MOCA: An ORM MOdel for Computational Accountability

Matteo Baldoni [*], Cristina Baroglio, Katherine M. May, Roberto Micalizio, and Stefano Tedeschi
Università degli Studi di Torino — Dipartimento di Informatica
c.so Svizzera 185, I-10149 Torino (Italy)
E-mail: firstname.lastname@unito.it

Abstract. Accountability is often seen as a key notion in distributed systems, both in the human world and in software, on top of which interaction is built, together with the sibling notion of responsibility. However, there is limited support for the specification and use of such concepts in computational settings. This paper proposes an information model for computational accountability, which describes what data have to be available to allow, in any situation of interest arising from a group of interacting parties, the investigation and identification of accountability. The characterization of accountability provided by the model is grounded in the notions of just expectation and control. The information model is expressed in Object-Role Modeling (ORM), since it is well-suited to capture the relational nature of the accountability concept. An advantage of our model is that a designer can verify the consistency of a domain w.r.t. a set of accountability requirements. The paper exemplifies this checking by means of an Answer Set Program (ASP).

Keywords: Accountability, Responsibility, ORM, Information Model

1. Introduction

Many contexts, both in the human world and in software, are characterized by the distribution of activities through a group of interacting parties: each member in the group takes care of a part of the activity, and the desired overall result is achieved only when each member behaves properly and properly interacts with the others. This happens both in human organizations, and in distributed systems like multi-agent systems (MAS).

As suggested in [15], an explicit handling of responsibility could be the base for modularizing complex software that is by nature distributed, and where multiple threads of execution run concurrently. In particular, the social sciences, from the seminal work in [25], identify in *accountability* the key notion on top of which interaction is built. In political sciences, e.g. [1], accountability is seen as a major driving force of individuals when it comes to deciding their own behavior. Accountability as a concept "emerges as a primary characteristic of governance where there is a sense of agreement and certainty about the legitimacy of expectations between the community members" [38]. In essence [27], accountability implies that some actors have the right to hold other actors to a set of standards, to judge whether they have fulfilled their responsibilities in light of these standards, and to impose sanctions if they determine that these responsibilities have not been met. In blame cultures [21], sanction is interpreted as punishment (and accountability is considered as close to blame), but in a broader perspective (see, for instance, [26] on tort law), the account giving becomes the crucial aspect, and in "sanction imposition" principles of remedy should be applied.

This paper pursues two research aims. First, it refines the characterization of accountability traced in [4,5,6]. For computational purposes, we consider accountability as a directed relationship between two parties: an account-giver who is held to give an account (i.e., an explanation) about a condition of interest, and an account-taker, who has the permission, under certain circumstances, to ask for an account. Re-

[*]Corresponding author. E-mail: baldoni@di.unito.it.

lying on a wide literature on accountability (including but not limited to [1,12,25,33,37,38,41]), we identify the main concepts and conditions that have to be met in order to realize such a computational perspective of accountability. Two notions, *expectation* and *control*, will emerge as pivotal for this purpose.

Second, the paper proposes MOCA, an *information model for accountability*, that is, a model that describes what kind of data needs to be available to develop systems that, in any situation of interest arising in a group of interacting agents, allow the identification of account-givers. The model, that we provide in Object-Role Modeling (ORM[1]) [28] due to the relational nature of the represented concepts, contributes to the development of systems that support the governance of a group of interacting parties. The model allows the verification of the consistency of a domain. In the paper, we have exemplified its realization by means of Answer Set Programming.

The proposal sets the ground for the development of approaches to MAS programming where interaction, coordination, and exception handling will rely on the sibling notions of responsibility and accountability.

Improvement on the conference paper and structure. This work improves the proposal presented at the AI*IA 2018 conference [8] as follows. The characterization of the concept of accountability Section 2 has been reviewed. The information model in Section 3 has been entirely revised in order to accommodate suggestions received at the conference; the new version of the model provides a better characterization of some central concepts, such as control and achievement. Section 4 discusses, as a novel contribution, an implementation of a consistency checker in ASP. Section 5 compares our model to other proposals for accountability.

2. Accountability and Responsibility

Responsibility, accountability, answerability, liability, causation, and sanction are all related terms, sometimes used as synonyms, other times used to capture different shades of meaning. A thorough ontological analysis of the term *accountability* is not in the scope of this work. We will provide just a minimal characterization, sufficient to understand the notions on which we rely for our information model. In essence, we understand accountability as representing the deceivingly simple concept of one principal holding another to account for his/her actions, both “good" and “bad." From this definition, we identify two integral pieces to accountability: 1) a relationship between two entities in which one feels a liability to account for his/her actions to the other; and 2) a process of accounting in which actions are declared, evaluated, and scored. Note that in this paper we focus on an interpretation of accountability where each principal is expected to act so as to contribute to social progress, and hence will be held to account for the satisfaction of that expectation. We do not consider, instead, those situations where a principal’s acting interferes with the some others’ tasks.

Feltus [24] uses accountability to qualify the notion of *responsibility*. He sees the latter as a unique charge assigned to an agent, which is always linked at least to one accountability. This view is compatible with the triangle model of responsibility [39], a psychological model by Schlenker *et al.* widely used in the context of human resource management, according to which the term bears two main understandings, each of which is investigated in depth by philosophers: one amounting to causation (who did it?) and the other to answerability (who deserves positive or negative treatment because of the event?). Schlenker *et al.* explain how responsibility, as individuals perceive it, depends on the strength of three linkages, each of which involves two out of three basic elements that are: prescriptions, events, and identities. Prescriptions come from regulative knowledge and (broadly speaking) concern what should be done or avoided. Events simply occur in the environment. Identities include but are not limited to roles of the individual that are relevant in the context. The three linkages, thus, respectively capture whether and to what extent: a prescription is considered to be applicable to an event, an event is considered relevant for an identity, a prescription is considered to bind an actor by virtue of his/her identity.

Accountability has distinctive traits that do not allow it to be a special kind of responsibility. First of all, it is a directed relationship; it involves two agents: the one who gives the account (*account-giver*) and the one who takes the account (*account-taker*) [12,33,37,39]. Indeed, following [39], accountability comes into being when an accredited public appraises a responsibility triangle. The account-taker can only be someone who has some kind of authority on the account-giver [18,38]. The authority may have origins of various type; for instance, it may be due to a principal-agent relationship or to a delegation. The account-taker is

[1]For an introduction to ORM, please check `http://www.orm.net/pdf/ORMwhitePaper.pdf`

sometimes called the *forum* [12]. Following ReMMO, accountability is in relationship with responsibility (it concerns a responsibility), and a responsibility may be subject to many accountabilities. Accountability may also involve a sanction, as a social consequence of the account-giver's achievement or non-achievement of what is expected, and of that entity providing or not an account.

