Rules of Order for Electronic Group Decision Making – A Formalization Methodology

Henry Prakken[1] and Thomas F. Gordon[2]

[1] Department of Computer Science
Utrecht University, The Netherlands
henry@cs.uu.nl
http://www.cs.uu.nl/staff/henry.html
[2] GMD – German National Research Center for Information Technology
Sankt Augustin, Germany
thomas.gordon@gmd.de
http://nathan.gmd.de/persons/thomas.gordon.html

Abstract. This paper reports on an ongoing research project, consisting of formalizing rules of order for group decision making, and implementing them as a procedural component of automated mediation systems for group decision making. The component should ultimately assist a human mediator in maintaining order at electronic meetings, and in giving advice to the participants on their options, rights and obligations in the decision making process. A main requirement for the system is that order can be maintained in a flexible way, allowing to set the rules aside when needed. This paper presents the first research result of the project: a way of formalizing rules of order that makes it possible to maintain order in such a flexible way.

1 Introduction

1.1 Background

Electronic meetings are an important object of research in Computer-Supported Cooperative Work (CSCW). One aspect of ordinary meetings is that they are usually governed by rules of order, which should ensure that the group involved in a meeting can conduct their business in a way that is both fair and effective. A natural research goal is how these benefits can also be made to hold for electronic meetings. This is an important research topic, since the participants in meetings are, unlike in many other CSCW applications, not always cooperative but often have conflicting goals and interests. In such circumstances it becomes important that a decision is made according to a procedure accepted by all those involved.

In this respect, electronic meetings provide both problems and prospects. A problem is how rules of order can be enforced given that the participants are not all physically present in one room. A prospect is that the setting of computer systems might enable different ways of maintaining order than in a traditional seting. This paper focuses on these issues with respect to a particular kind of

electronic meeting, viz. one which should result in collective decisions of the group that meets.

In particular, we shall report on research done in the context of the ZENO computer system, developed at the GMD Bonn [3]. This system serves as an automated assistance tool for human mediators of discussions and group decision processes. It is currently applied to urban planning procedures, in the context of the GeoMed project [5], funded by the European Union. One component of the ZENO system is a discussion forum that is accessible via the World Wide Web, and where participants can raise issues, state positions with respect to these issues, put forward arguments for or against a position, and collectively decide an issue. The system, which is fully implemented, provides automated tools for maintaining and inspecting the resulting argumentation structure, and for recording the decisions. It combines elements of Horst Rittel's issue-based information system (see e.g. [9]) with insights from logic and argumentation theory.

At present the use of ZENO's discussion forum is completely unregulated. However, one research goal of the ZENO project is to study how the benefits of rules of order can also be made available for electronic meetings. More specifically, the aim is to extend ZENO's discussion forum with rules of order, and with a corresponding module (ROBERT) that assists the human mediator in maintaining order at the forum, and in giving advice to the users of the forum on their options, rights and obligations in the discussion and decision making process.

As for the rules of order, in this project a choice has been made for Robert's Rules of Order (RRO), which is the standard procedure for deliberative societies of all kinds in the USA. This choice requires some explanation, since the applicability of RRO to electronic discussion is not obvious. In particular, they are meant for synchronous discussion, i.e. discussion in meetings where all participants are present in the same place, at the same time, and where each participant can immediately observe and respond to all procedural events that are taking place. By contrast, in electronic discussion forums participants often have no full knowledge of who else is taking part in a discussion, and communication can be delayed: messages that are sent before another message can arrive later, and so on. Therefore it is still an open question which rules of order are suitable for electronic meetings. Moreover, the answer might very well be different for different types of electronic meetings, depending, for instance, on the size of the meeting, and on the degree of cooperativeness of the participants.

Nevertheless, RRO have still been chosen for this project, for two reasons. Firstly, RRO are well-known, precisely formulated, and well-tested in practice, and it is therefore expected that, even if they are not directly suitable for electronic applications, their formalization will still give useful insights into the problems and prospects of adding a procedural component to automated mediation systems. Second, as far as we know, suitable rules of order for electronic and asynchronous discussion do not yet exist. In fact, developing such rules is one of the goals of the ZENO research project, and we think that a good way to develop

such rules is to first formalize and implement existing rules of order, and to then use the result to experiment with alternative adaptations of the rules.

It might also be asked what the value is of formalizing particular rules of order, when different kinds of meetings might require different kinds of rules of order, at diferent levels of detail. Our answer is that the ultimate aim of the ROBERT and ZENO project is not just to find a particular body of suitable rules of order for electronic meetings, but to arrive at an general 'ontology' of the world of meetings, and at a general methodology for extending mediation systems with rules of order. Such an ontology and methodology can be derived from 'first principles', but it can also be induced from a well-known, well-tested and elaborate example. We think that both research strategies are equally valid, but the latter strategy is the one we are following. And the results so far indicate that it is a good strategy.

