5

Temporal Logics

5.1 Introduction

In the previous two chapters we have introduced μ-calculus and ω-automata as basic formalisms for the specification and verification of reactive systems. We have already seen that every formula of our logic of ω-automata $\mathcal{L}_\omega$ can be translated to an equivalent μ-calculus formula, but that the converse is, in general, not possible. The reason for this is that μ-calculus formulas of alternation depth 2 were sufficient to capture $\mathcal{L}_\omega$, and hence, there are μ-calculus formulas of higher alternation depth that can not be described with $\mathcal{L}_\omega$. Nevertheless, specifications in $\mathcal{L}_\omega$ are, in general, more readable than μ-calculus formulas and are therefore better suited for the specification of reactive systems. Their expressiveness is still sufficient for most applications, and their ability to be used as graphical specification formalisms is advantageous for obtaining readable specifications.

However, if the size of automata grows (even less than ten states are sometimes enough) then it becomes, in many cases, difficult to figure out what is expressed by an ω-automaton. For this reason, more readable specification logics like temporal logics have been developed. The advantage of nesting temporal formulas into each other provides a natural way to write down structured specifications, whereas ω-automata are mostly based on flat, i.e., unstructured state transition diagrams (unless statechart-like approaches are used [236]).

Temporal logics are therefore convenient formalisms for specifying systems with a complex temporal behavior. They have been first proposed in 1977 for the specification of reactive systems by Pnueli [407, 408] (Kröger [294] used temporal logic in the same year for specifying sequential programs). Other authors that used temporal logics around the early eighties as specification language are Hailpern [228], Owicki [229], and Lamport [311].

There is, however, not a single temporal logic, since there are lot of ways a temporal logic can be defined. For example, some temporal logics consider the future as well as the past [274, 315, 326], while others only have

future time temporal operators. Some use temporal operators that include the present point of time, while others exclude it. Another distinction is the modeling of time: usually discrete points of time are considered, and it is assumed that every transition in the Kripke structure requires one unit of time. However, for special purposes, there are real-time temporal logics that are based on a continuous time model [9, 12, 30, 90, 91], and others that are based on a discrete time model where transitions may however require more than one unit of time [10, 11, 95, 238, 243, 316, 329, 330, 392, 427]. Furthermore, there are other variants such as first order temporal logics [1, 345], partial order temporal logics [406], and interval temporal logics [377, 450].

Throughout this chapter and this book, we will only consider propositional temporal logics with a discrete model of time. It is assumed that a transition in the Kripke structure requires one unit of time. Temporal logics of this kind are the most popular ones and have seen many success stories in the past [112, 114]. The mentioned criteria do not yet fix a particular temporal logic. There is still freedom for the set of temporal operators and the way these operators can be combined. In general, one distinguishes between *temporal* operators that express a temporal relationship along a computation path of a structure, and *path quantifiers* E and A that can be used to quantify over computation paths: We already know that $\mathsf{E}\varphi$ means that there is a path starting in the considered state that satisfies φ, and $\mathsf{A}\varphi$ means that for all paths starting in this state, the property φ holds. Examples for temporal operators are the ones that have been introduced in Section 2.2, i.e., X, G, F, U, W, B, $\underline{\mathsf{U}}$, $\underline{\mathsf{W}}$, and $\underline{\mathsf{B}}$, with the semantics as given in Section 2.2.

We briefly list some historical remarks and start with the genealogy of temporal operators. Kamp [274] used, in 1968, only one temporal operator and its corresponding past operator, namely the $[\cdot\ \underline{\mathsf{XU}}\ \cdot]$ operator that we can define as $[\varphi\ \underline{\mathsf{XU}}\ \psi] := \mathsf{X}[\varphi\ \underline{\mathsf{U}}\ \psi]$. Kamp proved that his logic is more expressive than previous ones that were only based on G and F. Moreover, he showed that his temporal logic is expressively equivalent to the first order theory of linear order (cf. Chapter 6). The same was proved in 1980 for the future time fragment by Gabbay, Pnueli, Shelah, and Stavi in [203], so that past time operators have been considered as unnecessary since then. Recently, past time operators have gained new attention, since it turned out that their usage makes temporal logic exponentially more succinct [316, 355].

Further temporal operators, such as the next-time X operator, the precede operator, and the weak U operator were introduced by Manna and Pnueli in 1979, 1982, and 1982, respectively in [343, 345, 346]. Despite the fact that different temporal operators may be used, the resulting logics will all be called LTL, if they can express Kamp's $[\cdot\ \underline{\mathsf{XU}}\ \cdot]$ operator. In 1985, it has been shown by Sistla and Clarke that the complexity of the LTL model checking problem is PSPACE-complete [143, 456]. Lichtenstein, Pnueli, and Zuck presented in the same year a model checking procedure that ran in time $O(|\mathcal{K}|\, 2^{c|\Phi|})$ (with some constant c) for every structure $\mathcal{K}$ and every LTL formula Φ [325].

In 1980, Emerson and Clarke [160], and in 1981, Ben-Ari, Manna, and Pnueli [36] explicitly used for the first time the path operators E and A. In [36] this was used to define a simple branching time temporal logic called UB (Unified system of Branching time) which is presented in detail in this chapter. In UB, path quantifiers and temporal operators occur in pairs. UB only used the temporal operators X, G, and F and provided hence the 'macro' operators EX, EG, EF, AX, AG, and AF. In the same year, Emerson and Clarke [160, 161] pointed out that UB can be interpreted as a fragment of the alternation-free μ-calculus. Moreover, they added the U operator and thus defined the temporal logic CTL which, although being strictly more expressive than UB, may still be interpreted as a fragment of the alternation-free μ-calculus. The expressiveness and complexity of CTL was investigated by Emerson and Halpern [162] and Sistla and Clarke [456] in 1985. In particular, it turned out that every CTL formula Φ can be checked on a structure $\mathcal{K}$ in time $O(|\mathcal{K}|\,|\Phi|)$. One year later, Emerson and Halpern defined the more powerful temporal logic CTL* [163] and proved in [165] that its model checking problem is PSPACE-complete.

After the introduction of linear time and branching time temporal logics, there has been an extensive debate [163, 172, 176, 310, 324, 497] on whether branching time logics like CTL and CTL* or linear time temporal logics like LTL are more suited to the specification and verification of reactive systems. In general, a linear time temporal logic consists of formulas of the form AΦ, where Φ does not contain path quantifiers at all, while branching time temporal logics allow nested path quantifiers. Depending on the choice of temporal operators and the way they are allowed to be nested, different linear time and branching-time temporal logics can be defined. We will consider some of them in the next section in more detail.

In [163], it was argued in favor of the linear time approach that for the verification of already existing systems, it is normally not necessary to reason about the existence of computation paths (this is something that can be specified in branching time temporal logics, but not in linear time temporal logics). [163] further argued that the usual specifications only require a property to hold for *all* computation paths, and hence linear time temporal logics are sufficient for that purpose. In comparison to CTL, linear time specifications tend to be more readable, since complex temporal properties require arbitrary nested temporal operators. Nested temporal operators are also allowed in CTL, but the syntax of CTL requires us to insert a path quantifier E or A in front of every temporal operator. This restricts the expressiveness and the readability of the formulas [434, 497].

After this, 'branching-time stroked back' [176]: Clearly, every linear time temporal logic can be extended to a corresponding branching time temporal logic by simply allowing path quantifiers to occur everywhere. Emerson and Lei showed in 1987 [176] that every model checking procedure for a linear time temporal logic can be extended to a model checking procedure for the corresponding branching time logic without changing the complexity of

the procedure. It is a powerful argument to prefer branching time temporal logics, since these contain the linear time logics as fragments, but do not increase the complexity of the verification procedure. Since then, the debate has continued, and beneath readability, expressiveness, and complexity other features like modular verification have also been considered. The interested reader is referred to Vardi's recent survey [497] for more pros and cons in this debate and also to Emerson's survey [158] that is still a rich source of interesting results.

Independent on the usage of a linear time or a branching time logic, a lack of expressiveness was seen by Wolper [519] in 1983. He pointed out that there are some simple properties that can not be expressed by temporal logics. For example, you can not express that a property holds *at least* for all even points of time. These properties can however be expressed in the μ-calculus and also by ω-automata. In fact, the result was known a long time before Wolper's remark: McNaughton and Papert had already shown in 1971 that the monadic first order theory of linear orders $\mathsf{MFO}_<$ (see Chapter 6) can be translated to noncounting automata (see Section 5.5.1) [364], and due to Kamp's result [274] (see Section 6.4), the same holds for temporal logic formulas. Wolper therefore suggested to extend temporal logics by grammar operators which essentially yields the flattened fragment of the logic $\mathcal{L}_\omega$ of the previous chapter. However, while it does no harm to extend temporal logics with automaton formulas to increase their expressiveness, the use of temporal operators is still recommended to increase the readability of specifications. Such hybrid logics are nowadays discussed for industrial usage like ForSpec [21] logic or Sugar logic [2, 20, 33].

Hence, with this and the previous two chapters, we now have three different formalisms: μ-calculus, ω-automata, and temporal logic (CTL^*). The expressiveness of these formalisms is strictly decreasing in this order, but the readability of the formalisms is strictly increasing. The expressiveness of CTL^* is sufficient for most applications. The additional expressiveness provided by ω-automata and the μ-calculus is rarely necessary. Hence, temporal logics like CTL^* are usually sufficient, and due to their better readability, their use is strongly recommended.

As the model checking problem of CTL^* is PSPACE-complete, a lot of researchers considered sublogics of CTL^* to find a good compromise between complexity and expressiveness. An extreme example is the computation tree logic CTL that is not very expressive, but very efficient (it is P-complete). In the next section, we will therefore discuss some sublanguages of CTL^* and compare them w.r.t. to their expressiveness and complexity. These sublanguages of CTL^* either restrict the set of available temporal operators or the possible combinations of these operators. To see which combinations can be removed without loosing expressive power, we will discuss two important transformations of CTL^* formulas: the 'elimination of quantified Boolean subformulas' (Section 5.3.3) and the 'duplication of path quantifiers' (Section 5.3.4). The first transformation allows us to eliminate applications of path

quantifiers to Boolean operators (for example $\mathsf{E}[\varphi \wedge \psi]$). The other transformation inserts path quantifiers at certain subformulas without changing the truth value (for example the LTL formula $\mathsf{AGF}\varphi$ is equivalent to the CTL formula $\mathsf{AGAF}\varphi$; however, note that even $\mathsf{EXEG}\varphi$ is equivalent to $\mathsf{EXG}\varphi$ and $\mathsf{EGX}\varphi$, but not equivalent to $\mathsf{EGEX}\varphi$).

Beyond proving the expressiveness of sublanguages of CTL*, the two transformations have a direct impact on the construction of verification procedures. Using these transformations, a direct translation to the temporal logic CTL, and hence to the alternation-free μ-calculus is possible for some CTL* formulas. Hence, these transformations are of practical importance, since they allow us to formulate specifications in syntactically richer logics, that are expressively equivalent to syntactically weaker logics. In particular, we will define the logic LeftCTL* and prove that LeftCTL* can be reduced by the mentioned transformations to CTL. As an example, the formula $\mathsf{E}[\mathsf{F}a \wedge \mathsf{FG}b]$ is a LeftCTL* formula, but not a CTL formula. The formula states that there is a computation in the system where a holds at least once, and from a certain point of time b must hold. By the transformations given in Section 5.3.3 and Section 5.3.4, we can translate this formula to the equivalent CTL formula given below:

$$\mathsf{EF}[a \wedge \mathsf{EFEG}b] \vee \mathsf{EFE}[b \,\underline{\mathsf{U}}\, (a \wedge \mathsf{EG}b)]$$

The above example clearly shows that LeftCTL* specifications can be more readable and more succinct than equivalent CTL specifications. In fact, by a recent result of Wilke [513], it follows that there are LeftCTL* specifications that require an exponential blow-up in the translation to CTL (a similar blow-up appears in the elimination of past temporal operators [316, 355]; see page 392). Adler and Immerman [4] improved this result in 2001 to a lower bound $|\varphi|!$. In fact, the translation of LeftCTL* to exponentially sized 'CTL' formulas given in Section 5.3.5 translates the formulas to CTL formulas where common subterms are shared. For this reason only a blow-up $O(|\Phi|\, 2^{2|\Phi|})$ is sufficient. Recently, Johannsen and Lange [266] proved that satisfiability checking of CTL$^+$ is 2EXPTIME-complete. As satisfiability checking of CTL is only EXPTIME-complete (as well as the sat-problem of the μ-calculus), it follows that every translation from CTL$^+$ to CTL (or to the μ-calculus) must necessarily have an exponential blow-up.

However, both logics, LeftCTL* and CTL, are expressively equivalent. The transformations in Section 5.3.3 and Section 5.3.4 are therefore important front-ends for CTL model checking tools so that specifications can be given in richer logics like LeftCTL*, while the model checking machinery can still be based on CTL.

In Section 5.4, we follow another approach: *instead of translating temporal logics to equivalent μ-calculus formulas, we translate them to equivalent ω-automata*, i.e., to formulas of $\mathcal{L}_\omega$. Of course, the obtained automata can in turn be translated to μ-calculus, but we need not necessarily do so since we

can also check the acceptance conditions in the product structures (Section 4.8.2).

In general, there are two different kinds of translations to ω-automata: procedures that construct the automata explicitly like [138, 185, 207, 208, 324, 386, 468, 501, 520], and others that derive a symbolic description of the automata like [89, 105, 131, 281, 437, 439]. The latter have the advantage that they run with linear runtime and memory requirements w.r.t. the length of the formulas, and that their result can be directly used for symbolic model checking. However, the different classes of translation are related as we will point out in Section 5.4.

While LeftCTL* and CTL can be translated to alternation-free μ-calculus formulas, the translation of CTL* to $\mathcal{L}_\omega$ requires to use the most powerful automaton class; for example, our translations yield $\mathsf{NDet}_{\mathsf{Streett}}$ automata. The translation of these automata to the μ-calculus requires alternation depth 2, which follows from results in [40, 381, 419]. The same holds for the reduction of the model checking problem with the help of the product structure (Section 4.8.2).

It is therefore natural to ask for (quantifier-free) temporal logics TL_κ such that every formula of TL_κ can be translated to an equivalent Det_κ automaton for $\kappa \in \{\mathsf{G}, \mathsf{F}, \mathsf{Prefix}, \mathsf{GF}, \mathsf{FG}, \mathsf{Streett}\}$. Hence, these logics completely correspond with the automata hierarchy that we have discovered in the previous chapter. Based on previous work of Manna and Pnueli [350–352], we will define temporal logics TL_κ, and show their translateability to Det_κ [437]. Moreover, we show their completeness, and their equal expressiveness to the classes of safety, guarantee, obligation, recurrence, persistence, and reactivity properties [350–352]. We will thereby also find a sublogic AFCTL* of CTL* that is strictly stronger than CTL and LeftCTL*, but that can still be translated to the alternation-free μ-calculus.

5.2 Branching Time Logics – Sublanguages of CTL*

In this section, we start with the definition of the most prominent logics CTL, LTL and CTL*, that are still the topic of many research projects. As already outlined, CTL and LTL were independently developed, and after a long debate, have been merged to the logic CTL* to overcome the disadvantages of CTL and LTL.

In particular, we explain in this section the different expressiveness of CTL and LTL and of the underlying temporal operators. CTL is still supported and favored by a lot of verification tools because it has the nice property that it can be checked in linear time. However, the logic has some limitations that makes its practical use inconvenient. Having seen the limitations of CTL that are not only due to the lack of expressiveness, but also due to a very limited syntax, we will consider, in Section 5.2.2, further temporal logics like

LeftCTL* that are equally expressive as CTL, but whose model checking problem is more complex. At first glance, this may sound as a disadvantage, but it is indeed an advantage: The higher complexity enables logics like LeftCTL* to be exponentially more succinct than CTL [4, 266, 513]. Moreover, we will see that every CTL formula is also a LeftCTL* formula, so that relying to the CTL subset leads to the same efficiency.

5.2.1 CTL, LTL and CTL*

We now start with the definition of CTL, LTL and CTL*. Although their origins are from a historical point of view very different, we will present CTL and LTL as subsets of CTL* (historically, CTL* has been obtained by combining CTL and LTL). The definition of these logics, and also of the forerunner UB of CTL, are given formally in the next definition.

Definition 5.1 (Sublanguages of CTL* (I)). *Given a finite set of variables V_Σ, the following grammar rules define the temporal logics* CTL*, UB, CTL, *and* LTL*:*

$$
\begin{array}{lll}
\mathsf{CTL}^* : & S ::= & V_\Sigma \mid \neg S \mid S \wedge S \mid S \vee S \mid \mathsf{E}P \mid \mathsf{A}P \\
& P ::= & S \mid \neg P \mid P \wedge P \mid P \vee P \mid \mathsf{X}P \mid \mathsf{G}P \mid \mathsf{F}P \\
& & \mid [P\ \mathsf{W}\ P] \mid [P\ \mathsf{U}\ P] \mid [P\ \mathsf{B}\ P] \\
& & \mid [P\ \underline{\mathsf{W}}\ P] \mid [P\ \underline{\mathsf{U}}\ P] \mid [P\ \underline{\mathsf{B}}\ P] \\
\mathsf{LTL} : & S ::= & \mathsf{A}P \\
& P ::= & V_\Sigma \mid \neg P \mid P \wedge P \mid P \vee P \mid \mathsf{X}P \mid \mathsf{G}P \mid \mathsf{F}P \\
& & \mid [P\ \mathsf{W}\ P] \mid [P\ \mathsf{U}\ P] \mid [P\ \mathsf{B}\ P] \\
& & \mid [P\ \underline{\mathsf{W}}\ P] \mid [P\ \underline{\mathsf{U}}\ P] \mid [P\ \underline{\mathsf{B}}\ P] \\
\mathsf{CTL} : & S ::= & V_\Sigma \mid \neg S \mid S \wedge S \mid S \vee S \mid \mathsf{E}P \mid \mathsf{A}P \\
& P ::= & \mathsf{X}S \mid \mathsf{G}S \mid \mathsf{F}S \\
& & \mid [S\ \mathsf{W}\ S] \mid [S\ \mathsf{U}\ S] \mid [S\ \mathsf{B}\ S] \\
& & \mid [S\ \underline{\mathsf{W}}\ S] \mid [S\ \underline{\mathsf{U}}\ S] \mid [S\ \underline{\mathsf{B}}\ S] \\
\mathsf{UB} : & S ::= & V_\Sigma \mid \neg S \mid S \wedge S \mid S \vee S \mid \mathsf{E}P \mid \mathsf{A}P \\
& P ::= & \mathsf{X}S \mid \mathsf{G}S \mid \mathsf{F}S
\end{array}
$$

In the above grammars, the formulas that are generated from the nonterminals S and P are state and path formulas, respectively. As can be seen, CTL* has no restrictions on its syntax: Temporal operators can be nested in an arbitrary manner to formulate temporal properties of paths. State formulas can be obtained by Boolean combination of other state formulas or by quantification of path formulas. LTL provides a powerful set of path formulas, however, state formulas of LTL are restricted to the very limited form $\mathsf{A}P$, where P must not contain further path quantifiers. CTL has other restrictions: temporal operators and path quantifiers have to occur in pairs, i.e., every path quantifier must be followed by a temporal operator, and conversely, every temporal operator must be preceded by a path quantifier. Finally, UB is the sublanguage of CTL that has only the temporal operators X, G, and F.

The choice among the binary temporal operators is thereby rather arbitrary. To reach the expressive power of $\mathsf{MFO}_<$, the monadic first order theory of linear orders, we only need the X operator and one of the binary operators. For example, the $\underline{\mathsf{U}}$ operator can define the other operators as follows:

$$\begin{array}{ll} \mathsf{G}\varphi = \neg\,[1\ \underline{\mathsf{U}}\ (\neg\varphi)] & \mathsf{F}\varphi = [1\ \underline{\mathsf{U}}\ \varphi] \\ [\varphi\ \mathsf{W}\ \psi] = \neg\,[(\neg\varphi \vee \neg\psi)\ \underline{\mathsf{U}}\ (\neg\varphi \wedge \psi)] & [\varphi\ \underline{\mathsf{W}}\ \psi] = [(\neg\psi)\ \underline{\mathsf{U}}\ (\varphi \wedge \psi)] \\ [\varphi\ \mathsf{B}\ \psi] = \neg\,[(\neg\varphi)\ \underline{\mathsf{U}}\ \psi] & [\varphi\ \underline{\mathsf{B}}\ \psi] = [(\neg\psi)\ \underline{\mathsf{U}}\ (\varphi \wedge \neg\psi)] \\ [\varphi\ \mathsf{U}\ \psi] = \neg\,[(\neg\psi)\ \underline{\mathsf{U}}\ (\neg\varphi \wedge \neg\psi)] & \end{array}$$

Analogously, each one of the binary temporal operators can define $\underline{\mathsf{U}}$:

$$\begin{array}{ll} [\varphi\ \underline{\mathsf{U}}\ \psi] = \neg\,[(\neg\psi)\ \mathsf{U}\ (\neg\varphi \wedge \neg\psi)] & \\ [\varphi\ \underline{\mathsf{U}}\ \psi] = \neg\,[(\neg\psi)\ \mathsf{W}\ (\varphi \rightarrow \psi)] & [\varphi\ \underline{\mathsf{U}}\ \psi] = [\psi\ \underline{\mathsf{W}}\ (\varphi \rightarrow \psi)] \\ [\varphi\ \underline{\mathsf{U}}\ \psi] = \neg\,[(\neg\varphi)\ \mathsf{B}\ \psi] & [\varphi\ \underline{\mathsf{U}}\ \psi] = [\psi\ \underline{\mathsf{B}}\ (\neg\varphi \wedge \neg\psi)] \end{array}$$

It is worthwhile noting that the until-operator that excludes the present, i.e., Kamp's version which is defined as $[\varphi\ \underline{\mathsf{XU}}\ \psi] := \mathsf{X}\,[\varphi\ \underline{\mathsf{U}}\ \psi]$, can express all other operators:

$$[\varphi\ \underline{\mathsf{U}}\ \psi] = \psi \vee \varphi \wedge [\varphi\ \underline{\mathsf{XU}}\ \psi] \qquad \mathsf{X}\varphi = [0\ \underline{\mathsf{XU}}\ \varphi]$$

Hence, we could define temporal logics with the only temporal operator $[\cdot\ \underline{\mathsf{XU}}\ \cdot]$. We have also the choice among the temporal operators for defining CTL. The classical definition is normally based on $\mathsf{E}\underline{\mathsf{U}}$, EG, and EX, which can be obtained due to the following rules:

$$\begin{array}{l} \mathsf{EF}\varphi = \mathsf{E}[1\ \underline{\mathsf{U}}\ \varphi] \\ \mathsf{E}[\varphi\ \mathsf{U}\ \psi] = \mathsf{E}[\varphi\ \underline{\mathsf{U}}\ \psi] \vee \mathsf{EG}\varphi \\ \mathsf{E}[\varphi\ \underline{\mathsf{B}}\ \psi] = \mathsf{E}[(\neg\psi)\ \underline{\mathsf{U}}\ (\varphi \wedge \neg\psi)] \\ \mathsf{E}[\varphi\ \underline{\mathsf{W}}\ \psi] = \mathsf{E}[(\neg\psi)\ \underline{\mathsf{U}}\ (\varphi \wedge \psi)] \\ \mathsf{E}[\varphi\ \mathsf{B}\ \psi] = \mathsf{E}[(\neg\psi)\ \underline{\mathsf{U}}\ (\varphi \wedge \neg\psi)] \vee \mathsf{EG}\neg\psi \\ \mathsf{E}[\varphi\ \mathsf{W}\ \psi] = \mathsf{E}[(\neg\psi)\ \underline{\mathsf{U}}\ (\varphi \wedge \psi)] \vee \mathsf{EG}\neg\psi \\ \mathsf{AX}\varphi = \neg\mathsf{EX}\neg\varphi \\ \mathsf{AG}\varphi = \neg\mathsf{E}[1\ \underline{\mathsf{U}}\ \neg\varphi] \\ \mathsf{AF}\varphi = \neg\mathsf{EG}\neg\varphi \\ \mathsf{A}[\varphi\ \mathsf{U}\ \psi] = \neg\mathsf{E}[(\neg\psi)\ \underline{\mathsf{U}}\ (\neg\varphi \wedge \neg\psi)] \\ \mathsf{A}[\varphi\ \mathsf{B}\ \psi] = \neg\mathsf{E}[(\neg\varphi)\ \underline{\mathsf{U}}\ \psi] \\ \mathsf{A}[\varphi\ \mathsf{W}\ \psi] = \neg\mathsf{E}[(\neg\psi)\ \underline{\mathsf{U}}\ (\neg\varphi \wedge \psi)] \\ \mathsf{A}[\varphi\ \underline{\mathsf{U}}\ \psi] = \neg\mathsf{E}[(\neg\psi)\ \underline{\mathsf{U}}\ (\neg\varphi \wedge \neg\psi)] \wedge \neg\mathsf{EG}\neg\psi \\ \mathsf{A}[\varphi\ \underline{\mathsf{B}}\ \psi] = \neg\mathsf{E}[(\neg\varphi)\ \underline{\mathsf{U}}\ \psi] \wedge \neg\mathsf{EG}\neg\varphi \\ \mathsf{A}[\varphi\ \underline{\mathsf{W}}\ \psi] = \neg\mathsf{E}[(\neg\psi)\ \underline{\mathsf{U}}\ (\neg\varphi \wedge \psi)] \wedge \neg\mathsf{EG}\neg\psi \end{array}$$

The above reduction to the basic operator sets $\mathsf{E}\underline{\mathsf{U}}$, EG, and EX has the disadvantage that in some cases two fixpoints are generated. Therefore, we recommend the following reductions to the basic operators $\mathsf{E}\underline{\mathsf{U}}$, EU, and EX:

$$\mathsf{AX}\varphi = \neg\mathsf{EX}\neg\varphi$$
$$\mathsf{EG}\varphi = \mathsf{E}[\varphi\ \mathsf{U}\ 0] \qquad \mathsf{AG}\varphi = \neg\mathsf{E}[1\ \underline{\mathsf{U}}\ \neg\varphi]$$
$$\mathsf{EF}\varphi = \mathsf{E}[1\ \underline{\mathsf{U}}\ \varphi] \qquad \mathsf{AF}\varphi = \neg\mathsf{E}[(\neg\varphi)\ \mathsf{U}\ 0]$$
$$\mathsf{A}[\varphi\ \mathsf{U}\ \psi] = \neg\mathsf{E}[(\neg\psi)\ \underline{\mathsf{U}}\ (\neg\varphi \wedge \neg\psi)]$$
$$\mathsf{A}[\varphi\ \underline{\mathsf{U}}\ \psi] = \neg\mathsf{E}[(\neg\psi)\ \mathsf{U}\ (\neg\varphi \wedge \neg\psi)]$$
$$\mathsf{E}[\varphi\ \mathsf{B}\ \psi] = \mathsf{E}[(\neg\psi)\ \mathsf{U}\ (\varphi \wedge \neg\psi)] \qquad \mathsf{A}[\varphi\ \mathsf{B}\ \psi] = \neg\mathsf{E}[(\neg\varphi)\ \underline{\mathsf{U}}\ \psi]$$
$$\mathsf{E}[\varphi\ \underline{\mathsf{B}}\ \psi] = \mathsf{E}[(\neg\psi)\ \underline{\mathsf{U}}\ (\varphi \wedge \neg\psi)] \qquad \mathsf{A}[\varphi\ \underline{\mathsf{B}}\ \psi] = \neg\mathsf{E}[(\neg\varphi)\ \mathsf{U}\ \psi]$$
$$\mathsf{E}[\varphi\ \mathsf{W}\ \psi] = \mathsf{E}[(\neg\psi)\ \mathsf{U}\ (\varphi \wedge \psi)] \qquad \mathsf{A}[\varphi\ \mathsf{W}\ \psi] = \neg\mathsf{E}[(\neg\psi)\ \underline{\mathsf{U}}\ (\neg\varphi \wedge \psi)]$$
$$\mathsf{E}[\varphi\ \underline{\mathsf{W}}\ \psi] = \mathsf{E}[(\neg\psi)\ \underline{\mathsf{U}}\ (\varphi \wedge \psi)] \qquad \mathsf{A}[\varphi\ \underline{\mathsf{W}}\ \psi] = \neg\mathsf{E}[(\neg\psi)\ \mathsf{U}\ (\neg\varphi \wedge \psi)]$$

Hence, we see that the choice among the binary operators is absolutely a matter of taste, both for LTL and CTL. One might ask at this point whether we can also express one of the binary operators also with G and F. Kamp has shown in his thesis that this is not the case [274]. We will also present a short proof for this fact below (see also page 450).

Consider a state s of an arbitrary structure $\mathcal{K} = (\mathcal{S}, \mathcal{I}, \mathcal{R}, \mathcal{L})$, and an arbitrary path $\pi \in \mathsf{Paths}_{\mathcal{K}}(s)$ through $\mathcal{K}$ that starts in s. Consider the corresponding sequence of labels, i.e., the infinite sequence $\lambda t.\mathcal{L}(\pi^{(t)})$. As V_Σ is finite, there is only a finite number of subsets of V_Σ, and hence only finitely many different possibilities to label a state in the structure. For this reason, it must be the case that at least one $\vartheta \subseteq V_\Sigma$ occurs infinitely often on $\lambda t.\mathcal{L}(\pi^{(t)})$, while others may occur only finitely often. Hence, among the labels $L_\pi := \{\mathcal{L}(\pi^{(t)}) \mid t \in \mathbb{N}\}$ there are some that occur infinitely often, and others that occur only finitely often. Let L_π^{fin} be the subset of L_π that consists of those $\vartheta \subseteq V_\Sigma$ that occur only finitely often on $\lambda t.\mathcal{L}(\pi^{(t)})$, and let L_π^{inf} be the remaining labels.

It is then clear that there is a number $j \in \mathbb{N}$ such that $\forall t \geq j.\, \mathcal{L}(\pi^{(t)}) \in L_\pi^{\mathsf{inf}}$ holds. The same holds for every $j' > j$, but not necessarily for all $j' < j$. So, let $j_\pi \in \mathbb{N}$ be the least number such that $\forall t \geq j_\pi.\, \mathcal{L}(\pi^{(t)}) \in L_\pi^{\mathsf{inf}}$ holds. We will see in the following lemma that the formulas that are build up with only the temporal operators G and F can not distinguish the order of states after the position j_π:

Lemma 5.2 (Temporal Logic without Binary Temporal Operators). *Given a structure $\mathcal{K} = (\mathcal{S}, \mathcal{I}, \mathcal{R}, \mathcal{L})$ over a set of variables V_Σ, a state $s \in \mathcal{S}$, and a path $\pi \in \mathsf{Paths}_{\mathcal{K}}(s)$. Let $j_\pi \in \mathbb{N}$ be the smallest number such that $\{\mathcal{L}(\pi^{(t)}) \mid t \geq j_\pi\} = \{\vartheta \subseteq V_\Sigma \mid \forall t_1.\exists t_2.\mathcal{L}(\pi^{(t_1+t_2)}) = \vartheta\}$ holds. Moreover, let $\mathsf{A}\varphi$ be a LTL formula that is built up only with Boolean operators and the temporal operators G and F. Then, the following holds for all positions $t_1, t_2 \geq j_\pi$ with $\mathcal{L}(\pi^{(t_1)}) = \mathcal{L}(\pi^{(t_2)})$:*

$$(\mathcal{K}, \pi, t_1) \models \varphi \quad \textit{iff} \quad (\mathcal{K}, \pi, t_2) \models \varphi$$

Proof. The proof is done by induction on the structure of φ. We only list the relevant cases (every formula can be reduced to an equivalent one using only these operators by rewriting with $\mathsf{G}\varphi = \neg\mathsf{F}\neg\varphi$ and $\varphi \vee \psi = \neg(\neg\varphi \wedge \neg\psi)$).

$\boxed{\varphi \in V_\Sigma}$: By the semantics of variables, we have $(\mathcal{K}, \pi, t_1) \models \varphi$ iff $\varphi \in \mathcal{L}(\pi^{(t_1)})$. By the assumption of the lemma, we have $\mathcal{L}(\pi^{(t_1)}) = \mathcal{L}(\pi^{(t_2)})$, so that the proposition holds for variables.

$\boxed{\neg\varphi}$: This case follows directly from the induction hypothesis.

$\boxed{\varphi \wedge \psi}$: This case follows directly from the induction hypothesis.

$\boxed{\mathsf{F}\varphi}$: Given $t_1, t_2 \geq j_\pi$ with $\mathcal{L}(\pi^{(t_1)}) = \mathcal{L}(\pi^{(t_2)})$, we have to prove that $(\mathcal{K}, \pi, t_1) \models \mathsf{F}\varphi$ holds iff $(\mathcal{K}, \pi, t_2) \models \mathsf{F}\varphi$ holds. We only show that $(\mathcal{K}, \pi, t_1) \models \mathsf{F}\varphi$ implies $(\mathcal{K}, \pi, t_2) \models \mathsf{F}\varphi$, since the other direction is symmetric. So, assuming that $(\mathcal{K}, \pi, t_1) \models \mathsf{F}\varphi$ holds, this means that there must be a $\delta_1 \in \mathbb{N}$ such that $(\mathcal{K}, \pi, t_1 + \delta_1) \models \varphi$ holds. We must now show that there is a $\delta_2 \in \mathbb{N}$, such that $(\mathcal{K}, \pi, t_2 + \delta_2) \models \varphi$ holds.
If $t_2 \leq t_1 + \delta_1$ holds, we can use $\delta_2 := (t_1 - t_2 + \delta_1)$ to satisfy the condition. In the remaining case, we have $t_2 < t_1 + \delta$. As $t_1 > j_\pi$ holds, it also follows that $t_1 + \delta_1 > j_\pi$ holds, which means that there are infinitely many states on the path with the same label $\mathcal{L}(\pi^{(t_1+\delta_1)})$. Hence, there are infinitely many k_i where $\mathcal{L}(\pi^{(t_1+\delta_1)}) = \mathcal{L}(\pi^{(t_1+k_i)})$ holds. By the induction hypothesis, it therefore follows that $(\mathcal{K}, \pi, t_1 + k_i) \models \varphi$ holds at these positions. Moreover, there is a $n \in \mathbb{N}$ such that $t_1 + k_n > t_2$ holds. Therefore, we can use $\delta_2 := t_1 + k_n - t_2$ to see that there is a δ_2 with $(\mathcal{K}, \pi, t_2 + \delta_2) \models \varphi$. □

The above lemma essentially says that the quantifier-free kernels φ of LTL formulas $\mathsf{A}\varphi$, that are build up only with Boolean operators and the temporal operators G and F, can not distinguish states $\pi^{(t_1)}$ and $\pi^{(t_2)}$ that have the same labels and are after a certain position on the path π. In particular, this means that these formulas can not distinguish between the order of such states on the path.

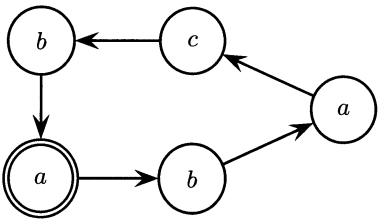


Fig. 5.1. G and F can not express U

The lemma does not hold for the binary temporal operators since these can distinguish the order of events. As an example, consider the structure given in Figure 5.1. There is only one path π leaving the initial state and all states on this path repeat infinitely often, hence, we have $j_\pi = 0$. Now, note that we have $(\mathcal{K}, \pi, 2) \not\models [a \,\mathsf{U}\, b]$, but $(\mathcal{K}, \pi) \models [a \,\mathsf{U}\, b]$. Moreover, we have $(\mathcal{K}, \pi, 2) \not\models \mathsf{X}b$, but $(\mathcal{K}, \pi) \models \mathsf{X}b$, so that we also see that the lemma can not be extended to X operators.

However, we can shift X operators inside so that they are finally applied to variables (or to other X operators). Abbreviating the formulas $\mathsf{X}^k x$ by a new variable x_k then allows us to derive the following result:

Theorem 5.3. *None of the binary operators* U, B, W, $\underline{\mathsf{U}}$, $\underline{\mathsf{B}}$, $\underline{\mathsf{W}}$ *can be expressed by the unary temporal operators* X, G *and* F.

It therefore follows immediately that UB is strictly less expressive than CTL, since the formula $\mathsf{E}[b\ \underline{\mathsf{U}}\ a]$ can not be expressed in UB. Conversely, it is clear that every UB formula is also a CTL formula.

To compare CTL and LTL, we note that there are CTL formulas that can not be expressed in LTL, and there are LTL formulas that can not be expressed in CTL. For example, no CTL formula is equivalent to $\mathsf{EGF}\varphi$, and no LTL formula is equivalent to $\mathsf{AXEX}\varphi$. It is furthermore well-known which formulas can be expressed in these logics. We briefly list some essential results for the consideration of these facts, and start with an inherent property of LTL formulas:

Lemma 5.4 (Invariance under Language Equivalence). *Given two structures* $\mathcal{K}_1 = (\mathcal{I}_1, \mathcal{S}_1, \mathcal{R}_1, \mathcal{L}_1)$ *and* $\mathcal{K}_2 = (\mathcal{I}_2, \mathcal{S}_2, \mathcal{R}_2, \mathcal{L}_2)$ *over* V_Σ*, two states* $s_1 \in \mathcal{S}_1$, $s_2 \in \mathcal{S}_2$*, and two paths* $\pi_1 \in \mathsf{Paths}_{\mathcal{K}_1}(s_1)$ *and* $\pi_2 \in \mathsf{Paths}_{\mathcal{K}_2}(s_2)$ *with* $\forall t. \mathcal{L}_1(\pi_1^{(t)}) = \mathcal{L}_2(\pi_2^{(t)})$*. For every* LTL *formula* $\mathsf{A}\varphi$*, the following holds for every* $t \in \mathbb{N}$*:*

$$(\mathcal{K}_1, \pi_1, t) \models \varphi \text{ iff } (\mathcal{K}_2, \pi_2, t) \models \varphi$$

The proof is by induction on φ, where it is sufficient to consider the cases $\neg\varphi$, $\varphi \wedge \psi$, $\mathsf{X}\varphi$, $[\varphi\ \underline{\mathsf{U}}\ \psi]$. We omit the proof, since it is very simple. The lemma says that for evaluating quantifier-free formulas only the sequence of labels is of interest. In other words, these formulas can not be used to describe the branching behavior of a structure, which is intuitively clear. Note that the same holds for automaton formulas $\mathcal{A}_\exists\,(Q, \Phi_\mathcal{I}, \Phi_\mathcal{R}, \Phi_\mathcal{F})$.

We will see in Section 5.4 that every LTL formula φ can be translated to an equivalent ω-automaton $\mathfrak{A}_\varphi \in \mathsf{NDet}_{\mathsf{GF}}$. Hence, we have $(\mathcal{K}, \pi, t) \models \varphi$ iff $(\mathcal{K}, \pi, t) \models \mathfrak{A}_\varphi$, and this in turn holds if the ω-word $\lambda t. \mathcal{L}(\pi^{(t)})$ is accepted by the automaton. However, as Kripke structures and automata are finite, we immediately find the following lemma that is a variant of the fact that ω-regular languages are not empty, iff they contain a rational word.

Lemma 5.5 (Satisfiability by Rational Paths). *Given a Kripke structure* $\mathcal{K} = (\mathcal{I}, \mathcal{S}, \mathcal{R}, \mathcal{L})$ *over* V_Σ *and a states* $s \in \mathcal{S}$*. Then, the following holds for every* LTL *formula* $\mathsf{A}\varphi$*:*

- $(\mathcal{K}, s) \models \mathsf{A}\varphi$ *iff for all* $\pi \in \mathsf{RatFairPaths}_\mathcal{K}(s)$ *we have* $(\mathcal{K}, \pi, 0) \models \varphi$
- $(\mathcal{K}, s) \models \mathsf{E}\varphi$ *iff there is a* $\pi \in \mathsf{RatFairPaths}_\mathcal{K}(s)$ *with* $(\mathcal{K}, \pi, 0) \models \varphi$

The previous two lemmas can now be used to prove the following theorem due to Clarke and Draghicescu [101]. This theorem provides a criterion to test whether a given CTL* formula can be expressed in LTL. If it can be expressed in LTL, the theorem even constructs such a formula:

Theorem 5.6 (Criterion for Reducibility from CTL* to LTL). *For every* CTL* *formula* Φ*, the following are equivalent:*

- *there is a* LTL *formula* $\mathsf{A}\varphi$ *that is equivalent to* Φ
- Φ *is equivalent to* $\mathsf{A}(\mathsf{remove}_{\mathsf{E},\mathsf{A}}(\Phi))$*, where* $\mathsf{remove}_{\mathsf{E},\mathsf{A}}(\Phi)$ *is obtained from* Φ *by deleting all path quantifiers.*

Proof. One direction is simple: given that $\Phi = \mathsf{A}(\mathsf{remove}_{\mathsf{E},\mathsf{A}}(\Phi))$ holds, then there is a LTL formula that is equivalent to Φ, namely $\mathsf{A}(\mathsf{remove}_{\mathsf{E},\mathsf{A}}(\Phi))$. To prove the other direction, suppose there is a LTL formula $\mathsf{A}\varphi$ that is equivalent to Φ. This means that $(*)$ for any state s of any structure $\mathcal{K}$, we have $(\mathcal{K}, s) \models \Phi$ iff $(\mathcal{K}, s) \models \mathsf{A}\varphi$. We then also have to show that $(\mathcal{K}, s) \models \mathsf{A}(\mathsf{remove}_{\mathsf{E},\mathsf{A}}(\Phi))$ holds.

We construct, for a rational path π with $\forall t.\pi^{(t+n)} = \pi^{((t \bmod \ell)+n)}$, a structure $\mathcal{K}_\pi = (\mathcal{I}_\pi, \mathcal{S}_\pi, \mathcal{R}_\pi, \mathcal{L}_\pi)$ as follows:

- $\mathcal{S}_\pi := \{0, \ldots, \ell - 1 + n\} \subseteq \mathbb{N}$
- $\mathcal{I}_\pi := \{0\}$
- $\mathcal{R}_\pi := \{(i, i+1) \mid i \in \{0, \ldots, \ell - 2 + n\}\} \cup \{(\ell - 1 + n, n)\}$
- $\mathcal{L}_\pi(i) := \mathcal{L}(\pi^{(i)})$ for $i \in \{0, \ldots, \ell - 1 + n\}$

Clearly, the structure has only one path, that is furthermore language equivalent to the original path π. Hence, it is easily seen that we have $(**)$ $(\mathcal{K}_\pi, 0) \models \Phi$ if and only if $(\mathcal{K}_\pi, 0) \models \mathsf{A}(\mathsf{remove}_{\mathsf{E},\mathsf{A}}(\Phi))$, which can be proved by induction. Therefore, given any structure $\mathcal{K}$ and any state s of it, the following equivalences hold:

$$
\begin{aligned}
(\mathcal{K}, s) \models \Phi &\overset{(*)}{\Leftrightarrow} (\mathcal{K}, s) \models \mathsf{A}\varphi \\
&\overset{(1)}{\Leftrightarrow} \forall \pi \in \mathsf{RatFairPaths}_{\mathcal{K}}(s).\ (\mathcal{K}, \pi, 0) \models \varphi \\
&\overset{(2)}{\Leftrightarrow} \forall \pi \in \mathsf{RatFairPaths}_{\mathcal{K}}(s).\ (\mathcal{K}_\pi, \pi, 0) \models \varphi \\
&\overset{(3)}{\Leftrightarrow} \forall \pi \in \mathsf{RatFairPaths}_{\mathcal{K}}(s).\ (\mathcal{K}_\pi, 0) \models \mathsf{A}\varphi \\
&\overset{(*)}{\Leftrightarrow} \forall \pi \in \mathsf{RatFairPaths}_{\mathcal{K}}(s).\ (\mathcal{K}_\pi, 0) \models \Phi \\
&\overset{(**)}{\Leftrightarrow} \forall \pi \in \mathsf{RatFairPaths}_{\mathcal{K}}(s).\ (\mathcal{K}_\pi, 0) \models \mathsf{A}(\mathsf{remove}_{\mathsf{E},\mathsf{A}}(\Phi)) \\
&\overset{(4)}{\Leftrightarrow} \forall \pi \in \mathsf{RatFairPaths}_{\mathcal{K}}(s).\ (\mathcal{K}_\pi, \pi, 0) \models \mathsf{remove}_{\mathsf{E},\mathsf{A}}(\Phi)) \\
&\overset{(5)}{\Leftrightarrow} \forall \pi \in \mathsf{RatFairPaths}_{\mathcal{K}}(s).\ (\mathcal{K}, \pi, 0) \models \mathsf{remove}_{\mathsf{E},\mathsf{A}}(\Phi) \\
&\overset{(6)}{\Leftrightarrow} (\mathcal{K}, s) \models \mathsf{A}(\mathsf{remove}_{\mathsf{E},\mathsf{A}}(\Phi))
\end{aligned}
$$

Step $(*)$ is due to our assumption, (1) is due to Lemma 5.5, and (2) due to Lemma 5.4. (3) holds since $\mathcal{K}_\pi$ has only one path. The next step is again due to our assumption $(*)$, this time applied to the single path structures $\mathcal{K}_\pi$. The next step $(**)$ is by construction of $\mathcal{K}_\pi$ as mentioned above. (4) holds since $\mathcal{K}_\pi$ has only the path π. (5) is clear by construction of $\mathcal{K}_\pi$, and (6) holds finally due to Lemma 5.5. □

The theorem can be used in two ways: Firstly, if we know that a given CTL* formula has an equivalent LTL formula, then we can determine an equivalent LTL formula very easily. Secondly, to decide whether a given CTL* formula Φ has an equivalent LTL formula or not, we have to check whether Φ and $\mathsf{A}(\mathsf{remove}_{\mathsf{E},\mathsf{A}}(\Phi))$ are equivalent. In particular, to show that a CTL* formula Φ is not expressible in LTL, we have to present a Kripke structure that interprets Φ and $\mathsf{A}(\mathsf{remove}_{\mathsf{E},\mathsf{A}}(\Phi))$ differently. For example, the CTL formula $\mathsf{AFAG}a$ is not equivalent to any LTL formula. To prove this, simply check that the structure given in Figure 5.2 satisfies $\mathsf{AFAG}a$, but not $\mathsf{AFG}a$.

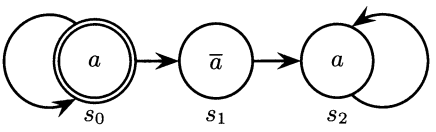


Fig. 5.2. Counterexample for $\mathsf{AFG}a = \mathsf{AFAG}a$

It is easily seen that $(\mathcal{K}, s_0) \models \mathsf{AFG}a$ holds, since every path starting in s_0 must either stay forever in s_0 or must leave s_0 via s_1 such that it must stay after that point of time forever in s_2. However, we have $(\mathcal{K}, s_0) \not\models \mathsf{AFAG}a$, since $[\![\mathsf{AG}a]\!]_{\mathcal{K}} = \{s_2\}$, and hence, $[\![\mathsf{AFAG}a]\!]_{\mathcal{K}} = \{s_1, s_2\}$. Hence, this structure shows that the LTL formula $\mathsf{AFG}a$ is not equivalent to the CTL formula $\mathsf{AFAG}a$, and the above theorem tells us additionally that no other LTL formula can therefore be equivalent to $\mathsf{AFAG}a$.

The structure given in Figure 5.2 shows also that $\mathsf{EGF}\varphi$ is not equivalent to $\mathsf{EGEF}\varphi$ and $\mathsf{EGX}\varphi$ is not equivalent to $\mathsf{EGEX}\varphi$. Note that $[\![\mathsf{EGF}\neg a]\!]_{\mathcal{K}} = \{\}$, $[\![\mathsf{EGEF}\neg a]\!]_{\mathcal{K}} = \{s_0\}$, $[\![\mathsf{EGX}\neg a]\!]_{\mathcal{K}} = \{\}$, and $[\![\mathsf{EGEX}\neg a]\!]_{\mathcal{K}} = \{s_0\}$. In a similar way, it is possible to show with another simple structure that $\mathsf{EXAX}\varphi$ can not be expessed in LTL.

Hence, we have obtained a criterion to check whether a CTL* formula, and hence, whether a CTL formula can be expressed in LTL. The opposite problem, namely to check if a LTL formula can be expressed in CTL turned out to be much harder and was unsolved for quite a long time. A breakthrough in that direction was obtained in 1998 by Kupferman and Vardi [303] who characterized the fragment of LTL that can be translated to the alternation-free μ-calculus:

Theorem 5.7 (LTL and Alternation-Free μ-Calculus). *A formula $\mathsf{A}\varphi \in \mathsf{LTL}$ is equivalent to an alternation-free μ-calculus formula iff φ is equivalent to some $\mathfrak{A} \in \mathsf{Det}_{\mathsf{GF}}$.*

We can not present the proof here since it requires some knowledge about tree automata. The above theorem characterizes the subset of LTL formulas that can be translated to the alternation-free μ-calculus, which is a superset of

CTL (as we will see in the next section). A tight characterization was finally obtained in 2000 by Maidl [338]. In particular, she defined the following logic:

Definition 5.8 (Semantic Intersection of CTL and LTL). *We define the logic* $\mathsf{LTL}_{\mathsf{det}}$ *as the least subset of* LTL *that satisfies the following rules:*

- $V_\Sigma \subseteq \mathsf{LTL}_{\mathsf{det}}$
- $\varphi \wedge \psi \in \mathsf{LTL}_{\mathsf{det}}$ *if* $\varphi, \psi \in \mathsf{LTL}_{\mathsf{det}}$
- $x \wedge \psi \vee \neg x \wedge \varphi \in \mathsf{LTL}_{\mathsf{det}}$ *if* $\varphi, \psi \in \mathsf{LTL}_{\mathsf{det}}$ *and* $x \in V_\Sigma$
- $\mathsf{X}\varphi \in \mathsf{LTL}_{\mathsf{det}}$ *if* $\varphi \in \mathsf{LTL}_{\mathsf{det}}$
- $[(x \wedge \varphi)\ \underline{\mathsf{U}}\ (\neg x \wedge \psi)] \in \mathsf{LTL}_{\mathsf{det}}$ *if* $\varphi, \psi \in \mathsf{LTL}_{\mathsf{det}}$ *and* $x \in V_\Sigma$
- $[(x \wedge \varphi)\ \mathsf{U}\ (\neg x \wedge \psi)] \in \mathsf{LTL}_{\mathsf{det}}$ *if* $\varphi, \psi \in \mathsf{LTL}_{\mathsf{det}}$ *and* $x \in V_\Sigma$

The idea is thereby to eliminate some 'overlaps': for example, to satisfy a formula $[\varphi\ \underline{\mathsf{U}}\ \psi]$ we must have a path where φ holds up to position t and where ψ holds at position $t+1$. It may be the case that φ also holds at $t+1$, which is an overlap that is eliminated in $\mathsf{LTL}_{\mathsf{det}}$. The same holds for disjunctions: the variable x selects one of the two subformulas to be satisfied. Maidl then proved the following result [338] and presented an algorithm to test whether for a $\mathsf{A}\varphi \in \mathsf{LTL}$, there is an equivalent $\mathsf{LTL}_{\mathsf{det}}$ formula.

Theorem 5.9 (Semantic Intersection of CTL and LTL). *A formula* $\mathsf{A}\varphi \in \mathsf{LTL}$ *is equivalent to a* CTL *formula iff* φ *is equivalent to a* $\mathsf{LTL}_{\mathsf{det}}$ *formula.*

5.2.2 Adding Syntactic Sugar to CTL

In the previous section, we have considered the temporal logics CTL, LTL, and CTL* and have compared their expressiveness. As CTL* includes both logics, it is clearly the best choice if we require maximal expressiveness. The model checking problem for CTL* is however PSPACE-complete [103, 172, 176], the same holds for the LTL model checking problem [456], while CTL model checking can be solved in polynomial time [103, 116, 443].

However, complexity and expressiveness are not the only issues. From a practical point of view, the readability of a specification logic is also a major criterion. Considering the readability, several authors claimed that CTL suffers from restrictions that limit its practical usage. These restrictions are not only due to a limited expressiveness, but also due to a unnecessarily hard restrictive syntax. For example, consider the following formula:

$$\mathcal{H}_n :\equiv \mathsf{E}\left[\left(\bigwedge_{i=0}^{n} \mathsf{F}\varphi_i\right) \wedge \bigwedge_{i=0}^{n} \mathsf{G}\left(\varphi_i \rightarrow \mathsf{XG}\neg\varphi_i\right)\right]$$

The formula $\mathcal{H}_n$ states that there must be a path that visits every set $[\![\varphi_i]\!]_{\mathcal{K}}$ exactly once. Hence, checking the formula on a structure (that is appropriately labeled) is equivalent to finding Hamilton paths in the structure which is known to be NP-complete [204].

We will define below the logic LeftCTL*, and will then see that $\mathcal{H}_n$ belongs to LeftCTL*. We already know that $\mathcal{H}_n$ is not a CTL formula, but in the next section we will develop techniques to translate every LeftCTL* formula to an equivalent CTL formula. Hence, it follows that there are formulas in CTL that are equivalent to $\mathcal{H}_n$, but these formulas are very large. In general, our transformations will generate, for a given LeftCTL* formula Φ, an equivalent CTL formula of size $O(|\Phi|!)$ and or $O(|\Phi|\, 2^{2|\Phi|})$ if common subterms are shared, so there may be an exponential blow-up. As LeftCTL* satisfiability checking is 2EXPTIME-complete [266], but CTL satisfiability checking is only EXPTIME-complete, it follows that the exponential blow-up can not be avoided. See also [4, 513] for lower bounds of this translation.

Hence, we see that not only the expressiveness itself is an issue, but also the succinctness and hence, the readability of a logic. This is clearly due to a higher complexity of the model checking problem: If a logic like LeftCTL* is exponentially more succinct than CTL, then it can no longer have a polynomial model checking procedure. In fact, the model checking problem of LeftCTL* is NP-hard (Section 5.6) and Δ_2^p-complete [103, 314, 443], hence better than the model checking problems of CTL* and LTL. Moreover, there are further good reasons to prefer LeftCTL* (if the expressiveness allows this), since in contrast to LTL or CTL*, LeftCTL* can be translated to CTL so that available model checking tools for CTL can also be used for model checking LeftCTL*. Furthermore, in Section 5.4.5, we will integrate the LeftCTL* to CTL translation in the translations for LTL and CTL*, so that LTL and CTL* model checking also benefits from this reduction.

Therefore, it is interesting to define sublogics of CTL* that can be reduced to CTL, but that allow more succinct specifications. To this end, a lot of other sublogics of CTL* have been considered [158, 163, 300, 434] in order to find a good compromise between readability, expressiveness, and complexity. From a theoretical point of view, these logics may be uninteresting, since most of them are equivalent to CTL. However, from a practical point of view, it is necessary to make specifications more readable and (even exponentially) more succinct, as the above example formula $\mathcal{H}_n$ has shown.

In this section, we therefore list further sublogics to motivate the translations from these logics to CTL, which may be used here as synonym for the alternation-free μ-calculus. The translations themselves are given in the following sections (Section 5.3.3 and Section 5.3.4). For a systematic classification of sublogics [163] introduced a hierarchy of branching time temporal logics. These are given in the following definition (see also [158]):

Definition 5.10 (Sublanguages of CTL* (II)). *Given a finite set of variables V_Σ, the following grammar rules define the state formulas for all branching time sublanguages of CTL* given below:*

$$S ::= V_\Sigma \mid \neg S \mid S \wedge S \mid S \vee S \mid \mathsf{E}P \mid \mathsf{A}P$$

Depending on the grammar rules for the path formulas, the following branching time sublanguages are defined:

$$
\begin{array}{ll}
\mathsf{B}(\mathsf{X}): & P ::= \mathsf{X}S \\
\mathsf{B}(\mathsf{F}): & P ::= \mathsf{F}S \\
\mathsf{B}(\mathsf{X},\mathsf{F}): & P ::= \mathsf{X}S \mid \mathsf{F}S \\
\mathsf{B}(\neg,\wedge,\mathsf{X},\mathsf{F}): & P ::= \neg P \mid P \wedge P \mid \mathsf{X}S \mid \mathsf{F}S \\
\mathsf{B}(\mathsf{X},\mathsf{F},\underline{\mathsf{U}}): & P ::= \mathsf{X}S \mid \mathsf{F}S \mid [S\ \underline{\mathsf{U}}\ S] \\
\mathsf{B}(\neg,\wedge,\mathsf{X},\mathsf{F},\underline{\mathsf{U}}): & P ::= \neg P \mid P \wedge P \mid \mathsf{X}S \mid \mathsf{F}S \mid [S\ \underline{\mathsf{U}}\ S] \\
\mathsf{B}(\neg,\wedge,\mathsf{X},\mathsf{F},\underline{\mathsf{U}},\mathsf{GF}): & P ::= \neg P \mid P \wedge P \mid \mathsf{X}S \mid \mathsf{F}S \mid [S\ \underline{\mathsf{U}}\ S] \mid \mathsf{GF}S
\end{array}
$$

Hence, $\mathsf{B}(M)$ allows, after a path quantifier, exactly formulas that start with an operator of the set M. As a further restriction, the arguments of temporal operators must be state formulas. Hence, temporal operators are not allowed to be nested without separating the nesting with a path quantifier. In the logic $\mathsf{B}(\neg,\wedge,\mathsf{X},\mathsf{F},\underline{\mathsf{U}},\mathsf{GF})$, we consider GF as a single macro operator, although it is composed of two temporal operators. Some authors, e.g. [158], write F^∞ and G^∞ instead of GF and FG, respectively, to stress that these macros should be treated as a single operators. We already know that $\mathsf{GF}a$ means that a holds infinitely often, while $\mathsf{FG}a$ means that a holds from a certain point of time on.

Hence, in the above terminology, $\mathsf{B}(\{\mathsf{X},\mathsf{G},\mathsf{F},\mathsf{W},\mathsf{U},\mathsf{B},\underline{\mathsf{W}},\underline{\mathsf{U}},\underline{\mathsf{B}}\})$ is our CTL logic, and the logic UB is $\mathsf{B}(\{\mathsf{X},\mathsf{G},\mathsf{F}\})$. LTL and CTL^* can not be expressed in this terminology, since these logics allow the nesting of temporal operators in an arbitrary manner. For the above logics, expressiveness and complexity results have been derived in [163] (cf. Figure 5.3). We list these results in this section and also list some simple remarks that prove some of these results. Other results are however deeper and require certain transformations that we will develop in Section 5.3.3 and Section 5.3.4. The following remarks are however easily seen:

1. For two logics $\mathsf{B}(M_1)$ and $\mathsf{B}(M_2)$ with $M_1 \subseteq M_2$ we trivially have $\mathsf{B}(M_1) \subseteq \mathsf{B}(M_2)$, and hence $\mathsf{B}(M_1) \lesssim \mathsf{B}(M_2)$.
2. For every set of operators M, we have $\mathsf{B}(\{\neg,\wedge\} \cup M) \approx \mathsf{B}(\{\neg,\wedge,\vee\} \cup M)$. Similar to the previous point, we can derive $\neg(\neg P \wedge \neg P)$ from P by the rules of $\mathsf{B}(\{\neg,\wedge\} \cup M)$.
3. $\mathsf{B}(\{\neg,\mathsf{F}\} \cup M) \approx \mathsf{B}(\{\neg,\mathsf{F},\mathsf{G}\} \cup M)$, since $\mathsf{G}\varphi = \neg\mathsf{F}\neg\varphi$ holds, and we can derive $\neg\mathsf{F}\neg S$ from P in $\mathsf{B}(\{\neg,\mathsf{F}\} \cup M)$.
4. For every set of operators M with $\{\neg,\wedge,\vee,\underline{\mathsf{U}}\} \subseteq M$, we have $\mathsf{B}(M) \approx \mathsf{B}(M \setminus \{\mathsf{G},\mathsf{F},\mathsf{W},\mathsf{U},\mathsf{B},\underline{\mathsf{W}},\underline{\mathsf{B}}\})$, since we have already seen that we can define these temporal operators with $\underline{\mathsf{U}}$. Alternatively, we can also use any of the binary temporal operators, since each of them can express $\underline{\mathsf{U}}$.
5. We will see in the next section that for every set of operators M with $\underline{\mathsf{U}} \in M$, we have $\mathsf{B}(M) \approx \mathsf{B}(M \setminus \{\neg,\wedge\})$ and also $\mathsf{B}(M) \approx \mathsf{B}(M \setminus \{\neg,\wedge,\vee\})$, i.e., we can eliminate applications of path quantifiers to subformulas starting with Boolean operators.

The above remarks already show that we have some freedom for the construction of equally expressive logics. One thing that was discovered quite early was that Boolean operations may occur after path quantifiers, even in

an arbitrary nesting depth (Section 5.3.3). This lead to the definition of UB^+ and CTL^+ that are defined in the above terminology as $\mathsf{B}(\{\neg, \wedge, \vee, \mathsf{X}, \mathsf{G}, \mathsf{F}\})$ and $\mathsf{B}(\{\neg, \wedge, \vee, \mathsf{X}, \mathsf{G}, \mathsf{F}, \mathsf{W}, \mathsf{U}, \mathsf{B}, \underline{\mathsf{W}}, \underline{\mathsf{U}}, \underline{\mathsf{B}}\})$. It can be shown that this transformation may require an exponential increase of the size of the formulas [4, 266, 513]. Note that this is not only a criterion on the complexity of a logic, but also a criterion on the succinctness.

Also, it has been known for a long time, that path quantifiers can be shifted inwards over some temporal operators, e.g., the formulas $\mathsf{AGF}a$ and $\mathsf{AGAF}a$ are equivalent. In contrast to the previous transformation, this transformation only requires a linear increase of the size of the formulas.

