Linear Temporal Logic and Büchi Automata

Bhumika Mittal

Abstract

This paper presents an exposition on Linear Temporal Logic (LTL) and Büchi Automata. Linear Temporal Logic provides a formal framework for expressing properties of infinite sequences, making it essential for reasoning about temporal behaviors in systems. Büchi automata serve as the computational counterpart to LTL, enabling algorithmic verification of temporal properties. We explore the syntax and semantics of LTL, the construction and properties of Büchi automata, and the fundamental connection between these two formalisms through the translation from LTL formulas to Büchi automata.

1 Introduction

Linear Temporal Logic is a modal logic designed to reason about sequences of events over time. Unlike classical propositional logic, which deals with static truth values, LTL allows us to express dynamic properties such as "eventually φ will hold" or "φ will hold until ψ becomes true." This temporal dimension makes LTL particularly well-suited for specifying and verifying properties of reactive systems, concurrent programs, and protocols that exhibit ongoing behavior.

The connection between LTL and automata theory, specifically Büchi automata, is fundamental to the field of formal verification. Büchi automata are finite automata that operate on infinite words, accepting a word if some designated "good" state is visited infinitely often. This acceptance condition naturally captures the infinite nature of temporal properties. The translation from LTL formulas to Büchi automata enables automated model checking, where we can algorithmically verify whether a system satisfies a given temporal specification.

In this paper, we begin with the preliminaries on games on graphs, which provide the mathematical foundation for understanding these concepts. We then develop the syntax and semantics of LTL, explore its expressiveness and equivalences, and examine Büchi automata in detail. Finally, we present the construction that translates any LTL formula into an equivalent Büchi automaton, establishing the deep connection between these two formalisms.

2 Preliminaries: Games on Graphs

Games played on graphs form a powerful tool to reason about various questions arising in Logic and Automata theory. This section introduces the fundamental concepts that form the basis of game-theoretic analysis.

2.1 What is a Game?

The first model of games that we will define are 2-player, zero-sum, turn-based, deterministic, perfect information games.

2.1.1 Players

The term 2-player games means that there are two people that are playing the game. We use various names to refer to the players. Commonly used names are: Eve and Adam, Player 0 and Player 1, Circle and Square corresponding to the graphical representation of games, Even and Odd corresponding to parity based games, Pathfinder and Verifier corresponding to representation using automaton, Min and Max corresponding to quantitative games. We use Eve and Adam to refer to players in qualitative games, and Min and Max to refer to players in quantitative games.

We will also be referring to 1-player games when there is only one player. In stochastic games, we have an additional random player which we refer to as the $\frac{1}{2}$ player. In games involving more than two players, we call them multiplayer games.

2.1.2 Graphs

A (directed) graph is given by a set V of vertices and a set E of edges given by the functions In, Out : $E \rightarrow V$: for an edge e we write $In(e)$ for the incoming vertex and $Out(e)$ for the outgoing vertex. We say that e is an outgoing edge of $In(e)$ and an incoming edge to $Out(e)$. To introduce an edge, it is convenient to write $e = v \rightarrow v'$ to express that $v = In(e)$ and $v' = Out(e)$. A path π is a finite or infinite sequence:

$$\pi = v_0 \rightarrow v_1 \rightarrow v_2 ...$$

We use first(π) to denote the first vertex of the path π and use last(π) to denote the last vertex of the path π . When a path is finite, we say that π starts from first(π) and ends in last(π). We will be using the notation $\pi_{\leq i}$ for the finite path $v_0 \rightarrow v_1 \rightarrow ... \rightarrow v_i$.

We define $Paths(G)$ which denote the set of finite paths in the graph G. To denote paths in a graph G starting from a vertex v, we say $Paths(G, v)$ .We define $Paths_\omega(G)$ which denote the set of infinite paths in a graph G. To denote paths in a graph G starting from a vertex v, we say $Paths_\omega(G, v)$.

We will use standard terminology associated with graphs:

- Successor: a vertex v' is a successor of v if $\exists$ $e \in E$ such that $In(e) = v$ and $Out(e) = v'$.
- Predecessor: a vertex v is a predecessor of v' , a vertex v' is reachable from v if $\exists$ a path starting from v that ends in v'.
- Outdegree: for a vertex v, the outdegree tells us the number of edges leaving that vertex.
- Indegree: for a vertex v, the indegree tells us the number of edges entering that vertex.
- Simple Path: a path where no vertices is repeated or more formally, a path π where $\pi = v_0 \rightarrow v_1 \rightarrow v_2 ... \rightarrow v_i$ such that each of the vertices are unique.
- Cycle: a path where the the first and last vertex coincide or more formally, a path π where $\pi = v_0 \rightarrow v_1 \rightarrow v_2 ... \rightarrow v_i$ such that $v_1 = v_i$.
- Simple cycle: a cycle that strictly does not contain another cycle.
- Self loop: an edge from a vertex to itself.
- Sink: a vertex with only self loops as outgoing edges.

2.1.3 Arenas

The arena is the place where the game is played, they are also called game structures or game graphs. Since, we have defined our model to be a turn-based game, wherein between two consecutive moves of a player, there needs to a be a move made by the opponent. In this turn-based setting, we divide the set of vertices into vertices controlled by each player. Since we are interested in 2-player games, we can define the set of vertices as $V = V_{Eve} \uplus V_{Adam}$ where V_{Eve} is the set of vertices controlled by Eve and V_{Adam} is the set of vertices controlled by Adam. We use circles to represent vertices belonging to V_{Eve} and squares to represent squares belonging to V_{Adam}.