To summarize the research context of this paper, it is part of a project that aims to develop a formalization methodology for rules of order for electronic discussions, to apply this methodology to an example, to implement the formalization as a module of ZENO's discussion forum, and then to test how the rules must be adapted to electronic discussions of various kinds. The underlying research goal is to provide an ontology of the world of meetings and a method for maintaining order at electronic discussion forums. The present paper focuses on the formalization of a particular set of rules of order, viz. RRO, and on an aspect of its implementation in ZENO.

1.2 Problem Statement

The problem discussed in this paper is the following: given that we want an automated mediation system that maintains order in electronic meetings in a flexible way, how can rules of order can be formalized in a way that enables such a flexible enforcement? With 'flexible' enforcement we mean two things. Firstly, the system should make violation of the rules of order by the participants physically possible, instead of being designed in such a way that only correct acts can be performed. And, secondly, it should be possible for the chair to set the rules aside when needed to conduct a meeting efficiently. The underlying assumption here is that a system which strictly enforces a certain procedure will not be attractive for the users. Similar concerns have been expressed with respect to workflow management systems, e.g. by Suchman [13], calling for research on more flexible workflow management systems.

The results reported in this paper are twofold: a methodology for formalizing rules of order that enables their flexible enforcement at electronic meetings, and an application of this methodology to an example, viz. Robert's Rules of Order.

1.3 Related Research

As for RRO, in the literature at least one earlier suggestion for using them for similar tasks can be found, viz. Page [7], who suggests their use for controlling communication between intelligent artificial agents. However, we have not found

whether Page has carried out his suggestion. Vreeswijk [15] also refers to Stary [12], who would have made a similar suggestion, but we have not been able to trace that publication. Formal and computational aspects of legal procedures have been studied by Vreeswijk, e.g. in [15], who has attempted to formalize aspects of Peter Suber's [14] NOMIC game, a game of which the purpose is to modify the rules of the game. Vreeswijk's insights are directly relevant for the ROBERT project, since many rules of order contain provisions for changing them (although this issue is not discussed in the present paper). Finally, Gordon [2] has formalized and implemented his normative model of procedural justice in civil pleading, which became a source of inspiration of the ZENO project.

1.4 Structure of this Paper

Following a brief overview of RRO in Section 2, in Section 3 a high-level design is proposed for maintaining order at electronic meetings in a flexible way. That section includes a discussion of the various types of order violation that are possible. Then in Section 4 our methodology for formalizing rules of order is presented, and illustrated by applying it to RRO. Finally, in Section 5 the main results of this paper are summarized, and the current state of the ROBERT project is briefly sketched.

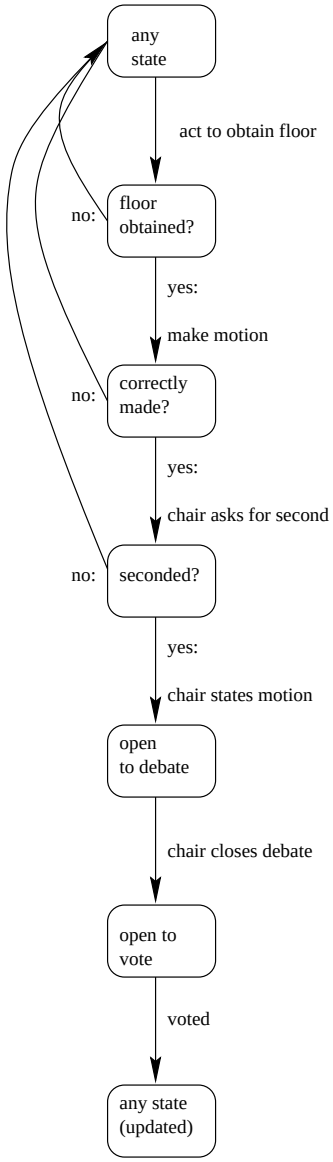
2 Overview of Robert's Rules of Order

This section briefly outlines the example rules of order chosen in our project, Robert's Rules of Order. These rules are based on parliamentary procedure in the USA. They were described by general H.M. Robert in 1876, and perfected by him for 35 years, in communication with many users of the rules. Over the years, Robert's rules have turned from a description into a definition of parliamentary procedure (cf. [7, p. 360]), and have become the standard rules of order for meetings of all kinds in the USA. Although both several watered-down and several extended versions have appeared over the years, the ROBERT project is based on the original text. The references in this paper are to a 1986 paperback publication of this text [10].

The 'world' of RRO is the world of meetings (more accurately, of sessions: each session is a series of meetings separated by adjournments). The main objects of this world are an *assembly*, consisting of *members*, which can have several *roles* (ordinary member, chair, secretary, ...), and finally, *issues*, or *questions* which are to be decided by the assembly.