All such considerations yield that, in order to properly tackle accountability in a computational system, it is necessary to identify the data that are specific to accountability and their relationships. We now summarize the key aspects of accountability that we rely upon in drawing the model together with the reference literature:

a) *Accountability implies agency.* If a principal does not possess the qualities to act "autonomously, interactively and adaptively," i.e. with agency, there is no reason to speak of accountability, for the agent would be but a tool, and a tool cannot be held accountable [40].
b) *Accountability requires but is not limited to causal significance.* The plain, physical causation (also called scientific causation in [39]), that does not necessarily involve *awareness* or *choice*, does not create responsibility, nor even accountability. [12,14] also support this view.
c) *Accountability does not hinder autonomy.* Indeed, accountability makes sense because of autonomy in deliberation [1,14,39,41].
d) *Accountability requires control.*
In [31], control is defined as the capability, possibly exercised indirectly via other agents, of bringing about events. Capability, thus, requires choice (i.e., acting to cause events), and/or awareness (i.e., knowing how to influence others). Due to our focus on accountability related to task completion (i.e., on bringing about a situation of interest), we interpret omissions (not acting) as non-achievements. The proposal in [19] gives a slightly different definition of control as the ability of an agent to maintain the truth value of a given state of affairs.
e) *Accountability requires observability.* In order to make correct judgments, a forum must be able to observe the necessary relevant information. However, in order to maintain modularity, a forum should not observe beyond its scope. For example, if a principal buys a product and the product is faulty, that principal holds the factory as a whole accountable. The factory, in turn, holds one of its members accountable for shoddy production. In other words, accountability determination is strictly related to a precise context. In each context, the forum must be able to observe events and/or actions strictly contained in its scope and decipher accountability accordingly. As context changes, accountability will also change. For this reason, a mechanism to compose different contexts and decide accountability comprehensively is essential.
f) *Accountability requires a mutually held expectation.* Accountability is a directed social relationship to regulate a behavior, that serves the purposes of sense-making and coordination in a group of interacting parties, all of whom share an agreement on how things should be done [25]. The role of expectation is widely recognized, see for example [1,25,41]. Both parties must be aware of such a relationship for it to have value (the account-taker to know when he/she has the right to ask for an account, the account-giver to know when and towards whom he/she is liable in case of request).
g) *Accountability is rights-driven.* One is held accountable by another who, in a certain context, has the claim-right to ask for the account. Particularly relevant on this aspect is the understanding of accountability that is drawn in tort law [18], where the account-taker is the only recognized authority who can ask for an account, and where the account-giver has a liability towards the account-taker (to explain when requested). Further analysis is carried out in [27].

3. Modeling Accountability with ORM

We have seen that accountability describes a relationship-centric approach, as it is fundamentally defined through relationships that permit one principal to take to account another. Without that relationship-permitting accounting, accountability would be reduced to a hybrid of traceability and blame-giving. To aid the modeling endeavor, use of ORM becomes a natural choice because of its built-in language advantage that places relationships at the center of its expressive power [28]. Throughout the process of modeling, we made use of both ORM and OWL (`https://www.w3.org/TR/owl2-syntax/`) to experiment with different tools and found that our instrument of choice, ORM, allowed us a more easily comprehensible description of accountability than OWL. We found the centrality of the relationship, rather than the entity, to be particularly apt at expressing our desired concepts in ORM. In particular, the possibility to place constraints on role groups in relationships proved invaluable for describing how relationships interact with and depend on one another. We recall that in this paper we focus on when principals do not act as they should (rather than act as they shouldn't). In other words, we are dealing with the *regulation of task completion* rather than task interference.

3.1. Accountability Requirements

For accountability to function, there must be a *base relationship* because of which a principal feels an obligation to account for his/her actions. The relationship entails a nuanced approach to unexpected outcomes or actions. For instance, a buyer may expect a seller to provide some goods and hold him/her to account should the seller not do so, but *not* in absence of payment. In other words, an unexpected action does not necessarily implicate wrongdoing, thanks to *mitigating circumstances* which circumscribe the scope of the accountability relationship – lack of payment is a mitigating circumstance. A straightforward case without mitigating circumstances would be a principal who acts with full understanding of his/her action effects and expected social role, that clearly caused the outcome/action in question, and could have chosen to act otherwise (i.e., has control over the given state of affairs). Thus, a forum would look for, among other qualities, *causation* as well as *autonomy* and *understanding* in action.

A forum's interest lies in assessing possibility to act, that is, if a principal had complete potential and autonomy as author of the outcome. We can also say he/she effectively caused the outcome through inaction contrary to social expectations (e.g., a merchant who refuses to provide goods that were paid as agreed). A forum must also determine a principal's *situational knowledge* of his/her expected agreed-upon role (e.g., the merchant must be aware the client expects him/her to ship the purchased items). Therefore, a model of accountability must respect the following requirements:

[R1] *Identify relationships of account giving between principals for certain outcomes;*
[R2] *Account for mitigating circumstances;*
[R3] *Establish a principal's qualities of:*
  .1 *Agency*
  .2 *Causal contribution to outcome;*
  .3 *Possibility/opportunity to act;*
[R4] *Allow for the passing of judgment.*

Requirement [R1] proves to be the simplest: we must place in *relationship* two principals (who are not necessarily individuals but may, for instance, amount to be organizations like a shipping company or an office) along with the agreed *expected outcomes*. Note that for us accountability relationships are always the result of an act that was explicitly, deliberately, and voluntarily performed by the account-giver, and that amount either to the creation of the accountability relationship by itself or to the acceptance of relationships that encode some "rules of the game" – e.g., when enacting some role.

[R2] refines the base relationship through *context*. Context contains a series of conditions that stipulate when there are no mitigating circumstances. We reiterate that we speak in the absence of interference from other system actors. Context, therefore, represents a kind of precondition to outcome's realization to be specified by the principal on whom the expectations lie. The specialized context also protects the principal by disallowing unattainable obligations. The accountability relationship consequently takes the form as described in [14].

[R3] presents us with a more complex modeling problem. For agency, we adopt the definition given in [11], "The person is an autonomous, intentional, and planning agent who is capable of distinguishing right and wrong and good and bad." A principal, arguably autonomous by design, satisfies the first part of the agency condition when he/she stipulates the ability to bring about an expectation from a context, that is, execute a plan, whether that be his/her own plan or a plan to hold another accountable. In order to know "good" from "bad," a principal must have foreknowledge of his/her social expectations, whose completion for all intents and purposes are "good" and "right," and whose non-completion are the only "bad" and "wrong" actions.

Likewise, for causal contribution, by declaring *control* over a *context* and *expectation*, a principal recognizes that one leads to another thanks to some intermediate action on her/his part. As a consequence, given context, the declared principal effectively *causes* either the outcome in question or its absence. A principal can subcontract out his/her plan, meaning another principal could be the "cause" of the first principal's outcome. From a modular viewpoint, the first retains his/her causal relevance for the outermost outcome as the "manager."

The possibility/opportunity to act condition completes a concept already covered by the previously discussed requirements. Thanks to a principal's *control* in a specified *context*, if that context comes about, that principal can *freely choose* to act or not act and bring about the desired *outcome*. Again, without interference through his/her control, that principal has the possibility and opportunity to act.

We choose to leave out the sanctioning piece, the requirement [R4], for future implementations of accountability. Our primary concern consists in identifying the information that is needed to *support* the passing of judgment and the possible consequent sanctions.

