Modelling Institutional Enactment in Multi-Agent Systems with Enact-as Rules

Maiquel de Brito, Jomi F. Hübner

[1] Federal University of Santa Catarina, Brazil

{maiquel.b,jomi.hubner}@ufsc.br

Abstract. *In Multi-Agent Systems, constitutive rules specify institutional facts from environmental facts through the process of constitution. However, the opposite direction – the impact of institutional facts on the environment – has received little attention. This paper addresses this gap by proposing enact-as rules to specify environmental effects produced by institutional states. A formal model of enactment is presented together with a language for specifying enact-as rules and their execution semantics. The proposal has been implemented and integrated with institutional and environmental infrastructures, and its use is illustrated through an application example. The results show how institutional states can be explicitly connected to environmental effects, providing a dedicated mechanism for modelling the interaction between institutions and environments in Multi-Agent Systems.*

1. Introduction

Our departing position is a Multi-Agent System (MAS) composed of three dimensions: agents, environment, and institutions [Boissier et al. 2019]. *Institutions* gather all elements that represent and manage the social aspects of the system, including organisations, norms, and constitutive rules. While the interaction between agents and their environment (e.g. through action and perception) and between agents and their institutions (e.g. through norm compliance) are vastly investigated, the interaction between institutions and the environment is just partially explored. A few works explore the impact of the environment on the institution (e.g. through count-as rules). In the opposite direction, the impact of the institution on the environment, is almost neglected (Fig. 1). Most of the current approaches assume that agents are in charge of interpreting the institutional state and act upon the environment accordingly. For instance, an obligation (which is an institutional fact) stating that Alice has to open a door (which is a fact in the environment) is an institutional instrument to change the environment with the help of Alice. While that is a reasonable approach, some cases may require not relying on (or burdening) agents with such activities. For example, the institutional fact of someone violating a traffic law (e.g. by crossing a red traffic light) may require the triggering of the environmental facts of printing and mailing a fine. However, no one is fined if the application of the sanction is up to an agent that autonomously decide not to carry it out. We are thus interested in another approach where artificial institutions impact the environment directly, without the agents on the way. The initial inspiration for our investigation comes from some results on non-artificial systems [Hildebrandt 2008, Schulz and Dankert 2016].

As another example, we may consider a University with objectives as to progress in the state of the art of human knowledge and educate people. Besides these high level

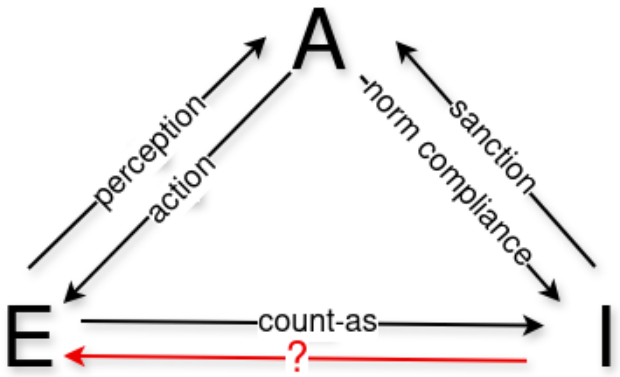

Figure 1. MAS dimensions and examples of concepts supporting their interactions. The research problem addressed in this work concerns the impact of institutions on the environment.

objectives, it also manages ordinary daily based objectives like register participants of classes, provide comfortable rooms for classes, etc. To achieve these objectives, it usually relies on tasks performed by the hands of agents, since *University* is an abstract entity with no concrete means to change the environment. For instance, a professor registers in a book the participants of his/her classes. To motivate agents to help with the University objectives, a common instrument is to define regulative norms: the professor is obliged to register participants or else s/he is subject to administrative sanctions. However, why burden agents with such ordinary and simple tasks that could be automated?

In this work, we investigate means to *automate* the achievement of institutional objectives when they are based on tasks that impact the environment and do not require any mediation of agents. A first research question is: how to specify the institutional feedback on environment? Since it involves both dimensions, it cannot be seen as a matter of either institutional or environmental specification alone. A second research question is: how is the dynamics of the interaction between institution and environment? Both dimensions have their own representations and dynamics, which must be conciliated with the dynamics of their interaction itself.

