Supporting Trust in Hybrid Intelligence Systems Using Blockchains

Hans-Georg Fill, Felix Härer
Digitalization and Information Systems Group
Department of Informatics, University of Fribourg
Bd. de Pérolles 90, 1700 Fribourg, Switzerland
hans-georg.fill@unifr.ch, felix.haerer@unifr.ch

Abstract

The combination of techniques from machine learning and knowledge engineering can lead to new types of information systems for processing data and knowledge by machines. A further step is to add humans in the loop as proposed by Hybrid Intelligence for amplifying human intellect and enabling machines to learn from humans. It then becomes essential to understand the provenance of data and knowledge and trace the accountability of humans and machine-based agents. This is a major prerequisite to establish trust in such systems. Thus, we propose a framework for addressing this aspect using blockchains as a trustful, decentralized ledger. We discuss how blockchains can support the attestation of patterns of data, knowledge, algorithms and human interventions as well as relations between these components. Furthermore, the search for existing patterns can be realized.

Introduction

The field of artificial intelligence has traditionally regarded *symbolic* as well as *non-symbolic* approaches for representing and processing knowledge (Russell and Norvig, 2012; Dorffner, 1991). Whereas the first direction aims to represent knowledge in the form of symbols and mechanisms for efficiently searching, rearranging or manipulating these symbols, non-symbolic approaches assume that machines can acquire knowledge through interactions with their environment. In these approaches, purely numeric representations are typically applied by recognizing desirable patterns, classifying, or clustering large amounts of data. Based on the resulting numerical models, similar patterns, clusters or classifications can be subsequently identified in other data.

However, both directions have deficiencies. Symbolic approaches are often limited by their requirement to express knowledge structures in rigid form that is amenable to machine-processing. Therefore, they typically revert to logic-based formalisms that fall short of representing fuzzy and heuristic aspects. On the other hand, numeric representation approaches rely on past data in large volumes and their results are hard to introspect. Therefore, it has been argued to combine both types of representation for mutual benefits (Minsky, 1991).

Recently, the discussion on such combinations has been taken up again. It could be shown that today several domains make use of AI systems that join machine learning techniques with knowledge reasoning approaches (Martin et al., 2019; Harmelen and Teije, 2019). In addition, the wide use of AI systems and their already large influence on our daily lives has raised the issue of trust. Particularly, cases where data and algorithms were used in ways that led to racial or gender biases have been extensively discussed in the media, e.g. (Obermeyer et al., 2019; Buolamwini and Gebru, 2018). Also in the context of knowledge reasoning, trust has been a core aspect. The provenance of data and the deductive processes used were identified as essential to decide whether results can be trusted, e.g. McGuinness (2004). In artificial intelligence, these considerations have been extended towards ethical considerations. It is claimed that machine-based reasoning should not only be transparent and able to explain itself but also consider societal values, moral and ethical aspects and the priorities of stakeholders in different cultural contexts (Dignum, 2018).

Another direction mentioned early in literature adds further benefits in terms of intelligent information processing. Whereas the goal of artificial intelligence is and has been the perfection of machines and their capabilities in solving tasks intelligently, the idea of *hybrid intelligence systems* promoted socio-technical systems that adapt interactively (Lomov and Venda, 1977). Thereby, particular human capabilities such as creativity, common sense, empathy or ethical responsibility are integrated with machine learning and reasoning approaches (Dellermann et al., 2019). Such *human-in-the-loop* systems have been described for example in the medical domain for enabling interactive machine learning (Holzinger, 2016) or by using visual analytics for letting humans interpret complex machine learning results (Hund et al., 2016).

Following these developments of combining machine learning and knowledge reasoning systems and the human-in-the-loop concept, the question arises how trust can be established for these hybrid intelligence systems. Despite the great technical capabilities offered by them, it needs to be ensured that their results match the expected outcomes in terms of reliability within a specific context. As already mentioned, the knowledge about the provenance of information is a central aspect to permit trust decisions (Artz and Gil, 2007), as well as the accountability, responsibility and transparency of how information is processed as discussed in the context of ethical AI (Dignum, 2017). Consequently, trust can be technically supported through security mechanisms in the sense of *policy-based trust* where the access to and the origin of information is regulated, e.g. using certificates to identify human and machine agents, as well as through the history of past interactions as apparent by *reputation-based trust* (Bonatti et al., 2005). In the following we assume that in both cases, humans are responsible for the initial creation of hybrid intelligence systems, the use of the right data, and the behavior these systems may exhibit - at least to the extent where they can be held accountable, which is not always clearly decidable in case of autonomous systems (Matthias, 2004).

Based on this assumption, it seems essential to make the composition of hybrid intelligence systems transparent so that everyone can verify the origin of data, algorithms, and human interventions as well as the usage of these components. Therefore, the provenance of the according information has to be recorded in a secure and tamper-proof fashion. Ideally, this should be done in an open accessible and thus transparent way that can be verified by any party. This would permit ensuring the traceability and thus the responsibility for all components of hybrid intelligence systems. As a means for meeting these requirements we consider in the following properties of blockchains. These permit the transparent, immutable, tamper-proof, and decentralized storage of information based on distinct consensus protocols and thus seem well-suited for supporting these tasks. For this purpose we discuss a framework for attesting the components and the results generated by hybrid intelligence systems on blockchains, thereby contributing to trust in these systems.

The paper is structured as follows: we will discuss foundations on the combination of machine learning and knowledge reasoning and explain the concepts behind hybrid intelligence. This will be followed by a brief characterization of blockchains. Subsequently, a framework for attesting hybrid intelligence components and their execution on blockchains will be described, followed by corresponding realization requirements in the form of smart contracts in pseudo-code. Finally, we will discuss execution options and discuss possible usage scenarios of the intended approach.

Foundations

In this section we briefly discuss foundations on the combination of machine learning and knowledge reasoning, hybrid intelligence, and blockchains to familiarize readers with the core concepts of these topics.

Combination of Machine Learning and Knowledge Reasoning

Machine learning and knowledge reasoning techniques are traditionally positioned opposite to each other, considering their methods and modes of operation. In particular, techniques of the former category are concerned with inductive learning on the basis of data without known internal structures. This kind of ”model-free” representation, a term found in (Pearl, 2018) is the input and output of the learning process. In its purest form, internal and unknown structures model everything obtained from learning. On the opposite side of the spectrum are knowledge reasoning techniques, which are concerned with deductive reasoning on explicit ”model-based” representations. Subject to logic, the explicit symbolic representation is used to infer new knowledge.

