STUDIES IN LOGIC, GRAMMAR AND RHETORIC 22 (35) 2010

Mariusz Giero
Institute of Sociology
University of Białystok

FORMALIZATION OF PROPOSITIONAL LINEAR TEMPORAL LOGIC IN THE MIZAR SYSTEM

Abstract: The paper describes formalization of some issues of propositional linear temporal logic (*PLTL*). We discuss encountered problems and applied solutions. The formalization was carried out in the Mizar system. In comparison with other systems, Mizar is famous for its large repository of computer checked mathematical knowledge and also for its user-friendly knowledge representation and proof language.

1. Introduction

Temporal logic [13, 6, 17] is a system of logic, whose language is enriched with temporal connectives. The purpose of temporal logic is reasoning about time (in philosophy), and about the behavior of systems evolving over time (in computer science). Enrichment with temporal connectives results in economy of wording and reasoning closer to intuition than if one uses classical first order logic. There are various temporal logics due to the choice of temporal operators, and what model of time is used. In computer science one usually considers propositional temporal logic with temporal operators *referring only to the future* and *linear flow of time* (isomorphic with the set of natural numbers). The main application of *PLTL* in computer science is the verification of finite-state reactive and concurrent systems (typical examples include microprocessors, operating systems, network protocols, etc.). One of the verification methods is a proof-theoretical method [18]. In this method, verification consists in *proving a theorem* expressing the desired property (e.g. freedom from deadlocks) in a formal deductive theory. The theory consists of formal language in which formulas are written, a set of axioms and inference rules. Propositions specifying the verified system are joined as premises to the theorem one is to prove. In the paper, we define such a deductive theory, with a future plan to be able to carry out verification by this method.

ISBN 978–83–7431–273–8 ISSN 0860-150X

Mizar is the name of a formal language designed by Andrzej Trybulec for writing strictly formalized mathematical definitions and proofs, but is also used as the name of a computer program which is able to check proofs written in this language [9, 15, 19, 1]. Mizar language is similar to the language used in mathematical texts. This makes it an attractive tool for mathematicians who do not have to put much effort in mastering it, gainig the opportunity to check their texts by a machine. Texts written in Mizar (called *articles*), positively verified by the program and positively reviewed by people, are added to the Mizar database of mathematical knowledge (*Mizar Mathematical Library – MML* for short). Currently, *MML* contains over 1,100 articles with more than 50,000 proved theorems, and almost 10,000 definitions of various fields, mostly from mathematics, but also computer science. It is considered to be the largest such database in the world. The list of most important theorems (available at [3]) includes, among others:

- The Fundamental Theorem of Algebra
- The Fundamental Theorem of Arithmetic
- The Fundamental Theorem of Calculus
- The Jordan Curve Theorem
- The Gödel Completeness Theorem

There is also defined Armstrong's deductive system in *MML,* used to infer all the functional dependencies on a relational database.

As an example of readability of Mizar text and its similarity to standard mathematical language let us present the proof of one of the set theory properties:

```
X /\ (Y \/ Z) c= (X /\ Y) \/ (X /\ Z)
proof
  let x be set;
  assume A1: x in X /\ (Y \/ Z);
  then x in Y \/ Z by XBOOLE_0:def 4;
  then A2: x in Y or x in Z by XBOOLE_0:def 3;
  x in X by A1, XBOOLE_0:def 4;
  then x in X /\ Y or x in X /\ Z by A2, XBOOLE_0:def 4;
  hence x in (X /\ Y) \/ (X /\ Z) by XBOOLE_0:def 3;
end;
```

The symbols `/\`, `\/`, `c=`, `in`, `or` denote in Mizar the conventional ones: $\cap$, $\cup$, $\subseteq$, $\in$, $\vee$. Expressions of the form: `by XBOOLE_0:def 4` are justifications with theorems contained in *MML* or local (labeled) propositions. `set` is the broadest type in the Mizar system. Any object extends to the type `set`.

The purpose of formalization is primarily to build a digital database of computer-checked mathematical knowledge. The advantages of such database in comparison with mathematics literature can not be overestimated. The knowledge is stored in one place. It is easier to access, search, process and, most importantly, it has a computer guarantee of correctness. It can also be used for didactic purposes [14, 5].

Formalization is often a tedious activity. Frequently, coding based on paper publication cannot be done directly. Definitions are not precise and formal, proofs (among others, of trivial facts that turn out not necessarily to be trivial) are left to the reader. It happens that the construct of a definition or a proof can not be introduced due to the limitations of a proof-checker system and the user has to invent an equivalent one. Much also depends on what is already formalized in the database of the system. If it lacks basic knowledge in a given field, one, at first, has to introduce it and after that can deal with desired formalization.

