Para-Disagreement Logics and Their Implementation Through Embedding in Coq and SMT

Bruno Woltzenlogel Paleo

1 Introduction

On closer inspection many apparent contradictions turn out to be mere disagreements between distinct sources of information. For example, if a source s_1 says P and a source s_2 says $\neg P$, their disagreement would only become an actual contradiction if we naively merged what they say into our own knowledge base. In this case, our own knowledge base would entail $P \wedge \neg P$ and would, therefore, be inconsistent. Although we could use traditional paraconsistent logics to avoid this kind of inconsistency's worst consequences, this would be an unsatisfactory approach, because the inconsistency in this case was clearly just a result of our indiscriminate use of knowledge originating from distinct mutually contradictory sources.

This paper proposes a new logical paradigm through which disagreements can be expressed and resolved. A *possible worlds* semantics is used (cf. Sect. 3), and each source denotes a world. Logical sentences of the form $@_s P$ express that source s claims P and denote that P is true at the world denoted by s. Within these logics, we can merge conflicting information more cautiously. For instance, our knowledge base would entail $(@_{s_1} P) \wedge (@_{s_2} \neg P)$ and, as desired, no inconsistency would follow from the disagreement between s_1 and s_2 with respect to P.

Section 4 explores a few different behaviours, attitudes and procedures that people typically use to resolve disagreements and to aggregate their opinions and beliefs in order to reach common collective decisions in the social groups to which they belong. They include *consensus*, *dictatorship*, *trust* and *voting*. All these disagreement resolution methods can be formalized within the proposed logical paradigm, and it may be considered that each method leads to a different logic conforming to the proposed paradigm. As discussed in Sect. 4.4, the voting method requires

B. Woltzenlogel Paleo (✉)
Canberra, ACT, Australia
e-mail: bruno.wp@gmail.com

W. Carnielli and J. Malinowski (eds.), *Contradictions, from Consistency to Inconsistency*, Trends in Logic 47, https://doi.org/10.1007/978-3-319-98797-2_7

formulating the possible worlds semantics in a way that is, from a technical perspective, subtly different from traditional formulations (cf. Sect. 3).

One of the main goals during the development of para-disagreement logics, besides simplicity and conceptual adequacy, was ease of implementation of automated reasoning tools for these logics. To show that this goal has been achieved, Sects. 5 and 6 exhibit and discuss semantical embeddings of the logics into, respectively, the non-extensional higher-order logic of the interactive proof assistant `Coq` and a fragment of first-order logic extended with datatypes and linear integer arithmetic supported by the automatic SMT-solver `Z3`.

2 Syntax

As shown in Definition 1, the basic language of our logics is the usual language of propositional logic extended with the usual box and diamond operators of modal logics [12], and the @ operator and the (here explicit) *in* nominal operator from hybrid logics [2]. This basic syntax is extended in Sect. 4.

Definition 1 Given countably infinite sets $\mathscr{P}$ and $\mathscr{S}$ of, respectively, propositional symbols and information sources, the set of formulas *formulas* $\mathscr{L}$ is the smallest set satisfying:

- if $p \in \mathscr{P}$, then $p \in \mathscr{L}$.
- if $\varphi \in \mathscr{L}$, then $\neg\varphi \in \mathscr{L}$.
- if $\varphi_1 \in \mathscr{L}$ and $\varphi_2 \in \mathscr{L}$, then:
  - $\varphi_1 \wedge \varphi_2 \in \mathscr{L}$.
  - $\varphi_1 \vee \varphi_2 \in \mathscr{L}$.
  - $\varphi_1 \rightarrow \varphi_2 \in \mathscr{L}$.
- if $\varphi \in \mathscr{L}$, then:
  - $\Box\varphi \in \mathscr{L}$.
  - $\Diamond\varphi \in \mathscr{L}$.
- if $s \in \mathscr{S}$ and $\varphi \in \mathscr{L}$, then $@_s\varphi \in \mathscr{L}$.
- if $s \in \mathscr{S}$, then $in(s) \in \mathscr{L}$.
- if $s \in \mathscr{S}$ and $g \in \mathscr{S}$, then $(s \in g) \in \mathscr{L}$.

3 Semantics

The semantics for para-disagreement logics, as described in Definitions 2 and 3, is conceptually similar to typical possible worlds semantics for modal logics, essentially differing only in the representation of world reachability.

Definition 2 A *model* is a tuple $M := \langle W, R, I_{\mathscr{S}}, I_{\mathscr{P}} \rangle$, where W is a countable set of *worlds*, $R : W \longrightarrow List[W]$ is a function that maps each world to the list of worlds reachable from it, $I_{\mathscr{S}} : \mathscr{S} \longrightarrow W$ is an interpretation function that maps each source to the world it denotes, and $I_{\mathscr{P}} : W \times \mathscr{P} \longrightarrow \{\top, \bot\}$ is an interpretation function that maps each world and atomic proposition to a truth value.

Definition 3 The *truth* of a formula φ in a world w of a model $M := \langle W, R, I_{\mathscr{S}}, I_{\mathscr{P}} \rangle$ is denoted $M \vDash_w \varphi$ and is defined recursively as follows:

- $M \vDash_w p$ for every $p \in \mathscr{P}$ such that $I_{\mathscr{P}}(w, p) = \top$
- $M \vDash_w \neg\varphi$ iff $M \nvDash_w \varphi$.
- $M \vDash_w \varphi_1 \wedge \varphi_2$ iff $M \vDash_w \varphi_1$ and $M \vDash_w \varphi_2$
- $M \vDash_w \varphi_1 \vee \varphi_2$ iff $M \vDash_w \varphi_1$ or $M \vDash_w \varphi_2$
- $M \vDash_w \varphi_1 \rightarrow \varphi_2$ iff $M \nvDash_w \varphi_1$ or $M \vDash_w \varphi_2$
- $M \vDash_w \Box\varphi$ iff $M \vDash_{w'} \varphi$ for every $w' \in R(w)$
- $M \vDash_w \Diamond\varphi$ iff $M \vDash_{w'} \varphi$ for some $w' \in R(w)$
- $M \vDash_w @_s\varphi$ iff $M \vDash_{I_{\mathscr{S}}(s)} \varphi$
- $M \vDash_w in(s)$ iff $I_{\mathscr{S}}(s) = w$
- $M \vDash_w s \in g$ iff $I_{\mathscr{S}}(s) \in R(I_{\mathscr{S}}(g))$

Definition 4 A set of logical sentences $\varphi_1, \ldots, \varphi_n$ *entails* another logical sentence φ, denoted $\varphi_1, \ldots, \varphi_n \vDash \varphi$, iff $M \vDash_w \varphi$ for all w and for every model M such that $M \vDash_w \varphi_i$ for $1 \leq i \leq n$ and for all w.

