Formalizing Robert's Rules of Order. An Experiment in Automating Mediation of Group Decision Making

Henry Prakken
GMD - German National Research Center for Information Technology
Schloss Birlinghoven, 53754 Sankt Augustin, Germany
henry@cs.vu.nl
http://www.cs.vu.nl/~henry

19 December 1997

Abstract

Robert's Rules of Order are the standard procedure for deliberative societies of all kinds in the USA. This paper reports on an ongoing experiment: formalizing these rules for the purpose of implementing them as a procedural component of automated mediation systems for discussion and group decision making. Robert's Rules of Order have been chosen for this experiment because they are well-known, precisely formulated, and well-tested in practice. Although they need to be adapted for electronic applications, their formalization should nevertheless give useful insights into the problems and prospects of adding a procedural component to automated mediation systems.

The research is carried out in the context of the ZENO mediation system, developed at the GMD Bonn. One of ZENO's components is a WWW-accessible discussion forum. The aim of the ongoing experiment is to extend this forum with rules of order, and with a corresponding mudule 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.

This paper reports on the first part of the experiment, formalizing Robert's Rules of Order in first-order predicate logic. This formal specification should be the basis for a more operational specification, and for the eventual implementation as a component of ZENO.

Contents

Preface

This paper is a report on my work at the FIT-KI, GMD Bonn, during my period of employment as a temporary researcher at the GMD, from August 1st till December 31st, 1997. During this period, I started a project in collaboration with Tom Gordon, in the context of the ZENO/GeoMed system, an automated mediation system for group discussion and decision making, being developed at the GMD. The project aims to extend ZENO's discussion forum with rules of order, and with a corresponding mudule for maintaining order at the forum. The present paper reports on the first part of the ROBERT project, formally specifiying a particular system of rules of order, viz. Robert's Rules of Order, the standard procedure for deliberative societies of all kinds in the USA.

The purpose of this paper is not to report on a finished piece of work, but to present the work that I did during my stay at the GMD. Therefore, this report ends 'in the middle' of the formalization process. Not all relevant parts of Robert's Rules of Order have yet been formalized, and some other parts have been formalized incompletely. Moreover, some of the formalized parts have not yet been completely tested and debugged. Nevertheless, my claim is that the formalization as it is included in the report is of sufficent quality to be presented before and criticised by an academic audience. Parts that are still too rough and preliminary to meet this standard have been left out of this report.

I hope that this report will be a fruitful basis for continuing my collaboration on this project with Tom Gordon and the rest of the ZENO/GeoMed team. I thank FIT-KI, GMD for their hospitality, and Hans Voss and Tom Gordon for inviting me, with financial support of VIM (A VIrtual Multicomputer), a project funded by the EC's Human Capital and Mobility programme.

Chapter 1

Introduction

Background

Robert's Rules of Order are the standard procedure for deliberative societies of all kinds in the USA. This paper reports on an ongoing experiment: formalizing these rules for the purpose of implementing them as a procedural component of automated mediation systems for discussion and group decision making. The project, which is called ROBERT, is carried out in the context of the ZENO computer system, developed at the GMD Bonn [Gordon, 1994b, Gordon & Karacapilidis, 1997]. 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 [Karacapilidis et al., 1997], 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, and put forward arguments for or against a position. The system provides automated tools for maintaining and inspecting the resulting argumentation structure, and also for recording decisions on the issues.

At present the use of the discussion forum is completely unregulated. However, underlying the Zeno project is the view that rationality has a procedural side, which for group decision processes means that the quality of the outcome of the process not only depends on the quality of the arguments put forward, but also on the properties of the disputation procedure in which the argumentation takes place. This view has been put forward by several philosophers, perhaps starting with Toulmin's [1958, pp. 7–8] advice that logicians who want to learn about reasoning in practice, should turn away from mathematics and instead study jurisprudence, since outside mathematics the validity of arguments would not depend on their syntactic form but on the disputational process in which they have been defended. According to Toulmin an argument is valid if it can stand against criticism in a properly conducted dispute, and the task of logicians is to find criteria for when a dispute has been conducted properly; moreover, he thinks that the law, with its emphasis on procedures, is an excellent place to find such criteria.

Although Toulmin was perhaps too polemic in his criticism of standard logic, the idea that correct reasoning not only depends on syntax and semantics but also on procedure has gained ground. For instance, Rescher [1977] has sketched a dialectical model of scientific reasoning, of which he claims, among other things, that it can explain the feasibility of inductive arguments: they must be accepted if they cannot be successfully challenged in a properly conducted scientific dispute. In legal philosophy Alexy [1978] has formulated a discourse theory of legal argumentation, based on the view that a legal decision is just if it is the outcome of a fair procedure. A similar view on argumentation in general underlies the so-called 'pragma-dialectical' school of argumentation theory [van Eemeren & Grootendorst, 1992]. And in Artificial Intelligence (AI) Loui [1998] has also defended a procedural view on rationality. According to him such a view can explain why nondeterministic reasoning can still be rational, viz. if this reasoning takes place in the context of a fair and effective protocol for dispute. Finally, Gordon's [1994,1995] Pleadings Game, which is an AI model of procedural justice in civil pleading, takes its point of departure in Alexy's views.

In order to integrate the logical and procedural aspects of rational argumentation, both Brewka & Gordon [1994] and Gordon & Karacipilidis [1997], and Prakken [1995,1997] have proposed a multi-layered view on argumentation. The first two layers assume a fixed body of information: the *logic* layer defines which arguments can be constructed with this information, and the *dialectical* layer determines, given certain evaluation criteria that are also in the information base, which arguments survive the competition with all conflicting arguments. The third, *procedural* layer drops the assumption of a fixed body of information and instead assumes that the information base is constructed dynamically during the dispute (thus the first two layers apply to each stage of a dispute). This layer defines the possible speech acts for doing so, and the norms for when these speech acts may or must be used (Accordingly, Brewka & Gordon [1994] and Gordon & Karacipilidis [1997] explicitly divide the procedural layer into a *speech act* layer and a *norm* layer.)[1]

Research goals of the ROBERT project

This multi-layered view on rational argumentation provides the background of the ROBERT project, which aims to implement the procedural layer in the context of the ZENO project. More specifically, the aim is to extend ZENO's discussion forum with rules of order, in particular, Robert's Rules of Order (RRO), 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. Although an ultimate goal is to integrate ROBERT with a formalization of the logical and dialectical layer, the present project completely abstracts from these two layers. In particular, ROBERT's (and RRO's) rules on debate do not assume any logical or dialectical structure of the debate.

The ROBERT project should result in an answer to several research questions. First of all, it should yield an ontology of the world of meetings, which can serve as a general basis for implemented procedures for discussion and group decision making. Second, the project should result in a methodology for extending mediation systems with rules

[1]Prakken [1997] also distinguishes a fourth, *strategic* or *heuristic* layer, which defines rational argumentation strategies within the procedural bounds of the third layer and the logical and dialectical bounds of the first two layers.

of order. Another research goal is to test how these rules of order can be implemented in a 'soft' way, i.e. such that they can be set aside when needed. The underlying assumption here is that system which strictly enforces a certain procedure will not be attractive for the users. Finally, the project should result in an answer whether regulating automated discussion and group decision making is useful at all, and if so, under which circumstances.

The choice for Robert's Rules of Order

The choice for RRO requires some explanation, since their applicability to electronic discussion is not obvious. RRO was meant for synchronous discussion, i.e. discussion in meetings where all members are present at the same time, in the same place, and where each member can immediately observe and respond to all procedural events that are taking place. By contrast, in electronic discussion forums members 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 RRO will need to be adapted for electronic applications,[2] which might be a considerable task.

Nevertheless, RRO have still been chosen for this experiment, for two reasons. Firstly, as far as I know, there are as yet no suitable rules of order for electronic and asynchronous discussion, and secondly, RRO are well-known, precisely formulated, and well-tested in practice, and it is therefore expected that their formalization will still give useful insights into the problems and prospects of adding a procedural component to automated mediation systems.

Related research

In the literature at least one earlier suggestion for using RRO for similar tasks can be found, viz. Page [1991], who suggests their use for controlling communication between intelligent artificial agents. However, I have not found whether Page has carried out his suggestion. Vreeswijk [1995] also refers to [Stary, 1991], who would have made a similar suggestion, but I have not been able to trace that publication. Formal and computational aspects of legal procedures have also been studied by Vreeswijk [e.g. 1995, 1996], who has attempted to formalize aspects of Peter Suber's [1990] 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 RRO contains rules for changing the rules of order (although these rules are left out of the present report). Finally, Gordon [1994,1995] has formalized and implemented his normative model of procedural justice in civil pleading, which became a source of inspiration of the ZENO project.

[2]And it needs to be slightly modernized; see e.g. [Robert, 1986, p. 118]: "The minutes should be neatly written with ink in the record book, leaving a margin for corrections ..."

Content of this report

The present paper reports on the ongoing formalization of RRO in first-order predicate logic, which is the first part of the ROBERT project. The complete project also consists of adapting the rules to electronic, asynchrounous group discussion, and implementing them as a component of ZENO's discussion forum. The present formalization is developed with this aim in mind.

In Chapter 2 an overview is given of Robert's Rules of Order, and it is specified which parts of RRO are to be included in ROBERT, and which of those parts have been formalized in this report. Then in Chapter 3 some implications are discussed of the intended implementation of ROBERT as a component of ZENO, after which in Chapter 4 the choice for first-order predicate logic as the specification formalism is motivated. That chapter also discusses how in a formalization of the ever changing world of meetings the notorious frame problem can be avoided. The chapters 5–8 then contain the heart of this report, a (partial) formalization of Robert's Rules of Order, which receives a preliminary evaluation in Chapter 9, as to how well it captures the normative aspects of RRO. Finally, the appendix contains some conceptual schemes.

Chapter 2

Robert’s Rules of Order: overview and relevant parts

This chapter gives overview of Robert’s Rules of Order is given, and then lists which parts of RRO are to be formalized in the ROBERT project, and which of those parts have already been formalized in this report.

2.1 Overview

Robert’s Rules of Order 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. [Page, 1991, 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 report are to a 1986 paperback publication of this text [Robert, 1986].

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.

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 society's purposes).

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 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. 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.

It seems obvious that this does not apply to all rules of RRO but only to certain less important formalities. Assume, for instance, that a motion that requires a 2/3 majority is in fact accepted by 56 % majority, without anyone present objecting. Now if mr X is a member of the society who was not present at the meeting, then it seems that mr X has a good case in court when appealing against the decision. However, if a trivial

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

motion has not been seconded and has, without a vote, been announced by the chair as accepted, and nobody present objects, then mr X's case in court seems without any chance.

2.2 What is to be included in ROBERT

The following points are to be formalised in the ROBERT project.

- Which procedural speech acts exist.
- When these acts are correctly performed.
- How an issue
  - can be brought before the assembly (making an appropriate speech act, seconded, stated).
  - is considered (debate opened, conducted, closed).
  - is dispensed with (decided, voted, or in some other way removed from before the assembly).
- The precedence order among motions.
- The procedural effect of motions (when made, when adopted and when rejected).
- How a session is opened, closed, adjourned and continued after adjournment.
- Orders of business.
- Rights and obligations of the chair.
- The quorum.
- Informal consideration of a question.
- How to change the rules of order, bye-laws or standing rules.

The main topics that will not be formalized are the rules on considering business in committees (RRO 28–32), decorum in debate (RRO 36), rights and duties of officers other than the chair (RRO 41), and the minutes (RRO 41).

2.3 What is formalized in this report

- Which procedural speech acts exist.

  All motions that are discussed in RRO have been formalized in a concept hierarchy, including the rules for determining the attribute values. This has not yet been done for the other acts, but many of these acts occur in rules of the present formalization.

- When these acts are correctly performed.

  The formalization divides this question into two subquestions: whether an act is in order (i.e. whether it is made at the correct moment: for instance, an appeal must be made directly after the appealed decision) and whether it is proper (i.e. whether it can be made at all in this form: for instance, only amendable motions can be amended). As for being in order, all conditions that hold for any kind of speech act have been formalized (Chapter 7). Order conditions that are special to certain motions have not yet been formalized. Conditions on whether an act is proper have not yet been formalized at all.

- How an issue

  – can be brought before the assembly (making an appropriate speech act, seconded, stated).

    This is fully formalized.

  – is considered (debate opened, conducted, closed).

    It is only formalized how a motion becomes open to debate.

  – is dispensed with (decided, voted, or in some other way removed from before the assembly).

    The cases where the chair has to decide have been formalized and the voting procedure has been partly formalized. The other ways of removing a question from before the assembly have been included in the motion hierarchy, but their procedural effects have not yet been formalized.

- The precedence order among motions.

  This has not yet been formalized.

- The procedural effect of motions (when made, when adopted and when rejected).

  This has not yet been formalized.

- How a session is opened, closed, adjourned and continued after adjournment.

  This has not yet been formalized.

- Orders of business.

  This has not yet been formalized.

- Rights and obligations of the chair.

  Some of the chair's obligations have been formalized in the parts on bringing a motion before the assembly, and on deciding and voting. The rest has not yet been formalized.

- The quorum.

  This has not yet been formalized.

- Informal consideration of a question.

  This has not yet been formalized.

- How to change the rules of order, bye-laws or standing rules.
  This has not yet been formalized.

Chapter 3

ROBERT as a component of ZENO

As stated in the introduction, the ROBERT system should ultimately be integrated with ZENO's discussion forum. This chapter 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 some design issues are then discussed, in particular with respect to how ROBERT should deal with violations of RRO by the users of ZENO's discussion forum (including the human mediator).

3.1 Two functions of ROBERT

As an implemented system, ROBERT can be included in the ZENO system for performing 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 RRO 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, or 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 present formalization should be such that it can be used for implementing both of these functions.

3.2 Tasks for ROBERT

To fulfill the just-sketched functions within ZENO, ROBERT should be able to perform at least the following two tasks:

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

In addition, ROBERT might be made to perform two further tasks:

3. Determine for any incorrect act which correct act might have been intended;
4. Determine how a certain procedural result can be obtained.

The third task probably requires that knowledge of a heuristic nature is added to ROBERT's knowledge base, while the fourth task requires the incorporation of a planner.

3.3 How ROBERT should deal with violations of RRO

Like any set of norms for human behaviour, the rules of RRO 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 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 that making a motion is impossible at this moment. Such an implementation of RRO would be what Jones & Sergot [1993] 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 RRO.

However, as Jones & Sergot remark, regimentation is not always a good idea, and ZENO's discussion forum is a good example. 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 RRO itself, which, as noted above, at various places formulates the principle that its formalities can be dispensed with as long as no member objects (e.g. RRO 1 and RRO 3, p. 34).

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 RRO;
2. It must be possible for the chair to set RRO aside when needed.

Let us look in more detail at the various ways in which RRO might be violated, and how ROBERT should deal with them.

Violations by ordinary participants

Ordinary participants can violate RRO 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).[1] 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 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), but to all participants, since all participants have the right to rise to a point of order (RRO 14). Furthermore, the 'sanction' for these kinds of violation is simply that the intended procedural effect does not occur. For instance, 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 chair can, in its role of the chair, violate RRO 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, 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 mover. A completely different kind of violation is when the chair incorrectly applies RRO: for instance, the chair incorrectly rules a motion out of order, or declares a motion 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 motion to end debate has been adopted, the next act of the chair must be putting the motion 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. The sanctions for this kind of violation are 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.

However, with the last kind of situation, erroneous application of RRO by the chair, things are different. Here it does not make much sense to formalize RRO in such a way that if, for instance, a participant starts debating a motion 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 RRO would make will have to be replaced by

[1]RRO does not explicitly distinguish these two notions; the distinction hase been introduced into the formalization to make it more structured.

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 RRO possible, so we must have at least some way of modelling erroneous application of RRO. Here is a sketch of a proposal, 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 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 RRO), or whether to sustain the input (i.e. to set RRO aside), in which case the system revises its state.

Note that this interaction procedure might actually 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.

Note also that when the chair sustains the erroneous input and opens the motion to debate, then in this 'subideal' state all other rules on RRO still apply to the motion, for instance, the rules on how to conduct debate, or on which motions can be made while another one is pending.

