Saarland University
Faculty of Mathematics and Computer Science (MI)
Department of Computer Science

Bachelor Thesis

Provable State-Space Minimization of Büchi Automata arising from LTL Specifications

submitted by
Peter Wita

submitted on
December 1st, 2016

Supervisor
Prof. Bernd Finkbeiner, Ph.D.

Advisor
Felix Klein, M.Sc.

Reviewers
Prof. Bernd Finkbeiner, Ph.D.
Swen Jacobs, Ph.D.

Eidesstattliche Erklärung

Ich erkläre hiermit an Eides Statt, dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel verwendet habe.

Statement in Lieu of an Oath

I hereby confirm that I have written this thesis on my own and that I have not used any other media or materials than the ones referred to in this thesis.

Einverständniserklärung

Ich bin damit einverstanden, dass meine (bestandene) Arbeit in beiden Versionen in die Bibliothek der Informatik aufgenommen und damit veröffentlicht wird.

Declaration of Consent

I agree to make both versions of my thesis (with a passing grade) accessible to the public by having them added to the library of the Computer Science Department.

Saarbrücken, _____ _____

(Datum/Date) (Unterschrift/Signature)

Abstract

The transformation of a Linear-time Temporal Logic (LTL) formula to a Büchi automaton is a widely used reduction and builds a crucial path in the verification against or the synthesis of a given specification. To deal with the exponential blowup in the state-space of the resulting automata, recent research already focused on reducing the number of states by creating and applying different constructions. Although some of them focus on a previous pushing in or pushing out of LTL operators, most of them lack in execution and argumentation of the effectiveness of such a simplification.
In this thesis, we consider the constructions from an LTL formula to a Büchi Automaton by Gastin and Oddoux and by Gerth et al., also called Tableau Construction. Subsequent we will examine both constructions in terms of the most familar equivalences over LTL and prove the discovered simplifications in the field of the resulting state space with the help of decent prove systems.
At the end we present for each mentioned construction a set of provable theorems over their LTL input, guaranteeing a state space minimization by applying them on the given formula before using the related construction.

Contents

1 Introduction

Linear-time Temporal Logic (LTL) was introduced in 1977 by Amir Pnueli [3] and became very popular by its intuitive operators for reasoning about program properties and behaviours. In [2] Sistla and Clark showed that the problem of LTL for model checking is PSPACE-complete, as well as its satisfiability and validity problem. Furthermore, as proved in [14, 11], the LTL realizability problem is 2-EXP-complete. Therefore, the expressiveness of LTL leads to many practical benefits, like the AMBA AHB case study [4], where LTL is used to specify its architecture. Moreover, this logic is part of a lot of discussions about hardware specifications in the Hardware Model Checking Competition (HWMCC). In synthesis, LTL provides advantages, for example, in the Reactive Synthesis Competition (SYNTCOMP) that focuses on research of reactive synthesis tools.

An important application for LTL is its transformation into a Büchi Automaton, for example, in LTL model checking. Besides the LTL model checker SPIN [8] and SPOT [1], whose implementations are based on a Tableau Construction by Gerth et al. [13], a lot of tools deal with the improvement of results based on the input of an LTL property. For an extended algorithm of the mentioned Tableau Construction, Etessami and Holzmann discussed in [10] a few techniques for preprocessing LTL formulas for their translation tool EQLTL by applying theoretic reductions and proved the correctness of their results via structural induction. Using equivalences over LTL, Somenzi and Bloem analyzed in [7] several rewriting rules for a given input property to reduce the size of its resulting automaton for any construction in general, implemented in their translation tool Wring. Another popular translation tool is LTL2BA, introduced by Gastin and Oddoux [12]. Instead of using a tableau, LTL2BA first creates a very weak Alternating Automaton, transforming it into a Generalized Büchi Automaton and finally achieving a Büchi Automaton. Some improvements of their algorithm are presented by Babiak et al. [15] providing their tool LTL3BA. In their work, they mention a few equivalences that can be applied on the input formula before starting the algorithm.

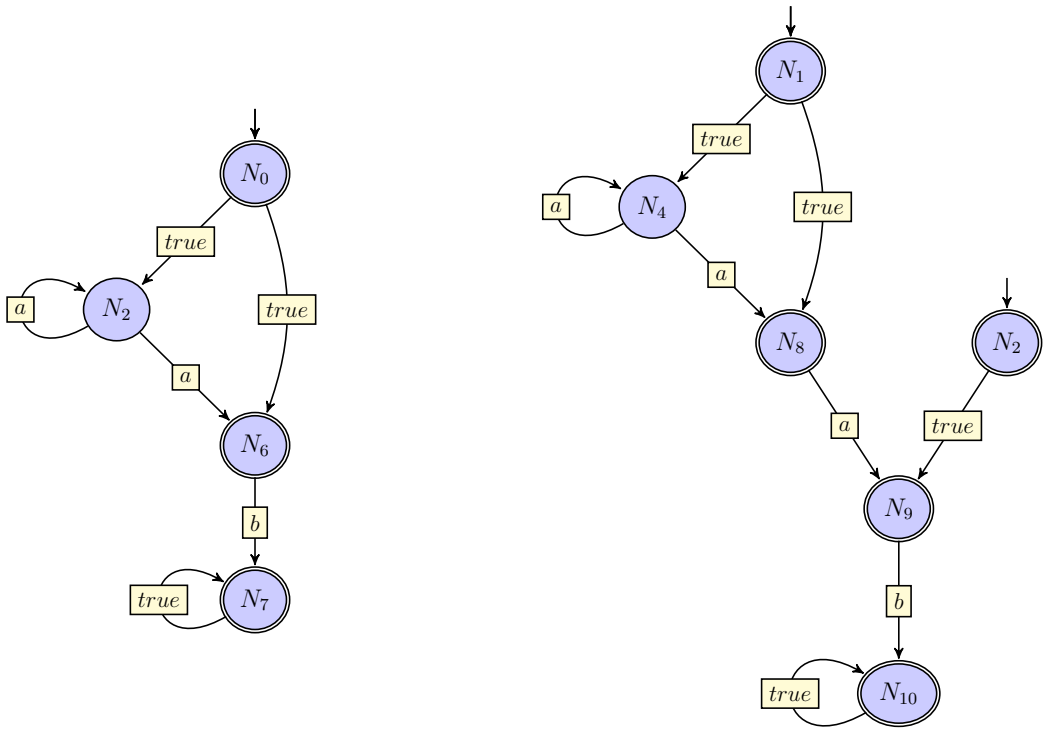

Figure 1.1: BA for $\bigcirc(a \,\mathcal{U}\, b)$ on the left, BA for $(\bigcirc a)\,\mathcal{U}(\bigcirc b)$ on the right

As in [7, 10, 15], we analyze the improvement in the number of states of a Büchi Automaton resulting from an LTL formula. Considering two mentioned constructions from LTL to Büchi Automata, a Tableau Construction by Gerth et al. [13] and a construction over very weak Alternating Automata by Gastin and Oddoux [12], stated in this paper as GO Construction, we discuss a much broader field of equivalences over LTL as in [7, 15], that should be applied to the LTL input before processing the construction. Furthermore, we verify the effectivness of our simplifications by giving an idea of proving the resulting state reductions via structural induction, as it was not proceeded in [7, 15], yet.

Concretely, given some LTL property $\bigcirc(a \,\mathcal{U}\, b)$, meaning that from the next state onwards, a has to be satisfied, until b is satisfied, and its equivalent representation $(\bigcirc a)\,\mathcal{U}(\bigcirc b)$, applying the mentioned Tableau Construction results in the Büchi Automata in Figure 1.1. We notice that pushing in the $\bigcirc$-operator causes two additional states in the resulting Büchi automaton. We can make such observations for a huge variant of equivalences, as we show as part of this

thesis.

In the next section we first present the knowledge base for Linear-time Temporal Logic as well as automata over infinite words, before presenting the considered constructions from LTL to BA, the Tableau Construction and the GO Construction. Afterwards in section 3 we present our observations for both constructions combined with the equivalences over LTL. Additionally we give an idea for proving each result. Finally, in section 4 we summarize our results.

2 Preliminaries

In this thesis, we consider automata over words defined by an alphabet Σ with its so called letters, denoted by a, b, c and so forth. By Σ^* we define the set of all finite words over an alphabet, such as the word *abc* with length three. However, this work refers to automata over infinite words, defining so called ω-languages, where the ω-symbol denotes the infinite set $\{1, 2, 3, ...\}$. The set Σ^ω defines the set of all infinite words over the alphabet Σ. We denote an infinite word by $w = w_0 w_1 w_2 ...$ with $\forall i.\ w_i \in \Sigma$.
In this chapter we talk about LTL formulas at first. We denote arbitrary LTL formulas by φ, ψ, also with added indices, like φ_1, φ_2 and so forth. Afterwards we introduce some kind of infinite word automata before defining two different constructions from LTL to BA that we use in further sections.

2.1 Linear-time Temporal Logic (LTL)

Building a crucial part in this thesis as input for our mentioned constructions, the linear-time temporal logic (LTL) was introduced to specify the specification of a system. We mostly stick to its notation presented in [12] and in [6].

2.1.1 Syntax

Given a set of atomic propositions AP, the syntax of LTL consists of the elements of AP, the components of propositional logic *true* and *false* and its standard boolean connectives *negation* and *conjunction*, including its derivatives, and the temporal operators $\bigcirc$ and $\mathcal{U}$. For $\mathcal{U}$, the derived temporal operators $\Diamond$ and $\Box$ are more commonly used. Futhermore, the derived temporal operator $\mathcal{R}$ ensures the negation of $\mathcal{U}$.
Note that because of the NNF precondition presented in Section 2.1.4, we do not have to

consider the derived boolean operators for implication and equivalence and therefore we do not mention them later on. From now on we assume that φ, ψ and their subformulas are arbitrary LTL formulas.

- *Propositional operators*: $p \in AP \cup \{true, false\}$, $\neg\varphi$, $\varphi \wedge \psi$
- *Temporal operators*: $\bigcirc\varphi$, $\varphi \,\mathcal{U}\, \psi$
- *Derived operators*
  - Disjunction: $\varphi \vee \psi \equiv \neg\varphi \wedge \neg\psi$
  - Implication: $\varphi \rightarrow \psi \equiv \neg\varphi \vee \psi$
  - Equivalence: $\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$
  - Release: $\varphi \,\mathcal{R}\, \psi \equiv \neg(\neg\varphi \,\mathcal{U}\, \neg\psi)$
  - Eventually: $\diamondsuit\varphi \equiv true \,\mathcal{U}\, \varphi$
  - Always: $\square\varphi \equiv \neg\diamondsuit\neg\varphi \equiv \neg(true \,\mathcal{U}\, \neg\varphi) \equiv false \,\mathcal{R}\, \varphi$

2.1.2 Semantics

Let $\alpha = \alpha_0\alpha_1... \in \Sigma^\omega$ with $\Sigma = 2^{AP}$ and φ be some arbitrary LTL formula. We define the relation $\alpha \models \varphi$ as follows:

- $\alpha \models p$ if $p \in \alpha_0$
- $\alpha \models \neg\varphi$ if $\alpha \not\models \varphi$
- $\alpha \models \varphi_1 \wedge \varphi_2$ if $\alpha \models \varphi_1$ and $\alpha \models \varphi_2$
- $\alpha \models \bigcirc\varphi$ if $\alpha_1\alpha_2... \models \varphi$
- $\alpha \models \varphi_1 \,\mathcal{U}\, \varphi_2$ if $\exists j \geq 0, \alpha_j\alpha_{j+1}... \models \varphi_2$ and $\forall 0 \leq i < j, \alpha_i\alpha_{i+1}... \models \varphi_1$

2.1.3 Equivalences

There are several equivalences over LTL concerning commutativity and distributivity laws that we use in this thesis for validating our constructions. Additionally, there are some other equivalences, the so called expansion laws of Until and Release, that build an important part for the understanding of our later algorithms.

- Commutativity Laws

$$\begin{aligned} \Diamond\bigcirc\varphi &\equiv \bigcirc\Diamond\varphi \\ \Box\bigcirc\varphi &\equiv \bigcirc\Box\varphi \end{aligned}$$

- Distributivity Laws

$$\begin{aligned} \bigcirc(\varphi \vee \psi) &\equiv (\bigcirc\varphi) \vee (\bigcirc\psi) \\ \bigcirc(\varphi \wedge \psi) &\equiv (\bigcirc\varphi) \wedge (\bigcirc\psi) \\ \bigcirc(\varphi \,\mathcal{U}\, \psi) &\equiv (\bigcirc\varphi) \,\mathcal{U}\, (\bigcirc\psi) \end{aligned}$$

$$\begin{aligned} \Diamond(\varphi \vee \psi) &\equiv (\Diamond\varphi) \vee (\Diamond\psi) \\ \Box(\varphi \wedge \psi) &\equiv (\Box\varphi) \wedge (\Box\psi) \end{aligned}$$

- Expansion Laws

$$\psi_1 \,\mathcal{U}\, \psi_2 = \psi_2 \vee (\psi_1 \wedge \bigcirc(\psi_1 \,\mathcal{U}\, \psi_2)). \tag{2.1}$$

$$\psi_1 \,\mathcal{R}\, \psi_2 = (\psi_1 \wedge \psi_2) \vee (\psi_2 \wedge \bigcirc(\psi_1 \,\mathcal{R}\, \psi_2)) \tag{2.2}$$

2.1.4 Negation Normal Form (NNF)

As a precondition, all considered constructions only accept LTL formulas that fulfill the *Negation Normal Form*. This property is satisfied if the given formula only consists of literals, conjunctions, disjunctions, $\bigcirc$-, $\mathcal{U}$- and $\mathcal{R}$-formulas. By applying multiple steps, each formula can be transformed into its NNF. We give an example for the formula $\varphi = \neg(\neg\Diamond a \leftrightarrow \neg\Box b)$.

1. Eleminate all $\leftrightarrow$ - Operators by applying

- $\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$

2. Eleminate all $\rightarrow$ - Operators by applying
   - $\varphi \rightarrow \psi \equiv \neg\varphi \vee \psi$
3. Eleminate all $\Diamond$ - Operators by applying
   - $\Diamond\varphi \equiv true\ \mathcal{U}\ \varphi$
4. Eleminate all $\Box$ - Operators by applying
   - $\Box\varphi \equiv false\ \mathcal{R}\ \varphi$
5. Push in all Negation - Operators by applying
   - $\neg true \equiv false, \neg false \equiv true, \neg p$ with $p \in AP$
   - $\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi$
   - $\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$
   - $\neg\bigcirc\varphi \equiv \bigcirc\neg\varphi$
   - $\neg(\varphi\ \mathcal{U}\ \psi) \equiv \neg\varphi\ \mathcal{R}\ \neg\psi$
   - $\neg(\varphi\ \mathcal{R}\ \psi) \equiv \neg\varphi\ \mathcal{U}\ \neg\psi$

Example 1. *We give an example for the formula* $\varphi = \neg(\neg\Diamond a \leftrightarrow \neg\Box b)$.

$$
\begin{aligned}
\varphi &\equiv && \neg(\neg\Diamond a \leftrightarrow \neg\Box b) \\
&\overset{1}{\equiv} && \neg((\neg\Diamond a \rightarrow \neg\Box b) \wedge (\neg\Box b \rightarrow \neg\Diamond a)) \\
&\overset{2}{\equiv} && \neg((\neg\neg\Diamond a \vee \neg\Box b) \wedge (\neg\neg\Box b \vee \neg\Diamond a)) \\
&\overset{3}{\equiv} && \neg((\neg\neg(\text{true}\ \mathcal{U}\ a) \vee \neg\Box b) \wedge (\neg\neg\Box b \vee \neg(\text{true}\ \mathcal{U}\ a))) \\
&\overset{4}{\equiv} && \neg((\neg\neg(\text{true}\ \mathcal{U}\ a) \vee \neg(\text{false}\ \mathcal{R}\ b)) \wedge (\neg\neg(\text{false}\ \mathcal{R}\ b) \vee \neg(\text{true}\ \mathcal{U}\ a))) \\
&\overset{5}{\equiv} && \neg((\text{true}\ \mathcal{U}\ a) \vee \neg(\text{false}\ \mathcal{R}\ b)) \vee \neg((\text{false}\ \mathcal{R}\ b) \vee \neg(\text{true}\ \mathcal{U}\ a)) \\
&\overset{5}{\equiv} && (\neg(\text{true}\ \mathcal{U}\ a) \wedge (\text{false}\ \mathcal{R}\ b)) \vee (\neg(\text{false}\ \mathcal{R}\ b) \wedge (\text{true}\ \mathcal{U}\ a)) \\
&\overset{5}{\equiv} && ((\text{false}\ \mathcal{R}\ \neg a) \wedge (\text{false}\ \mathcal{R}\ b)) \vee ((\text{true}\ \mathcal{U}\ \neg b) \wedge (\text{true}\ \mathcal{U}\ a))
\end{aligned}
$$

2.2 Automata over Infinite Words

To describe languages over infinite words, automata over infinite words have to be introduced. Their components are very similar to those of automata over finite words, however, their acceptance is defined on infinite words. Note also that we consider non-deterministic automata. An Automaton over infinite words gets defined by the tuple $A = (Q, \Sigma, I, \Delta, Acc)$ with

- Q is a finite set of states
- Σ is a finite alphabet
- $I \subseteq Q$ describes the initial states
- $\Delta \subseteq Q \times \Sigma \times Q$ is the transition function
- $Acc \subseteq Q^\omega$ describes the set of accepted words (accepting condition)

A run r on an automaton A over an infinite word α is an infinite sequence of states $r_0 r_1 ...$ such that $r_0 \in I$ and $\forall i \in \mathbb{N}, (r_i, \alpha_i, r_{i+1}) \in \Delta$. We say that such a run r is accepting if $r \in Acc$. By the set $\text{Inf(r)} = \{q \in Q \mid \forall m \in \mathbb{N}.\ \exists n \in \mathbb{N}.\ n \geq m \text{ and } q_m = q_n\}$ for some run $r \in Q^\omega$, we define all states that occur infinitely often in r. Similar to this, define $\text{Inf}(\alpha) = \{\sigma \in \Sigma \mid \forall m \in \mathbb{N}.\ \exists n \in \mathbb{N}.\ n \geq m \text{ and } \alpha(n) = \sigma\}$ as the set of all letters of Σ, that occur infinitely often in a word α. The language of the automaton A is defined as $L(A) = \{\alpha \in \Sigma^\omega \mid A \text{ accepts } \alpha\}$.

In this thesis such automata over infinite words build our intermediate results as well as our final results. For each of the mentioned constructions, we transform a given LTL formula input into various automata over infinite words, finally leading into a Büchi automaton (BA), that was introduced in 1962 by Julius Richard Büchi [9]. Besides BA, we also describe the conditions of coBüchi (cBA) and Generalized Büchi automata (GBA). We also consider so called Alternating Automata (AA) and their very weak variant VWAA.

