ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/283791364

Supporting Request Acceptance with Use Policies

Conference Paper · May 2014

DOI: 10.1007/978-3-319-25420-3_8

CITATIONS
4

READS
20

4 authors, including:

Thomas Christopher King
University of Oxford
11 PUBLICATIONS **34** CITATIONS
SEE PROFILE

Virginia Dignum
Delft University of Technology
312 PUBLICATIONS **3,556** CITATIONS
SEE PROFILE

Catholijn M. Jonker
Delft University of Technology
543 PUBLICATIONS **6,384** CITATIONS
SEE PROFILE

Some of the authors of this publication are also working on these related projects:

The Ninth International Automated Negotiating Agent Competition (ANAC 2018) View project

MSc Research View project

All content following this page was uploaded by Thomas Christopher King on 14 November 2015.

The user has requested enhancement of the downloaded file.

Supporting Request Acceptance with Use Policies

Thomas C. King, M. Birna van Riemsdijk, Virginia Dignum, and Catholijn M. Jonker

TU Delft, Delft, The Netherlands
{t.c.king-1, m.b.vanriemsdijk, m.v.dignum, c.m.jonker}@tudelft.nl

Abstract. This paper deals with the problem of automating the contribution of resources owned by people to do work for others, whilst providing a means for owners of resources to maintain autonomy over how, when and to whom their resources are used with the specification of use policies governing resources. We give representations of requests for resource usage as a set of conditional norms, and a use policy as specifying what norms should and should not be imposed on a resource, i.e. a set of meta-norms. Our main contribution is a reasoner built on the Event Calculus, that detects conflicts between requests and use policies, determining whether the request can be accepted.

1 Introduction

Increasingly, detailed environmental data is needed to support governments and citizens in making decisions affected by environmental conditions. Such decisions include determining where to go to avoid flooded areas or deciding how city water infrastructure can be improved based on its current effectiveness. Getting detailed data requires a large number of sensory resources, such as dense networks of precipitation sensors to monitor the rain.

Crowdsensing [9] is a means to cost-effectively acquire detailed data by requesting the use of a the mobile sensors people already own (i.e. crowdsourcing them), such as rain sensors on citizen's bicycles. We view a request for the use of a resource posed to its owner as a request for the owner to agree to the imposition of norms (obligations and prohibitions [1]) on the resource. An example of a request is for an ongoing agreement for a resource to be obligated to collect rain data when at a specific location and a prohibition from turning the sensor off until the data is collected. Given a large number of such agreements, detailed data can be gathered through the crowdsourcing of sensors.

However, if there are many requests for the use of a resource, feasibility demands the automation of their acceptance and rejection, whilst respecting an owner's desire to maintain autonomy over how, when and for whom their resource is used. Owners of resources need a means to specify a *use policy* governing how their resource may be used, and an automated process should reject requests if they conflict with the use policy. Whilst existing work detects conflicts between

norms (see [1]), such as when something is simultaneously obliged and prohibited, there lacks a way to detect conflicts between requests and use policies.

We address these issues by proposing a means to specify a use policy governing how a resource may be used, and an automated reasoner for rejecting requests if they conflict with the use policy. We view a use policy as specifying which norms should and should not be imposed. To exemplify, a use policy for a mobile sensor might state that anyone using it should be obliged to make a payment, and prohibit the prohibition of the sensor from being turned off. Given that a use policy obliges and prohibits the imposition of norms, we view it as a set of norms about norms, these are *meta-norms*.

Our proposed reasoner therefore detects conflicts between a request to use a resource and the meta-norms of a use policy that governs it, supporting the acceptance and rejection of requests. To detect conflicts we model norms and meta-norms in the Event Calculus [14], a logic of events and their effects over time. This allows us to determine if norms and meta-norms coincide and therefore detect if there are circumstances under which there are conflicts.

In the rest of the paper we first give an overview of existing work in the area (Section 2). Then, we give an overview of our approach (Section 3). In Section 4 we introduce the specification languages for requests and use policies and their informal meaning. The formal operational semantics are specified using the Event Calculus in Section 5. In Section 6 we illustrate the proposal with a formalisation of a running example. In Section 7 we describe an implementation of our proposal. We draw conclusions in Section 8.

2 Related Work

Our proposal fits into the broad area of normative multi-agent systems (see [1] for a recent literature survey), the formal study of which is deontic logic (see [8]). Much work has been done on reasoning about normative agreements such as contracts and compliance with norms (e.g. [5, 10, 18, 21, 22]) and multi-agent organisational frameworks for the verification of organisations as networks of contractual agreements (e.g. [7,12]). However, we focus on a pre-agreement stage, where the novelty of our proposal is the application of meta-norms to govern resource use.

There is already much work on reasoning about normative conflicts (for a review see [1]). In particular, Vasconcelos et al. [19] provide a normative conflict checking and resolution formalism based on logic and constraint programming. Unlike our work, they do not consider conflicts between norms and meta-norms, nor do they consider conditional norms and meta-norms about events. Instead, they detect normative conflicts between coinciding temporal obligations and prohibitions, where compliance with both is impossible. On the other hand, their approach does have an advantage in that it enables expressing norms with constraints such as an prohibition to stay within an area, and the detection of conflicting norms such as an obligation to be in a smaller part of that area.

Like our proposal, the work of Gunay et al. [11] uses the Event Calculus to detect normative conflicts. They consider conflicts between social commitments (norms bound to agents as a part of an agreement). Unlike our proposal, they do not consider meta-norms or conflicts between norms and meta-norms. Instead, they consider conflicts in terms of different simultaneous obligations to perform tasks that cannot cannot coincide. For example, two obligations to rent the same car to different people at the same time.