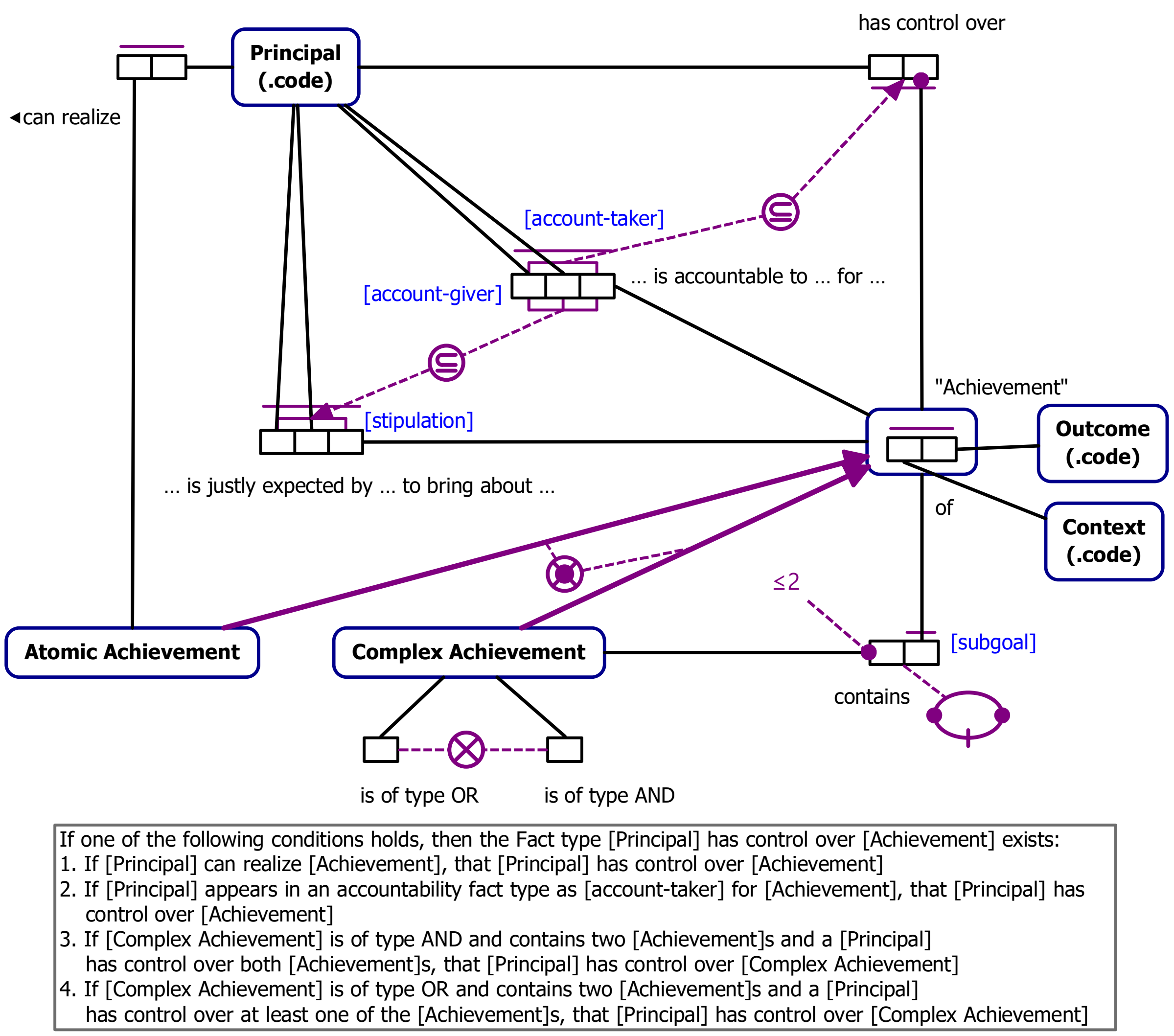
3.2. Building the Model

MOCA, our accountability information model, is shown in Figure 1 as an ORM model. In the following, we briefly report some ORM basics for those who are not familiar with this notation; for more details, see [28]. In ORM, all elements with the same name are the same entity/relationship. Entities/relationships can play identifiable (often named, e.g., *account-giver* or *subgoal*) roles inside relationships, that are explicitly depicted in the model. Entities can be identified by their rounded rectangular shape and relationships by their skinnier rectangular form that is divided into subrectangles. Each sub-rectangle identifies an individual role in a given relationship. Objectified relationships take the form of a relationship inside of an entity. Entities require a unique identifier, whose type is specified in parentheses underneath the entity name. Part (or all) of the roles that make up the relationship can be related to groups of roles inside another relationship by a constraint. Constraints are identifiable in the model by their purple color. For instance, in Figure 1 we use inclusive and exclusive *or*. Relationships represent facts. They are denoted by a reading that gives an intuition of the meaning of the relationship. Relationships can be further nuanced by some properties, for instance by graphically denoting reflexivity, a-/anti-/simmetry, and commutativity (e.g. *contains* is asymmetric).

Central to the model lies the relationship of accountability (accountability requirement [R1]), which contains two principals, one (account-giver) who is accountable to the other (account-taker), for an Achievement (... *is accountable to* ... *for* ...). An Achievement is a pair that is made of a Context and an Outcome, meaning that the interest in the Outcome raises in a specific Context –thus, the Context limits accountability for an Outcome. An Achievement can be structured into subgoals. The asymmetric relation *contains* represents such a structure. An Achievement that contains others is considered a Complex Achievement, otherwise it is an Atomic Achievement. An Achievement must be either Complex or Atomic but not both. Moreover, a Complex Achievement must contain at least one subgoal and at most two. Similarly, a Complex Achievement cannot simultaneously be of type AND (logical AND for its subgoals) and of type OR (logical OR for its subgoals).

The accountability relationship is further constrained so that a Principal who is accountable must have control over Achievement and that there must be a mutually held expectation on that Principal to act. In turn, a way to populate the fact type "Principal has control over Achievement" is by using the four conditions specified in the model note on the bottom of Figure 1. More precisely, first of all, a Principal is said to have control over an Achievement if that Principal can realize that Achievement. The notion of Achievement realization corresponds to the concept "direct control" and is captured in the fact type "Principal can realize Atomic Achievement." The Atomic Achievement restriction represents our understanding of Atomic Achievements as the basic building blocks that constitute more structured Achievements. Secondly, a Principal can have control over an Achievement if that Principal plays the role of account-taker in an accountability relationship for that same Achievement. Since the Principal can take another to account for that Achievement, that Principal has control. Thirdly, a Principal has control over a Complex Achievement of type AND if that same Principal controls all Achievements that play the role of subgoal to that Complex Achievement. Finally, a Principal has control over a Complex Achievement of type OR if that Principal controls at least one Achievement that plays the role of subgoal to that Complex Achievement. We note that, in the interest of information encapsulation, if a Principal has control over a Complex Achievement because of an accountability relationship, that Principal has no obligation to have control over the subgoals attached to that Complex Achievement.

Let us now discuss how MOCA meets the aforementioned requirements. We identify the following key nouns, which will take the shape of objects. *Principal* partially satisfies the agency requirement [R3] by representing an autonomous individual or organization



If one of the following conditions holds, then the Fact type [Principal] has control over [Achievement] exists:
1. If [Principal] can realize [Achievement], that [Principal] has control over [Achievement]
2. If [Principal] appears in an accountability fact type as [account-taker] for [Achievement], that [Principal] has control over [Achievement]
3. If [Complex Achievement] is of type AND and contains two [Achievement]s and a [Principal] has control over both [Achievement]s, that [Principal] has control over [Complex Achievement]
4. If [Complex Achievement] is of type OR and contains two [Achievement]s and a [Principal] has control over at least one of the [Achievement]s, that [Principal] has control over [Complex Achievement]

Fig. 1. ORM model for the accountability relationship.

who might potentially be thought of in legal terms as a "persona juris." *Context* and *Outcome* both represent sets of facts that characterize some states of interest. They are always associated with one another in the model, but the meaning of such association depends on further information. An Outcome represents a condition to achieve, i.e. a set of facts that should be brought about and that can be verified. Context can be interpreted in three ways: (1) the absence of mitigating circumstances (accountability requirement [R2]), (2) some preconditions under which a Principal has the possibility to pursue an Outcome, or (3) the condition under which an expectation that the Outcome will be achieved is activated. The relationships between such objects, based on the requirements that are explained in Section 3.1, are as follows:

1. *Accountability: Principal is accountable to Principal for Achievement.* Represents the actual atomic accountability relationship in which the first Principal can be held to account by the second for Achievement. It has three roles and can only exist if some of its elements are present both in a relationship of expectation (outgoing arrow towards ... *is justly expected by ... to bring about* ...), and in one of control (outgoing arrow towards ... *has control over* ...). This fact type satisfies both [R1] (relationship) and [R2] (mitigating circumstances) requirements.

