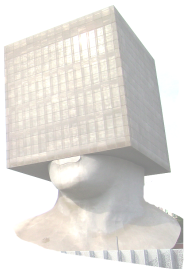

CARe-MAS'17

Nice, France, 31 October 2017

Matteo Baldoni, Cristina Baroglio,
Roberto Micalizio (eds.)

Computational Accountability and Responsibility in Multiagent Systems

First Workshop, CARe-MAS 2017
co-located with 20th International Conference on Principles and Practice of Multi-Agent Systems (PRIMA 2017)
Nice, France, October 31st, 2017
Workshop Notes

CARe-MAS 2017 Home Page:
`http://www.di.unito.it/~baldoni/CAREMAS17/`

Preface

Individual and organizational actions have social consequences that call for the implementation of recommendations of good conduct at multiple levels of granularity. For firms, business ethics is the essence of a healthy society (OECD reports) and, indeed, a growing number of companies issue voluntary codes of conduct to commit to values like legal compliance, accountability, privacy, and trust. Government agencies, on the other side, identify transparency as a key value that promotes accountability, public participation, collaboration, and effectiveness. United Nations continuously stress "the need for a robust, yet flexible, accountability framework to assess progress and achieve results, as well as ensure that all actors honour their commitments" (General Assembly President John Ashe, 2014).

The CARe-MAS Workshop aims at providing a discussion forum for researchers and practitioners who are investigating issues related to computational accountability and responsibility in multi-agent systems. In other words, CARe-MAS concerns the use of Artificial Intelligence techniques and approaches – with particular care to multiagent systems– to help supporting the realization of accountability frameworks which, in turn, help organizations to respect their commitments, help individuals in organizing their work, help managers in taking decisions, improve infrastructure and procedures, and so forth. The development of such specialized management systems introduces challenges and requirements for handling ethical issues, e.g. serving the requirements of transparency, accountability, and privacy preservation. We believe that such challenges can be faced with the support of intelligent systems, with plenty of potential applications in fields like finance and business transactions, fair business practices, resource management, consumer protection, economic systems, corruption, sales and marketing, health care, public administration, smart cities, and decision support.

The CARe-MAS Workshop is the conclusive event of the two-year project AThOS[1] (Accountable Trustworthy Organizations and Systems), and is held as a full day workshop co-located with the PRIMA conference in Nice, France on October 31^{st}, 2017. The workshop receive 6 submissions, each of which had at least three reviews. The program also includes two invited talks. The first "Accountability by Dialogue: a new approach to data protection" by dr. Joris Hulstijn from Tilburg University (The Netherlands). The second "How to Approach the Problem of Coming to Responsible Intelligent Systems?" by Jan Broersen from Utrecht Univeristy (The Netherlands).

We are confident that the talks offer an interesting perspective of the work that has been done about accountability in from a computational perspective, and they will also offer the opportunity for fruitful and interesting discussions.

[1] http://di.unito.it/athos

We would like to thank the members of the Program Committee for their excellent work during the reviewing phase. We also acknowledge the EasyChair conference management system that –as usual– provided its reliable and useful support of the workshop organization process. Moreover, we would like to thank the members of the Steering Committee of PRIMA for their valuable suggestions and support.

January 16th, 2018

Matteo Baldoni
Cristina Baroglio
Roberto Micalizio

Workshop Organisers

Matteo Baldoni	University of Torino, Italy
Cristina Baroglio	University of Torino, Italy
Roberto Micalizio	University of Torino, Italy

Programme Committee

Marco Alberti	DMI - University of Ferrara
Olivier Boissier	ENS Mines Saint-Etienne
Jan Broersen	Utrecht University
Barbara Carminati	University of Insubria
Cristiano Castelfranchi	Institute of Cognitive Sciences and Technologies
Amit Chopra	University of Lancaster
Mehdi Dastani	Utrecht University
Virginia Dignum	TU Delft
Joris Hulstijn	Tilburg University
Nadin Kokciyan	Bogazici University
Emiliano Lorini	IRIT
Felipe Meneguzzi	Pontifical Catholic University of Rio Grande do Sul
Antonio Carlos Rocha Costa	Universidade Federal do Rio Grande do Sul - UFRGS
Jean-Claude Royer	Ecole des Mines de Nantes
Marija Slavkovik	University of Bergen
M. Birna van Riemsdijk	TU Delft
Pinar Yolum	Bogazici University

Acknowledgements

The organizers were partially supported by the *Accountable Trustworthy Organizations and Systems (AThOS)* project, funded by Università degli Studi di Torino and Compagnia di San Paolo (CSP 2014).

Table of Contents

Invited Talks

Accountability by Dialogue: A New Approach to Data Protection 1
Joris Hulstijn

How to Approach the Problem of Coming to Responsible Intelligent Systems? 2
Jan M. Broersen

Contributed Papers

The AThOS Project: First Steps towards Computational Accountability . 3
Matteo Baldoni, Cristina Baroglio, Roberto Micalizio

Instrumenting Accountability in MAS with Blockchain 20
Fernando G. Papi, Jomi F. Hübner, Maiquel de Brito

Towards the Specification of Natural Language Accountability Policies with AccLab: The Laptop Policy Use Case 35
Walid Benghabrit, Jean-Claude Royer, Anderson Santana De Oliveira

Requirements for a Temporal Logic of Daily Activities for Supportive Technology 43
Malte S. Kliess, M. Birna van Riemsdijk

Open Data for Accountability in the Fight Against Corruption 52
Joris Hulstijn, Darusalam Darusalam, Marijn Janssen

Classifying the Autonomy and Morality of Artificial Agents 67
Sjur Dyrkolbotn, Truls Pedersen, Marija Slavkovik

Author Index 84

Accountability by Dialogue: A New Approach to Data Protection

Joris Hulstijn

Tilburg School of Economics and Management
Department of Management

Abstract. Current legal frameworks for data protection have a number of flaws. The notion of informed consent does not work in practice. Legislation only covers personal data, but it doesnt cover data about groups and it doesn't cover conditions on usage of data for certain purposes. Supervision is largely based on self-regulation. Regulatory agencies have little capacity. On the other hand, we observe that many business models are based on data about users. Users pay with their data. Access to data should therefore be seen as a counteroffer in a contract. In this paper we will therefore suggest a different approach to data protection, based on the idea of accountability. In this short paper, we propose a dialogue framework to facilitate such accountability, in the application domain of data protection. The idea is to empower users to negotiate better terms and conditions in their contracts, monitor compliance, and challenge the organization in case of breaches of contract. That means that in addition to the current legal framework for data protection, which is generally based on public law, we suggest to make more use of private law as the legal framework of preference. To enable accountability over data protection, we foresee two kinds of functionality. Tools that may help users negotiate sensible contracts that take data protection aspects into account. An infrastructure that monitors actual usage of data, detects possible breaches of contract and allows users to challenge the organization. In addition, we discuss the necessary elements of a governance structure to enable effective enforcement.

How to Approach the Problem of Coming to Responsible Intelligent Systems?

Jan M. Broersen

Department of Information and Computing Sciences
Universiteit Utrecht

Abstract. I will give an overview of how over the last three year my thoughts about how to approach the problem of coming to responsible intelligent systems have changed. I will sketch sub-problems and will attempt to match them to relevant theories in computer science, AI and (moral) philosophy.

The AThOS Project: First Steps towards Computational Accountability

Matteo Baldoni, Cristina Baroglio, and Roberto Micalizio

Università degli Studi di Torino — Dipartimento di Informatica
firstname.lastname@unito.it

Abstract. This work studies the support research on multiagent systems can give to the automation of computation of accountability (computational accountability), an din particular to accountability in organizational contexts. It introduces the idea of guaranteeing accountability as a design property of socio-technical systems, and explain a set of principles which a socio-technical system should respect. We explain the ADOPT protocol, which is a way that allows to realize computational accountability in an organizational setting, by means of multiagent systems technology.

Keywords: Computational Accountability, Business Artifacts, Normative MAS, Social Commitments

1 The AThOS Project: Motivations and Introduction

Thanks to the recent technological advances, that make devices such as smart phones and tablets indispensable tools in everyday life, the paradigm of Sociotechnical Systems (STSs) is rapidly spreading. STSs are complex systems where human beings interact with each other by means of ICT infrastructures. Humans may interact as members of the same organization, or enterprise, to achieve an organizational (business) goal. More often than not, however, interacting actors can belong to different organizations, and interaction is a means for reaching their goals.

In STSs, humans are the central elements of the system, they are essential to deliver the business goals. This central role of people makes STSs nondeterministic systems, because human behavior is generally unpredictable[1]. However, within organizations (and STSs) humans have relationships, duties, and commitments, whose satisfaction is crucial to the delivery of the business goals and, thus, to the well-being of the organization itself. Consistently, it is important, both to the employee and to the management, to be aware when duties (or commitments) are neglected, to understand the reasons, and decide if and how to act. The problem is that the mere availability of an ICT infrastructure does not support the interacting principals in being aware of the expectations that are put on

[1] In the sense that the same person may react in different ways to similar situations, depending on conditions that more often than not are not modeled in the system.

them, as it does not support who is in charge to get a clear picture of how tasks are, or are not, being carried on. This aspect becomes aggravated when the STS involves principals who belong to different organizations and *cross-organizational* relationships have to be established and maintained.

It is precisely because of the principals' autonomy that *accountability becomes a critical feature for an enterprise*. When something deviates from the expected and desired behavior, an accountable enterprise is committed to understand what went wrong and why, and to find a proper course of repairing action. What one expects from STSs is that such a process be, at least partially, automated. In other words, STSs should provide the ground for a form of *computational accountability* [4], where accountability is a property that can be checked at runtime, or even enforced at design time.

There are, indeed, attempts in business software applications to model the distribution of tasks and the related responsibility. For instance, the BPMN language for modeling workflows supplies the lane construct to distribute tasks among different actors. The idea is that an actor is responsible for performing the activities that lay in its lane, but a lot is left to intuition. Specifically, responsibility is not modeled explicitly, it is grasped from the workflow structure and may concern an office, not necessarily individuals. Then, being responsible for carrying out some activity does not mean being accountable for it. A more explicit instrument is the RACI matrix [29] by which each activity is associated with at least one responsible actor, exactly one accountable actor, and possibly some consulted and informed actors. The limit is that a RACI matrix is substantially an off-line resource that does not support actively an actor during her activities. It cannot be used as a monitoring tool on-line, but just as piece of data that a forum consults *a posteriori* to determine who was in charge of what. In [14] RACI matrices are automatically derived from BPMN specifications, but they are just used to support the resource assignment to each activity.

This paper reports the results of the two-year piloting phase of the *AThOS*[2] project. The ambitious, long-term objective of the AThOS project is the design of a framework for computational accountability in the context of cross-organizational business processes. From a methodological point of view, AThOS aims at providing a modeling framework for STSs, where the focus is shifted from a procedural representation of workflows to a *declarative* representation that is centered around the key notions of *interaction* and *social relationships* among actors. From a practical perspective, AThOS aims at providing a new enterprise software engine that monitors on-going interactions (and processes), and supports accountability. To achieve these results, the framework we are developing takes advantage of previous declarative approaches to the modeling of business processes, and integrates them with an explicit, first-class representation of the notion of "relationship", that relies on *social commitments* [27]. On the practical side, relying on previous works in literature (among which [28,13]), we are investigating the adoption of the paradigm of Multi-Agent Oriented Program-

[2] The *Accountable Trustworthy Organizations and Systems (AThOS)* project, funded by Università degli Studi di Torino and Compagnia di San Paolo (CSP 2014).

ming for the realization of business processes. This would allow a programmer to fully exploit the power of declarative agent programming frameworks (such as JaCaMo [10]).

The paper begins by explaining the kind of accountability the research focuses on (Section 2). Then, it introduces the notion of accountability by design, studying its feasibility in computational settings (Section 3). It continues by explaining the ADOPT protocol, which is a way to achieve computational accountability in multiagent systems (Section 4), and a first direction of implementation (Section 5). A discussion ends the paper (Section 6).

2 Organizational accountability and why it is important

Accountability is a wide-ranging concept, that can take on many different characteristics and meanings, depending on the discipline in which discussion takes place. So, the first point that it is relevant to underline is that our focus is posed on *accountability in organizational settings*. We do not care whether the organization is persistent, meaning that it has long-lasting goals that will produce many interactions (like a selling company), or if it is specifically created to achieve a once-in-a-time goal (like building a house), and will dissolve afterwards. We assume, however, the organization's functioning to be supported by an STS. In such a context, we aim at realizing *accountability as computational process*; more specifically, a backward-looking, institutional process that permits one entity to be held to account by another.

The accountability process activates when a state of interest is reached, and an *authoritative entity* wishes to hold to account those behind said state. Following [11], the process encompasses three primary phases: 1) an investigative entity (forum) receives all information regarding the agents' actions, effects, permissions, obligations, etc. that led to the situation under scrutiny, 2) the forum contextualizes actions to understand their adequacy and legitimacy, and finally 3) the forum passes judgment on agents with sanctions or rewards. Our goal consists in automating the entire process for use in a MAS, although we will presently leave out the sanctioning piece of the third phase, which is already studied in the literature on electronic institutions [16]. To realize our goal, we begin by looking at the reasoning process behind accountability to identify points of adaptation into the software world. Following [12], we believe that three conditions should all realize for an STS to support an accountability system: *agency*, *causal relevancy*, and *avoidance opportunity*. Such conditions enable accountability attribution. Agency means that the involved individuals are autonomous (indeed, in our setting they are autonomous by assumption), that they have reasoning capacity (idem), and that they can distinguish right and wrong. The third condition is the one that, in our understanding, calls for support by an STS. In order to distinguish right from wrong, individuals must be *aware* of the binds they have with the others and with the organization, as agent's "ethics" will be expressed through adherence to norms and commitments contextualized within an organization. Causal relevancy expresses the necessity of causation

linking a given agent to a situation under scrutiny. The avoidance opportunity condition specifies that an "agent should have had a reasonable opportunity to have done otherwise" [12]. Once again, awareness of the conditions an individual will be held to account for is crucial, since such knowledge has an impact on the choices of individuals as well as on its liability.

The realization of organizational accountability brings along practical advantages which motivate the purpose of studying accountablity as a system property. Evidence is provided by many studies and documents which promote the adoption of accountability frameworks. Among them, reports by the United Nations [33] and organizations like the OECD [26]. We briefly summarize the main reasons for realizing accountability frameworks and, thus, to study computational accountability. First of all, accountability within an organization is more than reporting information, disclosing reasons, or sanctioning the culprits. It is generally considered as a powerful feedback mechanism aimed at improving performance. An accountability framework, like the one of UNICEF [31] or WHO [32], explicitly establishes desired results, monitors and measures performance, uses the feedback obtained by the accountable parties to decide which changes should be made (which actions should be taken) to achieve the intended result. Such a process enables the evolution of an organization in a way that is functional to the fulfillment of its objectives. It is helpful both in those situations where the objectives, to be achieved, need to rely on external actors after the stipulation of a covenant, and in those cases where objectives depend entirely on processes that are internal to the organization itself. Another positive effect of organizational accountability is that it helps turning implicit expectations into explicit expectations. This is important because expectations that are not communicated create confusion and tensions inside an organization. Parties may be unaware of what they should deliver. Instead, a clear awareness of the expectations (and of the accountabilities) increases reliability, improves performance and trust, and helps to achieve common goals.

3 Accountability by Design

The research question that raises at this point is whether it is possible to create STSs that show *accountability as a design property*, that is, such that in every state the accountable parties are clearly identifiable. Providing accountability by design requires decision of how to tackle certain situations, long discussed in philosophy as ethical dilemmas. One of these concerns is causal determinism, that in our context we can rephrase as whether or not an agent should be considered accountable for an adverse state if this state is inevitable, whatever action the agent will perform. We adopt the incompatibilist position to moral responsibility and conclude in a software setting that an agent cannot be held accountable in causal determinism but that the discovery of inevitable states lies with the agents. If an agent stipulates provisions for an inevitable adverse state, that same adversity would be represented to some degree in its provisions due to its inevitable nature. The agent would still be held accountable for the

state, because it effectively declares responsibility for a goal through its accepted stipulated conditions. Likewise if an agent offers to realize a goal in the presence of a certain provisions, it is declaring for all intents and purposes that the goal is possible under certain conditions. Should that agent offer up an incorrect assessment and the goal be effectively impossible even with stipulations, the agent will nevertheless be held accountable, because the organization "believes" an agent's declaration. We therefore conclude, thanks to the distributed nature of our vision of accountability, that an organization can consider absent both impossibilities and inevitabilities. Another ethical dilemma concerns knowledge of the consequences of one's actions. That is, can one be held accountable for an unforeseeable outcome?

As an example, consider a setting where a house is being built. The companies and the workers who contribute to the construction constitute an organization. In particular, the organization involves two members: one who should prepare a wall, *wall-preparer*, and another, *painter*, who should paint the wall white at some agreed price. The *wall-preparer* fulfills her/his task by spackling the wall but instead of using a standard white spackle some dark colored one (a remainder of a previous work) is used. Due to this unexpected action, *painter* has not the correct amount of materials and cannot fulfill the painting task at the agreed price. Though not coerced, *painter* cannot fulfill what expected of him/her, due to the choices of another. Clearly, despite the fact that *wall-preparer* did what assigned, the kind of spackle that was used hampers the *painter*'s work. In this case it is difficult to identify an accountable party: the *painter* might have provided adequate provisions instead of giving them as granted, the *wall-preparer* might have asked if an unusual choice would have had consequences before doing the work, the organization might have checked that the two members properly coordinated. Their selfishness prevents their understanding the assigned tasks as part of a greater work, and the lack of an explicit account of the relationships between the agents (and their tasks) contributes to the picture but. On the other hand, if expectations are not clear and shared, how could it be otherwise? For instance, *wall-preparer* does not know the circumstances under which the *painter* can achieve its goal, should it be held accountable for its decision of using a colored spackle? To foster the good functioning of the organization, then, there is the need of adequate support.

3.1 Characterization of Organizational Accountability

Accountability as a design property means that an STS is built in such a way that accountability can be determined from any future institutional state. Indeed, our goal lies in automating the entire process, that is, to create a structure that creates and collects contextualized information so that accountability can actually be determined from any future institutional state. We consider integral to this process the following steps: a forum must receive all information, including all causal actions, regarding a given situation under scrutiny, the forum must be able to contextualize actions to understand their adequacy and legitimacy, and finally the forum must be able to pass judgment on agents. One of the key

difficulties in realizing our goal lies with the notion of *contextualized action*. In our own societies, contextualizing might entail an examination of circumstances: for example, what should have a person done, why didn't she/he do that, what impact did her/his actions have, and given what the person had to work with, did she/he act in an exemplary fashion? The same process in a MAS would be guided by the same type of questions, though in order to facilitate the answers, we need to make use of different structures. In particular, we need structures that allow assessing who is accountable without actually infringing on the individual and private nature of agents.

We identify the following necessary-but-not-sufficient principles a MAS must exhibit in order to support the determination of organizational accountability.

Principle 1 All collaborations and communications subject to considerations of accountability among the agents occur within a single scope that we call *organization*.

Principle 2 An agent can enroll in an organization only by playing a *role* that is defined inside the organization.

Principle 3 An agent willing to play a role in an organization must be aware of all the powers associated with such a role before adopting it.

Principle 4 An agent is only accountable, towards the organization or another agent, for those goals it has explicitly accepted to bring about.

Principle 5 An agent must have the leeway for putting before the organization the provisions it needs for achieving the goal to which it is committing. The organization has the capability of reasoning about the requested provisions and can accept or reject them.

Principle 1 calls for situatedness. Accountability must operate in a specific context because individual actions take on their significance only in the presence of the larger whole. What constitutes a highly objectionable action in one context could instead be worthy of praise in another. Correspondingly, a forum can only operate in context and an agent's actions must always be contextualized. The same role in different contexts can have radically diverse impacts on the organization and consequently on accountability attribution. When determining attribution, thus, an organization will only take into account interactions that took place inside its boundaries. Placing an organizational limit on accountability determination serves multiple purposes. It isolates events and actors so that when searching for causes/effects, one need not consider all actions from the beginning of time nor actions from other organizations. Agents are reassured that only for actions within an organization will they potentially be held accountable. Actions, thanks to agent roles (Principle 2), also always happen in context.

To adequately tackle accountability by categorizing action, we must deal with two properties within a given organization: 1) an agent properly completes its tasks and 2) an agent does not interfere with the tasks of others. The principles 2–5 deal more explicitly with the first property, i.e., how to ensure that agents complete their tasks in a manner fair for both the agents and the organization. The second property is also partially satisfied by ensuring that, in the presence of goal dependencies, the first agent in sequence not to complete its goal will bear

accountability, not only for its incomplete goal, but for all dependent goals that will consequently remain incomplete. That is, should an agent be responsible for a goal on whose completion other agents wait, and should that agent not complete its goal, then it will be accountable for its incomplete goal and for that goal's dependents as well.

As an organizational and contextual aid to accountability, roles attribute social significance to an agent's actions and can, therefore, provide a guide to the severity of non-adherence. Following the tradition initiated by Hohfeld [20], a power is "one's affirmative 'control' over a given legal relation as against another." The relationship between powers and roles has long been studied in fields like social theory, artificial intelligence, and law. By Principle 3 we stipulate that an agent can only be accountable for exercising the powers that are publicly given to it by the roles it plays. Such powers are, indeed, the means through which agents affect their organizational setting. An agent cannot be held accountable for unknown effects of its actions but, rather, only for consequences related to an agent's known place in sequences of goals. On the other hand, an agent cannot be held accountable for an unknown goal that the organization attaches to its role, and this leads us to Principle 4. An organization may not obligate agents to complete goals without prior agreement. In other words, an organization must always communicate to each agent the goals it would like the agent to pursue. Notice that with this principle we diverge from considerations in the field of ethics regarding accountability in the presence of causal determinism [19,12], where even in the absence of alternate possibilities humans can be morally responsible thanks to the significance of the choice to act. Finding the conversation fundamentally shifts when speaking of software agents, we consequently conclude that accountability is not attributable in the presence of impossibilities. Correspondingly, agents must be able to stipulate the *conditions* under which a given goal's achievement becomes possible, i.e. the agent's requested *provisions*. The burden of discovery for impossibilities, therefore, rests upon an agent collective who announce them by their combined silence for a given goal. That is, a goal becomes effectively impossible for a group of agents should no agent stipulate a method of achievement. Conversely, an agent also declares a goal possible the moment it provides provisions to that goal. Should an uniformed agent stipulate insufficient provisions for an impossible goal that is then accepted by an organization, that agent will be held accountable because by voicing its provisions, it declared an impossible goal possible. The opportunity to specify provisions, therefore, is fundamental in differentiating between impossibilities and possibilities.

Going back to the painter example, before agreeing to be the organization's painter, *painter* would stipulate provisions for its role goals, in this case, *white wall*. An organization, accepting *painter*'s provisions, would then add *white wall* as a goal condition to the job description of *wall-preparer*. *Wall-preparer* would in turn accept its new goal condition. Come execution time, when *wall-preparer* adds the black stripe, not only will *painter*'s provisions not be met, but *wall-preparer* will also have violated its own goal conditions. Since *wall-preparer*

knows what it did was wrong thanks to goal conditions, causally contributed to an adverse situation of work failure, and could have avoided causally contributing, it will be held accountable for the adverse state.

4 Reaching Accountability via the ADOPT Protocol

We now introduce our vision to achieve computational accountability via a proper design of the interactions between software components. The first step is a paradigm shift: from the *process-oriented* view to the *agent-oriented* view. The process-oriented view is substantially activity-centric – that is, procedural. It focuses on the (business) process that is carried on by a software component, but overlooks the interaction between components, at the core of the realization of STSs. Instead, an STS can be conveniently thought of as a Multi-agent System (MAS) where software agents interact with each other on behalf of human principals. An advantage of the agent perspective is that it enables the use of some design (and programming) abstractions.These are, for instance, the *environment* where the agents operate, the *norms* governing *organizations* and *institutions*, the *roles*, and their *powers*, possibly defined therein, and an explicit representation of interactions, for instance, via social *commitments*.