In reasoning about meta-norms, López et al. [15] propose a kind of legislative meta-norm to govern which norms an agent may issue or abolish. Their meta-norms define which norms can be introduced into the system and when. Unfortunately, they do not provide an implementation level mechanism for checking meta-norm/norm conflicts based on their operational semantics as we do.

Boella and Torre [3] also propose a kind of meta-norm acting as a permission issued by a higher authority to block the imposition of norms by lower level authorities. Using input/output logic (a logic of conditional norms [16]), their formalism produces for a given situation what may be obliged by lower-level authorities given everything that is permitted by higher-level authorities. The main difference with our work is that they give permissions the role of derogation and do not consider obligatory meta-norms. Furthermore, unlike in the aforementioned work, we detect conflict for norms with deadlines. Deadlines affect if norms can be simultaneously detached with meta-norms and thus cause conflict.

Meanwhile, Wansing [20] treats both obligatory and prohibitory meta-norms in deliberative-stit logic as being norms about the action of imposing norms. This is in a setting where there is a hierarchy of authorities, so for example, one authority obliges a lower authority to forbid an even lower authority. Like Wansing, we define a meta-norm as being about the event of a norm being imposed. We also explicitly represent norms as being from one authority to another, and meta-norms are implicitly so. Unlike Wansing's work, our work contributes a meta-norm/norm conflict detection mechanism.

In summary, there is much work on detecting conflicts between norms. There is also much work on reasoning about certain kinds of meta-norms. However, as far as we know there are no proposals for reasoning about conflicts between conditional norms and meta-norms about events.

3 Overview

Throughout the paper we consider a scenario where a municipality wants detailed statistics of rain hitting the ground near a newly built water square (an above-ground area that is both a recreational square, and a place to store rain water temporarily until there is capacity in the sewage system to handle the water[1]).

Assuming there is no dense network of stationary rain sensors in the area, the municipality might opt to make an ongoing agreement with the mobile sensors of people that frequent or pass by the square, such that whenever they are near

[1] http://www.raingain.eu/en/actualite/rotterdam-inaugurates-first-large-scale-square-water-storage-greenery-and-sport

the square they will collect and send rain data. For example, by recruiting users with an app on their mobile phone that communicates with rain sensors on their umbrella or bicycle to collect rain data, such that the sensors transmit the rain data they gather to their mobile phones which then send it to the requestor.

For simplicity we consider a process between two agents, depicted in Figure 1. These are, the resource provider agent governing a resource owned by a user and a requestor agent used by someone who wants to request use of the resource. The process begins with the user wanting data specifying the terms of the request, using the lexicon of a common ontology describing events, which their requestor agent then poses to the resource provider agent. The owner of the resource has specified the terms of its use policy, using the same common ontology of events.

The resource provider agent detects conflict between the use policy and request. This supports the automated governance of resources, so resource provider agent refuses requests that conflict with the use policy.

There are two types of conflict to consider. The first occurs if the request does not impose a norm in circumstances where the use policy obliges it to be imposed (e.g. an obligation to oblige a requestor to provide payment). The second type of conflict occurs if the request would impose norms in circumstances where they are prohibited by the use policy, (e.g. a prohibition on prohibiting the free movement of a resource).

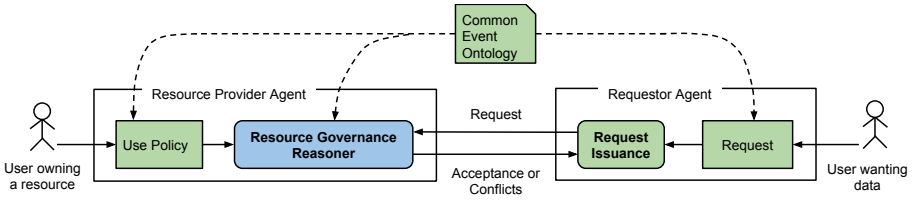

Fig. 1. Overview

In this paper, we focus on the reasoning about requests for resources governed by use policies, including the representations of requests and use policies. An overview of the resource governance reasoner architecture is given in Figure 2.

The resource governance reasoner we propose takes as inputs an ontology of events (what can happen), a request, and a use policy. The ontology of events is used to generate a sequence of events for the normative evaluator, in order to determine under which circumstances (after which events) which norms and meta-norms are detached simultaneously. Simultaneously detached norms and meta-norms are output as sets for the conflict detector. As a result, the reasoner returns the conflicts between a request and a use policy.

We model norms and meta-norms, detached and terminated according to a sequence of events, by using the Event Calculus [14] as the underlying formalism.

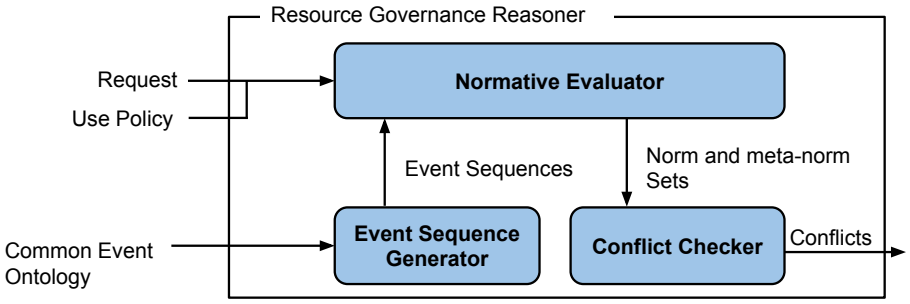

Fig. 2. The Architecture of the Reasoner

4 Requests and Use Policies

In this section we introduce the languages of requests as a set of norms and use policies as a set of meta-norms (4.1), then we proceed to discuss how norms and meta-norms relate to each other in terms of conflicts (4.2).