This idea is still sketchy, and its implementation involves several nontrivial technicalities, such as the belief revision procedure. Nevertheless, it seems to provide a good way for dealing with erroneous application of RRO by the chair, and at the same time for implementing the rule that 'anything goes until somebody objects'.

Summarizing this section on violations, two ways of dealing with violations will be used. For violations by ordinary participants, and for certain types of violations by the chair, a syntactic distinction will be made between required and actual behaviour, and the system will notify the chair and/or other participants when it detects a discrepancy between what happens and what should have happened. By contrast, a special type of violation by the chair, viz. erroneous application of RRO, is detected as a contradiction between procedural conclusions drawn by the system and those inputted by the chair.

Chapter 4

Choice of the formal specification tools

This paper's formalization of RRO is given with standard first-order logic. In the present chapter this choice will be motivated. Two main issues must be discussed: how to formalize the distinction between actual and required procedural behaviour, and how to formalize information about a constantly changing world.

4.1 Distinguishing actual and required behaviour

Above we required for several types of behaviour that ROBERT should be able to syntactically distinguish between actual and required behaviour. How can this be done? Various ways are possible, including the use of a full-fledged deontic logic. However, the present formalization stays within first-order logic. The normative character of RRO is captured by three special 'quasi-deontic' predicates, `Proper`, `In order`, `Correctly moved` and two surrogate deontic predicates, `Obliged to make` and `Obliged to decide or state`. The choice for this method is not irreversable, however, and therefore Chapter 9 will briefly compare it with a method using deontic logic.

The quasi-deontic predicates are convenient for formalizing prohibitions (`In order`) and obligations to make an act, if it is made, in a certain way (`Proper`). However, they are less suitable for obligations to perform a certain act, like the obligation for the chair to state a motion after it has been seconded. For such obligations the surrogate deontic predicate, `Obliged to make`, and in one case `Obliged to decide or state` will be used.

The use of quasi-deontic predicates is not so strange, since legal texts also often use such predicates, like 'tort' and 'criminal offence'. In fact, the Dutch criminal code hardly contains any deontic expression: it mainly defines the notion of criminal offence and several 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. On the other hand, the use of the surrogate deontic predicate `Obliged to make` is ad hoc, since no further definitions for this predicate are given. This will be further discussed in Chapter 9, which also briefly discusses an alternative style of formalizing RRO, with deontic logic.

4.2 Coping with the changing world of meetings

4.2.1 The choice of formalism

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 some 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 RRO.

In computer science and AI, formalizing changing worlds is a heavily studied topic. In computer science, many formalisms have been develop to specify the behaviour of computer programs, in order to prove their correctness or other properties, and as a basis for more operational specifications and for implementation (e.g. dynamic logic, petri nets, process algebra, temporal logic and several others). In AI the main focus has been on representing common-sense knowledge about actions and their effects in the world, modelled by e.g. situation calculus, event calculus, and again dynamic and temporal logic.

The present choice for standard first-order logic has not been made for deep reasons. Instead it was motivated by the short period of my stay at the GMD. The goal was to have a substantial part of the formalization finished by the end of this period, and therefore not much time could be wasted on investigating the pros and cons of the many available formalisms. Nevertheless, the choice is not final; when good reasons arise, the formalization can still be translated into another language. I am confident that this can be done without too much effort; if this is indeed the case, then the present specification in first-order logic has still proven useful.

4.2.2 The formalization method

4.3 Distinguishing actual and required behaviour

Above we required for several types of behaviour that ROBERT should be able to syntactically distinguish between actual and required behaviour. How can this be done? Various ways are possible, including the use of a full-fledged deontic logic. However, the present formalization stays within first-order logic. The normative character of RRO is captured by three special ‘quasi-deontic’ predicates, `Proper`, `In order`, `Correctly moved` and two surrogate deontic predicates, `Obliged to make` and `Obliged to decide or state`. The choice for this method is not irreversable, however, and therefore Chapter 9 will briefly compare it with a method using deontic logic.

The quasi-deontic predicates are convenient for formalizing prohibitions (`In order`) and obligations to make an act, if it is made, in a certain way (`Proper`). However, they are less suitable for obligations to perform a certain act, like the obligation for the chair to state a motion after it has been seconded. For such obligations the surrogate deontic predicate, `Obliged to make`, and in one case `Obliged to decide or state` will be used.

The use of quasi-deontic predicates is not so strange, since legal texts also often use such predicates, like ‘tort’ and ‘criminal offence’. In fact, the Dutch criminal code hardly contains any deontic expression: it mainly defines the notion of criminal offence and several 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. On the other hand, the use of the surrogate deontic predicate `Obliged to make` is ad hoc, since no further definitions for this predicate are given. This will be further discussed in Chapter 9, which also briefly discusses an alternative style of formalizing RRO, with deontic logic.

4.4 Coping with the changing world of meetings

4.4.1 The choice of formalism

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 some 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 RRO.

In computer science and AI, formalizing changing worlds is a heavily studied topic. In computer science, many formalisms have been develop to specify the behaviour of computer programs, in order to prove their correctness or other properties, and as a basis for more operational specifications and for implementation (e.g. dynamic logic, petri nets, process algebra, temporal logic and several others). In AI the main focus has been on representing common-sense knowledge about actions and their effects in the world, modelled by e.g. situation calculus, event calculus, and again dynamic and temporal logic.

The present choice for standard first-order logic has not been made for deep reasons. Instead it was motivated by the short period of my stay at the GMD. The goal was to have a substantial part of the formalization finished by the end of this period, and therefore not much time could be wasted on investigating the pros and cons of the many available formalisms. Nevertheless, the choice is not final; when good reasons arise, the formalization can still be translated into another language. I am confident that this can be done without too much effort; if this is indeed the case, then the present specification in first-order logic has still proven useful.

How can knowledge about states and state changes be formalized in first-order logic? This report uses the following method. 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 certain state is expressed as $\texttt{Pending question}(x, s)$, meaning that x is the pending question at s, and the speaker (who has the floor) in a certain state is expressed as $\texttt{Has floor}(y, s)$. 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 $\texttt{Motion}(m) \wedge \texttt{Seconded}(p, m, s)$. Cearly, this formalization method implies a commitment to a state-based ontology (as in situation calculus) instead of to an event-based ontology (as in event calculus).

Notational conventions

Let us define some notational conventions. Free variables are implicitly assumed to be universally quantified, as in logic programming. The material implication is denoted by $\Rightarrow$. 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. s' is the immediate successor or s and $'s$ its immediate predecessor. s^+ denotes a successor of s and s^- denotes a predecessor of s. In particular, for any s, s^+ and s^- it holds that $\texttt{Later}(s, s^+)$ and $\texttt{Later}(s^-, s)$ (Quantifiers in the rules and the definition of $\texttt{Later}$ will make this more precise). Person variables are written as y, y', ..., and variables for acts as x, x', ... or z, z', ... Finally, $\texttt{Type writer}$ strings are predicate symbols when they begin with a capital, otherwise they are function symbols.

Now the idea is that state changes are formalized with rules that have a term s in their antecedent predicates, and a term s' in their consequent predicates. For instance, the rule that a debatable motion becomes open to debate after it is stated by the chair can be written as

- $\texttt{Stated}(chair, x, s) \wedge \texttt{Debatable}(x) \Rightarrow \texttt{Open to debate}(x, s')$

However, readers familiar with the relevant AI literature will immediately recognize a problem, viz. the so-called ‘frame problem’ (see e.g. Shanahan [1997]). 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 next speaker at 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’:

- $\texttt{Open to debate}(x, s) \wedge \neg \ldots \Rightarrow \texttt{Open to debate}(x, s')$

where ... is the disjunction of all ways in which a motion ceases being open to debate. For various reasons this way of formalizing the effects of actions, where not only what has changed must be specified, but also what has not changed, is widely considered to be unattractive. In particular, the frame axioms are often quite complex, and reasoning with them is computationally expensive. Moreover, in actual common-sense reasoning it seems that it is simply assumed that things do not change, unless an explicit reason for change becomes known. This is a form of nonmonotonic reasoning (drawing plausible but uncertain conclusions in the absence of evidence of exceptions) and many nonmonotonic logics have been applied to model this kind of reasoning (including most applications of the situation calculus and the event calculus).

Can we use these nonmonotonic formalisms for our purposes? Unfortunately, none of these systems seems completely satisfactory: there are both theoretical problems and problems with computational efficiency. To avoid these problems, the present formalization uses a nonlogical, procedural component. The idea is that of any state of affairs that persists until it is changed by some event, the information is included in a data structure called the record. The record is not made relative to a state, but exists 'globally', and the values of its attributes are updated when needed: each time when the knowledge base derives a change in the value of some attribute (for instance, the pending question or the current speaker) its value on the record is changed. And each time when the logical reasoning process needs the value of a record attribute, a look-up at the record is performed. In more detail (illustrated with the pending question):

- The record is not represented as a logical theory but as a data structure that is seperate from the rest of the KB, i.e. in a different language.
- In the record the value of the above-mentioned predicates is recorded without referring to the state. So `Pending question`(m) instead of `Pending question`(m, s) (this shows that the languages of the KB and the record are different).
- When a conclusion `Pending question`(n, s) is derived from the knowledge base, then at the record the value of `Pending question` is changed from m to n (perhaps with other information on n, e.g. the type of question, the mover, and so on).
- A list is maintained of all questions that were once pending, with the begin and (if already known) end states. e.g. `Was pending`(m_1, s_1, s_2). This list can be inspected by the reasoner whenever information on past pending questions is needed. Simple functions must be implemented for determining, for instance, that a motion was pending within a certain interval.

Let us briefly compare this method with the logical alternative. An advantage of the purely logical formalization is that the history of a predicate like `Pending question` is available for reasoning: if ROBERT needs to know whether some motion was pending at any time, or at some particular time, it can find the answer with logical reasoning. By contrast, in our procedural representation the information that m was pending at s is lost as soon as at some s^+ some other motion becomes pending. This is the reason why the record must also keep track of the history of the meeting, in particular of the motions that were once pending. On the other hand, an advantage of the procedural method is that it makes the structure of the formalization easier; there is no need for closed world assumptions or for frame axioms.

Logicians might consider the procedural method to be ad hoc, but it can still be motivated on intuitive grounds. Think of a meeting where behind the chair stands a blackboard, at which the values of the record are written. Each time when 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 when the chair wants to know what is the pending question, s/he looks at the blackboard. This seems a perfectly natural way of conceiving what happens at actual meetings, so why not model it as directly as possible?

Let us finally list the predicates that in the present formalization are assumed to be part of the record.

- `Speaker` (or who `Has floor`). This says who is the speaker, if any.
- `Question stack`. This lists the motions that at any state are before the assembly (debated or decided), in the process of 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 either the motion that is in the process of being brought before the assembly, or being debated, or being decided.
- `Open to debate`. This says which motion is currently debated, if any.
- `Open to vote`. This says which motion is currently voted on, if any. For any motion x that is open to vote, there are two attributes *ayes*(x) and *noes*(x), which record the positive and negative votes cast.
- `Order of business`. This is a list of questions that must be dealt with, in a certain order. This list determines the pending question when no other events designate a pending question.
- `Orders of the day`. Some questions are assigned a special time. They become the pending question as soon as the chair announces them to be so, or when a member successfully makes the motion 'Call for the orders of the day'.
- `Motions to be called up`. Some motions are 'entered on the record' when made and seconded, and must be called up by a member in order to become the pending question (at present this only holds for `Motion to reconsider`, 27 RRO). Such a call does not need a vote (in contrast to motions to take from the table, or calls for the orders of the day).
- `Table`. This says which questions lay on the table. This predicate is invoked by the motions 'Lay on the table' and 'Take from the table'.
- `Postponed`. These are the motions that are postponed (to a certain time or indefinetely)
- `In committee`. This lists the motions that have been referred to a committee.
- `History`. This records the procedural acts that have been made during a session, as well as the decisions on the motions made.

Chapter 5

The formalization: definitions

This chapter defines several predicates that are used in the formalization. The first definitions define "Of type X and open for making" as "Open for X-ing". These predicates are about situations where only acts of a certain type can be made (for example, a seconding just after a motion has been correctly moved). For these cases the predicate `Open for making` is useful for expressing general rules about this situation (for instance, if an act that is open for making at s is prevented by some illegal act, it is still open for making at s'), while the special predicates, e.g. `Open for seconding`, make the antecedents of specific rules more readable. See Section 7.2.6 for a detailed discussion of the `Open for making` predicate.

** Note on implementation:*
If the specification is implemented in a rule-based language with a chaining-like inference mechanism, then only the if-part of the definitions should be stated, while their consequents (the 'Open for X-ing' predicates), should only be used in the antecedents of other rules. This is to avoid loops, as will be explained below.
** End note on implementation*

As for the intuitive reading of the various predicates, the first rule below reads as 'If x is a motion and x is open for making for person y at state s, then x is open for moving for person y at state s'. And the atomic formula `Seconding`(x, z) reads as 'x is a seconding of (motion) z'. Likewise for all other twoplace predicates.

- `Motion`$(x) \wedge$ `Open for making`$(y, x, s) \Leftrightarrow$ `Open for moving`(y, x, s)
- `Seconding`$(x, z) \wedge$ `Open for making`$(y, x, s) \Leftrightarrow$ `Open for seconding`(y, z, s)
- `Stating`$(x, z) \wedge$ `Open for making`$(y, x, s) \Leftrightarrow$ `Open for stating`(y, z, s)
- `Appeal`$(x, z) \wedge$ `Open for making`$(y, x, s) \Leftrightarrow$ `Open for appeal`(y, z, s)
- `Act to obtain floor`$(x) \wedge$ `Open for making`$(y, x, s) \Leftrightarrow$ `Open for obtaining floor`(y, x, s)
- `Yield floor`$(x) \wedge$ `Open for making`$(y, x, s) \Leftrightarrow$ `Open for yielding floor`(y, x, s)
- `Decision`$(x, z) \wedge$ `Open for making`$(y, x, s) \Leftrightarrow$ `Open for decision`(y, z, s)

-Act to obtain floor by mover$(x) \wedge$ Open for making$(y, x, s) \Leftrightarrow$ Open for obtaining floor by mover(y, x, s)
-Act to yield floor by mover$(x) \wedge$ Open for making$(y, x, s) \Leftrightarrow$ Open for yielding floor by mover(y, x, s)
-Entering on the record$(x, z) \wedge$ Open for making$(y, x, s) \Leftrightarrow$ Open for entering on the record(y, z, s)
-Putting affirmative$(x, z) \wedge$ Open for making$(y, x, s) \Leftrightarrow$ Open for putting affirmative(y, z, s)
-Putting negative$(x, z) \wedge$ Open for making$(y, x, s) \Leftrightarrow$ Open for putting negative(y, z, s)
-Decision or stating$(x, z) \wedge$ Open for making$(y, x, s) \Leftrightarrow$ Open for decision or stating(y, z, s)
-Withdrawal$(x, z) \wedge$ Open for making$(y, x, s) \Leftrightarrow$ Open for withdrawal(y, z, s)
-Affirmative vote$(x, z) \wedge$ Open for making$(y, x, s) \Leftrightarrow$ Open for affirmative voting(y, z, s)
-Negative vote$(x, z) \wedge$ Open for making$(y, x, s) \Leftrightarrow$ Open for negative voting(y, z, s)

Next we make similar rules for the various Obliged to ... predicates.

-Stating$(x, z) \wedge$ Obliged to make$(y, x, s) \Leftrightarrow$ Obliged to state(y, z, s)
-Decision$(x, z) \wedge$ Obliged to make$(y, x, s) \Leftrightarrow$ Obliged to decide(y, z, s)
-Entering on the record$(x, z) \wedge$ Obliged to make$(y, x, s) \Leftrightarrow$ Obliged to enter on the record(y, z, s)
-Putting affirmative$(x, z) \wedge$ Obliged to make$(y, x, s) \Leftrightarrow$ Obliged to put affirmative(y, z, s)
-Putting negative$(x, z) \wedge$ Obliged to make$(y, x, s) \Leftrightarrow$ Obliged to put negative(y, z, s)
-Decision or stating$(x, z) \wedge$ Obliged to make$(y, x, s) \Leftrightarrow$ Obliged to decide or state(y, z, s)

The following rules abbreviate “made an action of type X” to “X-ed”. For motions, we retain the term denoting the motion as an argument of the X-ed predicate. So, for instance, Moved to adjourn(y, x, s) means that person y moved to adjourn by motion to adjourn x at state s, and Appealed(y, x, z, s) means that person y appealed by appeal x against decision z at state s. And Decision(x, z) means that x is a decision of question z.