The author has defined a temporal logic called $\mathsf{LeftCTL}^*$ [434] that was designed as a superset of CTL^+ such that all available laws are exploited to eliminate Boolean operations that occur after path quantifiers (Section 5.3.3) and to shift path quantifiers inwards (see Section 5.3.4). A formal definition of the languages $\mathsf{LeftCTL}^*$ and CTL^+ amongst others is given by the following definition in form of BNF grammars:

Definition 5.11 (Sublanguages of CTL^* (III)). *Given a finite set of variables V_Σ, the following grammars define the logics UB^+, CTL^+, CTL^{++}, $\mathsf{LeftCTL}^{++}$, and $\mathsf{LeftCTL}^*$:*

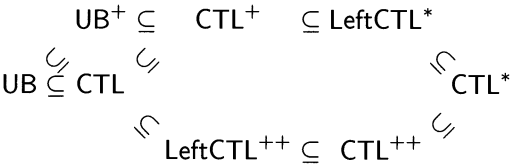
$$
\begin{array}{lll}
\mathsf{LeftCTL}^*: & S ::= & V_\Sigma \mid \neg S \mid S \wedge S \mid S \vee S \mid \mathsf{E}P_E \mid \mathsf{A}P_A \\
& P_E ::= & S \mid \neg P_A \mid P_E \wedge P_E \mid P_E \vee P_E \mid \mathsf{X}P_E \mid \mathsf{G}S \mid \mathsf{F}P_E \\
& & \mid [P_E\ \mathsf{W}\ S] \mid [S\ \mathsf{U}\ P_E] \mid [P_E\ \mathsf{B}\ S] \\
& & \mid [P_E\ \underline{\mathsf{W}}\ S] \mid [S\ \underline{\mathsf{U}}\ P_E] \mid [P_E\ \underline{\mathsf{B}}\ S] \\
& P_A ::= & S \mid \neg P_E \mid P_A \wedge P_A \mid P_A \vee P_A \mid \mathsf{X}P_A \mid \mathsf{G}P_A \mid \mathsf{F}S \\
& & \mid [P_A\ \mathsf{W}\ S] \mid [P_A\ \mathsf{U}\ S] \mid [S\ \mathsf{B}\ P_E] \\
& & \mid [P_A\ \underline{\mathsf{W}}\ S] \mid [P_A\ \underline{\mathsf{U}}\ S] \mid [S\ \underline{\mathsf{B}}\ P_E] \\
\mathsf{LeftCTL}^{++}: & S ::= & V_\Sigma \mid \neg S \mid S \wedge S \mid S \vee S \mid \mathsf{E}P_E \mid \mathsf{A}P_A \\
& P_E ::= & S \mid \mathsf{X}P_E \mid \mathsf{G}S \mid \mathsf{F}P_E \\
& & \mid [P_E\ \mathsf{W}\ S] \mid [S\ \mathsf{U}\ P_E] \mid [P_E\ \mathsf{B}\ S] \\
& & \mid [P_E\ \underline{\mathsf{W}}\ S] \mid [S\ \underline{\mathsf{U}}\ P_E] \mid [P_E\ \underline{\mathsf{B}}\ S] \\
& P_A ::= & S \mid \mathsf{X}P_A \mid \mathsf{G}P_A \mid \mathsf{F}S \\
& & \mid [P_A\ \mathsf{W}\ S] \mid [P_A\ \mathsf{U}\ S] \mid [S\ \mathsf{B}\ P_E] \\
& & \mid [P_A\ \underline{\mathsf{W}}\ S] \mid [P_A\ \underline{\mathsf{U}}\ S] \mid [S\ \underline{\mathsf{B}}\ P_E] \\
\mathsf{CTL}^{++}: & S ::= & V_\Sigma \mid \neg S \mid S \wedge S \mid S \vee S \mid \mathsf{E}P_1 \mid \mathsf{A}P_1 \\
& P_1 ::= & \mathsf{X}P \mid \mathsf{G}P \mid \mathsf{F}P \\
& & \mid [P\ \mathsf{W}\ P] \mid [P\ \mathsf{U}\ P] \mid [P\ \mathsf{B}\ P] \\
& & \mid [P\ \underline{\mathsf{W}}\ P] \mid [P\ \underline{\mathsf{U}}\ P] \mid [P\ \underline{\mathsf{B}}\ P] \\
& P ::= & S \mid \neg P \mid P \wedge P \mid P \vee P \mid P_1 \\
\mathsf{CTL}^+: & S ::= & V_\Sigma \mid \neg S \mid S \wedge S \mid S \vee S \mid \mathsf{E}P \mid \mathsf{A}P \\
& P ::= & \neg P \mid P \wedge P \mid P \vee P \mid \mathsf{X}S \mid \mathsf{G}S \mid \mathsf{F}S \\
& & \mid [S\ \mathsf{W}\ S] \mid [S\ \mathsf{U}\ S] \mid [S\ \mathsf{B}\ S] \\
& & \mid [S\ \underline{\mathsf{W}}\ S] \mid [S\ \underline{\mathsf{U}}\ S] \mid [S\ \underline{\mathsf{B}}\ S] \\
\mathsf{UB}^+: & S ::= & V_\Sigma \mid \neg S \mid S \wedge S \mid S \vee S \mid \mathsf{E}P \mid \mathsf{A}P \\
& P ::= & \neg P \mid P \wedge P \mid P \vee P \mid \mathsf{X}S \mid \mathsf{G}S \mid \mathsf{F}S
\end{array}
$$

As in the grammars before, the nonterminals S and P with or without indices describe sets of state or path formulas, respectively. LeftCTL* formulas allow arbitrary deep nested Boolean operators after path quantifiers, and also certain arbitrary deep nestings of temporal operators. However, the nesting of temporal operators is only allowed on one argument of the binary temporal operators. As the logic has been developed from a strictly weaker logic, given in [433], that only considers the W operator, and as the nestings are allowed only for the left hand arguments of W operators, the logic has been named as LeftCTL*. Note that the nesting side of U and B operators depends on whether the formula occurs after an E or A quantifier. Therefore, the grammar distinguishes between path formulas that occur after an E or A quantifier by the nonterminals P_E and P_A, respectively.

CTL^{++} contains all formulas of CTL* where no path quantifier is applied to an expression with a Boolean operator on top, i.e., each path quantifier in CTL^{++} must be followed by a temporal operator. Hence, we must distinguish in CTL^{++} between path formulas P_1 beginning with a temporal operator and general path formulas P.

LeftCTL^{++} is the intersection of LeftCTL* and CTL^{++}, i.e., the subset of LeftCTL*, where it is not allowed for Boolean operators to occur immediately after a path quantifier. Finally, CTL$^+$ is the generalization of CTL that allows arbitrary deep nestings of Boolean operators after path quantifiers. Path formulas of CTL$^+$ that start with a temporal operator are however, exactly the same as in CTL.

Note that we distinguish between the sets of formulas and the expressiveness of the logic. For the above logics, we have syntactically the following inclusion diagram (expressiveness results are given in Figure 5.3):



We have already noted that efficient verification tools are available for CTL, so that logics weaker than CTL are not of much interest. We have already mentioned (and will prove in detail later) that LeftCTL* is as expressive as CTL, but the logics differ dramatically in their succinctness.

Considering the expressiveness of the logics CTL and LeftCTL*, we can say that it is sufficient for most practical applications. However, there is one aspect that is sometimes required in practice that can not be specified in CTL, and hence not in LeftCTL*: There is no CTL formula that is equivalent to the CTL* formula EGFφ which states that there is a path where φ holds infinitely often [160, 163, 310]. However, for some applications, specifications of the form E[(GFφ) $\wedge$ Φ] or A[(GFφ) $\rightarrow$ Φ] occur naturally. For example, $\neg$E[(GF*req*) $\wedge$ G$\neg$*ack*] means that there is no computation path where a

request *req* is given infinitely often without acknowledge *ack*. Another example involves the fairness of an arbiter that administrates the access of n components to a shared resource. For a correct function of such a system, it is required that each component releases the shared resource after some time. The specification $\mathsf{A}\left[\left(\bigwedge_{i=0}^{n} \neg\mathsf{FG}own_i\right) \rightarrow \left(\bigwedge_{i=0}^{n} \neg\mathsf{FG}[req_i \wedge \neg ack_i]\right)\right]$ means that it is not the case that a component requests access while never being granted access (provided that each component releases the shared resource after some time).

Properties of this kind are therefore expressed by restricting the path quantifiers by *fairness constraints*, as the set of computation paths where the quantification is applied on is restricted only to certain *fair* paths instead of the set of all paths. We can alternatively express such restrictions in our Kripke structures by adding appropriate fairness constraints to the structure. If the fair sets $F_1, \ldots, F_f \subseteq \mathcal{S}$ that have to be visited infinitely often by a fair path can be identified with propositional formula α_i such that $[\![\alpha_i]\!]_{\mathcal{K}'} = F_i$ holds (where $\mathcal{K}'$ is the corresponding structure without fairness constraints), then some logics can express the restrictions by fairness constraints directly in their formulas. For example, in CTL* we just have to replace every subformula $\mathsf{E}\varphi$ by $\mathsf{E}\left[\left(\bigwedge_{i=1}^{f}\right) \wedge \varphi\right]$, and analogously, we replace every subformula $\mathsf{A}\varphi$ by $\mathsf{A}\left[\left(\bigwedge_{i=1}^{f}\right) \rightarrow \varphi\right]$. As these are also CTL* formulas, we see that CTL* can directly express fairness constraints.

CTL, and hence all expressively equivalent logics like LeftCTL*, are however not powerful enough to do this [160, 163, 310]. As such fairness constraints can not be expressed in CTL, an extension to fairness constraints of CTL has been proposed in [175] which on the one hand, extends the expressiveness of CTL, but on the other hand still retains a polynomial model checking algorithm. We will see this in the next section and list here an adaption of the last definition.

Definition 5.12 (Sublanguages of CTL* (IV)). *Given propositional formulas* $\alpha_1, \ldots, \alpha_f$, *we define* $\Phi := \bigwedge_{i=1}^{f} \mathsf{GF}\alpha_i$ *and the macros* $\mathsf{E}_\Phi\varphi := \mathsf{E}(\Phi \wedge \varphi)$ *and* $\mathsf{A}_\Phi\varphi := \mathsf{A}(\Phi \rightarrow \varphi)$. *The logics* FairCTL, FairCTL$^+$, FairLeftCTL^{++}, FairLeftCTL* *are obtained by changing the grammar rules for* CTL, CTL$^+$, LeftCTL^{++}, *and* LeftCTL* *in that all occurrences of* E *and* A *are replaced with* E_Φ *and* A_Φ, *respectively.*

The diagram for expressivenesses of the different logics we have encountered so far is given in Figure 5.3. We will prove the equal expressiveness results by the transformations in the following sections. For the strictness of the listed $\overset{\prec}{\approx}$ relations, we just note here without proof that the following example formulas can be expressed in some of the logics, but not in the other ones below in the hierarchy (see [162]).

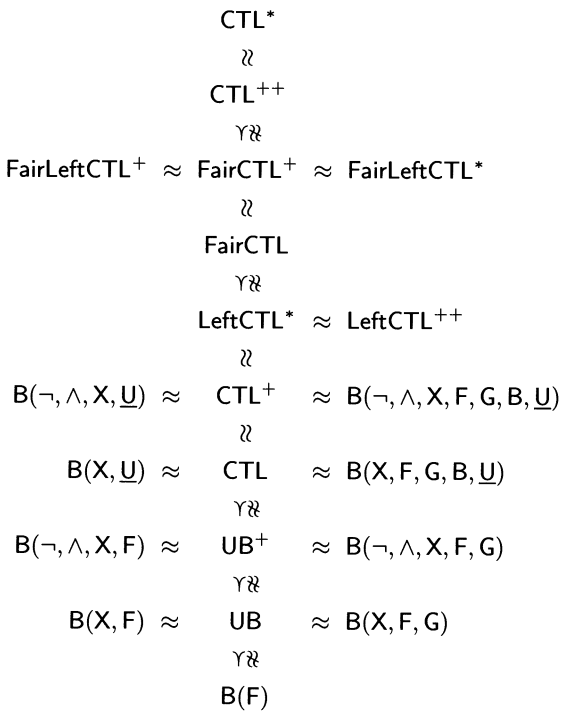


Fig. 5.3. Sublanguages of CTL* and their expressiveness

formula ($a, b \in V_\Sigma$)	expressible in	but not in
$\mathsf{E}[\mathsf{G}a \wedge \mathsf{F}b]$	UB^+	UB
$\mathsf{E}[a \,\underline{\mathsf{U}}\, b]$	CTL	UB^+
$\mathsf{EGF}a$	FairCTL	CTL
$\mathsf{AF}[a \wedge \mathsf{X}a]$	CTL*	FairCTL

Note that $\mathsf{E}[\mathsf{G}a \wedge \mathsf{F}b] = \mathsf{E}[a \,\underline{\mathsf{U}}\, (b \wedge \mathsf{EG}a)]$ holds, so that this formula can be expressed in CTL as well. Note further that $\mathsf{AF}[a \wedge \mathsf{X}a]$ can be translated to the alternation-free μ-calculus: $\mathsf{G}[\varphi \vee \mathsf{X}\varphi]$ can be easily translated to the ω-automaton $\mathcal{A}_\exists(\{q\}, \neg q, \Phi_\mathcal{R}, 1)$ with $\Phi_\mathcal{R} := \neg q \wedge \neg\varphi \wedge \mathsf{X}q \vee \varphi \wedge \neg\mathsf{X}q$ (see Figure 5.4). Hence, according to Theorem 4.67, we obtain the following μ-calculus formula for $\mathsf{EG}[\varphi \vee \mathsf{X}\varphi]$:

$$\mathsf{fix}\begin{bmatrix} y_{\{\}} \overset{\nu}{=} \varphi \wedge \Diamond y_{\{\}} \vee \neg\varphi \wedge \Diamond y_{\{q\}} \\ y_{\{q\}} \overset{\nu}{=} \varphi \wedge \Diamond y_{\{\}} \end{bmatrix} \mathsf{in}\ y_{\{\}}\ \mathsf{end}$$

Hence, we find the following equivalences:

- $\mathsf{EG}[\varphi \vee \mathsf{X}\varphi] = \nu y.\ \varphi \wedge \Diamond y \vee \neg\varphi \wedge \Diamond(\varphi \wedge \Diamond y)$
- $\mathsf{AF}[\varphi \wedge \mathsf{X}\varphi] = \mu y.\ (\varphi \vee \Box y) \wedge (\neg\varphi \vee \Box(\varphi \vee \Box y))$

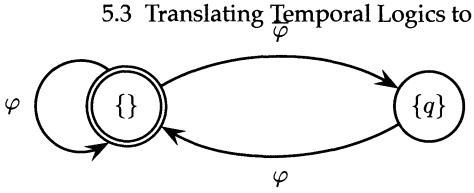


Fig. 5.4. State transition diagram of the automaton for $\mathsf{G}[\varphi \vee \mathsf{X}\varphi]$

Moreover, it is easily seen that the above formulas can be simplied as follows:

- $\mathsf{EG}[\varphi \vee \mathsf{X}\varphi] = \nu y.\ \varphi \wedge \Diamond y \vee \neg\varphi \wedge \Diamond(\varphi \wedge y)$
- $\mathsf{AF}[\varphi \wedge \mathsf{X}\varphi] = \mu y.\ (\varphi \vee \Box y) \wedge (\neg\varphi \vee \Box(\varphi \vee y))$

To formally prove the equivalence, one needs to have a decision procedure for the satisfiabilty of μ-calculus formulas as described in Section B.3.

5.3 Translating Temporal Logics to the μ-Calculus

As expected, we can translate every formula of CTL* to an equivalent μ-calculus formula. For example, a direct translation is given in [135], where a CTL* formula Φ is translated to an equivalent μ-calculus formula Ψ of length $O(2^{2^{|\Phi|}})$ and alternation depth 2. We will not follow this approach, and will only show in this section the translation of some simpler sublogics of CTL* to equivalent μ-calculus formulas. However, in Section 5.4, we will show how we can compute for arbitrary LTL formulas $\mathsf{A}\varphi$ an ω-automaton $\mathfrak{A}_\varphi$ that is equivalent to φ. The further translations of $\mathfrak{A}_\varphi$ according to the theorems of Section 4.8.3 will then show that we can translate every LTL formula $\mathsf{A}\varphi$ to an equivalent μ-calculus formula Φ of size $O(|\varphi|\, 2^{|\varphi|})$ and an alternation depth not higher than 2. Moreover, using the PQNF normal form of Definition 2.49 on page 85 allows us to apply these results to CTL* formulas. Hence, it follows that for every CTL* formula φ we can compute an equivalent μ-calculus formula Φ of single exponential size.

Hence, our translation of CTL* or LTL to the μ-calculus is done indirectly via an intermediate translation to ω-automata. We will present these procedures in detail in Section 5.4. In this section, we consider instead the direct translation of sublogics of CTL* to the μ-calculus. In particular, we will consider in this section the logics CTL, FairCTL, CTL^2 [300], and LeftCTL* [434]. The reason for this is that these logics can be translated in a very intuitive way to the μ-calculus. In fact, as these logics all refer to CTL, and CTL can be directly interpreted as fragment of the μ-calculus, we may also view the mentioned logics as fragments of the μ-calculus. However, while the translation of CTL, FairCTL, and CTL^2 yield linear sized μ-calculus formulas, the translation of LeftCTL* may result in an exponential blow-up. In contrast to the other mentioned logics, LeftCTL* is therefore not in a one-to-one correspondence with the μ-calculus.

The outline of this section is as follows: in the next section, we show that CTL and FairCTL formulas can be translated in linear time to equivalent μ-calculus formulas of alternation depth 1 and 2, respectively. In Section 5.3.2, we consider the temporal logic CTL^2 [300] of Kupferman and Grumberg. The idea of this logic is roughly as follows: while CTL provides macro operators that consist of pairs of a path quantifier and a temporal operator, CTL^2 provides macro operators that consist of a path quantifier and two nested temporal operators. In [300], it has been shown how CTL^2 can be translated to linear-sized μ-calculus formulas of alternation depth 2. We will follow these ideas in Section 5.3.2. Hence, the model checking algorithms we have already presented for the μ-calculus can be applied more or less directly to CTL, FairCTL, and CTL^2 and give polynomial time verification procedures.

The translation of LeftCTL* requires a bit more effort: In Section 5.3.3, we will describe how applications of path quantifiers to subformulas starting with Boolean operators can be eliminated. In Section 5.3.4, we moreover see that path quantifiers also induce the presence of path quantifiers in their subformulas, so that we can add path quantifiers at certain positions. These two transformations are sufficient to prove all of the equal expressiveness results of Figure 5.3. However, a naive use of our results would transform given LeftCTL* formulas to equivalent CTL formulas of size $O(|\Phi| \cdot |\Phi|!)$. We will therefore show in Section 5.3.5 that the translation can be improved by sharing subformulas in that we translate to the vectorized μ-calculus. As a result, we will obtain formulas of size $O(|\Phi|\, 2^{2|\Phi|})$, which will be a great improvement. We therefore obtain an exponential time verification procedure for LeftCTL*.

5.3.1 CTL and FairCTL as Fragments of the μ-Calculus

CTL and FairCTL are subsets of CTL*, but they can also be viewed as subsets of the μ-calculus. This was pointed out by Clarke and Emerson in [102, 160], and in fact, they designed CTL as a macro language for the μ-calculus. Hence, the translation of CTL and FairCTL to equivalent μ-calculus formulas is done by replacing the CTL and FairCTL operators with their definitions:

Theorem 5.13 (Translating CTL and FairCTL to μ-Calculus). *Given a fairness constraint $\Phi = \bigwedge_{i=1}^{n} \mathsf{GF}\alpha_i$ (with propositional formulas α_i), define*[1] *$\Phi_{\mathsf{fair}} = \nu z. \bigwedge_{i=1}^{f} \Diamond[\mu x. z \wedge \alpha_i \vee \Diamond x]$. The following equations rewrite every* FairCTL *formula φ to an equivalent μ-calculus formula of size $O(|\varphi|)$ and alternation depth 2:*

- $\mathsf{E}_\Phi \mathsf{X}\varphi = \Diamond(\Phi_{\mathsf{fair}} \wedge \varphi)$
- $\mathsf{E}_\Phi \mathsf{G}\varphi = \nu z.\ \varphi \wedge \bigwedge_{i=1}^{n} \Diamond[\mu x. z \wedge \alpha_i \vee \varphi \wedge \Diamond x]$
- $\mathsf{E}_\Phi \mathsf{F}\varphi = \mu x.\ \Phi_{\mathsf{fair}} \wedge \varphi \vee \Diamond x$

[1] Of course, every other μ-calculus formula to compute the fair states can be used as well for this purpose (cf. Section 3.7).

- $\mathsf{E}_\Phi\,[\varphi\ \underline{\mathsf{U}}\ \psi] = \mu x.\ \Phi_{\mathsf{fair}} \wedge \psi \vee \varphi \wedge \Diamond x$
- $\mathsf{E}_\Phi\,[\varphi\ \underline{\mathsf{W}}\ \psi] = \mu x.\ \Phi_{\mathsf{fair}} \wedge \psi \wedge \varphi \vee \neg(\Phi_{\mathsf{fair}} \wedge \psi) \wedge \Diamond x$
- $\mathsf{E}_\Phi\,[\varphi\ \underline{\mathsf{B}}\ \psi] = \mu x.\ \neg\psi \wedge ((\Phi_{\mathsf{fair}} \wedge \varphi) \vee \Diamond x)$
- $\mathsf{E}_\Phi\,[\varphi\ \mathsf{U}\ \psi] = \mathsf{E}_\Phi\,[\varphi\ \underline{\mathsf{U}}\ \psi] \vee \mathsf{E}_\Phi\mathsf{G}\varphi$
- $\mathsf{E}_\Phi\,[\varphi\ \mathsf{W}\ \psi] = \mathsf{E}_\Phi\,[\varphi\ \underline{\mathsf{W}}\ \psi] \vee \mathsf{E}_\Phi\mathsf{G}\neg\psi$
- $\mathsf{E}_\Phi\,[\varphi\ \mathsf{B}\ \psi] = \mathsf{E}_\Phi\,[\varphi\ \underline{\mathsf{B}}\ \psi] \vee \mathsf{E}_\Phi\mathsf{G}(\neg\varphi \wedge \neg\psi)$

For the other operators, note that the following negation laws hold:

$$\mathsf{A}_\Phi\mathsf{X}\varphi = \neg\mathsf{E}_\Phi\mathsf{X}(\neg\varphi)$$
$$\mathsf{A}_\Phi\mathsf{G}\varphi = \neg\mathsf{E}_\Phi\mathsf{F}(\neg\varphi) \qquad \mathsf{A}_\Phi\mathsf{F}\varphi = \neg\mathsf{E}_\Phi\mathsf{G}(\neg\varphi)$$
$$\mathsf{A}_\Phi\,[\varphi\ \mathsf{U}\ \psi] = \neg\mathsf{E}[(\neg\varphi)\ \underline{\mathsf{B}}\ \psi] \qquad \mathsf{A}_\Phi\,[\varphi\ \underline{\mathsf{U}}\ \psi] = \neg\mathsf{E}[(\neg\varphi)\ \mathsf{B}\ \psi]$$
$$\mathsf{A}_\Phi\,[\varphi\ \mathsf{B}\ \psi] = \neg\mathsf{E}_\Phi\,[(\neg\varphi)\ \underline{\mathsf{U}}\ \psi] \qquad \mathsf{A}_\Phi\,[\varphi\ \underline{\mathsf{B}}\ \psi] = \neg\mathsf{E}_\Phi\,[(\neg\varphi)\ \mathsf{U}\ \psi]$$
$$\mathsf{A}_\Phi\,[\varphi\ \mathsf{W}\ \psi] = \neg\mathsf{E}_\Phi\,[(\neg\varphi)\ \underline{\mathsf{W}}\ \psi] \qquad \mathsf{A}_\Phi\,[\varphi\ \underline{\mathsf{W}}\ \psi] = \neg\mathsf{E}_\Phi\,[(\neg\varphi)\ \mathsf{W}\ \psi]$$

For checking a CTL *formula, i.e. the special case where $\Phi :\equiv 1$ holds, we use the above equations with $\Phi_{\mathsf{fair}} := \Phi_{\mathsf{inf}} := \nu z.\Diamond z$, except for*

- $\mathsf{E}_\Phi\mathsf{G}\varphi = \nu z.\ \varphi \wedge \Diamond x$

Hence, the translation of CTL *yields alternation-free μ-calculus formulas.*

The fixpoints are rather clear: for example, consider $\mathsf{E}_\Phi\,[\varphi\ \underline{\mathsf{U}}\ \psi]$. By the semantics, a state satisfies this formula if it either satisfies ψ or if it satisfies φ and has a successor state that satisfies $\mathsf{E}_\Phi\,[\varphi\ \underline{\mathsf{U}}\ \psi]$. However, we have to be sure that the path that is thereby followed is a fair one. This is done by reaching out for $\Phi_{\mathsf{fair}} \wedge \psi$ instead of reaching out for only ψ. Note that the satisfaction of fairness constraints can be postponed for an arbitrary long (finite) time. The other fixpoints can be explained in the same manner; the only one that is more difficult is the one for $\mathsf{E}_\Phi\mathsf{G}\varphi$ which was discussed on page 165.

The situation is simplified for CTL, when no fairness constraints have to be considered. In principle, we could then use the same equations as before with $f = 1$ and $\alpha_1 = 1$. Hence, we would have $\Phi_{\mathsf{fair}} = \nu z.\Diamond[\mu x.z \vee \Diamond x]$, which is equivalent to $\nu z.\Diamond z$: both fixpoints determine the set of states that have at least one infinite path. In the case, where every state of the Kripke structure has at least one successor state, we can even use $\Phi_{\mathsf{fair}} = 1$, so that the situation is even simpler. This can be achieved by replacing $\mathcal{R}$ with the relation $\mathcal{R}_{\mathsf{inf}}$ as explained in Section 3.7.

Hence, by our previous results on the μ-calculus, we see that FairCTL formulas can be checked, at least in time $O(|\mathcal{K}|\,|\mathcal{S}|\,|\Phi|^2)$, according to Theorem 3.40, and even in time $O(|\mathcal{K}|\,|\mathcal{S}|\,f\,|\Phi|)$ with our remarks after Theorem 3.43 on page 165. [103] even shows that it can be done in time $O(|\mathcal{K}|\,|\mathcal{S}_{\mathsf{fair}}|\,|\Phi|)$, where $\mathcal{S}_{\mathsf{fair}}$ is the set of fair states, i.e. those that satisfy Φ_{fair}.

Model checking CTL formulas in Kripke structures with fairness constraints can obviously be reduced to checking the corresponding FairCTL formulas in the same structure without fairness constraints, so that we obtain the same results. However, the model checking problem is simpler when

there are no fairness constraints: In this case, we use the alternation-free formula $\Phi_{\mathsf{fair}} = \nu z.\Diamond z$ and $\mathsf{E}_{\Phi}\mathsf{G}\varphi = \nu z.\ \varphi \wedge \Diamond x$, so that an alternation-free formula is obtained. We therefore know that the complexity is $O(|\mathcal{K}|\,|\Phi|)$.

Theorem 5.14 (Complexity of CTL Model Checking). *For every Kripke structure $\mathcal{K} = (\mathcal{I}, \mathcal{S}, \mathcal{R}, \mathcal{L})$ and every CTL formula Φ, the computation of $[\![\Phi]\!]_{\mathcal{K}}$ can be done in time $O(|\mathcal{K}|\,|\Phi|)$. Given propositional formulas $\alpha_1, \ldots, \alpha_f$ as fairness constraints for a FairCTL formula Φ, the states where Φ holds can be computed in time $O(|\mathcal{K}|\,|\mathcal{S}|\,f\,|\Phi|)$.*

Note that we have assumed that the same fairness constraints were used for all path quantifiers. In principle, every occurrence of a path quantifier could have its own fairness constraints which would then require to compute the corresponding sets of fair states for every particular fixpoint iteration (due to a CTL operator). Moreover, we could consider different kinds of fairness constraints. Above, we considered the generalized Büchi constraints, where all sets $[\![\alpha_i]\!]_{\mathcal{K}}$ must be visited infinitely often. Instead, we could replace these constraints with the more general Streett condition $\bigwedge_{i=1}^{n} \mathsf{GF}\alpha_i \vee \mathsf{FG}\beta_i$. We have already seen in Theorem 4.65 on page 267 how the corresponding fair states can be computed.

5.3.2 CTL^2 as a Fragment of the μ-Calculus

In [300], Kupferman and Grumberg proposed another interesting logic called CTL^2 which is a strict superset of CTL, and a strict subset of CTL* (both in terms of expressiveness as well as in terms of syntax). The idea is inspired by generalizing the restrictions of CTL. Recall that CTL requires that path quantifiers and temporal operators occur in pairs, i.e., after a path quantifier there is exactly one temporal operator that is not within the scope of another path quantifier.

The idea of Kupferman and Grumberg was then to consider a similar logic that provides macro operators that consist of pairs of a path quantifier and one or two nested temporal operators (they also allowed a Boolean operator instead of a temporal operator). For this reason, it is clear that CTL is a strict subset of CTL^2. In particular, Kupferman and Grumberg considered the following logic:

$$
\begin{aligned}
S &::= V_\Sigma \mid \neg S \mid S \wedge S \mid S \vee S \mid \mathsf{E}P \mid \mathsf{A}P \\
P &::= P_1 \mid P_2 \\
P_1 &::= \mathsf{X}S \mid [S \,\underline{\mathsf{U}}\, S] \mid \neg P_1 \\
P_2 &::= \mathsf{X}P_1 \mid [P_1 \,\underline{\mathsf{U}}\, S] \mid [S \,\underline{\mathsf{U}}\, P_1] \mid \neg P \mid S \wedge P_1 \mid P_1 \wedge S
\end{aligned}
$$

In the above grammar, the formulas derived from S are again the state formulas and those derived from P are the path formulas. There are path formulas of degree 1 or 2, meaning path formulas that are derived from P_1 or P_2.

Those that are derived from P_1 start with a temporal operator whose arguments must be state formulas. Path formulas derived from P_2 allow instead one more operator, i.e., either a Boolean or another temporal operator.

The overall idea is better understood by discussing the following equivalent logic (we have however already eliminated the applications of the path quantifiers to Boolean operators):

$$
\begin{aligned}
S ::= {} & V_\Sigma \mid \neg S \mid S \wedge S \mid S \vee S \mid \mathsf{E}[S \vee T] \mid \mathsf{E}[S \wedge T] \\
T ::= {} & \mid \mathsf{EX}S \mid \mathsf{E}[S \,\underline{\mathsf{U}}\, S] \mid \mathsf{E}[S \;\mathsf{B}\; S] \\
& \mid \mathsf{EXX}S \mid \mathsf{EX}\,[S \,\underline{\mathsf{U}}\, S] \mid \mathsf{EX}\,[S \;\mathsf{B}\; S] \\
& \mid \mathsf{E}\,[\mathsf{X}S \,\underline{\mathsf{U}}\, S] \mid \mathsf{E}\,[[S \,\underline{\mathsf{U}}\, S] \,\underline{\mathsf{U}}\, S] \mid \mathsf{E}\,[[S \;\mathsf{B}\; S] \,\underline{\mathsf{U}}\, S] \\
& \mid \mathsf{E}\,[S \,\underline{\mathsf{U}}\, \mathsf{X}S] \mid \mathsf{E}\,[S \,\underline{\mathsf{U}}\, [S \,\underline{\mathsf{U}}\, S]] \mid \mathsf{E}\,[S \,\underline{\mathsf{U}}\, [S \;\mathsf{B}\; S]] \\
& \mid \mathsf{E}\,[\mathsf{X}S \;\mathsf{B}\; S] \mid \mathsf{E}\,[[S \,\underline{\mathsf{U}}\, S] \;\mathsf{B}\; S] \mid \mathsf{E}\,[[S \;\mathsf{B}\; S] \;\mathsf{B}\; S] \\
& \mid \mathsf{E}\,[S \;\mathsf{B}\; \mathsf{X}S] \mid \mathsf{E}\,[S \;\mathsf{B}\; [S \,\underline{\mathsf{U}}\, S]] \mid \mathsf{E}\,[S \;\mathsf{B}\; [S \;\mathsf{B}\; S]]
\end{aligned}
$$

As can be seen CTL^2 allows us to nest after a path quantifier at most two temporal operators, and in the same way, we could define CTL^3, CTL^4, and so on. The surprising fact is that most of these macro operators can be reduced to simple CTL formulas according to the following equations, where α, β, and γ are arbitrary state formulas:

- $\mathsf{EXX}\alpha = \mathsf{EXEX}\alpha$
- $\mathsf{EX}\,[\alpha \,\underline{\mathsf{U}}\, \beta] = \mathsf{EXE}\,[\alpha \,\underline{\mathsf{U}}\, \beta]$
- $\mathsf{EX}\,[\alpha \;\mathsf{B}\; \beta] = \mathsf{EXE}\,[\alpha \;\mathsf{B}\; \beta]$
- $\mathsf{E}\,[(\mathsf{X}\alpha) \,\underline{\mathsf{U}}\, \beta] = \beta \vee \mathsf{EXE}[\alpha \,\underline{\mathsf{U}}\, (\alpha \wedge \beta)]$
- $\mathsf{E}\,[[\alpha \,\underline{\mathsf{U}}\, \beta] \,\underline{\mathsf{U}}\, \gamma] = \gamma \vee \mathsf{E}[(\alpha \vee \beta) \,\underline{\mathsf{U}}\, (\beta \wedge \mathsf{EX}\gamma \vee \gamma \wedge \mathsf{E}[\alpha \,\underline{\mathsf{U}}\, \beta])]$
- $\mathsf{E}\,[[\alpha \;\mathsf{B}\; \beta] \,\underline{\mathsf{U}}\, \gamma] = \gamma \vee \mathsf{E}[(\neg\beta) \,\underline{\mathsf{U}}\, (\alpha \wedge \neg\beta \wedge \mathsf{EX}\gamma \vee \gamma \wedge \mathsf{E}[\alpha \;\mathsf{B}\; \beta])]$
- $\mathsf{E}\,[\alpha \,\underline{\mathsf{U}}\, \mathsf{X}\beta] = \mathsf{E}\,[\alpha \,\underline{\mathsf{U}}\, (\mathsf{EX}\beta)]$
- $\mathsf{E}\,[\alpha \,\underline{\mathsf{U}}\, [\beta \,\underline{\mathsf{U}}\, \gamma]] = \mathsf{E}\,[\alpha \,\underline{\mathsf{U}}\, (\mathsf{E}\,[\beta \,\underline{\mathsf{U}}\, \gamma])]$
- $\mathsf{E}\,[\alpha \,\underline{\mathsf{U}}\, [\beta \;\mathsf{B}\; \gamma]] = \mathsf{E}\,[\alpha \,\underline{\mathsf{U}}\, (\mathsf{E}\,[\beta \;\mathsf{B}\; \gamma])]$
- $\mathsf{E}\,[\mathsf{X}\alpha \;\mathsf{B}\; \beta] = \mathsf{E}\,[(\mathsf{EX}\alpha) \;\mathsf{B}\; \beta]$
- $\mathsf{E}\,[[\alpha \,\underline{\mathsf{U}}\, \beta] \;\mathsf{B}\; \gamma] = \mathsf{E}\,[(\mathsf{E}\,[\alpha \,\underline{\mathsf{U}}\, \beta]) \;\mathsf{B}\; \gamma]$
- $\mathsf{E}\,[[\alpha \;\mathsf{B}\; \beta] \;\mathsf{B}\; \gamma] = \mathsf{E}\,[(\mathsf{E}\,[\alpha \;\mathsf{B}\; \beta]) \;\mathsf{B}\; \gamma]$
- $\mathsf{E}\,[\alpha \;\mathsf{B}\; \mathsf{X}\beta] = \alpha \wedge \neg\mathsf{AX}\beta \vee \mathsf{EXE}[(\neg\beta)\;\mathsf{U}\;(\alpha \wedge \neg\beta \wedge \neg\mathsf{AX}\beta)]$
- $\mathsf{E}\,[\alpha \;\mathsf{B}\; [\beta \,\underline{\mathsf{U}}\, \gamma]] = \mathsf{E}[(\neg\gamma)\;\mathsf{U}\;(\alpha \wedge \mathsf{E}[(\neg\beta)\;\mathsf{B}\;\gamma])]$
- $\mathsf{E}\,[\alpha \;\mathsf{B}\; [\beta \;\mathsf{B}\; \gamma]] = \mathsf{E}[(\neg\beta \vee \gamma) \,\underline{\mathsf{U}}\, (\alpha \wedge \mathsf{E}[(\neg\beta) \,\underline{\mathsf{U}}\, \gamma])] \vee \mathsf{EG}\,[(\neg\beta) \,\underline{\mathsf{U}}\, \gamma]$

Hence, all macro operators of CTL^2 can be reduced to simple CTL formulas apart from the last equation that contains the subformula $\mathsf{EG}\,[\alpha \,\underline{\mathsf{U}}\, \beta]$. It is not difficult to see that $\mathsf{EG}\,[\alpha \,\underline{\mathsf{U}}\, \beta]$ is equivalent to $\mathsf{E}(\mathsf{G}(\alpha \vee \beta) \wedge \mathsf{GF}\beta)$, so that we once more encounter the fixpoint that we have considered in Theorem 3.43 on page 165. Therefore, we obtain the following result:

Theorem 5.15 (Translating CTL^2 to μ-Calculus). *Every formula $\Phi \in \mathsf{CTL}^2$ can be translated to an equivalent μ-calculus formula Ψ of alternation depth not higher than 2 and size $O(|\Phi|)$.*

In fact, most formulas can be translated to the alternation-free μ-calculus. Only the nesting $\mathsf{E}[\alpha\ \mathsf{B}\ [\beta\ \mathsf{B}\ \gamma]]$ requires to generate a fixpoint of alternation depth 2. For this reason, CTL^2 is powerful enough to express $\mathsf{EGF}\varphi = \mathsf{E}[0\ \mathsf{B}\ [0\ \mathsf{B}\ \varphi]]$, which is not possible in CTL.

It is natural to consider generalizations CTL^n of CTL^2 that provide macro operators where at most n temporal operators can appear after a path quantifier. Clearly, we can do the same with CTL^n, i.e., we could list all possible macro operators, and translate them to the μ-calculus. We will thereby obtain μ-calculus formulas of alternation depth ≤ 2 and of a size that is determined by n. No matter what the relationship between the size of the fixpoint formulas is, it is constant for a fixed n. Hence, all logics CTL^n have a polynomial model checking procedure.

5.3.3 Eliminating Quantified Boolean Expressions

One distinction between the temporal logics that we have considered is that some allow Boolean operators to occur after a path quantifier, while other logics do not allow this. In this section, we prove that logics that differ in this kind of formulas have the same expressiveness, provided they have one binary temporal operator like $\underline{\mathsf{U}}$. Hence, for example, the logics, CTL^* and CTL^{++}, CTL^+ and CTL, as well as $\mathsf{LeftCTL}^*$ and $\mathsf{LeftCTL}^{++}$, have the same expressiveness.

It must be noted however, that the transformation from the syntactically richer language to the weaker one gives a potentially enormous blow-up in the length of the formulas. The transformation we present here will turn a given formula of length n to an equivalent one of length $O(nn!)$. It can however be improved so that only an exponential blow-up occurs when common subformulas are shared.

The essential problem that has to be solved is that conjunctions can appear after a E quantifier. To see this, reduce any given temporal logic formula $\Phi \in \mathsf{CTL}^*$ first to the normal form $\mathsf{PQNF}(\Phi)$ (cf. 85), so that we only need to consider subformulas of the form $\mathsf{E}\Psi$ where Ψ contains no path quantifier. Computing further $\mathsf{TDNF}(\Psi)$ allows us then to shift the E quantifier over the eventually arising disjunctions by the law $\mathsf{E}[\varphi \vee \psi] = \mathsf{E}\varphi \vee \mathsf{E}\psi$. Some other laws of this kind are given in the next lemma.

Lemma 5.16. *Given arbitrary path formulas $\varphi, \psi, \Phi \in \mathsf{PF}_\Sigma$, and a state formula α, the following equations hold, where $\mathsf{E}_\Phi\varphi$ and $\mathsf{A}_\Phi\varphi$ are abbreviations for $\mathsf{E}[\Phi \wedge \varphi]$ and $\mathsf{A}[\Phi \rightarrow \varphi]$, respectively.*

$$\models \mathsf{E}_\Phi\neg\varphi = \neg\mathsf{A}_\Phi\varphi \qquad \models \mathsf{A}_\Phi\neg\varphi = \neg\mathsf{E}_\Phi\varphi$$
$$\models \mathsf{E}_\Phi\alpha = \mathsf{E}\Phi \wedge \alpha \qquad \models \mathsf{A}_\Phi\alpha = \mathsf{E}\Phi \rightarrow \alpha$$
$$\models \mathsf{E}_\Phi(\alpha \wedge \varphi) = \alpha \wedge \mathsf{E}_\Phi\varphi \qquad \models \mathsf{A}_\Phi(\alpha \wedge \varphi) = \alpha \wedge \mathsf{A}_\Phi\varphi$$
$$\models \mathsf{E}_\Phi(\varphi \vee \psi) = \mathsf{E}_\Phi\varphi \vee \mathsf{E}_\Phi\psi \qquad \models \mathsf{A}_\Phi(\varphi \wedge \psi) = \mathsf{A}_\Phi\varphi \wedge \mathsf{A}_\Phi\psi$$

Note that the special case $\Phi := 1$ does not allow us to replace E1 with 1: E1 holds in a given state iff there is an infinite path starting in that state. Hence, we could replace E1 with Φ_{inf} (cf. page 106).

Hence, we only have to consider the problem that E may be applied to a conjunction of formulas that start with temporal operators. The next lemma is one of the major steps that are used to handle this problem. It shows that both top-level conjunctions and disjunctions of a temporal operator can be eliminated.

Lemma 5.17 (Eliminating Homogeneous Conjunctions and Disjunctions). *The equations below can be used to eliminate top-level conjunctions or disjunctions of a temporal operator. The conjunction or disjunction is replaced by an equivalent formula that has as top-level operator the corresponding temporal operator. Corresponding theorems also hold for the strong versions of the temporal operators (cf. Lemma 2.48).*

- $\bigwedge_{i=1}^{n} \mathsf{X}\varphi_i = \mathsf{X}\left(\bigwedge_{i=1}^{n} \varphi_i\right)$ *and* $\bigvee_{i=1}^{n} \mathsf{X}\varphi_i = \mathsf{X}\left(\bigvee_{i=1}^{n} \varphi_i\right)$
- $\bigwedge_{i=1}^{n} \mathsf{G}\varphi_i = \mathsf{G}\left(\bigwedge_{i=1}^{n} \varphi_i\right)$ *and* $\bigvee_{i=1}^{n} \mathsf{G}\psi_i = \mathsf{G}\left(\bigwedge_{i=1}^{n} \psi_i \vee \bigvee_{k=1,k\neq i}^{n} \mathsf{G}\psi_k\right)$
- $\bigwedge_{i=1}^{n} \mathsf{F}\psi_i = \mathsf{F}\left(\bigvee_{i=1}^{n} \psi_i \wedge \bigwedge_{k=1,k\neq i}^{n} \mathsf{F}\psi_k\right)$ *and* $\bigvee_{i=1}^{n} \mathsf{F}\varphi_i = \mathsf{F}\left(\bigvee_{i=1}^{n} \varphi_i\right)$
- $\bigwedge_{i=1}^{n} [\varphi_i \,\mathsf{W}\, \psi_i] = \left[\left(\bigvee_{i=1}^{n} \varphi_i \wedge \psi_i \wedge \bigwedge_{k=1,k\neq i}^{n} [\varphi_k \,\mathsf{W}\, \psi_k]\right) \,\mathsf{W}\, \left(\bigvee_{k=1}^{n} \psi_k\right)\right]$
- $\bigvee_{i=1}^{n} [\varphi_i \,\mathsf{W}\, \psi_i] = \left[\left(\bigwedge_{i=1}^{n} \varphi_i \vee \neg\psi_i \vee \bigvee_{k=1,k\neq i}^{n} [\varphi_k \,\mathsf{W}\, \psi_k]\right) \,\mathsf{W}\, \left(\bigvee_{k=1}^{n} \psi_k\right)\right]$
- $\bigwedge_{i=1}^{n} [\varphi_i \,\mathsf{U}\, \psi_i] = \left[\left(\bigwedge_{k=1}^{n} \varphi_k\right) \,\mathsf{U}\, \left(\bigvee_{i=1}^{n} \psi_i \wedge \bigwedge_{k=1,k\neq i}^{n} [\varphi_k \,\mathsf{U}\, \psi_k]\right)\right]$
- $\bigvee_{i=1}^{n} [\varphi_i \,\mathsf{U}\, \psi_i] = \left[\left(\bigwedge_{i=1}^{n} \varphi_i \vee \bigvee_{k=1,k\neq i}^{n} [\varphi_k \,\mathsf{U}\, \psi_k]\right) \,\mathsf{U}\, \left(\bigvee_{k=1}^{n} \psi_k\right)\right]$
- $\bigwedge_{i=1}^{n} [\psi_i \,\mathsf{B}\, \varphi_i] = \left[\left(\bigvee_{i=1}^{n} \psi_i \wedge \bigwedge_{k=1,k\neq i}^{n} [\psi_k \,\mathsf{B}\, \varphi_k]\right) \,\mathsf{B}\, \left(\bigvee_{k=1}^{n} \varphi_k\right)\right]$
- $\bigvee_{i=1}^{n} [\psi_i \,\mathsf{B}\, \varphi_i] = \left[\left(\bigvee_{k=1}^{n} \psi_k\right) \,\mathsf{B}\, \left(\bigvee_{i=1}^{n} \varphi_i \wedge \neg \bigvee_{k=1,k\neq i}^{n} [\psi_k \,\mathsf{B}\, \varphi_k]\right)\right]$

Note that the right hand sides can also be written as conjunctions or disjunctions by applying the rules below (again, corresponding rules do also hold for the strong ver-

sions of the temporal operators). This allows us to replace homogeneous conjunctions by disjunctions and homogeneous disjunctions by conjunctions of the same operator.

$$
\begin{aligned}
[\varphi_1 \mathsf{W}\, \psi] \vee [\varphi_2 \mathsf{W}\, \psi] &= [(\varphi_1 \vee \varphi_2) \mathsf{W}\, \psi] \\
[\varphi_1 \mathsf{W}\, \psi] \wedge [\varphi_2 \mathsf{W}\, \psi] &= [(\varphi_1 \wedge \varphi_2) \mathsf{W}\, \psi] \\
[\varphi \mathsf{U}\, \psi_1] \vee [\varphi \mathsf{U}\, \psi_2] &= [\varphi \mathsf{U}\, (\psi_1 \vee \psi_2)] \\
[\varphi_1 \mathsf{U}\, \psi] \wedge [\varphi_2 \mathsf{U}\, \psi] &= [(\varphi_1 \wedge \varphi_2) \mathsf{U}\, \psi] \\
[\varphi_1 \mathsf{B}\, \psi] \vee [\varphi_2 \mathsf{B}\, \psi] &= [(\varphi_1 \vee \varphi_2) \mathsf{B}\, \psi] \\
[\varphi \mathsf{B}\, \psi_1] \wedge [\varphi \mathsf{B}\, \psi_2] &= [\varphi \mathsf{B}\, (\psi_1 \vee \psi_2)]
\end{aligned}
$$

Proof. The rules for eliminating conjunctions are dual to the rules for eliminating disjunctions, as can be easily seen by negating both sides of the equations and producing negation normal form afterwards. Hence, it is sufficient to prove only the rules for conjunctions. These proofs follow the same scheme that is mainly based on case distinctions on which one of the signals ψ_i will occur first. For example, consider the equation of the W-operator. In the first case, consider the case that none of the ψ_i's will ever hold. Then all W-expressions on the left and on the right hand side evaluate to 1 such that the equation holds. Now, consider the case that at least one of the ψ_i's will hold at least once, and assume that ψ_{i_0} ($i_0 \in \{1, \ldots, n\}$) is the one that holds first and that t_0 is the first point of time where ψ_{i_0} holds. First, the implication from left to right is shown. Clearly, at t_0 both φ_{i_0} and ψ_{i_0} must hold, otherwise $[\varphi_{i_0} \mathsf{W}\, \psi_{i_0}]$ would not hold. Moreover $\bigwedge_{k=1,k\neq i_0}^{n} [\varphi_i \mathsf{W}\, \psi_i]$ must hold since, the other ψ_k's have not occurred so far. The implication from right to left is similarly proved. The subterm $\Phi_{i_0} := \varphi_{i_0} \wedge \psi_{i_0} \wedge \bigwedge_{k=1,k\neq i_0}^{n} [\varphi_k \mathsf{W}\, \psi_k]$ of the disjunction on the right hand side is sufficient for the proof: clearly, $\bigvee_{k=1}^{n} \psi_k$ also holds at t_0 and from Φ_{i_0} it can then be concluded that φ_{i_0} has to hold. Hence, $[\varphi_{i_0} \mathsf{W}\, \psi_{i_0}]$ holds. Moreover, from Φ_{i_0} it can be concluded that $\bigwedge_{k=1,k\neq i_0}^{n} [\varphi_k \mathsf{W}\, \psi_k]$ holds. As none of the events ψ_k with $k \neq i_0$ have occurred before t_0, this is equivalent to the left hand side. □

The complexity of the transformations in the above lemma is however not harmless: For example, the left hand side of the rule for the W-operator has a length of $O(\sum_{i=1}^{n} |\varphi_i| + |\psi_i|)$, but the right hand side has the size $O(n \cdot \sum_{i=1}^{n} |\varphi_i| + |\psi_i|)$. This does not seem to be a problem as only a linear factor n has been added. However, the rules must be applied repeatedly to shift an E quantifier inwards and the right hand side contains n subproblems of the same problem of size $n-1$. Hence, if all conjunctions are to be removed by recursively applying the equations of Lemma 5.17, then a formula of size n results in a formula of size $O(n \cdot n!)$. Better algorithms that keep track of common subformulas are given in Section 5.3.5.

The above rules can be used to combine conjunctions of formulas that start with the same temporal operators to a single formula that starts with that operator. It is also possible to combine *heterogeneous conjunctions*. This can be done by converting the binary operators into one and use the combination laws of the previous lemma. The only additional point is to keep track of the operator strength.

For example, to find a combination law for conjunctions of B and $\underline{\mathsf{U}}$, we first express $[\gamma_j \mathsf{B} \delta_j]$ as $[(\neg\delta_j) \mathsf{U} (\gamma_j \wedge \neg\delta_j)]$, and then use the combination law for $\underline{\mathsf{U}}$. If there are no $\underline{\mathsf{U}}$ operators, we simply obtain (by the combination law for U):

$$\bigwedge_{j=1}^{n} [\gamma_j \mathsf{B} \delta_j] = \left[\left(\bigwedge_{k=1}^{n} \neg\delta_k \right) \mathsf{U} \left(\bigvee_{k \in J} \gamma_k \wedge \neg\delta_k \wedge \bigwedge_{j=1, j\neq k}^{n} [\gamma_j \mathsf{B} \delta_j] \right) \right]$$

Hence, we obtain the following lemma for combining heterogeneous conjunctions:

Lemma 5.18 (Eliminating Heterogeneous Conjunctions). *Given* CTL* *path formulas* $\{\alpha_i \mid i \in I\}$, $\{\beta_i \mid i \in I\}$, $\{\gamma_j \mid j \in J\}$, *and* $\{\delta_j \mid j \in J\}$ *where* $I \neq \{\}$, *the following formula is equivalent to* $\left(\bigwedge_{i \in I} [\alpha_i \underline{\mathsf{U}} \beta_i] \right) \wedge \left(\bigwedge_{j \in J} [\gamma_j \mathsf{B} \delta_j] \right)$, *where we abbreviate* $\Phi_I := \bigwedge_{i \in I} \alpha_i$ *and* $\Psi_J := \bigwedge_{j \in J} \neg\delta_j$:

$$\left[(\Phi_I \wedge \Psi_J) \underline{\mathsf{U}} \left(\begin{array}{l} \bigvee_{k \in I} \beta_k \wedge \left(\bigwedge_{i \in I \setminus \{k\}} [\alpha_i \underline{\mathsf{U}} \beta_i] \right) \wedge \left(\bigwedge_{j \in J} [\gamma_j \mathsf{B} \delta_j] \right) \vee \\ \bigvee_{k \in J} \gamma_k \wedge \neg\delta_k \wedge \left(\bigwedge_{i \in I} [\alpha_i \underline{\mathsf{U}} \beta_i] \right) \wedge \left(\bigwedge_{j \in J \setminus \{k\}} [\gamma_j \mathsf{B} \delta_j] \right) \end{array} \right) \right]$$

In the case $I = \{\}$, *the following equation holds:*

$$\bigwedge_{j \in J} [\gamma_j \mathsf{B} \delta_j] = \left[\Psi_J \mathsf{U} \left(\bigvee_{k \in J} \gamma_k \wedge \neg\delta_k \wedge \bigwedge_{j \in J \setminus \{k\}} [\gamma_j \mathsf{B} \delta_j] \right) \right]$$

Using the above lemma, we will now prove that CTL*, LeftCTL*, CTL$^+$, and UB$^+$ can be translated to CTL^{++}, LeftCTL^{++}, CTL, and UB, respectively. All of these translations are based on the same principle that is given in the above lemmas, namely to make successive case distinctions on the order of the events that are awaited.

A first observation that is used for that transformation is that the logics CTL*, LeftCTL*, CTL$^+$, and UB$^+$ are robust w.r.t. to negation normal form computation, i.e., if the function given in Lemma 2.48 is applied to a formula Φ of one of these logics, it yields a formula NNF(Φ) of the same logic. The same holds for the extensions of these logics by fairness constraints, which is a more general result, which can be seen by choosing one fairness constraint that is simply 1.

Lemma 5.19 (NNF Robustness). *The logics defined in Definition 5.11, as well as their extensions by fairness constraints according to Definition 5.12 are closed with respect to negation normal form computation.*

For the proof, it is sufficient to notice that the rules given in Lemma 2.48 preserve membership in the mentioned logics, which can be straightforwardly done by case distinctions. The above lemma allows us therefore to assume in the following that the given formulas are in negation normal form, i.e., negations can be neglected.

The next step reduces the temporal operators G, F, W, U, $\underline{\mathsf{W}}$, $\underline{\mathsf{B}}$ to $\underline{\mathsf{U}}$ and B. This is straightforwardly done for the logics except for FairLeftCTL^{++}, LeftCTL^{++}, FairLeftCTL* and LeftCTL*. For these logics, the replacement has to be done with some care: The problem is with the W and the $\underline{\mathsf{W}}$ operators. We have two rules for replacing W and the $\underline{\mathsf{W}}$ operators, as the following equations are valid:

$$[\varphi\ \mathsf{W}\ \psi] = [\psi\ \mathsf{B}\ (\neg\varphi \wedge \psi)] \qquad [\varphi\ \mathsf{W}\ \psi] = [(\neg\psi)\ \underline{\mathsf{U}}\ (\varphi \wedge \psi)] \vee [0\ \mathsf{B}\ \psi]$$

However, if $[\varphi\ \mathsf{W}\ \psi]$ is derivable from P_E, then the right hand side of the first equation is no LeftCTL* formula, and if $[\varphi\ \mathsf{W}\ \psi]$ is derivable from P_A, then the right hand side of the second equation is no LeftCTL* formula. Therefore, we have to consider for the replacement of W formulas, whether we have to replace a formula that is derived from the nonterminal P_E or from P_A. This is essentially shown in the next lemma.

Lemma 5.20 (Temporal Bases for FairLeftCTL*). *The following temporal operator bases can be used for* FairLeftCTL**:*

1. *For each* FairLeftCTL* *formula there is an equivalent* FairLeftCTL* *formula in NNF where only the temporal operators* X$\cdot$, $[\cdot\ \underline{\mathsf{U}}\ \cdot]$ *and* $[\cdot\ \mathsf{B}\ \cdot]$ *occur.*
2. *For each* FairLeftCTL* *formula there is an equivalent* FairLeftCTL* *formula in NNF where only the temporal operators* X$\cdot$, $[\cdot\ \mathsf{U}\ \cdot]$ *and* $[\cdot\ \underline{\mathsf{B}}\ \cdot]$ *occur.*

Proof. The first item is proved by the following rules that can be used to replace some grammar rules for P_A and P_E:

$P_A ::= \{[\cdot\ \underline{\mathsf{U}}\ \cdot], [\cdot\ \mathsf{B}\ \cdot]\}$	$P_E ::= \{[\cdot\ \underline{\mathsf{U}}\ \cdot], [\cdot\ \mathsf{B}\ \cdot]\}$
$\mathsf{G}P_A = [0\ \mathsf{B}\ (\neg P_A)]$	$\mathsf{G}S = [0\ \mathsf{B}\ (\neg S)]$
$\mathsf{F}S = [1\ \underline{\mathsf{U}}\ S]$	$\mathsf{F}P_E = [1\ \underline{\mathsf{U}}\ P_E]$
$[P_A\ \mathsf{W}\ S] = [S\ \mathsf{B}\ (\neg P_A \wedge S)]$	$[P_E\ \mathsf{W}\ S] = [(\neg S)\ \underline{\mathsf{U}}\ (P_E \wedge S)] \vee [0\ \mathsf{B}\ S]$
$[P_A\ \underline{\mathsf{W}}\ S] = [S\ \mathsf{B}\ (\neg P_A \wedge S)] \wedge [1\ \underline{\mathsf{U}}\ S]$	$[P_E\ \underline{\mathsf{W}}\ S] = [(\neg S)\ \underline{\mathsf{U}}\ (P_E \wedge S)]$
$[P_A\ \mathsf{U}\ S] = [P_A\ \underline{\mathsf{U}}\ S] \vee [0\ \mathsf{B}\ (\neg P_A)]$	$[S\ \mathsf{U}\ P_E] = [S\ \underline{\mathsf{U}}\ P_E] \vee [0\ \mathsf{B}\ (\neg S)]$
$[S\ \underline{\mathsf{B}}\ P_E] = [S\ \mathsf{B}\ P_E] \wedge [1\ \underline{\mathsf{U}}\ S]$	$[P_E\ \underline{\mathsf{B}}\ S] = [P_E\ \mathsf{B}\ S] \wedge [1\ \underline{\mathsf{U}}\ P_E]$

The second item is proved by the following rules that can be used to replace some grammar rules for P_A and P_E such that only $[\cdot\ \mathsf{U}\ \cdot]$ and $[\cdot\ \underline{\mathsf{B}}\ \cdot]$ occur apart from X:

$P_A ::= \{[\cdot\ \mathsf{U}\ \cdot], [\cdot\ \underline{\mathsf{B}}\ \cdot]\}$	$P_E ::= \{[\cdot\ \mathsf{U}\ \cdot], [\cdot\ \underline{\mathsf{B}}\ \cdot]\}$
$\mathsf{G}P_A = [P_A\ \mathsf{U}\ 0]$	$\mathsf{G}S = [S\ \mathsf{U}\ 0]$
$\mathsf{F}S = [S\ \underline{\mathsf{B}}\ 0]$	$\mathsf{F}P_E = [P_E\ \underline{\mathsf{B}}\ 0]$
$[P_A\ \mathsf{W}\ S] = [S\ \underline{\mathsf{B}}\ (\neg P_A \wedge S)] \vee [(\neg S)\ \mathsf{U}\ 0]$	$[P_E\ \mathsf{W}\ S] = [(\neg S)\ \mathsf{U}\ (P_E \wedge S)]$
$[P_A\ \underline{\mathsf{W}}\ S] = [S\ \underline{\mathsf{B}}\ (\neg P_A \wedge S)]$	$[P_E\ \underline{\mathsf{W}}\ S] = [(\neg S)\ \mathsf{U}\ (P_E \wedge S)] \wedge [S\ \underline{\mathsf{B}}\ 0]$
$[P_A\ \underline{\mathsf{U}}\ S] = [P_A\ \mathsf{U}\ S] \wedge [S\ \underline{\mathsf{B}}\ 0]$	$[S\ \underline{\mathsf{U}}\ P_E] = [S\ \mathsf{U}\ P_E] \wedge [P_E\ \underline{\mathsf{B}}\ 0]$
$[S\ \mathsf{B}\ P_E] = [S\ \underline{\mathsf{B}}\ P_E] \vee [(\neg P_E)\ \mathsf{U}\ 0]$	$[P_E\ \mathsf{B}\ S] = [P_E\ \underline{\mathsf{B}}\ S] \vee [(\neg S)\ \mathsf{U}\ 0]$

Using these rules, every occurrence of the other temporal operators can be eliminated. Note that if the left hand side in the above rules belongs to P_A or P_E, then the right hand side also belongs to P_A or P_E, respectively. As $\neg\,[a\ \underline{\mathsf{U}}\ b] = [(\neg a)\ \mathsf{B}\ b]$, $\neg\,[a\ \mathsf{B}\ b] = [(\neg a)\ \underline{\mathsf{U}}\ b]$, $\neg\,[a\ \mathsf{U}\ b] = [(\neg a)\ \underline{\mathsf{B}}\ b]$, and $\neg\,[a\ \underline{\mathsf{B}}\ b] = [(\neg a)\ \mathsf{U}\ b]$ hold, the negation normal form computation does not reintroduce one of the eliminated operators. □

Using the above lemmas, we can now to prove that CTL*, FairLeftCTL*, LeftCTL*, FairCTL$^+$, and CTL$^+$, are as expressive as the corresponding logics CTL^{++}, FairLeftCTL^{++}, LeftCTL^{++}, FairCTL, and CTL, respectively.