4.1 Representations

We use requests to represent what a requestor would like a provider to agree to doing and a use policy to govern what a provider agent will agree to. We begin with a motivating example to motivate our representations of requests.

Example. (Part One, Requestor) Rachel wants to monitor the level of rainfall around a newly built water square. There is no dense network of static sensors, so she wishes to make use of the existing mobile sensors people carry to gather the data. Thus, she requests people to make an ongoing agreement to provide her with precipitation data. The terms of the request are that whenever someone enters the water square, rain data should be gathered, once data is gathered it should be sent before the sensor leaves the area around the station, and the sensor should not be turned off until data has been gathered.

Norms concern agents, so we assume a set of agent names Ag. Norms are about events, which can be generated by or concern specific agents (e.g. an agent entering an area), or the environment (e.g. rain starting to fall). Thus, events are either non-agentive, denoted as propositions, or agentive denoted as propositions with an agent name in the subscript.

Definition 1. *(Events) Let EnvProp with typical element EP, and AgProp with typical element AP be mutually disjoint sets of propositions, respectively denoting non-agentive and agentive events. Also let Ag be a set of agent names with typical element a. The set Ev is the set of all elements ev expressible in the language:*

$$ev ::= EP \mid AP_a$$

Norms are obligations and prohibitions, respectively denoted with a deontic type of O or F. Norms oblige or prohibit an agent to ensure an aim A happens (an event) possibly before a deadline D (also an event).

A norm can be detached (become active) [1], either unconditionally or on the condition of an event happening. If a norm's detachment is conditional on an event, we say it is a conditional norm, otherwise it is an unconditional norm. Conditional norms are represented as a rule with the conditional event C placed in the antecedent and a norm as the consequent.

Finally, we follow the notion that when a norm is detached it is imposed on a debtor denoted with *DE*, towards a creditor denoted with *CR*. For example, a debtor can be an agent obliged to provide data within one minute towards a creditor requesting it, or a debtor can be an agent obliged to pay for data towards an agent providing the data.

Definition 2. *(Norm) Let O and F respectively denote the deontic types of obligation and prohibition, $DE, CR \in Ag$ be debtor and creditor agents, and $C, A, D \in Ev$ be events respectively denoting the condition, aim and deadline of the norm. An unconditional norm is denoted with $\langle ucn \rangle$ and a conditional norm is denoted with $\langle cn \rangle$. The set of norms N is the set of all elements $\langle n \rangle$ expressible in the language defined as:*

$$
\begin{array}{lcl}
\langle n \rangle & ::= & \langle cn \rangle \mid \langle ucn \rangle \\
\langle cn \rangle & ::= & C \ \mathit{THEN} \ \langle ucn \rangle \\
\langle ucn \rangle & ::= & O_{DE:CR}(A \ \mathit{BEFORE} \ D) \mid F_{DE:CR}(A \ \mathit{BEFORE} \ D) \mid \\
 & & O_{DE:CR}(A) \mid F_{DE:CR}(A)
\end{array}
$$

A request to use a resource, is the set of norms that if the request is accepted will be imposed on the resource.

Definition 3. *(Request) A request R is a set of norms such that $R \subseteq N$.*

Thus we can formalise the request R of Rachel (ra), to a resource owned by a person called Peter (pe), as:

$$
\begin{array}{lll}
R = \{ & \mathit{enter\_water\_square}_{pe} \ \mathit{THEN} \ O_{pe:ra}(\mathit{gather\_rain\_data}_{pe} & \\
 & \qquad \mathit{BEFORE} \ \mathit{send\_data}_{pe}), & \\
 & \mathit{enter\_water\_square}_{pe} \ \mathit{THEN} \ O_{pe:ra}(\mathit{send\_data}_{pe} & \\
 & \qquad \mathit{BEFORE} \ \mathit{leave\_water\_square}_{pe}), & \\
 & \mathit{enter\_water\_square}_{pe} \ \mathit{THEN} \ F_{pe:ra}(\mathit{sensing\_off}_{pe} & \\
 & \qquad \mathit{BEFORE} \ \mathit{gather\_rain\_data}_{pe}) \} &
\end{array}
$$

Use policies are used to govern a resource, such that requests a resource owner does not want to be automatically accepted will be detected as conflicting with the request, and rejected. We use the following example to motivate the expressivity of our use policy representation.

Example. (Part Two, Provider) Peter has a cellphone app that can measure rainfall in a location by communicating using bluetooth with a sensor on his umbrella. Peter usually carries his cellphone and umbrella with him, including when he walks past the water square. He is willing to donate the use of his sensing resource, but only if he does not have to stay in any particular area. His cellphone uses up a lot of energy when collecting data, so he wants to be allowed to turn the sensing off when the battery's energy becomes low. Finally, if he is obliged to provide data, then he wants to be paid within one day.

We represent a use policy as a specification of what norms should and should not be detached under specific circumstances. That is, it is a set of norms about norms, or meta-norms. This specification can be compared against a request.

Meta-norms can be conditional on an event or a norm being detached. For example, on the condition an obligation for data to be provided is detached, then it is obligatory for the agent that should be sent data to provide payment.

An unconditional meta-norm is detached by default, whilst a conditional meta-norm is detached on the condition an event occurring, or a norm specified in the condition is detached. Both conditional and unconditional meta-norms obligate or prohibit the detachment of a norm. For brevity we do not examine the case where a meta-norm has a deadline, but our framework can easily be extended to accommodate for this.