-Motion$(x) \wedge$ Made$(y, x, s) \Leftrightarrow$ Moved(y, x, s)
-Seconding$(x, z) \wedge$ Made$(y, x, s) \Leftrightarrow$ Seconded(y, z, s)
-Stating$(x, z) \wedge$ Made$(y, x, s) \Leftrightarrow$ Stated(y, z, s)
-Appeal$(x, z) \wedge$ Made$(y, x, s) \Leftrightarrow$ Appealed(y, x, z, s)
-Act to obtain floor$(x) \wedge$ Made$(y, x, s) \Leftrightarrow$ Acted to obtain floor(y, s)
-Yield floor$(x) \wedge$ Made$(y, x, s) \Leftrightarrow$ Yielded floor(y, s)
-Decision$(x, z) \wedge$ Made$(y, x, s) \Leftrightarrow$ Decided(y, z, s)
-Motion to reconsider$(z, x) \wedge$ Made$(y, x, s) \Leftrightarrow$ Moved to reconsider(y, z, x, s)

- Motion to fix time of adjournment$(x) \wedge$ Made$(y, x, s) \Leftrightarrow$ Moved to fix time of adjournment(y, x, s)
- Motion to adjourn$(x) \wedge$ Made$(y, x, s) \Leftrightarrow$ Moved to adjourn(y, x, s)
- Call for the orders of the day$(x) \wedge$ Made$(y, x, s) \Leftrightarrow$ Called for the orders of the day(y, x, s)
- Motion to suspend the rules$(x, r) \wedge$ Made$(y, x, s) \Leftrightarrow$ Moved to suspend the rules(y, x, r, s)
- Motion to lay on the table$(x, z) \wedge$ Made$(y, x, s) \Leftrightarrow$ Moved to lay on the table(y, x, z, s)
- Previous question$(x, z) \wedge$ Made$(y, x, s) \Leftrightarrow$ Moved the previous question(y, x, z, s)
- Act to call up$(x, z) \wedge$ Made$(y, x, s) \Leftrightarrow$ Acted to call up(y, x, z, s).
- Entering on the record$(x, z) \wedge$ Made$(y, x, s) \Leftrightarrow$ Entered on the record(y, z, s)
- Putting affirmative$(x, z) \wedge$ Made$(y, x, s) \Leftrightarrow$ Affirmative put(y, z, s)
- Putting negative$(x, z) \wedge$ Made$(y, x, s) \Leftrightarrow$ Negative put(y, z, s)
- Decision or stating$(x, z) \wedge$ Made$(y, x, s) \Leftrightarrow$ Decided or stated(y, z, s)
- Motion$(x) \wedge$ Correctly made$(y, x, s) \Leftrightarrow$ Correctly moved(y, x, s)
- Withdrawal$(x, z) \wedge$ Made$(y, x, s) \Leftrightarrow$ Withdrawn(y, z, s)

Finally, here is a definition of a slightly different type.

- Acted to call up$(y, x, z, s) \Leftrightarrow$ Called up(z, s)

Chapter 6

The formalization: the motion hierarchy

In this chapter the motion hierarchy of RRO is logically specified. A graphical overview is contained in the appendix. In later phases of the ROBERT project the hierarchy is to be extended with all types of procedural speech acts of RRO. The motions are represented as an inheritance hierarchy with exceptions, where each class has at most one superclass. As is well-known, the latter condition prevents ‘clashes of intuition’. Some attribute values are given directly, others by way of rules. When attribute values are explicitly specified for a certain class, the values of the same attribute of its superclass do not apply to the subclass. Note this way of formalizing themotion hierarchy does not commit to nonmonotonic reasoning methods; it is possible to translate the hierarchy into standard first-order logic, as will be shown below.

References to RRO

The formalization in this and the following three chapters contains many references to [Robert, 1986]. Expressions n RRO, RRO n, or just n, refer to sections of [Robert, 1986], and sometimes they are followed by page references to [Robert, 1986]. Alternatively, it is sometimes said that a certain formalization is based on the structure of RRO, or on common sense.

Format of the hierarchy

Let us now turn to the precise format of the motion hierarchy. Each type of motion is specified with the following scheme:

Type: $\texttt{Predicate}(x_1, \ldots, x_n)$ (Informal reading of predicate)

Superclass: [An act predicate]
Attributes:
- $(\neg)$ $\texttt{Attribute}_1(x_1, \ldots, x_m)$ $(m \in \{1, 2\})$ (Informal reading of attribute)
- ...

Rules:
A list of rules for determining attribute values.

When attributes are twoplace, the second argument is almost always a state variable, to capture that the value of the attribute depends on the state of the meeting.

This scheme translates into standard first-order formulas in the following way. Superclass relations are defined by rules

$\texttt{Type}(x) \Rightarrow \texttt{Superclass}(x)$

And attribute values, when not given by specific rules, are given as rules

$\texttt{Type}(x) \wedge \neg\, \texttt{Exception}_1(x) \wedge \ldots \wedge \neg\, \texttt{Exception}_n(x) \Rightarrow (\neg)\, \texttt{Attribute}_i(x)$

Here each $\texttt{Exception}_i(x)$ refers to an exception holding for a subclass. For more structured ways of formalizing exceptions see e.g. Prakken [1997].

6.1 Motions

Type: $\texttt{Motion}(x)$ (x is a motion)
Superclass: `Act`

Attributes:
- $\texttt{Debatable}(x)$ (motions are debatable)
- $\neg$ $\texttt{In order when another has floor}(x, s)$ (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 (See e.g. `Question of privilege`)
- $\texttt{Requires second}(x)$ (motions require a second)
- $\texttt{Required majority}(x, simple)$ (The required vote for motions is a simple majority)
- $\texttt{Decision mode}(x, vote)$ (motions are decided by vote (alternative: by chair's decision))

- `Applicable to it`(z,x)? See rules. (all except postpone indefinetely, 24)
- `Renewable`(x,s)? See rules. (Motions are renewable after the introduction of any motion that alters the state of affairs, 26).
- `Reconsiderable`(x) (motions are reconsiderable)
- $\neg$ `To be entered on the record when made` (motions need not be entered on the record when made (only exception: reconsider))

The attribute `Applicable to it` is intended to capture the subsidiary motion applicable to a motion.

Rules:
* *Formalizing* `Applicable to it`
-`Motion`$(x) \wedge$ `Subsidiary`$(z,x) \wedge \neg$ `Motion to postpone indefinetely`(z,x) $\Rightarrow$ `Applicable to it`(z,x)
-`Motion`$(x) \wedge$ `Motion to postpone indefinetely`$(z,x) \Rightarrow \neg$ `Applicable to it`(z,x) (See 24 RRO)

* *Formalizing* `Renewable`
This is formalized with a predicate `Altered state of affairs`(x,s,s^+), which informally reads as 'the state of affairs concerning x is different in s^+ than in s'. The reason why it is not formalized with an act that actually changes the state of affairs at a state t between s and s^+ is that after t but before s^+ the situation might be restored in its original form at s, and this is very difficult to formalize.

-`Moved`$(x,s_1) \wedge$ `Later`$(s_2,s_1) \wedge$ `Decided`$(y,x,s_2) \wedge$ `Later`$(s_3,s_2) \wedge$ `Altered state of affairs`$(x,s_1,s_3) \wedge \neg$ `Renewed between`$(x,s_1,s_3) \Rightarrow$ `Renewable`(x,s_3)
-`Moved`$(x,s_1) \wedge$ `Later`$(s_2,s_1) \wedge$ `Decided`$(y,x,s_2) \wedge$ `Later`$(s_3,s_2) \wedge (\neg$ `Altered state of affairs`$(x,s_1,s_3) \vee$ `Renewed between`$(x,s_1,s_3)) \Rightarrow \neg$ `Renewable`(x,s_3)

There is an *interpretation problem* here. The above rule assumes that a motion can only be renewed if it has been decided, but there are other ways to dispense with a motion. Take, for instance, the case where a motion is postponed indefinetely: can such a motion also be renewed? Maybe there are other, similar cases.

The idea is that the predicate `Altered state of affairs` can be defined by further rules, or even in terms of changes of the record. Alternatively, it could be useful to leave the interpretation of this predicate to the user.

Here is the definition of the predicate `Renewed between`.

-`Moved`$(y,x,s_1) \wedge$ `Later`$(s_1,s_2) \wedge \exists x', y', s_3($`Between`$(s_1,s_3,s_2) \wedge$ `Correctly moved`$(y',x',s_3) \wedge$ `Of same content`$(x,x')) \Leftrightarrow$ `Renewed between`(x,s_1,s_2)

* *Note on formalizing 'reconsiderable?'*
According to 27 RRO there is a time limit on the possibility to reconsider: "during the day or the day after .." and according to 60 RRO (p. 196) must be reconsidered during the same session. However, this is a matter of the right order conditions. The attribute `Reconsiderable` states whether a motion is reconsiderable 'in principle', and for

reconsiderable motions further rules specify when a reconsideration is in order.
** End note on formalizing ‘reconsiderable?’*

6.2 Privileged motions

Type: `Privileged motion`(x) (9)
Superclass: `Motion`

Attributes:
- $\neg$ `Debatable`(x) (privileged motions are not debatable, 9,35)

Fix time of adjournment (10)

Type: `Motion to fix time of adjournment`(x)
Superclass: See rule

Attributes:
-`Applicable to it`(z, x)? See rules. (All but `Motion to postpone indefinetely`, special rule for `Amendment` when privileged.)

Rules:
- `Moved to fix time of adjournment`$(y, x, s) \wedge$ `Pending`$(x', s) \wedge x \neq x' \Rightarrow$ `Privileged motion`(x)
- `Moved to fix time of adjournment`$(y, x, s) \wedge \neg\exists x'$`Pending`$(x', s) \wedge x \neq x' \Rightarrow$ `Principal motion`(x)

These two rules say that the motion to fix time of adjournment is privileged if made when another motion is pending, otherwise, it stands as a principal motion.

- `Motion to fix time of adjournment`$(x) \wedge$ `Privileged motion`(x) $\wedge$
`Amendment`$(z, x) \wedge$ `Changes time of`$(z, x) \Rightarrow$ `Applicable to it`(z, x)
- `Motion to fix time of adjournment`$(x) \wedge$ `Privileged motion`(x) $\wedge$
`Amendment`$(z, x) \wedge \neg$ `Changes time of`$(z, x) \Rightarrow \neg$ `Applicable to it`(z, x)

These two rules say that the motion to fix time of adjournment (when privileged) is only amendable by amending the time.

- `Motion to fix time of adjournment`$(x) \wedge$ `Privileged motion`(x) $\wedge$

$\neg$ `Amendment`$(z, x) \wedge \neg$ `Motion to postpone indefinetely`$(z, x) \Rightarrow$ `Applicable to it`(z, x)
- `Motion to fix time of adjournment`$(x) \wedge$ `Privileged motion`(x)
$\wedge$ `Motion to postpone indefinetely`$(z, x) \Rightarrow \neg$ `Applicable to it`(z, x)

These rules regulate the case where the subsidiary is not an amendment; then it is applicable to `Fix time of adjournment` iff it is not `Motion to postpone indefinetely`.

- *Comment 1:* A problem was how to formalize that this motion is only privileged if made when no other motion is pending. The alternative is to distinguish two types of motions, `Motion to fix time of adjournment (when other motion is pending)` and `Motion to fix time of adjournment (when no other motion is pending)`. The present formulation has been chosen since it allows reasoning about whether a certain instance of 'fix time of adjournment' is privileged or not: this seems a question that a chairman often has to answer.
- *Comment 2:* The text of 10 RRO contradicts with the list on p.12 RRO on whether this motion is amendable. 'Amendable' has been chosen since the table on p. 11 does not say that it cannot be amended. And various WWW pages also say that this motion is amendable. 10 RRO seems to be a trivial error.

Adjourn (11)

Type: `Motion to adjourn`(x)
Superclass: `Privileged motion`

Attributes:
- `Applicable to it`(z, x)? See rule.
- `Renewable`(x)? See rule. (Renewable iff there has been progress in debate, or any business transacted, RRO 26).
- $\neg$ `Reconsiderable`(x)

Rules:
- `Motion to adjourn`$(x) \wedge$ `Subsidiary`$(z, x) \Rightarrow \neg$ `Applicable to it`(z, x)

- `Moved to adjourn`$(y, x, s_1) \wedge$ `Later`$(s_2, s_1) \wedge$ `Business transacted between`$(s_1, s_2) \wedge \neg$ `Renewed before`$(x, s_2) \Rightarrow$ `Renewable`(x, s_2)
- `Moved to adjourn`$(y, x, s_1) \wedge$ `Later`$(s_2, s_1) \wedge \neg$ `Business transacted between`$(s_1, s_2) \Rightarrow \neg$ `Renewable`(x, s_2)
- `Moved to adjourn`$(y, x, s_1) \wedge$ `Later`$(s_2, s_1) \wedge$ `Renewed before`$(x, s_2) \Rightarrow$
$\neg$ `Renewable`(x, s_2)

- `Progress in debate between`$(s_1, s_2) \Rightarrow$ `Business transacted between`(s_1, s_2)

Here is the definition of the predicate Renewed before.

- Moved$(y, x, s_1) \wedge$ Later$(s_1, s_2) \wedge \exists x', y', s_3($Between$(s_1, s_3, s_2) \wedge$ Correctly moved$(y', x', s_3) \wedge$ Of same content$(x, x')) \Leftrightarrow$ Renewed before(x, s_2)

Question of privilege (12)

This motion concerns a question relating to the rights and privileges of the assembly, or any of its members.

Type: Question of privilege(x)
Superclass: Privileged motion

Attributes:
- Debatable(x) (9,35)
- In order when another has floor(x, s)? (see rule)
- Applicable to it(z, x)? See rule (all)

Rules:
- Question of privilege$(x) \wedge$ Requires immediate action$(x, s) \Rightarrow$ In order when another has floor(x, s)
- Question of privilege$(x) \wedge \neg$ Requires immediate action$(x, s) \Rightarrow \neg$ In order when another has floor(x, s)

- Question of privilege$(x) \wedge$ Subsidiary$(z, x) \Rightarrow$ Applicable to it(z, x)

Call for the orders of the day (13)

Some motions are assigned to a special time. When that time comes, this motion can be used to call them up.

Type: Call for the orders of the day(x)
Superclass: Privileged motion

Attributes:
- In order when another has floor(x, s)
- $\neg$ Requires second(x) (3,13)
- Applicable to it(z, x)? See rule (all but Amendment and Motion to postpone indefinetely)
- Renewable(x, s)? (See rule, RRO 61, p. 198)

Rules:

-Call for the orders of the day$(x) \wedge$ Subsidiary$(z,x) \wedge \neg$ Amendment(z,x) $\wedge \neg$ Motion to postpone indefinetely$(z,x) \Rightarrow$ Applicable to it(z,x)
-Call for the orders of the day$(x) \wedge$ Amendment$(z,x) \Rightarrow \neg$ Applicable to it(z,x)
-Call for the orders of the day$(x) \wedge$ Motion to postpone indefinetely$(z,x) \Rightarrow \neg$ Applicable to it(z,x)

-Called for the orders of the day$(y,x,s_1) \wedge$ Later$(s_2,s_1) \wedge$ Pending(x',s_1) $\wedge \neg$ Business(x',s_2) $\neg$ Renewed before$(x,s_2) \Rightarrow$ Renewable(x,s_2)
-Called for the orders of the day$(y,x,s_1) \wedge$ Later$(s_2,s_1) \wedge$ Pending(x',s_1) $\wedge$ Business$(x',s_2) \Rightarrow \neg$ Renewable(x,s_2)
-Called for the orders of the day$(y,x,s_1) \wedge$ Later$(s_2,s_1) \wedge$ Renewed before$(x,s_2) \Rightarrow \neg$ Renewable(x,s_2)

These rules say that this motion is renewable iff the question that was pending when it was first made, has been dealt with (61, p. 198)

6.3 Incidental motions

Type: Incidental motion(x) (8)
Superclass: Motion

Attributes:
- $\neg$ Debatable(x)
- Applicable to it(z,x)? See rules (all subsidiaries except Amendment and Motion to postpone indefinetely)

Rules:
- Incidental motion$(x) \wedge$ Subsidiary$(z,x) \wedge \neg$ Amendment$(z,x) \wedge \neg$ Motion to postpone indefinetely$(z,x) \Rightarrow$ Applicable to it(z,x)
-Incidental motion$(x) \wedge$ Amendment$(z,x) \Rightarrow \neg$ Applicable to it(z,x)
- Incidental motion$(x) \wedge$ Motion to postpone indefinetely(z,x)
$\Rightarrow$
$\neg$ Applicable to it(z,x)

Question of order (14)

Type: Question of order(x)
Superclass: Incidental motion

Attributes:
In order when another has floor(x, s)
- $\neg$ Requires second(x) (3)
- Decision mode$(x, chair)$
- Applicable to it(z, x)? See rules.