The formalization carried out by the author is based mostly on [13, Ch. 2, 3]. There are basics of propositional linear temporal logic with only future-referring connectives in these chapters. We defined syntax and semantics of language of this logic (Sec. 2). An axiomatic system of this logic was introduced (Sec. 3). We proved the soundness theorem and the deduction theorem for this system (Sec. 4, 5). Additionally, numerous minor lemmas and facts were proved. In each section, first, what was formalized is presented, and then how the formalization was carried out is described together with encountered problems and solutions applied. The Mizar code of the formalization is available in [7]. It can be accessed in the *MML* as the Mizar article with the identifier `LTLAXIO1`.

The carried out formalization extends knowledge on temporal logic already stored in the *MML* (see 2.2.3).

2. The language of *PLTL*

2.1. What was formalized

Definition 1.

Syntax of $PLTL$

$\phi ::= \bot \mid pl \mid (\phi \Rightarrow \phi) \mid (\phi \,\mathbf{Us}\, \phi)$

$pl ::= p \mid pl'$

$\bot$ is the *constant false*, pl is a *propositional letter* (for readability we denote propositional letters by p_1, $p_2, p_3, \ldots$ or p, q, r instead of p, p', p'', $\ldots$), $\Rightarrow$

is a classical connective called *implication*, and **Us** is a temporal connective called *strong until*. Others classical and temporal connectives are introduced as abbreviations:

Abbreviation.

$\neg\phi \stackrel{def}{=} (\phi \Rightarrow \bot)$(*negation*)
$\top \stackrel{def}{=} \neg\bot$ (*constant true*)
$\phi_1 \vee \phi_2 \stackrel{def}{=} (\neg\phi_1 \Rightarrow \phi_2)$(*disjunction*)
$\phi_1 \wedge \phi_2 \stackrel{def}{=} \neg(\phi_1 \Rightarrow \neg\phi_2)$ (*conjunction*)
$\phi_1 \Leftrightarrow \phi_2 \stackrel{def}{=} (\phi_1 \Rightarrow \phi_2) \wedge (\phi_2 \Rightarrow \phi_1)$ (*equivalence*)
$\mathbf{X}\,\phi \stackrel{def}{=} (\bot\,\mathbf{Us}\,\phi)$ (*next*)
$\phi_1\,\mathbf{U}\,\phi_2 \stackrel{def}{=} \phi_2 \vee (\phi_1 \wedge (\phi_1\,\mathbf{Us}\,\phi_2))$ (*weak until*)
$\mathbf{G}\,\phi \stackrel{def}{=} \phi \wedge \neg(\top\mathbf{U}\,\neg\phi)$ (*global*)
$\mathbf{F}\,\phi \stackrel{def}{=} \neg\mathbf{G}\,\neg\phi$ (*future*)
$\phi_1\,\mathbf{R}\,\phi_2 \stackrel{def}{=} \neg(\neg\phi_1\,\mathbf{U}\,\neg\phi_2)$ (*release*)

Definition 2.

Let PL be a set of all propositional letters. A **model** is a function $\mathcal{M} : \mathbb{N} \longrightarrow 2^{PL}$, where $\mathbb{N}$ is a set of natural numbers.

Definition 3.

The **satisfaction** of a $PLTL$ formula ϕ in model $\mathcal{M}$ at time point $n \in \mathbb{N}$ ($\mathcal{M}, n \models \phi$) is defined inductively as follows:

1. $\mathcal{M}, n \not\models \bot$, for every $n \in \mathbb{N}$
2. $\mathcal{M}, n \models pl$ iff $pl \in \mathcal{M}(n)$
3. $\mathcal{M}, n \models \phi_1 \Rightarrow \phi_2$ iff $\mathcal{M}, n \not\models \phi_1$ or $\mathcal{M}, n \models \phi_2$
4. $\mathcal{M}, n \models \phi_1\,\mathbf{Us}\,\phi_2$ iff
   there exists $i \in \mathbb{N}$, $i > 0$: $\mathcal{M}, n+i \models \phi_2$ and
   for every $j \in \mathbb{N}$, $1 \leqslant j < i$: $\mathcal{M}, n+j \models \phi_1$

According to the above definition, the semantics of temporal connectives introduced as abbreviations is as follows:

1. $\mathcal{M}, n \models \mathbf{X}\phi$ iff $\mathcal{M}, n+1 \models \phi$
2. $\mathcal{M}, n \models \mathbf{G}\phi$ iff for every $i \in \mathbb{N}$, $i \geqslant 0$: $\mathcal{M}, n+i \models \phi$
3. $\mathcal{M}, n \models \mathbf{F}\phi$ iff there exists $i \in \mathbb{N}$, $i \geqslant 0$: $\mathcal{M}, n+i \models \phi$
4. $\mathcal{M}, n \models \phi_1\,\mathbf{U}\,\phi_2$ iff
   there exists $i \in \mathbb{N}$, $i \geqslant 0$: $\mathcal{M}, n+i \models \phi_2$ and
   for every $j \in \mathbb{N}$, $j < i$: $\mathcal{M}, n+j \models \phi_1$

5. $\mathcal{M}, n \models \phi_1 \,\mathbf{R}\, \phi_2$ iff
   there exists $i \in \mathbb{N}$, $i \geqslant 0$: $\mathcal{M}, n+i \models \phi_1$ and
   for every $j \in \mathbb{N}$, $j \leq i$: $\mathcal{M}, n+j \models \phi_2$
   or
   for every $i \in \mathbb{N}$: $\mathcal{M}, n+i \models \phi_2$

Definition 4.

A formula ϕ is **valid** in model $\mathcal{M}$ (denoted: $\mathcal{M} \models \phi$) iff for every $n \in \mathbb{N}$: $\mathcal{M}, n \models \phi$

2.2 The formalization

2.2.1. Syntax

The formulas of *PLTL* are defined as finite sequences of natural numbers with the use of Polish notation (prefix notation). $\bot$ is the sequence $\langle 0 \rangle$, propositional letters are sequences of the form $\langle n \rangle$, where $n \geqslant 3$, implications are sequences whose the first element is 1, and *strong until* formulas are sequences whose the first element is 2 [7, see synonyms].

Definition 5.

The set of *PLTL* formulas is the set D such that:

1. $\langle 0 \rangle \in D$
2. if $n \geqslant 3$, then $\langle n \rangle \in D$
3. if $\phi_1, \phi_2 \in D$, then $\langle 1 \rangle^\frown \phi_1 {}^\frown \psi_2 \in D$
4. if $\phi_1, \phi_2 \in D$, then $\langle 2 \rangle^\frown \phi_1 {}^\frown \psi_2 \in D$
5. for every set D' satisfying 1–4 holds $D \subseteq D'$,

where $\frown$ is a concatenation of sequences. The set D is therefore the smallest set satisfying 1–4.

Example 6.

Let p, q, r be, respectively, sequences $\langle 3 \rangle$, $\langle 4 \rangle$, $\langle 5 \rangle$. The formula $(p \Rightarrow q)\,\mathbf{Us}\, r$ is represented by the sequence $\langle 2, 1, 3, 4, 5 \rangle$. The formula $p\,\mathbf{U}\,q$, after unfolding abbreviations: $(q \Rightarrow \bot) \Rightarrow ((p \Rightarrow ((p\,\mathbf{Us}\,q) \Rightarrow \bot)) \Rightarrow \bot)$, is represented by the sequence $\langle 1, 1, 4, 0, 1, 3, 1, 2, 3, 4, 0, 0 \rangle$.

This method of defining the set of formulas of a language (i. e. as finite sequences of natural numbers) has been used, among others, in [8] for the language of classical propositional logic without negations (*PPL*).

Definition 7.

Syntax of *PPL*

$\phi ::= \top \,|\, pl \,|(\phi \Rightarrow \phi)|(\phi \wedge \phi)$,

$pl ::= p \,|\, pl'$.

$\top$ is the sequence $\langle 0 \rangle$, propositional letters are sequences of the form $\langle n \rangle$, where $n \geqslant 3$, implications are sequences whose first element is 1, and conjunctions are sequences whose first element is 2. In terms of **syntax** the language of *PLTL* is no different from the language of *PPL* ($\bot$ corresponds to $\top$, **Us** corresponds to the binary connective $\wedge$, and the other two symbols *pl* and $\Rightarrow$ are the same). Thus, the definition of *PLTL* was not introduced from the beginning, repeating the definition of *PPL* but there was used one of the features of Mizar language allowing to define synonyms. The language of *PLTL* was defined as a synonym for *PPL*. Thanks to this, the definition took only a few lines of code [7, see synonyms] and it also allowed to make use of several theorems proved in [16], among others, the principal of structural induction for *PPL* which licenses the definition of function with domain *PPL* by structural recursion. Proving the analogous principle for *PLTL* would take about several hundred lines of code.