The main technical difference between the semantics described above and the usual possible worlds semantics is that reachability of worlds is represented not as a binary relation between worlds, but as a function that maps a world to a list of its reachable worlds. Consequently, in the models used in para-disagreement logics, the number of worlds reachable from any world is always finite (because lists are finite), whereas this number could be countably infinite when reachability is represented as a binary relation between worlds. Having only a finite number of reachable worlds is important for the disagreement resolution method described in Sect. 4.4.

Definition 5 A source s is a *group* in a model $M := \langle W, R, I_{\mathscr{S}}, I_{\mathscr{P}} \rangle$ iff $R(I_{\mathscr{S}}(w))$ is non-empty (i.e. the world denoted by s has reachable worlds). Any source s'

In para-disagreement logics, the formula $@_s\varphi$ can be read as "*s claims φ*" or "*s is of the opinion that φ*". The formula $s \in g$ can be read as "s is a *participant* in the *group* g". Para-disagreement logics are concerned with establishing relationships between the claims of a group and the claims of its participants (or even non-participants).

The notion of group does not need to be understood in a narrow sense. From a broader and abstract perspective, for example, a person whose opinions are strongly influenced by three newspapers could be modelled as a "group" where those media sources are participants; and a software system that combines data from several different databases could be modelled as a "group" that has those databases as participants.

4 Methods for Information Aggregation and Disagreement Resolution

The main concern from this point on is to investigate possible relationships between the opinions of a group and the opinions of its members. In general, the group needs to aggregate the opinions of its members and resolve disagreements. There are potentially many methods to do that, and para-disagreement logics do not aim to advocate in favour of one method, but rather to be flexible and expressive enough to allow a wide variety of methods to be formalized and possibly combined to reason simultaneously about groups with distinct disagreement resolution behaviours. The next subsections discuss a few.

4.1 *Consensus*

One of the simplest methods for a group to aggregate information from its members is *consensus*: if all members of a group g claim something, then g claims it as well. In logical form, the consensus axiom schema for a fixed[1] group g is:

$$@_g(\Box\varphi \to \varphi)$$

More strongly, a para-disagreement logic assuming the more general and well-known T axiom schema ($\Box\varphi \to \varphi$) would be stating that all groups respect consensus.

The obvious limitation of consensus is that it says nothing about the opinions of a group on matters on which its members disagree.

4.2 *Dictatorship*

Dictatorship is perhaps the simplest method to aggregate information in cases when there is disagreement. The opinion of the group is simply dictated by a distinguished source. The dictatorship axiom schema for a fixed group g with a fixed dictator d is[2]:

$$@_g(@_d\varphi \to \varphi)$$

Furthermore, to show, for instance, that the dictator d has dissidents in his dictatorship g, it suffices to find a proposition q for which $@_g(@_dq \wedge \Diamond\neg q)$. This

[1]By "fixed" it is meant that g is not a schema variable, but a constant. The consensus axiom schema does *not* hold for all g. It is the responsibility of a user of the logic to instantiate and assert the axiom schema for any particular g^* that is a consensus group.

[2]Note that d does not need to be a participant of g.

illustrates that it is easy to formalize dictatorships and reason about their situations in para-disagreement logics.

A serious problem of dictatorships is that they blindly follow the opinions of the dictator and simply ignore the opinions of other members of the group, who might have greater expertise on certain topics than the dictator.

4.3 Expertise-Restricted Trust

In human society, groups often follow restricted forms of dictatorship, in which the opinions and decisions of the group with respect to a proposition pertaining to a certain topic is dictated by a member of the group with declared expertise in that topic. For example, we are normally willing to trust the opinions of doctors on matters related to our health and the opinions of lawyers on legal matters, but not conversely; and a company's technological decisions are ultimately taken by the CTO, whereas its financial decisions by the CFO.

In order to be able to formalize groups with information aggregation based on expertise-restricted trust, it is necessary to extend the basic syntax of para-disagreement logic. One possibility is to extend it with the notion of topic and special operators to talk about expertise on topics and pertinence to topics. For example, the fact that an information source s has expertise on topic t could be expressed by the formula $E(s, t)$ and the fact that a proposition p is about a topic t could be expressed by the formula $A(p, t)$.[3] Assuming, for the sake of simplicity, that experts on a given topic do not disagree on propositions related to that topic, the axiom schema stating that g trusts its experts can be written as:

$$@_g((E(\varepsilon, \tau) \wedge A(\varphi, \tau)) \to (@_\varepsilon\varphi \to \varphi))$$

Despite the additional operators, the logic is still quantifier-free, and hence essentially propositional.

4.4 Voting

Another common method to manage disagreements, especially in democratic societies, is to vote. There exist various vote-counting methods [23] and, in principle, para-disagreement logics could be extended with any vote-counting method. But *majority voting* is one of the simplest, and it is already sufficiently interesting to illustrate a vote-based para-disagreement logic.

[3] A is assumed to be an intensional operator: the truth values of $A(p, t)$ and $A(q, t)$ may differ from each other even when $p \leftrightarrow q$.

The syntax of the logic needs to be extended with an intensional *vote* operator, a *majority* modality and an *aggregation* connective, as shown below.

Definition 6 Given countably infinite sets $\mathscr{P}$ and $\mathscr{S}$ of, respectively, propositional symbols and information sources, the set of formulas *formulas* $\mathscr{L}_V$ is the smallest set satisfying the clauses in Definition 1 (with $\mathscr{L}$ replaced by $\mathscr{L}_V$) and also:

- if $\varphi \in \mathscr{L}_V$, then $\mathrm{V}(\varphi) \in \mathscr{L}_V$.
- if $\varphi \in \mathscr{L}_V$, then $\mathfrak{M}\varphi \in \mathscr{L}_V$.
- if $g, s_1, \ldots, s_n \in \mathscr{S}$, then $g \prec [s_1, \ldots, s_n] \in \mathscr{L}_V$.

The intended readings of these new kinds of formulas are:

- $\mathrm{V}(\varphi)$: φ is a proposition at issue in the vote.
- $\mathfrak{M}\varphi$: the majority chooses φ (over $\neg\varphi$).
- $g \prec [s_1, \ldots, s_n]$: the group g aggregates the sources s_1, …, s_n.