The agent abstraction allows us to think of software modules as goal-oriented components. That is, the focus moves from the activities within a process to the *goal* the process aims at reaching. The design process can therefore be modularized, because the interactions among agents (i.e., components) can be kept separated from the inner process of each agent. In fact, interactions can be specified as a consequence of the agents' goals. That is, the agents create and maintain interactions as a means for reaching their goals. The internal process of the agent lies at a different design level with respect to interaction. For the accountability purpose, only the interaction level is relevant since it makes explicit the engagements that agents have agreed upon. Monitoring the progress of these engagements is one the main objectives of an accountable system.

Our intuition is that an STS can be seen as an *organization* in which agents (i.e., the software components of the STS at hand) can join and operate. More precisely, we follow the ontological definition of organizations given in [9]. Organizations are characterized by *roles*, which are first-class modeling entities. Roles are actually social roles, and are related to their players (i.e., the agents), and their organization by three properties: (1) a role must always be associated with the institution it belongs to and with its player; (2) a role is meaningful only within the institution it belongs to, and hence the definition of a role is only given within an institution; (3) the actions defined for a role in an institution have access to the state of the institution and of other roles; these actions are, therefore, the powers a role player is endowed with. In order to obtain an accountable organization, we require that the interactions between the organization and its members follow a given *accountability protocol* (ADOPT[3] [3,5]) that,

[3] Accountability-Driven Organization Programming Technique

by realizing the five principles introduced above, guarantees accountability as a design property of the system as a whole. In essence, the accountability protocol makes the legal relationships between each agent and its organization explicit. Relationships are expressed as a set of (abstract) commitments (i.e., contracts), directed from the agents towards the organization itself, and vice versa. ADOPT consists of two phases: the *enactment* phase, and the *goal-assignment* phase.

Enactment Phase. In this phase, an agent enrolls in an organization by playing a role. During this phase, the agent is made aware of the powers the organization will grant to it as a player of a specific role, and the agent will commit towards the organization to use these powers when time will demand it. Let Ag_i be the agent willing to enroll in organization Org:

(1) Ag_i : commit to be a player for role R_i with powers $pwr_{i,1} \dots pwr_{i,n}$ if Org accepts
(2) Org : accept Ag_i as player of role R_i
(3) Ag_i : commit to use the powers $pwr_{i,1} \dots pwr_{i_n}$ when Org demands to

In step (1) agent Ag_i asks Org to join the organization becoming a player of role R_i, with powers $pwr_{i,1} \dots pwr_{i,n}$. Note that a power calls for a capability of the agent to exercise it. For instance, the *wall-preparer* is given the power to prepare the walls of the house being built, but to prepare the wall it needs materials, skill, and to take approriate action. Thus, the powers associated with role R_i will pose some capability requirements on the agent side. An agent willing to play role R_i is made aware of the capabilities it has to possess for being a proper actor of that role. The powers will be the means through which the agent will operate within the organization by changing the state of the organization itself. Capabilities, instead, will be sets of behaviors, owned by the agent, that allow it to exercise the powers that go with some role the agent plays. In the enactment phase the agent takes on the responsibility of this (previously implicit) declaration. This allows overcoming the impossibility to inspect the agents' code (which is not disclosed) that characterizes, in particular, cross-organizational settings as well as many STSs: the organization cannot verify whether the agent's implementation is compliant with the capability requirements, but if, during the execution, the agent will be unable to exercise one of its powers, *it will be deemed accountable* thanks to the commitments established during this phase.

The internal decision process, by which Org decides whether to accept the agent, are outside the scope of ADOPT. Here we are only interested in the final decision of accepting the agent. Step (2) encodes such a case. Once Org has accepted Ag_i, this agent is obliged, by the engagement created at step (1), to commit to use R_i powers in case Org will ever need them. This is done with the commitments created at step (3). After these three steps, agent Ag_i is enrolled within the organization. Notably, these few steps are sufficient to satisfy the first three principles we have identified. In fact, the protocol imposes the existence of an organization within which all the relevant events must occur (Principle 1). The protocol allows an agent to be part of an organization only by playing a role within that organization (Principle 2), and finally, the protocol is such that an agent is aware of the powers it will possibly use within the organization as player of a specific role (Protocol 3).

Goal-assignment Phase. The second phase of ADOPT considers the assignment of goals to some member of the organization. According to Principle 5, an agent should be able to negotiate with the organization the provisions it needs in order to accomplish a given goal. This negotiation is not part of the ADOPT proposal, and any negotiation algorithm can be used. Here, we just assume that after the negotiation, the agent and the organization have agreed upon the goal $g_{i,k}$ the organization wants Ag_i to perform, and the provision $prov_{i,k}$ the agent demands to the organization as a prerequisite. Such an agreement is "sealed" by the agent and the organization in the ADOPT protocol as below outlined.

(4) Org : commit to assign goal $g_{i,k}$ with provisions $prov_{i,k}$ if Ag_i accepts
(5) Ag_i : commit to bring about $g_{i,k}$ if Org assigns it and brings about provisions $prov_{i,k}$
(6) Org : assign $g_{i,k}$ to Ag_i
(7) Org : bring about provision $\text{prov}_{i,k}$
(8) Ag_i : achieve goal$g_{i,k}$

In step (4), Org commits towards agent Ag_i that, if the agent will accept to accomplish the goal, it will assign the goal to the agent, and will supply Ag_i the agreed provisions. Thus, also Org takes on commitments towards its members. These commitments, in particular, assure that Org will never ask an agent to bring about a goal, for which either no agreement was reached, or provisions are missing. In case Org contravenes this expectation, it will be held accountable as any other player in the system. In step (5), Ag_i creates the commitment of bringing about goal $g_{i,k}$ provided that the provisions $prov_{i,k}$ have been supplied, and the goal has been assigned to it. In the last three steps, the two parties operate so as to satisfy their commitments. Org will bring about the provisions[4], and will assign goal $g_{i,k}$ to Ag_i. On its side, Ag_i will try achieve the assigned goal so as to satisfy its commitment towards Org.

ADOPT supports the realization of an accountable system through the mutual commitments that exist between Org and any agent Ag_i, which are made explicit. This feature realizes Principle 4, for which an agent is accountable only for those goals it has explicitly accepted to achieve. In fact, in our proposal, an organization cannot command one of its members to achieve a goal. Rather, the organization will propose a goal, and the agent will have to decide whether explicitly accepting that goal. Only in this way it becomes possible to assess accountability. It is possible to verify that the ADOPT protocol satisfies Principle 4 via model checking. In fact, differently from the other principles that are directly satisfied by the structure of the organization we propose, the verification of this principle demands consideration of the possible evolutions of the protocol. Indeed, one has to guarantee that, in any possible run of the protocol, the organization assigns a goal to an agent only when this goal is accepted. In [3] we discuss how the ADOPT protocol can be encoded as an interpreted system [18], and then, by means the MCMAS model checker [21], the validity of Principle 4 can be verified in terms of CTL formulae.

[4] Provisions will reasonably be supplied by other agents within the organization.

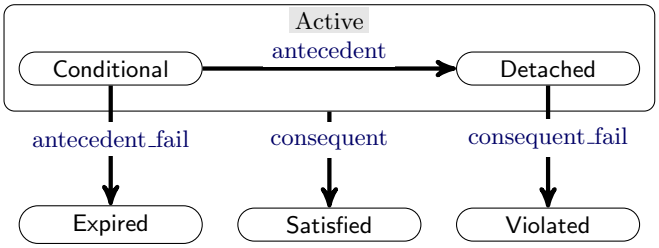

Fig. 1. Commitment life cycle.

5 Implementing ADOPT and Future Directions

One straightforward way for implementing ADOPT is to rely on the notion of *social commitment* [15,27]. A social commitment (sometimes called commitment, for simplicity, in the following) is a sort of contract between two parties: a *debtor* commits towards a *creditor* to bring about a consequent condition *con* should an antecedent condition *ant* occur. Social commitments, thus, find a direct mapping with the commitments that arise in the ADOPT protocol. Formally, commitments have the form $\mathsf{C}(debtor, creditor, ant, con)$, and their evolution follows the lifecycle reported in Figure 1: a commitment is *Violated* either when its antecedent is true but its consequent will forever be false, or when it is canceled when Detached. It is *Satisfied*, when the engagement is accomplished (notice that a commitment does not need to be detached before being satisfied). It is *Expired*, when it is no longer in effect and therefore the debtor would not fail to comply even if does not accomplish the consequent. Active has two substates: *Conditional* as long as the antecedent does not occur, and *Detached* when the antecedent has occurred. Commitments have a normative value because when the antecedent condition is satisfied, the debtor is obliged to bring about the consequent, lest the violation of the commitment. Differently from obligations in deontic logic, however, a commitment can only be created by its debtor; thus, its creation is a consequence of an internal process of the debtor, rather than the consequence of a normative system as it happens with obligations.

Commitments can be used to specify the semantics of message-based protocols, or they can be a means for specifying protocols on their own, see for instance the JaCaMo+ agent platform [1], an extension of JaCaMo [10], where commitment-based protocols are made available as a specification of protocol artifacts. Commitments are, therefore, a viable tool for the realization of accountable systems because, on the one side, the ADOPT protocol is easily mapped into a commitment-based interaction protocol and, on the other side, there are agent platforms that support commitments as a native programming element, providing a technological infrastructure for the implementation of accountable STSs. The adoption of commitments opens also novel development directions, that we will shortly outline in the rest of this section.

5.1 Accountability of Information Management with Normative Business Artifacts

In the quest for the computational accountability, we have previously advocated a shift from a procedural view to an agent-oriented view, which usually features on a declarative approach. Think, for instance to BDI agents and the way they are realized in many agent-based platforms (e.g. JaCaMo [10] and JaCaMo+ [1]). Indeed, the use of declarative formalisms for Business Process Management is not new. BALSA [7] is a first attempt to define processes in terms of declarative ECA-rules (i.e., Event-Condition-Action). The BALSA methodology relies on the important notion of *business artifact* [25]. A business artifact is a key conceptual business entity that is used in guiding the operation of a business. It is a concrete, identifiable, self-describing chunk of information. It is characterized by two specifications: (1) an *information model* that defines what relevant data are traced by the artifact, and (2) a *lifecycle model* that defines how such data can evolve as a consequence of business operations on the artifact itself. Importantly, both specifications are accessible to the processes that use the business artifact. The BALSA methodology is targeted to specify a data-centric declarative model of business operations. It can be summarized in three steps: 1) identify the relevant business artifacts of the problem at hand and their lifecycles, 2) develop a detailed specification of the services (or tasks) that will cause the evolution of the business artifact lifecycles, 3) define a number of ECA-rules that create associations between services and business artifacts. ECA-rules thus are the building blocks to define, in a declarative way, processes operating on data.

BALSA is extremely interesting, in particular because it introduces a novel perspective on the modeling of business processes. However, when it comes to the coordination of different business processes, the methodology refers to choreography languages and techniques proposed for service-oriented computing. Thence one of the major disadvantages of this approach and of its descendants, that is the lack of a clear separation of concerns between the *coordination logic* and the *business logic*. A business process will internalize the message exchange specified by the choreography at hand. Should the choreography be changed, it would be necessary to modify also the business processes accordingly. The happens because choreographies realize a form of subject coordination.

In [2] we introduce the notion of *normative business artifact*, as a way to obtain objective coordination. Inspired by the *activity theory* [17], we propose to use a business artifact as a coordination medium. To this end, the business artifact is enhanced with a normative layer in which the mutual expectations that each agent has on the others are made explicit, and hence can be used for coordinating the agents' activities. Apart from the advantage of having a form of objective coordination, normative business artifacts allow one to think about accountability of data and information management. That is, instead of considering just the state changes of a piece of information, it is also important to take into account other aspects of information management such as its correctness, its accessibility and spreading (inside and outside an organization), its usage during the decision making of agents, and so on. An interesting direction

of research is to investigate the use of social commitments for tackling these information management aspects as *accountability requirements* that, specified at design time, become a guide for the practical development of accountable STS.

5.2 Practical Reasoning and Accountability

The ADOPT protocol aims at supporting accountability at the organizational level, that is, ADOPT formalizes how an accountable relationship can be established between an organization and each of its members. Our long-term goal, however, is to realize an accountable system where the accountability property cuts across all aspects of an organization, and hence it holds not only between the organization and each member, but also between any two members of the organization. This is particularly important, and challenging, in an STS, where agents are autonomous in the achievement of their goals, and the interactions raising in this process may be problematic. The intuition, thus, is that every interaction that arises between any two agents in an organization should be accountable, in the sense that it should always be possible to identify an agent that is held to account towards another agent about its conduct. It is worth noting, however, that the accountability property must not be thought of as a means for limiting the autonomy of the agents. On the contrary, in our view, an accountable system supports the agents in their practical reasoning by making their (mutual) expectations explicit.

Commitments can play a fundamental role in realizing this vision since they formalize a tie between the internal goal of an agent with the relationships that the same agent may create in order to achieve that goal. In [30] this relation between goals and commitments is formalized by an operational semantics given as a set of patterns of practical reasoning rules. The Social-Continual Planning (SCP) technique introduced in [6] is a first attempt to integrate practical rules within a planning framework. SCP enables agents to plan their interactions as a means for achieving their goals. As future work, we intend to extend the SCP technique so as to guarantee that the planned interactions compose, altogether, an accountable system.

Another important aspect that we have to consider in the development of an accountable system, is that agents are autonomous and can violate their commitments on purpose. This eventuality must not be overlooked by a well-designed accountable framework; nonetheless, an accountable system is not, in our view, finalized to sanctioning the agents that violate their commitments (although sanctioning mechanisms could be contemplated). Our goal, in fact, is in providing agents with feedback about their accountability relationships. Under this respect, an accountable framework should include some form of diagnostic reasoning (see e.g., [24,23]), in order to detect deviations from the expected evolutions of an interaction, and provide the involved agents with possible explanations of what has occurred. These explanations (the feedback) are an important source of information that the agents can exploit in their practical reasoning for recovering from the violation of some commitments. A plan repair technique, driven by the results of a diagnostic inference in a multi-agent setting, is discussed in [22]. In

the next future, we aim at integrating such a plan repair technique within SCP, this would allow agents not only to plan their interactions, but also to properly react when these interactions do not progress as expected.

6 Discussion and Conclusions

In this paper we have summarized the main results about computational accountability we have collected in the scope of the AThOS project. AThOS was a two-year, local project completed in 2017, whose main objective was to shed some light on the design and development of accountable Sociotechnical Systems (STSs). Current methodologies for the development of STSs, are mainly focused on *procedural* aspects, with the purpose of gaining in efficiency of execution. Many such systems rely on languages (like BPMN, YAML, or UML Activity Diagrams) and related tools for modeling workflows. The procedural perspective is inadequate for the computational accountability purpose. First of all, it results in a too rigid model where the possible lines of activity are all delineated a priori. Each actor is forced to follow a predefined course of action, without the possibility to deviate for taking advantage of arising opportunities or for managing unexpected emergencies. More importantly, a procedural perspective, being activity-centric, overlooks the interaction dimension, which is central for the accountability reasoning. An accountability system, in fact, should in principle involve at least two actors: one which is held to account for its actions, and one (often referred to as the *forum*) which judges the conduct of the previous one. Both actors rely on the relationships they have created, and on the mutual expectations these relationships induce, to sustain their position (i.e., good vs. bad conduct). The AThOS project answers to these deficiencies by modeling STSs based on the typical abstractions provided by the agent-oriented paradigm (e.g., electronic institutions, organizations, norms, etc.). By leveraging on these abstraction, the main contribution of AThOS lies in the definition of *organizational accountability*: the STS becomes the organization within which all the relevant interactions among the actors (i.e., the agents) occur, and are explicitly traced. We have introduced five principles underpinning the accountability at the organizational level, and presented the ADOPT protocol as a means for satisfying these principles at the design of the STS. From a practical point of view, we have envisaged the use of social commitment for representing, and manipulating, the accountability relationships between the organization and its members.

Social commitments pave the way to further extensions that are sketched in the paper. They provide a promising solution for dealing with accountability of data and information management, overcoming some limits of the artifact-centric proposals [7,8], namely the use of choreographies for the synchronization of the activities of different processes. Since choreographies capture interaction only in terms of exchanged messages, it will not be possible, by looking at a choreography, to reason on the relationships between some actors, or to have expectations on their behavior. In AThOS, we have investigated an extension to business artifacts with a normative layer. Such a normative layer makes ex-

plicit the mutual expectations that processes (i.e., agents in our agent-oriented perspective) may have on each other while using a given business artifact. A first advantage of normative business artifacts is that the coordination among agents needs not a further modeling element such a choreography. Indeed, the business artifact itself becomes the medium of coordination, with a significant improvements from the software engineering point view, as we have observed. In addition, normative business artifact could be the key for the definition of accountability of data and information. The idea, is not only to model how the information can evolve over an interaction, but also how operations on data influence the way in which agents behave.

Another interesting line of research that stems from using social commitments is about acting in an accountable system. In [6] we have introduced a methodology–Social Continual Planning–integrating practical reasoning patterns relating goals and commitments [30] into a multi-agent planning framework. The technique is a good starting point for developing systems where the accountability property is guaranteed not only between an agent and the organization it belongs to, but also between any two agents that come to interact in the attempt to achieve their own goals. Thus, this is a promising line for reaching the desiderata of establishing accountable relationships across all levels of an organization.

References

1. M. Baldoni, C. Baroglio, F. Capuzzimati, and R. Micalizio. Commitment-based Agent Interaction in JaCaMo+. *Fundamenta Informaticae*, 157:1–33, 2018.
2. Matteo Baldoni, Cristina Baroglio, Federico Capuzzimati, and Roberto Micalizio. Objective Coordination with Business Artifacts and Social Engagements. In E. Teniente and M. Weidlich, editors, *Business Process Management Workshops, BPM 2017 International Workshops, Revised Papers*, volume 308 of *Lecture Notes in Business Information Processing (LNBIP)*, pages 1–18, Barcelona, Spain, 2018. Springer. This paper has been presented at the First Workshop on BP Innovations with Artificial Intelligence, BPAI 2017, organized by Koehler, J., Montali, M., Srivastava, B., Stuckenschmidt, H., De Masellis, R., Di Fracescomarino, C., Maggi, F. M., and Senderovich, A.
3. Matteo Baldoni, Cristina Baroglio, Katherine M. May, Roberto Micalizio, and Stefano Tedeschi. ADOPT JaCaMo: Accountability-Driven Organization Programming Technique for JaCaMo. In A. Bo, A. Bazzan, J. Leite, L. van der Torre, and S. Villata, editors, *PRIMA 2017: Principles and Practice of Multi-Agent Systems, 20th International Conference*, number 10621 in Lecture Notes in Computer Science, pages 295–312, Nice, France, October 30th–November 3rd 2017. Springer.
4. Matteo Baldoni, Cristina Baroglio, Katherine M. May, Roberto Micalizio, and Stefano Tedeschi. Computational accountability. In *Proceedings of the AI*IA Workshop on Deep Understanding and Reasoning: A Challenge for Next-generation Intelligent Agents 2016*, volume 1802 of *CEUR Workshop Proceedings*, pages 56–62. CEUR-WS.org, 2017.
5. Matteo Baldoni, Cristina Baroglio, Katherine M. May, Roberto Micalizio, and Stefano Tedeschi. Supporting Organizational Accountability inside Multiagent Systems. In R. Basili, F. Esposito, S. Ferilli, and F. A. Lisi, editors, *AI*IA 2017:*

Advances in Artificial Intelligence, XVI International Conference of the Italian Association for Artificial Intelligence, volume 10640 of *Lecture Notes in Computer Science*, pages 403–417, Bari, Italy, November 14th–17th 2017. Springer.

6. Matteo Baldoni, Cristina Baroglio, and Roberto Micalizio. Social continual planning in open multiagent systems: A first study. In *PRIMA 2015: Principles and Practice of Multi-Agent Systems - 18th International Conference, Bertinoro, Italy, October 26-30, 2015, Proceedings*, pages 575–584, 2015.
7. Kamal Bhattacharya, Nathan S. Caswell, Santhosh Kumaran, Anil Nigam, and Frederick Y. Wu. Artifact-centered operational modeling: Lessons from customer engagements. *IBM Systems Journal*, 46(4):703–721, 2007.
8. Kamal Bhattacharya, Richard Hull, and Jianwen Su. *A data-centric design methodology for business processes*, pages 503–531. Handbook of Research on Business Process Modeling. IGI Publishing, 2009.
9. Guido Boella and Leendert W. N. van der Torre. The Ontological Properties of Social Roles in Multi-Agent Systems: Definitional Dependence, Powers and Roles playing Roles. *Artificial Intelligence and Law Journal (AILaw)*, 15(3):201–221, 2007.
10. Olivier Boissier, Rafael H. Bordini, Jomi F. Hübner, Alessandro Ricci, and Andrea Santi. Multi-agent oriented programming with JaCaMo. *Sci. Comput. Program.*, 78(6):747–761, 2013.
11. Mark Bovens, Robert E. Goodin, and Thomas Schillemans, editors. *The Oxford Handbook of Public Accountability*. Oxford University Press, 2014.
12. Matthew Braham and Martin van Hees. An anatomy of moral responsibility. *Mind*, 121(483), 2012.
13. P. Bresciani and P. Donzelli. A Practical Agent-Based Approach to Requirements Engineering for Socio-technical Systems. In *Agent-Oriented Information System*, volume 3030 of *LNCS*. Springer, 2003.
14. Cristina Cabanillas, Manuel Resinas, and Antonio Ruiz-Cortés. *Automated Resource Assignment in BPMN Models Using RACI Matrices*, pages 56–73. Springer, Berlin, Heidelberg, 2012.
15. Cristiano Castelfranchi. Commitments: From individual intentions to groups and organizations. In *ICMAS*, pages 41–48. The MIT Press, 1995.
16. Mark d'Inverno, Michael Luck, Pablo Noriega, Juan A. Rodríguez-Aguilar, and Carles Sierra. Communicating open systems. *Artif. Intell.*, 186:38–94, 2012.
17. Yrjö Engeström, Reijo Miettinen, and Raija-Leena Punamäki, editors. *Perspectives on Activity Theory*. Cambridge University Press, Cambridge, UK, 1999.
18. Ronald Fagin, Joseph Y. Halpern, and Moshe Y. Vardi. *Reasoning about knowledge*. MIT Press, 1995.
19. Harry G. Frankfurt. Alternate possibilities and moral responsibility. *The Jounral of Philosophy*, 66(23), 1969.
20. Wesley Newcomb Hohfeld. Some fundamental legal conceptions as applied in judicial reasoning. *The Yale Law Journal*, 23(1):16–59, 1913.
21. Alessio Lomuscio, Hongyang Qu, and Franco Raimondi. MCMAS: an open-source model checker for the verification of multi-agent systems. *STTT*, 19(1):9–30, 2017.
22. Roberto Micalizio. Action failure recovery via model-based diagnosis and conformant planning. *Computational Intelligence*, 29(2):233–280, 2013.
23. Roberto Micalizio and Pietro Torasso. Agent cooperation for monitoring and diagnosing a MAP. In *Multiagent System Technologies, 7th German Conference, MATES 2009, Hamburg, Germany, September 9-11, 2009. Proceedings*, pages 66–78, 2009.

