On the Relationship between ω-Automata and Temporal Logic Normal Forms *

Alexander Bolotov[†], Michael Fisher[‡] and Clare Dixon[‡]
[†] Harrow School of Computer Science,
University of Westminster, Harrow HA1 3TP, U.K.
A.Bolotov@westminster.ac.uk
[‡] Department of Computer Science,
University of Liverpool, Liverpool L69 7ZF, UK.
{M.Fisher, C.Dixon}@csc.liv.ac.uk

keywords: temporal logic, normal forms, automata, theorem-proving, clausal resolution, automata.

Abstract

We consider the relationship between ω-automata and a specific logical formulation based on a normal form for temporal logic formulae. While this normal form was developed for use with execution and clausal resolution in temporal logics, we here show how it can represent, syntactically, ω-automata in a high-level way. Technical proofs of the correctness of this representation are given.

1 Introduction

Automata over infinite objects, termed ω-automata, were originally introduced as a tool for investigating the decidability of restricted classical first-order and second-order logics [3], but have also been extensively used in recent developments within wider areas of computer science. In particular, the success of the model checking approach [6, 12, 15, 20], when applied to the analysis of infinite computations, is primarily due to the incorporation of automata related methods.

On the other hand, in line with their original aim, ω-automata have been used to establish theoretical properties of a variety of logics used within computer science. In particular, automata-theoretic techniques give elegant and efficient methods for proving decidability of varieties of temporal logic [6]. Note that, in some cases, for example in the case of the branching-time logic CTL*, where the tableau construction is not directly applicable, automata-based methods are essential for proving decidability of such expressive systems [6]. In decidability results for temporal logic, Büchi automata [3] over infinite words and infinite trees are especially important.

*This work was partially supported by funding from EPSRC, under research grant GR/L87491

It is known [5] that Büchi word automata, which are as expressive as propositional linear-time μ-calculus [16], are themselves strictly more expressive than propositional linear-time temporal logic, PLTL [22]. A well-known example exhibiting this is the characterization of the property "p holds at every even moment of (a linear, discrete sequence of) time" (i.e. '$Even(p)$'), which is expressible using Büchi automata, but not PLTL [22].

In this paper we consider the relationship between ω-automata on infinite words and a specific logical formalism based on a normal form for temporal logic formulae, called $\mathrm{SNF}_{\mathrm{PLTL}}$ [8]. While this normal form was developed for use with clausal resolution in temporal logics [7, 10] and execution [8], we here show how it can represent, *syntactically*, ω-automata in a high-level way.

The general problem structure that we are trying to solve is given in Figure 1. Here we are attempting to analyze a system specification (S) and provide some (formal) verification of its properties (V).

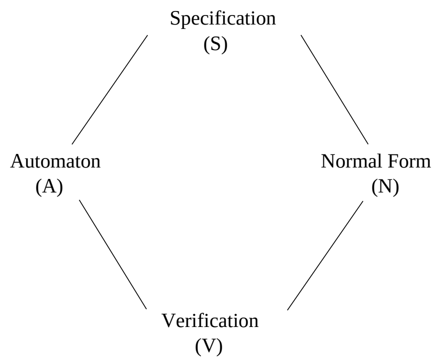

Figure 1: Specification-Verification Problem

If the specification (S) is given in a high-level language, for example a logic, then we can either translate (S) to an automaton (A) and then carry out as verification (V) an automaton emptiness check, or we can carry out the translation from (S) to a Normal Form (N) and then use as verification (V) some form of efficient deduction, for example clausal resolution. The complexity of the translation from (S) to an automaton (A) is usually exponential [21]. For example, a PLTL formula φ can be translated into a Büchi automaton with $2^{O(|\varphi|)}$ states [21]. In contrast, decidable in linear time [6, 21]. Note, however, that the situation is different when we carry out the translation of a PLTL formula φ into alternating automata [21, 1], where the translation (S)$\rightarrow$(A) results in an automaton of the size $O(|\varphi|)$, while checking non-emptiness involves an exponential blow-up.[1]

On the other hand, the complexity of the translation from (S) to (N) is polynomial or often linear [10]. In contrast, verification of formulae in the Normal Form ((N) $\rightarrow$ (V)) is usually exponential since it involves some form of proof (in our case, clausal

[1] No direct methods of checking non-emptiness of alternating automata are known. Usually an alternating automaton is simulated by a standard non-deterministic automaton and the known emptiness check is applied to the latter [1, 21].

resolution).[2]

Our particular concern here is what is the relationship between the Normal Form (N) given as the normal form for PLTL ($\mathrm{SNF_{PLTL}}$) and the Automaton (A) in the diagram above. We will show that $\mathrm{SNF_{PLTL}}$, which was originally developed as a normal form for the formulae of PLTL, can represent Büchi automata. The essential reason for this is that, in translating a problem specification into our normal form, we actually derive clauses within a fragment of quantified propositional linear-time temporal logic (QPLTL) [18, 14]. In particular, formulae within $\mathrm{SNF_{PLTL}}$ are existentially quantified. In order to utilise the normal form as part of a proof, we effectively skolemize the normal form producing temporal formulae without any quantification (i.e. PLTL).

Having established this relationship between $\mathrm{SNF_{PLTL}}$ and Büchi automata, we will be able to represent a problem specification directly as a set of formulae in the Normal Form and apply a resolution based verification technique to the latter. Also, taking into account that nondeterministic Büchi automata on infinite words recognize the same class of w-languages as nondeterministic Rabin or Streett word automata [20], we believe that varying the formulation of acceptance conditions in our syntactic representation of Büchi automata, our approach allows us to specify other types of ω-automata such as Rabin or Streett automata.

The remainder of the paper is organized as follows. In §2 we overview $\mathrm{SNF_{PLTL}}$ outlining its syntax in §2.1 and its semantics, in §2.2, together with an example, §2.3. In §3 we describe the representation of Büchi automata in QPLTL, defining this type of automata in §3.1, overviewing the logic QPLTL in §3.2, and the representation of Büchi automata in QPLTL in §3.3 following [14], which inspired the work presented in this paper. Consequently, in §4, we show how to obtain similar results using $\mathrm{SNF_{PLTL}}$ language. In §4.1 we describe the translation of Büchi automata into $\mathrm{SNF_{PLTL}}$ and in §4.2 establish its correctness, while in §4.3 we apply $\mathrm{SNF_{PLTL}}$ as a specification language to express the property '$Even(p)$'. Finally, in §5, we provide concluding remarks.

2 Normal form in clausal resolution for PLTL

The clausal resolution method for propositional linear-time temporal logic, PLTL, described in [7, 10], is divided into two main procedures. Firstly, a PLTL formula F is translated into its normal form, called *Separated Normal Form*, and abbreviated as $\mathrm{SNF_{PLTL}}(F)$. Then a resolution procedure is applied to $\mathrm{SNF_{PLTL}}(F)$.

The basic idea behind the normal form is to present information contained in the original PLTL formula, A, generating formulae that either relevant to the first state in a model, or constrain future behaviour of A. The transformation procedure uses fixpoint unwinding [2] and sub-formula renaming [17] in order to obtain an $\mathrm{SNF_{PLTL}}$ representation for an arbitrary PLTL formula.

2.1 Language of $\mathrm{SNF_{PLTL}}$

We define the language of $\mathrm{SNF_{PLTL}}$ based on the following symbols:

[2] However, we are often able to use either improved proof strategies [9] or restricted forms of the normal form [4] in order to improve the practical efficiency of such proof.

- atomic propositions $p, q, r, \ldots, p_1, q_1, r_1, \ldots, p_n, q_n, r_n, \ldots$;
- classical operators $\neg, \wedge, \Rightarrow, \vee$;
- classically defined constants **true** , **false** ;
- temporal operators
  - 'at the initial moment of time' **start** ;
  - 'always in the future' $\square$;
  - 'sometime in the future' $\diamondsuit$;
  - 'at the next moment in time' $\bigcirc$.

