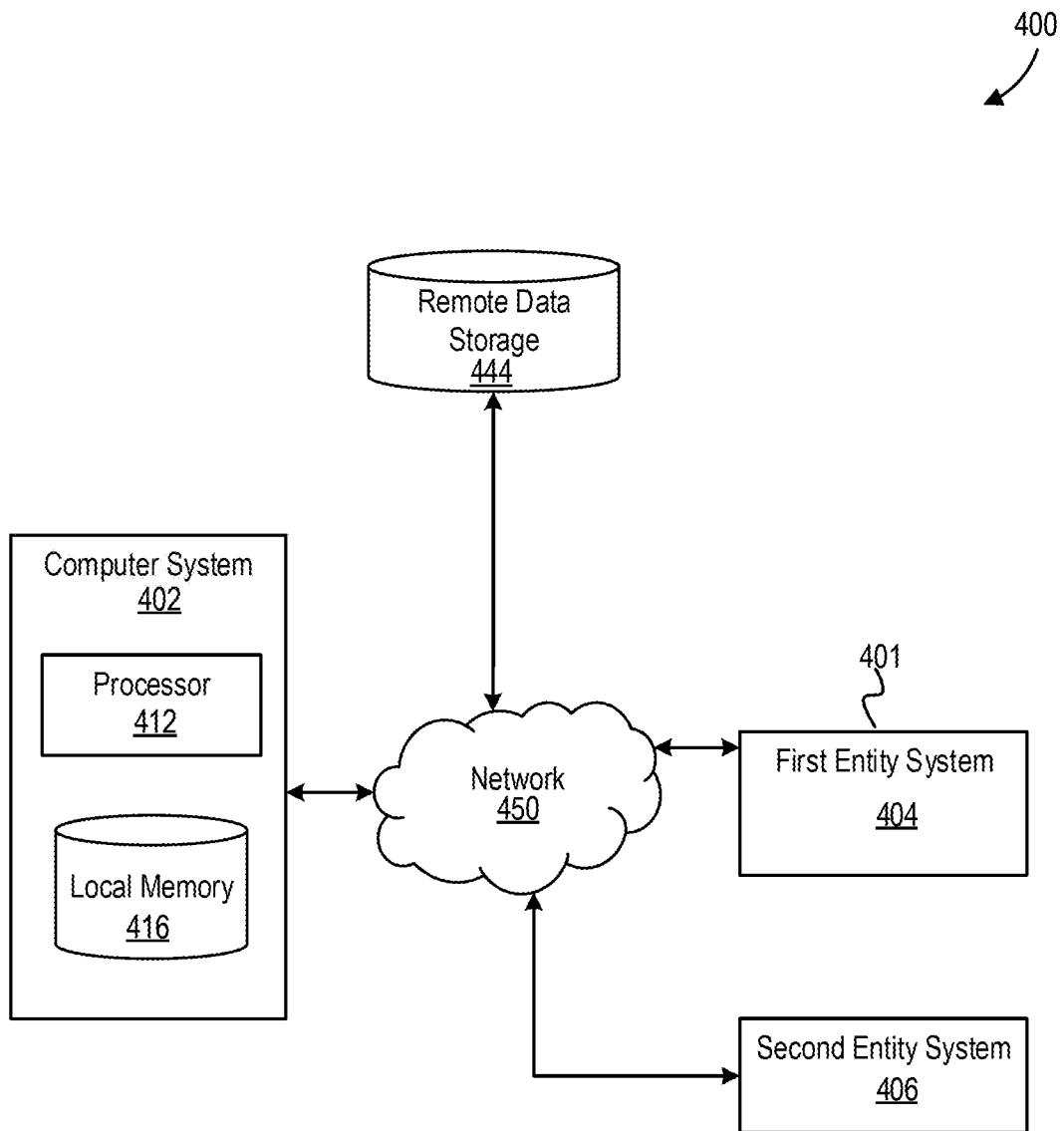


FIG. 4

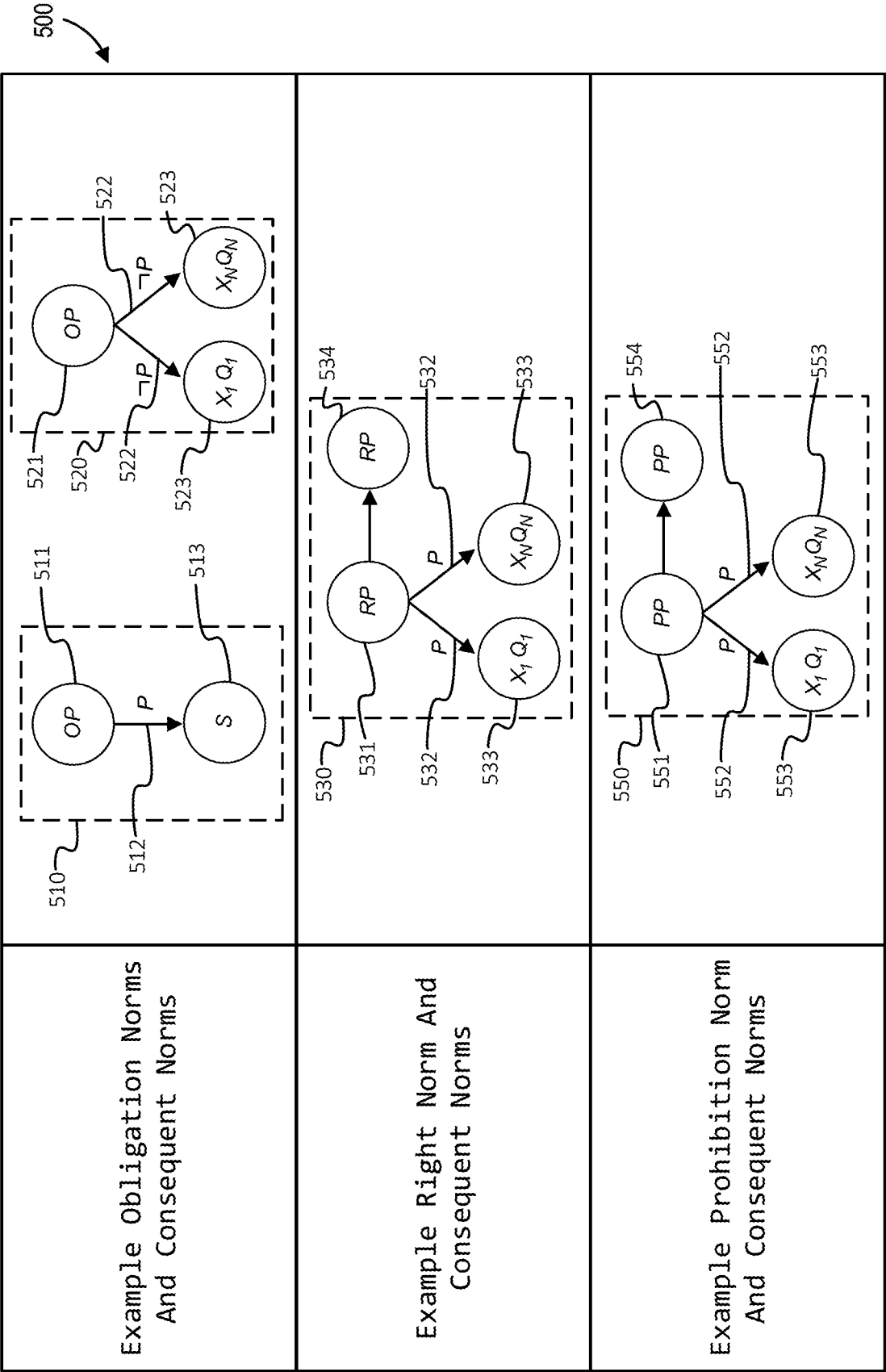


FIG. 5

600

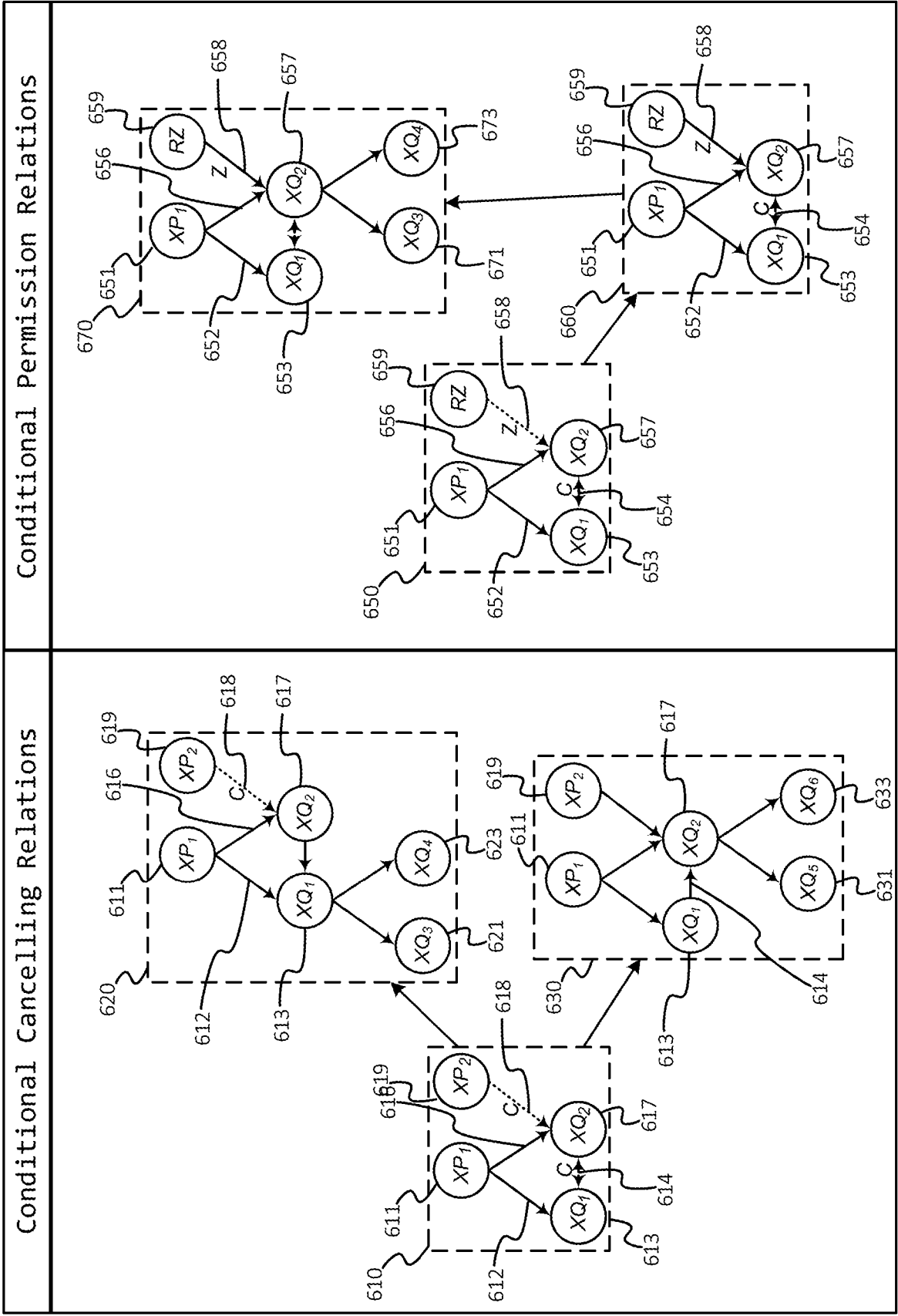


FIG. 6

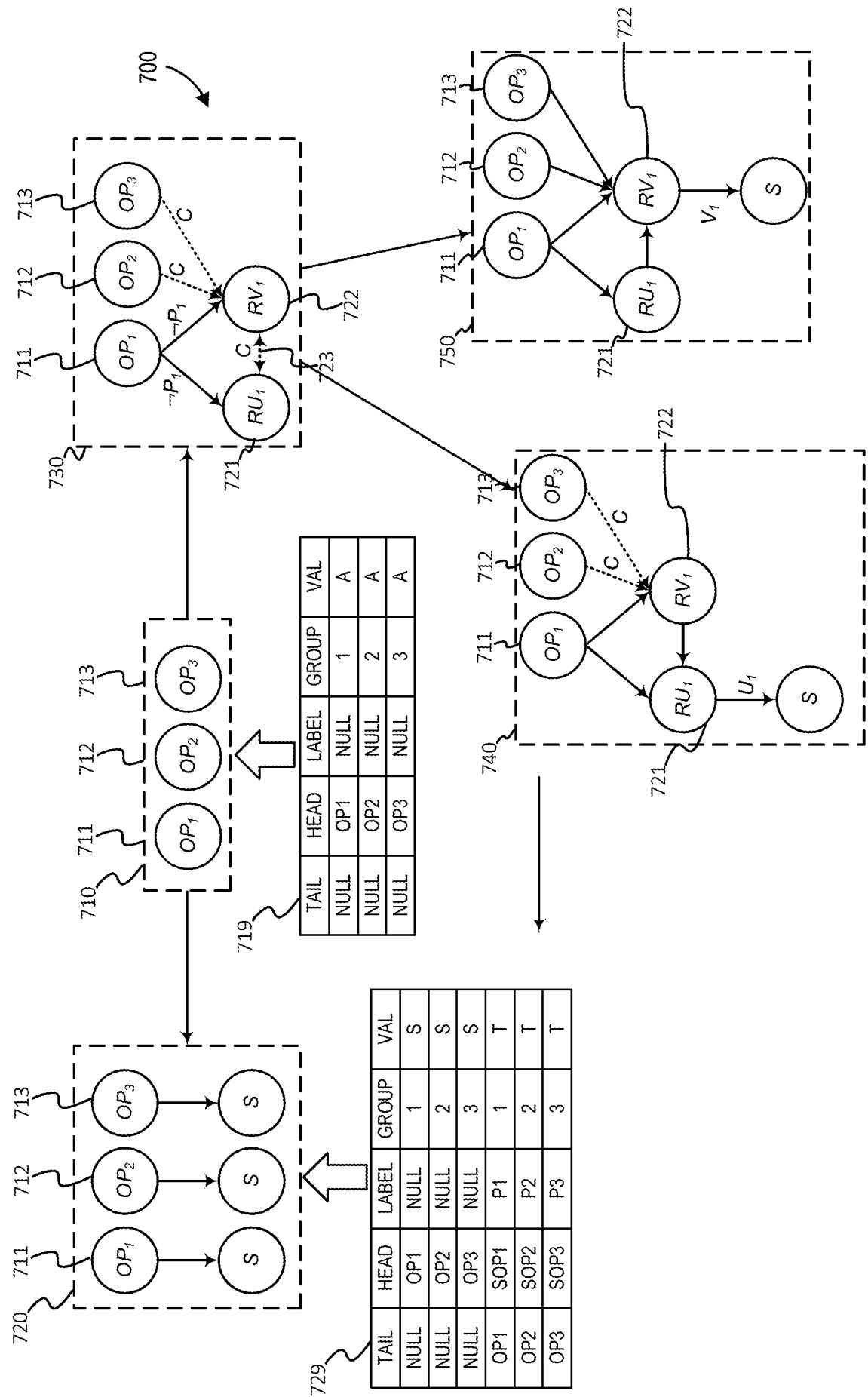


FIG. 7

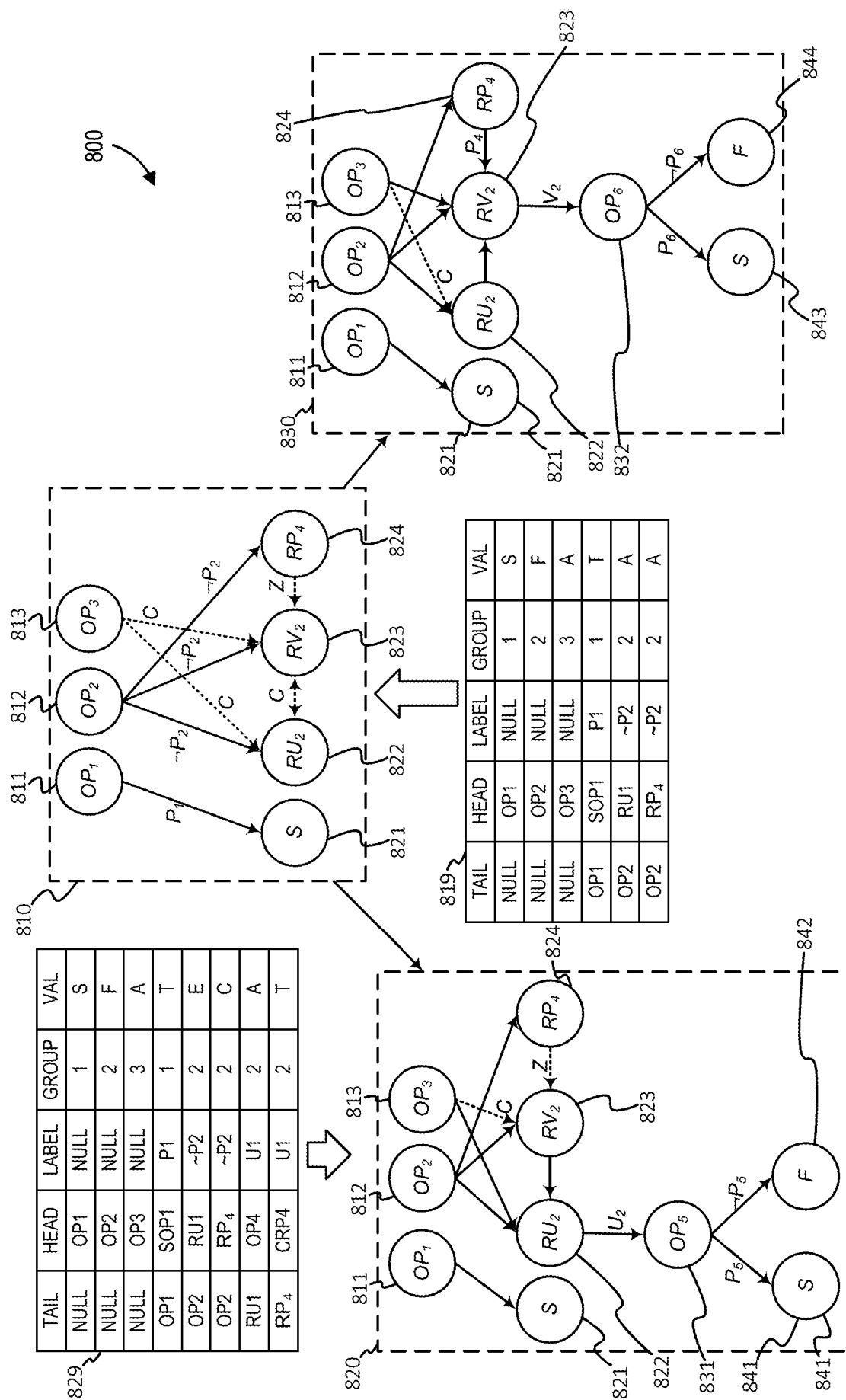


FIG. 8

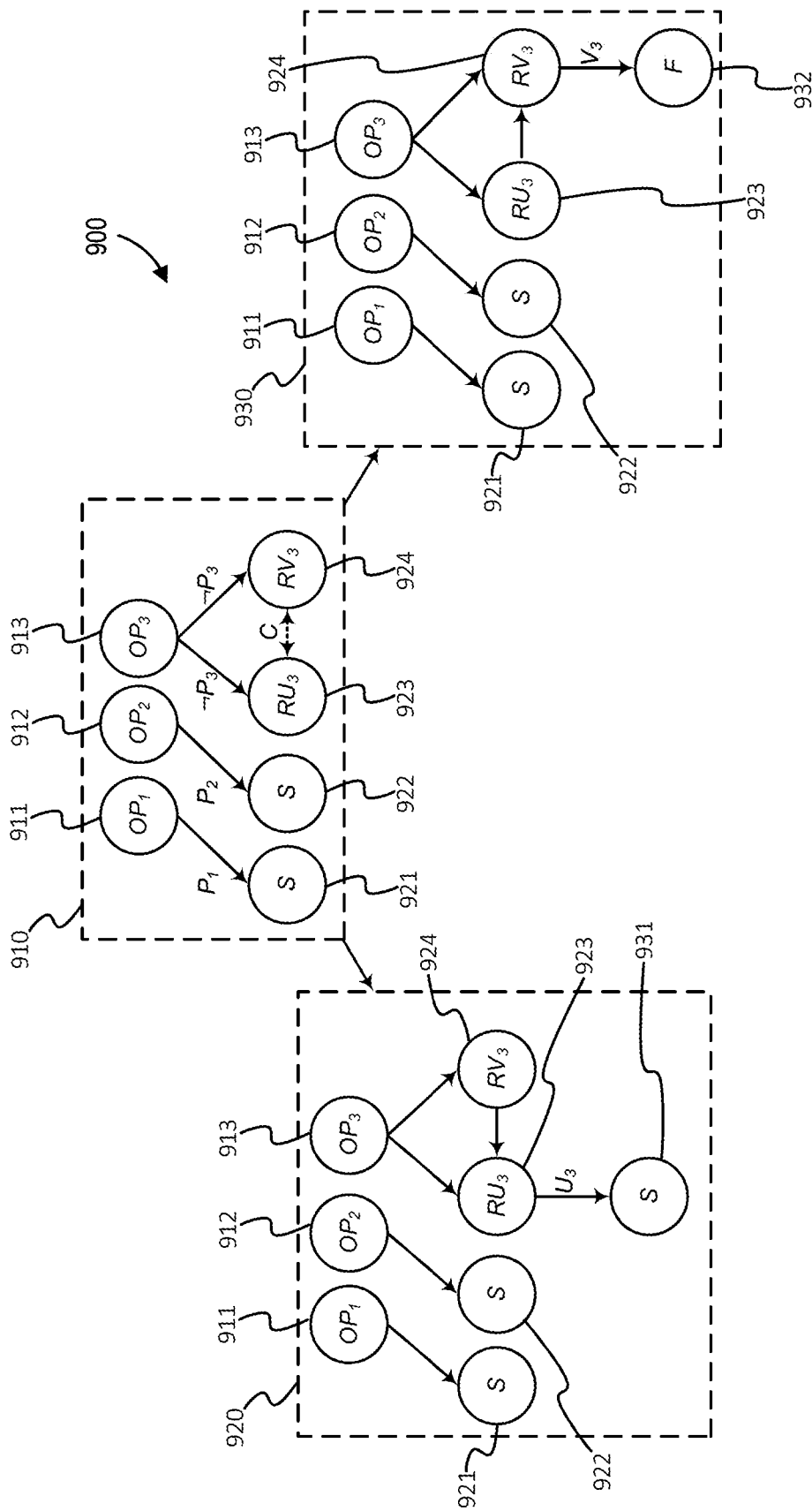


FIG. 9

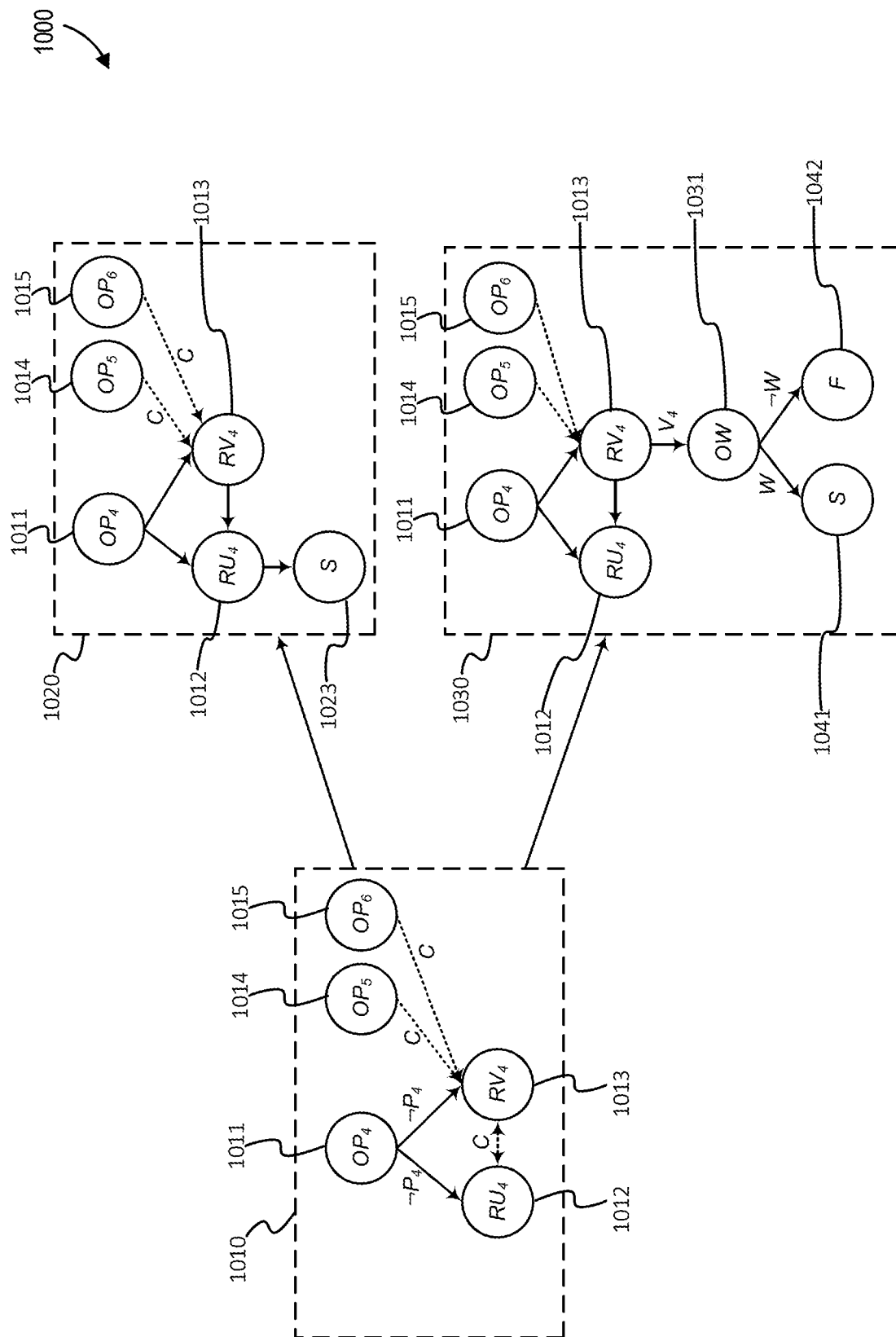


FIG. 10

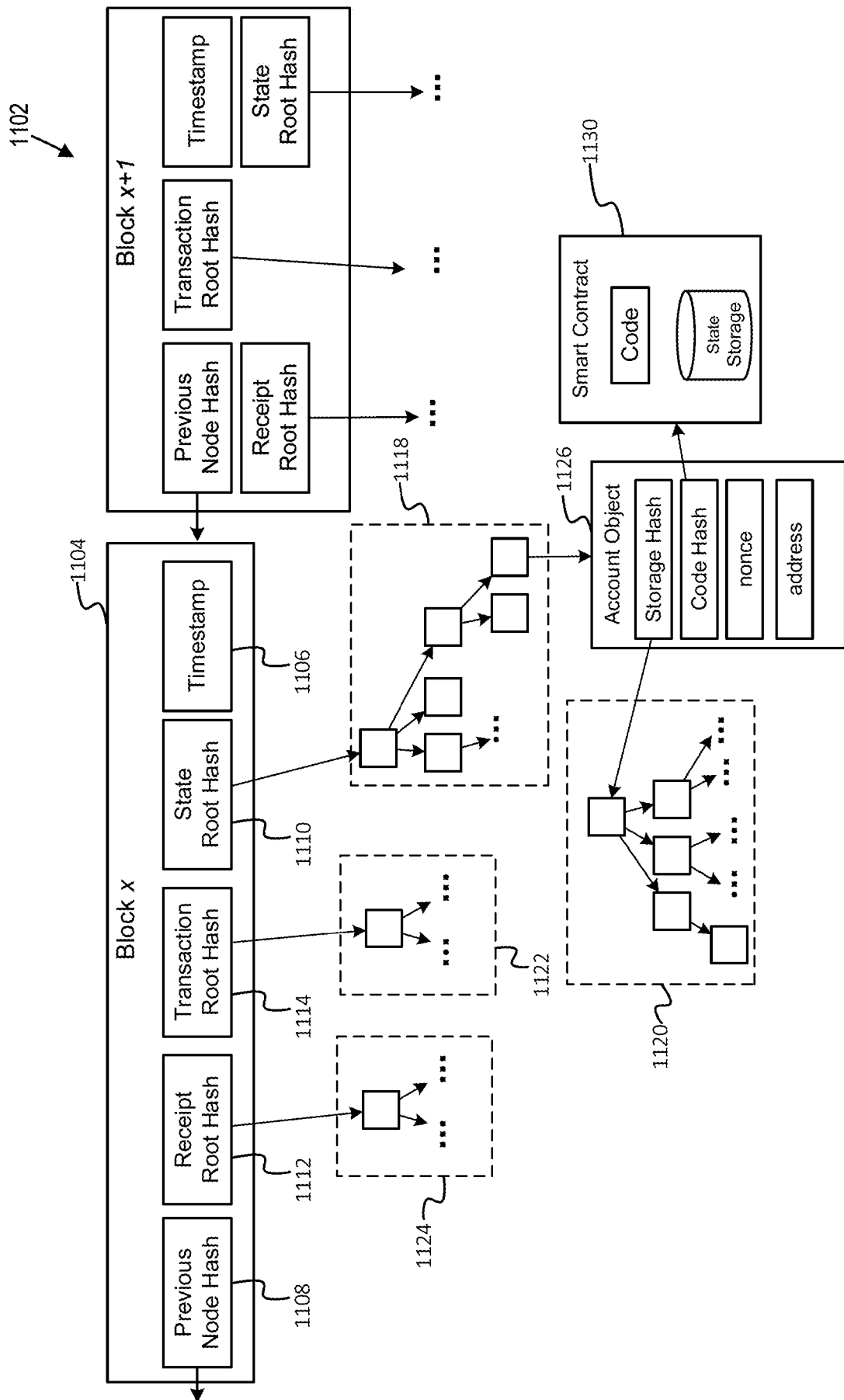


FIG. 11

12/17

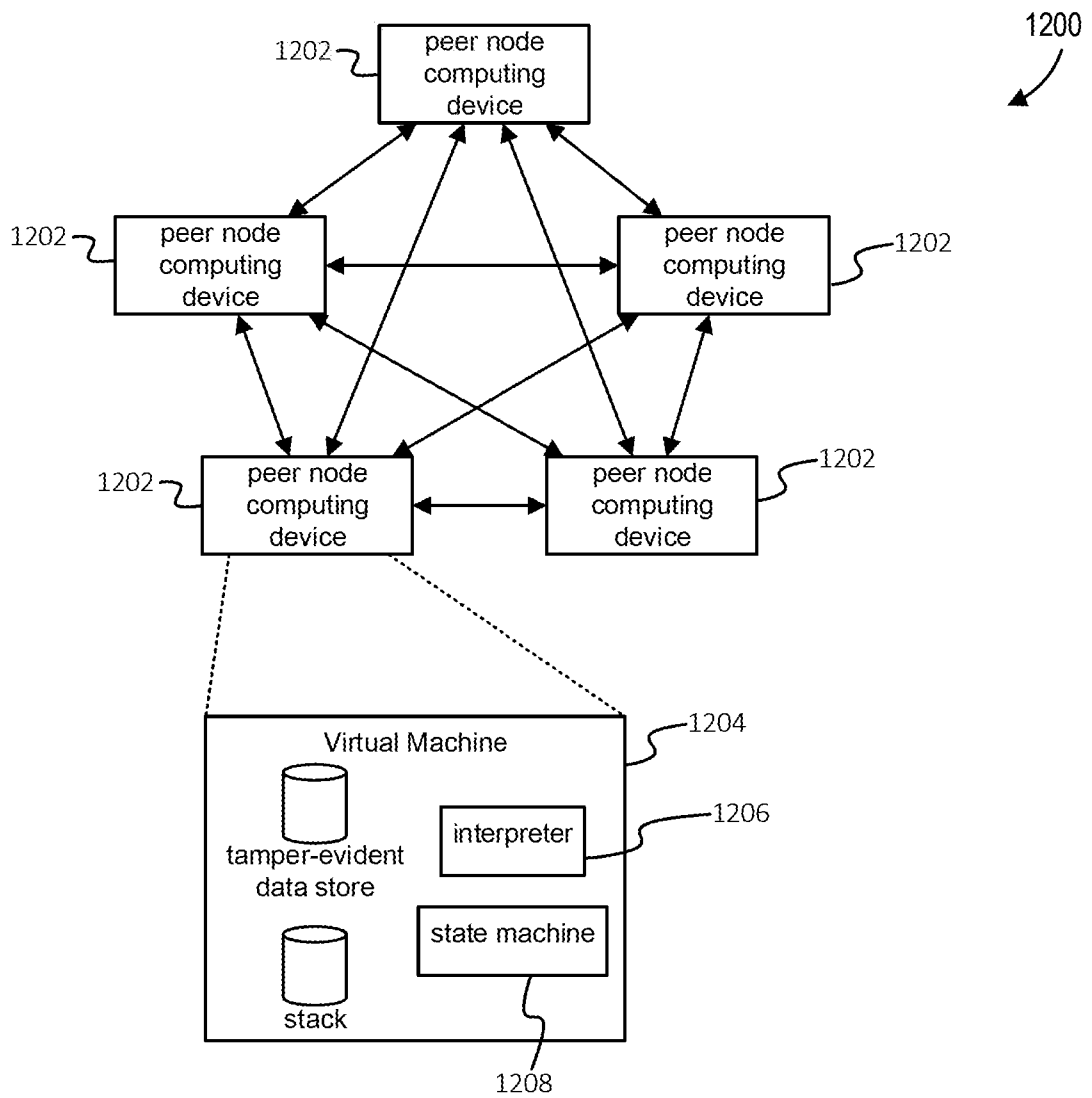


FIG. 12