Theorem 5.21. *The logics* CTL*, FairLeftCTL*, LeftCTL*, FairCTL$^+$, *and* CTL$^+$ *can be reduced to the corresponding subsets* CTL^{++}, FairLeftCTL^{++}, LeftCTL^{++}, FairCTL, *and* CTL, *respectively.*

Proof. We use the following replacement laws to remove the temporal operators G·, F·, [· U ·], [· W ·], [· $\underline{\mathsf{W}}$ ·], [· $\underline{\mathsf{B}}$ ·] (in case of FairLeftCTL* and LeftCTL*, we take special care in case of the W and $\underline{\mathsf{W}}$ operators, as given in the previous theorem):

$$\begin{array}{ll}
\mathsf{G}\varphi = [0\ \mathsf{B}\ (\neg\varphi)] & \mathsf{F}\varphi = [1\ \underline{\mathsf{U}}\ \varphi] \\
[\varphi\ \mathsf{U}\ \psi] = [\varphi\ \underline{\mathsf{U}}\ \psi] \vee [0\ \mathsf{B}\ (\neg\varphi)] & [\varphi\ \underline{\mathsf{B}}\ \psi] = [\varphi\ \mathsf{B}\ \psi] \wedge [1\ \underline{\mathsf{U}}\ \varphi] \\
[\varphi\ \mathsf{W}\ \psi] = [(\neg\psi)\ \underline{\mathsf{U}}\ (\varphi \wedge \psi)] \vee [0\ \mathsf{B}\ \psi] & [\varphi\ \underline{\mathsf{W}}\ \psi] = [(\neg\psi)\ \underline{\mathsf{U}}\ (\varphi \wedge \psi)] \\
[\varphi\ \mathsf{W}\ \psi] = [\psi\ \mathsf{B}\ (\neg\varphi \wedge \psi)] & [\varphi\ \underline{\mathsf{W}}\ \psi] = [\psi\ \mathsf{B}\ (\neg\varphi \wedge \psi)] \wedge [1\ \underline{\mathsf{U}}\ \psi]
\end{array}$$

The resulting formula Φ belongs still to the same logic. After this, we compute PQNF(Φ) of the obtained formula Φ, so that *it remains to consider formulas* Eφ *(or* E$_\Phi\varphi$*) of the same logic where* φ *does not contain any path quantifier. For these formulas, we prove the transformation by an induction over the maximal number of nestings of temporal operators in* φ. The base case, where φ is propositional is handled by Lemma 5.16: $\mathsf{E}_\Phi\varphi = \varphi \wedge \mathsf{E}_\Phi 1$. In this case, there are $n+1$ temporal operators in φ, and we compute TDNF(φ) and shift E (or E$_\Phi$) over possibly obtained disjunctions. Hence, we have now obtained a formula of the following form, where for CTL*, LeftCTL*, and CTL$^+$ the fairness constraint Φ is not apparent and the formulas e_j are state formulas (as no path quantifier can occur in e_j, it even follows that e_j is propositional):

$$\mathsf{E}_\Phi\left(\bigwedge_{j=1}^{n_1}[a_j\ \underline{\mathsf{U}}\ b_j] \wedge \bigwedge_{j=1}^{n_2}[c_j\ \mathsf{B}\ d_j] \wedge \bigwedge_{j=1}^{n_3} e_j \wedge \bigwedge_{j=1}^{n_4}\mathsf{X}f_j\right)$$

We now shift the X operator over the conjunction, and abbreviate $e := \bigwedge_{j=1}^{n_3} e_j$ and $f := \bigwedge_{j=1}^{n_4} f_j$. Moreover, we use Lemma 5.16 to obtain the following formula:

$$e \wedge \mathsf{E}_\Phi\left(\bigwedge_{j=1}^{n_1}[a_j\ \underline{\mathsf{U}}\ b_j] \wedge \bigwedge_{j=1}^{n_2}[c_j\ \mathsf{B}\ d_j] \wedge \mathsf{X}f\right)$$

Note that we can still not apply the induction hypothesis, since it may be the case that the maximal nesting of temporal operators is still the same. We now use Lemma 5.18 to convert the subformula $\bigwedge_{j=1}^{n_1} [a_j \,\underline{\mathsf{U}}\, b_j] \wedge \bigwedge_{j=1}^{n_2} [c_j \,\mathsf{B}\, d_j]$ to an equivalent one of the form $\left[\varphi \,\underline{\mathsf{U}}\, \left(\bigvee_{i\in I} \psi_i\right)\right]$ (in case $n_1 > 0$) or of the form $\left[\varphi \,\mathsf{U}\, \left(\bigvee_{i\in I} \psi_i\right)\right]$ (in case $n_1 = 0$ and $n_2 > 0$). In both cases, we proceed in the same manner (we only list one case):

In the case $n_1 > 0$, we have to convert $\mathsf{E}_\Phi\left(\left[\varphi \,\underline{\mathsf{U}}\, \left(\bigvee_{i\in I} \psi_i\right)\right] \wedge \mathsf{X}f\right)$. We use the distributive law $\left[\varphi \,\underline{\mathsf{U}}\, \left(\bigvee_{i\in I} \psi_i\right)\right] = \bigvee_{i\in I} [\varphi \,\underline{\mathsf{U}}\, \psi_i]$ and the recursion law $[\varphi \,\underline{\mathsf{U}}\, \psi_i] = \psi_i \vee \varphi \wedge \mathsf{X}[\varphi \,\underline{\mathsf{U}}\, \psi_i]$, distributive laws of $\wedge$ and $\vee$, and once again Lemma 5.16 to obtain:

$$
\begin{aligned}
&\mathsf{E}_\Phi\left(\left[\varphi \,\underline{\mathsf{U}}\, \left(\bigvee_{i\in I} \psi_i\right)\right] \wedge \mathsf{X}f\right)\\
&\quad\Leftrightarrow \mathsf{E}_\Phi \bigvee_{i\in I} \left([\varphi \,\underline{\mathsf{U}}\, \psi_i] \wedge \mathsf{X}f\right)\\
&\quad\Leftrightarrow \bigvee_{i\in I} \mathsf{E}_\Phi \left([\varphi \,\underline{\mathsf{U}}\, \psi_i] \wedge \mathsf{X}f\right)\\
&\quad\Leftrightarrow \bigvee_{i\in I} \mathsf{E}_\Phi \left(\psi_i \wedge \mathsf{X}f\right) \vee \mathsf{E}_\Phi \left(\varphi \wedge \mathsf{X}\left[[\varphi \,\underline{\mathsf{U}}\, \psi_i] \wedge f\right]\right)
\end{aligned}
$$

Hence, the repeated application of the above transformation steps eliminate each conjunction that appears after an E quantifier. □

We remark here that it is not possible to use the transformation that is used in the above proof for transforming the logic UB^+ to the corresponding logic UB. We can apply similar transformation steps as given in the above proof and obtain then formulas of the form $\mathsf{E}\left(\bigwedge_{j=1}^{n} \mathsf{F}a_j \wedge \mathsf{G}b \wedge \mathsf{X}f\right)$, where a_j, b, and f is propositional. Using Lemma 5.17, we can now replace $\bigwedge_{j=1}^{n_1} \mathsf{F}a_j$ by a disjunction of F formulas, and using distributive laws and, once more, Lemma 5.16 will reduce the problem further to subproblems of the form $\mathsf{E}[\mathsf{F}a \wedge \mathsf{G}b \wedge \mathsf{X}f]$. The problem is now that we can neither combine $\mathsf{F}a \wedge \mathsf{G}b$ to a F nor to a G formula. However, we can reduce it as follows: $\mathsf{F}a \wedge \mathsf{G}b = [b \,\underline{\mathsf{U}}\, (a \wedge \mathsf{G}b)]$, and prove thus that UB^+ can be reduced to CTL which is however subsumed by the above theorem since $\mathsf{UB}^+ \subseteq \mathsf{CTL}^+$ holds.

5.3.4 Adding Path Quantifiers

In the previous section, we have solved one of the problems that distinguishes some temporal logics, by presenting a transformation that allows us to eliminate Boolean operators that occur directly after a path quantifier. The remaining problem is that some logics allow only state formulas as arguments of temporal operators, and not general path formulas. For example, in CTL^{++} each path quantifier precedes a temporal operator, but not vice versa, while in CTL the converse is also true. $\mathsf{LeftCTL}^{++}$ is even more restricted than CTL^{++} since path formulas are only allowed on one side of a temporal binary operator. The following theorem is the background for the definition of the languages $\mathsf{LeftCTL}^{++}$, $\mathsf{LeftCTL}^*$, and $\mathsf{FairLeftCTL}^*$:

Theorem 5.22 (Adding Path Quantifiers). *For all path formulas φ and all state formulas ψ, the following equations hold for any fairness constraint $\Phi := \bigwedge_{i=1}^{n} \mathsf{GF}\alpha_i$ with propositional α_i:*

$$
\begin{array}{ll}
(1) \models \mathsf{E}_\Phi \mathsf{X}\varphi = \mathsf{E}_\Phi \mathsf{X}\mathsf{E}_\Phi\varphi & (2) \models \mathsf{A}_\Phi \mathsf{X}\varphi = \mathsf{A}_\Phi \mathsf{X}\mathsf{A}_\Phi\varphi \\
(3) \models \mathsf{E}_\Phi \mathsf{F}\varphi = \mathsf{E}_\Phi \mathsf{F}\mathsf{E}_\Phi\varphi & (4) \models \mathsf{A}_\Phi \mathsf{G}\varphi = \mathsf{A}_\Phi \mathsf{G}\mathsf{A}_\Phi\varphi \\
(5) \models \mathsf{E}_\Phi [\varphi \mathsf{W} \psi] = \mathsf{E}_\Phi [(\mathsf{E}_\Phi\varphi) \mathsf{W} \psi] & (6) \models \mathsf{A}_\Phi [\varphi \mathsf{W} \psi] = \mathsf{A}_\Phi [(\mathsf{A}_\Phi\varphi) \mathsf{W} \psi] \\
(7) \models \mathsf{E}_\Phi [\varphi \underline{\mathsf{W}} \psi] = \mathsf{E}_\Phi [(\mathsf{E}_\Phi\varphi) \underline{\mathsf{W}} \psi] & (8) \models \mathsf{A}_\Phi [\varphi \underline{\mathsf{W}} \psi] = \mathsf{A}_\Phi [(\mathsf{A}_\Phi\varphi) \underline{\mathsf{W}} \psi] \\
(9) \models \mathsf{E}_\Phi [\psi \mathsf{U} \varphi] = \mathsf{E}_\Phi [\psi \mathsf{U} (\mathsf{E}_\Phi\varphi)] & (10) \models \mathsf{A}_\Phi [\varphi \mathsf{U} \psi] = \mathsf{A}_\Phi [(\mathsf{A}_\Phi\varphi) \mathsf{U} \psi] \\
(11) \models \mathsf{E}_\Phi [\psi \underline{\mathsf{U}} \varphi] = \mathsf{E}_\Phi [\psi \underline{\mathsf{U}} (\mathsf{E}_\Phi\varphi)] & (12) \models \mathsf{A}_\Phi [\varphi \underline{\mathsf{U}} \psi] = \mathsf{A}_\Phi [(\mathsf{A}_\Phi\varphi) \underline{\mathsf{U}} \psi] \\
(13) \models \mathsf{E}_\Phi [\varphi \mathsf{B} \psi] = \mathsf{E}_\Phi [(\mathsf{E}_\Phi\varphi) \mathsf{B} \psi] & (14) \models \mathsf{A}_\Phi [\psi \mathsf{B} \varphi] = \mathsf{A}_\Phi [\psi \mathsf{B} (\mathsf{E}_\Phi\varphi)] \\
(15) \models \mathsf{E}_\Phi [\varphi \underline{\mathsf{B}} \psi] = \mathsf{E}_\Phi [(\mathsf{E}_\Phi\varphi) \underline{\mathsf{B}} \psi] & (16) \models \mathsf{A}_\Phi [\psi \underline{\mathsf{B}} \varphi] = \mathsf{A}_\Phi [\psi \underline{\mathsf{B}} (\mathsf{E}_\Phi\varphi)]
\end{array}
$$

The equations (10) and (12) do also hold if ψ is a path formula.

The proof of the equations is straightforward. In case of binary temporal operators the preceding path quantifier can be shifted to one of the arguments, so that this subformula is converted to a state formula. The logics LeftCTL^{++}, LeftCTL*, and FairLeftCTL* are defined in such a way that the above laws can be applied, i.e., that one of the arguments of the binary temporal operators is a state formula. Hence, together with the transformation of the previous section, the rewriting with the equations of Theorem 5.22 yields in pure CTL or FairCTL formulas.

Note that it is not possible to generalize the above equations. In particular, it is not possible to shift the path quantifier to both arguments. Otherwise we could transform every CTL* formula into an equivalent CTL formula, which is however not possible since CTL* is more expressive than CTL. Also, it is important to see that the condition in Theorem 5.22 that the formula ψ has to be a state formula is necessary. In particular, the following lemma holds:

Lemma 5.23. *In general, the following formulas are not valid:*

$$
\begin{array}{ll}
(1) \not\models \mathsf{E}[(\mathsf{E}\varphi) \mathsf{W} \psi] \rightarrow \mathsf{E}[\varphi \mathsf{W} \psi] & (2) \not\models \mathsf{A}[\varphi \mathsf{W} \psi] \rightarrow \mathsf{A}[(\mathsf{A}\varphi) \mathsf{W} \psi] \\
(3) \not\models \mathsf{E}[(\mathsf{E}\varphi) \underline{\mathsf{W}} \psi] \rightarrow \mathsf{E}[\varphi \underline{\mathsf{W}} \psi] & (4) \not\models \mathsf{A}[\varphi \underline{\mathsf{W}} \psi] \rightarrow \mathsf{A}[(\mathsf{A}\varphi) \underline{\mathsf{W}} \psi] \\
(5) \not\models \mathsf{E}[\varphi \mathsf{U} (\mathsf{E}\psi)] \rightarrow \mathsf{E}[\varphi \mathsf{U} \psi] & (6) \not\models \mathsf{E}[\varphi \underline{\mathsf{U}} (\mathsf{E}\psi)] \rightarrow \mathsf{E}[\varphi \underline{\mathsf{U}} \psi] \\
(7) \not\models \mathsf{EGE}\varphi \rightarrow \mathsf{EG}\varphi & (8) \not\models \mathsf{AF}\varphi \rightarrow \mathsf{AFA}\varphi \\
(9) \not\models \mathsf{E}[\varphi \mathsf{W} (\mathsf{E}\psi)] \rightarrow \mathsf{E}[\varphi \mathsf{W} \psi] & (10) \not\models \mathsf{E}[\varphi \underline{\mathsf{W}} (\mathsf{E}\psi)] \rightarrow \mathsf{E}[\varphi \underline{\mathsf{W}} \psi] \\
(11) \not\models \mathsf{E}[\varphi \mathsf{W} \psi] \rightarrow \mathsf{E}[\varphi \mathsf{W} (\mathsf{E}\psi)] & (12) \not\models \mathsf{E}[\varphi \underline{\mathsf{W}} \psi] \rightarrow \mathsf{E}[\varphi \underline{\mathsf{W}} (\mathsf{E}\psi)]
\end{array}
$$

(11) and (12) do not even hold under the additional restriction that φ is a state formula, but (9) and (10) hold in this case.

Proof. Consider state s_0 in the Kripke structure $\mathcal{K}$ in Figure 5.5. There are exactly two paths starting in state s_0. It is easy to see that on the one hand $(\mathcal{K}, s_0) \not\models \mathsf{E}[(\mathsf{F}a) \mathsf{W} (\mathsf{G}b)]$ holds, but on the other hand we have $(\mathcal{K}, s_0) \models \mathsf{E}[(\mathsf{EF}a) \mathsf{W} (\mathsf{G}b)]$ since $(\mathcal{K}, s_0) \models \mathsf{EF}a$ holds. This is due to the fact that the E path quantifier inside the W-expression allows us to choose the right path

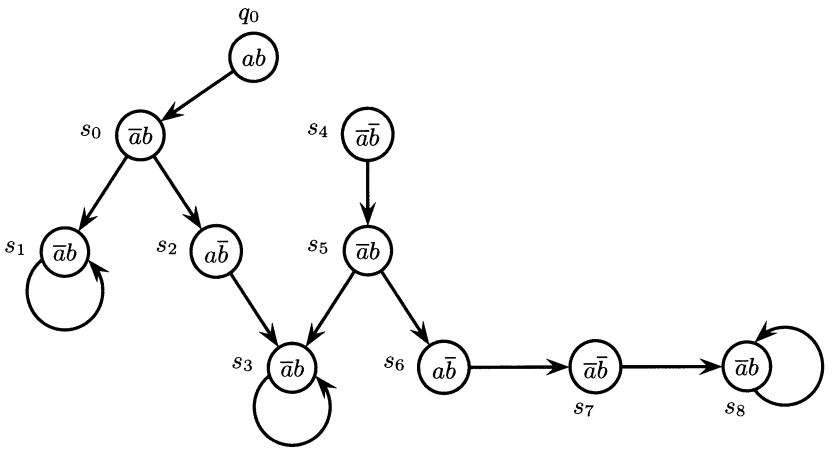


Fig. 5.5. A Kripke structure for disproving some formulas.

in state s_0 while the entire formula holds on the left hand path of s_0. Hence, (1) is disproved. We can disprove (3) by exchanging W by $\underline{\mathsf{W}}$ in the previous argumentation. (2) and (4) can be reduced to (3) and (1), respectively by the identity $\neg\,[\varphi\ \mathsf{W}\ \psi] = [(\neg\varphi)\ \underline{\mathsf{W}}\ \psi]$.

Now, consider state s_4 in Figure 5.5. Again there are two paths starting in state s_4. It can easily be seen that $(\mathcal{K}, s_4) \not\models \mathsf{E}[(\mathsf{F}a)\ \mathsf{U}\ (\mathsf{G}b)]$, but on the other hand $(\mathcal{K}, s_4) \models \mathsf{E}[(\mathsf{F}a)\ \mathsf{U}\ (\mathsf{EG}b)]$. The latter holds since we can choose in state s_5 the left alternative to validate $(\mathcal{K}, s_5) \models \mathsf{EG}b$, while the right alternative is chosen to satisfy the entire formula. Hence, (6) is disproved and the same argumentation also disproves (5).

Clearly, $\mathsf{EG}\varphi \rightarrow \mathsf{EGE}\varphi$ does hold. Hence, if (7) also holds then we could reduce $\mathsf{EGF}a$ to $\mathsf{EGEF}a$. However, this is a CTL formula, which contradicts the fact that no CTL formula is equivalent to $\mathsf{EGF}a$ [162]. (8) can be reduced to (7) by contraposition and pushing negation symbols inwards and outwards.

To disprove (11) and (12) consider state q_0: $(\mathcal{K}, q_0) \models \mathsf{E}[(\neg a)\ \mathsf{W}\ (\mathsf{G}b)]$ holds, since $[(\neg a)\ \mathsf{W}\ (\mathsf{G}b)]$ holds on the path $(q_0, s_0, s_2, s_3, \ldots)$. On the other hand, $(\mathcal{K}, q_0) \not\models \mathsf{E}[(\neg a)\ \mathsf{W}\ (\mathsf{EG}b)]$, since $(\mathcal{K}, q_0) \models \mathsf{EG}b$ holds (because of $(\mathcal{K}, (q_0, s_0, s_1, \ldots) \models \mathsf{G}b)$, but $(\mathcal{K}, q_0) \not\models \neg a)$. The same argumentation disproves (12) when W is exchanged by $\underline{\mathsf{W}}$.

To disprove (9) and (10), note again that $(\mathcal{K}, s_0) \not\models \mathsf{E}[(\mathsf{F}a)\ \mathsf{W}\ (\mathsf{G}b)]$. On the other hand $(\mathcal{K}, s_0) \models \mathsf{E}[(\mathsf{F}a)\ \mathsf{W}\ (\mathsf{EG}b)]$ holds, since $[(\mathsf{F}a)\ \mathsf{W}\ (\mathsf{EG}b)]$ holds along the path $s_0, s_2, s_3, \ldots$ (because $(\mathcal{K}, s_0) \models \mathsf{EG}b$ and $(\mathcal{K}, (s_0, s_2, s_3, \ldots)) \models \mathsf{F}a$). The same argumentation disproves (10) when W is exchanged by $\underline{\mathsf{W}}$. □

As the above formulas do not hold, path quantifiers can not be added on both sides of a binary temporal operator and the restriction that ψ must be a state formula is necessary in Theorem 5.22. It is also not possible to add a

path quantifier on the right side of the W-operator as shown by the formulas (9), (10), (11) and (12).

We have used fairness constraints of the form $\bigwedge_{j=1}^{f} \mathsf{GF}\alpha_i$ in our logics. We can even extend the above results to more general fairness constraints, namely to Boolean combinations of formulas $\mathsf{GF}\alpha_i$ with propositional formulas α_i. In this case, we proceed as follows: Given a formula $\Phi \in \mathsf{FairLeftCTL}^*$, we compute $\mathsf{PQNF}(\Phi)$ (cf. Figure 2.11), so that we only have to consider subformulas of the form $\mathsf{E}_\Phi \Psi$ of that logic where Ψ contains no path quantifiers. We now replace $\mathsf{E}_\Phi \Psi$ by $\mathsf{E}(\Phi \wedge \Psi)$, i.e., no longer treat E_Φ as a special path quantifier, and instead shift the fairness constraint to the quantified formula. We now transform Φ into $\mathsf{TDNF}(\Phi)$ (cf. Figure 2.12), and apply Lemma 5.16, so that we obtain a formula of the following form:

$$\bigvee_{i=0}^{n} \mathsf{E}\left(\bigwedge_{j=0}^{p_i} \mathsf{GF}\varphi_{i,j} \wedge \bigwedge_{j=0}^{q_i} \mathsf{FG}\psi_{i,j} \wedge \Psi \right)$$

Using the law $\bigwedge_{j=0}^{q_i} \mathsf{FG}\psi_{i,j} = \mathsf{FG} \bigwedge_{j=0}^{q_i} \psi_{i,j}$, we combine the negated fairness constraints $\mathsf{FG}\psi_{i,j}$ and obtain the following formula:

$$\bigvee_{i=0}^{n} \mathsf{E}\left(\underbrace{\left(\bigwedge_{j=0}^{p_i} \mathsf{GF}\varphi_{i,j} \right)}_{=:\Phi'} \wedge \underbrace{\left(\mathsf{FG} \bigwedge_{j=0}^{q_i} \psi_{i,j} \wedge \Psi \right)}_{=:\Psi'} \right)$$

With the above abbreviations, we have therefore obtained a formula $\mathsf{E}_{\Phi'}\Psi'$ with fairness constraints Φ' in the normal form $\bigwedge_{j=0}^{p_i} \mathsf{GF}\varphi_{i,j}$. Note now that Ψ' is a P_E formula if and only if Ψ is a P_E formula. Hence, this proves that we can convert any $\mathsf{FairLeftCTL}^*$ formula with general fairness constraints to one with fairness constraints in the mentioned normal form. For these formulas, we can then apply the rules given in Theorem 5.22.

For the logic FairCTL the same reduction can be performed. However, we then obtain a formula $\mathsf{E}_{\Phi'}\Psi'$, where Ψ' does not belong to a legal path formula of FairCTL, since it may start with a conjunction. In the previous section, we have however seen, that these conjunctions can be removed so that we finally end up with a FairCTL formula.

5.3.5 Translating LeftCTL* to Vectorized μ-Calculus

We have already seen how $\mathsf{LeftCTL}^*$ can be reduced to $\mathsf{LeftCTL}^+$ and even to CTL. Hence, we can use CTL model checking tools to construct a model checking tool for $\mathsf{LeftCTL}^*$. However, the complexity of the presented transformation has a big drawback: in each one of the rules of Lemma 5.17, the right hand side contains n subproblems of the same kind of size $n-1$. Therefore, the rules have to be applied $n-1$ times for the formulas on the left hand

sides such that a formula larger than $O(n!)$ is obtained. The combination of the presented transformation procedure with a CTL model checking procedure yields therefore a LeftCTL* model checking procedure whose complexity is also higher than $O(n!)$.

In this section, we therefore consider an optimization of the transformation from LeftCTL* to CTL. In particular, we will prove that the factorial blow-up of the naive transformation can be avoided by sharing subformulas in the generated formulas. The resulting transformation has then a complexity of $O(n2^{2n})$, i.e., it is exponential and therefore a strong improvement on the naive approach. Sharing of subformulas can be 'implemented' by translating to the vectorized μ-calculus instead of translating to CTL. Nevertheless, we will use the CTL operators in the following equation systems, even when we call them μ-calculus formulas.

The outline of this section is as follows: first, we will illustrate the drawback of the naive transformation by an example that leads to a factorial blow-up. After this, we define the new transformation procedure and prove its correctness in detail.

Illustrating Example

Consider the transformation of the LeftCTL* formula $\mathsf{E}\bigwedge_{i=1}^{n}\mathsf{F}a_i$ to an equivalent CTL formula according to the transformations presented so far. The problem is to shift the path quantifier E over the conjunction to the temporal operators. The essential idea of Lemma 5.17 in solving this problem was to make case distinctions on *all possible orders* of the events a_i. The conjunction is thereby turned into a disjunction where all permutations of $\{a_1, \ldots, a_n\}$ have to be considered. Having transformed the conjunction into a disjunction, the preceding E path quantifier can then be shifted inwards and duplicated for each F operator. For this reason, the following formula can be seen as a naive result of the transformation of $\mathsf{E}\bigwedge_{i=1}^{n}\mathsf{F}a_i$, where P_n is the set of all permutations of the numbers $1, \ldots, n$:

$$\mathsf{E}\bigwedge_{i=1}^{n}\mathsf{F}a_i = \bigvee_{\pi\in P_n}\mathsf{EF}\left(a_{\pi(1)} \wedge \mathsf{EF}\left(a_{\pi(2)} \wedge \ldots \wedge \mathsf{EF}\left(a_{\pi(n-1)} \wedge \mathsf{EF}a_{\pi(n)}\right)\ldots\right)\right)$$

It is easily seen that the length of the right hand size is of order $O(n \cdot n!)$. For this reason, the translation procedure from LeftCTL* to CTL has a worst case complexity that is higher than the exponential complexity obtained from corresponding translations to ω-automata (next section). This different complexity is due to the fact that automata benefit from sharing common situations in the form of separate states. We will show now that we can do the same by abbreviating common subformulas. Let us therefore consider where common subformulas arise in the translation procedure. The procedure repeatedly applies the following rule (cf. Lemma 5.17):

$$\bigwedge_{i=1}^{n} \mathsf{F} a_i = \mathsf{F} \bigvee_{i=1}^{n} \left(a_i \wedge \bigwedge_{l=1, l \neq i}^{n} \mathsf{F} a_l \right)$$

After the first application of the rule, n different subproblems of the size $n-1$ are obtained. Multiple rule applications of this rule generate equivalent subformulas. To see this, consider two successive rule applications:

$$\bigwedge_{i=1}^{n} \mathsf{F} a_i = \mathsf{F} \left(\bigvee_{i=1}^{n} a_i \wedge \mathsf{F} \left(\bigvee_{j=1, j \neq i}^{n} a_j \wedge \bigwedge_{l=1, l \notin \{i,j\}}^{n} \mathsf{F} a_l \right) \right)$$

Let us abbreviate for every subset $I \subseteq \{1, \ldots, n\}$ the subformula $\bigwedge_{k=1, k \notin I}^{n} \mathsf{F} a_k$ as φ_I. It is easily seen that each $\varphi_{\{p,q\}}$ occurs twice on the right hand side, for the values $(i,j) = (p,q)$ and $(i,j) = (q,p)$, since the order of the previous events a_i and a_j does not matter for this term. Hence, the interleaving of the previous event history can be neglected; all that matters is which of the events occurred. Sharing the subformula $\varphi_{\{p,q\}}$ would halve the size of the formula.

The procedure has to be recursively applied to the subformulas $\varphi_{\{p,q\}}$. The application to different formulas $\varphi_{\{p,q\}}$ and $\varphi_{\{p,r\}}$ will then generate more common subformulas, namely $\varphi_{\{p,q,r\}}$. Hence, the procedure has to consider common subformulas that are generated in different recursion calls, which makes a recursive treatment rather complicated.

Having seen where the factorial blow-up stems from, we now consider how it can be avoided by sharing *all* common subterms. For this reason, consider first an ω-automaton equivalent to $\bigwedge_{i=1}^{n} \mathsf{F} a_i$. A typical, intuitively obtained automaton uses a hypercube as transition relation. For example, the case $n = 2$ yields the automaton that is given in Figure 5.6.

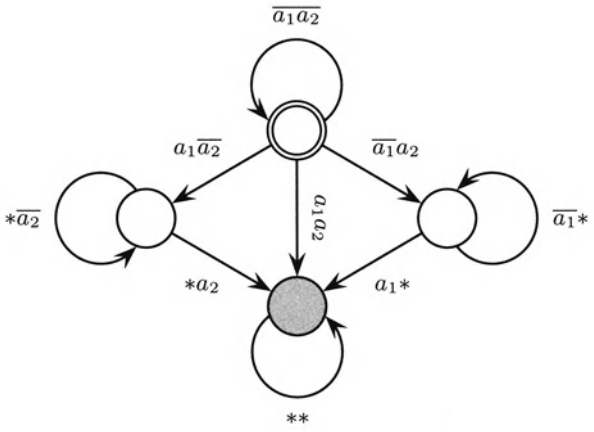


Fig. 5.6. State transition diagram of an automaton waiting concurrently for events a_1 and a_2.

In general, the states s_M of the automaton are associated with the set of events $M \subseteq \{a_1, \ldots, a_n\}$ that are awaited in that state. Hence, the initial state is $s_{\{1,\ldots,n\}}$ (drawn with double lines in Figure 5.6) and the final state is $s_{\{\}}$ (gray-shaded in Figure 5.6). From state s_M, there are transitions to each state s_N with $N \subseteq M$. The transition from s_M to s_N is taken if all events $\{a_k \mid k \in M \setminus N\}$ occur at once. The transitions at state s_M are independent of the events $\{a_k \mid k \in \{1, \ldots, n\} \setminus M\}$, since these events have already occurred before and are therefore no longer of interest. The acceptance condition of the automaton requires that the final state $s_{\{\}}$ must be visited at least once. This is equivalent to visiting it infinitely often and also to staying in it from a certain point of time on, since the only transition leaving from this state leads to the state itself.

Obviously, the automaton for $\bigwedge_{i=1}^{n} \mathsf{F} a_i$ has 2^n states, as the set $\{a_1, \ldots, a_n\}$ has 2^n subsets. Moreover, there are $\binom{n}{k}$ states s_M with $|M| = k$ and from each one of these 2^k transitions are leaving as M has 2^k subsets. Hence, there are $\sum_{k=0}^{n} \binom{n}{k} 2^k = (1+2)^n = 3^n$ edges.

We now represent the automaton in form of a shared temporal formula, where each state s_M of the automaton is represented by a subformula φ_M. In order to share these subformulas, we abbreviate them by a unique variable ϱ_M. We obtain the following shared temporal logic formula for the automaton:

$$\mathsf{E} \bigwedge_{i=1}^{n} \mathsf{F} a_i = \left(\begin{array}{ll} \mathsf{let} & \varrho_{\{\}} := 1 \\ \mathsf{and} & \varrho_M := \bigvee_{i \in M} \mathsf{F}\left(a_i \wedge \varrho_{M \setminus \{i\}}\right) \text{ for } \{\} \neq M \subseteq \{1, \ldots, n\} \\ \mathsf{in} & \mathsf{E} \varrho_{\{1,\ldots,n\}} \\ \mathsf{end} & \end{array} \right)$$

A simple induction proof over $|M|$ shows that $\varrho_M = \bigwedge_{i \in M} \mathsf{F} a_i$ holds (just consider Lemma 5.17). The length of the above formula can be easily computed: the length of the definition of ϱ_M is $O(|M|)$. As there are $\binom{n}{k}$ subsets with k elements of a set with n elements, we obtain for the length $D(n)$ of all definitions and hence the formula itself $D(n) \leq \sum_{k=0}^{n} k\binom{n}{k} = \sum_{k=1}^{n} k\binom{n}{k} = n2^{n-1}$. Therefore, the size of the above shared formula is even less than the size of the automaton that grows with $O(3^n)$. The reason for this is that the F operator includes the present, and therefore we can restrict the 'recursive calls' for ϱ_M to subsets $M \setminus \{i\}$. Hence, we have only $|M|$ recursive calls instead of $2^{|M|}$ (which is the number of transitions leaving a corresponding state in the automaton).

It is clear how the latter formula can be translated to CTL: We can now duplicate the path quantifier E for the inner temporal operators. Hence, the following shared CTL formula is obtained:

$$\mathsf{E}\bigwedge_{i=1}^{n}\mathsf{F}a_i = \left(\begin{array}{l}\mathsf{let}\quad \varrho_{\{\}} := 1\\ \mathsf{and}\ \varrho_M := \bigvee_{i\in M}\mathsf{EF}\left(a_i \wedge \varrho_{M\setminus\{i\}}\right)\ \mathsf{for}\ \{\} \neq M \subseteq \{1,\ldots,n\}\\ \mathsf{in}\quad \varrho_{\{1,\ldots,n\}}\\ \mathsf{end}\end{array}\right)$$

Note that the length is still of order $O(|\Phi|\, 2^{|\Phi|})$ and that our model checking procedure for CTL can directly make use of the shared formula by successively computing the sets of states where ϱ_M holds for increasing sets M. Therefore, we can check the formula $\mathsf{E}\bigwedge_{i=1}^{n}\mathsf{F}a_i$ in time $O(|\mathcal{K}|\, n2^n)$.

Transformation of LeftCTL* to Disjunctive Normal Form

Having seen the essential idea of the use of shared subformulas, we now start with the construction of an exponential-time translation procedure of arbitrary LeftCTL* formulas. We first replace all temporal operators by $\underline{\mathsf{U}}$, B and X according to Theorem 5.20 on page 308. After this, we separate the path quantified subformulas by computing the PQNF normal form by the algorithm given in Figure 2.49 such that, without loss of generality, we can consider in the following, LeftCTL* formulas of the form $\mathsf{E}\Phi$, where Φ contains only the temporal operators $\underline{\mathsf{U}}$, B and X and no path quantifier.

The first step of our transformation is to transform these formulas into a normal form. To explain how this normal form looks, we need some definitions in advance. In particular, we must be able to handle arbitrary deep nestings of $\underline{\mathsf{U}}$ and B. In order to write these nestings in a more concise manner, we use the following definitions for nested operators:

$$\begin{array}{l}[\langle\rangle\ \underline{\mathsf{U}}\ b] := b\\ [\langle a_1,\ldots,a_{p+1}\rangle\ \underline{\mathsf{U}}\ b] := [\langle a_1,\ldots,a_p\rangle\ \underline{\mathsf{U}}\ [a_{p+1}\ \underline{\mathsf{U}}\ b]]\\ [a\ \mathsf{B}\ \langle\rangle] := a\\ [a\ \mathsf{B}\ \langle b_1,\ldots,b_{p+1}\rangle] := [[a\ \mathsf{B}\ \langle b_1,\ldots,b_p\rangle]\ \mathsf{B}\ b_{p+1}]\end{array}$$

This means that $[\langle a_1,\ldots,a_p\rangle\ \underline{\mathsf{U}}\ b]$ abbreviates $[a_1\ \underline{\mathsf{U}}\ [a_2\ \underline{\mathsf{U}}\ \ldots[a_p\ \underline{\mathsf{U}}\ b]\ldots]]$ and $[a\ \mathsf{B}\ \langle b_1,\ldots,b_p\rangle]$ abbreviates $[\ldots[[a\ \mathsf{B}\ b_1]\ \mathsf{B}\ b_2]\ldots\ \mathsf{B}\ b_p]$. For an intuitive understanding of these formulas, note that $(\mathcal{K},\pi) \models [\langle a_1,\ldots,a_p\rangle\ \underline{\mathsf{U}}\ b]$ holds iff the following situation is found on π: let t_p be the first point on π where b holds, i.e., $(\mathcal{K},\pi,t_p) \models \neg b$ holds for $t < t_p$ and $(\mathcal{K},\pi,t_p) \models b$ holds. Then there must be numbers $0 = t_0 \leq t_1 \leq \ldots \leq t_p$ such that $(\mathcal{K},\pi,t) \models a_i$ for $t_{i-1} \leq t < t_i$. This means that the interval $\{t \mid 0 \leq t < t_p\}$ is divided into p intervals where the formulas a_i hold in sequence. Note, however, that some of these intervals may be empty.

$[a\ \mathsf{B}\ \langle b_1,\ldots,b_p\rangle]$ holds on a path π if there are points $t_0 < t_1 < \ldots < t_p$ on the path such that $(\mathcal{K},\pi,t_0) \models a$ and $(\mathcal{K},\pi,t_i) \models b_i$ holds. Alternatively, it may be the case that one of the b_i's never holds on the path. In this case, $[a\ \mathsf{B}\ \langle b_1,\ldots,b_i\rangle]$ trivially holds, and therefore the entire formula also holds.

Neither B nor $\underline{\mathsf{U}}$ is associative, i.e., neither $[[a\ \mathsf{B}\ b]\ \mathsf{B}\ c] = [a\ \mathsf{B}\ [b\ \mathsf{B}\ c]]$ nor $[[a\ \underline{\mathsf{U}}\ b]\ \underline{\mathsf{U}}\ c] = [a\ \underline{\mathsf{U}}\ [b\ \underline{\mathsf{U}}\ c]]$ holds. Therefore the order of the nestings is important. However, we have the following equations which can be easily proved by induction on p:

$$[\langle a_1, \ldots, a_{p+1}\rangle\ \underline{\mathsf{U}}\ b] = [a_1\ \underline{\mathsf{U}}\ [\langle a_2, \ldots, a_{p+1}\rangle\ \underline{\mathsf{U}}\ b]]$$
$$[a\ \mathsf{B}\ \langle b_1, \ldots, b_{p+1}\rangle] = [[a\ \mathsf{B}\ b_1]\ \mathsf{B}\ \langle b_2, \ldots, b_p\rangle]$$

Using the above shorthand notation for nestings, we can now define the following classes of temporal logic formulas:

Definition 5.24 (CUB Formulas). *Given propositional formulas $\alpha_{i,k}$, β_i for $i \in \{1, \ldots, n\}$ and $k \in \{1, \ldots, p_i\}$ and γ_j, $\delta_{j,k}$ for $j \in \{1, \ldots, m\}$ and $k \in \{1, \ldots, q_j\}$, then each* CTL* *path formula of the following form is a CUB formula of order 0:*

$$\left(\bigwedge_{i=1}^{n} [\langle \alpha_{i,1}, \ldots, \alpha_{i,p_i}\rangle\ \underline{\mathsf{U}}\ \beta_i]\right) \wedge \left(\bigwedge_{j=1}^{m} [\gamma_j\ \mathsf{B}\ \langle \delta_{j,1}, \ldots, \delta_{j,q_j}\rangle]\right)$$

CUB formulas of order $n+1$ have the same form, where β_i and γ_j are CUB formulas of order n.

It is easily seen that every CUB formula can be derived from the nonterminal P_E with the grammar rules of LeftCTL*, *hence, for every CUB formula Φ, we have* $\mathsf{E}\Phi \in$ LeftCTL*. *Moreover, the set of CUB formulas is closed with respect to conjunction.* Note that the definition of CUB formulas does not directly allow propositional formulas to occur in the conjunction. However, as $\varphi = [0\ \underline{\mathsf{U}}\ \varphi]$ holds, each propositional formula can be given as a $\underline{\mathsf{U}}$ formula. This means that for convenience, we do not explicitly treat propositional formulas, but embed them as temporal logic formulas of the form $[0\ \underline{\mathsf{U}}\ \varphi]$.

We are now ready to explain the normal form that is established in this section. It handles the propositional logic part of the LeftCTL* formula by transforming it into a disjunctive normal form similar to the one given in Figure 2.12 on page 88. The difference is however, that also right hand arguments of $\underline{\mathsf{U}}$ and left hand arguments of B are recursively transformed into this normal form. The algorithm is given in Figure 5.7; it does not however apply explicitly associative laws for $\wedge$ and $\vee$ (in the cases for $\varphi \wedge \psi$ and $\varphi \vee \psi$, respectively) which is also necessary.

Note that in the cases for $[\varphi\ \underline{\mathsf{U}}\ \psi]$ and $[\varphi\ \mathsf{B}\ \psi]$, the subformula φ is copied in the above algorithm for each minterm ψ_j. By the grammar of LeftCTL*, φ must be a state formula, and by our assumptions that no path quantifier must occur, it follows that φ is propositional. Hence, for our translation from LeftCTL* to CTL we can leave this formula unchanged. For this reason, is is not necessary to copy this formula for each factor, instead we should abbreviate φ by a new variable ϑ and copy ϑ instead of φ. Implementations of our translation procedure need to consider this, but the asymptotic complexities that are proved in this section are not influenced by this fact.

function DeepDNF(Φ)
  (* Φ *should be in negation normal form* *)
  case Φ **of**
    is_var(Φ): **return** Φ;
    $\neg\varphi$: **return** $\neg\varphi$; (* *since* $\varphi \in V_\Sigma$ *)
    $\varphi \wedge \psi$: $\bigvee_{i\in I} \varphi_i \equiv$ DeepDNF(φ);
      $\bigvee_{j\in J} \psi_j \equiv$ DeepDNF(ψ);
      : **return** $\bigvee_{i\in I}\bigvee_{j\in J} \varphi_i \wedge \psi_j$;
    $\varphi \vee \psi$: **return** DeepDNF(φ) $\vee$ DeepDNF(ψ);
    $\mathsf{X}\varphi$: $\bigvee_{i\in I} \varphi_i \equiv$ DeepDNF(φ); **return** $\bigvee_{i\in I} \mathsf{X}\varphi_i$;
    $[\varphi \,\underline{\mathsf{U}}\, \psi]$: $\bigvee_{j\in J} \psi_j \equiv$ DeepDNF(ψ); **return** $\bigvee_{j\in J} [\varphi \,\underline{\mathsf{U}}\, \psi_j]$;
    $[\varphi \,\mathsf{B}\, \psi]$: $\bigvee_{i\in I} \varphi_i \equiv$ DeepDNF(φ); **return** $\bigvee_{i\in I} [\varphi_i \,\mathsf{B}\, \psi]$;
  end
end

Fig. 5.7. Transformation into temporal disjunctive normal form

Theorem 5.25 (DeepDNF**).** *Given any* LeftCTL* *formula* $\mathsf{E}\Phi$, *where* Φ *contains no path quantifiers. There is an equivalent* LeftCTL* *formula* $\mathsf{E}\Phi_\vee$ *such that* $\Phi_\vee := \bigvee_{k\in M} \Phi_k$ *where* $\Phi_k := \bigwedge_{j=0}^{m_k} \mathsf{X}^k\varphi_{j,k}$ *with CUB formulas* $\varphi_{j,k}$. *Moreover,* $|\Phi_\vee| \in O(|\Phi|\, 2^{|\Phi|})$, *and in particular* $|M| \in O(2^{|\Phi|})$ *and* $|\Phi_k| \in O(|\Phi|)$ *hold.*

The proof of the above theorem is done by induction on Φ along the algorithm given in Figure 5.7. The correctness is clear, the only interesting laws that are used are $[a \,\underline{\mathsf{U}}\, (b_1 \vee b_2)] = [a \,\underline{\mathsf{U}}\, b_1] \vee [a \,\underline{\mathsf{U}}\, b_2]$ and $[(a_1 \vee a_2) \,\mathsf{B}\, b] = [a_1 \,\mathsf{B}\, b] \vee [a_2 \,\mathsf{B}\, b]$ and the distributivity of X over $\wedge$ and $\vee$. Note that also $[(a_1 \wedge a_2) \,\underline{\mathsf{U}}\, b] = [a_1 \,\underline{\mathsf{U}}\, b] \wedge [a_2 \,\underline{\mathsf{U}}\, b]$ and $[a \,\underline{\mathsf{B}}\, (b_1 \vee b_2)] = [a \,\underline{\mathsf{B}}\, b_1] \wedge [a \,\underline{\mathsf{B}}\, b_2]$ hold, but these laws are not used in Figure 5.7. The reason for not using the latter ones is that the conjunctions that would thereby be split are propositional formulas. Hence, our transformations would not benefit from these additional expansions.

Now, consider the lengths of the formulas. All cases apart from $\Phi \equiv \varphi \wedge \psi$ are non-critical, hence, we only consider this one here. By the induction hypothesis, we can compute DeepDNF(φ) $= \bigvee_{i\in K_1} \varphi_i$ and DeepDNF(ψ) $= \bigvee_{j\in K_2} \psi_j$ where $|K_1| \in O(2^{|\varphi|})$, $|\varphi_i| \in O(|\varphi|)$, $|K_2| \in O(2^{|\psi|})$, and $|\psi_j| \in O(|\psi|)$ holds. The result of the algorithm given in Figure 5.7 is then the formula $\bigvee_{i\in K_1}\bigvee_{j\in K_2} \varphi_i \wedge \psi_j$, which consists of $O(2^{|\varphi|}2^{|\psi|}) = O(2^{|\Phi|})$ factors of lengths $O(|\varphi| + |\psi|) = O(\Phi)$.

Having computed DeepDNF(Φ) for a given formula $\mathsf{E}\Phi$, we can shift the path quantifier inwards such that in the following, we only need to translate LeftCTL* formulas of the form $\mathsf{E}\bigwedge_{k=0}^{\ell} \mathsf{X}^k\Phi_k$ to CUB formulas Φ_k. In the next section, we will show, how X-free CUB formulas can be translated which means that we will handle the case $\ell = 0$. The results are then extended to arbitrary ℓ in the final subsection of this section.

Transformation of X-Free LeftCTL*

In this section, we show how CUB formulas of arbitrary order can be translated into a shared form such that an outermost E path quantifier could be shifted inwards without problem, i.e., whenever it is applied to a conjunction, then one of the arguments of the conjunction is a propositional formula. Hence, we can use the law $\mathsf{E}(\varphi \wedge \psi) = \varphi \wedge \mathsf{E}\psi$ (where φ is propositional). The essential lemma of this transformation is again the elimination of conjunctions that are quantified by an E path quantifier as given by Lemma 5.18.

Lemma 5.18 is essential for the transformation of CUB formulas. It corresponds to the rule $\bigwedge_{i=1}^{n} \mathsf{F}a_i = \mathsf{F}\left(\bigvee_{i=1}^{n} a_i \wedge \bigwedge_{l=1, l\neq i}^{n} \mathsf{F}a_l\right)$ we need for transforming the illustrative example. It is also easily seen that on the right hand side of the equations common subformulas occur. It is now important to describe how all of these common subterms can be collected. For this reason, first consider the transformation of a CUB formula of order 0, i.e., formulas of the form $(\bigwedge_{i=1}^{n} [\langle \alpha_{i,1}, \ldots, \alpha_{i,p_i} \rangle \mathbin{\underline{\mathsf{U}}} \beta_i]) \wedge \left(\bigwedge_{j=1}^{m} [\gamma_j \mathbin{\mathsf{B}} \langle \delta_{j,1}, \ldots, \delta_{j,q_j} \rangle]\right)$, where $\alpha_{i,k}$, β_i, γ_j, and $\delta_{j,k}$ are propositional.

In general, the problem is that several events are concurrently awaited, and we do not know which of them will be the first to occur. The solution is therefore based on making successive case distinctions on which of these events is the first one (or among the first ones) to occur. Up to the point of time where one of the events occurs, some other condition must hold. Additionally, we must deal with the fact that when one event occurs, it generates another event that is then awaited. The dependencies of the events that are awaited in the above formula are given in Figure 5.8.

Initially, the events of the first row in Figure 5.8 are awaited and until one of them occurs $\bigwedge_{i=1}^{n} \alpha_{i,1}$ and $\bigwedge_{j=1}^{m} \neg\delta_{j,q_j}$ must hold. If one of the events listed in the first row occurs, then this event is replaced by its successor event, i.e., the event in the same column directly below it. If more than one event occurs at the same point of time, then these are all replaced by their successor events. If finally, b_k or c_k occur, then the 'event family' of column k is released and only the events of the other columns are awaited. Hence, choosing from each column in Figure 5.8, at most one node leads to a situation where the events of that nodes are concurrently awaited.

Figure 5.8 precisely specifies the order in which the events are awaited: Events on the same column must occur in the order given by the column, while events on different columns may occur in an arbitrary order. The aim is now to represent the situations along with their dependencies as given in Figure 5.8 for arbitrary CUB formulas. For this reason, we must first find some representation of the 'event structure' of these formulas and represent the shared formula directly with the event structure. The extension of Figure 5.8 to an event structure for a CUB formula of order $k \geq 0$ is formally defined as follows (recall that PF_Σ is the set of path formulas):

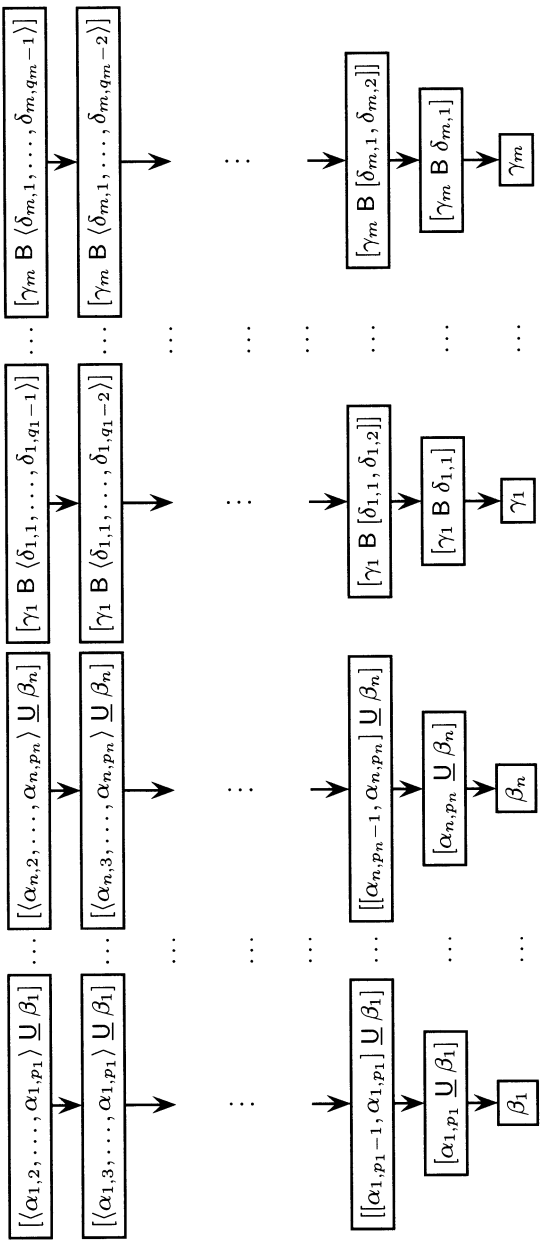


Fig. 5.8. Event structure for CUB formulas

Definition 5.26 (Event Structure of a CUB Formula). *An event structure for a CUB formula is a tuple $(\mathcal{I}, \mathcal{B}, \mathcal{S}, \mathcal{R}, \iota, \epsilon)$, where $\mathcal{S}$ is a finite set of states, $\mathcal{I} \subseteq \mathcal{S}$ is the set of initial states, and $\mathcal{R} \subseteq \mathcal{S} \times \mathcal{S}$ is the set of edges that is an acyclic relation. $\iota : \mathcal{S} \to \mathsf{PF}_\Sigma$ maps each state $s \in \mathcal{S}$ to a propositional formula $\iota(s)$ that is called the invariant of s. $\epsilon : \mathcal{S} \to \mathsf{PF}_\Sigma$ maps each state $s \in \mathcal{S}$ to a CUB formula $\epsilon(s)$ that is called the event of s. Finally, $\mathcal{B} \subseteq \mathcal{S}$ is the set of* B*-states of the structure. The event structure* $\mathsf{EVS}(\Phi)$ *for a particular CUB formula Φ is defined by the algorithm in Figure 5.9.*

function $\mathsf{EVS}(\Phi)$
  case Φ **of**
   is_prop(Φ):
      return $(\{\}, \{\}, \{\}, \{\}, \{\}, \{\})$;
   $[\langle \alpha_1, \ldots, \alpha_p \rangle \,\underline{\mathsf{U}}\, \varphi]$:
      $(\mathcal{I}_1, \mathcal{B}_1, \mathcal{S}_1, \mathcal{R}_1, \iota_1, \epsilon_1) := \mathsf{EVS}(\varphi)$;
      $\mathcal{S} := \mathcal{S}_1 \cup \{s_1, \ldots, s_p\}$;
      $\mathcal{R}_2 := \mathcal{R}_1 \cup \{(s_i, s_{i+1}) \mid i \in \{1, \ldots, p-1\}\}$;
      $\mathcal{R} := \mathcal{R}_2 \cup \{(s_p, q) \mid q \in \mathcal{I}_1\}$;
      $\iota := \iota_1 \cup \{(s_i, \alpha_i) \mid i \in \{1, \ldots, p\}\}$;
      $\epsilon := \epsilon_1 \cup \{(s_i, [\langle \alpha_{i+1}, \ldots, \alpha_p \rangle \,\underline{\mathsf{U}}\, \varphi]) \mid i \in \{1, \ldots, p\}\}$;
      return $(\{s_1\}, \mathcal{B}_1, \mathcal{S}, \mathcal{R}, \iota, \epsilon)$;
   $[\varphi \;\mathsf{B}\; \langle \delta_1, \ldots, \delta_p \rangle]$:
      $(\mathcal{I}_1, \mathcal{B}_1, \mathcal{S}_1, \mathcal{R}_1, \iota_1, \epsilon_1) := \mathsf{EVS}(\varphi)$;
      $\mathcal{B} := \mathcal{B}_1 \cup \{s_1, \ldots, s_p\}$;
      $\mathcal{S} := \mathcal{S}_1 \cup \{s_1, \ldots, s_p\}$;
      $\mathcal{R}_2 := \mathcal{R}_1 \cup \{(s_i, s_{i+1}) \mid i \in \{1, \ldots, p-1\}\}$;
      $\mathcal{R} := \mathcal{R}_2 \cup \{(s_p, q) \mid q \in \mathcal{I}_1\}$;
      $\iota := \iota_1 \cup \{(s_i, \neg\delta_{p-i+1}) \mid i \in \{1, \ldots, p\}\}$;
      $\epsilon := \epsilon_1 \cup \{(s_i, \neg\delta_{p-i+1} \wedge [\varphi \;\mathsf{B}\; \langle \delta_1, \ldots, \delta_{p-i} \rangle]) \mid i \in \{1, \ldots, p\}\}$;
      return $(\{s_1\}, \mathcal{B}, \mathcal{S}, \mathcal{R}, \iota, \epsilon)$;
   $\bigwedge_{i \in M} \varphi_i$:
      $(\mathcal{I}_i, \mathcal{B}_i, \mathcal{S}_i, \iota_i, \epsilon_i) := \mathsf{EVS}(\varphi_i)$;
      return $(\bigcup_{i \in M} \mathcal{I}_i, \bigcup_{i \in M} \mathcal{B}_i, \bigcup_{i \in M} \mathcal{S}_i, \bigcup_{i \in M} \iota_i, \bigcup_{i \in M} \epsilon_i)$;
 end
end

Fig. 5.9. Computation of the event structure of a CUB formula

The size of $\mathsf{EVS}(\Phi)$ *is linear in terms of $|\Phi|$, i.e. in particular $|\mathcal{S}|, |\mathcal{R}| \in O(|\Phi|)$.* In general the intention is that the event structure formalizes a *partial order* on the events that occur in a formula: whenever we know from the structure of the formula that an event e_1 is awaited after another event e_2 then there is a path through the event structure from a state s_1 to a state s_2 such that $\epsilon(s_1) = e_1$ and $\epsilon(s_2) = e_2$ holds. Hence, $\epsilon(s)$ is the event that is awaited when the structure is in state s. Until this event occurs, the formula $\iota(s)$ must hold.

The event structures of $[\langle \alpha_1, \ldots, \alpha_p \rangle \underline{\mathsf{U}} \varphi]$ and $[\varphi \,\mathsf{B}\, \langle \delta_1, \ldots, \delta_p \rangle]$ are obtained from the event structure of φ by adding new states $s_1, \ldots, s_p$ that are connected as a chain. s_1 is the new initial state and s_p is connected with each initial state of the event structure of φ. *Therefore, it is easily seen that the event structure of a CUB formula is a finite set of trees.* Figure 5.8 shows the event structure of a CUB formula of order 0, where each state is labeled with its event $\epsilon(s)$.

Each tree of the event structure of a CUB formula stems from a top-level formula of the form $[\langle \alpha_1, \ldots, \alpha_p \rangle \underline{\mathsf{U}} \varphi]$ or $[\varphi \,\mathsf{B}\, \langle \delta_1, \ldots, \delta_p \rangle]$ respectively. As the events in these formulas are, in general, independent of each other, we must consider all possible combinations of these basic events to form global events. For this reason, we need the following definition:

Definition 5.27 (Consistent Situations of an Event Structure). *Let $\mathcal{E} = (\mathcal{I}, \mathcal{B}, \mathcal{S}, \mathcal{R}, \iota, \epsilon)$ be an event structure. The set of consistent situations of $\mathcal{E}$ is defined as the least set that satisfies the following properties:*

1. *$\mathcal{I}$ is a consistent situation*
2. *if χ is a consistent situation, and $s \in \chi$, then $(\chi \setminus \{s\}) \cup \mathsf{suc}^{\mathcal{R}}_{\exists}(\{s\})$ is a consistent situation*

The set of consistent situations of $\mathcal{E}$ is denoted by $\mathfrak{S}(\mathcal{E})$.

The set of consistent situations $\mathfrak{S}(\mathcal{E})$ are those subsets of $\mathcal{S}$ that are obtained from the initial situation by replacing a state by its successor states. *It has to be noted, that switching from one situation to its successor situation does not necessarily need to consume time. This means that even in the first instant of time, each consistent situation may occur.* In the previous lemma, the consistent situations have been encoded by the tuples $(i_1, \ldots, i_n)$ and $(j_1, \ldots, j_m)$, where the event structure has been represented by incrementing one component of the tuples. As the consistent situations correspond to horizontal cuts through the event structure, the set of consistent situations can also be characterized as given in the following lemma:

Lemma 5.28 (Consistent Situations of an Event Structure). *Let $\mathcal{E} = (\mathcal{I}, \mathcal{B}, \mathcal{S}, \mathcal{R}, \iota, \epsilon)$ be an event structure and let $\prec$ be the partial order induced by $\mathcal{E}$ (i.e., the reflexive and transitive closure of $\mathcal{R}$). A set of states $\chi \subseteq \mathcal{S}$ is consistent iff $\forall s_1, s_2 \in \chi.\ s_1 \not\prec s_2 \wedge s_2 \not\prec s_1$.*

The consistency condition means that a consistent situation must not contain two states that lie on the same path of the structure. On the other hand, it is not required that a consistent situation must be complete, i.e., it need not necessarily contain states of all path of the structure. To determine an upper bound for the number of consistent situations of the event structure $\mathsf{EVS}(\Phi)$ of a CUB formula, we prove the following lemma:

Lemma 5.29. *Given a CUB formula Φ of an arbitrary order k and its event structure $\mathsf{EVS}(\Phi) = (\mathcal{I}, \mathcal{B}, \mathcal{S}, \mathcal{R}, \iota, \epsilon)$. There are at most $2^{|\Phi|}$ consistent situations of $\mathsf{EVS}(\Phi)$. Moreover, for each $\chi \in \mathfrak{S}(\mathcal{E})$ the inequation $|\chi| \leq |\Phi|$ holds.*

The proof is done inductively on the structure of Φ. The proof is easy, once it is seen that the following equations hold:

- $|\mathfrak{S}(\mathsf{EVS}(b))| = 1$ for each propositional formula b
- $|\mathfrak{S}(\mathsf{EVS}([\langle\alpha_1, \ldots, \alpha_p\rangle \mathsf{U} \varphi]))| = p + |\mathfrak{S}(\mathsf{EVS}(\varphi))|$
- $|\mathfrak{S}(\mathsf{EVS}([\varphi \mathsf{B} \langle\delta_1, \ldots, \delta_p\rangle]))| = p + |\mathfrak{S}(\mathsf{EVS}(\varphi))|$
- $\left|\mathfrak{S}(\mathsf{EVS}(\bigwedge_{\imath\in M} \varphi_\imath))\right| = \prod_{\imath\in M} |\mathfrak{S}(\mathsf{EVS}(\varphi_\imath))|$

Moreover, if we define $m(\mathcal{E}) := \max\{|\chi| \mid \chi \in \mathfrak{S}(\mathcal{E})\}$ as the least upper bound of the size of the consistent situations, then it is also easily seen that the following equations hold:

- $m(\mathsf{EVS}(b)) = 0$ for each propositional formula b
- $m(\mathsf{EVS}([\langle\alpha_1, \ldots, \alpha_p\rangle \mathsf{U} \varphi])) = m(\mathsf{EVS}(\varphi))$
- $m(\mathsf{EVS}([\varphi \mathsf{B} \langle\delta_1, \ldots, \delta_p\rangle])) = m(\mathsf{EVS}(\varphi))$
- $m(\mathsf{EVS}(\bigwedge_{\imath\in M} \varphi_\imath)) = \sum_{\imath\in M} m(\mathsf{EVS}(\varphi_\imath))$

The remaining proof is then done by induction on the order of the CUB formula.

We now show the translation of $\mathsf{E}\Phi$ for a CUB formula Φ to an equivalent alternation-free μ-calculus formula. Based on the construction of the event structure $\mathsf{EVS}(\Phi)$, we proceed as follows:

Theorem 5.30 (Translating CUB to Vectorized μ-Calculus). *Given a CUB formula Φ of an arbitrary order k and its event structure $\mathsf{EVS}(\Phi) = (\mathcal{I}, \mathcal{B}, \mathcal{S}, \mathcal{R}, \iota, \epsilon)$. We define for each consistent situation χ an equation to define a variable ϱ_χ as follows:*

$$\varrho_\chi := \begin{cases} \left[\left(\bigwedge_{s\in\chi} \iota(s)\right) \mathsf{U} \left(\bigvee_{s\in\chi} \xi_\chi(s)\right)\right] & : \text{if } \chi \not\subseteq \mathcal{B} \\ \left[\left(\bigwedge_{s\in\chi} \iota(s)\right) \mathsf{U} \left(\bigvee_{s\in\chi} \xi_\chi(s)\right)\right] & : \text{if } \chi \subseteq \mathcal{B} \end{cases}$$

$$\text{where } \xi_\chi(s) := \begin{cases} \varrho_{(\chi\setminus\{s\})\cup \mathsf{suc}^{\mathcal{R}}_{\exists}(\{s\})} & : \text{if } \mathsf{suc}^{\mathcal{R}}_{\exists}(\{s\}) \neq \{\} \\ \epsilon(s) \wedge \varrho_{\chi\setminus\{s\}} & : \text{otherwise} \end{cases}$$

Under these recursive definitions, Φ is equivalent to $\varrho_{\mathcal{I}}$. Hence, for each CUB formula Φ there is an alternation-free μ-calculus formula of size $O(|\Phi|\, 2^{|\Phi|})$ that is equivalent to $\mathsf{E}\Phi$.

Proof. It can be easily seen that all situations χ that are used as indices are consistent situations. First of all, one has to note that for each CUB formula Ψ with $\mathsf{EVS}(\Psi) = (\mathcal{I}, \mathcal{B}, \mathcal{S}, \mathcal{R}, \iota, \epsilon)$, we have

$$(*)\quad \Psi = \left(\bigwedge_{s\in\mathcal{I}\setminus\mathcal{B}} [\iota(s) \mathsf{U} \epsilon(s)]\right) \wedge \left(\bigwedge_{s\in\mathcal{I}\cap\mathcal{B}} [\iota(s) \mathsf{U} \epsilon(s)]\right)$$

This is easily seen since for each U formula φ_i new states $s_{\imath,1}, \ldots, s_{\imath,p_1}$ are introduced where $\varphi_\imath = [\iota(s_{\imath,1}) \mathsf{U} \epsilon(s_{\imath,1})]$. Similarly, for each B formula ψ_j new

states $s_{\jmath,1}, \ldots, s_{\jmath,q_\jmath}$ are introduced with $\psi_\jmath = [\iota(s_{\jmath,1}) \mathsf{U} \epsilon(s_{\jmath,1})]$. As $\mathcal{I}$ consists exactly of the states $s_{\imath,1}$ and $s_{\jmath,1}$, the above equation holds.