Definition 4. *(Meta Norms) Let $n, n' \in N$ denote norms, and $C \in Ev$ be an event. An unconditional meta-norm is denoted with $\langle ucmn \rangle$ and a conditional meta-norm is denoted with $\langle cmn \rangle$. The set MN is the set of all elements $\langle mn \rangle$ expressible in the language defined as:*

$$\begin{aligned} \langle mn \rangle &::= \langle cmn \rangle \mid \langle ucmn \rangle \\ \langle cmn \rangle &::= n \ THEN \ \langle ucmn \rangle \mid C \ THEN \ \langle ucmn \rangle \\ \langle ucmn \rangle &::= O(n') \mid F(n') \end{aligned}$$

A use policy is a set of meta-norms.

Definition 5. *(Use Policy) A use policy $UP \subseteq MN$ is a set of meta-norms.*

To exemplify, we formalise Peter's use policy *UP* which specifies how Rachel may use his resource, where his resource is denoted with *pe*, as[2]:

$$\begin{aligned} UP = \{ & O_{pe:ra}(send\_data_{pe} \ BEFORE \ leave\_water\_square) \ THEN \\ & \qquad O(O_{ra:pe}(pay_{ra} \ BEFORE \ tomorrow)), \\ & F(O_{pe:ra}(send\_data_{pe} \ BEFORE \ leave\_water\_square_{pe})), \\ & battery\_depleted_{pe} \ THEN \ F(F_{pe:ra}(sensing\_off_{pe} \ BEFORE \\ & \qquad gather\_rain\_data_{pe}))\} \end{aligned}$$

[2] More general use policies are possible by extending the framework with variables. This would allow non-specific debtors and creditors to be specified, and the expression of meta-norms about norms where we are not concerned with exact terms.

4.2 Conflict

Taking Rachel's request and Peter's use policy, we can intuitively see there are conflicts, which should be identified by the resource governance reasoner.

Example. (Part Three, Conflict) Rachel's request has been posed to Peter's agent in control of providing the resource. Rachel wants data sent before Peter's sensor leaves the water square area, however, Peter has stated that he wants to be free to move. If Rachel's request is accepted then Peter will be prohibited from turning his sensor off, yet, he has stated that when his cellphone's battery is depleted he must be allowed to do so. Finally, once Peter has fulfilled an obligation to provide data, he demands to be paid before tomorrow, but Rachel's request does not include such an obligation.

The normative reasoner identifies sets of norms and meta-norms that hold simultaneously given some circumstances (a sequence of events), a request, and a use policy. Meanwhile the conflict-detector takes as input sets of simultaneously holding norms and meta-norms and identifies conflicts by comparing those sets.

The conflict detector identifies two types of conflict. The first type of conflict is where a meta-norm holds that obliges a norm to be detached, but that norm is not detached. Conversely, the second type of conflict is where a meta-norm holds that prohibits a norm from being detached, but that norm is detached. We define conflict in terms of a set of norms and meta-norms that hold simultaneously. We assume the sets of norms and meta-norms to be self-consistent and focus on conflicts between norms and meta-norms.

Definition 6. *(Conflict) Let $N' \subset N$ be a set of unconditional norms and $MN' \subset MN$ be a set of unconditional meta-norms, denoting all of the simultaneously detached norms and meta-norms for some circumstances. We say that there is a conflict between N' and MN' if either of the following holds:*

$$(n \in N' \text{ and } F(n) \in MN') \text{ or } (n \notin N' \text{ and } O(n) \in MN')$$

To exemplify, consider a sequence of events generated by the event sequence generator component, that can hypothetically happen after the request is accepted, where Peter enters the water square and then the battery on his phone is depleted. For this situation, assuming nothing else has happened, the sets *DN* and *DMN* are the respective sets of simultaneously detached norms from Rachel's request and meta-norms from Peter's use policy:

$$\begin{aligned} DN = \{&F_{pe:ra}(sensing\_off_{pe} \ BEFORE \ gather\_rain\_data_{pe})\} \\ DMN = \{&O(O_{ra:pe}(pay_{ra} \ BEFORE \ tomorrow), \\ &F(F_{pe:ra}(sensing\_off_{pe} \ BEFORE \ gather\_rain\_data_{pe})), \\ &F(O_{pe:ra}(send\_data_{pe} \ BEFORE \ leave\_water\_square_{pe}))\} \end{aligned}$$

Both types of conflict are identified in this example when these sets of norms and meta-norms are compared. The first, is that a norm forbids turning the

sensing off, yet a meta-norm is simultaneously detached that forbids such a prohibition. Similarly, a norm obliges Peter to send data before he leaves the water square, but such an obligation is prohibited by a detached meta-norm. Finally, a meta-norm obliges the obligation for payment to be provided, but such an obligation is not detached at the same time.

5 The Event Calculus Normative Model

In this section we give the operational semantics for the detachment and termination of norms and meta-norms, and when they produce conflicts between a use policy and a request. We first re-introduce the Event Calculus (Section 5.1) which we subsequently use to define the operational semantics (Section 5.2).

5.1 Event Calculus

The Event Calculus is a logical-formalism specified by Kowalski and Sergot [14] for reasoning about events and their effects on which fluents hold and when. The Event Calculus provides an ontology of predicates (Table 1) for specifying in an Event Calculus theory the effects of events on initializing and terminating fluents, and what events happen at which time *points* (a narrative). The same ontology also provides predicates that specify, given the Event Calculus, an Event Calculus Theory and a narrative, which fluents hold at specific time *intervals*.

Predicate	Meaning
broken_during(*P, Start, End*)	P is terminated between time points *Start* and *End*.
happens_at(E, T)	The event E happens at time T.
holds_at(*P, T*)	The property P holds at time T.
holds_for(*P, Start, End*)	The property P holds from time points *Start* until *End*.
initially(P)	The property P holds at the first time point.
initiates_at(*E, P, T*)	The event E initiates the property P at time T.
terminates_at(*E, P, T*)	The event E terminates the property P at time T.