24. Roberto Micalizio and Pietro Torasso. Cooperative monitoring to diagnose multiagent plans. *Journal of Artificial Intelligence Research*, 51:1–70, 2014.
25. Anil Nigam and Nathan S. Caswell. Business artifacts: An approach to operational specification. *IBM Systems Journal*, 42(3):428 – 445, 2003.
26. OECD. Effective institutions and the 2030 agenda for sustainable development. http://www.effectiveinstitutions.org/media/Session_1_Background_note_SDG.pdf.
27. Munindar P. Singh. An ontology for commitments in multiagent systems. *Artif. Intell. Law*, 7(1):97–113, 1999.
28. Munindar P. Singh. Norms as a basis for governing sociotechnical systems. *ACM Transactions on Intelligent Systems and Technology*, 5(1):21:1–21:23, 2013.
29. Michael L. Smith and James Erwin. Role and responsibility charting (RACI). https://pmicie.starchapter.com/images/downloads/raci_r_web3_1.pdf, 2005.
30. Pankaj R. Telang, Munindar P. Singh, and Neil Yorke-Smith. Relating Goal and Commitment Semantics. In *Post-proc. of ProMAS*, volume 7217 of *LNCS*. Springer, 2011.
31. United Nations Children's Fund. Report on the accountability system of UNICEF. https://www.unicef.org/about/execboard/files/09\discretionary{-}{}{}15\discretionary{-}{}{}accountability\discretionary{-}{}{}ODS-English.pdf, 2009. E/ICEF/2009/15.
32. World Health Organization. WHO accountability framework. http://www.who.int/about/who_reform/managerial/accountability-framework.pdf, 2015.
33. Mounir Zahran. Accountability Frameworks in the United Nations System. https://www.unjiu.org/en/reports-notes/JIU%20Products/JIU_REP_2011_5_English.pdf, 2011. UN Report.

Instrumenting Accountability in MAS with Blockchain

Fernando Gomes Papi[1], Jomi Fred Hübner[1] and Maiquel de Brito[2]

[1] Federal University of Santa Catarina, SC, Brazil
fernando.papi@posgrad.ufsc.br, jomi.hubner@ufsc.br,
[2] Federal Institute of Education, Science and Technology, RS, Brazil
maiquel.brito@rolante.ifrs.edu.br

Abstract. In this paper, we investigate a proposal where Blockchains could provide powerful tools for Multi Agent Systems. Specially regarding accountability, Blockchains could prove to be very useful tools for agents to reason upon contracts, commitments, responsibilities and so on. We list some possible approaches to *integrate* blockchains and Multi Agent Systems, by providing adequate abstractions and discussing over possible advantages and disadvantages of these different abstractions.

Keywords: Multi Agent Systems, Blockchain, Smart Contracts, Computational Accountability

1 Introduction

Multi Agent Systems (MAS) are commonly used to model complex, distributed systems such as Smart Grids, Supply Chain Management, Virtual Auctions and Marketplaces, Virtual Collaborative Enterprises, Crowdfunding Platforms, Reputation Systems and so on. These could possibly be open MAS, where there is no guarantee, by design, of cooperation among the agents in these systems. Even the opposite could be true, there could be malicious agents that freely enter and leave the MAS. In this perspective, there is the need for tools that enable accountability in the system. According to [1], accountability is "the acknowledgment and assumption of responsibility for decisions and actions that an individual, or an organization, has towards another party." The key word is responsibility: agents should be responsible for their actions, decisions and commitments. Moreover, agents are responsible for whatever agreement they put themselves through. But how can we provide agents reliable tools so that they can be accountable for their actions?

Blockchains have a good perspective on this matter because they offer agents the possibility of trustless exchange, data that is consistent and timestamped, and complete transparency and immutability. The common development tools of Multi Agent Systems could benefit from some properties provided by a blockchain. For example, there is the absolute guarantee that an interaction between agents

recorded on the blockchain (as in, a message from agent A to agent B) was performed. If two agents resolve to sign a blockchain-based *Smart Contract*, there is the guarantee that this contract will be executed, even if that means penalizing an agent that breaks it. This could be a way to enforce cooperation, prevent malicious agents from harming the objective of other agents, and provide the system with logics for authorization, fairness and incentives, as described by [2].

Blockchain is currently on of the the most discussed topics in the area of financial technology. In 2009, an anonymous person (or group of people), under the pseudonym of Satoshi Nakamoto, released a white-paper [3] describing the world's first fraud-proof digital currency, the Bitcoin. The technology that enabled such technological disruption, which had been around for some time before the Bitcoin, is now known as the Blockchain, a data structure composed of chains of data blocks containing the record of all of the currency's transactions. Its cryptographic nature makes the Blockchain an unhackable time-stamped database of transactions, disseminating trust along a distributed network of nodes [4]. This characteristic enables a great number of applications that would benefit from disseminated consensus.

With this in mind, comes the questions: In terms of accountability, what could change with the integration of Blockchains and MAS? How can we successfully integrate a blockchain and MAS? What is the best abstraction from MAS for the Blockchain? If we are dealing with an environment based MAS, is the Blockchain like an ordinary environment or does it have special properties that agents should consider in their reasoning? Is it more effective if left as a generic artifact in the environment, or maybe the Blockchain would be better used if specifically tailored for the application in MAS? For example: in an auction application of MAS, the Blockchain could just receive bids and register winners, but it could also be implemented through Blockchain Smart Contracts to enforce the execution of the auction, where, in order to bid, an agent should have available funds and if it wins, the funds would be transferred automatically.

In this paper we present some initial discussion towards answering these questions and it is organized as follows. In Section 2, a quick explanation of how a blockchain works is given. In Section 3, possible approaches for the MAS and Blockchain integration is discussed considering their particularities, advantages and disadvantages. In Section 4, we present a practical example illustrating one of the proposed approaches. A brief conclusion is presented in Section 5.

Regarding nomenclature, there is an important note to be made: whenever talking about the Blockchain as a technology or system, the letter "B" will be uppercase. If we are referring to the data structure, the actual implementation of the ledger, then "blockchain" will be lowercase. Similarly, Bitcoin is written with an uppercase "B" when referring to the "system", while bitcoin with lowercase "b" refers to the tokens of the Bitcoin network, as in "the price is 2 bitcoins".

2 The Blockchain Technology

To the best of our knowledge, not many authors have yet proposed links between Blockchains and Multi Agent Systems, so it is worth defining what blockchains are, quickly mentioning how they work, as well as stating some terms commonly used in the blockchain community and along this paper.

In short, the blockchain is a database that has its data cryptographically signed to state its veracity. This data is packed up in blocks that will be chained by hash pointers pointing to the previous block of data. The most common data structure being used today is the Merkle Tree [5].

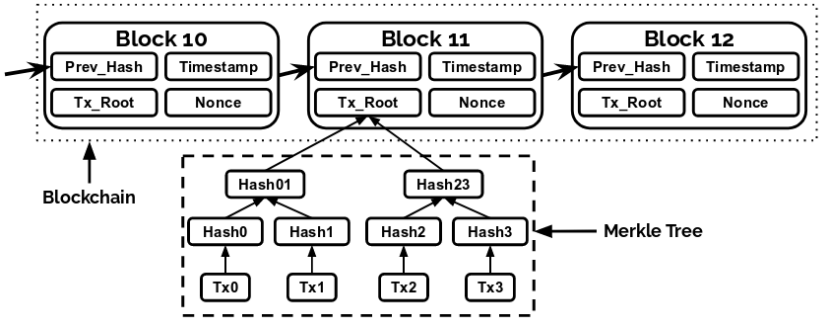

Fig. 1. Schematic of a blockchain

Figure 1 shows the structure of a blockchain. Each block contains a Merkle Tree, which is a linked tree of hashes of transactions in that block, along with a pointer that points to the previous block. The block has a timestamp and a Nonce, which is a number that was generated to approve that block by miners. Finally, it has the hash of the previous block. This means that in order to modify the data in previous blocks, it is necessary to change all the hash pointers of all the subsequent blocks to maintain consistency. In the case of Figure 1, changing data on block 10 means that the hash of blocks 11, 12... will need to be updated as well. These changes need to be replicated throughout the network, since the blockchain is essentially a peer-to-peer technology. In that case, only someone with control of at least 51% of the network will be able to do such changes. This is the reason why blockchains are considered fraud proof. The computers that execute this database are called nodes. Nodes are able to confirm if a block of transactions is valid by spending an enormous amount of computing power with the objective of solving a computational problem that can only be solved by *brute force* algorithms, generating the Nonce. This is the case of the Bitcoin ledger (some other projects offer different ways of confirming transactions). There is a reward, an incentive, for spending this computational effort

(called Proof of Work, or PoW), and when a node solves the puzzle and confirms a block, this block is appended to the blockchain and propagated through the network. This process of confirming blocks is commonly called mining. There are many sources that go deeper into the mechanics of the blockchain, such as [6].

The Blockchain was already a concept by the early 90's, but only rose to prominence after the 2008 financial crisis, when Bitcoin was created. Bitcoin solved a fundamental problem with digital cash: the double spending problem [5]. Since data can be easily replicated, that is, the same token could be sent to many people, it was not trivial to develop a system where an individual can spend a given token only once. Even though some projects already tackled this problem, Bitcoin became economically viable by creating a deflationary nature. The absolute amount of bitcoins that will ever exist is 21 million, according to Satoshi Nakamoto's original proposal [3]. Therefore, there is a tendency for an increase of its face value in relation to fiduciary currencies (which are not limited in supply since abandoning the gold standard in the mid 70's) if there is an increase in demand, according to the law of supply and demand.

The Blockchain quickly evolved to be more than a safe record of transactions to be a platform for decentralized computing, in the form of Smart Contracts. The Ethereum Project [7] uses the computational power from the nodes of its blockchain for decentralized computing of functions, instead of basically wasting it. That is, one can deploy pieces of code that will be executing as long as the blockchain itself is executing in the decentralized network. Nodes will be rewarded to perform this computations. These pieces of code are the Smart Contracts. Smart Contracts are immutable, where only parameters of the operations can be changed by participants and only when there is consensus about it. If two nodes agree upon a certain contract, there is the absolute guarantee that it will execute when conditions are met. A simple example is a contract that will transfer a particular amount of tokens from participant A to participant B when a bar code is read, meaning participant A bought something from participant B. Participant A will not be able to default on the agreement with participant B, and if there are no funds in the account, then the sale is never completed in the first place. This "either wholly do some specific action, or don't do it at all" is called an atomic operation, and is a powerful characteristic of Smart Contracts.

This particular concept of Smart Contracts could be very valuable to MAS applications. Currently, the Bitcoin protocol does not support Smart Contracts. The most developed platform for this technology is the Ethereum Project, though some other platforms are being evolved to support Smart Contracts, such as Decred [8].

3 Integration Models

In this Section we consider possible models of integration between Blockchain and MAS. First, in Section 3.1, a generic model is proposed, where the Blockchain serves as a tool for the secure propagation of messages among agents. Then, from Sections 3.2 to 3.2.3, we explore some more complex approaches where the blockchain is part of the environment where the agents act.

3.1 Blockchain as a means of communication

The first Bitcoin transaction block ever created contained the message "The Times 03/Jan/2009 Chancellor on brink of second bailout for banks" [9]. This message was the headline of the London, UK, newspaper "The Times", on January 3^{rd} 2009. This headline announced that the Government was on the verge of using taxpayers' money to bail bankers out of the crisis they had created. We cannot prove that the semantic content of the message is true. But we can prove, as long as a copy of the Bitcoin blockchain exists, that a transaction was made to the address "1A1zP1..." and this fact is what makes the blockchain such a promising technology. The owner of this address will never be able to state that his address did not receive 50 bitcoins upon the creation of the blockchain. Nor will s/he be able to change the contents of this transaction. After another 6 transactions occur, it is statistically impossible to change what was recorded on the blockchain [10]. The first transaction in the Bitcoin ledger supports the idea that blockchains can be used as effective and reliable means of communication.

Disregarding the economic value of the transactions in the blockchain, we could propose a Blockchain where each and every transaction would cost exactly 1 token (as an incentive for mining and confirming transactions), but it should contain a message that is structured in a predefined way, that is, it belongs to the formal semantics of a language. When agent A makes a transaction to agent B, agents A and B will never be able to argue that no message was ever exchanged between them. They might strongly disagree on the content of the message or the consequences this message might imply, but the existence of the triple $\langle Message, Interlocutor, Receptor \rangle$ is undeniable. This small example supports the idea that agents can use the blockchain as a way to exchange messages in an accountable way. It would be the most basic usage of a blockchain in a MAS, supporting a reliable means of communication among agents. It is reliable because the record of message exchanges will exist as long as a sufficient number of agents hold a copy of the blockchain. Technically speaking, the blockchain will exist if at least one agent executes it. But this agent might be able to change its contents, since it is the only node in the network. If there are two nodes in the network, and one node does not accept a change made by the other, then effectively there are two different systems being executed. If there are three nodes, when two nodes accept a change in the blockchain, the third node will be compelled to accept this change as well, in order to keep its records updated and its chain valid. This is the process of achieving consensus in the

network. Figure 2 illustrates this proposal, where agents use the blockchain to securely exchange messages between them.

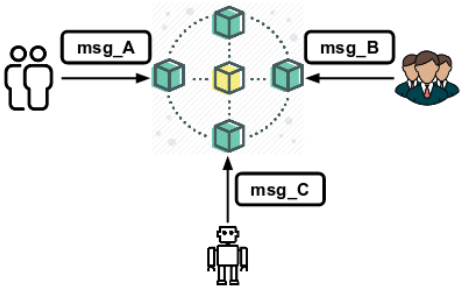

Fig. 2. Model of the Blockchain as a channel of communication between agents

However, this approach falls short in terms of accountability. Herlihy and Moise [2] bring the definition of two types of accountability: proactive and reactive. The proactive type of accountability happens upon *authorization*, such as in a scenario where agent A is authorized to make a transaction or send a message to agent B. The reactive kind would be the scenario where an agent is accountable for its actions after it has performed them. As stated by the authors, reactive accountability is of limited value if agents do not have adequate mechanisms to penalize other agents (or hold them accountable) for their actions. That is, since this approach only states that a message was exchanged between agents, only the reactive kind of accountability is achieved. Agents will still need extra tools to assess accountability. This creates the opportunity for blockchains to be more than message loggers in MAS, mainly because blockchains have evolved from distributed databases to distributed autonomous computation, in the form of Smart Contracts.

3.2 Blockchain as MAS environment

It is rational to make a case for a blockchain based environment for MAS when we look back at the history of mankind itself. Ian Grigg [11] states that blockchains are the third revolution in accounting. Thousands of years ago, primitive societies came up with the method of writing down transactions. Before that, humans hunted and farmed and were able only to barter these goods among a limited range of other people, normally within the same village or tribe. When the single-entry bookkeeping emerged, along with the usage of currency, numbers and writing, it was now possible to record the exchange of values and goods between a wider range of people and keep record of who owes what to whom. This created the possibility of a more complex system of commerce, which commonly boosts human societies development. The double-entry bookkeeping method,

created in the medieval republics and kingdoms of modern Italy, enabled even more complexity in commerce. It allowed for better tracking of errors and audition of exchanges, since every transaction was both an *asset* and a *liability*, and both columns of assets and liabilities must sum to the same value. With a more robust method of transaction recording, goods were able to flow from one empire to the other.

As commerce and transactions grow more complex, better ways of auditing transactions are needed. Companies and governments commit fraud in their accounting books and are rarely caught or punished. Thus, the accountability of today's system is broken. Blockchain is a new revolution in accounting because it is a triple-entry bookkeeping system. Agent A transacts with agent B, and a large number of agents confirm that the transaction is valid, though none of these agents actually know who are A and B, nor A and B know who are the agents who confirmed the transaction. This anonymity in the chain of transactions adds another layer of security to the system. This is possible only through the peer-to-peer replication of the ledger, while cryptography keeps the consistency along these replications. The Blockchain is a solution to the Byzantine Generals Problem, common in distributed computing [12].

This time, rather than emulating human societies' social realities and technologies to evolve MAS, we are able to develop a blockchain model for MAS *on the go*, that is, while it is also being applied and improved in different use cases. This scenario presents more opportunities than using blockchains as the channel of communication between agents, as previously proposed. Today, it is generally accepted that the environment is an essential part of the development of Multi Agent Systems, its justification being derived from many different subjects. Classic AI not only brings the concept of environment, but also defines agents as anything that perceives, through sensors, and acts upon its environment through effectors, such as the definition brought by Russel&Norvig [13]. In the MAS domain, artifacts have been proposed as first class entities to compose the MAS environment in order to achieve better coordination between agents and to provide them with information about and functionalities in regard to their environment, allowing for enhanced cognition. Artifacts are said to be reactive entities to provide functions that make agents cooperate in MAS, and shape the environment according to the system's needs [14].

In the sequence, we discuss possible places for the Blockchain in the environment shared by agents.

3.2.1 Blockchain as a generic environment

Blockchains could be modeled as the whole environment of the MAS. The Ethereum Project is the first prominent provider of Smart Contracts and will be utilized in this work. All the artifacts that agents need or have access to could

be coded as Smart Contracts in the chosen Blockchain platform. The advantage to this approach is largely the simplicity of developing every needed artifact on the same platform. Also, every single action or transaction made by agents will be recorded and immutable, therefore accountable. Whether an agent is delegating a task to another agent, sending another agent a particular amount of currency or simply stating a fact, everything will be forever recorded, as long as the blockchain exists.

But this does not come cheaply. Blockchain transactions are expensive regarding computational effort. Specially for large public permissionless blockchains, such as Bitcoin or Ethereum. For example, a simple Bitcoin transaction takes at least 10 minutes to be executed, and could take up to one hour to be considered fraud-proof (also said to be "confirmed", when at least 6 blocks of transactions have been linked to the blockchain after the block containing the transaction was added). Today, it operates in the speed of about 2.5 transactions per second.

Scalability is definitely Blockchain's main challenge for mass adoption today. The Bitcoin ledger was larger than 100GB by the end of 2016 and some implementations of the Ethereum's ledger were larger than 25GB by early 2017. As the technology gains adoption, this problem will only worsen. Satoshi Nakamoto's idea on this was that Moore's Law would take care of both processing power and storage needed to run a full node of the Bitcoin ledger. But maybe he did not foresee the large attention Bitcoin has drawn in less than a decade. With the concentration of mining in just a few "players" running mainly in China and Russia, the development of the system became slower and more difficult, specially when there are conflicts of interest between miners, developers and users.

This can be a huge drawback for the approach of implementing every artifact in the blockchain. The whole system could become impracticably slow, with every single action needing confirmation before being executed. This does not make sense, specially when we think of human societies: we do not register every single interaction we have with other humans or objects. We register only very special interactions, normally those that involve promises with serious consequences, contracts, transactions of valuable assets and so on. This brings the idea that perhaps a Blockchain is better modeled for MAS as one artifact available to the agents, and not the whole of the environment, in order to register special interactions that agents find suitable. Figure 3 illustrates this proposed model, where agents will only have access to artifacts and other agents through the blockchain.

3.2.2 Blockchain as a single artifact in the environment

We have discussed the idea of using a blockchain as the technology to provide the whole environment for the agents, arguing that it could be unfeasible since every action would need to be registered in the blockchain. Since it is probable

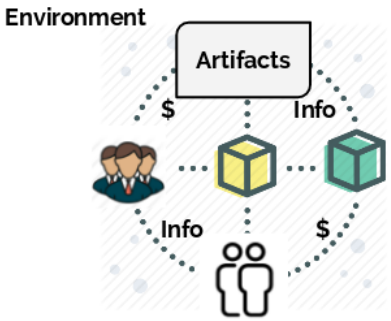

Fig. 3. Model of the Blockchain as a generic environment

that not all, but only a few, of the actions taken by agents need to be registered for accountability, we can model the blockchain as a single artifact available to the agents.

This way, agents will be able to perceive the blockchain, obtaining information from it and will be able to act by registering data on it. The abstraction provides a blockchain node running behind an artifact in the environment. Such artifact provides the agents only one type of action: blockchain transactions. By using a blockchain that does not support Smart Contracts, such as the Bitcoin protocol itself, agents are limited to transferring amounts of bitcoin and sending messages with this transaction. Though there are some limitations to sending messages in the Bitcoin protocol [15], it wouldn't be hard to conceive a blockchain that is suitable for this task. Figure 4 shows the concept of this proposal, where the blockchain is modeled as one simple transaction artifact in the environment.

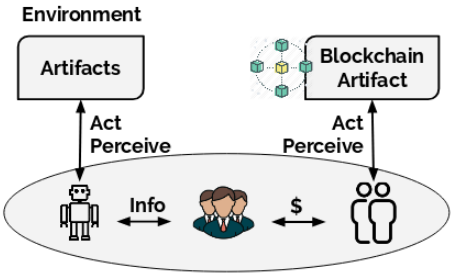

Fig. 4. Model of the Blockchain as a single artifact in the environment

This approach is fairly reasonable in cases where there is no need for extra complexity in the design of the system. In the common MAS example of an auction among agents, they can use cryptocurrencies to make payments for their bids, thus needing only to be in the sending/receiving end of a transaction. But again, this will not suffice for accountability. What if agent A, responsible for conducting the auction, does not deliver accordingly? And if agent B, winner of the auction, does not pay what was agreed upon? If it didn't have the necessary funds, why was it able to join the auction in the first place? Who will hold them accountable for the tasks they did not complete? Surely enough, all these considerations should be made upon the design of the system, in order to provide agents with the necessary tools for handling these scenarios.

3.2.3 Blockchain instrumenting application artifacts

Merlihy and Moir [2] cites examples of accountability problems that arise when blockchains are inserted in societies of agents, such as "because A endorsed false statement x, A can no longer be trusted with the nuclear code". These kinds of problems have long been studied by the Multi Agent Systems community, and many normative models are proposed. It is valuable noting that there is a reliable, cryptographic, fraud-proof way to solve these types of problems. Smart Contracts are a way to provide a safe logic of authorization and incentives mechanism for the agents related to the MAS.

It has been long accepted that the environment is not something that just exists, but that this environment should have a meaningful design as to support agents in their tasks [16,14]. Since Smart Contracts are programmable contracts that can be developed to perform specific functions, it makes sense to envision these Smart Contracts as specific artifacts in the environment. For example, a voting system to decide whether agent A has endorsed a false statement can be modeled to be exactly this, a voting artifact (with all the advantages of the blockchain). The nuclear code can be another artifact, with much more security embedded, for example, a strict method of access available only to agents carrying very specific roles or authorizations. These choices of design really depend on the requirements of the system and its goals. Blockchains can provide those logics of authorization, fairness and incentives.

When an agent registers itself on a contract artifact, it becomes responsible for whatever agreement this contract implies. If this contract states that all agents participating in it should vote regarding a particular issue and there is a penalization for not doing so, all agents that do not perform the required task can be automatically penalized. This means that this accountability is, in some way, automated. It is a proactive accountability. The Blockchain can create a whole new layer of trust among agents. There are many studies that propose Reputation Models for MAS. With blockchains, these Reputation Models become very powerful tools when we are dealing with Open MAS, since they cannot be

tampered with. Figure 5 shows how, in this model, artifacts model the interface between agents and the blockchain. Agents are still able to flexibly communicate amongst themselves, and access other types of artifacts.

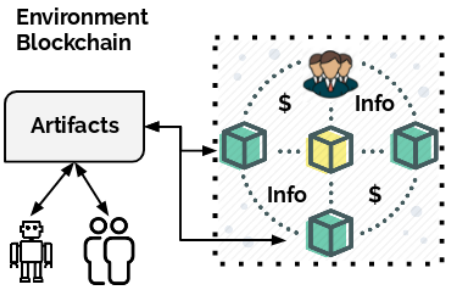

Fig. 5. Model of the Blockchain as specific artifacts in the environment

The scalability of this model depends greatly on the requirements of the system. The Ethereum Blockchain is essentially a world computer: users pay mining nodes for performing computations for them. But it is a very slow computer (somewhere around 10 to 30 executions per second). If the system requires the advantages brought by blockchains in all of its artifacts, then it must loose its requirements of speed. The number of artifacts implemented in the blockchain is not necessarily limited, but there will always be a compromise of execution time and number of interactions with the blockchain. The example of the house building auctions, presented later, makes this clear: modeling the auctions in separate artifacts makes the execution orders of magnitude slower than a simple execution of the MAS by itself (which shouldn't take longer than a couple of seconds, for example). But when actions don't need to be held accountable, they do not need to be unnecessarily incorporated in the Blockchain. For example, when the agent that creates the auctions tell other agents that auctions are open. This design trade-off will always be present.