RRO defines an extensive repertoire of procedural speech acts with which those present at a meeting can communicate. The primary topic treated by RRO is how to bring business before the assembly, and how to have this business dealt with. The main 'loop' of RRO is that a member has to act to obtain the floor, after which s/he should state a proposal (for which RRO uses the technical term 'motion'), which must be seconded by another member before the chair can open

the motion to debate by stating it. Debate is followed by a vote, after which new business can be introduced.



Main loop of RRO

This main loop has many exceptional cases, while also many complications can arise. As for the exceptions, some motions can be made while not having the floor, some do not need to be seconded, some are not debatable, and some motions are not decided by vote but by the chair. (A motion that satisfies all

these exceptions is a point of order). Virtually all of these exceptions are motions that, when adopted, have a certain procedural effect, like a point of order, an amendment, an appeal, an objection to the consideration of a question, a motion to adjourn, and so on. These procedural effects are one source of complications. Another source of complications is that certain motions can be made when another motion is pending and, when seconded, must be dealt with before the pending motion. This is captured by an order of precedence among motions, determining which motions can be made while another motion is pending.

The main precedence ordering is not defined on individual motions but on four categories of motions, which, in descending order of precedence are:

- *Privileged motions* (fix time of adjournment, adjourn, questions of privilege, orders of the day);
- *Incidental motions* (appeal/questions of order, objection to the consideration of a question, reading papers, withdrawal of a motion, suspension of the rules);
- *Subsidiary motions* (lay on the table, previous question[1], postpone to a certain day, commit, amend, postpone indefinetely);
- *Principal motions* (any other motion, usually motions related to the purposes for which the meeting is held).

The largest part of RRO is devoted to a discussion of all these types of motions. Their further order of precedence is defined, special conditions for when they are in order are given (e.g. an objection to a consideration of a question must be made immediately after the question has been introduced), the required majority for acceptance is defined, it is stated whether they can be made without having the floor, whether they require a second, whether they are debatable, renewable, amendable, reconsiderable, etc ..., and their procedural effects when made and when adopted are defined.

In addition to motions, RRO regulates the way debate and vote are conducted, the rights and duties of the officers of an assembly, the minutes, the functioning of committees, and some other things, like the quorum, and orders of business.

A main feature of RRO is that they acknowledge that sometimes it is better to temporarily put them aside (e.g. RRO 1 and RRO 3, p. 34). For instance, many questions of routine are not formulated as a motion and then seconded and stated; instead, the chair often announces after informal discussion that if no one objects, such an such is the action of the assembly. The general rule is that anything goes until any member objects, after which RRO must be strictly applied.

[1] This is a technical name for a motion to immediately put the pending question to vote.

3 How to Maintain Order at Electronic Meetings in a Flexible Way

As stated in the introduction, the formalization and implementation of RRO, or other rules of order, (called ROBERT) should ultimately be integrated with ZENO's discussion forum. This section discusses the functions that ROBERT can have within ZENO, and the corresponding tasks that it should be able to perform. On the basis of this discussion we propose how ROBERT can deal in a flexible way with violations of the rules of order by the users of ZENO's discussion forum (including the human mediator).

In the following, it should be kept in mind that ZENO's discussion forum is monitored by a human mediator, who not only acts as the chair but also processes the messages into a format that is readable for the ROBERT component.

3.1 Functions and Tasks of ROBERT within ZENO

As a component of ZENO, ROBERT can be made to perform two different functions.

1. As an autonomous expert system, giving advice to users of ZENO's discussion forum and to the human mediator, on procedural possibilities, rights, obligations. Here the human mediator independently maintains order at the forum, and ROBERT fulfills much the same role as a book copy of the rules of order at the chair's or a participant's table in an ordinary meeting.
2. Connected with the discussion forum, as a tool for maintaining order at the forum. Here ROBERT performs certain actions on behalf of the chair (or the secretary), like warning participants that they are out of order, and maintaining a list of decisions.

The aim of the ROBERT project is that the same core system can perform both functions (although for each of the functions probably some specific additional components are needed). Accordingly, the formalization of any rules of order should be such that it can be used for implementing both of these functions.

To fulfill these functions, ROBERT should be able to perform the following two tasks:

1. Update the current state of the procedure;
2. Determine of any procedural act whether it conforms to the rules.

The formalization of rules of order (in our case RRO) has to take these tasks into account. In Section 4 we shall explain how our current formalization does so.

3.2 How ROBERT Should Deal with Violations of the Rules of Order

Like any set of norms for human behaviour, rules of order can be violated. How should ROBERT deal with these violations? At first sight, one might think that

the implementation of ROBERT as a computer system yields an opportunity that human chairs of ordinary meetings rarely have: ZENO's discussion forum could be set up in such a way that violation becomes physically impossible. For instance, if in a meeting governed by RRO a participant wants to push a button 'Make motion' just after another participant has moved a motion that requires a second, the system might, instead of returning a window for typing the motion, return a window saying "You are out of order, change your input". Such an implementation of rules of order would be what Jones & Sergot [4] call 'regimentation' of norm-governed behaviour: the system is implemented in such a way that all meetings will as a matter of fact conform to the rules of order.