The semantics also has to be extended to cope with new syntactical constructions. An interpretation for the vote operator is needed, as shown in Definition 7, and the notion of truth needs to be extended to formulas containing the vote operator, the majority modality and the aggregation connective, as stated in Definition 8.

Definition 7 A *model* is a tuple $M := \langle W, R, I_{\mathscr{S}}, I_{\mathscr{P}}, I_V \rangle$, where W, R, $I_{\mathscr{S}}$ and $I_{\mathscr{P}}$ are as in Definition 2 and $I_V : W \times \mathscr{L}_V \longrightarrow \{\top, \bot\}$ is an interpretation function that maps each world and formula to a truth value.

Definition 8 The *truth* of a formula φ in a world w of a model $M := \langle W, R, I_{\mathscr{S}}, I_{\mathscr{P}} \rangle$ is denoted $M \vDash_w \varphi$ and is defined recursively using the cases shown in Definition 3 as well as the following:

- $M \vDash_w \mathrm{V}(\varphi)$ iff $I_V(w, \varphi) = \top$
- $M \vDash_w \mathfrak{M}\varphi$ iff there are more $w' \in R(w)$ such that $M \vDash_{w'} \varphi$ than $w'' \in R(w)$ such that $M \vDash_{w''} \neg\varphi$
- $M \vDash_w g \prec [s_1, \ldots, s_n]$ iff $R(I_{\mathscr{S}}(g)) = [I_{\mathscr{S}}(s_1), \ldots, I_{\mathscr{S}}(s_n)]$

It is now clearer why a technically different representation of reachability of worlds (using a function mapping worlds to (finite) lists of worlds, instead of a binary relation on worlds) was chosen. The definition of truth for a formula starting with a majority modality requires comparing the number of worlds w' for which φ is true with the number of worlds w'' for which $\neg\varphi$ is true. By requiring every world to have finitely many reachable worlds, the unclear corner case of comparing infinite numbers is simply avoided. The reason for choosing lists, in particular, is discussed in Sect. 5.

The majority voting axiom schema, stating that an arbitrary proposition is claimed by a fixed group g if the majority of its members asserts it and it is voted, can then be expressed as:

$$@_g((\mathfrak{M}\varphi \wedge \mathrm{V}(\varphi)) \rightarrow \varphi)$$

It is important to note that the vote operator is *intensional* with respect to its argument, in the following sense: the truth value of $\mathrm{V}(p)$ depends on p's intension (i.e. the proposition itself) and not only on p's extension (i.e. its denotation, its truth value). Consequently, $\mathrm{V}(p)$ and $\mathrm{V}(q)$ may have different truth values, even if p and q have the same truth values. Furthermore, if a group votes on a proposition p, this does not mean that the group omnisciently votes on all other propositions that it considers equivalent to p.

At this point one may wonder why the vote operator is needed. The following example, which is loosely inspired by the real-world case of the Brazilian president's impeachment in 2016, shows that a modified majority voting axiom schema without the vote guard (i.e. $@_g(\mathfrak{M}p \rightarrow p)$) would not be very useful, because then the opinions of a group would often be contradictory.

	C ?	$C \rightarrow I$?	I ?
Senator a	C	$C \rightarrow I$	I
Senator b	$\neg C$	$C \rightarrow I$	$\neg I$
Senator c	C	$\neg(C \rightarrow I)$	$\neg I$

Example 1 Suppose that a senate g composed of three senators a, b and c (i.e. $g \prec [a, b, c]$) has to decide whether the president should be impeached (proposition I). To come to that decision, the senators must deliberate on whether the president has committed a certain crime (proposition C) and whether that crime is a sufficient reason for impeachment ($C \rightarrow I$). Suppose that the senators think according to the following table:

Let S=$\{@_a(C \wedge (C \rightarrow I) \wedge I), @_b(\neg C \wedge (C \rightarrow I) \wedge \neg I), @_c(C \wedge \neg(C \rightarrow I) \wedge \neg I), g \prec [a, b, c]\}$. Then $S \vDash @_g(\mathfrak{M}C \wedge \mathfrak{M}(C \rightarrow I) \wedge \mathfrak{M}\neg I)$. If $@_g(\mathfrak{M}p \rightarrow p)$ is admitted, then:

- $S, @_g(\mathfrak{M}p \rightarrow p) \vDash @_g\neg I$, because $S \vDash @_g(\mathfrak{M}\neg I)$.
- $S, @_g(\mathfrak{M}p \rightarrow p) \vDash @_g I$, because $S \vDash @_g(\mathfrak{M}C)$, and thus $S, @_g(\mathfrak{M}p \rightarrow p) \vDash @_g C$, and $S \vDash @_g(\mathfrak{M}(C \rightarrow I)$, and thus $S, @_g(\mathfrak{M}p \rightarrow p) \vDash @_g(C \rightarrow I)$.

And hence g's opinions are contradictory (under S and $@_g(\mathfrak{M}p \rightarrow p)$) because both a proposition and its negation must be true. To avoid this, the senate must use the guarded majority voting axiom schema and decide a priori on which propositions it is going to vote. It may choose to vote on the conclusion (I) or on the premises (C and $C \rightarrow I$), but it shouldn't vote simultaneously on both. In either case, it is possible to reason about the outcome of the senate's decision process:

- in the first case (voting directly on the conclusion), $S, @_g((\mathfrak{M}p \wedge \mathrm{V}(p)) \rightarrow p), \mathrm{V}(I) \vDash @_g\neg I$ and hence the senate will decide not to impeach the president.
- in the second case (voting on the premises, and then deciding on the impeachment by logical reasoning), $S, @_g((\mathfrak{M}p \wedge \mathrm{V}(p)) \rightarrow p), \mathrm{V}(C), \mathrm{V}(C \rightarrow I) \vDash @_g I$ and hence the senate will decide to impeach the president.

One could also question the inclusion of the aggregation connective in the language and wonder if it would not be possible to use the diamond $\Diamond$ and the explicit nominal *in* operators instead, since together they can also state that a world is reachable from another. The following example discusses this.

