TOWARDS A UNIFIED THEOREM OF SOUNDNESS AND COMPLETENESS FOR RESOLUTION IN DIFFERENT 2-VALUED LOGICS

Cristian Masalagiu and Vasile Alaiba

Abstract. In this paper we analyse three well known logics (propositional, first order predicate and propositional linear temporal logics) using a unified methodology. For each logic, we start by defining the syntax by constructively building a language $\mathcal{L}$, followed by the semantics based on the notion of *structure* $\mathcal{S} : \mathcal{L} \rightarrow \mathbf{B}$, where $\mathbf{B}$ is the support set for a boolean algebra $\mathcal{B} = \langle \mathbf{B}, \bullet, +, ^{-} \rangle$. Next, the resolution method is enunciated by introducing the required normal form for each logic, the definition of step resolution and correctness results. Finally, we take the first steps towards a unified theorem of soundness and completeness that can be introduced and point out that our methodology, and especially the $\mathcal{S}$-notation of semantics, could be useful in building its proof.

1998 *ACM Computing Classification System*: I.2.4, F.4.1.

1. Introduction

The purpose of the paper is to define the notion of structure for several well known 2-valued logics and making the first steps towards a unified soundness and completeness theorem for resolution within these logics.

Throughout the paper, we shall use the term **SAT** for denoting the satisfiability problem for any logic. When thinking about testing the satisfiability of a formulae in a logic, it is well known that syntactic approaches have to be preferred. Resolution is a syntactic method of proof for the decidability or semi-decidability of SAT. However, resolution can only be applied after proving a soundness and completeness theorem: everything provable by resolution is true, and conversely.

Proving such a theorem is significantly different for each logic. For the beginning, we study a unified proof for **PL**, **PL1** and **LTL**. One of the sources of difference between proofs resides from the completely different semantics. For these reasons we believe that a uniform presentation of a logic language using a constructive syntax and structure based semantics can be useful for the development of a generic soundness and completeness proof, and maybe for other syntactic methods of **SAT** solving.

We start by presenting the syntax and semantics of classical logics: the propositional logic (**PL**) and the predicate logic (**PL1**). We will be using the classical definitions for the studied logics, but with variations that will ease the definition of semantics later on. Then we extend the proposed methodology to a temporal 2-valued logic: linear temporal logic (**LTL**). The first step is to introduce a constructive syntax and introduce semantics through a uniform $\mathcal{S}$-notation for each of the different 2-valued logics. Secondly, we analyse the resolution theorem in each of the mentioned logics and, afterwards, an associated soundness and completeness theorem (lemma) is needed.

The rest of the paper is organized as follows. In sections 2, 3 and 4 the syntax, semantics and resolution lemma and theorem are introduced for **PL**, **PL1** and **LTL** respectively. Section 5 extrapolates on the previous sections and introduces the first steps towards a unified soundness and completeness theorem for resolution.

2.Propositional Logic

Syntax

The syntax of a Propositional Logic is based upon an alphabet $\mathcal{A}lf = \mathcal{C} \cup \mathcal{A}t \cup \mathcal{P}$ where:

- $\mathcal{C} = \{\rceil, \vee, \wedge\}$ - a set of logical *connectives*
- $\mathcal{A}t = \{p_1, p_2, p_3, ...\}$ - a nonvoid, at most denumerable set. We call its elements *atoms.*
- $\mathcal{P}a = \{(,)\}$ - a set of parentheses.

Some specific words [3] over the alphabet above are called $\mathbf{PL}_{\mathcal{A}lf}$ formulae. The $\mathbf{PL}_{\mathcal{A}lf}$ syntax is constructively defined as follows (alternatively, we may use a context-free grammar).

We chose the following descriptive manner of the set of formulae because it will help us define the semantics uniformly later on.

Definition 1 *(**Propositional Logic Syntax**) Considering an alphabet $\mathcal{A}lf$ for $\boldsymbol{PL}$, then the corresponding set of formulae, denoted by $\boldsymbol{PL}_{\mathcal{A}lf}$ is:*
***Base**. (elementary formulae):*

- *$\mathcal{A}t \subseteq \boldsymbol{PL}_{\mathcal{A}lf}$ (the atoms are formulae).*

***Constructive Step** (new formulae from existing ones):*

- *If $F \in \boldsymbol{PL}_{\mathcal{A}lf}$ then $(\rceil F) \in \boldsymbol{PL}_{\mathcal{A}lf}$*
- *If $F_1 \in \boldsymbol{PL}_{\mathcal{A}lf}$ and $F_2 \in \boldsymbol{PL}_{\mathcal{A}lf}$ then $(F_1 \vee F_2) \in \boldsymbol{PL}_{\mathcal{A}lf}$*
- *If $F_1 \in \boldsymbol{PL}_{\mathcal{A}lf}$ and $F_2 \in \boldsymbol{PL}_{\mathcal{A}lf}$ then $(F_1 \wedge F_2) \in \boldsymbol{PL}_{\mathcal{A}lf}$*
- *If $F \in \boldsymbol{PL}_{\mathcal{A}lf}$ then $(F) \in \boldsymbol{PL}_{\mathcal{A}lf}$*
- *Nothing else belongs to $\boldsymbol{PL}_{\mathcal{A}lf}$*

Since the alphabet $\mathcal{A}lf$ does not fundamentally impact the syntax or the semantics of propositional logic, from now on we will only use the notation **PL** instead of $\mathbf{PL}_{\mathcal{A}lf}$. We shall not introduce *true* and *false* as special atoms as part of the language. We consider this an unnecessary intervention of the semantics into the syntax of **PL**.