Rules:
- Question of order$(x) \wedge$ Subsidiary$(z, x) \Rightarrow \neg$ Applicable to it(z, x)

Appeal (14)

An appeal can be made to any decision of the chair.

Type: Appeal(x, z)
Superclass: Incidental motion

Attributes:
- In order when another has floor(x, s) (See table on p.11)
- Debatable(x)? See rules.
- Applicable to it(z, x)? See rules. (only Lay on the table and Previous question, but only if appeal is debatable)

Rules:
- Appeal$(x, z) \wedge$ Relates to indecorum$(x) \Rightarrow \neg$ Debatable(x)
- Appeal$(x, z) \wedge$ Relates to transgressions of rules of speaking(x) $\Rightarrow \neg$ Debatable(x)
- Appeal$(x, z) \wedge$ Relates to priority of business$(x) \Rightarrow \neg$ Debatable(x)
- Appeal$(x, z) \wedge$ Made$(x, s) \wedge$ Previous question$(z) \wedge$ Pending$(z, s) \Rightarrow$ $\neg$ Debatable(x)
- Appeal$(x, z) \wedge \neg$ Relates to indecorum$(x) \wedge \neg$ Relates to transgressions of rules of speaking$(x) \wedge \neg$ Relates to priority of business(x)
$\wedge$
$(\neg$ Previous question$(z) \vee \neg$ Pending$(z, s)) \Rightarrow$ Debatable(x)

- Appeal$(x, z) \wedge$ Debatable$(x) \wedge$ Lay on the table$(z, x) \Rightarrow$ Applicable to it(z, x)
- Appeal$(x, z) \wedge$ Debatable$(x) \wedge$ Previous question$(z, x) \Rightarrow$ Applicable to it(z, x)

- Appeal$(x, z) \wedge$ Debatable$(x) \wedge$ Subsidiary$(z, x) \wedge \neg$ Lay on the table(z, x) $\wedge \neg$ Previous question$(z, x) \Rightarrow \neg$ Applicable to it(z, x)

- Appeal$(x, z) \wedge \neg$ Debatable$(x) \wedge$ Subsidiary$(z, x) \Rightarrow \neg$ Applicable to it(z, x)

Next we list the rules for when a chair's decision becomes open for appeal.

-`Motion`$(x) \wedge$ `Decision mode`$(x, chair) \wedge$ `Decided`$(chair, x, s) \Rightarrow$ `Open for appeal`(y, x, s')
-`Motion`$(x) \wedge$ `Open for appeal`$(y, x, s) \wedge \neg$ `Appeal`$(z, x) \Rightarrow \neg$ `Open for making`(y, z, s)

These rules (14,61) say that when the chair has just made a decision, appeal against the decision must and can be made at the next moment.

Objection to the consideration of a question (15)

The objective of this motion is to entirely remove the subject from before the assembly.

Type: `Objection to the consideration of a question`(x, z)
Superclass: `Incidental motion`

Attributes:
-`In order when another has floor`(x, s)
-`Applicable to it`(z, x)? See rules. (None)
-$\neg$ `Requires second`(x)
-`Required majority`$(x, 2/3)$

Rules:
-`Objection to the consideration of a question`$(x, z) \wedge$ `Subsidiary`(z', x)
$\Rightarrow \neg$ `Applicable to it`(z', x)

Request to read papers (16)

Type: `Request to read papers`(x)
Superclass: `Incidental motion`

Withdrawal of a motion (17)

Type: `Withdrawal of a motion (17)`(x, z)
Superclass: `Incidental motion`

Suspension of the rules (18)

Type: `Motion to suspend the rules`(x, r)
Superclass: `Incidental motion`

Attributes:
- `Required majority`(x, z)? See rule
- `Applicable to it`(z, x)? See rules (None)
- `Renewable`(x, s)? See rule
- $\neg$ `Reconsiderable`(x)

Rules:
-`Motion to suspend the rules`$(x, r) \wedge$ `Deprives maximally one-third of members of right`$(x) \Rightarrow$ `Required majority`$(x, unanimous)$
-`Motion to suspend the rules`$(x, r) \wedge \neg$ `Deprives maximally one-third of members of right`$(x, r) \Rightarrow$ `Required majority`$(x, 2/3)$

-`Motion to suspend the rules`$(x, r) \wedge$ `Subsidiary`$(z, x) \Rightarrow \neg$ `Applicable to it`(z, x)

-`Moved to suspend the rules`$(y, x, r, s_1) \wedge$ `Later`$(s_2, s_1) \wedge$ `Adjournment`(z, s_3) $\wedge$ `Between`(s_3, s_1, s_2) $\neg$ `Renewed before`$(x, s_2) \Rightarrow$ `Renewable`(x, s_2)
-`Moved to suspend the rules`$(y, x, r, s_1) \wedge$ `Later`$(s_2, s_1) \wedge$ `Adjournment`(z, s_3) $\wedge \neg$ `Between`$(s_3, s_1, s_2) \Rightarrow \neg$ `Renewable`(x, s_2)
-`Moved to suspend the rules`$(y, x, r, s_1) \wedge$ `Later`$(s_2, s_1) \wedge$ `Renewed before`$(x, s_2) \Rightarrow \neg$ `Renewable`(x, s_2)

-`Motion to suspend the rules`$(x, r) \wedge$ `Motion to suspend the rules`(x', r) $\wedge$ `Same purpose`$(x, x') \Rightarrow$ `Of same content`(x, x')
-`Motion to suspend the rules`$(x, r) \wedge$ `Motion to suspend the rules`(x', r) $\wedge \neg$ `Same purpose`$(x, x') \Rightarrow \neg$ `Of same content`(x, x')

These rules interpret the concept ‘suspensions of the same rule for the same purpose’ in 18 RRO as ‘suspensions with the same content’.

6.4 Subsidiary motions

Type: `Subsidiary motion`(x) (7)
Superclass: `Motion`

Lay on the table (19)

The objective of this motion is to temporarily lay a question aside, in order to take up more urgent business.

Type: `Motion to lay on the table`(x, z)
Superclass: `Subsidiary motion`

Attributes:
- $\neg$ `Debatable`(x)
- `Applicable to it`(z, x)? See rules (None)
- `Reconsiderable`(x)? See rule

Rules:
-`Motion to lay on the table`$(x, z) \wedge$ `Subsidiary`$(z, x) \Rightarrow \neg$ `Applicable to it`(z, x)

-`Moved to lay on the table`$(y, x, z, s) \wedge \exists x^+$ `Accepted`$(x, s^+) \Rightarrow$ $\neg$ `Reconsiderable`(x)
-`Moved to lay on the table`$(y, x, z, s) \wedge \neg \exists x^+$ `Accepted`$(x, s^+) \Rightarrow$ `Reconsiderable`(x)

Previous question (20)

This is a motion to immediately end debate on the pending question and put it to vote.

Type: `Previous question`(x, z)
Superclass: `Subsidiary motion`

Attributes:
- $\neg$ `Debatable`(x)
- `Required majority`$(x, 2/3)$
- `Applicable to it`(z, x)? See rules (none)
- `Reconsiderable`(x)? See rule

Rules:
-`Previous question`$(x, z) \wedge$ `Subsidiary`$(z, x) \Rightarrow \neg$ `Applicable to it`(z, x)

The previous question is reconsiderable iff not (partly) executed. This is formalized as a special order condition:

-`Moved the previous question`$(y, x, z, s_1) \wedge$ `Later`$(s_2, s_1) \wedge$ `Moved to reconsider`$(y', z, x, s_2) \wedge$ `Before reconsideration deadline`$(s_2, x) \wedge$ $\neg$ `Partly executed`$(x, s_2) \Rightarrow$ `Special order conditions fulfilled`(z, s_2)

- `Moved the previous question`$(y, x, z, s_1) \wedge$ `Later`$(s_2, s_1) \wedge$ `Moved to reconsider`$(y', z, x, s_2) \wedge$ `Partly executed`$(x, s_2) \Rightarrow \neg$ `Special order conditions fulfilled`(z, s_2)

Postpone to a certain day (21)

Type: `Motion to postpone to a certain day`(x, z)
Superclass: `Subsidiary motion`

Attributes:
- `Debatable`(x)? See rule
- `Applicable to it`(z, x)? See rules (previous question, and rule for amendment)
- `Reconsiderable`(x)? See rule

Rules:
- This motion is debatable, but not further than necessary for enabling the assembly to judge the propriety of the postponement. This rule will be formalized when the rules on debate are formalized.

- `Motion to postpone to a certain day`$(x, z) \wedge$ `Previous question`(x', x) $\Rightarrow$ `Applicable to it`(x', x)

- `Motion to postpone to a certain day`$(x, z) \wedge$ `Amendment`$(x', x) \wedge$ `Changes time of`$(x', x) \Rightarrow$ `Applicable to it`(x', x)
- `Motion to postpone to a certain day`$(x, z) \wedge$ `Amendment`$(x', x) \wedge$ $\neg$ `Changes time of`$(x', x) \Rightarrow \neg$ `Applicable to it`(x', x)

These two rules say that a motion to postpone to a certain day is only amendable by amending the time.

- `Motion to postpone to a certain day`$(x, z) \wedge \neg$ `Previous question`(x', x) $\wedge \neg$ `Amendment`$(x', x) \Rightarrow \neg$ `Applicable to it`(x', x)

Refer to a committee (22)

The object of this motion is to have the motion clarified by a committee, before it can be dealt with by the assembly.

Type: `Motion to refer to a committee`(x, z)
Superclass: `Subsidiary motion`

Attributes:
- `Applicable to it`(z,x)? See rules (Only amendment, but see rule)

Rules:
- `Motion to refer to a committee`$(x,z) \wedge$ `Amendment`$(x',x) \wedge$ (`Alters committee`$(x',x) \vee$ `Instructs committee`$(x',x)) \Rightarrow$ `Applicable to it`(x',x)
- `Motion to refer to a committee`$(x,z) \wedge$ `Amendment`$(x',x) \wedge \neg$ `Alters committee`$(x',x) \wedge \neg$ `Instructs committee`$(x',x) \Rightarrow \neg$ `Applicable to it`(x',x)

These two rules say that a motion to refer to a committee is only amendable by altering the committee, or giving it instructions.

- `Motion to refer to a committee`$(x,z) \wedge \neg$ `Amendment`$(x',x) \Rightarrow$ $\neg$ `Applicable to it`(x',x)

Amendment (23)

Type: `Amendment`(x,z)
Superclass: `Subsidiary motion`

Attributes:
- `Applicable to it`(z,x)? See rules (Previous question, amendment (but see rule) and see rule)
$\neg$ `Renewable`(x,s)

Rules:
- `Amendment`$(x,z) \wedge$ `Amendment`$(x',x) \wedge \neg \exists z'$ `Amendment`$(z,z') \Rightarrow$ `Applicable to it`(x',x)
- `Amendment`$(x,z) \wedge$ `Amendment`$(x',x) \wedge \exists z'$ `Amendment`$(z,z') \Rightarrow \neg$ `Applicable to it`(x',x)

These rules say that an amendment may be amended, unless it is itself an amendment of an amendment.

- `Amendment`$(x,z) \wedge$ `Previous question`$(x',x) \Rightarrow$ `Applicable to it`(x',x)
- `Amendment`$(x,z) \wedge \neg$ `Previous question`$(x',x) \wedge \neg$ `Amendment`$(x'x) \Rightarrow$ $\neg$ `Applicable to it`(x',x)

Filling blanks (23)

Type: `Filling blanks`(x,z,b) ('motion x fills blanks b of motion z')

Superclass: `Amendment`

Attributes:
- $\neg$ `Requires second`(x)

Different proposals for filling blanks are not treated as amendments of each other, but as alternative amendments of the motion with the blanks (23 RRO, p. 69). This is formalized by giving the predicate `Filling blanks` a third argument, for the blanks of the motion denoted by the second argument.

Nomination (23)

Type: `Nomination`(x, y, z) ('motion x nominates person y for z')
Superclass: `Amendment`

Attributes:
- $\neg$ `Requires second`(x) (Since RRO 23, p. 69 says that they are treated in a similar manner as filling blanks.)

A second nomination is not an amendment of a previous one, but an independent amendment, to be treated separately.

Postpone indefinetely (24)

Type: `Motion to postpone indefinetely`(x, z)
Superclass: `Subsidiary motion`

Attributes:
- `Applicable to it`(z, x)? See rules (only previous question)

Rules:
-`Motion to postpone indefinetely`$(x, z) \wedge$ `Previous question`(x', x)
$\Rightarrow$
`Applicable to it`(x', x)
-`Motion to postpone indefinetely`$(x, z) \wedge \neg$ `Previous question`(x', x)
$\Rightarrow \neg$ `Applicable to it`(x', x)

6.5 Principal motions

Principal motions (6) usually concern the society's purposes and therefore usually do not have procedural effects, but only effects on the external world. Any motion that does not classify as another type of motion, is a principal motion. RRO recognizes some specific types of principal motions that have procedural effects, which are listed below. However, this list is not exhaustive; a principal motion can have any content.

Type: `Principal motion`(x)
Superclass: `Motion`

Attributes:
- `Applicable to it`(z, x)? See rule (all)
- $\neg$ `Renewable`(x, s)

Rules:
- `Principal motion`(x) $\wedge$ `Subsidiary motion`(x') $\Rightarrow$ `Applicable to it`(x', x)

Strictly speaking there is an *interpretation problem* here, since RRO does not explicitly comment on whether the latter rule holds for all subsidiaries. But implicitly RRO seems to assume throughout that this is the case.

Take from the table (19)

This motion can be used to order that motions laying on the table (19) are taken up again by the assembly.

Type: `Motion to take from the table`(x, z)
Superclass: `Principal motion`

Attributes:
- $\neg$ `Debatable`(x)
- `Applicable to it`(z, x)? See rules (None, RRO 19, pp. 54,56)
- `Reconsiderable`(x)? See rule (only when rejected)

Rules:
- `Motion to take from the table`(x, z) $\wedge$ `Subsidiary motion`(x') $\Rightarrow$ $\neg$ `Applicable to it`(x', x)

- `Moved to take from the table`(y, x, z, s) $\wedge$ $\exists x^+$ `Accepted`(x, s^+) $\Rightarrow$ $\neg$ `Reconsiderable`(x)
- `Moved to take from the table`(y, x, z, s) $\wedge$ $\exists x^+$ `Rejected`(x, s^+) $\Rightarrow$ `Reconsiderable`(x)

Rescind (25)

When an assembly wishes to annul some previous action and no other procedural resource is available, it can still rescind the action.

Type: `Motion to rescind`(x, z)
Superclass: `Principal motion`

Reconsider (26,27)

This motion can be used to ask for a new vote on a previously decided motion.