However, as Jones & Sergot remark, regimentation is not always a good idea (see also [13], as mentioned above). ZENO's discussion forum is a good example of a system that should not be regimented. It seems that to be workable in practice, the system must not be too rigid: there is a real danger that if the system strictly enforces a certain procedure on the participants of a discussion, they will be discouraged from using the discusion forum. This is even acknowledged by many existing rules of order, such as RRO, which, as noted above, at various places formulates the principle that its formalities can be dispensed with as long as no member objects. Therefore, in our example we want that the system, instead of the above window, returns a window "You are out of order, do you want to sustain your input?"

Accordingly, a basic idea of the current project is that as an implemented system, ROBERT should satisfy the following constraints.

1. It must be physically possible for the users (including the chair) to violate the rules of order;
2. It must be possible for the chair to set the rules of order aside when needed.

We shall now look in more detail at the various ways in which rules of order might be violated, how ROBERT should deal with them, and what this means for a formalization of rules of order. Although we shall do so for our example rules of order, RRO, our observations apply to rules of order in general.

Violations by Ordinary Participants Ordinary participants can violate rules of order in only two ways, which are structurally similar, viz. by performing an act that is not in order (not at the right moment) or improper (not of the right kind).[2] ROBERT can deal with such violations as follows. As for knowledge representation, it suffices to specify under which conditions an act has the property of being out of order, or improper. Then every time ROBERT derives that an act is out of order or improper, it notifies all participants of the violation, possibly with advice on an action that is possible (for instance, a point of order). This message should not only be sent to the chair (i.e., the human mediator),

[2] RRO do not explicitly distinguish these two notions; the distinction has been introduced into the formalization to make it more structured. We expect this to be useful for other rules of order as well.

but to all participants, since usually all participants have the right to rise to a point of order (see in RRO section 14). Furthermore, the 'sanction' for these kinds of violation is simply that the intended procedural effect does not occur. For instance, according to RRO an incorrectly moved motion does not become open for seconding. And, of course, the chair can call the participant to order, and any other participant can rise to a point of order.

Violations by the Chair The mediator can, in its role of the chair, violate the rules of order in several different ways, which are not easy to deal with in a uniform manner. Firstly, it is possible that the chair does not perform an act that s/he must perform: for instance, in meetings governed by RRO, not stating a seconded and debatable motion, or not putting a motion to vote after debate has been closed. A variant of this kind of violation is when the chair incorrectly performs such an act: for instance, when stating a motion, the chair uses substantially different words than the person who made the motion.

A completely different kind of violation is when the chair incorrectly applies a rule of order: for instance, the chair incorrectly rules a proposal out of order, or declares a proposal adopted that needs a 2/3 vote but received only 56 % of the votes. Why is this a different kind of violation?

With the first two kinds it is easy to make a simple syntactic difference between the obligatory act and the act as it actually takes place. For instance, a formalized rule can say then when a proposal to end debate has been adopted, the next act of the chair must be putting the proposal to vote. Whether the chair indeed performs the obligatory action is then a matter of factual input to the system, just as with the behaviour of ordinary participants. In RRO and many other rules of order the sanctions for this kind of violation that ordinary participants can rise to a point of order, and that the obligation to perform the act stays in effect as long as it is not performed. Since the difference between actual and required behaviour is made with syntactic means, it is easy for ROBERT to detect such violations.

However, with the last kind of situation, erroneous application of a rule of order by the chair, things are different. Here it does not make much sense to formalize the rules of order in such a way that if, for instance, a participant starts debating a proposal before it has been opened to debate, the chair *ought to* rule the participant out of order. Instead, we want that ROBERT *infers* that the participant is out of order, and informs the chair about this fact, who can then accordingly rule the participant out of order. If otherwise, then virtually no rule application can be made automatically by the system; nearly every logical inference step that a reader of the rules of order would make will have to be replaced by factual input concerning the chair's actual behaviour. Clearly, such a system would not be very useful. On the other hand, we have just stated that ROBERT should make violations of the rules of order possible, so we must have at least some way of modelling erroneous application of a rule of order.

We have chosen for a design in terms of consistency checking and belief revision. The idea is that such violations are added as factual input by the chair,

after which the system detects and reports that the chair's input contradicts ROBERT's conclusions. For example, suppose that in a meeting governed by RRO that the chair mistakenly opens a motion to debate that has not yet been seconded. The system then asks the chair: note that according to my information the motion is not open to debate, so are you sure? Then the chair might ask: why is it not open to debate?, after which the system exlains why, viz. because the motion needs a second but has not yet been seconded. Then the chair might decide whether to follow the system and withdraw his/her input (i.e. to acknowledge violation of a rule of order), or whether to sustain the input (i.e. to set the rule aside), in which case the system revises its state.

Note that this interaction procedure might be very useful: it makes the chair (or other users) aware of which conclusions have to be changed if the user's input is to be sustained. And this might make the user aware of the mistakes s/he has made.