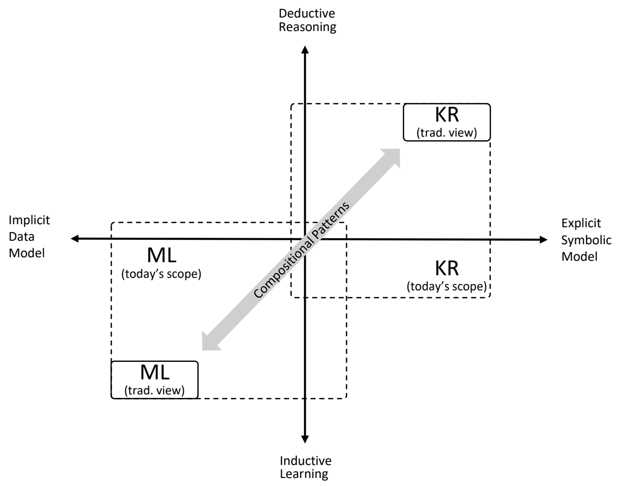

Figure 1: Machine Learning and Knowledge Reasoning

However, while the dichotomy is obvious when considering techniques such as multi-layer (deep) neural network architectures and OWL description logic, the scope of both machine learning and knowledge reasoning is becoming broader and partially overlapping (Harmelen and Teije, 2019), such that separate representational and processing dimensions can be assumed instead of a spectrum - see Figure 1. For example, artificial neural networks may be augmented with memory, computation and attention concepts (Vaswani et al., 2017; Hochreiter and Schmidhuber, 1997). Furthermore, Harmelen and Teije argue that recent approaches combine both techniques and can rather be described by the composition of their components. The authors describe a ”boxology” for this purpose, consisting of Machine Learning (ML) and Knowledge Reasoning (KR) elements, which can be combined with Data (Data) and Symbolic (Sym) input and output representations. For example, a basic combination of KR and ML with Sym and Data input as well as Sym and Data output is shown in Figure 2. More complex configurations encompass intermediate abstractions for learning and reasoning, the explanation of learning and reasoning, and meta-reasoning. In particular, the design of patterns with humans in the loop, as discussed later on, is of interest towards explainable AI and accountability.

Hybrid Intelligence

In the original conception of *hybrid intelligence systems*, humans were positioned as central figures where mechanical

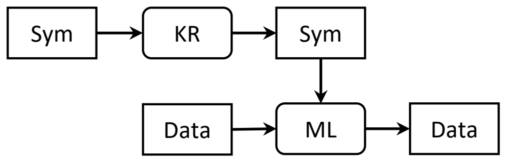

Figure 2: Boxology pattern for machine learning that considers also symbolic information on which reasoning has been applied (Harmelen and Teije, 2019, pattern (13))

components only act as tools for them (Lomov and Venda, 1977). The focus lied on the interaction between humans and machines as socio-technical systems, as opposed to purely technical systems in artificial intelligence. More recently, these concepts have seen a revival (Kamar, 2016a,b), including the allocation of considerable research funds (HI, 2019). Today, hybrid intelligence (HI) is regarded as the utilization of the particular strengths of humans and machines in such a way that individual or collective human intelligence is combined with artificial intelligence. Dellermann et al. define hybrid intelligence as the "the ability to accomplish complex goals by combining human and artificial intelligence to collectively achieve superior results than each of them could have done in separation and continuously improve by learning from each other." (Dellermann et al., 2019, p. 276).

It can be further distinguished between four sub-fields that are currently being investigated (HI, 2019): *collaborative HI*, where it is focused on the synergy of humans and intelligent agents for solving tasks, *adaptive HI*, that targets situations that have not been anticipated by the designers, e.g. in terms of variable team configurations and changing roles, *explainable HI*, where humans and machine agents need to explain their recognitions, goals, and actions to each other, and *responsible HI*, that addresses ethical and legal concerns as an integral part of HI systems.

As a consequence, in hybrid intelligence systems, human agents need to be considered on the same level as machine learning algorithms and reasoners, with distinct requirements in terms of information representation. Possible interfaces between humans and machines may be visualizations for representing results from machine calculations, e.g. (Hund et al., 2016), or human-adequate knowledge and data representations that can serve as input for machine learning and knowledge reasoning as e.g. found in conceptual modeling (Fill, 2017; Karagiannis and Buchmann, 2016).

Blockchains

Blockchains are a class of technologies for implementing distributed systems with verifiable storage and execution based on integrity-secured backward-linked blocks consisting of transactional data (Härer, 2019). In contrast to distributed database systems, blockchain systems allow for the public distribution and transparent validation of data among decentralized network participants. In addition to public blockchains, permissioned forms that limit write access exist as well as fully private variants thereby blurring the lines to the field of distributed databases.

While the initial innovation of a protocol for the verifiable storage and execution of monetary transactions has evolved to a general execution of smart contract programs (Buterin, 2013), the original proof-of-work mechanism for ensuring consensus among the peers of the underlying network remains largely unchanged so far. By this mechanism, the following properties can be established given a sufficiently large portion of mining nodes performing computational work which continuously verifies and proves past executions according to the protocol:

- *Integrity*: each block carries a hash value of the previous block, ensuring the integrity of all prior blocks.
- *Immutability*: the data structure is replicated throughout the peers of the work where integrity checking does not allow for changes of past blocks.
- *Traceability*: each transaction performed is recorded in a specific block, usually with a numeric identifier. A time stamp as part of the block's data specifies an approximate creation date and time for transactions.
- *Identification*: signatures bind the identity of users to blockchain addresses acting as source and destination of transactions. This property refers also to non-repudiation, i.e. the binding of a transaction to its source and destination is definitive.
- *Autonomous execution*: program code can be run as a smart contract using an instruction set specified by the protocol. In this way, the execution possesses the verifiability properties of the protocol.
- *Incentivized Operation*: rewards are issued to the peers performing the proof-of-work computations.

In such a system, the notion of trust therefore refers a single point of truth being established such that data and operations can be traced back to individual peers through digital signatures. In this context, blockchains are sometimes considered to establish trust without intermediaries.

Applications utilize the properties for example in the attestation of identities, knowledge, information or data (Härer and Fill, 2019), e.g. by certifications verifiable with an untrusted third party on the blockchain. Similarly, algorithms can be registered for providing transparency, in order to establish their integrity at a later point in time by another party. First concepts involving algorithms for the benefit of trust have been suggested for explainable artificial intelligence (XAI) at this point (Calvaresi et al., 2019; Nassar et al., 2020).

Regarding implementation, the discussion in this paper assumes a blockchain capable of smart contracts, such as Ethereum (Buterin, 2013; Wood, 2014). Here, smart contracts are byte code programs, stored and executed in autonomously operated contract accounts (CA), in contrast to externally owned accounts (EOA) defined by the public-private key pairs of users. Towards establishing trust, this platform constitutes a base layer for the identification, traceability and attestation of data and higher-level concepts.

Framework

Based on the aforementioned foundations we can now advance to describing our framework for supporting trust in hybrid intelligence systems through blockchains. The central idea thereby is to enable the traceability and thus the provenance of all components of hybrid intelligence systems. Thus, it needs to be recorded, which human agents, machine agents, data, and symbolic representations exist and how they interact for generating results. This can be done both at *design time*, i.e. when new hybrid intelligence systems are conceived, as well as at *run time* when concrete executions are observed. Thus it can be investigated, how results have been generated by HI systems and who can be held accountable for them.