The core of our answers is to allow the specification of rules that define the consequence in the environment for some institutional state. We call these rules *enact-as* rules. As defined by [Searle 1997], the institutional state is composed of *institutional facts* and the environment is composed of *brute facts*. While institutional facts are built on top of brute facts, brute facts exist completely independently of human agreement or institutions. Enact-as rules aim at the creation of brute facts from institutional facts. An example of enact-as rule is "if a class is running, the air conditioner is on". In this example, "class is running" is an institutional fact and "air conditioner is on" is a brute fact. Its constitution may depend on several other brute or institutional facts. For instance, we may consider the following count-as rule: "an agent playing professor and n agents playing student ($n > 5$) in a university room *count-as* the class is running." With the enact-as rule, as soon as the institutional fact "class is running" is constituted, the environment is automatically changed by turning on the air conditioner (increasing the comfort of the participants), without agents intervention.

The contribution of the proposal is contextualised in Artificial Institutions for MAS, as introduced in Section 2. The fundamentals of our proposal are presented in Sec. 3, with a concrete implementation in a specification language presented in Section 4 that is illustrated by an example in Section 5. Sections 6 and 7 discuss the main features and options of the proposal and compare it with related work.

2. Artificial Institutions

In human societies, *institutions* assign statuses, with associated functions (called *status functions*), to brute facts. Institutions can state, for example, that a piece of paper counts as (or has the status function of) a dollar bill or that a person counts as a president [Searle 1997, Searle 2010]. The process of status function assignment is called *constitution*. It follows *constitutive rules*, that have the form "X counts as Y in C", meaning that a brute fact X carries the status function Y in a context C.[1]

The Situated Artificial Institution (SAI) model departs from Searle's ideas to provide means for specifying and deploying institutions in MAS [de Brito et al. 2018].[2] In SAI, constitutive rules are tuples (x, y, t, m) and are interpreted as follows: the brute fact x, which can be either an agent, an environmental event, or an environmental state, counts as the status function y when the event t occurs, conditioned on the holding of the state m.[3] The set of constituted status functions compose the *constitutive state*. The SAI model has an associated language for specifying constitution. Fig. 3 shows an example of specification. The meaning of this specification in a particular application is described in Section 5. Some relevant details of the language are described in the following:

- Constitutive rules have the form `x count-as y when t while m`, where `x`, `y`, `t`, and `m` have the aforementioned meaning.
- Status functions are assignable to agents, events, and states, and are declared in the clause `status_functions`.
- In Fig. 3, the clause `when` is omitted in all the constitutive rules, meaning that the specified constitution does not depend on any event. The clause `while` can also be omitted, meaning that the constitution holds in any state.
- Sentences in the form `X is Y` are true when `X` actually counts as `Y`.
- The constitutive rule 3 specifies the constitution of a *freestanding Y*, when status function is constituted without being assigned to any `X`.
- The constitutive state records pairs (x, y), meaning that x is counting as y. An example of a representation of the constitutive state is illustrated in Table 1.

According to Searle, "*institutions are systems of constitutive rules*" [Searle 2010]. The idea is that the constituted status functions support other abstractions that make societies work (e.g. obligations, rights, duties, etc.). SAI also assumes that the constitutive state supports artificial institutions. Other social abstractions (e.g norms, organisations,

[1] This formulation of constitutive rules summarises different ways through which constitution is established in human societies (e.g. laws, regulations, or even collective agreement).

[2] There are other works on MAS inspired by the count-as notion. But they do not consider explicitly the status functions as the element to be constituted by the constitutive rules (e.g. [Aldewereld et al. 2010, Boella and van der Torre 2004, Fornara et al. 2008, Viganò and Colombetti 2008, Cliffe et al. 2007, Cardoso and Oliveira 2007, Dastani et al. 2009, Campos et al. 2009, Piunti et al. 2010, de Brito et al. 2013]).