Type: `Motion to reconsider`(x, z)
Superclass: `Principal motion`

Attributes:
- `Debatable`(x)? See rules (when reconsidered motion is debatable)
- `In order when another has floor`(x, s)
- `Applicable to it`(z, x)? See rules (all except amendment)
- $\neg$ `Reconsiderable`(x)
- `To be entered on the record when made`(x)? See rules

Rules:
- `Motion to reconsider`$(x, z) \wedge$ `Debatable`$(z) \Rightarrow$ `Debatable`(x)
- `Motion to reconsider`$(x, z) \wedge \neg$ `Debatable`$(z) \Rightarrow \neg$ `Debatable`(x)

The following is an *interpretation problem*: if the reconsideration concerns a debatable motion to which the previous question applied at the moment of vote, and the previous question is not yet exhausted so the motion is not debatable (p. 60 RRO), is then the motion to reconsider also not debatable?

- `Motion to reconsider`$(x, z) \wedge$ `Amendment`$(x', x) \Rightarrow \neg$ `Applicable to it`(x', x)
- `Motion to reconsider`$(x, z) \wedge$ `Subsidiary motion`$(x') \wedge \neg$ `Amendment`(x', x) $\Rightarrow$ `Applicable to it`(x', x)

Another *interpretation problem* is that RRO does not explicitly comment on whether postpone indefinetely/to certain day and commit apply to this motion. (This is a special case of the interpretation problem for all principal motions.)

The following rules say that a motion to reconsider all motions other than incidental or subsidiary motions is to be entered on the record; and a motion to reconsider an incidental or a subsidiary motion is to be entered unless the vote on that motion had the effect of removing the entire subject from before the assembly (RRO 27, pp. 77–8). (This latter predicate has to be defined by further rules.)

-Motion to reconsider$(x, z) \wedge$(Incidental motion$(z) \vee$Subsidiary motion$(z)) \wedge \neg$ Removes subject from before assembly$(z) \Rightarrow \neg$ To be
entered on the record when made(x)
-Motion to reconsider$(x, z) \wedge$(Incidental motion$(z) \vee$Subsidiary motion$(z)) \wedge$ Removes subject from before assembly$(z) \Rightarrow$ To be entered on the record when made(x)
-Motion to reconsider$(x, z) \wedge \neg$Incidental motion$(z) \wedge \neg$Subsidiary motion$(z) \Rightarrow$To be entered on the record when made(x)

Extending the limits of debate (34)

Type: Motion to extend the limits of debate(x, z)
Superclass: Principal motion

Attributes:
-$\neg$ Debatable(x)
-Required majority$(x, 2/3)$

Leave to continue speaking after indecorum (36)

When the chair has ruled that a speaker has violated decorum in debate, then, if any member objects, the speaker can only continue after obtaining leave from the assembly.

Type: Leave to continue speaking after indecorum(x)
Superclass: Principal motion

Attributes:
-$\neg$ Debatable(x)
-Applicable to it(z, x)? See rules (all except amendment)

Rules:
-Leave to continue speaking after indecorum$(x) \wedge$Amendment(x', x)
$\Rightarrow \neg$Applicable to it(x', x)
- Leave to continue speaking after indecorum$(x) \wedge$ Subsidiary motion$(x') \wedge \neg$Amendment$(x', x) \Rightarrow$Applicable to it(x', x)

Limiting debate (37)

Type: `Motion to limit debate`(x, z)
Superclass: `Principal motion`

Attributes:
- $\neg$ `Debatable`(x)
- `Required majority`$(x, 2/3)$

Closing debate at certain time (37)

Type: `Motion to close debate at certain time`(x, z)
Superclass: `Principal motion`

Attributes:
- $\neg$ `Debatable`(x)
- `Required majority`$(x, 2/3)$

To take up a question out of its proper order (39)

Type: `Motion to take up a question out of its proper order`(x, z)
Superclass: `Principal motion`

Attributes:
- `Required majority`$(x, 2/3)$

Make a special order (13)

A special order makes a question part of the orders of the day in such a way that it can be taken up disregarding any rule that might interfere with its consideration at the time specified.

Type: `Motion to make a special order`(x)
Superclass: `Principal motion`

Attributes:
- `Required majority`$(x, 2/3)$

Make a general order (13)

A special order also makes a question part of the orders of the day, but it does not suspend rules that might interfere with its consideration at the time specified.

Type: `Motion to make a general order`(x)
Superclass: `Principal motion`

Ordering of ballot (38)

This motion orders for voting by ballot.

Type: `Ordering of ballot`(x, x')
Superclass: `Principal motion`

Ordering of roll call (38)

This motion orders for voting by roll call.

Type: `Ordering of roll call`(x, x')
Superclass: `Principal motion`

Chapter 7

The formalization: general conditions for when speech acts are in order

This chapter formalizes the two conditions for when a procedural act has been correctly made (viz. when in order and proper) and then formalizes the three order conditions that hold for all acts, and the two additional order conditions that hold for all motions.

7.1 On when an act is correctly made

One of the main tasks of the chair of any meeting is to determine whether an act of any member (including the chair!) violates any rule of order. To capture this, the present section introduces the notion of an act that is `Correctly made`. An act is correctly made if it is `In order` (made at the right moment) and `Proper` (has the right properties).

- $\texttt{Act}(x) \wedge \texttt{Made}(y,x,s) \wedge \texttt{In order}(x,s) \wedge \texttt{Proper}(x,s) \Rightarrow \texttt{Correctly made}(y,x,s)$
- $\texttt{Act}(x) \wedge \texttt{Made}(y,x,s) \wedge (\neg\,\texttt{In order}(x,s) \vee \neg\,\texttt{Proper}(x,s)) \Rightarrow \neg\,\texttt{Correctly made}(y,x,s)$

RRO is not clear on what it means by ‘being out of order’, and it does not even have the notion of an ‘improper’ action. In the present formalization these terms are interpreted as follows. When an act has the right characteristics of its kind, but is performed at the wrong moment, we say that it is out of order (for example, making a principal motion without having the floor), and when and act does not have the right properties, so when it can never be made in that way, we say that it is improper (for instance, amending an unamendable motion).

The conditions on properness are specific to any type of motion, and will therefore (in a later phase) be included in the motion hierarchy, as are the motion specific order conditions. However, there are also general order conditions, applying to any type of act. This general theory of order conditions is presented in the rest of this chapter, and in the appendix it is graphically displayed as an AND/OR graph.

7.2 On when an act is in order

In the present section the general rules for the `In order` predicate are given. First the top level rules for acts other than motions are given, and then the more specific top level rules for motions. In the rest of the section the conditions for these top level rules are defined. Some of these definitions are shared by motions and other acts, and some are specific to motions.

7.2.1 Acts other than motions: top level structure

For acts that are not motions, RRO's top level structure for being in order is as follows.

-Act$(x) \wedge \neg$ Motion$(x) \wedge$ Made$(y,x,s) \wedge$ Floor condition fulfilled(x,y,s) $\wedge$ Mode condition fulfilled$(x,s) \wedge$ Special order conditions fulfilled$(x,s) \Rightarrow$ In order(x,s)
-Moved$(y,x,s) \wedge (\neg$ Floor condition fulfilled$(x,y,s) \vee \neg$ Mode condition fulfilled$(x,s) \vee \neg$ Special order conditions fulfilled$(x,s)) \Rightarrow$ $\neg$ In order(x,s)

The informal meaning of the conditions is explained in the following subsection.

7.2.2 Motions: top level structure

As for determining when making a motion is in order, the top level structure of RRO is as follows.

-Moved$(y,x,s) \wedge$ Floor condition fulfilled$(x,y,s) \wedge$ Precedence condition fulfilled$(x,s) \wedge$ Renewal condition fulfilled$(x,s) \wedge$ Mode condition fulfilled$(x,s) \wedge$ Special order conditions fulfilled$(x,s) \Rightarrow$ In order(x,s)
-Moved$(y,x,s) \wedge (\neg$ Floor condition fulfilled$(x,y,s) \vee \neg$ Precedence condition fulfilled$(x,s) \vee \neg$ Renewal condition fulfilled(x,s) $\vee \neg$ Mode condition fulfilled$(x,s) \vee \neg$ Special order conditions fulfilled$(x,s)) \Rightarrow \neg$ In order(x,s)

The conditions of these rules have the following meaning.

-Floor condition fulfilled(x,y,s) (2) means that the rules concerning having the floor do not prevent making the act (either one has the floor, or having the floor is not required).

- `Precedence condition fulfilled`(x, 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 fulfilled`(x, s) (26, 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 fulfilled`(x, 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 act.
- `Special order conditions fulfilled`(x, s) means that any special conditions for the relevant type of act are fulfilled. For motions these special conditions will in a later phase of the project be added to the motion hierarchy.

The rest of this chapter gives the top level rules for when these conditions are fulfilled.

7.2.3 Floor condition

The rules for the floor condition are shared by motions and other acts.

- `Act`$(x) \wedge$ `Made`$(y, x, s) \wedge \neg$ `In order when another has the floor`(x, s) $\wedge$ `Has floor`$(y, s) \Rightarrow$ `Floor condition fulfilled`(x, y, s)
- `Act`$(x) \wedge$ `Made`$(y, x, s) \wedge$ `In order when another has the floor`(x, s) $\Rightarrow$ `Floor condition fulfilled`(x, y, s)

- `Act`$(x) \wedge$ `Made`$(y, x, s) \wedge \neg$ `In order when another has the floor`(x, s) $\wedge \neg$ `Has floor`$(y, s) \Rightarrow \neg$ `Floor condition fulfilled`(x, y, s)

These rules (2, p. 11) say that an act fulfills the floor condition iff it is made while having the floor or can be made without having the floor.

7.2.4 Precedence condition

The rules for the precedence condition are special to motions.

- `Moved`$(y, x, s) \wedge \neg \exists z$`Pending`$(z, s) \Rightarrow$ `Precedence condition fulfilled`(x, s)
- `Moved`$(y, x, s) \wedge$ `Pending`$(z, s) \wedge x \geq z \Rightarrow$ `Precedence condition fulfilled`(x, z)
- `Moved`$(y, x, s) \wedge$ `Pending`$(z, s) \wedge \neg\, x \geq z \Rightarrow \neg$ `Precedence condition fulfilled`(x, z)

This says that making a motion satisfies the precedence condition iff (p. 12) either no question is pending or the pending question yields to the motion. The notion of a pending question is defined in Section 8.3.

Note that the predicate $\geq$, which in $x \geq y$ stands for 'y yields to x', is not asymmetric: two motions can yield to each other, which means that the one can be made while the other is pending.

7.2.5 Renewal condition

The rules for the renewal condition are also special to motions. Making a motion meets its renewal condition iff either it is made for the first time, or it is renewable (26 RRO). In formalizing this, we want to say that if a motion m_1 with a certain content has been made at s_1 and another motion m_2 with the same content has been made at a later state s_2, then m_2 meets its renewal condition at s_2 iff m_1 is renewable at s_2. This can be expressed with with a predicate `Of same content`(x, x'), defined above in Section 6.1.

- `Moved`$(y, x, s) \wedge \neg \exists x', y', s^-($`Moved`$(y', x', s^-) \wedge$ `Of same content`$(x, x'))$
$\Rightarrow$ `Renewal condition fulfilled`(x, s)
- `Moved`$(y, x, s_1) \wedge$ `Moved`$(y', x', s_2) \wedge$ `Later`$(s_2, s_1) \wedge$ `Of same content`(x, x')
$\wedge$ `Renewable`$(x, s_2) \Rightarrow$ `Renewal condition fulfilled`(x', s_2)
- `Moved`$(y, x, s_1) \wedge$ `Moved`$(y', x', s_2) \wedge$ `Later`$(s_2, s_1) \wedge$ `Of same content`(x, x')
$\wedge \neg$ `Renewable`$(x, s_2) \Rightarrow \neg$ `Renewal condition fulfilled`(x', s_2)

7.2.6 Mode condition

The following rules deal with all cases in which at a certain state only acts of a special type are allowed to be made. For example, after a secondable motion has been correctly moved, no other act may be made than a seconding of this motion (unless that act is of some special type, further specified by RRO). These situations will be formalized with the predicate `Open for making`, or with specialized versions `Open for X-ing`, like `Open for seconding`.[1] The intuitive meaning of this predicate is that if an act of a certain type is open for making, all acts that are not of this type are not in order (with exceptions). In this subsection this is formalized, via the intermediate condition `Mode condition fulfilled`.

Note that an act that is open for making does not have to be obligatory: for instance, a motion does not have to be seconded. The situation that a certain act must be performed at a certain moment is formalized as a special case of openness for making, with an `Obliged to make` predicate. (See Chapter 9 for more comments on the representation of deontic concepts.)

** Note on implementation:*
If a chaining-like inference mechanism is chosen, then any rule of the form

$\ldots \Rightarrow$ `Open for X-ing`(y, x, s)

is decomposed into

$\ldots \wedge$ `Of type X`$(z) \Rightarrow$ `Open for making`(y, z, s)

Moreover, of the definitions of the `Open for X-ing` predicates in Chapter 5 only

[1]The advantage of using the general predicate in addition to the special ones is that several rules need be stated just once; see in more detail below.

the if-parts will be formulated. Together, these two policies should avoid loops.
** End note on implementation:*

As just said, there are certain types of acts that can also be made when another act is open for making. These are the acts that can be made without having the floor: they always satisfy the mode condition.

Note, finally, that when an act of a certain type is not open for making, this does not mean that making it does not satisfy the mode condition; it just means that it is not the case that any next act must be of this type. Only if at the same time another act is open for making, the first act does not satisfy the mode condition.

The first three subsections make use of the left-hand sides of the definitions in Chapter 5.

Top level structure

- `Act`$(x) \wedge$ `Open for making`$(y, x, s) \wedge$ `Made`$(y, x, s) \Rightarrow$ `Mode condition fulfilled`(x, s)
- `Act`$(x) \wedge$ `Made`$(y, x, s) \wedge$ `Mode condition exception`$(x, s) \Rightarrow$ `Mode condition fulfilled`(x, s)
- `Act`$(x) \wedge \neg$ `Open for making`$(y, x, s) \wedge$ `Open for making`$(y', z, s) \wedge$ `Made`(y, x, s) $\wedge \neg$ `Mode condition exception`$(x, s) \Rightarrow \neg$ `Mode condition fulfilled`(x, s)

- `In order when another has the floor`$(x, s) \Leftrightarrow$ `Mode condition exception`(x, s)

The first two rules say that an act satisfies the mode condition if it is made while open for making (structure of RRO, common-sense), or if it is an exceptional case. The third rule says that if an act is not open for making, but another act is open for making, then making it violates the mode condition, unless an exception exists for the act (structure of RRO). The final rule then says that the only such exceptions are acts that can be made without having the floor. (interpretation of RRO)

- `Act`$(x) \wedge$ `Made`$(y, x, s) \wedge \neg \exists y', x'$`Open for making`$(y', x', s) \Rightarrow$ `Mode condition fulfilled`(x, s)

This rule expresses (based on system of RRO) that if none of the special situations concerning the mode condition obtains, i.e. if no act of a special type is required, then the mode condition is (trivially) satisified.

Finally, we formalize the special case of obligatory acts. Acts that are obliged are also open for making.

- `Obliged to make`$(y, x, s) \Rightarrow$ `Open for making`(y, x, s)

The reason why acts that ought to be made (such as the chair's obligation to state a motion after it has been seconded) are not formalized with rules with a $\neg$ `In order`(y, x, s) consequent, is that when violated, the obligation should stay in effect at the next moment. This is easier to formalize with the `Obliged to make` predicate, as will be defined below.

How an act ceases being open for making

Next we must regulate when an act ceases being open for making. The simplest case is when the act is made.

** Making the act*

- $\texttt{Act}(x) \wedge \texttt{Open for making}(y, x, s) \wedge \texttt{Made}(y, x, s) \Rightarrow \neg\, \texttt{Open for making}(y', x, s')$

This rule says that an act ceases being open for making when it is actually made.

However, when an act that is open for making is not made, several situations must be distinguished. We must first deal with two cases in which the act was not performed because something else came in between.

** Prevented acts*

The first of these two cases is that at s some other act is 'illegally' made, which prevents the making of the 'open' act, and so 'prolonges' the opennes for making (e.g. an act that is out of order is made at s) (structure of RRO, common sense). The definition of being prevented is as follows.