An arena is given by a graph and the sets V_{Eve} and V_{Adam}. In the context of games, vertices are also referred to as positions. We call an arena finite when there are finitely many vertices and edges. An important assumption called *perfect information* means that the players can see everything about how the game is played out, in particular they see the other players' moves. We also assume that all vertices have an outgoing edge.

2.1.4 Play

The interaction between the two players consists in moving a token on the vertices of the arena. Initially, the token is on some vertex, lets call this vertex v. We have learnt that each player controls a set of vertices. WLOG we can say Player 1 controls v and they choose some edge e such that $e = v \to v'$ and pushes the token along this edge. Now, the token is at vertex v'. The outcome of this interaction is a sequence of vertices that were traversed by the token: this is a path. Consequently, a path can also be called a *play* . Similar to paths, plays can be finite or infinite.

2.1.5 Strategies

A strategy (or a *policy*) is the full description of a players moves in all situations. Formally, we define strategy as a function that maps finite plays to edges. We use σ to denote a strategy.

$$\sigma : Paths \to E$$

Notation: We use σ to denote strategies of Eve and Max and τ to denote strategies of Adam and Min.

We say that a play $\pi = v_0 \to v_1 \to v_2 ...$ is consistent with a strategy σ of Eve if $\forall i$ such that $v_i \in V_{Eve}$ we have $\sigma(\pi_{\leq i}) = v_i \to v_{i+1}$ We can use the same idea to represent Adam's strategies as well. Once an initial vertex v and two strategies σ and τ have been fixed, there exists a unique infinite play starting from v and consistent with both strategies, it is written with the following notation $\pi^v_{\sigma,\tau}$. Note that the fact that it is infinite follows from our assumption that all vertices have an outgoing edge.

2.1.6 Winning Conditions

Condition or winning conditions is what Eve wants to achieve. There are two types of winning conditions based on the two types of game qualitative and quantitative conditions.

Qualitative condition is defined as $W \subseteq Paths_\omega$ where W is a set of paths such that if Eve takes a path π such that $\pi \in W$, Eve is taking a winning path (or play).

Quantitative condition is defined as $f : Paths_\omega \to R \cup \pm\infty$ where f assigns a real value to a path that Eve is taking and we want to maiximize the value of this path.

2.1.7 Objectives

2.1.8 Coloring Functions

Colouring function as the name suggests is a mapping between labelling edges of a graph and a set of colours. $c : E \to C$

We can extend the function c to also map a *play* or a path to a sequence of colours. $c : Paths_\omega \to C^\omega$

$$c(e_0e_1...) = c(e_0)c(e_1)...$$

A colouring function along with an objective induces a condition. Based on the type of objective: qualitative or quantitative, corresponding condition is induced. When we have a qualitative objective Ω where $\Omega(c) = \{\pi \in Paths_\omega : c(\pi) \in \Omega\}$and a colouring function c , we have the corresponding qualitative condition given by $\Omega(c)(\pi) = c(\pi) \in \Omega$.

Similarly, for a quantitative objective Φ where $\Phi : C^\omega \to R \cup \{\pm\infty\}$ and a colouring function c, we have the corresponding quantitative condition given by $\Phi(c)$.

$$\Phi(c)(\pi) = \Phi(c(\pi))$$

2.1.9 Games

1. A graph is a tuple $G = (V, E, In, Out)$ where is E is a set of edges and V is a set of vertices and $In, Out : E \to V$ defines the incoming and outgoing vertices of edges.
2. An arena is a tuple $A(G, V_{Eve}, V_{Adam})$ where G is a graph over the set of vertices V and $V = V_{Eve} \uplus V_{Adam}$. For quantitative games, we define $V = V_{Min} \uplus V_{Max}$.
3. A colouring function is a function $c : E \to C$ where C is a set of colours.

3 Linear Temporal Logic and Büchi Automata

3.1 Linear Temporal Logic

Linear Time Temporal Logic (LTL) is a way to describe infinite sequences, where each step in the sequence represents a specific state or situation. In LTL, we can express statements about the future of paths, such as a condition that eventually becomes true or remains true until another condition holds.

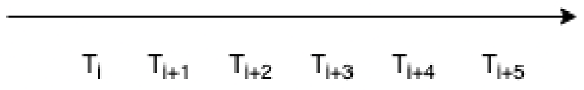

Figure 1: An LTL Trace

The reason we use LTL is that we wish to make statement about moves and positions that we make in the future. We should think of time as an arrow and future time as an infinite number of discrete points at which moves and positions can occur. The moment at which we currently exist can be thought of as T_i. Future moments are $T_{i+k}, k \geq 1$.

3.1.1 Syntax

We fix a countable set of atomic propositions $\mathcal{P} = \{p_0, p_1, \ldots\}$. Then Φ, the set of LTL formulas, is defined inductively as follows:

- Every member of $\mathcal{P}$ belongs to Φ.
- If α and β are formulas in Φ, then so are $\neg\alpha$, $\alpha \vee \beta$, $O\alpha$ and $\alpha\mathcal{U}\beta$.