2.2.1 Büchi Automata (BA) and coBüchi Automata (cBA)

A Büchi automaton is an automaton over infinite words described by the tuple $A = (Q, \Sigma, I, \Delta, \text{BÜCHI}(F))$ with $F \subseteq Q$ and its acceptance condition

$$\text{BÜCHI}(F) = \{\alpha \in Q^\omega \mid \text{Inf}(\alpha) \cap F \neq \emptyset\}.$$

Based on the Büchi acceptance condition, a run r on a Büchi automaton is accepting if the set F is visited infinitely often, i.e. the intersection of the set of all infinitely visited states in r and the final set F should not be empty.

Example 2. *In Figure 1.1 one can see the representations of the LTL specifications* $\bigcirc(a \,\mathcal{U}\, b)$ *and* $(\bigcirc a)\,\mathcal{U}\,(\bigcirc b)$ *as Büchi automata. By defining* $F = \{N_0, N_6, N_7\}$ *for the left and* $F = \{N_1, N_2, N_8, N_9, N_{10}\}$ *for the right automaton, we achieve Büchi automata that guarantee our specifications.*

A coBüchi automaton is defined by the tuple $A = (Q, \Sigma, I, \Delta, \text{coBÜCHI}(F))$ with $F \subseteq Q$ and the acceptance condition

$$\text{coBÜCHI}(F) = \{\alpha \in Q^\omega \mid \text{Inf}(\alpha) \cap F = \emptyset\}.$$

As the dual of the Büchi acceptance condition, a run r on a coBüchi automaton is accepting if the set F is only visited finitely often, that means, the intersection of all infinitely visited states in r and the final set F are empty.

Since coBüchi is the dual of the Büchi condition, we are able to transform a BA into a cBA and vice versa. Therefore it suffices to build the counterset $Q \setminus F$ to get the acceptance set for the dual automaton. So we conclude that $\text{BÜCHI}(F) = \text{coBÜCHI}(Q \setminus F)$ and correspondingly $\text{coBÜCHI}(F) = \text{BÜCHI}(Q \setminus F)$.

Example 3. *As mentioned in Example 2, we created Büchi automata with the acceptance sets* $\{N_0, N_6, N_7\}$ *and* $\{N_1, N_2, N_8, N_9, N_{10}\}$. *However we can create their cBA by simply building the countersets* $Q_1 \setminus \{N_0, N_6, N_7\} = \{N_2\}$ *and* $Q_2 \setminus \{N_1, N_2, N_8, N_9, N_{10}\} = \{N_4\}$. *Therefore, we can conclude that the left automaton in Figure 1.1 with acceptance condition BÜCHI(*$\{N_0, N_6, N_7\}$*) accepts the same language as with acceptance condition coBÜCHI(*$\{N_7\}$*).*

2.2.2 Generalized Büchi Automata (GBA)

A generalized Büchi automaton (GBA) is a Büchi automaton with a more general acceptance condition. It is identified by the tuple $A = (Q, \Sigma, I, \Delta, \text{GENBÜCHI}(F))$ with $F \subseteq 2^Q$ and the acceptance condition

$$\text{GENBÜCHI}(F) = \{\alpha \in Q^\omega \mid \forall\ F' \in F.\ \text{Inf}(\alpha) \cap F' \neq \emptyset\}.$$

Based on this acceptance condition, a run r on an automaton is accepting if on each set F' of F at least one member is visited infinitely often, that means, each intersection with all infinitely visited states and a set $F' \subseteq F$ is non-empty.

Example 4. *Recapturing Example 2, we could switch the Büchi conditions simply to Generalized Büchi conditions by BÜCHI($\{N_0, N_6, N_7\}$) = GENBÜCHI($\{\{N_0, N_6, N_7\}\}$) and BÜCHI($\{N_1, N_2, N_8, N_9, N_{10}\}$) = GENBÜCHI($\{\{N_1, N_2, N_8, N_9, N_{10}\}\}$).*

2.2.3 Alternating Automata (AA)

An alternating automaton (AA) is an automaton over infinite words with a different transition function. Instead of targeting a single state, a transition can target multiple states by building a boolean combination of its destinations. Therefore, an alternating automaton $A = (Q, \Sigma, I, \Delta, Acc)$ is defined as follows:

- Q is a finite set of states
- Σ is a finite alphabet
- $I \subseteq Q$ describes the initial states
- $\Delta \subseteq Q \times \Sigma \to \mathbb{B}^+(Q)$ is the transition function
- $Acc \subseteq Q^\omega$ describes the accepting condition

Note that all accepting conditions that we have talked about before are applicable to an alternating automaton as well.

Example 5. *In Figure 2.1 one can see a representation of the LTL specification $\bigcirc(a \wedge b)$ as an AA. The conjunction of our formula gets represented by an alternating transition.*

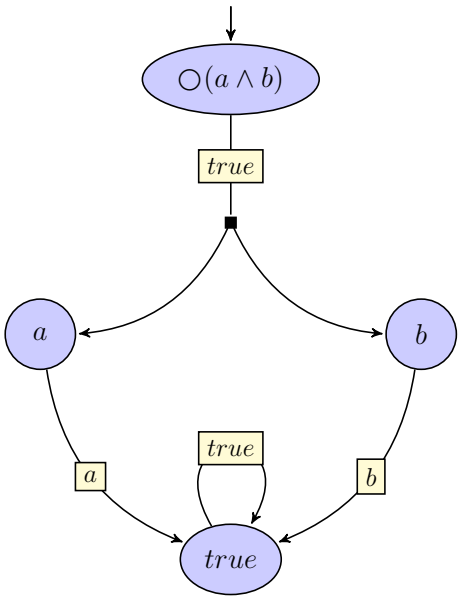

Figure 2.1: Example for VWAA of $\bigcirc(a \wedge b)$

2.2.4 Very Weak Alternating Automata (VWAA)

Very weak alternating automata (VWAA) are AA with an additional partial ordering $\succeq$ on their set of states Q, such that, $\forall q, q' \in Q$, if $q' \in \Delta(q)$ then $q \succeq q'$, that means, q' got a lower or equal value than q. Analogously, we could represent such an automaton as a directed acyclic graph with selfloops.

Example 6. *The automaton in Figure 2.1 fullfills our very weak condition, since none of its states goes back to a previous one. Therefore, the state labeled by true got the lowest value according to the partial ordering.*

2.2.5 GBA to BA

It is possible to transform a GBA $A = (Q, \Sigma, I, \Delta, F)$ with $F = \{F_0, F_1, ..., F_{n-1}\}$ into a BA $A' = (Q', \Sigma, I', \Delta', F')$ with the so called "round-robin Construction" or "counting construction" in [5]. Its idea is to add a counter i to each state, representing the current acceptance subset F_i that has to be visited and increasing the counter whenever such a state is reached. So if and only if some counter occurs infinitely often, we can conclude that a word is accepted.

- $Q' = Q \times \{0, 1, ..., n-1\}$
- $I' = I \times \{0\}$
- $\Delta'((q,i),a) = \begin{cases} \{(q',i) \mid q' \in \Delta(q,a)\} & \text{if } q \notin F_i, \\ \{(q', i+1 \mod n) \mid q' \in \Delta(q,a)\} & \text{otherwise.} \end{cases}$
- $F' = F_0 \times \{0\}$

Example 7. *As an example consider the automaton on Figure 2.2 with the final set* $F = \{\{N_0\}, \{N_1\}\}$*. This automaton ensures that transitions with* a *and* b *are taken infinitely often. After applying the Round-Robin Construction we achieve the automaton stated in Figure 2.3. Notice that our state space doubled in size since we got two sets in the set of final states.*

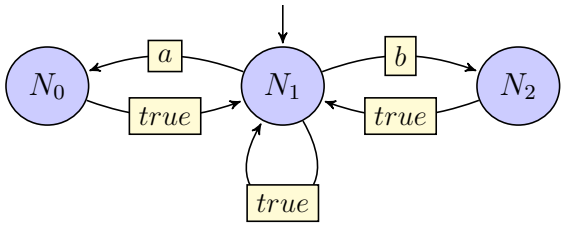

Figure 2.2: Example Input GBA for Round-Robin Construction

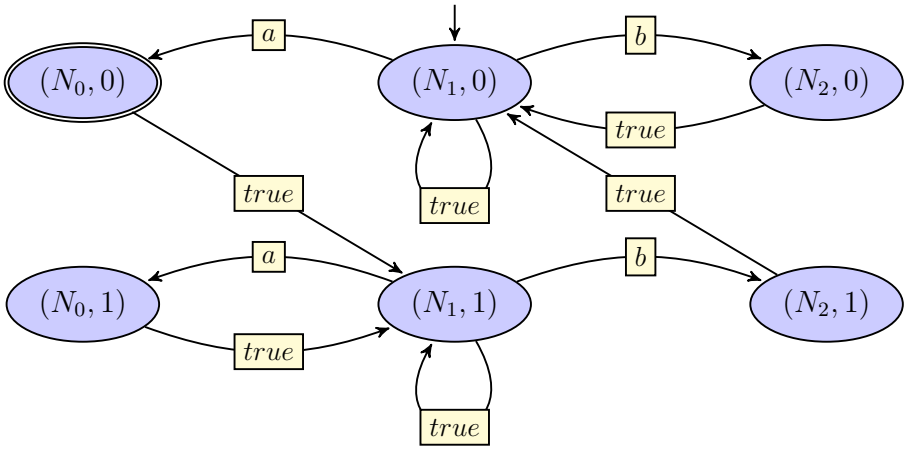

Figure 2.3: Example Output BA for Round-Robin Construction

2.3 Tableau Construction

The Tableau Construction, that was introduced by Gerth, Peled, Vardi and Wolper [13], is a tableau-based construction from LTL to BA. Its idea is to first creat a tableau out of the given LTL formula input based on a depth-first-search strategy. Afterwards a GBA can be created based on the created tableau. Finally, we can achieve a BA by applying the Round-Robin Construction on our GBA, as presented in Section 2.2.5.

2.3.1 LTL to Tableau

Given an LTL formula in NNF, we obtain a tableau containing six different columns, giving a sufficient description of a GBA. The first five columns describe the name of a possible node, its incoming edges, non-fullfilled requirements, already fullfilled requirements, representing also its outgoing labels, and requirements for its immediate successors, respectively. The sixth column gives information about its adding to the final GBA.

Data Structure

We will use the data structure below for the constructed tableau. The terms in brackets denote their abbreviation in the tableau.

- **Name** (Name) describes the unique name of a node.
- **Incoming** (Inc) describes a set of node names representing incoming edges. The special name *Init* marks an initial node.
- **New** (New) describes a set of temporal properties (formulas) that must hold.
- **Old** (Old) describes a set of temporal properties that already hold.
- **Next** (Next) describes a set of temporal properties that must hold in immediate successors.

Name	Inc	New	Old	Next	Added
N_0	Init	$\{\varphi\}$	$\emptyset$	$\emptyset$	

Table 2.1: Representation of the initial node in TC

- **Added** (Added) describes the added state of the node. We got the following states: *ADDED* - node is part of the resulting GBA; *SPLIT* - node triggered a split during the algorithm; *IGNORED* - the expansion of the node terminated without adding it to the resulting GBA.

We call a certain field of a node by its abbreviation and its name in brackets, like Name(N_0). We will represent our set of nodes in a table, for example, Table 2.1, that shows the representation of a single node. In the whole tableau construction we write all nodes created by the algorithm among each other.

Algorithm

We start our algorithm with the inital node N_0 that gets marked as inital. Although we add the starting formula φ in NNF to its *New* field and keep the other ones empty. One can see its representation in Table 2.1.

We start the expansion algorithm with the expansion of our initial node. Therefore, as long as its *New* field is not empty, we extract one of its formulas and proceed by a case analysis over the formula structure of LTL. For the following cases we assume some arbitrary node N_x with the *Incoming* field *Inc*, the *New* field $\{\varphi_0, \varphi_1, ..., \varphi_n\}$, the *Old* field *Old*, the *Next* field *Next* and the empty *Added* field. In the following we distinguish the upcoming cases for extraxting the formula φ_0 out of $New(N_x)$ without loss of generality since each formula of *New* will be considered during the expansion and, therefore, the order does not matter.

- $\varphi_0 = \psi_1 \wedge \psi_2$
  Given the conjunction φ_0, node N_x has to guarantee that ψ_1 and ψ_2 are fullfilled. Therefore, we can assume that the whole conjunction is satisfied by switching it to *Old* and adding both of its inner components to *New*.

Name	Inc	New	Old	Next	Added
N_x	Inc	$\{\varphi_1, ..., \varphi_n\}$	Old	Next	SPLIT
N_{x+1}	Inc	$\{\varphi_1, ..., \varphi_n\} \cup (\text{New}_1 \backslash \text{Old})$	$\text{Old} \cup \varphi_0$	$\text{Next} \cup \text{Next}_1$	
N_{x+2}	Inc	$\{\varphi_1, ..., \varphi_n\} \cup (\text{New}_2 \backslash \text{Old})$	$\text{Old} \cup \varphi_0$	Next	

Table 2.2: Representation of a split in TC

- $\varphi_0 = \bigcirc\psi$
  A $\bigcirc$-formula is satisfied on a node if its inner formula is fullfilled on the immediate successor. Therefore, we have to add ψ to *Next* and switch φ_0 to *Old*.

- $\varphi_0 = \psi_1 \,\mathcal{U}\, \psi_2$ or $\varphi_0 = \psi_1 \,\mathcal{R}\, \psi_2$ or $\varphi_0 = \psi_1 \vee \psi_2$
  Our algorithm calls this case a *split*, that means, we terminate the expansion of the current node N_x by setting its *Added* field to *SPLIT* and start the expansion of two fresh nodes N_{x+1}, N_{x+2}, one after the other starting with N_{x+1}. Their general representation is stated in Table 2.2.

  The easiest case for a split is a disjunction, since we just add its left side to N_{x+1} as New_1 and its right side to N_{x+2} as New_2. We keep $Next_1$ empty, since a disjunction does not have any temporal requirement.
  For an $\mathcal{U}$-formula we exploit its equivalent representation of the expansion law 2.1 to define an appropriate splitting rule. Therefore, we use the left side of the equivalent disjunction to define N_{x+2} by setting $\{\psi_2\}$ as New_2. Its right side builds a conjunction which can be directly expanded with our given rules. Therefore, we can set $\{\psi_1\}$ as New_1 and $\{\psi_1 \,\mathcal{U}\, \psi_2\}$ as $Next_1$.
  Similar to $\mathcal{U}$, we can exploit the equivalent representation of a $\mathcal{R}$ formula in its expansion law 2.2 to define an appropriate splitting rule. As before, the left side of the equivalent disjunction represents N_{x+2} by setting $\{\psi_1, \psi_2\}$ as New_2 and the right side builds the components of N_{x+1} by setting $\{\psi_2\}$ as New_1 and $\{\psi_1 \,\mathcal{R}\, \psi_2\}$ as $Next_1$.

- $p \in AP$, $\varphi_0 = p$ or $\varphi_0 = \neg p$ or $\varphi_0 = true$ or $\varphi_0 = false$
  Given a literal, that means an atomic proposition p or its negation $\neg p$, whenever its negation is element of *Old*, we would have a contradiction and, therefore, we can discard the current node, setting its *Added* field to *IGNORED*. Otherwise we add the literal to *Old* to ensure the satisfaction of the formula on our current node.
  Instead, if we got $true$ as formula, we can directly switch it to *Old*. If our formula

is equal to $false$ we can again discard the current node and set its *Added* field to *IGNORED*.

If the *New* field of a node is empty we have ensured that all of its specifications are fulfilled (the node is fully expanded) and, therefore, we can terminate its expansion. To avoid adding equivalent nodes, that means nodes with the same *Old* and *Next* fields, we have to search for such elements. If a fully expanded node N is equivalent to a node N' with *Added(N')* = *ADDED*, we set *Added(N)* to *IGNORED* and add its incoming field *Inc(N)* to the incoming field of the equivalent node *Inc(N')*. Therefore, we can easily reduce the number of states by just updating an equivalent node. Afterwards, we can continue the expansion of the next node in the tableau with an undefined *Added* field. Otherwise, if there is no equivalent node we set the state of N as *ADDED* and create a fresh node with a fresh name N' by setting {N} as its incoming fields, *Next(N)* as its *New* field and keeping its *Old* and *Next* fields empty. After that, we start the expansion of N'.
If there are no more undefined *Added* fields, the algorithm terminates and our GBA can be expressed by translating the *nodes set* of our tableau, that means the set of all nodes with an *ADDED* state.

Example 8. *In Table 2.3 you can observe the applying of the Tableau Construction to our frequently used LTL property* $\bigcirc(a \,\mathcal{U}\, b)$. *Note that we underline a formula in a New field whenever it gets expanded.*

Name	Inc	New	Old	Next	Added
N_0	{Init}	$\underline{\{\bigcirc(a \,\mathcal{U}\, b)\}}$	$\{\bigcirc(a \,\mathcal{U}\, b)\}$	$\{a \,\mathcal{U}\, b\}$	ADDED
N_1	$\{N_0\}$	$\underline{\{a \,\mathcal{U}\, b\}}$	$\emptyset$	$\emptyset$	SPLIT
N_2	$\{N_0,N_2\}$	$\underline{\{a\}}$	$\{a \,\mathcal{U}\, b,a\}$	$\{a \,\mathcal{U}\, b\}$	ADDED
N_4	$\{N_2\}$	$\underline{\{a \,\mathcal{U}\, b\}}$	$\emptyset$	$\emptyset$	SPLIT
N_5	$\{N_2\}$	$\underline{\{a\}}$	$\{a \,\mathcal{U}\, b,a\}$	$\{a \,\mathcal{U}\, b\}$	IGNORED
N_6	$\{N_0,N_2\}$	$\underline{\{b\}}$	$\{a \,\mathcal{U}\, b,b\}$	$\emptyset$	ADDED
N_7	$\{N_6,N_7\}$	$\emptyset$	$\emptyset$	$\emptyset$	ADDED
N_8	$\{N_7\}$	$\emptyset$	$\emptyset$	$\emptyset$	IGNORED
N_3	$\{N_0\}$	$\underline{\{b\}}$	$\{a \,\mathcal{U}\, b,b\}$	$\emptyset$	IGNORED

Table 2.3: Computation Tableau of $\bigcirc(a \,\mathcal{U}\, b)$

Name	Inc	New	Old	Next	Added
N_x	Inc_x	$\{a \mathcal{U} b, c \wedge (a \mathcal{U} b)\}$	Old_x	Next_x	SPLIT
N_{x+1}	Inc_x	$\{c\}$	$\text{Old}_x \cup \{a \mathcal{U} b\}$	$\text{Next}_x \cup \{a \mathcal{U} b\}$	
N_{x+2}	Inc_x	$\{c, a \mathcal{U} b\}$	Old_x	Next_x	

Table 2.4: Example for additional requirements of TCI

2.3.2 Optimizations

In their algorithm, Gerth, Peled, Vardi and Wolper mentioned multiple improvements for efficiency. In this thesis, we just consider those improvements that reduce the number of states of our nodes set. This can be achieved by not necessarily switching all formulas to the *Old* fields during the expansion to increase the number of equivalent nodes and, therefore, reducing the total size of the nodes set. The improved version of our tableau construction does not add the original conjunction, disjunction, $\mathcal{U}$- or $\mathcal{R}$-formula to the *Old* fields during the expansion as long as it does not build the right hand side of an $\mathcal{U}$ formula of our initial formula φ.
Note however, that we have to introduce the special requirement for the improved version that conjunctions in a *New* field always have to get preferred for an expansion. This is the case since otherwise a formula gets expanded that is a subformula of a conjunction as well and, therefore, it may happen that its expansion is done twice.