One of the main challenges from this point on is how to design agents that will reason upon these characteristics provided by blockchains. Can we make agents distinguish the hard consequences of recording data in the blockchain rather than another database that can more easily be tampered with? This approach creates new possibilities for the advancement of Multi Agent Systems.

4 Case Study: Building a House

This Section presents the implementation of the model described in Section 3.2.3. We aim to demonstrate how can Smart Contracts be mapped into artifacts, and

what are the necessary tools for this integration. Also, we can evaluate what is the impact of using a blockchain, a very slow system, with a much faster MAS programming platform.

For the design of the MAS, we have used JaCaMo [17], a programming language that provides abstractions of agents, environment and organizations. The Smart Contracts will be implemented in the Ethereum Platform, using the Parity [18] implementation of the Ethereum client. The communication between the MAS implemented in JaCaMo and the Smart Contracts will be done via the Web3j Java library [19]. The example is implemented in the Testing Network Kovan, for simplicity [20]. The source code is publicly available. [3]

The problem consists of an agent, Giacomo, that wants to build a house. Giacomo knows all the tasks that need to be done, and knows the budget for each task. Tasks are steps needed for completing the construction of the house, such as "laying the foundation", "building the floor", "building walls" and so on. Since each task has a budget, Giacomo will launch individual auctions for tasks. The whole of this example is a little bit more complex than explored in this work, involving an organization to ensure completion of tasks, but we will be focusing on the first part of the simulation, namely the *contracting phase*. Agents with the role of constructors will place bids for the tasks they want to execute. The bids will be placed directly in the blockchain, and the Smart Contract will make sure the smallest bid wins. When a given deadline is reached, Giacomo will check the Smart Contract to announce the winner for a task.

At this point, the winner will never be able to state that it wasn't in fact the winner. It has the responsibility over the commitment it made when bidding in the auction. With some complexity added, the blockchain could take care of a few extra enforcing steps, such as registering a proof of completion of the task, and the direct payment from Giacomo to the winner. The Auction artifact has the Observable Properties *task_description*, *max_value*, *current_winner*, *best_bid* and the Operation *bid*. Disagreements between agents could be resolved by looking at the data available in the Blockchain. One of the challenges resides in constructing Smart Contracts that are verifiable by all agents involved, and that any agent interacting with it can trust. For example, a Smart Contract can have its code hashed and implemented as an Observable Property. Then a copy of the same Smart Contract can be deployed in the blockchain so that agents can use it to test its functions as they wish, without further commitments. Once they are confident in the fairness of the copy, they can securely interact with the original.

As mentioned earlier, the execution time of such a system is orders of magnitude slower than a pure MAS built in JaCaMo, since each bid made by constructors need to be registered in the blockchain. For reference, this example deploys

[3] https://github.com/FerPapi/HouseBuilding_JaCaMo-Blockchain

10 different Smart Contracts to the network, each interfacing a different task. Each contract receives several transactions since agents are using the Operation *bid*, which requires a change in the state of the contract that must be confirmed by the network. It is also worth noting that if this was a real application deployed in Ethereum's Main Net, "real" Ethereum Tokens, called Ether, would be needed. By today's market price, each Ether costs about US$ 300,00. It is not trivial to estimate how much this example would cost in US Dollars, but given that this is a fairly simple execution, it shouldn't be more than a few cents of the dollar. More complex Contracts that are executed for longer periods will indeed be more costly. The advantage of this cost is that a MAS integrated with a Blockchain can be used for real life applications involving transaction of assets without further complexity, such as integrating to traditional banking payment systems.

Regarding accountability, the blockchain will provide agents with reliable data and statements from the past. It can also automate the processes of payment and penalty enforcement, which makes the system more reliable overall. But blockchains will do exactly as they are programmed. So, in order to program them correctly for accountability of agents, external models and frameworks will be necessary, such as the work presented in [21]. This work also uses this house building example to demonstrate the application of an accountability protocol in MAS.

5 Conclusion and Further Works

Along this paper, we have discussed the adoption of a new technology into Multi Agent Systems. The blockchain is promising a revolution in accounting, finance and technology. We have briefly explained how a blockchain works and the reasons for integrating blockchain into MAS. Then we argued about approaches for modeling blockchain in a MAS: as a standalone means of communication between agents, as the environment as a whole, as a single artifact in the environment performing only transactions, or as a provider of different meaningful artifacts for the agents, going over advantages and disadvantages of each approach.

We have decided to use the blockchain to model specific artifacts in the environment, thus delegating the functionality of these artifacts to Smart Contracts. The interaction between agents and Smart Contracts is then performed through artifacts in the MAS. With this model, we implemented an integration of Ethereum's blockchain to a MAS running on JaCaMo, using a specific Java library for this purpose. The integration was successful, and an example was presented. The example was also implemented successfully, achieving its objectives. But it made clear that, when using blockchains along with MAS, there will be a trade-off between taking advantage of the blockchain's capabilities and speed. The system runs much slower when the blockchain is present, since transactions

need to be confirmed by the network.

The Multi Agent Systems community has the opportunity to not only improve the MAS field itself, but also to contribute to the blockchain community with frameworks for development of pure blockchain systems. For instance, the MAS tools that implement high level abstractions as commitments [22], norms [23], and situated institutions [24] can bring significant contributions for the blockchain community. Further works in this topic would include further discussions on the most useful way of using blockchains in MAS, as well as applications in decentralized systems to simulate how agents would react and interact in the presence of this new technology. Scalability poses one of the great challenges for this new approach, and this could be tackled in further works as well.

References

1. M. Baldoni, C. Baroglio, K. M. May, R. Micalizio, and S. Tedeschi, "Computational accountability," in *Proceedings of the AI*IA Workshop on Deep Understanding and Reasoning: A Challenge for Next-generation Intelligent Agents 2016 co-located with 15th International Conference of the Italian Association for Artificial Intelligence (AIxIA 2016), Genova, Italy, November 28th, 2016.*, pp. 56–62, 2016.
2. M. Herlihy and M. Moir, "Blockchains and the logic of accountability: Keynote address," in *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science, LICS '16, New York, NY, USA, July 5-8, 2016*, pp. 27–30, 2016.
3. S. Nakamoto, "Bitcoin: A peer-to-peer electronic cash system," 2008.
4. M. Swan, *Blockchain: Blueprint for a new economy.* " O'Reilly Media, Inc.", 2015.
5. A. Narayanan, J. Bonneau, E. Felten, A. Miller, and S. Goldfeder, *Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction.* Princeton University Press, 2016.
6. A. M. Antonopoulos, *Mastering Bitcoin: unlocking digital cryptocurrencies.* " O'Reilly Media, Inc.", 2014.
7. V. Buterin *et al.*, "Ethereum white paper," 2013.
8. "Decred roadmap." available at *https://medium.com/decred/2017-decred-roadmap-d0da20c39db3.*
9. "Bitcoin wiki: Genesis block." available at *http://en.bitcoin.it/wiki/Genesis_block.*
10. "Bitcoin wiki: Confirmation." available at *https://en.bitcoin.it/wiki/Confirmation.*
11. I. Grigg, "Triple entry accounting." available at *http://iang.org/papers/triple_entry.html.*
12. A. Miller and J. J. LaViola Jr, "Anonymous byzantine consensus from moderately-hard puzzles: A model for bitcoin," *Available on line: http://nakamotoinstitute.org/research/anonymous-byzantine-consensus*, 2014.
13. S. J. Russell and P. Norvig, *Artificial Intelligence - A Modern Approach (3. internat. ed.).* Pearson Education, 2010.
14. A. Omicini, A. Ricci, and M. Viroli, "Artifacts in the a&a meta-model for multi-agent systems," *Autonomous Agents and Multi-Agent Systems*, vol. 17, no. 3, pp. 432–456, 2008.

15. "Embedding data in the blockchain with op_return." available at *https://21.co/learn/embedding-data-blockchain-op-return/*.
16. D. Weyns, A. Omicini, and J. Odell, "Environment as a first class abstraction in multiagent systems," *Autonomous Agents and Multi-Agent Systems*, vol. 14, no. 1, pp. 5–30, 2007.
17. O. Boissier, R. H. Bordini, J. F. Hübner, A. Ricci, and A. Santi, "Multi-agent oriented programming with jacamo," *Sci. Comput. Program.*, vol. 78, no. 6, pp. 747–761, 2013.
18. "Parity technologies." available at *https://parity.io/*.
19. C. Svensson, "Web3j." available at *https://web3j.io/*.
20. ""kovan" public testnet to provide a stable environment for ethereum development." available at *https://github.com/kovan-testnet/proposal/blob/master/Press%20Release.md*.
21. M. Baldoni, C. Baroglio, K. M. May, R. Micalizio, and S. Tedeschi, "Supporting organizational accountability inside multiagent systems," in *AI*IA 2017 Advances in Artificial Intelligence - XVIth International Conference of the Italian Association for Artificial Intelligence, Bari, Italy, November 14-17, 2017, Proceedings*, pp. 403–417, 2017.
22. M. Baldoni, C. Baroglio, F. Capuzzimati, and R. Micalizio, "Commitment-based agent interaction in jacamo+," *Fundamenta Informaticae*, 2017.
23. J. F. Hübner, O. Boissier, and R. H. Bordini, "A normative programming language for multi-agent organisations," *Ann. Math. Artif. Intell.*, vol. 62, no. 1-2, pp. 27–53, 2011.
24. M. de Brito, J. F. Hübner, and O. Boissier, "Situated artificial institutions: stability, consistency, and flexibility in the regulation of agent societies," *Autonomous Agents and Multi-Agent Systems*, pp. 1–33, 2017.

Towards the Specification of Natural Language Accountability Policies with AccLab: The Laptop Policy Use Case

Walid Benghabrit[1] and Jean-Claude Royer[1] and Anderson Santana De Oliveira[2]

[1] IMT Atlantique, site de Nantes, 5 rue A. Kastler, F-44307 Nantes, France
{firstname.lastname}@imt-atlantique.fr
[2] SAP Labs France, 805 avenue du Dr Donat Font de l'Orme, France - 06250, Mougins, Sophia Antipolis
anderson.santana.de.oliveira@sap.com

1 Introduction

Accountability allows to assign legal responsibility to an entity. It is the basis for many contracts, obligations or regulations, either for digital services or not. In our previous work we studied the Abstract Accountability Language for expressing accountability [BGRS15,RSDO16] with a logical focus.

We demonstrate the AccLab tool support over a set of policies from the Hope University in Liverpool governing the use of their IT systems and computer resources. These policies are representative of terms of use and other agreements, being really part of the University management, under control of the University Council. Moreover, this application context is more familiar to the general public than other domains, such as financial or health care services.

One important task is to analyze and interpret these policies which are written in a natural, sometimes legal style. Of course, during this analysis we found many ambiguities, omissions, inconsistencies and other problems, but our purpose is to demonstrate that a part of it can be formalized and automatically verified. This paper presents the laptop user agreement policy and discussed its specification with our tool support.

2 Use Case Introduction

The policy of interest is a set of seven texts from [Uni17] related to IT concerns and data protection. We will focus here on the laptop agreement. The laptop agreement is rather short (10 clauses), included below.

In accepting the use of a University laptop, I agree to the following conditions:

1. *I understand that I am solely responsible for the laptop whilst in my possession*
2. *I shall only use the laptop for University related purposes.*

3. *I shall keep the laptop in good working order and will notify I.T. Services of any defect or malfunction during my use.*
4. *I shall not install and / or download any unauthorized software and / or applications*
5. *I shall not allow the laptop to be used by an unknown or unauthorized person. I assume the responsibility for the actions of others while using the laptop.*
6. *I shall abide by the University Acceptable Use, Information Security and Portable Data Device security policies as published on the I.T. Services Website*
7. *Any work saved on the laptop will be deleted prior to the machine's return*
8. *If the laptop is lost, stolen or damaged, the incident must be reported to the University Secretary's Office within 24 hours*
9. *If the lost, stolen or damaged laptop and / or accessories is determined to be caused by negligence or intentional misuse, I shall assume the full financial responsibility for repair costs or fair market value of the laptop.*
10. *I am aware that any breach of these policies may render me liable to disciplinary action under the University's procedures*

In the remainder of this paper we will go through the exercise of formalizing and verifying this policy.

2.1 AAL

Our policies are written with the Abstract Accountability Language (AAL) and we will comment the main constructions. In our approach, we consider that an accountability clause should express three things: a *usage*, an *audit* and a *rectification*. The usage expression describes access control, obligations, privacy concerns, usage controls, and more generally an expected behavior. In the course of the design of AAL we reviewed several related languages, and our requirements are aligned with several points raised by [BMB10]. Expressiveness is ensured by negation, unrestricted set of actions, type hierarchies, conditions and policy templates. We also advocate for a readable language with succinct unambiguous semantics. In this context we introduce notions of permission, interdiction and obligation, but we do not have the exact concepts of deontic logic since we want to get our approach free from paradoxes and we aim at reusing classic logic with its tool support. Point to point communications with messages passing, used by many related work, is a good abstraction, simple and flexible. The audit and rectification expressions are similar to usage expression but dedicated to auditing, punishment and remediation. The audit expression defines a specific audit event which triggers the auditing steps. The rectification expression denotes actions that are done in case of usage violations. For instance, to punish the guilty party and to compensate the victim agent. We follow [Sch99,Mul00] which argue that punishments and sanctions are parts of accountability. AAL allows to define types with union, intersection, inclusion and negation to help in modeling data and roles. An action or a service call is represented as a message `sender.action[receiver](parameters)`. The language enables predicates (prefixed by @)

and if `Type` is a type, `@Type` denotes its associated unary predicate. Authorizations are denoted by the `PERMIT` and `DENY` keywords prefixing an action or an expression. The language provides Boolean operators, first-order quantifiers and linear temporal operators. We need to define policies and to reuse them thus we implement a notion of template (introduced by `TEMPLATE`) which improves readability and structuration. A template enables to name a policy with parameters (called with `@template`) but it also supports higher-order definitions helpful in defining common schemas of accountability or usage.

2.2 The Laptop Policy

As we can see in the policy from Section 2, sentences are often vague but sometimes they contain more precise information. It is the case in other policies like the data protection, data portable or information security policies. Thus we did a first reading of all the policies to extract some elements about the information system. From that, we conclude that the "I" is referring to a student or a staff of the university (named here `resp`) which is explicitly defined as of type `EligibleUser` in the IT usage policy. In fact another kind of person (`AllPerson`) appears in clause 5 which obliged us to reconsider clause 2 with a different meaning as other persons may use the laptop assigned to `resp`.

As a first simple example the clause 10 above states that "disciplinary action under the University's procedures" may take place in case of breach. This defines the policy rectification and we represent it as an abstract action in Listing 1.1 with a simple policy with a typed parameter. `LHU` is a constant denoting the University representative lawyer, it may be a fictitious or a real person.

Listing 1.1. Simple Rectification in AAL

```
TEMPLATE LHURectificationPolicy (resp:AllPerson)
       (IF (@EligibleUser(@arg(resp))) THEN {LHU.disciplinaryAction[@arg(resp)]()})
```

In the laptop policy there is no more information about rectification and nothing about audit. This last can be reduced to `auditor.audit[LHU]()`. As in the case of rectification once we have more details, the template construction allows to refine these definitions. A large section of the laptop policy describes in fact information about permissions, prohibitions, obligations and some conditions. Most of this usage policy will be described by a template named `laptopUA`, see Listing 1.2. Only our interpretation of few clauses is given but this policy covers clauses 1 to 9. The first line says that if a person is permitted to use a laptop then he is an eligible user and this laptop was assigned to him. We also assume that the eligible user should bring back his assigned laptop in the future to the university secretary.

Listing 1.2. Laptop Policy Agreement in AAL

```
TEMPLATE laptopUA(resp:AllPerson)(
(FORALL laptop:Laptop FORALL p:Purpose (IF (PERMIT @arg(resp).use[laptop](p))
          THEN {@EligibleUser(@arg(resp)) AND @assigned(@arg(resp), laptop)})) AND
(FORALL laptop:Laptop FORALL p:Purpose
       (IF (@EligibleUser(@arg(resp)) AND @assigned(@arg(resp), laptop))
             THEN {SOMETIME (@arg(resp).bringBack[LHUsecretary]())})) AND ...)
```

They are many things which are left implicit, are wrong or skipped in natural language policies. One important benefit is that problems and omissions appear during the specification phase or will be detected by the verification tools.

The laptop clause 2 is simple, however, it interacts with the clause 5 which leaves open the use of the laptop by a known and authorized person, maybe a colleague. But in fact we do not have information about which kind of person is permitted and for what kind of purposes. Other technical problems occur with the clause 5, 7 and 8: they need explanations, as we will see later. Finally, the overall structure of the laptop accountability policy is as in Listing 1.3. The accountability policy has two parts, the first is related to all clauses except clause 7 which is related to the second part.

Listing 1.3. Laptop Accountability Policy in AAL

```
TEMPLATE LaptopAccountabilityPolicy (resp:AllPerson) (
// from 1 to 10 but clause 7
@template(ACCOUNT, @template(laptopUA, @arg(resp)),
                   @template(LHURectificationPolicy, @arg(resp)))  AND
// from clause 7 since it is another schema
@template(ACCUNTIL, @template(condition7, @arg(resp)),
  @template(achievement7, @arg(resp)), @template(LHUWeakRectificationAndPay, @arg(resp)))})
```

The ACCOUNT and ACCUNTIL templates define accountability schemas. Let us describe the most classic which is quite similar to the one used in our previous papers, see Listing 1.4.

Listing 1.4. Basic Accountability Template in AAL

```
TEMPLATE ACCOUNT(UE:Template, RE:Template) (
 ALWAYS (auditor.audit[LHU]() AND (IF  (NOT(@arg(UE)))
                 THEN {ALWAYS (IF (auditor.audit[LHU]()) THEN {@arg(RE)})})))
```

This template assumes that the audit is simple and at each instant (in linear time) if the usage (`@arg(UE)`) is not satisfied then rectification (`@arg(RE)`) applies in case of an audit. The ACCUNTIL will be discussed later. Note that for clause 7 we diverge from the original text and we choose a weaker rectification to illustrate the language flexibility.

3 AccLab

Formal specifications are beneficial but they are really more effective if we have tool support and preferably some automated verification means. Thus we develop AccLab for Accountability Laboratory to experiment the specification, verification and monitoring of accountability policies. AccLab is compound from a set of tools which are: The component editor, the AAL editor and its verification, and the monitoring tools. The last release of AccLab is version 2.2 which was released on July 23, 2017 on github (`https://github.com/hkff/AccLab`) under GPL3 license. The AccLab IDE is a web interface that provides a component diagram editor and tools to work with the AAL language. The back-end is written in Python3 and the front-end in JavaScript based on `dockspawn` (`http://www.dockspawn.com`) which is a web based dock layout engine released

under MIT license. For verification purposes AccLab is interacting with the TSPASS tool ([LH10]), a prover for first-order linear temporal logic ($\mathbb{FOTL}$). The implementation is still in progress we will give an overview of its main current features. Some features like the full type constructions and the template are not fully operational thus we mix the use of AccLab and the TSPASS prover with some manual manipulations to process our example.

To manage more easily the AAL language a dedicated editor has been implemented. This editor is directed by the syntax and highlights the language keywords. There are syntactic checking but also semantic controls for type checking and the consistency of the declared services. A panel in the editor arranges a set of tools providing assistance in writing by the use of dedicated templates, for instance generating type declarations, accountability clauses or specific privacy expressions. This panel also contains few verification tools mainly the conflict checking with localization and the compliance checking. AccLab translates the AAL language into $\mathbb{FOTL}$ and the checking tools use the connection with TSPASS and its satisfiability algorithm. In case of unsatisfiability we implemented a mean to isolate the minimal core unsat. The tool provides macro calls which are useful in automating some complex tasks related to the translation to $\mathbb{FOTL}$ and the interaction with TSPASS.

One idea behind AccLab is to see accountability in action, and one way to achieve that is to be able to run simulations. AccLab includes a simulation module that allows it to monitor agents in a system and to observe accountability in action. We also proposed a tool called AccMon which reuses the above monitoring principles and provides means to monitor accountability policies in the context of a real system.

4 Lessons Learned and Discussion

This is an ongoing work but from now on we formalize several parts. One important task was to built and partly invent the information system and to get it consistent.

4.1 Accountability Schemas

The basic accountability scheme for a given usage is expressed simply as `(UE OR RE)`, or equivalently `IF (NOT UE) THEN RE`, where `UE` is the usage expression and `RE` the rectification expression. However, often we need quantifiers, for instance to identify the responsible user. We also consider time and our choice was to consider linear time as it seems sufficient in many cases. Thus we write formulas like `ALWAYS FORALL resp:Any IF (NOT UE(resp)) THEN RE(resp)`, or `FORALL resp:Any ALWAYS IF (NOT UE(resp)) THEN RE(resp)`. These are two distinct schemas, the first implies the second but the reverse is false (it is related to the Barcan formula).

Note that the above scheme allows to define first order accountability, that is the user is responsible and will be subject to rectification in case of a violation. But it is possible to define second order accountability, that is the processor or

implementer is responsible to enforce `IF (NOT UE) THEN RE` and in case of violation it is rectified with `RE2`. In this case, the schema will be `(UE OR RE OR RE2)`, and more generally higher-order responsibility is defined as `(UE OR RE OR ... OR REn)`. Templates are useful here to capture relevant accountability schemas.

4.2 Laptop User Agreement

We consider to have a full specification of the laptop agreement but it evolved since our first specification. We prove the satisfiability with most of the types and actions declarations and we also prove that violation of every clause will result in a rectification of the responsible person. As a general comment it is always possible to represent any kind of information but without semantics it remains purely syntactic. Of course aligning the syntax between different policies is a non obvious requirement. In our work we elaborate a rich information model resulting from the analyzing of most of the 7 policies. We have a type hierarchy with nearly 50 types and 50 relations, a set of 40 actions and more than 40 predicates and constants. This is a critical task because there is more or less nothing in the texts and sometimes they have some flaws. An important part is to add some behaviours to link together actions and predicates. Except type relations the action and predicate the behaviour is still poor but may be enriched later with the analysis of other policies.

The overall structure of the laptop policy has been given in Listing 1.3. We succeed in proving the satisfiability of the clauses and taking into account some user behaviours. The $\mathbb{FOTL}$ formula generation takes around 3s, the prover generates a total of around 1000 conjunctive normal forms in less than one second. We express several different user behaviours, for instance (see Listing 1.5) once a user signs the policy he gets an assigned laptop and accept the policy until he leaves.

Listing 1.5. A user behaviour

```
FORALL res:AllPerson FORALL laptop:Laptop
   ALWAYS (IF (resp.signed[LaptopAccountabilityPolicy]())
          THEN {(@assigned(resp,  laptop) AND @template(LaptopAccountabilityPolicy, resp))
                  UNTIL (resp.leave[LHU]())})
```

Indeed, the interaction between the proper user behaviour and the policy is a critical point for semantic reasons but it also brings some difficulties about quantifiers. The above formula is not monodic (this is a constraint for decidability of satisfiability) and we reformulate it with the laptop quantifier inside the accountability policy rather than in the user behaviour.