Semantics
The **PL** semantics is built as an extension of an *assignation function* that associates a Boolean value (0 or 1) to any elementary formula. This is done again constructively, according to the definition of the syntax. The Boolean value for any new formula is based on the way this formula is made and on the values of the old formulae.

Definition 2 *(**Boolean algebra**) A **Boolean algebra** is a structure $\mathcal{B} = \langle \boldsymbol{B}, \bullet, +, ^- \rangle$ where $\boldsymbol{B}$ is a nonvoid set, $\bullet, + : \boldsymbol{B} \times \boldsymbol{B} \rightarrow \boldsymbol{B}$ are two binary functions and $^- : \boldsymbol{B} \rightarrow \boldsymbol{B}$ is a unary function. The functions obey the laws: commutativity, associativity, distributivity, absorption, contradiction/tautology [1].*

Remind that in a Boolean algebra **B**, the following laws are also true:

1. $x_1 + x_2 + x_3 + ... + x_n = 1$ if and only if there exists $i \in \mathbf{N}; 0 < i \leq n$ so that $x_i = 1$

2. $x_1 \bullet x_2 \bullet x_3 \bullet ... \bullet x_n = 1$ if and only if for all $i \in \mathbf{N}; 0 < i \leq n$ we have that $x_i = 1$

Definition 3 *(**Assignation Function**) $\mathcal{A}s : \mathcal{A}t \to \boldsymbol{B}$ is defined by:*

$$\mathcal{A}s(p) = \begin{cases} 0, & \textit{if } p \textit{ is false} \\ 1, & \textit{if } p \textit{ is true} \end{cases}$$

Theorem 1 (8) *For every assignation $\mathcal{A}s$ there is a unique extension function $\mathcal{S} : \boldsymbol{PL} \to \boldsymbol{B}$ called structure which satisfies:*

- *$\mathcal{S}(A) = \mathcal{A}s(A)$ for every $A \in \mathcal{A}t$*
- *$\mathcal{S}(\rceil F) = \overline{\mathcal{S}(F)}$, for every $F \in \boldsymbol{PL}$*
- *$\mathcal{S}((F_1 \vee F_2)) = \mathcal{S}(F_1) + \mathcal{S}(F_2)$, for every $F_1, F_2 \in \boldsymbol{PL}$*
- *$\mathcal{S}((F_1 \wedge F_2)) = \mathcal{S}(F_1) \bullet \mathcal{S}(F_2)$, for every $F_1, F_2 \in \boldsymbol{PL}$*

Definition 4 *(**Strong and Weak Equivalence**) Let F_1 and F_1 be **PL** formulae. F_1 is strongly equivalent with F_2, denoted $F_1 \equiv F_2$, if for all structures $\mathcal{S}$, we have $\mathcal{S}(F_1) = \mathcal{S}(F_2)$. F_1 is weakly equivalent with F_2, denoted $F_1 \equiv_w F_2$, if every time there is a structure $\mathcal{S}_1$ such as $\mathcal{S}_1(F_1) = 1$ (F_1 is satisfiable) then a structure $\mathcal{S}_2$ exists such as $\mathcal{S}_2(F_2) = 1$ (F_2 is satisfiable) and conversely.*

We will use the notation $\perp$ for the empty clause, which is not satisfiable in any structure: $\mathcal{S}(\perp) = 0$, for every $\mathcal{S}$.

Resolution

The **SAT** problem can be solved in $O(2^n)$ using brute-force algorithms (that assign all combinations of *true-false* to each formula). The time complexity can be reduced to linear (deterministic) time for certain types of formulae (e.g., Horn formulae) [8].

The known algorithms use as input a formula into **C**onjunctive **N**ormal **F**orm (**CNF**). The transformation of any formula in an equivalent **CNF** could require more complex algorithms [5].

Theorem 2 (8) *For every formula F from **PL** there is a **CNF** formula from **PL** that is strongly equivalent with F .*

Definition 5 ***(Resolvent)*** *Let C_1, C_2 and R be clauses of* ***PL*** *and L a literal. R is a resolvent of C_1 and C_2 if and only if $L \in C_1$, $\overline{L} \in C_2$ and $R = (C_1 \setminus \{L\}) \cup (C_2 \setminus \{\overline{L}\}) = Res_L(C_1, C_2)$.*

The propositional resolution can be applied for two clauses C_1 and C_2 if one of them contains a literal ($L \in C_1$) and the other one contains its complementary literal ($\overline{L} \in C_2$). Both L and $\overline{L}$ will be eliminated and the result (called a resolvent and denoted as R) will be a disjunction of the remaining literals contained in both C_1 and C_2 [5].

Definition 6 ***(Resolvent set)*** *Let $F \in LP$ be a formula (a set of clauses). Then the resolvent set of F is:*

$$Res(F) = F \cup \{Res(C_1, C_2) | C_1, C_2 \in F\}$$

We introduce the following notations:

$$Res^0(F) = F$$

$$Res^1(F) = Res(F)$$

$$Res^n(F) = Res(Res^{n-1}(F)), \text{ for } n > 1$$

$$Res^*(F) = \bigcup_{n \geq 0} Res^n(F)$$

Lemma 1 *Let $F \in$* ***PL*** *a formula in* ***CNF****, $C_1, C_2 \in F$ clauses. If $R = Res(C_1, C_2)$ is a resolvent for C_1 and C_2, then F is strongly equivalent with $F \cup \{R\}$ (for any structure S, $S(F) = S(F \cup \{R\})$).*

Theorem 3 (8) ***(Resolution theorem; soundness and completeness)*** *Let F be a set of clauses from* ***PL****. F is unsatisfiable if and only if $\perp \in Res^*(F)$ (the empty clause belongs to the set of resolvents of F).*

3. Predicate Logic

Syntax

The Predicate Logic (**PL1**) may be viewed as a generalization of **PL**. The concepts of *constant*, *variable*, *functional symbol*, *predicate symbol*, *term* and *quantors* are introduced. To syntactically represent them, a new alphabet is needed.