For structuring the framework we reverted to the boxology elements proposed by Harmelen and Teije for combining machine learning and knowledge reasoning (Harmelen and Teije, 2019), i.e. we include components for knowledge reasoning (KR), machine learning (ML), model-free representations (Data) and symbolic representations (Sym). We extend it by adding human agents (HA) to allow for the construction of human-in-the-loop systems. All components can be related to each other through a relation element as shown in Figure 3. The recording on a blockchain then takes place in the form of *attestations*, which can either be triggered through human agents or machine agents.

In the following sections, the components and relationships of the framework are discussed. Subsequently, we will specify smart contracts for conducting the mentioned attestations and discuss possible realization options.

Components and Relations

The following sections argue for designing knowledge reasoning and machine learning systems with the explicit involvement of human agents for utilizing complementary strengths and as a source of trust and accountability. For this reason, the pattern-based specification of such a system is detailed here by its components and relations. They comprise Human Agents (HA), Knowledge Reasoning (KR) and Machine Learning (ML) components as well as representational Data and Symbolic (Sym) components and their relations. The notation for processing components and representational components can be seen in Figure 3.

In order for HA to be the source of trust and accountability of a specific pattern, its components are subject to attestation on a blockchain where they are linked to the identities of human agents. Subsequently, algorithms and the execution with concrete representations are subject to attestation triggered by humans or machines.

Initialization through Human Agents In contrast to views where machines are held accountable for their actions, this approach assumes the source of trust to be a human agent. Acting as a designer for a specific pattern and its algorithms, a human agent is able to provide explainability of the design, human comprehension, emotional understanding in decisions concerning moral and ethics as well as accountability. For the identity to be linked to the design, it needs to fulfil three requirements. (1) The identity must be unique, (2) the existence of the identity must be publicly verifiable, and (3) it must be linked to a human identity for accountability and for others to trust it.

However, the role and benefit of human involvement through comprehension, emotional understanding and other factors is not to be reduced to accountability, as it might be considered the least desired factor when algorithms are executed autonomously with little control of human agents.

One possibility is the registration of self-managed identities on a blockchain, proposed in the form of self-sovereign identity concepts (Lundkvist et al., 2016). At a minimum, (1) and (2) can be realized by public-private key pairs belonging to externally owned accounts on a blockchain such as Ethereum, which can be extended with additional personal attributes if required. The link to a human identity (3) has to be proven outside of the blockchain system, e.g. through digital signatures from government-issued electronic identity such as eID in Europe (Shehu, Pinto, and Correia, 2019).

For the purposes of discussing the framework, the following initialization through a human agent HA is assumed prior to the design of a pattern with its implementing algorithms and data:

- Generation of an externally owned account $\mathsf{EOA} = (\mathsf{EOA}_{\mathsf{Sec}}, \mathsf{EOA}_{\mathsf{Pub}}, \mathsf{EOA}_{\mathsf{A}})$ where $\mathsf{EOA}_{\mathsf{Sec}}$ is a private key from which a public key $\mathsf{EOA}_{\mathsf{Pub}}$ and a public account address $\mathsf{EOA}_{\mathsf{A}}$ are derived. In principle, any form of a public identifier ID might be provided.
- Creation of a signature S using an electronic ID certificate identified by $\mathsf{eID}_{\mathsf{Pub}}$ such that a message $\mathsf{M} = \mathsf{EOA}_{\mathsf{A}}$ is signed and verifiable with S through the certificate authority of the eID.
- Registration of S and M with a smart contract through a blockchain transaction originating from $\mathsf{EOA}_{\mathsf{A}}$, proving that the identity knows $\mathsf{EOA}_{\mathsf{Sec}}$.
- Future transactions for design time and run time attestations originating from an address $\mathsf{EOA}_{\mathsf{A'}}$ are valid if a signature $\mathsf{S'}$ is found for $\mathsf{EOA}_{\mathsf{A'}}$ in the smart contract and if $\mathsf{S'}$ can be verified, proving the identity is $\mathsf{eID}_{\mathsf{Pub}}$.

Following the design of a pattern, attestation transactions triggered by human or machine agents are bound to HA. Individual transactions for components and relations are carried out according to the following sections.

Machine Agents The term machine agent here refers to the processing of algorithms for knowledge reasoning and machine learning. At design time, this concerns the traceability of the components KR and ML with their implementing algorithms over time.

In principle, two dimensions might be considered for recording and attesting algorithms over time. (1) the *abstraction level*, ranging from securing the algorithm in its source code representation as a whole to a fine-grained representation of individual syntactic elements (Falleri et al., 2014; Fluri et al., 2007), and (2) the *differencing approach*, distinguishing a state-based or operation-based representation of changes (Koegel et al., 2009; Lippe and van Oosterom, 1992). State-based approaches permit showing the

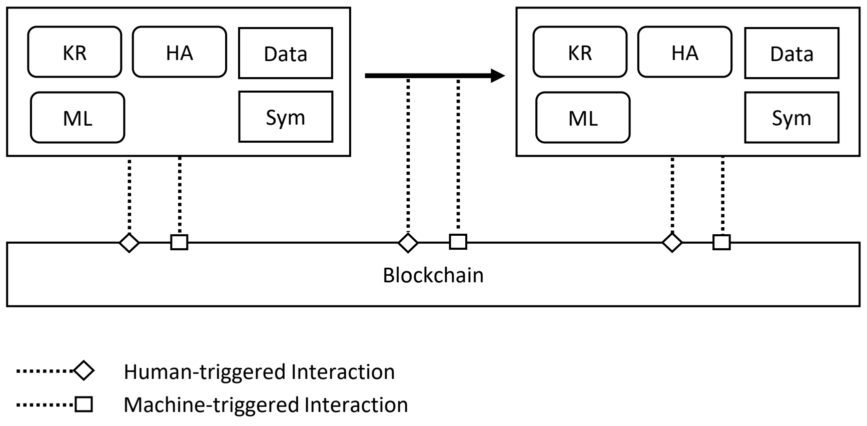

Figure 3: Framework for describing the components and relationships which utilize the blockchain in human-triggered or machine-triggered attestations. The components are based on the design pattern proposed by (Harmelen and Teije, 2019) and extended to human agents.

differences between individually recorded states of an algorithm's source code. Implementations of this approach in version control systems such as Git[1] require an intentional commit for recording the current state. Given two states, it is only possible to reconstruct their differences. In contrast, operations-based approaches permit the reconstruction of change operations made between two states by recording all operations or atomic state changes individually. Instead of detecting deletions and insertions of source code, individual operational changes to syntax elements can be detected, however, versioning needs to be aware of the syntax and language used.