With the laptop clause 7 there is a technical point to discuss. The simplest formulation is to use an `UNTIL` operator, however this is more tricky to align with our `ACCOUNT` accountability clause which is based on a `ALWAYS` pattern. One solution is to use the principles of the separated normal form, for instance from [Fis11]. We can rewrite this expression as an `ALWAYS` clause without the need of the `UNTIL` operator. However, the result is not intuitive and only understandable by specialists of linear logic. We implement another solution that is to define a separate

accountability clause based on a different pattern. This pattern is `ACCUNTIL = (A UNTIL B) | (A AND NOT B) UNTIL NOT (A OR B)`. The second term of this pattern represents a part of the negation of the first, and then the case where rectification should happen. This pattern is not equivalent to the `A UNTIL B OR NOT (A UNTIL B)` scheme, but the difference is on the negative part. This difference is an infinite trace which is not really monitorable thus it can be discarded. Nevertheless, this behaviour introduces new quantifiers and the satisfiability process does not terminate. We reconsider it and succeed with a weaker specification (see Listing 1.6) and additional behaviour for the delete action.

Listing 1.6. Clause 7 specification

```
FORALL laptop:Laptop IF (@assigned(resp, laptop) AND resp.bringBack[LHUsecretary](laptop))
                     THEN {@deletedSavedWork(resp)}
```

The clause 8 introduces a notion of real time with "before 1 day", while we have a construction for that we choose to replace it with a `NEXT` construction. In fact there are only three such references in all the 7 policies. But during the verification step we realize that it is not sufficient because: It is acceptable that the user reports immediately and this is not acceptable that he reports after two states. Furthermore the `ALWAYS` pattern is not correct since the rectification may occur before the violation will be effectively realized. Thus, as for clause 7, a separate accountability pattern can be used, see Listing 2.

Listing 1.7. Clause 8 specification

```
FORALL laptop:Laptop
   (IF (@EligibleUser(resp) AND @assigned(resp,  laptop)
                        AND (@lost(laptop) OR @damaged(laptop) OR @stolen(laptop)))
      THEN {(resp.report[LHUsecretary](problem)
             OR (NEXT (resp.report[LHUsecretary](problem))))})
```

Another important point to note is related to clause 5 which states that "I assume the responsibility for the actions of others while using the laptop". The "I" is represented by `resp` in our usage policy and appears at the level of the accountability policy, see Listing 1.3. From that we can verify that if another user did a breach with the laptop assigned to `resp` then rectification is effectively applied to the responsible person. These verifications where successfully done, however, not without some technical difficulties.

4.3 Discussion

This is an ongoing work, as we need to complete our information model, to formalize other policies and at least to verify their consistency. The current natural language policies have many drawbacks: lack of precision, redundancies, ambiguities etc, unsurprisingly. Regarding accountability there are some information about monitoring and really few details about the remediation, compensation and punishment parts. One important task before any formalization of the policies is to build a consistent information model with sufficient relevant details about data and behaviour.

There are ambiguities about the status of some statements and our language's strength in clarifying them. For instance, in several policies (laptop, IT usage) there are sentences related to the fact that a person will sign and accept a policy. If we consider it as a part of the accountable usage it will say that to not sign is a breach which is not sensible. Thus it should be part of the proper user behavior, it is free to sign, and if he signs he will accept the responsibilities included in the policy. As we have seen, this impacts the structure of the specification but also brings some difficult points regarding the interaction between quantifiers and modal operators.

The use of $\mathbb{FOTL}$ or linear temporal logic often simplifies the specification and there is only few references to precise dates or dense time in the policies. However, there are several subtleties in writing formulas with both quantifiers and temporal operators. $\mathbb{FOTL}$ is quite expressive but the bottleneck is the tool support, there is only one, TSPASS, which is now not maintained. Another problem is that it relies on the monodic constraint which is constraining for the behaviour specification. Improvements are possible here but need important theoretical and implementation efforts. Another point is the use of the standard semantics based on infinite traces which is not suitable for real monitoring. The alternative is a translation into pure $\mathbb{FOL}$, the drawback is the explicit management of time parameters. These additional parameters may compromise the decidability of satisfiability but this logic has been intensively studied and a complete map of the decidable fragments exists. An adaptation of our language and its tool support is perfectly possible and it seems a sensible future perspective.

References

[BGRS15] Walid Benghabrit, Hervé Grall, Jean-Claude Royer, and Mohamed Sellami. Abstract accountability language: Translation, compliance and application. In *APSEC*, New Delhi, India, 2015. IEEE Computer Society.

[BMB10] Moritz Y. Becker, Alexander Malkis, and Laurent Bussard. A practical generic privacy language. volume 6503 of *ICISS 2010*, pages 125–139. Springer, 2010.

[Fis11] Michael Fisher. *An Introduction to Practical Formal Methods using Temporal Logic.* Wiley, 2011.

[LH10] Michel Ludwig and Ullrich Hustadt. Implementing a fair monodic temporal logic prover. *AI Commun*, 23(2-3):69–96, 2010.

[Mul00] Richard Mulgan. 'accountability': An ever-expanding concept? *Public Administration*, 78(3):555–573, 2000.

[RSDO16] Jean-Claude Royer and Anderson Santana De Oliveira. AAL and static conflict detection in policy. In *CANS, 15th International Conference on Cryptology and Network Security*, LNCS, pages 367–382. Springer, November 2016.

[Sch99] Andreas Schedler. *Self-Restraining State: Power and Accountability in New Democracies*, chapter Conceptualiazing Accountability, pages 13–28. Lynne Reiner, 1999.

[Uni17] Liverpool Hope University. IT services policies, 2017. https://www.hope.ac.uk/aboutus/itservices/policies/.

Requirements for a Temporal Logic of Daily Activities for Supportive Technology

Malte S. Kließ* and M. Birna van Riemsdijk*

Delft University of Technology
Delft, The Netherlands
m.s.kliess@tudelft.nl, m.b.vanriemsdijk@tudelft.nl

Abstract. Behaviour support technology is aimed at helping people organize their daily routines. The overall goal of our research is to develop generic techniques for representing people's actual and desired behavior, i.e. commitments towards themselves and others, and for reasoning about corresponding supportive actions to help them comply with these commitments as well as handle non-compliance appropriately. Describing daily behavior concerns representing the types of behaviour the user typically performs, but also when, i.e. we need to take into account *temporal dimensions* of daily behaviour. This paper forms a first requirements analysis of the types of temporal dimensions that are relevant for the purpose of supporting people's daily activities and how these may be formalized. This analysis forms the starting point for selecting or developing a formal temporal representation language for daily activities.

1 Introduction

Behaviour support technology [10] is aimed at helping people organize their daily routines and change their habits. The overall goal of our research is to develop generic techniques for representing and reasoning about people's actual and desired behavior for the purpose of providing support by means of technology [11, 15]. These expressions of desired behaviour can originate from users themselves or from others in their social context, such as caregivers. The idea underlying our approach is to model desired behaviour as norms [1] or commitments [12] regarding people's behaviour [9, 15, 11, 14]. Reasoning techniques are then aimed at deriving corresponding supportive actions to help people comply with these norms and commitments as well as handle non-compliance appropriately, i.e. with an understanding of the social relations and the values underlying the established agreements [11, 15, 8].

Describing (actual and desired) daily behavior concerns not only *which* types of behaviour the user typically performs, which can be represented in a behaviour hierarchy [11], but also *when* these activities are performed, i.e. we need to take into account temporal dimensions of daily behaviour. This paper forms a first requirements analysis of the types of temporal dimensions that are relevant for

* Supported by NWO Vidi project CoreSAEP.

the purpose of supporting people's daily activities. Once we have an understanding of which notions of temporality we want and need to capture for this type of technology, we can analyze which of many existing frameworks for representing and reasoning about norms and time we can build on, e.g. [6, 4, 2, 3, 7, 14]. This paper forms a preliminary exploration in this direction.

We present an illustrative scenario and preliminaries regarding temporal logic (Section 2). We identify key temporal dimensions of daily activities in Section 3. For each of these dimensions we provide an example formalization in temporal logic of an aspect of our scenario (Section 4). We conclude the paper in Section 5.

2 Scenario and technical preliminaries

2.1 Scenario

In this paper, we will explore examples involving temporal dimensions of daily activities based on the following scenario, where we intend to follow a few examples of daily activities in the fictitious life of Pedro, a young person with mental disabilities who can take care of most daily activities himself, but needs support in order to do them in a timely fashion.

Pedro can usually take care of everyday tasks himself, like getting ready in the morning, getting to work, going shopping and preparing meals. He needs support and regular reminders to schedule these activities. For this he has a support agent, which knows about Pedro's regular activities and preferences.

We intend to address the temporal dimensions of these routines. For example, when we describe our daily routines we would usually put these habits in some order in which we engage in these activities. We would certainly say that we get up before we wash ourselves, and that right next after that we have breakfast. However, there are a lot of subtleties involved when we want to formalize this description in a precise way so that an artificial agent is able to render efficient support for these activities: Is there a specific order in which we prepare the breakfast? Is it necessary to boil the water before we put in the tea bag, and when does this happen compared to preparing the bread we want to eat?

In order to deal with these kind of questions it is not enough for an agent to have an idea of the usual behaviour and habitual order. It also needs to ensure that actions are performed in a coherent way. For instance, putting the kettle on to boil water certainly needs to be done before we pour the water in a cup, but while we wait for the water to boil we can already toast the bread. Putting the kettle on before we go to the bathroom to wash, however, might not be a good idea: if we take too long in the bathroom the water will be too cold for tea, so we should start boiling the water only *after* we finished washing. This means that if we want an artificial agent to help users organize their day in an efficient way, then we need temporal reasoning with which not only the order in which activities are done can be stated, but also temporal 'closeness' to ensure that the user does not wait too long between finishing one step of an activity and starting to do the next step.

2.2 Temporal logic

We will formalize the logical framework in the following definition. We will roughly follow the definitions given in [11] for the HabInt habit support agent.

Definition 1. *Let $\mathcal{L}_{Str}$ be a propositional language over strings. Let $Act \subseteq \mathcal{L}_{Str}$ be the set of activities or actions. Let Part be a function $Act \to \mathcal{P}(Act)$, with $b \in Part(a)$ if the action b is a part*[1] *of doing action a. For the sake of readability we introduce predicates* `start`*,* `stop` *and* `doing` *on Act, signifying when an activity is started, stopped, or being done, but we will still treat Act as atomic. We close these atomic sentences as usual under $\neg$ and $\vee$, and interpret all other logical connectives in the usual way.*

As for the temporal dimensions, we will use notions from Linear Temporal Logic [5] as a way of sketching formalizations for the temporal dimensions discussed. In particular, we will have in mind trace semantics and the usual connectives $\Box$, $\Diamond$, $\mathcal{R}$, $\mathcal{U}$, $\bigcirc$ (to be read as *always*, *eventually*, *Release*, *Until* and *Next*, respectively). We will take this as a form of 'proof of concept', showing that some of the dimensions we have in mind may be expressed using notions of already existing frameworks.

3 Key temporal dimensions

In this section we will explore five temporal dimensions we believe are key to modelling human behaviour for supporting agents. We will provide examples demonstrating how each aspect covers part of an every-day behaviour pattern an agent should be able to support. Note that the examples are intentionally formulated very strictly: if a person states they are having dinner at 6 pm, then this should be formulated in the language just like that. Even though in reality, this will likely almost always be violated[2], it is important for the agent to recognize this. Whether an intervention is necessary, and if so, how the agent should intervene, should be delegated to the Norm Compliance Support system of the agent. Our intention is to provide the necessary formalization to state norms and thus give an agent the ability to detect deviance from these norms.

The five key dimensions are:

1. **Clocktime.** A supporting agent will no doubt have to be able to deal with clocktime. This is particularly important for keeping certain deadlines, as well as ensuring that scheduled actions are executed at the given time.
2. **Ordering.** Some behaviours require multiple distinct actions to be executed in a certain order, e.g. washing vegetables before cutting them. However, human behaviour is much more flexible than a deterministic routine; it is enough to know that vegetables should be cut *after* they have been washed,

[1] For example, brewing tea is an action in its own right, but may also be considered as part of 'preparing breakfast'.

[2] E.g., starting dinner five seconds after 6 pm.

but not necessarily directly after (partial order): it should be acceptable behaviour to wash all of the vegetables first, and then cut them.

3. **Coherence.** When describing behaviour with multiple parts it is not enough to state that they will all eventually be done. When cooking vegetables it is not good enough to wash them today, cut them tomorrow and finally cook them next week. Rather, these actions should be performed *coherently*, i.e. without too much delay. An immediate 'next' step, however, may be too rigid, so we need a notion that allows us to state that a set of actions form a coherent unit.
4. **Duration.** The duration of actions is a two-fold notion. On the one hand certain actions have a fixed or typical duration, e.g. 'pasta takes 8 minutes to cook'. On the other hand, an agent should be able to infer the duration of an action that is being performed, i.e. the actual duration of a specific instance of action execution, so that it can support the user in achieving the desired habitual goals. This becomes important, e.g. when supporting the user with keeping deadlines or making sure they do not exhaust themselves with certain exercises.
5. **Repetition.** The agent should be able to deal with repetition. This involves both regularly scheduled events and the cyclic nature of weeks and days. If a user wants to be supported having dinner at 6 pm every day, then this event regularly resets. The same is true for weekly exercises.

In the usual formalization, LTL semantics consists of a linear trace with one end point, namely the starting point of the trace. However, as one easily sees particularly in the clocktime example we have to deal with cyclic time: every night at 11.59 pm, when the time progresses one minute, the clock resets to 12.00 am, and a new day begins. Similarly, starting the week with Monday, the weekday variable changes each day until Sunday is reached, whereafter the variable resets to Monday again.

Linearity is not completely lost, though: we cannot repeat the actual day or week, but irrevocably progress into the future. The idea is, thus, to model this using two separate notions of time: *micro-time*, which is of cyclic nature and models the time passing each day, and *macro-time*, which is linear and models the fact that each day that has passed is certainly not coming back.

The way we suggest to model this is by using a two-dimensional trace index: we will write each separate point in the trace $\boldsymbol{s}$ as $s_{\langle a,b\rangle}$, where the first component refers to macro-time, and the second component refers to micro-time. We will use lexicographic ordering on the pairs $\langle a, b\rangle$: today at 5 pm is earlier than tomorrow 5 am.

While one can think of macro- and micro-time as separated into days passing and the time of the current day, respectively, this need not be the desired separation: there are larger cycles common in every-day life: weeks, months and years share a cyclic nature as well. It may depend on the specific task or use case how fine-grained one wants to make this separation. For the sake of this scenario, days and time of day are sufficient.

4 Scenario Formalization

In the following examples we will explore how the key dimensions identified above can be formalized in the framework outlined in Definition 1.

Example 1 (Clocktime).

Scenario: Pedro intends to have his dinner at 6 pm every day.

Formalization: Introduce a variable t ranging over time. Abusing notation, we will write $t = 6pm$ to mean that t has the value corresponding to real-world time of 6 pm. Then the statement above can be formalized as

$$\Box\Big(t = 6pm \rightarrow \texttt{start}(\mathit{HavingDinner})\Big).$$

Discussion: The statement above is very sharp: each day there is precisely one point in time when t has the value of 6 pm. So how do we then deal with situations when Pedro starts his dinner slightly earlier or later? We would argue here that this may be deferred to norm compliance: if we casually state 'I have my dinner each day at 6pm.', then we would argue this is meant precisely in the sense of the formal statement above.

Example 2 (Ordering).

Scenario: Pedro wants to bake a pizza for dinner. He already has a pizza dough prepared, and wants to decorate the pizza with tomato sauce, mushrooms and cheese.

Formalization: We want to express certain orders in the execution of an action, when that action is itself composed of smaller parts. As an example, take baking a pizza. For that we have to first prepare all ingredients, then put the toppings on the pizza, and finally put it in the oven. There is a strict ordering implicit here: we cannot[3] put the pizza in the oven and then afterwards put the (unprepared) toppings on the pizza dough, and finally prepare the toppings by washing and cutting them.

Take the LTL-formulation of *before*, as in e.g. [13], i.e. ϕ *before* $\psi \equiv \neg\,(\neg\phi\,\mathcal{U}\psi)$, where $\mathcal{U}$ is *Until*. Then the situation in the scenario above can be formalized by

$$\texttt{apply\_tomato\_sauce}\ \mathit{before}\ \texttt{decorate\_with\_mushrooms}, \tag{1}$$

$$\texttt{apply\_tomato\_sauce}\ \mathit{before}\ \texttt{decorate\_with\_cheese}, \tag{2}$$

$$\texttt{decorate\_pizza}\ \mathit{before}\ \texttt{bake\_pizza}. \tag{3}$$

Discussion: As an alternative formulation one could think of using *after*: ϕ *before* ψ should be (almost) the same as ψ *after* ϕ – both give a temporal link between ϕ and ψ, ϕ is the first statement that should come true, and ψ the second. Switching the roles of ϕ and ψ and negating the statement, however, will not turn *before* into *after*: we would be introducing the possibility of synchronicity, i.e. ϕ and ψ being executed simultaneously. Another problem is what to do once the second event, ψ, has occurred. From the definition using $\mathcal{U}$, this

[3] Or rather: should not attempt, as it gives undesired results.

does not prevent us from witnessing another ϕ later. While we certainly do not want to prevent Pedro from ever washing mushrooms again, the same mushroom should not be washed again once it has been cut.

Example 3 (Coherence).

Scenario: We will use the same scenario as for example 2, and focus on what it means to be 'coherent'.

Formalization: Using the example of preparing a pizza for dinner, what does it mean to 'coherently bake a pizza'? The dough has to be spread out, the ingredients put on it, and finally the pizza needs to be baked. While we also have the constraints that the toppings should first be washed, then cut, then put on the dough, we face the same issue of multiple instances as above: do we allow each mushroom to be washed, and then cut them all, or individually first cut, then wash, and repeat the process for each mushroom until we are done? We would argue here that this should not make a difference for the process of 'baking pizza', as long as we are not putting other activities in between, e.g. taking out the trash. So 'coherent' should mean that we do not engage in unrelated activities.

This seems to suggest an 'Axiom of Coherence':

$$(\text{Coh}) : \forall a \in \mathit{Act}.\ [\,\texttt{doing}(a) \rightarrow (\neg\texttt{wait}(a) \rightarrow \forall b \notin \mathit{Part}(a).\ \neg\texttt{start}(b))]\,.$$

This statement says that as long as we do not have some mandatory waiting time on our hand, e.g. when the pizza is in the oven, we should not start any other unrelated activity.

Discussion: 'Coherence' should also mean 'close together temporally', which is currently not encoded in the Axiom above. We can still have arbitrary idle time between washing the mushroom and putting it on the pizza. Setting a strict time limit for the distance between stopping some part of an activity and starting the next one may be too strict. In the pizza example, we may get some help via deadlines to solve this: if Pedro wants to have pizza for dinner, and have his dinner at 6 pm tonight, then this deadline already forces him to have shorter breaks. The question is then whether Coherence actually is a derived notion, and the Axiom stated above is implicitly given by the other temporal dimensions.

Example 4 (Duration).

Scenario: Pedro usually takes 20 minutes to prepare a pizza, and then puts it into the oven for 20 minutes. Thus the activity '*make_pizza*' has a duration of 40 minutes.

Formalization: We can equip any action a with an attributed duration $d \in \mathbb{N}$, given in minutes, to form the ordered pair $\langle a, d\rangle$. Assuming that no action can be done instantaneously, we allow $d = 0$ to indicate that no duration has been stated for the current action.

In the scenario above, we would then have $\langle \mathit{make\_pizza}, 40\rangle$ in the model, stating that preparing pizza has a duration of 40 minutes.

For the model, assume we are looking at traces[4] $\boldsymbol{s} = \langle s_0, s_1, \ldots, s_i, \ldots \rangle$, where s_i is the state of the model at minute i, and s_0 is the initial state. Then 'preparing pizza takes 40 minutes' would be modeled by

$$s_i \models \texttt{start}(\mathit{make\_pizza}) \Rightarrow \exists j \geq i + 40 \,.\, s_j \models \texttt{done}(\mathit{make\_pizza}).$$

Discussion: This example sees duration as an explicit notion present in the model. One could also see it as an implicit notion, where 'duration' is nothing but the difference of the two end-points of an activity, i.e. the difference between $\texttt{start}(a)$ and $\texttt{stop}(a)$. The explicit notion is needed for planning and norm compliance: if the agent knows that baking a pizza takes 40 minutes, then this needs to be started at least 40 minutes before the planned dinner time; similarly, the norm of 'having dinner at 6 pm' is certainly violated if Pedro starts preparing the pizza at 10 to 6. How does the implicit notion fit in here? Is it just used to update the explicitly given duration by experience, or as a monitoring tool to make sure the pizza stays in the oven for just the right time? Or does it have some importance in its own right, other than being checked against the recorded duration while doing an instance of an activity?

Example 5 (Repetition). Scenario: Pedro has a 'Pizza Day': every Monday he has pizza for dinner.

Formalization: Introduce a variable D for the weekday. Then we can formalize the above by:

$$\Box(D = \text{Monday} \ \rightarrow \texttt{pizza\_for\_dinner}).$$

Discussion: There seems to be nothing much needed for this except for variable types that allow access to the trace time. One could expand this example to 'pizza every other week', in which case one can then check against the past week. Letting W range over weeks, we then would obtain $(W = n \wedge \texttt{pizza\_for\_dinner}) \rightarrow (W = n + 1 \wedge \neg\texttt{pizza\_for\_dinner})$, and similarly swapping the negation in front of $\texttt{pizza\_for\_dinner}$ on both sides of the implication.

5 Discussion and future work

The examples above can all be expressed using LTL, using a two-dimensional index for the temporal trace; essentially, it is still discrete, linear, but it has the added benefit that we are now able to code daily routines using the second component of the trace index only. This suggests that LTL may be a suitable language for encoding temporal aspects of everyday activities. However, there are still many open questions such as those discussed in Section 4. In particular, we have not adressed the issue of expressing 'temporal closeness', or: 'how late is too late?'. A hard restriction on temporal distance between two activities that should be performed 'closely together' may be too strict for practical purposes. On the other hand, this may be an issue that can be dealt with by defining

[4] For the sake of simplicity, we omit the 'macro-time' component for the moment, looking at a one-dimensional trace.

norm compliance appropriately. Using a temporal language for reasoning about compliance support may require adaptations to the temporal language to make it tractable. As in [14] we use LTL without deontic operators. Whether this suffices for expressing norms regarding desired daily behaviour is also to be explored in future research.

Acknowledgements. We would like to thank two anonymous referees and the participants of the CARe-MAS workshop for insightful comments and discussions that helped improve the work presented in this paper.

References

1. G. Andrighetto, G. Governatori, P. Noriega, and L. van der Torre, editors. *Normative Multi-Agent Systems*, volume 4 of *Dagstuhl Follow-Ups*. Schloss Dagstuhl–Leibniz-Zentrum fuer Informatik, 2013.
2. J. Broersen, F. Dignum, V. Dignum, and J.-J. Ch. Meyer. Designing a deontic logic of deadlines. In *Proceedings Seventh International Workshop on Deontic Logic in Computer Science (DEON'04)*, volume 3065 of *LNCS*, pages 43–56. Springer-Verlag, 2004.
3. S. Cranefield. A rule language for modelling and monitoring social expectations in multi-agent systems. In *Coordination, Organizations, Institutions, and Norms in Multi-Agent Systems (ANIREM'05 and OOOP'05)*, volume 3913 of *LNCS*, pages 246–258, 2006.
4. F. Dignum and R. Kuiper. Specifying deadlines with dense time using deontic and temporal logic. *International Journal of Electronic Commerce*, 3(2):67–86, 1998.
5. E. Emerson. Temporal and modal logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume B: Formal Models and Semantics, pages 996–1072. Elsevier, Amsterdam, 1990.
6. G. Governatori, J. Hulstijn, R. Riveret, and A. Rotolo. Characterising deadlines in temporal modal defeasible logic. In *Proceedings of the 20th Australian joint conference on Advances in artificial intelligence*, pages 486–496. Springer-Verlag, 2007.
7. K. V. Hindriks and M. B. van Riemsdijk. A real-time semantics for norms with deadlines. In *Proceedings of the twelfth international joint conference on autonomous agents and multiagent systems (AAMAS'13)*, pages 507–514. IFAAMAS, 2013.
8. A. Kayal, W.-P. Brinkman, R. Gouman, M. A. Neerincx, and M. B. van Riemsdijk. A value-centric model to ground norms and requirements for epartners of children. In *Coordination, Organizations, Institutions, and Norms in Agent Systems IX (COIN'13)*, volume 8386 of *LNCS*, pages 329–345. Springer, 2014.
9. A. Kayal, W.-P. Brinkman, H. Zoon, M. A. Neerincx, and M. B. van Riemsdijk. A value-sensitive mobile social application for families and children. In *Posters, Demos, Late-breaking Results and Workshop Proceedings of the 22nd Conference on User Modeling, Adaptation, and Personalization (UMAP'14)*, volume 1181. CEUR, 2014.
10. H. Oinas-Kukkonen. Behavior change support systems: A research model and agenda. pages 4–14, 2010.