Example 2 Consider the same scenario from Example 1 above, but let S' be S with the statement $g \prec [a, b, c]$ replaced by $@_g(\Diamond in(a) \wedge \Diamond in(b) \wedge \Diamond in(c))$. Superficially S' and S may appear equivalent, but in fact they are not. S' admits models where the world denoted by g has other reachable worlds besides those denoted by a, b and c, whereas in S the worlds reachable from the world denoted by g are exactly only those denoted by a, b and c. Therefore, $g \prec [a, b, c]$ is actually a stronger statement than $@_g(\Diamond in(a) \wedge \Diamond in(b) \wedge \Diamond in(c))$. A consequence of this fact is that, whereas with the former statement S, $@_g((\mathfrak{M}p \wedge \mathrm{V}(p)) \rightarrow p), \mathrm{V}(I) \models @_g\neg I$, with the latter statement S', $@_g((\mathfrak{M}p \wedge \mathrm{V}(p)) \rightarrow p), \mathrm{V}(I) \not\models @_g\neg I$, because S' gives only partial information about the worlds reachable from the world denoted by g. There is a model M of S', $@_g((\mathfrak{M}p \wedge \mathrm{V}(p)) \rightarrow p), \mathrm{V}(I)$ where the world denoted by g has several other reachable worlds where I is true and then $M \models @_g I$.

Instead of extending the language with the aggregation connective, the language could have been extended with quantification over sources and equality of sources. In this case, a statement such as $g \prec [a, b, c]$ could be replaced by $@_g(\Diamond in(a) \wedge \Diamond in(b) \wedge \Diamond in(c) \wedge \forall x.(\Diamond in(x) \rightarrow x = a \vee x = b \vee x = c))$. However, not only the aggregation connective is simpler, more concise and more convenient, but it also eases the implementation of counting, as discussed in the next two sections.

5 Embedding of Para-Disagreement Logics in Coq

A tool to support formal reasoning within para-disagreement logics can be implemented through a shallow embedding of the semantics of para-disagreement logics in the Coq proof assistant [11], which is based on the calculus of inductive constructions [20] for a non-extensional type-theoretical higher-order logic. The first step is to declare a type for worlds and the reachability function:

```
Parameter W: Type. (* Type for worlds *)
Parameter r: W -> list W. (* Reachability function *)
```

Then the type of propositions of para-disagreement logics is defined as the function type of functions that take a world and return a Coq proposition (i.e Prop):

```
Definition o := W -> Prop. (* Type of modal propositions *)
```

The propositional connectives operating on the lifted modal propositions are defined as functions taking modal propositions and a world, and returning a Coq proposition. Through currying (partial application of functions to arguments), such connectives can also be seen as taking modal propositions and returning a modal proposition (i.e. a function that takes a world and returns a Coq proposition). Notations are declared to allow the new lifted connectives to be written down exactly as Coq's built-in connectives, but with an “m” prefix.

```
Definition mnot (p: o)(w: W) := ˜ (p w).
Notation "m˜  p" := (mnot p) (at level 74, right associativity).

Definition mand (p q:o)(w: W) := (p w) / (q w).
Notation "p m/ q" := (mand p q)
   (at level 79, right associativity).

Definition mor (p q:o)(w: W) := (p w)   (q w).
Notation "p m q" := (mor p q) (at level 79, right associativity).

Definition mimplies (p q:o)(w: W) := (p w) -> (q w).
Notation "p m-> q" := (mimplies p q)
   (at level 99, right associativity).

Definition mequiv (p q:o)(w: W) := (p w) <-> (q w).
Notation "p m<-> q" := (mequiv p q)
   (at level 99, right associativity).
```

The use of a reachability function mapping worlds to their reachable worlds instead of a binary reachability relation between worlds requires that the box and diamond modalities be defined in a different way, using an auxiliary function that traverses the list of reachable worlds.

```
(* auxiliary function to checks if an element is in a list. *)
Fixpoint is_in A: Type (x: A) (l: list A) := match l with
  | nil => False
  | (cons h tail) => x = h  (is_in x tail)
end.

(* Box Modal Operator *)
Definition box (p: o) :=
           fun w => forall w1, (is_in w1 (r w)) -> (p w1).

(* Diamond Modal Operator *)
Definition dia (p: o) :=
           fun w => exists w1, (is_in w1 (r w)) / (p w1).
```

The @ modality and the explicit nominal operator *in* borrowed from hybrid logics are defined as expected, and a notation is declared to allow the special symbol `@` to stand for the modality.

```
Definition At (w: W)(p: o) := fun w0: W => (p w).
Notation "'@' w p" := (At w p)
  (at level 200, w ident, right associativity) : type_scope.

Definition In (w: W) := fun w0 => w = w0.
```

For the sake of simplicity, it is assumed here that the set of source symbols $\mathscr{S}$ and the set of worlds W denoted by the sources coincide. In other words, the interpretation function $I_{\mathscr{S}}$ is assumed to be the identity function. In the `Coq` embedding, this is reflected in the usage of the type `W` not only for worlds but also for sources, as seen in the type of the @ modality.

The embedding of the basic language of para-disagreement logics in Coq is completed with the definition of the aggregation connective and its corresponding notation:

```
Definition aggregation (g: W)(l: list W): o :=
           fun w: W => (r g) = l.
Notation "g '<<' l" := (aggregation g l)
  (at level 70) : type_scope.
```

And finally, quotes are used as notation for truth of a modal proposition in all worlds:

```
Definition UniversallyTrue (p: o) := forall w, p w.
Notation "' p '" := (UniversallyTrue p).
Ltac mv := match goal with [|- (UniversallyTrue _)] => intro end.
```

With all the basic language ready, it is time to move on to the extensions described in Sect. 4.4. As the majority modality requires counting, an auxiliary `count` function is defined:

```
Parameter dec: forall (f: o)(w: W), f w + ~ (f w).

Fixpoint count (p: o) (l: list W) := match l with
  | [] => 0
  | head::tail => if (dec p head)
                  then (1 + (count p tail))
                  else (count p tail)
end.
```

The function `count` needs to traverse all the reachable worlds and count on how many of them the modal proposition is true. Lists are the simplest traversable and collection datatype, and that is why it was chosen here and also as the return type of the reachability function in Definition ??. The parameter `dec` is needed to conform with the typing requirements of the "`if` ...`then` ...`else` ..." expression.