Definition 1 (Separated Normal Form) Separated normal form is a formula

$$\square \bigwedge_{i=1}^{n} A_i$$

where each of the "A_i" is known as a '*clause*' and must be one of the following (see Figure 2) with each particular α_i, β_j, γ_k, δ_m and l representing literals, i.e. propositions or their negations.

$$\begin{array}{lcll}
\mathbf{start} & \Rightarrow & \bigvee_i \alpha_i & \text{(an } \textit{initial} \text{ clause)} \\
\bigwedge_j \beta_j & \Rightarrow & \bigcirc \bigvee_k \gamma_k & \text{(a } \textit{step} \text{ clause)} \\
\bigwedge_m \delta_m & \Rightarrow & \diamondsuit l & \text{(a } \textit{sometime} \text{ clause)}
\end{array}$$

Figure 2: Varieties of $\mathrm{SNF}_{\mathrm{PLTL}}$ Clauses

The intuition behind this formulation is that the initial clauses provide the initial conditions while each step or sometime clause represents a constraint upon the future behaviour of the formula, given the current conjunction of literals. Thus, the step clauses constrain successor states and the sometime clauses constrain sequences of states over a longer period.

The semantics of $\mathrm{SNF}_{\mathrm{PLTL}}$ is given in the next section. However, to illustrate Definition 1 we exhibit a simple example of $\mathrm{SNF}_{\mathrm{PLTL}}$ clauses explaining its intuitive interpretation. Consider the following $\mathrm{SNF}_{\mathrm{PLTL}}$ expression:

$$\square((\mathbf{start} \Rightarrow x) \wedge (x \Rightarrow \diamondsuit z) \wedge (\mathbf{true} \Rightarrow \bigcirc \neg z)) \tag{1}$$

Formula 1 expresses the following properties of the linear-time model:

- x is satisfied in the initial state (given by the first conjunct);
- once x is satisfied, z must occur eventually (given by the second conjunct); and
- $\neg z$ occurs everywhere apart from possibly in the initial state in a model (given by the third conjunct).

$\langle \mathcal{M}, t_i \rangle$	$\models p$	iff	$I(t_i, p) = true,$ for atomic proposition p
$\langle \mathcal{M}, t_i \rangle$	$\models \neg A$	iff	$\langle \mathcal{M}, t_i \rangle \not\models A$
$\langle \mathcal{M}, t_i \rangle$	$\models A \wedge B$	iff	$\langle \mathcal{M}, t_i \rangle \models A$ and $\langle \mathcal{M}, t_i \rangle \models B$
$\langle \mathcal{M}, t_i \rangle$	$\models A \vee B$	iff	$\langle \mathcal{M}, t_i \rangle \models A$ or $\langle \mathcal{M}, t_i \rangle \models B$
$\langle \mathcal{M}, t_i \rangle$	$\models A \Rightarrow B$	iff	$\langle \mathcal{M}, t_i \rangle \not\models A$ or $\langle \mathcal{M}, t_i \rangle \models B$
$\langle \mathcal{M}, t_i \rangle$	$\models$ **start**	iff	$i = 0$
$\langle \mathcal{M}, t_i \rangle$	$\models \Box B$	iff	for each j, if $i \leq j$ then $\langle \mathcal{M}, t_j \rangle \models B$
$\langle \mathcal{M}, t_i \rangle$	$\models \Diamond B$	iff	there exists j such that $i \leq j$ and $\langle \mathcal{M}, t_j \rangle \models B$
$\langle \mathcal{M}, t_i \rangle$	$\models \bigcirc B$	iff	$\langle \mathcal{M}, t_{i+1} \rangle \models B$

Figure 3: Semantics of $\mathrm{SNF}_{\mathrm{PLTL}}$

Note that although the language of PLTL includes '$\mathcal{U}$' and '$\mathcal{W}$' operators as well as the operators used in $\mathrm{SNF}_{\mathrm{PLTL}}$, in the latter language we use only '$\Box$', '$\Diamond$', and '$\bigcirc$' operators. This is possible since in the transformation procedure the '$\Box$'[3], '$\mathcal{U}$' and '$\mathcal{W}$' operators are replaced by their fixpoint definitions. We will provide the corresponding rules in section §2.3, together with other main transformation rules.

2.2 Interpretation of $\mathrm{SNF}_{\mathrm{PLTL}}$

Models of a set of $\mathrm{SNF}_{\mathrm{PLTL}}$ clauses are tuples $\mathcal{M} = \langle T, \leq, I \rangle$ where

- T is a discrete, linear sequence of states, isomorphic to Natural Numbers, $\mathcal{N}$;
- $\leq$ is a binary relation over $\mathcal{N}$; and
- I is an interpretation function mapping atomic propositional symbols to truth values at each state, i.e $I : T \times Prop \rightarrow \{true, false\}$.

In Figure 3 we define a relation '$\models$', which evaluates an $\mathrm{SNF}_{\mathrm{PLTL}}$ clause at a state t_i in a model $\mathcal{M}$.

Definition 2 (Satisfiability) An $\mathrm{SNF}_{\mathrm{PLTL}}$ clause, C, is satisfiable if, and only if, there exists a model $\mathcal{M}$ such that $\langle \mathcal{M}, t_0 \rangle \models C$.

Definition 3 (Validity) An $\mathrm{SNF}_{\mathrm{PLTL}}$ clause, C, is valid if, and only if, it is satisfiable in every possible model, i.e. for each $\mathcal{M}$, $\langle \mathcal{M}, t_0 \rangle \models C$.

[3] In the language of $\mathrm{SNF}_{\mathrm{PLTL}}$ the $\Box$ operator is used only as an operator which surrounds the conjunction of clauses which is needed for the correctness of the translation. In the translation of a PLTL formula φ into $\mathrm{SNF}_{\mathrm{PLTL}}$, we remove any occurrence of the '$\Box$' operator in φ substituting it by a corresponding fixpoint definition. This is based on the fact that this operator can be understood as an infinite sequence of the '$\bigcirc$' operations and, thus, can be represented in $\mathrm{SNF}_{\mathrm{PLTL}}$ by a set of the step clauses (see [10] for details).

2.3 Example Transformation

We precede our example of transformation into $\mathrm{SNF_{PLTL}}$ by describing the main stages of this procedure referring the interested reader to [10] for further details. Recall that $\mathrm{SNF_{PLTL}}$ is used as a normal form for PLTL formulae to which resolution rules could be applied in order to test their validity [8, 10]. Thus, assuming that our aim is to check if a PLTL formula φ is valid, we first negate it, and then push negations inward until they are only applied to literals variables. This gives $NNF(\varphi)$, the *Negation Normal Form* of φ. Then we anchor the latter to **start**, obtaining $\Box(\mathbf{start} \Rightarrow NNF(\varphi))$ and, following the algorithm in [8, 10], gradually apply transformation rules in order to obtain the desired form of $\mathrm{SNF_{PLTL}}$ clauses. Abbreviating the corresponding transformation procedure by τ, we next describe its main stages.

First, we reduce the nesting of the temporal operators providing renaming of the complex subformulae, arguments of these operators, by new propositions. As a result, any temporal operator applies to literals.

Secondly, we remove unwanted temporal operators, $\Box$, $\mathcal{U}$, and $\mathcal{W}$. The corresponding rules are based upon fixpoint definitions of these modalities [2]. In the formulation of these rules below, Q abbreviates a classical formula, α and β are literals, and γ is a new proposition.

Removal of $\Box$

$$\tau[\Box(Q \Rightarrow \Box\alpha)] \quad \longrightarrow \quad \left\{ \begin{array}{l} \tau[\Box(Q \Rightarrow \alpha \wedge \gamma)] \;\wedge \\ \tau[\Box(\gamma \Rightarrow \bigcirc(\alpha \wedge \gamma))] \end{array} \right\}$$

Removal of $\mathcal{U}$

$$\tau[\Box(Q \Rightarrow \alpha\,\mathcal{U}\,\beta)] \quad \longrightarrow \quad \left\{ \begin{array}{l} \tau[\Box(Q \Rightarrow \Diamond\beta)] \;\wedge \\ \tau[\Box(Q \Rightarrow (\beta \vee (\alpha \wedge \gamma)))] \;\wedge \\ \tau[\Box(\gamma \Rightarrow \bigcirc(\beta \vee (\alpha \wedge \gamma)))] \end{array} \right\}$$

Removal of $\mathcal{W}$

$$\tau[\Box(Q \Rightarrow \alpha\,\mathcal{W}\,\beta)] \quad \longrightarrow \quad \left\{ \begin{array}{l} \tau[\Box(Q \Rightarrow (\beta \vee (\alpha \wedge \gamma)))] \;\wedge \\ \tau[\Box(\gamma \Rightarrow \bigcirc(\beta \vee (\alpha \wedge \gamma)))] \end{array} \right\}$$