[3] The interpretation of constitutive rules in SAI is detailed in [de Brito et al. 2015].

etc.) run on top of the constitutive state [de Brito et al. 2019]. From this assumption, this work considers that an *institutional state* is the combination of a constitutive state and some other institutional components (e.g. normative state, organisational state, etc.).

3. A minimal model of Enactment Rules

Inspired by Searle's proposal of "X count-as Y in C", we propose to write *enact-as rules* as "`S` *enact-as* `X`". The informal semantics is that when the institution is in a state `S`, the brute fact `X` is expected to be produced in the environment. For instance, the rule "`ongoing class` *enact-as* `HVAC on`" means that if we have the institutional fact "ongoing class", the brute fact "HVAC on" should be produced.

Regarding the nature of S in "`S` *enact-as* `X`", it is a set of institutional facts representing an institutional state of the MAS relevant for the enact-as rule. This state is composed of constitutions produced by count-as rules, obligations produced by norms, sanctions, and other kinds of institutional facts. For example, if S is composed of (i) *ongoing class* and (ii) a person counting as a *student*, the enact-as rule is triggered in institutional states that hold these two institutional facts.

Regarding the nature of X in "`S` *enact-as* `X`", it refers to an *expected* brute fact in the environment after the rule is triggered. As actions used in agent plans or automatic planning [Ghallab et al. 2016], X represents a fact that is expected to be realised in the environment. In the previous example, this fact is that the HVAC is turned on. These components of the enact-as rule are formalised in definitions 1 to 4.

Definition 1 (Institutional state) *Let I^* be the set of all possible institutional facts, which represent properties of all the institutional components of the system; an institutional state $I \subseteq I^*$ is the set of the actually standing institutional facts.*

Definition 2 (Institutional literal) *Let $\bar{I} = I^* \cup \{\neg i \mid i \in I^*\}$ be a set of institutional literals, for any $p \in I^*$:*

$$I \models p \quad \textit{iff} \quad p \in I \tag{1}$$

$$I \models \neg p \quad \textit{iff} \quad p \notin I \tag{2}$$

Definition 3 (Enact condition) *The S in the enact-as rule is a set of institutional literals ($S \subseteq \bar{I}$). S holds in some institutional state I if all its literals hold:*

$$I \models S \quad \textit{iff} \quad \forall p \in S : I \models p \tag{3}$$

Definition 4 (Enact effect) *Let X^* be the set of all possible brute facts in the MAS, the X in the enact-as rule is one of them ($x \in X^*$).*

Definition 5 (Enact-as Rule) *An enact-as rule is a pair (S, x) where (i) $S \subseteq \bar{I}$ is the enact condition and (ii) $x \in X^*$ is an enact effect. The set of all enact-as rules is given by $\mathcal{E}$ ($\mathcal{E} \subseteq 2^{\bar{I}} \times X^*$).*

The interpretation of enact-as rules is based on the continuous evaluation of which rules are to be triggered. For triggered rules, the corresponding X value is delivered to an environment infrastructure that should be capable of producing the brute fact X in the environment. More precisely, the set of all enact effects from an institutional state I is given by the function E defined as follows:

$$E(I) = \{x \mid (S, x) \in \mathcal{E},\ I \models S\} \tag{4}$$

4. Enact-as Language

The minimal model provides the essential elements of enact-as rules. Using this model in practice requires to define a language to express its elements. Furthermore, it may be necessary to take some decisions required by the practical application of the model that are not the concern of its abstract presentation. This section introduces a language to specify enact-as rules and discusses the decisions taken to extend the minimal model towards practical applications.

