Formalizing Arrow's theorem

Freek Wiedijk
freek@cs.ru.nl

Institute for Computing and Information Sciences
Radboud University Nijmegen
Toernooiveld 1, 6525 ED Nijmegen, The Netherlands

Abstract. We present a small project in which we encoded a proof of Arrow's theorem – probably the most famous results in the economics field of social choice theory – in the computer using the Mizar system. We both discuss the details of this specific project, as well as describe the process of *formalization* (encoding proofs in the computer) in general.

Keywords: formalization of mathematics, Mizar, social choice theory, Arrow's theorem, Gibbard-Satterthwaite theorem, proof errors.

1 Introduction

1.1 The project

Some time ago Krzysztof Apt suggested to me to formalize a proof of Arrow's theorem. He told me that this theorem is considered important by economists, and that for this reason a formalization of that theorem might generate some attention from the economics community. Also it would show that formalization can be useful beyond mathematics and computer science.

Formalizing is the activity of creating a *formalization*, a very detailed representation of a mathematical theory (both consisting of definitions and theorems) in the computer. Creating a formalization is done using a computer program called a *proof checker* or *proof assistant*. Such a proof assistant helps with the creation of the formalization – although the main part of the work is done by the human user – and checks everything for correctness. Once a proof of a theorem has been formalized, it is impossible for that theorem to be false. (Human beings are fallible, but computers are very good at checking even the smallest details. For a discussion of the reliability aspect of formalization, see Section 9 below.)

For a while I wondered whether I should follow Krzysztof's suggestion or not. He had suggested that I might formalize one of the proofs from John Geanakoplos' *Three Brief Proofs of Arrow's Impossibility Theorem* [Geanakoplos 2001]. These proofs indeed are brief, as each of them is about a single page long. Now a reasonable rule of thumb when formalizing is that it takes about one week of full time work to formalize a textbook page. For this reason my expectation was that if I would try this, it would take me one week of work, and it would be straight-forward and uneventful. It was not clear to me what I would have shown if that expectation happened to be true. (It turned out that, after I made a false start – described in Section 4 below – of about two days, I formalized the first of the three proofs in another three days. This second attempt indeed was uneventful. My expectation had been quite accurate.)

In the end I decided that, despite my doubts about the usefulness of the project, I would write the formalization anyway. My main reason for this was that I wanted to show Krzysztof what a formalization of Arrow's theorem would be like, and since it was not a lot of time that I was investing, it would not be a big loss if the project turned out to be not too interesting.

1.2 The proof assistant

When formalizing, the first step is to thoroughly understand the proof that will be formalized. Once one is using a proof assistant, the technical details of the formalization process distract quite a bit from the understanding of the proof. The time lost by taking the time to thoroughly understand the proof before starting the formalization is more than paid back by the time gained during the formalization process. (One could imagine a proof assistant so helpful that it would assist in exploring the informal proof, in which case this first step would not be necessary, but unfortunately such a proof assistant does not exist yet.) In the case of Arrow's theorem however, this first phase had not to take very long as John Geanakoplos' paper was quite clear and explicit.

Another first step is to select the proof assistant to be used for the formalization. Here is a list of important current proof assistants that were options for this:

Mizar. A proof assistant for mathematics, developed in Poland under the direction of Andrzej Trybulec at the University of Białystok. Development of Mizar started in the early seventies and continues until today. Mizar is based on set theory and has the feature that formalizations look quite similar to non-formalized mathematics. Mizar offers basic automation of proof steps, but this automation cannot be extended by the users of the system (only by the system's developers). Mizar has the largest library of formalized mathematics of the current systems. This library, called MML (Mizar Mathematical Library), currently consists of 2.1 million lines of code, proving more than 55 thousand lemmas. Mizar has been used to formalize a major part of a graduate level textbook in mathematical logic [Bancerek & Rudnicki 2003].

Coq. A proof assistant for both software correctness and mathematics, developed in France at the INRIA institute. Development of Coq was started in the eighties by Gérard Huet and Thierry Coquand, but has since then been continued by many people. Coq is based on a foundational competitor of set theory called *type theory*, which is a synthesis between logical systems of Jean-Yves Girard and Per Martin-Löf. For this reason the natural way of reasoning in Coq is *constructive*, which means that you only can prove something to exist if a computer can be programmed to calculate it. However Coq also supports the usual, *classical*, kind of mathematics. Formalized proofs in Coq do not look like normal proofs, and only can be understood by 'replaying' them on the computer. Coq has been used for the most impressive formalization thus far, the formalization of the Four Color Theorem by Georges Gonthier in 2004 [Gonthier 2006]. For this formalization an improved input language for Coq called *ssreflect* was implemented. Coq also has been used to prove a realistic C compiler correct, in the Compcert project of Xavier Leroy [Leroy 2006, Blazy et al 2006].

HOL. A proof assistant both for hardware and software correctness as well as for mathematics, developed in the UK by Mike Gordon at the University of Cambridge. HOL was developed in the eighties as a close successor to the LCF system from the early seventies. The LCF and HOL systems are not based on set theory or type theory, but on a weaker and typed variant of set theory called *higher order logic.* (Because it is typed, higher order logic shares some concepts with type theory.) This is also what gives the system its name. The HOL logic and its implementation are the simplest that one can find among the proof assistants listed here. A HOL proof, like a Coq proof, does not look like a normal proof, and also needs 'replaying' on the computer to be intelligible. The HOL proof language is rather difficult to learn. The HOL system supports strong proof automation, and the user of the system can easily extend this by implementing programs that generate parts of proofs.

HOL is one of the most well-known proof assistants, and for that reason has been implemented several times:

HOL4. This is the current version of the original HOL system. It is still in development at the University of Cambridge in the UK, and probably the most popular version of HOL. HOL4 has been used by Anthony Fox to prove the correctness of a real micro-processor (the ARM6 micro-architecture) [Fox 2003].

HOL Light. This is a lean and clean reimplementation of HOL by John Harrison, made in the late nineties as part of his PhD research at the University of Cambridge. Originally this system was being referred to as *HOL Done Right* [Harrison 1995]. Currently it is being used at Intel for verification of algorithms used in floating point processors. The HOL Light system has the largest library of formalized mathematics of the HOL variants, and provides strong proof automation. The HOL Light library contains many interesting theorems, and is much better organized than the Mizar library. HOL Light has been used by John Harrison to formalize a proof of the prime number theorem that uses a significant amount of complex analysis [Harrison 2008].

ProofPower. ProofPower is a reimplementation of HOL by Rob Arthan and Roger Jones, to have a version of HOL that supports the Z notation for set theory. Originally it was a commercial product, but nowadays it is open software. ProofPower also has been significantly used for mathematics.

Isabelle. A direct successor to the HOL system, that however is different from HOL in many ways now. Isabelle has been developed in the UK by Larry

Paulson at the University of Cambridge and in Germany by Tobias Nipkow and Makarius Wenzel at the Technische Universität München. An important difference between Isabelle and HOL is that Isabelle did not hardwire the mathematical foundations into the system, but keeps it as a parameter of the system. (However, the HOL implementation on top of Isabelle, called Isabelle/HOL, is the only variant of the system that is significantly used.) Another difference between Isabelle and HOL is that Isabelle has a readable proof language inspired by the Mizar language called Isar. This means that Isabelle can be used to make formalizations that are readable like normal mathematics. Like HOL, Isabelle has been used extensively, both for computer science applications as well as for formalization of mathematics.

PVS. A proof assistant for hardware and software correctness. PVS was developed in the nineties in the US by Natarajan Shankar and John Rushby of SRI International. PVS means 'Prototype Verification System', but the system is far more than a prototype. It actually is one of the most popular proof assistants for computer science applications today. PVS has strong proof automation, and also has an integrated *model checker*, which makes it possible to explore finite state transition systems automatically. PVS is based on a variant of higher order logic, but one that is substantially different from the foundation of HOL and Isabelle/HOL. A PVS formalization is like a Coq or HOL formalization in that it only can be understood by replaying it on a computer. (In PVS the proof parts of the formalization are not even stored in a humanly readable way.) PVS has been used less for non-computer science related mathematics than the systems mentioned above.

ACL2. A proof assistant for hardware and software correctness. It is the direct successor to *nqthm*, the 'Boyer-Moore prover', that was developed in the seventies in the US by Bob Boyer and J Moore. ACL2 is quite close to nqthm but was developed from scratch by J Moore and Matt Kaufmann at the University of Texas. Both nqthm and ACL2 have been used for mathematics, but the foundation of the system, *primitive recursive arithmetic*, is too weak to express advanced mathematics. ACL2 is very closely related to the Lisp programming language (ACL2 means 'A Computational Logic for Applicative Common Lisp'), and ACL2's mathematical definitions all are executable as Lisp programs. ACL2 is more of an *automated* theorem prover than an *interactive* theorem prover: one does not enter the proofs into the system oneself, but has the system discover these proofs in a guided fashion. The proofs that ACL2 discovers are printed in a very verbose but readable English format.