In order to allow changes to be publicly observed and traced on a blockchain, existing versioning approaches can be adapted for storing individual states or operations. Given the source code of an algorithm, the following attestation is assumed:

- Assignment of a unique identifier ID, optionally in the form of a resource locator for providing public access to the algorithm. For example, UUID version 4 (Leach, Mealling, and Salz, 2005) might be used to locally assign a randomly generated identifier, possibly as part of a URL.
- Calculation of a set of hash values H_A for each abstraction of the algorithm's source code, e.g. code blocks defining a state or individual operations.
- Recording of the pattern component type $c \in \{\mathsf{KR}, \mathsf{ML}\}$, ID, H_A, and the current block number in a smart contract given that a valid signature of HA is provided.
- Changes to the chosen abstraction can be detected publicly by a change of any of the values in H_A compared to the versions stored previously.

Data The representational components Data and Sym concern the run time of algorithms according to a given pattern. Given the notion of model-free data without knowledge of its internal composition, the requirement is for it to be traceable regarding its provenance and changes over time.

The storage of data on a blockchain has several limitations. While it is theoretically possible to store the input and output data of algorithms on a blockchain as a whole, the data volume, veracity, and velocity (Laney, 2001) impose requirements difficult to meet for distributed data management systems and particularly blockchain systems. For example, large machine learning data sets are not suited for today's blockchains due to the limited number of transactions per second and block size (Kim, Kwon, and Cho, 2018). While data availability therefore cannot be assumed, traceability might be realized without it through the attestation of data over time. On the basis of the attestation concept, traceability can be achieved through the issuance of a permanent identifier in combination with its binding to a human or machine agent. In addition, the identifier here is also assumed to locate data through traditional client-server-based access.

The following scheme is assumed for the human- or machine-triggered attestation of data:

- Assignment of a unique identifier ID similar to the attestation of algorithms, e.g. using a UUID and, possibly, a URL which contains it.
- Calculation of $\mathsf{H}_{\mathsf{Data}}(B)$ as a cryptographic hash function, e.g. (Dworkin, 2015), applied to binary data B.
- Recording of the representational component with ID, $\mathsf{H}_{\mathsf{Data}}(B)$, the component's type Data, and the current block number in a smart contract given that a valid signature of HA is provided.
- In future utilizations of B, its integrity is considered valid if the re-computed value $\mathsf{H}_{\mathsf{Data}}(B)$ is present in the smart contract. In this case, an attestation is provided through the smart contract and additional information can be retrieved from it. In particular, the date and time when the data was recorded according to the block number, an iden-

[1]https://git-scm.com/

tifier and, possibly, locator, and the provenance according to the signature of HA can be retrieved.

Sym The Sym component stands for symbolic knowledge representations that are used as inputs and outputs of classical reasoning systems as defined in (Harmelen and Teije, 2019). This includes for example ontologies, rules, knowledge graphs or linked data.

For the attestation of Sym, the applicability of the data attestation scheme outlined in the previous section is assumed. However, by considering the internal structures, more fine-grained attestations become possible. Depending on the concrete representation, e.g. a resource description framework (RDF) graph, logical expressions, or an ontology, an attestation requires an appropriate abstraction level. One example is the attestation of ontologies on the basis of Knowledge Blockchains as outlined in (Fill, 2019; Fill and Härer, 2018). Other knowledge representations such as RDF triplets can be subject to attestation in a similar fashion. Given a method $\mathsf{H}_{\mathsf{Sym}}(K)$ for observing an abstraction of a knowledge representation K over time, the following attestation scheme is assumed:

- Assignment of a unique identifier ID similar to the attestation of algorithms, e.g. using a UUID and, possibly, a URL which contains it.
- Calculation of $\mathsf{H}_{\mathsf{Sym}}(K)$ as a cryptographic hash function applied to a knowledge representation K. Given the example of ontologies, $\mathsf{H}_{\mathsf{Sym}}(K)$ represents the root hash value of a Merkle tree that is created from the data in an ontology - see (Fill, 2019) for details.
- Recording of the representational component Sym, ID, $\mathsf{H}_{\mathsf{Sym}}(K)$ and the current block number in a smart contract given that a valid signature of HA is provided.
- Similar to the attestation of Data, K is considered valid in future utilizations if $\mathsf{H}_{\mathsf{Sym}}(K)$ can be established, e.g. by reconstruction of a Merkle tree resulting in this particular value. However, K might be any fine-grained representation here, e.g. classes of a subclass relation of an ontology. For each abstraction K, the date and time, the block number, an identifier and, possibly, locator, and the provenance according to the signature of HA can be retrieved.

Relations between Components The design of a pattern is finalized by the specification of its relations between the components established previously. After the aforementioned attestations of individual components, their pairwise specification on the basis of the assigned attestation identifiers is required. Therefore, relations in the form of $(\mathsf{ID}_{C1}, \mathsf{ID}_{C2})$ for component pairs $\mathsf{C1}$ and $\mathsf{C2}$ are recorded on the blockchain for completing the design of a pattern.

In this process, the pairwise specification of components is restricted by the components' types and the combinations permitted for them. Primarily, the constraints are imposed by the notion of processing. Components of this type imply an input or output to be present in the form of a representational component. According to the patterns stemming from the literature analysis by Harmelen and Teije (2019), the knowledge reasoning (KR) and machine learning (ML) components can be found in combination with Data or symbolic (Sym) representations. Table 1 summarizes possible relations between the components.

		Processing		
		Human Agent (HA)	Knowledge Reasoning (KR)	Machine Learning (ML)
Repre-sentation	Data	(1)	(3)	(5)
	Sym	(2)	(4)	(6)

Table 1: Possible Relations between Representation and Processing Components of the Framework

Knowledge reasoning systems are usually designed for Sym representations as input and output (3). Additional systems were found (Harmelen and Teije, 2019) for relation (4) in two instances, where raw data and symbolic input was applied in combination for KR (Pattern 11) and in another case where input data for ML was also used as input and output of KR in order for KR to try to interpret and explain the abstractions gained from machine learning (Pattern 8).

Machine learning systems mostly process model-free Data representations as input and output (5). However, there exist a variety of systems using relation (6) (Harmelen and Teije, 2019). In particular, systems operating on symbolic inputs may also produce symbolic output through learning (Pattern 3 and 11) or an intermediate data output in case of embeddings, which are an input for ML again in order to produce Sym (Pattern 4). Other examples include the learning of ontologies (Sym) from Data inputs (Pattern 5), Sym output for explanation of ML (Pattern 6 and 7), the production of Sym output by ML to prepare learning or reasoning (Patterns 9 and 10), learning with Data an additional Sym input (Patterns 12 and 13), and for ML to learning knowledge reasoning using Sym inputs and outputs (Pattern 15). As an exceptional case, there also exist complex relations where Sym is composed out of multiple components (Pattern 16).