3.3 Formalization Requirements

Our design proposal is still general, and its implementation involves several non-trivial technicalities, such as the belief revision procedure and matters of user-interface design. Nevertheless, for present purposes it is specific enough to check a formalization of rules of order on adequacy. In particular, on the basis of our analysis we can state the following formalization requirements.

- The formalization must cope with the changing world of meetings, so that the system can update the state of the procedure each time something relevant happens.
- For violations by ordinary participants, and for certain types of violations by the chair, a syntactic distinction must be made between required and actual behaviour, so that the system can detect and report such violations.
- By contrast, a special type of violation by the chair, viz. erroneous application of a rule of order, should be detected as a contradiction between procedural conclusions drawn by the system and those typed in by the chair.
- Our aim of flexible enforcement of rules of order requires that when the chair sustains erroneous input, then in this 'subideal' state all other rules of order still apply. For instance, when a chair opens debate on an undebatable proposal, and no participant objects, the rules on how to conduct debate and on which proposal can be made while another one is pending, should still apply.

4 Formalizing Rules of Order

In the present section we shall present a formalization methodology that respects the requirements of the previous section, and illustrate it with our formalization of RRO. Our methodology uses the language and semantics of standard first-order predicate logic (FOL), and assumes that reasoning with a formalization is

valid first-order reasoning. It should be noted that the methodology is not meant to result in a directly exectutable logic program, but rather in a formal specification for a designer of an expert system; further decisions on implementation are still necessary.

As for our choice for standard first-order logic (which leaves open the possibility of a more structured format using description or terminological logics), at several points computer science and Artificial Intelligence provide alternative formalisms (such as nonmonotonic logics for formalizing change, and deontic logics for formalizing normative concepts). We shall briefly discuss them when relevant. However, the pragmatic constraints of the ZENO project have led us not to use them. FOL is well-understood and sufficiently expressive, and many ways to implement it exist. Moreover, FOL enables a style of formalization that can be easily implemented in standard expert system tools. Nevertheless, even when other formalisms are chosen, our formalization has its use, since it is much easier to change one formalization into another than to formalize a natural language text.

We next explain some details on notation. As for the logical symbols, $\neg$ stands for logical 'not', $\wedge$ for 'and', $\Rightarrow$ for 'if ... then', and $\Leftrightarrow$ for 'if and only if'. When relevant, first-order predicates have an argument for a state term. A discretely and linearly ordered set of states is asssumed. State variables are written as possibly indexed or primed s, while person variables are written as y, y', ..., and variables for acts as x, x', ... or z, z', ... `Type writer` strings are predicate symbols when they begin with a capital, otherwise they are function symbols. If an argument of a predicate symbol has more than one letter, like *chair*, it is an object constant. Finally, formulas with free variables are implicitly assumed to be universally quantified, as in logic programming.

In the present paper we use, for purpose of presentation, a quasi-natural-language notation, which is inspired by [6]. For instance, we write

x `Is stated by` *chair* `at` s

instead of the more standard FOL notation

`Is stated by`$(x, chair, s)$

which we have used in [8], or even

`Is_stated_by`$(x, chair, s)$

Note that in the expression

$\neg$ x `Is stated by` *chair* `at` s

the negation symbol $\neg$ does not apply to the term x (which would not be well-formed in FOL) but to the entire expression; in standard notation:

$\neg$ `Is_stated_by`$(x, chair, s)$

The choice between these styles of formalizing is not something which is essential for our methodology. We note that [6] defines a systematic way to convert standard FOL notation into the above quasi-natural-language form.

4.1 Procedural Speech Acts

Among other things, rules of order define the possible procedural speech acts, usually as a taxonomy of types and subtypes. We first discuss the representation of such a taxonomy. This is specified as an inheritance hierarchy with exceptions, where each class has at most one superclass. Each class of speech acts has certain attributes with specified values. Some attribute values are given directly, others by way of rules. When attribute values are not explicitly specified for a certain class, it inherits the values of its immediate superclass. In [8] it is specified how this hierarchy can be translated into predicate logic formulas.

We now illustrate our speech act representation with the specification of a motion according to RRO.

Type: x `Is a motion`
Superclass: x `Is an act`
Attributes:
- x `Is debatable` (motions are debatable)
- $\neg$ x `Is in order when another has floor at` s (motions are not in order when another has the floor. This attribute has a second argument for the state because sometimes its value depends on the situation)
- x `Requires second` (motions require a second)
- `Required majority for` x `is` *simple* (The required vote for motions is a simple majority)
- `Decision mode of` x `is` *vote* (motions are decided by vote (alternative: by chair's decision))
- z `Is applicable to` x? See rules. (all subsidiary motions except postpone indefinitely)
- x `Is renewable at` s? See rules.
- x `Is reconsiderable` (motions are reconsiderable)
- $\neg$ x `Is to be entered on the record when made` (motions need not be entered on the record when made (only exception: reconsider))

The attribute `Is applicable to` captures the subsidiary motions that are applicable to a motion.