- $\texttt{Act}(x) \wedge \texttt{Open for making}(y, x, s) \wedge \texttt{Act}(z) \wedge \texttt{Made}(y', z, s) \wedge \neg\, \texttt{Correctly made}(y', z, s) \Rightarrow \texttt{Prevented}(x, s)$
- $\texttt{Act}(x) \wedge \texttt{Open for making}(y, x, s) \wedge \neg\exists y', z\, (\texttt{Made}(y' z, s) \wedge \neg\, \texttt{Correctly made}(y', z, s)) \Rightarrow \neg\, \texttt{Prevented}(x, s)$

This rule says that an act that is open for making is prevented when another act occurs that is not open for making and that is not correctly made (not in order or not proper).

We now say that an act that was not made because it was prevented stays open for making.

- $\texttt{Act}(x) \wedge \texttt{Open for making}(y, x, s) \wedge \neg\, \texttt{Made}(y, x, s) \wedge \texttt{Prevented}(x, s) \Rightarrow \texttt{Open for making}(y, x, s')$

** Surpressed acts*

The second case with an interfering act is when that act interferes 'legally'. This is the case when that act can be made without the floor. For example, if x is open for making at s, but somebody rises to a point of order at s, then at s' the chair must decide the point of order. In such a case I say that x is surpressed: x temporarily ceases being open for making, until the other business has been dealt with. (all based on common-sense and on interpretation of the structure of RRO).

- $\texttt{Act}(x) \wedge \texttt{Open for making}(y, x, s) \wedge \texttt{Act}(z) \wedge \texttt{Correctly made}(y', z, s)$
$\wedge \neg\, \texttt{Of same type}(x, z) \Rightarrow \texttt{Surpressed}(x, z, s)$
- $\texttt{Act}(x) \wedge \texttt{Open for making}(y, x, s) \wedge \neg\exists y', z\, (z \neq x \wedge \texttt{Correctly made}(y', z, s))$
$\Rightarrow \neg\, \texttt{Surpressed}(x, z, s)$
- $\texttt{Surpressed}(x, z, s) \Rightarrow \neg\, \texttt{Open for making}(y, x, s)$

Unfortunately, the first of these rules needs an ad hoc device. This rule must cover the case where the act that was open for making for a person (say, a seconding) was not made because somebody else instead correctly performed that act (somebody else seconded the motion). In that case the act ceases being open for making by the first person (as regulated by the earlier-given rules). This is captured by the predicate `Of same type`. This is ad hoc for two reasons: the meaning of this predicate is not captured by the logic of the representation, as it should, and it is not certain whether this solution really covers all the cases: it is conceivable that a certain act is open for making by a person, and that the fact that another person correctly makes the same act still surpresses it. But RRO does not seem to contain such cases.

The following rules formalize that a surpressed act remains surpressed until the surpressing act has been dealt with, after which the surpressed act again becomes open for making.

- $\texttt{Surpressed}(x, z, s) \wedge \texttt{Business}(z, s') \Rightarrow \texttt{Surpressed}(x, z, s')$
- $\texttt{Surpressed}(x, z, s) \wedge \neg\, \texttt{Business}(z, s') \Rightarrow \neg\, \texttt{Surpressed}(x, z, s')$

- $\texttt{Surpressed}(x, z, s) \wedge \neg\, \texttt{Surpressed}(x, z, s') \Rightarrow \texttt{Open for making}(y, x, s')$

The following rule says that something is business iff it is in the question stack. For more on the latter notion, see Section 8.3. The idea is that the stack contains those motions that at any state are before the assembly (debated or decided), in the process of being brought before the assembly (the phase from being correctly moved to being stated), or temporarily set aside by another motion with higher precedence.

- $x \in \texttt{question stack}(s) \Leftrightarrow \texttt{Business}(x, s)$

In the following chapter, which is on introducing and dealing with business, it is for several acts defined how they become open for making.

** Failing to make when not surpressed and not prevented*

Finally, we must formalize the case of not-performed acts that were not prevented or surpressed. An act that was not made and not prevented ceases being open for making, unless the act was also obliged. The rationale of the latter exception is that when an obligatory act is not performed, it must still be performed at the next possible moment. For instance, when the chair forgets to state a seconded motion, or does something else instead, then at the next state s/he must still state the motion. The exceptional rule for obligatory acts in turn also has an exception, viz. when the obligatory act was surpressed: in that case the above rules for surpression apply, so that the act temporarily ceases open for making.

- Act(x) $\wedge$ Open for making(y, x, s) $\wedge$ $\neg$ Made(y, x, s) $\wedge$ $\neg$ $\exists x^-$ Obliged to make(y, x, s^-) $\wedge$ $\neg$ Prevented(x, s) $\Rightarrow$ $\neg$ Open for making(y, x, s')
- Act(x) $\wedge$ Open for making(y, x, s) $\wedge$ $\neg$ Made(y, x, s) $\wedge$ $\exists x^-$ Obliged to make(y, x, s^-) $\wedge$ $\neg$ Surpressed(x, z, s) $\Rightarrow$ Open for making(y, x, s')

Note that the first rule includes the case that there is silence at s (e.g. no one appeals or seconds). The last rule regulates the special case that an obligatory act was not performed. In order to make the formalization not too complicated, it is not the formula with `Obliged to make` that is prolonged from s to s'; only the implied formula with `Open for making` is prolonged, otherwise many of the rules for `Open for making` would have to be repeated for `Obliged to make`. Instead, the rule's condition says that the act was once obligatory in the past. But, of course, this is an ad hoc solution.

What if somebody (for instance, the chair) who is obliged to perform a certain act, keeps failing to perform it? In that case the openness for making keeps being prolonged, which seems to fit nicely with actual meetings. A procedural resource for members in such cases with reluctant chairs is to rise to a point of order (14).

Chapter 8

The formalization: introducing and dealing with business

This chapter formalizes how business can be brought before the assembly (Section 8.1), and how successfully introduced business can be decided (Section 8.2). The rules for debate have not yet been formalized, but the notion of a pending question is precisely defined, in Section 8.3.

8.1 On bringing a question before the assembly

There are two basic ways to bring business before the assembly (1 RRO): making a communication, and making a motion. This section formalizes the second way. The general procedure for bringing a question before the assembly is to first obtain the floor (with some exceptions) (2 RRO), and then (3 RRO) to make a motion, which has to be seconded by another member (with some exceptions), and has to be stated by the chair (except when the chair must decide the motion). (For debatable motions the mover must first be given the floor, 2 RRO, note at close, and 19 RRO, p. 56). The question is then open for decision, which can take place in three ways: decision by the chair (in a few designated cases), debate followed by vote (otherwise, when the motion is debatable), and immediate vote (otherwise, when the motion is not debatable).

Acts that should be performed by the chair are formalized using the `Obliged to make` predicate defined in Subsection 7.2.6. This formalization method makes it possible to represent that the chair has violated his or her obligations (as required in Chapter 3), and it makes it possible to deal with situations where the chair is temporarily unable to perform the required act, because of an interrupting act of a member (whether legal or illegal). In the latter case, the rules of Section 7.2.6 prolonge the obligatoriness of the act for being made.

Negative conditions for `Open for ...` predicates

In the rest of this report the `Open for ...` predicates are implicitly assumed to be *completed*. The completion of a predicate is a material implication with the negated predicate in the consequent, and with as antecedent the disjunction of the negations of antecedents of all positive rules for the predicate. (The only reason why these completion rules are left implicit is by way of notational convention; no negation-as-failure constructs are intended.)

We also implicitly include rules that say that if an act of a certain type is open for making, all acts that are not of this type are not open for making. To make one of these rules explicit by way of example, consider the implicit rule for a seconding.

- `Open for seconding`$(y, x, s) \wedge \neg$ `Seconding`$(z, x) \Rightarrow \neg$ `Open for making`(y', z, s)

Obtaining the floor

So far just one rule has been formalized, dealing with the case where the chair invites any speaker.

- `Invites speaker`$(chair, s) \Rightarrow$ `Open for obtaining floor`(y, s')
- `Open for obtaining floor`$(y, s) \wedge \neg$ `Act to obtain floor`$(x) \Rightarrow \neg$ `Open for making`(y, x, s)

These rules (common sense) say that when the chair has just asked who wants to speak next, any next act must be an act to obtain the floor (unless, of course, it satisfies the mode exception).

Making a motion

Recall from Section 7.1 that any act, so also a motion, is `Correctly made` if and only if it is `In order` (made at the right moment) and `Proper` (has the right properties).

Seconding a motion

- `Correctly moved`$(y, x, s) \wedge$ `Requires second`$(x) \wedge y' \neq y \Rightarrow$ `Open for seconding`(y', x, s')
- `Open for seconding`$(y, x, s) \wedge$ `Seconded`$(y, x, s) \Rightarrow$ `Second condition fulfilled`(y, x, s)
- `Correctly moved`$(y, x, s) \wedge \neg$ `Requires second`$(x) \Rightarrow$ `Second condition fulfilled`(y, x, s)

These rules say that a motion x that was correctly made at s by y, is open for seconding at s' if it requires a second (2, note at close, p. 56). The rules also define an intermediate predicate `Second condition fulfilled`, which predicate is conventient for reducing the number of rules needed below for regulating what happend afterwards (similar intermediate predicates will be used at various occasions.)

These rules have an implicit way of dealing with the case that the chair mistakenly thinks that a just-moved motion does not require a second: if s/he immediately states the motion, this violates the mode condition, since stating was not open for making and is not a mode condition exception.

Mover's right to speak before motion is stated

-`Second condition fulfilled`$(y, x, s) \wedge$ `Debatable`$(x) \Rightarrow$ `Floor open to mover`(y, x, s')
-`Floor open to mover`$(y, x, s) \Rightarrow$ `Open for obtaining floor by mover`$(y, x, s) \wedge$ `Open for yielding floor by mover`(y, x, s)

These rules say that when a motion has fulfilled its second condition, then (except when the motion is to be entered on the record) the mover has the right to claim or yield the floor, before the motion is stated by the chair (2, note at close, and 19, p. 56). `Floor open to mover` is a convenient intermediate predicate.

-`Second condition fulfilled`$(y, x, s) \wedge$ `To be entered on the record when made`$(x) \Rightarrow$ `Obliged to enter on the record`$(chair, x, s')$
-`Called up`$(y, x, s) \wedge$ `Debatable`$(x) \Rightarrow$ `Floor open to mover`(y, x, s')
-`Called up`$(y, x, s) \wedge \neg$ `Debatable`$(x) \Rightarrow$ `Obliged to state`$(chair, x, s')$

These three rules regulate the special case for motions that are to be entered on the record when made (i.e. motions to reconsider, see 27 RRO). The first rule says that it must be entered on the record by the chair when seconded, and the second and third rule say that if a motion that is on the record (as 'to be called up') is called up, the situation is as for all other motions after their seconding condition is fulfilled.

An *interpretation problem* here is that in cases where a motion must be entered on the record (presently only a reconsideration, 27), RRO is not clear on the precise moment at which the mover can claim the floor: is it before it is entered on the record, or after it has been called up? The present formalization chooses for the latter option. This is since it seems to make more sense to give the mover this opportunity just before debate starts than just before entering the motion on the record.

- `Open for yielding floor by mover`$(y, x, s) \wedge$ `Yielded floor`(y, s)
$\Rightarrow$ `Mover condition fulfilled`(x, s)
-`Open for obtaining floor by mover`$(y, x, s) \wedge$ `Acted to obtain floor`(y, s)
$\wedge$ `Yielded floor`$(y, s^+ \wedge$ `Has floor between`$(x, s, s^+)) \wedge \neg$ `Has changed`(y, x, s^+)
$\wedge \neg$ `Withdrawn`$(y, x, s^+) \Rightarrow$ `Mover condition fulfilled`(y, x, s^+)

-`Second condition fulfilled`$(y, x, s) \wedge \neg$ `Debatable`$(x) \Rightarrow$ `Mover condition fulfilled`(x, s)

These rules define an intermediate concept covering all situations concerning the right of the mover to obtain the floor before his motion is stated. This concept includes the situations where the mover has either abstained from or used the right to obtain the floor, or does not have this right. The predicate `Has floor between` is used to ensure that yielding the floor pertains to the motion x and not to some later motion, much later on in a meeting.

The special acts `Open for yielding/obtaining floor by mover` are used in order to make sure that when such an act is made after being surpressed or prevented, it follows that the mover condition is fulfilled (otherwise special surpression and prevention rules would have to be given for the predicate `Floor open to mover`, in addition to the general rules on surpression and prevention given below).

The following rule deals with the case that the mover has changed the motion after it was seconded. In that case the second may be withdrawn, and the motion is again open for seconding.

-`Open for obtaining floor by mover`$(y,x,s) \wedge$ `Acted to obtain floor`(y,s) $\wedge$ `Yielded floor`$(y,s^+) \wedge$ `Has floor between`$(x,s,s^+) \wedge$ `Has changed`(y,x,s^+) $\wedge$ `Seconding`$(z,x) \wedge$ `Made`$(y',z,s^-) \Rightarrow$ `Open for withdrawal`$(y',z,s^{+'})$

-`Motion`$(x) \wedge$ `Seconding`$(z,x) \wedge$ `Open for withdrawal`$(y,z,s) \wedge$ `Withdrawn`(z,s) $\Rightarrow$ `Open for seconding`(x,s')

Note that these two rules leave room for a ‘procedural loop’, where a mover keeps modifying a motion after it has been seconded. It is a matter of interpretation whether RRO also leaves room for this loop.

Stating a motion

-`Mover condition fulfilled`$(x,s) \wedge$ `Decision mode`$(x,vote) \Rightarrow$ `Obliged to state`$(chair,x,s')$

This rule says that once the mover has been dealt with, a motion that requires a vote becomes open for stating (3,54).

Starting to consider a motion

Considering a motion comes in three forms: decision by the chair, vote, and debate followed by vote. Let us first consider decisions by the chair (14,15,61e). Here a complication is that the chair may at once submit the question to the assembly, by stating it (14). In fact, this means that the chair has an obligation to a disjunctive action: decide or state. Since the present formalization does not use deontic logic, an ad hoc solution is necessary, by defining a disjunctive act `Decision or stating`.

-`Decision`$(x,z) \vee$ `Stating`$(x,z) \Rightarrow$ `Decision or stating`(x,z)

-`Mover condition fulfilled`$(x,s) \wedge$ `Decision mode`$(x,chair) \Rightarrow$ `Obliged to decide or state`$(chair,x,s')$

This rule says that when the chair must decide a motion, s/he has the choice to decide it (after which appeal is open: see motion hierarchy), or to submit it at once to the assembly by stating it (14).

`-Open for decision or stating`$(chair, x, s) \Rightarrow x =$ `top(question stack`(s')`)`
`-Open for decision or stating`$(chair, x, s) \wedge$ `Stated`$(chair, x, s) \Rightarrow$ `Open to vote`(x, s')
`-Open for stating`$(chair, x, s) \wedge$ `Stated`$(chair, x, s) \wedge$ `Decision mode`$(x, vote)$ $\wedge$ `Debatable`$(x) \Rightarrow$ `Open to debate`$(x, s') \wedge x =$ `top(question stack`(s')`)`
`-Open for stating`$(chair, x, s) \wedge$ `Stated`$(chair, x, s) \wedge$ `Decision mode`$(x, vote)$ $\wedge \neg$ `Debatable`$(x) \Rightarrow$ `Open to vote`$(x, s') \wedge x =$ `top(question stack`(s')`)`

After being stated, a motion is open for debate followed by vote (if debatable, 5,38,54), or for immediate vote (if not debatable, 38,54). The above rules on starting to consider a motion rule trigger a change of the record, concerning the values of `Open to debate` and `Open to vote`. Note that when a motion m is the top of the question stack, later events can push m down the question stack or even remove m from it, either temporarily (for instance, by laying it on the table) or definetely (for instance, by postponing it indefinetely).

The rules for how debate is conducted have not yet been formalized.