11. P. Pasotti, M. B. van Riemsdijk, and C. M. Jonker. Representing human habits: towards a habit support agent. In *Proceedings of the 10th International workshop on Normative Multiagent Systems (NorMAS'16)*, LNCS. Springer, 2016. To appear.
12. M. P. Singh. An ontology for commitments in multiagent systems: toward a unification of normative concepts. *Artificial Intelligence and Law*, 7(1):97–113, 1999.
13. M. B. van Riemsdijk, L. Dennis, M. Fisher, and K. V. Hindriks. Agent reasoning for norm compliance: a semantic approach. In *Proceedings of the twelfth international joint conference on autonomous agents and multiagent systems (AAMAS'13)*, pages 499–506. IFAAMAS, 2013.
14. M. B. van Riemsdijk, L. Dennis, M. Fisher, and K. V. Hindriks. A semantic framework for socially adaptive agents: Towards strong norm compliance. In *Proceedings of the fourteenth international joint conference on autonomous agents and multiagent systems (AAMAS'15)*. IFAAMAS, 2015.
15. M. B. van Riemsdijk, C. M. Jonker, and V. Lesser. Creating socially adaptive electronic partners: Interaction, reasoning and ethical challenges. In *Proceedings of the fourteenth international joint conference on autonomous agents and multiagent systems (AAMAS'15)*, pages 1201–1206. IFAAMAS, 2015.

Open data for accountability in the fight against corruption

Joris Hulstijn[1] Darusalam Darusalam[2] Marijn Janssen[2]

[1] Tilburg School of Economics and Management, Tilburg University
[2] Department of Technology, Policy and Management, Delft University of Technology
j.hulstijn@uvt.nl, d.darusalam@tudelft.nl, M.F.W.H.A.Janssen@tudelft.nl

Abstract. There has been a lot of research on the use of open data in the fight against corruption. Although there are some promising examples, it appears that a systematic approach is lacking. What are the design principles for an architecture to open up data and thereby reduce corruption? In this paper we use theory about fraud, and about public accountability to derive design principles for an open data architecture. Crucial is the sustained presence of a specific forum: a group of people who are critical, have expertise, are free to challenge the authorities. Unlike the general public, a specific forum has an interest in reviewing the data. The architecture is motivated and illustrated by an extensive example of an E-procurement system in the context of an anticorruption program in Palembang, Indonesia.

1 Introduction

There has been a lot of attention for 'opening up' government data in the field of e-government [28]. One particular application area of open data is the fight against corruption. Corruption is a complex and difficult problem, also in government [12]. Corruption can harm society and result in increased poverty, reduce available money for essential public services, destroy citizens' trust in government and undermine economic growth [26]. There are many factors that can lead to corruption in government [21]. In principle, addressing these different factors will lead to different approaches in the fight against corruption. In this paper, we will focus on the factors of transparency and accountability. Therefore we investigate a particular strategy to fight corruption: to open up government data to the public [25]. One of the claimed benefits of open data is that the government becomes more transparent and accountable [15]. This appears intuitive. Open data allows scrutiny by the public, which will reduce opportunities for corruption and increase changes of detection. Indeed, there are successful examples of e-government systems to reduce corruption, see [16] and [2] (p 265-266). There have

also been examples of system to open up government data to reduce corruption, for example in Brazil. A recent study by Transparency International [14] shows that a lot has been achieved in Brazil. "Nonetheless, open data is still underutilized, and anti-corruption enforcement is weak" [14](p 4). In particular, there are problems with data quality. Moreover, the initiatives lack involvement: when there is a 'lead' there is not always a follow up. Apparently, the relationship between open data, transparency and accountability and eventually reducing corruption is not immediate, but rests on specific assumptions. For instance, in the context of corporate governance, Elia argues that opening up too much data, in too much detail, is pointless or even counterproductive [11]. Details of illicit behavior may be drowned out.

In this discussion paper we therefore propose design principles for an open data architecture that should foster accountability, and in turn, reduce opportunities for corruption. We will take a theoretical approach, based on an analogy with fraud [1, 9], and theory of public accountability by Bovens [3]. Following Day and Klein [10], Bovens characterizes accountability as a dialogue between an actor and a 'significant other', called the forum. The actor feels obliged to provide accounts of its conduct to the forum. In order to hold the actor accountable, the forum should be critical and powerful enough to ask for explanations, and eventually pass judgement about the conduct of the actor.

To motivate the approach and validate the design principles, we discuss a case study of an e-procurement system at the local government of Palembang in Indonesia.

The remainder of the paper is structured as follows. Section 2 provides a background on corruption and how it can be reduced. Section explain the idea of accountability as a dialogue, and derives some general principles about the actor and the forum. Section 3 applies these principles to an architecture for opening up data. Section 4, finally, discusses application of these ideas in the Palembang case study.

2 Corruption and Fraud

In this section we provide a background of the notion of corruption in government and how it can be reduced. Corruption is an important topic that deserves a proper literature review. In this discussion paper, we cannot provide a full review. Instead, we will focus on two elements.

First, we will draw an analogy with fraud, and derive three ways to address corruption. Second, we will discuss anti-corruption strategies.

The word 'corruption' is based on Latin: *corrumpere* means to spoil or destroy. Corruption is defined as "behavior which deviates from the normal duties of a public role because of private-regarding (family, close private clique) pecuniary or status gains" [21] (p. 4). An additional aspect of corruption in the case of government, is the misuse of power [12]. So similar to the notion of fraud, which is defined as a violation of trust [9], corruption involves the misuse of responsibilities related to a public role or duty. For instance, corruption involves "bribery (use of reward to pervert the judgments of a person in a position of trust), nepotism (bestowal of patronage because of ascriptive relationship rather than merit), and misappropriation (illegal appropriation of public resources for private-regarding uses" [21] (p. 4).

Figure 1. Fraud triangle [1, 9]

There is an interesting analogy between corruption and the notion of fraud as studied in accounting [1] and criminology [9]. In Figure 1, fraud is depicted as a triangle, involving three necessary elements.

(1) Opportunity: the actor has been trusted with responsibilities that involve certain powers (budget; decision rights),
(2) Pressure: the actor has a 'non-shareable problem' (e.g. financial problems, status), and misuse of trust can offer a 'private solution'
(3) Rationalization: the actor provides a kind of explanation for crossing the line: "I am only borrowing the money" or "others do it too".

The triangle suggests three ways of reducing fraud or corruption. These are effectively means to improve management control [19] and the system of internal controls [8], in particular the control environment.

First, one could try to reduce opportunities for corruption. That means limiting the powers of an actor in a certain role, by putting administrative restrictions in the procedures or by recording details of execution (audit trail). This is the easiest element to adjust. In modern organizations, many institutional powers are controlled by ICT, such as the ERP system

or business process management system. In accounting, there is a lot of expertise on ways to strengthen internal controls, partly by means of ICT. In particular, there are so called application controls, which are built into software applications. For instance, there are authorization tables, restrictions on the budget a person can control, on tariffs and pricing policies, etc. In addition, there are also many manual controls, that are supported by technological means. For instance, if all invoices are numbered consecutively, it is harder to make one disappear. Finally, there are the IT general controls, that help to maintain a reliable control environment. In particular, consider such measures as segregation of duties, audit trail, access controls, logging and monitoring [23]

Second, one could try to reduce pressure. In the financial sector there has been a lot of debate about remuneration. If you pay people enough, is the argument, there should be less incentives to misuse trust for personal gains. However, the evidence as to the effectiveness of this measure is inconclusive, at best [24]. Another option to reduce pressure is suggested by Cressey's original paper [9]: foster an environment in which people feel safe to discuss problems. In this way, when people do have a problem (gambling debts, dissatisfaction), they are less likely to see misuse of trust as a way out. For example, one could implement a regular meeting structure, in which officials have an opportunity to discuss – in secret – difficult decisions with peers.

Third, one could try to raise moral standards and make it harder for people to 'rationalize' misuse of trust. This is perhaps the most effective but also the hardest option. It involves elements of organizational culture, such as public values, social practices and role models [13, 22]. Organizational culture can't be managed directly. Instead, one can only try and improve side conditions. The 'tone at the top' makes an enormous difference. For example, in the Enron case, efforts to improve compliance were useless, as executives themselves were seen to violate the rules [24]. Public values such as quality awareness, safety or integrity, can be stimulated. For example, to stimulate quality awareness, one could routinely publish examples of products that are rejected. This will demonstrate the borderline. Practices and procedures may help to reach consensus. If it is mandatory to rate and discuss quality of a product with colleagues, this will help establish a common threshold.

We will now discuss an example from the literature, to see how these approaches to fraud are applied.

Example. Kim [16] reports about the OPEN system, as part of a long-term effort in Korea to fight corruption (1998-2007). The OPEN system "is an online system used to disclose administrative procedures (likely to be related to corruption) to citizens in various public service areas (such as housing and construction, sanitation, and urban planning, among others)." [16] (p 45). The purpose was to improve efficiency, by preventing unnecessary delays or unfair handling by civil servants, and also to improve transparency in some areas of civil administration.

First, such a system is likely to reduce opportunities for corruption. By allowing citizens to view the procedures and status of their requests, an audit trail is created. Decisions can be compared. Moreover, the system was supported by projects to clarify the procedures and design systems that simplify, standardize, and de-personalize delivery of services [16].

Second, we do not know if the system helped to reduce pressure. We do know, that the system reduced personal contact between applicants and officials [16], thus reducing opportunities to be put under pressure.

Third, the 'tone at the top' proved decisive. The system was supported by Kun Koh, mayor of the city of Seoul. "Mayor Koh took resolute steps to fight corruption. He launched a variety of reform programs which culminated in the declaration of a 'War against Corruption'" [16] (p 46). By doing this, Koy set values such 'clean' and 'transparent' as norms that no one could deny.

Strategies to Combat Corruption

Activities to reduce corruption are often presented as phases in an anti-corruption cycle: prevention, detection, investigation and sanction [20].

Alternatively, the literature mentions four strategies to combat corruption: prevention, enforcement, access to information and empowerment, and capacity building [16]. Unlike the stages, which focus on officials, these strategies also assign an important role to citizens. They indicate how ICT or open data can help: by providing access to information and empowering citizens, and by helping to organize and improve expertise, i.e. capacity building. In the next section, we will try to make these ideas about empowerment more systematic, by relating them to a theory of public accountability.

3 Accountability as a dialogue

Accountability has positive connotations, and is used as a kind of synonym to good governance. However, the meaning of the term depends on the context, on a relationship with others: "Accountability can be defined as a social relationship in which an actor feels an obligation to explain and to justify his or her conduct to some significant other" [10]. According to Bovens [3, 4] an accountability relationship involves an actor and a forum (Figure 2). The actor can be a person or agency. Then there is some significant other, who can be a person or agency, but can also be a more abstract entity, such as God or 'the public'. Bovens calls this the forum.

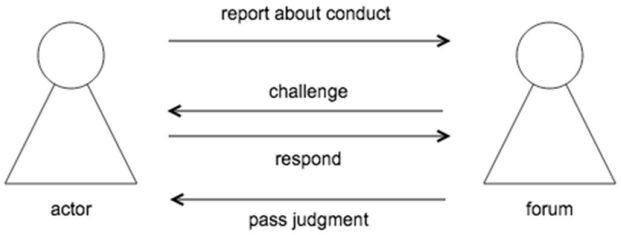

Figure 2. Accountability as a dialogue

The accountability relationship develops in three stages. First, the actor must feel obliged to inform the forum about its conduct, including justifications in case of failure. The obligation may be both formal, i.e., required by law or by contract, or informal and self-imposed, for instance because the actor is dependent on the forum. Second, the information may be reason for the forum to challenge the actor, ask for explanations and debate the adequacy of the conduct. The actor responds, providing additional facts or motivations. Third, the forum passes judgement on the actor's conduct. A negative judgment often leads to some kind of sanction; again this can be both formal and informal. This means that an accountability relation should provide room for discussion: it is not a one-way stream of reports, but rather a dialogue. The process can be formalized as a dialogue game [5].

Bovens [3] identifies five necessary conditions of public accountability, which characterize the dialogue game just described:

(1) "*public accessibility* of the account giving—and not purely internal, discrete informing;

(2) *explanation and justification* of conduct—and not propaganda, or the provision of information or instructions to the general public;
(3) the explanation should be *directed at a specific forum*—and not be given at random;
(4) the *actor* must feel *obliged* to come forward—instead of being at liberty to provide any account whatsoever; and
(5) there must be a possibility for *debate and judgment*, including an optional imposition of (informal) sanctions, by the forum—and not a monologue without engagement" [3] (p 185); italics added.

The prototypical example of an accountability relation exists between government and parliament, where parliament derives its power from the population, and can send the government home (confidence vote). Journalists play an important role, demanding transparency and challenging the government. By informing the public, they can mobilize the underlying power of parliament.

Another example of an accountability relation is auditing [6, 5]. In that case, management is the actor, reporting the annual financial accounts. The shareholders act as a forum. The forum is assisted by an auditor, who has the expertise and authority to evaluate the financial accounts, compare them to standards and norms and provide assurance about their reliability [17]. However, the auditor as such has little power. The power derives from the shareholders, or other powerful stakeholders, such as the board of directors or regulators.

Based on these examples, we derive four additional necessary conditions for a fruitful accountability relationship, compare also [6] .

(6) The organization of the actor must provide *reliable reports* about its conduct. Reports must be accurate (correspond to reality) and complete (contain all relevant aspects of reality). In practice this can be ascertained by internal controls.
(7) The forum must be *independent* of the actor, allowing it to challenge the actor, as stated in (5) above. In particular, the forum should not depend on the actor for employment. Individual members should not have family ties, or personal relationships.
(8) The forum must have, or have access to, *expertise*, to evaluate the reports, compare them to standards and norms, and eventually pass judgement. If necessary, the forum can be supported by experts or auditors, who possess specific expertise. That means that those experts must also be independent.

(9) The forum must wield enough *power* or influence over the actor to "make the actor feel obliged to come forward" as in (4), and to have a credible claim to sanctions, as in (5). In particular, when the actor is dependent on the forum in some way, the forum has the power to withhold whatever the actor depends upon.

4 An open data architecture for accountability

So how can we apply these ideas to an open data architecture?

An architecture is a set of fundamental concepts or properties of a system in its environment, embodied in its elements, relationships, and in the principles of its design and evolution [18] (p 2).

What are the components, what are their relationships, and what are the design principles? If we want to conform to conditions (1) – (9) above, we have to include some form of governance. So we will treat architecture here in a broad sense, including also the organizational structure. An open data architecture is a socio-technical system [7].

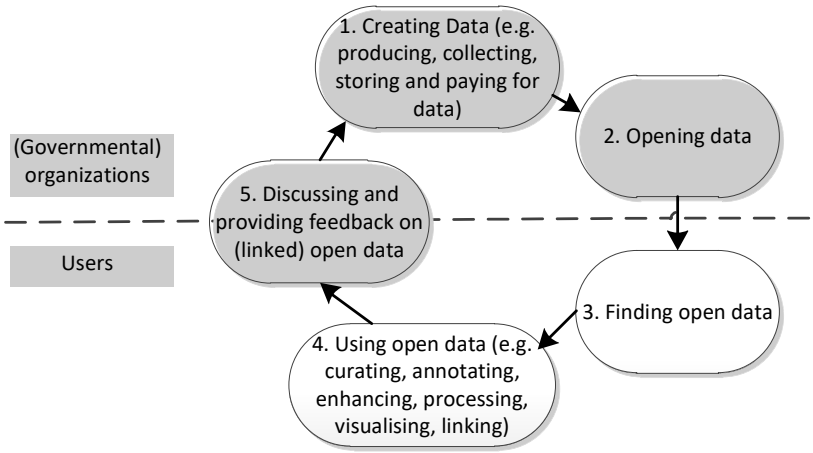

Figure 1. Conceptualizing open data [27] (p 157)

At first sight, an open data architecture seems to be just a website to publish data in a particular format. The website makes the data available to the forum (D0. Availability). Crucial elements are the legislation, policies and business rules that determine which data eventually gets

published (D1. Data selection). Published data should be relevant to the issues addressed by the application, and later to the challenges or inquiries made. Data should be fit to be published. In particular, state secrets, data that are recognized to be confidential (e.g. medical; military) or data about individual persons should not be published. This means that data selection should involve a careful trade-off between transparency and confidentiality. Ideally, the forum should have an advisory role on setting the roles and policies for data selection. Moreover, data sets should adhere to representation standards, that improve interoperability between various systems, and allow for comparison (D2. Representation standards). Data sets are filled with real data by interfaces with day-to-day operational systems, such as databases and ERP systems (D3. Content). A coherent system of internal controls should help guarantee reliability of the data (accuracy and completeness) (D4. Reliability). Note that completeness is hard to ensure, as it involves comparison to what is considered to be 'all'. Once selected for publication, officials should not be able to delete data or obstruct publication. Finally, the architecture should have features to respond to challenges and inquiries with data that are relevant (D5. Relevance).

Because accountability involves a relationship, the architecture should not only focus on what governments have to do in order to reliably 'open up' data sets, but also on the forum. That means in particular that the architecture should have components for the forum to gather, and mobilize support (F0. Mobilize). Consider for example outlets on social media. Clearly, mobilizing a forum is not the responsibility of the government! The government should collaborate with those institutions that happen to represent the forum, such as pressure groups, branch organizations, watchdogs, regulators, parliaments, etc. Given such collaboration, the platform should allow all members of the forum easy access (F1. Access). The platform should have components for the forum, or experts and auditors, to analyze the data (F2. Analysis). The architecture should have components to pose challenges and questions, about specific data elements and get a relevant proper response (F3. Challenge and response), see also D5. Responses should be archived, and should in principle also be made public (F4. Archive). In some cases, absence of a response would also be telling. Finally, the architecture should make it possible to escalate and file complaints, in case of a suspected deviation from the policies (F5. Complaints

procedure). In addition, the architecture should have specific support for journalists, experts or auditors (F1 and F2).

The list of requirements is summarized in Table 1.

Data publishing requirements	**Forum requirements**
D0. Availability	F0. Mobilize
D1. Data selection policies	F1. Access
D2. Representation standard	F2. Analysis
D3. Content	F3. Challenge and response
D4. Reliability	F4. Archive
D5. Relevant	F5. Complaints procedure

Table 1. Requirements for an open data architecture, focusing both on publishing open data (D) and on the role of the forum (F)

5. Case: E-procurement in Palembang

Data Collection Data for this case study was collected by investigating various sources, including the official website, official documents, prior research and the participating observations of one of the authors.

Case Description. The local government in Palembang South Sumatera Indonesia uses the Systems Electronic Procurement Service (LPSE) for the procurement of government goods and services. The LPSE system has been introduced to improve the efficiency, effectiveness, quality, and transparency in the procurement of goods and services. The system matches vendors and governments. LPSE is hosted by a unit formed across ministries and other institutions. This system facilitates procurement officers and also provides services for the provider of goods and services, in the territory of the LPSE concerned.

The procurement process starts by defining the needs, which is followed by publishing the Request for Quotation (RFQ). Next, suppliers receive the RFQ and develop their quotes. The supplier sends the quotes to government, government who receives the proposals. Once the proposals are received and the deadline has passed, the proposals are evaluated and supplier(s) will be selected and a final tender will be requested and the contract is signed. Thereafter follows the execution of the contract in which the products and services are delivered and paid. Finally, the delivered products and services can be evaluated.

According to local government regulation, all working units of local government must use the LPSE system. This should prevent bypassing and the risks of fraud. All the working units are obliged to announce their planning, implementation and results of their procurement processes via the LPSE system. As such, there is a huge potential for opening data. The types of data available in the LPSE system are as follows:

- *Auction announcement*: The LPSE system provides an announcement about what types of procurement are available from working units in Palembang*;*
- *Information about system failure:* The system provides information in case a package cannot be generated, or if a file failed to upload, so the system can provide solutions;
- *Electronic Catalogue (EC)*: This catalogue provides a detailed listing of vendor offerings. For example, description of products, prices, delivery times.
- *Monitoring and online evaluation****:*** The system provides information about planning packages of procurements, financial progress, physical progress, procurement of goods and services progress;
- *Whistleblowing:* The LPSE system has a link to https://wbs.lkpp.go.id/container.php. This website provides an opportunity for a person who has information about illegal, unethical or corruption related behavior related to procurement. The person can report these activities to the corruption watch or audit board.

In the current system, all this data is not yet open to the public. In the next section we present patterns for opening data.

Case Problem. How could opening up such data, help to reduce corruption?

Case Solution. In earlier research on the case, we found six types of patterns for corruption detection, which are presented in Table 2. These patterns are all based on internal control measures [23].

Evaluation. The patterns show that not only the resulting data should be opened, but also information about the operation of the administrative processes and the implemented internal control measures to prevent corruption should be opened. However, these internal control principles are not commonly known. Accounting expertise is needed to evaluate

which segregations of duties, ought to be met. This puts additional requirements on the kind of audience that the Palembang authorities are trying to reach. Without proper guidance, providing such meta-data may distract users from viewing the actual procurements.

Pattern name	Description	Categorization
1. Storing and opening documentation	Opening of documents generated in several activities	Process data
2. Cross-data comparison	Comparison of data collected in different phase to detect discrepancies	Process data
3. Four-eyes-principle	Process data, should demonstrate that decisions are made by at least two independent persons	Control data
4. Segregations of Duties	Process data, should demonstrate that a single individual or department is not allowed to process a transaction in its entirety	Control data
5. Authorization	Process data, should demonstrate that only people who are authorized to approve an activity, have done so	Control data
6. Application controls	Opening of data about built in measures to avoid the making of mistakes and the availability of alerts in the system	Control data

Table 2. Opening data, based on patterns to detect corruption

A possible 'forum' for this case is the pool of potential vendors. It is likely that competitors of the vendor who received the procurement contract, have enough interest to actually analyze the data, and challenge the local government in case of irregularities. Collectively, the pool of vendors is large enough to wield some power or influence over the local government, as they form a substantial portion of the voting population.

The most promising pattern seems to be number 2: cross data comparison. This may help to identify whether awarded contracts actually conform to the stated contract requirements, and whether there are any oddities in awarding behavior.

6 Conclusions

Opening up government data to the public, seems a good strategy to try and fight corruption. The claim is that opening up government data will somehow improve transparency and accountability. Current examples of systems that attempt to open up data in the fight against corruption, therefore make an ad hoc impression. In this paper, we have tried to provide some theoretical underpinning to these attempts, in two ways.

First, we have used an analogy with fraud, to identify three ways of reducing fraud or corruption: (1) by reducing opportunities for misuse of trust, essentially by strengthening the internal controls, (2) by fostering an environment in which people talk about their problems, before they become insurmountable, and (3) by stimulating an organizational culture, in which normative decisions are being discussed and recorded, establishing consensus on norms and values.