For the Arrow theorem formalization project I decided to use the Mizar proof assistant, which is one of the three proof assistants that I know well (Mizar, Coq and HOL). A gentle introduction to the Mizar system is my *Writing a Mizar article in nine easy steps* [Wiedijk 2007].

Mizar makes it possible to have the formalized version of a proof be quite close to the normal English version. When I started the Arrow's theorem formalization this was my reason for selecting Mizar. It would make the project interesting to see how close to the text in John Geanakoplos' paper I could get with my formalization. (However, in the end I did not pursue this aspect of the project, as described at the end of Section 4 below.)

1.3 Earlier formalizations of Arrow's theorem

Some time months after I finished my formalization of Arrow's theorem, I attended the TTVSI ('Tools and Techniques for Verification of System Infrastructure') workshop in London, which was held on March 25 and 26, 2008. It was a celebration of the sixtieth birthday of the creator of the HOL system, Mike Gordon. At this workshop there was a talk by one of the creators of the Isabelle system, Tobias Nipkow from the Technische Universität München in Germany. In this talk he presented a formalization of Arrow's theorem under the title *A Bit of Social Choice Theory in HOL: Arrow and Gibbard-Satterthwaite.* An abstract of this talk was included in the proceedings of this workshop [Nipkow 2008].

In Tobias' talk I learned that Arrow's theorem had already been formalized by him long before I had done it. (However, I had submitted it to the Mizar library MML and its accompanying journal *Formalized Mathematics* in August 2007. My formalization still might be the first *published* formalization of Arrow's theorem. The details of the submission process to the MML of my Arrow formalization are described in Section 13 below.) Tobias had formalized Arrow's theorem in Isabelle/HOL in 2002, and afterwards also had formalized the *Gibbard-Satterthwaite theorem.*

During the talk Tobias mentioned that both Arrow's theorem and the Gibbard-Satterthwaite theorem also had been formalized (and in Isabelle/HOL as well) by Peter Gammie who was at that time at the University of New South Wales and National ICT Australia. Peter finished his formalization of Arrow's theorem in December 2006 and Gibbard-Satterthwaite in April 2007.

Here I will not compare my formalization with the Isabelle formalizations of Tobias and Peter. That would turn this paper into a comparison between the Mizar and Isabelle systems, which is not its intended topic. Instead I will focus on the process of writing my Mizar formalization of Arrow's theorem.

Tobias stated in the questions session after his talk at the TTVSI workshop that his formalization had not been particularly difficult, which suggests that his experiences with the formalization of Arrow's theorem had been similar to mine.

1.4 Outline

The rest of the paper describes my formalization of Arrow's theorem in Mizar. Section 2 gives a brief introduction to Arrow's theorem. Section 3 presents the proof that I formalized. Section 4 describes my first, aborted, attempt at a formalization, while Sections 5 to 7 present the formalization that was finished. Section 5 explains how I formalized orders and preorders. Section 6 presents a fragment of the formalization in detail, and Section 7 gives an example of how the formalization can be used to get very precise information from the proof.

Section 8 discusses whether flaws in the original proof were found by formalizing it, while Sections 9 and 10 discuss the question whether we can now guarantee that the proof will not have any further mistakes. Section 11 discusses possible variations on the statement of the theorem in the formalization (of which one also was formalized). Sections 12 and 13 describe how the formalization was submitted to the library of the Mizar system. Finally, in Section 14 we conclude with some possibilities to extend the work that is presented here.

Below we will present fragments of the Mizar formalization of Arrow's theorem. For this we will use the version of the formalization that is in the library of version '`7.8.10_4.99.1005`' of the Mizar system. This formalization is also on the web on its own at the URLs:

```
http://www.cs.ru.nl/~freek/mizar/arrow-7.8.10_4.99.1005.miz
http://www.cs.ru.nl/~freek/mizar/arrow-7.8.10_4.99.1005.abs
```

(The first file is the full formalization. The second file is the 'abstract' of this formalization, in which all proofs have been automatically removed.)

2 Arrow's theorem

Arrow's theorem says that it is impossible to have a *fair* rule for combining the preferences of a group of individuals. The theorem says that the only rule that satisfies some combination of reasonable restrictions (given below) is to take one of the individuals to be a *dictator*, and to just always follow that individual's preferences. That clearly is not a fair rule.

Arrow's theorem is also known as the *impossibility theorem*, the *dictatorship theorem*, or *Arrow's paradox*. The theorem was first proved by the economist Kenneth Arrow (a 1972 Nobel prize winner in economics) as part of his 1950 PhD work [Arrow 1950]. However, one might claim that the first *fully* correct proof was given by Richard Routley in 1979 [Routley 1979].

There are various versions of Arrow's theorem. Here is a basic version:

Let there be a finite number N of individuals that have to *rank* a finite set of alternatives A. We want a *rule* for combining the preferences of each individual into a preference for the whole group. Here a preference is a linear order on the set of alternatives, from least to most acceptable. The rule for combining these preferences now should be seen as a *constitution* for the democratic process of combining the individual preferences.

Here are the two requirements that we would like this rule to satisfy:

- If for two alternatives $a, b \in A$ all the individuals prefer b to a, then the outcome for the rule should be that the group as a whole also prefers b to a. This is the requirement of the rule *respecting unanimity.*
- For two alternatives $a, b \in A$, it should not be possible for an individual to change the group decision on the order of a and b by manipulating his or her preference for a third alternative c. This is the requirement of *independence of irrelevant alternatives.*

Arrow's theorem says that if the size of A is at least three, then the only possibility for a rule satisfying these two requirements is to be:

– There is an individual n, called the *dictator*, such that the rule is to take the group preference *always* to be identical to the preference of this individual. This is the property of the rule being a *dictatorship.*

(If the size of A is equal to two, then the theorem does not apply. In that case deciding by majority works. It is easy to check that if there are only two alternatives, then that satisfies the two requirements.)

Before we look into why naive attempts at a fair rule do not work, and how one proves this theorem, let us first look at some basic variations on the theorem. First, one can allow *ties* between alternatives, either in the preferences of the individuals, in the preference for the whole group (the outcome of the rule), or both. It turns out that surprisingly this does not make much of a difference. Second, one can drop the finiteness condition on the number of individuals, on the number of alternatives, or both. Again, this does not make a difference for the theorem (although in that case the proof that we present below will not work anymore).

Here is the notation that we will use for preference between alternatives. We will write $a \leq b$ when either b is preferred to a or when a and b are considered to be equivalent (when 'ties' are allowed). We will write $a < b$ when $a \leq b$ but not $b \leq a$. We always will have that between any two alternatives a and b at least one of $a \leq b$ or $b \leq a$ holds: this means that $a < b$ coincides with the the negation of $b \leq a$. In fact, in the Mizar formalization this is the *definition* of $a < b$, as shown on page 23. You should think of a preference as a linearly ordered sequence of small clusters of alternatives, where all alternatives within a cluster are considered to be equivalent.

Now why is it that naive approaches for taking the group decision do not work?

The naive solution for choosing which one of two alternatives $a, b \in A$ to prefer for the group as a whole, is to tally how many people prefer a to b and how many prefer b to a. If N is odd, there will be a majority vote for this, and then it seems natural to follow this for the group as a whole.

The problem with this solution is that it does not lead to a *transitive* group preference. Suppose the preferences of three individuals are $a < b < c$ and $b < c < a$ and $c < a < b$ (three cyclic permutations), where we write $a < b$ to mean that the individual prefers b to a. Now in that situation two out of three people prefer b to a, so in the group decision one would like to have $a < b$. However, similarly one would have $b < c$ and $c < a$. But that does not constitute a transitive relation! Which means that this approach does not work.

Now there seems to be a way out of this problem if one is allowed to have *ties* in the group decision. That would mean that in the example (because the group apparently is not able to decide between a, b and c) one could consider the alternatives to be all equivalent in the group preference. An order in which ties are allowed (formally: which is allowed to be not antisymmetric) is called a *preorder.* However, even with ties the naive majority rule still does not work.

In that case the third person can affect the outcome of the vote by moving a up (instead of $c < a < b$ his preferences then becomes $c < b < a$), in which case the majority vote becomes $b < c < a$. Instead of b being equivalent to c then c is preferred to b, and therefore the relationship between b and c has changed by only one person changing his preference for a. Clearly this violated the requirement of independence of irrelevant alternatives.

Another approach to try to make a fair rule is to have two alternatives be considered equivalent whenever there is no unanimity about it. However, that also does not work. If the three preferences are $a < b < c$ and $b < a < c$ and $a < c < b$, then this rule would make a equivalent to b, b equivalent to c, but $a < c$. Clearly again this is not transitive.

Apparently naive attempts at a good rule that satisfies the two requirements without being a dictatorship do not work. Arrow's theorem states that no non-naive attempts will give a good voting rule either.