Once the `count` function is available, defining the majority modality can be easily done as follows:

```
Definition M (p: o) :=
           fun w: W => ((count p (r w)) > (count (m~ p) (r w))).
```

Next the intensional vote operator is declared, together with an axiom stating that it is invariant with respect to negation of its argument.

```
(* vote operator *)
Parameter V: o -> o.

Axiom vote_invariant_wrt_negation:
     'mforall p, (V p) m<-> (V (m~ p)) '.
```

And then finally the majority axiom schema can be declared:

```
Axiom majority_axiom: 'mforall p, ((V p) m-> ((M p) m-> p))'.
```

Now that the para-disagreement logic is fully embedded, the impeachment example can be formalized as shown below:

```
(* The three senators *)
Parameters a b c: W.

(* The senate containing the three senators *)
Parameter g: W.
Axiom e: '(g << [a; b ; c])'.


(* The two atomic propositions *)
(* proposition that the president committed a crime *)
Parameters C: o.
(* proposition that the president should be impeached *)
Parameters I: o.

(* The senators' opinions *)
Axiom a_claims_C: '(@ a C)'.
Axiom a_claims_C_implies_I: '@ a (C m-> I)'.
(* a's third opinion is not independent *)
Lemma a_claims_I: '@ a I ' .
Proof. mv.
apply (a_claims_C_implies_I w). apply (a_claims_C w).
Qed.

Axiom b_claims_not_C: '@ b (m~ C)'.
Axiom b_claims_C_implies_I: '@ b (C m-> I)'.
Axiom b_claims_not_I: '@ b (m~ I)'.

Axiom c_claims_C: '(@ c C)'.
Axiom c_claims_not_C_implies_I: '@ c (m~ (C m-> I))'.
Axiom c_claims_not_I: '(@ c (m~ I))'.

(* The propositions that have been voted in the senate *)
Axiom C_is_voted: '@ g (V C)'.
Axiom C_implies_I_is_voted: '@ g (V (C m-> I))'.
```

From the axioms stated above, it is now possible to prove that the majority claims that the president committed a crime:

```
Lemma majority_claims_C: '@ g (M C)'.
Proof.
mv.
unfold At; unfold M.
rewrite (e w).
assert (C a); [apply (a_claims_C w) | auto].
assert ((m~ C) b); [apply (b_claims_not_C w) | auto].
assert (C c); [apply (c_claims_C w) | auto].
unfold count.
destruct (dec C a); [auto | contradiction].
destruct (dec C c); [auto | contradiction].
destruct (dec (m~ C) b); [auto | contradiction].
destruct (dec (m~ C) a); [contradiction | auto].
destruct (dec C b); [contradiction | auto].
destruct (dec (m~ C) c); [contradiction | auto].
Qed.
```

However, the proof above is tedious, requiring the user to interactively count the senators that (dis)agree with the claim. In order to automate the counting, a new tactic can be implemented using Coq's Ltac language, as shown below:

```
Ltac count db :=
  match goal with
  |- context [if dec ?q ?x then _ else _] =>
      destruct (dec q x);
      firstorder with db;
      count db
  end.
```

The count tactic receives a hint database db of axioms and tries to automatcally decide the conditions of if-then-else statements in the goal using the firstorder tactic with the given database. It is a recursive tactic that reapplies itself until it fails.

With shallow embeddings, it is also often the case that definitions need to be unfolded for a goal to be proven. To automate the unfolding, the unfold_pdl tactic defined below repeatedly tries to unfold all defined connectives, quantifiers and operators included in the modal unfold hint database occurring both in the conclusion and in the hypotheses of the goal.

```
Create HintDb modal.
Hint Unfold mimplies mequiv mnot mor mand
              dia box A E M At In UniversallyTrue count: modal.

Ltac unfold_pdl := try mv; repeat autounfold with modal;
                           repeat autounfold with modal in * |-.
```

Automation can be improved further with a tactic that combines the previously defined tactics with Coq's built-in auto and autorewrite tactics:

```
Ltac pdl_solve kb rb := unfold_pdl; autorewrite with rb;
                        try auto with kb modal; try count kb.
```

In the case of the impeachment example, the databases of facts and rewrite equalities can be created as shown below:

```
Create HintDb db.
Hint Resolve a b c g e C I: db.
Hint Resolve C_is_voted C_implies_I_is_voted: db.
Hint Resolve a_claims_C a_claims_C_implies_I a_claims_I: db.
Hint Resolve b_claims_not_C
             b_claims_C_implies_I
             b_claims_not_I: db.
Hint Resolve c_claims_C
             c_claims_not_C_implies_I
             c_claims_not_I: db.

Create HintDb rb.
Hint Rewrite e: rb.
```

The previous lemma can now be proved fully automatically:

```
Lemma majority_claims_C: '@ g (M C)'.
Proof.
pdl_solve db rb.
Qed.
```

And other lemmas can be proven fully automatically as well:

```
Lemma majority_claims_C_implies_I: '@ g (M (C m-> I))'.
Proof.
pdl_solve db rb.
Qed.

Lemma majority_claims_not_I: '@ g (M (m~ I))'.
Proof.
pdl_solve db rb.
Qed.
```

Full automation is not possible when the lemma to be proven depends on axioms and lemmas that have not been included in the hint databases. Furthermore `Coq`'s `auto` tactic ignores axioms and lemmas that have a universally quantified head, because such axioms and lemmas can match any goal and, therefore, the proof search may not terminate. The majority axiom schema has a universally quantified head, and that is why it has not been included in the hint database. Consequently, lemmas that depend on this axiom currently cannot be proven fully automatically. Nevertheless, in such cases, it often suffices to apply the majority axiom, include previously proved lemmas in the local context with `pose`, and solve the remaining goals automatically with the provided tactic.

```
Lemma g_claims_C: '(@ g C)'.
Proof.
pdl_solve db rb.
apply majority_axiom.
  pose C_is_voted; pdl_solve db rb.
  pose majority_claims_C; pdl_solve db rb.
Qed.
```

A less automatic but more efficient alternative to `pose` and `pdl_solve` is shown below for a similar lemma.

```
Lemma g_claims_C_implies_I: '@ g (C m-> I)'.
Proof.
pdl_solve db rb.
apply (majority_axiom g);
  [apply (C_implies_I_is_voted w) |
   apply (majority_claims_C_implies_I w)].
Qed.
```

And finally the president's impeachment can be shown:

```
Theorem g_claims_I: '(@ g I)'.
Proof.
pdl_solve db rb.
apply (g_claims_C_implies_I w).
exact (g_claims_C w).
Qed.
```

6 Embedding of Para-Disagreement Logics in SMT

To automate reasoning in para-disagreement logics even further, SMT-solvers (for satisfiability modulo theories) such as `Z3` [17], which are capable of dealing with lists, recursive function definitions and linear integer arithmetic are a natural choice.