Some attribute values are defined by logical rules. For instance, the attribute `Is renewable at` receives its value by rules that say that motions are renewable after the introduction of any motion that alters the state of affairs).

What is also specified by rules is the special order conditions for a speech act, and the procedural effects of performing a motion and of adopting it. For

instance, according to RRO an objection to the consideration of a motion is only in order when made immediately after that motion has been introduced, making such an objection has the procedural effect that the pending question is changed to the objection, and adopting it has the effect that the objected motion is removed from before the assembly.

To illustrate inheritance and exceptions, consider the specification of RRO's class of incidental motions.

Type: x `Is an incidental motion`
Superclass: x `Is a motion`
Attributes:
- $\neg$ x `Is debatable`
- z `Is applicable to` x? See rules (all subsidiary motions except amendment and motion to postpone indefinetely)

Thus the class of incidental motions inherits all its attribute values from the class of motions, except the values for `Is debatable` and `Is applicable to`. In the latter case this is since the rules for which subsidiaries are applicable are different than those for motions in general, and thus override these rules.

4.2 Coping with the Changing World of Meetings

The world of meetings is a constantly changing world. Speakers obtain or yield the floor, and motions are introduced, debated and decided. Accordingly, different *states* of a meeting can be distinguished, with different speakers, different pending questions, and several other differences. States are *changed* by procedural speech acts (moving, seconding, acting to obtain the floor, voting, etc ...), according to their procedural effects as defined by the rules of order.

In computer science and Artificial Intelligence (AI) formalizing changing worlds is a heavily studied topic. In AI a debate has been going on between those who 'want to do it all in logic', e.g. [11], and those who admit procedural elements in their specification. We have chosen for a method of the latter kind, essentially based on the so-called STRIPS approach to planning [1].

The logical component of our method is as follows. In the knowledge base, procedural facts are not just true or false, but true or false relative to a state of a meeting. Accordingly, a state is conceived as a first-order object, and aspects (attributes) of a state are expressed with predicates having the state as an argument. For instance, the pending question of a state s is expressed as x `Is the pending question at` s, and the speaker (who has the floor) at state s is expressed as y `Has the floor at` s. Events occurring in a state are expressed likewise. For instance, that a motion m is seconded at s by person p can be expressed as m `Is a motion` $\wedge$ m `Is seconded by` p `at` s.

State changes are formalized as follows. For any state s, we denote its immediate successor with s'.[3] Then a state change is defined by rules that have a term

[3] A full formalization should contain axioms that justify this intended reading.

s in their antecedent predicates, and a term s' in their consequent predicates. For instance, RRO's rule that a debatable motion becomes open to debate after it is stated by the chair can be written as

x `Is stated by` *chair* `at` $s \wedge x$ `Is debatable` $\Rightarrow$
x `Is open to debate at` s'

The procedural element of our method (adapted from STRIPS) comes in to solve the following problem, which in AI is called the 'frame problem'. Assume that we have derived that a certain motion m is open to debate at s, and assume also that a participant p becomes the new speaker at the next moment s'. Then we want to conclude that m is still open to debate at s'. However, in standard first-order logic this can only be derived if the knowledge base also contains the following rule, a so-called 'frame axiom':

x `Is open to debate at` $s \wedge$ *'nothing relevant happens'* $\Rightarrow$
x `Is open to debate at` s'

where *'nothing relevant happens'* is the negated disjunction of all ways in which a motion ceases being open to debate at s'. For various reasons this way of formalizing the effects of actions, where for each state not only what has changed must be specified, but also what has not changed, is widely considered to be unattractive. In logic, so-called nonmonotonic logics have been developed, in which it can be *assumed* that things do not change unless an explicit reason for change becomes known.

However, for the pragmatic reasons sketched above, we shall not use one of those logics, but instead add an extralogical component to our method. The idea is that any state of affairs that persists until it is changed by some event, is an attribute of a data structure called the *record*. The record is not made relative to a state, but exists 'globally'. So it says m `Is the pending question` instead of m `Is the pending question at` s_1. Its attribute values are updated when needed: each time the knowledge base derives a change in the value of some attribute, its value on the record is changed. For instance, when a conclusion n `Is the pending question at` s_2 is derived from the knowledge base, then at the record the value of `Is the pending question` is changed from m to n. And each time the logical reasoning process needs the value of a record attribute, a look-up at the record is performed.

An intuitive way to understand this method is to think of a meeting where behind the chair stands a blackboard, at which the values of the record are written. Each time an event triggers a change in, say, the pending question, the chair erases the old value and writes down the new one. And each time the chair wants to know what is the pending question, s/he looks at the blackboard.

As for the content of the record, the general rule is that any procedural property of which we want to assume that it persists until it is explicitly changed, is an attribute of the record. In addition, the record keeps track of the procedural

acts that have been made during a session, as well as the decisions on the motions made. This component is useful when information is needed about the past, for example, when of a motion that cannot be renewed it must be known whether it has already been made.