Abbreviations of other connectives were introduced as *functors* with the `equals` construct [9, p. 18]. The Mizar code for the connective **U** is as follows:

```
definition let p, q be Element of LTLB_WFF ;
  func p 'U' q -> Element of LTLB_WFF equals :: LTLAXIO1:def 8
  q 'or' (p '&&' (p 'Us' q));
end;
```

where `LTLB_WFF` denotes the set of *PLTL* formulas, `'or'` denotes $\vee$, and `'&&'` denotes $\wedge$.

2.2.2. Semantics

It was not possible to formalize the concept of satisfaction directly as it is placed conventionally in literature, i.e., as a recursive predicate (def. 3), due to the fact that Mizar does not allow for this kind of definitions. However, the Mizar system allows defining recursive functions, and with the use of the recursive function $SAT_{\mathcal{M}}$ the satisfaction was defined [7, def. 11]:

Definition 8.

$\mathcal{M}, n \models \phi$ iff $SAT_{\mathcal{M}}(n, \phi) = 1$,

where $SAT_{\mathcal{M}}$ is defined as follows:

Definition 9.

Let $\mathcal{M}$ be a model. $SAT_{\mathcal{M}} : \mathbb{N} \times PLTL \to \{0, 1\}$ is a function satisfying the following conditions:

1. $SAT_{\mathcal{M}}(n, \bot) = 0$, for every $n \in \mathbb{N}$
2. $SAT_{\mathcal{M}}(n, pl) = 1$ iff $pl \in \mathcal{M}(n)$
3. $SAT_{\mathcal{M}}(n, \phi_1 \Rightarrow \phi_2) = 1$ iff $imp(SAT_{\mathcal{M}}(n, \phi_1), SAT_{\mathcal{M}}(n, \phi_2)) = 1$
4. $SAT_{\mathcal{M}}(n, \phi_1 \mathbf{Us}\, \phi_2) = 1$ iff
   there exists $i \in \mathbb{N}$, $i > 0$: $SAT_{\mathcal{M}}(n + i, \phi_2) = 1$ and
   for every $j \in \mathbb{N}$, $1 \leqslant j < i$: $SAT_{\mathcal{M}}(n + j, \phi_1) = 1$,

where *imp* is the Boolean function of implication defined in [2].

The notion of validity in model is then defined directly as the definition 8 says. The Mizar code of the definition of validity is as follows:

```
definition
  let M be LTLModel;
  let p be Element of LTLB_WFF;
  pred M |= p means :: LTLAXIO1:def 12
    for n being Element of NAT holds (SAT M).[n,p] = 1;
end;
```

The function $SAT_{\mathcal{M}}$ was defined as a *functor* which required proving of correctness conditions, i. e., showing that such an object exists (*existence* condition) and is uniquely determined (*uniqueness* condition) [9, p. 17]. There is no general theorem which licenses definition of function by structural recursion. Each case requires a separate proof. In this paper, we made use of the fact that *PLTL* is syntactically synonym of *PPL* and the theorem about the existence of a recursive function over the set of *PPL* formulas [16] was applied.

There were proved theorems [7, Th. 5–13] about the correctness of semantics of connectives introduced as abbreviations (def. 3). The Mizar code for temporal operator **G** is as follows:

```
theorem :: LTLAXIO1:10
for A being Element of LTLB_WFF
for n being Element of NAT
for M being LTLModel holds
((SAT M).[n,('G' A)] = 1 iff
 for i being Element of NAT holds (SAT M).[(n+i),A]=1
)
```

2.2.3. Related Work

There are three Mizar articles concerning temporal logic in *MML*: `MODELC_1` (on computational tree logic) [10], `MODELC_2` (on propositional linear time temporal logic) [11] and `MODELC_3` (on programs verification with the use of Büchi automaton) [12]. In [11] there was defined the following syntax of *PLTL*:

$$\phi ::= pl \,|\neg\phi|\, (\phi \wedge \phi) \,|\, (\phi \vee \phi) \,|\, \mathbf{X}\phi \,|\, (\phi\, \mathbf{U}\, \phi) \,|\, (\phi\, \mathbf{R}\, \phi)$$
$$pl ::= p \,|\, pl'$$

Adding to the language connectives, which can be introduced as abbreviations results in extension of proofs. For example, if one wants to use the principal of structural induction, instead of two cases (two connectives) one has to consider six (for each connective). So, the author of this paper decided to redefine the language of *PLTL* introducing the optimal number of connectives. In [11] formulas of *PLTL* are also formalized as finite sequences of natural numbers with the use of Polish notation. However, the set of these formulas, introduced in [11, def. 9], can be formalized in a shorter and more readable way using the Mizar concept of *attributes* [9, p. 15]. That was done in [8] for *PPL* and we based our definition (def. 5) for *PLTL* on the solution.