Having removed unwanted temporal operators, we apply the transformation procedure τ obtaining the desired form of $\mathrm{SNF_{PLTL}}$ clauses. Many of the rules used here are classical transformations. (In the example below we refer to these rules as SIMP). Other rules (see [10]) invoke temporal reasoning. One of these rules called 'Temp', applies to $A \Rightarrow B$, where A and B are purely classical formulae, introducing a temporal context. Since this technique will be used in the example below, here we formulate the corresponding rule:

TEMP

$$\tau[\Box(A \Rightarrow B)] \quad \longrightarrow \quad \left\{ \begin{array}{l} \tau[\Box(\mathbf{start} \Rightarrow \neg A \vee B) \;\wedge \\ \tau[\Box(\mathbf{true} \Rightarrow \bigcirc(\neg A \vee B))] \end{array} \right\}$$

As an example, let us consider the translation into $\mathrm{SNF_{PLTL}}$ of the PLTL *induction* principle.

$$\Box(p \Rightarrow \bigcirc p) \Rightarrow (p \Rightarrow \Box p) \tag{2}$$

To check that (2) is valid we negate it, obtaining $\neg(\Box(p \Rightarrow \bigcirc p) \Rightarrow (p \Rightarrow \Box p))$ and then derive the *Negation Normal Form* of (2):

$$\Box(p \Rightarrow \bigcirc p) \land p \land \Diamond\neg p \qquad (3)$$

In carrying out the translation of (3) into $\mathrm{SNF}_{\mathrm{PLTL}}$, below we omit the outer '$\Box$' connective that surrounds the conjunction of clauses and, for convenience, consider a set of clauses rather than the conjunction.

Thus, we first anchor (3) to **start**, and rename a complex subformula on the right hand side of the generated formula by a new proposition, x (steps 2 and 3). Then we simplify the formula in step 3 by splitting conjunctions on its right hand side (steps 4–6).

1.	**start**	$\Rightarrow$	$\Box(p \Rightarrow \bigcirc p) \land p \land \Diamond\neg p$	given
2.	**start**	$\Rightarrow$	x	1, Renaming
3.	x	$\Rightarrow$	$\Box(p \Rightarrow \bigcirc p) \land p \land \Diamond\neg p$	1, Renaming
4.	x	$\Rightarrow$	$\Box(p \Rightarrow \bigcirc p)$	3, SIMP
5.	x	$\Rightarrow$	p	3, SIMP
6.	x	$\Rightarrow$	$\Diamond\neg p$	3, SIMP

At this stage we reduce the nesting of temporal operators (in 4) by renaming the embedded subformula $p \Rightarrow \bigcirc p$ by a new proposition, y.

7.	x	$\Rightarrow$	$\Box y$	4, Renaming
8.	y	$\Rightarrow$	$(p \Rightarrow \bigcirc p)$	4, Renaming

Now, since the '$\Box$' operator applies to a literal (in 7), we remove it deriving formulae 9 and 10 below, and introducing a new variable, t.

9.	x	$\Rightarrow$	$y \land t$	7, Removal of $\Box$
10.	t	$\Rightarrow$	$\bigcirc(y \land t)$	7, Removal of $\Box$

Further steps of the translation give us the desired $\mathrm{SNF}_{\mathrm{PLTL}}$ clauses. Firstly, we transform 8 to obtain the following:

11.	$y \land p$	$\Rightarrow$	$\bigcirc p$	8, SIMP

Then we simplify formulae 9 and 10 splitting the conjunction in 9 (steps 12 and 13) and distributing $\bigcirc$ over the conjunction in 10 (steps 14 and 15).

12.	x	$\Rightarrow$	y	9, SIMP
13.	x	$\Rightarrow$	t	9, SIMP
14.	t	$\Rightarrow$	$\bigcirc y$	10, SIMP
15.	t	$\Rightarrow$	$\bigcirc t$	10, SIMP

Finally, note that steps 5, 12 and 13 are purely classical expressions. Here we introduce a temporal context incorporating the rule 'Temp':

16.	**start**	$\Rightarrow$	$\neg x \lor p$	5, TEMP
17.	**true**	$\Rightarrow$	$\bigcirc(\neg x \lor p)$	5, TEMP
18.	**start**	$\Rightarrow$	$\neg x \lor y$	12, TEMP
19.	**true**	$\Rightarrow$	$\bigcirc(\neg x \lor y)$	12, TEMP
20.	**start**	$\Rightarrow$	$\neg x \lor t$	13, TEMP
21.	**true**	$\Rightarrow$	$\bigcirc(\neg x \lor t)$	13, TEMP

$$
\begin{array}{llcl}
1. & \textbf{start} & \Rightarrow & x \\
2. & \textbf{start} & \Rightarrow & p \\
3. & \textbf{start} & \Rightarrow & y \\
4. & \textbf{start} & \Rightarrow & t \\
5. & y \wedge p & \Rightarrow & \bigcirc p \\
6. & t & \Rightarrow & \bigcirc y \\
7. & t & \Rightarrow & \bigcirc t \\
8. & x & \Rightarrow & \Diamond \neg p
\end{array}
$$

Figure 4: Normal form for the negation of PLTL induction.

The normal form of the given PLTL formula is represented by the set of formulae 2, 6, 11, 14–21. However, the following reasoning can be applied to this set of clauses, producing a semantically equivalent, but simpler, set.

First, note that the initial clause 2 and the initial clauses 16, 18 and 20, following the semantics of SNF_{PLTL}, can only be satisfied together in a model $\mathcal{M}$, if the initial state of $\mathcal{M}$ satisfies p, y and t. Thus, without loss of generality we can consider clauses **start** $\Rightarrow p$, **start** $\Rightarrow y$ and **start** $\Rightarrow t$ instead of the clauses 16, 18 and 20.

This reasoning is simulated in the clausal resolution technique by a *step* resolution operation [8], which, for example, being applied to **start** $\Rightarrow x$ and **start** $\Rightarrow \neg x \vee p$, results in **start** $\Rightarrow p$ (see [8, 10] for details of the clausal resolution method).

Also, as it follows from the temporal resolution algorithm [10], clauses 17, 19 and 21 can not effectively participate in the subsequent refutation, hence, they are redundant. Therefore, we can omit these clauses without loss of generality.

As a result we obtain a simplified set of the SNF_{PLTL} clauses for (3). We present this set in Figure 4 such that we first give initial clauses, then step clauses and, finally, any sometime clauses.

Note that, in this set of clauses, all propositions except p are new - they have been introduced during the translation of (3) either via renaming or via removing unwanted temporal operators. The truth values of these new propositions are associated, for example, in the case of renaming, with the truth values of the renamed formulae [8, 17].

It is easy to establish that the set of clauses in Figure 4 is unsatisfiable. Indeed, consider an arbitrary model $\mathcal{M}$. Clauses 1–4 indicate that x, p, y, t are satisfied at the initial moment t_0 of a model $\mathcal{M}$. Further, if y, p, t, which occur on the left hand side of clauses 5–7, are satisfied at some moment t_i $(0 \leq i)$, then p must occur at every moment t_j, where $i < j$. Therefore, since y, p and t are satisfied at t_0, and since t_0 also satisfies p, we conclude that p must be satisfied at every state of $\mathcal{M}$. However, since x is satisfied at the initial moment t_0, and clause 8 requires that once x is satisfied, $\neg p$ must eventually occur, there must be a moment t_k $(0 \leq k)$ which satisfies $\neg p$. Thus, t_k must satisfy both p and $\neg p$ — hence, a contradiction.
A resolution method defined in [7], if applied to the set of clauses in Figure 4, produces a refutation, thus, indicating that (3), a negation normal form of induction, is

unsatisfiable, hence, that (2), the induction principle itself, is valid.

3 Representation of Büchi automata in QPLTL

The main stages of our translation of Büchi automaton into normal form are similar to those used in [14], where the authors describe the encoding of Büchi automata in terms of quantified propositional temporal logic (QPLTL). Thus, we precede the presentation of our result by introducing Büchi automata, the logic QPLTL and the translation of Büchi automata into QPLTL.

3.1 Nondeterministic Büchi automata

w-automata on infinite words were introduced by Büchi as a tool for investigating the decidability of restricted classical second-order logics [3], monadic second order logic of one successor (S1S).

Definition 4 A Büchi word automaton, $\mathcal{A}$, (or a Büchi automaton over a structure $\mathcal{M}$) is a tuple $\langle \Sigma,\ S,\ F_0,\ \delta,\ F_A \rangle$ where:

- Σ is an alphabet;
- $S = s_0, s_1, \ldots s_k$ is a finite set of states;
- $F_0 \subseteq S$ is a set of initial states;
- $\delta = S \times \Sigma \longrightarrow 2^S$ is a non-deterministic transition function;
- $F_A \subseteq S$ is a set of accepting states.