Now, we can prove that for each event structure computed by the algorithm given in Figure 5.9 for a CUB formula, the following holds[2]:

$$(**)\ \forall s \in \mathcal{S}.\ \mathsf{suc}^{\mathcal{R}}_{\exists}(\{s\}) \neq \{\} \rightarrow \left[\epsilon(s) = \left(\bigwedge_{t \in \mathsf{suc}^{\mathcal{R}}_{\exists}(\{s\}) \setminus \mathcal{B}} [\iota(t) \underline{\mathsf{U}} \epsilon(t)]\right) \wedge \left(\bigwedge_{t \in \mathsf{suc}^{\mathcal{R}}_{\exists}(\{s\}) \cap \mathcal{B}} [\iota(t) \mathsf{U} \epsilon(t)]\right)\right]$$

To see that it holds, note that states are introduced in the algorithm for computing $\mathsf{EVS}(\Phi)$ in Figure 5.9 only for formulas that start with a temporal operator $\underline{\mathsf{U}}$ or B. Assume that the states $s_1, \ldots, s_p$ are introduced for $[\langle \alpha_1, \ldots, \alpha_p \rangle \underline{\mathsf{U}} \varphi]$. Then we have by definition of Figure 5.9 for $i \in \{1, \ldots, p\}$ that $\iota(s_i) := \alpha_i$, $\epsilon(s_i) := [\langle \alpha_{i+1}, \ldots, \alpha_p \rangle \underline{\mathsf{U}} \varphi]$ holds. Moreover, for $1 \leq i < p$, we have $\mathsf{suc}^{\mathcal{R}}_{\exists}(\{s_i\}) := \{s_{i+1}\}$, and hence, $\epsilon(s_i) = [\iota(s_{i+1}) \underline{\mathsf{U}} \epsilon(s_{i+1})]$ such that $(**)$ holds for $1 \leq i < p$. For s_p, we need equation $(*)$ above: firstly, we have $\mathsf{suc}^{\mathcal{R}}_{\exists}(\{s_p\}) := \mathcal{I}_\varphi$, where $\mathcal{I}_\varphi$ are the initial states of $\mathsf{EVS}(\varphi)$. The rest follows as $\epsilon(s_p) := \varphi$, hence by $(*)$, we have $\epsilon(s_p) = \bigwedge_{s \in \mathcal{I}_\varphi \setminus \mathcal{B}_\varphi} [\iota(s) \underline{\mathsf{U}} \epsilon(s)] \wedge \bigwedge_{s \in \mathcal{I}_\varphi \cap \mathcal{B}_\varphi} [\iota(s) \mathsf{U} \epsilon(s)]$.

Having seen that $(**)$ holds, we can now use it for proving the following equation by induction on the order of the consistent situations χ:

$$(***)\ \varrho_\chi = \bigwedge_{s \in \chi \setminus \mathcal{B}} [\iota(s) \underline{\mathsf{U}} \epsilon(s)] \wedge \bigwedge_{s \in \chi \cap \mathcal{B}} [\iota(s) \mathsf{U} \epsilon(s)]$$

The induction base is clear since for the maximal situation $\{\}$, we have $\varrho_{\{\}} = [1 \mathsf{U} 0] = 1$. In the induction step, we first consider the case $\chi \not\subseteq \mathcal{B}$, such that we can apply the first equation of Lemma 5.18 to transform $(***)$ as follows: $\varrho_\chi = \left[\left(\bigwedge_{s \in \chi} \iota(s)\right) \underline{\mathsf{U}} \left(\bigvee_{s \in \chi} \epsilon(s) \wedge \varrho_{\chi \setminus \{s\}}\right)\right]$. In case $\chi \subseteq \mathcal{B}$, we use the second equation of Lemma 5.18 to obtain the same proof goal where $\underline{\mathsf{U}}$ is replaced with U. In each case, it therefore remains to show that for every state $s \in \chi$ of any consistent situation χ the equation $\xi_\chi(s) = \epsilon(s) \wedge \varrho_{\chi \setminus \{s\}}$ holds. For the case $\mathsf{suc}^{\mathcal{R}}_{\exists}(\{s\}) = \{\}$, this holds by definition of $\xi_\chi(s)$. In case $\mathsf{suc}^{\mathcal{R}}_{\exists}(\{s\}) \neq \{\}$, we argue as follows (where we need the induction hypothesis):

$$\begin{aligned}
\xi_\chi(s) &\stackrel{def}{=} \varrho_{(\chi \setminus \{s\}) \cup \mathsf{suc}^{\mathcal{R}}_{\exists}(\{s\})} \\
&\stackrel{(***)}{=} \textstyle\bigwedge_{s \neq t \in \chi \setminus \mathcal{B}} [\iota(t) \underline{\mathsf{U}} \epsilon(t)] \wedge \bigwedge_{s \neq t \in \chi \cap \mathcal{B}} [\iota(t) \mathsf{U} \epsilon(t)] \wedge \\
&\quad \textstyle\bigwedge_{t \in \mathsf{suc}^{\mathcal{R}}_{\exists}(\{s\}) \setminus \mathcal{B}} [\iota(t) \underline{\mathsf{U}} \epsilon(t)] \wedge \bigwedge_{t \in \mathsf{suc}^{\mathcal{R}}_{\exists}(\{s\}) \cap \mathcal{B}} [\iota(t) \mathsf{U} \epsilon(t)] \\
&\stackrel{(**)}{=} \varrho_{\chi \setminus \{s\}} \wedge \epsilon(s)
\end{aligned}$$

□

[2] Note that by construction of $\mathsf{EVS}(\Phi)$, we have that $\mathsf{suc}^{\mathcal{R}}_{\exists}(\{s\}) = \{\}$ holds iff Φ is propositional. However, we only need to prove $(**)$.

The construction of the definitions ϱ_χ implies that in the situation χ, all formulas $\iota(s)$ for $s \in \chi$ must hold until one of the events $\bigvee_{s\in\chi} \xi_\chi(s)$ occurs. If $\xi_\chi(s)$ occurs then χ is replaced by its successor event of the tree where s stems from. If no successor event of that tree is available, then this tree is released, i.e., removed from χ. Therefore, the construction is a generalization of the construction given in Figure 2.12.

The given upper bound is tight, i.e., there are examples that really match it (for example, take the order 0 for the CUB formula with one nesting of temporal operators). Note further that it is easily possible to shift an outermost E path quantifier inside the formula: the only cases where conjunctions occur are the subformulas $\epsilon(s) \wedge \varrho_{\chi\setminus\{s\}}$, which are generated as events $\xi_\chi(s)$ when $\mathrm{suc}^{\mathcal{R}}_{\exists}(\{s\}) = \{\}$. We have already noted in the proof of the above theorem that in this case, $\epsilon(s)$ is propositional. Therefore, we can shift E over $\epsilon(s)$ with the rule $\mathsf{E}\epsilon(s) \wedge \varrho_{\chi\setminus\{s\}} = \epsilon(s) \wedge \mathsf{E}\varrho_{\chi\setminus\{s\}}$.

Theorem 5.25 together with Theorem 5.30 allows the transformation of arbitrary LeftCTL* formulas Φ without X operators to alternation-free μ-calculus formulas of size $O(|\Phi|\, 2^{2|\Phi|})$. We will not prove this here, since this result is proved for general LeftCTL* formulas in the next subsection.

Transformation of Full LeftCTL*

In order to develop a transformation of full LeftCTL* to vectorized μ-calculus formulas, we must additionally handle the X-operator. Recall that the computation of the DeepDNF(Φ) as given in Theorem 5.25 of a given LeftCTL* formula Φ is a disjunction of at most $2^{|\Phi|}$ formulas of the form $\bigwedge_{k=0}^{\ell} \mathsf{X}^k\Phi_k$ where each Φ_k is a CUB formula of an arbitrary order. We will now show, how a vectorized μ-calculus formula for the subformulas $\bigwedge_{k=0}^{\ell} \mathsf{X}^k\Phi_k$ can be computed. The idea of that computation is best explained by considering the simple case where two CUB formulas $\Phi_\imath \wedge \mathsf{X}\Phi_{\imath+1}$ are to be combined. It is important to see that in the first instant of time, only the ι conditions of $\Phi_\imath$ must be satisfied and only the ϵ-conditions of $\Phi_\imath$ are awaited. Hence, in the first instant of time, $\Phi_{\imath+1}$ is totally neglected. Depending on the events that hold in the first instant of time, the situation of the event structure EVS($\Phi_\imath$) can be an arbitrary one, since the transition from one situation to a successive one does not necessarily consume time. In the second instant of time, we must hence combine *all* situations of EVS($\Phi_\imath$) with *all* situations of EVS($\Phi_{\imath+1}$). For this reason, we need to compute the event structure EVS($\Phi_\imath \wedge \Phi_{\imath+1}$), and for the general case EVS($\bigwedge_{k=0}^{\ell} \Phi_k$).

The next theorem shows how this can be established for an arbitrary conjunction $\bigwedge_{k=0}^{\ell} \mathsf{X}^k\Phi_k$ of CUB formulas Φ_k. The key is that beginning with EVS(Φ_0), we unroll the definitions of the substitution variables by applying the recursion laws $[a \underline{\mathsf{U}}\, b] = b \vee a \wedge \mathsf{X}\,[a \underline{\mathsf{U}}\, b]$ and $[a \,\mathsf{U}\, b] = b \vee a \wedge \mathsf{X}\,[a \,\mathsf{U}\, b]$. Inside the scope of the thereby generated X operator, we know that one instant of time has been consumed and therefore we must consider there also the situations of Φ_1. The same procedure is repeated until we have reached the ℓ-th

point of time, where all situations of $\mathsf{EVS}(\Phi_0)$, ..., $\mathsf{EVS}(\Phi_\ell)$ are considered. The detailed construction is given in the following theorem:

Theorem 5.31. *Given CUB formulas $\Phi_0, \ldots, \Phi_\ell$ of arbitrary order and their event structures $\mathsf{EVS}(\Phi_k) = (\mathcal{I}_k, \mathcal{B}_k, \mathcal{S}_k, \mathcal{R}_k, \iota_k, \epsilon_k)$. Define $\Phi := \bigwedge_{k=0}^{\ell} \Phi_k$ and $\Psi := \bigwedge_{k=0}^{\ell} \mathsf{X}^k \Phi_k$. Moreover, let $(\mathcal{I}, \mathcal{B}, \mathcal{S}, \mathcal{R}, \iota, \epsilon) := \mathsf{EVS}(\Phi)$. Then we define equations for variables ϱ_χ^k for $k \in \{0, \ldots, \ell\}$ and for $\chi \in \bigcup_{i=0}^{k} \mathfrak{S}(\mathsf{EVS}(\Phi_i))$ as follows:*

$$\varrho_\chi^k := \begin{cases} \left(\bigvee_{s \in \chi} \xi_\chi^k(s)\right) \vee \left(\bigwedge_{s \in \chi} \iota(s)\right) \wedge \mathsf{X}\varrho_{\chi \cup \mathcal{I}_{k+1}}^{k+1} & : \text{if } k < \ell \\ \left[\left(\bigwedge_{s \in \chi} \iota(s)\right) \mathbin{\underline{\mathsf{U}}} \left(\bigvee_{s \in \chi} \xi_\chi^k(s)\right)\right] & : \text{if } k = \ell \text{ and } \chi \not\subseteq \mathcal{B} \\ \left[\left(\bigwedge_{s \in \chi} \iota(s)\right) \mathsf{U} \left(\bigvee_{s \in \chi} \xi_\chi^k(s)\right)\right] & : \text{if } k = \ell \text{ and } \chi \subseteq \mathcal{B} \end{cases}$$

$$\text{where for each } k \; \xi_\chi^k(s) := \begin{cases} \varrho_{(\chi \setminus \{s\}) \cup \mathsf{suc}_\exists^{\mathcal{R}}(\{s\})}^k & : \text{if } \mathsf{suc}_\exists^{\mathcal{R}}(\{s\}) \neq \{\} \\ \epsilon(s) \wedge \varrho_{\chi \setminus \{s\}}^k & : \text{otherwise} \end{cases}$$

Under these recursive definitions, Ψ is equivalent to $\varrho_{\mathcal{I}_0}^0$. Moreover, the size of all of the above definitions is of order $O(|\Psi|\, 2^{|\Psi|})$.

Proof. The proof is done by induction on ℓ. For $\ell = 0$, the theorem is equivalent to Theorem 5.30. Moreover, from the proof of Theorem 5.30 it follows that $\varrho_{\mathcal{I}_k}^\ell = \Phi_k$ holds for each $\ell \in \mathbb{N}$. For $\ell > 0$, we prove by recursion on j that the following holds:

$$(*) \quad \varrho_\chi^{\ell-k} = \varrho_\chi^\ell \wedge \bigwedge_{j=1}^{k} \mathsf{X}^j \varrho_{\mathcal{I}_{j+\ell-k}}^\ell \text{ for } k \in \{0, \ldots, \ell\} \text{ and } \chi \in \bigcup_{j=0}^{\ell-k} \mathfrak{S}(\mathsf{EVS}(\Phi_j))$$

Note that $\xi_\chi^{\ell-k}(s) = \xi_\chi^\ell(s) \wedge \bigwedge_{j=1}^{k} \mathsf{X}^j \varrho_{\mathcal{I}_{j+\ell-k}}^\ell$ can be directly derived from $(*)$ (this is easily seen by the definition of $\xi_\chi^{\ell-j}(s)$). Before proving $(*)$, we show that the theorem can indeed be derived from it. For this reason, instantiate $(*)$ with $k = \ell$ and $\chi = \mathcal{I}_0$, such that it takes the form $\varrho_{\mathcal{I}_0}^0 = \bigwedge_{j=0}^{\ell} \mathsf{X}^j \varrho_{\mathcal{I}_j}^\ell$. The rest follows from the fact $\varrho_{\mathcal{I}_k}^\ell = \Phi_k$, which we have already showed in the proof of Theorem 5.30.

Consider now the proof of $(*)$ that starts with induction on k: for $k = 0$ the equation degenerates to $\varrho_\chi^\ell = \varrho_\chi^\ell$, which is clearly true for each χ. In the induction step, we have to show $(*)$ where we can use the following induction hypothesis:

$$(**) \quad \varrho_\chi^{\ell-k+1} = \varrho_\chi^\ell \wedge \bigwedge_{j=1}^{k-1} \mathsf{X}^j \varrho_{\mathcal{I}_{j+\ell-k+1}}^\ell \text{ for each } \chi \in \bigcup_{j=0}^{\ell-k+1} \mathfrak{S}(\mathsf{EVS}(\Phi_j))$$

We proceed with another induction, this time on χ. In the induction base, we have to prove $(*)$ for $\chi = \{\}$, i.e. we have to prove $\varrho_{\{\}}^{\ell-k} = \bigwedge_{j=1}^{k} \mathsf{X}^j \varrho_{\mathcal{I}_{j+\ell-k}}^{\ell}$ (note that $\varrho_{\{\}}^{\ell} = [1 \,\mathsf{U}\, 0] = 1$). This is proved with our induction hypothesis $(**)$ as follows:

$$\begin{aligned}
\varrho_{\{\}}^{\ell-k} &\stackrel{def}{=} \left(\bigvee_{s\in\{\}} \xi_{\{\}}^{\ell-k}(s)\right) \vee \left(\bigwedge_{s\in\{\}} \iota(s)\right) \wedge \mathsf{X}\varrho_{\{\}\cup\mathcal{I}_{\ell-k+1}}^{\ell-k+1} \\
&= 0 \vee 1 \wedge \mathsf{X}\varrho_{\{\}\cup\mathcal{I}_{\ell-k+1}}^{\ell-k+1} \\
&= \mathsf{X}\varrho_{\mathcal{I}_{\ell-k+1}}^{\ell-k+1} \stackrel{(**)}{=} \mathsf{X}\left[\varrho_{\mathcal{I}_{\ell-k+1}}^{\ell} \wedge \bigwedge_{j=1}^{k-1} \mathsf{X}^j \varrho_{\mathcal{I}_{j+\ell-k+1}}^{\ell}\right] \\
&= \mathsf{X}\left[\bigwedge_{j=0}^{k-1} \mathsf{X}^j \varrho_{\mathcal{I}_{j+\ell-k+1}}^{\ell}\right] \\
&= \bigwedge_{j=0}^{k-1} \mathsf{X}^{j+1} \varrho_{\mathcal{I}_{j+\ell-k+1}}^{\ell} \\
&= \bigwedge_{j=1}^{k} \mathsf{X}^j \varrho_{\mathcal{I}_{j+\ell-k}}^{\ell}
\end{aligned}$$

In the induction step, it remains to prove $(*)$ for our fixed $k > 0$ and a consistent situation $\chi \neq \{\}$, where we can additionally use the following induction hypothesis $(***)$:

$$(***)\quad \varrho_X^{\ell-k} = \varrho_X^{\ell} \wedge \bigwedge_{j=1}^{k} \mathsf{X}^j \varrho_{\mathcal{I}_{j+\ell-k}}^{\ell} \text{ for each } X \prec \chi$$

The remaining proof is then established as follows, where $k > 0$ and $\chi \neq \{\}$ holds and the assumptions $(**)$ and $(***)$ are used:

$$\begin{aligned}
\varrho_{\chi}^{\ell-k} &\stackrel{def}{=} \left(\bigvee_{s\in\chi} \xi_{\chi}^{\ell-k}(s)\right) \vee \left(\bigwedge_{s\in\chi} \iota(s)\right) \wedge \mathsf{X}\varrho_{\chi\cup\mathcal{I}_{\ell-k+1}}^{\ell-k+1} \\
&\stackrel{(**)}{=} \left(\bigvee_{s\in\chi} \xi_{\chi}^{\ell-k}(s)\right) \vee \left(\bigwedge_{s\in\chi} \iota(s)\right) \wedge \mathsf{X}\left[\varrho_{\chi\cup\mathcal{I}_{\ell-k+1}}^{\ell} \wedge \bigwedge_{j=1}^{k-1} \mathsf{X}^j \varrho_{\mathcal{I}_{j+\ell-k+1}}^{\ell}\right] \\
&= \left(\bigvee_{s\in\chi} \xi_{\chi}^{\ell-k}(s)\right) \vee \left(\bigwedge_{s\in\chi} \iota(s)\right) \wedge \mathsf{X}\left[\varrho_{\chi}^{\ell} \wedge \varrho_{\mathcal{I}_{\ell-k+1}}^{\ell} \wedge \bigwedge_{j=1}^{k-1} \mathsf{X}^j \varrho_{\mathcal{I}_{j+\ell-k+1}}^{\ell}\right] \\
&= \left(\bigvee_{s\in\chi} \xi_{\chi}^{\ell-k}(s)\right) \vee \left(\bigwedge_{s\in\chi} \iota(s)\right) \wedge \mathsf{X}\left[\varrho_{\chi}^{\ell} \wedge \bigwedge_{j=0}^{k-1} \mathsf{X}^j \varrho_{\mathcal{I}_{j+\ell-k+1}}^{\ell}\right] \\
&= \left(\bigvee_{s\in\chi} \xi_{\chi}^{\ell-k}(s)\right) \vee \left(\bigwedge_{s\in\chi} \iota(s)\right) \wedge \mathsf{X}\varrho_{\chi}^{\ell} \wedge \bigwedge_{j=1}^{k} \mathsf{X}^j \varrho_{\mathcal{I}_{j+\ell-k}}^{\ell} \\
&\stackrel{(***)}{=} \left(\bigvee_{s\in\chi} \xi_{\chi}^{\ell}(s) \wedge \bigwedge_{j=1}^{k} \mathsf{X}^j \varrho_{\mathcal{I}_{j+\ell-k}}^{\ell}\right) \vee \left(\bigwedge_{s\in\chi} \iota(s)\right) \wedge \mathsf{X}\varrho_{\chi}^{\ell} \wedge \bigwedge_{j=1}^{k} \mathsf{X}^j \varrho_{\mathcal{I}_{j+\ell-k}}^{\ell} \\
&= \left[\left(\bigvee_{s\in\chi} \xi_{\chi}^{\ell}(s)\right) \vee \left(\bigwedge_{s\in\chi} \iota(s)\right) \wedge \mathsf{X}\varrho_{\chi}^{\ell}\right] \wedge \bigwedge_{j=1}^{k} \mathsf{X}^j \varrho_{\mathcal{I}_{j+\ell-k}}^{\ell} \\
&= \varrho_{\chi}^{\ell} \wedge \bigwedge_{j=1}^{k} \mathsf{X}^j \varrho_{\mathcal{I}_{j+\ell-k}}^{\ell}
\end{aligned}$$

The last step holds due to the recursion laws of $\underline{\mathsf{U}}$ and U (note that ϱ_{χ}^{ℓ} is either a $\underline{\mathsf{U}}$ or a U formula). Now, consider the length of the definitions and let $\Psi = \bigwedge_{k=0}^{\ell} \mathsf{X}^k \Phi_k$. Clearly, Ψ is a CUB formula and the application of Theorem 5.31 results in the definitions for ϱ_{χ}^{ℓ}. According to Lemma 5.29 we know that there are at most $2^{|\Psi|}$ consistent situations of a CUB formula Ψ and moreover $|\chi| \leq |\Psi|$ holds. However, we have additional costs in the above constructions, as possibly $\ell + 1$ definitions $\varrho_{\chi}^{0}, \ldots, \varrho_{\chi}^{\ell}$ for a single situation χ are necessary. Hence, we have in the above construction at most $(\ell + 1)2^{|\Psi|}$ definitions of length $O(|\Psi|)$ resulting in a complexity of order $O((\ell + 1)\,|\Psi|\,2^{|\Psi|})$. This is

certainly not greater than $O(|\Psi|\, 2^{\ell} 2^{|\Psi|})$ and this in turn is not greater than $O((\ell+|\Psi|)2^{\ell+|\Psi|})$. The latter is the same as $O(|\Phi|\, 2^{|\Phi|})$, since we have $O(|\Phi|) = O(\ell + |\Psi|)$. □

This completes the translation of LeftCTL* to vectorized alternation-free μ-calculus. The following theorem summarizes the results which improve the complexity of the former naive transformation.

Theorem 5.32 (Translating LeftCTL* to Alternation-Free μ-Calculus). *Given an arbitrary* LeftCTL* *formula Φ, there is an equivalent vectorized alternation-free μ-calculus formula of length $O(|\Phi|\, 2^{2|\Phi|})$. This implies that there is a model checking procedure for* LeftCTL* *formulas that runs in time $O(|\mathcal{K}|\, |\Phi|\, 2^{2|\Phi|})$ and a satisfiability checking procedure that runs in time $2^{O(|\Phi| 2^{2|\Phi|})}$.*

Proof. We first compute PQNF(Φ) of the given formula Φ and therefore only need to consider LeftCTL* formulas of the form EΦ' where Φ' does not contain any path quantifier at all. To handle these formulas, we compute DeepDNF(Φ') of Φ' by the algorithm given in Figure 5.7. Hence, we obtain a formula $\bigvee_{i \in M} \Phi_i$ with $|M| \in O(2^{\Phi})$, where each Φ_i has a length of order $O(|\Phi|)$ and is of the form $\bigwedge_{k=0}^{\ell_i} \mathsf{X}^k \varphi_{i,k}$ for some CUB formulas $\varphi_{i,k}$. The last theorem can be used to translate each Φ_i in a shared form of size $O(|\Phi_i|\, 2^{|\Phi_i|})$ such that the E path quantifiers can be shifted inwards with only a linear increase of the size. Hence, the overall costs are of order $O(|M|\, |\Phi_i|\, 2^{|\Phi_i|}) = O(|M|\, |\Phi|\, 2^{|\Phi|}) = O(|\Phi|\, 2^{2|\Phi|})$.

The proposition for the complexity of the satisfiability check is to due to satisfiability checking of CTL formulas that can be done in time $2^{O(|\Phi|)}$ [158]. □

We note that the upper bound of the transformation does not seem not to be tight, i.e., no example has been found that leads to an increase of $O(|\Phi|\, 2^{2|\Phi|})$. However, as our illustrating example of Subsection 5.3.5 showed, there are examples that lead to an increase of $O(|\Phi|\, 2^{|\Phi|})$. In either case however, the result shown in this section proves that the transformation from LeftCTL* can become exponential if vectorized μ-calculus formulas are used. Note, however, the lower bounds established in [4, 266, 513].

5.4 Translating Temporal Logics to ω-Automata

In the previous section, we have considered translations of the logics CTL, CTL^2, LeftCTL*, FairCTL, and FairLeftCTL* to the μ-calculus, so that we have obtained efficient decision procedures for these logics. There are however more powerful temporal logics whose model checking problems we have not considered so far. We will therefore consider in this section model checking algorithms for more general logics, in particular for CTL*. However, we will

not directly translate to the μ-calculus. Instead, we will show how quantifier-free formulas (even with past time temporal operators) can be efficiently translated to equivalent ω-automata. As we already know how to translate the branching time logic $\mathcal{L}_\omega$ of ω-automata to the μ-calculus, we can also translate arbitrary branching time temporal logic formulas to the μ-calculus.

The ω-automaton $\mathfrak{A}_\Phi$ that is computed for a given formula Φ has, at most, $O(2^{|\Phi|})$ states. Nevertheless, our translation procedure is able to compute that automaton in linear time $O(|\Phi|)$ because it generates a *symbolic representation* in terms of our ω-automaton formulas (it is easy to see that we can write for every $i \in \mathbb{N}$ an ω-automaton formula $\mathfrak{A}_i = \mathcal{A}_\exists(Q, \Phi_\mathcal{I}, \Phi_\mathcal{R}, \Phi_\mathcal{F})$ of size $O(i)$ with $O(2^i)$ reachable states). The translation from ω-automata to the μ-calculus can not however benefit from the symbolic representation of the ω-automata. Hence, we get an exponential blow-up there: given an arbitrary temporal logic formula Φ, we can compute an equivalent μ-calculus formula of exponential size in terms of $|\Phi|$.

This blow-up can probably not be avoided due to complexity results: CTL* model checking is PSPACE-complete (Section 5.6). We have already seen that model checking of alternation-free μ-calculus formulas can be done in linear time. So, if there were a polynomial translation from a temporal logic to equivalent alternation-free μ-calculus formulas, this would result in a polynomial time model checking procedure for that logic. Hence, for temporal logics like LeftCTL*, LTL, and CTL*, it is probably not possible to obtain μ-calculus formulas of polynomial size (in general), since their model checking problems are NP-hard (even Δ_2^p-complete) and PSPACE-complete (Section 5.6), respectively.

Similar to the automaton hierarchy (cf. Figure 4.33), we will also define a corresponding hierarchy for temporal logics. Recall that any of the automaton classes has a deterministic representative: For example, we can choose the classes Det_G, Det_F, $\mathsf{Det}_\mathsf{Prefix}$, Det_FG, Det_GF, and $\mathsf{Det}_\mathsf{Streett}$ as representatives. We have also seen that our translation from Det_GF, and $\mathsf{Det}_\mathsf{Streett}$ to the μ-calculus results in μ-calculus formulas with alternation depth 2, while the classes Det_G, Det_F, $\mathsf{Det}_\mathsf{Prefix}$, Det_FG can be translated to the alternation-free μ-calculus. Therefore, we are interested in temporal logics TL_G, TL_F, $\mathsf{TL}_\mathsf{Prefix}$, and TL_FG that correspond with the automaton classes Det_G, Det_F, $\mathsf{Det}_\mathsf{Prefix}$, and Det_FG because these formulas can be translated to the alternation-free μ-calculus and can therefore be checked more efficiently.

The outline of this section is therefore as follows: in the next subsection, we will present a simple translation from quantifier-free temporal logic formulas to equivalent $\mathsf{NDet}_\mathsf{Streett}$ automata. We will then show in Section 5.4.2 how this translation can be improved by considering the monotonicity of temporal operators. This observation necessitates the definition of the temporal logics TL_G, TL_F, $\mathsf{TL}_\mathsf{Prefix}$, TL_FG, TL_GF, and $\mathsf{TL}_\mathsf{Streett}$ in Section 5.4.3. We construct efficient procedures for translation of these logics to their corresponding automaton class in Section 5.4.4. We then investigate the tight relationship between the logic LeftCTL* and the logics TL_FG and TL_GF in Sec-

tion 5.4.5 and show there that the *model checking problems* of these logics can be directly translated to equivalent CTL model checking problems. For this reason, we will also define a logic AFCTL* that can be translated to the alternation-free μ-calculus. In Section 5.4.5, we collect all these results in a final translation procedure that is able to translate CTL* model checking problems to equivalent FairCTL model checking problems.

In the following, we will often consider the quantifier-free fragment of CTL*, and often we furthermore allow past temporal operators. Hence, for reasons of conciseness, we will define the following temporal logic LTL$_p$:

Definition 5.33 (Temporal Path Formulas LTL$_p$). *Given a finite set of variables V_Σ, the following grammar rules define the temporal logic* LTL$_p$*:*

$$\begin{aligned} P ::= {} & S \mid \neg P \mid P \wedge P \mid P \vee P \\ & \mid \mathsf{X}P \mid \mathsf{G}P \mid \mathsf{F}P \\ & \mid \overleftarrow{\mathsf{X}}P \mid \overleftarrow{\underline{\mathsf{X}}}P \mid \overleftarrow{\mathsf{G}}P \mid \overleftarrow{\mathsf{F}}P \\ & \mid [P\ \mathsf{U}\ P] \mid [P\ \mathsf{B}\ P] \mid [P\ \underline{\mathsf{U}}\ P] \mid [P\ \underline{\mathsf{B}}\ P] \\ & \mid [P\ \overleftarrow{\mathsf{U}}\ P] \mid [P\ \overleftarrow{\mathsf{B}}\ P] \mid [P\ \overleftarrow{\underline{\mathsf{U}}}\ P] \mid [P\ \overleftarrow{\underline{\mathsf{B}}}\ P] \end{aligned}$$

We have excluded the W and $\underline{\mathsf{W}}$ operators, to reduce the number of cases. Clearly, we could do the same with some other operators, but want to retain the others to show up some similarities. The algorithms in the remainder of this section do also only consider the above set of temporal operators, sometimes even further restricted to the next and previous operators, the strong-until and the weak-before (or the weak until).

5.4.1 The Basic Translation from LTL$_p$ to NDet$_{\text{Streett}}$

Translations from temporal logics to equivalent ω-automata have been intensively studied. Early research, as done by Vardi and Wolper considered translations to equivalent Büchi automata, where the automaton has been explicitly constructed [386]. For that reason, Vardi and Wolper have defined the set of *elementary subformulas* of a given formula Φ which is the set of all subformulas of Φ that start with a temporal operator. The procedure has been refined in a couple of further papers [138, 185, 207, 208, 468, 520]. We will not directly present Vardi and Wolper's procedure, but its underlying ideas are still contained in the procedures that we will present.

Assume that Φ has the elementary formulas $\{\varphi_1, \ldots, \varphi_n\}$. The states of the ω-automaton that is to be constructed consist of the truth values of these elementary formulas. For this reason, we need n state variables $\{q_1, \ldots, q_n\}$ to encode the state set, and therefore we already see where the exponential blow-up comes from. Clearly, for any run through the automaton, we want $q_i \leftrightarrow \varphi_i$ to hold. For this reason, the transition relation of the automaton must respect the semantics of the temporal operators that occur in φ_i. Having this view, we will describe in this section how to define a transition relation for

the ω-automaton such that the desired equivalences $q_i \leftrightarrow \varphi_i$ hold for all elementary subformulas of the given LTL_p formula Φ. In particular, we are able to define a transition relation of size $O(|\Phi|)$ so that our translation procedure will also run in time $O(|\Phi|)$, which is not possible for explicit constructions of the automaton. Procedures that translate temporal logic formulas to symbolic descriptions have been considered in [88, 89, 105, 131, 281, 439].

For example, consider the formula $\mathsf{FG}a \rightarrow \mathsf{GF}a$. The elementary subformulas are abbreviated as follows: $q_1 = \mathsf{G}a$, $q_2 = \mathsf{F}q_1$, $q_3 = \mathsf{F}a$, $q_4 = \mathsf{G}q_3$, so that the resulting formula is $\Psi \equiv q_2 \rightarrow q_4$. Therefore, our ω-automaton will be constructed with the state variables $\{q_1, q_2, q_3, q_4\}$, and its transitions should be such that the above equations are respected. For this reason, the following recursion laws must be respected by the transition relation of our automaton:

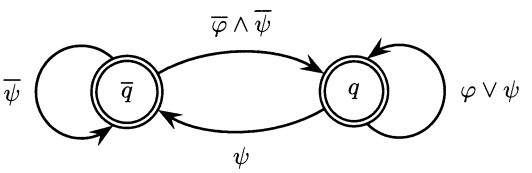
Lemma 5.34 (Recursion Laws of Temporal Operators). *The following equations are generally valid for all formulas φ and ψ:*

$$\mathsf{G}\varphi = \varphi \wedge \mathsf{XG}\varphi \qquad \mathsf{F}\varphi = \varphi \vee \mathsf{XF}\varphi$$
$$[\varphi\ \mathsf{U}\ \psi] = \psi \vee \varphi \wedge \mathsf{X}[\varphi\ \mathsf{U}\ \psi] \qquad [\varphi\ \underline{\mathsf{U}}\ \psi] = \psi \vee \varphi \wedge \mathsf{X}[\varphi\ \underline{\mathsf{U}}\ \psi]$$
$$[\varphi\ \mathsf{B}\ \psi] = \neg\psi \wedge (\varphi \vee \mathsf{X}[\varphi\ \mathsf{B}\ \psi] \qquad [\varphi\ \underline{\mathsf{B}}\ \psi] = \neg\psi \wedge (\varphi \vee \mathsf{X}[\varphi\ \underline{\mathsf{B}}\ \psi]$$
$$\overleftarrow{\mathsf{G}}\varphi = \varphi \wedge \overleftarrow{\mathsf{X}}\overleftarrow{\mathsf{G}}\varphi \qquad \overleftarrow{\mathsf{F}}\varphi = \varphi \vee \overleftarrow{\underline{\mathsf{X}}}\overleftarrow{\mathsf{F}}\varphi$$
$$[\varphi\ \overleftarrow{\mathsf{U}}\ \psi] = \psi \vee \varphi \wedge \overleftarrow{\mathsf{X}}[\varphi\ \overleftarrow{\mathsf{U}}\ \psi] \qquad [\varphi\ \overleftarrow{\underline{\mathsf{U}}}\ \psi] = \psi \vee \varphi \wedge \overleftarrow{\underline{\mathsf{X}}}[\varphi\ \overleftarrow{\underline{\mathsf{U}}}\ \psi]$$
$$[\varphi\ \overleftarrow{\mathsf{B}}\ \psi] = \neg\psi \wedge (\varphi \vee \overleftarrow{\mathsf{X}}[\varphi\ \overleftarrow{\mathsf{B}}\ \psi] \qquad [\varphi\ \overleftarrow{\underline{\mathsf{B}}}\ \psi] = \neg\psi \wedge (\varphi \vee \overleftarrow{\underline{\mathsf{X}}}[\varphi\ \overleftarrow{\underline{\mathsf{B}}}\ \psi]$$

It is to be noted that the strong and weak operators share exactly the same recursion laws and that the recursion laws of the past time modalities correspond with the recursion laws of the corresponding future time modality. Furthermore, for the past time modalities, we must use the strong previous operator in the recursion laws for the strong operators and the weak one for the recursion laws of the weak operators.

Hence, given that both φ and ψ would be propositional formulas, we could built an ω-automaton by simply substituting a new state variable q for the elementary subformula that should be abbreviated by q. For example, for the formula $[\varphi\ \underline{\mathsf{U}}\ \psi]$, the transition relation should then be $q \leftrightarrow \psi \vee \varphi \wedge \mathsf{X}q$. Using this transition relation, we know that q is a solution of the fixpoint equation $x \leftrightarrow \psi \vee \varphi \wedge \mathsf{X}x$.

However, this does not help much at the moment: We already know that $[\varphi\ \mathsf{U}\ \psi]$ and $[\varphi\ \underline{\mathsf{U}}\ \psi]$ are also solutions of this equation, and therefore there is more than one solution. However, it could be the case that there could possibly be other solutions, too. For this reason, we now consider the solutions of the above fixpoint equations in detail. The following lemma lists the essential properties of the fixpoint equation $q \leftrightarrow \psi \vee \varphi \wedge \mathsf{X}q$ and will be of important use for a lot of theorems in this section. We will see that besides $[\varphi\ \mathsf{U}\ \psi]$ and $[\varphi\ \underline{\mathsf{U}}\ \psi]$, there are no further solutions, and this will make the automata even unambiguous, i.e., every accepted word has exactly one accepting run.



ψ	φ	$q \leftrightarrow \psi \vee \varphi \wedge \mathsf{X}q$
0	0	$q \leftrightarrow 0$
0	1	$q \leftrightarrow \mathsf{X}q$
1	0	$q \leftrightarrow 1$
1	1	$q \leftrightarrow 1$

Fig. 5.10. ω-automaton for discussing the fixpoint equation $q \leftrightarrow \psi \vee \varphi \wedge \mathsf{X}q$

Lemma 5.35 (Solutions of $q \leftrightarrow \psi \vee \varphi \wedge \mathsf{X}q$). *Assuming that* $\mathsf{G}[q \leftrightarrow \psi \vee \varphi \wedge \mathsf{X}q]$ *initially holds, then the following facts can be derived (for the initial point of time):*

(1) $\mathsf{G}[\psi \rightarrow q] \wedge \mathsf{G}[\neg\varphi \wedge \neg\psi \rightarrow \neg q] \wedge \mathsf{G}[\varphi \wedge \neg\psi \rightarrow (q \leftrightarrow \mathsf{X}q)]$
(2) $\mathsf{G}[\mathsf{F}(\varphi \rightarrow \psi) \rightarrow (q \leftrightarrow [\varphi \,\underline{\mathsf{U}}\, \psi])]$
(3) $\mathsf{G}\Big(\mathsf{F}(\varphi \rightarrow \psi) \Rightarrow \overleftarrow{\mathsf{G}}(q \leftrightarrow [\varphi \,\underline{\mathsf{U}}\, \psi]) \,\Big|\, \mathsf{G}q \vee \mathsf{G}\neg q\Big)$
(4) $\mathsf{G}[q \leftrightarrow [\varphi \,\underline{\mathsf{U}}\, \psi]] \vee \mathsf{G}[q \leftrightarrow [\varphi \,\mathsf{U}\, \psi]]$
(5) $\mathsf{G}\Big(\mathsf{F}(q \rightarrow \psi) \Rightarrow \overleftarrow{\mathsf{G}}(q \leftrightarrow [\varphi \,\underline{\mathsf{U}}\, \psi]) \,\Big|\, \mathsf{G}(q \wedge \varphi \wedge \neg\psi)\Big)$
(6) $\mathsf{G}\Big(\mathsf{F}(q \rightarrow \psi) \Rightarrow \overleftarrow{\mathsf{G}}(q \leftrightarrow [\varphi \,\underline{\mathsf{U}}\, \psi]) \,\Big|\, \mathsf{G}(q \leftrightarrow [\varphi \,\mathsf{U}\, \psi])\Big)$
(7) $\mathsf{G}\Big(\mathsf{F}(q \rightarrow \psi) \Rightarrow \overleftarrow{\mathsf{G}}(q \leftrightarrow [\varphi \,\underline{\mathsf{U}}\, \psi]) \,\Big|\, \overleftarrow{\mathsf{G}}(q \leftrightarrow [\varphi \,\mathsf{U}\, \psi]) \wedge \mathsf{G}(q \leftrightarrow [\varphi \,\mathsf{U}\, \psi])\Big)$
(8) $\mathsf{G}\Big[(q \leftrightarrow [\varphi \,\underline{\mathsf{U}}\, \psi]) \leftrightarrow \overleftarrow{\mathsf{G}}(q \leftrightarrow [\varphi \,\underline{\mathsf{U}}\, \psi])\Big]$
(9) $\mathsf{G}[(q \leftrightarrow [\varphi \,\underline{\mathsf{U}}\, \psi]) \leftrightarrow \mathsf{F}(q \rightarrow \psi)]$
(10) $\mathsf{G}[(q \leftrightarrow [\varphi \,\mathsf{U}\, \psi]) \leftrightarrow \mathsf{F}(\varphi \rightarrow q)]$

Before turning to the proof, we first discuss some of the above facts. (1) means that the truth value of any solution q of the fixpoint equation is immediately and uniquely determined at any point of time where either ψ or $\neg\varphi \wedge \neg\psi$ holds. In the remaining case, where $\varphi \wedge \neg\psi$ holds, the truth value is the same as that of the next point of time.

(2) details this in that it additionally says that whenever $\varphi \rightarrow \psi$ holds (note that $\neg(\varphi \wedge \neg\psi)$ is equivalent to $\varphi \rightarrow \psi$) at some point of time, then we definitely have $q \leftrightarrow [\varphi \,\underline{\mathsf{U}}\, \psi]$ (and also $q \leftrightarrow [\varphi \,\mathsf{U}\, \psi]$). The same holds if only $\mathsf{F}(\varphi \rightarrow \psi)$ holds at some point of time. (3) gives even more details: if $\mathsf{F}(\varphi \rightarrow \psi)$ holds at some point of time, then we have $q \leftrightarrow [\varphi \,\underline{\mathsf{U}}\, \psi]$ (and also $q \leftrightarrow [\varphi \,\mathsf{U}\, \psi]$) for now and all previous moments of time. (3) also states what happens if $\mathsf{F}(\varphi \rightarrow \psi)$ does not hold: then there are two solutions for q: one that is constantly 1, the other one is constantly 0. (4) specifies the set of all solutions of the fixpoint equations: we either have $q \leftrightarrow [\varphi \,\underline{\mathsf{U}}\, \psi]$ or $q \leftrightarrow [\varphi \,\mathsf{U}\, \psi]$ for all moments of time. *Hence, the equation* $q \leftrightarrow \psi \vee \varphi \wedge \mathsf{X}q$ *has exactly two solutions!* $(5-7)$ specifies in more detail in which case we have $q \leftrightarrow [\varphi \,\underline{\mathsf{U}}\, \psi]$ or $q \leftrightarrow [\varphi \,\mathsf{U}\, \psi]$, respectively, and $(8-10)$ are further reformulations of (5). Now, consider the proofs of these facts:

Proof. We prove the facts in the listed order, where we consider an arbitrary path π of an arbitrary Kripke structure $\mathcal{K}$. As a shorthand notation, we simply write $\varphi^{(t)}$ instead of $(\mathcal{K}, \pi, t) \models \varphi$ to be more concise in the following proofs:

(1): These facts are obtained by propositional simplifications of the fixpoint equation: For example, if $\varphi \wedge \neg\psi$ holds, then $q \leftrightarrow \psi \vee \varphi \wedge \mathsf{X}q$ simplifies to $q \leftrightarrow \mathsf{X}q$.

(2): Let t_0 be an arbitrary point of time, where $\mathsf{F}(\varphi \rightarrow \psi)$ holds. Then, there is a number t_1 such that (2.1) $\varphi \rightarrow \psi$ holds at time $t_0 + t_1$. In particular, we can choose t_1 to be the smallest number with this property, so that we additionally have (2.2) $\forall t < t_1.\varphi^{(t_0+t)} \wedge \neg\psi^{(t_0+t)}$. By (1), it then follows that (2.3) $\forall t < t_1. \rightarrow q^{(t_0+t)} = q^{(t_0+t+1)}$. This means that the truth value of any solution of the fixpoint equation remains constant until $\varphi \wedge \neg\psi$ is read. Hence, the truth value of $q^{(t_0+t_1)}$ determines the truth value for all $q^{(t_0+t)}$ with $t < t_1$, i.e., we have (2.4) $\forall t \leq t_1.q^{(t_0+t)} = q^{(t_0+t_1)}$, and in particular that (2.5) $q^{(t_0)} = q^{(t_0+t_1)}$ ((2.5) can be proved without (2.4) by induction on t_1). Now, we consider two cases according to (2.1):
In the first case, we have (2.1.1) $\psi^{(t_0+t_1)}$, and hence obtain by (1) that $q^{(t_0+t_1)}$ holds, and hence, by (2.5) that $q^{(t_0)}$ holds. By (2.2), (2.1.1), and the semantics of the temporal operators, we also have $[\varphi \,\underline{\mathsf{U}}\, \psi]^{(t_0)}$, so that the equation $q^{(t_0)} = [\varphi \,\underline{\mathsf{U}}\, \psi]^{(t_0)}$ holds.
In the second case, we have (2.1.2) $\neg\varphi^{(t_0+t_1)} \wedge \neg\psi^{(t_0+t_1)}$, and hence obtain by (1) that $\neg q^{(t_0+t_1)}$ holds, and hence, by (2.5) that $\neg q^{(t_0)}$ holds. By (2.2), (2.1.2), and the semantics of the temporal operators, we also have $\neg\,[\varphi \,\underline{\mathsf{U}}\, \psi]^{(t_0)}$, so that the equation $q^{(t_0)} = [\varphi \,\underline{\mathsf{U}}\, \psi]^{(t_0)}$ holds.

(3): We have to prove the implications $\mathsf{G}\left[\mathsf{F}(\varphi \rightarrow \psi) \rightarrow \overleftarrow{\mathsf{G}}\,(q \leftrightarrow [\varphi \,\underline{\mathsf{U}}\, \psi])\right]$ and $\mathsf{G}\,[\neg\mathsf{F}(\varphi \rightarrow \psi) \rightarrow (\mathsf{G}q \vee \mathsf{G}\neg q)]$. The first implication follows more or less directly from (2): Assume that t_0 is an arbitrary point of time where $[\mathsf{F}(\varphi \rightarrow \psi)]^{(t_0)}$ holds, then we also have for all t_1 with $t_1 \leq t_0$ that $[\mathsf{F}(\varphi \rightarrow \psi)]^{(t_1)}$ holds. Thus we can use (2) to derive for all t_1 with $t_1 \leq t_0$ that $[q \leftrightarrow [\varphi \,\underline{\mathsf{U}}\, \psi]]^{(t_1)}$ holds.
To prove $\mathsf{G}\,[\neg\mathsf{F}(\varphi \rightarrow \psi) \rightarrow (\mathsf{G}q \vee \mathsf{G}\neg q)]$, assume that at some point of time t_0, we would have $\neg[\mathsf{F}(\varphi \rightarrow \psi)]^{(t_0)}$, i.e. (3.1) $\forall t.\varphi^{(t_0+t)} \wedge \neg\psi^{(t_0+t)}$. From the fixpoint equation, it then follows that (3.2) $\forall t.q^{(t_0+t)} = q^{(t_0+t+1)}$, so that we conclude by induction on t that (3.3) $\forall t.q^{(t_0+t)} = q^{(t_0)}$. Depending on $q^{(t_0)}$, we thus have either $[\mathsf{G}q]^{(t_0)}$ or $[\mathsf{G}\neg q]^{(t_0)}$.

(4) We consider two cases: In the first case, we assume that $\mathsf{GF}(\varphi \rightarrow \psi)^{(0)}$ holds. Thus, we have $\mathsf{F}(\varphi \rightarrow \psi)^{(t_0)}$ for all t_0, and hence by (2) that $q^{(t_0)} = [\varphi \,\underline{\mathsf{U}}\, \psi]^{(t_0)}$ holds. In the second case, we have $\mathsf{FG}(\varphi \wedge \neg\psi)^{(0)}$, hence there must be a point of time t_0, with (4.1) $\forall t \geq t_0.\varphi^{(t_0+t)} \wedge \neg\psi^{(t_0+t)}$. By (3), it therefore follows that (4.2) $[\mathsf{G}q]^{(t_0)}$ or $[\mathsf{G}\neg q]^{(t_0)}$ holds. Combining (4.1) and (4.2) results in (4.3) $[\mathsf{G}(q \wedge \varphi \wedge \neg\psi)]^{(t_0)}$ or $[\mathsf{G}(\neg q \wedge \varphi \wedge \neg\psi)]^{(t_0)}$. Hence, we consider two cases according to (4.3.1): If $[\mathsf{G}(q \wedge \varphi \wedge \neg\psi)]^{(t_0)}$ holds, then it follows by the semantics of the temporal operators that

$[\mathsf{G}(q \leftrightarrow [\varphi\ \mathsf{U}\ \psi])]^{(t_0)}$, and if $[\mathsf{G}(\neg q \wedge \varphi \wedge \neg\psi)]^{(t_0)}$ holds, then it follows by the semantics of the temporal operators that $[\mathsf{G}(q \leftrightarrow [\varphi\ \underline{\mathsf{U}}\ \psi])]^{(t_0)}$.

(5): We first prove that $\mathsf{G}[\neg\mathsf{F}(q \rightarrow \psi) \rightarrow \mathsf{G}(q \wedge \varphi \wedge \neg\psi]$ holds: Let t_0 be an arbitrary point of time, where $\neg[\mathsf{F}(q \rightarrow \psi)]^{(t_0)}$ holds. Then, we have $[\mathsf{G}(q \wedge \neg\psi)]^{(t_0)}$, so that by our fixpoint equation $[\mathsf{G}\varphi]^{(t_0)}$ follows, which already proves our second implication.

Now we prove that $\mathsf{G}[\mathsf{F}(q \rightarrow \psi) \rightarrow (q \leftrightarrow [\varphi\ \underline{\mathsf{U}}\ \psi])]$ holds: Let t_0 be an arbitrary point of time, where $[\mathsf{F}(q \rightarrow \psi)]^{(t_0)}$ holds. This is obviously equivalent to (5.1) $[\mathsf{F}\neg q]^{(t_0)} \vee [\mathsf{F}\psi]^{(t_0)}$, so that we make a case distinction: In the first case, we have (5.1.1) $[\mathsf{F}\psi]^{(t_0)}$, and thus also (5.1.2) $[\mathsf{F}(\varphi \rightarrow \psi)]^{(t_0)}$. Hence, our proposition follows from (2).

In the second case, we have (5.1.2) $[\mathsf{F}\neg q]^{(t_0)} \wedge \neg[\mathsf{F}\psi]^{(t_0)}$. Hence, there is a number t_1 such that (5.1.3) $\neg q^{(t_0+t_1)}$ holds. In particular, we can choose t_1 to be the least number with this property, so that we additionally have (5.1.4) $\forall t < t_1.q^{(t_0+t)}$. By (5.1.2), (5.1.4) and our fixpoint equation, we now obtain (5.1.5) $\forall t < t_1.\varphi^{(t_0+t)} \wedge \neg\psi^{(t_0+t)} \wedge q^{(t_0+t)} \wedge q^{(t_0+t+1)}$. But then, it follows that $t_1 = 0$ must hold: If $0 < t_1$ holds, then we could instantiate $t := t_1 - 1$ in (5.1.5), so that we obtain $q^{(t_0+t_1)}$, and hence a contradiction to (5.1.3). Therefore, we have (5.1.6) $t_1 = 0$, and thus by (5.1.3) that (5.1.7) $\neg q^{(t_0)}$. We now make a case distinction on (4): In the first case, we have $\mathsf{G}[q \leftrightarrow [\varphi\ \underline{\mathsf{U}}\ \psi]]$, and therefore our proposition does trivially hold. In the second case, we have $\mathsf{G}[q \leftrightarrow [\varphi\ \mathsf{U}\ \psi]]$, and thus by (5.1.7) that $\neg[\varphi\ \mathsf{U}\ \psi]^{(t_0)}$. As strong operators imply the weak ones, this implies by contraposition $\neg[\varphi\ \underline{\mathsf{U}}\ \psi]^{(t_0)}$ which also proves our proposition in this case.

It is now straightforward to prove that our first implication holds: Just note that if $\mathsf{F}\Phi$ holds at some point of time t_0 then it also holds at any point of time t_1 with $t_1 \leq t_0$.

(6): The first implication follows immediately from (5). For the second one, let t_0 be arbitrarily chosen. We obtain from (5) that $[\mathsf{G}(q \wedge \varphi \wedge \neg\psi)]^{(t_0)}$ holds, and this implies by the semantics of the temporal operators that also $[\mathsf{G}(q \leftrightarrow [\varphi\ \mathsf{U}\ \psi])]^{(t_0)}$ holds.

(7): We only prove $\mathsf{G}\left(\neg\mathsf{F}[q \rightarrow \psi] \rightarrow \overleftarrow{\mathsf{G}}(q \leftrightarrow [\varphi\ \underline{\mathsf{U}}\ \psi])\right)$, the rest follows immediately from (6). Assume first that (7.1) $[\mathsf{GF}[q \rightarrow \psi]]^{(0)}$ holds. Then, it follows by (5) that $[\mathsf{G}[q \leftrightarrow [\varphi\ \underline{\mathsf{U}}\ \psi]]]^{(0)}$ must hold for any solution q of the fixpoint equation. As $[\varphi\ \mathsf{U}\ \psi]$ is by (4) a solution of the fixpoint equation, we thus have $[\mathsf{G}[[\varphi\ \mathsf{U}\ \psi] \leftrightarrow [\varphi\ \underline{\mathsf{U}}\ \psi]]]^{(0)}$, so that our proposition holds.

If, on the other hand, (7.2) $[\mathsf{FG}[q \wedge \neg\psi]]^{(0)}$ holds, then there is a point of time t_0 where (7.3) $[\mathsf{G}[q \wedge \neg\psi]]^{(t_0)}$ holds. We consider the least t_0 with that property, i.e. we have (7.4) $\forall t < t_0.[\mathsf{F}[q \rightarrow \psi]]^{(t)}$. If $t_0 = 0$ holds, then our proposition follows by (7.3) and (6). Hence, assume further (7.5) $0 < t_0$, so that we can instantiate $t := t_0 - 1$ in (7.4) to obtain (7.6) $[\mathsf{F}[q \rightarrow \psi]]^{(t_0-1)}$, and thus together with (5) that $[\overleftarrow{\mathsf{G}}[q \leftrightarrow [\varphi\ \underline{\mathsf{U}}\ \psi]]]^{(t_0-1)}$ holds for any solution of the fixpoint equation. As $[\varphi\ \mathsf{U}\ \psi]$ is by (4) a solution of the

fixpoint equation, we thus have $[\overleftarrow{\mathsf{G}}[[\varphi \mathsf{U} \psi] \leftrightarrow [\varphi \underline{\mathsf{U}} \psi]]]^{(t_0-1)}$, and hence $[\overleftarrow{\mathsf{G}}[q \leftrightarrow [\varphi \underline{\mathsf{U}} \psi]]]^{(t_0-1)}$.

(8) and (9): These are just simple consequences of (5).

(10): The implication from left to right is simple: Assume that (10.1) $q^{(t_0)} = [\varphi \mathsf{U} \psi]^{(t_0)}$ holds, but $\mathsf{F}[\varphi \rightarrow q]$ will not hold at some point of time t_0. Hence, this means that (10.2) $\mathsf{G}[\varphi \wedge \neg q]$ must hold at t_0. But then, due to (10.1) and (10.2) we have that (10.3) $\neg [\varphi \mathsf{U} \psi]^{(t_0)}$, and hence, (10.4) $[(\neg\varphi) \underline{\mathsf{B}} \psi]^{(t_0)}$. This implies however that (10.5) $\mathsf{F}(\neg\varphi)^{(t_0)}$, which contradicts (10.2). Hence, (10.1) and (10.2) can not both hold, so that (10.1) implies the negation of (10.2).

For the implication from right to left, we first note that by fact (4) of the lemma, we already know that either $\mathsf{G}[q \leftrightarrow [\varphi \underline{\mathsf{U}} \psi]]$ or $\mathsf{G}[q \leftrightarrow [\varphi \mathsf{U} \psi]]$ hold. In the latter case, there is nothing to prove, so consider the former one, i.e., we have (10.6) $\mathsf{G}[q \leftrightarrow [\varphi \underline{\mathsf{U}} \psi]]$. Consider an arbitrary point of time t_0. Given that $q^{(t_0)}$ holds, we have by (10.6) that $[\varphi \underline{\mathsf{U}} \psi]^{(t_0)}$, and as strong operators imply the weak ones, we also have $[\varphi \mathsf{U} \psi]^{(t_0)}$. So, consider the remaining case where (10.7) $\neg q^{(t_0)}$ holds. By (10.6), we also have (10.8) $\neg [\varphi \underline{\mathsf{U}} \psi]^{(t_0)}$ and have to prove that $\neg [\varphi \mathsf{U} \psi]^{(t_0)}$ holds. It is sufficient to show that $[\mathsf{F}\neg\varphi]^{(t_0)}$ holds, since under this assumption (10.8) implies $\neg [\varphi \mathsf{U} \psi]^{(t_0)}$. By our assumption, we have (10.9) $[\mathsf{F}(\varphi \rightarrow q)]^{(t_0)}$, so that we consider two cases:

- $[\mathsf{F}\neg\varphi]^{(t_0)}$: As our case assumption is identical to the goal of our proof, there is nothing to prove.
- $[\mathsf{F}q]^{(t_0)}$: Let t_1 be the least point of time after t_0 where $q^{(t_0+t_1)}$ holds. Due to the fixpoint equation of q, it follows that we can only move from $\neg q$ to q when $\neg\varphi \wedge \neg\psi$ is read. Hence, we have $\neg\varphi^{(t_0+t_1)}$, which implies $[\mathsf{F}\neg\varphi]^{(t_0)}$.

The previous lemma is quite powerful. We will see that it provides us with the main hints for many improvements of the translation from temporal logics to ω-automata. At the moment, only fact (4) of the lemma is of interest for us since it precisely specifies the set of solutions of our fixpoint equations. We can also determine in the same manner the set of solutions of the other fixpoint equations. As a result, we will see in the next theorem that the fixpoint equations for the past time operators have a unique solution[3] and that the fixpoint equations for the future time operators have exactly two solutions that are given by the strong and weak variant of the operator (for G and F, we view 0 and 1 as counterparts).

[3] This is due to the fact that all past time operators can be defined by primitive recursion: For example, $[\varphi \overleftarrow{\underline{\mathsf{U}}} \psi]^{(0)} := 0$ and $[\varphi \overleftarrow{\underline{\mathsf{U}}} \psi]^{(t+1)} := \psi^{(t)} \vee \varphi^{(t)} \wedge [\varphi \overleftarrow{\underline{\mathsf{U}}} \psi]^{(t)}$. Note that each primitive recursive definition has a unique solution.

Theorem 5.36 (Temporal Fixpoint Equations). *The following equivalences are generally valid for all formulas φ and $\psi \in \mathsf{LTL}_p$. These equivalences determine the set of solutions of the fixpoint equations of Lemma 5.34:*

$$\begin{aligned}
\mathsf{G}[q \leftrightarrow \varphi \wedge \mathsf{X}q] &\Leftrightarrow \mathsf{G}[q \leftrightarrow \mathsf{G}\varphi] \vee \mathsf{G}[q \leftrightarrow 0] \\
\mathsf{G}[q \leftrightarrow \varphi \vee \mathsf{X}q] &\Leftrightarrow \mathsf{G}[q \leftrightarrow \mathsf{F}\varphi] \vee \mathsf{G}[q \leftrightarrow 1] \\
\mathsf{G}[q \leftrightarrow \psi \vee \varphi \wedge \mathsf{X}q] &\Leftrightarrow \mathsf{G}[q \leftrightarrow [\varphi\ \mathsf{U}\ \psi]] \vee \mathsf{G}[q \leftrightarrow [\varphi\ \underline{\mathsf{U}}\ \psi]] \\
\mathsf{G}[q \leftrightarrow \neg\psi \wedge (\varphi \vee \mathsf{X}q)] &\Leftrightarrow \mathsf{G}[q \leftrightarrow [\varphi\ \mathsf{B}\ \psi]] \vee \mathsf{G}[q \leftrightarrow [\varphi\ \underline{\mathsf{B}}\ \psi]] \\
\mathsf{G}[q \leftrightarrow \varphi \wedge \overleftarrow{\mathsf{X}}q] &\Leftrightarrow \mathsf{G}[q \leftrightarrow \overleftarrow{\mathsf{G}}\varphi] \\
\mathsf{G}[q \leftrightarrow \varphi \vee \overleftarrow{\underline{\mathsf{X}}}q] &\Leftrightarrow \mathsf{G}[q \leftrightarrow \overleftarrow{\mathsf{F}}\varphi] \\
\mathsf{G}[q \leftrightarrow \psi \vee \varphi \wedge \overleftarrow{\mathsf{X}}q] &\Leftrightarrow \mathsf{G}[q \leftrightarrow [\varphi\ \overleftarrow{\mathsf{U}}\ \psi]] \\
\mathsf{G}[q \leftrightarrow \psi \vee \varphi \wedge \overleftarrow{\underline{\mathsf{X}}}q] &\Leftrightarrow \mathsf{G}[q \leftrightarrow [\varphi\ \overleftarrow{\underline{\mathsf{U}}}\ \psi]] \\
\mathsf{G}[q \leftrightarrow \neg\psi \wedge (\varphi \vee \overleftarrow{\mathsf{X}}q)] &\Leftrightarrow \mathsf{G}[q \leftrightarrow [\varphi\ \overleftarrow{\mathsf{B}}\ \psi]] \\
\mathsf{G}[q \leftrightarrow \neg\psi \wedge (\varphi \vee \overleftarrow{\underline{\mathsf{X}}}q)] &\Leftrightarrow \mathsf{G}[q \leftrightarrow [\varphi\ \overleftarrow{\underline{\mathsf{B}}}\ \psi]]
\end{aligned}$$

Proof. All implications from right to the left follow simply from Lemma 5.34, since we can replace q with the corresponding temporal subformula. The implications from left to the right for the past time temporal operators simply follow from the uniqueness of primitive recursion. The remaining implications from left to right of the future time operators are more challenging. For the equation $\mathsf{G}[q \leftrightarrow \psi \vee \varphi \wedge \mathsf{X}q]$, we have already given the proof in the previous lemma. We now instantiate $p := \neg q$ in fact (4) of the previous lemma. It then says that if $\mathsf{G}[\neg p \leftrightarrow \neg\psi \wedge (\varphi \vee \neg\mathsf{X}p)]$ holds, then we have either $\mathsf{G}[\neg p \leftrightarrow [\varphi\ \underline{\mathsf{U}}\ \psi]]$ or $\mathsf{G}[\neg p \leftrightarrow [\varphi\ \mathsf{U}\ \psi]]$. Driving the negations first to the right hand side and then inwards, then results in our proposition for the B operator. In the same manner we can reduce the remaining equations to (4) of the previous lemma.

Hence, if the transition relation of an automaton implies for a state variable q that $q \leftrightarrow \psi \vee \varphi \wedge \mathsf{X}q$ invariantly holds, then we can immediately conclude that q behaves either like $[\varphi\ \underline{\mathsf{U}}\ \psi]$ or as $[\varphi\ \mathsf{U}\ \psi]$. Moreover, if the transition relation assures that $\mathsf{G}[q \leftrightarrow \psi \vee \varphi \wedge \overleftarrow{\underline{\mathsf{X}}}q]$ holds, then q will definitely behave exactly as $[\varphi\ \overleftarrow{\underline{\mathsf{U}}}\ \psi]$.

However, there are three problems for our translation procedure: first, the subformulas φ and ψ must be propositional, so that the discussed equations can be used as part of a transition relation of an ω-automaton. This can always be achieved when we work in a bottom-up manner through the syntax tree of the given formula. The second problem is that we are not allowed to use the operators $\overleftarrow{\mathsf{X}}$ and $\overleftarrow{\underline{\mathsf{X}}}$ in the transition relation of an ω-automaton. However, the fixpoint equations for the past time operators, use these operators. Hence, if we reach a subformula $[\varphi\ \overleftarrow{\mathsf{U}}\ \psi]$ with propositional arguments φ and ψ, we need to define a new state variable with the transition relation $q \leftrightarrow \psi \vee \varphi \wedge \overleftarrow{\mathsf{X}}q$ and replace $[\varphi\ \overleftarrow{\mathsf{U}}\ \psi]$ by q. As $\overleftarrow{\mathsf{X}}$ should not occur in the transition relation, we are not allowed to do this. Instead, we 'delay' our state variable q by one point of time, by using $\mathsf{X}q \leftrightarrow \psi \vee \varphi \wedge q$ in the transition

relation. The question is now, what solutions does this new fixpoint equation have. We will see in the next theorem that there are two solutions of this equation, namely $[\varphi \overleftarrow{\mathsf{U}} \psi]$ and $[\varphi \overleftarrow{\underline{\mathsf{U}}} \psi]$. The uniqueness of the solution is now lost, since the equation $\mathsf{X}q \leftrightarrow \psi \vee \varphi \wedge q$ allows us to define q differently at the initial point of time. Before presenting a theorem that states all these facts, we introduce a concise notation for substitution that makes the remainder more readable.