2. *Expectation: Principal is justly expected by Principal to bring about Stipulation.* Partially satisfies the foreknowledge requirement of the agency condition in order to distinguish "right" from "wrong." By this relationship, the principal recognizes that if, given the opportunity, realizing the achievement is "good" and not doing so is "bad." With the word justly, we require that both principals are in agreement over the expectation.
3. *Principal has control over Achievement.* Control expresses contextual autonomy in that a Principal can effectively decide whether or not to bring about an Achievement and act on that decision. Control is constrained by the four rules listed.
4. *Principal can realize Achievement.* This type of control represents a modularization of knowledge. If a Principal could realize Achievement (that is, given Context it could realize Outcome), that Principal is effectively declaring knowledge and ability.
5. *Complex Achievement contains Achievement.* A Complex Achievement is an Achievement that can be divided into subgoals. That division then plays into the rules for control. Depending on the type of Complex Achievement, a Principal, in order to have control over it, must have control over one or more of its subgoals. Consequently, the relationship bears importance not only for structural definition but also for control.
6. *Complex Achievement is of type OR.* The population of the unary relationship means that the two subgoals it contains do not require the completion of both subgoals but at least one.
7. *Complex Achievement is of type AND* The population of the unary relationship means that the two subgoals it contains require the completion of both subgoals.

The model includes the following constraints between the above-listed relationships. The subset constraint between relationships *Accountability* and *Expectation* in Figure 1 satisfies the "agency requirement" (Section 3.1, accountability requirement [R3.1]) by ensuring that accountability is only possible in the presence of a previously established expectation. A Principal A, in order to be accountable to Principal B for a given Achievement, must be justly expected by Principal B to realize Achievement. In other words, it is not enough for a Principal to exert autonomy, there must also be a socially established agreement of expectation. Simply put, in order to be accountable, a Principal must not only exert situational autonomy, but must also be expected to do so.

Subset constraint between account-giver and Achievement in *Accountability*, on the one hand, and with *Control* on the other, instead satisfies both the "causal" as well as the "possibility/opportunity" condition ([R3.2] and [R3.3]). Similarly, it links mitigating circumstances to a principal-specified context. The constraint expresses in words that a Principal can be accountable for an Achievement only if that same Principal also has control over that Achievement, which means over the Outcome in the given Context. That is, in order to be accountable for an Outcome, a Principal must exhibit causal relevance over, and have possibility to realize, that Outcome and, thus, be a crucial influence in the truth value result of Outcome in Context.

4. Consistency Checking with ASP

The ORM model we have presented maintains all the relevant pieces of information for supporting accountability. A straightforward consequence is that, given facts about accountabilities, control, expectations concerning a specific domain, it is possible to verify their consistency. This can be done in many ways. For instance, by using NORMA[2], the development tool for ORM, or, more concretely, by, first, turning the ORM model into a relational model and, then, by applying the integrity constraints to the recorded facts [28, Chapter 11]. The way that we discuss in the present paper relies on Answer Set Programming (ASP) [30][3]. The constraints encoded within MOCA can be expressed in ASP through rules and integrity constraints, whereas model fact types are mapped into predicates. The consistency would then result in the satisfiability of the corresponding ASP program, i.e., the existence of an ASP model in which every achievement is controlled by a principal in accordance with the defined accountability requirements.

The use of ASP presents certain advantages in that it can check the consistency of a specification, but also, should the specification be partial, it can propose possible ways for completing it in a way that preserves consistency. Furthermore, the designer can add domain-specific constraints to check whether they are compatible with the required accountability rela-

[2]https://www.ormfoundation.org/.

[3]We made use of the clingo solver [32], available at `https://potassco.org/clingo/`.

tionships. For example, should a principal not be capable of realizing a certain achievement, the designer would specify the lack of control with the introduction of a negated predicate. The model(s) constructed by the solver, if any, can be then used as a basis to design a responsibility distribution that fits a set of imposed accountabilities, as discussed in [2].

In the following, we briefly discuss the rules and constraints that allow the encoding in ASP of the MOCA information model for accountability and conclude the section with a short example.[4]

We use the following predicates to denote the fact types expressed in the model:

`achievement(P,Q)` represents an achievement with context `P` and outcome `Q`.
`contains(A,A1)` where `A` and `A1` are achievements.
`complex(A)` achievement `A` is a complex achievement.
`atomic(A)` achievement `A` is an atomic achievement.
`typeAnd(A)` achievement `A` is of type AND.
`typeOr(A)` achievement `A` is of type OR.
`exp(X,Y,A)` principal `X` is justly expected by principal `Y` to realize achievement `A`.
`canRealize(X,A)` this predicate corresponds to the "can realize" fact type that denotes the notion of direct control. Principal `X` can realize achievement `A`.
`ctrl(X, A)` principal `X` controls achievement `A`. This control predicate is more general than the `can Realize(X,A)` predicate and corresponds to the fact type "has control over."
`a(X,Y,A)` represents the basic accountability relationship in which principal `X` is accountable to principal `Y` for achievement `A`.

Accountability, Expectation, and Can Realize. Let us now consider the most important consistency rules. The first one captures the subset constraint between accountability and expectation relationships:

```
exp(X,Y,A) :- a(X,Y,A).
```

The rule states that an accountability predicate implies an expectation predicate that involves the same principals and achievement.

The following three rules define when a principal should be capable of exerting direct control over an achievement (otherwise there would be an inconsistency).

[4]The full ASP code of the example is available at `http://di.unito.it/accountabilityasp`.

```
canRealize(X,A) :- a(X,_,A),
                   not a(_,X,A),
                   atomic(A).

canRealize(X,A1) :- a(X,_,A),
                    typeAnd(A),
                    contains(A,A1),
                    not a(_,X,A1),
                    atomic(A1).

0 {canRealize(X,A1):
      a(X,_,A), not a(_,X,A1),
      contains(A,A1), atomic(A1)} 2 :- typeOr(A)
```

The first rule provides stipulation for basic atomic achievements: if a principal is accountable for an atomic achievement and can hold no other principal accountable for that same achievement, then the principal must necessarily exert direct control over that achievement. The second and third rules, on the other hand, work with complex achievements that contain atomic achievements. In the second rule, we have a complex achievement (`A`) of type AND, which means that if principal `X` is accountable for realizing achievement `A`, and has no other principal to bring to account for one of its subgoals (`A1`), principal `X` must exert direct control over atomic achievement `A1`. Finally, the third rule, which is an aggregate rule, specifies that if an achievement `A` is of type OR, principal `X` who is accountable for `A` can exert direct control over at most two atomic achievement subgoals for which principal `X` cannot hold any other principal accountable. The numbers in the rules are the minimum and maximum cardinalities of the aggregate. Specifically, in the represented case they express the fact that, in a given ASP model, there can be at most two predicates `canRealize(X,A1)`, for a given principal `X` and an atomic achievement `A1` contained in a complex achievement `A` of type OR. Naturally, should principal `X` be able to hold another principal accountable for one of the subgoals, `X` has no obligation to realize it him/herself and correspondingly need not create even one `canRealize` fact. We will later show that principal `X` must have control over at least one of `A`'s subgoals with an integrity constraint over the `ctrl` predicate.

Structured Achievements. The following set of rules and integrity constraints shape the structure of the achievements. They can be either complex or atomic depending on whether or not they contain subgoals:

```
complex(achievement(P,Q)) :-
              achievement(P,Q),
              contains(achievement(P,Q),_).
```

The rule states that if an achievement is present in a `contains` predicate, meaning it contains subgoals, then that achievement is complex.

```
atomic(achievement(P,Q)) :-
               achievement(P,Q),
               not contains(achievement(P,Q),_).
```

Analogously, an atomic achievement contains no subgoals.