Definition 7 *The syntax of the predicate logic (**PL1**) is based upon an alphabet* $\mathcal{A}lf = \mathcal{X} \cup (\bigcup_{i=0}^{\infty} \mathcal{P}_i) \cup (\bigcup_{i=0}^{\infty} \mathcal{F}_i) \cup \mathcal{C}_1 \cup \mathcal{C}_2 \cup \mathcal{P}a$*, where:*

- $\mathcal{X} = \{x_1, x_2, ...\}$ *- the set of variables (at most denumerable)*
- $\mathcal{P} = \{\mathcal{P}_0, \mathcal{P}_1, ...\}$ *- the set of classes of predicates (at most denumerable); each* $\mathcal{P}_i$ *is in its turn an at most denumerable set of predicate symbols of arity i, (*$i \in N$*)*
- $\mathcal{F} = \{\mathcal{F}_0, \mathcal{F}_1, ...\}$ *- the set of classes of functions (at most denumerable); same as above, each* $\mathcal{F}_i$ *is in its turn an at most denumerable set of functional symbols of arity i, (*$i \in N$*)*
- $\mathcal{C}_1 = \{\rceil, \vee, \wedge\}$ *- the set of logical connectives*
- $\mathcal{C}_2 = \{(\forall x) | x \in \mathcal{X}\} \cup \{(\exists x) | x \in \mathcal{X}\}$ *- the set of universal and existential quantors (or quantifiers)*
- $\mathcal{P}a = \{(,)\}$ *- the set of parentheses*

Definition 8 *(terms). The set of terms is denoted by* $\mathcal{T}$ *and it is constructively defined as:*
Base

- $\mathcal{X} \subseteq \mathcal{T}$ *and* $\mathcal{F}_0 \subseteq \mathcal{T}$ *(variables and constants are terms).*

Constructive Step

- *If* $n \in N^*$*,* $f \in \mathcal{F}_n$ *and* $t_1, t_2, ..., t_n \in \mathcal{T}$*, then* $f(t_1, t_2, ..., t_n) \in \mathcal{T}$
- *Nothing else is a term*

Semantics

Definition 9 *(**Assignation**) An assignation is a pair* $\mathcal{A}s = \langle \mathcal{U}, \mathcal{I} \rangle$ *in which* $\mathcal{U}$ *is a non-empty set called universe, and* $\mathcal{I}$ *is a function (also called interpretation)*

$$\mathcal{I} : \mathcal{X} \cup \mathcal{P} \cup \mathcal{F} \to \mathcal{U} \cup [\mathcal{U}^* \to \boldsymbol{B}] \cup [\mathcal{U}^* \to \mathcal{U}]$$

which satisfies the conditions:

- *If $x \in \mathcal{X}$, then $\mathcal{I}(x) \in \mathcal{U}$*
- *If $P \in \mathcal{P}_n$, then $\mathcal{I}(P) \in [\mathcal{U}^n \to \boldsymbol{B}]$*
- *If $F \in \mathcal{F}_n$, then $\mathcal{I}(F) \in [\mathcal{U}^n \to \mathcal{U}]$*

In the previous definition, the notation $[A \to B]$ denotes the set of total functions having the domain A and the codomain B and $[A^* \to B]$ is the set of functions from A to B having any number of arguments, including 0 - meaning an element of B. Thus the semantic interpretation of a variable in the assignation $\mathcal{A}s$ is an element from the universe, the interpretation of a predicate having n arguments is a function from ${\mathcal{U}_\mathcal{S}}^n$ to $\{0,1\}$ and the semantics of a functional symbol having n arguments is a function from ${\mathcal{U}_\mathcal{S}}^n$ to $\mathcal{U}_\mathcal{S}$. If there are no confusions, $\mathcal{I}$ will be also denoted by $\mathcal{S}$.

Theorem 4 *For every assignation $\mathcal{A}s = \langle \mathcal{U}, \mathcal{I} \rangle$ there is a unique extension function*

$\mathcal{S} : \mathcal{X} \cup \mathcal{P} \cup \mathcal{F} \cup \mathcal{T} \cup \mathbf{PL1} \to \mathcal{U} \cup [\mathcal{U}^* \to \boldsymbol{B}] \cup [\mathcal{U}^* \to \mathcal{U}] \cup \boldsymbol{B}$ called structure and defined as follows:

$\mathcal{S}(a) = \mathcal{I}(a)$ - for every $a \in \mathcal{X} \cup \mathcal{P} \cup \mathcal{F}$.

Choosing $t \in \mathcal{T}$ means that either t is a constant or variable, or for every $n \in N^*$ and any $t_1, t_2, ..., t_n \in \mathcal{T}$ and any functional symbol $f \in \mathcal{F}_n$, we have $t = f(t_1, t_2, ..., t_n)$. In this case $\mathcal{S}(t) = \mathcal{S}(f)(\mathcal{S}(t_1), \mathcal{S}(t_2), ..., \mathcal{S}(t_n)) \in \mathcal{U}_\mathcal{S}$. We have now finished the process of defining $\mathcal{S}'$ over $\mathcal{X} \cup \mathcal{P} \cup \mathcal{F} \cup \mathcal{T}$.