The connectives $\neg$ and $\vee$ correspond to the usual Boolean connectives "not" and "or," respectively. The modality O is to be read as 'next', while the binary modality $\mathcal{U}$ is to be read as 'until'. Thus, $O\alpha$ is "next α", while $\alpha\mathcal{U}\beta$ is "α until β."

3.1.2 Semantics

A *model* is a function $M : \mathbb{N}_0 \to 2^{\mathcal{P}}$. What this means is that P is the list of all the propositions that we use. $2^{\mathcal{P}}$ is the power set of $\mathcal{P}$. At any given moment some subset of $\mathcal{P}$ is true. All such possible truth assignments belong to $2^{\mathcal{P}}$. The function M describes how the truth of atomic propositions changes as time progresses.

We write $M, i \models \alpha$ to denote that "α is true at time instant i in the model M". This notion is defined inductively, according to the structure of α:

- $M, i \models p$, where $p \in \mathcal{P}$, iff $p \in M(i)$.
- $M, i \models \neg\alpha$ iff $M, i \not\models \alpha$.
- $M, i \models \alpha \vee \beta$ iff $M, i \models \alpha$ or $M, i \models \beta$.

  Uptil now we have used conventions and operators that might already have been familiar to the reader.

- $M, i \models O\alpha$ iff $M, i+1 \models \alpha$.

  This means that we are at a given moment i and at the next moment $(i+1)$, the statement α will be true.

- $M, i \models \alpha\mathcal{U}\beta$ iff there exists $k \geq i$ such that $M, k \models \beta$ and for all j such that $i \leq j < k$, $M, j \models \alpha$.

  What we mean to say here is that there is some future moment in time when β holds true and α shall hold true for every moment until that moment.

A formula α is said to be satisfiable if there exist a model M and an instant i such that $M, i \models \alpha$.

3.1.3 Derived Connectives

Let us introduce the constants $\top$ and $\bot$ representing "true" and "false." We can write $\top$ as, for example, $p_0 \vee \neg p_0$ and $\bot$ as $\neg\top$. Based on $\alpha\mathcal{U}\beta$, we can introduce the modality $\Diamond$ (possibly), read as "eventually", and the modality $\Box$ (necessarily), read as "henceforth". We can write $\Diamond\alpha$ for $\top\mathcal{U}\alpha$ and $\Box\alpha$ for $\neg\Diamond\neg\alpha$. Formally, these can be defined as follows; for a model M,

- $M, i \models \Diamond\alpha$ iff there exists $k \geq i$ such that $M, k \models \alpha$.
- $M, i \models \Box\alpha$ iff for all $k \geq i$, $M, k \models \alpha$.

Lemma 3.1. *LTL with the 'Until' ($\mathcal{U}$) and 'Next' ($\mathcal{X}$) operators is strictly more expressive than LTL restricted to only the 'Globally' ($\Box$), 'Finally' ($\Diamond$), and 'Next' ($\mathcal{X}$) operators.*

Proof. Let's define $\mathsf{LTL}(\mathcal{U}, \mathcal{X})$ as LTL with the 'Until' ($\mathcal{U}$) and 'Next' ($\mathcal{X}$) operators and $\mathsf{LTL}(\Box, \Diamond, \mathcal{X})$ as LTL restricted to only the 'Globally' ($\Box$), 'Finally' ($\Diamond$), and 'Next' ($\mathcal{X}$) operators. We need to show that $\mathsf{LTL}(\mathcal{U}, \mathcal{X})$ is strictly more expressive than $\mathsf{LTL}(\Box, \Diamond, \mathcal{X})$.

Let true be denoted as $\top$ and false as $\bot$. We know that 'Globally' ($\Box$) and 'Finally' ($\Diamond$) can be expressed using 'Until' ($\mathcal{U}$) as follows:

$$\Diamond\phi \equiv \top\mathcal{U}\phi$$
$$\Box\phi \equiv \neg\Diamond\neg\phi$$

This means 'Until' ($\mathcal{U}$) and 'Next' ($\mathcal{X}$) operators can express everything that 'Globally' ($\Box$), 'Finally' ($\Diamond$), and 'Next' ($\mathcal{X}$) operators can express. To show strict expressiveness, we need to show that 'Until' and 'Next' can express something that 'Globally', 'Finally' and 'Next' cannot.

Let $\mathcal{P} = \{p_0, p_1, \cdots\}$ be a countable set of atomic propositions. $M : \mathbb{N} \to 2^{\mathcal{P}}$ is a model. We denote $M, i \models \alpha$ if the formula α is true in the model M at time i. Let us consider the following property P:

$$\exists k \geq i \ni M, k \models \beta \wedge \forall j \in [i, k), M, j \models \alpha$$

This property states that α holds until β holds. Trivally, P can be expressed using the 'Until' operator as $\alpha\mathcal{U}\beta$ in $\mathsf{LTL}(\mathcal{U}, \mathcal{X})$. Now, let's show that P cannot be expressed in $\mathsf{LTL}(\Box, \Diamond, \mathcal{X})$.

Assume there exists an $\mathsf{LTL}(\Box, \Diamond, \mathcal{X})$ formula φ equivalent to $\alpha\mathcal{U}\beta$. Let n be the maximum number of nested $\mathcal{X}$ operators in φ.