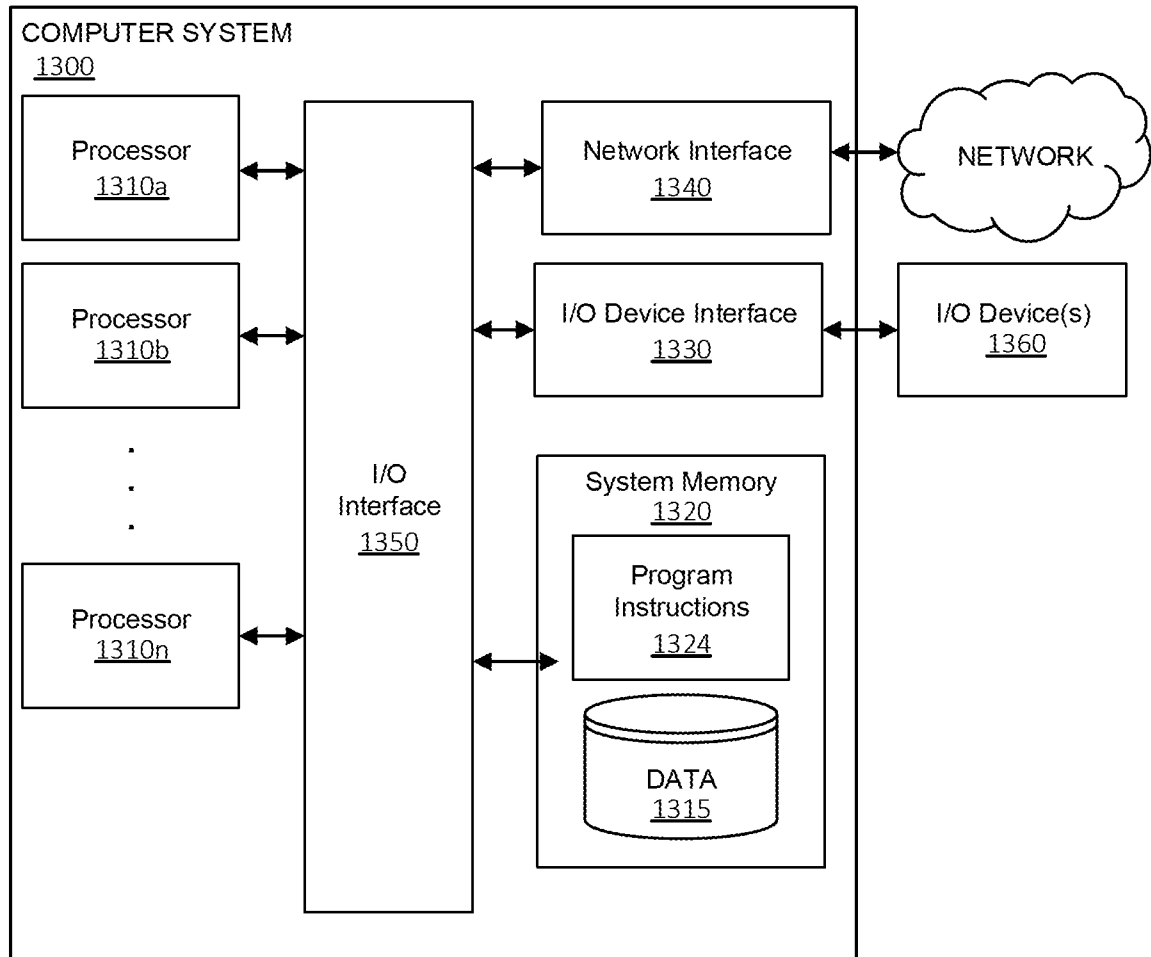


FIG. 13

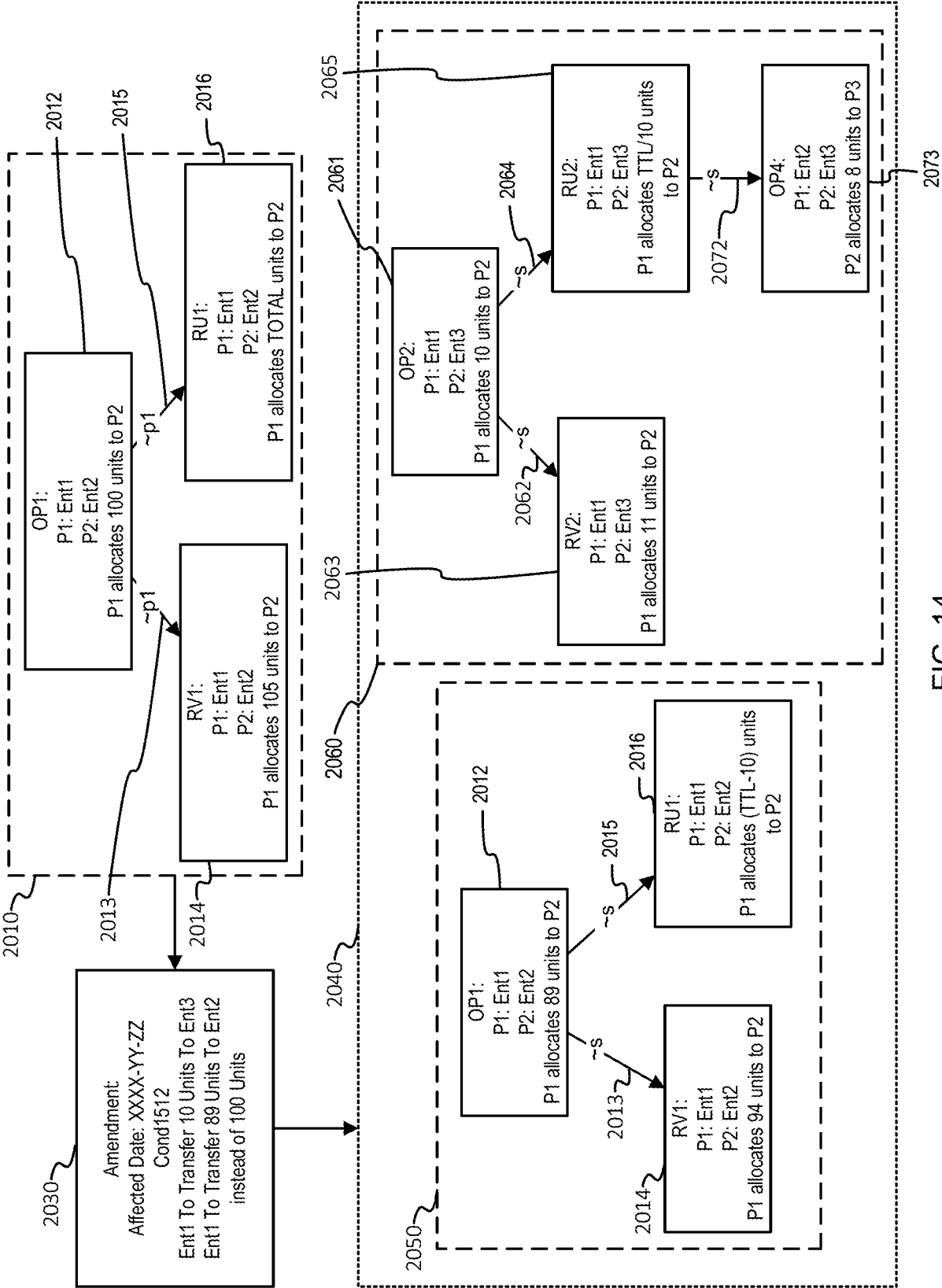


FIG. 14

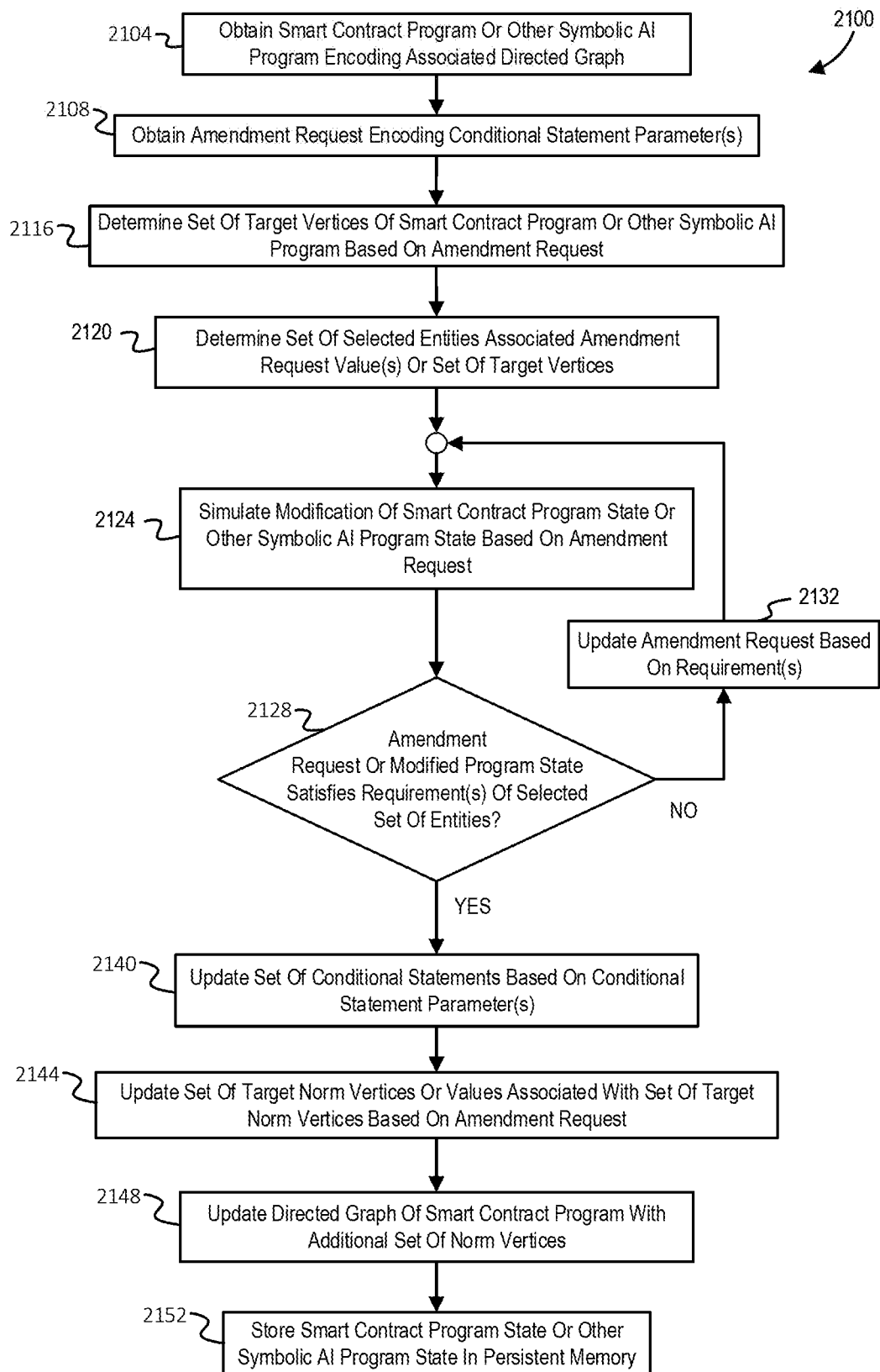


FIG. 15

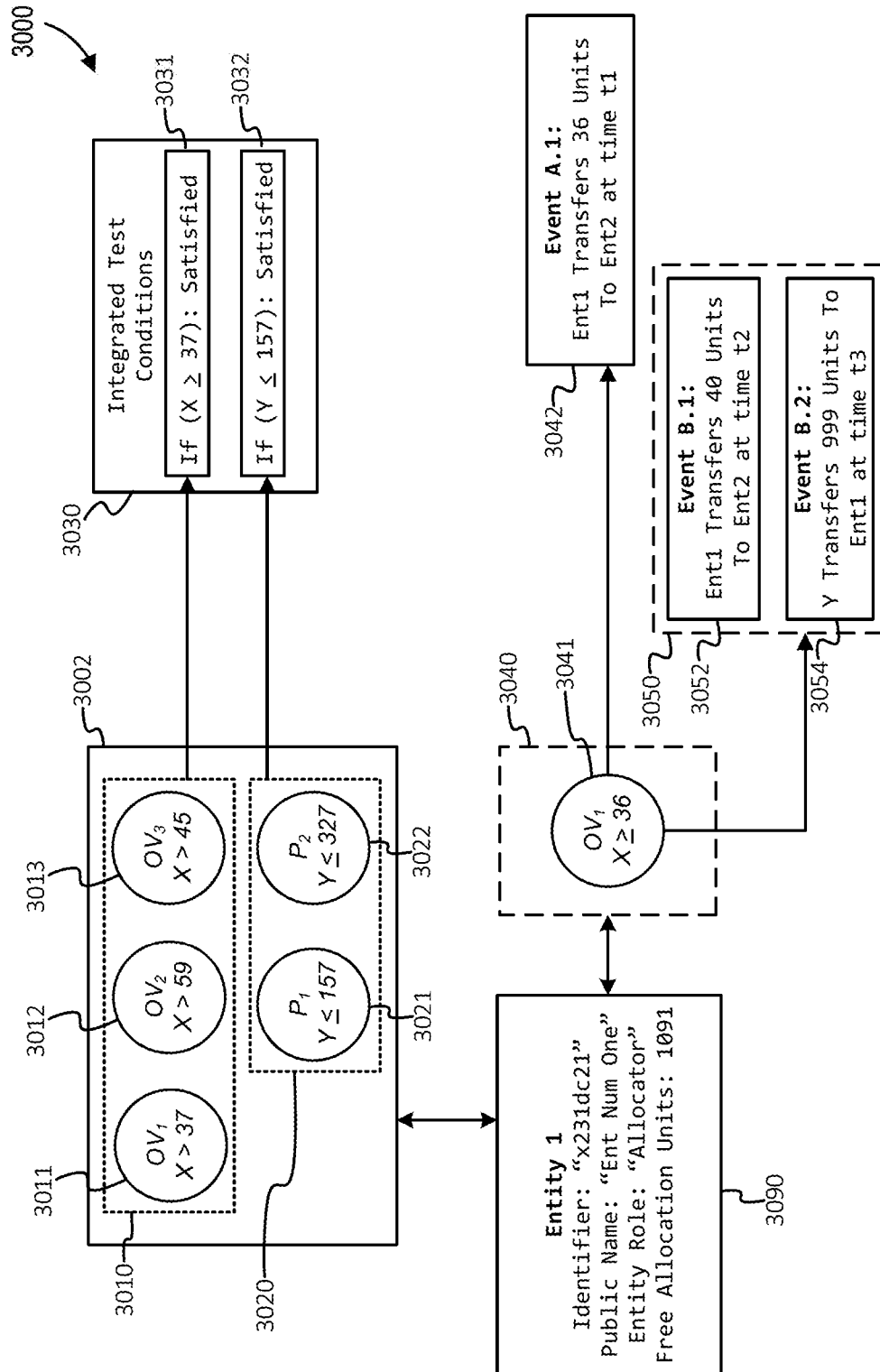


FIG. 16

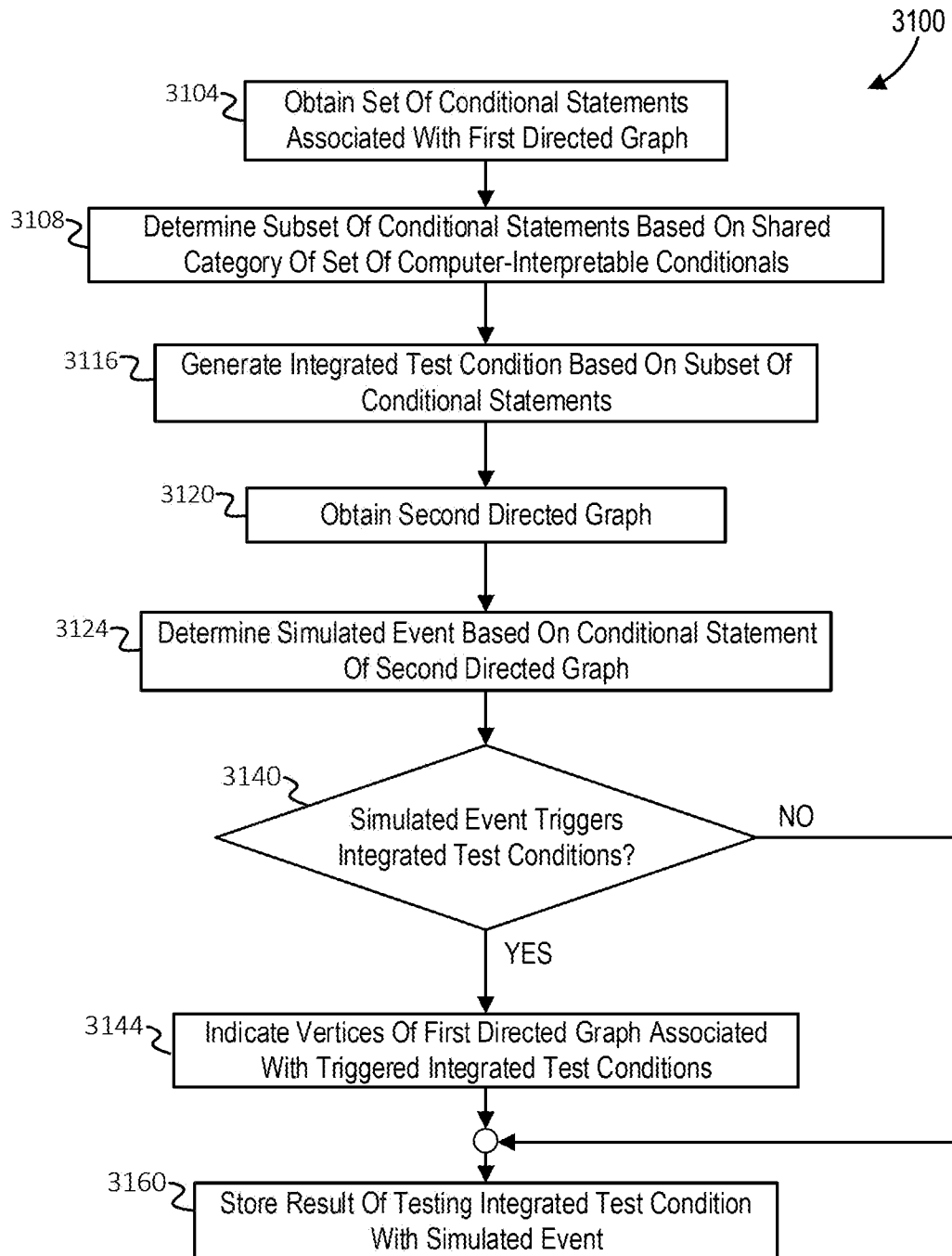


FIG. 17

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2020/049777**A. CLASSIFICATION OF SUBJECT MATTER****G06F 16/2455(2019.01)i, G06F 16/33(2019.01)i, G06F 16/182(2019.01)i, G06F 16/23(2019.01)i**

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