Example 9. *To make this requirement more clear, consider the tableau fragment on Table 2.4. Here we assume that on some node N_x there is an $\mathcal{U}$-formula, that is not a righthand side of another $\mathcal{U}$-formula, and a conjunction containing the same formula in a New field. The expansion of the $\mathcal{U}$-formula triggers a split where we will not include the $\mathcal{U}$-formula to the Old field of the second node N_{x+2} by the optimization rule and our assumption. However, because of this, the conjunction adds the same $\mathcal{U}$-formula to the New field of N_{x+2} and yields another split. By expanding conjunctions at first, we would have prevented such a blowup.*

In the following sections we explicitly differentiate between the normal (TCO) and improved version (TCI) of the tableau construction and compare their results.

2.3.3 Tableau to GBA

The translation of a nodes set to a GBA $A = (Q, \Sigma, I, \Delta, F)$ maps the given information to our automaton. Our states set Q is represented by the unique names of our nodes set and all nodes with *Init* as incoming state are inital. Our alphabet is brought together by all literals in the *Old* field of the nodes set. The transitions can be taken from the *Inc* fields and the conjunctions of literals of incoming nodes, alternatively *true* if no literals exist, as labels. Finally, for each $\mathcal{U}$-subformula of our initial formula, we build an own set for the generalized Büchi condition, where each set is filled by node names of our nodes set that do not have the respected $\mathcal{U}$-formula in their *Old* field or that have the right side of the $\mathcal{U}$-formula in their *Old* field. A more theoretical view of this transformation is stated below.

- $Q = \{\text{Name(N)} \mid \text{N} \in \text{nodes set}\}$
- $\Sigma = \{\psi \mid \exists \text{ N s.t. N} \in \text{nodes set} \wedge \psi \in \textit{Old(N)} \wedge \varphi \text{ is a literal}\} \cup \{\textit{true}\}$
  We call the labeling *Label(N)* of a node N the conjunction of all literals of *Old(N)*. If *Old(N)* doesn't have any literals we determine $\textit{Label(N)} = \textit{true}$.
- $I = \{\text{Name(N)} \mid \text{N} \in \text{nodes set} \wedge \textit{Init} \in \textit{Inc(N)}\}$
- $\Delta := \forall \text{ N} \in \text{nodes set}, \forall \text{ N'} \in \textit{Inc(N)}, \Delta(\text{N'}, \text{Label(N')}) = \text{N}$
- $F = \forall\, \mathcal{U}\text{subformulas } \varphi' \text{ of } \varphi \text{ with } \varphi' = \psi_1\, \mathcal{U}\, \psi_2, \exists F' \subseteq 2^Q \in \text{F such that}$

$$F' := \{\text{Name(N)} \mid \text{N} \in \text{nodes set} \wedge (\varphi' \notin \textit{Old(N)} \vee \psi_2 \in \textit{Old(N)})\}$$

Finally, we can achieve a BA by applying the Round-Robin Construction as introduced in Section 2.2.5.

Example 10. *Continuing our tableau of Example 8, we achieve the following components of a GBA:*

- $Q = \{N_0, N_2, N_6, N_7\}$
- $\Sigma = \{a, b\}$
- $I = \{N_0\}$

- $\Delta(N_0) = \{(true, N_2), (true, N_6)\}$,
  $\Delta(N_2) = \{(a, N_2), (a, N_6)\}$,
  $\Delta(N_6) = \{(b, N_7))\}$,
  $\Delta(N_7) = \{(true, N_7)\}$
- $F = \{\{N_0, N_6, N_7\}\}$

The graphical representation of the resulting BA, using the mentioned Round-Robin construction is exactly the same as presented at the beginning of this thesis in Figure 1.1.

2.4 GO Construction

The construction by Gastin and Oddoux [12] does not succumb any algorithm and, therefore, we do not need any data structure. Its concept creates a VWAA with a coBüchi condition out of our LTL formula φ in NNF first, turning this automaton into a GBA and finally obtaining a BA through a Round-Robin construction as already shown in 2.2.5.

2.4.1 LTL to VWAA

The transformation of an LTL formula φ to a VWAA is based on some computation combined with two fresh operators $\otimes$ for handling conjunctions and $\overline{\psi}$, giving us the disjunctive normal form of a formula and, therefore, allowing us to only observe temporal subformulas at our states. Given Σ as the set of all literals of φ, Q as the set of temporal subformulas of φ and two sets $J_1, J_2 \in 2^\Sigma \times 2^Q$, our additional operators are defined as follows:

$$J_1 \otimes J_2 = \{(\alpha_1 \wedge \alpha_2, e_1 \wedge e_2) \mid (\alpha_1, e_1) \in J_1 \text{ and } (\alpha_2, e_2) \in J_2\} \tag{2.3}$$

$$\overline{\psi} := \begin{cases} \overline{\psi_1 \wedge \psi_2} = \{e_1 \wedge e_2 \mid e_1 \in \overline{\psi_1} \text{ and } e_2 \in \overline{\psi_2}\} \\ \overline{\psi_1 \vee \psi_2} = \overline{\psi_1} \cup \overline{\psi_2} \\ \overline{\psi} = \{\varphi\} \text{ if } \psi \text{ is a temporal formula} \end{cases} \tag{2.4}$$

Regarding those definitions, we can define the components of our VWAA by defining Q and Σ as above. Inital states can be computed by $\overline{\varphi}$. Based on this set we compute the transitions

through the function Δ that is mentioned below, by computing the transitions of each resulting subformula of Δ, as long as it is not already computed. Notice that conjunctions as target states represent alternating transitions, producing two seperate states. Note also that we use conjunctions as the transition labels. However, from now on we consider such labels as both, conjunctions and sets of literals. F is then the set of all $\mathcal{U}$ formulas in φ, defining a coBüchi rather than a Büchi acceptance set. A more compact view is stated below.

- $Q = \{\psi \mid \psi \text{ is a subformula of } \varphi\}$
- $\Sigma = \{\psi \mid \psi \text{ is a literal of } \varphi\} \cup \{true\}$
- $I = \overline{\varphi}$
- Δ is defined as follows
  - $$\delta := \begin{cases} \delta(true) = \{(true, true)\} \\ \delta(p) = \{(p, true)\} \text{ where } p \text{ is a literal} \\ \delta(\neg p) = \{(\neg p, true)\} \text{ where } \neg p \text{ is a literal} \\ \delta(\bigcirc \psi) = \{(true, e) \mid e \in \overline{\psi}\} \\ \delta(\psi_1 \,\mathcal{U}\, \psi_2) = \Delta(\psi_2) \cup (\Delta(\psi_1) \otimes \{(\Sigma, \psi_1 \,\mathcal{U}\, \psi_2)\}) \\ \delta(\psi_1 \,\mathcal{R}\, \psi_2) = \Delta(\psi_2) \otimes (\Delta(\psi_1) \cup \{(\Sigma, \psi_1 \,\mathcal{R}\, \psi_2)\}) \end{cases}$$
  - $$\Delta := \begin{cases} \Delta(\psi_1 \vee \psi_2) = \Delta(\psi_1) \cup \Delta(\psi_2) \\ \Delta(\psi_1 \wedge \psi_2) = \Delta(\psi_1) \otimes \Delta(\psi_2) \\ \Delta(\psi) = \delta(\psi) \text{ if } \psi \text{ is a temporal formula} \end{cases}$$
- $F = \{\psi \mid \psi = \psi_1 \,\mathcal{U}\, \psi_2 \text{ and } \psi \text{ is a subformula of } \varphi\}$

Example 11. *Consider the formula $\bigcirc(a \wedge b)$. Applying those definitions produces the already*

presented VWAA on Figure 2.1. Its computation of its initial state looks like follows:

$$\begin{aligned}
\Delta(\bigcirc(a \wedge b)) &= \delta(\bigcirc(a \wedge b)) \\
&= \{(true, e) \mid e \in \overline{a \wedge b}\} \\
&= \{(true, e) \mid e \in \{e_1 \wedge e_2 \mid e_1 \in \overline{a} \text{ and } e_2 \in \overline{b}\}\} \\
&= \{(true, e) \mid e \in \{e_1 \wedge e_2 \mid e_1 \in \{a\} \text{ and } e_2 \in \{b\}\}\} \\
&= \{(true, e) \mid e \in \{a \wedge b\}\} \\
&= \{(true, a \wedge b)
\end{aligned}$$

Therefore, the state $\bigcirc(a \wedge b)$ got an alternating transition with label true leading into states a and b.

2.4.2 VWAA to GBA

Given a VWAA $A = (Q, \Sigma, I, \Delta, F)$, we first have to get rid of the alternating characterization of our automaton to obtain a GBA $A' = (Q', \Sigma, I, \Delta', F')$. Therefore, we consider conjunctions of subformulas to represent the target states of alternating transitions. Thus, our new state set Q' is the powerset of Q identifying conjunctions of states. As a consequence, we have to compute new transitions Δ' starting in I. To compute the transitions of a conjunction we combine the computed Δ' of all elements through $\otimes$. Afterwards, we are able compute the set of all $\preceq$-minimal transitions, which intuitively is the set of all transitions that take the highest amount of labels to go to the lowest number of states. Different from [12], where acceptance conditions over transition sets were observed, we compute the classical final sets for GBA over states by creating a set f for each formula of F and including all those states into f that do not have the respected formula as a subformula. A more theoretical view is stated below.

- $Q' = 2^Q$, i.e. conjunctions of states
- $\Delta'(q_1 \wedge q_2 \wedge ... \wedge q_n) = \bigotimes_{i=1}^{n} \Delta(q_i)$, reduced by computing the $\preceq$-minimal transitions where $t \preceq t'$ if $t = (e, \alpha, e'), t' = (e, \alpha', e''), \alpha \subseteq \alpha', e'' \subseteq e'$
- $F' :=$ $\forall$ formula f of F create set $f' \subseteq Q'$ in F' s.t. $\forall q \in f'$. f is not a subformula of q

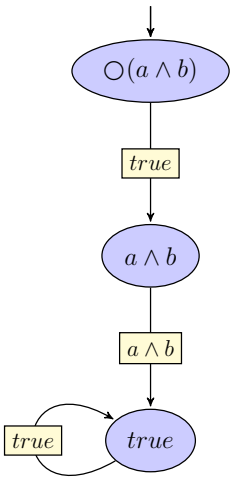

Figure 2.4: Example GBA for $\bigcirc(a \wedge b)$ via GOC

Finally, we achieve a BA by using the Round-Robin Construction, mentioned in Section 2.2.5.

Example 12. *Continuing our computation of the VWAA for $\bigcirc(a \wedge b)$, we observe the GBA in Figure 2.4 by applying the construction above.*

3 Evaluation

A primary goal of this thesis is the detection of likely equivalences for LTL formulas to minimize the state space of the resulting Büchi automata after applying one of our constructions. Therefore, we consider each of the given equivalences in the presented Tableau Construction [13], as well as in the presented GO Construction [12] to detect possible minimizations. To keep it as general as possible and to get an approach for our later proofs, we have to observe arbitrary LTL formulas in NNF as inner formulas of our equivalences. This leads us to proofs via structural induction over the formula structure of LTL, where we have to include all operators of LTL caused by the NNF requirement. Considering an arbitrary formula φ, we have to observe seven different cases for its representation in our structural induction proofs: $a \in AP, \neg a, \psi_1 \wedge \psi_2, \psi_1 \vee \psi_2, \bigcirc \psi, \psi_1 \,\mathcal{U}\, \psi_2, \psi_1 \,\mathcal{R}\, \psi_2$. Furthermore, given a formula with two arbitray formulas would lead to $7^2 = 49$ possibilites that all have to get observed.
However, a more complicated part will be the handling of arbitrary formulas in the algorithms of both constructions itself. Hence, we have to create an appropriate solution for each construction by extending their current representation. After presenting those extensions, we present our observations divided into the mentioned equivalences over LTL.

3.1 Tableau Construction

3.1.1 Extended Representation of Tableau Construction

The main question for the Tableau Construction is how to handle an arbitrary formula in a *New* field that has to be processed during the node expansion. We skip such a formula until only undefined arbitrary formulas are elements of the *New* field. Then, we skip the node, keeping its status empty, and begin to expand the next node. We consider such nodes as a

black box where possibly more nodes are part of it. In later formulas we give each black box with *New* field φ a variable $x_\varphi \geq 1$ to denote its number of states. Note however, that such a variable is an upper bound because of equivalent nodes between black boxes.
In automata we draw black box nodes with rectangles instead of circles. Also we will name them in a special way, for example, $TC(\psi)[N_x]$ means that we consider in this node the resulting GBA of the Tableau Construction of the formula ψ where N_x denotes the name of the node that caused the black box.
However, a big question is how to compare black box variables and normal constant numbers for our later proofs. This is somehow difficult since the black box can be any arbitrary number and in addtion, it is also possible that black boxes share several nodes. Therefore, we assume that each blackbox variable describes at least one node, but we do not compare them with normal constants.
Finally, we have to argue about the blowup of the number of states from a GBA to a BA via the Round-Robin Construction. The blowup of the mentioned construction depends on the number of final sets of its input GBA. In the Tableau Construction, this set only depends on the number of $\mathcal{U}$-formulas that is the same for each equivalence except for the Distributivity between Eventually and Disjunction, which we consider seperately. Since the same number of acceptance sets causes the multiplication of both state spaces of an equivalence with the same factor, this construction will not change our results over GBA and, therefore, it is enough to just observe Generalized Büchi Automata, except for the mentioned distributivity law.

3.1.2 Commutativity of Eventually and Next

As previously mentioned, we consider proofs by structural induction and, therefore, have to observe all operators of LTL which leads to a total of fourteen cases for an equivalence with one variable. However, since there is no difference between most of the cases, we just observe the inputs $\psi_1 \,\mathcal{U}\, \psi_2$ and $\psi_1 \,\mathcal{R}\, \psi_2$ for this subsection to give an idea for how such proofs work. Note also that we only consider the whole computation for this case, since for larger formulas we achieve tables with a oversized amount of nodes. For such computations we only observe the resulting GBAs. Furthermore, as we already argued in section 3.1.1, we do not consider the acceptance sets of our GBAs in here since they are negligible for our number of states and build huge sets of nodes.
Table 3.1 shows the computation of the formula $\Diamond(\bigcirc(\psi_1 \,\mathcal{U}\, \psi_2))$ in the Tableau Construction.

As mentioned in section 3.1.1 we are not able to completely expand some nodes because of the arbitrary formulas. Such nodes are marked by an empty *Added* field. Its transformation to a GBA can be observed in Figure 3.1. We compare it with Table 3.2 that builds the computation of the equivalent formula $\bigcirc(\diamondsuit(\psi_1 \,\mathcal{U}\, \psi_2))$ and its GBA in Figure 3.2. The main difference of both computations can be observed by comparing the nodes N_6 of both tables. While both got the same *New* and *Next* fields, the additional formula $true\,\mathcal{U}(\psi_1 \,\mathcal{U}\, \psi_2)$ in the *Old* field of the second node causes a difference between the nodes N_7, N_{10} and N_8, N_{14}. Therefore, we got two additional nodes for the computation of $\bigcirc(\diamondsuit(\psi_1 \,\mathcal{U}\, \psi_2))$. Their related automata show the difference more clearly, where the node $TC(\psi_1)[N_7]$ builds the combination of the node pair N_7, N_{10} and $TC(\psi_2)[N_{11}]$ the combination of N_8, N_{14}.
Those computations yield the following upper bounds for the size of the state space in their GBAs, where $x_y \geq 0$ describes the related black box with the *New* field y for y being any formula:

$$\begin{array}{lccc} & |TC(\diamondsuit(\bigcirc(\psi_1 \,\mathcal{U}\, \psi_2)))| & \leq & |TC(\bigcirc(\diamondsuit(\psi_1 \,\mathcal{U}\, \psi_2)))| \\ \Leftrightarrow & 2 + x_{\psi_1} + x_{\psi_2} & \leq & 2 + 2 \cdot x_{\psi_1} + 2 \cdot x_{\psi_2} \\ \Leftrightarrow & x_{\psi_1} + x_{\psi_2} & \leq & 2 \cdot x_{\psi_1} + 2 \cdot x_{\psi_2} \\ \Leftrightarrow & 0 & \leq & x_{\psi_1} + x_{\psi_2} \end{array}$$

The computations of the second case $\psi_1 \,\mathcal{R}\, \psi_2$ can be observed in Table 3.3 and 3.4. We discard the related automata since this case is very similar to the previous one. Again, by comparing the nodes N_6 of both tables we can conclude that the additional nodes of the second automaton are caused by the additional formula $true\,\mathcal{U}(\psi_1 \,\mathcal{R}\, \psi_2)$ in its *Old* field.
Therefore we can directly compute the following upper bound for the state space size of their related GBAs:

$$\begin{array}{lccc} & |TC(\diamondsuit(\bigcirc(\psi_1 \,\mathcal{R}\, \psi_2)))| & \leq & |TC(\bigcirc(\diamondsuit(\psi_1 \,\mathcal{R}\, \psi_2)))| \\ \Leftrightarrow & 2 + x_{\psi_1} + x_{\psi_2} & \leq & 2 + 2 \cdot x_{\psi_1} + 2 \cdot x_{\psi_2} \\ \Leftrightarrow & x_{\psi_1} + x_{\psi_2} & \leq & 2 \cdot x_{\psi_1} + 2 \cdot x_{\psi_2} \\ \Leftrightarrow & 0 & \leq & x_{\psi_1} + x_{\psi_2} \end{array}$$

Thus, we can conclude our first result for a state space minimization in the original Tableau Construction. Its proof is stated below as well. Note however, that we just give the whole proof idea once since it is a straight forward structural induction. In the following sections we just consider the cases with the most interesting observations.