Let M and M' be the following two models in $\mathsf{LTL}(\Box, \Diamond, \mathcal{X})$.

- Model M: For all $i \leq n$, $M, i \models \alpha$, and $M, n+1 \models \beta$. Thus, $M, 0 \models \alpha\mathcal{U}\beta$.
- Model M': For all $i \leq n-1$, $M', i \models \alpha$; $M', n \models \neg\alpha$; and $M', n+1 \models \beta$. Here, $M', 0 \not\models \alpha\mathcal{U}\beta$, as α fails at n before β holds at $n+1$.

Claim: M and M' are indistinguishable to φ at time 0, i.e., $M, 0 \models \varphi \iff M', 0 \models \varphi$.

By structural induction on φ, we can show that for any subformula ψ of φ and any $k \leq n$, $M, k \models \psi \iff M', k \models \psi$.

Atomic Formulas: For $p \in \mathcal{P}$, $M, k \models p$ iff $p \in M(k)$. Since M and M' agree on all positions $\leq n$, the claim holds for $k \leq n$.

Boolean combinations: If $\psi = \psi_1 \wedge \psi_2$ or $\psi = \neg\psi_1$, then $M, k \models \psi$ iff $M, k \models \psi_1$ and $M, k \models \psi_2$ or $M, k \not\models \psi_1$, respectively. By induction, M and M' agree on the truth of ψ_1 and ψ_2 at $k \leq n$, so the claim holds for ψ.

Temporal operators: We can show the claim holds for the temporal operators individually as follows:

- For $\psi = \mathcal{X}\psi_1$: By definition, $M, k \models \mathcal{X}\psi_1$ iff $M, k+1 \models \psi_1$. Since $k \leq n$, $k+1 \leq n+1$. For $k < n$, M and M' agree at $k+1$; for $k = n$, $M, n+1 \models \psi_1$ iff $M', n+1 \models \psi_1$. However, φ contains at most n nested $\mathcal{X}$ operators, so ψ_1 cannot reference positions beyond n. Thus, M and M' agree at all positions $\leq n$.
- For $\psi = \Box\psi_1$: $M, k \models \Box\psi_1$ iff $\forall j \geq k, M, j \models \psi_1$. However, ψ_1 can only reference positions up to $k+n$ (due to the n-step limit of $\mathcal{X}$). Beyond n, M' diverges, but ψ_1 cannot "observe" these positions. Thus, the evaluation of $\Box\psi_1$ depends only on positions $\leq n$, where M and M' agree.

- For $\psi = \Diamond\psi_1$: Similar to $\Box$, $\Diamond\psi_1$ depends on the existence of some $j \geq k$ where $M, j \models \psi_1$. Within the n-steps, M and M' agree, so $\Diamond\psi_1$ evaluates equivalently.

Thus, $M, 0 \models \varphi \iff M', 0 \models \varphi$. However, by construction:

$$M, 0 \models \alpha\mathcal{U}\beta \quad \text{and} \quad M', 0 \not\models \alpha\mathcal{U}\beta.$$

This contradicts the assumption that φ is equivalent to $\alpha\mathcal{U}\beta$. Therefore, no such φ exists in $\mathsf{LTL}(\Box, \Diamond, \mathcal{X})$.

Thus, $\mathsf{LTL}(\mathcal{U}, \mathcal{X})$ is strictly more expressive than $\mathsf{LTL}(\Box, \Diamond, \mathcal{X})$. $\Box$

3.1.4 Equivalences

Let φ, ψ, and ρ be LTL formulas. The following list lists some useful equivalences that extend standard equivalences among the usual logical operators.

1. Distributivity

$$\begin{aligned}
X(\varphi \vee \psi) &\equiv (X\varphi) \vee (X\psi) \quad \text{(Since } X \text{ applies to both operands independently)}\\
X(\varphi \wedge \psi) &\equiv (X\varphi) \wedge (X\psi) \quad \text{(Similarly, } X \text{ distributes over } \wedge)\\
X(\varphi\mathcal{U}\psi) &\equiv (X\varphi)\mathcal{U}(X\psi) \quad \text{(By induction over time steps)}\\
\Diamond(\varphi \vee \psi) &\equiv (\Diamond\varphi) \vee (\Diamond\psi) \quad \text{(Since at least one of } \varphi \text{ or } \psi \text{ must be true in the future)}\\
\Box(\varphi \vee \psi) &\equiv (\Box\varphi) \wedge (\Box\psi) \quad \text{(Since } \varphi \text{ and } \psi \text{ must both hold in all future states)}\\
\rho\mathcal{U}(\varphi \vee \psi) &\equiv \varphi\mathcal{U}\psi \vee (\psi\mathcal{U}\varphi) \quad \text{(Rewriting } \mathcal{U} \text{ in terms of disjunction)}\\
(\varphi \wedge \psi)\mathcal{U}\rho &\equiv (\varphi\mathcal{U}\rho) \wedge (\psi\mathcal{U}\rho) \quad \text{(By distributivity of } \mathcal{U} \text{ over } \wedge)
\end{aligned}$$

2. Negation Propagation

$$\begin{aligned}
\neg X\varphi &\equiv X\neg\varphi \quad \text{(Since negation propagates through next-time operator)}\\
\neg\Diamond\varphi &\equiv \Box\neg\varphi \quad \text{(Since } \Diamond \text{ and } \Box \text{ are dual)}\\
\neg\Box\varphi &\equiv \Diamond\neg\varphi \quad \text{(By duality of always and eventually)}
\end{aligned}$$