G06F 16/2455; G06F 17/30; G06F 7/00; G06Q 20/38; H04L 9/06; G06F 16/33; G06F 16/182; G06F 16/23

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Korean utility models and applications for utility models

Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

eKOMPASS(KIPO internal) & Keywords: smart contract, directed graph, program state, conditional statement, parameter, entity, criteria, update

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 2019-0050854 A1 (INTEL CORPORATION) 14 February 2019 claim 5	1-15
A	KR 10-2019-0092564 A (ALIBABA GROUP HOLDING LIMITED) 07 August 2019 claim 1	1-15
A	JULIEN M. HENDRICKX, 'Graphs and Networks for the Analysis of Autonomous Agent Systems', 29 February 2008, pp.1-251 [retrieved on 2020.11.24.]. Retrieved from < https://perso.uclouvain.be/julien.hendrickx/availablepublications/Thesis_Julien_Hendrickx.pdf > page 23	1-15
A	US 2019-0095909 A1 (NCHAIN HOLDINGS LIMITED) 28 March 2019 claims 1-15	1-15
A	US 2008-0126450 A1 (JUSTIN O'NEILL et al.) 29 May 2008 claim 3	1-15



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents:

"A" document defining the general state of the art which is not considered to be of particular relevance

"D" document cited by the applicant in the international application

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search

07 December 2020 (07.12.2020)

Date of mailing of the international search report

08 December 2020 (08.12.2020)

Name and mailing address of the ISA/KR