Definition 5.37 (Formula Templates). *Given a formula Φ with some occurrences of a variable x, we abbreviate for convenience $\Phi\langle\varphi\rangle_x := [\Phi]_x^\varphi$.*

The main reason for the introduction of the above abbreviation is to make the following formulas more readable in that a lot of subscripts and superscripts with complex formulas are avoided.

Note that the notation allows us to single out *some* of the occurrences of a subformula: For example, given the formula $\mathsf{G}a \vee \neg\mathsf{G}a$, we can use the templates $\Phi_1 := x \vee \neg\mathsf{G}a$, $\Phi_2 := \mathsf{G}a \vee \neg x$, or $\Phi_3 := x \vee \neg x$ for one or both occurrences of the subformula $\mathsf{G}a$. For example, we then have $\Phi_1\langle\mathsf{F}a\rangle_x := \mathsf{F}a \vee \neg\mathsf{G}a$, $\Phi_2\langle\mathsf{F}a\rangle_x := \mathsf{G}a \vee \neg\mathsf{F}a$, and $\Phi_3\langle\mathsf{F}a\rangle_x := \mathsf{F}a \vee \neg\mathsf{F}a$.

Theorem 5.38 (Defining Past Time Modalities by ω-Automata). *Given a formula $\Phi \in \mathsf{LTL}_p$ with some occurrences of a variable x, and propositional formulas φ and ψ, the following equations are initially valid (we use the notation of Definition 5.37):*

$$
\begin{aligned}
\Phi\langle\overleftarrow{\mathsf{X}}\varphi\rangle_x &\Leftrightarrow \mathcal{A}_\exists\,(\{q\}, q, \mathsf{X}q \leftrightarrow \varphi, \Phi\langle q\rangle_x)\\
\Phi\langle\overleftarrow{\mathsf{X}}\varphi\rangle_x &\Leftrightarrow \mathcal{A}_\exists\,(\{q\}, \neg q, \mathsf{X}q \leftrightarrow \varphi, \Phi\langle q\rangle_x)\\
\Phi\langle\overleftarrow{\mathsf{G}}\varphi\rangle_x &\Leftrightarrow \mathcal{A}_\exists\,(\{q\}, q, \mathsf{X}q \leftrightarrow \varphi \wedge q, \Phi\langle\varphi \wedge q\rangle_x)\\
\Phi\langle\overleftarrow{\mathsf{F}}\varphi\rangle_x &\Leftrightarrow \mathcal{A}_\exists\,(\{q\}, \neg q, \mathsf{X}q \leftrightarrow \varphi \vee q, \Phi\langle\varphi \vee q\rangle_x)\\
\Phi\langle[\varphi \overleftarrow{\mathsf{U}} \psi]\rangle_x &\Leftrightarrow \mathcal{A}_\exists\,(\{q\}, q, \mathsf{X}q \leftrightarrow \psi \vee \varphi \wedge q, \Phi\langle\psi \vee \varphi \wedge q\rangle_x)\\
\Phi\langle[\varphi \overleftarrow{\underline{\mathsf{U}}} \psi]\rangle_x &\Leftrightarrow \mathcal{A}_\exists\,(\{q\}, \neg q, \mathsf{X}q \leftrightarrow \psi \vee \varphi \wedge q, \Phi\langle\psi \vee \varphi \wedge q\rangle_x)\\
\Phi\langle[\varphi \overleftarrow{\mathsf{B}} \psi]\rangle_x &\Leftrightarrow \mathcal{A}_\exists\,(\{q\}, q, \mathsf{X}q \leftrightarrow \neg\psi \wedge (\varphi \vee q), \Phi\langle\neg\psi \wedge (\varphi \vee q)\rangle_x)\\
\Phi\langle[\varphi \overleftarrow{\underline{\mathsf{B}}} \psi]\rangle_x &\Leftrightarrow \mathcal{A}_\exists\,(\{q\}, \neg q, \mathsf{X}q \leftrightarrow \neg\psi \wedge (\varphi \vee q), \Phi\langle\neg\psi \wedge (\varphi \vee q)\rangle_x)
\end{aligned}
$$

Using the above equations in a bottom-up traversal over the syntax tree of a given formula allows us already to translate any temporal logic formula without future time modalities into an equivalent ω-automaton. It is remarkable that all transitions are *deterministic* and that the (intermediate) acceptance condition is only formed by the abbreviated formula $\Phi\langle q\rangle_x$ alone. Using the flattening law below (cf. Lemma 4.9), shows that at the end of our traversal, we end up with an ω-automaton of the form $\mathcal{A}_\exists\,(\{q_1, \ldots, q_n\}, \Phi_\mathcal{I}, \Phi_\mathcal{R}, \Phi)$ where Φ is a propositional formula.

$$\mathcal{A}_\exists\,(Q_\Phi, \Phi_\mathcal{I}, \Phi_\mathcal{R}, \mathcal{A}_\exists\,(Q_\Psi, \Psi_\mathcal{I}, \Psi_\mathcal{R}, \Psi_\mathcal{F})) \Leftrightarrow \mathcal{A}_\exists\,(Q_\Phi \cup Q_\Psi, \Phi_\mathcal{I} \wedge \Psi_\mathcal{I}, \Phi_\mathcal{R} \wedge \Psi_\mathcal{R}, \Psi_\mathcal{F})$$

Note further that $\Phi_{\mathcal{I}}$ is thereby a conjunction of possibly negated state variables q_i and therefore determines exactly one initial state. *Hence, we can translate any temporal logic formula without future time operators into a deterministic ω-automaton with a propositional acceptance condition Φ.* What restrictions are there for an input sequence to be accepted by such an automaton? Clearly, as the automaton is deterministic, a corresponding run through the transition system of the automaton exists, and it is even unique. Hence, the input sequence and its run only need to fulfill the acceptance condition $\Psi_{\mathcal{F}}$ of the automaton, and as $\Psi_{\mathcal{F}}$ is propositional, it only refers to the initial input of the sequence. This is not surprising: as we do not have future operators, we can only refer to the initial point of time, and at that point of time, any formula with past operators is equivalent to a propositional formula (hence, we do not need ω-automata at all in this case).

To extend our translation procedure to more interesting cases, we need to handle future time operators in a similar way, and must therefore face the following problem: Simply abbreviating a subformula as e.g. $[\varphi \mathsf{U} \psi]$ by a new state formula q in that we add the transition relation $q \leftrightarrow \psi \vee \varphi \wedge \mathsf{X}q$ is not enough: According to Theorem 5.36, q can then either behave as $[\varphi \mathsf{U} \psi]$ or as $[\varphi \underline{\mathsf{U}} \psi]$. As the truth value of the formula normally depends on the strength of the temporal operators, we must additionally fix the strength of the temporal expression that has been abbreviated. The next theorem shows that this can be done by adding a suitable fairness constraint of the form $\mathsf{GF}\Psi$.

Theorem 5.39 (Defining Future Time Modalities by ω-Automata). *Given a formula $\Phi \in \mathsf{LTL}_p$ with some occurrences of a variable x, and propositional formulas φ and ψ, the following equations are initially valid (we use the notation of Definition 5.37):*

$$
\begin{aligned}
\Phi\langle \mathsf{X}\varphi\rangle_x &\Leftrightarrow \mathcal{A}_\exists\,(\{q\}, 1, q \leftrightarrow \mathsf{X}\varphi, \Phi\langle q\rangle_x)\\
\Phi\langle \mathsf{X}\varphi\rangle_x &\Leftrightarrow \mathcal{A}_\exists\,(\{q_0, q_1\}, 1, (q_0 \leftrightarrow \varphi) \wedge (q_1 \leftrightarrow \mathsf{X}q_0), \Phi\langle q_1\rangle_x)\\
\Phi\langle \mathsf{G}\varphi\rangle_x &\Leftrightarrow \mathcal{A}_\exists\,(\{q\}, 1, q \leftrightarrow \varphi \wedge \mathsf{X}q, \Phi\langle q\rangle_x \wedge \mathsf{GF}[\varphi \rightarrow q])\\
\Phi\langle \mathsf{F}\varphi\rangle_x &\Leftrightarrow \mathcal{A}_\exists\,(\{q\}, 1, q \leftrightarrow \varphi \vee \mathsf{X}q, \Phi\langle q\rangle_x \wedge \mathsf{GF}[q \rightarrow \varphi])\\
\Phi\langle [\varphi \mathsf{U} \psi]\rangle_x &\Leftrightarrow \mathcal{A}_\exists\,(\{q\}, 1, q \leftrightarrow \psi \vee \varphi \wedge \mathsf{X}q, \Phi\langle q\rangle_x \wedge \mathsf{GF}[\varphi \rightarrow q])\\
\Phi\langle [\varphi \underline{\mathsf{U}} \psi]\rangle_x &\Leftrightarrow \mathcal{A}_\exists\,(\{q\}, 1, q \leftrightarrow \psi \vee \varphi \wedge \mathsf{X}q, \Phi\langle q\rangle_x \wedge \mathsf{GF}[q \rightarrow \psi])\\
\Phi\langle [\varphi \mathsf{B} \psi]\rangle_x &\Leftrightarrow \mathcal{A}_\exists\,(\{q\}, 1, q \leftrightarrow \neg\psi \wedge (\varphi \vee \mathsf{X}q), \Phi\langle q\rangle_x \wedge \mathsf{GF}[q \vee \psi])\\
\Phi\langle [\varphi \underline{\mathsf{B}} \psi]\rangle_x &\Leftrightarrow \mathcal{A}_\exists\,(\{q\}, 1, q \leftrightarrow \neg\psi \wedge (\varphi \vee \mathsf{X}q), \Phi\langle q\rangle_x \wedge \mathsf{GF}[q \rightarrow \varphi])
\end{aligned}
$$

Proof. All implications from the left to the right are simple: Instantiate for q the subformula to be abbreviated and use Lemma 5.34 to prove the transition relation. The formulas that result from the fairness constraints can be easily proved by considering suitable instances of points of time. Hence, consider now the implications from the right to the left. The implications for $\Phi\langle \mathsf{X}\varphi\rangle_x$ are simple. The equations for $\Phi\langle [\varphi \mathsf{U} \psi]\rangle_x$ and $\Phi\langle [\varphi \underline{\mathsf{U}} \psi]\rangle_x$ follow directly from facts (10) and (9) of Lemma 5.35, respectively. The equation for $\Phi\langle [\varphi \underline{\mathsf{B}} \psi]\rangle_x$ and $\Phi\langle [\varphi \mathsf{B} \psi]\rangle_x$ can be reduced to the equations for $\Phi\langle [\varphi \mathsf{U} \psi]\rangle_x$ and $\Phi\langle [\varphi \underline{\mathsf{U}} \psi]\rangle_x$, respectively, by replacing $q := \neg p$ and driving the negation inwards.

The equations of the above theorem show how we can assure that the newly generated variables q_i can be fixed as either the strong or the weak version of an operator by adding additional fairness constraints. These fairness constraints distinguish between the two solutions of the fixpoint equation and safely select one of the two solutions.

While the choice of the transition relation is uniquely determined, we can choose different fairness constraints that might be more or less efficient to check. For example, we could also use $\mathsf{GF}[\neg q \vee \neg\varphi \vee \psi]$ instead of $\mathsf{GF}[q \rightarrow \psi]$ for the abbreviation of $\Phi\langle[\varphi \underline{\mathsf{U}} \psi]\rangle_x$. Also, we could use $\mathsf{GF}[q \vee \neg\varphi \vee \psi]$ instead of $\mathsf{GF}[\varphi \rightarrow q]$ for the abbreviation of $\Phi\langle[\varphi \mathsf{U} \psi]\rangle_x$. It can not be predicted which version of the fairness constraint will be more efficient to prove, but one might speculate that it is a good strategy to choose them as small as possible. This advocates the versions we have chosen.

Another point has to be noted concerning the X operator: Intuitively, one would like to use the first equation $\Phi\langle\mathsf{X}\varphi\rangle_x \Leftrightarrow \mathcal{A}_\exists(\{q\}, 1, q \leftrightarrow \mathsf{X}\varphi, \Phi\langle q\rangle_x)$, which is also valid. However, as φ might also contain occurrences of input variables, we would then refer to the next input in the transition relation of the automaton. In this case, the formula $\mathcal{A}_\exists(\{q\}, 1, q \leftrightarrow \mathsf{X}\varphi, \Phi\langle q\rangle_x)$ is not well-formed. To circumvent this, we introduce two state variables q_0 and q_1 to store the information of the previous input we need to fix the transition relation appropriately.

The last two theorems can be used to translate an arbitrary LTL_p formula Φ to an equivalent ω-automaton $\mathfrak{A}_\Phi$ as given in Figure 5.11. The function implements a bottom-up traversal over the syntax tree and applies the rules of Theorem 5.38 and Theorem 5.39 during this traversal. As we must add fairness constraints, it is unnecessarily difficult to handle them with the intermediate acceptance condition. We have therefore added for convenience another argument to our automaton formulas:

Definition 5.40 (Automaton Formulas with Constraints). *For any finite set of temporal logic formulas $\mathcal{F} \subseteq \mathsf{LTL}_q$, we define*

- $\mathcal{A}_\exists(Q, \mathcal{I}, \mathcal{R}, \mathcal{F}, \Phi) := \mathcal{A}_\exists\left(Q, \mathcal{I}, \mathcal{R}, \Phi \wedge \bigwedge_{\xi\in\mathcal{F}} \xi\right)$
- $\mathcal{A}_\forall(Q, \mathcal{I}, \mathcal{R}, \mathcal{F}, \Phi) := \mathcal{A}_\forall\left(Q, \mathcal{I}, \mathcal{R}, \left(\bigwedge_{\xi\in\mathcal{F}} \xi\right) \rightarrow \Phi\right)$

According to the above definition, the set of constraints $\mathcal{F}$ belongs to the acceptance condition, so that nothing new is achieved by the above definition. For the following, it is however convenient to use the above definition in order to split the acceptance condition into two parts. It is moreover straightforward to define a product operation on automaton formulas with constraints:

Definition 5.41 (Product of Constrained Automaton Formulas). *Given two automaton formulas* $\mathcal{A}_\exists(\mathcal{Q}_1, \mathcal{I}_1, \mathcal{R}_1, \mathcal{F}_1, \mathcal{A}_1)$ *and* $\mathcal{A}_\exists(\mathcal{Q}_2, \mathcal{I}_2, \mathcal{R}_2, \mathcal{F}_2, \mathcal{A}_2)$*, we define their product as follows:*

$$\begin{aligned}&\mathcal{A}_\exists(\mathcal{Q}_1, \mathcal{I}_1, \mathcal{R}_1, \mathcal{F}_1, \mathcal{A}_1) \times \mathcal{A}_\exists(\mathcal{Q}_2, \mathcal{I}_2, \mathcal{R}_2, \mathcal{F}_2, \mathcal{A}_2)\\ &\qquad := \mathcal{A}_\exists(\mathcal{Q}_1 \cup \mathcal{Q}_2, \mathcal{I}_1 \wedge \mathcal{I}_2, \mathcal{R}_1 \wedge \mathcal{R}_2, \mathcal{F}_1 \cup \mathcal{F}_2, \mathcal{A}_1 \wedge \mathcal{A}_2)\end{aligned}$$

We can easily prove that $\mathfrak{A}_1 \times \mathfrak{A}_1 = \mathfrak{A}_1 \wedge \mathfrak{A}_2$ holds. Using this definition, we can now implement our basic translation algorithm as given in Figure 5.11. Obviously, this implementation merely applies the rules of Theorem 5.38 and Theorem 5.39 in a recursive manner. It is thereby important that the algorithm translates the formulas in a bottom-up manner, i.e., it translates first the subformulas of a considered formula into equivalent ω-automata and then constructs an automaton for the entire formula using the equations of Theorem 5.38 and Theorem 5.39. For example, consider how the formula $\left[a \,\underline{\mathsf{U}}\, \left(\overleftarrow{\mathsf{G}}\, [b \,\mathsf{U}\, c]\right)\right]$ is translated by successively applying the rules of the previous theorems:

$$\begin{aligned}&\left[a \,\underline{\mathsf{U}}\, \overleftarrow{\mathsf{G}}\, [b \,\mathsf{U}\, c]\right]\\ &\quad = \mathcal{A}_\exists\left(\{q_1\}, 1, q_1 \leftrightarrow c \vee b \wedge \mathsf{X}q_1, \{\mathsf{GF}[b \rightarrow q_1]\}, \left[a \,\underline{\mathsf{U}}\, \overleftarrow{\mathsf{G}}\, q_1\right]\right)\\ &\quad = \mathcal{A}_\exists\left(\begin{array}{l}\{q_1, q_2\}, q_2, [q_1 \leftrightarrow c \vee b \wedge \mathsf{X}q_1] \wedge [\mathsf{X}q_2 \leftrightarrow q_1 \wedge q_2]\,,\\ \{\mathsf{GF}[b \rightarrow q_1]\}, [a \,\underline{\mathsf{U}}\, q_2]\end{array}\right)\\ &\quad = \mathcal{A}_\exists\left(\begin{array}{l}\{q_1, q_2, q_3\}, q_2,\\ [q_1 \leftrightarrow c \vee b \wedge \mathsf{X}q_1] \wedge [\mathsf{X}q_2 \leftrightarrow q_1 \wedge q_2] \wedge [q_3 \leftrightarrow q_2 \vee a \wedge \mathsf{X}q_3]\,,\\ \{\mathsf{GF}[b \rightarrow q_1], \mathsf{GF}[q_3 \rightarrow q_2],\}, q_3\end{array}\right)\end{aligned}$$

Hence, we see that our translation yields for every future temporal operator (apart from X) one fairness constraint for the $\mathsf{NDet}_{\mathsf{Streett}}$ automaton. Moreover, the acceptance condition that is finally obtained is propositional, but of course the generated fairness constraints must also be considered. The following theorem shows that the resulting automata are really $\mathsf{NDet}_{\mathsf{Streett}}$ automata.

Theorem 5.42 (Translating Temporal Logic to ω-Automata). *Given any* LTL_p *formula* Φ*, and* $\mathcal{A}_\exists(\mathcal{Q}, \mathcal{I}, \mathcal{R}, \mathcal{F}, \Phi_0) = \mathsf{Streett}(\Phi)$*, where* Streett *is defined as in Figure 5.11, then the following holds:*

(1) $\mathsf{Streett}(\Phi)$ *runs in time* $O(|\Phi|)$.
(2) Φ_0 *is propositional*
(3) *each* $\xi \in \mathcal{F}$ *is of the form* $\mathsf{GF}\xi'$ *with a propositional* ξ'
(4) $\Phi \leftrightarrow \mathcal{A}_\exists\left(\mathcal{Q}, \mathcal{I} \wedge \Phi_0, \mathcal{R}, \bigwedge_{\xi \in \mathcal{F}} \xi\right)$ *is initially valid*

Proof. The proof of (1) and (2) easily follow by induction along the structure of Φ. For (1), just note that $|\mathcal{Q}| \leq |\Phi|$, $|\mathcal{I}| \leq |\Phi|$, $|\mathcal{R}| \leq |\Phi|$, $|\mathcal{F}| \leq |\Phi|$,

function Streett(Φ)
  case Φ **of**
    is_prop(Φ): **return** $\mathcal{A}_\exists(\{\}, 1, 1, \{\}, \Phi)$;
    $\neg\varphi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \varphi') \equiv$ Streett(φ);
      return $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \neg\varphi')$;
    $\varphi \wedge \psi$: $\mathcal{A}_\exists(\mathcal{Q}_\Phi, \mathcal{I}_\Phi, \mathcal{R}_\Phi, \mathcal{F}_\Phi, \varphi' \wedge \psi') \equiv$ Streett(φ) $\times$ Streett(ψ);
      return $\mathcal{A}_\exists(\mathcal{Q}_\Phi, \mathcal{I}_\Phi, \mathcal{R}_\Phi, \mathcal{F}_\Phi, \varphi' \wedge \psi')$;
    $\varphi \vee \psi$: $\mathcal{A}_\exists(\mathcal{Q}_\Phi, \mathcal{I}_\Phi, \mathcal{R}_\Phi, \mathcal{F}_\Phi, \varphi' \wedge \psi') \equiv$ Streett(φ) $\times$ Streett(ψ);
      return $\mathcal{A}_\exists(\mathcal{Q}_\Phi, \mathcal{I}_\Phi, \mathcal{R}_\Phi, \mathcal{F}_\Phi, \varphi' \vee \psi')$;
    $\mathsf{X}\varphi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \varphi') \equiv$ Streett(φ); $q :=$ new_var;
      return $\mathcal{A}_\exists(\mathcal{Q}_\varphi \cup \{q\}, \mathcal{I}_\varphi, \mathcal{R}_\varphi \wedge (q \leftrightarrow \mathsf{X}\varphi'), \mathcal{F}_\varphi, q)$;
    $[\varphi \,\underline{\mathsf{U}}\, \psi]$: $\mathcal{A}_\exists(\mathcal{Q}_\Phi, \mathcal{I}_\Phi, \mathcal{R}_\Phi, \mathcal{F}_\Phi, \varphi' \wedge \psi') \equiv$ Streett(φ) $\times$ Streett(ψ);
      $q :=$ new_var; $\mathcal{R}_q := [q \leftrightarrow \psi' \vee \varphi' \wedge \mathsf{X}q]$; $\mathcal{F}_q := \{\mathsf{GF}[q \rightarrow \psi']\}$;
      return $\mathcal{A}_\exists(\mathcal{Q}_\Phi \cup \{q\}, \mathcal{I}_\Phi, \mathcal{R}_\Phi \wedge \mathcal{R}_q, \mathcal{F}_\Phi \cup \mathcal{F}_q, q)$;
    $[\varphi \,\mathsf{B}\, \psi]$: $\mathcal{A}_\exists(\mathcal{Q}_\Phi, \mathcal{I}_\Phi, \mathcal{R}_\Phi, \mathcal{F}_\Phi, \varphi' \wedge \psi') \equiv$ Streett(φ) $\times$ Streett(ψ);
      $q :=$ new_var; $\mathcal{R}_q := [q \leftrightarrow \neg\psi' \wedge (\varphi' \vee \mathsf{X}q)]$; $\mathcal{F}_q := \{\mathsf{GF}[q \vee \psi']\}$;
      return $\mathcal{A}_\exists(\mathcal{Q}_\Phi \cup \{q\}, \mathcal{I}_\Phi, \mathcal{R}_\Phi \wedge \mathcal{R}_q, \mathcal{F}_\Phi \cup \mathcal{F}_q, q)$;
    $\overleftarrow{\mathsf{X}}\varphi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \varphi') \equiv$ Streett(φ); $q :=$ new_var;
      return $\mathcal{A}_\exists(\mathcal{Q}_\varphi \cup \{q\}, \mathcal{I}_\varphi \wedge q, \mathcal{R}_\varphi \wedge (\mathsf{X}q \leftrightarrow \varphi'), \mathcal{F}_\varphi, q)$;
    $\overleftarrow{\underline{\mathsf{X}}}\varphi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \varphi') \equiv$ Streett(φ); $q :=$ new_var;
      return $\mathcal{A}_\exists(\mathcal{Q}_\varphi \cup \{q\}, \mathcal{I}_\varphi \wedge \neg q, \mathcal{R}_\varphi \wedge (\mathsf{X}q \leftrightarrow \varphi'), \mathcal{F}_\varphi, q)$;
    $[\varphi \,\overleftarrow{\underline{\mathsf{U}}}\, \psi]$: $\mathcal{A}_\exists(\mathcal{Q}_\Phi, \mathcal{I}_\Phi, \mathcal{R}_\Phi, \mathcal{F}_\Phi, \varphi' \wedge \psi') \equiv$ Streett(φ) $\times$ Streett(ψ);
      $q :=$ new_var; $r_q := \psi' \vee \varphi' \wedge q$; $\mathcal{R}_q := [\mathsf{X}q \leftrightarrow r_q]$; $\mathcal{I}_q := \neg q$;
      return $\mathcal{A}_\exists(\mathcal{Q}_\Phi \cup \{q\}, \mathcal{I}_\Phi \wedge \mathcal{I}_q, \mathcal{R}_\Phi \wedge \mathcal{R}_q, \mathcal{F}_\Phi, r_q)$;
    $[\varphi \,\overleftarrow{\mathsf{B}}\, \psi]$: $\mathcal{A}_\exists(\mathcal{Q}_\Phi, \mathcal{I}_\Phi, \mathcal{R}_\Phi, \mathcal{F}_\Phi, \varphi' \wedge \psi') \equiv$ Streett(φ) $\times$ Streett(ψ);
      $q :=$ new_var; $r_q := \neg\psi' \wedge (\varphi' \vee q)$; $\mathcal{R}_q := [\mathsf{X}q \leftrightarrow r_q]$; $\mathcal{I}_q := q$;
      return $\mathcal{A}_\exists(\mathcal{Q}_\Phi \cup \{q\}, \mathcal{I}_\Phi \wedge \mathcal{I}_q, \mathcal{R}_\Phi \wedge \mathcal{R}_q, \mathcal{F}_\Phi, r_q)$;
  end case
end function

Fig. 5.11. The basic translation from temporal logic to ω-Automata

$\forall \xi \in \mathcal{F}.\, |\xi| \leq |\Phi|$, and $|\Phi_0| \leq |\Phi|$ holds. (3) is directly seen by observing that the algorithm does only add constraints of the mentioned form to the constraints set. (4) is the essential property to be proved. As already explained in the explanations for the implementation of Streett, it can be simply proved by consulting the equations of Theorem 5.38 and Theorem 5.39 in a recursive manner. Note that the initial validity is sufficient for this purpose. To prove (4), simply recall that there the added transition relations are fixpoint equations with exactly two solutions and that our constraints choose exactly one of these solutions. Hence, any input sequence has exactly one *accepting* run. In general, there may however be further runs that do not satisfy the fairness constraints.

The acceptance condition $\Phi_0 \wedge \bigwedge_{i=1}^{n} \mathsf{GF}\varphi_i$ used in fact (3) is somewhat disturbing since it does not directly fit into the automaton classes we have considered in the previous chapter. As Φ_0 is however a propositional formula, it

only adds an additional restriction for the initial point of time on the input sequence and its run. Hence, it is natural to add it to the initial condition, as shown in the above theorem.

To illustrate the algorithm once more, let us apply the procedure to another example: We want to translate the formula $\mathsf{FG}a \rightarrow \mathsf{GF}a$ to an equivalent ω-automaton.

$$\mathsf{FG}a \rightarrow \mathsf{GF}a = \mathcal{A}_\exists \left(\begin{array}{l} \{q_1, q_2, q_3, q_4\}, q_2 \rightarrow q_4, \\ [q_1 \leftrightarrow a \wedge \mathsf{X}q_1] \wedge [q_2 \leftrightarrow q_1 \vee \mathsf{X}q_2] \wedge \\ [q_3 \leftrightarrow a \vee \mathsf{X}q_3] \wedge [q_4 \leftrightarrow q_3 \wedge \mathsf{X}q_4], \\ [\mathsf{GF}[a \rightarrow q_1]] \wedge [\mathsf{GF}[q_2 \rightarrow q_1]] \wedge \\ [\mathsf{GF}[q_3 \rightarrow a]] \wedge [\mathsf{GF}[q_3 \rightarrow q_4]] \end{array} \right)$$

Our algorithm therefore translates any temporal logic formula to an equivalent $\mathsf{NDet}_{\mathsf{Streett}}$ automaton with a special acceptance condition of the form $\bigwedge_{i=0}^{n} \mathsf{GF}\xi_i$. These automata are sometimes also called generalized Büchi automata, but we prefer to view them as special $\mathsf{NDet}_{\mathsf{Streett}}$ automata. As we have already seen, $\mathsf{NDet}_{\mathsf{Streett}}$ is equal expressive as $\mathsf{NDet}_{\mathsf{GF}}$, we could also reduce the result to $\mathsf{NDet}_{\mathsf{GF}}$. From the viewpoint of verification, the reduction to a Büchi automaton with a single fairness constraint is however not necessary, and even expensive. According to Theorem 3.43 on page 165, we can check the $\bigwedge_{i=0}^{n} \mathsf{GF}\xi_i$ with a fixpoint formula of alternation depth 2 in time $O(f\,|\mathcal{K}|\,|\mathcal{S}|)$.

5.4.2 Exploitation of Monotonicity

The abbreviation rules given in Theorems 5.38 and 5.39 allow us to successively abbreviate elementary subformulas by a new variable that will be used as a state variable of the desired ω-automaton. The state transition relation of these variables is chosen such that the state variable obeys the recursion laws of the corresponding temporal operator, and therefore, as we have already seen by Theorem 5.36 the state variable will either behave like the strong or weak variant of the temporal expression. To fix the operator strength, we have used suitable initial conditions in case of past temporal operators (cf. Theorem 5.38), and fairness constraints in case of future temporal operators (cf. Theorem 5.39). Based on these theorems, the algorithm in Figure 5.11 is therefore able to compute for any LTL_p formula an equivalent $\mathsf{NDet}_{\mathsf{Streett}}$ automaton. *Although this automaton is not deterministic, we emphasize that any input sequence has a unique accepting run. In particular, Φ and $\neg\Phi$ yield in very similar automata, that differ only in their initial conditions (we obtain $\neg\Phi_0$ instead of Φ_0 in Theorem 5.42).* The basic translation procedure as given in Figure 5.11 has however the drawback that it introduces for *every* future time temporal operator (apart from X) a fairness constraint. We will now see that there are situations where the fairness constraint is not necessary, and in the following section, we will furthermore see that the fairness constraints can often be replaced with simpler reachability constraints.

We have already seen by Theorems 5.36 and 5.39 that the fairness constraints are added to fix the strength of the temporal operators that are abbreviated. Consequently, if we simply omit the fairness constraints, then we leave it unspecified whether the abbreviated operator was a weak or strong one. This immediately proves the following equivalences[4]:

$$\begin{aligned}
\mathcal{A}_\exists(\{q\}, 1, q \leftrightarrow \varphi \wedge \mathsf{X}q, \Phi\langle q\rangle_x) &\Leftrightarrow \Phi\langle \mathsf{G}\varphi\rangle_x \vee \Phi\langle 0\rangle_x \\
\mathcal{A}_\exists(\{q\}, 1, q \leftrightarrow \varphi \vee \mathsf{X}q, \Phi\langle q\rangle_x) &\Leftrightarrow \Phi\langle \mathsf{F}\varphi\rangle_x \vee \Phi\langle 1\rangle_x \\
\mathcal{A}_\exists(\{q\}, 1, q \leftrightarrow \psi \vee \varphi \wedge \mathsf{X}q, \Phi\langle q\rangle_x) &\Leftrightarrow \Phi\langle[\varphi\ \mathsf{U}\ \psi]\rangle_x \vee \Phi\langle[\varphi\ \underline{\mathsf{U}}\ \psi]\rangle_x \\
\mathcal{A}_\exists(\{q\}, 1, q \leftrightarrow \neg\psi \wedge (\varphi \vee \mathsf{X}q), \Phi\langle q\rangle_x) &\Leftrightarrow \Phi\langle[\varphi\ \mathsf{B}\ \psi]\rangle_x \vee \Phi\langle[\varphi\ \underline{\mathsf{B}}\ \psi]\rangle_x
\end{aligned}$$

Hence, whenever the strength of the temporal operator that is on top of the subformula Ψ that is to be abbreviated in $\Phi\langle\Psi\rangle_x$ does not influence the truth value of the entire formula, then we can omit the fairness constraints. The question now is when will this be the case.

Note that the propositional formula $(p \rightarrow q) \rightarrow (p \vee q \leftrightarrow q)$ is a tautology and that each strong operator implies the corresponding weak version. Therefore, we conclude by modus ponens that the equations $[\varphi\ \mathsf{U}\ \psi] \vee [\varphi\ \underline{\mathsf{U}}\ \psi] = [\varphi\ \mathsf{U}\ \psi]$ and $[\varphi\ \mathsf{B}\ \psi] \vee [\varphi\ \underline{\mathsf{B}}\ \psi] = [\varphi\ \mathsf{B}\ \psi]$ hold. However, this is not yet sufficient for our purpose, as we should establish equations where formulas as $[\varphi\ \mathsf{U}\ \psi]$ or $[\varphi\ \mathsf{B}\ \psi]$ occur as subformulas in a surrounding formula.

Again regarding the tautology $(p \rightarrow q) \rightarrow (p \vee q \leftrightarrow q)$, we must find criteria that assure that from $\mathsf{G}[y \rightarrow z]$ it follows (1) $\mathsf{G}[\Phi\langle y\rangle_x \rightarrow \Phi\langle z\rangle_x]$ and (2) $\mathsf{G}[\Phi\langle z\rangle_x \rightarrow \Phi\langle y\rangle_x]$. In the first case, we could then conclude that $\Phi\langle[\varphi\ \mathsf{U}\ \psi]\rangle_x \vee \Phi\langle[\varphi\ \underline{\mathsf{U}}\ \psi]\rangle_x = \Phi\langle[\varphi\ \mathsf{U}\ \psi]\rangle_x$ holds and in the second case, we have $\Phi\langle[\varphi\ \mathsf{U}\ \psi]\rangle_x \vee \Phi\langle[\varphi\ \underline{\mathsf{U}}\ \psi]\rangle_x = \Phi\langle[\varphi\ \underline{\mathsf{U}}\ \psi]\rangle_x$ (similar for the other operators). Hence, we must study the monotonicity of the temporal logic operators as given in the next lemma:

Lemma 5.43 (Monotonicity of Temporal Operators). *Given that* $\mathsf{G}[\alpha \rightarrow \underline{\alpha}]$ *and* $\mathsf{G}[\beta \rightarrow \underline{\beta}]$ *holds on a path, then the following implications hold on the path:*

[4] For reasons of completeness, we also mention that the following equations hold for the past operators. These equations show that the initialization of the corresponding state variables are used here to fix the strength of the temporal operators. However, as we already mentioned, there is no need to modify the translation of past time operators.

$$\begin{aligned}
\mathcal{A}_\exists(\{q\}, 1, \mathsf{X}q \leftrightarrow \varphi \wedge q, \Phi\langle q\rangle_x) &\Leftrightarrow \Phi\langle \overleftarrow{\mathsf{X}}\,\overleftarrow{\mathsf{G}}\,\varphi\rangle_x \vee \Phi\langle 0\rangle_x \\
\mathcal{A}_\exists(\{q\}, 1, \mathsf{X}q \leftrightarrow \varphi \vee q, \Phi\langle q\rangle_x) &\Leftrightarrow \Phi\langle \overleftarrow{\underline{\mathsf{X}}}\,\overleftarrow{\mathsf{F}}\,\varphi\rangle_x \vee \Phi\langle 1\rangle_x \\
\mathcal{A}_\exists(\{q\}, 1, \mathsf{X}q \leftrightarrow \psi \vee \varphi \wedge q, \Phi\langle q\rangle_x) &\Leftrightarrow \Phi\langle \overleftarrow{\mathsf{X}}[\varphi\ \overleftarrow{\mathsf{U}}\ \psi]\rangle_x \vee \Phi\langle \overleftarrow{\underline{\mathsf{X}}}[\varphi\ \overleftarrow{\underline{\mathsf{U}}}\ \psi]\rangle_x \\
\mathcal{A}_\exists(\{q\}, 1, \mathsf{X}q \leftrightarrow \neg\psi \wedge (\varphi \vee q), \Phi\langle q\rangle_x) &\Leftrightarrow \Phi\langle \overleftarrow{\mathsf{X}}[\varphi\ \overleftarrow{\mathsf{B}}\ \psi]\rangle_x \vee \Phi\langle \overleftarrow{\underline{\mathsf{X}}}[\varphi\ \overleftarrow{\underline{\mathsf{B}}}\ \psi]\rangle_x
\end{aligned}$$

$$\mathsf{G}[\neg\underline{\alpha} \rightarrow \neg\alpha] \qquad \mathsf{G}[\mathsf{X}\alpha \rightarrow \mathsf{X}\underline{\alpha}]$$
$$\mathsf{G}[\alpha \wedge \beta \rightarrow \underline{\alpha} \wedge \underline{\beta}] \qquad \mathsf{G}[\alpha \vee \beta \rightarrow \underline{\alpha} \vee \underline{\beta}]$$
$$\mathsf{G}[\mathsf{G}\alpha \rightarrow \mathsf{G}\underline{\alpha}] \qquad \mathsf{G}[\mathsf{F}\alpha \rightarrow \mathsf{F}\underline{\alpha}]$$
$$\mathsf{G}[[\alpha\ \mathsf{U}\ \beta] \rightarrow [\underline{\alpha}\ \mathsf{U}\ \underline{\beta}]] \qquad \mathsf{G}[[\alpha\ \underline{\mathsf{U}}\ \beta] \rightarrow [\underline{\alpha}\ \underline{\mathsf{U}}\ \underline{\beta}]]$$
$$\mathsf{G}[[\alpha\ \mathsf{B}\ \underline{\beta}] \rightarrow [\underline{\alpha}\ \mathsf{B}\ \underline{\beta}]] \qquad \mathsf{G}[[\alpha\ \underline{\mathsf{B}}\ \underline{\beta}] \rightarrow [\underline{\alpha}\ \underline{\mathsf{B}}\ \underline{\beta}]]$$
$$\mathsf{G}[\overleftarrow{\mathsf{X}}\alpha \rightarrow \overleftarrow{\mathsf{X}}\underline{\alpha}] \qquad \mathsf{G}[\overleftarrow{\underline{\mathsf{X}}}\alpha \rightarrow \overleftarrow{\underline{\mathsf{X}}}\underline{\alpha}]$$
$$\mathsf{G}[\overleftarrow{\mathsf{G}}\alpha \rightarrow \overleftarrow{\mathsf{G}}\underline{\alpha}] \qquad \mathsf{G}[\overleftarrow{\mathsf{F}}\alpha \rightarrow \overleftarrow{\mathsf{F}}\underline{\alpha}]$$
$$\mathsf{G}[[\alpha\ \overleftarrow{\mathsf{U}}\ \beta] \rightarrow [\underline{\alpha}\ \overleftarrow{\mathsf{U}}\ \underline{\beta}]] \qquad \mathsf{G}[[\alpha\ \overleftarrow{\underline{\mathsf{U}}}\ \beta] \rightarrow [\underline{\alpha}\ \overleftarrow{\underline{\mathsf{U}}}\ \underline{\beta}]]$$
$$\mathsf{G}[[\alpha\ \overleftarrow{\mathsf{B}}\ \underline{\beta}] \rightarrow [\underline{\alpha}\ \overleftarrow{\mathsf{B}}\ \beta]] \qquad \mathsf{G}[[\alpha\ \overleftarrow{\underline{\mathsf{B}}}\ \underline{\beta}] \rightarrow [\underline{\alpha}\ \overleftarrow{\underline{\mathsf{B}}}\ \beta]]$$

We can now use the above laws in a recursive manner to establish our criteria (1) and (2) above. To be precise and concise, we introduce the following notion of positive and negative subformulas in a temporal logic formula.

Definition 5.44 (Positive and Negative Occurrences). *The following rules define the signum of an occurrence of a subformula in a surrounding formula Φ (we furthermore add the fact that Φ itself occurs positively in Φ):*

- *Given that one of the formulas $\mathsf{X}\varphi$, $\mathsf{G}\varphi$, $\mathsf{F}\varphi$, $\overleftarrow{\mathsf{X}}\varphi$, $\overleftarrow{\underline{\mathsf{X}}}\varphi$, $\overleftarrow{\mathsf{G}}\varphi$, and $\overleftarrow{\mathsf{F}}\varphi$ has a positive/negative occurrence in Φ, then the occurrence of φ is positive/negative.*
- *Given that $\neg\varphi$ has a positive/negative occurrence in Φ, then the occurrence of φ is negative/positive.*
- *Given that one of the formulas $\varphi \wedge \psi$, $\varphi \vee \psi$, $\mathsf{X}\varphi$, $\mathsf{G}\varphi$, $\mathsf{F}\varphi$, $[\varphi\ \mathsf{U}\ \psi]$, $[\varphi\ \underline{\mathsf{U}}\ \psi]$, $\overleftarrow{\mathsf{X}}\varphi$, $\overleftarrow{\underline{\mathsf{X}}}\varphi$, $\overleftarrow{\mathsf{G}}\varphi$, $\overleftarrow{\mathsf{F}}\varphi$, $[\varphi\ \overleftarrow{\mathsf{U}}\ \psi]$, $[\varphi\ \overleftarrow{\underline{\mathsf{U}}}\ \psi]$ has a positive/negative occurrence in Φ, then both occurrences of φ and ψ are positive/negative.*
- *Given that one of the formulas $[\varphi\ \mathsf{B}\ \psi]$, $[\varphi\ \underline{\mathsf{B}}\ \psi]$, $[\varphi\ \overleftarrow{\mathsf{B}}\ \psi]$, and $[\varphi\ \overleftarrow{\underline{\mathsf{B}}}\ \psi]$ has a positive/negative occurrence in Φ, then the occurrence of φ is positive/negative, and the occurrence of ψ is negative/positive.*

If all occurrences of a variable x in a template formula Φ are positive/negative, then we write $\Phi\langle\varphi\rangle_x^+/\Phi\langle\varphi\rangle_x^-$ for $\Phi\langle\varphi\rangle_x$.

Using the notion of positive and negative occurrences of subformulas, we can now use the previous lemma in a recursive manner to prove the following important lemma:

Lemma 5.45 (Substitution Lemma). *Given a formula Φ that contains a variable x, the following equations are valid for every path of every structure:*

- *In any case, $\mathsf{G}[\varphi \leftrightarrow \psi]$ implies $\mathsf{G}[\Phi\langle\varphi\rangle_x \leftrightarrow \Phi\langle\psi\rangle_x]$.*
- *If x has only positive occurrences in Φ, then $\mathsf{G}[\varphi \rightarrow \psi]$ implies $\mathsf{G}[\Phi\langle\varphi\rangle_x \rightarrow \Phi\langle\psi\rangle_x]$.*
- *If x has only negative occurrences in Φ, then $\mathsf{G}[\varphi \rightarrow \psi]$ implies $\mathsf{G}[\Phi\langle\psi\rangle_x \rightarrow \Phi\langle\varphi\rangle_x]$.*

The proof of all facts can be obtained by a straightforward induction on the structure of $\Phi\langle x\rangle_x$.

The key observation is therefore that every positive occurrence of a weak temporal future operator and every negative occurrence of a strong temporal future operator can be simply translated into a transition relation without adding initial conditions or further parts for an acceptance condition. Therefore, positive occurrences of weak temporal operators and negative occurrences strong future temporal operators are completely harmless for a translation to equivalent ω-automata: The translation procedure of the previous section can be modified such that, in these cases, no fairness constraint is added. This means that the transformation rules given in Theorem 5.39 for the future time operators are now refined by the following theorem (the rules for the past time operators given in Theorem 5.38 remain unchanged):

Theorem 5.46 (Defining Future Time Modalities w.r.t. Positive and Negative Occurrences). *Given a formula $\Phi \in \mathsf{LTL}_p$ where a variable x has only positive occurrences, and propositional formulas φ and ψ, the following equations are generally valid:*

$$\begin{aligned}
\Phi\langle \mathsf{G}\varphi\rangle_x^+ &\Leftrightarrow \mathcal{A}_\exists\,(\{q\}, 1, q \leftrightarrow \varphi \wedge \mathsf{X}q, \Phi\langle q\rangle_x^+)\\
\Phi\langle [\varphi\ \mathsf{U}\ \psi]\rangle_x^+ &\Leftrightarrow \mathcal{A}_\exists\,(\{q\}, 1, q \leftrightarrow \psi \vee \varphi \wedge \mathsf{X}q, \Phi\langle q\rangle_x^+)\\
\Phi\langle [\varphi\ \mathsf{B}\ \psi]\rangle_x^+ &\Leftrightarrow \mathcal{A}_\exists\,(\{q\}, 1, q \leftrightarrow \neg\psi \wedge (\varphi \vee \mathsf{X}q), \Phi\langle q\rangle_x^+)\\
\Phi\langle [\varphi\ \mathsf{W}\ \psi]\rangle_x^+ &\Leftrightarrow \mathcal{A}_\exists\,(\{q\}, 1, q \leftrightarrow \psi \wedge \varphi \vee \neg\psi \wedge \mathsf{X}q), \Phi\langle q\rangle_x^+)
\end{aligned}$$

Given a formula $\Phi \in \mathsf{LTL}_p$ where a variable x has only negative occurrences, and propositional formulas φ and ψ, the following equations are valid:

$$\begin{aligned}
\Phi\langle \mathsf{F}\varphi\rangle_x^- &\Leftrightarrow \mathcal{A}_\exists\,(\{q\}, 1, q \leftrightarrow \varphi \vee \mathsf{X}q, \Phi\langle q\rangle_x^-)\\
\Phi\langle [\varphi\ \underline{\mathsf{U}}\ \psi]\rangle_x^- &\Leftrightarrow \mathcal{A}_\exists\,(\{q\}, 1, q \leftrightarrow \psi \vee \varphi \wedge \mathsf{X}q, \Phi\langle q\rangle_x^-)\\
\Phi\langle [\varphi\ \underline{\mathsf{B}}\ \psi]\rangle_x^- &\Leftrightarrow \mathcal{A}_\exists\,(\{q\}, 1, q \leftrightarrow \neg\psi \wedge (\varphi \vee \mathsf{X}q), \Phi\langle q\rangle_x^-)\\
\Phi\langle [\varphi\ \underline{\mathsf{W}}\ \psi]\rangle_x^- &\Leftrightarrow \mathcal{A}_\exists\,(\{q\}, 1, q \leftrightarrow \psi \wedge \varphi \vee \neg\psi \wedge \mathsf{X}q), \Phi\langle q\rangle_x^-)
\end{aligned}$$

Proof. We only consider $\Phi\langle [\varphi\ \mathsf{U}\ \psi]\rangle_x^+ \Leftrightarrow \mathcal{A}_\exists\,(\{q\}, 1, q \leftrightarrow \psi \vee \varphi \wedge \mathsf{X}q, \Phi\langle q\rangle_x^+)$ in case that any occurrence of x in Φ is positive. By Lemma 5.45, it therefore follows from $\mathsf{G}[\alpha \rightarrow \beta]$ that $\mathsf{G}[\Phi\langle\alpha\rangle_x^+ \rightarrow \Phi\langle\beta\rangle_x^+]$ holds. This, in turn implies by the tautology $(p \rightarrow q) \rightarrow (p \vee q \leftrightarrow q)$ that $\mathsf{G}[\Phi\langle\alpha\rangle_x^+ \vee \Phi\langle\beta\rangle_x^+ = \Phi\langle\beta\rangle_x^+]$ holds. Using $\alpha := [\varphi\ \underline{\mathsf{U}}\ \psi]$ and $\beta := [\varphi\ \mathsf{U}\ \psi]$ allows us therefore to immediately conclude that $\mathsf{G}[\Phi\langle [\varphi\ \underline{\mathsf{U}}\ \psi]\rangle_x^+ \vee \Phi\langle [\varphi\ \mathsf{U}\ \psi]\rangle_x^+ = \Phi\langle [\varphi\ \mathsf{U}\ \psi]\rangle_x^+]$ holds. The rest follows from Theorem 5.36, because this theorem implies the following equation: $\mathcal{A}_\exists\,(\{q\}, 1, q \leftrightarrow \psi \vee \varphi \wedge \mathsf{X}q, \Phi\langle q\rangle_x^+) \Leftrightarrow \Phi\langle [\varphi\ \mathsf{U}\ \psi]\rangle_x^+ \vee \Phi\langle [\varphi\ \underline{\mathsf{U}}\ \psi]\rangle_x^+$. Considering whether the occurrence of a subformula that is to be abbreviated by a new state variable q is positive or negative allows us therefore to omit the addition of fairness constraints in the above cases. In the other cases, namely when weak operators occur negatively or strong operators occur positively, we still use the rules of Theorem 5.39. Hence, in these cases, fairness constraints are still added. A refinement of the algorithm Figure 5.11 is therefore given in Figure 5.12.

function $\mathsf{TopProp}_\sigma(\Phi)$
case Φ **of**
isProp(Φ): **return** $\mathcal{A}_\exists(\{\},1,1,\{\},\Phi)$;
$\neg\varphi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi,\mathcal{I}_\varphi,\mathcal{R}_\varphi,\mathcal{F}_\varphi,\varphi') \equiv \mathsf{TopProp}_{\neg\sigma}(\varphi)$;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi,\mathcal{I}_\varphi,\mathcal{R}_\varphi,\mathcal{F}_\varphi,\neg\varphi')$;
$\varphi \wedge \psi$: $\mathcal{A}_\exists(\mathcal{Q}_\Phi,\mathcal{I}_\Phi,\mathcal{R}_\Phi,\mathcal{F}_\Phi,\varphi' \wedge \psi') \equiv \mathsf{TopProp}_\sigma(\varphi) \times \mathsf{TopProp}_\sigma(\psi)$;
return $\mathcal{A}_\exists(\mathcal{Q}_\Phi,\mathcal{I}_\Phi,\mathcal{R}_\Phi,\mathcal{F}_\Phi,\varphi' \wedge \psi')$;
$\varphi \vee \psi$: $\mathcal{A}_\exists(\mathcal{Q}_\Phi,\mathcal{I}_\Phi,\mathcal{R}_\Phi,\mathcal{F}_\Phi,\varphi' \wedge \psi') \equiv \mathsf{TopProp}_\sigma(\varphi) \times \mathsf{TopProp}_\sigma(\psi)$;
return $\mathcal{A}_\exists(\mathcal{Q}_\Phi,\mathcal{I}_\Phi,\mathcal{R}_\Phi,\mathcal{F}_\Phi,\varphi' \vee \psi')$;
$\mathsf{X}\varphi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi,\mathcal{I}_\varphi,\mathcal{R}_\varphi,\mathcal{F}_\varphi,\varphi') \equiv \mathsf{TopProp}_\sigma(\varphi)$; $q :=$ new_var;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi \cup \{q\},\mathcal{I}_\varphi,\mathcal{R}_\varphi \wedge (q \leftrightarrow \mathsf{X}\varphi'),\mathcal{F}_\varphi,q)$;
$[\varphi\ \mathsf{U}\ \psi]$: $\mathcal{A}_\exists(\mathcal{Q}_\Phi,\mathcal{I}_\Phi,\mathcal{R}_\Phi,\mathcal{F}_\Phi,\varphi' \wedge \psi') \equiv \mathsf{TopProp}_\sigma(\varphi) \times \mathsf{TopProp}_\sigma(\psi)$;
$q :=$ new_var; $\mathcal{R}_q := [q \leftrightarrow \psi' \vee \varphi' \wedge \mathsf{X}q]$;
$\mathcal{F}_q :=$ **if** σ **then** $\{\mathsf{GF}[q \to \psi']\}$ **else** $\{\}$;
return $\mathcal{A}_\exists(\mathcal{Q}_\Phi \cup \{q\},\mathcal{I}_\Phi,\mathcal{R}_\Phi \wedge \mathcal{R}_q,\mathcal{F}_\Phi \cup \mathcal{F}_q,q)$;
$[\varphi\ \mathsf{B}\ \psi]$: $\mathcal{A}_\exists(\mathcal{Q}_\Phi,\mathcal{I}_\Phi,\mathcal{R}_\Phi,\mathcal{F}_\Phi,\varphi' \wedge \psi') \equiv \mathsf{TopProp}_\sigma(\varphi) \times \mathsf{TopProp}_{\neg\sigma}(\psi)$;
$q :=$ new_var; $\mathcal{R}_q := [q \leftrightarrow \neg\psi' \wedge (\varphi' \vee \mathsf{X}q)]$;
$\mathcal{F}_q :=$ **if** σ **then** $\{\}$ **else** $\{\mathsf{GF}[q \vee \psi']\}$;
return $\mathcal{A}_\exists(\mathcal{Q}_\Phi \cup \{q\},\mathcal{I}_\Phi,\mathcal{R}_\Phi \wedge \mathcal{R}_q,\mathcal{F}_\Phi \cup \mathcal{F}_q,q)$;
$\overleftarrow{\mathsf{X}}\varphi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi,\mathcal{I}_\varphi,\mathcal{R}_\varphi,\mathcal{F}_\varphi,\varphi') \equiv \mathsf{TopProp}_\sigma(\varphi)$; $q :=$ new_var;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi \cup \{q\},\mathcal{I}_\varphi \wedge q,\mathcal{R}_\varphi \wedge (\mathsf{X}q \leftrightarrow \varphi'),\mathcal{F}_\varphi,q)$;
$\overleftarrow{\mathsf{X}}\varphi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi,\mathcal{I}_\varphi,\mathcal{R}_\varphi,\mathcal{F}_\varphi,\varphi') \equiv \mathsf{TopProp}_\sigma(\varphi)$; $q :=$ new_var;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi \cup \{q\},\mathcal{I}_\varphi \wedge \neg q,\mathcal{R}_\varphi \wedge (\mathsf{X}q \leftrightarrow \varphi'),\mathcal{F}_\varphi,q)$;
$[\varphi\ \overleftarrow{\mathsf{U}}\ \psi]$: $\mathcal{A}_\exists(\mathcal{Q}_\Phi,\mathcal{I}_\Phi,\mathcal{R}_\Phi,\mathcal{F}_\Phi,\varphi' \wedge \psi') \equiv \mathsf{TopProp}_\sigma(\varphi) \times \mathsf{TopProp}_\sigma(\psi)$;
$q :=$ new_var; $r_q := \psi' \vee \varphi' \wedge q$; $\mathcal{R}_q := [\mathsf{X}q \leftrightarrow r_q]$;
return $\mathcal{A}_\exists(\mathcal{Q}_\Phi \cup \{q\},\mathcal{I}_\Phi \wedge \neg q,\mathcal{R}_\Phi \wedge \mathcal{R}_q,\mathcal{F}_\Phi,r_q)$;
$[\varphi\ \overleftarrow{\mathsf{B}}\ \psi]$: $\mathcal{A}_\exists(\mathcal{Q}_\Phi,\mathcal{I}_\Phi,\mathcal{R}_\Phi,\mathcal{F}_\Phi,\varphi' \wedge \psi') \equiv \mathsf{TopProp}_\sigma(\varphi) \times \mathsf{TopProp}_{\neg\sigma}(\psi)$;
$q :=$ new_var; $r_q := \neg\psi' \wedge (\varphi' \vee q)$; $\mathcal{R}_q := [\mathsf{X}q \leftrightarrow r_q]$;
return $\mathcal{A}_\exists(\mathcal{Q}_\Phi \cup \{q\},\mathcal{I}_\Phi \wedge q,\mathcal{R}_\Phi \wedge \mathcal{R}_q,\mathcal{F}_\Phi,r_q)$;
end case
end function

Fig. 5.12. Translation of LTL_p improved by monotonicity laws

Theorem 5.47 (Correctness of $\mathsf{TopProp}_\sigma(\Phi)$). *Given any formula $\Phi \in \mathsf{LTL}_p$, and the automaton formula $\mathcal{A}_\exists(\mathcal{Q},\mathcal{I},\mathcal{R},\mathcal{F},\Phi_0) = \mathsf{TopProp}_\sigma(\Phi)$ obtained by the algorithm* TopProp *given in Figure 5.12, then the following holds:*

(1) $\mathsf{TopProp}_\sigma(\Phi)$ *runs in time $O(|\Phi|)$.*
(2) *Φ_0 is propositional*
(3) *Each $\xi \in \mathcal{F}$ is of the form $\mathsf{GF}\xi'$ where ξ' is propositional.*
(4) *For $\sigma = 1$, the equation $\Phi \leftrightarrow \mathcal{A}_\exists\left(\mathcal{Q},\Phi_0 \wedge \mathcal{I},\mathcal{R},\bigwedge_{\xi\in\mathcal{F}}\xi\right)$ is initially valid.*
(5) *For $\sigma = 0$, the equation $\neg\Phi \leftrightarrow \mathcal{A}_\exists\left(\mathcal{Q},\neg\Phi_0 \wedge \mathcal{I},\mathcal{R},\bigwedge_{\xi\in\mathcal{F}}\xi\right)$ is initially valid.*

Note that the refined algorithm given in Figure 5.12 differs from the one given in Figure 5.11 only in that the signum of the currently considered subformula is given as a further argument σ, and that the fairness constraints for the future time temporal operators are added only when necessary according to the above theorem. The proof is essentially the same as for the Streett algorithm, however for fact (4), we additionally consult Theorem 5.46. (5) is easily seen by considering what the algorithm does with negations: One can easily see that $\mathcal{A}_\exists(\mathcal{Q}, \mathcal{I}, \mathcal{R}, \mathcal{F}, \neg\Phi_0) = \mathsf{TopProp}_\sigma(\neg\Phi)$ holds iff $\mathcal{A}_\exists(\mathcal{Q}, \mathcal{I}, \mathcal{R}, \mathcal{F}, \Phi_0) = \mathsf{TopProp}_{\neg\sigma}(\Phi)$ holds.

5.4.3 Borel Classes of Temporal Logic

We have seen in the last two sections how arbitrary LTL_p formulas can be translated to equivalent $\mathsf{NDet}_{\mathsf{Streett}}$ automata. We already know that there are weaker classes of ω-automata, but we also know that the temporal logic formula $\mathsf{GF}\varphi \vee \mathsf{FG}\psi$ can not be expressed by the lower Borel classes. Hence, for the translation of arbitrary LTL_p formulas, we really need the most powerful Borel class $\mathsf{NDet}_{\mathsf{Streett}}$. It is nevertheless reasonable to ask for temporal logics that correspond with the lower Borel classes. In particular, it is important to know the logics that correspond with the classes Det_G, Det_F, $\mathsf{Det}_{\mathsf{Prefix}}$, and $\mathsf{Det}_{\mathsf{FG}}$, because these classes can be translated to the alternation-free μ-calculus, while the remaining classes $\mathsf{Det}_{\mathsf{GF}}$ and $\mathsf{NDet}_{\mathsf{Streett}}$ require μ-calculus formulas of alternation depth 2.

In this section, we will therefore define temporal logics TL_κ in analogy to the automaton classes Det_κ for $\kappa \in \{\mathsf{G}, \mathsf{F}, \mathsf{Prefix}, \mathsf{FG}, \mathsf{GF}, \mathsf{Streett}\}$. After this, we will prove some simple properties of these logics. For example, we will see that TL_G and TL_F, and that also $\mathsf{TL}_{\mathsf{GF}}$ and $\mathsf{TL}_{\mathsf{FG}}$ are dual to each other (this means that the negations of one class are in the dual class). Moreover, we will see that $\mathsf{TL}_{\mathsf{Prefix}}$ is the Boolean closure of both TL_G and TL_F, and that $\mathsf{TL}_{\mathsf{Streett}}$ is the Boolean closure of both $\mathsf{TL}_{\mathsf{GF}}$ and $\mathsf{TL}_{\mathsf{FG}}$.

Furthermore, we will see that our translation function TopProp is already able to translate TL_G to NDet_G. As TL_F is dual to TL_G, and as NDet_G is equally expressive to Det_G, this immediately implies that TL_F can be translated to Det_F. However, this indirect translation of TL_F to Det_F requires an exponential runtime due to the determinization step. In the next section, we will also develop further techniques to construct a linear time translation procedure from TL_F to NDet_F.

To conclude this section, we consider extraction procedures that are able to split any temporal logic formula into a part that belongs to one of the classes TL_κ, and another part that is translated with TopProp. These extraction procedures work roughly as follows: They follow the grammar rules of the sublogics TL_κ, and abbreviate each subformula that violates these grammar rules by applying the function TopProp to that subformula. This also explains the name of the previous function TopProp. It is also an extraction

procedure that extracts the largest top-level propositional part. The other extraction functions are defined in a similar way.