3. Special Temporal Properties

$$\begin{aligned}
&\Diamond\varphi \equiv \Diamond\Diamond\varphi \quad \text{(Since if something happens eventually, it also happens eventually eventually)}\\
&\Box\varphi \equiv \Box\Box\varphi \quad \text{(Since if something is always true, it is always always true)}\\
&\Box\varphi \equiv \varphi \wedge X(\Box\varphi) \quad \text{(Induction principle for always operator)}\\
&\Diamond\varphi \equiv \varphi \vee X(\Diamond\varphi) \quad \text{(Induction principle for eventually operator)}
\end{aligned}$$

3.1.5 Examples

Here are some examples of the kinds of assertions we can make in temporal logic.

- The formula $\Diamond\Box\alpha$ says that α is eventually a "stable property" of the system—
$M, i \models \Diamond\Box\alpha \iff \exists j \geq i, \forall k \geq j, M, k \models \alpha$.
- On the other hand, $\Box\Diamond\beta$ asserts that β holds infinitely often.

3.2 Büchi Automata

A Büchi automaton is a non-deterministic finite-state automaton which takes infinite words as input. A word is accepted if the automaton goes through some designated "good" states infinitely often while reading it.

We begin with some notation for infinite words. Let Σ be a finite alphabet. An infinite word $\alpha \in \Sigma^\omega$ is an infinite sequence of symbols from Σ. We shall represent α as a function $\alpha : \mathbb{N}_0 \to \Sigma$, where $\mathbb{N}_0$ is the set $\{0, 1, 2, \dots\}$ of natural numbers. Thus, $\alpha(i)$ denotes the letter occurring at the ith position.

In general, if S is a set and σ an infinite sequence of symbols over S—in other words, $\sigma : \mathbb{N}_0 \to S$—then $\inf(\sigma)$ denotes the set of symbols from S which occur infinitely often in σ. Formally,

$$\inf(\sigma) = \{s \in S \mid \exists^\omega n \in \mathbb{N}_0 : \sigma(n) = s\},$$

where $\exists^\omega$ denotes the quantifier "there exist infinitely many".

3.2.1 Automata

An automaton is defined as a triple $A = (S, \to, S_{\text{in}})$, where:

- S is the set of states,
- $S_{\text{in}} \subseteq S$ is the set of initial states,
- $\to \subseteq S \times \Sigma \times S$ represents the transition relation.

A transition is typically written as $s \xrightarrow{a} s'$ to indicate that $(s, a, s') \in \to$. An automaton is considered deterministic if S_{in} contains exactly one state and the transition relation $\to$ is a function from $S \times \Sigma$ to S. We will primarily deal with non-deterministic automata.

A Büchi automaton is defined as a pair (A, G), where $A = (S, \to, S_{\text{in}})$ is an automaton, and $G \subseteq S$ is a set of designated "good" states. The automaton (A, G) accepts an infinite input word $\alpha : \mathbb{N}_0 \to \Sigma$ if there exists a run ρ of A on α such that $\inf(\rho) \cap G \neq \emptyset$. Since G is finite, at least one state $g \in G$ must appear infinitely often in ρ. In graphical terms, viewing the state space of a Büchi automaton as a directed graph, an accepting run follows an infinite path that:

- Begins at some initial state $s \in S_{\text{in}}$,
- Reaches a "good" state $g \in G$,
- Eventually loops back to g infinitely often.

The language recognized by (A, G), denoted as $L(A, G)$, consists of all infinite words accepted by (A, G). A set $L \subseteq \Sigma^\omega$ is called Büchi-recognizable if there exists a Büchi automaton (A, G) such that $L = L(A, G)$.

3.2.2 Runs

Given an automaton $A = (S, \to, S_{\text{in}})$ and an input word $\alpha : \mathbb{N}_0 \to \Sigma$, a run of A on α is an infinite sequence $\rho : \mathbb{N}_0 \to S$ that satisfies:

- The initial state condition: $\rho(0) \in S_{\text{in}}$,
- The transition condition: for all $i \in \mathbb{N}_0$, $\rho(i) \xrightarrow{\alpha(i)} \rho(i+1)$.

Thus, a run represents a valid sequence of states traversed by the automaton while processing the input. Due to non-determinism, a single input can yield multiple possible runs. Moreover, some inputs may not admit any valid runs if they reach a state where no further transitions are possible. In contrast, a deterministic automaton ensures that every input is associated with exactly one unique run.

3.2.3 Closure properties

The class of Büchi-recognizable languages is closed under Boolean operations. This means that if L_1 and L_2 are Büchi-recognizable languages, then the following languages are also Büchi-recognizable:

Union To demonstrate closure under union, suppose we have two Büchi automata, (A_1, G_1) and (A_2, G_2). We can construct an automaton (A, G) such that

$$L(A, G) = L(A_1, G_1) \cup L(A_2, G_2).$$

Here, A is defined as the disjoint union of A_1 and A_2. Since multiple initial states are allowed, the initial states from both automata are preserved. If a run in A begins in an initial state from A_1, it remains within the states of A_1, and similarly for A_2. Therefore, we define the set of good states G as the union $G_1 \cup G_2$.