Semantics of *PLTL* is defined in [11] by abstract algebra with operations corresponding to connectives. The definition is complex and far from the conventional one known from literature. So, we decided to introduce a new definition (def. 9) which seems to be as close as possible to the conventional one.

3. Axiomatization

3.1. What was formalized

3.1.1 Axioms

The set of axioms consists of:

(A1) all tautologies of classical propositional logic of the *PLTL* language and all the formulas of the form:

(A2) $\neg\mathbf{X}\phi \Leftrightarrow \mathbf{X}\neg\phi$

(A3) $\mathbf{X}(\phi_1 \Rightarrow \phi_2) \Rightarrow (\mathbf{X}\phi_1 \Rightarrow \mathbf{X}\phi_2)$

(A4) $\mathbf{G}\phi \Leftrightarrow \phi \wedge \mathbf{XG}\phi$

(A5) $\phi_1 \mathbf{Us}\, \phi_2 \Leftrightarrow \mathbf{X}\phi_2 \vee \mathbf{X}(\phi_1 \wedge (\phi_1 \mathbf{Us}\, \phi_2))$

(A6) $\phi_1 \mathbf{Us}\, \phi_2 \Rightarrow \mathbf{XF}\phi_2$

A formula is of the form of (A1) if it results from a tautology of classical propositional logic by consistently replacing the propositional letters of the tautology by formulas of *PLTL*.

Example 10.
The formula $\mathbf{X}p \vee \neg\mathbf{X}p$ is of the form of (A1). It results from the tautology $p \vee \neg p$ by replacing p by $\mathbf{X}p$.

Axioms of the form of (A2) assert linearity of time, (A4), (A5) are the fixpoint characterizations of operators **G** and **Us**, respectively, and axioms of the form of (A6) indicate that the strong version of *until* is used.

3.1.2. Inference rules

There are three inference rules:

(MP): $\phi_1, \phi_1 \Rightarrow \phi_2 \ / \phi_2$

(NEX): $\phi \ / \ \mathbf{X}\phi$

(IND): $\phi_1 \Rightarrow \phi_2, \phi_1 \Rightarrow \mathbf{X}\phi_1 \ / \ \phi_1 \Rightarrow \mathbf{G}\phi_2$

3.2. The formalization

3.2.1. Axioms

Formalization of the axioms of the form (A1) directly from [13] would require, among others, a definition of replacing operation and a definition of a subformula. We have defined these axioms in another equivalent way, seemingly easier and shorter in formalization. We have used a modified zero-one method for verification whether a formula is a classical propositional logic tautology. The modification consists in the fact that we valuate all the formulas (in particular case also propositional letters) being an argument of an implication but not being an implication themselves. In other words, we valuate $\perp$, propositional letters and *until* formulas, only if they are arguments of an implication (the only classical connective of *PLTL*). The exception is the $\perp$ formula which is always valuated by 0. Then, we calculate the value of the verified formula in accordance with the Boolean function of implication. If, at any such valuation, the value is 1, the formula is of the form (A1), otherwise it is not. If a verified formula is not an implication, we valuate the whole formula. So this type of formula is not of the form (A1), because its value can be 0.

Formalization was carried out in two steps. First, we defined the function V_f for calculating the value of the formula:

Definition 11.

Let $f : PLTL \to \{0,1\}$ be a function and ϕ be a formula of *PLTL*. The value of the formula ϕ at valuation f calculates the function $V_f : PLTL \to \{0,1\}$ such that:

$$V_f(\phi) = \begin{cases} 0, & \text{if } \phi = \bot \\ f(\phi), & \text{if } \phi \in pl \text{ or } \phi = \phi_1 \mathbf{Us}\ \phi_2 \\ imp(V_f(\phi_1), V_f(\phi_2)), & \text{if } \phi = \phi_1 \Rightarrow \phi_2 \end{cases}$$

where *imp* is the Boolean function of implication defined in [2], and then we defined the formulas of the form of (A1) as follows:

Definition 12.

A formula ϕ is of the form of (A1) iff for every function $f : PLTL \to \{0,1\}$ holds $V_f(\phi) = 1$.