To start with, consider the following definition of the temporal logic Borel classes that are defined in analogy to the automaton hierarchy:

Definition 5.48 (Temporal Borel Classes). *We define the logics* TL_G *and* TL_F *by the following grammar rules, where* TL_G *and* TL_F *is the set of formulas that can be derived from the nonterminal* P_G *and* P_F*, respectively:*

$$\begin{array}{l|l}
P_\mathsf{G} ::= V_\Sigma & P_\mathsf{F} ::= V_\Sigma \\
\quad \mid \neg P_\mathsf{F} \mid P_\mathsf{G} \wedge P_\mathsf{G} \mid P_\mathsf{G} \vee P_\mathsf{G} & \quad \mid \neg P_\mathsf{G} \mid P_\mathsf{F} \wedge P_\mathsf{F} \mid P_\mathsf{F} \vee P_\mathsf{F} \\
\quad \mid \mathsf{X} P_\mathsf{G} \mid [P_\mathsf{G}\ \mathsf{U}\ P_\mathsf{G}] \mid [P_\mathsf{G}\ \mathsf{B}\ P_\mathsf{F}] & \quad \mid \mathsf{X} P_\mathsf{F} \mid [P_\mathsf{F}\ \underline{\mathsf{U}}\ P_\mathsf{F}] \mid [P_\mathsf{F}\ \underline{\mathsf{B}}\ P_\mathsf{G}] \\
\quad \mid \overleftarrow{\mathsf{X}} P_\mathsf{G} \mid [P_\mathsf{G}\ \overleftarrow{\mathsf{U}}\ P_\mathsf{G}] \mid [P_\mathsf{G}\ \overleftarrow{\mathsf{B}}\ P_\mathsf{F}] & \quad \mid \overleftarrow{\mathsf{X}} P_\mathsf{F} \mid [P_\mathsf{F}\ \overleftarrow{\underline{\mathsf{U}}}\ P_\mathsf{F}] \mid [P_\mathsf{F}\ \overleftarrow{\underline{\mathsf{B}}}\ P_\mathsf{G}] \\
\quad \mid \overleftarrow{\mathsf{X}} P_\mathsf{G} \mid [P_\mathsf{G}\ \overleftarrow{\mathsf{U}}\ P_\mathsf{G}] \mid [P_\mathsf{G}\ \overleftarrow{\mathsf{B}}\ P_\mathsf{F}] & \quad \mid \overleftarrow{\mathsf{X}} P_\mathsf{F} \mid [P_\mathsf{F}\ \overleftarrow{\mathsf{U}}\ P_\mathsf{F}] \mid [P_\mathsf{F}\ \overleftarrow{\mathsf{B}}\ P_\mathsf{G}]
\end{array}$$

The temporal logic $\mathsf{TL}_{\mathsf{Prefix}}$ *is the set of temporal logic formulas that can be derived from the nonterminal* P_{Prefix} *with the following additional grammar rules:*

$$P_{\mathsf{Prefix}} ::= P_\mathsf{G} \mid P_\mathsf{F} \mid \neg P_{\mathsf{Prefix}} \mid P_{\mathsf{Prefix}} \wedge P_{\mathsf{Prefix}} \mid P_{\mathsf{Prefix}} \vee P_{\mathsf{Prefix}}$$

Moreover, we define the logics TL_GF *and* TL_FG *by the following grammar rules, where* TL_GF *and* TL_FG *are the set of formulas that can be derived from the nonterminal* P_GF *and* P_FG*, respectively:*

$$\begin{array}{l|l}
P_\mathsf{GF} ::= V_\Sigma & P_\mathsf{FG} ::= V_\Sigma \\
\quad \mid \neg P_\mathsf{FG} \mid P_\mathsf{GF} \wedge P_\mathsf{GF} \mid P_\mathsf{GF} \vee P_\mathsf{GF} & \quad \mid \neg P_\mathsf{GF} \mid P_\mathsf{FG} \wedge P_\mathsf{FG} \mid P_\mathsf{FG} \vee P_\mathsf{FG} \\
\quad \mid \mathsf{X} P_\mathsf{GF} \mid [P_\mathsf{GF}\ \underline{\mathsf{U}}\ P_\mathsf{F}] \mid [P_\mathsf{F}\ \underline{\mathsf{B}}\ P_\mathsf{FG}] & \quad \mid \mathsf{X} P_\mathsf{FG} \mid [P_\mathsf{FG}\ \underline{\mathsf{U}}\ P_\mathsf{FG}] \mid [P_\mathsf{FG}\ \underline{\mathsf{B}}\ P_\mathsf{GF}] \\
\quad \mid [P_\mathsf{GF}\ \mathsf{U}\ P_\mathsf{GF}] \mid [P_\mathsf{GF}\ \mathsf{B}\ P_\mathsf{FG}] & \quad \mid [P_\mathsf{G}\ \mathsf{U}\ P_\mathsf{FG}] \mid [P_\mathsf{FG}\ \mathsf{B}\ P_\mathsf{F}] \\
\quad \mid \overleftarrow{\mathsf{X}} P_\mathsf{GF} \mid [P_\mathsf{GF}\ \overleftarrow{\underline{\mathsf{U}}}\ P_\mathsf{GF}] \mid [P_\mathsf{GF}\ \overleftarrow{\underline{\mathsf{B}}}\ P_\mathsf{FG}] & \quad \mid \overleftarrow{\mathsf{X}} P_\mathsf{FG} \mid [P_\mathsf{FG}\ \overleftarrow{\underline{\mathsf{U}}}\ P_\mathsf{FG}] \mid [P_\mathsf{FG}\ \overleftarrow{\underline{\mathsf{B}}}\ P_\mathsf{GF}] \\
\quad \mid \overleftarrow{\mathsf{X}} P_\mathsf{GF} \mid [P_\mathsf{GF}\ \overleftarrow{\mathsf{U}}\ P_\mathsf{GF}] \mid [P_\mathsf{GF}\ \overleftarrow{\mathsf{B}}\ P_\mathsf{FG}] & \quad \mid \overleftarrow{\mathsf{X}} P_\mathsf{FG} \mid [P_\mathsf{FG}\ \overleftarrow{\mathsf{U}}\ P_\mathsf{FG}] \mid [P_\mathsf{FG}\ \overleftarrow{\mathsf{B}}\ P_\mathsf{GF}]
\end{array}$$

Finally, $\mathsf{TL}_{\mathsf{Streett}}$ *is the set of temporal logic formulas that can be derived from the nonterminal* P_{Streett} *with the following additional grammar rules:*

$$P_{\mathsf{Streett}} ::= P_\mathsf{GF} \mid P_\mathsf{FG} \mid \neg P_{\mathsf{Streett}} \mid P_{\mathsf{Streett}} \wedge P_{\mathsf{Streett}} \mid P_{\mathsf{Streett}} \vee P_{\mathsf{Streett}}$$

It is easily seen that every occurrence of a subformula of a TL_G formula that is derived from the nonterminal P_G has a positive occurrence. Analogously, any occurrence of a subformula that is derived from the nonterminal P_F in a TL_G formula is negative. This is easily verified by induction on the structure of TL_G and TL_F formulas. Moreover, we have the following properties:

Lemma 5.49 (Relationship between Borel Classes). *The following properties hold for the temporal Borel classes:*

(1) $\Phi \in \mathsf{TL}_\mathsf{F}$ *iff* $\neg\Phi \in \mathsf{TL}_\mathsf{G}$ *and* $\Phi \in \mathsf{TL}_\mathsf{G}$ *iff* $\neg\Phi \in \mathsf{TL}_\mathsf{F}$
(2) $\Phi \in \mathsf{TL}_\mathsf{FG}$ *iff* $\neg\Phi \in \mathsf{TL}_\mathsf{GF}$ *and* $\Phi \in \mathsf{TL}_\mathsf{GF}$ *iff* $\neg\Phi \in \mathsf{TL}_\mathsf{FG}$

(3) $\mathsf{TL}_\mathsf{F} \subseteq \mathsf{TL}_\mathsf{FG}$ *and* $\mathsf{TL}_\mathsf{G} \subseteq \mathsf{TL}_\mathsf{GF}$
(4) $\mathsf{TL}_\mathsf{G} \subseteq \mathsf{TL}_\mathsf{FG}$ *and* $\mathsf{TL}_\mathsf{F} \subseteq \mathsf{TL}_\mathsf{GF}$
(5) $\mathsf{TL}_\mathsf{Prefix} \subseteq \mathsf{TL}_\mathsf{FG}$ *and* $\mathsf{TL}_\mathsf{Prefix} \subseteq \mathsf{TL}_\mathsf{GF}$, *hence* $\mathsf{TL}_\mathsf{Prefix} \subseteq \mathsf{TL}_\mathsf{FG} \cap \mathsf{TL}_\mathsf{GF}$

Proof. (1), (2), (3), and (4) can be easily proved by induction along the formulas. Using (3) and (4), a further induction along the grammar of $\mathsf{TL}_\mathsf{Prefix}$ then proves that (5) holds. □

(1) states the duality of the logics TL_F and TL_G, and (2) states the duality of the logics TL_FG and TL_GF. (3) and (4) shows that $\mathsf{TL}_\mathsf{G} \cup \mathsf{TL}_\mathsf{F} \subseteq \mathsf{TL}_\mathsf{FG} \cap \mathsf{TL}_\mathsf{GF}$ holds. Moreover, it is easily seen that $\mathsf{TL}_\mathsf{Prefix}$ is the Boolean closure of both TL_F and TL_G, and (5) states that even $\mathsf{TL}_\mathsf{Prefix}$ is contained in $\mathsf{TL}_\mathsf{FG} \cap \mathsf{TL}_\mathsf{GF}$. In a similar way, it is easily seen that $\mathsf{TL}_\mathsf{Streett}$ is the Boolean closure of both TL_FG and TL_GF. Therefore, we have the following lemma:

Lemma 5.50 (Boolean Closures of TL_κ). *Each of the logics* TL_κ *for* $\kappa \in \{\mathsf{G}, \mathsf{F}, \mathsf{Prefix}, \mathsf{GF}, \mathsf{FG}, \mathsf{Streett}\}$ *is closed under conjunction and disjunction. Moreover,* $\mathsf{TL}_\mathsf{Prefix}$ *and* $\mathsf{TL}_\mathsf{Streett}$ *are closed under negation, and are therefore the Boolean closures of* $\mathsf{TL}_\mathsf{G}/\mathsf{TL}_\mathsf{F}$ *and* $\mathsf{TL}_\mathsf{FG}/\mathsf{TL}_\mathsf{GF}$, *respectively.*

The proofs are immediately seen by considering the grammar rules of the logics. It is furthermore remarkable that we can convert each formula of any of the classes into negation normal form without leaving the considered class. This property may be used to eliminate the mutual recursive definition of the classes.

Lemma 5.51 (Robustness of TL_κ under Negation Normal Form). *For any formula from the logics* TL_κ *for* $\kappa \in \{\mathsf{G}, \mathsf{F}, \mathsf{Prefix}, \mathsf{GF}, \mathsf{FG}, \mathsf{Streett}\}$ *there is an equivalent formula of the same class in negation normal form.*

We now turn to the translation of the logics TL_κ to their corresponding automaton classes Det_κ. For this reason, note again that any occurrence of a subformula of a TL_G formula that is derived from the nonterminal P_G has a positive occurrence, and that any occurrence of a subformula that is derived from the nonterminal P_F in a TL_G formula is negative. Therefore, it immediately follows that each TL_G will be translated by function TopProp of Figure 5.12 to an ω-automaton without any fairness or other constraint. This, together with the previous lemmas yield the following result on the translateability:

Theorem 5.52 (Temporal Borel Classes (I)). *The following holds for the temporal logics* TL_G, TL_F, *and* $\mathsf{TL}_\mathsf{Prefix}$:

- *For any formula* $\Phi \in \mathsf{TL}_\mathsf{G}$, *there is an equivalent* ω*-automaton* $\mathfrak{A}_\Phi \in \mathsf{Det}_\mathsf{G}$.
- *For any formula* $\Phi \in \mathsf{TL}_\mathsf{F}$, *there is an equivalent* ω*-automaton* $\mathfrak{A}_\Phi \in \mathsf{Det}_\mathsf{F}$.
- *For any formula* $\Phi \in \mathsf{TL}_\mathsf{Prefix}$, *there is an equivalent* ω*-automaton* $\mathfrak{A}_\Phi \in \mathsf{Det}_\mathsf{Prefix}$.

Proof. The first proposition is easily shown by induction on the given $\mathsf{TL_G}$ formula, where we can assume by the previous lemma that the given formula is given in negation normal form. Moreover, we can replace B operators by corresponding U operators and still preserve membership in $\mathsf{TL_G}$. The same holds for the strong before and the past time before operators. According to Theorem 5.47, we can use the algorithm of Figure 5.12, to obtain an automaton formula $\mathcal{A}_\exists\,(Q, \mathcal{I} \wedge \Phi_0, \Phi_\mathcal{R}, \bigwedge_{i=0}^{n} \xi_i)$. Due to the grammar rules of $\mathsf{TL_G}$, we can however easily see that $n = 0$ holds, i.e., that no fairness constraints were generated. Hence, the automaton formula is a (special) $\mathsf{NDet_G}$ automaton (one might add the 'constraint' G1). Note finally, that $\mathsf{NDet_G}$ is equally expressive as $\mathsf{Det_G}$.

The second proposition follows by duality of $\mathsf{TL_G}$ and $\mathsf{TL_F}$, and the duality of $\mathsf{Det_G}$ with $\mathsf{Det_F}$. The third proposition is then proved by an easy induction on the given $\mathsf{TL_{Prefix}}$ formula Φ: The base cases where $\Phi \in \mathsf{TL_G}$ or $\Phi \in \mathsf{TL_F}$ holds follow immediately from the first two propositions, since $\mathsf{Det_G} \cup \mathsf{Det_F} \subset \mathsf{Det_{Prefix}}$ holds. The remaining cases follow by the Boolean closure of $\mathsf{Det_{Prefix}}$. □

Hence, we have already seen the correspondence of the logics $\mathsf{TL_G}$, $\mathsf{TL_F}$, and $\mathsf{TL_{Prefix}}$ with the automaton classes $\mathsf{Det_G}$, $\mathsf{Det_F}$, and $\mathsf{Det_{Prefix}}$. Using the function TopProp of Figure 5.12, gives us an algorithm for translating $\mathsf{TL_G}$ to $\mathsf{NDet_G}$. However, our translation from $\mathsf{TL_F}$ to $\mathsf{Det_F}$ is still indirect: We translate the negated formula which belongs to $\mathsf{TL_G}$, determinize the automaton and negate it. Clearly, this procedures suffers from an exponential blow-up due to the determinization. We will find another algorithm for directly translating $\mathsf{TL_F}$ to $\mathsf{NDet_F}$ within only linear runtime that is similar to TopProp in the next section.

Consider now the logics $\mathsf{TL_{FG}}$ and $\mathsf{TL_{GF}}$. These logics have been defined similar to the logics $\mathsf{TL_F}$ and $\mathsf{TL_G}$. To make this correspondence more apparent, note that we could first replace the rules $P_\mathsf{GF} ::= V_\Sigma$ and $P_\mathsf{FG} ::= V_\Sigma$ with $P_\mathsf{GF} ::= P_\mathsf{Prefix}$ and $P_\mathsf{FG} ::= P_\mathsf{Prefix}$, since $\mathsf{TL_{Prefix}} \subseteq \mathsf{TL_{FG}} \cap \mathsf{TL_{GF}}$. After this, we can eliminate the rules $P_\mathsf{FG} ::= [P_\mathsf{G}\ \mathsf{U}\ P_\mathsf{FG}]$ and $P_\mathsf{FG} ::= [P_\mathsf{FG}\ \mathsf{B}\ P_\mathsf{F}]$ as shown below:

- $[P_\mathsf{G}\ \mathsf{U}\ P_\mathsf{FG}] = \underbrace{[P_\mathsf{G}\ \underline{\mathsf{U}}\ P_\mathsf{FG}]}_{\in \mathsf{TL_{FG}}} \vee \underbrace{[P_\mathsf{G}\ \mathsf{U}\ 0]}_{\in \mathsf{TL_G} \subseteq \mathsf{TL_{Prefix}}}$
- $[P_\mathsf{FG}\ \mathsf{B}\ P_\mathsf{F}] = \underbrace{[P_\mathsf{FG}\ \underline{\mathsf{B}}\ P_\mathsf{F}]}_{\in \mathsf{TL_{FG}}} \vee \underbrace{[(\neg P_\mathsf{F})\ \mathsf{U}\ 0]}_{\in \mathsf{TL_G} \subseteq \mathsf{TL_{Prefix}}}$

Eliminating these rules (and the corresponding two of the grammar rules of P_GF) shows the correspondence between $\mathsf{TL_{FG}}$ and $\mathsf{TL_{GF}}$ with $\mathsf{TL_F}$ and $\mathsf{TL_G}$, respectively. The difference is then that only P_FG and P_GF accept P_Prefix formulas where P_F and P_G require variables.

Hence, $\mathsf{TL_{FG}}$ allows arbitrary positive nestings with strong future and arbitrary past operators, but not with weak future operators: once a weak future operator is used in a positive occurrence, this must be $\mathsf{TL_G}$ formula,

where only weak future and arbitrary past operators may be nested. Note that if we were to allow strong operators to appear as arguments of weak operators, we could express the formula $\mathsf{GF}\varphi$ that can not be expressed by any $\mathsf{Det}_{\mathsf{FG}}$ automaton.

Analogously, $\mathsf{TL}_{\mathsf{GF}}$ allows arbitrary positive nestings with weak future and arbitrary past operators, but not with strong future operators: once a strong future operator is used in a positive occurrence, this must be a TL_{F} formula, where only strong future and arbitrary past operators may be nested. In particular, these restriction do not allow us to generate the formula $\mathsf{FG}\varphi$ that can not be expressed by any $\mathsf{Det}_{\mathsf{GF}}$ automaton.

Before considering the efficient translations of all the logics TL_κ to NDet_κ, we consider how top-level formulas of these classes can be extracted from arbitrary LTL_p formulas. For this reason, we implement the functions TopFG and TopG as given in Figures 5.14 and 5.13. The principle of these functions is the same: They follow the grammar rules of the sublogics, and abbreviate each subformula that violates these grammar rules by applying the function TopProp. Hence, the functions TopFG, TopG, and TopProp are closely related to each other in that they extract the largest top-level $\mathsf{TL}_{\mathsf{FG}}$, TL_{G}, and propositional formulas. The precise specification of the functions TopFG and TopG is given in the following theorem.

Theorem 5.53 (Correctness of TopFG and TopG). *Given any formula $\Phi \in \mathsf{LTL}_p$, and the resulting automaton formula $\mathcal{A}_\exists(\mathcal{Q}, \mathcal{I}, \mathcal{R}, \mathcal{F}, \Phi_0) = \mathsf{TopG}_\sigma(\Phi)$ obtained by the algorithm given in Figure 5.13, then the following holds:*

(1) $\mathsf{TopG}_\sigma(\Phi)$ *runs in time* $O(|\Phi|)$.
(2) *If* $\sigma = 1$ *holds, we have* $\Phi_0 \in \mathsf{TL}_{\mathsf{G}}$, *otherwise, we have* $\Phi_0 \in \mathsf{TL}_{\mathsf{F}}$.
(3) *Each* $\xi \in \mathcal{F}$ *is of the form* $\mathsf{GF}\xi'$ *where* ξ' *is propositional.*
(4) *For* $\sigma = 1$, *the equation* $\Phi = \mathcal{A}_\exists\left(\mathcal{Q}, \mathcal{I}, \mathcal{R}, \Phi_0 \wedge \bigwedge_{\xi \in \mathcal{F}} \xi\right)$ *is initially valid.*
(5) *For* $\sigma = 0$, *the equation* $\neg\Phi = \mathcal{A}_\exists\left(\mathcal{Q}, \mathcal{I}, \mathcal{R}, \neg\Phi_0 \wedge \bigwedge_{\xi \in \mathcal{F}} \xi\right)$ *is initially valid.*

Given any formula $\Phi \in \mathsf{LTL}_p$, and the automaton formula $\mathcal{A}_\exists(\mathcal{Q}, \mathcal{I}, \mathcal{R}, \mathcal{F}, \Phi_0) = \mathsf{TopFG}_\sigma(\Phi)$ obtained by the algorithm given in Figure 5.14, then the following holds:

(1) $\mathsf{TopFG}_\sigma(\Phi)$ *runs in time* $O(|\Phi|)$.
(2) *If* $\sigma = 1$ *holds, we have* $\Phi_0 \in \mathsf{TL}_{\mathsf{FG}}$, *otherwise, we have* $\Phi_0 \in \mathsf{TL}_{\mathsf{GF}}$.
(3) *Each* ξ_i *is of the form* $\mathsf{GF}\xi'$ *where* ξ' *is propositional.*
(4) *For* $\sigma = 1$, *the equation* $\Phi = \mathcal{A}_\exists\left(\mathcal{Q}, \mathcal{I}, \mathcal{R}, \Phi_0 \wedge \bigwedge_{\xi \in \mathcal{F}} \xi\right)$ *is initially valid.*
(5) *For* $\sigma = 0$, *the equation* $\neg\Phi = \mathcal{A}_\exists\left(\mathcal{Q}, \mathcal{I}, \mathcal{R}, \neg\Phi_0 \wedge \bigwedge_{\xi \in \mathcal{F}} \xi\right)$ *is initially valid.*

function $\mathsf{TopG}_\sigma(\Phi)$
case Φ **of**
is_prop(Φ): **return** $\mathcal{A}_\exists(\{\}, 1, 1, \{\}, \Phi)$;
$\neg\varphi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \varphi') \equiv \mathsf{TopG}_{\neg\sigma}(\varphi)$;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \neg\varphi')$;
$\varphi \wedge \psi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \varphi') \equiv \mathsf{TopG}_\sigma(\varphi)$;
$\mathcal{A}_\exists(\mathcal{Q}_\psi, \mathcal{I}_\psi, \mathcal{R}_\psi, \mathcal{F}_\psi, \psi') \equiv \mathsf{TopG}_\sigma(\psi)$;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi \cup \mathcal{Q}_\psi, \mathcal{I}_\varphi \wedge \mathcal{I}_\psi, \mathcal{R}_\varphi \wedge \mathcal{R}_\psi, \mathcal{F}_\varphi \cup \mathcal{F}_\psi, \varphi' \wedge \psi')$;
$\varphi \vee \psi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \varphi') \equiv \mathsf{TopG}_\sigma(\varphi)$;
$\mathcal{A}_\exists(\mathcal{Q}_\psi, \mathcal{I}_\psi, \mathcal{R}_\psi, \mathcal{F}_\psi, \psi') \equiv \mathsf{TopG}_\sigma(\psi)$;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi \cup \mathcal{Q}_\psi, \mathcal{I}_\varphi \wedge \mathcal{I}_\psi, \mathcal{R}_\varphi \wedge \mathcal{R}_\psi, \mathcal{F}_\varphi \cup \mathcal{F}_\psi, \varphi' \vee \psi')$;
$\mathsf{X}\varphi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \varphi') \equiv \mathsf{TopG}_\sigma(\varphi)$;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \mathsf{X}\varphi')$;
$[\varphi\ \underline{\mathsf{U}}\ \psi]$: **if** σ **then return** $\mathsf{TopProp}_\sigma(\Phi)$ **end**;
$\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \varphi') \equiv \mathsf{TopG}_\sigma(\varphi)$;
$\mathcal{A}_\exists(\mathcal{Q}_\psi, \mathcal{I}_\psi, \mathcal{R}_\psi, \mathcal{F}_\psi, \psi') \equiv \mathsf{TopG}_\sigma(\psi)$;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi \cup \mathcal{Q}_\psi, \mathcal{I}_\varphi \wedge \mathcal{I}_\psi, \mathcal{R}_\varphi \wedge \mathcal{R}_\psi, \mathcal{F}_\varphi \cup \mathcal{F}_\psi, [\varphi'\ \underline{\mathsf{U}}\ \psi'])$;
$[\varphi\ \mathsf{B}\ \psi]$: **if** $\neg\sigma$ **then return** $\mathsf{TopProp}_\sigma(\Phi)$ **end**;
$\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \varphi') \equiv \mathsf{TopG}_\sigma(\varphi)$;
$\mathcal{A}_\exists(\mathcal{Q}_\psi, \mathcal{I}_\psi, \mathcal{R}_\psi, \mathcal{F}_\psi, \psi') \equiv \mathsf{TopG}_{\neg\sigma}(\psi)$;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi \cup \mathcal{Q}_\psi, \mathcal{I}_\varphi \wedge \mathcal{I}_\psi, \mathcal{R}_\varphi \wedge \mathcal{R}_\psi, \mathcal{F}_\varphi \cup \mathcal{F}_\psi, [\varphi'\ \mathsf{B}\ \psi'])$;
$\overleftarrow{\mathsf{X}}\varphi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \varphi') \equiv \mathsf{TopG}_\sigma(\varphi)$;
return $\mathcal{A}_\exists\left(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \overleftarrow{\mathsf{X}}\varphi'\right)$;
$\overleftarrow{\underline{\mathsf{X}}}\varphi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \varphi') \equiv \mathsf{TopG}_\sigma(\varphi)$;
return $\mathcal{A}_\exists\left(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \overleftarrow{\underline{\mathsf{X}}}\varphi'\right)$;
$[\varphi\ \overleftarrow{\mathsf{U}}\ \psi]$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \varphi') \equiv \mathsf{TopG}_\sigma(\varphi)$;
$\mathcal{A}_\exists(\mathcal{Q}_\psi, \mathcal{I}_\psi, \mathcal{R}_\psi, \mathcal{F}_\psi, \psi') \equiv \mathsf{TopG}_\sigma(\psi)$;
return $\mathcal{A}_\exists\left(\mathcal{Q}_\varphi \cup \mathcal{Q}_\psi, \mathcal{I}_\varphi \wedge \mathcal{I}_\psi, \mathcal{R}_\varphi \wedge \mathcal{R}_\psi, \mathcal{F}_\varphi \cup \mathcal{F}_\psi, [\varphi'\ \overleftarrow{\mathsf{U}}\ \psi']\right)$;
$[\varphi\ \overleftarrow{\mathsf{B}}\ \psi]$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \varphi') \equiv \mathsf{TopG}_\sigma(\varphi)$;
$\mathcal{A}_\exists(\mathcal{Q}_\psi, \mathcal{I}_\psi, \mathcal{R}_\psi, \mathcal{F}_\psi, \psi') \equiv \mathsf{TopG}_{\neg\sigma}(\psi)$;
return $\mathcal{A}_\exists\left(\mathcal{Q}_\varphi \cup \mathcal{Q}_\psi, \mathcal{I}_\varphi \wedge \mathcal{I}_\psi, \mathcal{R}_\varphi \wedge \mathcal{R}_\psi, \mathcal{F}_\varphi \cup \mathcal{F}_\psi, [\varphi'\ \overleftarrow{\mathsf{B}}\ \psi']\right)$;
end case
end function

Fig. 5.13. Extraction of top-level TL_G ($\sigma = 1$) and TL_F ($\sigma = 0$) formulas

The proof of the above theorem is not too difficult: Properties $(1-3)$ are easily proved by induction. Properties $(4-5)$ are proved in the same manner as the correctness of TopProp was proved, i.e., by means of Theorems 5.38, 5.39, and 5.46.

One might wonder whether in the definition of TL_FG, the grammar rules $P_\mathsf{FG} := [P_\mathsf{G}\ \mathsf{U}\ P_\mathsf{FG}]$ and $P_\mathsf{FG} := [P_\mathsf{FG}\ \mathsf{B}\ P_\mathsf{F}]$ could be generalized to $P_\mathsf{FG} := [P_\mathsf{FG}\ \mathsf{U}\ P_\mathsf{FG}]$ and $P_\mathsf{FG} := [P_\mathsf{FG}\ \mathsf{B}\ P_\mathsf{FG}]$. This is not possible since the formula $[[1\ \underline{\mathsf{U}}\ \varphi]\ \mathsf{U}\ 0]$ would then belong to TL_FG. However, as this formula is equiva-

function $\mathsf{TopFG}_\sigma(\Phi)$
case Φ **of**
is_prop(Φ): **return** $\mathcal{A}_\exists(\{\},1,1,\{\},\Phi)$;
$\neg\varphi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi,\mathcal{I}_\varphi,\mathcal{R}_\varphi,\mathcal{F}_\varphi,\varphi') \equiv \mathsf{TopFG}_{\neg\sigma}(\varphi)$;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi,\mathcal{I}_\varphi,\mathcal{R}_\varphi,\mathcal{F}_\varphi,\neg\varphi')$;
$\varphi \wedge \psi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi,\mathcal{I}_\varphi,\mathcal{R}_\varphi,\mathcal{F}_\varphi,\varphi') \equiv \mathsf{TopFG}_\sigma(\varphi)$;
$\mathcal{A}_\exists(\mathcal{Q}_\psi,\mathcal{I}_\psi,\mathcal{R}_\psi,\mathcal{F}_\psi,\psi') \equiv \mathsf{TopFG}_\sigma(\psi)$;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi \cup \mathcal{Q}_\psi, \mathcal{I}_\varphi \wedge \mathcal{I}_\psi, \mathcal{R}_\varphi \wedge \mathcal{R}_\psi, \mathcal{F}_\varphi \cup \mathcal{F}_\psi, \varphi' \wedge \psi')$;
$\varphi \vee \psi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi,\mathcal{I}_\varphi,\mathcal{R}_\varphi,\mathcal{F}_\varphi,\varphi') \equiv \mathsf{TopFG}_\sigma(\varphi)$;
$\mathcal{A}_\exists(\mathcal{Q}_\psi,\mathcal{I}_\psi,\mathcal{R}_\psi,\mathcal{F}_\psi,\psi') \equiv \mathsf{TopFG}_\sigma(\psi)$;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi \cup \mathcal{Q}_\psi, \mathcal{I}_\varphi \wedge \mathcal{I}_\psi, \mathcal{R}_\varphi \wedge \mathcal{R}_\psi, \mathcal{F}_\varphi \cup \mathcal{F}_\psi, \varphi' \vee \psi')$;
$\mathsf{X}\varphi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi,\mathcal{I}_\varphi,\mathcal{R}_\varphi,\mathcal{F}_\varphi,\varphi') \equiv \mathsf{TopFG}_\sigma(\varphi)$;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi,\mathcal{I}_\varphi,\mathcal{R}_\varphi,\mathcal{F}_\varphi,\mathsf{X}\varphi')$;
$[\varphi\ \underline{\mathsf{U}}\ \psi]$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi,\mathcal{I}_\varphi,\mathcal{R}_\varphi,\mathcal{F}_\varphi,\varphi') \equiv \mathsf{TopFG}_\sigma(\varphi)$;
if σ **then** $\mathcal{A}_\exists(\mathcal{Q}_\psi,\mathcal{I}_\psi,\mathcal{R}_\psi,\mathcal{F}_\psi,\psi') \equiv \mathsf{TopFG}_\sigma(\psi)$
else $\mathcal{A}_\exists(\mathcal{Q}_\psi,\mathcal{I}_\psi,\mathcal{R}_\psi,\mathcal{F}_\psi,\psi') \equiv \mathsf{TopG}_\sigma(\psi)$ **end**;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi \cup \mathcal{Q}_\psi, \mathcal{I}_\varphi \wedge \mathcal{I}_\psi, \mathcal{R}_\varphi \wedge \mathcal{R}_\psi, \mathcal{F}_\varphi \cup \mathcal{F}_\psi, [\varphi'\ \underline{\mathsf{U}}\ \psi'])$;
$[\varphi\ \mathsf{B}\ \psi]$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi,\mathcal{I}_\varphi,\mathcal{R}_\varphi,\mathcal{F}_\varphi,\varphi') \equiv \mathsf{TopFG}_\sigma(\varphi)$;
if σ **then** $\mathcal{A}_\exists(\mathcal{Q}_\psi,\mathcal{I}_\psi,\mathcal{R}_\psi,\mathcal{F}_\psi,\psi') \equiv \mathsf{TopG}_{\neg\sigma}(\psi)$
else $\mathcal{A}_\exists(\mathcal{Q}_\psi,\mathcal{I}_\psi,\mathcal{R}_\psi,\mathcal{F}_\psi,\psi') \equiv \mathsf{TopFG}_{\neg\sigma}(\psi)$ **end**;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi \cup \mathcal{Q}_\psi, \mathcal{I}_\varphi \wedge \mathcal{I}_\psi, \mathcal{R}_\varphi \wedge \mathcal{R}_\psi, \mathcal{F}_\varphi \cup \mathcal{F}_\psi, [\varphi'\ \mathsf{B}\ \psi'])$;
$\overleftarrow{\mathsf{X}}\varphi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi,\mathcal{I}_\varphi,\mathcal{R}_\varphi,\mathcal{F}_\varphi,\varphi') \equiv \mathsf{TopFG}_\sigma(\varphi)$;
return $\mathcal{A}_\exists\left(\mathcal{Q}_\varphi,\mathcal{I}_\varphi,\mathcal{R}_\varphi,\mathcal{F}_\varphi,\overleftarrow{\mathsf{X}}\varphi'\right)$;
$\overleftarrow{\underline{\mathsf{X}}}\varphi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi,\mathcal{I}_\varphi,\mathcal{R}_\varphi,\mathcal{F}_\varphi,\varphi') \equiv \mathsf{TopFG}_\sigma(\varphi)$;
return $\mathcal{A}_\exists\left(\mathcal{Q}_\varphi,\mathcal{I}_\varphi,\mathcal{R}_\varphi,\mathcal{F}_\varphi,\overleftarrow{\underline{\mathsf{X}}}\varphi'\right)$;
$[\varphi\ \overleftarrow{\underline{\mathsf{U}}}\ \psi]$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi,\mathcal{I}_\varphi,\mathcal{R}_\varphi,\mathcal{F}_\varphi,\varphi') \equiv \mathsf{TopFG}_\sigma(\varphi)$;
$\mathcal{A}_\exists(\mathcal{Q}_\psi,\mathcal{I}_\psi,\mathcal{R}_\psi,\mathcal{F}_\psi,\psi') \equiv \mathsf{TopFG}_\sigma(\psi)$;
return $\mathcal{A}_\exists\left(\mathcal{Q}_\varphi \cup \mathcal{Q}_\psi, \mathcal{I}_\varphi \wedge \mathcal{I}_\psi, \mathcal{R}_\varphi \wedge \mathcal{R}_\psi, \mathcal{F}_\varphi \cup \mathcal{F}_\psi, [\varphi'\ \overleftarrow{\underline{\mathsf{U}}}\ \psi']\right)$;
$[\varphi\ \overleftarrow{\mathsf{B}}\ \psi]$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi,\mathcal{I}_\varphi,\mathcal{R}_\varphi,\mathcal{F}_\varphi,\varphi') \equiv \mathsf{TopFG}_\sigma(\varphi)$;
$\mathcal{A}_\exists(\mathcal{Q}_\psi,\mathcal{I}_\psi,\mathcal{R}_\psi,\mathcal{F}_\psi,\psi') \equiv \mathsf{TopFG}_{\neg\sigma}(\psi)$;
return $\mathcal{A}_\exists\left(\mathcal{Q}_\varphi \cup \mathcal{Q}_\psi, \mathcal{I}_\varphi \wedge \mathcal{I}_\psi, \mathcal{R}_\varphi \wedge \mathcal{R}_\psi, \mathcal{F}_\varphi \cup \mathcal{F}_\psi, [\varphi'\ \overleftarrow{\mathsf{B}}\ \psi']\right)$;
end case
end function

Fig. 5.14. Extraction of top-level $\mathsf{TL}_{\mathsf{FG}}$ ($\sigma = 1$) and $\mathsf{TL}_{\mathsf{GF}}$ ($\sigma = 0$) formulas

lent to $\mathsf{GF}\varphi$, and the latter can not be expressed by $\mathsf{Det}_{\mathsf{FG}}$ automata, this generalization would really extend the expressiveness of $\mathsf{TL}_{\mathsf{FG}}$, so that it could no longer be translated to $\mathsf{Det}_{\mathsf{FG}}$. The same holds for the impossible generalizations $P_{\mathsf{FG}} := [P_{\mathsf{Prefix}}\ \mathsf{U}\ P_{\mathsf{FG}}]$ and $P_{\mathsf{FG}} := [P_{\mathsf{FG}}\ \mathsf{B}\ P_{\mathsf{Prefix}}]$.

In fact, we will prove in Section 5.5.3 that our definitions are complete in some sense, i.e., the logics that we have defined are essentially the strongest that can be translated to the desired automaton classes. Before we proceed

with considerations of the completeness, we should however construct further translation procedures for the remaining classes.

5.4.4 Reducing Temporal Borel Classes to Borel Automata

We have seen in Section 5.4.2 that the basic translation as given in Section 5.4.1 can be improved when the monotonicity of operators is taken into account. In the previous section, we have moreover defined a hierarchy of temporal logics TL_κ that is analogous with the automaton hierarchy. Moreover, we have already seen in the previous section that the logics TL_G, TL_F, and $\mathsf{TL}_{\mathsf{Prefix}}$ can be translated to equivalent Det_G, Det_F, and $\mathsf{Det}_{\mathsf{Prefix}}$ automata. Moreover, the function TopProp translates any TL_G formula in linear time to an equivalent NDet_G automaton, so that we have an efficient translation for this logic. Our current translation from TL_F to NDet_F is, however, indirect: We first negate the TL_F formula Φ to obtain the TL_G formula $\neg\Phi$, which is then translated by the function TopProp to a NDet_G automaton $\mathfrak{A}_{\neg\Phi}$. Up to this point, all computations are done within $O(|\Phi|)$ time, but now our current translation needs to determinize $\mathfrak{A}_{\neg\Phi}$, so that we can compute its complement. This complement is then our desired Det_F automaton. The translation of $\mathsf{TL}_{\mathsf{Prefix}}$ to $\mathsf{Det}_{\mathsf{Prefix}}$ is done in the same spirit, where we have to additionally compute the Boolean closures of the obtained Det_G and Det_F automata.

In this section, we will develop another translation that does not suffer from the exponential blow-up due to the determinization. This translation however, only translates TL_F to NDet_F instead of translating it to Det_F, but this is completely sufficient for verification purposes. The translation that we will now develop is a modification of the function TopProp. Recall that the key idea of TopProp is to abbreviate elementary subformulas by state variables of the ω-automaton to be constructed. As the abbreviations of the future operators are only determined up to the operator strength with the transition relations, the function TopProp uses fairness constraints to fix the desired operator strength when necessary.

We will now see that for TL_F formulas, a simper *reachability constraint* of the form $\mathsf{F}\varphi$ can be used instead of a fairness constraint to fix the operator strength. The key idea is as follows: Assume that $[\varphi \mathbin{\underline{\mathsf{U}}} \psi]$ holds at some point of time t_0. To check this, we must only inspect finitely many points of time, namely those from t_0 up to the first point of time after t_0, where ψ holds. In this interval, we must evaluate the subformulas φ and ψ, and need to check if the semantics of $[\varphi \mathbin{\underline{\mathsf{U}}} \psi]$ are respected. Hence, it is *not necessary* to define state variables q_φ and q_ψ such that we have $\mathsf{G}[q_\varphi \leftrightarrow \varphi]$ and $\mathsf{G}[q_\psi \leftrightarrow \psi]$. Instead, it would be sufficient if the equations $q_\varphi \leftrightarrow \varphi$ would *hold long enough* so that we can evaluate the formula $[\varphi \mathbin{\underline{\mathsf{U}}} \psi]$. The syntax of TL_F and TL_FG guarantees that no outer operator requires the infinitely often checking of truth values. It is sufficient to evaluate the subformulas in a finite interval.

For example, consider a formula of the form $[(\Phi\langle[\varphi \mathbin{\underline{\mathsf{U}}} \psi]\rangle_x) \mathbin{\underline{\mathsf{U}}} \gamma]$, where the formulas φ and ψ are propositional. Furthermore, assume that x does not

occur in the scope of a future temporal operator inside the formula Φ. As in the previous algorithms, we abbreviate the subformula $[\varphi \underline{\mathsf{U}} \psi]$ with a new state variable q by using the transition relation $q \leftrightarrow \psi \vee \varphi \wedge \mathsf{X}q$. We already know (cf. Theorem 5.36) that this restricts q so that either $\mathsf{G}[q \leftrightarrow [\varphi \mathsf{U} \psi]]$ or $\mathsf{G}[q \leftrightarrow [\varphi \underline{\mathsf{U}} \psi]]$ holds, and thus the behavior of q is not yet uniquely fixed. As already outlined above, the crucial observation is that it is not necessary to determine the unique behavior of q for *all points of time*. Instead, it is *sufficient to uniquely determine it up to the point of time where γ holds for the first time.* Note that the latter would not be sufficient if x would occur in the scope of a future temporal operator, as e.g., in $[(\mathsf{G}x) \underline{\mathsf{U}} \gamma]$.

How can we do this without adding fairness constraints? According to fact (6) of Lemma 5.35 (on page 333), we already know that if $\mathsf{F}[q \rightarrow \psi]$ holds at some point of time t_0, then all solutions of the fixpoint equation $\mathsf{G}[q \leftrightarrow \psi \vee \varphi \wedge \mathsf{X}q]$ have the same truth value at least up to this point of time t_0. In particular, we then have $\overleftarrow{\mathsf{G}}[q \leftrightarrow [\varphi \underline{\mathsf{U}} \psi]]$. Let t_1 be the first point of time where γ becomes true. Hence, we must establish in some way that $\mathsf{F}[q \rightarrow \psi]$ holds at t_1 (or after t_1, which implies that it will also hold at t_1). A simple attempt could therefore be to use the formula $[(\Phi\langle q \rangle_x) \underline{\mathsf{U}} (\gamma \wedge \mathsf{F}[q \rightarrow \psi])]$, and a detailed proof will show that it is correct.

We note again that it is necessary that x must not occur in the scope of a future time temporal operator in Φ: The key idea of the above theorem is that we need to evaluate q only in a finite interval, i.e., up to some point of time. However, iff x were to occur in the scope of a future time temporal operator in Φ, then we require it to correct the operator strength for all times, which must then be done by a fairness constraint instead.

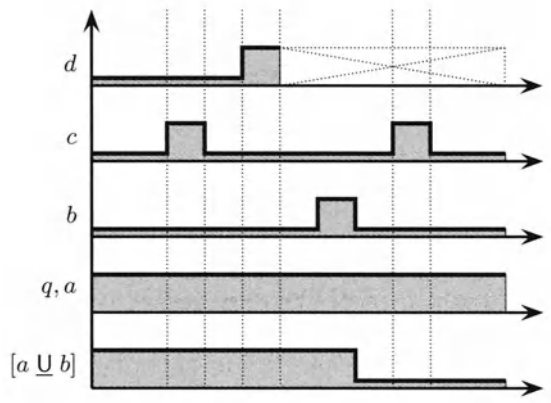


Fig. 5.15. An example to show the extension of finite intervals of interest

However, even if the interval of interest remains finite, it may be extended by operator nestings. To see this, consider the formula $[[[a \underline{\mathsf{U}} b] \underline{\mathsf{U}} c] \underline{\mathsf{U}} d]$ where we assume that a, b, c, and d are variables. Of course, we start by

abbreviating $[a \underline{\mathsf{U}}\, b]$ with a new state variable q in that we add the transition relation $q \leftrightarrow b \vee a \wedge \mathsf{X}q$. To fix the operator strength, we must now establish that the constraint $\mathsf{F}[q \rightarrow b]$ holds at the end of the finite interval where we need the truth value of q. Now, note that it is not sufficient to add the constraint that follows:

$$\mathcal{A}_\exists\,(\{q\}, 1, q \leftrightarrow b \vee a \wedge \mathsf{X}q, [[q \underline{\mathsf{U}}\, c]\ \underline{\mathsf{U}}\ (d \wedge \mathsf{F}[q \rightarrow b])])$$

Our constraint has been put at the first point of time where d holds, and therefore, we can guarantee that $q \leftrightarrow [a \underline{\mathsf{U}}\, b]$ holds up to this point of time. However, the further nesting in another temporal operator shows that this is not sufficient: If we assume that a, b, c, and d, have the values as given in the timing diagram of Figure 5.15, then we see that $q \leftrightarrow [a \underline{\mathsf{U}}\, b]$ holds up to the first point after t_2 where b holds, but after that, we only have $q \leftrightarrow [a\ \mathsf{U}\ b]$. The equation $q \leftrightarrow [a \underline{\mathsf{U}}\, b]$ should however hold up to t_3, so that we see that we must place the constraint at point t_3. Therefore, the constraint $\mathsf{F}[q \rightarrow b]$ must be put in the next upper event that is awaited, i.e., our intermediate result should be the following automaton formula:

$$\mathcal{A}_\exists\,(\{q\}, 1, q \leftrightarrow b \vee a \wedge \mathsf{X}q, [[q \underline{\mathsf{U}}\ (c \wedge \mathsf{F}[q \rightarrow b])]\ \underline{\mathsf{U}}\, d])$$

After that, we would like to further abbreviate $[q \underline{\mathsf{U}}\ (c \wedge \mathsf{F}[q \rightarrow b])]$ with another state variable. As the subformulas are not propositional, we would need to abbreviate $\mathsf{F}[q \rightarrow b]$ in advance, which yields in an infinite loop (we abbreviate a reachability constraint by another reachability constraint that also needs to be abbreviated, and so on).

We will see in the following that this problem can be circumvented. The key idea is thereby the same as the one used by function TopProp, i.e., we traverse the syntax tree of the formula in a bottom-up manner. This means, we first translate the subformulas of a formula to an equivalent ω-automaton and construct then the entire automaton. However, we can not simply drop the G operators of the GF-constraints of Theorem 5.39 and use the resulting equations. As we have seen by the above example, we must nest the generated constraints in the same way as the formulas are nested where they stem from.

Hence, a first attempt is to assume that we already have proved that the equations $\varphi \leftrightarrow \mathcal{A}_\exists\,(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \varphi')$ and $\psi \leftrightarrow \mathcal{A}_\exists\,(\mathcal{Q}_\psi, \mathcal{I}_\psi, \mathcal{R}_\psi, \mathcal{F}_\psi, \psi')$, where we may assume that the constraint sets $\mathcal{F}_\varphi$ and $\mathcal{F}_\psi$ contain reachability constraints of the form $\mathsf{F}\xi'$ with propositional ξ'. The task is now to generate an automaton that is equivalent to $[\varphi \underline{\mathsf{U}}\, \psi]$ out of the already available automata. Due to the presence of past operators we will only be able to generate automata that are initially equivalent to the original formula. This is sufficient for verification purposes, but prevents us from nesting the automata: To translate the formula $[\varphi \underline{\mathsf{U}}\, \psi]$, we need to establish the equivalence of φ to an automaton not only for the initial point of time, but for a finite interval that is determined by ψ. Also, the initial equivalence of ψ with

an automaton is not of much use, since we need the equation for later points of time.

We are therefore interested in proving the initial validity of automaton formulas whose acceptance condition is of the form $\zeta_\Phi \rightarrow \overleftarrow{\mathsf{G}}[\Phi_0 \leftrightarrow \Phi]$, where ζ_Φ is an appropriate reachability constraint. This will allow us to generate, more or less, the same automata with the acceptance condition ζ_Φ. For a detailed proof, we will however need to construct universal automata. As universal automaton formulas may however trivially hold in the case where there is no product path in the product structure, we must additionally prove the existence of product paths. Therefore, we need the following lemma to establish the invariants of our algorithm:

Lemma 5.54 (Fixing Operator Strength by F-Constraints (I)). *The following implications are initially valid and show how one can recursively construct universal ω-automata from TL_F formulas:*

$$\bullet\quad \left(\begin{array}{l} \mathsf{G}(\varphi \rightarrow \varphi_0) \wedge \mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathsf{G}\zeta_\varphi) \wedge \\ \mathsf{G}(\psi \rightarrow \psi_0) \wedge \mathcal{A}_\exists(\mathcal{Q}_\psi, \mathcal{I}_\psi, \mathcal{R}_\psi, \mathsf{G}\zeta_\psi) \wedge \\ \mathcal{A}_\forall\left(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathsf{G}\left(\zeta_\varphi \rightarrow \overleftarrow{\mathsf{G}}[\varphi_0 \leftrightarrow \varphi]\right)\right) \wedge \\ \mathcal{A}_\forall\left(\mathcal{Q}_\psi, \mathcal{I}_\psi, \mathcal{R}_\psi, \mathsf{G}\left(\zeta_\psi \rightarrow \overleftarrow{\mathsf{G}}[\psi_0 \leftrightarrow \psi]\right)\right) \wedge \\ \rightarrow \mathcal{A}_\forall\left(\begin{array}{l} \mathcal{Q}_\varphi \cup \mathcal{Q}_\psi \cup \{q\}, \mathcal{I}_\varphi \wedge \mathcal{I}_\psi, \\ \mathcal{R}_\varphi \wedge \mathcal{R}_\psi \wedge (q \leftrightarrow \psi_0 \vee \varphi_0 \wedge \mathsf{X}q), \\ \mathsf{G}\left(\mathsf{F}([q \rightarrow \psi_0] \wedge \zeta_\varphi \wedge \zeta_\psi) \rightarrow \overleftarrow{\mathsf{G}}(q \leftrightarrow [\varphi \,\underline{\mathsf{U}}\, \psi])\right) \end{array}\right) \end{array}\right)$$

$$\bullet\quad \left(\begin{array}{l} \mathsf{G}(\varphi_0 \rightarrow \varphi) \wedge \mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathsf{G}\zeta_\varphi) \wedge \\ \mathsf{G}(\psi_0 \rightarrow \psi) \wedge \mathcal{A}_\exists(\mathcal{Q}_\psi, \mathcal{I}_\psi, \mathcal{R}_\psi, \mathsf{G}\zeta_\psi) \wedge \\ \mathcal{A}_\forall\left(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathsf{G}\left(\zeta_\varphi \rightarrow \overleftarrow{\mathsf{G}}[\varphi_0 \leftrightarrow \varphi]\right)\right) \wedge \\ \mathcal{A}_\forall\left(\mathcal{Q}_\psi, \mathcal{I}_\psi, \mathcal{R}_\psi, \mathsf{G}\left(\zeta_\psi \rightarrow \overleftarrow{\mathsf{G}}[\psi_0 \leftrightarrow \psi]\right)\right) \wedge \\ \rightarrow \mathcal{A}_\forall\left(\begin{array}{l} \mathcal{Q}_\varphi \cup \mathcal{Q}_\psi \cup \{q\}, \mathcal{I}_\varphi \wedge \mathcal{I}_\psi, \\ \mathcal{R}_\varphi \wedge \mathcal{R}_\psi \wedge (q \leftrightarrow \psi_0 \vee \varphi_0 \wedge \mathsf{X}q), \\ \mathsf{G}\left(\mathsf{F}([q \vee \neg\varphi_0] \wedge \zeta_\varphi \wedge \zeta_\psi) \rightarrow \overleftarrow{\mathsf{G}}(q \leftrightarrow [\varphi \,\mathsf{U}\, \psi])\right) \end{array}\right) \end{array}\right)$$

$$\bullet\quad \left(\begin{array}{l} \mathsf{G}(\varphi \rightarrow \varphi_0) \wedge \mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathsf{G}\zeta_\varphi) \wedge \\ \mathsf{G}(\psi_0 \rightarrow \psi) \wedge \mathcal{A}_\exists(\mathcal{Q}_\psi, \mathcal{I}_\psi, \mathcal{R}_\psi, \mathsf{G}\zeta_\psi) \wedge \\ \mathcal{A}_\forall\left(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathsf{G}\left(\zeta_\varphi \rightarrow \overleftarrow{\mathsf{G}}[\varphi_0 \leftrightarrow \varphi]\right)\right) \wedge \\ \mathcal{A}_\forall\left(\mathcal{Q}_\psi, \mathcal{I}_\psi, \mathcal{R}_\psi, \mathsf{G}\left(\zeta_\psi \rightarrow \overleftarrow{\mathsf{G}}[\psi_0 \leftrightarrow \psi]\right)\right) \wedge \\ \rightarrow \mathcal{A}_\forall\left(\begin{array}{l} \mathcal{Q}_\varphi \cup \mathcal{Q}_\psi \cup \{q\}, \mathcal{I}_\varphi \wedge \mathcal{I}_\psi, \\ \mathcal{R}_\varphi \wedge \mathcal{R}_\psi \wedge (q \leftrightarrow \neg\psi_0 \wedge (\varphi_0 \wedge \mathsf{X}q)), \\ \mathsf{G}\left(\mathsf{F}([q \rightarrow \varphi_0] \wedge \zeta_\varphi \wedge \zeta_\psi) \rightarrow \overleftarrow{\mathsf{G}}(q \leftrightarrow [\varphi \,\underline{\mathsf{B}}\, \psi])\right) \end{array}\right) \end{array}\right)$$

- $$\left(\begin{array}{l} \mathsf{G}(\varphi_0 \rightarrow \varphi) \wedge \mathcal{A}_\exists\,(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathsf{G}\zeta_\varphi) \wedge \\ \mathsf{G}(\psi \rightarrow \psi_0) \wedge \mathcal{A}_\exists\,(\mathcal{Q}_\psi, \mathcal{I}_\psi, \mathcal{R}_\psi, \mathsf{G}\zeta_\psi) \wedge \\ \mathcal{A}_\forall\left(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathsf{G}\left(\zeta_\varphi \rightarrow \overleftarrow{\mathsf{G}}\,[\varphi_0 \leftrightarrow \varphi]\right)\right) \wedge \\ \mathcal{A}_\forall\left(\mathcal{Q}_\psi, \mathcal{I}_\psi, \mathcal{R}_\psi, \mathsf{G}\left(\zeta_\psi \rightarrow \overleftarrow{\mathsf{G}}\,[\psi_0 \leftrightarrow \psi]\right)\right) \wedge \\ \rightarrow \mathcal{A}_\forall\left(\begin{array}{l} \mathcal{Q}_\varphi \cup \mathcal{Q}_\psi \cup \{q\}, \mathcal{I}_\varphi \wedge \mathcal{I}_\psi, \\ \mathcal{R}_\varphi \wedge \mathcal{R}_\psi \wedge (q \leftrightarrow \neg\psi_0 \wedge (\varphi_0 \wedge \mathsf{X}q)), \\ \mathsf{G}\left(\mathsf{F}\,([q \vee \psi_0] \wedge \zeta_\varphi \wedge \zeta_\psi) \rightarrow \overleftarrow{\mathsf{G}}\,(q \leftrightarrow [\varphi\ \mathsf{B}\ \psi])\right) \end{array}\right) \end{array}\right)$$

The above implications are used in the following to develop an algorithm for translating TL_F formulas to equivalent NDet_F automata. This algorithm (given in Figure 5.16) will recursively apply the above implications to end up with an equivalent ω-automaton. One should not be bothered about the universal automata that are used in the above equations. The algorithm will generate existential automata, but to prove its correctness, the above formulas are used as they stand. For the proof of these implications, we only note that they can be easily reduced to formulas where no automaton formulas occur. For example, the first implication can be reduced to the following 'automaton-free' implication:

$$\left(\begin{array}{l} \mathsf{G}(\varphi \rightarrow \varphi_0) \wedge \mathsf{G}(\zeta_\varphi \rightarrow \overleftarrow{\mathsf{G}}\,[\varphi_0 \leftrightarrow \varphi]) \wedge \\ \mathsf{G}(\psi \rightarrow \psi_0) \wedge \mathsf{G}(\zeta_\psi \rightarrow \overleftarrow{\mathsf{G}}\,[\psi_0 \leftrightarrow \psi]) \wedge \\ \mathsf{G}(q \leftrightarrow \psi_0 \vee \varphi_0 \wedge \mathsf{X}q) \\ \rightarrow \mathsf{G}\left(\mathsf{F}\,([q \rightarrow \psi_0] \wedge \zeta_\varphi \wedge \zeta_\psi) \rightarrow \overleftarrow{\mathsf{G}}\,(q \leftrightarrow [\varphi\ \underline{\mathsf{U}}\ \psi])\right) \end{array}\right)$$

The remaining proofs of these implications all run in the same lines, and rely again on the important Lemma 5.35 of page 333 (the proofs are however rather technical and tedious, so that we skip them here). The implications of the lemma required as assumptions that the subsequent automata have product paths with any Kripke structure that satisfy their constraints. This was established by adding existential ω-automata as preconditions of the implications. In order to apply the above implications in a recursive algorithm, we also need to establish these formulas as invariants. For this reason, we will now prove the following lemma:

Lemma 5.55 (Fixing Operator Strength by F-Constraints (II)). *The following implications are initially valid, and show that for any path π in any structure $\mathcal{K}$, there is a product path that satisfies the constraint used in Lemma 5.54:*

- $$\left(\begin{array}{l} \mathcal{A}_\exists\,(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathsf{G}\zeta_\varphi) \wedge \mathcal{A}_\exists\,(\mathcal{Q}_\psi, \mathcal{I}_\psi, \mathcal{R}_\psi, \mathsf{G}\zeta_\psi) \wedge \\ \rightarrow \mathcal{A}_\exists\left(\begin{array}{l} \mathcal{Q}_\varphi \cup \mathcal{Q}_\psi \cup \{q\}, \mathcal{I}_\varphi \wedge \mathcal{I}_\psi, \\ \mathcal{R}_\varphi \wedge \mathcal{R}_\psi \wedge (q \leftrightarrow \psi_0 \vee \varphi_0 \wedge \mathsf{X}q), \\ \mathsf{GF}\,([q \rightarrow \psi_0] \wedge \zeta_\varphi \wedge \zeta_\psi) \end{array}\right) \end{array}\right)$$
- $$\left(\begin{array}{l} \mathcal{A}_\exists\,(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathsf{G}\zeta_\varphi) \wedge \mathcal{A}_\exists\,(\mathcal{Q}_\psi, \mathcal{I}_\psi, \mathcal{R}_\psi, \mathsf{G}\zeta_\psi) \wedge \\ \rightarrow \mathcal{A}_\exists\left(\begin{array}{l} \mathcal{Q}_\varphi \cup \mathcal{Q}_\psi \cup \{q\}, \mathcal{I}_\varphi \wedge \mathcal{I}_\psi, \\ \mathcal{R}_\varphi \wedge \mathcal{R}_\psi \wedge (q \leftrightarrow \psi_0 \vee \varphi_0 \wedge \mathsf{X}q), \\ \mathsf{GF}\,([q \vee \neg\varphi_0] \wedge \zeta_\varphi \wedge \zeta_\psi) \end{array}\right) \end{array}\right)$$

- $$\left(\begin{array}{l}\mathcal{A}_\exists\left(\mathcal{Q}_\varphi,\mathcal{I}_\varphi,\mathcal{R}_\varphi,\mathsf{G}\zeta_\varphi\right)\wedge\mathcal{A}_\exists\left(\mathcal{Q}_\psi,\mathcal{I}_\psi,\mathcal{R}_\psi,\mathsf{G}\zeta_\psi\right)\wedge\\ \rightarrow\mathcal{A}_\exists\left(\begin{array}{l}\mathcal{Q}_\varphi\cup\mathcal{Q}_\psi\cup\{q\},\mathcal{I}_\varphi\wedge\mathcal{I}_\psi,\\ \mathcal{R}_\varphi\wedge\mathcal{R}_\psi\wedge\left(q\leftrightarrow\neg\psi_0\wedge\left(\varphi_0\wedge\mathsf{X}q\right)\right),\\ \mathsf{GF}\left(\left[q\rightarrow\varphi_0\right]\wedge\zeta_\varphi\wedge\zeta_\psi\right)\end{array}\right)\end{array}\right)$$
- $$\left(\begin{array}{l}\mathcal{A}_\exists\left(\mathcal{Q}_\varphi,\mathcal{I}_\varphi,\mathcal{R}_\varphi,\mathsf{G}\zeta_\varphi\right)\wedge\mathcal{A}_\exists\left(\mathcal{Q}_\psi,\mathcal{I}_\psi,\mathcal{R}_\psi,\mathsf{G}\zeta_\psi\right)\wedge\\ \rightarrow\mathcal{A}_\exists\left(\begin{array}{l}\mathcal{Q}_\varphi\cup\mathcal{Q}_\psi\cup\{q\},\mathcal{I}_\varphi\wedge\mathcal{I}_\psi,\\ \mathcal{R}_\varphi\wedge\mathcal{R}_\psi\wedge\left(q\leftrightarrow\neg\psi_0\wedge\left(\varphi_0\wedge\mathsf{X}q\right)\right),\\ \mathsf{GF}\left(\left[q\vee\psi_0\right]\wedge\zeta_\varphi\wedge\zeta_\psi\right)\end{array}\right)\end{array}\right)$$

Proof. The proof of the implications is straightforward. Just note that the added transition relation for the new variable q determines q to behave like either a weak or strong temporal future operator. The constraints that are introduced moreover imply the fairness constraints used to fix the operator strengths of the mentioned temporal future operator. Therefore, we may simply identify q with the mentioned temporal expression ($[\varphi_0 \,\underline{\mathsf{U}}\, \psi_0]$ for the first implication). □

The above two lemmas are essential in the construction of an algorithm that will translate TL_F to NDet_F by the abbreviation technique. The main part of this translation is done by the algorithm given in Figure 5.16. We will furthermore see in Theorem 5.57 that the same algorithm can also be applied to TL_FG with the same effect: It does not translate to NDet_F or $\mathsf{NDet}_\mathsf{FG}$, but to automata with a more complicated acceptance condition that we can easily reduce to NDet_F or $\mathsf{NDet}_\mathsf{FG}$. As $\mathsf{TL}_\mathsf{Prefix} \subseteq \mathsf{TL}_\mathsf{FG}$ holds, the algorithm is furthermore able to handle $\mathsf{TL}_\mathsf{Prefix}$ formulas.