Name	Inc	New	Old	Next	Added
N_0	{Init}	$\underline{\{true\,\mathcal{U}(\bigcirc(\psi_1\,\mathcal{U}\,\psi_2))\}}$	$\emptyset$	$\emptyset$	SPLIT
N_1	{N_1,Init}	$\underline{\{true\}}$	{$true\,\mathcal{U}(\bigcirc(\psi_1\,\mathcal{U}\,\psi_2))$,$true$}	{$true\,\mathcal{U}(\bigcirc(\psi_1\,\mathcal{U}\,\psi_2))$}	ADDED
N_3	{N_1}	$\underline{\{true\,\mathcal{U}(\bigcirc(\psi_1\,\mathcal{U}\,\psi_2))\}}$	$\emptyset$	$\emptyset$	SPLIT
N_4	{N_1}	$\underline{\{true\}}$	{$true\,\mathcal{U}(\bigcirc(\psi_1\,\mathcal{U}\,\psi_2))$,$true$}	{$true\,\mathcal{U}(\bigcirc(\psi_1\,\mathcal{U}\,\psi_2))$}	IGNORED
N_5	{N_1,Init}	$\underline{\{\bigcirc(\psi_1\,\mathcal{U}\,\psi_2)\}}$	{$true\,\mathcal{U}(\bigcirc(\psi_1\,\mathcal{U}\,\psi_2))$,$\bigcirc(\psi_1\,\mathcal{U}\,\psi_2)$}	{$\psi_1\,\mathcal{U}\,\psi_2$}	ADDED
N_6	{N_5}	$\underline{\{\psi_1\,\mathcal{U}\,\psi_2\}}$	$\emptyset$	$\emptyset$	SPLIT
N_7	{N_5,N_7}	$\underline{\{\psi_1\}}$	{$\psi_1\,\mathcal{U}\,\psi_2$}	{$\psi_1\,\mathcal{U}\,\psi_2$}	
N_9	{N_7}	$\underline{\{\psi_1\,\mathcal{U}\,\psi_2\}}$	$\emptyset$	$\emptyset$	SPLIT
N_{10}	{N_7}	$\underline{\{\psi_1\}}$	{$\psi_1\,\mathcal{U}\,\psi_2$}	{$\psi_1\,\mathcal{U}\,\psi_2$}	IGNORED
N_{11}	{N_5,N_7}	$\underline{\{\psi_2\}}$	{$\psi_1\,\mathcal{U}\,\psi_2$}	$\emptyset$	
N_8	{N_5}	$\underline{\{\psi_2\}}$	{$\psi_1\,\mathcal{U}\,\psi_2$}	$\emptyset$	IGNORED
N_2	{Init}	$\underline{\{\bigcirc(\psi_1\,\mathcal{U}\,\psi_2)\}}$	{$true\,\mathcal{U}(\bigcirc(\psi_1\,\mathcal{U}\,\psi_2))$,$\bigcirc(\psi_1\,\mathcal{U}\,\psi_2)$}	{$\psi_1\,\mathcal{U}\,\psi_2$}	IGNORED

Table 3.1: Computation Tableau of $\Diamond(\bigcirc(\psi_1\,\mathcal{U}\,\psi_2))$

Name	Inc	New	Old	Next	Added
N_0	{Init}	$\underline{\{\bigcirc(true\,\mathcal{U}(\psi_1\,\mathcal{U}\,\psi_2))\}}$	{$\bigcirc(true\,\mathcal{U}(\psi_1\,\mathcal{U}\,\psi_2))$}	{$true\,\mathcal{U}(\psi_1\,\mathcal{U}\,\psi_2)$}	ADDED
N_1	{N_0}	$\underline{\{true\,\mathcal{U}(\psi_1\,\mathcal{U}\,\psi_2)\}}$	$\emptyset$	$\emptyset$	SPLIT
N_2	{N_0,N_2}	$\underline{\{true\}}$	{$true\,\mathcal{U}(\psi_1\,\mathcal{U}\,\psi_2)$,$true$}	{$true\,\mathcal{U}(\psi_1\,\mathcal{U}\,\psi_2)$}	ADDED
N_4	{N_2}	$\underline{\{true\,\mathcal{U}(\psi_1\,\mathcal{U}\,\psi_2)\}}$	$\emptyset$	$\emptyset$	SPLIT
N_5	{N_2}	$\underline{\{true\}}$	{$true\,\mathcal{U}(\psi_1\,\mathcal{U}\,\psi_2)$,$true$}	{$true\,\mathcal{U}(\psi_1\,\mathcal{U}\,\psi_2)$}	IGNORED
N_6	{N_2}	$\underline{\{\psi_1\,\mathcal{U}\,\psi_2\}}$	{$true\,\mathcal{U}(\psi_1\,\mathcal{U}\,\psi_2)$}	$\emptyset$	SPLIT
N_7	{N_0,N_2}	$\underline{\{\psi_1\}}$	{$true\,\mathcal{U}(\psi_1\,\mathcal{U}\,\psi_2)$,$\psi_1\,\mathcal{U}\,\psi_2$}	{$\psi_1\,\mathcal{U}\,\psi_2$}	
N_9	{N_7}	$\underline{\{\psi_1\,\mathcal{U}\,\psi_2\}}$	$\emptyset$	$\emptyset$	SPLIT
N_{10}	{N_{10},N_7}	$\underline{\{\psi_1\}}$	{$\psi_1\,\mathcal{U}\,\psi_2$}	{$\psi_1\,\mathcal{U}\,\psi_2$}	
N_{12}	{N_{10}}	$\underline{\{\psi_1\,\mathcal{U}\,\psi_2\}}$	$\emptyset$	$\emptyset$	SPLIT
N_{13}	{N_{10}}	$\underline{\{\psi_1\}}$	{$\psi_1\,\mathcal{U}\,\psi_2$}	{$\psi_1\,\mathcal{U}\,\psi_2$}	IGNORED
N_{14}	{N_{10},N_7}	$\underline{\{\psi_2\}}$	{$\psi_1\,\mathcal{U}\,\psi_2$}	$\emptyset$	
N_{11}	{N_7}	$\underline{\{\psi_2\}}$	{$\psi_1\,\mathcal{U}\,\psi_2$}	$\emptyset$	IGNORED
N_8	{N_0,N_2}	$\underline{\{\psi_2\}}$	{$true\,\mathcal{U}(\psi_1\,\mathcal{U}\,\psi_2)$,$\psi_1\,\mathcal{U}\,\psi_2$}	$\emptyset$	
N_3	{N_0}	$\underline{\{\psi_1\,\mathcal{U}\,\psi_2\}}$	{$true\,\mathcal{U}(\psi_1\,\mathcal{U}\,\psi_2)$}	$\emptyset$	SPLIT
N_{15}	{N_0}	$\underline{\{\psi_1\}}$	{$true\,\mathcal{U}(\psi_1\,\mathcal{U}\,\psi_2)$,$\psi_1\,\mathcal{U}\,\psi_2$}	{$\psi_1\,\mathcal{U}\,\psi_2$}	IGNORED
N_{16}	{N_0}	$\underline{\{\psi_2\}}$	{$true\,\mathcal{U}(\psi_1\,\mathcal{U}\,\psi_2)$,$\psi_1\,\mathcal{U}\,\psi_2$}	$\emptyset$	IGNORED

Table 3.2: Computation Tableau of $\bigcirc(\Diamond(\psi_1\,\mathcal{U}\,\psi_2))$

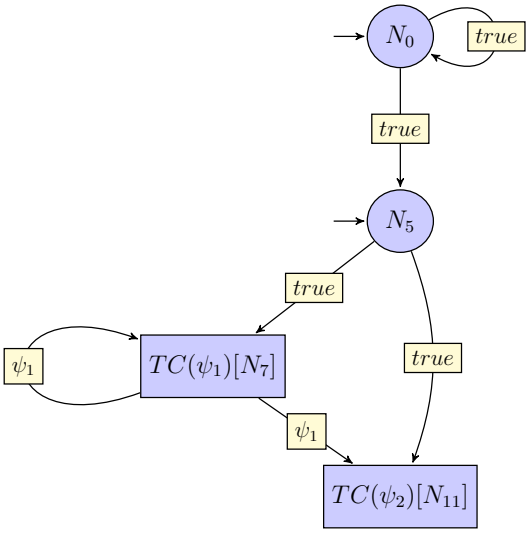

Figure 3.1: GBA of $\Diamond(\bigcirc(\psi_1 \,\mathcal{U}\, \psi_2))$ via TCO

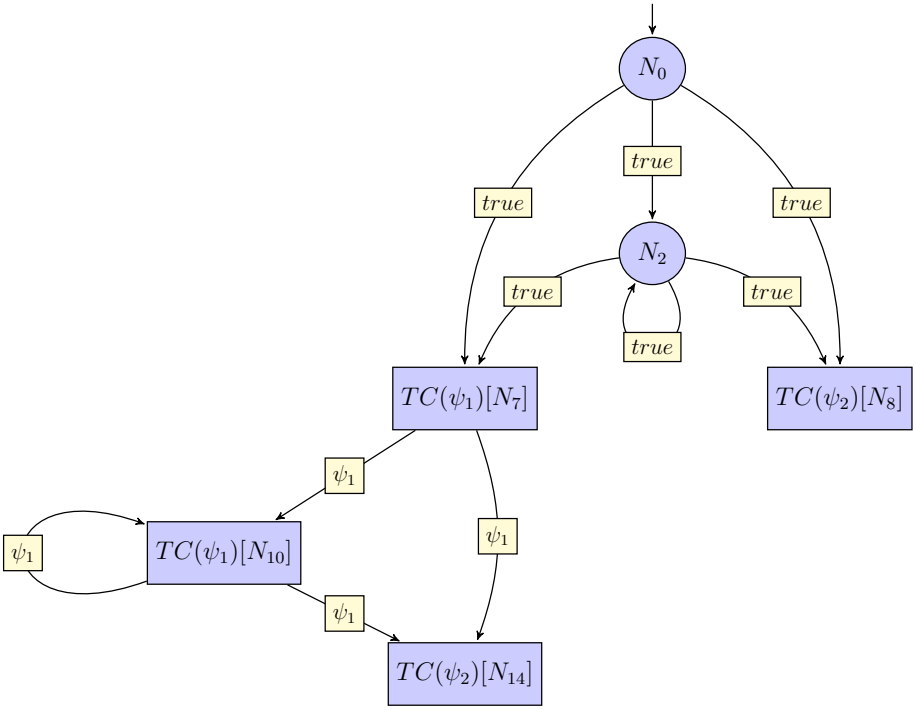

Figure 3.2: GBA of $\bigcirc(\Diamond(\psi_1 \,\mathcal{U}\, \psi_2))$ via TCO

Name	Inc	New	Old	Next	Added
N_0	{Init}	$\underline{\{true\,\mathcal{U}(\bigcirc(\psi_1\,\mathcal{R}\,\psi_2))\}}$	$\emptyset$	$\emptyset$	SPLIT
N_1	{N_1,Init}	$\underline{\{true\}}$	$\{true\,\mathcal{U}(\bigcirc(\psi_1\,\mathcal{R}\,\psi_2)),true\}$	$\{true\,\mathcal{U}(\bigcirc(\psi_1\,\mathcal{R}\,\psi_2))\}$	ADDED
N_3	{N_1}	$\underline{\{true\,\mathcal{U}(\bigcirc(\psi_1\,\mathcal{R}\,\psi_2))\}}$	$\emptyset$	$\emptyset$	SPLIT
N_4	{N_1}	$\underline{\{true\}}$	$\{true\,\mathcal{U}(\bigcirc(\psi_1\,\mathcal{R}\,\psi_2)),true\}$	$\{true\,\mathcal{U}(\bigcirc(\psi_1\,\mathcal{R}\,\psi_2))\}$	IGNORED
N_5	{N_1,Init}	$\underline{\{\bigcirc(\psi_1\,\mathcal{R}\,\psi_2)\}}$	$\{true\,\mathcal{U}(\bigcirc(\psi_1\,\mathcal{R}\,\psi_2)),\bigcirc(\psi_1\,\mathcal{R}\,\psi_2)\}$	$\{\psi_1\,\mathcal{R}\,\psi_2\}$	ADDED
N_6	{N_5}	$\underline{\{\psi_1\,\mathcal{R}\,\psi_2\}}$	$\emptyset$	$\emptyset$	SPLIT
N_7	{N_5,N_7}	$\underline{\{\psi_2\}}$	$\{\psi_1\,\mathcal{R}\,\psi_2\}$	$\{\psi_1\,\mathcal{R}\,\psi_2\}$	
N_9	{N_7}	$\underline{\{\psi_1\,\mathcal{R}\,\psi_2\}}$	$\emptyset$	$\emptyset$	SPLIT
N_{10}	{N_7}	$\underline{\{\psi_2\}}$	$\{\psi_1\,\mathcal{R}\,\psi_2\}$	$\{\psi_1\,\mathcal{R}\,\psi_2\}$	IGNORED
N_{11}	{N_5,N_7}	$\underline{\{\psi_1,\psi_2\}}$	$\{\psi_1\,\mathcal{R}\,\psi_2\}$	$\emptyset$	
N_8	{N_5}	$\underline{\{\psi_1,\psi_2\}}$	$\{\psi_1\,\mathcal{R}\,\psi_2\}$	$\emptyset$	IGNORED
N_2	{Init}	$\underline{\{\bigcirc(\psi_1\,\mathcal{R}\,\psi_2)\}}$	$\{true\,\mathcal{U}(\bigcirc(\psi_1\,\mathcal{R}\,\psi_2)),\bigcirc(\psi_1\,\mathcal{R}\,\psi_2)\}$	$\{\psi_1\,\mathcal{R}\,\psi_2\}$	IGNORED

Table 3.3: Computation Tableau of $\Diamond(\bigcirc(\psi_1\,\mathcal{R}\,\psi_2))$

Name	Inc	New	Old	Next	Added
N_0	{Init}	$\underline{\{\bigcirc(true\,\mathcal{U}(\psi_1\,\mathcal{R}\,\psi_2))\}}$	$\{\bigcirc(true\,\mathcal{U}(\psi_1\,\mathcal{R}\,\psi_2))\}$	$\{true\,\mathcal{U}(\psi_1\,\mathcal{R}\,\psi_2)\}$	ADDED
N_1	{N_0}	$\underline{\{true\,\mathcal{U}(\psi_1\,\mathcal{R}\,\psi_2)\}}$	$\emptyset$	$\emptyset$	SPLIT
N_2	{N_0,N_2}	$\underline{\{true\}}$	$\{true\,\mathcal{U}(\psi_1\,\mathcal{R}\,\psi_2),true\}$	$\{true\,\mathcal{U}(\psi_1\,\mathcal{R}\,\psi_2)\}$	ADDED
N_4	{N_2}	$\underline{\{true\,\mathcal{U}(\psi_1\,\mathcal{R}\,\psi_2)\}}$	$\emptyset$	$\emptyset$	SPLIT
N_5	{N_2}	$\underline{\{true\}}$	$\{true\,\mathcal{U}(\psi_1\,\mathcal{R}\,\psi_2),true\}$	$\{true\,\mathcal{U}(\psi_1\,\mathcal{R}\,\psi_2)\}$	IGNORED
N_6	{N_2}	$\underline{\{\psi_1\,\mathcal{R}\,\psi_2\}}$	$\{true\,\mathcal{U}(\psi_1\,\mathcal{R}\,\psi_2)\}$	$\emptyset$	SPLIT
N_7	{N_0,N_2}	$\underline{\{\psi_2\}}$	$\{true\,\mathcal{U}(\psi_1\,\mathcal{R}\,\psi_2),\psi_1\,\mathcal{R}\,\psi_2\}$	$\{\psi_1\,\mathcal{R}\,\psi_2\}$	
N_9	{N_7}	$\underline{\{\psi_1\,\mathcal{R}\,\psi_2\}}$	$\emptyset$	$\emptyset$	SPLIT
N_{10}	{N_{10},N_7}	$\underline{\{\psi_2\}}$	$\{\psi_1\,\mathcal{R}\,\psi_2\}$	$\{\psi_1\,\mathcal{R}\,\psi_2\}$	
N_{12}	{N_{10}}	$\underline{\{\psi_1\,\mathcal{R}\,\psi_2\}}$	$\emptyset$	$\emptyset$	SPLIT
N_{13}	{N_{10}}	$\underline{\{\psi_2\}}$	$\{\psi_1\,\mathcal{R}\,\psi_2\}$	$\{\psi_1\,\mathcal{R}\,\psi_2\}$	IGNORED
N_{14}	{N_{10},N_7}	$\underline{\{\psi_1,\psi_2\}}$	$\{\psi_1\,\mathcal{R}\,\psi_2\}$	$\emptyset$	
N_{11}	{N_7}	$\underline{\{\psi_1,\psi_2\}}$	$\{\psi_1\,\mathcal{R}\,\psi_2\}$	$\emptyset$	IGNORED
N_8	{N_0,N_2}	$\underline{\{\psi_1,\psi_2\}}$	$\{true\,\mathcal{U}(\psi_1\,\mathcal{R}\,\psi_2),\psi_1\,\mathcal{R}\,\psi_2\}$	$\emptyset$	
N_3	{N_0}	$\underline{\{\psi_1\,\mathcal{R}\,\psi_2\}}$	$\{true\,\mathcal{U}(\psi_1\,\mathcal{R}\,\psi_2)\}$	$\emptyset$	SPLIT
N_{15}	{N_0}	$\underline{\{\psi_2\}}$	$\{true\,\mathcal{U}(\psi_1\,\mathcal{R}\,\psi_2),\psi_1\,\mathcal{R}\,\psi_2\}$	$\{\psi_1\,\mathcal{R}\,\psi_2\}$	IGNORED
N_{16}	{N_0}	$\underline{\{\psi_1,\psi_2\}}$	$\{true\,\mathcal{U}(\psi_1\,\mathcal{R}\,\psi_2),\psi_1\,\mathcal{R}\,\psi_2\}$	$\emptyset$	IGNORED

Table 3.4: Computation Tableau of $\bigcirc(\Diamond(\psi_1\,\mathcal{R}\,\psi_2))$

Theorem 1. *Given an LTL formula $\varphi = \bigcirc(\Diamond\psi)$ for an arbitrary LTL formula ψ, one can reduce the number of states of a BA created by the Tableau Construction without any improvement from φ by transforming φ to its equivalent representation $\varphi = \Diamond(\bigcirc\psi)$.*

Proof 1. *By structural induction over ψ. We compare for each possible structure of ψ both sides of the equivalence $\bigcirc(\Diamond\psi) \equiv \Diamond(\bigcirc\psi)$.*

- *($\psi = a$) If we apply the Tableau Construction to the equivalent formulas $\bigcirc(\Diamond a)$ and $\Diamond(\bigcirc a)$, we end up in the same number of states of four for their GBA and, therefore, we conclude that they always have the same size. The same holds for $\psi = \neg a$.*
- *($\psi = \psi_1 \wedge \psi_2$) Applying the Tableau Construction to both sides ends up in the same upper bound computations of $2 + x_{\psi_1,\psi_2}$. Therefore, both GBAs have the same size.*
- *($\psi = \psi_1 \vee \psi_2$) Both formulas result in the same number of states with $2 + x_{\psi_1} + x_{\psi_2}$.*
- *($\psi = \bigcirc\psi_1$) Both formulas have the upper bounds $3 + x_{\psi_1}$.*
- *($\psi = \psi_1 \,\mathcal{U}\, \psi_2$) As mentioned above, the formula $\Diamond(\bigcirc a)$ got less states on its upper bound as its equivalent formula $\bigcirc(\Diamond a)$. Thus, one should always prefer the first representation.*
- *($\psi = \psi_1 \,\mathcal{R}\, \psi_2$) Here we achieve the same observations as for $\psi_1 \,\mathcal{U}\, \psi_2$.*

Finally, since each case of our structural inductions either prefers none of its representations or prefers the representation $\Diamond(\bigcirc\psi_1)$, we can conclude our result.