Table 1. The Event Calculus ontology of predicates

We choose the Event Calculus due to its modelling of inertial fluents and its efficient implementations [2, 4, 6]. Inertial fluents are required because we treat norms and meta-norms as fluents, and the informal notions of detached norms and meta-norms mean they continue to be detached until either their aim or deadline occurs. Efficiency is important, due to the time constraints that can be expected when accepting or rejecting a request.

Although many variations of the Event Calculus exist [17] we use the *simple* Event Calculus, where from here-on we usually omit the word simple. In the following, we give an axiomatisation of the Event Calculus adapted from [6] with the addition of a commonly used axiom for an initial state.

Axioms are given as Prolog style horn-clauses. Keeping with convention, symbols starting with upper-case denote variables and lower-case denote constants. Since the Event Calculus explicitly deals with time, we assume an infinitely countable set of time instances $\mathbb{T}$ with typical element t_i where $i \in \mathbb{N} \cup \{\infty\}$. The operators $<$ and $\leq$ are assumed to be specified for all members of the set $\mathbb{T}$, with the expected meaning. Finally, $\neg$ is interpreted as negation-as-failure, making the Event Calculus non-monotonic.

The first axiom, *ec1*, states that any fluent stated to initially hold is initiated at the first time point.

$$\mathit{initiates\_at}(\mathit{initially}(P), P, 0) \leftarrow \mathit{initially(P)} \tag{ec1}$$

The next two axioms specify the intervals fluents hold for. Axiom *ec2* states that a fluent holds in an interval beginning immediately after the initiation event and ending at the termination event. Axiom *ec3* deals with the case where there is no terminating event for a fluent.

$$\begin{aligned} \mathit{holds\_for}(P,\ \mathit{Start},\ \mathit{End}) \leftarrow\ & \mathit{initiates\_at}(\mathit{Ei},\ P,\ \mathit{Start}) \wedge \\ & \mathit{terminates\_at}(\mathit{Et},\ P,\ \mathit{End}) \wedge \\ & \mathit{End} > \mathit{Start} \wedge \\ & \neg \mathit{broken\_during}(P,\ \mathit{Start},\ \mathit{End}) \end{aligned} \tag{ec2}$$

$$\begin{aligned} \mathit{holds\_for}(P,\ \mathit{Start},\ t_\infty) \leftarrow\ & \mathit{initiates\_at}(\mathit{Ei},\ P,\ \mathit{Start}) \wedge \\ & \neg \mathit{broken\_during}(P,\ \mathit{Start},\ t_\infty) \end{aligned} \tag{ec3}$$

Axiom *ec4* states that a fluent is broken during an interval if a terminating event occurs during that interval. We specify the axiom such that it provides a 'weak-interpretation' of the *initiates_at* predicate [6], where the same initiation event occurring consecutively does not imply there was a terminating event in-between.

$$\begin{aligned} \mathit{broken\_during}(P,\ \mathit{Start},\ \mathit{End}) \leftarrow\ & \mathit{terminates\_at}(E,\ P,\ T) \wedge \\ & \mathit{Start} < T < \mathit{End} \end{aligned} \tag{ec4}$$

Finally, axiom *ec5* states which time points a fluent holds at.

$$\mathit{holds\_at}(P,\ T) \leftarrow \mathit{holds\_for}(P,\ \mathit{Start},\ \mathit{End})\ \wedge \mathit{Start} < T \leq \mathit{End} \tag{ec5}$$

Given the Event Calculus specification, the effects of events can be specified using the schemas *ec6* for the initialisation of a fluent and *ec7*, taken from [5].

$$\begin{aligned} \mathit{initiates\_at}(E,\ P,\ T) \leftarrow\ & \mathit{happens\_at}(E,\ T)\ \wedge \mathit{holds\_at}(P_1, T) \\ & \wedge \ldots\ \wedge \mathit{holds\_at}(P_n, T) \end{aligned} \tag{ec6}$$

$$\begin{aligned} \mathit{terminates\_at}(E,\ P,\ T) \leftarrow\ & \mathit{happens\_at}(E,\ T)\ \wedge \mathit{holds\_at}(P_1, T) \\ & \wedge \ldots\ \wedge \mathit{holds\_at}(P_n, T) \end{aligned} \tag{ec7}$$

5.2 Normative Evaluation and Conflict Checking

Our normative evaluation and conflict checking semantics uses the Event Calculus for reasoning about which norms and meta-norms hold when, and when they conflict. The two resource governance reasoner components (see Figure 2), the Normative Evaluator and the Conflict Checker, are defined as sets of Event Calculus rules.

In the following, we use the predicates $o/4$ and $f/4$ to respectively represent obligations and prohibitions, with the first two parameters respectively being the debtor and creditor, the third the aim, and the fourth the deadline event or $\perp$ to indicate no deadline. $o/1$ and $f/1$ are predicates representing meta-norms, where the parameter is a norm. We use the predicate *ifthen*/2 to represent conditional and unconditional norms and meta-norms, the first parameter is the condition or $\top$ if it is unconditional, the second parameter is the norm or meta-norm.

As with work on social commitment modelling [5] we assume that two events cannot occur at the same time. However, we make an exception for the event of a norm being detached, which can often occur at the same time as a non-detachment event and other norms being detached.

Normative Operational Semantics. The operational semantics of norms and meta-norms correspond to the Normative Evaluator component (see Figure 2), specifying when a norm or meta-norm is and is not detached. The semantics are specified with axioms for the *initiates_at*/3 and *terminates_at*/3 Event Calculus predicates, which state an event (the first term) respectively detaches or terminates a norm or meta-norm (the second term) when the event happens (the last term). These axioms are defined with respect to the *happens_at*/2 predicate which describes when an event happens.