International Application Division

Korean Intellectual Property Office

189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea



Facsimile No. +82-42-481-8578

Authorized officer

YANG JEONG ROK

Telephone No. +82-42-481-5709



INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.

PCT/US2020/049777

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2019-0050854 A1	14/02/2019	CN 110971652 A DE 102019122933 A1	07/04/2020 02/04/2020
KR 10-2019-0092564 A	07/08/2019	AU 2018-212186 B2 BR 112019014589 A2 CA 3048574 A1 CN 107025559 A CN 107025559 B EP 3547241 A1 JP 2020-507840 A MX 2019008502 A PH 12019501498 A1 SG 11201906003 A TW 201828220 A US 10782998 B2 US 10817330 B2 US 2019-0310878 A1 US 2020-0174824 A1 US 2020-0285507 A1 WO 2018-137564 A1	10/09/2020 18/02/2020 02/08/2018 08/08/2017 18/09/2020 02/10/2019 12/03/2020 13/09/2019 08/06/2020 27/08/2019 01/08/2018 22/09/2020 27/10/2020 10/10/2019 04/06/2020 10/09/2020 02/08/2018
US 2019-0095909 A1	28/03/2019	BR 112018071814 A2 CA 3019267 A1 CN 109074566 A EP 3449446 A1 GB 2564207 A JP 2019-515534 A KR 10-2019-0004310 A PH 12018502269 A1 SG 11201808758 A WO 2017-187395 A1	19/02/2019 02/11/2017 21/12/2018 06/03/2019 09/01/2019 06/06/2019 11/01/2019 09/09/2019 29/11/2018 02/11/2017
US 2008-0126450 A1	29/05/2008	AU 2007-325567 A1 CN 101542467 A EP 2087436 A1 JP 2010-511249 A KR 10-2009-0085135 A WO 2008-067018 A1	05/06/2008 23/09/2009 12/08/2009 08/04/2010 06/08/2009 05/06/2008