Visiting a state s_i $(0 \leq i \leq k)$ and reading π_m $(0 \leq m)$, a component of an infinite word $\omega = \pi_0\pi_1\pi_2 \ldots$, the automaton makes a *move* to one of the states, members of S, say, to s_j, according to the non-deterministic transition function δ.

A *run*, $\tau_{\mathcal{A}}$, of a Büchi automaton $\mathcal{A}$ over the word ω is an infinite sequence of states where the first state is one of the initial states, elements of F_0, and every other state is reached from its immediate predecessor by a move.

A run, $\tau_{\mathcal{A}}$, is *successful* if there is a state $s \in F_A$ such that s appears in $\tau_{\mathcal{A}}$ infinitely often.[4] An automaton $\mathcal{A}$ accepts the infinite word ω (in other terms, the language recognized by $\mathcal{A}$ is *not empty*) if it has a successful run $\tau_{\mathcal{A}}$.

3.2 Quantified Propositional Temporal Logic (QPLTL)

Quantified propositional temporal logic, QPLTL, introduced in [18][5] extends linear-time temporal logic by allowing propositional quantification (see also [19, 14] for

[4] Note that by varying the acceptance condition we induce different types of ω automata [20].

[5] Usually this logic is known as QPTL. We adopt here the abbreviation 'QPLTL' to emphasize that the propositional quantification is introduced here in the framework of *linear-time* (but not, for example *branching-time*) temporal logic.

details). Note that, while in [19] the logic QPLTL is formulated using only future-time operators, in [14], the authors presented a complete axiomatization for QPLTL formulated with both past and future-time operators. As it is known for linear-time temporal logic [11], given the underlying temporal structures have the initial moment of time, the introduction of the past-time operators does not increase the expressiveness of the logic and is usually provided only for convenience. Since in our representation of Büchi automata in SNF, we essentially use the ideas of representation of Büchi automata in QPLTL [14], below we present the syntax and semantics of QPLTL incorporating both linear and future-time operators, following closely [14].

3.2.1 QPLTL syntax.

We define the language of QPLTL based on the following symbols:

- atomic propositions $p, q, r, \ldots, p_1, q_1, r_1, \ldots, p_n, q_n, r_n, \ldots$;
- classical operators $\neg, \wedge, \Rightarrow, \vee$;
- temporal operators

<u>Future</u>:		<u>Past</u>:	
'always in the future'	$\square$,	'always in the past'	$\blacksquare$,
'at sometime in the future'	$\diamondsuit$,	'at sometime in the past'	$\blacklozenge$,
'until'	$\mathcal{U}$,	'since'	$\mathcal{S}$,
'at the next moment in time'	$\bigcirc$,	'weak previous'	$\bullet$,

- (propositional) quantifiers
  - 'universal quantifier' $\forall$,
  - 'existential quantifier' $\exists$.

Well-formed QPLTL formulae are defined as follows:

- Every atomic proposition is a formula.
- If A and B are formulae then so are

  $\neg A,\ A \vee B,\ A \wedge B,\ A \Rightarrow B,$

  $\square A,\ \diamondsuit A,\ A\,\mathcal{U}\,B,\ \bigcirc A,$

  $\blacksquare A,\ \blacklozenge A,\ A\,\mathcal{S}\,B,\ \bullet A.$

- If x is an atomic proposition and A is a formula then $\forall x A$ and $\exists x A$ are formulae.

3.2.2 QPLTL semantics

Similar to $\mathrm{SNF}_{\mathrm{PLTL}}$ clauses, we interpret formulae of QPLTL over linear, discrete structures $\mathcal{M} = \langle T, \leq, I \rangle$ which satisfy the conditions mentioned in §2.2.
Below we define a relation '$\models$', which evaluates QPLTL formulae at a state t_i in a model $\mathcal{M}$. For the clarity, we present the semantics separately for the temporal operators (Figure 5) and propositional quantification (Figure 6), omitting the cases of the Boolean connectives which are similar to those given in the semantics for $\mathrm{SNF}_{\mathrm{PLTL}}$ in §2.2.

$\langle \mathcal{M}, t_i \rangle$	$\models \Box B$	iff	for each j, if $i \leq j$ then $\langle \mathcal{M}, t_j \rangle \models B$;
$\langle \mathcal{M}, t_i \rangle$	$\models \Diamond B$	iff	there exists j such that $i \leq j$ and $\langle \mathcal{M}, t_j \rangle \models B$;
$\langle \mathcal{M}, t_i \rangle$	$\models \bigcirc B$	iff	$\langle \mathcal{M}, t_{i+1} \rangle \models B$;
$\langle \mathcal{M}, t_i \rangle$	$\models A \mathcal{U} B$	iff	there exists j such that $i \leq j$ and $\langle \mathcal{M}, t_j \rangle \models B$ and for each k if $i \leq k < j$ then $\langle \mathcal{M}, t_k \rangle \models A$;
$\langle \mathcal{M}, t_i \rangle$	$\models \blacksquare B$	iff	for each j, if $0 \leq j \leq i$ then $\langle \mathcal{M}, t_j \rangle \models B$;
$\langle \mathcal{M}, t_i \rangle$	$\models \blacklozenge B$	iff	there exists j such that $0 \leq j \leq i$ and $\langle \mathcal{M}, t_j \rangle \models B$;
$\langle \mathcal{M}, t_i \rangle$	$\models \bullet B$	iff	$i = 0$, or $0 < i$ and $\langle \mathcal{M}, t_{i-1} \rangle \models B$;
$\langle \mathcal{M}, t_i \rangle$	$\models A \mathcal{S} B$	iff	there exists j such that $j \leq i$ and $\langle \mathcal{M}, t_j \rangle \models B$ and for each k if $j < k \leq i$ then $\langle \mathcal{M}, t_k \rangle \models A$.

Figure 5: Semantics of QPLTL for temporal operators

$\langle \mathcal{M}, t_i \rangle \models \forall x B$	iff	for every model $\mathcal{M}'$ which differs from $\mathcal{M}$ at most in the evaluation of x, $\langle \mathcal{M}', t_i' \rangle \models B$;
$\langle \mathcal{M}, t_i \rangle \models \exists x B$	iff	there exists a model $\mathcal{M}'$ such that it differs from $\mathcal{M}$ at most in the evaluation of x and $\langle \mathcal{M}', t_i' \rangle \models B$.

Figure 6: Semantics of QPLTL for propositional quantification

Definition 5 (Satisfiability) A QPLTL formula, Q, is satisfiable if, and only if, there exists a model $\mathcal{M}$ such that $\langle \mathcal{M}, t_0 \rangle \models Q$.

Definition 6 (Validity) A QPLTL formula, Q, is valid if, and only if, it is satisfiable in every possible model, i.e. for each $\mathcal{M}$, $\langle \mathcal{M}, t_0 \rangle \models Q$.

3.3 Representation of Büchi automata in QPLTL

In [14] a complete axiom system for QPLTL formulated with both future and past-time operators is presented. In the proof of completeness of this axiom system, which is based on the fact that QPLTL is expressively equivalent to Büchi automata, the construction of the QPLTL formula characterizing a Büchi automaton is essential. Thus, in [14] the authors represent a Büchi automaton as a QPLTL expression whose construction is defined in stages as below.

Given a Büchi automaton $\mathcal{A} = \langle \Sigma,\ S,\ F_0,\ \delta,\ F_A \rangle$, let $S = s_0 \ldots s_k$. Let $s_a \ldots s_m$ $(0 \leq a \leq m \leq k)$ be the initial states, i.e. members of F_0, let $s_n \ldots s_r$ $(0 \leq$

$n \leq r \leq k$) be the accepting states, i.e. members of F_A, and let a set of variables $Q = q_0, \ldots q_k$ consist of new propositions of QPLTL language such that q_i $(0 \leq i \leq k)$ encodes the state s_i of the automaton. Let $Q_0 \subseteq Q$ abbreviate a subset of Q which encodes the initial states, i.e. members of F_0 and $Q_A \subseteq Q$ abbreviate a subset set of Q which encodes the accepting states, i.e. members of F_A.