The first two axioms state that any unconditional norm holds initially.

$$\mathit{initially}(o(\mathit{DE},\ \mathit{CR},\ \mathit{A},\ \mathit{D})) \leftarrow \mathit{ifthen}(\top, o(\mathit{DE},\ \mathit{CR},\ \mathit{A},\ \mathit{D}))$$
(Obl. Uncond. Norm Detachment)

$$\mathit{initially}(f(\mathit{DE},\ \mathit{CR},\ \mathit{A},\ \mathit{D})) \leftarrow \mathit{ifthen}(\top, f(\mathit{DE},\ \mathit{CR},\ \mathit{A},\ \mathit{D}))$$
(Pro. Uncond. Detachment)

Obligatory meta-norms have different detachment semantics from norms. An unconditional obligatory meta-norm is detached initially only if it is not simultaneously satisfied with the detachment of a norm. We do not check if the norm it obliges is *already* detached, although this is certainly possible we take the meaning of an obligatory meta-norm to be that it obliges the detachment of a norm at the time it is itself detached. If the obligatory meta-norm is satisfied as soon as it is detached, then there is no conflict and so it will not be taken into account by the conflict checker.

$$\mathit{initially}(o(\mathit{Norm})) \leftarrow \mathit{ifthen}(\top, o(\mathit{Norm})) \wedge \neg \mathit{initially}(\mathit{Norm})$$
(Obl. MN Uncond. Detachment)

Unconditional prohibitory meta-norms, however, are detached regardless of whether the norm they prohibit is detached. Thus, their detachment follows the same form as norms, formulated in the next axiom.

$$initially(f(Norm)) \leftarrow ifthen(\top, f(Norm)) \qquad \text{(Pro. MN Uncond. Detachment)}$$

The next two axioms, give the conditional detachment of norms, stating that when the condition of a conditional norm happens, the norm is detached.

$$initiates\_at(C, o(DE,\ CR,\ A,\ D), T) \leftarrow ifthen(C, o(DE,\ CR,\ A,\ D)) \wedge happens\_at(C, T)$$
(Obl. Cond. Detachment)

$$initiates\_at(C, f(DE,\ CR,\ A,\ D), T) \leftarrow ifthen(C, f(DE,\ CR,\ A,\ D)) \wedge happens\_at(C, T)$$
(Pro. Cond. Detachment)

The next axiom states that a conditional obligatory meta-norm is detached when its condition occurs, unless it is satisfied at the same time with the detachment of the norm it obliges (as with its unconditional variant). Again, if the obligatory meta-norm is detached and simultaneously satisfied, the conflict checker will not take it into account, because there is no conflict.

$$initiates\_at(C1, o(Norm), T) \leftarrow ifthen(C1, o(Norm)) \wedge happens\_at(C1, T) \wedge \neg initiates\_at(C2,\ Norm,\ T)$$
(Obl. MN Cond. Detachment)

Conditional prohibitory meta-norms, have the same detachment semantics as conditional norms.

$$initiates\_at(C, f(Norm), T) \leftarrow ifthen(C, f(Norm)) \wedge happens\_at(C, T)$$
(Pro. MN Cond. Detachment)

We treat the detachment of a norm as an event, the event of the norm being imposed on an agent. This is required for the detachment of meta-norms that are conditional on the event of a norm being detached and the satisfaction of obligatory meta-norms which is the event of a norm being detached.

$$happens\_at(o(DE,\ CR,\ A,\ D),\ T) \leftarrow ifthen(C, o(DE,\ CR,\ A,\ D)) \wedge happens\_at(C, T)$$
(Obl. Detachment Event)

$$happens\_at(f(DE,\ CR,\ A,\ D),\ T) \leftarrow ifthen(C, f(DE,\ CR,\ A,\ D)) \wedge happens\_at(C, T)$$
(Pro. Detachment Event)

We now turn our attention to the termination of detached norms. A detached obligation is terminated if its aim is achieved, whilst a detached prohibition is

terminated if its deadline occurs. Thus, although we do not explicitly model violations, under these semantics a norm persists after it is violated until it is fulfilled. Alternative semantics can be accommodated for in the future.

$$terminates\_at(A, o(DE,\ CR,\ A,\ D), T) \leftarrow happens\_at(A, T) \quad \text{(Obl. Aim Termination)}$$

$$terminates\_at(D, f(DE,\ CR,\ A,\ D), T) \leftarrow happens\_at(D, T) \quad \text{(Pro. Deadl. Termination)}$$

As with obligatory norms, obligatory meta-norms are terminated when their aim (the detachment of a norm) occurs:

$$terminates\_at(Norm, o(Norm), T) \leftarrow happens\_at(Norm,\ T) \quad \text{(Obl. MN Aim Termination)}$$

Prohibitory meta-norms, like their norm counterparts, are not terminated when their aim occurs (a norm they prohibit is detached). Thus, due to not having a deadline, they are not terminated at all.

Conflict Detection Semantics. The conflict detection semantics correspond to the Conflict Checker component (see Figure 2). The semantics are given as axioms for the predicate *conflict*/3, which states a meta-norm is causing conflict (the first term), from when (the second term) and until when (the last term). As conceptually defined in Definition 6, conflict is determined based on which norms and meta-norms hold for the same period of time, this is given by the results of the Normative Operational Semantics.