The previous lemmas are sufficient the prove the essential step towards an algorithm for efficiently translating TL_F to NDet_F. For this reason, consider the algorithm given in Figure 5.16. We will see that this algorithm is our desired translation algorithm. To formally prove its correctness, we will however first prove the following lemma:

Lemma 5.56 (Correctness of $\mathsf{Borel}_\sigma(\Phi)$). *Given an arbitrary formula $\Phi \in \mathsf{TL}_\mathsf{FG}$, and the automaton $\mathcal{A}_\exists(\mathcal{Q},\mathcal{I},\mathcal{R},\mathcal{F},\Phi_0) = \mathsf{Borel}_1(\Phi)$ obtained by the algorithm given in Figure 5.16. Then, the following formulas are initially valid, where we abbreviate $\zeta_\Phi := \bigwedge_{\xi\in\mathcal{F}}\xi$ for conciseness:*

(1) $\mathcal{A}_\exists\left(\mathcal{Q},\mathcal{I},\mathcal{R},\mathsf{G}\zeta_\Phi\right)$
(2) $\mathcal{A}_\forall\left(\mathcal{Q},\mathcal{I},\mathcal{R},\mathsf{G}\left(\zeta_\Phi\rightarrow\overleftarrow{\mathsf{G}}\left[\Phi_0\leftrightarrow\Phi\right]\right)\right)$
(3) $\mathcal{A}_\exists\left(\mathcal{Q},\mathcal{I},\mathcal{R},\mathsf{G}\left(\zeta_\Phi\wedge\left[\Phi_0\leftrightarrow\Phi\right]\right)\right)$

Of course, this implies that also the following formulas are initially valid:

(1′) $\mathcal{A}_\exists\left(\mathcal{Q},\mathcal{I},\mathcal{R},\zeta_\Phi\right)$
(2′) $\mathcal{A}_\forall\left(\mathcal{Q},\mathcal{I},\mathcal{R},\left(\zeta_\Phi\rightarrow\left[\Phi_0\leftrightarrow\Phi\right]\right)\right)$
(3′) $\mathcal{A}_\exists\left(\mathcal{Q},\mathcal{I},\mathcal{R},\zeta_\Phi\wedge\left[\Phi_0\leftrightarrow\Phi\right]\right)$

function $\mathsf{Borel}_\sigma(\Phi)$
case Φ **of**
$\mathsf{is\_prop}(\Phi)$: **return** $\mathcal{A}_\exists(\{\}, 1, 1, \{\}, \Phi)$;
$\neg\varphi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \varphi') \equiv \mathsf{Borel}_{\neg\sigma}(\varphi)$;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \neg\varphi')$;
$\varphi \wedge \psi$: **return** $\mathsf{Borel}_\sigma(\varphi) \times \mathsf{Borel}_\sigma(\psi)$;
$\varphi \vee \psi$: $\mathcal{A}_\exists(\mathcal{Q}_\Phi, \mathcal{I}_\Phi, \mathcal{R}_\Phi, \mathcal{F}_\Phi, \varphi' \wedge \psi') \equiv \mathsf{Borel}_\sigma(\varphi) \times \mathsf{Borel}_\sigma(\psi)$;
return $\mathcal{A}_\exists(\mathcal{Q}_\Phi, \mathcal{I}_\Phi, \mathcal{R}_\Phi, \mathcal{F}_\Phi, \varphi' \vee \psi')$;
$\mathsf{X}\varphi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \varphi') \equiv \mathsf{Borel}_\sigma(\varphi)$; $q :=$ new_var;
if $\mathcal{F}_\Phi \neq \{\}$ **then** $\mathcal{F}_\Phi := \left\{\mathsf{X}\bigwedge_{\xi\in\mathcal{F}_\Phi}\xi\right\}$ **end**;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi \cup \{q\}, \mathcal{I}_\varphi, \mathcal{R}_\varphi \wedge (q \leftrightarrow \mathsf{X}\varphi'), \mathcal{F}_\varphi, q)$;
$[\varphi\ \mathsf{U}\ \psi]$: $\mathcal{A}_\exists(\mathcal{Q}_\Phi, \mathcal{I}_\Phi, \mathcal{R}_\Phi, \mathcal{F}_\Phi, \varphi' \wedge \psi') \equiv \mathsf{Borel}_\sigma(\varphi) \times \mathsf{Borel}_\sigma(\psi)$;
$q :=$ new_var; $\mathcal{R}_q := [q \leftrightarrow \psi' \vee \varphi' \wedge \mathsf{X}q]$;
if σ **then** $\mathcal{F}_\Phi := \left\{\mathsf{F}\left[(q \rightarrow \psi') \wedge \bigwedge_{\xi\in\mathcal{F}_\Phi}\xi\right]\right\}$ **end**;
return $\mathcal{A}_\exists(\mathcal{Q}_\Phi \cup \{q\}, \mathcal{I}_\Phi, \mathcal{R}_\Phi \wedge \mathcal{R}_q, \mathcal{F}_\Phi, q)$;
$[\varphi\ \mathsf{B}\ \psi]$: $\mathcal{A}_\exists(\mathcal{Q}_\Phi, \mathcal{I}_\Phi, \mathcal{R}_\Phi, \mathcal{F}_\Phi, \varphi' \wedge \psi') \equiv \mathsf{Borel}_\sigma(\varphi) \times \mathsf{Borel}_{\neg\sigma}(\psi)$;
$q :=$ new_var; $\mathcal{R}_q := [q \leftrightarrow \neg\psi' \wedge (\varphi' \vee \mathsf{X}q)]$;
if $\neg\sigma$ **then** $\mathcal{F}_\Phi := \left\{\mathsf{F}\left[(q \vee \psi') \wedge \bigwedge_{\xi\in\mathcal{F}_\Phi}\xi\right]\right\}$ **end**;
return $\mathcal{A}_\exists(\mathcal{Q}_\Phi \cup \{q\}, \mathcal{I}_\Phi, \mathcal{R}_\Phi \wedge \mathcal{R}_q, \mathcal{F}_\Phi, q)$;
$\overleftarrow{\mathsf{X}}\varphi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \varphi') \equiv \mathsf{Borel}_\sigma(\varphi)$; $q :=$ new_var;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi \cup \{q\}, \mathcal{I}_\varphi \wedge q, \mathcal{R}_\varphi \wedge (\mathsf{X}q \leftrightarrow \varphi'), \mathcal{F}_\varphi, q)$;
$\overleftarrow{\underline{\mathsf{X}}}\varphi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \varphi') \equiv \mathsf{Borel}_\sigma(\varphi)$; $q :=$ new_var;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi \cup \{q\}, \mathcal{I}_\varphi \wedge \neg q, \mathcal{R}_\varphi \wedge (\mathsf{X}q \leftrightarrow \varphi'), \mathcal{F}_\varphi, q)$;
$[\varphi\ \overleftarrow{\mathsf{U}}\ \psi]$: $\mathcal{A}_\exists(\mathcal{Q}_\Phi, \mathcal{I}_\Phi, \mathcal{R}_\Phi, \mathcal{F}_\Phi, \varphi' \wedge \psi') \equiv \mathsf{Borel}_\sigma(\varphi) \times \mathsf{Borel}_\sigma(\psi)$;
$q :=$ new_var; $r_q := \psi' \vee \varphi' \wedge q$; $\mathcal{R}_q := [\mathsf{X}q \leftrightarrow r_q]$;
return $\mathcal{A}_\exists(\mathcal{Q}_\Phi \cup \{q\}, \mathcal{I}_\Phi \wedge \neg q, \mathcal{R}_\Phi \wedge \mathcal{R}_q, \mathcal{F}_\Phi, r_q)$;
$[\varphi\ \overleftarrow{\mathsf{B}}\ \psi]$: $\mathcal{A}_\exists(\mathcal{Q}_\Phi, \mathcal{I}_\Phi, \mathcal{R}_\Phi, \mathcal{F}_\Phi, \varphi' \wedge \psi') \equiv \mathsf{Borel}_\sigma(\varphi) \times \mathsf{Borel}_{\neg\sigma}(\psi)$;
$q :=$ new_var; $r_q := \neg\psi' \wedge (\varphi' \vee q)$; $\mathcal{R}_q := [\mathsf{X}q \leftrightarrow r_q]$;
return $\mathcal{A}_\exists(\mathcal{Q}_\Phi \cup \{q\}, \mathcal{I}_\Phi \wedge q, \mathcal{R}_\Phi \wedge \mathcal{R}_q, \mathcal{F}_\Phi, r_q)$;
end case
end function

Fig. 5.16. Translation of $\mathsf{TL}_{\mathsf{FG}}$ to $\mathsf{NDet}_{\mathsf{FG}}$ (first part)

Proof. The proof of (1) is done by a simple induction on Φ, where we only need to consider the implications of Lemma 5.55. Using this, we can prove (2) by another induction on Φ, where the induction steps are proved by application of the implications of Lemma 5.54. The required assumptions are proved by (1). Note that we have not listed implications for the past operators in Lemma 5.54 and 5.55, since no constraints are generated there. However, the induction steps require similar implications for these cases, which are straightforward to prove. (3) is a simple consequence of (1) and (2): Consider an arbitrary path π of an arbitrary structure $\mathcal{K}$. By (1), it follows that there is at least one product path ξ that satisfies ζ_Φ on all of its states. Hence, we can derive by (2) that $\mathsf{G}[\Phi_0 \leftrightarrow \Phi]$ holds on ξ, so that we obtain (3). (1′),

(2′), and (3′) are simple consequences of (1), (2), and (3), respectively, where we only consider the first point of the path. □

The previous lemma is the essential part required to prove the correctness of our translation. It almost directly implies the correctness of our translation procedure Borel for all $\mathsf{TL}_{\mathsf{FG}}$ formulas. There is only one point that has to be considered: We have to eliminate the following 'grammar rules':

$$P_{\mathsf{GF}} ::= [P_{\mathsf{GF}} \,\underline{\mathsf{U}}\, P_{\mathsf{F}}] \mid [P_{\mathsf{F}} \,\underline{\mathsf{B}}\, P_{\mathsf{FG}}] \quad P_{\mathsf{FG}} ::= [P_{\mathsf{G}} \,\mathsf{U}\, P_{\mathsf{FG}}] \mid [P_{\mathsf{FG}} \,\mathsf{B}\, P_{\mathsf{F}}]$$

We have already seen that this can be done by rewriting the formulas with the following equivalences:

- $[P_{\mathsf{GF}} \,\underline{\mathsf{U}}\, P_{\mathsf{F}}] = [P_{\mathsf{GF}} \,\mathsf{U}\, P_{\mathsf{F}}] \wedge [P_{\mathsf{F}} \,\underline{\mathsf{B}}\, 0]$
- $[P_{\mathsf{F}} \,\underline{\mathsf{B}}\, P_{\mathsf{FG}}] = [P_{\mathsf{F}} \,\mathsf{B}\, P_{\mathsf{FG}}] \wedge [P_{\mathsf{F}} \,\underline{\mathsf{B}}\, 0]$
- $[P_{\mathsf{G}} \,\mathsf{U}\, P_{\mathsf{FG}}] = [P_{\mathsf{G}} \,\underline{\mathsf{U}}\, P_{\mathsf{FG}}] \vee [P_{\mathsf{G}} \,\mathsf{U}\, 0]$
- $[P_{\mathsf{FG}} \,\mathsf{B}\, P_{\mathsf{F}}] = [P_{\mathsf{FG}} \,\underline{\mathsf{B}}\, P_{\mathsf{F}}] \vee [(\neg P_{\mathsf{F}}) \,\mathsf{U}\, 0]$

It is however not recommended to rewrite the formulas with the above equations in advance, since this could provoke a blow-up of the formula. Instead, the above equations should be considered while using the algorithm given in Figure 5.16. Therefore, we can state the following theorem:

Theorem 5.57 (Translating $\mathsf{TL}_{\mathsf{FG}}$ with Borel). *Given any formula $\Phi \in \mathsf{TL}_{\mathsf{FG}}$ that does not require the following rules for its derivation with the grammar of Definition 5.48:*

$$P_{\mathsf{GF}} ::= [P_{\mathsf{GF}} \,\underline{\mathsf{U}}\, P_{\mathsf{F}}] \mid [P_{\mathsf{F}} \,\underline{\mathsf{B}}\, P_{\mathsf{FG}}] \quad P_{\mathsf{FG}} ::= [P_{\mathsf{G}} \,\mathsf{U}\, P_{\mathsf{FG}}] \mid [P_{\mathsf{FG}} \,\mathsf{B}\, P_{\mathsf{F}}]$$

For the ω-automaton $\mathcal{A}_\exists\,(\mathcal{Q}, \mathcal{I}, \mathcal{R}, \mathcal{F}, \Phi_0) := \mathsf{Borel}_\sigma(\Phi)$ that is obtained by the algorithm Borel *as implemented in Figure 5.16, the following holds:*

(1) $\mathsf{Borel}_\sigma(\Phi)$ *runs in time* $O(|\Phi|)$.
(2) Φ_0 *is propositional.*
(3) *Each* $\xi \in \mathcal{F}$ *can be derived from the nonterminal* C_{F} *by the following grammar rules:*

$$P_{\mathsf{Prop}} ::= V_\Sigma \mid \neg P_{\mathsf{Prop}} \mid P_{\mathsf{Prop}} \wedge P_{\mathsf{Prop}} \mid P_{\mathsf{Prop}} \vee P_{\mathsf{Prop}} \mid P_{\mathsf{Prop}} \rightarrow P_{\mathsf{Prop}}$$
$$C_{\mathsf{F}} ::= \mathsf{F} P_{\mathsf{Prop}} \mid \mathsf{F}\,(P_{\mathsf{Prop}} \wedge C_{\mathsf{F}}) \mid \mathsf{X} C_{\mathsf{F}} \mid C_{\mathsf{F}} \wedge C_{\mathsf{F}}$$

(4) *For* $\sigma = 1$, *the equation* $\Phi = \mathcal{A}_\exists \left(\mathcal{Q}, \mathcal{I}, \mathcal{R}, \Phi_0 \wedge \bigwedge_{\xi \in \mathcal{F}} \xi \right)$ *is initially valid.*

(5) *For* $\sigma = 0$, *the equation* $\Phi = \mathcal{A}_\exists \left(\mathcal{Q}, \mathcal{I}, \mathcal{R}, \neg\Phi_0 \wedge \bigwedge_{\xi \in \mathcal{F}} \xi \right)$ *is initially valid.*

(6) *For* $\sigma = 1$ *and* $\Phi \in \mathsf{TL}_{\mathsf{G}}$, *we have* $\mathcal{F} = \{\}$, *and thus* $\mathsf{Borel}_1(\Phi) = \mathsf{TopProp}_1(\Phi)$.
(7) *For* $\sigma = 0$ *and* $\Phi \in \mathsf{TL}_{\mathsf{F}}$, *we have* $\mathcal{F} = \{\}$, *and thus* $\mathsf{Borel}_0(\Phi) = \mathsf{TopProp}_0(\Phi)$.

Proof. $(1-3)$ are easily seen by inspecting the algorithm. To prove (4), we split the equation into two implications, where we abbreviate $\zeta_\Phi := \bigwedge_{\xi \in \mathcal{F}} \xi$ in the following. The first subgoal requires us to prove that $\Phi \rightarrow \mathcal{A}_\exists(\mathcal{Q}, \mathcal{I}, \mathcal{R}, \zeta_\Phi \wedge \Phi_0)$ is initially valid. This is equivalent to the initial validity of $\mathcal{A}_\exists(\mathcal{Q}, \mathcal{I}, \mathcal{R}, \Phi \rightarrow \zeta_\Phi \wedge \Phi_0)$. As $\zeta_\Phi \wedge [\Phi_0 = \Phi] \rightarrow (\Phi \rightarrow \zeta_\Phi \wedge \Phi_0)$ is a propositional tautology, our subgoal follows immediately from fact $(3)'$ of Lemma 5.56.

It remains to prove that $\mathcal{A}_\exists(\mathcal{Q}, \mathcal{I}, \mathcal{R}, \zeta_\Phi \wedge \Phi_0) \rightarrow \Phi$ is initially valid. Again, we can easily verify that this is equivalent to the initial validity of the formula $\mathcal{A}_\forall(\mathcal{Q}, \mathcal{I}, \mathcal{R}, [\zeta_\Phi \wedge \Phi_0] \rightarrow \Phi)$. Note that we now have obtained a universal automaton formula. As $[\zeta_\Phi \wedge \Phi_0] \rightarrow \Phi$ is equivalent to $\zeta_\Phi \rightarrow (\Phi_0 \rightarrow \Phi)$, this can be concluded from fact $(2')$ of Lemma 5.56. The other facts are easily seen by inspection of the algorithm. □

To be precise, property (3) above does not hold as it stands. There are cases, where empty conjunctions are added as a constraint, i.e., when $\mathcal{F}_\Phi = \{\}$ holds. In these cases, the algorithm generates a constraint of the form $\mathsf{F}[\zeta \wedge 1]$ (where ζ is propositional), but we assume that this will be immediately simplified to $\mathsf{F}\zeta$.

Hence, we almost have an efficient algorithm to translate the logics TL_F, $\mathsf{TL}_{\mathsf{Prefix}}$, and $\mathsf{TL}_{\mathsf{FG}}$. However, the result is not yet a NDet_κ automaton, as the acceptance condition is not yet of the desired form.

We will now show that the generated acceptance condition can be converted to an equivalent NDet_F or $\mathsf{NDet}_{\mathsf{FG}}$ automaton. To this end, we call the formulas that appear as acceptance conditions of the automata that are constructed by the Borel algorithm C_F formulas. It is not hard to see that the equations of the following lemma can be used to compute for any C_F formula an equivalent $\mathsf{NDet}_{\mathsf{FG}}$ automaton:

Lemma 5.58 (Translating C_F to $\mathsf{NDet}_{\mathsf{FG}}$). *Given a propositional formula φ, and the two $\mathsf{NDet}_{\mathsf{FG}}$ automaton formulas $\mathfrak{A}_\Phi = \mathcal{A}_\exists(Q_\Phi, \Phi_\mathcal{I}, \Phi_\mathcal{R}, \mathsf{FG}\Phi_\mathcal{F})$, and $\mathfrak{A}_\Psi = \mathcal{A}_\exists(Q_\Psi, \Psi_\mathcal{I}, \Psi_\mathcal{R}, \mathsf{FG}\Psi_\mathcal{F})$, the following equations hold:*

- $\mathsf{F}\varphi = \mathcal{A}_\exists(\{q\}, \neg q, \neg q \wedge \neg\varphi \wedge \neg\mathsf{X}q \vee \neg q \wedge \varphi \wedge \mathsf{X}q \vee q \wedge \mathsf{X}q, \mathsf{FG}q)$
- $\mathsf{F}(\varphi \wedge \mathfrak{A}_\Phi) = \mathcal{A}_\exists \begin{pmatrix} Q \cup \{p\}, \neg p \vee \Phi_\mathcal{I}, \\ [\neg p \wedge \neg\mathsf{X}p] \vee \\ [\neg p \wedge \mathsf{X}(p \wedge \Phi_\mathcal{I} \wedge \varphi)] \vee \\ [p \wedge \Phi_\mathcal{R} \wedge \mathsf{X}p], \\ \mathsf{FG}(p \wedge \Phi_\mathcal{F}) \end{pmatrix}$
- $\mathsf{X}[\mathfrak{A}_\Phi] = \mathcal{A}_\exists \begin{pmatrix} Q \cup \{p\}, \\ \neg p, \\ [\neg p \wedge \mathsf{X}p \wedge \mathsf{X}\Phi_\mathcal{I}] \vee [p \wedge \Phi_\mathcal{R} \wedge \mathsf{X}p], \\ \mathsf{FG}\Phi_\mathcal{F} \end{pmatrix}$
- $\varphi \wedge \mathfrak{A}_\Phi = \mathcal{A}_\exists(Q_\Phi, \Phi_\mathcal{I} \wedge \varphi, \Phi_\mathcal{R}, \mathsf{FG}\Phi_\mathcal{F})$
- $\mathfrak{A}_\Phi \wedge \mathfrak{A}_\Psi = \mathcal{A}_\exists(Q_\Phi \cup Q_\Psi, \Phi_\mathcal{I} \wedge \Psi_\mathcal{I}, \Phi_\mathcal{R} \wedge \Psi_\mathcal{R}, \mathsf{FG}(\Phi_\mathcal{F} \wedge \Psi_\mathcal{F}))$

Obviously, the equations correspond with the grammar rules of C_{F}: The idea is to compute within a bottom-up traversal over the C_{F} formulas an equivalent $\mathsf{NDet}_{\mathsf{FG}}$ automaton out of previously computed $\mathsf{NDet}_{\mathsf{FG}}$ automata for its subformulas. This is quite different from our previous translation procedures, which are best understood as extraction procedures that extract some sort of largest top-level formulas: Subformulas that violate some grammar rules are abbreviated by new state variables.

Hence, we are now able to translate within linear runtime any $\mathsf{TL}_{\mathsf{FG}}$ formula to an equivalent one of $\mathsf{NDet}_{\mathsf{FG}}$. The above lemma can also be formulated with $\mathsf{NDet}_{\mathsf{F}}$ automata; which is not surprising since we already know that $\mathsf{NDet}_{\mathsf{FG}} \approx \mathsf{NDet}_{\mathsf{F}}$ holds. However, one has to take care with the following problem: We can efficiently compute the conjunction of $\mathsf{NDet}_{\mathsf{FG}}$ automata due to the following equivalence $\mathsf{FG}\varphi \wedge \mathsf{FG}\psi = \mathsf{FG}(\varphi \wedge \psi)$. In general, the equation $\mathsf{F}\varphi \wedge \mathsf{F}\psi = \mathsf{F}(\varphi \wedge \psi)$ is not valid, and we therefore used watchdogs to compute the conjunction of $\mathsf{NDet}_{\mathsf{F}}$ automata in Lemma 4.11.

However, this additional effort is not necessary for the automata that are computed for the C_{F} formulas, as these $\mathsf{NDet}_{\mathsf{F}}$ automata have a special property: Their acceptance condition is of the form $\mathsf{F}\Phi_{\mathcal{F}}$ and it is equivalent to $\mathsf{FG}\Phi_{\mathcal{F}}$, which means that once a run satisfies the formula $\Phi_{\mathcal{F}}$, then all its remaining states satisfy $\Phi_{\mathcal{F}}$, too. Hence, the acceptance conditions are suffix-closed (cf. page 209). For this reason, we have the following lemma for the reduction of C_{F} formulas to $\mathsf{NDet}_{\mathsf{F}}$ automata:

Lemma 5.59 (Translating C_{F} to $\mathsf{NDet}_{\mathsf{F}}$). *Given a propositional formula φ, and the two $\mathsf{NDet}_{\mathsf{F}}$ automaton formulas $\mathfrak{A}_\Phi = \mathcal{A}_\exists(Q_\Phi, \Phi_\mathcal{I}, \Phi_\mathcal{R}, \mathsf{F}\Phi_\mathcal{F})$, and $\mathfrak{A}_\Psi = \mathcal{A}_\exists(Q_\Psi, \Psi_\mathcal{I}, \Psi_\mathcal{R}, \mathsf{F}\Psi_\mathcal{F})$ which are equivalent to the $\mathsf{NDet}_{\mathsf{FG}}$ automaton formulas $\mathcal{A}_\exists(Q_\Phi, \Phi_\mathcal{I}, \Phi_\mathcal{R}, \mathsf{FG}\Phi_\mathcal{F})$ and $\mathcal{A}_\exists(Q_\Psi, \Psi_\mathcal{I}, \Psi_\mathcal{R}, \mathsf{FG}\Psi_\mathcal{F})$, respectively, then the following equations hold:*

- $\mathsf{F}\varphi = \mathcal{A}_\exists(\{q\}, \neg q, \neg q \wedge \neg\varphi \wedge \neg\mathsf{X}q \vee \neg q \wedge \varphi \wedge \mathsf{X}q \vee q \wedge \mathsf{X}q, \mathsf{F}q)$
- $\mathsf{F}(\varphi \wedge \mathfrak{A}_\Phi) = \mathcal{A}_\exists \begin{pmatrix} Q \cup \{p\}, \neg p \vee \Phi_\mathcal{I}, \\ [\neg p \wedge \neg\mathsf{X}p] \vee \\ [\neg p \wedge \mathsf{X}(p \wedge \Phi_\mathcal{I} \wedge \varphi)] \vee \\ [p \wedge \Phi_\mathcal{R} \wedge \mathsf{X}p], \\ \mathsf{F}(p \wedge \Phi_\mathcal{F}) \end{pmatrix}$
- $\mathsf{X}[\mathfrak{A}_\Phi] = \mathcal{A}_\exists \begin{pmatrix} Q \cup \{p\}, \\ \neg p, \\ [\neg p \wedge \mathsf{X}p \wedge \mathsf{X}\Phi_\mathcal{I}] \vee [p \wedge \Phi_\mathcal{R} \wedge \mathsf{X}p], \\ \mathsf{F}\Phi_\mathcal{F} \end{pmatrix}$
- $\varphi \wedge \mathfrak{A}_\Phi = \mathcal{A}_\exists(Q_\Phi, \Phi_\mathcal{I} \wedge \varphi, \Phi_\mathcal{R}, \mathsf{F}\Phi_\mathcal{F})$
- $\mathfrak{A}_\Phi \wedge \mathfrak{A}_\Psi = \mathcal{A}_\exists(Q_\Phi \cup Q_\Psi, \Phi_\mathcal{I} \wedge \Psi_\mathcal{I}, \Phi_\mathcal{R} \wedge \Psi_\mathcal{R}, \mathsf{F}(\Phi_\mathcal{F} \wedge \Psi_\mathcal{F}))$

Furthermore, the acceptance conditions $\mathsf{F}\mathcal{F}$ of all automata on the right hand sides can also be replaced with $\mathsf{FG}\mathcal{F}$.

Unfortunately, we still have to apply the first equation to the C_{F} formulas that are generated by the grammar rule $C_{\mathsf{F}} ::= \mathsf{F}P_{\mathsf{Prop}}$, since the reachability

constraints that are introduced by our algorithms do not have the terminal property (which means that F could be replaced with FG). Nevertheless, we now have the following extension of Theorem 5.52:

Corollary 5.60 (Temporal Borel Classes). *For any formula $\Phi \in \mathsf{TL}_\kappa$ with $\kappa \in \{\mathsf{G}, \mathsf{F}, \mathsf{Prefix}, \mathsf{GF}, \mathsf{FG}, \mathsf{Streett}\}$, there is an equivalent ω-automaton $\mathfrak{A}_\Phi \in \mathsf{Det}_\kappa$.*

5.4.5 Reductions to CTL/LeftCTL* Model Checking

The algorithms of the previous sections enable us to translate any temporal logic formula to an equivalent existential ω-automaton. We are therefore able to solve the model checking problem for all temporal logics, and even the satisfiability problem for linear time temporal logics. The following two lemmas show how these problems are handled. First of all, the *model checking problem* can be reduced to an automaton problem:

Lemma 5.61 (Temporal Logic Model Checking). *Given any LTL_p formula Φ, an automaton $\mathfrak{A}_\Phi = \mathcal{A}_\exists(\mathcal{Q}_\Phi, \mathcal{I}_\Phi, \mathcal{R}_\Phi, \mathcal{F}_\Phi)$ that is initially equivalent to Φ, and a further automaton $\mathfrak{A}_{\neg\Phi} := \mathcal{A}_\exists(\mathcal{Q}_{\neg\Phi}, \mathcal{I}_{\neg\Phi}, \mathcal{R}_{\neg\Phi}, \mathcal{F}_{\neg\Phi})$ that is initially equivalent to $\neg\Phi$. Moreover, let $\mathcal{K}_\Phi = \mathsf{Struct}(\mathfrak{A}_\Phi)$ and $\mathcal{K}_{\neg\Phi} = \mathsf{Struct}(\mathfrak{A}_{\neg\Phi})$ be the associated Kripke structures of $\mathfrak{A}_\Phi$ and $\mathfrak{A}_{\neg\Phi}$, respectively, then the following hold:*

- $[\![\mathsf{E}\Phi]\!]_\mathcal{K} = [\![\mathsf{E}\mathfrak{A}_\Phi]\!]_\mathcal{K} = \{s \in \mathcal{S} \mid \exists \vartheta \subseteq \mathcal{Q}_\Phi \cup V_\Sigma.(\mathcal{K} \times \mathcal{K}_\Phi, (s, \vartheta)) \models \mathcal{I}_\Phi \wedge \mathsf{E}\mathcal{F}_\Phi\}$
- $[\![\mathsf{A}\Phi]\!]_\mathcal{K} = \{s \in \mathcal{S} \mid \forall \vartheta \subseteq \mathcal{Q}_{\neg\Phi} \cup V_\Sigma.(\mathcal{K} \times \mathcal{K}_{\neg\Phi}, (s, \vartheta)) \models \mathcal{I}_{\neg\Phi} \rightarrow \mathsf{A}\neg\mathcal{F}_{\neg\Phi}\}$

If the above lemma is used in connection with the PQNF normal form (page 85), it is also possible to extend it to CTL* formulas. To prove its correctness, simply recall how we reduced ω-automata model checking to equivalent μ-calculus problems in Theorem 4.65: *Given an automaton formula $\mathfrak{A} = \mathcal{A}_\exists(\mathcal{Q}, \mathcal{I}, \mathcal{R}, \mathcal{F})$, then the following are equivalent for any structure $\mathcal{K}$, and any state s of $\mathcal{K}$:*

- $(\mathcal{K}, s) \models \mathsf{E}\mathcal{A}_\exists(\mathcal{Q}_\Phi, \mathcal{I}_\Phi, \mathcal{R}_\Phi, \mathcal{F})$
- *there is a $\vartheta \subseteq \mathcal{Q} \cup \mathcal{V}$ such that $(\mathcal{K} \times \mathcal{K}_\mathfrak{A}, (s, \vartheta)) \models \mathcal{I} \wedge \mathsf{E}\mathcal{F}$ and $\mathcal{L}(s) = \vartheta \cap \mathcal{V}$*

Hence, we see that our translations allow us to efficiently solve the model checking problems. The next lemma shows how the validity and satisfiability problems for linear time temporal logic can be solved:

Lemma 5.62 (Satisfiability Problem of LTL_p). *Given any LTL_p formula Φ over the variables V_Σ, an automaton $\mathfrak{A}_\Phi = \mathcal{A}_\exists(\mathcal{Q}_\Phi, \mathcal{I}_\Phi, \mathcal{R}_\Phi, \mathcal{F}_\Phi)$ that is initially equivalent to Φ, and a further automaton $\mathfrak{A}_{\neg\Phi} := \mathcal{A}_\exists(\mathcal{Q}_{\neg\Phi}, \mathcal{I}_{\neg\Phi}, \mathcal{R}_{\neg\Phi}, \mathcal{F}_{\neg\Phi})$ that is initially equivalent to $\neg\Phi$. Moreover, let $\mathcal{K}_\Phi = \mathsf{Struct}(\mathfrak{A}_\Phi)$ and $\mathcal{K}_{\neg\Phi} = \mathsf{Struct}(\mathfrak{A}_{\neg\Phi})$ be the associated Kripke structures of $\mathfrak{A}_\Phi$ and $\mathfrak{A}_{\neg\Phi}$, respectively, then the following hold:*

- *Φ is valid iff $[\![\mathcal{I}_{\neg\Phi} \wedge \mathsf{E}\mathcal{F}_{\neg\Phi}]\!]_{\mathcal{K}_{\neg\Phi}} = \{\}$*
- *Φ is satisfiable iff $[\![\mathcal{I}_\Phi \wedge \mathsf{E}\mathcal{F}_\Phi]\!]_{\mathcal{K}_\Phi} \neq \{\}$*

The PQNF normal form does not however help us to construct a satisfiability checking procedure for CTL*. Satisfiability checking can be obtained by reductions to ω-tree automata (see Appendix B.3). The satisfiability checking problem for CTL* is, moreover, 2EXPTIME-complete [165, 177].

Moreover, we have already shown in Theorem 4.65 how the standard acceptance conditions of the ω-automata of the Borel hierarchy are reduced to equivalent μ-calculus formulas. Some of them can even be reduced to the alternation-free μ-calculus, and some even to CTL.

Having this view, we see that it is not problematic when an acceptance condition is a larger CTL formula than simple ones like $\mathsf{EG}\Phi$, $\mathsf{EF}\Phi$, or $\mathsf{EFEF}\Phi$ (with propositional Φ). For this reason, the primary goal in efficiently solving the model checking problem for temporal logics is not a reduction to simple ω-automata. Instead, a reduction to 'some' automaton formula with acceptance condition $\mathcal{F}$, so that $\mathsf{E}\mathcal{F}$ can be reduced to a CTL formula, will do. Now, recall that we have already found a temporal logic, namely LeftCTL*, that can be translated to CTL, and that has nontrivial path formulas.

Hence, a translation to ω-automata with acceptance condition $\mathcal{F}$, so that $\mathsf{E}\mathcal{F} \in$ LeftCTL holds, is more interesting for practice than a reduction to standard automata.* This way we can save states for the automaton construction and generate rather larger acceptance conditions instead. For this reason, note that it was completely superfluous to reduce the C_F formulas that appeared as intermediate acceptance conditions to equivalent $\mathsf{NDet}_\mathsf{FG}$ automata, since for any C_F formula Φ, we have $\mathsf{E}\Phi \in$ LeftCTL*. Our translation from LeftCTL* to CTL can then be used for the final translation to a CTL model checking problem.

Hence, we should extract LeftCTL* formulas instead of TL_FG formulas. To be more precise, we now define the following linear-time temporal logics TL_PA and TL_PE that are obviously related to LeftCTL*:

Definition 5.63 (The Logics TL_PE and TL_PA). *We define the logics TL_PE and TL_PA by the following grammar rules, where TL_PE and TL_PA are the sets of formulas that can be derived from the nonterminals P_PE and P_PA, respectively:*

$$
\begin{array}{ll}
P_\mathsf{Prop} ::= V_\Sigma \mid \neg P_\mathsf{Prop} \mid P_\mathsf{Prop} \wedge P_\mathsf{Prop} \mid P_\mathsf{Prop} \vee P_\mathsf{Prop} & \\
P_\mathsf{PA} ::= P_\mathsf{Prop} & P_\mathsf{PE} ::= P_\mathsf{Prop} \\
\quad \mid \neg P_\mathsf{PE} \mid P_\mathsf{PA} \wedge P_\mathsf{PA} \mid P_\mathsf{PA} \vee P_\mathsf{PA} & \quad \mid \neg P_\mathsf{PA} \mid P_\mathsf{PE} \wedge P_\mathsf{PE} \mid P_\mathsf{PE} \vee P_\mathsf{PE} \\
\quad \mid \mathsf{X}P_\mathsf{PA} \mid \mathsf{G}P_\mathsf{PA} \mid \mathsf{F}P_\mathsf{Prop} & \quad \mid \mathsf{X}P_\mathsf{PE} \mid \mathsf{G}P_\mathsf{Prop} \mid \mathsf{F}P_\mathsf{PE} \\
\quad \mid [P_\mathsf{PA} \,\underline{\mathsf{U}}\, P_\mathsf{Prop}] \mid [P_\mathsf{Prop} \,\underline{\mathsf{B}}\, P_\mathsf{PE}] & \quad \mid [P_\mathsf{Prop} \,\underline{\mathsf{U}}\, P_\mathsf{PE}] \mid [P_\mathsf{PE} \,\underline{\mathsf{B}}\, P_\mathsf{Prop}] \\
\quad \mid [P_\mathsf{PA} \,\mathsf{U}\, P_\mathsf{Prop}] \mid [P_\mathsf{Prop} \,\mathsf{B}\, P_\mathsf{PE}] & \quad \mid [P_\mathsf{Prop} \,\mathsf{U}\, P_\mathsf{PE}] \mid [P_\mathsf{PE} \,\mathsf{B}\, P_\mathsf{Prop}]
\end{array}
$$

The logics TL_PE and TL_PA are closely related to the branching time temporal logic LeftCTL* that we have already studied: It is not hard to see that for any TL_PE formula Φ the formula $\mathsf{E}\Phi$ belongs to LeftCTL*, and analogously, that for any TL_PA formula Φ the formula $\mathsf{A}\Phi$ belongs to LeftCTL*. It is furthermore not difficult to see that the logics TL_PE and TL_PA are subsets of the logics TL_FG and TL_GF, respectively, and that TL_PE and TL_PA are dual to each other.

Lemma 5.64 (Relationship between $\mathsf{TL}_{\mathsf{PE}}/\mathsf{TL}_{\mathsf{PA}}$ and $\mathsf{TL}_{\mathsf{FG}}/\mathsf{TL}_{\mathsf{GF}}$).

- $\mathsf{TL}_{\mathsf{PE}} \subseteq \mathsf{TL}_{\mathsf{FG}}$ *and* $\mathsf{TL}_{\mathsf{PA}} \subseteq \mathsf{TL}_{\mathsf{GF}}$
- *For any* $\Phi \in \mathsf{TL}_{\mathsf{PE}}$, *we have* $\neg\Phi \in \mathsf{TL}_{\mathsf{PA}}$, *and for any* $\Phi \in \mathsf{TL}_{\mathsf{PA}}$, *we have* $\neg\Phi \in \mathsf{TL}_{\mathsf{PE}}$.

However, $\mathsf{TL}_{\mathsf{PE}}$ is significantly weaker than $\mathsf{TL}_{\mathsf{FG}}$, in particular, neither TL_{G} nor TL_{F} is a subset of $\mathsf{TL}_{\mathsf{PE}}$. For example, we have already mentioned that $\mathsf{AF}[a \wedge \mathsf{X}a]$ can neither be expressed in CTL nor in FairCTL [162], hence, this formula can also not be expressed in LeftCTL*. However, the formula $\mathsf{F}[a \wedge \mathsf{X}a]$ is a $\mathsf{TL}_{\mathsf{GF}}$ formula.

LeftCTL* has however the advantage of a relatively simple translation to the alternation-free μ-calculus, even a translation to CTL. For this reason, we will now define an extraction procedure TopPE similar to TopFG, but assume that we start the extraction from a $\mathsf{TL}_{\mathsf{FG}}$ formula (this formula may be obtained by TopFG). With that assumption, we can abbreviate the subformulas without generating fairness constraints as shown in Figure 5.17.

Theorem 5.65 (Correctness of $\mathsf{TopPE}_\sigma(\Phi)$). *Given any formula* $\Phi \in \mathsf{TL}_{\mathsf{FG}}$, *and the resulting automaton formula* $\mathcal{A}_\exists(\mathcal{Q}, \mathcal{I}, \mathcal{R}, \Phi_0) = \mathsf{TopPE}_1(\Phi)$ *obtained by the algorithm given in Figure 5.17, then the following hold:*

(1) $\mathsf{TopPE}_\sigma(\Phi)$ *runs in time* $O(|\Phi|)$.
(2) $\Phi_0 \in P_{\mathsf{PE}}$
(3) $\Phi = \mathcal{A}_\exists(\mathcal{Q}, \mathcal{I}, \mathcal{R}, \Phi_0)$ *is initially valid.*

The proof is not too complicated. In particular, the same explanations as given in the correctness proof of Borel hold, i.e., the introduced abbreviations must hold long enough as controlled by the generated C_{F} constraints. These constraints can now however be integrated into the remaining $\mathsf{TL}_{\mathsf{PE}}/\mathsf{TL}_{\mathsf{PA}}$ formulas. Note that we do not have the problem that we had in translating $\mathsf{TL}_{\mathsf{FG}}$ formulas: Algorithm Borel could not be used to handle these formulas since it was not clear where to put the reachability constraints for the subformulas that have been generated by the 'asymmetric' grammar rules of $\mathsf{TL}_{\mathsf{FG}}$. The problem does not occur here since we no longer need to abbreviate the until operators, even if weak ones occur positively.

A possible translation procedure could now work as follows: we first extract the largest $\mathsf{TL}_{\mathsf{FG}}$ formula by generating fairness constraints. After that, we further extract the largest $\mathsf{TL}_{\mathsf{PE}}$ formula, and translate the remaining automaton problem to an equivalent CTL model checking problem. *For this reason, as much as possible is kept in the acceptance condition, and as less as possible is translated into states and state transitions.*

However, this will not always be the best solution: Recall that the translation from LeftCTL* to CTL requires an exponential runtime in the worst case. So, there might be cases, where a translation to 'state variables' can be more efficient. Nevertheless, it is reasonable to still extract $\mathsf{TL}_{\mathsf{PE}}$ formulas even if

function $\mathsf{TopPE}_\sigma(\Phi)$
case Φ **of**
is_prop(Φ): **return** $\mathcal{A}_\exists(\{\}, 1, 1, \Phi)$;
$\neg\varphi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \varphi') \equiv \mathsf{TopPE}_{\neg\sigma}(\varphi)$;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \neg\varphi')$;
$\varphi \wedge \psi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \varphi') \equiv \mathsf{TopPE}_\sigma(\varphi)$;
$\mathcal{A}_\exists(\mathcal{Q}_\psi, \mathcal{I}_\psi, \mathcal{R}_\psi, \psi') \equiv \mathsf{TopPE}_\sigma(\psi)$;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi \cup \mathcal{Q}_\psi, \mathcal{I}_\varphi \wedge \mathcal{I}_\psi, \mathcal{R}_\varphi \wedge \mathcal{R}_\psi, \varphi' \wedge \psi')$;
$\varphi \vee \psi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \varphi') \equiv \mathsf{TopPE}_\sigma(\varphi)$;
$\mathcal{A}_\exists(\mathcal{Q}_\psi, \mathcal{I}_\psi, \mathcal{R}_\psi, \psi') \equiv \mathsf{TopPE}_\sigma(\psi)$;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi \cup \mathcal{Q}_\psi, \mathcal{I}_\varphi \wedge \mathcal{I}_\psi, \mathcal{R}_\varphi \wedge \mathcal{R}_\psi, \varphi' \vee \psi')$;
$\mathsf{X}\varphi$: $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \varphi') \equiv \mathsf{TopPE}_\sigma(\varphi)$;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathsf{X}\varphi')$;
$[\varphi\ \mathsf{U}\ \psi]$: **if** σ **then** $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \varphi') \equiv \mathsf{Borel}_\sigma(\varphi)$;
$\mathcal{A}_\exists(\mathcal{Q}_\psi, \mathcal{I}_\psi, \mathcal{R}_\psi, \psi') \equiv \mathsf{TopPE}_\sigma(\psi \wedge \bigwedge_{\xi\in\mathcal{F}_\varphi} \xi)$;
else $\mathcal{A}_\exists(\mathcal{Q}_\psi, \mathcal{I}_\psi, \mathcal{R}_\psi, \mathcal{F}_\psi, \psi') \equiv \mathsf{Borel}_\sigma(\psi)$;
$\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \varphi') \equiv \mathsf{TopPE}_\sigma(\varphi \wedge \bigwedge_{\xi\in\mathcal{F}_\psi} \xi)$;
end;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi \cup \mathcal{Q}_\psi, \mathcal{I}_\varphi \wedge \mathcal{I}_\psi, \mathcal{R}_\varphi \wedge \mathcal{R}_\psi, [\varphi'\ \mathsf{U}\ \psi'])$;
$[\varphi\ \underline{\mathsf{U}}\ \psi]$: **if** σ **then** $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \varphi') \equiv \mathsf{Borel}_\sigma(\varphi)$;
$\mathcal{A}_\exists(\mathcal{Q}_\psi, \mathcal{I}_\psi, \mathcal{R}_\psi, \psi') \equiv \mathsf{TopPE}_\sigma(\psi \wedge \bigwedge_{\xi\in\mathcal{F}_\varphi} \xi)$;
else $\mathcal{A}_\exists(\mathcal{Q}_\psi, \mathcal{I}_\psi, \mathcal{R}_\psi, \mathcal{F}_\psi, \psi') \equiv \mathsf{Borel}_\sigma(\psi)$;
$\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \varphi') \equiv \mathsf{TopPE}_\sigma(\varphi \wedge \bigwedge_{\xi\in\mathcal{F}_\psi} \xi)$;
end;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi \cup \mathcal{Q}_\psi, \mathcal{I}_\varphi \wedge \mathcal{I}_\psi, \mathcal{R}_\varphi \wedge \mathcal{R}_\psi, [\varphi'\ \underline{\mathsf{U}}\ \psi'])$;
$[\varphi\ \mathsf{B}\ \psi]$: **if** σ **then** $\mathcal{A}_\exists(\mathcal{Q}_\psi, \mathcal{I}_\psi, \mathcal{R}_\psi, \mathcal{F}_\psi, \psi') \equiv \mathsf{Borel}_{\neg\sigma}(\psi)$;
$\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \varphi') \equiv \mathsf{TopPE}_\sigma(\varphi \wedge \bigwedge_{\xi\in\mathcal{F}_\psi} \xi)$;
else $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \varphi') \equiv \mathsf{Borel}_\sigma(\varphi)$;
$\mathcal{A}_\exists(\mathcal{Q}_\psi, \mathcal{I}_\psi, \mathcal{R}_\psi, \psi') \equiv \mathsf{TopPE}_{\neg\sigma}(\psi \wedge \bigwedge_{\xi\in\mathcal{F}_\varphi} \xi)$;
end;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi \cup \mathcal{Q}_\psi, \mathcal{I}_\varphi \wedge \mathcal{I}_\psi, \mathcal{R}_\varphi \wedge \mathcal{R}_\psi, [\varphi'\ \mathsf{B}\ \psi'])$;
$[\varphi\ \underline{\mathsf{B}}\ \psi]$: **if** σ **then** $\mathcal{A}_\exists(\mathcal{Q}_\psi, \mathcal{I}_\psi, \mathcal{R}_\psi, \mathcal{F}_\psi, \psi') \equiv \mathsf{Borel}_{\neg\sigma}(\psi)$;
$\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \varphi') \equiv \mathsf{TopPE}_\sigma(\varphi \wedge \bigwedge_{\xi\in\mathcal{F}_\psi} \xi)$;
else $\mathcal{A}_\exists(\mathcal{Q}_\varphi, \mathcal{I}_\varphi, \mathcal{R}_\varphi, \mathcal{F}_\varphi, \varphi') \equiv \mathsf{Borel}_\sigma(\varphi)$;
$\mathcal{A}_\exists(\mathcal{Q}_\psi, \mathcal{I}_\psi, \mathcal{R}_\psi, \psi') \equiv \mathsf{TopPE}_{\neg\sigma}(\psi \wedge \bigwedge_{\xi\in\mathcal{F}_\varphi} \xi)$;
end;
return $\mathcal{A}_\exists(\mathcal{Q}_\varphi \cup \mathcal{Q}_\psi, \mathcal{I}_\varphi \wedge \mathcal{I}_\psi, \mathcal{R}_\varphi \wedge \mathcal{R}_\psi, [\varphi'\ \underline{\mathsf{B}}\ \psi'])$;
otherwise: $\mathcal{A}_\exists(\mathcal{Q}_\Phi, \mathcal{I}_\Phi, \mathcal{R}_\Phi, \mathcal{F}_\Phi, \Phi') \equiv \mathsf{Borel}_\sigma(\Phi)$;
return $\mathcal{A}_\exists\left(\mathcal{Q}_\Phi, \mathcal{I}_\Phi \wedge \Phi', \mathcal{R}_\Phi, \bigwedge_{\xi\in\mathcal{F}_\Phi} \xi\right)$;
end case
end function

Fig. 5.17. Extraction of top-level LeftCTL* $\mathsf{TL}_{\mathsf{PE}}$ ($\sigma = 1$) and $\mathsf{TL}_{\mathsf{PA}}$ ($\sigma = 0$) formulas

no further translation to a CTL model checking problem is desired. The reason for this is as follows: *Temporal operators that are abbreviated by* Borel *using a reachability constraint introduce a state variable to abbreviate the subformula, and a reachability constraint. A later translation using the closure theorems of Lemma 5.58, will then add further state variables.* Therefore, one temporal operator may introduce more than one state variable with this approach.

To circumvent this overhead, we now show that this is not necessary for $\mathsf{TL}_{\mathsf{PE}}$ formulas in that we develop a translation from $\mathsf{TL}_{\mathsf{PE}}$ to $\mathsf{NDet}_{\mathsf{FG}}$ that is independent of the function Borel. Our alternative translation is based on closure theorems just like the translation of the C_{F} formulas, i.e., we construct in a bottom-up manner $\mathsf{NDet}_{\mathsf{FG}}$ from the $\mathsf{NDet}_{\mathsf{FG}}$ automaton that has been generated by one of the subformulas. This computation is done with the following closure theorems.

Lemma 5.66 (Translating $\mathsf{TL}_{\mathsf{PE}}$ to $\mathsf{NDet}_{\mathsf{FG}}$). *Given a propositional formula φ, and the $\mathsf{NDet}_{\mathsf{FG}}$ automaton formulas $\mathfrak{A}_\Phi = \mathcal{A}_\exists(Q_\Phi, \Phi_\mathcal{I}, \Phi_\mathcal{R}, \mathsf{FG}\Phi_\mathcal{F})$, and $\mathfrak{A}_\Psi = \mathcal{A}_\exists(Q_\Psi, \Psi_\mathcal{I}, \Psi_\mathcal{R}, \mathsf{FG}\Psi_\mathcal{F})$, the following equations hold:*

- $\mathfrak{A}_\Phi \wedge \mathfrak{A}_\Psi = \mathcal{A}_\exists(Q_\Phi \cup Q_\Psi, \Phi_\mathcal{I} \wedge \Psi_\mathcal{I}, \Phi_\mathcal{R} \wedge \Psi_\mathcal{R}, \mathsf{FG}(\Phi_\mathcal{F} \wedge \Psi_\mathcal{F}))$
- $\mathfrak{A}_\Phi \vee \mathfrak{A}_\Psi = \mathcal{A}_\exists \begin{pmatrix} Q_\Phi \cup Q_\Psi \cup \{p\}, \\ \neg p \wedge \Phi_\mathcal{I} \vee p \wedge \Psi_\mathcal{I}, \\ \neg p \wedge \Phi_\mathcal{R} \wedge \neg \mathsf{X} p \vee p \wedge \Psi_\mathcal{R} \wedge \mathsf{X} p, \\ \mathsf{FG}(\neg p \wedge \Phi_\mathcal{F} \vee p \wedge \Psi_\mathcal{F}) \end{pmatrix}$
- $\mathsf{X}[\mathfrak{A}_\Phi] = \mathcal{A}_\exists \begin{pmatrix} Q \cup \{p\}, \\ \neg p, \\ [\neg p \wedge \mathsf{X} p \wedge \mathsf{X}\Phi_\mathcal{I}] \vee [p \wedge \Phi_\mathcal{R} \wedge \mathsf{X} p], \\ \mathsf{FG}\Phi_\mathcal{F} \end{pmatrix}$
- $[\varphi \,\underline{\mathsf{U}}\, \mathfrak{A}_\Phi] = \mathcal{A}_\exists \begin{pmatrix} Q \cup \{p\}, \neg p \vee \Phi_\mathcal{I}, \\ [\neg p \wedge \varphi \wedge \neg \mathsf{X} p] \vee \\ [\neg p \wedge \varphi \wedge \mathsf{X} p \wedge \mathsf{X}\Phi_\mathcal{I}] \vee \\ [p \wedge \Phi_\mathcal{R} \wedge \mathsf{X} p], \\ \mathsf{FG}(p \wedge \Phi_\mathcal{F}) \end{pmatrix}$
- $[\varphi \,\mathsf{U}\, \mathfrak{A}_\Phi] = \mathcal{A}_\exists \begin{pmatrix} Q \cup \{p\}, \neg p \vee \Phi_\mathcal{I}, \\ [\neg p \wedge \varphi \wedge \neg \mathsf{X} p] \vee \\ [\neg p \wedge \varphi \wedge \mathsf{X} p \wedge \mathsf{X}\Phi_\mathcal{I}] \vee \\ [p \wedge \Phi_\mathcal{R} \wedge \mathsf{X} p], \\ \mathsf{FG}(\neg p \vee \Phi_\mathcal{F}) \end{pmatrix}$
- $[\mathfrak{A}_\Phi \,\underline{\mathsf{B}}\, \varphi] = \mathcal{A}_\exists \begin{pmatrix} Q \cup \{p\}, \neg \varphi \wedge (\neg p \vee \Phi_\mathcal{I}), \\ [\neg p \wedge \neg \varphi \wedge \neg \mathsf{X} p] \vee \\ [\neg p \wedge \neg \varphi \wedge \mathsf{X} p \wedge \mathsf{X}\Phi_\mathcal{I} \wedge \neg \mathsf{X}\varphi] \vee \\ [p \wedge \Phi_\mathcal{R} \wedge \mathsf{X} p], \\ \mathsf{FG}(p \wedge \Phi_\mathcal{F}) \end{pmatrix}$

- $[\mathfrak{A}_\Phi \mathsf{B}\ \varphi] = \mathcal{A}_\exists \begin{pmatrix} Q \cup \{p\}, \neg\varphi \wedge (\neg p \vee \Phi_\mathcal{I}), \\ [\neg p \wedge \neg\varphi \wedge \neg\mathsf{X}p] \vee \\ [\neg p \wedge \neg\varphi \wedge \mathsf{X}p \wedge \mathsf{X}\Phi_\mathcal{I} \wedge \neg\mathsf{X}\varphi] \vee \\ [p \wedge \Phi_\mathcal{R} \wedge \mathsf{X}p], \\ \mathsf{FG}(\neg p \vee \Phi_\mathcal{F}) \end{pmatrix}$

The rationale behind the above equations is as follows: in case of disjunctions, we add an additional state variable p that may initially be either 1 or 0. According to the transition relation, p will always maintain its initial value. Hence, depending on the initial value of p the automaton obtained for the disjunctive closure collapses to either one of the given automata.

In the closure for the X operator, the additional state variable will initially be 0 and will be 1 for all $t > 0$. Hence, the transition relation formalizes that at time $t = 1$, we are in an initial state, i.e. a state satisfying $\Phi_\mathcal{I}$, and for all points of time $t \geq 1$, the transitions encoded by $\Phi_\mathcal{R}$ are enabled.

The closure of $\underline{\mathsf{U}}$ is constructed as follows: We add new initial states encoded by $\neg p$ (note that $\neg p \vee \Phi_\mathcal{I} = \neg p \vee p \wedge \Phi_\mathcal{I}$ is valid) and retain also the previous initial states (that are extended by a true value of p). We may loop in the new initial states $\neg p$ whenever φ holds, but may also leave these states with the same input to reach an initial state of $p \wedge \Phi_\mathcal{I}$. The second alternative then enables the automaton $\mathfrak{A}$. The closures for $[\mathfrak{A}\ \mathsf{B}\ \varphi]$ and $[\mathfrak{A}\ \underline{\mathsf{B}}\ \varphi]$ are similar, just note that $[\varphi\ \mathsf{B}\ \psi] = [(\neg\psi)\ \mathsf{U}\ (\varphi \wedge \neg\psi)]$ and $[\varphi\ \underline{\mathsf{B}}\ \psi] = [(\neg\psi)\ \underline{\mathsf{U}}\ (\varphi \wedge \neg\psi)]$ are valid.

Büchi automata are sometimes further classified into *weak* and *terminal* Büchi automata [40, 52, 221, 381]. A Büchi automaton is weak iff there exists a partition of its states into classes $Q_0, \ldots, Q_{f+n}$ that are partially ordered so that the transitions of the automaton never move from Q_i to Q_j unless $Q_j \preceq Q_i$ holds. Thus, every cycle is completely contained in a class Q_i. Given that Ψ_i represents the situations of Q_i, it is furthermore assumed that the acceptance condition $\mathsf{GF}\Phi_0$ is given as a disjunction $\Phi_0 = \bigvee_{i \in \mathcal{F}} \Psi_i$ of classes Q_i. Furthermore, a weak Büchi automaton is terminal if the classes Ψ_i contained in the acceptance condition Φ_0 correspond with minimal state sets Q_i or direct successors of minimal state sets w.r.t. $\preceq$, i.e., $\Phi_0 = \bigvee_{i=0}^{f} \Psi_i$. It is easily seen that weak and terminal Büchi automata are special forms of $\mathsf{NDet}_{\mathsf{FG}}$ and $\mathsf{NDet}_{\mathsf{F}}$ automata [52, 502], respectively: one can simply replace the acceptance condition $\mathsf{GF}\Phi_0$ by $\mathsf{FG}\Phi_0$ and $\mathsf{F}\Phi_0$, respectively [52]. Conversely, one can construct for every $\mathsf{NDet}_{\mathsf{FG}}$ and $\mathsf{NDet}_{\mathsf{F}}$ automaton a weak and terminal Büchi automaton [502]. Note that the automata obtained by the closure theorems fulfill the terminal condition.

Recall that the idea of our new translation procedure for $\mathsf{TL}_{\mathsf{PE}}$ is to compute ω-automata in a bottom-up traversal through the syntax tree of the formula, where the ω-automaton of an elementary formula is obtained from the ω-automata of its arguments by application of a corresponding closure operation. For example, consider the formula $[\Phi\ \mathsf{U}\ \Psi]$ where Φ is propositional and we already have translated Ψ to an equivalent $\mathsf{NDet}_{\mathsf{FG}}$ automaton for $\mathfrak{A}_\Psi$.

We then use the above closure theorems to construct an $\mathsf{NDet}_{\mathsf{FG}}$ automaton for $[\Phi\ \mathsf{U}\ \Psi]$ out of $\mathfrak{A}_\Psi$. *It is important to note that by the above closure theorems, every temporal operator is replaced with only a single state variable!* This makes it, in general, more efficient than Borel.

In order to complete this translation, we must show how the base cases of our recursive treatment are obtained. This means that we have to show how only the $\mathsf{TL}_{\mathsf{PE}}$ formulas that have only propositional subformulas (apart from themselves) are translated. These cases are very simple and can, for example, be solved as shown in the following lemma.

Lemma 5.67. *Given that φ and ψ are propositional formulas, then the following rules can be used for a translation to* $\mathsf{NDet}_{\mathsf{FG}}$*:*

- $[\varphi\ \mathsf{U}\ \psi] = \mathcal{A}_\exists\,(\{q\}, \neg q, \neg q \wedge \varphi \wedge \neg\psi \wedge \neg\mathsf{X}q \vee \neg q \wedge \psi \wedge \mathsf{X}q \vee q \wedge \mathsf{X}q, \mathsf{FG1})$
- $[\varphi\ \underline{\mathsf{U}}\ \psi] = \mathcal{A}_\exists\,(\{q\}, \neg q, \neg q \wedge \varphi \wedge \neg\psi \wedge \neg\mathsf{X}q \vee \neg q \wedge \psi \wedge \mathsf{X}q \vee q \wedge \mathsf{X}q, \mathsf{FG}q)$
- $[\varphi\ \mathsf{B}\ \psi] = [(\neg\psi)\ \mathsf{U}\ (\varphi \wedge \neg\psi)]$
- $[\varphi\ \underline{\mathsf{B}}\ \psi] = [(\neg\psi)\ \underline{\mathsf{U}}\ (\varphi \wedge \neg\psi)]$
- $$\mathsf{X}\varphi = \mathcal{A}_\exists \begin{pmatrix} \{p, q\}, \\ \neg p \wedge \neg q, \\ \neg p \wedge \neg q \wedge \mathsf{X}p \wedge \neg\mathsf{X}q \vee \\ p \wedge \neg q \wedge \varphi \wedge \mathsf{X}p \wedge \mathsf{X}q \vee \\ p \wedge q \wedge \mathsf{X}p \wedge \mathsf{X}q, \\ \mathsf{FG1} \end{pmatrix}$$

The reader may draw the state transition diagrams of the above mentioned automata to see that these automata are indeed equivalent to the mentioned temporal logic formulas.

Hence, having extracted $\mathsf{TL}_{\mathsf{FG}}$ formulas, and having furthermore extracted $\mathsf{TL}_{\mathsf{PE}}$ formulas, we have the choice of whether we would like to translate the $\mathsf{TL}_{\mathsf{PE}}$ formulas to a 'CTL formula' or to a $\mathsf{NDet}_{\mathsf{FG}}$ automaton using the above closures.

Hence, we now have many possibilities at hand to translate LTL_p model checking problems to equivalent fixpoint problems. A simple algorithm that combines all of these procedures is listed in Figure 5.18. We assume that BorelFG is the function that consists of the removal of the 'asymmetric' grammar rules as explained in Theorem 5.57 on page 362. The algorithm has three Boolean valued parameters *OnlyMono*, *useBorel*, and *toCTL* to control the translation. If *OnlyMono* is true, then the algorithm will simply use the function TopProp to compute an equivalent $\mathsf{NDet}_{\mathsf{Streett}}$ automaton for Φ. Note, that this function considers the monotonicity improvement and will therefore already translate TL_{G} formulas to equivalent $\mathsf{NDet}_{\mathsf{G}}$ automata. However, for any positive/negative occurrence of a weak/strong future temporal operator, a fairness constraint is generated.

If this is not wanted, we have to set *OnlyMono* $= 1$. In this case, we first use the function TopFG to extract the largest top-level $\mathsf{TL}_{\mathsf{FG}}$ Φ_0 of Φ. The remaining parts are thereby translated by generating fairness constraints as it

```
function Automaton(OnlyMono, useBorel, toCTL, Φ)
  if OnlyMono then return TopProp_1(Φ) end;
  A∃(Q_0, I_0, R_0, F_0, Φ_0) ≡ TopFG_1(Φ);
  if useBorel then (* translate to reachability automaton *)
    A∃(Q_1, I_1, R_1, F_1, Φ_1) ≡ BorelFG_1(Φ_0);
    I_1 := I_1 ∧ Φ_1;
    Φ_1 := ⋀_{ξ∈F_1} ξ
  else (* extract top-level TL_PE formula *)
    A∃(Q_1, I_1, R_1, Φ_1) ≡ TopPE_1(Φ_0)
  end;
  (* in any case, we now have Φ_1 ∈ TL_PE *)
  if toCTL then (* translate to CTL *)
    Φ_2 := LeftCTL2CTL(Φ_1);
    return A∃(Q_0 ∪ Q_1, I_0 ∧ I_1, R_0 ∧ R_1, F_0, Φ_2)
  else (* apply closure theorems *)
    A∃(Q_2, I_2, R_2, Φ_FG) ≡ close(Φ_1);
    return A∃(Q_0 ∪ Q_1 ∪ Q_2, I_0 ∧ I_1 ∧ I_2, R_0 ∧ R_1 ∧ R_2, F_0, Φ_FG)
  end
end function
```

Fig. 5.18. The final translation from LTL_p to ω-Automata or CTL model checking

would have been done in TopProp. The extracted $\Phi_0 \in \mathsf{TL}_{\mathsf{FG}}$, however, can be further translated without the introduction of fairness constraints. Depending on the other parameters *useBorel* and *toCTL* this is done as follows: if *useBorel* is true, we apply the function BorelFG to reduce Φ_0 to a reachability automaton; otherwise we reduce Φ_0 by TopPE to an ω-automaton whose acceptance condition Φ_0 belongs to $\mathsf{TL}_{\mathsf{PE}}$. Of course, the latter will leave more parts of Φ_0 in the acceptance condition, while the function BorelFG will introduce more state variables and a smaller acceptance condition.

Since we replace Φ_1 with the conjunction of the reachability constraints in case *useBorel* is true, it then follows (independent of the value of *useBorel*) that $\Phi_1 \in \mathsf{TL}_{\mathsf{PE}}$. We therefore have two possibilities for the further translation: If *toCTL* is true, we translate $\mathsf{E}\Phi_1$ to an equivalent CTL formula, and drop all path quantifiers in the result (a later translation to a CTL model checking problem will then only have to add these path quantifiers again). If, on the other hand, *toCTL* is false, then we use the closure theorems to compute an equivalent $\mathsf{NDet}_{\mathsf{FG}}$ automaton for Φ_1. After that, we collect the different automaton parts, so that finally the desired ω-automaton is obtained.