In our formalization of RRO (and probably of any rules of order), some important record attributes are the following:

- *The speaker.* This says who is the speaker, i.e., who has the floor, if any.
- *The question stack.* This lists the motions that at any state are before the assembly (debated or decided), being brought before the assembly (the phase from being correctly moved to being stated), or temporarily set aside by another motion with higher precedence. The top of the question stack is:
- *The pending question.* This is the question that is currently before the assembly. It is the motion that is either being brought before the assembly, or being debated, or being decided.
- *What is open to debate.* This says which motion is currently debated, if any.
- *The session history.* This records the procedural acts made during a session, as well as the decisions on the motions made.

This completes our discussion of how the first formalization requirement of Section 3.3 can be met, viz. how the dynamic aspect of meetings can be formalized. We now turn to the other three requirements, which are about violation and flexible enforcement of rules of order.

4.3 Distinguishing Actual and Required behaviour

The second requirement on formalizations of rules of order is that for several types of behaviour they make a syntactic distinction between actual and required behaviour. How can this be done? Various ways are possible, including the use of a full-fledged deontic logic. Deontic logic is a branch of modal logic, which adds to standard logic the logical operators O for 'obligatory', P for 'permitted', and F for 'forbidden'. Thus it becomes possible, for instance, to say O *chair* `Puts the affirmative of` $motion_1$ `at` s_1, which says that the chair must put the affirmative vote on a particular motion at state s_1. However, as explained above, the present formalization stays within first-order logic. The normative character of rules of order is captured by three special 'quasi-deontic' predicates, `Is proper at`, `Is in order at`, `Correctly makes at` and a surrogate deontic predicate `Is obliged to make at`, which can be used to define more special versions `Is obliged to ... at`. The quasi-deontic predicates are used in the following 'top level' rules.

x `Is an act` $\wedge$ y `Makes` x `at` s $\wedge$ x `Is in order at` s $\wedge$ x `Is proper at` s $\Leftrightarrow$ y `Correctly makes` x `at` s

This rule says that a procedural act is correctly made if and only if it is in order and proper. In our formalization of RRO, the latter predicates are defined

in further rules. For motions, the 'top level' definition of `Is in order at` is as follows.

x `Is a motion` $\wedge$ y `Makes` x `at` s $\wedge$
y `Fulfills floor condition of` x `at` s $\wedge$
y `Fulfills precedence condition of` x `at` s $\wedge$
`Renewal condition of` x `is fulfilled at` s $\wedge$
`Mode condition of` x `is fulfilled at` s $\wedge$
`Special order conditions of` x `are fulfilled at` s
$\Leftrightarrow$ x `Is in order at` s

The atomic expressions in the conditions of these rules have the following intuitive reading (the page numbers refer to [10]).

- y `Fulfills floor condition of` x `at` s (pp. 27–32) means that the rules concerning having the floor do not prevent making the motion (either one has the floor, or having the floor is not required).
- y `Fulfills precedence condition of` x `at` s (p. 12) means that no pending question prevents making the motion (either there is no pending question, or the pending question yields to the moved motion).
- `Renewal condition of` x `is fulfilled at` s (pp. 178/9) means that the rules on renewing motions do not prevent making the motion (either it can be renewed, or it is moved for the first time).
- `Mode condition of` x `is fulfilled at` s says that the rules requiring special acts at certain moments (e.g. seconding when a motion that requires second has been made) do not prevent making the motion.
- `Special order conditions of` x `are fulfilled at` s means that any special conditions for the relevant type of motion are fulfilled.

The quasi-deontic predicates are convenient for formalizing prohibitions (`Is in order at`) and obligations to make an act, if it is made, in a certain way (`Is proper at`). However, they are less suitable for obligations to perform a certain act, like in RRO the obligation for the chair to state a motion after it has been seconded. For such obligations the surrogate deontic predicate, `Is obliged to make at` (or special versions) will be used, as in, for instance, the following rule of RRO's voting procedure:

x `Is open to vote at` s $\wedge$ $\neg$ `Ballot is ordered for` x `at` s $\wedge$ $\neg$ `Roll call is ordered for` x `at` s $\Rightarrow$
chair `Is obliged to put the affirmative of` x `at` s

This rule says that when a motion is open to vote (e.g. since debate has closed) and no ballot or roll call has been ordered, the chair is obliged to put the affirmative.

The use of quasi-deontic predicates is not so strange, since the law also often uses such predicates, like 'tort' and 'criminal offence' instead of the deontic term 'forbidden'. For example, the Dutch criminal code hardly contains any deontic expression: it mainly defines the notion of criminal offence and its subcategories, and specifies the penalties for when actual behaviour satisfies these categories. It is left to the citizens to pragmatically infer from these penalties that they had better not commit criminal offences.