The first type of conflict occurs when a meta-norm obliges the detachment of a norm and that norm is not detached. If this is the case, then the obligatory meta-norm will hold for some time until it is satisfied. If the obligatory meta-norm holds, then the norm it obliges was not detached at the same time or subsequently and so there is a conflict.

$$conflict(o(Norm), Start, End) \leftarrow holds\_for(o(Norm), Start, End) \quad \text{(Obl. MN Conflict)}$$

The final axiom states that given two overlapping intervals where a norm holds and a prohibitory meta-norm holds, there is a conflict if the norm is prohibited by the meta-norm. We assume two predicates $minimum(T, T', Min)$ and $maximum(T, T', Max)$ for defining when two periods of time, $T, T' \in \mathbb{T}$, overlap. The predicate *minimum*/3 holds iff *Min* is the minimum of the two time points, and the predicate *maximum*/3 holds iff *Max* is the maximum.

$$\begin{aligned} conflict(f(Norm), Start, End) \leftarrow & holds\_for(f(Norm), MNStart, MNEnd) \wedge \\ & holds\_for(Norm, NStart, NEnd) \wedge \\ & maximum(NStart, MNStart, Start) \wedge \\ & minimum(NEnd, MNEnd, End) \ \wedge Start < End \end{aligned} \quad \text{(Pro. MN Conflict)}$$

6 Illustration

In this section we illustrate how our formalism works for the running example of Rachel's request and Peter's use policy. First, we assume the following narrative is produced by a sequence generator component (to be sure all conflicts are detected, all possible event sequences would need to be checked):

$$E1 = happens\_at(enter\_water\_square_{pe}, 1), E2 = happens\_at(battery\_depleted_{pe}, 2),$$
$$E3 = happens\_at(gather\_rain\_data_{pe}, 3), E4 = happens\_at(send\_data_{pe}, 4)$$

From this narrative, we can infer the following norm detachment events:

$$E5 = happens\_at(O_{pe:ra}(gather\_rain\_data_{pe}\ BEFORE\ send\_data_{pe}), 1),$$
$$E6 = happens\_at(O_{pe:ra}(send\_data_{pe}\ BEFORE\ leave\_water\_square_{pe}), 1),$$
$$E7 = happens\_at(F_{ra:pe}(sensing\_off_{pe}\ BEFORE\ gather\_rain\_data_{pe}), 1)$$

Given these events, Figure 3 depicts which norms, meta-norms and conflicts hold and when.

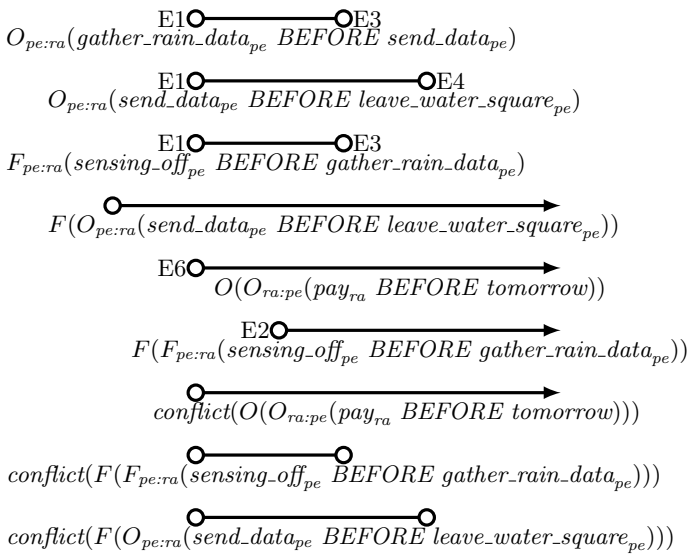

Fig. 3. An example with several conflicts. — indicates the interval a fluent holds for, O—O is a terminated interval, whilst O→ is an interval that continues forever.

As we intuitively expect, there is a conflict because Peter is obliged to send data, but Rachel is not obliged to pay him. Another conflict occurs because Peter is forbidden from turning his device off, but because his phone's battery

has become low he wants to maintain this right. Finally Peter is obliged to send data before leaving the water square, but such an obligation is forbidden.

7 Implementation

Our proposal is implemented [13] for a prototype of a crowdsensing system used to gather accurate rain data with user's mobile sensing devices (such as rain sensors on bicycles) in a simulated environment. In the prototype we simulate users using NetLogo[3] that can both request other users to gather rain data and form an ad-hoc network to help transmit the data, and provide rain data and participate in an ad-hoc network to help transmit the data.

Each user in the system has a use policy governing their resource and a resource governance reasoner, implemented in Prolog, for the automated acceptance and rejection of requests for the use of their resource on the basis of its use policy. Our implementation uses pre-formulated requests (sets of norms) for users to send to others and provides a graphical user interface for the editing of Use Policies governing the devices of the individual simulated users.

The proposal in this paper closely corresponds to our implementation. Two Definite Clause Grammars (DCGs) are specified corresponding to the formal definition of the norm and meta-norm specification languages defined earlier, with an appropriate lexicon for the rain gathering scenario. The DCGs are used to check the requests and use policies are well-formed.

The resource governance reasoner consists of Prolog theories that directly correspond to the rules for the operational semantics specified in this paper, and a combinatorial Event Sequence Generator for producing Event Calculus narratives. We combine a request, a use policy, the operational semantics for the normative evaluator and conflict checker, and Event Calculus narratives into a single Prolog theory which we query for conflicts using a Prolog engine.

8 Conclusions

In this paper our main contribution was a novel temporal event-based reasoner for determining if there are conflicts between a request to use a resource and a use policy governing a resource's usage. This allows owners of resources to maintain autonomy over how, when and to whom their resources are used. Taking the notion of a request for the use of a resource as a set of norms, we gave a representation of a use policy, specifying what norms a request should and should not impose on a resource under some circumstances, as a set of meta-norms. Our reasoner detects whether a request can be accepted with respect to a a use policy or if there are conflicts necessitating rejecting the request. Our proposal is particularly robust because the operational semantics ensures a conflict is detected between a norm and a meta-norm only based on whether they can be detached simultaneously.