For the design of patterns involving the learning and reasoning of human agents (HA), their relation to other HA, KR, or ML components is indirectly made through explicit knowledge, usually in the form of Sym representations, leading to relation (2). However, knowledge of an inherent structure cannot always be assumed, such that Data and Sym are possible components in relation (1) to and from HA.

Realization Requirements for Smart Contracts

Based on the outlined process for the attestation of individual components with their relations, the following smart contract details the data required and operations necessary to implement attestations in practice. Blockchain platforms supporting the execution of smart contracts in a manner similar to Ethereum or Hyperledger are suited for implementing the abstract specification provided in the following paragraphs.

Firstly, the data structures required are shown in part 1 of the smart contract listing. Here, the smart contract establishes a typing system in the form of enumerations (line 1 ff.), abstract data types with according data structures (line 4 ff.) and global variables for mappings between the abstract data types provided (line 31 ff.). The typing system distinguishes human agents (HA), machine agents (MA) of processing type KR and ML and the representational types (R) Data and Sym. Individual HA's identity data in the according data type requires the storage of a signature, an externally owned account (EOA) as well as a block number indicating when attestations have been conducted. Similarly, MA must record an address for traceability to HA with a block number, in addition to the attestation hash value, the processing type and a URL. Representations use the same structure which only differs in the representation type. Relations are stored as tuples of components where each component is identified by a UUID. In addition, the attestation block and address with the component type are recorded to allow for lookups of a UUID without type information. Global variables are defined for mapping data structures to relate each UUID to one HA, MA, representational type or relation.

Smart Contract Part 1: Data Types and Mappings

```
1  Enum ComponentType { HA, MA, R }
2  Enum ProcessingType { KR, ML }
3  Enum RepresentationType { Data, Sym }
4  DataStruct HumanAgent {
5      Signature sig;
6      Address eoa;
7      Integer block;
8  }
9  DataStruct MachineAgent {
10     ProcessingType type;
11     ByteArray hashValue;
12     Address addr;
13     Integer block;
14     URL url;
15 }
16 DataStruct Representation {
17     RepresentationType type;
18     ByteArray hashValue;
19     Address addr;
20     Integer block;
21     URL url;
22 }
23 DataStruct Relation {
24     UUID uuidFrom;
25     UUID uuidTo;
26     ComponentType typeFrom;
27     ComponentType typeTo;
28     Address addr;
29     Integer block;
30 }
31 Map (UUID => HumanAgent) humanAgent;
32 Map (UUID => MachineAgent) machineAgent;
33 Map (UUID => Representation) representation;
34 Map (UUID => Relation) relation;
```

The operations required to perform attestations are outlined in parts 2 - 5. In the registration of HA in part 2, a signature needs to be provided by HA. A UUID is randomly generated and the sender address is read from the transaction (2) such that it can be the subject of a signature validation. I.e., the signature including a public key of HA and the signed address of the sender must be valid. In this case, the aforementioned data is recorded under the assigned UUID.

Smart Contract Part 2: HA Identity Registry

```
1  Function identityRegistry(sig: Signature) : void {
2      Address snd = Transaction.sender;
3      if signatureValidation(sig, snd) then
4          UUID uuid = generateRandomUUIDV4();
5          humanAgent[uuid].block = Block.nr;
6          humanAgent[uuid].eoa = snd;
7          humanAgent[uuid].sig = sig;
8      end
9  }
```

After the initialization by HA is conducted, the identity data stored for it is retrieved and validated for every registration of algorithms, representations, and relations in parts 3 - 5. The validation occurs in the same fashion in these parts. Considering, e.g. the registration of algorithms in part 5, line 5, the two requirements for registration can be seen. In particular, the transaction sender must match the HA referenced by its UUID and the signature has to be valid according to the properties mentioned in the previous paragraph. With a randomly generated UUID for MA, the algorithms' hash value, processing type, URL and sender are stored. In the same manner, the recording of representations of the types Data and Sym can occur as in part 4.

Smart Contract Part 3: MA Algorithm Registry

```
1  Function algorithmRegistry(type: ProcessingType,
     data: ByteArray, url: URL, uuidHA: UUID) : void {
2      Address snd = Transaction.sender;
3      Address eoa = humanAgent[uuidHA].eoa;
4      Signature sig = humanAgent[uuidHA].sig;
5      if snd == eoa && signatureValidation(sig, eoa)
         then
6          UUID uuid = generateRandomUUIDV4();
7          machineAgent[uuid].block = Block.nr;
8          machineAgent[uuid].addr = snd;
9          machineAgent[uuid].type = type;
10         machineAgent[uuid].url = url;
11         machineAgent[uuid].hashValue =
             hashFunction(data);
12     end
13 }
```

The registration of relations in part 5 takes the pair of components by their UUID and types as parameters, together with the discussed identifier of HA. Another UUID is generated in order to store the relation itself. The retrieval

Smart Contract Part 4: Representation Registry

```
Function representationRegistry(type: RepresentationType, data: ByteArray, url: URL, uuidHA: UUID) : void {
    $Address$ snd = Transaction.sender;
    $Address$ eoa = humanAgent[uuidHA].eoa;
    $Signature$ sig = humanAgent[uuidHA].sig;
    if snd == eoa && signatureValidation(sig, eoa) then
        $UUID$ uuid = generateRandomUUIDV4();
        representation[uuid].block = Block.nr;
        representation[uuid].addr = snd;
        representation[uuid].type = type;
        representation[uuid].url = url;
        representation[uuid].hashValue = hashFunction(data);
    end
}
```

of a relation by UUID therefore yields access to the components provided by their typing information through the mapping data structures in part 1, line 31 ff..

Smart Contract Part 5: Relation Registry

```
Function relationRegistry(uuidFrom: UUID, uuidTo: UUID, typeFrom: ComponentType, typeTo: ComponentType, uuidHA: UUID) : void {
    $Address$ snd = Transaction.sender;
    $Address$ eoa = humanAgent[uuidHA].eoa;
    $Signature$ sig = humanAgent[uuidHA].sig;
    if snd == eoa && signatureValidation(sig, eoa) then
        $UUID$ uuid = generateRandomUUIDV4();
        relation[uuid].block = Block.nr;
        relation[uuid].addr = snd;
        relation[uuid].uuidFrom = uuidFrom;
        relation[uuid].uuidTo = uuidTo;
        relation[uuid].typeFrom = typeFrom;
        relation[uuid].typeTo = typeTo;
    end
}
```

Newly registered identities, algorithms, representations, and relations can be monitored over time by anyone with access to the blockchain that contains the smart contract with the global variables *humanAgent*, *machineAgent*, *representation*, and, *relation* (part 1, line 31 ff.). After a registration and its UUID have been observed, an attestation can be carried out by any third party through the per-UUID retrieval of a global variable and the validation of its values. In the case of an identity, the validation is determined by checking the signature in *humanAgent* which must be a valid signed message of the EOA address also stored in *humanAgent*. For an algorithm or representation, the hash values stored in *machineAgent* and *representation* are required to match recomputed hash values of the abstractions used, e.g. syntax elements for algorithms, binary data and symbolic representations such as ontology classes. Lastly, relations are subject to validation by checking whether the components contained in *relation* exist. Each component is identified by a UUID required to be stored in *humanAgent*, *machineAgent*, or *representation*. Given a successful validation, the stored block number and identity address can be considered valid and might be retrieved additionally. A continuous monitoring and attestation process is easily carried out, triggered by receiving a new block from the network over the course of synchronizing the blockchain as usual.