8.2 On dispensing with a question

This section should ultimately formalize all ways in which successfully introduced business becomes finished: either it is decided, or it is dispensed with in some other way, viz. by a withdrawal, or by a successful objection to its consideration or motion to postpone it indefinetely. The appendix contains a picture (finished business) that graphically represents all ways in which business can be dispensed with. At present, only voting has been (partly) formalized.

8.2.1 Voting

We now formalize the normal case, where debate is closed by putting the motion to vote (94 RRO). When the chair thinks debate has ended, s/he asks the assembly whether it is ready to vote. If nobody rises, the chair puts (in the normal case) the affirmative, and after the affirmative votes have been cast and nobody wants to re-open the debate (to which every member has the right, 38, p. 105), the chair puts the negative. At that point debate is closed, i.e. nobody can re-open debate. After the negative votes have also been cast, the chair announces the result. At that point the motion ceases to be pending, i.e. it is removed from the question stack.

Exceptions to the normal case are when the assembly has ordered vote by ballot or roll call.

`-Open to debate`$(x, s) \wedge$ `‘Ready for the question?’`$(chair, x, s) \wedge \neg\exists y$

`Acted to obtain floor`$(y,s') \Rightarrow$ `Open to vote`(x,s'')

The last condition of this rule captures the fact that any one can re-open debate before the negative has been put (38, p. 105). How does this deal with the case were somebody acts to obtain the floor after the affirmative has been put, but then makes a motion that is not in order? This is covered by the same rules as that apply at any other time during debate!

-`Open to vote`$(x,s) \wedge \neg$ `Ballot ordered`$(x,s) \wedge \neg$ `Roll call ordered`(x,s)
$\Rightarrow$ `Obliged to put affirmative`$(chair,x,s)$
-`Affirmative put`$(chair,x,s) \Rightarrow$ `Open for affirmative voting`(y,x,s')
-`Open for affirmative voting`$(y,x,s) \wedge$ `Affirmative votes cast`(x,s,n)
$\Rightarrow ayes(x,s) = n \wedge \neg$ `Open for affirmative voting`(y,x,s')

The reason why this special rule is needed in addition to the general rule in Section 7.2.6 is that, unlike all other acts that can be open for making, here not one but many acts of the required type are needed to cease the act from being open for making (I assume that all affirmative votes are being cast at the same state).

-`Affirmative votes cast`$(x,s) \wedge \neg\exists x,y$ `Acted to obtain floor`(y,x,s')
$\Rightarrow$ `Obliged to put negative`$(chair,x,s'')$
-`Negative put`$(chair,x,s) \Rightarrow \neg$ `Open to debate`$(x,s') \wedge$ `Open for negative voting`(x,s')
-`Negative votes cast`$(x,s,n) \Rightarrow noes(x,s) = n \wedge$ `Voting completed`(x,s)

Next we define which other acts fulfil the mode condition when they are made when affirmative or negative votes are open for making.

-`Open for affirmative voting`$(y,x,s) \wedge \neg$ `Affirmative vote`$(x') \Rightarrow$
$\neg$ `Open for making`(y,x',s)

Note that these rules capture in an implicit way that when somebody re-opens the discussion, the affirmative votes must be ignored (38, p. 105). This is assured by making the number of ayes and noes recorded by the record. Any later derivation of a new number causes an update on the record.

-`Ordering of Ballot`$(x,x') \wedge$ `Accepted`$(x,s) \Leftrightarrow$ `Ballot ordered`(x',s)
-`Ordering of Roll call`$(x,x') \wedge$ `Accepted`$(x,s) \Leftrightarrow$ `Roll call ordered`(x',s)

The following rules leave the precise nature of voting by ballot or roll call implicit.

-`Voted by ballot`$(x,s) \Rightarrow$ `Voting completed`(x,s)
-`Voted by roll call`$(x,s) \Rightarrow$ `Voting completed`(x,s)
-`Voting completed`$(x,s) \wedge$ `Required majority`$(x,n) \wedge ayes(x,s)\,/\,ayes(x,s)+$
$noes(x,s) \geq n \Rightarrow$ `Accepted`(x,s)
-`Voting completed`$(x,s) \wedge$ `Required majority`$(x,n) \wedge ayes(x,s)\,/\,ayes(x,s)+$

noes$(x, s) \ngeq n \Rightarrow$ `Rejected`(x, s)

`Accepted`$(x, s) \vee$ `Rejected`$(x, s) \Leftrightarrow$ `Decided`(x, s)
`Decided`$(x, s) \Rightarrow x \notin$ `question stack`(s)

These rules use some new act types.

- `Affirmative vote`$(x) \Rightarrow$ `Vote`(x)
- `Negative vote`$(x) \Rightarrow$ `Vote`(x)
- `Vote`$(x) \Rightarrow$ `Act`(x)

This formalization of the voting procedure is not yet complete. The following items must still be formalized.

- The chair is obliged to announce the result (p. 104).
- When the result is announced, any member can ask for a 'division' (pp. 104/5).
- Before the result has finally and conclusively been announced, any member can change their vote (unless the vote was by ballot) (p. 105).
- No one can vote on a question affecting only him/herself.
- What happens with a tie vote (p. 106).
- Voting rights of the chair (pp. 106, 112–114).
- Voting on nominations (p. 106).
- Mover may close debate (?).

8.3 The pending question

I now turn to the notion of a pending question.[1] RRO is not completely clear on when a notion is pending. Clearly, a notion that is open to debate, decision or vote is pending, but what about a motion that is open for seconding or stating? Let us consider an example. Suppose debate is open on a principal motion m_1, then a member moves to adjourn (m_2), which motion has precedence over m_1, so m_2 is open for seconding. Then immediately afterwards somebody rises to a point of order (m_3); now m_3 has precedence over m_1 but yields to m_2. Suppose now that `Pending` is defined such that m_1 is still pending: then m_3 is also in order, and both m_2 and m_3 are open for seconding. This is clearly not in line with the structure of RRO, which assumes that at any time at most one question can be before the assembly.

This undesired situation can be avoided as follows. The idea is to define a question stack of motions that at any state are before the assembly (debated or decided), in the

[1]The appendix contains a picture ('unfinished business') with all modes that unfinished business can have.

process of 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 following rule says how a motion is added on top of the question stack.

- `Correctly moved`$(y, x, s) \Rightarrow x =$ `top`(`question stack`$(s'))$

Note that at s' the motion x is also open for seconding (unless it needs no second). Note also that the procedural way of updating the record makes x stay in the question stack (albeit perhaps sometimes pushed down) until it is explicitly removed from the stack. Let us now look at which occasions the latter happens. This is when a motion is decided (dealt with above in Subsection 8.2.1), when another motion is adopted which removes x from before the assembly (to be added to Section 8.2), when the motion is not seconded, or when it is withdrawn by the mover before it is stated. The latter two cases are formalized as follows.

- `Open for seconding`$(y, x, s) \wedge \neg\exists y'$ `Seconded`$(y'x, s) \wedge \neg$ `Prevented`(x, s) $\wedge \neg$ `Surpressed`$(x, s) \Rightarrow x \notin$ `question stack`(s')
- `Moved`$(y, x, s) \wedge$ `Withdrawn`$(y, x, s^+) \Rightarrow x \notin$ `question stack`$(s^{+'})$.

The idea is that whenever a new motion is added on top of the question stack, the other motions are pushed downwards. This will be captured by the way the record is maintained, not by logical reasoning.

The notion of a pending question is now simply defined as the top of the question stack.

- $x =$ `top`(`question stack`$(s)) \Leftrightarrow$ `Pending`(x, s)

Note, however, that it is not completely obvious whether this precisely captures RRO's notion of a pending question. Although it does with respect to the parts of RRO that have been formalized in this report, in the remaining parts it might also occur with a more narrow meaning, as 'open to debate, vote or decision'. I leave this for future inspection, and for the time being define `Pending` as above.

Chapter 9

Evaluating the formalization of normative concepts

Although RRO expresses norms for what is obligatory, forbidden or permitted in meetings, the above formalization does not use deontic logic. Instead, the normative character of RRO is captured by three special 'quasi-deontic' predicates, `Proper`, `In order`, `Correctly moved` and two surrogate deontic predicates, `Obliged to make` and `Obliged to decide or state`. This method is mainly used for pragmatic reasons (viz. the time constraints). In a later phase of the project this choice might very well be changed. Let us therefore try to make a first comparison between the first-order method and a formalization using deontic logic (leaving a full investigation for future research).[1]

The first-order translation of deontic notions gives up some advantages of deontic logic, such as the generality of its consistency conditions. Let us restrict ourselves to standard deontic logic (SDL), a modal logic of type KD. In SDL no two incompatible acts can be obligatory at the same time: DL validates

$$\neg (O\varphi \wedge O\neg\varphi)$$

(where O stands for 'it is obligatory that'). Moreover, if SDL is extended with a standard necessity operator $\Box$, then it also validates the following formula:

$$\Box \neg (\varphi \wedge \psi) \Rightarrow \neg (O\varphi \wedge O\psi)$$

Clearly, the `Obliged to make` operator does not satisfy the first-order counterparts of these axioms. Moreover, the following two first-order formulas are consistent.

- $\texttt{Made}(p, m, s) \Rightarrow \neg\, \texttt{In order}(p, m, s)$
- $\neg\, \texttt{Made}(p, m, s) \Rightarrow \neg\, \texttt{In order}(p, m, s)$

[1]See for more elaborate discussions on the usefulness of deontic logic for representing normative knowledge Jones & Sergot [1993] and Bench-Capon [1994].

Yet it might be said that if RRO makes it impossible to be in order, it violates a basic principle of rational legislation, a principle which is nicely captured by SDL.

The language of SDL is also more expressive than our use of first-order deontic predicates. For instance, in SDL any compound formula can occur within the scope of a deontic operator. In the present formalization SDL's additional expressiveness was not needed, with one exception: at the end of Section 8.1 a disjunctive obligation was formalized, viz. the chair's obligation to either decide or put to vote motions which have *chair* as decision mode. Clearly, the predicate `Obliged to decide or state` used in this formalization is ad hoc.

Another advantage of SDL is its most general way of making a syntactic distinction between actual and required behaviour, which makes a most general definition possible of a violation: a behaviour φ is a violation of a set of norms if that set implies $O\neg\varphi$. By contrast, in this report's formalization three kinds of violation are possible (for certain p, a and s): the derivation of $\neg$ `Correctly made`(p,a,s), the derivation of `Obliged to make`$(p,a,s) \wedge \neg$ `Made`(p,a,s) and the derivation of `Obliged to decide or state`$(p,a,s) \wedge \neg$ `Decided or stated`(p,a,s). And as more special purpose deontic predicates are introduced, more types of violations become possible as well. However, it remains to be seen whether this is more than just a theoretical advantage, since the task of actually computing consistency of a deontic formalization might be as hard as computing for this report's formalization whether it is possible not to violate it.

Let us next briefly examine some possible ways of using deontic logic for representing RRO. The simplest way is by replacing the quasi-deontic predicates by expressions of a deontic logic. It turns out that if we want to retain the intermediate conditions for being in order, such a formalization can be very similar to the above one. The top level structure becomes as follows (where P stands for 'it is permitted that'; note that SDL validates $P\varphi \equiv \neg O\neg\varphi$).

- `Act`$(y,x) \wedge$ `In order`$(x,s) \wedge$ `Proper`$(x,s) \Rightarrow$ `PMade`(y,x,s)
- `Act`$(y,x,s) \wedge \neg$ `In order`$(x,s) \Rightarrow \neg$ `PMade`(y,x,s)
- `Act`$(y,x,s) \wedge \neg$ `Proper`$(x,s) \Rightarrow \neg$ `PMade`(y,x,s)

Clearly this deontic formalization does not add much to the above one in Chapter 7: the only real change is a change of the predicate `Correctly made` into the construct `PMade`. A more elaborate use of deontic predicates is, of course, possible. We could, for instance, formalize all cases where an act is not in order or proper as prohibitions. For example:

- `Made before`$(y,x,s) \wedge \neg$ `Renewable`$(x) \Rightarrow \neg$ `PMade`(y,x,s)

- $\neg$ `In order when another has the floor`$(x,s) \wedge \neg$ `Has floor`(y,s)
$\Rightarrow$
$\neg$ `PMade`(y,x,s)

However, a complication arises, since we also need positive rules for `PMade`(y,x,s), since some rules have this formula in their antecedent, for instance, in

-Motion$(x) \wedge$ Made$(y, x, s) \wedge$ PMade$(y, x, s) \wedge$ Requires second$(x) \wedge y' \neq y$
$\Rightarrow$ PSeconded(y', x, s')

Now the problem is that at least one positive rule for PMade(y, x, s) has a very large antecedent, which conjoins the negations of the antecedents of all negative rules for PMade(y, x, s). Here the reason why above the five conditions for being in order were used becomes apparent: these conditions *structure* the relevant information concerning being in order. Note also that deontic operators do not prevent the need for the `Open for ...` predicates. This is since we must be able to say under which conditions a permission to make a certain act (e.g. a seconding) ceases or is prolonged when it is not used, and the most general and structured way to do so is by way of the rules given in Subsection 7.2.6.

In sum, it is not immediately clear what the advantages are of a more elaborate use of (standard) deontic logic. SDL's general consistency conditions might be hard to compute, its added expressiveness is only useful on a few occasions, and the lack of more special deontic notions might be a disadvantage. After all, the law also often uses more specific, quasi-deontic concepts (such as criminal offence, tort, negligence and many other ones), and maybe the law has good reason to to do so. Perhaps the notions of a permitted, obligatory and forbidden act are sometimes too general in the sense that they do not express what *kind* of permission, obligation or prohibition is involved. In the present formalization, it seems that the five order conditions are very useful for structuring the normative aspects of RRO.

Bibliography

[Alexy, 1978] Alexy, R. *Theorie der juristischen Argumentation. Die Theorie des rationalen Diskurses als eine Theorie der juristischen Begründung*. Frankfurt am Main: Suhrkamp Verlag. (in german)

[Brewka & Gordon, 1994] Brewka, G. & Gordon, T.F. 1994. How to buy a Porsche, and approach to defeasible decision making. In *Working Notes of the AAAI-94 Workshop on Computational Dialectics*, 28–38. Seattle, Washington.

[Bench-Capon, 1994] Bench-Capon, T.J.M. Deontic logic: who needs it? In J.A. Breuker (ed.), *Proceedings of the ECAI-94 Workshop on Artificial Normative Reasoning*, 69–78.

[van Eemeren & Grootendorst, 1992] Eemeren, F.H. van & Grootendorst, R. *Argumentation, Communication, and Fallacies. A Pragma-dialectical Perspective*. Hillsdale, NJ: Lawrence Erlbaum Associates.

[Gordon, 1994a] Gordon, T.F. The Pleadings Game: an exercise in computational dialectics. *Artificial Intelligence and Law* 2: 239–292.

[Gordon, 1994b] Gordon, T.F. Computational Dialectics. In: *Workshop Kooperative Juristische Informationssysteme, GMD Studien* Nr. 241, 25–36

[Gordon, 1995] Gordon, T.F. *The Pleadings Game. An Artificial Intelligence Model of Procedural Justice*. Dordrecht: Kluwer Academic Publishers.

[Gordon & Karacapilidis, 1997] Gordon, T.F., & Karacapilidis, N. The Zeno argumentation framework. In *Proceedings of the Sixth International Conference on Artificial Intelligence and Law*, 10–18. New York: ACM Press.

[Jones & Sergot, 1993] 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*. Chicester: John Wiley and Sons, 275–307.

[Karacapilidis et al., 1997] Karacapilidis, N.I., Papadias, D., Gordon, T. & Voss, H. Collaborative Environmental Planning with GeoMed. *European Journal of Operational Research, Special Issue on Environmental Planning*, Vol. 102, No. 2, 335–346.

[Loui, 1998] Loui, R.P. Process and policy: resource-bounded non-demonstrative reasoning. To appear in *Computational Intelligence* 14: 1.

[Page, 1991] Page, C.V. Principles for democratic control of bounded-rational, distributed, knowledge agents. *Proceedings of the European Simulation Conference*, ed. E. Mosekilde, pp. 359–361.