*We continue by defining $\mathcal{S}$ over **PL1**. This will be done, as usual, in a constructive way.*

***Base** Let $A \in \mathcal{A}t$. In this situation we have either $A = P \in \mathcal{P}_0$ or $A = P(t_1, t_2, ..., t_n), n \in N^*, t_1, t_2, ..., t_n \in \mathcal{T}$. In the first case $\mathcal{S}'$ is already defined ($\mathcal{S}'(P) = \mathcal{S}(P) \in \boldsymbol{B}$). Otherwise we put:*

$\mathcal{S}'(P) = \mathcal{S}(P)(\mathcal{S}'(t_1), \mathcal{S}'(t_2), ..., \mathcal{S}'(t_n)) \in \boldsymbol{B}$

***Constructive step**. The following cases have to be considered*

- *$F = (\rceil F_1)$. Then $\mathcal{S}'(F) = \overline{\mathcal{S}'(F_1)}$*
- *$F = (F_1 \vee F_2)$. Then $\mathcal{S}'(F) = \mathcal{S}'(F_1) + \mathcal{S}'(F_2)$*
- *$F = (F_1 \wedge F_2)$. Then $\mathcal{S}'(F) = \mathcal{S}'(F_1) \bullet \mathcal{S}'(F_2)$*
- *$F = (\forall x)(F_1)$. Then $\mathcal{S}'(F) = 1$ if and only if for each $u \in \mathcal{U}_s$ we have ${\mathcal{S}'}_{[x/u]}(F_1) = 1$ where ${\mathcal{S}'}_{[x/u]}$ is an interpretation which differs from $\mathcal{S}'$, regarding the fact that $\mathcal{S}(x) = u$*

- $F = (\exists x)(F_1)$. *Then* $\mathcal{S}'(F) = 1$ *if and only if there is at least one element* $u \in \mathcal{U}_s$ *such that* $\mathcal{S}'_{[x/u]}(F_1) = 1$

The truth of a formula in **PL1** depends essentially on the way the universe is chosen. Thus the semantics is also built as a function $S : \mathbf{PL1} \rightarrow \mathbf{B}$, based on the syntax of the formula and on the operators of the underlying Boolean Algebra. To check if a formula can be satisfied or not means to test it against the Herbrand Universe [8].

Resolution

For this class of formulae the decidability problem (**SAT**) cannot be solved, so, we cannot decide "for sure" if the formula can be satisfied or not. More precisely the **SAT** problem is undecidable for e.g. predicate logic *with equality* but semidecidable for "pure" **PL1**. That is, there exists an algorithm that having as input a **SNF** formula F, stops with the answer "YES" if F is satisfiable. If F is unsatisfiable, it may run forever.

In the following paragraphs we introduce a couple of definitions and theorems that will allow us to describe the application of the resolution principle for **PL1**.

Definition 10 ***(Skolem Normal Form)*** *A formula* $F \in$ ***PL1*** *is in Skolem normal form if it is in prenex normal form with only universal quantifiers.* F *is of form* $(\forall x_1)...(\forall x_n)F^*$, *where* F^* *is the matrix of the formula* F *and* $\{x_1, x_2, ...x_n\}$ *the set of free variables appearing in* F^*

If F is a formula in Skolem normal form and F^* is in **CNF**, then the formula is said to be in **Clausal Skolem Normal Form (CSNF)**.

Theorem 5 *For every* ***PL1*** *formula there exists an weakly equivalent formula in* ***CSNF***.

Definition 11 ***(Unifiable set)*** *A finite set of literals* $L = \{L_1, L_2, ...L_k\}$ *is unifiable if a substitution sub exists such that* $|L_{sub}| = 1$.

If such a substitution exists it is called a *unifier* of the set of literals L. A unifier *sub* is called a *most general unifier* if for every unifier sub' there is a substitution s such that $sub' = sub \cdot s$.

Theorem 6 ***(Unification*** *[7]) Every unifiable set of literals has a most general unifier.*

Definition 12 ***(Resolvent)*** *Let $C_1, C_2, R \in$ **PL1** be clauses. Then R is a resolvent of C_1 and C_2 if:*

- *there exists two variable renaming substitutions s_1 and s_2 such that C_1s_1 and C_2s_2 have no common variables*
- *there is a set of literals $L_1, L_2, ..., L_m \in C_1s_1$ and $L'_1, L'_2, ..., L'_n \in C_2s_2$, $m, n \in N^*$, such that the set $L = \{\overline{L_1}, \overline{L_2}, ..., \overline{L_m}, L'_1, L'_2, ..., L'_n\}$ is unifiable with sub a most general unifier.*
- *$R = ((C_1s_1 \setminus \{L_1, L_2, ..., L_m\}) \cup (C_1s_1 \setminus \{L'_1, L'_2, ..., L'_n\}))sub$*

Lemma 2 *Let $F \in$ **PL1** a formula in **CSNF**, $C_1, C_2 \in F$ clauses. If $R = Res(C_1, C_2)$ is a resolvent for C_1 and C_2, then F is strongly equivalent with $F \cup \{R\}$ (for any structure S, $S(F) = S(F \cup \{R\})$).*

Theorem 7 ***(Resolution theorem; soundness and completeness)****. Let $F \in$ **LP1** be a set of clauses (a formula in **CSNF**). F is unsatisfiable if and only if $\perp \in Res^*(F)$ (the empty clause belongs to the resolvent set of F).*

4. Linear Temporal Logic

Linear **T**emporal **L**ogic (**LTL**) is a *modal logic* in which modalities refer to time. In **LTL** the *future* is seen as a sequence of *states* that defines a *path* [9].