Intersection For the intersection of Büchi automata, consider constructing a product automaton that processes the input simultaneously with both automata. For finite words, acceptance requires that each component reaches an accepting state at the end of the input. However, for infinite inputs, both components must visit their good states infinitely often. A complication arises since the visits to good states might not be synchronized between the two components. The key idea is to monitor an infinite subsequence of visits to good states in each automaton. Initially, we wait for the first component to enter a good state; upon its occurrence, we shift our attention to the second component, waiting for it to reach a good state, and then switch back, continuing this alternating process. This ensures that both components visit their respective good states infinitely often.

Formally, let (A_1, G_1) and (A_2, G_2) be two Büchi automata with $A_i = (S_i, \to_i, S_i^{\text{in}})$ for $i = 1, 2$. We define the product automaton (A, G) as follows:

- The state space is
$$S = S_1 \times S_2 \times \{1, 2\}.$$
- The transition relation $\to$ is defined by:
  - $(s_1, s_2, 1) \xrightarrow{a} (s_1', s_2', 1)$ if $s_1 \xrightarrow{a}_1 s_1'$, $s_2 \xrightarrow{a}_2 s_2'$, and $s_1 \notin G_1$.
  - $(s_1, s_2, 1) \xrightarrow{a} (s_1', s_2', 2)$ if $s_1 \xrightarrow{a}_1 s_1'$, $s_2 \xrightarrow{a}_2 s_2'$, and $s_1 \in G_1$.
  - $(s_1, s_2, 2) \xrightarrow{a} (s_1', s_2', 2)$ if $s_1 \xrightarrow{a}_1 s_1'$, $s_2 \xrightarrow{a}_2 s_2'$, and $s_2 \notin G_2$.
  - $(s_1, s_2, 2) \xrightarrow{a} (s_1', s_2', 1)$ if $s_1 \xrightarrow{a}_1 s_1'$, $s_2 \xrightarrow{a}_2 s_2'$, and $s_2 \in G_2$.
- The set of initial states is
$$S^{\text{in}} = \{(s_1, s_2, 1) \mid s_1 \in S_1^{\text{in}},\ s_2 \in S_2^{\text{in}}\}.$$
- The set of good states is defined as
$$G = S_1 \times G_2 \times \{2\}.$$
(Alternatively, one could choose $G = G_1 \times S_2 \times \{1\}$.)

In this automaton, each state carries an extra tag indicating which component's good state is currently being monitored. An input is accepted if the tag alternates infinitely often, ensuring that both components visit their good states infinitely often.

Emptiness In many applications, it is crucial to determine whether a Büchi automaton accepts any words at all. Recall that an accepting run must begin in an initial state, eventually reach a good state, and then repeatedly return to that good state infinitely often.

If we ignore the labels on the transitions, the state space of a Büchi automaton (A, G) can be viewed as a directed graph $G_A = (V_A, E_A)$, where $V_A = S$ and there is an edge $(s, s') \in E_A$ if there exists some $a \in \Sigma$ such that $s \xrightarrow{a} s'$. A subset $X \subseteq V_A$ forms a strongly connected component (SCC) if for every pair of vertices $v, v' \in X$, there exists a path from v to v'.

Thus, the language $L(A, G)$ is non-empty if and only if there exists a strongly connected component X in G_A that is reachable from an initial state and contains at least one good state $g \in G$.

3.2.4 Deterministic vs Non-deterministic

An *infinite word* is an infinite-length sequence (specifically, an ω-length sequence) of symbols, and an ω-language is simply a set of such infinite words. The ω-regular languages extend the concept of regular languages to these infinite words.

It turns out that the class of deterministic Büchi automata is not expressive enough to capture all ω-regular languages. In particular, there is no deterministic Büchi automaton that recognizes the language

$$(0 \cup 1)^* 0^\omega,$$

which consists exactly of those infinite words in which the symbol 1 occurs only finitely many times.

To see why, assume for contradiction that there exists a deterministic Büchi automaton A with a set of final states F that recognizes $(0 \cup 1)^* 0^\omega$. Notice that the word 0^ω (an infinite sequence of 0s) is accepted by A. Therefore, after reading a finite prefix of 0^ω, say after the i_0th symbol, A enters a state in F.

Now, consider the infinite word

$$0^{i_0} 1 0^\omega.$$

Since this word is also accepted by A, there must be some i_1 such that after reading the prefix $0^{i_0} 1 0^{i_1}$, the automaton reaches a state in F. Repeating this construction, one obtains the infinite word

$$0^{i_0} 1 0^{i_1} 1 0^{i_2} \ldots,$$

which causes A to visit states in F infinitely often. However, this word does not belong to $(0 \cup 1)^* 0^\omega$ since it contains infinitely many 1s. This contradiction shows that no deterministic Büchi automaton can recognize the language $(0 \cup 1)^* 0^\omega$.

3.2.5 Generalized Büchi automata

When relating temporal logic to Büchi automata, it is often useful to consider a variant with a more complex acceptance condition. A *generalized Büchi automaton* is given as a structure $(A, G_1, G_2, \ldots, G_k)$, where $A = (S, \to, S_{\text{in}})$ is an automaton and, for each $i \in \{1, 2, \ldots, k\}$, $G_i \subseteq S$.