The function f valuates any formula, but V_f uses only the values that f assigns to propositional letters and *until* formulas which are not arguments of an implication. If we do not need all values of f, then we could restrict the domain of f to the set of propositional letters and *until* formulas, but it is not necessary. It is always better to have for an object as broad type as possible. Some difficulty in understanding this definition can cause the fact that f can assign 1 to $\bot$, while V_f always assigns 0. However, there is no inconsistency, because this particular value of f is not used by V_f. The restriction of a function f would be worth to consider in case of implementation of the method, it could affect how long the program would run.

Example 13.

The formula $\mathbf{X}p \vee \neg\mathbf{X}p$ is of the form of (A1). We evaluate the function V_f for the formula:

$$\begin{aligned} V_f(\mathbf{X}p \vee \neg\mathbf{X}p) &= V_f(((\bot \mathbf{Us}\, p) \Rightarrow \bot) \Rightarrow ((\bot \mathbf{Us}\, p) \Rightarrow \bot)) \\ &= imp(imp(f(\bot \mathbf{Us}\, p), 0), imp(f(\bot \mathbf{Us}\, p), 0) \end{aligned}$$

and we have:

if $f(\bot \mathbf{Us}\, p) = 0$,

then $V_f(Xp \vee \neg Xp) = imp(imp(0,0), imp(0,0)) = imp(1,1) = 1$,

and also

if $f(\bot \mathbf{Us}\, p) = 1$,

then $V_f(Xp \vee \neg Xp) = imp(imp(1,0), imp(1,0)) = imp(0,0) = 1$.

The function V_f is recursive. As in the definition of the function $SAT_{\mathcal{M}}$, we had to show that such a function exists and we also used the principal of structural induction proved in [16]. The Mizar code of the definition of V_f (denoted in Mizar as: `VAL f`) is as follows:

```
definition
  let f be Function of LTLB_WFF,BOOLEAN;
  func VAL f -> Function of LTLB_WFF,BOOLEAN means
    :: LTLAXIO1:def 15
    for A, B being Element of LTLB_WFF
    for n being Element of NAT holds
    (it.TFALSUM = 0 & it.(prop n)=f.(prop n) &
    it.(A=>B)=(it.A)=>(it.B) &
    it.(A 'Us' B)=f.(A 'Us' B));
  correctness;
end;
```

The set of axioms is defined as the smallest subset AKS of the set of $PLTL$ formulas, i.e., the set satisfying the following conditions:

1. If ϕ is of the form of A1, then $\phi \in AKS$
2. If ϕ is of the form of A2–A6, then $\phi \in AKS$
3. for every set AKS' satisfying 1–2 holds $AKS \subseteq AKS'$

3.2.2. Inference rules

The rules were defined as relations on the set of $PLTL$. The definition of the MP-rule is as follows:

$$(\phi_1, \phi_2, \phi_3) \in MP \text{ iff } \phi_2 = \phi_1 \Rightarrow \phi_3$$

The Mizar code of the definition is as follows:

```
definition
  let p,q,r be Element of LTLB_WFF ;
  pred p,q MP_rule r means :: LTLAXIO1:def 19
  q = p => r;
end;
```

4. The Soundness Theorem

Theorem.

For every set of formulas $\mathcal{F}$ and a formula ϕ holds: if $\mathcal{F} \vdash \phi$, then $\mathcal{F} \models \phi$.

The Mizar code of the theorem is as follows:

```
theorem :: LTLAXIO1:41
  for A being Element of LTLB_WFF
  for F being Subset of LTLB_WFF st F |- A
  holds F |= A;
```

Semantic consequence, $\mathcal{F} \models \phi$, introduces the following definition:

```
definition
  let F be Subset of LTLB_WFF;
  let p be Element of LTLB_WFF ;
  pred F |= p means :: LTLAXIO1:def 14
    for M being LTLModel st M |= F holds M |= p;
end;
```

A set of formulas $\mathcal{F}$ is valid in model $\mathcal{M}$ (denoted in Mizar: `M |= F`) iff every formula being element of F is valid (as stated in def. 4) in model $\mathcal{M}$.

Syntactic consequence, $\mathcal{F} \vdash \phi$ (a derivation of a formula ϕ from a set of formulas $\mathcal{F}$) is defined conventionally, i.e. as a finite sequence of formulas such that:

- elements of the sequence are: axioms or elements of the set $\mathcal{F}$ or formulas derived from the earlier element(s) by inferences rules
- the last element of the sequence is the formula ϕ.