From the grammar specified in Fig. 2, an enact program is a set $\mathcal{E}$ of enact-as rules. Each enact-as rule is an enact condition S, expressed through an *institutional formula*, connected to an enact effect x. An institutional formula is a logical formula composed of atoms and their negation, connected through the operators $\vee$ (`|`) and $\wedge$ (`&`) when needed.

$$
\begin{aligned}
\mathit{enact\_program}\ (\mathcal{E}) &::= \mathit{enact\_rule}+ \\
\mathit{enact\_rule}\ (S, x) &::= S\ \texttt{"enact-as"}\ x\ \texttt{"."} \\
S &::= \mathit{institutional\_formula} \\
x &::= \mathit{atom} \\
\mathit{institutional\_formula} &:: \mathit{atom} \mid \texttt{not}\ \ \mathit{institutional\_formula} \mid \\
&\quad\ \mathit{atom}\ (\texttt{"|"} \mid \texttt{"\&"})\ \mathit{institutional\_formula} \\
\mathit{atom} &::= \mathit{predicate}\ \ \texttt{"("}\ \ \mathit{terms}\ \ \texttt{")"}
\end{aligned}
$$

Figure 2. Grammar of the enact-as language in EBNF notation. $\mathit{predicate}$ **and** terms **are defined as usual in FOL.**

In the proposed language, institutional literals are represented by *predicates*, with the default syntax and semantics given by the first-order logic. The reasons for this decision are: (i) institutional facts usually establish properties of the domain elements (e.g. *running(classroom)*) or relations among them (e.g. *at(bob,room1)*); (ii) institutional facts produced by existing models and practical implementations are usually expressed by predicates (e.g. [Hübner et al. 2011, de Brito et al. 2019]); (iii) elements involved in these properties and relations may be replaced by variables in the enact-programs, enhancing their expressivity.

In the minimal model, the triggering condition of the enactment is represented by a set of institutional literals, implicitly interpreted as a conjunction (Definition 3). In the proposed language, it is represented by a logical formula, allowing the use of conjunction, disjunction, and negation. In the minimal model, disjunctions over the institutional state can be represented through a set of enact-as rules (e.g. the fact that *a or b enact x* can be expressed by the rules (a, x) and (b, x)). However, including the operator $\vee$ avoids writing multiple rules to represent disjunctions.

The interpretation of an enact-as program $\mathcal{E}$ results from the interpretation of its enact-as rules and the consequent creation of enact effects. The state of a running an enact-as program is a pair (B, E), where B is a set of institutional facts from I that has already been interpreted and E is a set of enact effects. The enact effects in E are recorded as pairs (S, x) where S is an enact condition representing the state that produced the enactment

and x is the enact effect. The initial state is $(\{\}, \mathsf{E})$ where

$$\mathsf{E} = \{(S\theta, x\theta) \mid (S, x) \in \mathcal{E}, I \models S\theta\} \tag{5}$$

and θ is a substitution of variables by values.

The state of an enact-as program evolves according to transition rules (6) and (7). In the first rule, when a new institutional fact i is in the institutional state I ($\mathsf{i} \in I$) but is not yet interpreted ($\mathsf{i} \notin \mathsf{B}$), it is added to B. The second rule removes facts from B when they are not in I any more. Besides, for both rules, the new set E of enact effects includes (i) those existing effects that are a consequence of a still holding institutional state, and (ii) new effects, produced by enact-as rules whose formula S, under a substitution θ, is true with respect to the new institutional state.

$$\frac{\mathsf{i} \in I \wedge \mathsf{i} \notin \mathsf{B}}{(\mathsf{B}, \mathsf{E}) \longrightarrow (\mathsf{B}', \mathsf{E}')} \tag{6}$$

$$\frac{\mathsf{i} \notin I \wedge \mathsf{i} \in \mathsf{B}}{(\mathsf{B}, \mathsf{E}) \longrightarrow (\mathsf{B}'', \mathsf{E}')} \tag{7}$$

s.t.:

$$\begin{aligned}
\mathsf{B}' &= \mathsf{B} \cup \{\mathsf{i}\} \\
\mathsf{B}'' &= \mathsf{B} \setminus \{\mathsf{i}\} \\
\mathsf{E}' &= \{(\mathsf{S}, \mathsf{x}) \mid (\mathsf{S}, \mathsf{x}) \in \mathsf{E}, I \models \mathsf{S}\} \cup \\
&\quad \{(S\theta, x\theta) \mid (S, x) \in \mathcal{E}, I \models S\theta\}
\end{aligned}$$

An interpreter for the proposed language has been developed and integrated with SAI for deployment in JaCaMo applications [Boissier et al. 2020]. In this integration, the constitutive state produced by the SAI execution is the institutional state considered by the enact-as language interpreter.[4] The execution of an enact-as program integrated with SAI is described in Section 5.