Further, let t and y be new propositions of QPLTL such that $t \neq y \neq q_i$, for any $0 \leq i \leq k$. Finally, let **start** be defined as $\bullet$ **false**.[6]
Now, the following QPLTL formula defines a marker, t, which is determined to be true in the current state of the automaton and false elsewhere.

$$S_n(t) = \bullet\blacksquare\neg t \wedge t \wedge \bigcirc \square\neg t \tag{4}$$

Next, a concept of a successful run is represented as

$$acc_{r\mathcal{A}}(y) = init_{\mathcal{A}}(y) \wedge run_{\mathcal{A}}(y) \wedge acc^{B}_{\mathcal{A}}(y) \tag{5}$$

The '$init_{\mathcal{A}}$' component, defined below, determines a proposition y to be true at one of the initial states.

$$init_{\mathcal{A}}(y) = \bigvee_{q_i \in Q_0} [y \equiv q_i] \tag{6}$$

Considering $\sigma(s_i, s_j)$ as a propositional assertion which is true if $\delta(s_i, \pi_k) = s_j$, for some element, π_k, of an infinite word, the '$run_{\mathcal{A}}$' component is given by

$$run_{\mathcal{A}}(y) = \square \bigvee_{q_i, q_j \in Q} [(y = q_i) \wedge \bigcirc(y = q_j) \wedge \sigma(s_i, s_j)] \tag{7}$$

The latter requires that a proposition y is true at the state s_i and at the next moment in time, s_j, when the automaton is moving from a state s_i to a state s_j, according to the transition function δ.

The '$acc^{B}_{\mathcal{A}}(y)$' component represents the Büchi acceptance condition:

$$acc^{B}_{\mathcal{A}}(y) = \square\diamondsuit \bigvee_{q_i \in Q_A} [y = q_i] \tag{8}$$

Thus, (8) requires that y is true infinitely often in one of the accepting states.
Now, having defined (5), the last component, $app_{r\mathcal{A}}(y)$, needed for representation of Büchi automaton, is composed as

$$app_{r\mathcal{A}}(y) = \blacklozenge\, (\textbf{start} \wedge acc_{r\mathcal{A}}(y)). \tag{9}$$

Finally, the desired QPLTL formula $\xi_{\mathcal{A}}$, a characteristic formula for a Büchi automaton, $\mathcal{A}$, is given as

$$\xi_{\mathcal{A}} = \exists t \exists y (S_n(t) \wedge app_{r\mathcal{A}}(y)) \tag{10}$$

[6]As we mentioned, the introduction of the past-time operators in QPLTL does not increase its expressiveness being provided only for convenience. This, for example, allows the authors of [14] to define the constant **start** in terms of other operators. Similarly, $\mathrm{SNF}_{\mathrm{PLTL}}$ can be defined either with past-time operators or only with future-time operators. We gave the preference to the latter case, as this formulation of $\mathrm{SNF}_{\mathrm{PLTL}}$ is more convenient for defining our resolution technique. As a consequence, we must introduce the constant, **start**, as one of the initial symbols of the language.

where t is a marker which indicates the current state of the automaton and y is a proposition which encodes its accepting run.

Given a successful run, τ, of a Büchi automaton, $\mathcal{A} = \langle \Sigma,\ S,\ F_0,\ \delta,\ F_A \rangle$, over an infinite word $\omega = \pi_0\pi_1\pi_2\ldots$, we can build a linear-time model $\mathcal{M} = \langle T, \leq, I \rangle$, such that $\langle \mathcal{M}, t_0 \rangle \models \pi_0, \langle \mathcal{M}, t_1 \rangle \models \pi_1,\ \ldots\ \langle \mathcal{M}, t_n \rangle \models \pi_n\ \ldots$.

Now, as it follows from [14], we can define a new model, $\mathcal{M}'$, which is the same as $\mathcal{M}$ except for the interpretation of the new propositions $q_0, \ldots q_k, t$ and y which were used in the construction of the characteristic formula $\xi_{\mathcal{A}}$, following the construction of 10, such that $\mathcal{M}' \models \xi_{\mathcal{A}}$. (This is a reformulation of Claim 15 in [14], which states that every Büchi automaton, $\mathcal{A}$, is congruent to its characteristic formula $\xi_{\mathcal{A}}$. The relevant terminology and corresponding definitions can be found in [14].)

As we mentioned, the above construction was essential to the authors of [14] in proving the completeness of the axiom system for QPLTL. We will utilize similar ideas in our representation of Büchi automata in terms of $\mathrm{SNF}_{\mathrm{PLTL}}$, which is described in the next section.

4 Representation of Büchi automata in $\mathrm{SNF}_{\mathrm{PLTL}}$

Here, given a Büchi automaton $\mathcal{A}$, we construct its characteristic formula, $SNF_{\mathcal{A}}$, as a set of $\mathrm{SNF}_{\mathrm{PLTL}}$ clauses. The main stages of the construction of $SNF_{\mathcal{A}}$, encoding of the set of the initial states, representing the run of the automaton and the acceptance condition are similar to those in [14]. However, due to the specifics of the $\mathrm{SNF}_{\mathrm{PLTL}}$ notation, where we use only future-time temporal operators, we do not invoke the backward recursion.

4.1 Translation of Büchi automata into $\mathrm{SNF}_{\mathrm{PLTL}}$

Let $\mathcal{A} = \langle \Sigma,\ S,\ F_0,\ \delta,\ F_A \rangle$, be a Büchi automaton such that

- $\Sigma = 2^{Prop}$ is an alphabet;
- $S = s_0 \ldots s_k$ is a finite set of states;
  - $s_a \ldots s_m$ $(0 \leq a \leq m \leq k)$ are the initial states, members of F_0,
  - $s_n \ldots s_r$ $(0 \leq n \leq r \leq k)$ are the accepting states, members of F_A;

Also, we assume that the states of a Büchi automaton are labelled as follows. For every proposition $p \in Prop$, given $\delta(s_i, \pi) = s_j$, if $p \in \pi$ then $p \in L(s_j)$ else if $p \notin \pi$ then $\neg p \in L(s_j)$. In our translation we will explicitly encode this labelling. Now, in order to describe the construction of the characteristic clause set $SNF_{\mathcal{A}}$ (expression (15) below) we introduce formulae (11)–(14), which represent the set of initial states, the transition function, the labelling and the acceptance condition of a Büchi automaton. Note that each of these formulae is thought of as a conjunction of formulae in the scope of the outer $\square$ operator. Let $q_0, \ldots, q_k$ be new propositions of our $\mathrm{SNF}_{\mathrm{PLTL}}$ language such that q_i $(0 \leq i \leq k)$ encodes the state s_i of the automaton.

The set F_0 is specified by the following $\mathrm{SNF}_{\mathrm{PLTL}}$ expression.

$$SNF_{init_{\mathcal{A}}}: \quad \mathbf{start} \Rightarrow q_1 \vee \ldots \vee q_m \tag{11}$$

Further, we introduce the following set of clauses to represent the transition function of the automaton. Note that some of the expressions below should be further translated (by simple manipulations) into required form of $\mathrm{SNF_{PLTL}}$ clauses. For every i ($0 \leq i \leq k$) and for any j ($j \neq i$),

$$SNF_{tran_{\mathcal{A}}}: \qquad (12)$$

$$\begin{array}{lrcl}
1. & q_i & \Rightarrow & \bigcirc \delta_{s_i} \\
2. & \mathbf{start} & \Rightarrow & \neg q_i \vee \bigwedge\limits_{i \neq j} (\neg q_j) \\
3. & \mathbf{true} & \Rightarrow & \bigcirc \left[\neg q_i \vee \bigwedge\limits_{i \neq j} (\neg q_j) \right]
\end{array}$$

where δ_{s_i} encodes the disjunction of states which the automaton may move to when it visits s_i according to the transition function δ, or δ_{s_i} is **false** if s_i does not have successor states. Formulae 2 and 3 above encode the fact that for any i, if q_i is true at some state s_i then every proposition q_j, different from q_i, is false there, i.e. the automaton can only be at one state at any particular time.