An achievement must be either complex or atomic but not both:

```
:- complex(A), atomic(A).

:- achievement(P,Q),
   not complex(achievement(P,Q)),
   not atomic(achievement(P,Q)).
```

The following encodes the asymmetric constraint on the contains relationship in which an achievement cannot contain another achievement, in which it is also contained:

```
:- contains(A1,A2),contains(A2,A1).
```

Only complex achievements have either AND or OR type:

```
:- typeOr(achievement(P,Q)),
   not complex(achievement(P,Q)).

:- typeAnd(achievement(P,Q)),
   not complex(achievement(P,Q)).
```

A complex achievement of type AND or of type OR must contain exactly two subgoals, which is expressed through the following aggregate rules and integrity constraints:

```
2 { containsDef(A,A1) : contains(A,A1) } 2 :-
      typeOr(A).
:- contains(A,A1),
   not containsDef(A,A1),
   typeOr(A).

2 { containsDef(A,A1) : contains(A,A1) } 2 :-
      typeAnd(A).
:- contains(A,A1),
   not containsDef(A,A1),
   typeAnd(A).
```

Control Rules. Finally, with the following integrity constraint, we specify that a principal cannot directly realize a complex achievement. This constraint is encoded in the information model by the entities that can participate in the "can realize" relationship; only an atomic achievement can be realized by a principal:

```
:- complex(A), canRealize(_,A).
```

The following rules and integrity constraints characterize the derivation of the control predicate. In particular, it can be derived in four ways, as described in the information model's control rules.

The first rule states that if a principal can realize an achievement, that principal has control over the achievement:

```
ctrl(X,A) :- canRealize(X,A).
```

The second rule states that if a principal can hold another to account for the accomplishment of an achievement, the former has control over the achievement:

```
ctrl(X,A) :- a(_,X,A).
```

If principal `X` is the account-taker in an accountability relationship for achievement `A`, we can derive that principal `X` has control over `A`.

The following rule specifies the third rule that handles complex achievements of type AND:

```
ctrl(X,A) :- typeAnd(A),
             contains(A,A1),
             contains(A,A2),
             ctrl(X,A1),
             ctrl(X,A2),
             A1!=A2.
```

Principal `X` has control over complex achievement `A` of type AND, if `X` controls both of `A`'s subgoals.

Moreover, we must impose that if principal `X` is accountable for a complex achievement of type AND, `X` should exert control over all `A`'s subgoals:

```
:- typeAnd(A),
   a(X,_,A),
   contains(A, A1),
   not ctrl(X, A1).
```

The fourth and final rule derives control in the presence of a complex achievement of type OR:

```
ctrl(X,A) :- typeOr(A),
             contains(A,A1),
             ctrl(X,A1).
```

Principal `X` has control over complex achievement `A` of type OR, if `X` controls at least one of `A`'s two subgoals.

Similarly to the previous case, the following integrity constraint imposes that if principal `X` is accountable for a complex achievement, `A`, of type OR, `X` must control at least one of `A`'s subgoals:

```
:- typeOr(A),
   a(X,_,A),
   contains(A, A1),
```

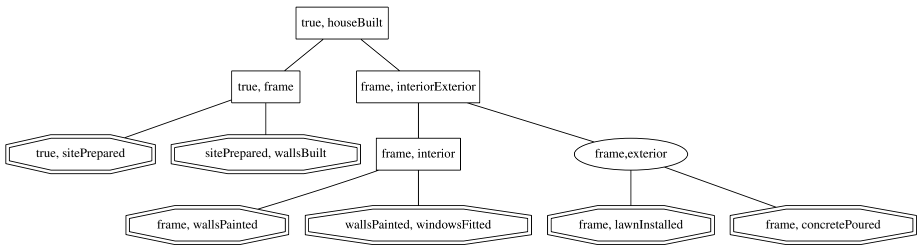


Fig. 2. Decomposition of the (true,houseBuilt) achievement.

```
    contains(A, A2),
    A1!=A2,
    not ctrl(X, A1),
    not ctrl(X, A2).
```

Finally, the last integrity constraint specifies that a principal `X` that is accountable for an achievement `A` must necessarily exert control over `A`:

```
:- a(X,_,A), not ctrl(X,A).
```

4.1. *Building-a-House Example*

To illustrate the use of the ASP program, we rely on a use case, inspired to the *building-a-house* example from [10]. The overall objective (to build a house for the house owner) can be decomposed into several subgoals, that are frame completion, interior completion, and exterior completion. These subgoals, are complex goals as well. Frame completion encompasses the site preparation and construction of walls. Interior completion includes painting the walls and the installation of windows. Finally, exterior completion involves either the installation of the lawn or the pouring of concrete. We can express goals in terms of achievements. Figure 2 graphically shows the decomposition of the overall (root) achievement, i.e., `(true, houseBuilt)`, into simpler achievements. The shape of the nodes represents their type. Rectangular nodes represent complex achievements of type AND, whereas elliptical nodes represents achievement of type OR. Octagonal nodes represent instead atomic achievements. The described decomposition is encoded in ASP as follows:

```
achievement(true, houseBuilt).
achievement(true, frame).
achievement(frame, interiorExterior).
achievement(frame, interior).
achievement(frame, exterior).
achievement(true, sitePrepared).
achievement(sitePrepared, wallsBuilt).
achievement(frame, wallsPainted).
achievement(wallsPainted, windowsFitted).
achievement(frame, lawnInstalled).
achievement(frame, concretePoured).
```

```
contains(achievement(true, houseBuilt),
         achievement(true, frame)).
contains(achievement(true, houseBuilt),
         achievement(frame, interiorExterior)).
contains(achievement(frame, interiorExterior),
         achievement(frame, interior)).
contains(achievement(frame, interiorExterior),
         achievement(frame, exterior)).
contains(achievement(true, frame),
         achievement(true, sitePrepared)).
contains(achievement(true, frame),
         achievement(sitePrepared, wallsBuilt)).
contains(achievement(frame, interior),
         achievement(frame, wallsPainted)).
contains(achievement(frame, interior),
         achievement(wallsPainted, windowsFitted)).
contains(achievement(frame, exterior),
         achievement(frame, lawnInstalled)).
contains(achievement(frame, exterior),
         achievement(frame, concretePoured)).

typeAnd(achievement(true, houseBuilt)).
typeAnd(achievement(frame, interiorExterior)).
typeAnd(achievement(true, frame)).
typeAnd(achievement(frame, interior)).
typeOr(achievement(frame, exterior)).
```

The house construction will involve many principals, each of which will be in charge of some specific parts of the overall achievement. In particular, we want to model a set of accountability relationships as requirements and check the possible existence of a model that satisfies them. We state that a contractor will be accountable to the owner for the overall achievement. The contractor will then rely on three managers (frame, interior, and exterior), for the corresponding subgoals. Similarly, each manager will have principals accountable towards them for the realization of the individual atomic goals. These accountability requirements can be captured by the following predicates:

```
a(contractor, owner,
  achievement(true, houseBuilt)).
a(frameManager, contractor,
  achievement(true, frame)).
a(interiorManager, contractor,
  achievement(frame, interiorExterior)).
a(interiorManager, interiorManager,
  achievement(frame, interior)).
a(exteriorManager, interiorManager,
  achievement(frame, exterior)).
```

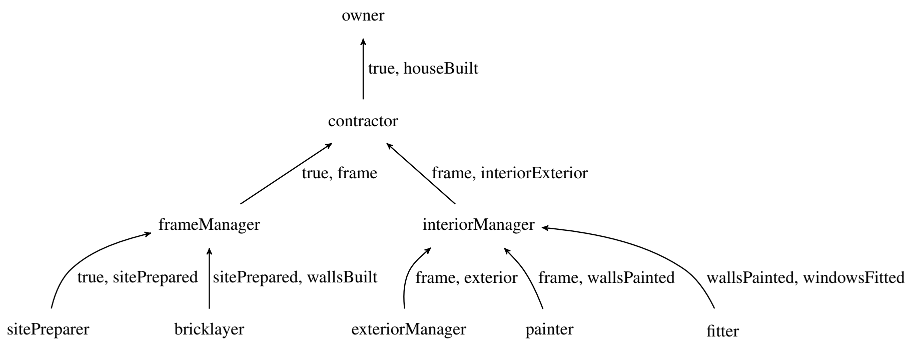


Fig. 3. Graphical description of the accountability relationships devised for the building-a-house scenario.