Requirements and Options for Execution

At run time, the execution of algorithms might be fully or partially autonomous and require the ability of reviewing execution traces in order to allow for according design time changes.

Concerning traceability in the execution at run time, it is limited by the distributed execution environment. In general, there exist three approaches in this area. First, the execution might be *blockchain-based* if it occurs directly on the infrastructure of a blockchain. Secondly, a trusted execution outside the blockchain might be *software-based* through performing proofs. Thirdly, *hardware-based* executions in trusted environments might be employed.

In the case of a *blockchain-based* execution, an untrusted and global peer-to-peer network can be assumed, where the blockchain infrastructure extends beyond the boundaries of known internal networks. Therefore, the execution itself is traceable only in the form of a smart contract, which is verifiably executed by the nodes of the network. While the execution of (unsupervised) learning algorithms and the application of learned models through smart contracts is explored (Harris and Waggoner, 2019), the feasibility of running knowledge reasoning and, in particular, machine learning algorithms directly on a blockchain cannot be assumed for the general case due to the scalability limitations of blockchains.

For trusted execution in general, approaches on the run time level providing a verifiable execution of algorithms can be divided into software- and hardware-based approaches.

Software-based approaches rely on proofs calculated during the execution, such that either the execution instructions performed or the resulting data output can be verified. For example, zero-knowledge proofs have been used in trusted execution schemes independent of (Ben-Sasson et al., 2013) and specifically for blockchain-based execution (Morais et al., 2019). In a distributed execution environment, blockchain-based approaches are advantageous due to the global state and global verifiability they provide. Scalability limitations also impact verifiable execution for applications such as machine learning here, even though, specialized approaches are beginning to appear (Ghodsi, Gu, and Garg, 2017).

Hardware-based approaches concern the area of trusted computing in combination with blockchain infrastructures (Hardjono and Smith, 2019; Luo et al., 2019). Trusted platform modules and secure enclaves in processing units are

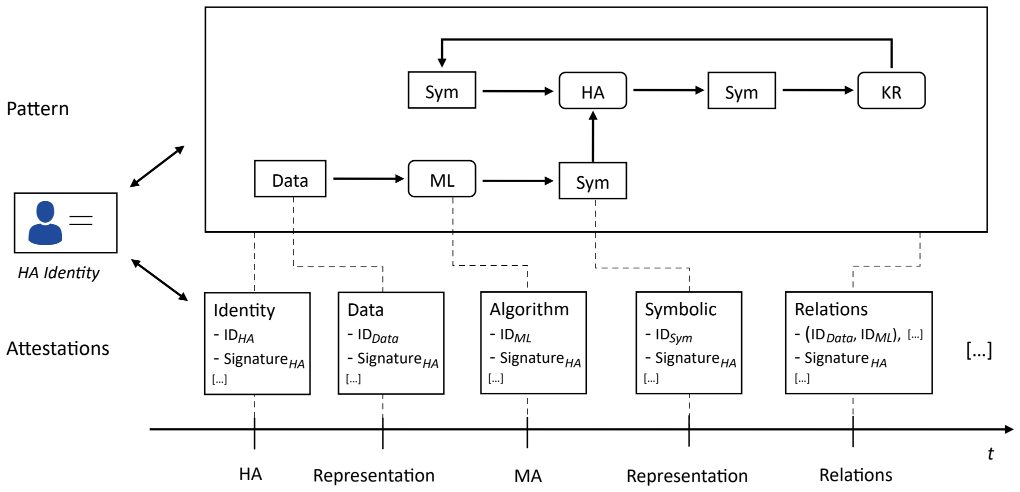

Figure 4: Example of a Hybrid System (Ontology Learning) with attestation of pattern and its instances

widespread today. In these realizations, there exists a trusted hardware element that provides a higher level of trust in the data stored and executions performed due to isolation from the main processing units and storage components. However, a variety of specialized execution instructions such as Intel SGX[2] and storage formats specific to individual platforms such as the Android keystore system[3] hamper standardization today.

Exemplary Use Cases

The construction of HI systems through the framework encompasses the specification of patterns, their attestation in smart contracts and an execution according to the aforementioned requirements. Use cases outlining the assumed benefits of this approach relative to current KR and ML systems are discussed in the following subsections. While the framework imposes significant design-time and run-time overhead, it might be applied in scenarios requiring transparency, distribution and trust. In particular, systems involving human and machine agents, e.g. considering algorithmic profiling, self-driving cars, or medical diagnoses. A pattern employed by these instances, where human and machine actors depend on each other, might be composed as in Figure 4. In general, the registration of an HA identity is carried out at first, followed by components and relations. The pattern assumed here consists of learning of an intermediate abstraction for reasoning, e.g. for creating and maintaining an ontology classifying diseases with data input processed through ML and oversight by human actors. Structurally, the pattern is also partially similar to pattern (10) (Harmelen and Teije, 2019), reminiscent of Alpha Go. However, it is extended here as an example for a learning and reasoning system with feedback and learning from a human agent. Similarly to pattern (10), Data is an input for an ML step to produce an intermediate Sym input for KR. However, we do not assume direct input to KR here. Instead, the Sym input first is subject to HA for quality control by help of an additional Sym input, e.g. from a domain ontology. With this additional knowledge, HA may choose to verify and augment the ML Sym output for reasoning through KR. Furthermore, results of the reasoning are used to enrich future Sym input for aiding HA. Similar to pattern (10), symbolic structures are constructed ultimately, however, with HA and KR as two intermediate abstractions. Assuming the attestation of this exemplary pattern, the following applications related to transparency, distribution and trust become apparent.

Example 1: Traceability Traceability pertains to the human agent involved as well as the components of the pattern at design time and run time.

Regarding the human agent, an identity is established through the externally owned blockchain account belonging to it. Using signatures, this account may be bound to another identity system such as officially issued eID cards.

The components and relations designed by the human agent are created on the basis of its identity. Initially, the HA identity is registered as an anchor for further attestations of Data, the ML and KR algorithms, the Sym representations following and all relations. In scenarios where accountability is required, e.g. for scenarios critical to security and human safety, the attestation of the design becomes relevant at run time. Even though ML and KR might be performed by machine agents in full or partial autonomy, their algorithms with input and output data are bound to their initial designer HA.