Next, we represent the labelling of the states of the automaton by the following set of clauses, constructed for every q_i and every $x_i \in L(s_i)$.

$$SNF_{lab_{\mathcal{A}}}: \qquad (13)$$

$$\begin{array}{lrcl}
1. & \mathbf{start} & \Rightarrow & \neg q_i \vee \bigwedge\limits_{x_i \in L(s_i)} x_i \\
2. & \mathbf{true} & \Rightarrow & \bigcirc (\neg q_i \vee \bigwedge\limits_{x_i \in L(s_i)} x_i)
\end{array}$$

The Büchi acceptance condition is given by the following set of $\mathrm{SNF_{PLTL}}$ clauses.

$$SNF_{acc^B_{\mathcal{A}}}: \qquad (14)$$

$$\begin{array}{lrcl}
1. & \mathbf{start} & \Rightarrow & y \\
2. & y & \Rightarrow & \Diamond l \\
3. & l & \Rightarrow & \bigcirc z \\
4. & z & \Rightarrow & \Diamond l \\
5. & \mathbf{start} & \Rightarrow & \neg l \vee q_n \vee \ldots \vee q_r \\
6. & \mathbf{true} & \Rightarrow & \bigcirc (\neg l \vee q_n \vee \ldots \vee q_r)
\end{array}$$

where $q_n \ldots q_r$ encode the accepting states of the automaton and y, l and z are new propositions.

Finally, let $SNF'_{init_{\mathcal{A}}}$, $SNF'_{tran_{\mathcal{A}}}$, $SNF'_{lab_{\mathcal{A}}}$ and $SNF'_{acc^B_{\mathcal{A}}}$ be obtained from $SNF_{init_{\mathcal{A}}}$, $SNF_{tran_{\mathcal{A}}}$, $SNF_{lab_{\mathcal{A}}}$ and $SNF_{acc^B_{\mathcal{A}}}$, respectively, by translating their components into the required form of $\mathrm{SNF_{PLTL}}$ clauses and dropping the outer $\Box$ operators.

Now, a Büchi automaton $\mathcal{A}_B$ is characterized by the following $\mathrm{SNF_{PLTL}}$ expression known as a *characteristic clause set* and abbreviated by $SNF_{\mathcal{A}}$.

$$SNF_{\mathcal{A}}: \quad \Box(SNF'_{init_{\mathcal{A}}} \wedge SNF'_{tran_{\mathcal{A}}} \wedge SNF'_{lab_{\mathcal{A}}} \wedge SNF'_{acc^B_{\mathcal{A}}}) \qquad (15)$$

4.2 Correctness of the Translation

Theorem 7 Given a Büchi word automaton $\mathcal{A}_B$, we can construct a characteristic clause set, $SNF_{\mathcal{A}}$, such that $\mathcal{A}_B$ has an accepting run, $\tau_{\mathcal{A}}$ (over an infinite word ω), if and only if, $SNF_{\mathcal{A}}$ is satisfiable.

Proof We first show that given a non-empty Büchi word automaton $\mathcal{A}_B$ and $\tau_{\mathcal{A}}$ as its accepting run over an infinite word $\omega = \pi_0\pi_1\pi_2\ldots$, there is a model which satisfies the characteristic clause set, $SNF_{\mathcal{A}}$, i.e. which satisfies formula (15).

Recall that, given a non-empty automaton, $\mathcal{A}_B = \langle \Sigma,\ S,\ F_0,\ \delta,\ F_A\rangle$, where $S = s_0, s_1, s_2, \ldots s_k$, and its accepting run, $\tau_{\mathcal{A}}$ over an infinite word $\omega = \pi_0\pi_1\pi_2\ldots$, each state, $s_j \in \tau_{\mathcal{A}}$, is labelled by L_{s_j}, a member of ω. Now, let $\tau_{\mathcal{A}}$ be an infinite sequence of states $s_a^0, s_b^1, s_c^2, \ldots$ such that $s_a, s_b, s_c \in S$ $(0 \leq a, b, c \leq k)$ and the upper index, i, of the state s_j^i (where $j = a, b, c\ldots$ and $0 \leq i$) indicates that s_j is the $i-th$ state visited by the automaton on run $\tau_{\mathcal{A}}$. Let f be a function such that for every such i, $f(i) = s_j$ if, and only if, s_j is the i-th state visited by the automaton on run τ_A. Thus, we can unwind the run $\tau_{\mathcal{A}}$ of the automaton into an infinite sequence of states $T = t_0, t_1, t_2, \ldots$ and build an intermediate model structure $\mathcal{M} = \langle T, \leq, I\rangle$, such that $t_i \in T$ $(0 \leq i)$ is identified with the label of the $i - th$ state of the run $\tau_{\mathcal{A}}$. Therefore, for every $x_k \in L(s_j^i)$, $I(t_i, x_k) = true$.

Now we construct a model $\mathcal{M}'$ which satisfies a characteristic formula (15) built according with our translation.

Let a model $\mathcal{M}' = \langle T', \leq, I'\rangle$ be the same as $\mathcal{M}$ except for the interpretation of the new propositions $q_0, \ldots q_k, y, l$ and z which satisfy the following conditions:

(i) for every proposition q_j, $(0 \leq j \leq k)$ and every state $t_i \in T'$, $\langle \mathcal{M}', t_i\rangle \models q_j \Leftrightarrow f(i) = s_j$;

(ii) $\langle \mathcal{M}', t_0\rangle \models y$ and for every $t_n \neq t_0$, $\langle \mathcal{M}', t_n\rangle \not\models y$;

(iii) l is satisfied exactly at those states that satisfy q_n (where q_n encodes the state $s_n \in F_A$ which the automaton hits infinitely often) and false elsewhere;

(iv) z is satisfied by every state t_m $(1 \leq m)$ such that t_{m-1} satisfies l.

Thus, by the construction of $\mathcal{M}'$, (11) is satisfied in it. Also, from (i), for any $t_m \in T'$, and any i $(0 \leq i \leq k)$, if $\langle \mathcal{M}', t_m\rangle \models q_i$ then $\langle \mathcal{M}', t_m\rangle \models L(s_i)$. Hence, (13) is satisfied in $\langle \mathcal{M}'\rangle$.

Further, from the definition of the run and from (i), for any q_i, if the left hand side of the condition 1 of (12) is satisfied at some state $t_m \in T'$, then, by construction of $\mathcal{M}'$, $\langle \mathcal{M}', t_{m+1}\rangle \models \delta_{s_i}$. Also, by definition of $\mathcal{M}'$, conditions 2 and 3 are satisfied, therefore, (12) is satisfied in $\mathcal{M}'$.

Finally, consider the $SNF_{acc^B_{\mathcal{A}}}$ component of (15). Since τ is the accepting run, it hits at least one of the accepting states, say s_r $(s_r \in s_1 \ldots s_n)$, infinitely often. The condition 1 of (14) is satisfied by (ii). Also, since s_r is visited by the automaton (at least once), then, by (ii) and (iii), the condition 2 of (14) is satisfied. Further, if s_r is also one of the initial states of the automaton then q_r must be true at t_0. Alternatively, if $s_r \notin F_0$ then, by (iii), t_0 does not satisfy l, i.e. $\langle \mathcal{M}', t_0\rangle \models \neg l$, hence, condition 5 is satisfied. Further, for every state $t_m \in T'$, either $\langle \mathcal{M}', t_m\rangle \models q_r$ or $\langle \mathcal{M}', t_m\rangle \models \neg l$ in which case, by (iii), $t_m \not\models l$, hence $t_m \models \neg l$. Therefore, condition 6 of (14) is satisfied in $\mathcal{M}'$. Finally, since s_r is visited by the automaton infinitely often, due to (iii) and (iv), conditions 3 and 4 of (14) are satisfied. Therefore, all conditions required by (14) have been shown satisfiable in $\mathcal{M}'$. Hence, $\mathcal{M}'$ satisfies formulae (11)-(14), and, by the semantics of the $\Box$ operator, it satisfies (15) as required.

Now, assume that we are given a satisfiable set, C, of $\mathrm{SNF}_{\mathrm{PLTL}}$ clauses, (15). Following [10], we use C to construct a behaviour graph, $\mathcal{G} = \langle N, E \rangle$, where N is a (finite) set of states and E is a set of transitions, i.e. a structure from which models that satisfy C can be extracted. (Construction of behaviour graph has been used in [10] as a part of the completeness proof for the temporal resolution method for PLTL.) Members of N are pairs $(\mathcal{V}, \mathcal{E})$, where $\mathcal{V}$ is an evaluation of all the propositions in C and $\mathcal{E}$ is a subset of eventualities in C. Transitions in E are between nodes that satisfy step clauses according to the following criteria: if the left hand side of each step clause within (15) is satisfied in a state v_i, then transitions are made to every state which satisfies the right hand side of all step clauses after having deleting the $\bigcirc$ operator. Having constructed a finite graph as above we delete those paths which can not give us models for the set of clauses. Here the standard set of deletion rules is applied: every state without successors is deleted and if the left hand side of the eventuality clauses is satisfied at v_i but there is no path from v_i to the state which satisfies l, then v_i and all its successors are deleted. According to [10], a reduced behaviour graph (i.e. the one obtained after deletions) is not empty if, and only if, the set of clauses is satisfiable. Therefore, the output reduced finite graph, $\mathcal{G}' = \langle N', E' \rangle$, constructed for (15), is not empty.