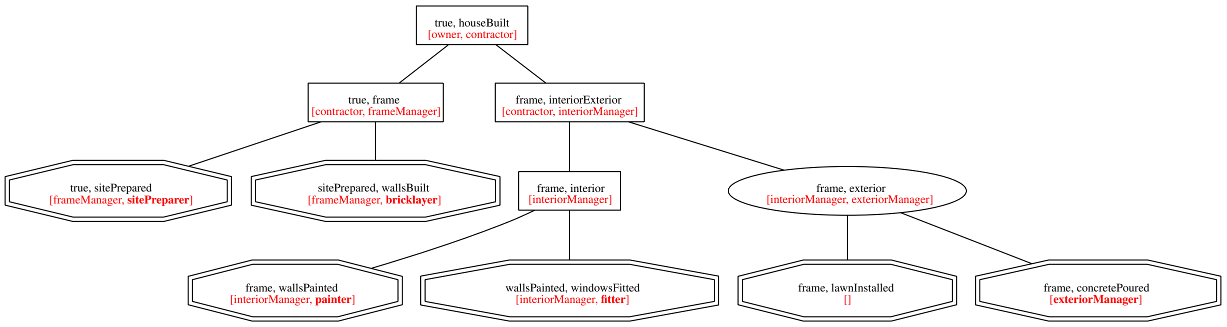


Fig. 4. Graphical description of one of the models found by the ASP solver for the building-a-house scenario. Each node is enriched by the list of principals that have control over it.

```
a(sitePreparer, frameManager,
  achievement(true, sitePrepared)).
a(bricklayer, frameManager,
  achievement(sitePrepared, wallsBuilt)).
a(painter, interiorManager,
  achievement(frame, wallsPainted)).
a(fitter, interiorManager,
  achievement(wallsPainted, windowsFitted)).
```

Figure 3 shows a graphical description of all the above accountability relationships. Each node represents a principal while edges are accountabilities directed from the account-taker to the account-giver and labeled with the corresponding achievement.

By running the program, the solver tells us first of all if the model is satisfiable, which means that we have a correct specification in which all the relevant achievements can be properly controlled. Secondly, the solver provides us with the models that contain information about the necessary expectations and controls over achievements to satisfy the accountability relationships. In our case, three models exist.

As an illustration, here follows the `canRealize` predicates produced by the solver for one of the models:

```
canRealize(sitePreparer,
         achievement(true,sitePrepared)).
canRealize(bricklayer,
         achievement(sitePrepared,wallsBuilt)).
canRealize(painter,
         achievement(frame,wallsPainted)).
canRealize(fitter,
         achievement(wallsPainted,windowsFitted)).
canRealize(exteriorManager,
         achievement(frame,concretePoured)).
```

These predicates encode a possible solution, that satisfies the accountability requirements expressed above, in terms of which principal should directly realize the atomic achievements. These predicates are the basic blocks over which control predicates over more complex achievements are built. The presence of three models, in our case, is due to the presence of a complex achievement of type OR. Indeed, having control over such a predicate could result in different `canRealize` predicates over the contained atomic achievements. To illustrate, let's consider a complex achievement `A` of type OR, which contains two atomic achievements `A1` and `A2`. A given principal `X` could have control over `A` because he/she can realize `A1`, even if he/she cannot realize `A2`, and vice versa. Still, `X` could be able to realize both `A1` and `A2` and he/she would have control over `A`, as well.

Figure 4 is built on the decomposition shown in Figure 2, by enriching each node with the list of principals who have control over the corresponding achievement, according to the ASP model produced by the solver. Different models would result in the same decomposition but with different control predicates. The figure highlights in bold those principals that have control over the achievements because they can directly realize them.

Note that despite no principal controls `achievement (frame,lawnInstalled)`, the parent node corresponding to `achievement(frame,exterior)` is properly controlled. This is due to the fact that `achievement(frame,exterior)` is a complex achievement of type OR. In order to control such an achievement, it is sufficient to control one of the two subgoals. This is true in our case since `achievement(frame,concretePoured)` is controlled by exteriorManager. An alternative, equally acceptable, ASP model would result in no principal controlling `achievement(frame,concretePoured)`, provided that `achievement(frame,lawnInstalled)` is properly controlled.

As discussed above, the designer can also specify further application-dependent constraints. For instance, we could indicate that the fitter is not capable, or enabled, of fitting doors, only windows, with the fact:

```
-canRealize(painter,
            achievement(frame, wallsPainted)).
```

The addition of this fact would render the specification unsatisfiable because it is inconsistent with the accountability requirement `a(painter, interiorManager, achievement(frame, wallsPainted))`. Conversely, if we state that the interior manager is not capable of directly performing the walls painting, the specification would remain satisfiable. Indeed, the manager exhibits control over the achievement thanks to the accountability relationship of the painter, in which he/she is account-taker.

5. Related Work

The proposal for an information model of accountability, explained in this work, is related to other proposals, not only from the artificial intelligence research area, that tackle aspects or uses of accountabilities. We briefly describe the most relevant ones and explain the connections to our proposal.

In [14], the concept of accountability requirement is defined as a directed relationship between two principals (account-giver and account-taker), an antecedent, and a consequent, which together constitute the mutual and conditional expectation between two parties. Should the antecedent become true, account-giver becomes accountable for the expected consequent to account-taker. An organization serves as a context for the accountability requirements. Principals enter the organization by playing at least one of its roles. Accountability requirements include commitments, authorizations, prohibitions, and empowerment. All the foreseen declinations of the accountability requirement only implicitly entail a notion of control.

The 2002 December Report of the Auditor General of Canada [34] recognizes accountability as a critical element of a representative democratic government, and proposes a well-known (and widely accepted) definition of accountability that takes recent developments in public management and governance into account. Here, accountability is defined as "a relationship based on obligations to demonstrate, review, and take responsibility for performance, both the results achieved in light of agreed expectations and the means used." All the elements of the definition are explained in Exhibit 9.1 [35]. The model we propose captures this understanding of accountability in a straightforward way. Of particular importance is the notion of *agreed expectation* stemming *from either a formal or informal agreement* as a key part of accountability. As underlined in our model, the creation of a just expectation relationship requires an agreement between the parties; an internal expectation, generated from one's opinions, is not enough to hold another accountable. Another important point is that the key focus of accountability is on results accomplished or not accomplished, i.e. on outcomes, for which principals explicitly take responsibility. Exhibit 9.2 explains that performance should be clearly linked with each party's capacity (e.g., skills) to deliver. This is what we capture with the relation *can-realize*. Exhibit 9.2 also explains that besides the expected accomplishments, also the operating constraints that must be respected should be explicit, understood, and agreed upon. Operating constraints amount to those mitigating circumstances which, in our model, are explicitly captured by the context and agreed upon by the principals involved.

In [17], Cranefield *et al.* discuss the notion of accountable autonomy within the context of practical reasoning agents. The authors start by listing some requirements that an agent should satisfy to be capable of performing practical reasoning on accountability. Then, they focus on the answerability aspect, which is defined as an obligation to provide an answer under some circumstances, and they provide a formalism that is meant to support the creation of replies done by accountable agents. In particular, the formalism specifies which pieces of information are needed in order to create the replies, among these, the length of the retrospective time, the maximum time allowed for the answer, the query language. As such, the proposal, though related, is complementary to the one addressed in this paper.