Generally, temporal logics are used in system verification, properties of the system being expressed as formulae that can be *true* or *false* in different moments of the system's life. Different properties of the system such as *safety* (something "bad" will never happen) or *liveness* (something "good" will always happen - even if we do not know when) are such kind of system characteristics that can be verified using **LTL**. As usual, because semantic algorithms are rather difficult to implement on such issues, we have to redirect our attention on the syntactic approach, following the scheme already suggested.

The syntax for **LTL** will be defined in the same way as that for **PL** or **PL1**.

Syntax

The alphabet of **LTL** is $\mathcal{A}lf = \mathcal{A} \cup \overline{\mathcal{A}} \cup \mathcal{P}a \cup \mathcal{C}_1 \cup \mathcal{C}_2$, where:

- $\mathcal{A} = \{p_1, p_2, p_3, p_4, ...\}$ and $\overline{\mathcal{A}} = \{\rceil p_1, \rceil p_2, \rceil p_3, ..., \}$ - the sets of propositions/atoms (positive and negative)

- $\mathcal{P}a = \{(,)\}$ - the set of parentheses
- $\mathcal{C}_1 = \{\rceil, \vee, \wedge\}$ - the set of logical connectives
- $\mathcal{C}_2 = \{\circ, \Box, \Diamond, U, \tilde{U}\}$ - the set of *temporal operators/connectives* for future

We put $SP = \{\{p_1, \rceil p_1\}, \{p_2, \rceil p_2\}, \{p_3, \rceil p_3\}, ..., \{p_n, \rceil p_n\}...\}$ - a set of sets, each of them containing a positive and a negative proposition, each positive proposition being "the contrary" of the negative one in the same set.

In a set that is formed by a proposition and the negative form of the same proposition, it is not important if the proposition in the positive form is *true* or not. The idea is that every element in the SP set contains for sure a *true* proposition and a *false* one.

Definition 13 *The set of atoms is then defined as:* $\mathcal{A}t = \{true, false\} \cup BA$, *where* $BA = \bigcup_{i \in N} \{x_i | x_i \in \{p_i, \rceil p_i\}\}$.

In this set it is impossible to have at the same time a proposition $p \in \mathcal{A}$ *and the same proposition in its negative form (or vice versa).*

Definition 14 *(**LTL** syntax)*

Base

- *If* $p \in \mathcal{A}t$ *then* $p \in \textbf{LTL}$

Constructive step

- *If* $F_1 \in \textbf{LTL}$ *and* $F_2 \in \textbf{LTL}$, *then* $(F_1 \vee F_2) \in \textbf{LTL}$
- *If* $F_1 \in \textbf{LTL}$ *and* $F_2 \in \textbf{LTL}$, *then* $(F_1 \wedge F_2) \in \textbf{LTL}$
- *If* $F \in \textbf{LTL}$, *then* $(\circ F) \in \textbf{LTL}$
- *If* $F \in \textbf{LTL}$, *then* $(\Box F) \in \textbf{LTL}$
- *If* $F \in \textbf{LTL}$, *then* $(\Diamond F) \in \textbf{LTL}$
- *If* $F_1, F_2 \in \textbf{LTL}$, *then* $(F_1 U F_2) \in \textbf{LTL}$
- *If* $F_1, F_2 \in \textbf{LTL}$, *then* $(F_1 \tilde{U} F_2) \in \textbf{LTL}$
- *If* $F \in \textbf{LTL}$, *then* $(F) \in \textbf{LTL}$

- *Nothing else belongs to **LTL***

At a semantic level, some of the temporal connectives can be written with the help of the others. That is why the language can be reduced by eliminating the U and the $\tilde{U}$ operators. U and $\tilde{U}$ can be written using the $\Box$ and $\Diamond$ connectors, but this is not a simple task (it implies *recursion and fixed points*). Many authors use U and $\tilde{U}$ because $\Box$ F and $\Diamond$ F can be expressed directly with the help of the others. Namely:

- $\Diamond F = (\mathit{true}\ UF)$
- $\Box F = (\mathit{false}\ \tilde{U}F)$
- $\Diamond F = \rceil(\Box(\rceil F))$
- $F_1UF_2 = \Diamond F_2 \wedge ((F_1UF_2) \vee \Box F_1)$
- $F_1\tilde{U}F_2 = (F_2U(F_1 \wedge F_2) \vee (\Box F_2)$
- $F_1\tilde{U}F_2 = \rceil(\rceil F_1U\rceil F_2)$

Semantics Based on Paths

The classical semantics for **LTL** formulae is based on *paths* (a path is an ordered set of states, also known as a *run*). Every state of the path contains a set of atoms that are *true* in that state. We can consider a function $\pi : \mathbf{N} \rightarrow 2^{\mathcal{A}t}$ that, applied to an integer value i returns the set of atoms that are true in the i^{th} state of the path. Thus, we may consider, $\pi^i : \mathbf{N} \rightarrow 2^{\mathcal{A}t}$, $\pi^i(j) = \pi(i+j), \forall j \in \mathbf{N}$. In particular we have $\pi^1(j) = \pi(1+j), \forall j \in N$.