Now, based on the properties of the reduced behaviour graph $\mathcal{G}' = \langle N', E' \rangle$, we can define a Büchi automaton $\mathcal{A}_B = \langle \Sigma,\ S,\ F_0,\ \delta,\ F_A \rangle$ as follows.

- $\Sigma = 2^{Prop_C}$, where $Prop_C$ is a set of propositions of the clause set C;
- S is a finite set, N', of states of $\mathcal{G}'$;
- F_0 is a set of those states that satisfy all the initial clauses occurring within (15);
- F_A is a set of those states which satisfy proposition l which occurs on the right hand side of the eventuality clauses of (15);
- δ is a transition function such that for any $s_i, s_j \in N'$, $(s_i \times \sigma \longrightarrow s_j) \in \delta$ if $\langle s_i, s_j \rangle \in E'$ and σ is the valuation in s_j.

The $SNF_{acc^B_{\mathcal{A}}}$ formula represented by the set of clauses (14) guarantees that there is a run, $\tau_{\mathcal{A}}$, through such reduced finite state system which starts at one of the initial states and hits one of the accepting states (which satisfies l) - this is due to clauses 1-2. Further, since each successor of this state will satisfy z as required by the satisfiability condition for clause 3, a run $\tau_{\mathcal{A}}$ will eventually hit a state which satisfies l. Due to the finiteness of the state system, we are, therefore, assured that $\tau_{\mathcal{A}}$ hits the accepting state infinitely often.

4.3 Expressing 'even(p)' in $\mathrm{SNF}_{\mathrm{PLTL}}$

As we mentioned in the introduction, a famous example which shows that Büchi automata are strictly more expressive than PLTL [22], is the property "p occurs at every even moment of time" (of some computation). Specifications of this property can be found, in particular, in terms of Büchi automata [22], QPLTL [16, 13]. In Figure 7 we give a Büchi automaton which expresses this property, then construct a characteristic clause set as described in the previous section. Further, we give a simpler, direct, specification of 'even(p)' directly in terms of $\mathrm{SNF}_{\mathrm{PLTL}}$ and show that the latter is satisfiable if, and only if, so is the characteristic clause set.

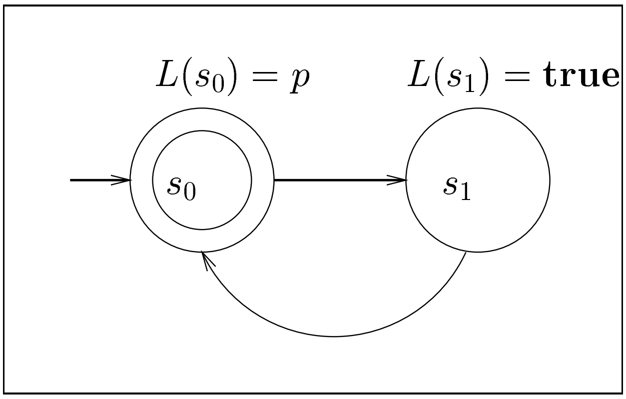

Figure 7: Büchi automaton for "even(p)"

The state structure of this Büchi automaton, $\mathcal{A}_{even}$ is very simple, it consists only of two states s_0 and s_1, such that s_0 is at the same time the initial and the accepting state. A run which this automaton accepts is an infinite sequence of states, where every even state satisfies p.

To construct a characteristic clause set, we first introduce a set of new propositions q_0, q_1 which encode the states s_0 and s_1, respectively. Now, the (singleton) set of accepting states of $\mathcal{A}_{even}$ is specified by the following formula:

$$SNF_{init_{\mathcal{A}_{even}}} : \quad \mathbf{start} \Rightarrow q_0 \tag{16}$$

Next, the transition function of the automaton $\mathcal{A}_{even}$ is given by the following set of $\mathrm{SNF}_{\mathrm{PLTL}}$ clauses:

$$SNF_{tran_{\mathcal{A}_{even}}} : \tag{17}$$

$$\begin{array}{rrcl} 1. & q_0 & \Rightarrow & \bigcirc q_1 \\ 2. & q_1 & \Rightarrow & \bigcirc q_0 \\ 3. & \mathbf{start} & \Rightarrow & (\neg q_0 \vee \neg q_1) \\ 4. & \mathbf{true} & \Rightarrow & \bigcirc(\neg q_0 \vee \neg q_1) \end{array}$$

The labelling of the states of the automaton is represented by the following set of clauses:

$$SNF_{lab_{\mathcal{A}_{even}}} : \tag{18}$$

$$\begin{array}{rrcl} 1. & \mathbf{start} & \Rightarrow & (\neg q_0 \vee p) \\ 2. & \mathbf{true} & \Rightarrow & \bigcirc(\neg q_0 \vee p) \end{array}$$

$$
\begin{array}{lrcl}
1. & \mathbf{start} & \Rightarrow & q_0 \\
2. & \mathbf{start} & \Rightarrow & \neg q_1 \\
3. & \mathbf{start} & \Rightarrow & p \\
4. & \mathbf{start} & \Rightarrow & y \\
5. & q_0 & \Rightarrow & \bigcirc q_1 \\
6. & q_1 & \Rightarrow & \bigcirc q_0 \\
7. & l & \Rightarrow & \bigcirc z \\
8. & \mathbf{true} & \Rightarrow & \bigcirc(\neg l \vee q_0) \\
9. & \mathbf{true} & \Rightarrow & \bigcirc(\neg q_0 \vee \neg q_1) \\
10. & \mathbf{true} & \Rightarrow & \bigcirc(\neg q_0 \vee p) \\
11. & y & \Rightarrow & \Diamond l \\
12. & z & \Rightarrow & \Diamond l
\end{array}
$$

Figure 8: Characteristic set for the automaton 'even(p)'.

Next, to express the Büchi acceptance condition for $\mathcal{A}_{even}$ we introduce new propositions y, l and z and proceed as follows:

$$
SNF_{acc^{B}_{\mathcal{A}_{even}}} : \qquad
\begin{array}{lrcl}
1. & \mathbf{start} & \Rightarrow & y \\
2. & y & \Rightarrow & \Diamond l \\
3. & l & \Rightarrow & \bigcirc z \\
4. & z & \Rightarrow & \Diamond l \\
5. & \mathbf{start} & \Rightarrow & \neg l \vee q_0 \\
6. & \mathbf{true} & \Rightarrow & \bigcirc(\neg l \vee q_0)
\end{array}
\tag{19}
$$

Now, a Büchi automaton $\mathcal{A}_{even}$ is characterized by the set of clauses (16) — (19). Note, however, that this merging makes some of the clauses redundant as they do not affect the satisfiability of the whole set, while other clauses can be simplified. For example, clause 5 of (19) is subsumed by clause (16) and thus, can be omitted. Also, clause 3 of (17) and clause 1 of (18) can be satisfied together with clause (16) only if **start** $\Rightarrow \neg q_1$ and **start** $\Rightarrow p$ are valid. Thus, we present the simplified characteristic set of clauses for $\mathcal{A}_{even}$ in Figure 8 such that we first give initial clauses, then step clauses and, finally, any sometime clauses.

Now we show how the property 'even(p)' can be directly represented as a much simpler set of $\mathrm{SNF_{PLTL}}$ clauses, avoiding the construction of the automaton.

$$
\text{`}even\ p' : \qquad
\begin{array}{lrcl}
1. & \mathbf{start} & \Rightarrow & q_0 \\
2. & \mathbf{start} & \Rightarrow & p \\
3. & \mathbf{true} & \Rightarrow & \bigcirc(\neg q_0 \vee p) \\
4. & q_0 & \Rightarrow & \bigcirc q_1 \\
5. & q_1 & \Rightarrow & \bigcirc q_0
\end{array}
\tag{20}
$$

First, note that it is easy to establish that the set of clauses (20) indeed defines the desired property. Also, we can see that the set (20) is simply a subset of the set of clauses given in Figure 8. On the other hand, given a model for (20), it is straightforward that we can augment it by the new propositions y, l and z such that the set of clauses in Figure 8 becomes satisfiable. Therefore, the characteristic clause set for $\mathcal{A}_{even}$ is satisfiable if, and only if, the set of clauses (20) is satisfiable.