The proof of the theorem is based on [13]. From assumption, we have that there exists a finite sequence of formulas f (as described above) such that the last element of the sequence is ϕ. Then, we prove that for every element of the sequence f holds $\mathcal{F} \models f_i$, where $1 \leqslant i \leqslant |f|$, applying the following induction rule [4, sch. 4]:

$$\forall i \in \mathbb{N}\ P(i) \text{ iff } \forall i \in \mathbb{N}\ (\forall k \in \mathbb{N}\ (k < i \Rightarrow P(k)) \Rightarrow P(i)), \quad \text{(indrule)}$$

where predicate $P(i)$ is defined as $P(i) \stackrel{def}{=} \mathcal{F} \models f_i$

Since $P(i)$ holds for every element of the sequence f, it also holds for the last element and the thesis is proved. The proof required considering all the cases of what f_i can be. For each case, semantic consequence was showed [7, Th. 24–27, 37]:

- any axiom is semantic consequence of $\mathcal{F}$
- any element of $\mathcal{F}$ is semantic consequence of $\mathcal{F}$
- for each of the rules of inference: if the premises is semantic consequence of $\mathcal{F}$, then so is the conclusion.

5. The Deduction Theorem

Theorem.

For every set of formulas $\mathcal{F}$ and formulas ϕ_1, ϕ_2 holds: if $\mathcal{F} \cup \{\phi_1\} \vdash \phi_2$, then $\mathcal{F} \vdash \mathbf{G}\phi_1 \Rightarrow \phi_2$.

The Mizar code of the theorem is as follows:

```
theorem :: LTLAXIO1:57
  for p,q being Element of LTLB_WFF
  for F being Subset of LTLB_WFF st F \/ {p}|- q
  holds F|-('G' p) => q;
```

The proof of the theorem [13] is analogous to the previous one. From assumption, we have that there exists a finite sequence of formulas f such that the last element of the sequence is ϕ_2. Then, we prove that for every element of the sequence f holds $\mathcal{F} \vdash \mathbf{G}\phi_1 \Rightarrow f_i$, where $1 \leqslant i \leqslant |f|$, applying induction rule (*indrule*). We define $P(i)$ as follows:

$$P(i) \stackrel{def}{=} \mathcal{F} \vdash \mathbf{G}\phi_1 \Rightarrow f_i.$$

Since $P(i)$ holds for every element of the sequence f, it also holds for the last element and the thesis is proved.

Sketch of the proof:

We consider all the cases of what f_i can be:

1. The formula $f_i \Rightarrow (\mathbf{G}\phi_1 \Rightarrow f_i)$ is an axiom of the (A1) form, so $\mathcal{F} \vdash f_i \Rightarrow (\mathbf{G}\phi_1 \Rightarrow f_i)$. If f_i is an axiom or $f_i \in \mathcal{F}$, then $\mathcal{F} \vdash f_i$. Then, we infer $P(i)$ applying (MP) rule.
2. The formula $\mathbf{G}\phi_1 \Leftrightarrow \phi_1 \wedge \mathbf{XG}\phi_1$ is an axiom of the (A4) form, so $\mathcal{F} \vdash \mathbf{G}\phi_1 \Leftrightarrow \phi_1 \wedge \mathbf{XG}\phi_1$. If $f_i = \phi_1$, then we infer $P(i)$ applying (MP) rule and the classical logic tautology (particular case of (A1)).
3. If f_i is derived from a inference rule, e.g., (*NEX*) rule, then there exists $j < i$: $f_i = \mathbf{X}f_j$. $P(j)$ holds under the assumption of the induction. Then, we infer $\mathcal{F} \vdash \mathbf{G}\phi_1 \Rightarrow \mathbf{X}f_j$ applying successively:
   - the NEX rule: $\mathcal{F} \vdash \mathbf{X}(\mathbf{G}\phi_1 \Rightarrow f_j)$
   - axiom of the (A3) form and the MP rule: $\mathcal{F} \vdash \mathbf{XG}\phi_1 \Rightarrow \mathbf{X}f_j$
   - axiom of the (A4) form, the classical tautology and the MP rule: $\mathcal{F} \vdash \mathbf{G}\phi_1 \Rightarrow \mathbf{XG}\phi_1$
   - axiom of the (A1) form (transitivity of implication) and the MP rule
4. If f_i is derived from the *(MP)* or (*IND*) rule, we proceed by analogy to p. 3.