5. Application Example

Consider a system of virtual conference rooms that registers any participant as an *occupant*. Each virtual room may be used for a different purpose (e.g. business meetings, social meetings, lectures, etc.). In this system, the set of constitutive rules shown in Fig. 3 can turn a room into a *classroom*.[5] These rules define that (i) *bob* is the teacher (rule 1), (ii) anyone else in the room is a student (rule 2), (iii) the class is in session when there is a teacher and at least two students in the room (rule 3), and (iv) the event of class ending is produced when the agent counting as the *teacher* signs out of the room (rule 4). Additionally, classes supported by this system require that (i) student attendance is recorded; (ii) lecture notes are published after the session; and (iii) the room activity is recorded only while the class is in session. The enactment specification in Fig. 4 links the fulfilments of these requirements to the institutional facts produced by the constitutive rules.

```
status_functions:
   agents: student, teacher.
   events: finished(class).
   states: running(class).

constitutive_rules:
//the agent bob is the teacher of this room
1: bob
   count-as teacher
   while occupant(room,bob).

//any agent in the room is a student, except the teacher
2: Ag
   count-as student
   while occupant(room,Ag) & not(Ag is teacher) & Someone is teacher.

//a class is running when there is at least two students
3: count-as running(class)
   while Someone is teacher & Agent1 is student & Agent2 is student &
         Agent1 \== Agent2.

//the lecture ends when the teacher leaves the room
4: signOut[sai__agent(Agent)]
   count-as finished(class)
   while Agent is teacher.
```

Figure 3. Example of constitutive specification

```
//student attendance are registered when the class is in session
1: count_as(Agent, student) & running(class)
   enact-as registered(attendance,Agent).

//lecture notes are published when the class ends.
2: finished(class)
   enact-as published(lecture_notes).

//the activity is recorded when the class is in session
3: running(class)
   enact-as on(lecture_recording).

//there is not recording when the class is not in session
4: not(running(class))
   enact-as off(lecture_recording).
```

Figure 4. Example of enactment specification

Step	Brute Fact	Constitutive State (I)	Enact Effects (x from E)
0	(initial state)	{}	{off(lecture_recording)}
1	tom enters	{}	{off(lecture_recording)}
2	bob enters	{(bob,teacher), (tom,student)}	{off(lecture_recording)}
3	ana enters	{(bob, teacher), (tom,student), (ana, student), ($\perp$,running(class))}	{registered(attendance,tom), registered(attendance,ana), on(lecture_recording)}
4	bob signs out	{(signOut(bob), finished(class)}	{off(lecture_recording), published(lecture_notes)}

Table 1. Example of the execution of the system. In the constitutive state, a pair *(x,y)* means that *x* counts as *y*.

To illustrate the dynamics of the enact program, we can consider four steps, summarised in Table 1 and described in the following:

Step 0 Initially, there is no fact counting as a running class. Therefore, lecture recording remains off (enact-as rule 4).

Step 1 *tom* enters the room. Since no constitutive rule applies to this fact, the institutional state remains unchanged.

Step 2 *bob* enters the room. From constitutive rule 1, this fact constitutes *bob* as a *teacher*. Since there is a *teacher* in the room, *tom* counts as a *student* (constitutive rule 2).

Step 3 *ana* enters the room. From the constitutive rule 2, she counts as a *student*. In addition, the class is considered to be in session since there are two students (constitutive rule 3). This fact enacts student attendance registration (enact-as rule 1) and lecture recording (enact-as rule 3).

Step 4 *bob* leaves the room. This fact counts as the class ending (constitutive rule 4). This enacts the publishing of the lecture notes (enact-as rule 2). In addition, the institutional fact of *running(class)* no longer holds (constitutive rule 3), which enacts the halt of the lecture recording (enact-as rule 4). Finally, *bob* no longer counts as a *teacher* (constitutive rule 1), and neither *tom* and *ana* count as *students* (constitutive rule 2), but these facts do not enact any effect.