A protocol to ensure accountability as a design property in an organizational setting is proposed in [5,6,7]. The authors distill five principles for supporting accountability, which make use of the same concepts from the definition of multi-agent organizations (role, goal, and power). A commitment-based protocol is, then, defined with the aim of guaranteeing the aforementioned principles, both in the construction of the organization and, through an enhanced monitoring functionality, while it operates. The MOCA model we proposed here is more general: it does not explicitly include concepts like organization and role, but those concepts are still derivable. Principle one states that:

> (1) *All the collaborations subject to considerations of accountability among principals occur within a single scope called organization.*

Naturally, in order to properly identify relationships of accountability, all agreements, including expectations and declarations of control, must belong to the same scope to satisfy the subset constraints for accountability. The second and third principles concern the process of enrollment of an agent into an organization.

> (2) *A principal can enroll in an organization only by playing a role that is defined inside the organization.*

> (3) *A principal willing to play a role in an organization must be aware of all the powers associated with such a role before adopting it.*

Principle (3) adheres to definitions in [9], in which the ability to affect the institutional state is provided by the acquisition of some powers associated with the role itself at enactment time. The definition of a role delimits the range of actions an agent, who plays that role, can perform (i.e., its powers) inside the organization, thereby encoding the outcomes over which it has control. From the point of view of the ORM model, a role is essentially a set of achievements (i.e. outcomes and contexts) that are grouped together with a purpose of organizing work. Playing a role, therefore, means having control (through powers) over a set of outcomes associated with it, and being aware of the fact that there could be a social expectation related to their realization. The next principle states that:

> (4) *A principal is accountable, towards the organization or another agent, for those goals s/he has explicitly accepted to bring about.*

From the declaration of just (i.e., mutually agreed) expectation, in conjunction with the subset constraint from accountability, we can conclude that an agent must agree to a goal (through the expectation relationship) before that agent can be considered accountable. The last principle states that:

> (5) *A principal must have the leeway for putting before the organization the provisions s/he needs for achieving the goal to which s/he is committing. The organization has the capability of reasoning on the requested provisions and can accept or reject them.*

By specifying control, a principal stipulates contextual conditions under which he/she can realize an outcome. Without a principal's declaration of control, he/she cannot be held accountable thanks to the subset constraint and will not be considered committed to an outcome.

The works described in [2,3] are set in the context of normative MAS organizations. As the authors claim, and as pointed out in [13,15], normative organizations obfuscate accountability because they lack an explicit representation of the relationships between the agents inside the organization. Thus, a proposal is made to enrich MAS organizations, written in $\mathcal{M}$OISE [29], with an explicit representation of accountabilities and responsibilities that is coherent with our information model. In this way, the organizational model not only provides a structure for goal distribution but it becomes a means for coordinating the assumption of responsibilities done by the agents. Finally, the notion of *accountability fitting* is introduced to specify when a given responsibility distribution covers a set of accountability specifications. From an organization designer's perspective, the duties defined in responsibilities represent requirements with which the agents must comply. An accountability specification can be defined and verified with respect to a responsibility distribution independently of the actual agents that will play a given role.

6. Conclusions

Accountability is a central concept in many fields that study human interaction. It is studied by sociology (e.g., [22,25,36]), and psychologists provide evidence that accountability increases the salience of goals [37]. Ethnomethodologists postulate that social behavior is configured by relying on the same mechanisms through which it is explained and which indeed give meaning to social action, e.g. [25]. Management studies, e.g. [38], consider it a framework for managing expectations.

In this work, we have explained which data needs to be available in order to support the realization of accountability inside a a distributed system, like a MAS. This proposal is a necessary step in order to fill the lacks of current proposals that already exploit the notion of responsibility to capture the assignment of a task to an agent (e.g., [16,20,23,24]).

A relevant exception in this landscape is the ReMMo conceptual model by Feltus [24], which proposes a conceptualization of how responsibility is structured. Interestingly, this is also the first proposal in its field that brings into the picture the notion of accountability. ReMMo, in fact, defines responsibility as “a charge assigned to a unique actor to signify its accountabilities concerning a unique business task.” Still ReMMO leaves questions unanswered. In particular, it does not provide an information model that ties the concepts to the environment, which would capture constraints on how data evolve. Such a layer is necessary both for attributing responsibilities and, given a situation of interest, for identifying those who answer for it (forward-looking and backward-looking responsibility, according to [43]).

The relationships which, in MOCA, tie accountability, expectation, and control together, are versatile and can be used in many ways. The availability of accountability facts influences the behavior of the whole group of interacting agents, both those who play the role of account-givers and those who play the role of account-takers. For instance, a principal, knowing that another principal is accountable for some achievement, will draw expectations on the behavior of that principal, and on the control exercised by that principal, and it will orient its own behavior consequently. Another principal, knowing that he/she is accountable for an achievement, will likely consider the outcome as a goal when the related context holds. In general, we can say that accountability facts have an impact on future actions because they increase awareness, as explained in [1,37]. Moreover, we discuss in [8], by relying on a practical example, how the proposed model overcomes some weaknesses in tackling goal distribution in business processes.

In the future, it would be interesting to combine the accountability model with a representation of *compensations*, e.g. [42], that should be executed when an outcome is not achieved. Accountability in presence of interfering actions represents another promising and equally important area to investigate. Task interference poses just as important a piece to determining accountability as task completion. Future work will see the concept built from the information model presented here to obtain a whole-picture view.

Acknowledgements

This research was partially funded by Regione Piemonte grant number 320-30 “CANP - La Casa nel Parco” and by European Union’s Horizon 2020 research and innovation programme under grant agreement number 693514 “Towards #WeGovernment: Collective and participative approaches for addressing local policy challenges”.

References

[1] Paul A. Anderson. Justifications and precedents as constraints in foreign policy decision-making. *American Journal of Political Science*, 25(4):738–761, 1981.

[2] Matteo Baldoni, Cristina Baroglio, Olivier Boissier, Katherine M. May, Roberto Micalizio, and Stefano Tedeschi. Accountability and responsibility in agent organizations. In *PRIMA 2018: Principles and Practice of Multi-Agent Systems, 21st International Conference*, volume 11224 of *Lecture Notes in Computer Science*, pages 261–278. Springer, 2018.

[3] Matteo Baldoni, Cristina Baroglio, Olivier Boissier, Roberto Micalizio, and Stefano Tedeschi. Accountability and Agents for Engineering Business Processes. In R. H. Bordini, L. A. Dennis, and Y. Lesperance, editors, *Proc. of the 7th International Workshop on Engineering Multi-Agent Systems, EMAS 2019, held in conjuction with AAMAS 2019*, Montreal, Canada, May 13-14 2019.

[4] Matteo Baldoni, Cristina Baroglio, Katherine M. May, Roberto Micalizio, and Stefano Tedeschi. Computational Accountability. In F. Chesani, P. Mello, and M. Milano, editors, *Deep Understanding and Reasoning: A challenge for Next-generation Intelligent Agents, URANIA 2016*, volume 1802, pages 56–62, Genoa, Italy, December 2016. CEUR, Workshop Proceedings.

[5] Matteo Baldoni, Cristina Baroglio, Katherine M. May, Roberto Micalizio, and Stefano Tedeschi. ADOPT JaCaMo: Accountability-driven organization programming technique for JaCaMo. In *PRIMA 2017: Principles and Practice of*

Multi-Agent Systems, 20th International Conference, volume 10621 of *Lecture Notes in Computer Science*, pages 295–312. Springer, 2017.