□

However, by using the improved Tableau Construction, described in 2.3.1, one can get rid of the additional formula in the *Old* field and, therefore, we do not observe any differences between the formulas $\Diamond(\bigcirc\psi)$ and $\bigcirc(\Diamond\psi)$ for an arbitrary formula ψ. In Table 3.5, the computation with the improved algorithm of the Tableau Construction for the formula $\bigcirc(\Diamond(\psi_1 \,\mathcal{U}\, \psi_2))$ is described. Thus, we can compute a new upper bound for the size of its state set as $|TC(\bigcirc(\Diamond(\psi_1 \,\mathcal{U}\, \psi_2)))| = 2 + x_{\psi_1} + x_{\psi_2}$. This refutes Theorem 1 for the improved version of the Tableau Construction.

Name	Inc	New	Old	Next	Added
N_0	{Init}	$\underline{\{\bigcirc(true\ \mathcal{U}(\psi_1\ \mathcal{U}\ \psi_2))\}}$	$\{\bigcirc(true\ \mathcal{U}(\psi_1\ \mathcal{U}\ \psi_2))\}$	$\{true\ \mathcal{U}(\psi_1\ \mathcal{U}\ \psi_2)\}$	ADDED
N_1	$\{N_0\}$	$\underline{\{true\ \mathcal{U}(\psi_1\ \mathcal{U}\ \psi_2)\}}$	$\emptyset$	$\emptyset$	SPLIT
N_2	$\{N_0,N_2\}$	$\underline{\{true\}}$	$\{true\ \mathcal{U}(\psi_1\ \mathcal{U}\ \psi_2),true\}$	$\{true\ \mathcal{U}(\psi_1\ \mathcal{U}\ \psi_2)\}$	ADDED
N_4	$\{N_2\}$	$\underline{\{true\ \mathcal{U}(\psi_1\ \mathcal{U}\ \psi_2)\}}$	$\emptyset$	$\emptyset$	SPLIT
N_5	$\{N_2\}$	$\underline{\{true\}}$	$\{true\ \mathcal{U}(\psi_1\ \mathcal{U}\ \psi_2),true\}$	$\{true\ \mathcal{U}(\psi_1\ \mathcal{U}\ \psi_2)\}$	IGNORED
N_6	$\{N_2\}$	$\underline{\{\psi_1\ \mathcal{U}\ \psi_2\}}$	$\emptyset$	$\emptyset$	SPLIT
N_7	$\{N_0,N_2,N_7\}$	$\underline{\{\psi_1\}}$	$\{\psi_1\ \mathcal{U}\ \psi_2\}$	$\{\psi_1\ \mathcal{U}\ \psi_2\}$	
N_9	$\{N_7\}$	$\underline{\{\psi_1\ \mathcal{U}\ \psi_2\}}$	$\emptyset$	$\emptyset$	SPLIT
N_{10}	$\{N_7\}$	$\underline{\{\psi_1\}}$	$\{\psi_1\ \mathcal{U}\ \psi_2\}$	$\{\psi_1\ \mathcal{U}\ \psi_2\}$	IGNORED
N_{11}	$\{N_0,N_2,N_7\}$	$\underline{\{\psi_2\}}$	$\{\psi_1\ \mathcal{U}\ \psi_2\}$	$\emptyset$	
N_8	$\{N_2\}$	$\underline{\{\psi_2\}}$	$\{\psi_1\ \mathcal{U}\ \psi_2\}$	$\emptyset$	IGNORED
N_3	$\{N_0\}$	$\underline{\{\psi_1\ \mathcal{U}\ \psi_2\}}$	$\emptyset$	$\emptyset$	SPLIT
N_{12}	$\{N_0\}$	$\underline{\{\psi_1\}}$	$\{\psi_1\ \mathcal{U}\ \psi_2\}$	$\{\psi_1\ \mathcal{U}\ \psi_2\}$	IGNORED
N_{13}	$\{N_0\}$	$\underline{\{\psi_2\}}$	$\{\psi_1\ \mathcal{U}\ \psi_2\}$	$\emptyset$	IGNORED

Table 3.5: Computation Tableau of $\bigcirc(\Diamond(\psi_1\ \mathcal{U}\ \psi_2))$ (Improved)

3.1.3 Commutativity of Always and Next

For any arbitrary formula φ, the formulas $\Box(\bigcirc\varphi)$ and $\bigcirc(\Box\varphi)$ result in the same BA by our Tableau Construction and its improved version. For both cases we can compute the following upper bounds:

$$\begin{aligned} & |TC(\Box(\bigcirc\varphi))| \leq |TC(\bigcirc(\Box\psi))| \\ \Leftrightarrow\ & 1 + x_{\psi_1} + x_{\psi_2} \leq 1 + x_{\psi_1} + x_{\psi_2} \end{aligned}$$

Therefore we do not prefer any of both cases.

3.1.4 Distributivity of Disjunction and Next

First of all the right side of the equivalence will always cause more fixed states than the left side because of the disjunction as the outer operator. Such a disjunction always produces two initial states. However, for our original Tableau Construction it is not the case that the left side always outperforms the right one. For each equivalence of the form $\bigcirc(\varphi \vee (\psi_1\ \mathcal{U}\ \psi_2)) \equiv$

$(\bigcirc\varphi) \vee (\bigcirc(\psi_1 \,\mathcal{U}\, \psi_2)$ and $\bigcirc(\varphi \vee (\psi_1 \,\mathcal{R}\, \psi_2)) \equiv (\bigcirc\varphi) \vee (\bigcirc(\psi_1 \,\mathcal{R}\, \psi_2)$ the right side should always be preferred. This is caused by splitting the disjunction, which is directly followed by another split with either the operator $\mathcal{U}$ or the operator $\mathcal{R}$. As already explained in 3.1.2 such a nested split will create at least two more arbitrary nodes. Figure 3.3 and 3.4 show this property with the example $\varphi = a$ and $\psi = \psi_1 \,\mathcal{U}\, \psi_2$. As previously mentioned, the nodes $TC(\psi_1)[N_6], TC(\psi_1)[N_9]$ and $TC(\psi_2)[N_7], TC(\psi_2)[N_{13}]$ of the automaton on Figure 3.3 correspond to the nodes $TC(\psi_1)[N_7]$ and $TC(\psi_2)[N_{11}]$ of Figure 3.4. For this example we can compute the following upper bounds for $x_{\psi_1}, x_{\psi_2} \geq 1$:

$$\begin{array}{lccc} & |TC(\bigcirc(a \vee (\psi_1 \,\mathcal{U}\, \psi_2)))| & \geq & |TC((\bigcirc a) \vee (\bigcirc(\psi_1 \,\mathcal{U}\, \psi_2)))| \\ \Leftrightarrow & 3 + 2 \cdot x_{\psi_1} + 2 \cdot x_{\psi_2} & \geq & 4 + x_{\psi_1} + x_{\psi_2} \\ \Leftrightarrow & x_{\psi_1} + x_{\psi_2} & \geq & 1 \end{array}$$

So we can conclude another result for the Tableau Construction.

Theorem 2. *Given an LTL formula $\varphi_1 = \bigcirc(\psi_1 \vee \psi_2)$ and its equivalent representation $\varphi_2 = (\bigcirc\psi_1) \vee (\bigcirc\psi_2)$ for arbitrary LTL formulas ψ_1, ψ_2. If at least one of the formulas ψ_1, ψ_2 is an $\mathcal{U}$ or $\mathcal{R}$ formula, one should always prefer the representation of φ_2 to reduce the number of states of the BA resulting from the original usage of the Tableau Construction. Otherwise one should always prefer the representation of φ_1.*

However, the improved version of our Tableau Construction prevents the problem of such a nested split. Therefore, the only distinction between the BA computations of the formulas $\bigcirc(\psi_1 \vee \psi_2)$ and $(\bigcirc\psi_1) \vee (\bigcirc\psi_2)$ with the improved Tableau Construction is located in the different number of inital states that get computed.

Theorem 3. *Given an LTL formula $\varphi = (\bigcirc\psi_1) \vee (\bigcirc\psi_2)$ for arbitrary LTL formulas ψ_1, ψ_2, one can reduce the number of states of a BA created by the improved Tableau Construction from φ by transforming φ to its equivalent representation $\varphi = \bigcirc(\psi_1 \vee \psi_2)$.*

3.1.5 Distributivity of Conjunction and Next

As long as φ or ψ are unequal to an $\mathcal{U}$- or an $\mathcal{R}$-formula, the outcome of both formulas as input of the original Tableau Construction is exactly the same. However, if φ, ψ are

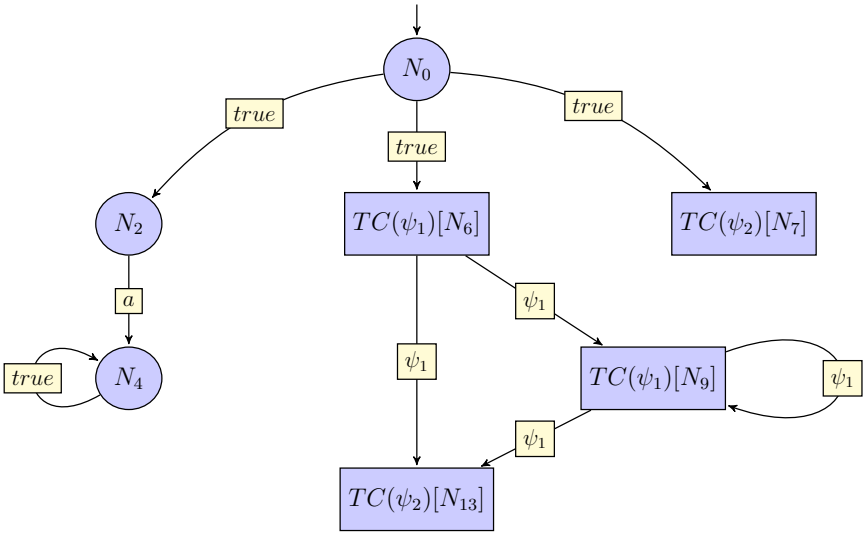

Figure 3.3: GBA of $\bigcirc(a \vee (\psi_1 \mathcal{U} \psi_2))$ by TCO

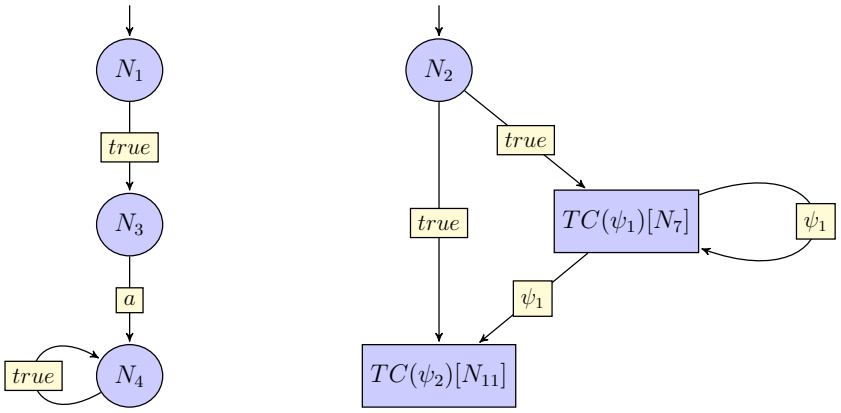

Figure 3.4: GBA of $(\bigcirc a) \vee (\bigcirc(\psi_1 \mathcal{U} \psi_2))$ by TCO

both either an $\mathcal{U}$ or an $\mathcal{R}$ formula, the created GBA of the formula $(\bigcirc\varphi) \wedge (\bigcirc\psi)$ will outperform the formula $\bigcirc(\varphi \wedge \psi)$ in the number of states. As an example, assume the equivalence $\bigcirc((\psi_1 \,\mathcal{U}\, \psi_2) \wedge (\psi_3 \,\mathcal{U}\, \psi_4)) \equiv (\bigcirc(\psi_1 \,\mathcal{U}\, \psi_2)) \wedge (\bigcirc(\psi_3 \,\mathcal{U}\, \psi_4))$. As first step of the expansion the left formula will get rid of its $\bigcirc$-operator, whereas the right one will eleminate its conjunction. However, the following conjunction of the left formula will cause twice as much states with the *New* fields $\{\{\alpha, \beta\} \mid \alpha \in \{\psi_1, \psi_2\}, \beta \in \{\psi_3, \psi_4\}\}$ as the formula on the right side since there is one state with the additional conjunction in its *Old* field and one without, making them different in the criterion of equivalent nodes. Therefore, we can make the following computation for the upper bound of the state space size of both GBAs for $x_{\psi_1}, x_{\psi_2}, x_{\psi_3}, x_{\psi_4}, x_{\psi_2,\psi_3}, x_{\psi_2,\psi_4}, x_{\psi_1,\psi_3}, x_{\psi_1,\psi_3} \geq 1$:

$$
\begin{array}{lrcl}
 & |TC(\bigcirc((\psi_1 \,\mathcal{U}\, \psi_2) \wedge (\psi_3 \,\mathcal{U}\, \psi_4)))| & \geq & |TC((\bigcirc(\psi_1 \,\mathcal{U}\, \psi_2)) \wedge (\bigcirc(\psi_3 \,\mathcal{U}\, \psi_4)))| \\
\Leftrightarrow & 1 + x_{\psi_1} + x_{\psi_2} + x_{\psi_3} + x_{\psi_4} + 2 \cdot x_{\psi_2,\psi_3} + 2 \cdot x_{\psi_2,\psi_4} + 2 \cdot x_{\psi_1,\psi_3} + 2 \cdot x_{\psi_1,\psi_3} & & \\
\geq & 1 + x_{\psi_1} + x_{\psi_2} + x_{\psi_3} + x_{\psi_4} + x_{\psi_2,\psi_3} + x_{\psi_2,\psi_4} + x_{\psi_1,\psi_3} + x_{\psi_1,\psi_3} & & \\
\Leftrightarrow & x_{\psi_2,\psi_3} + x_{\psi_2,\psi_4} + x_{\psi_1,\psi_3} + x_{\psi_1,\psi_3} & & \\
\geq & 0 & &
\end{array}
$$

With this knowledge we can create another result for the given equivalence in the context of the original Tableau Construction:

Theorem 4. *Given an LTL formula $\varphi = \bigcirc(\psi_1 \wedge \psi_2)$ for arbitrary LTL formulas ψ_1, ψ_2. If either $\psi_1 = \psi_3 \,\mathcal{U}\, \psi_4$ or $\psi_1 = \psi_3 \,\mathcal{R}\, \psi_4$ and also either $\psi_2 = \psi_5 \,\mathcal{U}\, \psi_6$ or $\psi_2 = \psi_5 \,\mathcal{R}\, \psi_6$ for $\psi_3, \psi_4, \psi_5, \psi_6$ being arbitrary LTL formulas, one can reduce the number of states of a BA created by the original Tableau Construction without any improvement from φ by transforming φ to its equivalent representation $\varphi = (\bigcirc\psi_1) \wedge (\bigcirc\psi_2)$.*

However, the improved version of the Tableau Construction will not add the conjunction to the *Old* field by its definition. As a consequence, for any arbitrary LTL formula ψ_1, ψ_2, the formulas $\bigcirc(\psi_1 \wedge \psi_2)$ and $(\bigcirc\psi_1) \wedge (\bigcirc\psi_2)$ will produce exactly the same GBAs as output of the improved Tableau Construction.

3.1.6 Distributivity of Until and Next

For our original Tableau Construction, the preferred side of the equivalence $\bigcirc(\varphi \,\mathcal{U}\, \psi) \equiv (\bigcirc\varphi) \,\mathcal{U}\, (\bigcirc\psi)$ may differ for each input. Therefore, it would not make sense to define preferred

rules for each different case of our structural induction itself. Instead we can obtain some interesting results for our improved Tableau Construction, where the left side of our equivalence always outperforms the right one, caused by multiple reasons. Therefore, we observe the GBAs resulting from the improved Tableau Construction with the familar equivalent inputs $\bigcirc(a \,\mathcal{U}\, b)$ and $(\bigcirc a)\,\mathcal{U}(\bigcirc b)$ in Figure 1.1. One may notice that the two initial states of the right automaton could be reduced to one. Their existence is caused by the outer $\mathcal{U}$-formula that always creates two initial nodes because of the definition of a split. Also the pair of nodes N_4, N_8 could be combined to just one node with incoming node N_1, its selfloop N_4 and its outgoing transition to N_9 with label a. The redundant blowup of two states is caused by expansion of a node with *New* field $\{(\bigcirc a)\,\mathcal{U}(\bigcirc b), a\}$. The resulting split of this node creates two addtional added nodes with the previous *New* fields, $\{a, \bigcirc a\}$, $\{a, \bigcirc b\}$, representing N_4, N_8 respectively. Instead, the automaton on its left just creates one node, N_2.