3 The proof

We formalized the first proof from John Geanakoplos' *Three Brief Proofs of Arrow's Impossibility Theorem* [Geanakoplos 2001]. (This was the same proof that Tobias Nipkow formalized in Isabelle/HOL.) The three proofs in that paper successively get more abstract. Generally abstract mathematics is easier to formalize than concrete mathematics. For this reason the first proof from the paper seemed the most challenging.

Formalization becomes particularly difficult when in a proof an appeal is made to 'visual intuition'. In the first proof there were little pictures of alternatives being moved around in a preference. I wondered whether this would make the formalization difficult. (In the end it turned out that it was not that difficult to do this 'visual' reasoning formally. The way that we handled this is described in Section 5.2 below.)

The statement of Arrow's theorem in the formalization was:

```
reserve A,N for finite non empty set;
reserve a,b for Element of A;
reserve i,n for Element of N;
reserve o for Element of LinPreorders A;
reserve p,p' for Element of Funcs(N,LinPreorders A);
reserve f for Function of Funcs(N,LinPreorders A),LinPreorders A;

theorem Th14:
  (for p,a,b st for i holds a <_p.i, b holds a <_f.p, b) &
  (for p,p',a,b st
      for i holds (a <_p.i, b iff a <_p'.i, b) & (b <_p.i, a iff b <_p'.i, a)
    holds a <_f.p, b iff a <_f.p', b) &
  card A >= 3 implies
  ex n st for p,a,b st a <_p.n, b holds a <_f.p, b
```

(This is a slightly modified version of lines 530–542 of the formalization. We removed the variable reservations that were not used in the statement.)

We will now explain this statement in some detail. The Mizar syntax for the universal quantifier

`for` x `st` A `holds` B

should be read as

$$\forall x\ (A \Rightarrow B)$$

(the '`st` A' part also may be omitted), while the syntax for the existential quantifier

`ex` x `st` A

should be read as

$$\exists x\ A$$

In both cases '`st`' abbreviates 'such that'.

Now one should read

```
a <_o, b
```

as

$$a <_o b$$

and take it to mean 'b is preferred to a in preference o'. Using this notation 'b is preferred to a by individual i' becomes

```
a <_p.i, b
```

and 'b is preferred to a by the group preference' becomes

```
a <_f.p, b
```

The variable p describes the function that maps individuals to their preferences, and the variable f describes the *rule* for combining these preferences together.

The statement of Arrow's theorem in the formalization has the form

$$A_1 \wedge A_2 \wedge A_3 \Rightarrow A_4$$

in which a free variable f occurs, and in which the four subformulas are:

$A_1 =$ `for p,a,b st for i holds a <_p.i, b holds a <_f.p, b`

This is the requirement of *unanimity.* If each individual i prefers a to b, then that also holds in the group preference $f(p)$.

$A_2 =$ `for p,p',a,b st` B_2 `holds a <_f.p, b iff a <_f.p', b`
$B_2 =$ `for i holds (a <_p.i, b iff a <_p'.i, b) & (b <_p.i, a iff b <_p'.i, a)`

This is the requirement of *independence of irrelevant alternatives.* If we have two situations p and p' that only differ in preferences for alternatives different from a and b (this is the condition labeled B_2), then the group preferences $f(p)$ and $f(p')$ for those two situations also match on a and b.

$A_3 =$ `card A >= 3`

This is the requirement that there are at least three alternatives.

$A_4 =$ `ex n st for p,a,b st a <_p.n, b holds a <_f.p, b`

This states that there is a *dictator* called n. There exists an n such that if n prefers b to a, then that also will hold in the group preference $f(p)$.

The proof of Arrow's theorem consists of four steps. (These are the four steps of John Geanakoplos' proof, which are reflected in four steps in the formal Mizar proof.)

- First one proves that if every individual puts an alternative at the lowest or at the highest place (so no one puts it between other possibilities) then in the group ranking it also is at an extremal place.
  Here is how this statement was rendered in the Mizar formalization (lines 550–552):

  ```
    defpred extreme[Element of LinPreorders A,Element of A] means
    (for a st a <> $2 holds $2 <_$1, a) or (for a st a <> $2 holds a <_$1, $2);
  A4: for p,b st for i holds extreme[p.i,b] holds extreme[f.p,b]
  ```

  The expression `extreme[`o`,`a`]` is defined to mean that alternative a is at the extreme point of the preference o. (In this definition the text '`$2 <_$1, a`' means 'the second argument of this macro is less than `a` in the preference which is the first argument of the macro.) The third line states the first step in the proof of Arrow's theorem. It is labeled `A4` (the labels in the proof are `A1`, `A2`, `A3`, and so on). The proof of this step in the formalization consists of lines 553–619.
  We will present the proof of this fact below, but first will continue with the statement of the second step in the proof.

- Second, one proves that for each alternative b there is a person $n(b)$ and a specific situation tailored to this person, such that $n(b)$ by just changing his preference for b can move the group preference for b from the bottom to the top.
  In Mizar the counterpart for this step is stated in lines 620–624:

  ```
  A20: for b holds ex nb,pI,pII st (for i st i <> nb holds pI.i = pII.i) &
    (for i holds extreme[pI.i,b]) & (for i holds extreme[pII.i,b]) &
    (for a st a <> b holds b <_pI.nb, a) &
    (for a st a <> b holds a <_pII.nb, b) &
    (for a st a <> b holds b <_f.pI, a) & (for a st a <> b holds a <_f.pII, b)
  ```

  This says that there are two situations `pI` and `pII`, which only differ in the preference of `nb`, such that the group preference for `b` is extremal in both cases, but because `nb` moves `b` from lowest to highest position, in the group preference it goes from lowest to highest as well. The proof of this statement is lines 625–797 of the formalization.

Clearly the formalized statement gives more specific (and even redundant) information than the informal statement. The reason for this is that in that way the formalization becomes easier. If the less specific statement had been used, work would have been needed later to get back information that already is present in the proof of the statement.

- Third one shows that $n(b)$ is a dictator over each pair of alternatives a and c that both are different from b.
  In Mizar this is lines 798–801:

  ```
  A53: for b holds ex nb,pI,pII st (for i st i <> nb holds pI.i = pII.i) &
    (for i holds extreme[pI.i,b]) &
    (for a st a <> b holds b <_f.pI, a) & (for a st a <> b holds a <_f.pII, b) &
    (for p,a,c st a <> b & c <> b & a <_p.nb, c holds a <_f.p, c)
  ```

  which is proved in lines 802–917.
  Again this gives more information than needed, which again is just for convenience, to prevent the need to 'regenerate' information that one already has. The informal statement of the third step corresponds to just the fourth line of the formal statement.
- The fourth step says that $n(c) = n(b)$ for any alternative c different from b. Hence this $n(b)$ is a dictator for all alternatives, and the proof is done.
  In the Mizar version this is not a single statement, as there the *existence* of $n(b)$ from the previous steps is asserted, but this existence is not turned into a function. For this reason the proof asserts the existence of $n(b)$ in lines 919–924, the existence of $n(c)$ in lines 939–942, and then in line 943:

  ```
    nc = nb
  ```

  which is proved in lines 944–968.

The following table summarizes how the lines of the formalization are distributed over these four steps in the proof:

statement of the theorem	lines 530–542
step 1 of the proof	lines 550–619
step 2 of the proof	lines 620–797
step 3 of the proof	lines 798–917
step 4 of the proof	lines 919–968
the full theorem	lines 530–971
the full formalization	lines 1–1121

The part of the formalization before the proof consists of the definition and properties of the linear orders and preorders that we used. This will be discussed in Section 5 below. The part of the formalization after the proof derives a variant of Arrow's theorem. This is discussed in Section 11 below.

We will now explain how the four steps of the proof are proved:

- *If every individual puts an alternative b at an extremal place then in the group ranking it always also is at an extremal place.*
  The proof goes by contradiction. Suppose that in the group ranking it would not be at an extremal place, so that in the group ranking we have $a \leq b \leq c$ for some a and c. We are going to move a and c around in everyone's preference in such a way that the relative positions of both a and c with respect to b do not change. (That still leaves us a lot of freedom as b is at the far end of each individual's preference. We can move a and c almost everywhere as long as we leave b at the far end.)
  We will move a and c in such a way that have $c < a$ for everyone (specifically: move a just above c for everyone), because then by unanimity in the group preference we also will have $c < a$. That will then give a contradiction with the fact that we had $a \leq b$ and $b \leq c$ from the relative positions of a and c with respect to b (which did not change, and which therefore also did not change in the group preference).