[6] Matteo Baldoni, Cristina Baroglio, Katherine M. May, Roberto Micalizio, and Stefano Tedeschi. Supporting organizational accountability inside multiagent systems. In *AI*IA 2017 Advances in Artificial Intelligence*, volume 10640 of *Lecture Notes in Computer Science*, pages 403–417. Springer, 2017.

[7] Matteo Baldoni, Cristina Baroglio, Katherine M. May, Roberto Micalizio, and Stefano Tedeschi. Computational accountability in MAS organizations with ADOPT. *Applied Sciences*, 8(4):1–29, 2018. Special issue "Multi-Agent Systems".

[8] Matteo Baldoni, Cristina Baroglio, and Roberto Micalizio. Goal distribution in business process models. In *AI*IA 2018 – Advances in Artificial Intelligence*, volume 11298 of *Lecture Notes in Computer Science*, pages 252–265. Springer, 2018.

[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*, 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. *Science of Computer Programming*, 78(6):747–761, 2013.

[11] Matthew Braham and Martin van Hees. An anatomy of moral responsibility. *Mind*, 121(483):601–634, 2012.

[12] Brigitte Burgemeestre and Joris Hulstijn. *Handbook of Ethics, Values, and Technological Design*, chapter Design for the Values of Accountability and Transparency, pages 303–333. Springer, 2015.

[13] Amit K. Chopra, Fabiano Dalpiaz, Fatma Basak Aydemir, Paolo Giorgini, John Mylopoulos, and Munindar P. Singh. Protos: Foundations for engineering innovative sociotechnical systems. In *2014 IEEE 22nd International Requirements Engineering Conference (RE)*, pages 53–62. IEEE, 2014.

[14] Amit K. Chopra and Munindar P. Singh. The thing itself speaks: Accountability as a foundation for requirements in sociotechnical systems. In *2014 IEEE 7th International Workshop on Requirements Engineering and Law (RELAW)*, pages 22–22. IEEE, 2014.

[15] Amit K. Chopra and Munindar P. Singh. From social machines to social protocols: Software engineering foundations for sociotechnical systems. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 903–914, 2016.

[16] Rosaria Conte and Mario Paolucci. Responsibility for societies of agents. *Journal of Artificial Societies and Social Simulation*, 7(4):1–3, 2004.

[17] Stephen Cranefield, Nir Oren, and Wamberto Vasconcelos. Accountability for practical reasoning agents. In *Agreement Technologies 6th International Conference, AT 2018, Bergen, Norway, December 6-7, 2018, Revised Selected Papers*, volume 11327 of *LNAI*, pages 33–48. Springer, 2019.

[18] Stephen Darwall. *Morality, Authority, and Law: Essays in Second-Personal Ethics I*, chapter Civil Recourse as Mutual Accountability. Oxford University Press, 2013.

[19] Mehdi Dastani, Emiliano Lorini, John-Jules Meyer, and Alexander Pankov. Other-condemning anger = blaming accountable agents for unattainable desires. In *Proceedings of the 16th Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '17, pages 1520–1522. IFAAMAS, 2017.

[20] Mehdi Dastani and Vahid Yazdanpanah. Distant group responsibility in multi-agent systems. In *PRIMA 2016: Principles and Practice of Multi-Agent Systems*, volume 9862 of *Lecture Notes in Computer Science*, pages 261–278. Springer, 2016.

[21] Melvin J. Dubnick. Blameworthiness, trustworthiness, and the second-personal standpoint: Foundations for an ethical theory of accountability. Presented at EGPA Annual Conference, Group VII: Quality and Integrity of Governance. Edinburgh, Scotland, September 2013.

[22] Emile Durkheim. *De la division du travail social*. PUF, 1893.

[23] Andrew Eshleman. Moral responsibility. *The Stanford Encyclopedia of Philosophy*, 2014.

[24] Christophe Feltus. *Aligning Access Rights to Governance Needs with the Responsability MetaModel (ReMMo) in the Frame of Enterprise Architecture*. PhD thesis, University of Namur, Belgium, March 2014.

[25] Harold Garfinkel. *Studies in ethnomethodology*. Prentice-Hall, 1967.

[26] John C. P. Goldberg and Benjamin C. Zipursky. Torts as wrongs. *Texas Law Review*, 88, 2010.

[27] Ruth W. Grant and Robert O. Keohane. Accountability and Abuses of Power in World Politics. *The American Political Science Review*, 99(1):29–43, 2005.

[28] Terry Halpin and Tony Morgan. *Information Modeling and Relational Databases*. Morgan Kaufmann Publishers, 2008.

[29] Jomi F. Hubner, Jaime S. Sichman, and Olivier Boissier. Developing organised multiagent systems using the MOISE+ model: Programming issues at the system and agent levels. *International Journal of Agent-Oriented Software Engineering*, 1(3/4):370–395, 2007.

[30] Vladimir Lifschitz. What is answer set programming? In *Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 3*, AAAI'08, pages 1594–1597. AAAI Press, 2008.

[31] Elisa Marengo, Matteo Baldoni, Cristina Baroglio, Amit K. Chopra, Viviana Patti, and Munindar P. Singh. Commitments with regulations: reasoning about safety and control in REGULA. In *The 10th International Conference on Autonomous Agents and Multiagent Systems*, volume 2 of *AAMAS '11*, pages 467–474. IFAAMAS, 2011.

[32] Martin, Roland Kaminski, Benjamin Kaufmann, Max Ostrowski, Torsten Schaub, and Sven Thiele. A user's guide to gringo, clasp, clingo, and iclingo, 2010.

[33] Helen Nissenbaum. Accountability in a computerized society. *Science and Engineering Ethics*, 2(1):25–42, 1996.

[34] Office of the Auditor General of Canada. 2002 December Report of the Auditor General of Canada: Chapter 9, 2002. `http://www.oag-bvg.gc.ca/internet/English/parl_oag_200212_09_e_12403.html`.

[35] Office of the Auditor General of Canada. Exhibit 9.1 – The elements of accountability, 2002. `http://www.oag-bvg.gc.ca/internet/English/att_20021209xe01_e_12282.html`.

[36] Talcott Parsons. *The Structure of Social Action*. Collier-Macmillan, London, 1968.

[37] Andrew Quinn and Barry R. Schlenker. Can accountability produce independence? Goals as determinants of the impact of accountability on conformity. *Personality and Social Psychology Bulletin*, 28(4):472–483, 2002.

[38] Barbara S. Romzek and Melvin J. Dubnick. Accountability in the Public Sector: Lessons from the Challenger Tragedy. *Pub-*

lic Administration Review, 47(3):227–238, 1987.

[39] Barry R. Schlenker, Thomas W. Britt, John Pennington, Murphy Rodolfo, and Kevin Doherty. The triangle model of responsibility. *Psychological Review*, 101(4):632–652, 1994.

[40] Judith Simon. *The Onlife Manifesto: Being human in a hyperconnected era*, chapter Distributed Epistemic Responsibility in a Hyperconnected Era, pages 145–159. Springer Open, 2015.

[41] Lucy Suchman. *Discourse, Tools, and Reasoning: Essays on Situated Cognition*, chapter Centers of Coordination: A Case and Some Themes, pages 41–62. Springer, 1997.

[42] Amy Unruh, James Bailey, and Kotagiri Ramamohanarao. A framework for goal-based semantic compensation in agent systems. In *Safety and Security in Multiagent Systems*, volume 4324 of *Lecture Notes in Computer Science*, pages 130–146. Springer, 2009.

[43] Ibo van de Poel. *Moral Responsibility: Beyond Free Will and Determinism*, chapter The Relation Between Forward-Looking and Backward-Looking Responsibility, pages 37–52. Springer, 2011.