Definition 15 ***(The classical semantics of LTL) Base***

- $\pi \models \mathit{true}$ *and* $\pi \not\models \mathit{false}$
- $\pi \models p$ *if and only if* $p \in \pi(0)$, *and* $\pi \models \rceil p$ *if and only if* $p \notin \pi(0)$ *(for every* $p \in \mathcal{A}$*)*

Constructive step

- $\pi \models (F_1 \wedge F_2)$ *if and only if* $\pi \models F_1$ *and* $\pi \models F_2$

- $\pi \models (F_1 \vee F_2)$ *if and only if* $\pi \models F_1$ *or* $\pi \models F_2$
- $\pi \models (\circ F)$ *if and only if* $\pi^1 \models F$
- $\pi \models (\Diamond F_1)$ *if and only if there is* $j > 0$ *such that* $\pi^j \models F_1$
- $\pi \models (\Box F_1)$ *if and only if for every* $j > 0$ *we have* $\pi^j \models F_1$
- $\pi \models (F)$ *if and only if* $\pi \models F$
- $\pi \models (F_1 U F_2)$ *if and only if there exists* $j \in \boldsymbol{N}$ *such that* $\pi^j \models F_2$ *and for each* $i \in \boldsymbol{N}, i < j$ *we have that* $\pi^i \models F_1$
- $\pi \models (F_1 \tilde{U} F_2)$ *if and only if there exists* $j \in \boldsymbol{N}$ *such that* $\pi^j \models \rceil F_2$ *and for that* j*, there exists* $i \in \boldsymbol{N}, i < j$ *such that* $\pi^i \models F_1$

Each of the above laws has to be considered for any path π.

It can be concluded that this operational semantics is not defined in the same manner as the one for **PL** or **PL1**. In the previous case we have spoken about a function from the set of formulae into **B**, also named a structure. Let us call it the $\mathcal{S}$ **- notation**.

Semantics Using the $\mathcal{S}$ - notation

We start again by an initial Boolean interpretation of atoms.

Definition 16 ***(Assignation)*** *An assignation is a pair* $\mathcal{A}s = \langle \pi, \mathcal{I} \rangle$, *where* $\pi : \mathrm{N} \to 2^{\mathcal{A}t}$ *and* $\mathcal{I} : N \times \mathcal{A}t \to \boldsymbol{B}$ *defined by* $\mathcal{I}(i,p) = 1$ *if* $p \in \pi(i)$ *and* 0 *otherwise.*

For an assignation $\mathcal{A}s = \langle \pi, \mathcal{I} \rangle$, we will use the notation $\mathcal{A}s^k$ to denote a new assignation starting at state k in the path π: $\mathcal{A}s^k = \langle \pi^k, \mathcal{I}' \rangle$, where $\mathcal{I}'(i,p) = \mathcal{I}(i+k,p)$.

Below, $\sum$ and $\prod$ refer to the *Boolean sum* and the *Boolean product*, respectively.

Theorem 8 *For each assignation* $\mathcal{A}s = \langle \pi, \mathcal{I} \rangle$ *there exists a unique extension function* $\mathcal{S} : \boldsymbol{LTL} \to \boldsymbol{B}$ *called structure and defined as follows:*

Base

- *If* $F \in \mathcal{A}t$ *then* $\mathcal{S}(F) = \mathcal{I}(0, F)$

In the following we will use the notation $\mathcal{S}_k$ as the extension of the assignation $\mathcal{A}s^k$.

Constructive step

- *If* $F = (F_1 \vee F_2)$ *then* $\mathcal{S}(F) = \mathcal{S}(F_1 \vee F_2) = \mathcal{S}(F_1) + \mathcal{S}(F_2)$
- *If* $F = (F_1 \wedge F_2)$ *then* $\mathcal{S}(F) = \mathcal{S}(F_1 \wedge F_2) = \mathcal{S}(F_1) \bullet \mathcal{S}(F_2)$
- *If* $F = (\circ F_1)$ *then* $\mathcal{S}(F) = \mathcal{S}_1(F_1)$
- *If* $F = (\Diamond F_1)$ *then*
$$\mathcal{S}(F) = \sum_{k \in \mathrm{N}} \mathcal{S}_k(F_1)$$
- *If* $F = (\Box F_1)$ *then*
$$\mathcal{S}(F) = \prod_{k \in \mathrm{N}} \mathcal{S}_k(F_1)$$
- *If* $F = (F_1 U F_2)$ *then* $\mathcal{S}(F) = \mathcal{S}(F_1 U F_2) = 1$ *if and only if there exists* $u \in \boldsymbol{N}$ *such that* $\mathcal{S}_u(F_2) \bullet \mathcal{S}_{u-1}(F_1) \bullet \mathcal{S}_{u-2}(F_1) \bullet ... \bullet \mathcal{S}_1(F_1) = 1$
- *If* $F = (F_1 \tilde{U} F_2)$ *then* $\mathcal{S}(F) = \mathcal{S}(F_1 \tilde{U} F_2) = 1$ *if and only if there exists* $u \in \boldsymbol{N}$ *such that we have* $\overline{(\mathcal{S}_u(F_2))} = 1$ *and for that u, there is a* $k \in \boldsymbol{N}, k < u$ *and* $\mathcal{S}_k(F_1) = 1$

We have to prove:

- *If* $F = (F_1 \vee F_2)$ *then* $\mathcal{S}(F) = \mathcal{S}(F_1 \vee F_2) = \mathcal{S}(F_1) + \mathcal{S}(F_2)$.
$\mathcal{S}(F) = \mathcal{S}(F_1 \vee F_2)$ *means that* $F_1 \vee F_2 \in \pi$ *and that means that either* F_1 *or* F_2 *belongs to* π: $F_1 \in \pi$ *or* $F_2 \in \pi$. *We will consider two cases: If* $F_1 \in \pi$ *means that* $\mathcal{S}(F_1) = 1 = \mathcal{S}(F_1) + a$, $\forall a \in \boldsymbol{B}$; *Considering* $a = \mathcal{S}(F_2)$ *(that belongs to* $\boldsymbol{B}$*) the result is obvious.*
The second case $F_2 \in \pi$ *can be treated in the same manner resulting that* $\mathcal{S}(F_2) = 1 = \mathcal{S}(F_1) + \mathcal{S}(F_2)$.
- *If* $F = (F_1 \wedge F_2)$ *then* $\mathcal{S}(F) = \mathcal{S}(F_1 \wedge F_2) = \mathcal{S}(F_1) \bullet \mathcal{S}(F_2)$ *is somehow similar with the first case.*
$\mathcal{S}(F) = \mathcal{S}(F_1 \wedge F_2)$ *means that* $F_1 \wedge F_2 \in \pi$ *and that means that both* F_1 *and* F_2 *belong to* π: $F_1 \in \pi$ *and* $F_2 \in \pi$. $\mathcal{S}(F_1) = 1$ *and* $\mathcal{S}(F_2) = 1$ *meaning that* $\mathcal{S}(F_1) \bullet \mathcal{S}(F_2) = 1 \bullet 1 = 1$.

- *If* $F = (\circ F_1)$ *then* $\mathcal{S}(F) = \mathcal{S}_1(F_1)$ $F \in \pi \Leftrightarrow (\circ F_1) \in \pi$ *and according to* π*, the formula* $\circ F_1$ *is true in the state* π *iff* F_1 *is true in the next state:* $\pi(i+1)$ *which is also denoted by* π^1 *(* $F_1 \in \pi^1$*). We can write that as:* $\mathcal{S}(F) = \mathcal{S}(\circ F_1) = \mathcal{S}_1(F_1)$.

- *If* $F = \Diamond F_1$ *then*
$$\mathcal{S}(F) = \sum_{k \in \mathrm{N}} \mathcal{S}_k(F_1)$$
If $F = (\Diamond F_1)$ *then* $\mathcal{S}(F) = 1$ *if and only if there is at least one state* j *and* $F_1 \in \pi^j$*. If* $F_1 \in \pi^j$ *that means that* $\mathcal{S}_j(F_1) = 1$ *and because* $\mathcal{S}(F)$ *is defined as a sum of all next states, also it includes the state* π^j *in which* $\mathcal{S}_j(F_1) = 1$*. The sum of any number of boolean variables is 1 if there is at least one variable with the value 1. That means that* $\mathcal{S}(F) = 1$ *if and only if*
$$\sum_{k \in \mathrm{N}} \mathcal{S}_k(F_1) = 1$$

- *If* $F = \Box F_1$ *then*
$$\mathcal{S}(F) = \prod_{k \in \mathrm{N}} \mathcal{S}_k(F_1)$$
If $F = (\Box F_1)$ *then* $\mathcal{S}(F) = 0$ *if and only if there is at least one state* j *and* $F_1 \notin \pi^j$*. If* $F_1 \notin \pi^j$ *that means that* $\mathcal{S}_j(F_1) = 0$ *and because* $\mathcal{S}(F)$ *is defined as a product of all next states, it also includes the state* π^j *in which* $\mathcal{S}_j(F_1) = 0$*. The product of any number of boolean variables is 0 if there is at least one variable with the value 0. That means that* $\mathcal{S}(F) = 0$ *if and only if*
$$\sum_{k \in \mathrm{N}} \mathcal{S}_k(F_1) = 0$$

Otherwise, if the sum is 0, it means that there is no j *state (where* j *is bigger than* 0*) that satisfies the formula* F_1 *(*$\nexists j, j > 0$ *so* $\mathcal{S}_j(F_1) = 1$*), which means that* $(\Diamond F) \notin \pi \Leftrightarrow \pi \nvDash F$.

*We have got again a set of formulae (**LTL**) and a semantic definition of truth, based on structures.*

To prove the uniqueness of $\mathcal{S}$*, suppose that there exists another homomorphic extension of* $\mathcal{S}$*, denoted by* $\mathcal{S}'$*. Knowing that all the objects implied,* $\mathcal{S}$*,* $\mathcal{S}'$*, are functions in the mathematical sense, the fact that* $\mathcal{S} = \mathcal{S}'$ *is immediate proved by constructive induction.*

Resolution

Temporal resolution is fairly complex compared to the PL and PL1 equivalent methods. From the existing approaches [2, 4, 6] we chose to follow Fisher's description of *clausal temporal resolution* [2].

In order to introduce the normal form necessary for the temporal resolution, we need to introduce a special symbol *start*, which has the special property that it holds only at the beginning of time, in any structure: $\mathcal{S}(start) = 1$ and $\mathcal{S}_k(start) = 0$, for every $k \in N^*$.

Definition 17 ***(Separated Normal Form)*** *A formula $F \in LTL$ is in separated normal form (**SNF**) if it is of the form*

$$F = \Box \bigwedge_i C_i$$

*where each C_i is a **LTL** clause of one of the following forms:*

$$start \Rightarrow \bigvee_a L_a$$

$$\bigwedge_b L_b \Rightarrow \circ \bigvee_c L_c$$

$$\bigwedge_d L_d \Rightarrow \Diamond L$$

and L_a, L_b, L_c, L_d and L are literals.

Definition 18 ***(Resolvent)*** *Let $C_1, C_2, R \in$ **LTL** be clauses. Then $R = Res(C_1, C_2)$ is a resolvent of C_1 and C_2 if it can be derived using any of the resolution rules (step resolution, temporal resolution or augmentation).*

For a full description of the resolution rules mentioned in the definition of a resolvent the reader is advised to consult [2].