An infinite input word α is accepted by the automaton $(A, G_1, G_2, \dots, G_k)$ if there exists a run ρ of A on α such that for every $i \in \{1, 2, \dots, k\}$ the set of states visited infinitely often satisfies

$$\inf(\rho) \cap G_i \neq \emptyset.$$

The language recognized by $(A, G_1, G_2, \dots, G_k)$, denoted by

$$L(A, G_1, G_2, \dots, G_k),$$

consists of all infinite words accepted by the automaton.

The following proposition immediately follows from the definition:

Proposition 3.1. *Let $(A, G_1, G_2, \dots, G_k)$ be a generalized Büchi automaton. Then*

$$L(A, G_1, G_2, \dots, G_k) = \bigcap_{i=1}^{k} L(A, G_i).$$

In other words, every language recognized by a generalized Büchi automaton is also Büchi-recognizable.

3.3 From LTL to Büchi Automata

Before we formally get into the proof let us consider how we could go about proving that any LTL can be written as a Buechi automata. We could say that if every operation on our propositions we represent using LTL can be converted into an automata, then the entire LTL formula can be converted into a buechi automata. Let us now show this step-by-step:

Definition 3.1. Closure Let α be an LTL formula. Then CL'(α) is the smallest set of formulas such that

- $\alpha \in \mathrm{CL}'(\alpha)$
- If $\neg\beta \in \mathrm{CL}'(\alpha)$, then $\beta\mathrm{CL}'(\alpha)$
- If $\beta \vee \gamma \in \mathrm{CL}'(\alpha)$, then $\beta, \gamma \in \mathrm{CL}'(\alpha)$.
- If $O\beta \in \mathrm{CL}'(\alpha)$, then $\beta \in \mathrm{CL}'(\alpha)$.
- If $\beta U \gamma \in \mathrm{CL}'(\alpha)$, then $\beta, \gamma, O(\beta U \gamma) \in \mathrm{CL}'(\alpha)$.

Finally, $\mathrm{CL}(\alpha) = \mathrm{CL}'(\alpha) \cup \{\neg\beta \mid \beta \in \mathrm{CL}'(\alpha)\}$, where we identify $\neg\neg\beta$ with β.

Definition 3.2. Atoms Let α be a formula. Then $A \subseteq \mathrm{CL}(\alpha)$ is an atom if:

- $\forall\beta \in \mathrm{CL}(\alpha)$, $\beta \in A \iff \neg\beta \notin A$.
- $\forall\beta \vee \gamma \in \mathrm{CL}(\alpha)$, $\beta \vee \gamma \in A \iff \beta \in A$ or $\gamma \in A$.
- $\forall\beta U \gamma \in \mathrm{CL}(\alpha)$, $\beta U \gamma \in A \iff \gamma \in A$ or $\beta, O(\beta U \gamma) \in A$.

Let $\mathcal{AT}$ be the set of all atoms of α.

Now we must show that if all $\alpha \in \mathcal{AT}$ can be converted into an automata, then α can be converted into an buechi automatata

Definition 3.3. Voc(α), denotes the "vocabulary" of α, is a subset of $\mathcal{P}$ which is mentioned in Voc(α). If we consider a formula to be operations on a set of propositions, Voc(α) is this set.

Constructing a Büchi automaton for α We first construct an automaton $A_\alpha = (S, \rightarrow, S_{\text{in}})$ over the alphabet $2^{\text{Voc}(\alpha)}$, where:

- $S = \mathcal{AT}$
- Let $A, B \in \mathcal{AT}$ and $P \subseteq \text{Voc}(\alpha)$. Then $A \xrightarrow{P} B$ iff the following hold:
  - $P = A \cap \text{Voc}(\alpha)$.
  - For all $O_\beta \in \text{CL}(\alpha)$, $O_\beta \in A$ iff $\beta \in B$.
- $S_{\text{in}} = \{A \in \mathcal{AT} \mid \alpha \in A\}$.

Let $\{\beta_1 U \gamma_1, \beta_2 U \gamma_2, \ldots, \beta_k U \gamma_k\}$ be the set of until formulas which appear in $\text{CL}(\alpha)$. We add a generalized Büchi acceptance condition $(G_1, G_2, \ldots, G_k)$ where for each i,

$$G_i = \{A \in \mathcal{AT} \mid \beta_i U \gamma_i \notin A \text{ or } \gamma_i \in A\}.$$

Now we only have left to prove that $\forall \beta \in \text{CL}(\alpha)$ and $\forall i \in \mathbb{N}_0$, we show that $M, i \models \beta$ iff $\beta \in A_i$. We do this based on our definition of atoms, as we can prove that each operation can be converted to an automata.

Definition 3.4. Let φ be the set of formulas of LTL and let α be a formula in φ.

Theorem 3.2. *Let M be an infinite word over $2^{Voc(\alpha)}$. Then $M \in L(\mathcal{A}_\alpha, G_1, G_2, \cdots, G_k)$ iff $M, 0 \models \alpha$.*

Proof ($\Rightarrow$) Let $M = P_0 P_1 \ldots$ be an infinite word over $\text{Voc}(\alpha)$ which is accepted by $(A_\alpha, G_1, G_2, \ldots, G_k)$. Let $A_0 A_1 \ldots$ be an accepting run of A_α on M. For all $\beta \in \text{CL}(\alpha)$ and for all $i \in \mathbb{N}_0$, we show that $M, i \models \beta$ iff $\beta \in A_i$.