[3] A multi-agent modelling environment http://ccl.northwestern.edu/netlogo/

There are many interesting avenues for future work, we go over some of the most immediate extensions here. By extending the representation of meta-norms to include an 'Or Else' option, such as 'you should not obligate me to do X, but if you do then you should not forbid me to do Y' we will be able to investigate computing a partial ordering of ideal and sub-ideal requests. This can support better decision making in a resource governing agent, such as which requests are the best to accept when there are many offered, or support negotiation with a qualitative model of preferences. Extending our work to support negotiation would also require a conflict resolution mechanism, such that counter-offers can be made to the agent requesting the use of the resource by modifying the original request and sending it back as part of an interaction protocol.

Finally, a limitation of our proposal is that we use a propositions as the terms of norms and meta-norms. This makes sense since it allows us to express concrete norms in a request. For future work we can extend the representation languages to make use of variables in first-order logic. This will allow a user to express meta-norms such as 'you are prohibited from obligating me to do anything' and even meta-norms with constraints such as 'you are forbidden from obligating me to pay you anything over €10'.

Acknowledgements

This work is supported by the SHINE project of TU Delft, funded by the Delft Institute for Research on ICT.

References

1. G. Andrighetto, G. Governatori, P. Noriega, and L. van der Torre. Normative Multi-Agent Systems. *Dagstuhl Follow-Ups*, 4, 2013.
2. A. Artikis, M. Sergot, and G. Paliouras. Run-time composite event recognition. In *Proceedings of the 6th ACM International Conference on Distributed Event-Based Systems*, pages 69–80, New York, New York, USA, 2012. ACM Press.
3. G. Boella and L. van der Torre. Permissions and obligations in hierarchical normative systems. In *Proceedings of the 9th International Conference on Artificial Intelligence and Law*, pages 109–118, 2003.
4. F. Chesani, P. Mello, M. Montali, and P. Torroni. A logic-based, reactive calculus of events. *Fundamenta Informaticae*, 105((1-2)):135–161, 2010.
5. F. Chesani, P. Mello, M. Montali, and P. Torroni. Representing and monitoring social commitments using the event calculus. *Autonomous Agents and Multi-Agent Systems*, 27(1):85–130, June 2012.
6. L. Chittaro and A. Montanari. Efficient temporal reasoning in the cached event calculus. *Computational Intelligence*, 12(3), 1996.
7. V. Dignum. *A Model for Organizational Interaction: Based on Agents, Founded in Logic*. PhD thesis, University of Utrecht, Utrecht, The Netherlands, 2003.
8. D. Gabbay, J. Horty, X. Parent, R. van der Meyden, and L. van der Torre, editors. *Handbook of Deontic Logic and Normative Systems vol. 1*. 2013.
9. R. Ganti, F. Ye, and H. Lei. Mobile crowdsensing: current state and future challenges. *IEEE Communications Magazine*, 49(11):32–39, Nov. 2011.

10. G. Governatori. Representing business contracts in RuleML. *International Journal of Cooperative Information Systems*, 14(2-3):181–216, 2005.
11. A. Günay and P. Yolum. Detecting conflicts in commitments. *Declarative Agent Languages and Technologies IX*, pages 51–66, 2012.
12. J. Hübner, J. S. Sichman, and O. Boissier. A model for the structural, functional, and deontic specification of organizations in multiagent systems. *LNCS*, 2507:118–128, 2002.
13. T. C. King, Q. Liu, G. Polevoy, M. de Weerdt, V. Dignum, M. B. van Riemsdijk, and M. Warnier. Request Driven Social Sensing (Demonstration). In A. Lomuscio, P. Scerri, A. Bazzan, and M. Huhns, editors, *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2014)*, Paris, France, 2014.
14. R. Kowalski and M. Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–95, 1986.
15. F. L. y. López, M. Luck, and M. D'Inverno. A normative framework for agent-based systems. *Computational and Mathematical Organization Theory*, 12(2-3):227–250, 2006.
16. X. Parent and L. van der Torre. Input/output Logic. In D. Gabbay, J. Horty, X. Parent, R. van der Meyden, and L. van der Torre, editors, *Handbook of Deontic Logic and Normative Systems Vol. 1*, pages 499 – 544. 2013.
17. M. Shanahan. The event calculus explained. *Artificial Intelligence Today*, pages 409–430, 1999.
18. M. B. van Riemsdijk, L. A. Dennis, M. Fisher, and K. V. Hindriks. Agent Reasoning for Norm Compliance A Semantic Approach. In *Proceedings of the 12th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2013)*, pages 499–506, Saint Paul, Minnesota, USA, 2013.
19. W. W. Vasconcelos, M. J. Kollingbaum, and T. J. Norman. Normative conflict resolution in multi-agent systems. *Autonomous Agents and Multi-Agent Systems*, 19(2):124–152, Nov. 2008.
20. H. Wansing. Nested deontic modalities: Another view of parking on highways. *Erkenntnis*, 49(2):185–199, 1998.
21. P. Yolum and M. Singh. Reasoning about commitments in the event calculus: An approach for specifying and executing protocols. *Annals of Mathematics and Artificial Intelligence*, 42(1-3):227–253, 2004.
22. P. Yolum and M. P. Singh. Flexible Protocol Specification and Execution : Applying Event Calculus Planning using Commitments. In *The First International Joint Conference on Autonomous Agents and Multiagent Systems: Part 2*, pages 527–534, 2002.

View publication stats