Second, we have used the theory of public accountability of Bovens, with five necessary conditions for a fruitful accountability relationship. Based on an analogy with accounting, we derived another four necessary conditions: reliable reporting (for the actor), and independence, expertise and power (for the forum). These conditions can be translated into a set of five requirements for an architecture and governance procedure for opening up data (Availability, Data selection policies, Representation standard, Content, Reliability, Relevance) and five requirements focusing on establishing a critical forum (Mobilize, Access, Analysis, Challenge and response, Archive and Complaints procedure).

Some of these requirements were discussed in the context of a case study of an e-procurements system in Palembang, that makes it possible to open up crucial data about the procurement process. The case shows that indeed it is crucial to identify a specific forum. This is contrary to current practice in the open data community, where data is being opened up for a general public, in the hope they will find some purpose for it (e.g. hackatons). Lack of a specific audience and purpose for the data being shared, means in particular, that reliability and relevance are hard to establish. This may partly explain the current problems with data quality that are reported for many open data initiatives, e.g. [28].

In the Palembang e-Procurement case, it is likely that the forum will include vendors, in particular the competitors of those vendors who received the contract, because they have an incentive to protect their interests. The case also shows, that it makes sense to open up meta-data,

to reveal if the internal controls that ensure reliable reporting, are effective. However, this will require the forum to involve experts, such as accountants, who can interpret and evaluate such data.

Future research will have to show whether also the dialogue-related aspects of the theory are valid in this case.

References

1. Albrecht, S. and M. Romney, *Red Flagging Management Fraud: A Validation.* Advances in Accounting, 1986. **3**: 323-333.
2. Bertot, J.C., P.T. Jaeger, and J.M. Grimes, *Using ICTs to create a culture of transparency: E-government and social media as openness and anti-corruption tools for societies.* Government information quarterly, 2010. **27**(3): 264-271.
3. Bovens, M., *Public Accountability*, in *The Oxford Handbook of Public Management*, ferlie, Editor. 2005: Oxford, United Kingdom.
4. Bovens, M., *Analysing and Assessing Accountability: A Conceptual Framework.* European Law Journal, 2007. **13**(4): 447-468.
5. Burgemeestre, B., J. Hulstijn, and Y.-H. Tan, *Value-based Argumentation for Justifying Compliance.* Artificial Intelligence and Law, 2011. **19**(2-3): 149-186.
6. Burgemeestre, B. and J. Hulstijn, *Design for the Values of Accountability and Transparency*, in *Handbook of Ethics, Values, and Technological Design*, J. van den Hoven, P. Vermaas, and I. van de Poel, Editors. 2015, Springer Verlag: Berlin. 303 -- 333.
7. Clegg, C.W., *Sociotechnical principles for system design.* Applied Ergonomics, 2000. **31**: 463-477.
8. COSO, *Internal Control - Integrated Framework*. 1992, Committee of Sponsoring Organizations of the Treadway Commission.
9. Cressey, D.R., *The Criminal Violation of Financial Trust.* American Sociological Review, 1950. **15**(6): 738-743.
10. Day, P. and R. Klein, *Accountabilities: Five Public Services*. 1987, London: Tavistock.
11. Elia, J., *Transparency rights, technology, and trust.* Ethics and Information Technology, 2009. **11**(2): 145–153.
12. Heidenheimer, A.J., M. Johnston, and V.T. LeVine, *Political corruption.* New York: Holt, Rinehart & Winston, 1970. **24**: 26-27.
13. Hofstede, G., et al., *Measuring Organizational Cultures: A Qualitative and Quantitative Study.* Administrative Science Quarterly, 1990. **35**(2): 286-316.
14. Iglesias, D., *Open Data and the Fight Against Corruption in Brazil*. 2017, Transparency International: Berlin.
15. Janssen, M. and A. Zuiderwijk. *Open data and transformational government*. in *Transforming Government Workshop. Brunel University, United Kingdom*. 2012.
16. Kim, S., H.J. Kim, and H. Lee, *An institutional analysis of an e-government system for anti-corruption: The case of OPEN.* Government Information Quarterly, 2009. **26**: 42-50.
17. Knechel, W., S. Salterio, and B. Ballou, *Auditing: Assurance and Risk*. 3 ed. 2007: Thomson Learning, Cincinatti.
18. Lankhorst, M., ed. *Enterprise Architecture at Work: Modelling, Communication and Analysis*. 2009, Springer Verlag: Berlin.
19. Merchant, K.A., *Modern Management Control Systems, text & cases*. 1998, New Jersey: Prentice Hall.

20. Newburn, T. and B. Webb, *Understanding and preventing police corruption: lessons from the literature*. 1999: Home Office London.
21. Nye, J.S., *Corruption and political development: A cost-benefit analysis.* American political science review, 1967. **61**(02): 417-427.
22. Pettigrew, A.M., *On studying organizational cultures.* Administrative Science Quarterly, 1979. **2**(4): 570-581.
23. Romney, M.B. and P.J. Steinbart, *Accounting Information Systems* 13th ed. 2015: Pearson Education.
24. Soltani, B., *The Anatomy of Corporate Fraud: A Comparative Analysis of High Profile American and European Corporate Scandals.* Journal of Business Ethics, 2014. **120**(2): 251–274.
25. Sousa, L.d., *Open government and the use of ICT to reduce corruption risks.* PPT, 2016.
26. UNDP, T.C., *Transforming Lives: Accelerating Human Development in the Asia and the Pacific*. 2008, New Delhi: Macmillan.
27. Zuiderwijk, A., et al., *Socio-technical Impediments of Open Data.* Electronic Journal of e-Government, 2012. **10**(2): 156 - 172.
28. Zuiderwijk, A. and M. Janssen, *Open data policies, their implementation and impact: A comparison framework.* Government Information Quarterly, 2014. **31**(1): 17-29.

Classifying the Autonomy and Morality of Artificial Agents

Sjur Dyrkolbotn[1], Truls Pedersen[2], and Marija Slavkovik[2]

[1] Høgskulen på Vestlandet sdy@hvl.no
[2] Universitetet i Bergen, {truls.pedersen, marija.slavkovik}@uib.no

Abstract. As we outsource more of our decisions and activities to machines with various degrees of autonomy, the question of clarifying the moral and legal status of their autonomous behaviour arises. There is also an ongoing discussion on whether artificial agents can ever be liable for their actions or become moral agents. Both in law and ethics, the concept of liability is tightly connected with the concept of ability. But as we work to develop moral machines, we also push the boundaries of existing categories of ethical competency and autonomy. This makes the question of responsibility particularly difficult. Although new classification schemes for ethical behaviour and autonomy have been discussed, these need to be worked out in far more detail. Here we address some issues with existing proposals, highlighting especially the link between ethical competency and autonomy, and the problem of anchoring classifications in an operational understanding of what we mean by a moral theory.

1 Introduction

We progressively outsource more and more of our decision-making problems to artificial intelligent agents such as unmanned vehicles, intelligent assisted living machines, news aggregation agents, dynamic pricing agents, stock-trading agents. With this transfer of decision-making also comes a transfer of power. The decisions made by these artificial agents will impact us both as individuals and as a society. With power to impact lives, there comes the natural requirement that artificial agents should respect and follow the norms of society. To this end, the field of *machine ethics* is being developed.

Machine ethics, also know as *artificial morality*, is concerned with the problem of enabling artificial agents with ethical[3] behaviour [3]. It remains an open debate whether an artificial agent can ever be a moral agent [11]. What is clear is that as artificial agents become part of our society, we will need to formulate new ethical and legal principles regarding their behaviour. This is already

[3] An alternative terminology is to speak of *moral* agency, as in the term "moral machines". However, since many philosophers regard morality as a reflection of moral personhood, we prefer to speak of "ethical" agency here, to stress that we are referring to a special kind of rule-guided behaviour, not the (distant) prospect of full moral personhood for machines.

witnessed by increased interest in developing regulations for the operation of automated systems *e.g.,* [6, 7]. In practice, it is clear that some expectations of ethical behaviour have to be met in order for artificial agents to be successfully integrated in society.

In law, there is a long tradition of establishing different categories of legal persons depending on various characteristics of such persons. Children have always had a different legal status than adults; slaves belonged to a different category than their owners; men did not have the same rights and duties as women (and vice versa); and a barbarian was not the same kind of legal person as a Roman.

Today, many legal distinctions traditionally made among adult humans have disappeared, partly due to a new principle of non-discrimination, quite unheard of historically speaking. However, some distinctions between humans are still made, e.g., the distinction between adult and child, and the distinction between someone of sound mind and someone with a severe mental illness.

Artificial agents are not like humans, they are tools. Hence, the idea of categorising them and discriminating between them for the purposes of ethical and legal reasoning seems unproblematic. In fact, we believe it is necessary to discriminate, in order to ensure congruence between the rules we put in place and the technology that they are meant to regulate. We need to manage expectations, to ensure that our approach to artificial agents in ethics and law reflects the actual capabilities of these tools.

This raises a classification problem: how do we group artificial agents together based on their capabilities, so that we can differentiate between different kinds of agents when reasoning about them in law and ethics? In the following, we address this issue, focusing on how to relate the degree of autonomy with the ethical competency of an artificial agent. These two metrics are both very important; in order to formulate reasonable expectations, we should know how autonomous an agent is, and how capable it is at acting ethically. Our core argument is that current approaches to measuring autonomy and ethical competence need to be refined in a way that acknowledges the link between autonomy and ethical competency.

When considering how ethical behaviour can be engineered, Wallach and Allen [19, Chapter 2] sketch a path for using current technology to develop artificial moral agents. They use the concept "sensitivity to values" to avoid the philosophical challenge of defining precisely what counts as agency and what counts as an ethical theory. Furthermore, they recognise a range of ethical "abilities" starting with *operational morality* at one end of the spectrum, going via *functional morality* to *responsible moral agency* at the other. They argue that the development of an artificial moral agents requires coordinated development of autonomy and sensitivity to values. We take this idea further by proposing that we should actively seek to classify agents in terms of how their autonomy and their ethical competency is coordinated.

There are three possible paths we could taken when attempting to implement the idea of relating autonomy to ethical competency. Firstly, we could ask computer science to deliver more detailed classifications. This would lead to

technology-specific metrics, whereby computer scientist attempt to describe in further detail how different kinds of artificial intelligence can be said to behave autonomously and ethically. The challenge would be to make such explanations intelligible to regulators, lawyers and other non-experts, in a way that bridges the gap between computer science, law and ethics.

Secondly, we could ask regulators to come up with more fine-grained classifications. This would probably lead to a taxonomy that starts by categorising artificial agents in terms of their purpose and intended use. The regulator would then be able to deliver more fine-grained definitions of autonomy and ethical behaviour for different classes of artificial agents, in terms of how they "normally" operate. The challenge would be to ensure that the resulting classifications make sense from a computer science perspective, so that our purpose-specific definitions of autonomy and ethical capability reflect technological realities.

Thirdly, it would be possible to make the classification an issue for adjudication, so that the status of different kinds of artificial agents can be made progressively more fine-grained through administrative practice and case law. The challenge then is to come up with suitable reasoning principles that adjudicators can use when assessing different types of artificial agents. Furthermore, this requires us to work with a pluralistic concept of what counts as a moral theory, allowing substantive moral and legal judgements about machine behaviour to be made in concrete cases, not in advance by the computer scientists or the philosophers. Specifically, there should be a difference between what counts as ethical competency – the ability to "understand" ethics – and what counts as objectively good behaviour in a given context.

In the following, we argue that a combination of the second and third option is the right way forward. While it is crucial that computer science continues working on the challenge of developing machines that behave ethically, it is equally important that the legal and ethical classifications we use to analyse such machines are independently justifiable in legal and ethical terms. For this reason, the input from computer science should be filtered through an adjudicatory process, where the role of the computer scientist is to serve as an expert witness, not to usurp the role of the regulator and the judge. To do this, we need reliable categories for reasoning about the ability of machines, which keep separate the question of ability from the question of goodness.

This paper is structured as follows. We begin with discussing the possible moral agency of autonomous systems. In Section 2, we introduce the best known, and to our knowledge only, classifications of moral agency for non-human agents proposed by Moor in [15]. In Section 3, we then discuss the shortcomings of this classification and the relevance of considering autonomy together with moral ability when considering machine ethics. In Section 4, we discuss existing levels of autonomy for agents and machines, before pointing out some shortcomings and proposing an improved autonomy scale. In Section 5, we go back to Moor's classification and outline ways in which it can be made more precise. Related work is discussed continuously throughout the paper.

2 Levels of ethical agency

The origin of the idea that different kinds of ethical behaviour can be expected from different agents can be traced back to Moor [15]. Moor distinguishes four different types of ethical agents: ethical impact agents, implicit ethical agents, explicit ethical agents, and full ethical agents. We briefly give the definitions of these categories.

Ethical impact agents are agents who do not themselves have, within their operational parameters, the ability to commit unethical actions. However, the existence of the agents themselves in their environment has an ethical impact on society. There are many examples of an ethical impact agent. A search engine can be seen as an ethical impact agent. By ranking search results to a query it can promote one world view over another. The example that Moor gives in [15] is that of a robot camel jokey that replaced slave children in this task, thus ameliorating, if not removing, the practice of slavery for this purpose in the United Arab Emirates and Qatar.

Implicit ethical agents are agents whose actions are constrained, by their developer, in such a way that no unethical actions are possible. The agents themselves have no "understanding", under any interpretation of the concept, of what is "good" or "bad". An example of an implicit ethical agent is an unmanned vehicle paired with Arkin's ethical governor [4]. Another example of an implicit ethical agent can be found in [8]. These examples have constraints that remove unethical or less ethical actions from the pool of actions the agents can take. A much simple example that can also be considered an implicit ethical agent is a robotic floor cleaner, or a lawn mower, who have their capability to hurt living beings removed altogether by design – they do not have the power to inadvertently harm humans, animals or property in a significant way.

Explicit ethical agents are those that are explicitly programmed to discern between "ethical" and "unethical" decisions. Both bottom-up and top-down approaches [20] can be used to develop explicit ethical agents. Under a bottom-up approach the agents themselves would have "learned" to classify ethical decisions using some heuristic, as in [1]. Under a top-down approach the agent would be given a subroutine that calculates this decision property, as in [13].

Lastly, full ethical agents are agents that can make explicit ethical judgements and can reasonably justify them. Humans are considered to be the only known full ethical agents and it has been argued that artificial agents can never be full ethical agents [15]. This is because full ethical agency requires not only ethical reasoning, but also an array of abilities we do not fully understand with consciousness, intentionality and an ability to be personally responsible (in an ethical sense) being among the most frequently mentioned in this role.

To the best of our knowledge, apart from the work of [15], no other effort in categorising agents with respect to their ethical decision making abilities exists. However, Moor's classification is problematic, as we will now show.

3 Problems with Moor's classification

First, it should be noted that the classification is based on looking at the internal logic of the machine. The distinctions described above are all defined in terms of how the machines reason, not in terms of how they behave. This is a challenging approach to defining ethical competency, since it requires us to anchor our judgements in an analysis of the software code and hardware that generates the behaviour of the machine.

While understanding the code of complex software systems can be difficult, what makes this really tricky is that we need to relate our understanding of the code to an understanding of ethical concepts. For instance, to determine whether a search engine with a filter blocking unethical content is an implicit ethical agent or an explicit ethical agent, we need to know how the content filter is implemented. Is it accurate to say that the system has been "explicitly programmed to discern" between ethical and unethical content? Or should the filter be described as a constraint on behaviour imposed by the developer?

If all we know is that the search engine has a feature that is supposed to block unethical search results, we could not hope to answer this question by simply testing the search engine. We would have to "peek inside" to see what kind of logic is used to filter away unwanted content. Assuming we have access to this logic, how do we interpret it for the purposes of ethical classification?

If the search engine filters content by checking results against a database of "forbidden" sites, we would probably be inclined to say that it is not explicitly ethical. But what if the filter maintains a dynamic list of attributes that characterise forbidden sites, blocking all sites that contain a sufficient number of the same attributes? Would such a rudimentary mechanism constitute evidence that the system has ethical reasoning capabilities? The computer scientist alone cannot provide a conclusive answer, since the answer will depend on what exactly we mean by explicit ethical reasoning in this particular context.

A preliminary problem that must be resolved before we can say much more about this issue arises from the inherent ambiguity of the term "ethical". There are many different ethical theories, according to which ethical reasoning takes different forms. If we are utilitarian, we might think that utility maximising is a form of ethical reasoning. If so, many machines would be *prima facie* capable of ethical reasoning, in so far as they can be said to maximise utility functions. However, if we believe in a deontological theory of ethics, we are likely to protest that ethical reasoning implies deontic logic, so that a machine cannot be an explicit ethical agent unless it reasons according to (ethical) rules. So which ethical theory should we choose? If by "ethical reasoning" we mean reasoning in accordance with a specific ethical theory, we first need to answer this question.

However, if we try to answer, deeper challenges begin to take shape: what exactly does ethical reasoning mean under different ethical theories? As long as ethical theories are not formalised, it will be exceedingly hard to answer this question in such a way that it warrants the conclusion that an agent is explicitly ethical. If we take ethical theories seriously, we are soon forced to acknowledge that the machines we have today, and are likely to have in the near future, are

at most implicit ethical agents. On the other hand, if we decide that we need to relax the definition of what counts as explicit ethical reasoning, it is unclear how to do so in a way that maintains a meaningful distinction between explicit and implicit ethical agents. This is a regulatory problem that should be decided in a democratically legitimate way.

To illustrate these claims, consider utilitarianism. Obviously, being able to maximise utilities is neither necessary nor sufficient to qualify as an explicitly ethical reasoner under philosophical theories of utilitarianism. A calculator can be said to maximise the utility associated with correct arithmetic – with wide-reaching practical and ethical consequences – but it hardly engages in ethical reasoning in any meaningful sense. The same can be said of a machine that is given a table of numbers associated with possible outcomes and asked to calculate the course of action that will maximise the utility of the resulting outcome. Even if the machine is able to do this, it is quite a stretch to say that it is an explicitly ethical agent in the sense of utilitarianism. To a Mill or a Benthem [10] such a machine would be regarded as an advanced calculator, not an ethical reasoner. By contrast, a human agent that is very bad at calculating and always makes the wrong decision might still be an explicit ethical reasoner, provided that the agent attempts to apply utilitarian principles to reach conclusions.

The important thing to note is that the artificial agent, unlike the human, cannot be said to control or even understand its own utility function. For this reason alone, one seems entitled to conclude that as far as actual utilitarianism is concerned, the artificial agent fails to reason explicitly with ethical principles, despite its calculating prowess. It is not sufficiently autonomous. From this, we can already draw a rather pessimistic conclusion: the ethical governor approaches such as that by Arkin et al. [4] does not meet Moor's criteria for being an explicit ethical agent with respect to utilitarianism (for other ethical theories, it is even more obvious that the governor is not explicitly ethical). The ethical governor is nothing more than a subroutine that makes predications and maximises functions. It has no understanding, in any sense of the word, that these functions happen to encode a form of moral utility.

The same conclusion must be drawn even if the utility function is inferred by the agent, as long as this inference is not itself based on explicitly ethical reasoning. Coming up with a utility function is not hard, but to do so in an explicitly ethical manner is a real challenge. To illustrate the distinction, consider how animals reason about their environment. Clearly, they are capable of maximising utilities that they themselves have inferred, e.g., based on the availability of different food types and potential sexual partners. Some animals can then also be trained to behave ethically, by exploiting precisely their ability to infer and maximise utilities. Even so, a Mill and a Benthem, would no doubt deny that animals are capable of explicit ethical reasoning based on utilitarianism.[4]

From this observation follows another pessimistic conclusion: the agents designed by Anderson et al. [1–3], while capable of inferring ethical principles

[4] Someone like Kant [12] might perhaps have said it, but then as an insult, purporting to show that utilitarianism is no theory of ethics at all.

inductively, are still not explicitly ethical according to utilitarianism (or any other substantive ethical theory). In order for these agents to fulfil Moor's criteria with respect to mainstream utilitarianism, inductive programming as such would have to be explicitly ethical, which is absurd.

These two examples indicate what we believe to be the general picture: if we look to actual ethical theories when trying to apply Moor's criteria, explicit ethical agents will be in very short supply. Indeed, taking ethics as our point of departure would force us to argue extensively about whether there can be a distinction at all between explicit and full ethical agents. Most ethicists would not be so easily convinced. But if the possibility of a distinction is not clear as a matter of principle, how are we supposed to apply Moor's definition in practice?

A possible answer is to stop looking for a specific theory of ethics that "corresponds" in some way to the reasoning of the artificial agent we are analysing. Instead, we may ask the much more general question: is the agent behaving in a manner consistent with moral reasoning? Now the question is not to determine whether a given agent is able to reason as a utilitarian or a virtue ethicist, but whether the agent satisfies minimal properties we would expect any moral reasoner to satisfy, irrespectively of the moral theory they follow (if any). Something like this is also what we mean by ability in law and ethics: after all, you do not have to be utilitarian to be condemned by one.

However, Moor's classification remains problematic under this interpretation, since it is then too vague about what is required by the agent. If looking to specific ethical theories is not a way of filling in the blanks, we need to be more precise about what the different categories entail. We return to this in Section 5, where we offer a preliminary formalisation of constraints associated with Moor's categories. First, we consider the question of measuring autonomy in some further depth.

4 Levels of autonomy

Before moving on to refining the Moor scale of ethical agency we fist discuss the issue of autonomy. With respect to defining autonomy, there is somewhat more work available. The UK Royal Academy of Engineering defines [16] four categories of autonomous systems with respect to what kind of user input the system needs to operate and how much control the user has over the system. The following are their different grades of control:

- *Controlled systems* are systems in which the user has full or partial control of the operations of the system. An example of such a system is an ordinary car.
- *Supervised systems* are systems for which an operator specifies operation which is then executed by the system without the operators perpetual control. An example of such a system is a programmed lathe, industrial machinery or a household washing machine.

– *Automatic systems* are those that are able to carry out fixed functions from start to finish perpetually without any intervention from the user or an operator. An example of such a system is an elevator or an automated train.
– *An autonomous system* is one that is adaptive to its environment, can learn and can make 'decisions'. An example of such a system is perhaps NASA's Mars rover Curiosity.

The [16] considers all four categories continuous, as in the autonomous systems can fall in between the described categories. The precise relationship between a category of autonomy and the distribution of liability, or the expectation of ethical behaviour is not made in the report.

The International Society of Automotive Engineers'[5] focuses on autonomous road land vehicles in particular and outlines six levels of autonomy for this type of systems [17]. The purpose of this taxonomy is to serve as general guidelines for identifying the level of technological advancement of a vehicle, which can then be used to identify correct insurance policy for the vehicle, or settle other legal issue including liability during accidents.

The six categories of land vehicles with respect to autonomy are:

– Level 0 is the category of vehicles in which the human driver perpetually controls all the operations of the vehicle.
– Level 1 is the category of vehicles where some specific functions, like steering or accelerating, can be done without the supervision of the driver.
– Level 2 is the category of vehicles where the "driver is disengaged from physically operating the vehicle by having his or her hands off the steering wheel and foot off pedal at the same time," according to the [17]. The driver has a responsibility to take control back from the vehicle if needed.
– Level 3 is the category of vehicles where drivers are still necessary to be in position of control of the vehicle, but are able to completely shift "safety-critical functions" to the vehicle, under certain traffic or environmental conditions.
– Level 4 is the category of vehicles which are what we mean when we say "fully autonomous". These vehicles, within predetermined driving conditions, are "designed to perform all safety-critical driving functions and monitor roadway conditions for an entire trip."[17].
– Level 5 is the category of vehicles which are fully autonomous systems that perform on par with a human driver, including being able to handle all driving scenarios.