6. Related Work

The institutional feedback from software systems on the environment is often seen as a means of regimentation (i.e. preventing individuals from deviating from an expected behaviour) in works that do not focus on MAS. Two essential perspectives are identified: (i) "coding" norms in the environment; and (ii) implementing norms in a software system that manages the environment to constrain the behaviour of the individuals. The first case concerns the environment design, which includes the resources to limit the individuals' behaviour (e.g. a key locks a door to implement a norm forbidding everyone to

[4]Other institutional facts, such as normative and organisational facts, will be addressed in future work.

[5]Example available at `http://github.com/maiquelb/enact/tree/master/examples/smart_room`.

enter a room). The second case, aligned with this work, concerns the enactment design: software is designed to produce effects in the environment that restrict or replace human behaviour [Hildebrandt 2008, Schulz and Dankert 2016]. For example, a smart car could simply refuse to start if sensors detect that the driver is not properly buckled up.

The work by [Schulz and Dankert 2016] acknowledges that the institutional feedback may not only restrict individuals' behaviour but also replace it. In this direction, [Just and Latzer 2016] argues that algorithms can "construct the reality". For example, in search engines and social media platforms, algorithms build the available content, which influences the individuals' behaviour (even not constraining it). Our proposal is aligned with this perspective when it assumes that the enactment may produce facts that replace a human operator (e.g. managing the attendance registering, lecture recording, and lecture notes publishing in the example of Section 5).

In the MAS field, [Cunha et al. 2022] explores the relation between the institution and environment through the concept of *purpose*. A purpose expresses the influence of the brute facts that constitute status functions upon the environment. This differs from the enactment addressed in this work, which concerns environmental effects produced by constitution (and other institutional facts), rather than the environmental facts that produce the institutional facts. Environmental effects of the institutional facts are addressed by [Campos et al. 2009], which proposes *staff agents* capable of acting upon the environment according to the institutional state. Their actions are not limited to regimentation, but may encompass any behaviour supported by their reasoning and acting capabilities. Nevertheless, the institutional feedback remains subject to the autonomy of the staff agents. Enact-as rules are explicitly considered by [Piunti et al. 2010]. They are proposed at a structural level, connecting the dynamics of components implementing organisational infrastructures to the dynamics of the components that implement the environment. The works by [Campos et al. 2009, Piunti et al. 2010]) mix the specification and management of enactment with other MAS elements (agents in [Campos et al. 2009], and organisation and environment platforms in [Piunti et al. 2010]). Going a step further, our work addresses enactment as a distinct MAS element, with its own definitions, semantics, language, and dynamics. Having a general-purpose enact-as language, decoupled of any institutional and environmental infrastructure, makes it possible to specify enactment through proper abstractions, without mixing it with other MAS elements. Similarly, a dedicated enact-as interpreter enables enactment to be computed independently of the dynamics of institutional and environmental platforms.

7. Discussion

With our proposal, an MAS can achieve institutional objectives without relying on agents that comply with norms, thereby reducing their autonomy, since they can no longer choose not to comply. This reduction in autonomy can be tackled by agents themselves specifying enact-as rules. However, the agent power to obey or not a norm is not of the same nature as the power to change the enact-as rules. Thus, the initial solution is not suitable when we want to power the agents with autonomy to follow norms but not for changing them. Indeed, with the proposed approach we have the option to implement artificial institutions where agent's autonomy is reduced. This autonomy issues are thus a design decision and a feature that can be exploited by developers.

Another implication of our proposal is that we have a new element affecting the environment. Besides being changed by itself and agents, now the environment can be changed by the institution. While agents are autonomous to decide such changes, the institution follows clearly defined enact-as *rules*.