- *For each alternative b there is a person $n(b)$ and a specific situation tailored to this person, such that $n(b)$ by just changing his preference for b can move the group preference for b from the bottom to the top.*
  This is proved by considering the following process. Start with everyone with b at the very bottom. In this situation by unanimity b will also be on the bottom in the group preference.
  One by one have the individuals move b in their preference from the very bottom to the very top. From the previous fact we know that after this move in the group preference b then still either has to be at the very bottom or at the very top. As long as in the group preference b is at the bottom we keep doing this. When the b flips to the top, we have found our individual $n(b)$ and the situation from the statement.
  There has to occur a flip from the bottom to the top at *some* point, because after everyone has had his turn moving b to the top, by unanimity b also will be at the top in the group preference.
  We will call the situation just before $n(b)$ moved b from bottom to top p_{I}, and the situation just after this move we will call p_{II}.

- *The individual $n(b)$ from the previous step is a dictator over each a and c that is different from b.*
  Suppose that $n(b)$ has the preference $a < c$. We are going to show that then in the group preference we also have $a < c$.
  To do this, for each individual but $n(b)$ move b to the extremal position that it has in the situations p_{I} and p_{II} from the previous step. However for $n(b)$ put it in between a and c. This means that $n(b)$ now has $a < b < c$.
  This does not affect the placing of a with respect to c, so the group preference between a and c is not affected by this, and we still need to show that in this new situation $a < c$.
  Now we will show that in the group preference after the preferences for b were moved in this way, both $a < b$ and $b < c$ hold.

We get $b < c$, because the relative positions of b and c are now for everyone the same as in p_{I}, and there we had $b < c$ (as there b was less than *everything*), so we get that in our group preference we also have $b < c$.
By the same argument for p_{II} we also get $a < b$. Hence by transitivity we find $a < c$ and we are done.

- *For any b and c the dictators $n(b)$ and $n(c)$ coincide.*
Take some a different from b and c. Dictator $n(b)$ can move b from the very bottom to the very top. In particular, he can affect the order between a and b that way. But according to the previous step $n(c)$ is the dictator for the group preference between a and b. Hence only $n(c)$ can affect that order, and we find that $n(b)$ and $n(c)$ are the same individual.

4 An aborted formalization

When I first started the formalization, I decided that I would try to mimic the paper as much as possible. This is the technology of *formal proof sketches*, where one makes a skeleton of a formalization as close as possible to the informal original [Wiedijk 2004].

However, it turned out that this proof was not really suited to this approach, and I aborted this attempt. John Geanakoplos' proof contains sentences like (from the middle of page 2):

> [...] *this would continue to hold even if every individual moved c above a, because that could be arranged without disturbing any ab or cb votes.*

This is too far from the formal statements of Mizar to make a formal proof sketch that is sufficiently close to this to be attractive. (Of course the sentence can easily be expressed using the Mizar language, but the result will not at all be structurally similar to the natural language.)

For my first version of the formalization I defined the terminology that is used in the paper. For instance I introduced a type called '`Alternatives`' for finite sets that have at least three elements, and a type '`Constitution`' for functions that determine a group preference from the preferences of the individuals. The statement to be proved then became:

```
reserve A for Alternatives;
reserve N for non zero (natural number);

theorem
  for f being Constitution of A,N st
    f is independent_of_irrelevant_alternatives & f is respecting_unanimity
  holds f is a_dictatorship;
```

However, this is not what I put in my file, as in Mizar such a statement is more naturally expressed as a so-called *cluster*:

```
registration let A,N;
  cluster
    independent_of_irrelevant_alternatives respecting_unanimity ->
      a_dictatorship Constitution of A,N;
  ...
end;
```

A cluster is the way that one provides Mizar's type system with information about properties of 'attributes', argumentless type modifiers. This cluster would mean that every time the type of a Mizar term would contain the attributes `independent_of_irrelevant_alternatives` and `respecting_unanimity`, the Mizar type system would also automatically give it the attribute `a_dictatorship`. The advantage of a cluster is that the type system will infer properties of terms expressed by attributes automatically, relieving the user from that kind of reasoning.

The three definitions that I had in my file for the terminology in the cluster were:

```
definition let A,N,f;
  attr f is independent_of_irrelevant_alternatives means
    for p,p',a,b st
      for i holds (a <_p.i, b iff a <_p'.i, b) & (b <_p.i, a iff b <_p'.i, a)
    holds a <_f.p, b iff a <_f.p', b;
end;

definition let A,N,f;
  attr f is respecting_unanimity means
    for p,a,b st for i holds a <_p.i, b holds a <_f.p, b;
end;

definition let A,N,f;
  attr f is a_dictatorship means
    ex n st for p,a,b st a <_p.n, b holds a <_f.p, b;
end;
```

(The bodies of these definitions clearly are three parts of the statement that I formalized in my eventual formalization.) The definitions of the types involved in this were:

```
definition let A,N;
  mode Constitution of A,N is
    Function of Funcs(N,LinPreorders A),LinPreorders A;
end;

notation let A;
  synonym Alternative of A for Element of A;
end;

notation let N;
  synonym Individual of N for Element of N;
end;
```

Clearly I could have put this notation back on top of my final formalization. However I decided not to do this. Generally I like to only have definitions for notions that are used more than once. In this case the only use of these definitions would have been cosmetic, and it would have made the final statement a bit less transparent.

5 Moving alternatives around in preferences

5.1 Modelling orders and preorders

Arrow's theorem and its proof is about orders and preorders. I had to decide how to model this in my formalization. One consideration for this was that I did not have a single fixed order, but many orders simultaneously. In Mizar there is a large amount of theory about orders, but not one in which the order is explicitly indicated in the notation. For that reason I had to introduce my own definitions. (When I submitted the formalization to the Mizar library the referee did not like that at all. See the description of this submission process in Section 13 below.)

In my definitions of orders and preorders I could have used some notions already defined in the Mizar library, but it would not have saved me time. Relating my notions to the existing ones would just have added work.

I could have used the extreme possibility to represent the orderings as set of pairs and to just write

```
[a,b] in o
```

which is Mizar notation for

$$\langle a, b\rangle \in o$$

However, I really wanted to have the less-than symbol in my statements. For this reason I introduced definitions that allowed me to write an ASCII counterpart to my preferred notation

$$a <_o b$$

The closest that one can get to this in Mizar is to use either

```
a < b ,o
```

('$a < b$ in the order o'), or

```
a <_o, b
```

At first I used the first possibility, but after a while I changed my mind and used the second.

After I introduced my own definitions of orders and preorders, I had to develop the basic properties of these notions, like antisymmetry and transitivity and so on. That took lines 83–344 of the formalization. I put in a small token theorem to link my notions to the rest of the library by proving in lines 201–283:

```
definition
  let A;
  redefine func LinOrders A means :Def3:
    for R being set holds R in it iff R is connected Order of A;
  ...
end;
```

5.2 Moving elements around

Once I had my orders and preorders defined, I had to think about how to model moving elements around. But it turned out that I did not need that! All I needed was to prove that certain orders existed in which the relative positions on a few elements were the same as in the original order.

For instance, instead of defining an operation to move an element a to the bottom (and keeping the rest of the elements in the same order), all I needed was to prove a lemma (lines 421–448 of the formalization):

```
theorem Th10:
  a <> b & a <> c implies ex o st
    a <_o, b & a <_o, c &
    (b <_o, c iff b <_o', c) & (c <_o, b iff c <_o', b);
```

This says that if I have an order o', then I can find an order o where b and c are in the same order as in o', and a is below both of them.

Altogether I had seven lemmas like this, which I proved in lines 346–524.

6 A small sample of Mizar code

To give an impression of what the proof of Arrow's theorem looks like in Mizar code, I will now transcribe the formalization of the first step in this proof. Here is the Mizar version (with line numbers in the right margin):

```
A4: for p,b st for i holds extreme[p.i,b] holds extreme[f.p,b]             552
  proof                                                                    553
    assume not thesis;                                                     554
    then consider p,b such that                                            555
A5:   (ex a st a <> b & a <=_f.p, b) & (ex c st c <> b & b <=_f.p, c) &    556
      for i holds extreme[p.i,b];                                          557
    consider a' such that                                                  558
A6:   a' <> b & a' <=_f.p, b by A5;                                        559
    consider c' such that                                                  560
A7:   b <> c' & b <=_f.p, c' by A5;                                        561
    ex a,c st a <> b & b <> c & a <> c & a <=_f.p, b & b <=_f.p, c         562
    proof                                                                  563
      per cases;                                                           564
      suppose                                                              565
A8:       a' <> c';                                                        566
        take a',c';                                                        567
        thus thesis by A6,A7,A8;
      end;
      suppose
A9:       a' = c';
        consider d such that
A10:      d <> b & d <> a' by A3,Th2;
        per cases by Th4;
        suppose
A11:        d <=_f.p, b;
          take d,c';
          thus thesis by A7,A9,A10,A11;
        end;
        suppose
A12:        b <=_f.p, d;
          take a',d;
          thus thesis by A6,A10,A12;
        end;
      end;
    end;
    then consider a,c such that
A13:  a <> b & b <> c & a <> c & a <=_f.p, b & b <=_f.p, c;
    defpred P[Element of N,Element of LinPreorders A] means
    (a <_p.$1, b iff a <_$2, b) & (b <_p.$1, a iff b <_$2, a) &
    (b <_p.$1, c iff b <_$2, c) & (c <_p.$1, b iff c <_$2, b) & c <_$2, a;
A14: for i holds ex o st P[i,o]
    proof
      let i;
      per cases by A5;
      suppose for c st c <> b holds b <_p.i, c;
        then
A15:      b <_p.i, a & b <_p.i, c by A13;
        consider o such that
A16:      b <_o, c & c <_o, a by A13,Th7;
        take o;
        thus thesis by A15,A16,Th4,Th5;
      end;
      suppose for a st a <> b holds a <_p.i, b;
        then
A17:      a <_p.i, b & c <_p.i, b by A13;
        consider o such that
A18:      c <_o, a & a <_o, b by A13,Th7;
        take o;
        thus thesis by A17,A18,Th4,Th5;
      end;
    end;
    consider p' being Function of N,LinPreorders A such that
A19:  for i holds P[i,p'.i] from FUNCT_2:sch 3(A14);
    reconsider p' as Element of Funcs(N,LinPreorders A) by FUNCT_2:11;
    a <=_f.p', b & b <=_f.p', c by A2,A13,A19;
    then a <=_f.p', c & c <_f.p', a by A1,A19,Th5;
    hence contradiction;                                        618
  end;                                                          619
```