The main difference to the previous embedding in `Coq` is that the standard language[4] of SMT-solvers is not a higher-order typed language but a first-order multi-sorted language. This already causes difficulty when declaring the sort/type for propositions. In constrast to the embedding in `Coq`, where propositions had a defined function type from worlds to `Coq`'s built-in `Prop`, function types/sorts are not available in the first-order multi-sorted first-order language of SMT-solvers. Therefore, propositions are assumed to be of a primitive (declared but undefined) sort of arity 0.

```
(declare-sort o 0) ;; sort for propositions
(declare-sort W 0) ;; sort for worlds
```

As in the `Coq` embedding, reachability is declared as a function from worlds to lists of worlds:

```
(declare-fun r (W) (List W))
```

As a consequence of the fact that the sort `o` of propositions is now a primitive sort and not a function type from worlds to the meta-logic's propositions, the connectives and modal operators cannot be simply defined as functions, as they were in the `Coq` embedding. Instead, they must first be declared without definition:

```
(declare-fun mnot (o) o)
(declare-fun mimp (o o) o)
(declare-fun mand (o o) o)
(declare-fun mor (o o) o)
(declare-fun box (o) o)
(declare-fun dia (o) o)
(declare-fun M (o) o)
(declare-fun vote (o) o)
```

And then their intended meanings have to be axiomatized with the help of a truth predicate:

```
(declare-fun T (o W) Bool) ;; (T p w) = "p is true at world w"

(assert (forall ((w W) (p o))
            (iff (T (mnot p) w) (not (T p w))   ) ))

(assert (forall ((w W) (p o) (q o))
            (iff (T (mimp p q) w) (=> (T p w) (T q w)) ) ))

(assert (forall ((w W) (p o) (q o))
            (iff (T (mand p q) w) (and (T p w) (T q w)) ) ))

(assert (forall ((w W) (p o) (q o))
            (iff (T (mor p q) w) (or (T p w) (T q w)) ) ))
```

The axiomatizations of box, diamond and majority make use of auxiliary recursive function definitions that traverse the list of reachable worlds and output a formula that checks, respectively, whether the given proposition is true in all, in at least one, and in the majority of reachable worlds:

```
;; Box
(define-fun-rec TInAll ((p o) (l (List W)) ) Bool
  (or (= l (as nil (List W)))
      (and (T p (head l)) (TInAll p (tail l)))    ))

(assert (forall ((w W) (p o)) (iff (T (box p) w)
                                   (TInAll p (r w))   ) ))

;; Diamond
(define-fun-rec TInOne ((p o) (l (List W)) ) Bool
  (and (not (= l (as nil (List W))))
       (or (T p (head l)) (TInOne p (tail l)))    ))

(assert (forall ((w W) (p o)) (iff (T (dia p) w)
                                   (TInOne p (r w))   ) ))

;; Majority Modality
(define-fun-rec count ((p o) (l (List W)) ) Int
  (ite (= l (as nil (List W))) 0
       (ite (T p (head l)) (+ 1 (count p (tail l)))
                           (count p (tail l))          )))

(assert (forall ((w W) (p o))(iff (T (M p) w)
                                 (> (count p (r w))
                                    (count (mnot p) (r w))))))
```

For convenience, a *validity* predicate is defined, analogously to the quotes in the `Coq` embedding.

```
(define-fun V ((p o)) Bool (forall ( (w W) ) (T p w)) )
```

And finally the majority axiom schema can be asserted:

```
(assert (forall ((p o)) (V (mimp (mand (vote p) (M p)) p) ) ))
```

Now that the para-disagreement logic has been embedded, the impeachment example can be formalized. First the senators and the senate itself are declared as worlds:

```
(declare-fun senA () W)
(declare-fun senB () W)
(declare-fun senC () W)
(declare-fun senate () W)
```

Then the fact that the senate aggregates the three senators is asserted.

```
(assert (= (r senate)
           (insert senA
           (insert senB
           (insert senC (as nil (List W))))) ))
```

And the atomic propositions are declared and the senators' opinions can be asserted:

```
(declare-fun c () o) ;; the president committed a crime
(declare-fun i () o) ;; the president should suffer impeachment

(assert (T c senA))
(assert (T (mimp c i) senA))
(assert (T i senA))

(assert (T (mnot c) senB))
(assert (T (mimp c i) senB))
(assert (T (mnot i) senB))

(assert (T c senC))
(assert (T (mnot (mimp c i)) senC))
(assert (T (mnot i) senC))
```

The facts that the senators decided to vote on whether the president committed a crime and on whether the crime should imply impeachment are asserted.

```
(assert (T (vote c) senate))
(assert (T (vote (mimp c i)) senate))
```

To ask the SMT-solver whether the previous assertions entail impeachment, the negation of the impeachment conjecture should be asserted, as shown below. This is so because SMT-solvers are refutational theorem provers, proving conjectures by contradiction.

```
(assert (not (T i senate)) )
```

Finally, the commands `check-sat` and `get-proof` should be invoked.

```
(check-sat)
(get-proof)
```

Unfortunately, `Z3` fails to prove this conjecture, probably because SMT-solvers have incomplete quantifier instantiation heuristics, which may be failing to find the correct instantiation when there are defined connectives and operators. Fortunately, `Z3` can solve the problem with just a slight help. If the following unfolding of the majority axiom schema is manually asserted, `Z3` is able to do all the remaining logical and arithmetical reasoning fully automatically, and outputs a proof that is 35842 characters long.

```
(assert (forall ((p o) (w W))
   (=> (and (T (vote p) w) (T (M p) w)) (T p w) )))
```

7 Related Work

The idea of using modal logics to handle (apparent) contradictions can be traced back at least to Jaskowski's *discussive logics* [16]. However, the para-disagreement logics proposed here use the @ modality, thereby overcoming well-known issues [19] faced by Jaskowki due to his use of the $\Diamond$ modality instead. As in Jaskowki's logics, the $\Box$ modality acts like a consensus operator. $\Box P$ expresses that everybody claims P. Together with the T axiom ($\Box P \rightarrow P$), the behavior of $\Box$ is reminiscent of the $\circ$ consistency operator of *logics of formal inconsistency* with principles of gentle explosion [13].