We can make such an observation also for formulas with arbitrary inner formulas, for example, the equivalence $\bigcirc((\bigcirc\psi_1)\,\mathcal{U}\, b) \equiv (\bigcirc(\bigcirc\psi_1))\,\mathcal{U}(\bigcirc b)$ where the computation with the left and right input is shown in Figure 3.5 and Figure 3.6. As mentioned before, the second automaton got two inital states because of the outer $\mathcal{U}$-formula. Also the black box states $TC(\psi_1)[N_7], TC(\psi_1)[N_{11}]$ build the additional blowup as in the previous equivalence and could be reduced to one state as $TC(\psi_1)[N_5]$ in the automaton of $\bigcirc((\bigcirc\psi_1)\,\mathcal{U}\, b)$. Additionally, we notice another redundant state N_5 on the second automaton that is caused by the nested $\bigcirc$-operator that can be avoided by the left side of the equivalence.
The computations of the upper bounds of our last equivalence confirm our assertions for $x_{\psi_1}, x_{\psi_1,b} \geq 1$:

$$\begin{array}{rccc} & |TC(\bigcirc((\bigcirc\psi_1)\,\mathcal{U}\, b)| & \leq & |TC((\bigcirc(\bigcirc\psi_1))\,\mathcal{U}(\bigcirc b))| \\ \Leftrightarrow & 4 + x_{\psi_1} + x_{\psi_1,b} & \leq & 6 + 2 \cdot x_{\psi_1} + x_{\psi_1,b} \\ \Leftrightarrow & 0 & \leq & 2 + x_{\psi_1} + x_{\psi_1,b} \end{array}$$

Theorem 5. *Given an LTL formula $\varphi = (\bigcirc\psi_1)\,\mathcal{U}(\bigcirc\psi_2)$ for arbitrary LTL formulas ψ_1, ψ_2, one can reduce the number of states of a BA created by the improved Tableau Construction from φ by transforming φ to its equivalent representation $\varphi = \bigcirc(\psi_1 \,\mathcal{U}\, \psi_2)$.*

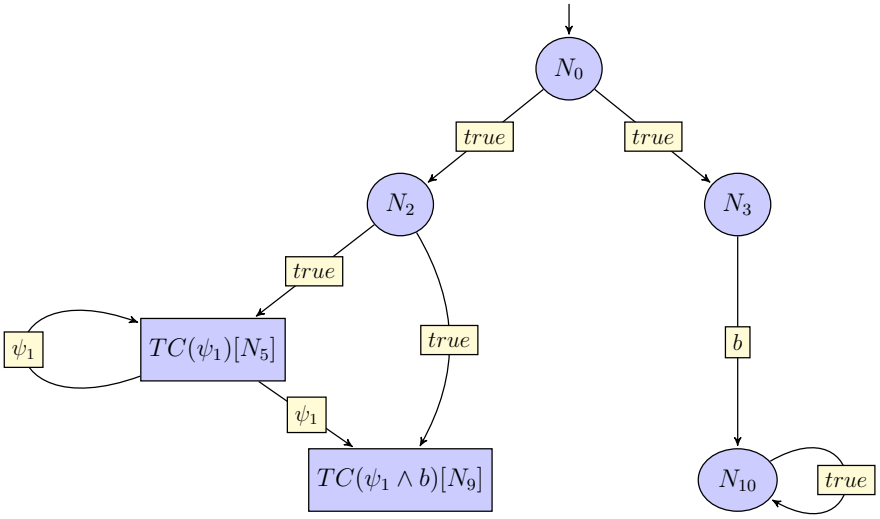

Figure 3.5: GBA of $\bigcirc((\bigcirc\psi_1)\,\mathcal{U}\,b)$ via TCI

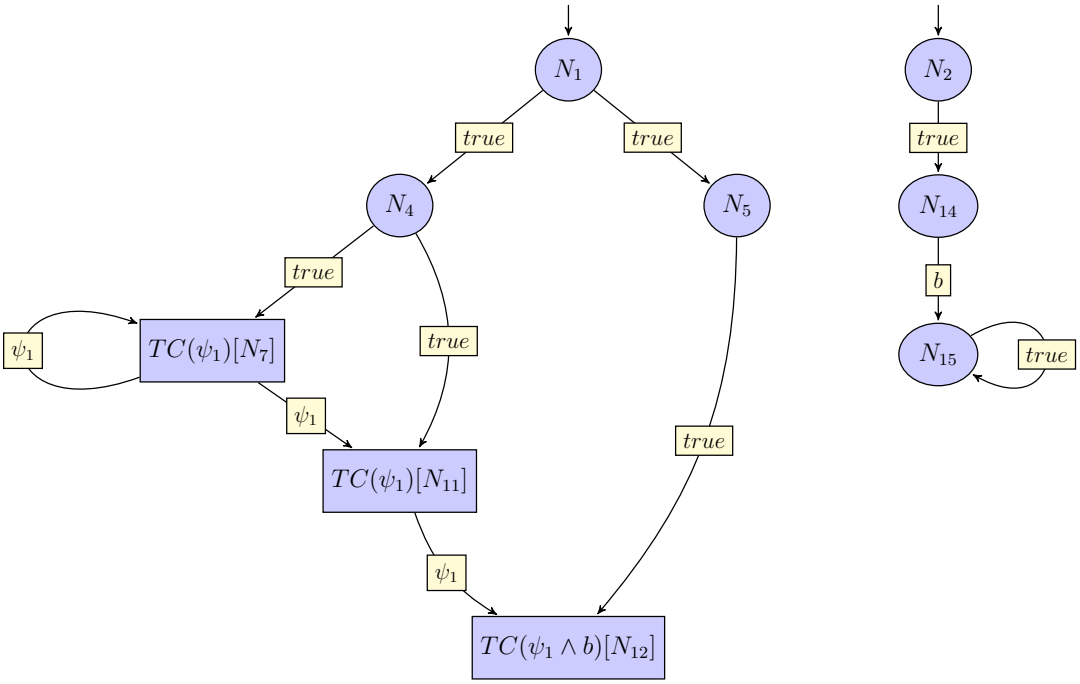

Figure 3.6: GBA of $(\bigcirc(\bigcirc\psi_1)\,\mathcal{U}(\bigcirc b)$ via TCI

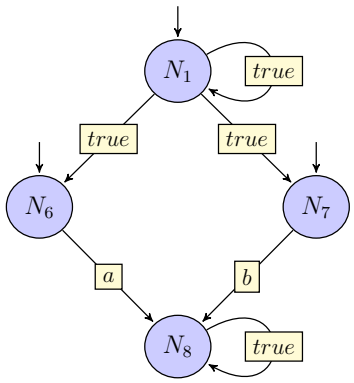

Figure 3.7: GBA of $\Diamond(a \vee b)$ via TCO

3.1.7 Distributivity of Eventually and Disjunction

Given the input equivalence $\Diamond(\varphi \vee \psi) \equiv (\Diamond\varphi) \vee (\Diamond\psi)$ we can observe a huge difference between both sides in the number of states as input of the original Tableau Construction. As an example consider the GBAs for the input formula $\Diamond(a \vee b)$ in Figure 3.7 and 3.8 for the input formula $(\Diamond a) \vee (\Diamond b)$ with the difference of five additional states in the second automaton. All of them are caused by the disjunction as outer formula whose split during the expansion causes the initial states N_3 and N_{14}, as well as N_4 and N_{15}. Respectively, those pairs only differ by an additional disjunction in their *Old* field. Finally, the pairs N_3, N_6 and N_{14}, N_{17} only differ in the disjunction formula in *Old* as well and therefore, it builds this blowup. We could map from the second automaton to the first one by mapping N_3, N_6, N_{14}, N_{17} to N_1, and N_4, N_{14} to N_6 and N_{15}, N_{21} to N_7. Such a mapping is possible for each possibility of our structural induction and therefore we can conclude another result for this thesis.

Theorem 6. *Given an LTL formula $\varphi = (\Diamond\psi_1) \vee (\Diamond\psi_2)$ for arbitrary LTL formulas ψ_1, ψ_2, one can reduce the number of states of a GBA as well as of a BA created by the original Tableau Construction from φ by transforming φ to its equivalent representation $\varphi = \Diamond(\psi_1 \vee \psi_2)$.*

Note that we do not have to observe the final sets of the GBAs since we always choose the formula with the lower number of $\mathcal{U}$-formulas.

However, we can make some interesting observations with the improved version of the Tableau Construction that are very similar to the results of Result 3.1.4 concerning the original Tableau

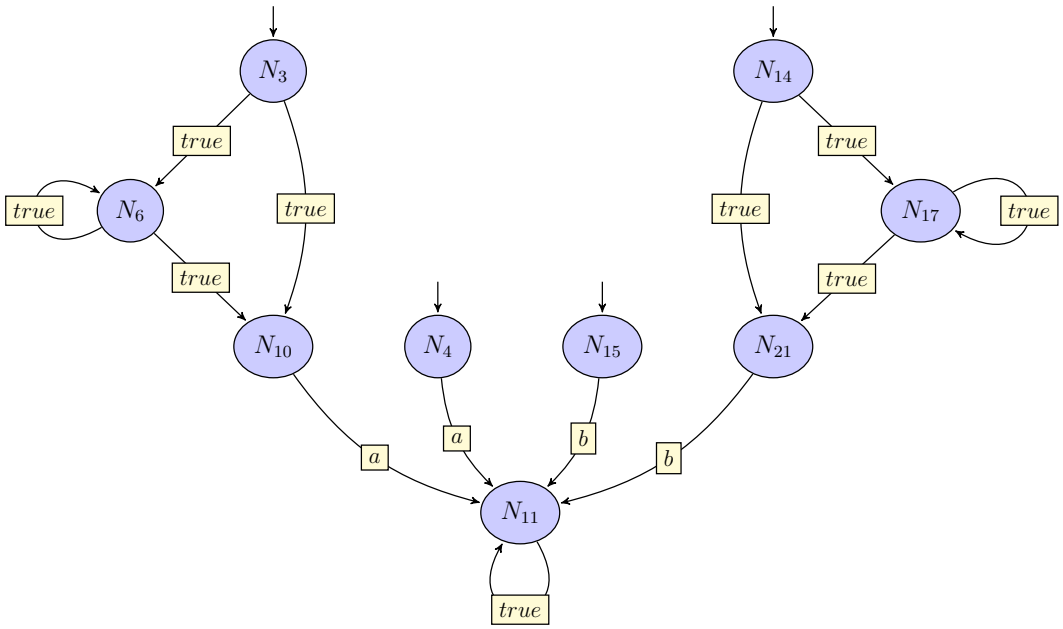

Figure 3.8: GBA of $(\Diamond a) \vee (\Diamond b)$ via TCO

Construction with input $\bigcirc(a \vee (\psi_1 \,\mathcal{U}\, \psi_2)) \equiv (\bigcirc a) \vee (\bigcirc(\psi_1 \,\mathcal{U}\, \psi_2))$. As in Figure 3.4 we observe the same subformulas for input $(\Diamond\psi_1) \vee (\Diamond\psi_2)$ in Figure 3.9. The first observation is that an $\mathcal{U}$ will be represented with two instead of three states in an $\mathcal{U}$-subformula which excludes three redundant states of the non-improved version and also outperforms the outcome of $\Diamond(a \vee (\psi_1 \,\mathcal{U}\, \psi_2))$ in the improved version. Another great advantage is the exclusion of additional states for the inital states that directly got a transition with a or that directly perform either ψ_1 or ψ_2 as in the non-improved version and in the result of the formula $\Diamond(a \vee (\psi_1 \,\mathcal{U}\, \psi_2))$. Thus, whenever an $\mathcal{U}$ or a $\mathcal{R}$ formula is a component in the given equivalence, the right side always outperforms the left side.

With some computation we get the following upper bounds for our current example with $x_{\psi_1}, x_{\psi_2} \geq 1$:

$$\begin{array}{rcc} |TC(\Diamond(a \vee (\psi_1 \,\mathcal{U}\, \psi_2)))| & \geq & |TC((\Diamond a) \vee (\Diamond(\psi_1 \,\mathcal{U}\, \psi_2)))| \\ \Leftrightarrow \quad 3 + 2 \cdot x_{\psi_1} + 2 \cdot x_{\psi_2} & \geq & 4 + x_{\psi_1} + x_{\psi_2} \\ \Leftrightarrow \quad x_{\psi_1} + x_{\psi_2} & \geq & 1 \end{array}$$

Theorem 7. *Given an LTL formula $\varphi_1 = \Diamond(\psi_1 \vee \psi_2)$ and its equivalent representation $\varphi_2 = (\Diamond\psi_1) \vee (\Diamond\psi_2)$ for arbitrary LTL formulas ψ_1, ψ_2. If at least one of the formulas ψ_1, ψ_2 is an $\mathcal{U}$ or $\mathcal{R}$ formula, one should always prefer the representation of φ_2 to reduce the*

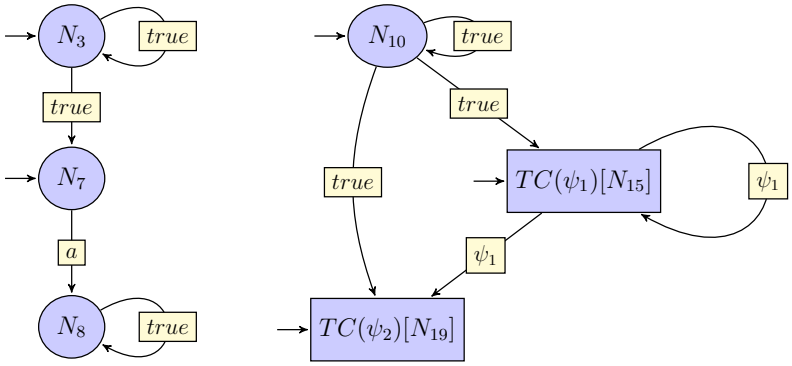

Figure 3.9: GBA of $(\Diamond a) \vee (\Diamond(\psi_1 \,\mathcal{U}\, \psi_2))$ via TCI

*number of states of the **GBA** resulting from the improved Tableau Construction. Otherwise one should always prefer the representation of* φ_1.

Note that we just consider the GBAs for this Theorem since the formula $(\Diamond\psi_1) \vee (\Diamond\psi_2)$ will always produce more states as its equivalent formula in the resulting BA from the Round-Robin Construction because of its additional $\mathcal{U}$ formula.

3.1.8 Distributivity of Always and Conjunction

Because of the additional conjunction in the right formula $(\Box\varphi) \wedge (\Box\psi)$ of our equivalence, each of its automata resulting from the original Tableau Construction got one additional state, one without and one with the conjunction in their *Old* fields. This is caused by expanding a conjunction in combination with the directly followed splitting rules of the $\mathcal{R}$ operators. However, if at least one of the components φ, ψ is an $\bigcirc$ formula, one may notice that both formulas produce exactly the same output. This is the case since the second formula can spare the additional $\bigcirc$ transition because of its nevertheless existing additional state, whereas the first formula is not able to skip it. However, we can conclude that the left formula outperforms the right one for this construction.

Theorem 8. *Given an LTL formula* $\varphi = (\Box\psi_1) \wedge (\Box\psi_2)$ *for arbitrary LTL formulas* ψ_1, ψ_2*, one can reduce the number of states of a BA created by the improved Tableau Construction from* φ *by transforming* φ *to its equivalent representation* $\varphi = \Box(\psi_1 \wedge \psi_2)$.

However, the improved version of the Tableau Construction will not add the conjunction to the *Old* field by its definition. Thus, for any arbitrary LTL formula ψ_1, ψ_2, $\Box(\psi_1 \wedge \psi_2)$ and $(\Box\psi_1) \wedge (\Box\psi_2)$ produce exactly the same GBAs as input of the improved Tableau Construction.

3.1.9 Conclusions on Tableau Construction

Summarizing, we observed many interesting results for the original Tableau Construction as well as for its improved version. All of them are stated in Table 3.6 to get a compact overview over all equivalences that are recommended to apply before starting the construction of the Büchi Automaton. Note again that, different to the original algorithm, some equivalences produce the same number of states in the improved version and therefore, it is not necessary to transfer those rules to the improved Tableau Construction as well.

Equivalences	TC (Original)	TC (Improved)
Commutativity of Eventually and Next	$\bigcirc\Diamond\varphi \Rightarrow \Diamond\bigcirc\varphi$	—
Commutativity of Always and Next	—	—
Distributivity of Disjunction and Next	$\bigcirc(\varphi \vee \psi) \Rightarrow^1 (\bigcirc\varphi) \vee (\bigcirc\psi)$ $(\bigcirc\varphi) \vee (\bigcirc\psi) \Rightarrow^2 \bigcirc(\varphi \vee \psi)$	$\bigcirc(\varphi \vee \psi) \Rightarrow (\bigcirc\varphi) \vee (\bigcirc\psi)$
Distributivity of Conjunction and Next	$\bigcirc(\varphi \wedge \psi) \Rightarrow^1 (\bigcirc\varphi) \wedge (\bigcirc\psi)$	—
Distributivity of Until and Next	—	$(\bigcirc\varphi)\,\mathcal{U}\,(\bigcirc\psi) \Rightarrow \bigcirc(\varphi\,\mathcal{U}\,\psi)$
Distributivity of Eventually and Disjunction	$(\Diamond\varphi) \vee (\Diamond\psi) \Rightarrow \Diamond(\varphi \vee \psi)$	$\Diamond(\varphi \vee \psi) \Rightarrow^{1,4} (\Diamond\varphi) \vee (\Diamond\psi)$ $(\Diamond\varphi) \vee (\Diamond\psi) \Rightarrow^2 \Diamond(\varphi \vee \psi)$
Distributivity of Always and Conjunction	$(\Box\varphi) \wedge (\Box\psi) \Rightarrow \Box(\varphi \wedge \psi)$	—

Table 3.6: Table of Results for Tableau Construction - Original vs. Improved

Annotations for Table 3.6

1 This implication does only hold if φ is an $\mathcal{U}$ / $\mathcal{R}$ - formula **or** ψ is an $\mathcal{U}$ / $\mathcal{R}$ - formula.

2 This implication does hold for all other except the restricted cases.

3 This implication does only hold if φ is an $\mathcal{U}$ / $\mathcal{R}$ - formula **and** ψ is an $\mathcal{U}$ / $\mathcal{R}$ - formula.

4 This implication does only hold for the construction of a GBA.

3.2 GO Construction

3.2.1 Extended Representation of GO Construction

For this construction we have to redefine some of our current definitions to handle arbitrary formulas, starting with our additional operator $\overline{\psi}$ where ψ is some arbitrary undefined formula.

$$\overline{\psi} := \begin{cases} \vdots \\ \overline{\psi} = \{\varphi\} \text{ if } \psi \text{ is an arbitrary formula} \\ \vdots \end{cases}$$

As before we regard our arbitrary formulas as black boxes. So our definition of $\overline{\psi}$ as $\{\psi\}$ itself could also be a conjunction or a disjunction which would lead to a duplication of all ingoing as well as outgoing transitions. However, the number of states can still be represented by a single black box variable. The same holds for our modified transition functions for an arbitrary formula ψ.

$$\Delta := \begin{cases} \vdots \\ \Delta(\psi) = \{(\psi, \emptyset)\} \\ \vdots \end{cases} \qquad \delta := \begin{cases} \vdots \\ \delta(\psi) = \{(\psi, \emptyset)\} \\ \vdots \end{cases}$$

Since we can not make assumtions over successor states of an arbitrary formula in our construction, except for self loops, we define the symbol $\emptyset$ as an undefined state. It is represented via an outgoing transition labeled with its given label that does not lead into any state. However, with such a new state, we have to modify our $\otimes$-operator as well.

For any given tuple $J = (\alpha_1, e)$ we build its conjunction with the state $(\alpha_2, \emptyset)$ by building the conjunction of both label sets α_1, α_2 resulting in a pair with the states set e.

$$(\alpha_1, \emptyset) \otimes (\alpha_2, e) = \begin{cases} (\alpha_1 \wedge \alpha_2, \underline{e}) & \text{if } e \neq \mathit{true} \text{ and no literal} \\ (\alpha_1 \wedge \alpha_2, e) & \text{if } e \text{ is literal} \\ (\alpha_1 \wedge \alpha_2, \emptyset) & \text{if } e = \mathit{true} \end{cases}$$

In the graphical representation, all states e that were combined with our state $\emptyset$ are black box states, denoted by an underline.

As before, we represent such blackbox states graphically with rectangles instead of circles. Their labeling looks like e.g. $GOC(\psi)$, meaning that we observe either the VWAA or the GBA of the output of the GO Construction on the formula ψ. In later computations we also denote the number of states of such black boxes via some variable $x_\psi \geq 1$ where ψ denotes the labeling of the respective state.

As before in the Tableau Construction, the comparison between constant variables and black box variables is not possible, caused by shared states between black boxes as well as the possibility of finding counterexamples. Therefore, we do not compare such variables against each other.