As an alternative for our proposal, one may question: why not simply implement the environment so that a brute fact (crossing the red traffic light) directly produces another brute fact (send fine)? What is the reason to place an institution between these two brute facts? Of course, some applications can be entirely implemented in the environment dimension. However, for applications that require or are better implemented with an institution, we can enforce institutional consistency with enact-as rules. Some environmental states are the consequence of the institutional state, these states are aligned and the former is justified by the latter. In the case where the institutional specification is modified, its consequences can be verified, since they are explicitly defined in enact-as rules. If those consequences were (implicitly) implemented entirely in the environment, this verification would be difficult to implement.

Another case that supports enact-as rules as proposed is, of course, when the conditions for the enactment include institutional facts that are created based on a complex institutional context that includes not only brute facts. For instance, the institutional fact `play(A,driver)`, used in an enact-as rule, might be constituted only if some agent enters into a car (a brute fact) and has a valid license (another institutional fact with potentially a complex constitution). Briefly, we can use enact-as rules when brute facts are not sufficient reasons to produce the enactment effects.

One could argue that enactment effects are the environmental counterparts of institutional facts, implying a tight correspondence between what occurs in the institution and what occurs in the environment. However, the proposed approach treats enactment as a *consequence* of institutional states rather than as a *counterpart* of individual institutional facts. This choice has some implications: (i) brute facts produced through enactment may result from the absence of institutional facts (e.g., `not S enact-as X`); (ii) brute facts may be consequences of complex institutional states involving conjunctions, disjunctions, and negations of institutional facts; (iii) there is no coupling between the nature of institutional facts and that of the resulting brute facts (for example, in SAI, the type of a constituted status function does not constrain the type of brute fact it may produce). The view of enactment as consequence also affects the integration of the enact-as interpreter with JaCaMo. Although enactment effects are formally dropped when their triggering institutional state ceases to hold (cf. Expression 7), actual revocation at the platform level is managed by a dedicated engine in the JaCaMo integration.

8. Conclusion

The main contribution of this work is to address the problem of the interaction between institutions and environments in MAS, complementing the count-as rules with enact-as rules. Regarding our research questions, the proposed answer is a new independent specification that states the consequences of some institutional states on the environment. We have an initial language for writing these specifications and an interpreter that monitors

the institutional states and computes brute facts that should be produced in the environment.

As future work, (i) we plan to further investigate the (lack of) participation of agents in the effects of institutions in the environment; (ii) investigate the use of the approach to implement sanctions as the result of norm violation; (iii) evaluate the proposal in more applications as well as its computational performance; (iv) investigate the nature of the element x of the enact-as rules; and (v) integrate the enact-as interpreter with other platforms in addition to JaCaMo.

References

Aldewereld, H., Álvarez-Napagao, S., Dignum, F., and Vázquez-Salceda, J. (2010). Making norms concrete. In van der Hoek, W., Kaminka, G. A., Lespérance, Y., Luck, M., and Sen, S., editors, *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems: volume 1-Volume 1*, pages 807–814, Toronto, Canada. International Foundation for Autonomous Agents and Multiagent Systems.

Boella, G. and van der Torre, L. W. N. (2004). Regulative and constitutive norms in normative multiagent systems. In Dubois, D., Welty, C. A., and Williams, M.-A., editors, *Principles of Knowledge Representation and Reasoning: Proceedings of the Ninth International Conference (KR2004), Whistler, Canada, June 2-5, 2004*, pages 255–266. AAAI Press.

Boissier, O., Bordini, R. H., Hübner, J., and Ricci, A. (2020). *Multi-Agent Oriented Programming: Programming Multi-Agent Systems Using JaCaMo*. MIT Press.

Boissier, O., Bordini, R. H., Hübner, J. F., and Ricci, A. (2019). Dimensions in programming multi-agent systems. *The Knowledge Engineering Review*, 34.

Campos, J., López-Sánchez, M., Rodríguez-Aguilar, J., and Esteva, M. (2009). Formalising Situatedness and Adaptation in Electronic Institutions. In Hübner, J., Matson, E., Boissier, O., and Dignum, V., editors, *Coordination, Organizations, Institutions and Norms in Agent Systems IV*, volume 5428 of *Lecture Notes in Computer Science*, pages 126–139. Springer Berlin Heidelberg, Berlin, Heidelberg.