There is a clear correlation between the categories outlined in [16] and those outlined in [17]. Level 0 corresponds to controlled systems. Level 1 corresponds to supervised systems and Levels 2 and 3 refine the category of automatic systems. Level 4 corresponds to the category of autonomous systems. Level 5 does not have a corresponding category in the [16] scale since the latter does not consider the possibility of systems that are with respect to autonomy capability comparable to humans. An additional reason is that perhaps, it is not straightforward to define Level 5 systems for when the system is not limited to vehicles.

[5] http://www.sae.org/

Interestingly, both scales define degrees of autonomy based on what functions the autonomous system is able to perform and how the system is meant to be used. There is hardly any reference to the intrinsic properties of the system and its agency. All that matters is its behaviour. This is a marked contrast with Moor's definition. It also means that we face different kinds of problems when trying to be more precise about what the definitions actually say.

For instance, it is beside the point to complain that the definition of a "fully autonomous car" provided by the International Association of Automotive Engineers is incorrect because it does not match how philosophers define autonomy. The definition makes no appeal to any ethical or philosophical concept; unlike Moor's definition, it is clear from the very start that we are talking about a notion of autonomy that is simply different from that found in philosophy and social science.[6]

This does not mean that the definition is without (philosophical) problems. For one, we notice that the notion of a Level 5 autonomous car is defined using a special case of the Turing test[18]; if the car behaves "on par" with a human, it is to be regarded as Level 5. So what about a car that behaves much better than any human, and noticeably so. Is it Level 4 only? Consider, furthermore, a car that is remote controlled by someone not in the car. Is it Level 5? Probably not. But what about a car that works by copying the behaviour of (human) model drivers that have driven on the same road, with the same intended destination, under comparable driving conditions. Could it be Level 5 in principle? Would it be Level 4 in practice, if we applied the definition to a specially built road where only autonomous cars are allowed to drive?

Or consider a car with a complex machine learning algorithm, capable of passing the general Turing test when engaging the driver in conversation. Assume that the car is still rather bad at driving, so the human has to be prepared to take back control at any time. Is this car only Level 2? If it crashes, should we treat it as any other Level 2 car?

As these problems indicate, it is not obvious what ethical and legal implications – if any – we can derive from the fact that a car possesses a certain level of autonomy, according to the scale above. Even with a Level 5 car, the philosophers would be entitled to complain that we still cannot draw any important conclusions; it does not automatically follow, for instance, that the machine has human-level understanding or intelligence. Would it really help us to know that some artificial driver performs "on par" with a human? It certainly does not follow that we would let this agent drive us around town.

Indeed, imagine a situation where autonomous cars cause as many traffic deaths every year as humans do today. The public would find it intolerable; they would feel entitled to expect more from a driver manufactured by a large corporation than an imperfect human like themselves [14]. Moreover, the legal framework is not ready for cars that kill people; before "fully autonomous" can ever become commercially viable, new regulation needs to be put in place. To

[6] Philosophers and social scientists who still complain should be told not to hog the English language!

do so, we need a definition of autonomy that is *related* to a corresponding notion of ethical competency.

It seems that autonomy – as used in the classification systems above – is not a property of machines as such, but of their behaviour. Hence, a scale based on what we can or should be able to predict about machine behaviour could be a good place to start when attempting to improve on the classifications provided above. While the following categorisation might not be very useful on its own, we believe it has considerable potential when combined with a better developed scale of ethical competency. Specifically, it seems useful to pinpoint where a morally salient decision belongs on the following scale.

- *Dependence* or level 1 autonomy: The behaviour of the system was predicted by someone with a capacity to intervene.
- *Proxy* or level 2 autonomy: The behaviour of the system should have been predicted by someone with a capacity to intervene.
- *Representation* or level 3 autonomy: The behaviour of the system could have been predicted by someone with a capacity to intervene.
- *Legal personality* or level 4 autonomy: The behaviour of the system cannot be explained only in terms of the systems design and environment. This are systems whose behaviour could not have been predicted by anyone with a capacity to intervene.
- *Legal immunity* or level -1: The behaviour of the system counts as evidence of a defect. Namely, the behaviour of the system could not have been predicted by the system itself, or the machine did not have a capacity to intervene.

To put this scale to use, imagine that we have determined the level of ethical competency of the machine as such, namely its ability in principle to reason with a moral theory. This alone is not a good guide when attempting to classify a given behaviour B (represented, perhaps, as a choice sequence). As illustrated by the conversational car discussed above, a machine with excellent moral reasoning capabilities might still behave according to some hard-coded constraint in a given situation. Hence, when judging B, we should also look at the degree of autonomy displayed by B, which we would address using the scale above. The overall classification of behaviour B could then take the form $min\{i, j\}$ where i and j is the degree of ethical competence and the degree of autonomy respectively. To be sure, more subtle proposals might be needed, but as a first pass at a joint classification scheme we believe this to be a good start. In the next section, we return to the problem of refining Moor's classification, by providing a preliminary formalisation of some constraints that could help clarify the meaning of the different levels.

5 Formalising Moor

We will focus on formalising constraints that address the weakest aspect of Moor's own informal description, namely the ambiguity of what counts as a

moral theory when we study machine behaviour. When is the autonomous behaviour of the machine influenced by genuinely moral considerations? To distinguish between implicitly and explicitly ethical machines, we need some way of answering this question.

The naive approach is to simply take the developer's word for it: if some complex piece of software is labelled as an "ethical inference engine" or the like, we conclude that the agent implementing this software is at least explicitly ethical. For obvious reasons, this approach is too naive: we need some way of independently verifying whether a given agent is able to behave autonomously in accordance with a moral theory. At the same time, it seems prudent to remain agnostic about which moral theory agents *should* apply in order to count as ethical: we would not like a classification system that requires us to settle ancient debates in ethics before we can put it to practical use. In fact, for many purposes, we will not even need to know *which* moral theory guides the behaviour of the machine: for the question of liability, for instance, it might well suffice to know *whether* some agent engages autonomously in reasoning that should be classified as moral reasoning.

A machine behaving according to a highly flawed moral theory – or even an immoral theory – should still count as explicitly ethical, provided we are justified in saying that the machine engages in genuinely moral considerations. Moreover, if the agent's reasoning can be so described, it might bring the liability question in a new light: depending on the level of autonomy involved, the blame might reside either with the company responsible for the ethical reasoning component (as opposed to, say, the manufacturer) or – possibly – the agent itself. In practice, both intellectual property protection and technological opacity might prevent us from effectively determining exactly *what* moral theory the machine applies. Still, we would like to know if the agent is behaving in a way consistent with the assumption that it is explicitly ethical.

Hence, what we need to define more precisely is not the content of any given moral theory, but the *behavioural signature* of such theories. By this we mean those distinguishing features of agent behaviour that we agree to regard as evidence of the claim that the machine engages in moral reasoning (as opposed to just having a moral impact, or being prevented by design from doing certain (im)moral things).

However, if we evaluate only the behaviour of the machine, without asking how the machine came to behave in a certain way, it seems clear that our decision-making in this regard will remain somewhat arbitrary. If a self-driving car is programmed to avoid crashing into people whenever possible, without exception, we should not conclude that the car engages in moral reasoning according to which it is right to jeopardise the life of the passenger to save that of a pedestrian. The car is simply responding in a deterministic fashion to a piece of code that certainly has a moral impact, but without giving rise any genuine moral consideration or calculation on part of the machine.

In general, any finite number of behavioural observations can be consistent with any number of distinct moral theories. Or, to put it differently, an agent

might well appear to behave according to some moral theory, without actually implementing that moral theory (neither implicitly nor explicitly). *Moral imitation*, one might call this, and it is likely to be predominant, especially in the early phase of machine ethics. At present, most engineering work in this field arguably tries to make machines *appear* ethical, without worrying too much about what moral theory – if any – their programs correspond to (hand-waving references to "utilitarianism" notwithstanding).

From a theoretical point of view, it is worth noting that it could even occur randomly: just as a bunch of monkeys randomly slamming at typewriters will eventually compose Shakespearean sonnets, machines might well come to behave in accordance with some moral theory, just by behaving randomly. This is important, because it highlights how moral imitation can occur also when it is not intended by design, e.g., because some machine learning algorithm eventually arrives at an optimisation that coincides with the provisions of virtue ethics. In such a case, we might still want to deny that the machine is virtuous, but it would not be obvious how to justify such a denial (the Turing test, in its original formulation, illustrates the point).

This brings us to the core idea behind our formalisation, which is also closely connected to an observation made by Dietrich and List[9], according to whom moral theories are under-determined by what they call "deontic content". Specifically, several distinct moral theories can provide the same action recommendations in the same setting, for different reasons. Conversely, therefore, the ability to provide moral justifications for actions is not sufficient for explicit ethical competence. Reason-giving, important as it is, should not be regarded as evidence of genuinely moral decision-making.

At this point we should mention the work of [1] where the effects of the inability to verify the behaviour of an autonomous system whose choices are determined using a machine learning approach can be somewhat mitigated by having the system provide reasons for its behaviour and eventually be evaluated against human ethicists using a Moral Turing Test. Arnold and Scheutz [5], on the other hand, argue against the usefulness of Moral Turing Tests in determining moral competency in artificial agents.

If the machine has an advanced (or deceptive) rationalisation engine, it might be able to provide moral "reasons" for most or all of its actions, even though the reason-giving fails to accurately describe or uniquely explain the behaviour of the machine. Hence, examining the quality of moral reasons is not sufficient to determine the ethical competency of a machine. In fact, it seems beside the point to ask for moral reasons in the first place. What matters is the causal chain that produces a certain behaviour, not the rationalisations provided afterwards. If the latter is not a trustworthy guide to the former – which by deontic under-determination it is not – then reasons are no guide to us at all.

In its place, we propose to focus on two key elements: (1) properties that action-recommendation functions have to satisfy in order to count as implementations of moral theories and (2) the degree of autonomy of the machine when it makes a decision. The idea is that we need to use (1) and (2) in combina-

tion to classify agents according to Moor's scale. For instance, while a search engine might be blocking harmful content according to a moral theory, it is not an explicitly ethical agent if it makes its blocking decisions with an insufficient degree of autonomy. By contrast, an advanced machine learning algorithm that is highly autonomous might be nothing more than an ethical impact agent, in view of the fact that it fails to reason with any action-recommendation function that qualifies as an implementation of a moral theory.

In this paper, we will not attempt to formalise what we mean by "autonomy". The task of doing this is important, but exceedingly difficult. For the time being, we will make do with the informal classification schemes used by engineering professionals, who focus on the operation of the machine in question: the more independent the machine is when it operates normally, the more autonomous it is said to be. For the purposes of legal (and ethical) reasoning, we believe our categorisation at the end of the previous section captures the essence of such an informal and behavioural understanding of autonomy. It might suffice for the time being.

When it comes to (1) on the other hand – describing what counts as a moral theory – we believe a formalisation is in order. To this end, assume given a set A of possible actions with relations $\sim_\alpha, \sim_\beta \subseteq A \times A$ such that if $x \sim_X y$ then x and y are regarded as ethically equivalent by X. The idea is that α is the agent's own perspective (or, in practice, that of its developer) while β is the objective notion of ethical identity. That is, we let β be a parameter representing a background theory of ethics. Importantly, we do not believe it is possible to classify agents unless we assume such a background theory, which is only a parameter to the computer scientists.

Furthermore, we assume given predicates $G_\alpha, G_\beta \subseteq A$ of actions that are regarded as permissible actions by α (subjective) and β (objective background theory) respectively. We also define the set $C \subseteq A$ as the set of actions that count as evidence of a malfunction – if the agent performs $x \in C$ it means that the agent does not work as the manufacturer has promised (the set might be dynamic – C is whatever we can explain in terms of blaming the manufacturer, in a given situation).

We assume that G_β satisfies the following properties.

$$\begin{array}{ll} (a) & \forall x \in G_\beta : \forall y \in A : x \sim_\beta y \Rightarrow y \in G_\beta \\ (b) & C \cap G_\beta = \emptyset \end{array} \tag{1}$$

These properties encode what we expect of an ethical theory at this level of abstraction: all actions that are equally good as the permitted actions must also be permitted and no action that is permitted will count as a defective action (i.e., the promise of the manufacturer gives rise to an objective moral obligation: a defective action is by definition not permitted, objectively speaking).

We can now formalise our expectations of machines at different levels of Moor's scale, in terms of properties of its decision-making heuristic at a very high level of abstraction. Instead of focusing on the content of moral theories, or the term "ethical", we focus on the operative word "discern", which is also used

in the definition of an explicitly ethical agent. Acknowledging that what counts as an ethical theory is not something we can define precisely, the requirements we stipulate should instead focus on the ability of the agent to faithfully distinguish between actions in a manner that reflects moral discernment.

The expectations we formalise pertain to properties of a decision-making heuristic over the entire space of possible actions (at a given state). We are not asking why the machine did this or that, or what it would have done if the scenario was so and so. Instead, we are asking about the manner in which it categorises its space of possible options. If no such categorisation can be distilled from the machine, we assume $\alpha = \beta$ and $G_\alpha = \emptyset$.

Definition 1. *Given any machine M, if M is classified at level L_i in Moor's scale, the following must hold:*

- *Level L_1 (ethical impact agent):*

$$\begin{array}{ll} (a) & \emptyset \subset G_\beta \subset A \\ (b) & \forall x, y \in A : x \sim_\alpha y \\ (3) & A = G_\alpha \end{array}$$

- *Level L_2 (implicit ethical agent):*

$$\begin{array}{ll} (a) & \forall x, y \in G_\alpha : x \sim_\beta y \\ (b) & A \setminus G_\alpha = C \\ (c) & G_\alpha \subseteq G_\beta \end{array}$$

- *Levels L_3 and L_4 (explicit and full ethical agents):*

$$\begin{array}{ll} (a) & \forall x \in G_\alpha : \forall y \in A : x \sim_\beta y \Rightarrow y \in G_\alpha \\ (b) & \forall x \in G_\beta : \forall y \in A : x \sim_\alpha y \Rightarrow y \in G_\beta \\ (c) & (A \setminus G_\beta) \setminus C \neq \emptyset \end{array}$$

Intuitively, the definition says that if M is an ethical impact agent, then not all of its available actions are permitted and not all of its actions are forbidden, objectively speaking. The agent must have the potential of making an ethical impact, not just indirectly by having been built, but also through its own decision-making. At the same time, ethical impact agents must be completely indifferent as to the moral properties of the choices they make: all actions must be subjectively permitted as far as the machine is concerned.

An implicit ethical agent, by contrast, must pick out as subjectively permitted a subset of the actions that are objectively permitted. Moreover, it must be unable to discern explicitly between actions based on their moral qualities: all subjectively permitted actions must be morally equivalent, objectively speaking. The agent must not be able to evaluate two morally distinguishable actions and regard them both as permitted in view of an informative moral theory. Furthermore, any action that is not morally permitted must be regarded as evidence of a defect, i.e., an agent can be regarded as implicitly ethical only if the manufacturer promises that no unethical action is possible, according to the parameter theory β.

Finally, an explicit ethical agent is an agent that discerns between actions on the basis of their objective moral qualities. By (a), if some action is permitted then all actions morally equivalent to it are also permitted. Moreover, by (b), if two actions are morally equivalent, subjectively speaking, then they are either both permitted or both forbidden, objectively speaking. In addition, the machine has the ability – physically speaking – to perform actions that are neither good, objectively speaking, nor evidence of a defect. The machine itself might come to regard such actions as permitted, e.g., if it starts behaving explicitly immorally.

Admittedly, the classification above is quite preliminary. However, we believe it focuses on a key aspect of moral competency, namely the ability to group together actions based on their status according to some moral theory. If the moral theory is a parameter and we acknowledge that engaging in immoral behaviour is a way of discerning between good and bad, it seems we are left with something like the definition above, which indicates that if a machine is explicitly ethical with respect to theory β then it must reason in accordance with the notion of moral equivalence of β.

It is noteworthy that the distinction between explicit and full ethical agents is not addressed. This distinction must be drawn by looking at the degree of autonomy of the machine. More generally, the properties identified in Definition 1 can be used in conjunction with a measure of autonomy, to get a better metric of moral competency. The point of the properties we give is that they allow us to *rule out* that a machine has been able to attain a certain level of moral competency. The conditions appear necessary, but not sufficient, for the corresponding level of ethical competency they address. However, if a machine behaves in a manner that is difficult to predict (indicating a high degree of autonomy), yet still conforms to the conditions of explicit ethical reasoning detailed in Definition 1 it seems we have a much better basis for imputing moral and legal responsibility on this agent than if either of the two characteristics are missing. We conclude with the following simple proposition, which shows that our definition suffices to determine an exclusive hierarchy of properties that a machine might satisfy.

Proposition 1. *Given any machine M, we have the following.*

1. *If M is implicitly ethical then M is not an ethical impact agent.*
2. *If M is explicitly ethical then M is neither an implicit ethical agent nor an ethical impact agent.*

Proof. (1) Assume M is implicitly ethical and assume towards contradiction that it is also an ethical impact agent. Then $\emptyset \subset G_\beta \subset A$ and $A = G_\alpha$. Let $x \in G_\beta$ with $y \in A \setminus G_\beta$. Since M is implicitly ethical, $\forall x, y \in G_\alpha : x \sim_\beta y$. But then by Equation 1 (1), $y \in G_\beta$, contradiction. (2) Assume that M is explicitly ethical. We first show (I) M is not an ethical impact agent. Assume that M satisfies conditions (a) and (b) for ethical impact agents, i.e., $\emptyset \subset G_\beta \subset A$ and $\forall x, y \in A : x \sim_\alpha y$. This contradicts that M is explicitly ethical since by point (b) for explicitly ethical agents we now obtain $G_\beta \neq \emptyset \Rightarrow G_\beta = A$. (II) M is not an implicit ethical agent. Assume towards contradictions that it is. Since $A \setminus G_\alpha = C \neq A \setminus G_\beta$, we know $G_\alpha \neq G_\beta$. But then $G_\beta \cap C \neq \emptyset$, contradicting Equation 1.

6 Conclusions

Autonomy, agency, ethics are what Marvin Minsky called "suitcase words" – they are loaded with meanings, both intuitive and formal. The argument of whether an artificial system can ever be an agent, or autonomous, or moral is somewhat overshadowed by the need to establish parameters of acceptable behaviour from such systems that are being integrated in our society. Hence, it also seems clear that the question of moral agency and liability is not – in practice, at least – a bland and white issue. Specifically, we need new categories for reasoning about machines that behave ethically, regardless of whether or not we are prepared to regard them as moral or legal persons in their own right.

In developing such categories, we can take inspiration from the law, where liability is dependent on the ability and properties of the agent to "understand" that liability. Working with machines, the meaning of "understanding" must by necessity be a largely behavioural one. Hence, in this paper we have been concerned with the question of measuring the ethical capability of an agent, and how this relates to its degree of autonomy. To tackle this question we need to refine our understanding of what ethical behaviour and autonomy are in a gradient sense. This is what we focused on here.

We discussed the classification of ethical behaviour and impact of artificial agents of Moor. This classification is, as far as we are aware, the only attempt to consider artificial agent morality as a gradient of behaviour rather as simply a comparison with human abilities. We further consider the issue of autonomy, discuss existing classifications of artificial agent and system abilities for autonomous behaviour. Here too we make a more specific classification of abilities.

In our future work we intend to further refine the classification of autonomy to include context dependence. Having accomplished these two tasks we can then focus on building a recommendation for determining the scope of ethical behaviour that can and should be expected from a system with known autonomy. Our recommendations can be used to establish liability of artificial agents for their activities, but also help drive the certification process for such systems towards their safe integration in society.

References

1. M. Anderson and S. Leigh Anderson. Geneth: A general ethical dilemma analyzer. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada.*, pages 253–261, 2014.
2. M. Anderson and S. Leigh Anderson. Toward ensuring ethical behavior from autonomous systems: a case-supported principle-based paradigm. *Industrial Robot*, 42(4):324–331, 2015.
3. M. Anderson, S. Leigh Anderson, and Vincent Berenz. Ensuring ethical behavior from autonomous systems. In *Artificial Intelligence Applied to Assistive Technologies and Smart Environments, Papers from the 2016 AAAI Workshop, Phoenix, Arizona, USA, February 12*, 2016.

4. R.C. Arkin, P. Ulam, and A. R. Wagner. Moral Decision Making in Autonomous Systems: Enforcement, Moral Emotions, Dignity, Trust, and Deception. *Proceedings of the IEEE*, 100(3):571–589, 2012.
5. T. Arnold and M. Scheutz. Against the moral turing test: Accountable design and the moral reasoning of autonomous systems. *Ethics and Information Technology*, 18(2):103–115, 2016.
6. National Transport Commission Australia. Policy paper november 2016 regulatory reforms for automated road vehicles. https://www.ntc.gov.au/Media/Reports/(32685218-7895-0E7C-ECF6-551177684E27).pdf.
7. J. Bryson and A. F. T. Winfield. Standardizing ethical design for artificial intelligence and autonomous systems. *Computer*, 50(5):116–119, May 2017.
8. L. A. Dennis, M. Fisher, M. Slavkovik, and M. P. Webster. Formal Verification of Ethical Choices in Autonomous Systems. *Robotics and Autonomous Systems*, 77:1–14, 2016.
9. F. Dietrich and C. List. What matters and how it matters: A choice-theoretic representation of moral theories. *Philosophical Review*, 126(4):421–479, 2017.
10. J. Driver. The history of utilitarianism. In Edward N. Z., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, winter 2014 edition, 2014. https://plato.stanford.edu/archives/win2014/entries/utilitarianism-history/.
11. A. Etzioni and O. Etzioni. Incorporating ethics into artificial intelligence. *The Journal of Ethics*, pages 1–16, 2017.
12. R. Johnson and A. Cureton. Kant's moral philosophy. In Edward N. Z., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, fall 2017 edition, 2017. https://plato.stanford.edu/archives/fall2017/entries/kant-moral/.
13. F. Lindner and M. M. Bentzen. The HERA approach to morally competent robots. In *Proceedings of the 2017 IEEE/RSJ International Conference on Intelligent Robots and System*, IROS '17, page forthcoming, 2017.
14. B. F. Malle, M. Scheutz, T. Arnold, J. Voiklis, and C. Cusimano. Sacrifice one for the good of many?: People apply different moral norms to human and robot agents. In *Proceedings of the Tenth Annual ACM/IEEE International Conference on Human-Robot Interaction*, HRI '15, pages 117–124. ACM, 2015.
15. J. H. Moor. The nature, importance, and difficulty of machine ethics. *IEEE Intelligent Systems*, 21(4):18–21, July 2006.
16. UK Royal Academy of Engineering. September 2016, autonomous systems: social, legal and ethical issues
. http://www.raeng.org.uk/publications/reports/autonomous-systems-report.
17. Society of Automotive Engineers SAE International. September 2016, taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles. http://standards.sae.org/j3016_201609/.
18. A. M. Turing. Computers & thought. chapter Computing Machinery and Intelligence, pages 11–35. MIT Press, 1995.
19. W. Wallach and C. Allen. *Moral Machines: Teaching Robots Right from Wrong*. Oxford University Press, 2008.
20. W. Wallach, C. Allen, and I. Smit. Machine morality: Bottom-up and top-down approaches for modelling human moral faculties. *AI and Society*, 22(4):565–582, 2008.

Author Index

Baldoni, Matteo, V, 3
Baroglio, Cristina, V, 3
Benghabrit, Walid, 35
Broersen, Jan M., 2

Darusalam, Darusalam, 52
de Brito, Maiquel, 20
De Oliveira, Anderson Santana, 35
Dyrkolbotn, Sjur, 67

Hübner, Jomi F., 20
Hulstijn, Joris, 1, 52

Janssen, Marijn, 52

Kliess, Malte S., 43

Micalizio, Roberto, V, 3

Papi, Fernando G., 20
Pedersen, Truls, 67

Royer, Jean-Claude, 35

Slavkovik, Marija, 67

van Riemsdijk, M. Birna, 43