Lemma 3 *Let $F \in$ **LTL** a formula in **SNF**, $C_1, C_2 \in F$ clauses. If $R = Res(C_1, C_2)$ is a resolvent for C_1 and C_2, then F is strongly equivalent with $F \cup \{R\}$ (for any structure S, $S(F) = S(F \cup \{R\}))$.*

Theorem 9 ***(Resolution theorem; soundness and completeness)****. Let $F \in$ **LTL** be a set of clauses (a formula in **SNF**). F is unsatisfiable if and only if $\bot \in Res^*(F)$ (the empty clause belongs to the resolvent set of F).*

5. Towards a Unified Soundness and Completeness Theorem for Resolution

Up to this point we analysed three logics, **PL**, **PL1** and **LTL**, using the same methodology. For each logic, we introduced a constructive syntax, thus formally describing the notion of formula. Following the syntax, a definition of semantics based on the concept of structure, or $\mathcal{S}$-notation, is defined. The semantics are not fundamentally different from the classic definitions, as the purpose of the paper is to introduce a unified methodology for a set of logics, rather than adding specific results for a single logic. The notion of resolution is examined in all logics as a means of deriving new clauses from existing ones.

We can now draw from the previous sections and point toward a set of generic results. For the following paragraphs, let $\mathcal{L}$ be a logic defined using our methodology (for now one of PL, PL1 and LTL).

Lemma 4 *Let $F \in \mathcal{L}$ a formula in clausal normal form, $C_1, C_2 \in F$ clauses. If $R = Res(C_1, C_2)$ is a resolvent for C_1 and C_2, then F is strongly equivalent with $F \cup \{R\}$ (for any structure S, $S(F) = S(F \cup \{R\})$).*

By the phrase *clausal normal form* we understand the corresponding normal form in each of the languages.

Theorem 10 ***(Resolution theorem; soundness and completeness).*** *Let $F \in \mathcal{L}$ be a set of clauses (a formula in clausal normal form). F is unsatisfiable if and only if $\perp \in Res^*(F)$ (the empty clause belongs to the resolvent set of F).*

The proof of the theorem in different logics is based upon a lemma similar to the one enunciated above and some proofs depend on others, like for example in PL1 proving the resolution theorem requires results regarding the Herbrand extension.

6. Conclusions and Future Research

Although there are still a few steps to make to achieve a unified soundness and completeness theorem for resolution that can be applied in many different kinds of 2-valued logics, we believe that the work done so far is interesting from at least two points of view. First, it provides a well defined methodology for analysis of a 2-valued logic, based on constructive definitions and the

$\mathcal{S}$-notation. Second, it shows that although there are significant differences between logics, some results can be enunciated in a similar manner.

Our main idea is to suggest a uniform framework for treating the syntax, semantics, and satisfiability of logical languages. The main step is to have a constructive definition for semantics, starting from a corresponding definition of the syntax, i.e. a structure $\mathcal{S} : \mathcal{L} \to \mathbf{B}$. Then, knowing that $\mathcal{S}$ is recursive (or, at least, recursively denumerable), the idea is to derive a syntactic method to compute it and a soundness and completeness theorem related to a notion of truth, materialized as a boolean algebra.

We plan to apply the same methodology for other logics like computation tree logic, temporal logic of actions and some logics of belief. The notion of structure could be applied to other methods requiring a connection between syntax and semantics. A first step is considering the field of automated theorem proving in general, not limited to resolution. It would be interesting to study in the same uniform framework other types of semantics for 2-valued logics - for instance algebraic. In the future, we might create a universal checker for testing satisfiability of a formula in a certain logic, starting from a syntax and a semantics as described in this paper.

References

[1] P. Dwinger. Introduction to Boolean algebras. Wrzburg: Physica Verlag, 1971.

[2] M. Fisher, C. Dixon, and M. Peim. Clausal temporal resolution. ACM Trans. Comput. Logic, 2:1256, January 2001.

[3] J.E. Hopcroft and J.D. Ullman. Introduction to Automata Theory, Languagesand Computation. Addison-Wesley Publishing, 1979.

[4] U. Hustadt, B. Konev, and R. A. Schmidt. Deciding monodic fragments by temporal resolution. In R. Nieuwenhuis, editor, Proceedings of the 20th International Conference on Automated Deductio n (CADE-20), volume 3632 of Lecture Notes in Artificial Intelligence, pages 204218. Springer, 2005.

[5] M. Huth and M. Ryan. Logic in Computer Science - modelling and reasoning about systems. Cambridge University Press, 2004.

[6] B. Konev, A. Degtyarev, C. Dixon, M. Fisher, and U. Hustadt. Mechanising first-order temporal resolution. Inf. Comput., 199(1-2):5586, 2005.

[7] J.A. Robinson. A Machine-oriented Logic Based on the Resolution Principle. Journal of the Association for Computing Machinery, 1965.

[8] U. Schoening. Logic for Computer Scientists. Birkhaeuser Berlin, Germany, 1989.

[9] P. Wolper. The Algorithmic Verification of Reactive Systems (course notes). Universite de Liege, 2000.

Cristian Masalagiu
Faculty of Computer Science
Alexandru Ioan Cuza University
General Berthelot 16, Iasi 700483, Romania
email:*mcristy@info.uaic.ro*

Vasile Alaiba
Faculty of Computer Science
Alexandru Ioan Cuza University
General Berthelot 16, Iasi 700483, Romania
email:*alaiba@info.uaic.ro*