The proof is by induction on the structure of β.

- $\beta = p \in P$

  Then $M, i \models p$ iff $p \in P_i$ iff $p \in A_i$.

- $\beta = \neg\gamma$

  Then $M, i \models \beta$ iff $M, i \not\models \gamma$ iff (by the induction hypothesis) $\gamma \notin A_i$ iff $\beta \in A_i$ (by the definition of an atom).

- $\beta = \gamma \vee \delta$

  Then $M, i \models \beta$ iff $(M, i \models \gamma$ or $M, i \models \delta)$ iff—by the induction hypothesis—$(\gamma \in A_i$ or $\delta \in A_i)$ iff—by the definition of an atom—$\gamma \vee \delta \in A_i$.

- $\beta = O\gamma$

  Then $M, i \models \beta$ iff $M, i + 1 \models \gamma$ iff (by the induction hypothesis) $\gamma \in A_{i+1}$ iff (by the definition of A_i, $P_i \rightarrow A_{i+1}$) $O\gamma = \beta \in A_i$.

- $\beta = \gamma \mathcal{U} \delta$

  Suppose that $M, i \models \beta$. We must show that $\beta \in A_i$. By the semantics of the modality U, there exists $k \geq i$ such that $M, k \models \delta$ and for all j, $i \leq j < k$, $M, j \models \gamma$. We show that $\beta \in A_i$ by a second induction on $k - i$.

  Base case: $(k - i = 0)$

  Then $k = i$ and $M, i \models \delta$. By the main induction hypothesis, $\delta \in A_i$, whence, from the definition of atoms, $\gamma \mathcal{U} \delta = \beta \in A_i$.

  Induction step: $(k - i = \ell > 0)$

By the semantics of the modality U, $M, i \models \gamma$ and $M, i{+}1 \models \gamma U \delta$. Then, by the secondary induction hypothesis, $\gamma \mathcal{U} \delta \in A_{i+1}$. From the definition of A_i, $P_i \rightarrow A_{i+1}$, we then have $O(\gamma \mathcal{U} \delta) \in A_i$ (recall that if $\gamma \mathcal{U} \delta \in \mathrm{CL}(\alpha)$ then $O(\gamma \mathcal{U} \delta) \in \mathrm{CL}(\alpha)$ as well). By the main induction hypothesis, we also have $\gamma \in A_i$. Combining these facts, from the definition of an atom, $\gamma \mathcal{U} \delta = \beta \in A_i$.

Conversely, suppose $\beta \in A_i$. We must show that $M, i \models \beta$. Recall that we have indexed the until formulas in $\mathrm{CL}(\alpha)$ by $1, 2, \ldots, k$. Let m be the index of β.

Since $A_0 A_1 \ldots$ is an accepting run of $(A_\alpha, G_1, G_2, \ldots, G_k)$, there must exist a $k \geq i$ such that $A_k \in G_m$. Choose the least such k. Once again, we do a second induction on $k - i$ to show that $M, i \models \beta$.

Base case: $(k - i = 0)$

If $k = i$, then $A_i \in G_m$. Since $\gamma \mathcal{U} \delta \in A_i$, the only way for A_i to be in G_m is for δ to also belong to A_i. Then, by the main induction hypothesis, $M, i \models \delta$, whereby $M, i \models \gamma \mathcal{U} \delta$ as well.

Induction step: $(k - i = \ell > 0)$

Since $A_i \notin G_m$, $\delta \notin A_i$. From the definition of atoms, both γ and $O(\gamma \mathcal{U} \delta)$ must be in A_i. Since $A_i P_i \rightarrow A_{i+1}$, it must be the case that $\gamma U \delta \in A_{i+1}$. By the secondary induction hypothesis, $M, i+1 \models \gamma \mathcal{U} \delta$ while by the main induction hypothesis $M, i \models \gamma$. From the semantics of the modality $\mathcal{U}$, it then follows that $M, i \models \gamma \mathcal{U} \delta$ as well.

(⇐) Suppose that $M = P_0 P_1 \ldots$ such that $M, 0 \models \alpha$. We have to show that $M \in L(A_\alpha, G_1, G_2, \ldots, G_k)$. In other words, we have to exhibit an accepting run of the automaton on M.

For $i \in \mathbb{N}_0$, define A_i to be the set $\{\beta \in \mathrm{CL}(\alpha) \mid M, i \models \beta\}$. It is easy to verify that each A_i is an atom. We can also verify that each adjacent pair of atoms A_i and A_{i+1} satisfy the conditions specified for the existence of an arrow $A_i P_i \rightarrow A_{i+1}$. Finally, since $M, 0 \models \alpha$, we have $\alpha \in A_0$, so A_0 is an initial state in A_α. From all this, it follows that $A_0 A_1 \ldots$ is a run of the automaton on M.

To check that it is an accepting run, we have to verify that each good set G_m is met infinitely often. Suppose this is not the case—i.e., for some G_m and some index $k \in \mathbb{N}_0$, for all $j \geq k$, $A_j \notin G_m$. In other words, for all $j \geq k$, the mth until formula in $\mathrm{CL}(\alpha)$, $\gamma_m U \delta_m$, belongs to A_j and $\delta_m \notin A_j$. This leads to a contradiction, proving our claim. □