Preferential and *Distance-based* paraconsistent logics [3] form an interesting class of logic that handles inconsistencies by considering most preferred or least distant valuations of a theory in order to determine the logical consequences of the theory. Although the use of preferential and numerical approaches may suggest a similarity with the voting-based para-disagreement logic presented in Sect. 4.4 or with para-disagreement logics where an information source is preferred (e.g. as in Sect. 4.3), the similarities are superficial and the logics are actually very different, simply because para-disagreement logics are *not* paraconsistent logics. Para-disagreement logics are classical, monotonic, modal logics, where the principle of explosion holds. Their goal is to deal with consistent theories containing formulas such as $@_{s_1} P \wedge @_{s_2} \neg P$, which express a disagreement between s_1 and s_2. In paraconsistent logics (including preferential and distance-based), on the other hand, the concern is to avoid the principle of explosion in the presence of inconsistencies in theories with formulas such as $P \wedge \neg P$. Another difference between para-disagreement logics and preferential or distance-based paraconsistent logics is that the latter's preferential or distance-based mechanisms for avoiding explosion in the presence of inconsistencies is extra-logical and rigidly built-in as part of the semantics, whereas the former's disagreement resolution mechanisms are expressible syntactically in the logic itself through axiom schemata that can be flexibly modified and even combined to suit various domains of applications, as not all disagreements ought to be resolved in the same way.

As (apparent) contradictions can be common for AI agents and databases handling data from various sources or from different points in time, it is not surprising that many tasks, such as *belief revision* [1], *information/data integration* [18], *database repair* [10] and *consistent query answering* [14, 21], share an interest with para-disagreement logics on the topic of tackling (apparent) contradictions. One important distinguishing characteristic of the framework of para-disagreement logics is that it does not advocate for a specific way of handling apparent contradictions but rather provides an expressive language that allows disagreements to be explicitly modeled and allows a wide variety of disagreement resolution mechanisms to be asserted through axiom schemata, as non-exhaustively exemplified in Sect. 4. Due to this generality, it may be the case that some concrete approaches proposed for belief revision, database repair, information integration and consistent query answering could be defined as concrete para-disagreement logics within the framework described here. In such cases, the para-disagreement logic framework would serve as an alternative classical modal foundation to define these approaches, which are often described from a non-classical, non-monotonic and paraconsistent standpoint. Despite its generality, however, certainly not all database-related approaches are amenable to be described as para-disagreement logics. An essential requirement in the para-disagreement logic framework is the ability to distinguish and name the sources of contradictory information. In a practical database setting, this requirement is not always satisfied. For instance: a single source may already contain contradictory information; or, maybe, even though the contradictory information originates from different sources or time points, it is not known anymore from which source or time point each piece of information originated. Database-related techniques that target such situations are clearly outside the scope of the para-disagreement logic framework.

The embedding of para-disagreement logics in `Coq` follows an approach previously used in the embedding of the modal logics **K**, **KB** and **S5** in `Coq` [8, 9] and related to the embedding of the same logics in `Isabelle` [5, 7] and TPTP THF [6]. However, that approach had to be modified (as explained in the previous sections), because para-disagreement logics require a different technical encoding of reachability between worlds, and it also had to be extended with arithmetical reasoning for counting worlds. Furthermore, the work presented here also discusses automation of reasoning in para-disagreement logic within `Coq`, whereas the previous work in [9] was concerned with interactive reasoning only.

Thanks to the maturity, efficiency and popularity of SAT-solvers, theorem provers (e.g. [15, 22]) for non-classical and modal logics have been implemented recently with architectures that use SAT-solvers as black-boxes. In contrast, the work presented here uses an SMT-solver. As the logics of SMT-solvers are more expressive than the classical propositional logic of SAT-solvers, non-classical and modal logics (even complex ones requiring arithmetical reasoning such as para-disagreement logics) can be fully embedded within the logics of SMT-solvers, and these solvers can then be used directly, with no need to build a separate prover having an SMT-solver as a black-box component.

8 Conclusion

The para-disagreement logics presented here constitute a new paradigm to deal with apparent contradictions that occur when different agents or sources of information have conflicting opinions about some propositions. Four different disagreement resolution methods were discussed, with special emphasis on a majority voting method. However, it is important to note that para-disagreement logics are a general framework that, in principle, can support other (possibly more sophisticated) disagreement resolution methods as well.

The development of para-disagreement logics required a formulation of possible worlds semantics that is technically different from the usual. Their embedding into the meta-logics of `Coq` and SMT-solvers also pushed further the state-of-the-art of the embedding approach, as it required the use of arithmetics, which was not necessary in previous work on simpler modal logics. At the same time, the successful (almost full) automation of para-disagreement logical reasoning within Coq and Z3 attests the current level of maturity of these tools even for a domain of application for which they were not originally intended. And indeed, the embeddings described here expand the range of applications of classical interactive and automated theorem provers to the area of paraconsistent reasoning, broadly understood, at least when contradictions are merely apparent as a result of disagreement between clearly identifiable sources.

Although the focus here was on *propositional* para-disagreement logics, this was so just because the propositional level was sufficient to discuss the essence of para-disagreement logics. The embedding into the meta-logic of SMT-solvers could be easily extended to quantifier-free first-order logic, and the embedding into the meta-logic of Coq can be easily extended to rigid higher-order logic with constant or varying domains (i.e. with actualistic or possibilistic quantifiers).

As para-disagreement logics target *apparent* inconsistencies (e.g. disagreements such as $@_{s_1} P \wedge @_{s_2} \neg P$), they should be regarded as a complement, and not a replacement, to paraconsistent logics, which handle actual inconsistencies (e.g. $P \wedge \neg P$).

Acknowledgements The impeachment example used throughout this paper is inspired by a similar example presented by Viktor Wang in the ANU-Unicamp Paraconsistency Workshop. Der Author ist Stipendiat der Österreichischen Akademie der Wissenschaften (APART).