[Prakken, 1995] Prakken, H. From logic to dialectics in legal argument. *Proceedings of the Fifth International Conference on Artificial Intelligence and Law*, 165–174. New York: ACM Press.

[Prakken, 1997] Prakken, H. *Logical Tools for Modelling Legal Argument. A Study of Defeasible Reasoning in Law.* Dordrecht etc.: Kluwer Law and Philosophy Library.

[Rescher, 1977] Rescher, N. *Dialectics: a Controversy-oriented Approach to the Theory of Knowledge*. Albany, N.Y.: State University of New York Press.

[Robert, 1986] Robert, H.M. *Robert's Rules of Order. The Standard Guide to Parliamentary Procedure*. New York etc. Bantam Books.

[Shanahan, 1997] Shanahan, M.P. *Solving the Frame Problem*. Cambridge, MA: MIT Press.

[Stary, 1991] Stary, C. Modelling decision support for rational agents. *Proceedings of the European Simulation Conference*, ed. E. Mosekilde, pp. 351–356.

[Suber, 1990] Suber, P. *The Paradox of Self-amendment: a Study of Logic, Law, Omnipotence, and Change.* New York: Peter Lang.

[Toulmin, 1958] Toulmin, S.E. *The Uses of Argument*. Cambridge: Cambridge University Press.

[Vreeswijk, 1995] 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.* Presented at the *4th Int. Colloquium on Cognitive Science (ICCS-95),* Donostia, San Sebastian, Spain.

[Vreeswijk, 1996] Vreeswijk, G.A.W. Representation of formal dispute with a standing order. *Research Report MATRIX, University of Limburg, Maastricht*.

Some WEB sites devoted to Robert's Rules of Order are:

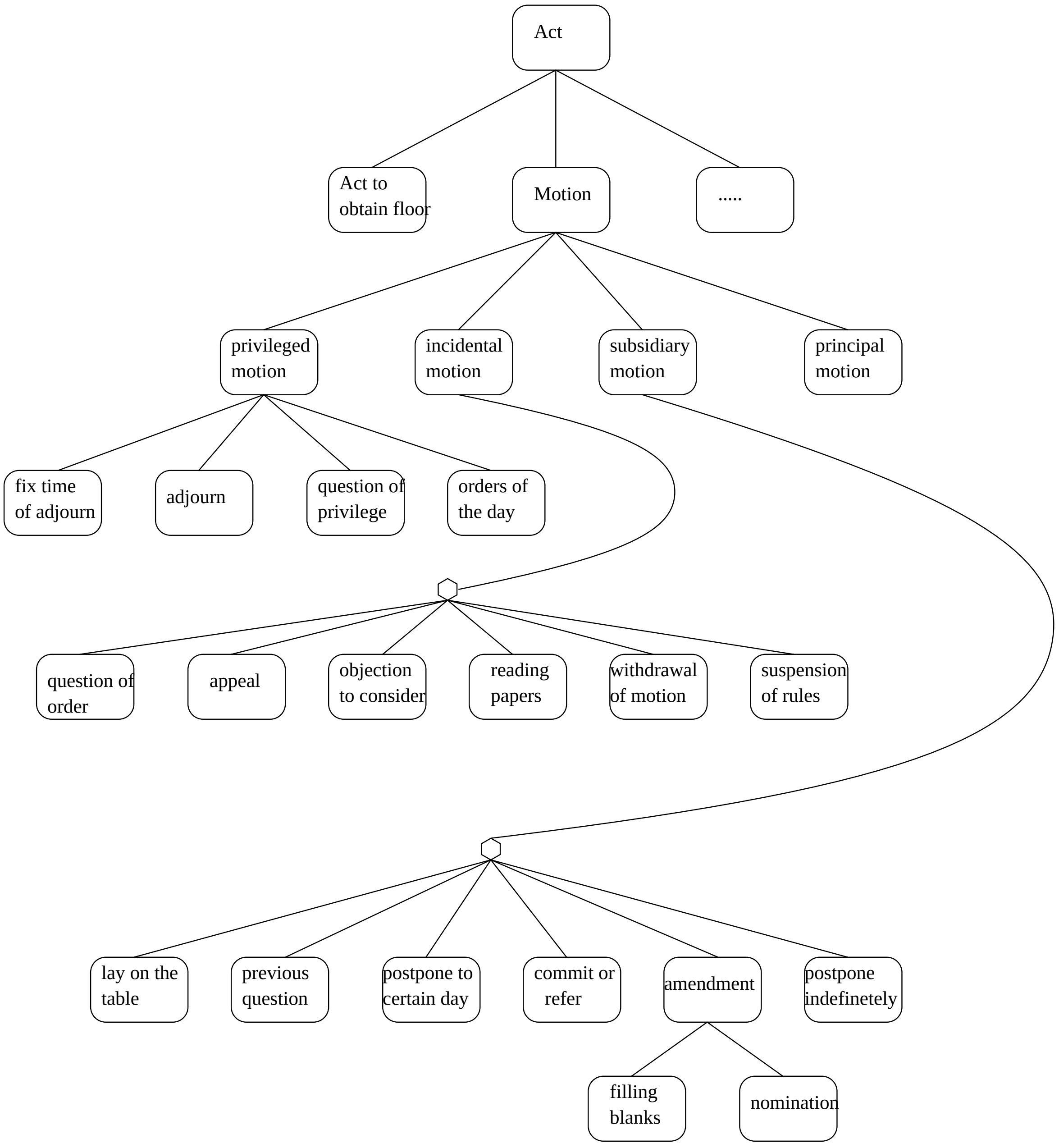
http://www.csufresno.edu/speechcomm/cagle-p3.htm
http://www.connix.com/ aip/index.htm
http://www.psychiatry.ubc.ca/dept/rulesord/preface.htm

Appendix A

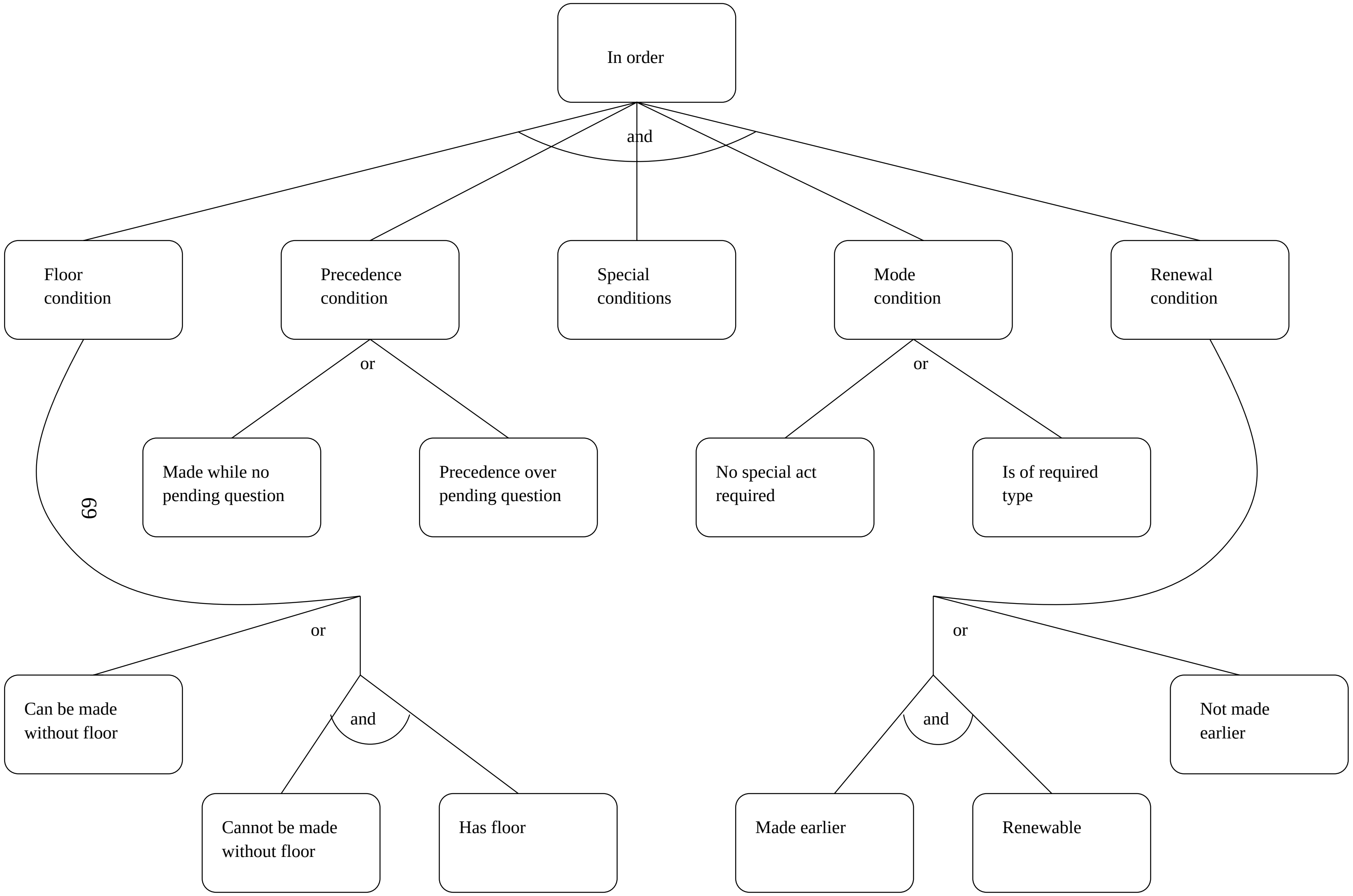
Some Pictures

This appendix contains graphical representations of some of the important notions of RRO:

- The figure *Act hierarchy* contains an (incomplete) conceptual hierarchy of types of speech acts occuring in RRO (see Chapter 6).
- The *And/Or graph for when a motion is in order* helps to understand Chapter 7.
- The figure *finished business* displays all ways in which introduced business can be dispensed with (see Section 8.2).
- Finally, the figure *unfinished business* depicts all modes that unfinished business can have (see Section 8.3).



Act hierarchy



And/Or graph for when a motion is in order

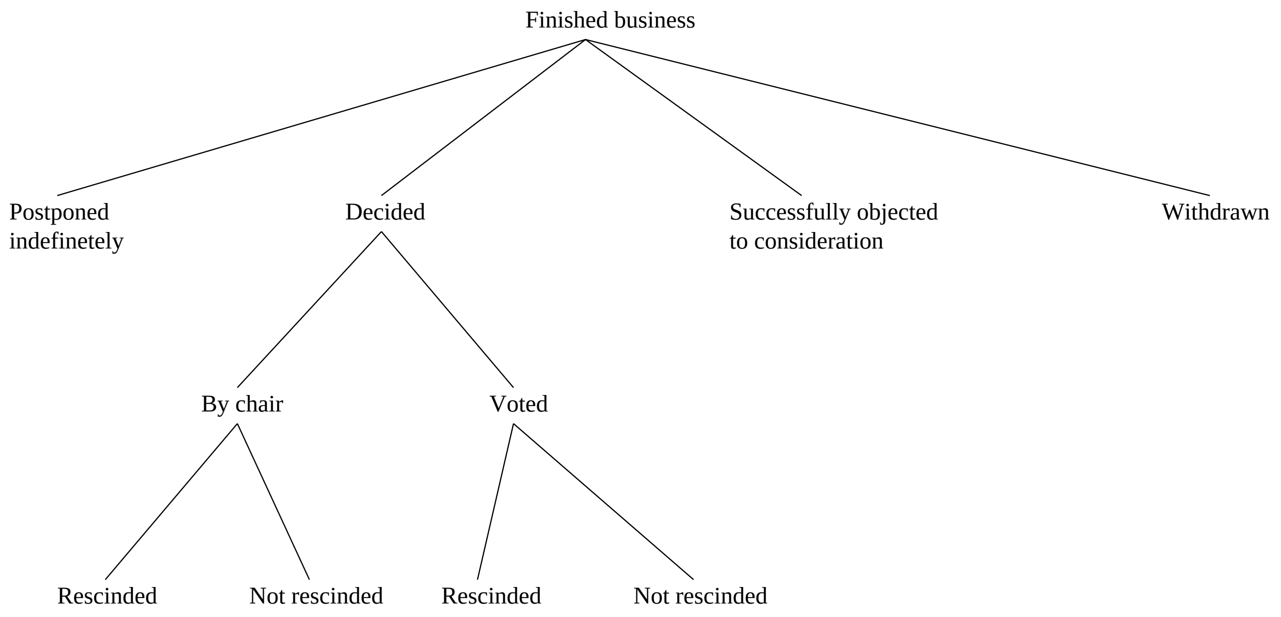


Fig: finished business

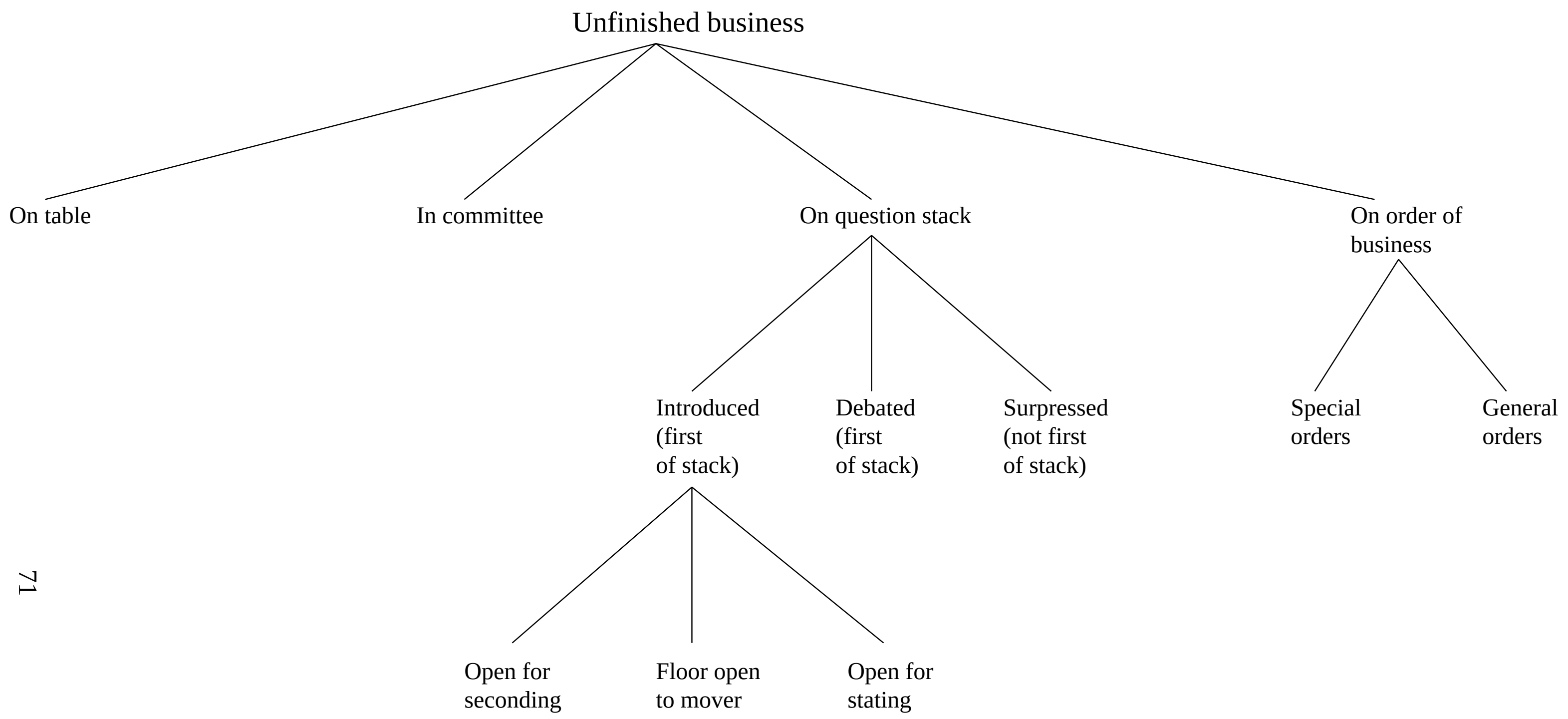


Fig: unfinished business