In English this proof can be read as follows:

> We are going to show that if for each individual i the alternative b is at the extreme point of i's preference, then it is also at the extreme point of the group preference *(line 552)*.
> Suppose by contradiction that this is not true *(line 554–557)*. Then there are a' and c' (both different from b, but maybe not different from each other), with in the group preference $a' \leq b \leq c'$ *(lines 558–561)*. We use this to find a and c that again satisfy $a \leq b \leq c$, but where we also have $a \neq c$ *(line 562)*. For this we do a case distinction *(line 564)*: either already $a' \neq c'$, in which case we can take $a = a'$ and $c = c'$ *(lines 565–569)*, or we have $a' = c'$ *(line 570–571)*. In this latter case we use the fact that there are at least three alternatives *(this is the assumption labeled* `A3`*)* to obtain an alternative d different from both b and $a' = c'$ *(lines 572–573)*. Now either $d \leq b$ in which case we can take $a = d$ and $c = c'$ *(lines 575–579)* or $b \leq d$ in which case $a = a'$ and $c = d$ works *(lines 580–584)*.
> We therefore have $a \leq b \leq c$ with $a \neq c$ *(line 587–588)*. Now for each individual i change the preference to one in which b is in the same position relative to a and c, but where $c < a$ *(lines 589–592 and lines 613–615)*. It is easy to see that such a preference always exists *(lines 592)*, by a case distinction *(lines 595)* with the kind of extreme that b was for i *(the statement that b was an extreme was on line 556 with label* `A5`*)*. Either b is the lowest choice of i *(line 596)*, in which case any preference with $b < c < a$ will work *(lines 599–602)*. Or b is the highest choice *(line 604)*, in which case we use $c < a < b$ *(lines 607–610)*.
> With the modified preferences, independence of irrelevant alternatives *(the statement labeled* `A2`*)* gives us that in the group preference b still is in the same relative order to a and c as before, i.e., we still have there that $a \leq b \leq c$. By transitivity this gives $a \leq c$, but we also now have that all individuals have preference $c < a$, so by unanimity *(the statement labeled* `A1`) that also holds in the group preference *(line 617)*. Together this clearly is a contradiction *(line 618)*, which finishes the proof *(line 619)*.

This proof is a bit more subtle that the proof in the paper of John Geanakoplos. We will discuss this in Section 8.

7 Where is the cardinality assumption used?

Mizar requires one to make explicit what is the relation between the steps in the proof through the use of labels, that are subsequently referred to using the keyword '`by`'.

Furthermore the system has a utility called `relprem` that will point out which of those labels actually are not necessary. This utility is one among many similar ones (`relinfer`, `reliters`, `trivdemo`, `chklab`, `inacc`, etc.) that are often called the *irrelevant utilities*, as they point out which parts of the formalization are irrelevant. These utilities all have been run on the Arrow's formalization. Therefore all the references to labels that occur in the formalization are *needed* to make the formalization correct.

We can use this to gain a better understanding of the proof. As a specific example, consider the assumption in the theorem that the cardinality of the set of alternatives is at least *three*. Where in the proof is this used?

In the formalization this assumption is stated as

```
A3: card A >= 3;
```

Therefore, we should be looking for justifications of the form

```
by ..., A3, ...;
```

It turns out that there are three of these:

- The first use of the assumption is on line 573 of the formalization, in the first step of the proof, which was detailed in the previous section. It is used to deal with the possibility that we get $a \leq b \leq c$ with $a = c$, in which case the movement of c with respect to a will not be possible.
- The second use of the assumption is on line 717 of the formalization, in the second step in the proof. Actually the only thing that is used there is that there are at least *two* alternatives. This is used to make the proof easier: we are looking for an individual where b moves from the bottom to the top. If the number of alternatives is equal to one, then it is at the top already, and the proof breaks the way it is formalized.
  It probably is possible to get rid of this second use of the assumption by being a bit more careful in the way that we determine $n(b)$.
- The third use of the assumption is on line 938, in the fourth step in the proof. This corresponds to the sentence 'take some a different from b and c' on page 13 above.

The possibility to do this kind of very careful proof analysis is one of the attractive sides of using formalization technology.

8 Did formalization show errors in the informal proof?

Formalization is also called *proof checking*, and what is the point of checking something if it will not find errors? Of course Arrow's theorem is very well known, and the paper that was formalized has been widely read. Still an interesting question is: did I find any errors?

The answer is that, no, I did not find real errors. (Nor did Tobias Nipkow according to his talk in London.) However, I *did* find 'omissions' of some trivial cases. The formalized proof seemed to have more case distinctions than I experienced when I read the original proof. All these case distinctions do not amount to much, but when formalizing one still has to navigate all those possibilities, thinking ‘ah yes, in *that* case of course it is trivial.’ Those cases were not (consciously) present in my mind when I studied the proof without the computer’s help.

The only case distinction where one might claim that the proof from John Geanakoplos’ paper *really* misses something is the one that already was presented in Section 6. In the paper it reads:

> *Suppose to the contrary that for such a profile and for distinct a, b, c, the social preference put $a \geq b$ and $b \geq c$. By independence of irrelevant alternatives, this would continue to hold even if every individual moved c above a,* [...]

However, it is not shown that this moving around of c with respect to a is possible, as we do not have the information that $a \neq c$! From the way that a and c are found we know that a and c both are different from b, but this is not sufficient to establish that $a \neq c$.

Of course the situation with $a = c$ is easy to deal with as well (as the detailed proof in Section 6 shows), but it might be claimed that the proof by John Geanakoplos is incomplete because it does not make this case explicit. At least, I had not seen it coming, and was surprised when Mizar forced me to think about it.

9 Is it now absolutely certain that the proof is flawless?

Human beings are fallible, and any non-trivial computer program has bugs. Everyone knows that. Therefore, how reliable is it when a proof assistant tells me that the proof that I formalized is flawless? Is it thinkable that the proof still might be wrong in some way?

We are getting into the very dangerous territory of *philosophy* here. The answer is: of course it might be wrong! *Absolute* certainty does not exist in any way. For example, the existence of India is not certain either. One might be mistaken about that as well. (Even if you happen to be in India, that does not guarantee that India exists. You might very well be mistaken about where you are.) But questioning the existence of India is not interesting. And therefore the question of *absolute* certainty is not interesting. If the certainty of the correctness of my proof is of the same magnitude as the certainty of India existing, there is no point in discussing that certainty.

There is one *significant* aspect in which a proof might have a problem despite the fact that it has been coded and verified with a proof assistant. This is the possibility that the definitions of the notions that were used in the formalization do not correspond to your understanding of them. In that case your formalization proved *something*, which means that it is not strictly speaking *false*, but it might not have proved the thing that you thought you proved, so in that sense it then is incorrect. (This has happened in the Isabelle system once. There was a definition of a prime number being a number with precisely two divisors, and several formalized lemmas about this notion. However, the 'mistake' was that the numbers in this formalization were integers, which meant that prime numbers p had *four* divisors: 1, p, -1 and $-p$. All those theorems, which were of the form 'if we have a prime number, then it has the following properties' in fact were true, as there *were* no 'prime numbers'. Still, one could not consider those theorems to be really flawless.) A proof assistant can establish that *if* the definitions are correct, then the theorem is correct, but it cannot establish that the *definitions* themselves are correct.