As before for our Tableau Construction, we also have to argue about the final states of our GBA in view of using our Round-Robin Construction to create a BA. After creating a VWAA, we obtain a single state in our coBüchi acceptance condition for each $\mathcal{U}$-formula. Afterwards the acceptance set of our GBA cam be obtained by creating for each such formula in our co-Büchi acceptance set an own set including all conjunction states that do not cover the $\mathcal{U}$-formula. Therefore, we conclude that the size of the GBA acceptance set depends on the number of $\mathcal{U}$-formulas in our NNF formula. Thus, again we only have to consider this case for the distributivity of eventually and conjunction and can ignore the acceptance set of other automata.

3.2.2 Commutativity of Eventually and Next

Based on the extended representation given above, we consider at first the commutativity of eventually and next for our GO Construction. However, for this commutativity law we can only make observations that are sobering.
Starting with the positive results, we know that the formula $\Diamond\bigcirc(\psi_1 \,\mathcal{U}\, \psi_2)$ always outperforms its equivalent representation $\bigcirc\Diamond(\psi_1 \,\mathcal{U}\, \psi_2)$ for arbitrary formulas ψ_1, ψ_2. To prove this, we compute both sides over the VWAA to the GBA, starting with the transitions of the first formula.

$$
\begin{aligned}
& \Delta(true \,\mathcal{U}\, \bigcirc(\psi_1 \,\mathcal{U}\, \psi 2)) \\
= \;& \delta(true \,\mathcal{U}\, \bigcirc(\psi_1 \,\mathcal{U}\, \psi 2)) \\
= \;& \Delta(\bigcirc(\psi_1 \,\mathcal{U}\, \psi_2)) \cup (\Delta(true) \otimes \{(true, true \,\mathcal{U}\, \bigcirc(\psi_1 \,\mathcal{U}\, \psi 2))\}) \\
= \;& \delta(\bigcirc(\psi_1 \,\mathcal{U}\, \psi_2)) \cup (\delta(true) \otimes \{(true, true \,\mathcal{U}\, \bigcirc(\psi_1 \,\mathcal{U}\, \psi 2))\}) \\
= \;& \{(true, e) \mid e \in \overline{\psi_1 \,\mathcal{U}\, \psi_2)}\} \cup (\{(true, true)\} \otimes \{(true, true \,\mathcal{U}\, \bigcirc(\psi_1 \,\mathcal{U}\, \psi 2))\}) \\
= \;& \{(true, \psi_1 \,\mathcal{U}\, \psi_2)\} \cup \{(true, true \,\mathcal{U}\, \bigcirc(\psi_1 \,\mathcal{U}\, \psi 2))\} \\
= \;& \{(true, \psi_1 \,\mathcal{U}\, \psi_2), (true, true \,\mathcal{U}\, \bigcirc(\psi_1 \,\mathcal{U}\, \psi 2))\}
\end{aligned}
$$

$$
\begin{aligned}
& \Delta(\psi_1 \,\mathcal{U}\, \psi_2) \\
= \;& \delta(\psi_1 \,\mathcal{U}\, \psi_2) \\
= \;& \Delta(\psi_2) \cup (\Delta(\psi_1) \otimes \{(true, \psi_1 \,\mathcal{U}\, \psi_2)\}) \\
= \;& \{(\psi_2, \emptyset)\} \cup (\{(\psi_1, \emptyset)\} \otimes \{(true, \psi_1 \,\mathcal{U}\, \psi_2)\}) \\
= \;& \{(\psi_2, \emptyset)\} \cup \{(\psi_1, \underline{\psi_1 \,\mathcal{U}\, \psi_2})\} \\
= \;& \{(\psi_2, \emptyset), (\psi_1, \underline{\psi_1 \,\mathcal{U}\, \psi_2})\}
\end{aligned}
$$

Note that this VWAA only has two states whereas the second one has three as the following computation shows.

$$
\begin{aligned}
& \Delta(\bigcirc(true \,\mathcal{U}\, (\psi_1 \,\mathcal{U}\, \psi_2))) \\
= \;& \delta(\bigcirc(true \,\mathcal{U}\, (\psi_1 \,\mathcal{U}\, \psi_2))) \\
= \;& \{(true, e) \mid e \in \overline{true \,\mathcal{U}\, (\psi_1 \,\mathcal{U}\, \psi_2)}\} \\
= \;& \{(true, true \,\mathcal{U}\, (\psi_1 \,\mathcal{U}\, \psi_2))\}
\end{aligned}
$$

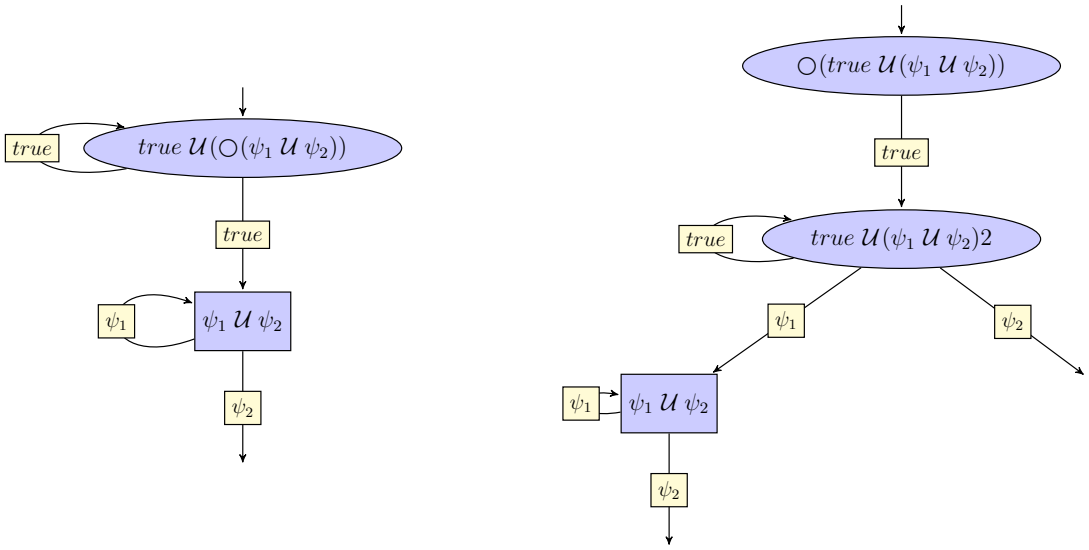

Figure 3.10: GBA for $\Diamond\bigcirc(\psi_1 \,\mathcal{U}\, \psi_2)$ on the left, GBA for $\bigcirc\Diamond(\psi_1 \,\mathcal{U}\, \psi_2)$ on the right

$$
\begin{aligned}
& \Delta(true \,\mathcal{U}\, (\psi_1 \,\mathcal{U}\, \psi_2)) \\
= \;& \delta(true \,\mathcal{U}\, (\psi_1 \,\mathcal{U}\, \psi_2)) \\
= \;& \Delta(\psi_1 \,\mathcal{U}\, \psi_2) \cup (\Delta(true) \otimes \{(true, true \,\mathcal{U}\, (\psi_1 \,\mathcal{U}\, \psi_2))\}) \\
= \;& \{(\psi_2, \emptyset), (\psi_1, \underline{\psi_1 \,\mathcal{U}\, \psi_2})\} \cup (\delta(true) \otimes \{(true, true \,\mathcal{U}\, (\psi_1 \,\mathcal{U}\, \psi_2))\}) \\
= \;& \{(\psi_2, \emptyset), (\psi_1, \underline{\psi_1 \,\mathcal{U}\, \psi_2})\} \cup (\{(true, true)\} \otimes \{(true, true \,\mathcal{U}\, (\psi_1 \,\mathcal{U}\, \psi_2))\}) \\
= \;& \{(\psi_2, \emptyset), (\psi_1, \underline{\psi_1 \,\mathcal{U}\, \psi_2}), (true, true \,\mathcal{U}\, (\psi_1 \,\mathcal{U}\, \psi_2))\}
\end{aligned}
$$

The graphical representations of both GBAs in Figure 3.10 are exactly the same as their VWAA, except for the accepting conditions, since they do not have any alternating transitions. However, while the automaton for $\Diamond\bigcirc(\psi_1 \,\mathcal{U}\, \psi_2)$ directly got a transition from its inital state to $\psi_1 \,\mathcal{U}\, \psi_2$, the other automaton takes a detour to an additional state, caused by the $\bigcirc$-operator at its beginning. Therefore, $\Diamond\bigcirc(\psi_1 \,\mathcal{U}\, \psi_2)$ always outperforms its equivalent formula $\bigcirc\Diamond(\psi_1 \,\mathcal{U}\, \psi_2)$ in the number of states for our GO Construction.
A similar result can be observed for $\psi = \psi_1 \,\mathcal{R}\, \psi_2$, where the left formula of the equivalence should always be preferred. As mentioned before, the observation of this equivalence yields some negative results as well, meaning that we are unable to determine a preferred side for the inputs $\psi = \psi_1 \wedge \psi_2$ and $\psi_1 \vee \psi_2$. We show this by giving two different inputs for ψ_1, ψ_2, where each input prefers the respective other side. Therefore, given the input $\psi_1 = a \,\mathcal{U}\, b$ and $\psi_2 = c \,\mathcal{U}\, d$ for the equivalence $\Diamond\bigcirc(\psi_1 \wedge \psi_2) \equiv \bigcirc\Diamond(\psi_1 \wedge \psi_2)$, the resulting GBA of

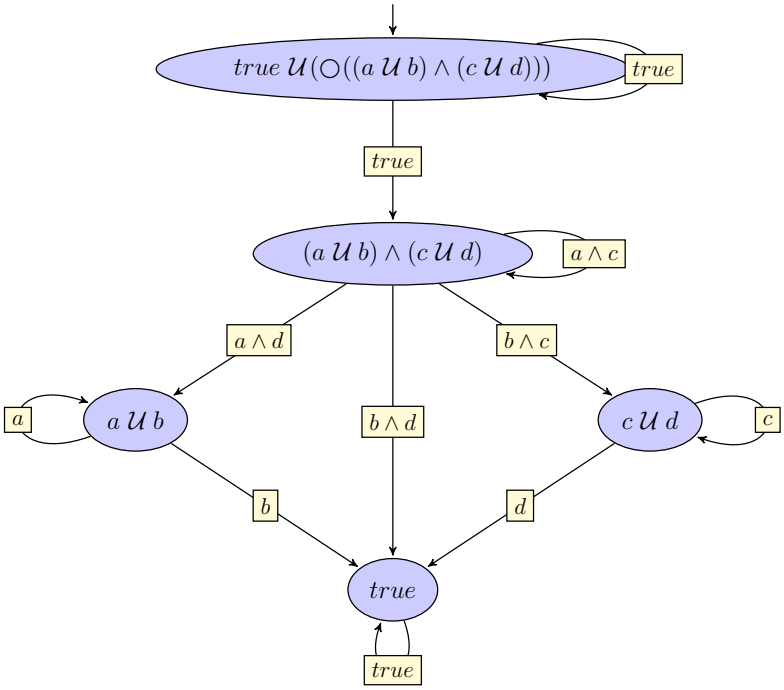

Figure 3.11: GBA of $\Diamond\bigcirc((a \mathcal{U} b) \wedge (c \mathcal{U} d))$ via GOC

the GO Construction on the left side 3.11 got less states (5) than the automaton on the right side (6) 3.12. However, given the input $\psi_1 = a \vee b$ and $\psi_1 = c \vee d$, the right side 3.14 outperforms the left one 3.13 in the number of states by 3 versus 6. Note that we can make similar observations for $\psi = \psi_1 \wedge \psi_2$.

Therefore, we are unable to give a result covering all cases of our structural induction. Instead, we can give partial results that can be proven by structural induction as well.

Theorem 9. *Given an LTL formula $\varphi_1 = \bigcirc(\Diamond\psi)$ and its equivalent representation $\varphi_2 = \Diamond(\bigcirc\psi)$ for an arbitrary LTL formula ψ. If ψ is an $\mathcal{U}$ or $\mathcal{R}$ formula, one should always prefer the representation of φ_2 to reduce the number of states of the BA resulting from the GO construction.*

3.2.3 Commutativity of Always and Next

This equivalence is very similar to the commutativity law before. While we produce useful results for $\mathcal{U}$ and $\mathcal{R}$ formulas, we can not make any assumptions for conjunctions and

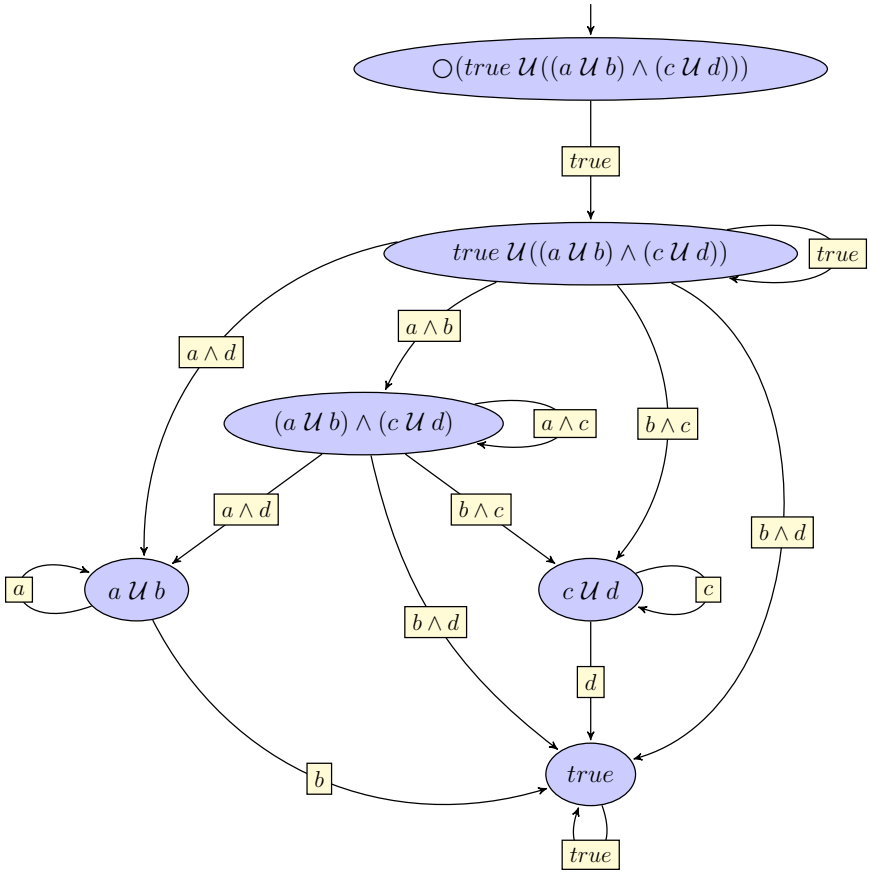

Figure 3.12: GBA of $\bigcirc\Diamond((a\ \mathcal{U}\ b) \wedge (c\ \mathcal{U}\ d))$ via GOC

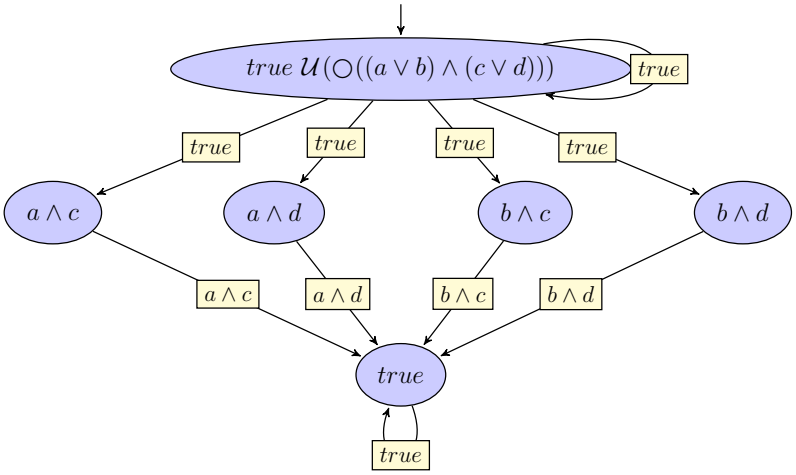

Figure 3.13: GBA of $\Diamond\bigcirc((a \vee b) \wedge (c \vee d))$ via GOC

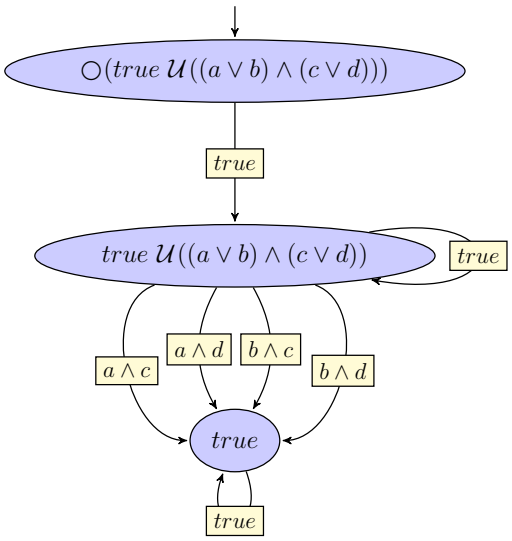

Figure 3.14: GBA of $\bigcirc\Diamond((a \vee b) \wedge (c \vee d))$ via GOC

disjunctions as input. A counter example would be $\psi = \psi_1\ \mathcal{U}\ \psi_2$ and $\psi = \psi_1 \vee \psi_2$, each preferring another side of our equivalence.
However, our partial results lead into another small observation that can be proven again via structural induction.

Theorem 10. *Given an LTL formula $\varphi_1 = \bigcirc(\Box\psi)$ and its equivalent representation $\varphi_2 = \Box(\bigcirc\psi)$ for an arbitrary LTL formula ψ. If ψ is an $\mathcal{U}$ or $\mathcal{R}$ formula, one should always prefer the representation of φ_2 to reduce the number of states of the BA resulting from the GO construction.*

3.2.4 Distributivity of Disjunction and Next

The right side of the equivalence $\bigcirc(\varphi \vee \psi) \equiv (\bigcirc\varphi) \vee (\bigcirc\psi)$ always leads to more states than its equivalent side because of its outer conjunction. By computing the initial states of our VWAA via the overline function, a cunjunction always leads to two initial states by definition. Therefore, one should always prefer to push out the $\bigcirc$ operator in this situation. Figure 3.15 represents this argumentation. While the GBA for $\bigcirc(a \vee b)$ on the left got only one initial state, the GBA for $(\bigcirc a) \vee (\bigcirc b)$ got always two. This leads to another result.