Example 2: Crowd Sourcing of Learning and Reasoning Due to their origin, most blockchains natively support the transfer of virtual currency. An externally owned account bound to an identity has a balance denominated in a currency such as Ether. Similarly, contract accounts in blockchains such as Ethereum hold a balance for making monetary transaction triggered by a smart contract.

[2]https://software.intel.com/en-us/sgx

[3]https://developer.android.com/training/articles/keystore

Once deployed, a smart contract can autonomously execute transactions compensating for machine learning and knowledge reasoning tasks performed by human agents or machine agents. In principle, the fully autonomous and reward-based creation of symbolic data and algorithms, e.g. ontologies and query logic, is possible with known identities. Depending on the incentive structure and game theoretical outcomes, the explicit involvement of human agents for quality control might be desirable considering a pattern such as Figure 4. For example, today, the collaborative creation of ontologies such as for the ICD uses KR manually through the Protégé software (Horridge et al., 2019). Here, according interfaces can be added for the collaborative creation of ontologies by multiple distributed parties with KR or ML capabilities.

Example 3: Trust With the identity system outlined, a reputation-based trust model as well as policy-based one can be implemented. Reputation-based trust is commonly employed on the internet today. The concept of a rating system for the determination of trust applies also to smart contracts, assuming a sufficient number of users. On the other hand, a policy-based trust can establish definitive rules for known identities when ML and KR tasks are executed by machine agents. With increasing autonomy, this aspect becomes relevant, for example, in applications critical to the safety of human users, e.g. considering the algorithms developed for self-driving cars. In this instance of policy-based trust, algorithms developed in the engineering phase and updates applied at a later point in time can be traced back to the initial development. In principle, the validity of data in any system can only be established to the extent of the real-world behavior observable and verifiable. Therefore, the transparent and tamper-proof record enables observation and validation to an extent, as it might be performed internally, by regulators, competitors, or the general public.

Conclusion and Outlook

The emergence of hybrid intelligence systems as combinations of knowledge reasoning and machine learning with human-in-the-loop, symbolic, and data representations offer great potentials. Towards the creation of trust in such systems, blockchains can aid in tracing the provenance of all involved components and their relations to their originators in the form of human agents. However, this ensures trust only on a technical level and in terms of accountability through attestation. The integration of further trust aspects such as ethical and moral responsibility is thereby implicitly covered to the extent of the involvement of humans. It will need to be further investigated how such aspects can be explicitly ensured, e.g. through verifying components that are to be attested. Further work will be required for analyzing potentially huge sets of pattern combinations and for deriving further meta-insights into their optimal usage.

References

Artz, D., and Gil, Y. 2007. A survey of trust in computer science and the semantic web. *Journal of Web Semantics* 5(2):58 – 71. Software Engineering and the Semantic Web.

Ben-Sasson, E.; Chiesa, A.; Genkin, D.; Tromer, E.; and Virza, M. 2013. SNARKs for C: Verifying Program Executions Succinctly and in Zero Knowledge. In Canetti, R., and Garay, J. A., eds., *Advances in Cryptology – CRYPTO 2013*, Lecture Notes in Computer Science, 90–108. Berlin, Heidelberg: Springer.

Bonatti, P.; Duma, C.; Olmedilla, D.; and Shahmehri, N. 2005. An integration of reputation-based and policy-based trust management. In *In Semantic Web Policy Workshop*.

Buolamwini, J., and Gebru, T. 2018. Gender shades: Intersectional accuracy disparities in commercial gender classification. In Friedler, S. A., and Wilson, C., eds., *Proceedings of the 1st Conference on Fairness, Accountability and Transparency*, volume 81 of *Proceedings of Machine Learning Research*, 77–91. New York, NY, USA: PMLR.

Buterin, V. 2013. Ethereum: The Ultimate Smart Contract and Decentralized Application Platform. http://web.archive.org/web/20131228111141/ http://vbuterin.com/ethereum.html. accessed on 2019-02-28.

Calvaresi, D.; Mualla, Y.; Najjar, A.; Galland, S.; and Schumacher, M. 2019. Explainable Multi-Agent Systems Through Blockchain Technology. In *Explainable, Transparent Autonomous Agents and Multi-Agent Systems, EXTRAAMAS 2019*, volume 11763 of *Lecture Notes in Computer Science*. Springer International Publishing.

Dellermann, D.; Calma, A.; Lipusch, N.; Weber, T.; Weigel, S.; and Ebel, P. 2019. The future of human-ai collaboration: A taxonomy of design knowledge for hybrid intelligence systems. In *Hawaii International Conference on System Sciences (HICSS)*. URL: http://hdl.handle.net/10125/59468 accessed 2019-11-11.

Dignum, V. 2017. Responsible autonomy. In *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, 4698–4704.

Dignum, V. 2018. Ethics in artificial intelligence: introduction to the special issue. *Ethics and Information Technology* 20(1):1–3.

Dorffner, G. 1991. *Konnektionismus - Von neuronalen Netzwerken zu einer 'natürlichen' KI.* Springer.

Dworkin, M. J. 2015. SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions. Technical Report NIST FIPS 202, National Institute of Standards and Technology.

Falleri, J.-R.; Morandat, F.; Blanc, X.; Martinez, M.; and Monperrus, M. 2014. Fine-grained and Accurate Source Code Differencing. In *Proceedings of the 29th ACM/IEEE International Conference on Automated Software Engineering*, ASE '14, 313–324. New York, NY, USA: ACM.

Fill, H.-G., and Härer, F. 2018. Knowledge Blockchains: Applying Blockchain Technologies to Enterprise Modeling. In *Proceedings of the 51st Hawaii International Conference on System Sciences (HICSS-51)*, 4045–4054.

Fill, H.-G. 2017. SeMFIS: A Flexible Engineering Platform for Semantic Annotations of Conceptual Models. *Semantic Web (SWJ)* 8(5):747–763.

Fill, H. 2019. Applying the concept of knowledge blockchains to ontologies. In Martin, A.; Hinkelmann, K.; Gerber, A.; Lenat, D.; van Harmelen, F.; and Clark, P., eds., *AAAI 2019 Spring Symposium on Combining Machine Learning with Knowledge Engineering*. CEUR-WS.org.

Fluri, B.; Wuersch, M.; PInzger, M.; and Gall, H. 2007. Change Distilling:Tree Differencing for Fine-Grained Source Code Change Extraction. *IEEE Transactions on Software Engineering* 33(11):725–743.

Ghodsi, Z.; Gu, T.; and Garg, S. 2017. SafetyNets: Verifiable Execution of Deep Neural Networks on an Untrusted Cloud. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, 4675–4684. USA: Curran Associates Inc.

Hardjono, T., and Smith, N. M. 2019. Decentralized Trusted Computing Base for Blockchain Infrastructure Security. *CoRR* abs/1905.04412.