Our third formalization requirement is that erroneous application of a rule of order by the chair is detected in terms of a contradiction check. Our formalization method also meets this requirement. When, for instance, the chair incorrectly rules a motion in order, s/he inputs a fact $Motion_1$ `Is in order at` s_1. Since this is incorrect, the system will derive $\neg$ $Motion_1$ `Is in order at` s_1 and then recognize the contradiction with the chair's input.

Finally, as for the last formalization requirement, here is how the chair can set the rules of order aside on one point without ignoring them completely. In our example, s/he can do so by sustaining the erroneous input $Motion_1$ `Is in order at` s_1. The belief revision component of the system then withdraws its own contradictory conclusion and the system then further reasons with the chair's input. Note that thus setting the rules of order aside on a certain point does not render the rules inapplicable on other points. For instance, the system can (if the other relevant conditions are also fulfilled) derive (for the maker p of the motion) p `Correctly makes` $Motion_1$ `at` s_1, after which other rules apply as usual, for instance, a rule saying that when a motion that needs a second is correctly moved, the motion is open for being seconded.

5 Conclusion

In this paper we first presented a (high level) design for how order can be maintained at electronic meetings in a flexible way. The main features of the design are

- The system keeps track of the changing state of a meeting;
- The system recognizes and reports violations of the rules of order;
- The system does not physically enforce the rules of order;
- The system allows the chair to set the rules of order aside when needed.

Then we stated some requirements for any formalization of rules of order that is to be used in such a design, after which we presented a formalization methodology that meets these requirements.

The current state of the ROBERT project is as follows. The ZENO discussion forum is fully implemented, but as yet it contains nothing of the above design. However, the formalization of RRO with the above-sketched methodology is reaching its completion. The main formalization problems have been solved, RRO's top level structure is formalized, and most of the details are filled in. The state of the formalization at the moment of the writing of this paper is reported in [8]. That report also contains further discussions of some of the alternative formalisms that were briefly mentioned in this paper.

References

1. Fikes, R.E. & Nilsson, N.J. STRIPS: a new approach to the application of theorem proving to problem solving. *Artificial Intelligence* 2 (1971), 189–208.
2. Gordon, T.F. *The Pleadings Game. An Artificial Intelligence Model of Procedural Justice*. Kluwer Academic Publishers, Dordrecht (1995).
3. Gordon, T.F., & Karacapilidis, N. The Zeno argumentation framework. In *Proceedings of the Sixth International Conference on Artificial Intelligence and Law*, ACM Press, New York (1997) 10–18.
4. Jones, A.J.I. & Sergot, M.J. On the characterisation of law and computer systems: the normative systems perspective. In J.-J.Ch. Meyer & R.J. Wieringa (eds.): *Deontic Logic in Computer Science: Normative System Specification*. John Wiley and Sons, Chicester (1993) 275–307.
5. Karacapilidis, N.I., Papadias, D., Gordon, T.F & Voss, H. Collaborative environmental planning with GeoMed. *European Journal of Operational Research*, Special Issue on Environmental Planning, Vol. 102, No. 2 (1997) 335–346.
6. Kowalski, R.A. Legislation as logic programs. In Z. Bankowski, I. White & U. Hahn (eds.): *Informatics and the Foundations of Legal Reasoning*. Law and Philosophy Library, Kluwer Academic Publishers, Dordrecht etc. (1995) 325–356.
7. Page, C.V. Principles for democratic control of bounded-rational, distributed, knowledge agents. *Proceedings of the European Simulation Conference*, ed. E. Mosekilde, (1991) 359–361.
8. Prakken, H. Formalizing Robert's Rules of Order. An Experiment in Automating Mediation of Group Decision Making. GMD Report 12 (1998), GMD – German National Research Center for Information Technology, Sankt Augustin, Germany. Electronically available at http://nathan.gmd.de/projects/zeno/publications.html
9. Rittel, H.W.J. & Webber, M.M. Dilemmas in a general theory of planning. *Policy Sciences* (1973), 155–169.
10. Robert, H.M. *Robert's Rules of Order. The Standard Guide to Parliamentary Procedure*. Bantam Books, New York etc. (1986).
11. Shanahan, M.P. *Solving the Frame Problem*. MIT Press, Cambridge, MA (1997).
12. Stary, C. Modelling decision support for rational agents. *Proceedings of the European Simulation Conference*, ed. E. Mosekilde (1991) 351–356.
13. Suchman, L. Do categories have politics? The language/action perspective reconsidered. *Computer-Supported Cooperative Work* 2 (1994) 177–190.
14. Suber, P. *The Paradox of Self-amendment: a Study of Logic, Law, Omnipotence, and Change*. Peter Lang, New York (1990).
15. Vreeswijk, G.A.W. Formalizing Nomic: working on a theory of communication with modifiable rules of procedure. Technical report CS 95-02, Dept. of Computer Science, University of Limburg, Maastricht, The Netherlands (1995).