Bugs in the proof assistant also *might* be a reason that a proof might be incorrect despite it having been formalized, but this is not a significant risk (and it will even be smaller in the long term) for the following four reasons:

- First of all, bugs in a proof assistant are similar to bugs in a compiler. In practice, when a program is wrong it hardly ever is caused by compiler bugs. When you write a program and it does not behave the way you expect it to, you do not expect compiler bugs to be the cause. And proof assistant bugs are even less harmful than compiler bugs, as a compiler need to generate object code, while a proof assistant just needs to 'generate' a Boolean value (whether the proof is correct or not), which generally is a simpler process. And people *try* to make their formalizations correct, so it is not that the formalizer is working *against* the proof assistant.
  I do not know of a proof of a false statement ever *accidentally* having been accepted by a proof assistant because of bugs in the system. Certainly there have been bugs that allowed one to prove false statements. But in that case those 'proofs' were contrived examples, and not real proofs that people believed to be correct. Also, every time a bug in a proof assistant was found the library of the system still was okay after the bug had been fixed.
  The correctness of a proof assistant is similar to the correctness of a compiler in another way. The fact that the system has behaved correctly on many many inputs gives a lot of trust in the system.
  (There is an asymmetry here. Every time the kernel of a system is modified, it might have become incorrect. Now the system will be tested on its library, which means that if it starts to reject correct formalizations, this has a large probability of being noticed. However, generally there will not be a regular dual test to investigate whether it has started accepting incorrect formalizations.)

- Then there is a very powerful method to further minimalize the risk of a proof assistant giving the wrong answer: to use a *micro-kernel architecture.* In this approach one has only a very small part of the proof assistant (the *logical kernel*) guarantee the correctness of the mathematics. This is also called the *LCF architecture*, after the LCF system from the seventies that used this approach for the first time. Yet another name for this approach is the *de Bruijn criterion* (although in the proposal of N.G. de Bruijn from

which this name was derived, the kernel was not part of the program but a separate checker.)
The current serious system that has the smallest kernel in this style, is the HOL Light system by John Harrison. Its logical kernel is only 394 lines long (less than 20 pages of code, of which about half is comments.)

- Another way to reduce the risk of mistakenly accepting a proof is the approach of *multiple implementations*. This is not used so much in its pure form. However, there are systems to convert the mathematical library of one system to be usable in another [Naumov et al 2001, Obua & Skalberg 2006, Asperti et al 2007], and then the second system rechecks the library of the first, which is a form of having multiple implementations.
  This approach also is applicable to reduce worries about the correctness of the compiler and processors that are used to run the proof assistant. One can gain confidence in the correctness of the behavior of the proof assistant by using different compilers for the same programming language (this often is made more difficult by the fact that the system is written for a specific language for which only one compiler exists) and run it on different architectures.

- And finally one can prove the (kernel of the) system to be correct, and not just on paper but as a *formalization*. This has not *really* been done yet for a serious proof assistant, but it will happen in the foreseeable future. Currently there already have been experiments with correctness proofs of simplified versions of logical kernels, like the Coq in Coq project [Barras & Werner 1997, Barras 1999] and the HOL in HOL project [Harrison 2006].
  (Of course there is a circularity in using a system to verify its own correctness. However, the trust that one gains about the correctness in this way clearly is orders of magnitudes higher than the trust that one can have without.)
  Also proof assistant technology is sufficiently advanced that serious compilers and processors can be proved correct using a proof assistant. This again allows one to get the certainty of a proof being correct closer to that of India existing.

Another possible problem, which takes us back to philosophy, is that the logical foundation of the system might turn out to be inconsistent (i.e., that one can prove false theorems from it). Consider a proof assistant based on Cantor's naive set theory (which is inconsistent because it has the Russell paradox) or one based on Martin-Löf's first version of type theory (which is called *type-in-type*: this system is inconsistent because of Girard's paradox). If one formalizes a proof in such a system, strictly speaking one would not have established anything, as *any* statement can be proved in such a system. Still, even if one only has checked a proof in a system that is inconsistent, but where one needs to work very hard to exploit that inconsistency, it does not seem a serious possibility that one will do so *accidentally*.

10 How do we know that this really is Arrow's theorem?

According to the previous section one should not seriously worry about whether the theorem that was proved holds or not. However, one *could* worry about whether it actually was Arrow's theorem. It might be the case that we made a mistake in the *statement* of the theorem in such a way that it states something different from Arrow's theorem. It might for instance be the case that the assumptions that we put in the statement happened to be contradictory, leaving the statement vacuously true.

It is not possible to be *utterly* certain that in the definitions of the notions and the statement of the theorem no mistakes were made, but there are three ways in which we still can be *reasonably* certain about this:

- First of all, *we generally can keep the statement simple, and the chain of definitions of the notions in the statement reasonably short.* This is one reason for not wanting to introduce too much terminology (which was the approach described in Section 4 that I aborted after a short while), but to leave the content of the statement explicit.
  The definitions that occur in the Mizar statement of Arrow's theorem (that was given on page 8) are for the orders and preorders:

  ```
  reserve A for non empty set;
  reserve a,b,c for Element of A;

  definition
    let A;
    func LinPreorders A means
      for R being set holds R in it iff
      R is Relation of A & (for a,b holds [a,b] in R or [b,a] in R) &
      (for a,b,c st [a,b] in R & [b,c] in R holds [a,c] in R);
  end;

  definition
    let A;
    func LinOrders A -> Subset of LinPreorders A means
      for R being Element of LinPreorders A holds R in it iff
      for a,b st [a,b] in R & [b,a] in R holds a = b;
  end;

  reserve o for Element of LinPreorders A;

  definition
    let A,o,a,b;
    pred a <=_o, b means
      [a,b] in o;
  end;

  notation
    let A,o,a,b;
    synonym b >=_o, a for a <=_o, b;
    antonym b <_o, a for a <=_o, b;
    antonym a >_o, b for a <=_o, b;
  end;
  ```

If one traces back from these definitions and from the statement of the theorem, one also encounters the notions of

- the types of the *elements* and *subsets* of a given set
- *pairs* (written `[a,b]`)
- *relations* over a given set A
- *cardinality* of finite sets as a natural number
- the order relation $\leq$ on the natural numbers
- the number 3

These all are very standard notions in the Mizar library MML, that have been used in a very large number of formalizations, and therefore one can really trust that their formalized definitions correspond to the standard ones.

- Second, *one can 'justify' the formal definitions in the formalization by proving many lemmas about them.* For instance the Arrow formalization proves eleven lemmas (labeled `Th3` to `Th13`) about orders and preorders.
  The fact that it is possible to prove many familiar properties of the notions that are defined, gives some confidence that the formal definitions corresponds to our intuitive understanding of them.
- Third, *the fact that one can accurately follow the informal proof in the formal system is evidence that the formal notions correspond to the original versions.* The fact that we were able to exactly follow John Geanakoplos' proof is evidence that the formal notions that we are reasoning about are in fact the notions that occur in that proof.

When I first started on the formalization of Arrow's theorem, it took me a while to find the exact right formal statement that corresponded to the statement from the paper. This surprised me! I had thought that it would be quite apparent what the right statement would be. However, it turned out that there were various possible formal interpretations, and that only 'the right one' made the formal proof work.

The fact that the statement is about preorders (that is, the preferences that it reasons about allow 'ties' between alternatives) made this even harder.

For example, at first I had interpreted the property of *respecting unanimity* to be the fact that if every individual has $a \leq b$, then the group preference also has $a \leq b$. However, that turned out to be the wrong reading! With this interpretation the theorem does not even hold, as in that case one can for the group preferences take all alternatives to be equivalent. All the properties then are satisfied without there having to be a dictator. (The alternative interpretation of $a < b$ for every individual implying $a < b$ for the group turned out the right one. To my surprise, that interpretation is *not* stronger than the wrong interpretation. Each of them can be true with the other one being false.)

As another example I wondered whether it would be sufficient to just take the simple

```
for p,p',a,b st
  for i holds a <_p.i, b iff a <_p'.i, b
holds a <_f.p, b iff a <_f.p', b)
```

as the interpretation of *independence of irrelevant alternatives* instead of

```
for p,p',a,b st
  for i holds (a <_p.i, b iff a <_p'.i, b) & (b <_p.i, a iff b <_p'.i, a)
holds a <_f.p, b iff a <_f.p', b)
```

which is now in the formalization. That *would* have worked. However it would have made the theorem stronger, as this simpler statement is a more restrictive property.

There is a third interpretation issue that I missed at first. The proof in John Geanakoplos' paper does *not* establish *full* dicatorship. If the dictator takes two elements to be equivalent, then the proof does not guarantee that the group also takes them to be equivalent. This means that the dictator can force elements apart, but not together. (This problem does not occur in the variant of the theorem in the next section.)

11 A variant of the theorem

After I finished the formalization of the proof by John Geanakoplos, I wondered whether this was the strongest result possible. It seemed to me that if one restricts the options of the individuals more, then it becomes even stronger.

For this reason I considered the theorem in which the *group* was allowed to have ties between the alternatives, but the *individuals* were *not* allowed this freedom.