Härer, F., and Fill, H.-G. 2019. Decentralized Attestation of Conceptual Models Using the Ethereum Blockchain. In *21st IEEE International Conference on Business Informatics (CBI 2019)*.

Harmelen, F. v., and Teije, A. t. 2019. A boxology of design patterns forhybrid learningand reasoning systems. *Journal of Web Engineering* 18(1):97–124.

Harris, J. D., and Waggoner, B. 2019. Decentralized & Collaborative AI on Blockchain. In *2019 IEEE International Conference on Blockchain (Blockchain)*.

HI - The Hybrid Intelligence Centre. 2019. Hybrid Intelligence (HI): augmenting human intellect. http://www.hybrid-intelligence-centre.nl/wp-content/uploads/2019/10/HI-Application-public-version.pdf. Accessed: 2019-11-01.

Hochreiter, S., and Schmidhuber, J. 1997. Long Short-Term Memory. *Neural Computation* 9(8):1735–1780.

Holzinger, A. 2016. Interactive machine learning for health informatics: when do we need the human-in-the-loop? *Brain Informatics* 3(2):119–131.

Horridge, M.; Gonçalves, R. S.; Nyulas, C. I.; Tudorache, T.; and Musen, M. A. 2019. WebProtégé: A Cloud-Based Ontology Editor. In *Companion Proceedings of The 2019 World Wide Web Conference on - WWW '19*, 686–689. San Francisco, USA: ACM Press.

Hund, M.; Böhm, D.; Sturm, W.; Sedlmair, M.; Schreck, T.; Ullrich, T.; Keim, D. A.; Majnaric, L.; and Holzinger, A. 2016. Visual analytics for concept exploration in subspaces of patient groups - making sense of complex datasets with the doctor-in-the-loop. *Brain Informatics* 3(4):233–247.

Härer, F. 2019. *Integrierte Entwicklung und Ausführung von Prozessen in dezentralen Organisationen. Ein Vorschlag auf Basis der Blockchain*. University of Bamberg Press. Dissertation. doi 10.20378/irbo-55721.

Kamar, E. 2016a. Directions in hybrid intelligence: Complementing AI systems with human intelligence. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI 2016, New York, NY, USA, 9-15 July 2016*, 4070–4073.

Kamar, E. 2016b. Hybrid workplaces of the future. *XRDS: Crossroads, The ACM Magazine for Students* 23(2):22–25.

Karagiannis, D., and Buchmann, R. A. 2016. Linked open models: Extending linked open data with conceptual model information. *Inf. Syst.* 56:174–197.

Kim, S.; Kwon, Y.; and Cho, S. 2018. A Survey of Scalability Solutions on Blockchain. In *2018 International Conference on Information and Communication Technology Convergence (ICTC)*, 1204–1207.

Koegel, M.; Herrmannsdoerfer, M.; Helming, J.; and Li, Y. 2009. State-based vs. Operation-based Change Tracking. In *MODELS '09 MoDSE-MCCM Workshop*, 10.

Laney, D. 2001. 3D data management: Controlling data volume, velocity, and variety. Technical report, META Group.

Leach, P. J.; Mealling, M.; and Salz, R. 2005. A Universally Unique IDentifier (UUID) URN Namespace. https://tools.ietf.org/html/rfc4122.

Lippe, E., and van Oosterom, N. 1992. Operation-based Merging. In *Proceedings of the Fifth ACM SIGSOFT Symposium on Software Development Environments*, SDE 5, 78–87. New York, NY, USA: ACM.

Lomov, B. F., and Venda, V. F. 1977. Human factors: Problems of adapting systems for the interaction of information to the individual: The theory of hybrid intelligence. *Proceedings of the Human Factors Society Annual Meeting* 21(1):1–9.

Lundkvist, C.; Heck, R.; Torstensson, J.; Mitton, Z.; and Sena, M. 2016. uport a platform for self-sovereign identity. http://blockchainlab.com/pdf/uPort_whitepaper_DRAFT20161020.pdf. accessed on 2019-02-28.

Luo, Y.; Fan, J.; Deng, C.; Li, Y.; Zheng, Y.; and Ding, J. 2019. Accountable Data Sharing Scheme Based on Blockchain and SGX. In *2019 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC)*, 9–16.

Martin, A.; Hinkelmann, K.; Gerber, A.; Lenat, D.; van Harmelen, F.; and Clark, P., eds. 2019. *Proceedings of the AAAI 2019 Spring Symposium on Combining Machine Learning with Knowledge Engineering (AAAI-MAKE 2019) Stanford University, Palo Alto, California, USA, March 25-27, 2019.*, *Stanford University, Palo Alto,* *California, USA, March 25-27, 2019*, volume 2350 of *CEUR Workshop Proceedings*. CEUR-WS.org.

Matthias, A. 2004. The responsibility gap: Ascribing responsibility for the actions of learning automata. *Ethics and Information Technology* 6(3):175–183.

McGuinness, D. L. 2004. Question answering on the semantic web. *IEEE Intelligent Systems* 19(1):82–85.

Minsky, M. L. 1991. Logical versus analogical or symbolic versus connectionist or neat versus scruffy. *AI Magazine* 12(2):34.

Morais, E.; Koens, T.; van Wijk, C.; and Koren, A. 2019. A survey on zero knowledge range proofs and applications. *SN Applied Sciences* 1(8):946.

Nassar, M.; Salah, K.; ur Rehman, M. H.; and Svetinovic, D. 2020. Blockchain for explainable and trustworthy artificial intelligence. *WIREs Data Mining and Knowledge Discovery* 10(1):e1340.

Obermeyer, Z.; Powers, B.; Vogeli, C.; and Mullainathan, S. 2019. Dissecting racial bias in an algorithm used to manage the health of populations. *Science* 366(6464):447–453.

Pearl, J. 2018. Theoretical Impediments to Machine Learning With Seven Sparks from the Causal Revolution. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining - WSDM '18*, 3–3. Marina Del Rey, CA, USA: ACM Press.

Russell, S., and Norvig, P. 2012. *Artificial Intelligence: A Modern Approach (German Edition)*. Pearson.

Shehu, A.-s.; Pinto, A.; and Correia, M. E. 2019. On the interoperability of european national identity cards. In Novais, P.; Jung, J. J.; Villarrubia González, G.; Fernández-Caballero, A.; Navarro, E.; González, P.; Carneiro, D.; Pinto, A.; Campbell, A. T.; and Durães, D., eds., *Ambient Intelligence – Software and Applications –, 9th International Symposium on Ambient Intelligence*, 338–348. Cham: Springer International Publishing.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Neural Information Processing Systems (NIPS)*, 5998–6008. Curran Associates, Inc.

Wood, G. 2014. Ethereum: A Secure Decentralised Generalised Transaction Ledger. https://gavwood.com/paper.pdf. accessed on 2019-02-28.