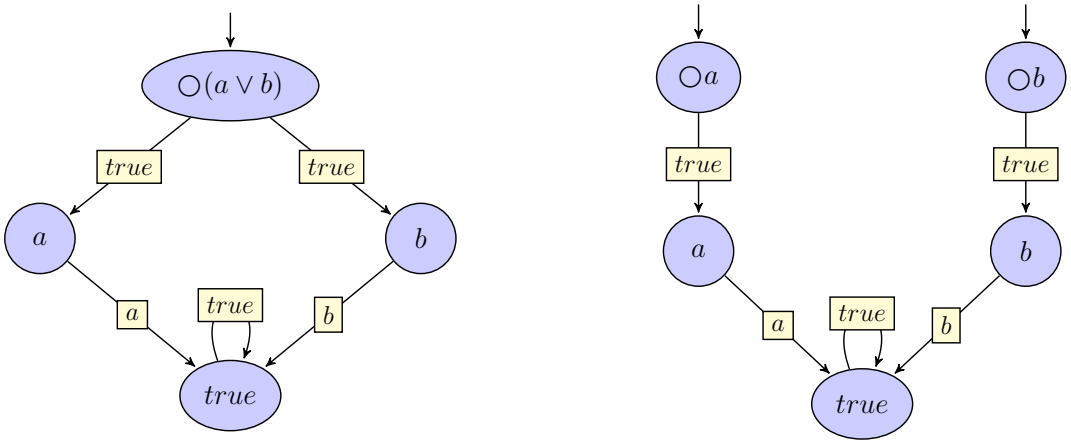

Figure 3.15: GBA for $\bigcirc(a \vee b)$ on the left, GBA for $(\bigcirc a) \vee (\bigcirc b)$ on the right

Theorem 11. *Given an LTL formula $\varphi = (\bigcirc\psi_1) \vee (\bigcirc\psi_2)$ for any arbitrary LTL formulas ψ_1, ψ_2, one can reduce the number of states of a BA created by the GO Construction by transforming φ to its equivalent representation $\varphi = \bigcirc(\psi_1 \vee \psi_2)$.*

3.2.5 Distributivity of Conjunction and Next

For any arbitrary formulas φ, ψ, the formulas $\bigcirc(\varphi \wedge \psi)$ and $(\bigcirc\varphi) \wedge (\bigcirc\psi)$ result in the same BA by our GO Construction. Therefore we do not prefer any of both cases.

3.2.6 Distributivity of Until and Next

For the distributivity of Until and Next we can produce a lot of partial results. However, for some other cases we are unable to determine one as the preferred side of the equivalence because of possible counter examples. Therefore, we only present the different partial results, leaving other cases untouched.

First of all, we notice that whenever one arbitrary formulas of the equivalence $\bigcirc(\varphi \,\mathcal{U}\, \psi) \equiv (\bigcirc\varphi) \,\mathcal{U}\, (\bigcirc\psi)$ is a literal, one should always prefer the left side, keeping the $\bigcirc$-operator outside. Therefore, consider the inputs $\varphi = a$ and $\psi = \psi_1 \wedge \psi_2$ with the graphical representation of their GBA in Figure 3.16. Whereas on the left side we first get rid of the $\bigcirc$-operator, leading

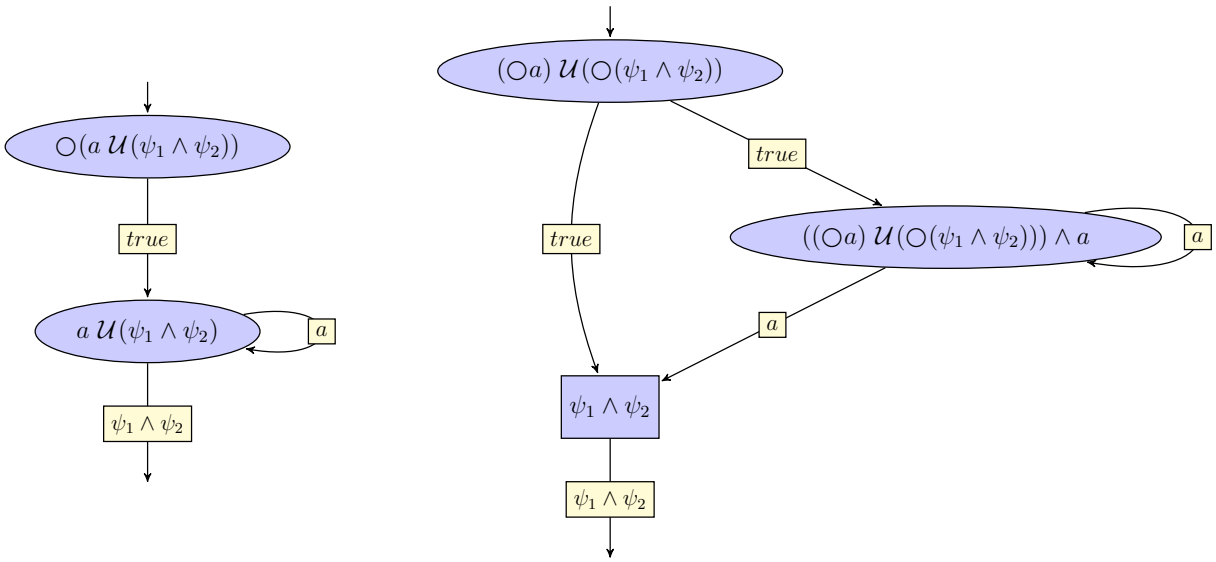

Figure 3.16: GBA for $\bigcirc(a\,\mathcal{U}(\psi_1 \wedge \psi_2)$ on the left, GBA for $(\bigcirc a)\,\mathcal{U}(\bigcirc(\psi_1 \wedge \psi_2))$ on the right

into a single $\mathcal{U}$-formula state, on the right side the inner $\bigcirc$ operators of the $\mathcal{U}$ formula triggers transitions to two different states where it either demands the satisfaction of $\psi_1 \wedge \psi_2$ or ensures the satisfaction of a until satisfying the conjunction. The difference is based on the additional black box state of the right automaton the can be avoided by preventing $\bigcirc$-operators within an $\mathcal{U}$-formula. This observation leads us to our first result that can get proved via structural induction.

Theorem 12. *Given an LTL formula $\varphi_1 = (\bigcirc\psi_1)\,\mathcal{U}(\bigcirc\psi_2)$ and its equivalent representation $\varphi_2 = \bigcirc(\psi_1\,\mathcal{U}\,\psi_2)$ for arbitrary LTL formulas ψ_1, ψ_2. If at least one of the formulas ψ_1, ψ_2 is a literal, one should always prefer the representation of φ_2 to reduce the number of states of the BA resulting from the GO Construction.*

A similar result can be achieved by observing a formula $\bigcirc\psi$ as at least one of the inputs ψ_1, ψ_2. Therefore, consider the automata in Figure 3.17 and Figure 3.18 for the formulas $\bigcirc((\bigcirc\psi)\,\mathcal{U}(\psi_1 \wedge \psi_2))$ and $\bigcirc(\bigcirc\psi)\,\mathcal{U}\,\bigcirc(\psi_1 \wedge \psi_2)$ respectively. Again, the first automaton outperforms the second one which is again the case because of the $\bigcirc$-formula within an $\mathcal{U}$-formula forcing our construction to create additional states instead of direct transitions.

Theorem 13. *Given an LTL formula $\varphi_1 = (\bigcirc\psi_1)\,\mathcal{U}(\bigcirc\psi_2)$ and its equivalent representation $\varphi_2 = \bigcirc(\psi_1\,\mathcal{U}\,\psi_2)$ for arbitrary LTL formulas ψ_1, ψ_2. If at least one of the formulas ψ_1, ψ_2*

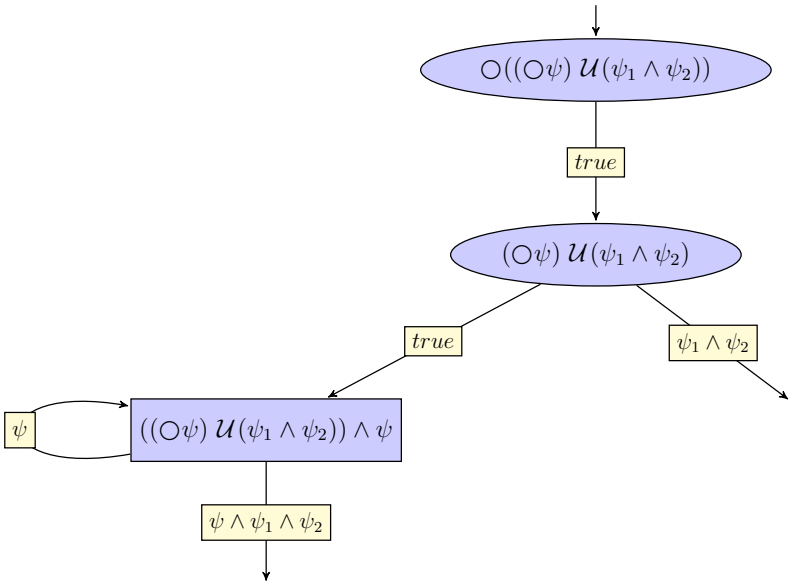

Figure 3.17: GBA of $\bigcirc((\bigcirc\psi)\,\mathcal{U}\,(\psi_1 \wedge \psi_2))$ via GOC

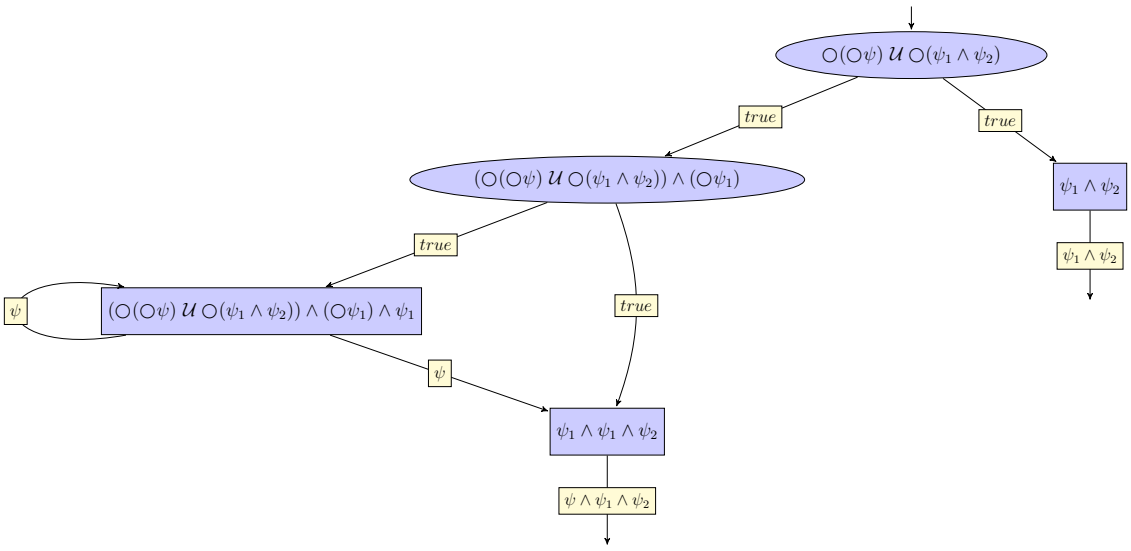

Figure 3.18: GBA of $\bigcirc(\bigcirc\psi)\,\mathcal{U}\,\bigcirc(\psi_1 \wedge \psi_2)$ via GOC

is a formula of the form $\bigcirc\psi$ for some arbitrary LTL formula ψ, one should always prefer the representation of φ_2 to reduce the number of states of the BA resulting from the GO Construction.

Finally we have already seen that an $\mathcal{U}$-formula as direct subformula of another $\mathcal{U}$-subformula can be critical, for example, considering Theorem 3.2.2. Similar to this result, we can make the observation for this equivalence that if both formulas φ, ψ are either an $\mathcal{U}$ or an $\mathcal{R}$ formula, it is beneficial to push in the $\bigcirc$ operators, leading us to our final result for this equivalence.

Theorem 14. *Given an LTL formula $\varphi_2 = \bigcirc(\psi_1 \,\mathcal{U}\, \psi_2)$ and its equivalent representation $\varphi_2 = (\bigcirc\psi_1)\,\mathcal{U}\,(\bigcirc\psi_2)$ for arbitrary LTL formulas ψ_1, ψ_2. If both formulas ψ_1, ψ_2 are either $\mathcal{U}$ or $\mathcal{R}$ formulas, one should always prefer the representation of φ_1 to reduce the number of states of the BA resulting from the GO Construction.*

3.2.7 Distributivity of Eventually and Disjunction

Similar to our observations over the distributivity of Disjunction and Next in 3.2.4, the righthand side of the equivalence $\Diamond(\varphi \vee \psi) \equiv (\Diamond\varphi) \vee (\Diamond\psi)$ always leads to more states than the left-hand side because the outer disjunction always causes multiple initial states. Therefore, we can directly conclude our result for this equivalence.

Theorem 15. *Given an LTL formula $\varphi = (\Diamond\psi_1) \vee (\Diamond\psi_2)$ for any arbitrary LTL formulas ψ_1, ψ_2, one can reduce the number of states of a BA created by the GO Construction by transforming φ to its equivalent representation $\varphi = \Diamond(\psi_1 \vee \psi_2)$.*

Note that this observation also would be the case if we compare the number of $\mathcal{U}$ formulas of both formulas. Since the right side of the equivalence got more such formulas than the left one, we could automatically conclude that the BA resulting from the Round-Robin Construction would produce a multiple of the states of its GBA.

3.2.8 Distributivity of Always and Conjunction

Unfortunately, for each possible input φ, ψ of the equivalence $\Box(\varphi \wedge \psi) \equiv (\Box\varphi) \wedge (\Box\psi)$, the automata always produce the same numer of states, however, always with different black box states that can not be compared. Therefore, it is not possible to determine any side of our equivalence that is more likely.

3.2.9 Conclusions on GO Construction

Summarizing, we observed only a few positive results for the GO construction, because we were able to construct counter examples in many cases. However, again all results are stated in Table 3.7 to get a compact overview over all equivalences that are recommended to apply before starting the construction of the Büchi Automaton.

Equivalences	GO Construction)
Commutativity of Eventually and Next	$\bigcirc(\Diamond\varphi) \Rightarrow^1 \Diamond(\bigcirc\varphi)$
Commutativity of Always and Next	$\bigcirc(\Box\varphi) \Rightarrow^1 \Box(\bigcirc\varphi)$
Distributivity of Disjunction and Next	$(\bigcirc\varphi) \vee (\bigcirc\psi) \Rightarrow \bigcirc(\varphi \vee \psi)$
Distributivity of Conjunction and Next	—
Distributivity of Until and Next	$(\bigcirc\varphi)\, \mathcal{U}\, (\bigcirc\psi) \Rightarrow^2 \bigcirc(\varphi\, \mathcal{U}\, \psi)$ $(\bigcirc\varphi)\, \mathcal{U}\, (\bigcirc\psi) \Rightarrow^3 \bigcirc(\varphi\, \mathcal{U}\, \psi)$ $\bigcirc(\varphi\, \mathcal{U}\, \psi) \Rightarrow^4 (\bigcirc\varphi)\, \mathcal{U}\, (\bigcirc\psi)$
Distributivity of Eventually and Disjunction	$(\Diamond\varphi) \vee (\Diamond\psi) \Rightarrow \Diamond(\varphi \vee \psi)$
Distributivity of Always and Conjunction	—

Table 3.7: Table of Results for GO Construction

Annotations for Table 3.7

1 This implication does only hold if φ is an $\mathcal{U}$ / $\mathcal{R}$ - formula.

2 This implication does only hold if φ is a literal **or** ψ is a literal.

3 This implication does only hold if φ is a $\bigcirc$ - formula **or** ψ is an $\bigcirc$ - formula.

4 This implication does only hold if φ is an $\mathcal{U}$ / $\mathcal{R}$ - formula **and** ψ is an $\mathcal{U}$ / $\mathcal{R}$ - formula.

4 Conclusions

In this thesis we displayed and analyzed two different constructions from LTL specifications to their representing Büchi Automata. Using extended representations in the algorithms of our considered constructions, we found preferable equivalences over LTL that should be applied to the input formula to possibly reduce the number of states of its resulting automaton. Moreover, we gave an idea on how to prove the efficiency of the obtained simplifications by using structural induction.
Primarily, the resulting simplifications of this thesis are highly beneficial for translation tools that are based on one of our considered constructions. Improving those tools by implementing our results will provide automata with a minimized state space, as we proved. Also researches can focus on similar constructions and analyze them in the same manner as presented in this thesis. As well, inspired by the approach of our analyses, it may be possible to transfer them to completely other constructions to find preferable equivalences.

List of Figures

List of Tables

Bibliography

[1] A. Duret-Lutz, D. Poitrenaud. SPOT: an Extensible Model Checking Library using Transition-based Generalized Büchi Automata. *Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS'04)*, pages 76 – 83, 2004.

[2] A. P. Sistla, E. M. Clarke. The complexity of propositional linear temporal logics. *Journal of the ACM (JACM)*, pages 733 – 749, 1985.

[3] Amir Pnueli. The temporal logic of programs. *Proceedings of the 18th Annual Symposium on Foundations of Computer Science (SFCS'77)*, pages 46–57, 1977.

[4] ARM Ltd. Amba specification, 1999. Available from www.arm.com.

[5] C. Baier, J. S. Katoen. *Principles of Model Checking*, chapter Generalized Büchi Automata, pages 195 – 197. The MIT Press, 2008.

[6] E. Grädel, W. Thomas, T. Wilke (Eds.). *Automata, Logics, and Infinite Games*. Springer, 2002.

[7] F. Somenzi, R. Bloem. Efficient Büchi Automata from LTL Formulae. *Computer Aided Verification*, pages 248–263, 2000.

[8] G. Holzmann. The model checker SPIN. *IEEE Transactions on Software Engineering*, pages 279 – 295, 1997.

[9] J. R. Büchi. On a Decision Method in Restricted Second Order Arithmetic. *The Collected Works of J. Richard Büchi*, pages 425 – 435, 1962.

[10] K. Etessami, G. J. Holzmann. Optimizing Büchi automata. *11th International Conference University Park, PA, USA, August 22–25, 2000 Proceedings*, pages 153–168, 2000.

[11] O. Kupferman, M. Y. Vardi. Safraless Decision Procedures. *Proceedings of the 46th Annual IEEE Symposium on Foundations of Computer Science (FOCS'05)*, pages 531 – 542, 2005.

[12] P. Gastin, D. Oddoux. Fast LTL to Büchi Automata Translation. *Computer Aided Verification*, pages 53–65, 2001.

[13] R. Gerth, D. Peled, M. Y. Vardi and P. Wolper. Simple On-the-fly Automatic Verification of Linear Temporal Logic. *Protocol Specification, Testing and Verification XV*, pages 3–18, 1996.

[14] S. Safra. On The Complexity of ω-Automata. *Proceedings of the 29th Annual Symposium on Foundations of Computer Science (SFCS'88)*, pages 319 – 327, 1988.

[15] T. Babiak, M. Křetínský, V. Řehák, J. Stretjček. LTL to Büchi Automata Translation: Fast and More Deterministic. *Tools and Algorithms for the Construction and Analysis of Systems*, pages 95–109, 2012.