After some thought it seemed clear to me that this did not help much: if one could have a non-dictatorship rule for that situation, then one could extend it to preferences with ties by consistently breaking any ties in the individuals' preferences first (according to some fixed order on the alternatives) and then feeding that variant into the rule.

However, I was not entirely certain that I was not deluding myself with this argument. For this reason I decided to formalize this argument in Mizar too. That is, I then derived from the statement that I already had proved the variant (lines 530–532 and 977–984):

```
reserve A,N for finite non empty set;
reserve a,b for Element of A;
reserve i,n for Element of N;
reserve p,p' for Element of Funcs(N,LinOrders A);
reserve f for Function of Funcs(N,LinOrders A),LinPreorders A;

theorem
  (for p,a,b st for i holds a <_p.i, b holds a <_f.p, b) &
  (for p,p',a,b st
      for i holds a <_p.i, b iff a <_p'.i, b
    holds a <_f.p, b iff a <_f.p', b) &
  card A >= 3 implies
  ex n st for p,a,b holds a <_p.n, b iff a <_f.p, b
```

This is almost the same as the original, but with *orders* for the individuals' preferences, instead of *preorders*.

Also, because of this modification, the property of being independent of irrelevant alternatives could now be phrased a bit more economically. Just to be sure I made no mistake with this, the fact that for *orders* the simpler statement is equivalent to the larger statement was also proved as a lemma (lines 479–482):

```
theorem
  for o,o' being Element of LinOrders A holds
    ((a <_o, b iff a <_o', b) & (b <_o, a iff b <_o', a)) iff
    (a <_o, b iff a <_o', b)
```

Deriving this variant on Arrow's theorem from the one in John Geanakoplos' paper took quite a bit more work than I had expected from my informal understanding of the argument. This part of the formalization is lines 975–1120.

12 A journal of formalizations

The library of Mizar formalizations has two versions:

- A computer library of Mizar input text. This is called the *Mizar Mathematical Library* or MML. Another name for it is the *Journal of Formalized Mathematics* or JFM. The name depends on whether one thinks of it as a computer library or as a mathematical journal.
  Each formalization is in the MML in two versions:
  - The full formalization (called the 'article'). In the case of the Arrow formalization, this is the file `arrow.miz`.
  - An automatically generated version of the formalization (called the 'abstract') in which all proofs have been removed. In the case of the Arrow formalization, this is the file `arrow.abs`.
- A paper journal with abstracts in the English language (everything but the proofs), that are fully automatically generated from the Mizar input. This journal is called *Formalized Mathematics* or FM. Although in practice no one ever looks at these abstracts (everyone uses the MML files), it is nice that it exists too.

The version of the formalization of Arrow's theorem in *Formalized Mathematics* is slightly over three pages long [Wiedijk 2007a]. The statement of Arrow's theorem in this 'pretty-printed' version of the formalization is (on page 173):

---

3. Arrow's Theorem

For simplicity, we follow the rules: A, N are finite non empty sets, a, b are elements of A, i, n are elements of N, p, p' are elements of (LinPreorders $A)^N$, and f is a function from (LinPreorders $A)^N$ into LinPreorders A.

We now state the proposition

(14) Suppose that

(i) for all p, a, b such that for every i holds $a <_{p(i)} b$ holds $a <_{f(p)} b$,

(ii) for all p, p', a, b such that for every i holds $a <_{p(i)} b$ iff $a <_{p'(i)} b$ and $b <_{p(i)} a$ iff $b <_{p'(i)} a$ holds $a <_{f(p)} b$ iff $a <_{f(p')} b$, and

(iii) card $A \geq 3$.

Then there exists n such that for all p, a, b such that $a <_{p(n)} b$ holds $a <_{f(p)} b$.

The fact that the Mizar group calls *Formalized Mathematics* a *journal*, and even sends the authors reprints of the abstracts that are published in it, is a bit misleading. A journal is meant to be read, while this text generation is just a nice *tour de force*, with no one ever really looking at the abstracts after they have been printed. In a certain sense, the 'journal' called *Formalized Mathematics* mainly is there to satisfy university administrators who want to see publications.

However, related to this there is a real issue. Writing a formalization can be a lot of work, and it is important that it is done to exercise the technology. However, the academic credit that a researcher gets out of it is minor. One can publish a small report about it, but that is not in proportion to the amount of work that went into it. (Recently a new journal called *Journal of Formalized Reasoning* was started by Andrea Asperti, to be a platform for reports about formalizations. However, until now it has not had many submissions.)

13 Referee reports for a formalization

Generally libraries of formalizations (like the *contribs* of the Coq system and the Archive of Formal Proofs for Isabelle) are just collections of formalizations without much attention to unity and quality. The Mizar people are taking this aspect more seriously than the other proof assistant communities, but still the Mizar library is quite inhomogeneous.

Recently the Mizar community has decided to pay more attention to this aspect. I was very surprised that after I submitted my work on Arrow's theorem to their library, instead of just getting a nice 'thank you for your work, we will add it to our library' e-mail, I got *referee reports*. It suddenly felt surpringly much like submitting a article to a scientific journal.

I got three referee reports that gave my formalization the following judgments:

- *Confidence:* A, A, A

  A = very confident
  B = quite confident
  C = not very confident

- *The decision:* B, A, A

  A. accept as is (editorial changes only, can be done by the editor)
  B. accept, requires changes by the author to be approved by the editor
  C. reject, substantial author's revisions needed before resubmission for another review
  D. decision delayed, MML revision needed
  E. reject, no hope of getting anything of value

- *Presentation:* 2, 2, 3
- *The quality of formalization:* 2, 2, 3
- *Significance for MML:* 3, 2, 3

  0 – very poor
  1 – poor
  2 – good
  3 – very good

Apart from these judgments, the referees had some objections:

- All referees thought that I had divided the formalization in too many subsections. (This is done in Mizar using the keyword '`begin`'.) For instance the first referee wrote:

  > *In my opinion, the number of sections is too high as for file with no more than 1200 lines; it is o'k for the Mizar article, in 'Formalized Mathematics' it will look rather ugly.*
  > *I would expect some more material in here but I let Editors to decide what to do.*

  Apparently this referee also thought the article was too short. (I had been trying to keep the formalization as short and compact as possible. If I had written a less compact formalization, I would not have gotten this criticism.)

- The first referee did not like it that I had introduced my own version of orders and preorders, instead of using the already existing definitions in the Mizar library:

  > *Main shortcoming of the formalization is that the Mizar apparatus is not fully used; instead of the ordering wrt the relation, appropriate structure can be used. Also redefinition for 'Orders' seems to be unnecessary. I understand however that the point is in the one-to-one translation from its informal counterpart.*

  However, I *had* to introduce my own definitions to get the notation that I wanted. I could have used some definitions from the library in them, but then I just would have had a bit more work to translate between those definitions and mine, and it would not have made the definitions that I added more useful to others.

- The third referee had an issue related to a technical point that I had been fighting with. I needed the '`p.i`' that occur in the formalization to have the appropriate type, in order for them to work with the '`a <_o, b`' notations. However, the definitions in the Mizar library did not give me the right types. (Specifically: variables that have type `Element of LinOrders A` also automatically have type `Element of LinPreorders A`, and if `p` has type `Element of Funcs(N,LinOrders A)`, then `p.i` automatically gets type `Element of LinOrders A`. However, the expression `p.i` did *not* get type `Element of LinPreorders A`.)

  To get around this problem with the Mizar type system, I had added the 'redefinition' (lines 27–37):

```
definition
  let A,B' be non empty set;
  let B be non empty Subset of B';
  let f be Function of A,B;
  let x be Element of A;
  redefine func f.x -> Element of B;
  ...
end;
```

  (To express my irritation with the fact that the Mizar library made the types behave in a difficult to understand way, I had put a comment '`Mizar weirdness`' in front of this. The first referee thought that this was 'too informal'.)

  Now the third referee wanted me (for a reason that I still do not fully grasp) to use a different notation for function application with this redefinition:

  > I would rather use a new notation for the redefinition of Element:
  >
  > ```
  > notation
  >   let A,B' be non empty set;
  >   let B be non empty Subset of B';
  >   let f be Function of A,B;
  >   let x be Element of A;
  >     synonym f(.)x for f.x;
  > end;
  > ```
  >
  > Keeping the original notation might be troublesome for people using the definition in other articles. It may override other redefinitions.
  > I have attached corrected text. I do not insist on '`(.)`', maybe the Author could find something better.

  My reaction to this was that I found it completely unacceptable to have to use a different notation for the very standard function application operation, just because I needed it to have the correct type. It would mean doing something very bad (using strange notation for something common) to prevent the *possibility* of something a little bad (a definition being hidden) from happening.

When I submitted the formalization I had expected to have been finished with it already, and I did not feel like putting in much more time. For this reason I just addressed the first point by reducing the number of subsections and making my comments a bit more polite. Then I mailed the Mizar people that essentially I had no interest in addressing the other two points and that they could take it or leave it. After which they took it.