The proof required showing numerous minor facts (sometimes trivial), among others:

- every axiom is syntactic a consequence of any set of formulas [7, Th. 42]
- for each of the rules of inference: if the premises is a syntactic consequence of a set of formulas, then so is the conclusion [7, Th. 43–45]

- if $\phi_1 \Rightarrow \phi_2$ is a classical propositional logic tautology and ϕ_1 is a syntactic consequence of a set of formulas, then so is ϕ_2. For example, if $\mathcal{F} \vdash p \Rightarrow q \wedge r$, then $\mathcal{F} \vdash p \Rightarrow q$. This property and others of this kind were derived from the MP-rule and the formulas of the form of (A1) [7, Th. 46–52]

The Deduction Theorem of classical propositional logic does not hold generally in *PLTL*.

6. Future Work

To finish formalization of the deductive system of *PLTL,* the completeness theorem must be proved: if $\mathcal{F} \vDash \phi$, then $\mathcal{F} \vdash \phi$. Formalization of deductive systems for other temporal logics (*CTL – Computational Tree Logic, CTL** – a combination of *PLTL* and *CTL*) would be a valuable extension of the *MML* database. Another valuable extension would be Mizar articles on specification of hardware and software. Then, having a deductive system and being able to formalize specification, Mizar could be used as a tool for hardware and software verification by proof-theoretical method, i.e., by proving the theorem expressing the desired property (e.g. mutual exclusion) in the deductive system. The correctness of the proof would be computer checked (by Mizar).

REFERENCES

[1] Mizar Home Page, http://mizar.org.

[2] On the Arithmetic of Boolean Values. Available at http://mizar.uwb.edu.pl/version/current/html/xboolean.html.

[3] Bancerek G., The MML Query Home Page. Available at http://mmlquery.mizar.org/mmlquery/fillin.php?filledfilename=mml-facts.decoded.mqt&argument=number+1.

[4] Bancerek G., The Fundamental Properties of Natural Numbers. *Formalized Mathematics*, 1 (1): 40–46, 1990.

[5] Borak E. and Zalewska A., Mizar Course in Logic and Set Theory. In *Proceedings of Calculemus '07 / MKM '07 Proceedings of the 14th symposium on Towards Mechanized Mathematical*, pages 191–204, Berlin Heidelberg , 2007, Springer-Verlag.

[6] Gabbay D. M., Finger M. and Reynolds M., *Temporal Logic: Mathematical Foundations and Computational Aspects*, volume 1, Oxford University Press, New York, 1994.

[7] Giero M., The Axiomatization of Propositional Linear Time Temporal Logic, Available at http://mizar.uwb.edu.pl/version/current/html/ltlaxio1.html.

[8] Grabowski A., Hilbert Positive Propositional Calculus, *Formalized Mathematics*, 8 (1): 69–72, 1999.

[9] Grabowski A., Korniowicz A. and Naumowicz A., Mizar in a Nutshell, In Asperti A., Harrison J. and Munoz C., editors, *Journal of Formalized Reasoning*, volume 3 of *User Interface I*, pages 153–245, 2010.

[10] Ishida K., Model Checking. Part I, *Formalized Mathematics*, 14 (4): 171–186, 2006.

[11] Ishida K., Model Checking. Part II, *Formalized Mathematics*, 16 (3): 231–245, 2008.

[12] Ishida K. and Shidama Y., Model Checking. Part III, *Formalized Mathematics*, 16 (4): 339–353, 2008.

[13] Kröger F. and Merz S., *Temporal Logic and State Systems*, Springer-Verlag, Berlin Heidelberg, 2008.

[14] Naumowicz A., Teaching How to Write a Proof. In *Proceedings of ETAPS 2008 satellite workshop Formal Methods in Computer Science Education (FORMED2008)*, pages 91–100, Budapest, 2008.

[15] Trybulec A., Some Features of the Mizar Language, In *Procedings of ESPRIT Workshop*, Torino, 1993.

[16] Trybulec A., Defining by Structural Induction in the Positive Propositional Language, *Formalized Mathematics*, 8 (1): 133–137, 1999.

[17] Trzęsicki K., *Logika temporalna. Wybrane zagadnienia*, Wydawnictwo UwB, Białystok 2008.

[18] Trzęsicki K., Temporal Logic Model Checkers as Applied in Computer Science, *Studies in Logic, Grammar and Rhetoric*, 17 (30): 13–125, 2009.

[19] Wiedijk F., Writing a Mizar article in nine easy steps, Available at http://www.cs.ru.nl/~freek/mizar/mizman.pdf.

Mariusz Giero
Institute of Sociology
University of Białystok
giero@uwb.edu.pl