References

1. Alchourrón, C.E., P. Gärdenfors, and D. Makinson. 1985. On the logic of theory change: partial meet contraction and revision functions. *The Journal of Symbolic Logic* 50 (2): 510–530. https://doi.org/10.2307/2274239.
2. Areces, C., and P. Blackburn. 2010. Special issue on hybrid logics. *Journal of Applied Logic* 8(4): 303–304. https://doi.org/10.1016/j.jal.2010.10.001.
3. Arieli, O. 2008. Distance-based paraconsistent logics. *International Journal of Approximate Reasoning* 48(3): 766–783. https://doi.org/10.1016/j.ijar.2007.07.002.
4. Barrett, C., L.M. de Moura, S. Ranise, A. Stump, and C. Tinelli. 2010. The SMT-LIB initiative and the rise of SMT - (HVC 2010 award talk). In: S. Barner, I.G. Harris, D. Kroening, O. Raz (eds.) Hardware and Software: Verification and Testing - 6th International Haifa Verification Conference, HVC 2010, Haifa, Israel, October 4-7, 2010. Revised Selected Papers, *Lecture Notes in Computer Science*, vol. 6504, p. 3. Springer. https://doi.org/10.1007/978-3-642-19583-9_2.
5. Benzmüller, C., and B. Woltzenlogel Paleo. 2013. Gödel's god in isabelle/hol. Archive of Formal Proofs 2013. http://afp.sourceforge.net/entries/GoedelGod.shtml.
6. Benzmüller, C., and B. Woltzenlogel Paleo. 2014. Automating Gödel's ontological proof of god's existence with higher-order automated theorem provers. In *ECAI 2014 - 21st European Conference on Artificial Intelligence, 18-22 August 2014, Prague, Czech Republic - Including Prestigious Applications of Intelligent Systems (PAIS 2014)*, ed. by T. Schaub, G. Friedrich, B. O'Sullivan *Frontiers in Artificial Intelligence and Applications*, vol. 263, pp. 93–98. IOS Press. https://doi.org/10.3233/978-1-61499-419-0-93.
7. Benzmüller, C., B. Woltzenlogel Paleo. 2014. On logic embeddings and Gödel's god. In *Recent Trends in Algebraic Development Techniques*, *Theoretical Computer Science and General*

Issues, ed. by M. Codescu, R. Diaconescu, I. uu, vol. 9463. Springer. https://doi.org/10.1007/978-3-319-28114-8.
8. Benzmüller, C., and B. Woltzenlogel Paleo. 2015. Higher-order modal logics: Automation and applications. In *Reasoning Web. Web Logic Rules - 11th International Summer School 2015, Berlin, Germany, July 31 - August 4, 2015, Tutorial Lectures*, ed. by W. Faber, A. Paschke, *LNCS*, vol. 9203, pp. 32–74. Springer. https://doi.org/10.1007/978-3-319-21768-0_2.
9. Benzmüller, C., and B. Woltzenlogel Paleo. 2015. Interacting with modal logics in the Coq proof assistant. In *Computer Science - Theory and Applications - 10th International Computer Science Symposium in Russia, CSR 2015, Listvyanka, Russia, July 13–17, 2015, Proceedings*, pp. 398–411. https://doi.org/10.1007/978-3-319-20297-6_25.
10. Bertossi, L.E. 2011. *Database Repairing and Consistent Query Answering*. Synthesis Lectures on Data Management. Morgan & Claypool Publishers. https://doi.org/10.2200/S00379ED1V01Y201108DTM020.
11. Bertot, Y., and P. Casteran. 2004. *Interactive Theorem Proving and Program Development*. Springer.
12. Blackburn, P., M. de Rijke, and Y. Venema. 2001. *Modal Logic*. New York: Cambridge University Press.
13. Carnielli, W., M.E. Coniglio, and J. Marcos. 2007. *Logics of Formal Inconsistency*, pp. 1–93. Dordrecht: Springer Netherlands. https://doi.org/10.1007/978-1-4020-6324-4_1.
14. Chomicki, J. 2008. Consistent query answering: The first ten years. In *Scalable Uncertainty Management, Second International Conference, SUM 2008, Naples, Italy, October 1-3, 2008. Proceedings*, ed. by S. Greco, T. Lukasiewicz, *Lecture Notes in Computer Science*, vol. 5291, pp. 1–3. Springer. https://doi.org/10.1007/978-3-540-87993-0_1.
15. Claessen, K., and D. Rosén. 2015. SAT modulo intuitionistic implications. In *Logic for Programming, Artificial Intelligence, and Reasoning - 20th International Conference, LPAR-20 2015, Suva, Fiji, November 24-28, 2015, Proceedings*, ed. by M. Davis, A. Fehnker, A. McIver, A. Voronkov, *Lecture Notes in Computer Science*, vol. 9450, pp. 622–637. Springer. https://doi.org/10.1007/978-3-662-48899-7_43.
16. da Costa, N.C.A., and F.A. Doria. 1995. On jaskowski's discussive logic. *Studia Logica* 54(1): 33–60. https://doi.org/10.1007/BF01058531.
17. de Moura, L.M., and N. Bjørner. 2008. Z3: an efficient SMT solver. In *Tools and Algorithms for the Construction and Analysis of Systems, 14th International Conference, TACAS 2008, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2008, Budapest, Hungary, March 29-April 6, 2008. Proceedings*, ed. by C.R. Ramakrishnan, J. Rehof, *Lecture Notes in Computer Science*, vol. 4963, pp. 337–340. Springer. https://doi.org/10.1007/978-3-540-78800-3_24.
18. Lenzerini, M. 2002. Data integration: A theoretical perspective. In *Proceedings of the Twenty-first ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, June 3-5, Madison, Wisconsin, USA*, ed. by L. Popa, S. Abiteboul, P.G. Kolaitis, pp. 233–246. ACM. https://doi.org/10.1145/543613.543644.
19. Marcos, J. 2005. Modality and paraconsistency. *The Logica Yearbook* 2004: 213–222.
20. Paulin-Mohring, C. 2015. Introduction to the calculus of inductive constructions. In *All about Proofs, Proofs for All, Mathematical Logic and Foundations*, ed. by D. Delahaye, B. Woltzenlogel Paleo. London: College Publications.
21. Staworko, S., J. Chomicki, and J. Marcinkowski. 2012. Prioritized repairing and consistent query answering in relational databases. *Annals of Mathematics and Artificial Intelligence* 64(2–3): 209–246. https://doi.org/10.1007/s10472-012-9288-8.
22. Zohar, Y., and A. Zamansky. 2016. Gen2sat: An automated tool for deciding derivability in analytic pure sequent calculi. In *Automated Reasoning - 8th International Joint Conference, IJCAR 2016, Coimbra, Portugal, June 27 - July 2, 2016, Proceedings*, ed. by N. Olivetti, A. Tiwari, *Lecture Notes in Computer Science*, vol. 9706, pp. 487–495. Springer. https://doi.org/10.1007/978-3-319-40229-1_33.
23. Zwicker, W.S. 2016. Introduction to the theory of voting. In *Handbook of Computational Social Choice*, ed. by F. Brandt, V. Conitzer, U. Endriss, J. Lang, A.D. Procaccia, pp. 23–56. Cambridge University Press. https://doi.org/10.1017/CBO9781107446984.003.