14 Conclusions

14.1 Formalization can be useful outside mathematics and computer science

Outsiders to the field of formalization might expect that its use is restricted to mathematical logic and computer science, or maybe also to pure mathematics, but not beyond that. The example from this paper shows otherwise.

It is important for formalization technology to be widely applicable, that this technology is exercised in many different domains. The formalization presented in this paper can be seen as being part of that effort.

I believe that in any field where one reasons in a mathematical style, even in an applied science like economics, one can make good use of proof assistants.

14.2 Possible future work

There are various ways in which one might continue the work described in this paper.

One might also try to formalize the other two proofs from the *Three Brief Proofs of Arrow's Impossibility Theorem.*

The *Gibbard-Satterthwaite theorem*, is generally mentioned together with Arrow's theorem. Therefore it is natural to formalize this theorem after formalizing Arrow's theorem. This is what both Tobias Nipkow and Peter Gammie did.

After I finished my formalization, Krzysztof Apt pointed out to me Philip Reny's *Arrow's Theorem and the Gibbard-Satterthwaite Theorem: A Unified Approach* [Reny 2000]. In this article both Arrow's theorem and the Gibbard-Satterthwaite theorem are proved, with both proofs next to each other in two columns, showing how closely they are related. It would be interesting to investigate whether one could do the same thing in Mizar, that is, to have two Mizar proofs run in parallel in a similar manner.

Of course one also can look into formalization of other theorems from economics. There are many other theorems from the field of social choice theory (like the ones in [Taylor 2005]) that one could formalize.

A nice theorem *outside* of social choice theory that might be interesting for formalization is a theorem by Kenneth Arrow called the *Arrow-Debreu theorem* [Varian 1992]. Krzysztof Apt already sent me a chapter from a set of course notes that explains this proof [Papadimitriou 2008]. The Arrow-Debreu theorem happens to be a direct consequence of Brouwer's fixed point theorem, which already has been formalized in various proof assistants. However, the version of this theorem in Mizar is not the n-dimensional simplex version that is needed for the Arrow-Debreu theorem, but just a 2-dimensional version. This means that the real work in formalizing the Arrow-Debreu theorem in Mizar would be to formalize the more-dimensional generalization of Brouwer's fixed point theorem.

14.3 Investing into formalization

Formalization is a relatively labor-intensive activity, but on the other hand it is not impossibly difficult. The work reported on in this paper altogether took about one work-week (including the false start). Writing the final formalization took about three work-days, which in between other kinds of work took slightly over one week.

The fact that the project described here only took this modest amount of time strongly suggests that more people should look into formalizing their work. This will allow them to be able to then really trust their results, and to gain a very precise understanding of their proofs.

Acknowledgments. Thanks to Krzysztof Apt for suggesting the project to me. Thanks to Tobias Nipkow and Peter Gammie for telling me about their formalizations in Isabelle/HOL and for helpful comments. Thanks to the referees of the Mizar library for their feedback on my work. Thanks to the referees of this journal for many helpful comments on this paper.

References

[Arrow 1950] Kenneth Arrow. A Difficulty in the Concept of Social Welfare. *Journal of Political Economy*, 58(4):328–346, August 1950.

[Asperti et al 2007] Andrea Asperti, Claudio Sacerdoti Coen, Enrico Tassi, and Stefano Zacchiroli. Crafting a Proof Assistant. In Thorsten Altenkirch and Conor McBride, editors, *Types for Proofs and Programs, International Workshop, TYPES 2006, Nottingham, UK, April 18-21, 2006, Revised Selected Papers*, volume 4502 of *Lecture Notes in Computer Science*, pages 18–32. Springer, 2007.

[Bancerek & Rudnicki 2003] Grzegorz Bancerek and Piotr Rudnicki. A Compendium of Continuous Lattices in Mizar. *Journal of Automated Reasoning*, 29(3–4):189–224, 2003.

[Barras 1999] Bruno Barras. *Auto-validation d'un système de preuves avec familles inductives.* Thèse de doctorat, Université Paris 7, November 1999.

[Barras & Werner 1997] Bruno Barras and Benjamin Werner. Coq in Coq. ⟨`http://pauillac.inria.fr/~barras/coqincoq.ps.gz`⟩.

[Blazy et al 2006] Sandrine Blazy, Zaynah Dargaye, and Xavier Leroy. Formal Verification of a C Compiler Front-end. In *FM 2006: Int. Symp. on Formal Methods*, volume 4085 of *Lecture Notes in Computer Science*, pages 460–475. Springer, 2006.

[Fox 2003] Anthony Fox. Formal Specification and Verification of ARM6. In David A. Basin and Burkhart Wolff, editors, *Theorem Proving in Higher Order Logics, 16th International Conference TPHOLs 2003, Rome, Italy, September 8-12, 2003, Proceedings*, volume 2758 of *Lecture Notes in Computer Science*, pages 25–40. Springer, 2003.

[Geanakoplos 2001] John Geanakoplos. Three Brief Proofs of Arrow's Impossibility Theorem. Technical Report 1123RRR, Cowles Foundation, June 2001.

[Gonthier 2006] Georges Gonthier. A computer-checked proof of the Four Colour Theorem. ⟨`http://research.microsoft.com/~gonthier/4colproof.pdf`⟩, 2006.

[Harrison 1995] John Harrison. HOL Done Right. ⟨`http://www.cl.cam.ac.uk/users/jrh/papers/holright.ps.gz`⟩, 1995.

[Harrison 2006] John Harrison. Towards self-verification of HOL Light. In Ulrich Furbach and Natarajan Shankar, editors, *Proceedings of the Third International Joint Conference IJCAR 2006*, volume 4130 of *Lecture Notes in Computer Science*, pages 177–191, Seattle, WA, 2006. Springer.

[Harrison 2008] John Harrison. Formalizing an Analytic Proof of the Prime Number Theorem (extended abstract). In Richard Boulton, Joe Hurd, and Konrad Slind, editors, *Tools and Techniques for Verification of System Infrastructure*, pages 17–22, London, 2008. The Royal Society.

[Leroy 2006] Xavier Leroy. Formal Certification of a Compiler Back-end, or: Programming a Compiler with a Proof Assistant. In *POPL'06*, Charleston, South Carolina, USA, 2006.

[Naumov et al 2001] Pavel Naumov, Mark-Oliver Stehr, and José Meseguer. The HOL/NuPRL Proof Translator: A Practical Approach to Formal Interoperability. In R.J. Boulton and P.B. Jackson, editors, *The 14th International Conference on Theorem Proving in Higher Order Logics*, volume 2152 of *LNCS*, pages 329–345. Springer-Verlag, 2001.

[Nipkow 2008] Tobias Nipkow. A Bit of Social Choice Theory in HOL: Arrow and Gibbard-Satterthwaite. In Richard Boulton, Joe Hurd, and Konrad Slind, editors, *Tools and Techniques for Verification of System Infrastructure*, page 9, London, 2008. The Royal Society.

[Obua & Skalberg 2006] Steven Obua and Sebastian Skalberg. Importing HOL into Isabelle/HOL. In Ulrich Furbach and Natarajan Shankar, editors, *IJCAR*, volume 4130 of *Lecture Notes in Computer Science*, pages 298–302. Springer, 2006.

[Papadimitriou 2008] Christos Papadimitriou. CS294-1 Algorithmic Aspects of Game Theory, Lecture 2: January 23, 2008. ⟨`http://www.cs.berkeley.edu/~kunal/cs294-1-lec2.ps`⟩.

[Reny 2000] Philip Reny. Arrow's Theorem and the Gibbard-Satterthwaite Theorem: A Unified Approach, June 2000.

[Routley 1979] Richard Routley. Repairing Proofs of Arrow's General Impossibility Theorem and Enlarging the Scope of the Theorem. *Notre Dame Journal of Formal Logic*, XX(4), October 1979.

[Taylor 2005] Alan D. Taylor. *Social Choice and the Mathematics of Manipulation.* Outlooks. Cambridge University Press, 2005.

[Varian 1992] Hal R. Varian. *Microeconomic Analysis.* W. W. Norton, 3rd edition, 1992.

[Wiedijk 2004] Freek Wiedijk. Formal Proof Sketches. In Stefano Berardi, Mario Coppo, and Ferruccio Damiani, editors, *Types for Proofs and Programs: Third International Workshop, TYPES 2003, Torino, Italy, April 30 – May 4, 2003, Revised Selected Papers*, volume 3085 of *LNCS*, pages 378–393, 2004.

[Wiedijk 2007] Freek Wiedijk. Writing a Mizar article in nine easy steps. ⟨`http://www.cs.ru.nl/~freek/mizar/mizman.pdf`⟩, 2007.

[Wiedijk 2007a] Freek Wiedijk. Arrow's Impossibility Theorem. *Formalized Mathematics*, 15(4):171–174, 2007.