5 Discussion

We have shown that normal form used for clausal resolution method for PLTL is expressive enough to give the succinct high-level syntactic representation of one of the types of ω-automata, Büchi automata.

Thus, we can redraw Figure 1 producing a new diagram given in Figure 9. Here, we represent the expressiveness results established in this paper by showing the possibility of the translation of an automaton (A) into the normal form (N) and vice versa.

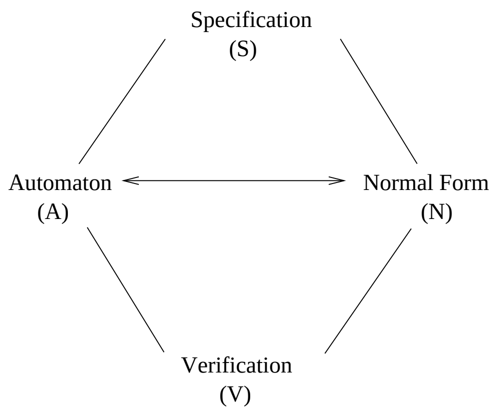

Figure 9: Specification-Verification Problem Reviewed

Although the main stages of our translation of a Büchi automaton into normal form are similar to the encoding of the latter in terms of QPLTL ([14]), we believe that our representation of Büchi automata in terms of $\mathrm{SNF}_{\mathrm{PLTL}}$ is more practical. To the best of our knowledge, the only known deductive system for the logic QPLTL is the *axiom system* developed in [14], while a well developed clausal resolution technique can be applied to a set of $\mathrm{SNF}_{\mathrm{PLTL}}$ clauses [10].

However, we consider the contribution of this work not only in that it answers a particular question on the expressiveness of the temporal logic normal form, but represents a step towards solving the general problem structure described in the introduction. As we mentioned in the introduction, in using an automata based approach to analyse a system specification, the most expensive step is the translation of the latter into the corresponding automaton. Based on the expressiveness equivalence of automata and normal form, we can directly translate a problem specification into the normal form and apply the temporal resolution technique as a verification method. We believe that application of $\mathrm{SNF}_{\mathrm{PLTL}}$ is useful when a system specification given in a high-level language, such as temporal logic, is required to represent complex temporal properties. In this case, generating an automaton can be especially costly, while the translation into normal form will involve only a linear blow-up mostly due to the introduction of new propositions [10]. Therefore, we consider a detailed analysis of the cases that are "easy" for automata/normal form representation as an important direction of the future work.

Also, in the future we will investigate the representation of other types of ω-automata acceptance conditions in $\mathrm{SNF}_{\mathrm{PLTL}}$, which will enable us to give an $\mathrm{SNF}_{\mathrm{PLTL}}$ specification for Rabin and Street automata. Noting also the similarity between $\mathrm{SNF}_{\mathrm{PLTL}}$ clauses and alternating automata, we believe that $\mathrm{SNF}_{\mathrm{PLTL}}$ can be used as a syntactic representation of the latter. This, in turn, will enable us to apply temporal resolution as a non-emptiness check for alternating automata.

Finally, taking into account the equivalence between Büchi automata and linear-time fixpoint calculus [16, 2], the result established in this work enables us to investigate an interesting problem of the translation from linear-time fixpoint calculus into $\mathrm{SNF}_{\mathrm{PLTL}}$, and thus, to potentially apply the clausal resolution method to temporal fixpoint logic.

Acknowledgements. The authors would like to thank Damian Niwinsky, Igor Walukiewicz and Giacomo Lenzi for the useful discussions, and anonymous referees for the useful comments and suggestions.

References

[1] O. Bernholtz, M. Y. Vardi, and P. Wolper. An automata-theoretic approach to branching-time model checking. In *Computer Aided Verification, Proc. 6th Int. Workshop*, volume 818 of *Lecture Notes in Computer Science*, pages 142–155, Stanford, California, June 1994. Springer-Verlag.

[2] J. Bradffield. and C. Stirling. Modal logics and mu-calculi. In J. Bergstra, A. Ponse, and S. Smolka, editors, *Handbook of Process Algebra*, pages 293–330. Elsevier, North-Holland, 2001.

[3] J. R. Büchi. On a decision method in restricted second-order arithmetics. In *Proc.of International Congress of Logic, Methodology and Philosophy of Science*, pages 1–12. Stanford University Press, 1962.

[4] C. Dixon, M. Fisher, and M. Reynolds. Execution and Proof in Horn-Clause Temporal Logic. In H. Barringer, M. Fisher, D. Gabbay, and G. Gough, editors, *Advances in Temporal Logic*, volume 16 of *Applied Logic Series*, pages 413–433. Kluwer, 2000. Proceedings the Second International Conference on Temporal Logic (ICTL).

[5] E. A. Emerson. Temporal and Modal Logic. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science: Volume B, Formal Models and Semantics.*, pages 996–1072. Elsevier, 1990.

[6] E. A. Emerson. Automated reasoning about reactive systems. In *Logics for Concurrency: Structures Versus Automata, Proc. of International Workshop*,

volume 1043 of Lecture Notes in Computer Science, pages 41–101. Springer-Verlag, 1996.

[7] M. Fisher. A Resolution Method for Temporal Logic. In *Proc. of the XII International Joint Conference on Artificial Intelligence (IJCAI)*, pages 99–104, 1991.

[8] M. Fisher. A normal form for temporal logic and its application in theorem-proving and execution. *Journal of Logic and Computation*, 7(4):429–456, 1997.

[9] M. Fisher and C. Dixon. Guiding Clausal Temporal Resolution. In H. Barringer, M. Fisher, D. Gabbay, and G. Gough, editors, *Advances in Temporal Logic*, volume 16 of *Applied Logic Series*, pages 167–184. Kluwer, 2000. Proceedings the Second International Conference on Temporal Logic (ICTL).

[10] M. Fisher, C. Dixon, and M. Peim. Clausal temporal resolution. *ACM Transactions on Computational Logic (TOCL)*, 1(2):12–56, 2001.

[11] D. Gabbay, A. Phueli, S. Shelah, and J. Stavi. On the temporal analysis of fairness. In *Proceedings of 7th ACM Symposium on Principles of Programming Languages*, pages 163–173, Las Vegas, Nevada, 1980.

[12] J. Y. Halpern and M. Y. Vardi. Model checking vs. theorem proving - a manifesto. In V. Lifschitz, editor, *Artificial Intelligence and Mathematical Theory of Computation*, pages 151–176. Academic Press, 1991.

[13] R. Kaivola. *Using Automata to Characterise Fixed Point Temporal Logics.* PhD thesis, University of Edinburgh, 1997.

[14] Y. Kesten and A. Pnueli. A complete deductive system for QPTL. In *Proceedings of the 10th Annual IEEE Symposium of Logic in Computer Science*, pages 2–12, 1995.

[15] O. Kupferman and M. Y. Vardi. Model checking of safety properties. In *Computer Aided Verification (CAV' 99), Proc. of Interntational Conference*, volume 1633 of Lecture Notes in Computer Science, pages 172–183. Springer-Verlag, 1999.

[16] O. Lichtenstein. *Decidability, Completeness, and Extensions of Linear-Time Temporal Logic.* PhD thesis, The Weizmann Institute, Rehovot, Israel, 1991.

[17] D. A. Plaisted and S. A. Greenbaum. A structure-preserving clause form translation. *Journal of Symbolic Computation*, 2(3):293–304, 1986.

[18] A. Sistla. *Theoretical issues in the Desing and Verification of Distibuted Systems.* PhD thesis, Harvard University, 1983.

[19] A. Sistla, M. Y. Vardi, and P. Wolper. The complementation problem for Büchi automata with application to temporal logic. *Theoretical Computer Science*, (49):217–237, 1987.

[20] W. Thomas. Languages, automata, and logic. In G. Rozenberg and A. Salomaa, editors, *Handbook of Formal Language Theory*, pages 389–455. Springer-Verlag, 1997.

[21] M. Y. Vardi. An automata-theoretic approach to linear temporal logic. In *Logics for Concurrency: Structures Versus Automata, Proc. of International Workshop*, volume 1043 of Lecture Notes in Computer Science, pages 238–266. Springer-Verlag, 1996.

[22] P. Wolper. Temporal logic can be more expressive. *Information and Control*, 56(1–2):72–99, 1983.