Cardoso, H. L. and Oliveira, E. C. (2007). Institutional reality and norms: Specifying and monitoring agent organizations. *Int. J. Cooperative Inf. Syst.*, 16(1):67–95.

Cliffe, O., De Vos, M., and Padget, J. (2007). Answer set programming for representing and reasoning about virtual institutions. In Inoue, K., Satoh, K., and Toni, F., editors, *Computational Logic in Multi-Agent Systems*, volume 4371 of *Lecture Notes in Computer Science*, pages 60–79. Springer Berlin Heidelberg, Berlin, Heidelberg.

Cunha, R., Hübner, J. F., and de Brito, M. (2022). A conceptual model for situating purposes in artificial institutions. *Revista de Informática Teórica e Aplicada*, 29(1).

Dastani, M., Grossi, D., Meyer, J.-J., and Tinnemeier, N. (2009). Normative multi-agent programs and their logics. In Meyer, J.-J. and Broersen, J., editors, *Knowledge Representation for Agents and Multi-Agent Systems*, volume 5605 of *LNCS*. Springer.

de Brito, M., Hübner, J. F., and Boissier, O. (2015). Bringing constitutive dynamics to situated artificial institutions. In *Proc. of 17th Portuguese Conference on Artificial Intelligence (EPIA 2015)*, volume 9273 of *LNCS*, pages 624–637. Springer.

de Brito, M., Hübner, J. F., and Boissier, O. (2018). Situated artificial institutions: stability, consistency, and flexibility in the regulation of agent societies. *Autonomous Agents and Multi-Agent Systems*, 32(2):219–251.

de Brito, M., Hübner, J. F., and Boissier, O. (2019). Coupling the normative regulation with the constitutive state management in situated artificial institutions. *The Knowledge Engineering Review*, 34.

de Brito, M., Hübner, J. F., and Bordini, R. H. (2013). Programming institutional facts in multi-agent systems. In Aldewereld, H. and Sichman, J. S., editors, *Coordination, Organizations, Institutions, and Norms in Agent Systems VIII: 14th International Workshop, COIN 2012, Held Co-located with AAMAS 2012, Valencia, Spain, June 5, 2012, Revised Selected Papers*, volume 7756 of *LNCS*, pages 158–173. Springer.

Fornara, N., Viganò, F., Verdicchio, M., and Colombetti, M. (2008). Artificial institutions: a model of institutional reality for open multiagent systems. *Artificial Intelligence and Law*, 16(1):89–105.

Ghallab, M., Nau, D., and Traverso, P. (2016). *Automated Planning and Acting*. Cambridge University Press, Cambridge.

Hildebrandt, M. (2008). Legal and technological normativity: More (and less) than twin sisters. *Techné: Research in Philosophy and Technology*, 12(3).

Hübner, J. F., Boissier, O., and Bordini, R. H. (2011). A normative programming language for multi-agent organisations. *Annals of Mathematics and Artificial Intelligence*, 62(1-2):27–53.

Just, N. and Latzer, M. (2016). Governance by algorithms: Reality construction by algorithmic selection on the internet. *Media Culture & Society*, 39.

Piunti, M., Boissier, O., Hübner, J. F., and Ricci, A. (2010). Embodied organizations: a unifying perspective in programming agents, organizations and environments. In Fornara, N. and Vouros, G., editors, *Workshop on Coordination, Organization, Institutions and Norms in agent systems (COIN10@MALLOW)*, volume 627. CEUR.

Schulz, W. and Dankert, K. (2016). 'governance by things' as a challenge to regulation by law. *Internet Policy Review*, 5(2).

Searle, J. (2010). *Making the Social World:The Structure of Human Civilization*. Oxford University Press.

Searle, J. R. (1997). *The Construction of Social Reality*. Free Press.

Viganò, F. and Colombetti, M. (2008). Model checking norms and sanctions in institutions. In Sichman, J. S. a., Padget, J., Ossowski, S., and Noriega, P., editors, *Coordination, Organizations, Institutions, and Norms in Agent Systems III*, volume 4870 of *LNCS*, pages 316–329. Springer.