Hence, the function given in Figure 5.18 summarizes all of the previous translations in that these can be obtained by using particular instances. In general, it is not clear in advance, which parameters will finally lead to a model checking problem that can be efficiently solved. For this reason, it is necessary in practice to be able to use different translations. To conclude this

section, we embed our temporal logics TL_κ in CTL* so that we obtain the following 'alternation-free' fragment of CTL*:

Definition 5.68 (The Logic AFCTL*). *We define the logic* AFCTL* *as the set of formulas that can be derived from the nonterminal S by the following grammar rules:*

$$S ::= V_\Sigma \mid \neg S \mid S \wedge S \mid S \vee S \mid \mathsf{E}P_{\mathsf{FG}} \mid \mathsf{A}P_{\mathsf{GF}}$$

$$\begin{array}{ll}
P_{\mathsf{G}} ::= S & P_{\mathsf{F}} ::= S \\
\quad \mid \neg P_{\mathsf{F}} \mid P_{\mathsf{G}} \wedge P_{\mathsf{G}} \mid P_{\mathsf{G}} \vee P_{\mathsf{G}} & \quad \mid \neg P_{\mathsf{G}} \mid P_{\mathsf{F}} \wedge P_{\mathsf{F}} \mid P_{\mathsf{F}} \vee P_{\mathsf{F}} \\
\quad \mid \mathsf{X}P_{\mathsf{G}} \mid [P_{\mathsf{G}} \,\mathsf{U}\, P_{\mathsf{G}}] \mid [P_{\mathsf{G}} \,\mathsf{B}\, P_{\mathsf{F}}] & \quad \mid \mathsf{X}P_{\mathsf{F}} \mid [P_{\mathsf{F}} \,\mathsf{U}\, P_{\mathsf{F}}] \mid [P_{\mathsf{F}} \,\mathsf{B}\, P_{\mathsf{G}}] \\
\quad \mid \underline{\overleftarrow{\mathsf{X}}} P_{\mathsf{G}} \mid [P_{\mathsf{G}} \,\underline{\overleftarrow{\mathsf{U}}}\, P_{\mathsf{G}}] \mid [P_{\mathsf{G}} \,\underline{\overleftarrow{\mathsf{B}}}\, P_{\mathsf{F}}] & \quad \mid \underline{\overleftarrow{\mathsf{X}}} P_{\mathsf{F}} \mid [P_{\mathsf{F}} \,\underline{\overleftarrow{\mathsf{U}}}\, P_{\mathsf{F}}] \mid [P_{\mathsf{F}} \,\underline{\overleftarrow{\mathsf{B}}}\, P_{\mathsf{G}}] \\
\quad \mid \overleftarrow{\mathsf{X}} P_{\mathsf{G}} \mid [P_{\mathsf{G}} \,\overleftarrow{\mathsf{U}}\, P_{\mathsf{G}}] \mid [P_{\mathsf{G}} \,\overleftarrow{\mathsf{B}}\, P_{\mathsf{F}}] & \quad \mid \overleftarrow{\mathsf{X}} P_{\mathsf{F}} \mid [P_{\mathsf{F}} \,\overleftarrow{\mathsf{U}}\, P_{\mathsf{F}}] \mid [P_{\mathsf{F}} \,\overleftarrow{\mathsf{B}}\, P_{\mathsf{G}}]
\end{array}$$

$$\begin{array}{ll}
P_{\mathsf{GF}} ::= S & P_{\mathsf{FG}} ::= S \\
\quad \mid \neg P_{\mathsf{FG}} \mid P_{\mathsf{GF}} \wedge P_{\mathsf{GF}} \mid P_{\mathsf{GF}} \vee P_{\mathsf{GF}} & \quad \mid \neg P_{\mathsf{GF}} \mid P_{\mathsf{FG}} \wedge P_{\mathsf{FG}} \mid P_{\mathsf{FG}} \vee P_{\mathsf{FG}} \\
\quad \mid \mathsf{X}P_{\mathsf{GF}} \mid [P_{\mathsf{GF}} \,\underline{\mathsf{U}}\, P_{\mathsf{F}}] \mid [P_{\mathsf{F}} \,\underline{\mathsf{B}}\, P_{\mathsf{FG}}] & \quad \mid \mathsf{X}P_{\mathsf{FG}} \mid [P_{\mathsf{FG}} \,\underline{\mathsf{U}}\, P_{\mathsf{FG}}] \mid [P_{\mathsf{FG}} \,\underline{\mathsf{B}}\, P_{\mathsf{GF}}] \\
\qquad \mid [P_{\mathsf{GF}} \,\mathsf{U}\, P_{\mathsf{GF}}] \mid [P_{\mathsf{GF}} \,\mathsf{B}\, P_{\mathsf{FG}}] & \qquad \mid [P_{\mathsf{G}} \,\mathsf{U}\, P_{\mathsf{FG}}] \mid [P_{\mathsf{FG}} \,\mathsf{B}\, P_{\mathsf{F}}] \\
\quad \mid \underline{\overleftarrow{\mathsf{X}}} P_{\mathsf{GF}} \mid [P_{\mathsf{GF}} \,\underline{\overleftarrow{\mathsf{U}}}\, P_{\mathsf{GF}}] \mid [P_{\mathsf{GF}} \,\underline{\overleftarrow{\mathsf{B}}}\, P_{\mathsf{FG}}] & \quad \mid \underline{\overleftarrow{\mathsf{X}}} P_{\mathsf{FG}} \mid [P_{\mathsf{FG}} \,\underline{\overleftarrow{\mathsf{U}}}\, P_{\mathsf{FG}}] \mid [P_{\mathsf{FG}} \,\underline{\overleftarrow{\mathsf{B}}}\, P_{\mathsf{GF}}] \\
\quad \mid \overleftarrow{\mathsf{X}} P_{\mathsf{GF}} \mid [P_{\mathsf{GF}} \,\overleftarrow{\mathsf{U}}\, P_{\mathsf{GF}}] \mid [P_{\mathsf{GF}} \,\overleftarrow{\mathsf{B}}\, P_{\mathsf{FG}}] & \quad \mid \overleftarrow{\mathsf{X}} P_{\mathsf{FG}} \mid [P_{\mathsf{FG}} \,\overleftarrow{\mathsf{U}}\, P_{\mathsf{FG}}] \mid [P_{\mathsf{FG}} \,\overleftarrow{\mathsf{B}}\, P_{\mathsf{GF}}]
\end{array}$$

Intuitively, AFCTL* formulas are those CTL* formulas where any subformula of the form $\mathsf{E}\Phi$ with quantifier-free Φ 'correspond' to $\mathsf{TL}_{\mathsf{FG}}$. This 'correspondence' is exactly the same as the correspondence between LeftCTL* and the formulas $\mathsf{E}\Phi$ with $\Phi \in \mathsf{TL}_{\mathsf{PE}}$. To make this 'correspondence' more precise, we convert the formulas into PQNF normal form (cf. Theorem 2.49):

Theorem 5.69 (The Logic AFCTL*). *Any formula Φ of the logic* AFCTL* *over the variables V_Σ can be reduced in linear time to its* PQNF *normal form, where $\varphi_i \in \mathsf{TL}_{\mathsf{FG}}$ holds and Ψ is a propositional formula over $V_\Sigma \cup \{x_1, \ldots, x_n\}$:*

$$\Phi = \mathsf{let} \begin{bmatrix} x_1 = \mathsf{E}\varphi_1 \\ \vdots \\ x_n = \mathsf{E}\varphi_n \end{bmatrix} \mathsf{in}\ \Psi\ \mathsf{end}$$

Hence, any formula Φ of the logic AFCTL* *over the variables V_Σ can be reduced in linear time into a formula of the form below, where $\mathfrak{A}_i \in \mathsf{NDet}_{\mathsf{FG}}$ holds and Ψ is a propositional formula over $V_\Sigma \cup \{x_1, \ldots, x_n\}$:*

$$\Phi = \mathsf{let} \begin{bmatrix} x_1 = \mathsf{E}\mathfrak{A}_1 \\ \vdots \\ x_n = \mathsf{E}\mathfrak{A}_n \end{bmatrix} \mathsf{in}\ \Psi\ \mathsf{end}$$

Hence, the model checking problem for AFCTL* *can be reduced to the alternation-free μ-calculus.*

The above defined logic AFCTL* has some interesting properties from both the practical and the theoretical side. For the practice, it is clearly a very interesting logic, since it can be efficiently checked: Using the algorithm BorelFG and then the translation by the closure theorems, allows us to compute the normal form mentioned in the above theorem in time $O(|\Phi|)$. Note that this implies that the size of the formula is also linear w.r.t. $|\Phi|$.

Therefore, the logic is interesting from the practical side. For the theoretical side, it is closely related with Kupferman and Vardi's result [303] that $\mathsf{E}\Phi$ with $\Phi \in \mathsf{LTL}_p$ can be translated to the alternation-free μ-calculus iff Φ can be translated to $\mathsf{Det}_{\mathsf{FG}}$. However, this does not imply that AFCTL* is the alternation-free fragment of the CTL*, even though we prove in the next section that if $\Phi \in \mathsf{LTL}_p$ can be translated to $\mathsf{Det}_{\mathsf{FG}}$, then there is an equivalent formula in $\mathsf{TL}_{\mathsf{FG}}$. Hence, the classes TL_κ are complete with respect to the classes Det_κ. However, this is not sufficient to prove that AFCTL* is equally expressive as the alternation-free fragment of CTL*.

Let us now compare LeftCTL* and AFCTL*, which is equivalent to comparing $\mathsf{TL}_{\mathsf{PE}}$ with $\mathsf{TL}_{\mathsf{FG}}$. Of course, it immediately follows from the syntactical inclusion $\mathsf{TL}_{\mathsf{PE}} \subseteq \mathsf{TL}_{\mathsf{FG}}$ that AFCTL* is at least as expressive as LeftCTL* and hence, as CTL. Clearly, this need not necessarily mean that AFCTL* is more expressive than LeftCTL*.

However, as the formula $\mathsf{E}[[p \vee \mathsf{X}p] \mathsf{U} 0]$ belongs to AFCTL*, but it is well-known not to be expressible in CTL, it immediately follows that LeftCTL* $\precnsim$ AFCTL* holds (just note that $\mathsf{E}[[p \vee \mathsf{X}p] \mathsf{U} 0]$ is equivalent to $\mathsf{EG}[p \vee \mathsf{X}p]$, which is the negation of $\mathsf{AF}[\neg p \wedge \neg \mathsf{X}p]$, whose inequivalence to CTL formulas is shown in [162]). We therefore have the following result:

Theorem 5.70. *The logics* $\mathsf{TL}_{\mathsf{PE}}$, $\mathsf{TL}_{\mathsf{PA}}$, *and* LeftCTL* *are strictly less expressive than the logics* $\mathsf{TL}_{\mathsf{FG}}$, $\mathsf{TL}_{\mathsf{GF}}$, *and* AFCTL*, *respectively.*

Of course, some AFCTL* formulas that are syntactically not LeftCTL* formulas can be replaced by equivalent LeftCTL* formulas. For example, the following is such a reduction:

$$[[\alpha \mathbin{\underline{\mathsf{U}}} \beta] \mathbin{\underline{\mathsf{U}}} \gamma] = \begin{pmatrix} \gamma \vee \\ [(\alpha \vee \beta) \mathbin{\underline{\mathsf{U}}} (\beta \wedge [\beta \mathbin{\underline{\mathsf{U}}} \gamma])] \vee \\ [(\alpha \vee \beta) \mathbin{\underline{\mathsf{U}}} [(\alpha \wedge \neg\beta) \mathbin{\underline{\mathsf{U}}} (\gamma \wedge [\alpha \mathbin{\underline{\mathsf{U}}} \beta])]] \end{pmatrix}$$

Further examples of that kind are the reductions that we used to translate CTL^2 to CTL (cf. Section 5.3.2). Dams has shown in [136] that the formula $[(r \vee [p \mathbin{\underline{\mathsf{U}}} (q \wedge [q \mathbin{\underline{\mathsf{U}}} r])]) \mathbin{\underline{\mathsf{U}}} (s \wedge [s \mathbin{\underline{\mathsf{U}}} (t \wedge [t \mathbin{\underline{\mathsf{U}}} u]))]]$ can not be written without a left nesting of until operators (and has explained that this is somehow the simplest example). However, it is easily seen that this formula belongs to TL_{F} and can thus be translated to a $\mathsf{Det}_{\mathsf{F}}$ automaton.

As further related work, Etessami and Wilke considered in [188] the until hierarchy of temporal logic, i.e., the sets of temporal logic formulas where k until operators are nested, and proved that this hierarchy of formulas is

strictly increasing in its expressiveness. So, it is not always possible to eliminate nestings of until operators.

Hence, not any formula of AFCTL* can be translated to CTL, but it can nevertheless be translated to the alternation-free μ-calculus. Please note that the difference between AFCTL* and LeftCTL* is that AFCTL* does allow P_{G} formulas, where LeftCTL* needs to have a state formula S. This allows us to construct formulas such as the ones above that are not expressible in $\mathsf{TL}_{\mathsf{PE}}$.

5.5 Completeness and Expressiveness of Temporal Logic

In the previous section, we have developed translation procedures from temporal logics to ω-automata. In particular, we have seen that the classes TL_κ can be translated to the corresponding classes Det_κ. We now turn to the question of completeness, i.e., whether Det_κ automata can be translated to TL_κ. We will see that this is not the case, which immediately follows from various well-known results that imply that temporal logic is in general not as expressive as ω-automata, since the latter provide some form of second order quantification (see next chapter). However, we will see that the classes TL_κ are complete w.r.t. noncounting Det_κ automata (cf. Definition 4.61), which are those Det_κ automata that do not have this second order capability.

To this end, we will first list some well-known results on the expressiveness of temporal logics in Section 5.5.1. The expressiveness of LTL_p has many characterizations, and is still a topic of research. Unfortunately, most proofs of this issue are highly complex and are often obtained indirectly by other formalisms (cf. Theorem 5.76 and 5.82). In general, there are two techniques to prove these results: One is based on the Krohn-Rhodes theorem [151, 209, 309, 341, 342, 475], which concerns the decomposition theory of semigroups. The other follows Schützenberger's proof [402, 444, 445] that is based on a certain induction over the syntactic monoids of languages. In Section 5.5.1, we will follow Wilke's direct proof [512] of the fact that every deterministic noncounting automaton can be translated to LTL_p. The proof is constructive and provides an algorithm to compute temporal logic formulas for given automata.

In Section 5.5.2, we show the completeness of the logics TL_κ with respect to noncounting Det_κ automata. To this end, we show that the 'past fragments' of our logics TL_κ correspond with Manna and Pnueli's normal forms for safety, guarantee, obligation, recurrence, persistence, and reactivity properties. By the techniques of Section 5.5.2, it also follows that noncounting deterministic ω-automata can be translated to LTL_p.

Finally, we will prove the completeness of the future time fragments of the logics TL_κ in Section 5.5.3, in that we show that the logics TL_κ have the separation property: Every formula with future and past temporal operators can be converted in a normal form where future and past operators are sep-

arated. In particular, all past operators can therefore be eliminated in these logics (w.r.t. initial validity).

5.5.1 Noncounting Automata and Temporal Logic

We have already mentioned that ω-automata are strictly more expressive than temporal logics. The reason for this will become clearer in the next chapter where we will prove that LTL_p corresponds with the first order theory of linear order, while ω-automata correspond with the second order theory of linear order. Hence, ω-automata provide means for second order quantification, which is not available in LTL_p. For example, there is no LTL_p formula that can express that a proposition φ should hold at every even point of time [518, 519]. In general, automata have the ability to store information in their states which enables them to 'count modulo a constant number n'. This is not possible in any of the temporal logics we consider in this book.

The inability of temporal logics to 'count' has been known for a long time. In 1960, Büchi proved that the monadic second order logic of one successor S1S can be reduced to $\mathsf{NDet}_{\mathsf{GF}}$ automata [80, 81, 455] (see next Chapter), and thereby initiated research in the field of ω-automata. It is immediately seen that S1S is equal expressive as the monadic second order logic of linear orders $\mathsf{MSO}_<$ (cf. next Chapter). Hence, it was a major achievement of Büchi to show that the mathematical theories S1S and $\mathsf{MSO}_<$ are both decidable, and that decision procedures can be obtained by a suitable reduction to automata problems. At that time, an important problem was finding characterizations for the *first* order logic of linear orders $\mathsf{MFO}_<$. In 1968, Kamp [274] proved that $\mathsf{MFO}_<$ is equal expressive as our logic LTL_p and therefore presented a predicate logic characterization of temporal logic. Hence, it became clear that LTL_p is the subset of $\mathsf{MSO}_<$ that is restricted to first order quantification. But still a characterization of the subset of ω-automata that correspond with $\mathsf{MFO}_<$ was unknown. The question was then answered by McNaughton and Papert [364] in 1971. They introduced the notion of 'noncounting' automata and proved the equal expressiveness of $\mathsf{MFO}_<$ and star-free regular expressions.

Therefore, considering again the issue of completeness of the logics TL_κ, we must refine our question in that we ask *whether it is possible to translate any noncounting* Det_κ *automaton to an equivalent* TL_κ *formula*. We will see in this section, that the answer to that question is 'yes' for all classes κ, and therefore we have our desired completeness results for the logics TL_κ. To establish this result, we will not directly pose our considerations on noncounting Det_κ automata. Instead, we will consider another characterization of these automata which is directly related to temporal logic. We will consider temporally defined automata as introduced by Lichtenstein, Pnueli, and Zuck [326], and later used by Maler and Pnueli [342] in 1990, and Manna and Pnueli [352] in 1992.

The intuitive idea of temporally defined automata is that we first forget about their acceptance conditions, thus we consider semiautomata $\mathfrak{A} = (\Sigma, \mathcal{S}, \mathcal{I}, \mathcal{R})$. We then consider the *regular languages* $L_{s,s'}$ whose words have a run leading from state s to state s'. *A semiautomaton $\mathfrak{A}$ is then temporally defined iff all languages $L_{s,s'}$ can be characterized by temporal logic formulas.* Of course, we have to make precise how 'characterized' is to be understood. To this end, we restrict our consideration to automata whose input alphabet is encoded by Boolean variables V_Σ that we furthermore use to build formulas of LTL_p. As we are only interested in finite words at the moment, we only consider LTL_p formulas that do not contain future temporal operators:

Definition 5.71 (End-Satisfiability of Past Formulas). *Given a pure past formula φ, we define the following set* $\mathsf{ESat}(\varphi)$ *of finite words over* 2^{V_Σ}*:*

$$\mathsf{ESat}(\varphi) := \left\{ (\alpha^{(0)}, \ldots, \alpha^{(\ell)}) \in (2^{V_\Sigma})^* \;\middle|\; \begin{array}{l} \exists \mathcal{K}. \exists \pi \in \mathsf{Paths}_{\mathcal{K}}(\pi^{(0)}). \\ \quad (\mathcal{K}, \pi, \ell) \models \varphi \wedge \\ \quad \forall t \leq \ell.\ \alpha^{(t)} = \mathcal{L}(\pi^{(t)}) \end{array} \right\}$$

We write $(\alpha^{(0)}, \ldots, \alpha^{(\ell)}) \models \varphi$ to denote that $(\alpha^{(0)}, \ldots, \alpha^{(\ell)}) \in \mathsf{ESat}(\varphi)$ holds.

For example, for $V_\Sigma = \{a, b, c\}$, we have $(\{\}, \{b\}, \{a, c\}, \{a\}, \{a, c\}) \models [a \mathbin{\overleftarrow{\mathsf{U}}} b]$. It is not complicated to determine $\mathsf{ESat}(\varphi)$ for a given formula LTL_p where only past temporal operators occur: We construct the automaton $\mathfrak{A}_\varphi$ that is obtained for a pure past formula φ by the repeated application of the laws of Theorem 5.38 on page 338. We thereby obtain an automaton formula $\mathfrak{A}_\varphi = \mathcal{A}_\exists(Q, \varphi_\mathcal{I}, \varphi_\mathcal{R}, \varphi_\mathcal{F})$, where $\varphi_\mathcal{F}$ is propositional. Moreover, the automaton has a deterministic and complete transition relation $\varphi_\mathcal{R}$, and exactly one initial state ι. $\mathsf{ESat}(\varphi)$ is then obtained as the union of languages $L_{\iota,s'}$ where $L_{\iota,s'}$ is the set of finite words that are read by $\mathfrak{A}_\varphi$ from the initial state ι such that the computation ends in a state where with the last letter of the word the condition $\varphi_\mathcal{F}$ holds.

Proceeding this way, it is not difficult to see that the syntactic monoid $\mathsf{SM}(\mathsf{ESat}(\varphi))$ of the language $\mathsf{ESat}(\varphi)$ is aperiodic. We can prove this by induction on the number of temporal operators in φ. The induction base is thereby trivial, since for propositional formulas φ, we obtain the automaton $\mathcal{A}_\exists(\{\}, \varphi, 1, 1)$ with no state variables, and hence, only one state. The induction step follows from the following lemma, where we also introduce the *cascade product* of two automata:

Lemma 5.72 (Aperiodicity of Cascade Product). *Given semiautomata $\mathfrak{A} = (\Sigma, \mathcal{S}^\mathfrak{A}, \mathcal{I}^\mathfrak{A}, \mathcal{R}^\mathfrak{A})$ and $\mathfrak{B} = (\Sigma \times \mathcal{S}^\mathfrak{A}, \mathcal{S}^\mathfrak{B}, \mathcal{I}^\mathfrak{B}, \mathcal{R}^\mathfrak{B})$, whose transition monoids $\mathsf{TM}(\mathfrak{A})$ and $\mathsf{TM}(\mathfrak{B})$ are aperiodic. Then, the transition monoid $\mathsf{TM}(\mathfrak{C})$ of any*[5] *semiautomaton $\mathfrak{C} := (\Sigma, \mathcal{S}^\mathfrak{A} \times \mathcal{S}^\mathfrak{B}, \mathcal{I}^\mathfrak{C}, \mathcal{R}^\mathfrak{C})$ with $\mathcal{R}^\mathfrak{C}$ defined as below (called the cascade product of $\mathfrak{A}$ and $\mathfrak{B}$) is also aperiodic:*

$$((s_1, s_2), \sigma, (s_1', s_2')) \in \mathcal{R}^\mathfrak{C} :\Leftrightarrow (s_1, \sigma, s_1') \in \mathcal{R}^\mathfrak{A} \wedge (s_2, (\sigma, s_1), s_2') \in \mathcal{R}^\mathfrak{B}$$

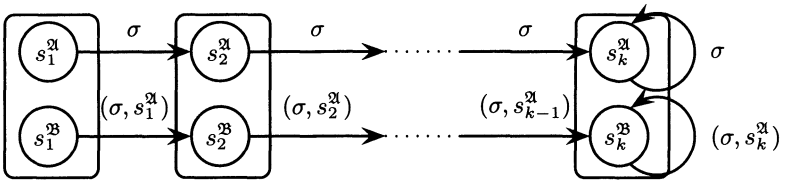


Fig. 5.19. A path through the cascade product of two automata $\mathfrak{A}$ and $\mathfrak{B}$

Proof. For the proof, we consider what happens when we repeatedly read a letter $\sigma \in \Sigma$ in the cascade product $\mathfrak{C}$. The corresponding run is given in Figure 5.19. As $\mathsf{TM}(\mathfrak{A})$ is aperiodic, there is a number n such that $\mathcal{R}^{\mathfrak{A}}_{\sigma^n} = \mathcal{R}^{\mathfrak{A}}_{\sigma^{n+1}}$ holds. Hence, after having read n times σ, we end in a self-loop in the semiautomaton $\mathfrak{A}$. Up to this point, we have read different letters $(\sigma, s_1^{\mathfrak{A}})$, $(\sigma, s_2^{\mathfrak{A}})$, ... in the semiautomaton $\mathfrak{B}$, but after this point, we only read $(\sigma, s_n^{\mathfrak{A}})$, since $\mathfrak{A}$ is now caught in a self-loop. As $\mathsf{TM}(\mathfrak{B})$ is also aperiodic, it follows that after some further transitions, also $\mathfrak{B}$ must be caught in a self-loop, so that it follows that $\mathsf{TM}(\mathfrak{C})$ is also aperiodic. □

We can moreover apply the above lemma to the $\mathsf{NDet}_{\mathsf{Streett}}$ automata that are obtained by our translations of LTL_p by function Streett of Figure 5.11 on page 342 and function TopProp of Figure 5.12 on page 347. *Indeed, the automata that are constructed by these procedures are cascade products of automata with only two states (every temporal operator of the formula corresponds with such a two-state automaton.* This decomposition is closely related with the decomposition stated by the Krohn-Rhodes decomposition theorem [151, 209, 309, 475].

We can therefore conclude that we can translate every $\varphi \in \mathsf{LTL}_p$ to an equivalent ω-automaton $\mathfrak{A}_\varphi$ whose transition monoid $\mathsf{TM}(\mathfrak{A})$ is aperiodic. Hence, it follows by Theorem 4.58 on page 259 that the syntactic monoids of the languages $L_{s,s'}$ for every pair (s, s') are aperiodic.

In the following, regular languages with aperiodic syntactic monoid are called 'star-free'. This notion is due to the further characterization that these languages can be described with star-free regular expressions. Hence, we obtain the following result for the regular languages $\mathsf{ESat}(\varphi)$:

Theorem 5.73 (LTL_p and Noncounting Automata). *For every formula φ of the past fragment of LTL_p, the syntactic monoid $\mathsf{SM}(\mathsf{ESat}(\varphi))$ of the language $\mathsf{ESat}(\varphi)$ is aperiodic. In particular, the automata constructed by function* Streett *of Figure 5.11 on page 342 and function* TopProp *of Figure 5.12 on page 347 are noncounting.*

The above theorem establishes one part of the relationship between noncounting automata and temporal logic. The other part is more complicated: We have to show that for a noncounting automaton $\mathfrak{A}$ the languages $L_{s,s'}$ can be described by pure past temporal logic formulas $\varphi_{s,s'}$ such that $L_{s,s'} =$

[5] We have the choice for $\mathcal{I}^{\mathfrak{C}}$.

$\mathsf{ESat}(\varphi_{s,s'})$ holds. To this end, we first note that we have the following relationship between the languages $L_{s,s'}$ and the classes of the transition monoid (note that the union is a finite one):

$$L_{s,s'} := \bigcup_{(s,s')\in\mathcal{R}_\alpha} [\alpha]_{\equiv_{\mathfrak{A}}}$$

Hence, it is obviously enough to characterize the finitely many classes $[\alpha]_{\equiv_{\mathfrak{A}}}$ by pure past formulas, i.e., to define pure past formulas φ_α such that $[\alpha]_{\equiv_{\mathfrak{A}}} = \mathsf{ESat}(\varphi_\alpha)$ holds. This is shown in the following theorem, whose proof has been taken from [512]:

Theorem 5.74 (Translating Noncounting Automata to LTL_p). *If a deterministic and complete semiautomaton $\mathfrak{A} = (\Sigma, \mathcal{S}, \mathcal{I}, \mathcal{R})$ is noncounting, then every class $[\alpha]_{\equiv_{\mathfrak{A}}}$ of its transition monoid can be characterized by a pure past temporal logic formula φ_α, i.e., φ_α satisfies the equation $[\alpha]_{\equiv_{\mathfrak{A}}} = \mathsf{ESat}(\varphi_\alpha)$.*

Proof. Given a deterministic and complete semiautomaton $\mathfrak{A} = (\Sigma, \mathcal{S}, \mathcal{I}, \mathcal{R})$ that is noncounting, we will construct a temporal logic formula φ_α to describe $[\alpha]_{\equiv_{\mathfrak{A}}}$. To simplify the notation, we use the letters of the alphabet Σ as atomic propositions of the temporal logic. The correct form would require that Σ is the powerset of the variables of the temporal logic, and that a letter ϑ is translated to $\mathsf{minterm}_{V_\Sigma}(\vartheta)$.

The following proof is done by a well-founded induction over all deterministic complete noncounting semiautomata. For this purpose, we define the following binary relation on semiautomata:

$$\begin{aligned}(\Sigma^{\mathfrak{A}}, \mathcal{S}^{\mathfrak{A}}, \mathcal{I}^{\mathfrak{A}}, \mathcal{R}^{\mathfrak{A}}) &\preceq (\Sigma^{\mathfrak{B}}, \mathcal{S}^{\mathfrak{B}}, \mathcal{I}^{\mathfrak{B}}, \mathcal{R}^{\mathfrak{B}}) \\ &:\Leftrightarrow \left|\mathcal{S}^{\mathfrak{A}}\right| < \left|\mathcal{S}^{\mathfrak{B}}\right| \vee \left|\mathcal{S}^{\mathfrak{A}}\right| = \left|\mathcal{S}^{\mathfrak{B}}\right| \wedge \left|\Sigma^{\mathfrak{A}}\right| \leq \left|\Sigma^{\mathfrak{B}}\right|\end{aligned}$$

It is easily seen that $\preceq$ is reflexive and transitive, and will be antisymmetric when we identify semiautomata with the same number of states and inputs. Furthermore, there is then a least semiautomaton (up to isomorphism) that has only one state and one input letter. Hence, $\preceq$ is a well-founded ordering that can be used for a well-founded induction. In the induction base, we consider a semiautomaton $\mathfrak{A} = (\Sigma, \mathcal{S}, \mathcal{I}, \mathcal{R})$ with $|\mathcal{S}| = |\Sigma| = 1$. This is trivial: if $\Sigma = \{\sigma\}$, then define $\varphi_\alpha := \overleftarrow{\mathsf{G}}\sigma$.

For the induction step, we consider two cases: in the first case, assume that there is no $\sigma \in \Sigma$ such that $\mathsf{suc}_\exists^{\mathcal{R}_\sigma}(\mathcal{S}) \subsetneq \mathcal{S}$. Hence, for all $\sigma \in \Sigma$, we have $\mathsf{suc}_\exists^{\mathcal{R}_\sigma}(\mathcal{S}) = \mathcal{S}$, which implies due to the noncounting property (cf. Theorem 4.63 on page 263) that $\mathsf{suc}_\exists^{\mathcal{R}_\sigma}(\{s\}) = \{s\}$ holds. Hence, all transitions under all input letters $\sigma \in \Sigma$ are self-loops, and hence we have for each word $\alpha \in \Sigma^*$ the equation $[\alpha]_{\equiv_{\mathfrak{A}}} = \Sigma^*$. The language Σ^* is characterized by the formula $\overleftarrow{\mathsf{G}} \bigvee_{\sigma\in\Sigma} \sigma$.

The remaining case of the induction step is more difficult. Here, we have at least one input $\sigma \in \Sigma$ where $\mathsf{suc}_\exists^{\mathcal{R}_\sigma}(\mathcal{S}) \subsetneq \mathcal{S}$. This means, there is at least

one state $s \in \mathcal{S}$ that can not be reached from any other state by reading σ. We use this input letter to construct other automata with either fewer states or the same states, but fewer inputs. To this end, it will be convenient to abbreviate $\Sigma_\sigma := \Sigma \setminus \{\sigma\}$ and $\mathcal{S}_\sigma := \mathrm{suc}_\exists^{\mathcal{R}_\sigma}(\mathcal{S})$. Using this, we define two further semiautomata (we ignore the initial sets since they do not matter):

- $\mathfrak{B} := (\Sigma_\sigma, \mathcal{S}, \{(s,a,s') \in \mathcal{R} \mid a \in \Sigma_\sigma\})$
- $\mathfrak{C} := (\Sigma^{\mathfrak{C}}, \mathcal{S}_\sigma, \mathcal{R}^{\mathfrak{C}})$ with input alphabet $\Sigma^{\mathfrak{C}} := \{[\gamma\sigma]_{\equiv_{\mathfrak{A}}} \mid \gamma \in \Sigma_\sigma^*\}$ and transition relation $(s, [\gamma\sigma]_{\equiv_{\mathfrak{A}}}, s') \in \mathcal{R}^{\mathfrak{C}} :\Leftrightarrow (s,s') \in \mathcal{R}^{\mathfrak{A}}_{\gamma\sigma}$

The semiautomata $\mathfrak{B}$ and $\mathfrak{C}$ are used in the following to construct the formula φ_α. Note that the input letters of $\mathfrak{C}$ are classes $[\gamma\sigma]_{\equiv_{\mathfrak{A}}}$ with $\gamma \in \Sigma_\sigma^*$, i.e., classes of words on Σ that end with the letter σ. Hence, we have to consider finite words of these classes for $\mathfrak{C}$, i.e., words $[\gamma_1\sigma]_{\equiv_{\mathfrak{A}}}[\gamma_2\sigma]_{\equiv_{\mathfrak{A}}}\dots[\gamma_n\sigma]_{\equiv_{\mathfrak{A}}}$ with $\gamma_i \in \Sigma_\sigma^*$. We first list some facts of these semiautomata that obviously hold:

Fact 1: $\mathfrak{B} \preceq \mathfrak{A}$ and $\mathfrak{C} \preceq \mathfrak{A}$, and both $\mathfrak{B}$ and $\mathfrak{C}$ are deterministic, complete and noncounting, so that the induction hypothesis can be applied both to $\mathfrak{B}$ and $\mathfrak{C}$. Therefore, we can find by the induction hypothesis for all classes $[\beta]_{\equiv_{\mathfrak{B}}}$ and $[\gamma\sigma]_{\equiv_{\mathfrak{C}}}$, characterizing formulas φ_β and $\varphi_{\gamma\sigma}$.

Fact 2: for every word $\alpha \in \Sigma_\sigma^*$, we have $[\alpha]_{\equiv_{\mathfrak{A}}} \cap \Sigma_\sigma^* = [\equiv_\alpha]_{\mathfrak{B}}$

Fact 3: for every word $\gamma_1\sigma\gamma_2\sigma\dots\gamma_n\sigma$ with $\gamma_i \in \Sigma_\sigma^*$, we have $\mathcal{R}^{\mathfrak{A}}_{\gamma_1\sigma\dots\gamma_n\sigma} = \mathcal{R}^{\mathfrak{C}}_{[\gamma_1\sigma]_{\equiv_{\mathfrak{A}}}\dots[\gamma_n\sigma]_{\equiv_{\mathfrak{A}}}}$

Fact 4: for all words $\gamma_1, \dots, \gamma_n \in \Sigma_\sigma^*$, we have the following:

$$
\begin{aligned}
&\big[[\gamma_1\sigma]_{\equiv_{\mathfrak{A}}} \dots [\gamma_n\sigma]_{\equiv_{\mathfrak{A}}}\big]_{\mathfrak{C}} \\
&\quad = \{[\delta_1\sigma]_{\equiv_{\mathfrak{A}}} \dots [\delta_n\sigma]_{\equiv_{\mathfrak{A}}} \mid [\gamma_1\sigma]_{\equiv_{\mathfrak{A}}} \dots [\gamma_n\sigma]_{\equiv_{\mathfrak{A}}} \equiv_{\mathfrak{C}} [\delta_1\sigma]_{\equiv_{\mathfrak{A}}} \dots [\delta_n\sigma]_{\equiv_{\mathfrak{A}}}\} \\
&\quad = \{[\delta_1\sigma]_{\equiv_{\mathfrak{A}}} \dots [\delta_n\sigma]_{\equiv_{\mathfrak{A}}} \mid \mathcal{R}^{\mathfrak{C}}_{[\gamma_1\sigma]_{\equiv_{\mathfrak{A}}}\dots[\gamma_n\sigma]_{\equiv_{\mathfrak{A}}}} = \mathcal{R}^{\mathfrak{C}}_{[\delta_1\sigma]_{\equiv_{\mathfrak{A}}}\dots[\delta_n\sigma]_{\equiv_{\mathfrak{A}}}}\} \\
&\quad = \{[\delta_1\sigma]_{\equiv_{\mathfrak{A}}} \dots [\delta_n\sigma]_{\equiv_{\mathfrak{A}}} \mid \mathcal{R}^{\mathfrak{A}}_{\gamma_1\sigma\dots\gamma_n\sigma} = \mathcal{R}^{\mathfrak{A}}_{\delta_1\sigma\dots\delta_n\sigma}\} \\
&\quad = \{[\delta_1\sigma]_{\equiv_{\mathfrak{A}}} \dots [\delta_n\sigma]_{\equiv_{\mathfrak{A}}} \mid [\gamma_1\sigma\dots\gamma_n\sigma]_{\equiv_{\mathfrak{A}}} = [\delta_1\sigma\dots\delta_n\sigma]_{\equiv_{\mathfrak{A}}}\} \\
&\quad = \{[\delta_1\sigma\dots\delta_n\sigma]_{\equiv_{\mathfrak{A}}} \mid [\gamma_1\sigma\dots\gamma_n\sigma]_{\equiv_{\mathfrak{A}}} = [\delta_1\sigma\dots\delta_n\sigma]_{\equiv_{\mathfrak{A}}}\} \\
&\quad = \{[\gamma_1\sigma\dots\gamma_n\sigma]_{\equiv_{\mathfrak{A}}}\}
\end{aligned}
$$

We next consider a particular language $[\alpha]_{\equiv_{\mathfrak{A}}}$ and partition it into the set of words with none, one, or more than one occurrences of the letter $\sigma \in \Sigma$:

$$
[\alpha]_{\equiv_{\mathfrak{A}}} = \underbrace{\big([\alpha]_{\equiv_{\mathfrak{A}}} \cap \Sigma_\sigma^*\big)}_{L_0} \cup \underbrace{\big([\alpha]_{\equiv_{\mathfrak{A}}} \cap \Sigma_\sigma^*\sigma\Sigma_\sigma^*\big)}_{L_1} \cup \underbrace{\big([\alpha]_{\equiv_{\mathfrak{A}}} \cap \Sigma_\sigma^*\sigma\Sigma^*\sigma\Sigma_\sigma^*\big)}_{L_{>1}}
$$

Now, note the following:

- $L_0 := [\alpha]_{\equiv_{\mathfrak{A}}} \cap \Sigma_\sigma^*$ is the set of words of $[\alpha]_{\equiv_{\mathfrak{A}}}$ that do not contain the letter σ. As by fact 2, $[\alpha]_{\equiv_{\mathfrak{A}}} \cap \Sigma_\sigma^* = [\alpha]_{\equiv_{\mathfrak{B}}}$ holds, we can characterize this language by a pure past formula according to the induction hypothesis for $\mathfrak{B}$.

- $L_1 := [\alpha]_{\equiv_{\mathfrak{A}}} \cap \Sigma_\sigma^* \sigma \Sigma_\sigma^*$ is the set of words of $[\alpha]_{\equiv_{\mathfrak{A}}}$ that contain exactly one occurrence of the letter σ. Hence, we have $[\alpha]_{\equiv_{\mathfrak{A}}} \cap \Sigma_\sigma^* \sigma \Sigma_\sigma^* = \{\gamma\sigma\gamma' \mid \gamma, \gamma' \in \Sigma_\sigma^* \wedge \gamma\sigma\gamma' \equiv_{\mathfrak{A}} \alpha\}$. We can transform this infinite description of L_1 into a finite one by showing that L_1 is the following *finite* union:

$$[\alpha]_{\equiv_{\mathfrak{A}}} \cap \Sigma_\sigma^* \sigma \Sigma_\sigma^* = \bigcup_{\substack{\gamma, \gamma' \in \Sigma_\sigma^*, \\ \gamma\sigma\gamma' \equiv_{\mathfrak{A}} \alpha}} [\gamma]_{\equiv_{\mathfrak{B}}} \sigma \Sigma_\sigma^* \cap \Sigma_\sigma^* \sigma [\gamma']_{\equiv_{\mathfrak{B}}}$$

  For the proof, simply consider the two inclusions $\subseteq$ and $\supseteq$: Given that $\gamma\sigma\gamma' \in L_1$ with $\gamma, \gamma' \in \Sigma_\sigma^*$ and $\gamma\sigma\gamma' \equiv_{\mathfrak{A}} \alpha$, then $\gamma\sigma\gamma'$ also belongs the set on the right hand side. In particular, $\gamma\sigma\gamma'$ belongs to $[\gamma]_{\equiv_{\mathfrak{B}}} \sigma \Sigma_\sigma^*$ and also to $\Sigma_\sigma^* \sigma [\gamma']_{\equiv_{\mathfrak{B}}}$. For the converse, consider a word that belongs to the set on the right hand side. It is obviously of the form $\gamma\sigma\gamma'$ with $\gamma, \gamma' \in \Sigma_\sigma^*$ and $\gamma\sigma\gamma' \equiv_{\mathfrak{A}} \alpha$ and therefore belongs to L_1.
  By the induction hypothesis, we already have formulas $\Psi_\gamma^{\mathfrak{B}}$ and $\Psi_{\gamma'}^{\mathfrak{B}}$ that characterize the languages $[\gamma]_{\equiv_{\mathfrak{B}}}$ and $[\gamma']_{\equiv_{\mathfrak{B}}}$, respectively. Hence, it is not too complicated to construct a characterizing formula for L_1 in that we construct characterizing formulas for $[\gamma]_{\equiv_{\mathfrak{B}}} \sigma \Sigma_\sigma^*$ and $\Sigma_\sigma^* \sigma [\gamma']_{\equiv_{\mathfrak{B}}}$ (see [512] for the formulas).
- $L_{>1} := [\alpha]_{\equiv_{\mathfrak{A}}} \cap \Sigma_\sigma^* \sigma \Sigma^* \sigma \Sigma_\sigma^*$ is the set of words of $[\alpha]_{\equiv_{\mathfrak{A}}}$ that contain at least two occurrences of the letter σ. These words are of the form $\beta\sigma\gamma\beta'$ with $\beta, \beta' \in \Sigma_\sigma^*$ and a word γ ending with the letter σ. By fact 4 that we mentioned before, we have already seen that for words $\gamma \in \Sigma^*\sigma$, we have $\big[[\gamma]_{\equiv_{\mathfrak{A}}}\big]_{\equiv_{\mathfrak{C}}} = \{[\gamma]_{\equiv_{\mathfrak{A}}}\}$. The classes of $\mathsf{TM}(\mathfrak{C})$ are therefore singleton sets that consist of those classes $[\gamma]_{\equiv_{\mathfrak{A}}}$ of $\mathsf{TM}(\mathfrak{A})$, where the word γ ends with the letter σ. If we therefore define the mapping $\hbar$ such that $\hbar(\big[[\gamma]_{\equiv_{\mathfrak{A}}}\big]_{\equiv_{\mathfrak{C}}}) := [\gamma]_{\equiv_{\mathfrak{A}}}$ holds, then we have $[\beta\sigma\gamma\beta']_{\equiv_{\mathfrak{A}}} = [\beta]_{\equiv_{\mathfrak{A}}} [\sigma]_{\equiv_{\mathfrak{A}}} [\gamma]_{\equiv_{\mathfrak{A}}} [\beta']_{\equiv_{\mathfrak{A}}} = [\beta]_{\equiv_{\mathfrak{A}}} [\sigma]_{\equiv_{\mathfrak{A}}} \hbar(\big[[\gamma]_{\equiv_{\mathfrak{A}}}\big]_{\equiv_{\mathfrak{C}}}) [\beta']_{\equiv_{\mathfrak{A}}}$, and therefore:

$$[\alpha]_{\equiv_{\mathfrak{A}}} \cap \Sigma_\sigma^* \sigma \Sigma^* \sigma \Sigma_\sigma^* = \bigcup_{\substack{\beta, \beta' \in \Sigma_b^*, \\ \gamma \in \Sigma^*\sigma, \\ \beta\sigma\gamma\beta' \equiv_{\mathfrak{A}} \alpha}} [\beta]_{\equiv_{\mathfrak{B}}} \sigma \Sigma^* \cap \Sigma_\sigma^* \sigma \hbar(\big[[\gamma]_{\equiv_{\mathfrak{A}}}\big]_{\equiv_{\mathfrak{C}}}) \Sigma_\sigma^* \cap \Sigma^* \sigma [\beta']_{\equiv_{\mathfrak{B}}}$$

  The proof of the above equation is again simply obtained by considering the two set inclusions of the sets on the left hand side and on the right hand side. Note that the union is again finite since $\mathsf{TM}(\mathfrak{B})$ and $\mathsf{TM}(\mathfrak{C})$ are finite monoids. It now follows by induction hypothesis that we can find characterizing pure past formulas to characterize all the classes $[\beta]_{\equiv_{\mathfrak{B}}}$ and $\big[[\gamma]_{\equiv_{\mathfrak{A}}}\big]_{\equiv_{\mathfrak{C}}}$ of the monoids $\mathsf{TM}(\mathfrak{B})$ and $\mathsf{TM}(\mathfrak{C})$. It is then again not too difficult to construct characterizing formulas for $[\beta]_{\equiv_{\mathfrak{B}}} \sigma \Sigma^*$, $\Sigma^* \sigma [\beta']_{\equiv_{\mathfrak{B}}}$, and $\Sigma_\sigma^* \sigma \hbar(\big[[\gamma]_{\equiv_{\mathfrak{A}}}\big]_{\equiv_{\mathfrak{C}}}) \Sigma_\sigma^*$ (see [512] for these formulas).

□

The above two theorems establish therefore the close relationship between temporal logic and noncounting automata. The above proof is a variant

of Schützenberger's proof that avoids, however, group theory. An indirect translation from noncounting automata to $\mathsf{MFO}_<$ and then to temporal logic is even simpler. The reader may rework the above proof for a translation to $\mathsf{MFO}_<$. The final translation from $\mathsf{MFO}_<$ to temporal logic is shown in the next chapter.

Theorem 5.75 (LTL_p and Noncounting Automata). *For every regular language $L \subseteq \Sigma^*$, the following is equivalent:*

- *there is a pure past formula φ with $L = \mathsf{ESat}(\varphi)$*
- *there is a temporally defined automaton that accepts L*
- *there is a noncounting automaton that accepts L*
- $\mathsf{SM}(L)$ *is aperiodic*

Note however, that there are counting automata that are nevertheless equivalent to LTL_p formulas, but then there are equivalent noncounting automata. For example, consider the automaton given in Figure 5.20. It is easily seen that its transition monoid is aperiodic with period 2. Nevertheless, the set of words that have a run through the automaton is $\mathsf{ESat}(\overleftarrow{\mathsf{G}}\,\varphi)$.

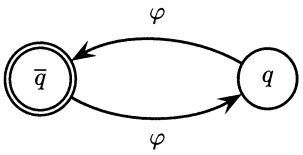


Fig. 5.20. A counting NDet_G automaton that is equivalent to $\mathsf{G}\varphi$

Beneath the above characterizations of the languages whose syntactic monoid is aperiodic, a variety of different other characterizations has been found in the past. It is beyond the scope of this book to give complete proofs for all of these characterizations, but we will prove a further one in the next chapter, namely the relationship with the monadic first order theory of linear order $\mathsf{MFO}_<$. Other characterizations are listed in the following theorem without proof:

Theorem 5.76 (Characterization of Temporal Logic on Finite Words). *For every regular language $L \subseteq \Sigma^*$, the following properties are equivalent:*

(1) there is a pure past formula Φ with $L = \mathsf{ESat}(\Phi)$
(2) there is a temporally defined automaton that accepts L
(3) there is a noncounting automaton that accepts L
(4) there is a star-free regular expression that describes L
(5) there is a $n > 0$ such that for all $\alpha, \beta, \gamma \in \Sigma^*$, $\alpha\beta^n\gamma \in L$ iff $\alpha\beta^{n+1}\gamma \in L$
(6) there is a $\mathsf{MFO}_<$ formula that describes L

We have already proved the equivalences (1)=(2) and (1)=(3). Equivalences (3)=(4)=(6) are due to McNaughton and Papert [364], and a revised proof of the equivalence between (4) and (6) has been published by Perrin and Pin in [403]. Equivalence (1)=(6) is due to Kamp [274] and Gabbay, Pnueli, Shelah and Stavi [203]. The equivalence between (2) and (3) is also proved in Zuck [326, 527], a further proof sketch can also be found in [351, 352]. Further direct translation from noncounting automata to temporal logic can be found in [120, 341, 342, 511]. The equivalence between (4) and (5) is known as Schützenberger's theorem [402, 444, 445].

A completely different approach to prove Schützenberger's result is obtained by the Krohn-Rhodes decomposition theorem which is beyond the scope of this book The interested reader may consult [151, 209, 309, 341, 342, 475]. We can however prove the following corollary of the Krohn-Rhodes theorem:

Theorem 5.77 (Krohn-Rhodes Decomposition). *Let $\mathfrak{A}$ be an acceptor such that* $\mathsf{SM}(\mathsf{Lang}(\mathfrak{A}))$ *is aperiodic. Then there are automata $\mathfrak{A}_1, \ldots, \mathfrak{A}_n$ with $\mathfrak{A}_i = (\Sigma, \{s_1^i, s_2^i\}, \mathcal{I}^i, \mathcal{R}^i)$ such that $\mathfrak{A}$ is equivalent to the cascade product of these automata. Furthermore, the transitions $\mathcal{R}_\sigma^i$ of any letter σ fall into one of the two following categories:*

- *Reset Inputs: There is a state s_j^i in $\mathfrak{A}_i$ such that* $\mathsf{suc}_\exists^{\mathcal{R}_\sigma}(\{s_1^i, s_2^i\}) = \{s_j^i\}$.
- *Identity Inputs:* $\mathsf{suc}_\exists^{\mathcal{R}_\sigma}(\{s_1^i\}) = \{s_1^i\}$ *and* $\mathsf{suc}_\exists^{\mathcal{R}_\sigma}(\{s_2^i\}) = \{s_2^i\}$

Due to our previous results, the proof is remarkably simple: We translate the automaton $\mathfrak{A}$ to a pure past formula $\varphi_{\mathfrak{A}}$ and use our translation procedure TopProp to translate $\varphi_{\mathfrak{A}}$ back to an automaton. We have already seen that this automaton is a cascade product of simple automata having two states. Inspecting the automata used in Theorem 5.38 and Theorem 5.39 shows that all their inputs are either resets or identities.

5.5.2 Completeness of the Borel Classes

In the previous section, we have developed translations that allowed us to translate the temporal logics TL_κ to the corresponding automaton classes Det_κ. In the previous subsection, we have moreover seen that the converse is not possible, since each class Det_κ contains counting automata that can not be translated to LTL_p, and hence neither to TL_κ. However, we can now consider the *subsets* TDet_κ *of* Det_κ *that consists of the noncounting automata.* Equivalently, we can characterize TDet_κ *as the set of temporally defined* Det_κ *automata.*

We now prove the completeness of the temporal logics TL_κ, which means that we have to show that we can translate every TDet_κ automaton to an equivalent TL_κ formula.

Theorem 5.78. *For any class* $\kappa \in \{\mathsf{G}, \mathsf{F}, \mathsf{Prefix}, \mathsf{GF}, \mathsf{FG}, \mathsf{Streett}\}$, *we have* $\mathsf{TDet}_\kappa \approx \mathsf{TL}_\kappa$. *Moreover, we have* $\mathsf{LTL}_p \approx \mathsf{TL}_{\mathsf{Streett}}$.

Proof. The direction $\mathsf{TL}_\kappa \gtrapprox \mathsf{TDet}_\kappa$ has already been proved, so that it remains to prove $\mathsf{TDet}_\kappa \gtrapprox \mathsf{TL}_\kappa$ here. Given a TDet_κ automaton $\mathfrak{A}$, we show how an equivalent formula $\Phi_\mathfrak{A} \in \mathsf{TL}_\kappa$ can be constructed. For example, consider the case $\kappa = \mathsf{FG}$ (all other cases are proved in an analogous manner). Hence, we have $\mathfrak{A} = \mathcal{A}_\exists\,(Q, \Phi_\mathcal{I}, \Phi_\mathcal{R}, \mathsf{FG}\Phi_\mathcal{F})$ where we can assume that $\Phi_\mathcal{I}$ and $\Phi_\mathcal{F}$ are propositional formulas over Q. Hence, these formulas encode subsets of states $Q_\mathcal{I} \subseteq 2^Q$ and $Q_\mathcal{F} \subseteq 2^Q$.

As $\mathfrak{A}$ is temporally defined, it follows that for all $\iota, \vartheta \subseteq Q$ there is a pure past formula $\varphi_{\iota,\vartheta}$ such that $L_{\iota,\vartheta} = \mathsf{ESat}(\varphi_{\iota,\vartheta})$ holds. We define

$$\Phi_\mathfrak{A} := \mathsf{FG} \bigvee_{\iota \in Q_\mathcal{I}, \vartheta \in Q_\mathcal{F}} \varphi_{\iota,\vartheta}$$

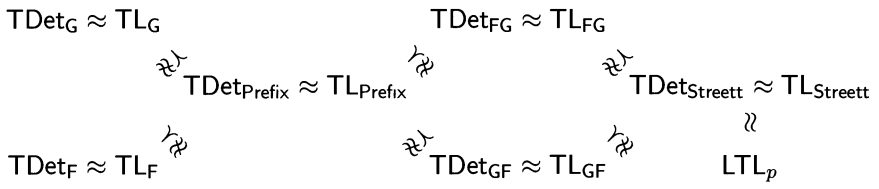
It is obvious that $\Phi_\mathfrak{A}$ and $\mathfrak{A}$ are equivalent, i.e., for any structure $\mathcal{K}$ and any path π through $\mathcal{K}$, we have $(\mathcal{K}, \pi) \models \mathfrak{A} \leftrightarrow \Phi_\mathfrak{A}$.

In the same manner, we construct formulas of the form $\mathsf{G}\psi$, $\mathsf{F}\psi$, $\mathsf{FG}\psi$, $\mathsf{GF}\psi$, $\bigwedge_{i=1}^{f} \mathsf{G}\varphi \vee \mathsf{F}\psi$, and $\bigwedge_{i=1}^{f} \mathsf{GF}\varphi \vee \mathsf{FG}\psi$ to construct TL_κ formulas of given TDet_κ automata for all mentioned classes κ.

The equal expressiveness $\mathsf{LTL}_p \approx \mathsf{TL}_{\mathsf{Streett}}$ follows from the fact that we can translate every LTL_p formula via our translation TopProp on page 347 to an equivalent noncounting $\mathsf{NDet}_{\mathsf{Streett}}$ automaton. The rest follows by the translation to $\mathsf{TL}_{\mathsf{Streett}}$ as indicated above. □

The above theorem allows us furthermore to prove the strictness of the expressivenesses of the TL_κ logics, since we can now transfer these results from the corresponding results of the automaton hierarchy to the logics TL_κ. In particular, we have the following theorem that establishes the analogon of the Borel hierarchy as a hierarchy of temporal logics:

Theorem 5.79 (Equal Expressiveness of TL_κ and TDet_κ). *For any* $\kappa \in \{\mathsf{G}, \mathsf{F}, \mathsf{Prefix}, \mathsf{FG}, \mathsf{GF}, \mathsf{Streett}\}$, *we have* $\mathsf{TL}_\kappa \approx \mathsf{TDet}_\kappa$, *and therefore the following hierarchy of these logics:*



Hence, the six classes TL_κ form the same hierarchy as the automata, and correspond with the Borel hierarchy. Considering the construction in the proof of Theorem 5.78 reveals that we only constructed TL_κ formulas of a very special form: The formulas that we constructed had the form of the acceptance condition of the corresponding automaton, where propositional formulas were replaced with pure past formulas.

It therefore follows that these formulas are also expressively complete w.r.t. TDet_κ, i.e., every formula in TDet_κ can be rewritten in such a restricted

form. We therefore give the following definition which is due to Manna and Pnueli who were the first who investigated the temporal logic hierarchy in [350–352].

Definition 5.80 (Normal Forms for the Classes TL_κ). *Given that φ_i and ψ_i are pure past formulas for $i = 0, \ldots, n$, we define the classes of safety, liveness, obligation, persistence, fairness, and reactivity formulas as follows:*

name of normal form	*abbreviation*	*syntactic representation*
safety	$\mathsf{NF_G}$	$\mathsf{G}\varphi_0$
guarantee	$\mathsf{NF_F}$	$\mathsf{F}\varphi_0$
obligation	$\mathsf{NF_{Prefix}}$	$\bigwedge_{i=0}^{n} \mathsf{G}\varphi_i \vee \mathsf{F}\varphi_i$
persistence	$\mathsf{NF_{FG}}$	$\mathsf{FG}\varphi_0$
recurrence	$\mathsf{NF_{GF}}$	$\mathsf{GF}\varphi_0$
reactivity	$\mathsf{NF_{Streett}}$	$\bigwedge_{i=0}^{n} \mathsf{FG}\varphi_i \vee \mathsf{GF}\varphi_i$

It is easily seen that safety, guarantee, obligation, persistence, fairness, and reactivity formulas are syntactic subsets of $\mathsf{TL_G}$, $\mathsf{TL_F}$, $\mathsf{TL_{Prefix}}$, $\mathsf{TL_{FG}}$, $\mathsf{TL_{GF}}$, and $\mathsf{TL_{Streett}}$, respectively. The converse is obviously not true, since the logics TL_κ are syntactically much richer than the corresponding set NF_κ. By the proof of Theorem 5.78, we moreover have the following result:

Corollary 5.81. *For any class $\kappa \in \{\mathsf{G}, \mathsf{F}, \mathsf{Prefix}, \mathsf{GF}, \mathsf{FG}, \mathsf{Streett}\}$, we have $\mathsf{NF}_\kappa \approx \mathsf{TL}_\kappa$. In particular, we have $\mathsf{NF_{Streett}} \approx \mathsf{LTL}_p$*

Due to the closures of the logics TL_κ, it follows that also all logics NF_κ must be closed under disjunction and conjunction. Moreover, $\mathsf{NF_{Streett}}$ and $\mathsf{NF_{Prefix}}$ must be closed under all Boolean operations. This can also be shown directly:

$$\mathsf{G}\varphi \wedge \mathsf{G}\psi = \mathsf{G}[\varphi \wedge \psi] \qquad \mathsf{FG}\varphi \wedge \mathsf{FG}\psi = \mathsf{FG}[\varphi \wedge \psi]$$
$$\mathsf{G}\varphi \vee \mathsf{G}\psi = \mathsf{G}\left[\overleftarrow{\mathsf{G}}\varphi \vee \overleftarrow{\mathsf{G}}\psi\right] \qquad \mathsf{FG}\varphi \vee \mathsf{FG}\psi = \mathsf{FG}\left[\psi \vee \overleftarrow{\mathsf{X}}[\varphi \overleftarrow{\mathsf{U}} (\varphi \wedge \neg\psi)]\right]$$
$$\mathsf{F}\varphi \wedge \mathsf{F}\psi = \mathsf{F}\left[\overleftarrow{\mathsf{F}}\varphi \wedge \overleftarrow{\mathsf{F}}\psi\right] \qquad \mathsf{GF}\varphi \wedge \mathsf{GF}\psi = \mathsf{GF}\left[\psi \wedge \overleftarrow{\mathsf{X}}[(\neg\psi) \overleftarrow{\mathsf{U}} \varphi]\right]$$
$$\mathsf{F}\varphi \vee \mathsf{F}\psi = \mathsf{F}[\varphi \vee \psi] \qquad \mathsf{GF}\varphi \vee \mathsf{GF}\psi = \mathsf{GF}[\varphi \vee \psi]$$

None of the classes of safety, guarantee, persistence, and fairness formulas are however closed under negation: The closure under negation would immediately imply that the some corresponding ω-automaton classes would coincide and therefore would contradict Theorem 4.33.

Finally, consider now what happens if a NF_κ formula is translated by TopProp to an equivalent ω-automaton: as the formulas mainly consist of pure past temporal logic subformulas, the rules of Theorem 5.38 are sufficient for the translation. These rules, replace the pure past temporal logic subformulas by propositional ones and therefore introduce new state variables that

correspond with the elementary subformulas of the given formula. Hence, given a formula $\Phi \in \mathsf{NF}_\kappa$, we obtain a *deterministic* TDet_κ-automaton. Hence, the formulas NF_κ can be viewed in some way as determinizations of TL_κ.

We conclude this section with related results on the characterization of the star-free ω-regular languages. These results are summarized as follows:

Theorem 5.82 (Characterization of Temporal Logic on Infinite Words). *For every ω-regular language $L \subseteq \Sigma^\omega$, the following properties are equivalent:*

(1) there is a LTL_p formula Φ whose models are the words in L
(2) there is a temporally defined Streett automaton that accepts L
(3) there is a noncounting Streett automaton that accepts L
(4) there are a star-free regular expressions $\alpha_1, \ldots, \alpha_n$ and $\beta_1, \ldots, \beta_n$ such that $L = \mathsf{Lang}(\sum_{i=1}^n \alpha_i \beta_i^\omega)$ and $\mathsf{Lang}(\beta_i \beta_i) \subseteq \mathsf{Lang}(\beta_i)$
(5) there are a star-free regular expressions $\alpha_1, \ldots, \alpha_n$ and $\beta_1, \ldots, \beta_n$ such that $L = \mathsf{Lang}(\sum_{i=1}^n \mathsf{Lim}(\alpha_i) \& \overline{\mathsf{Lim}(\beta_i)})$
(6) there are no words $\alpha, \beta, \gamma, \delta \in \Sigma^*$ such that for infinitely many n we have $\alpha\beta^n\gamma\delta^\omega \in L$ and also for infinitely many n we have $\alpha\beta^n\gamma\delta^\omega \notin L$
(7) there is a $\mathsf{MFO}_<$ formula that describes Φ

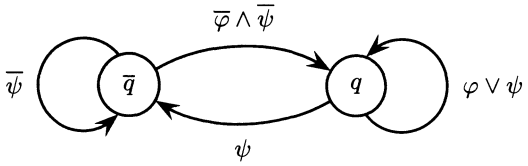


Fig. 5.21. Semiautomaton $\mathfrak{A}_\mathsf{U}$ with transition relation $q \leftrightarrow \psi \vee \varphi \wedge \mathsf{X}q$

We illustrate the constructions mentioned in the previous theorems by an example. To this end, we consider the semiautomaton $\mathfrak{A}_\mathsf{U}$ that is used for the abbreviation of subformulas $[\varphi \,\underline{\mathsf{U}}\, \psi]$ in our translations to ω-automata. Recall that this semiautomaton has two states that are encoded by a single state variable q. The transition relation is $q \leftrightarrow \psi \vee \varphi \wedge \mathsf{X}q$, and the state transition diagram is shown in Figure 5.21.

We first compute the transition monoid $\mathsf{TM}(\mathfrak{A}_\mathsf{U})$ of $\mathfrak{A}_\mathsf{U}$ and then characterize its classes with pure past LTL_p formulas. To this end, note first that we have to consider four inputs that constitute our alphabet Σ_U, namely $\overline{\varphi}\overline{\psi}$, $\overline{\varphi}\psi$, $\varphi\overline{\psi}$, and $\varphi\psi$, and therefore we obtain the following matrices:

$\mathcal{M}_{\overline{\varphi}\overline{\psi}} = \begin{pmatrix} 1 & 1 \\ 0 & 0 \end{pmatrix}$	$\mathcal{M}_{\overline{\varphi}\psi} = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}$	$\mathcal{M}_{\varphi\overline{\psi}} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}$	$\mathcal{M}_{\varphi\psi} = \begin{pmatrix} 0 & 0 \\ 1 & 1 \end{pmatrix}$
$q = 0$	$q = 1$	$q = \mathsf{X}q$	$q = 1$

Computing their closure under matrix multiplications reveals that $\mathsf{TM}(\mathfrak{A}_\mathsf{U})$ consists of three matrices, and furthermore, we have the following operation table:

$\times$	$\mathcal{M}_{\varphi\overline{\psi}}$	$\mathcal{M}_{\overline{\varphi}\overline{\psi}}$	$\mathcal{M}_{\overline{\varphi}\psi}$
$\mathcal{M}_{\varphi\overline{\psi}}$	$\mathcal{M}_{\varphi\overline{\psi}}$	$\mathcal{M}_{\overline{\varphi}\overline{\psi}}$	$\mathcal{M}_{\overline{\varphi}\psi}$
$\mathcal{M}_{\overline{\varphi}\overline{\psi}}$	$\mathcal{M}_{\overline{\varphi}\overline{\psi}}$	$\mathcal{M}_{\overline{\varphi}\overline{\psi}}$	$\mathcal{M}_{\overline{\varphi}\overline{\psi}}$
$\mathcal{M}_{\overline{\varphi}\psi}$	$\mathcal{M}_{\overline{\varphi}\psi}$	$\mathcal{M}_{\overline{\varphi}\psi}$	$\mathcal{M}_{\overline{\varphi}\psi}$

We furthermore know by Theorem 4.54 of page 258 that the elements of $\mathsf{TM}(\mathfrak{A}_\mathsf{U})$ correspond with the classes $[\beta]_{\mathfrak{A}_\mathsf{U}}$ of the induced equivalence relation $\equiv_{\mathfrak{A}_\mathsf{U}}$ of $\mathfrak{A}_\mathsf{U}$. Hence, we have to characterize these three classes by pure past LTL_p formulas. This can be done by the algorithm indicated in the proof of Theorem 5.74, but it is not too hard to see that the following hold:

- $[\varphi\psi]_{\equiv_{\mathfrak{A}_\mathsf{U}}} = \{\alpha \in \Sigma_\mathsf{U}^* \mid \mathcal{M}_\alpha = \mathcal{M}_{\varphi\psi}\} = \mathsf{ESat}(\overleftarrow{\mathsf{F}}(\psi \wedge \overleftarrow{\mathsf{X}}\,\overleftarrow{\mathsf{G}}(\varphi \wedge \neg\psi)))$
- $[\overline{\varphi}\psi]_{\equiv_{\mathfrak{A}_\mathsf{U}}} = \{\alpha \in \Sigma_\mathsf{U}^* \mid \mathcal{M}_\alpha = \mathcal{M}_{\overline{\varphi}\psi}\} = [\varphi\psi]_{\equiv_{\mathfrak{A}_\mathsf{U}}}$
- $[\varphi\overline{\psi}]_{\equiv_{\mathfrak{A}_\mathsf{U}}} = \{\alpha \in \Sigma_\mathsf{U}^* \mid \mathcal{M}_\alpha = \mathcal{M}_{\varphi\overline{\psi}}\} = \mathsf{ESat}(\overleftarrow{\mathsf{G}}(\varphi \wedge \neg\psi))$
- $[\overline{\varphi}\overline{\psi}]_{\equiv_{\mathfrak{A}_\mathsf{U}}} = \{\alpha \in \Sigma_\mathsf{U}^* \mid \mathcal{M}_\alpha = \mathcal{M}_{\overline{\varphi}\overline{\psi}}\} = \mathsf{ESat}(\overleftarrow{\mathsf{F}}(\neg\varphi \wedge \neg\psi \wedge \overleftarrow{\mathsf{X}}\,\overleftarrow{\mathsf{G}}(\varphi \wedge \neg\psi)))$

Using these characterizations, we can now characterize the regular languages $L_{s,s'}$ by pure past formulas (recall that these languages are the unions of those classes $[\alpha]_{\equiv_\mathfrak{A}}$ with $(s, s') \in \mathcal{R}_\alpha$):

- $L_{\overline{q},\overline{q}} = [\varphi\overline{\psi}]_{\equiv_{\mathfrak{A}_\mathsf{U}}} \cup [\overline{\varphi}\overline{\psi}]_{\equiv_{\mathfrak{A}_\mathsf{U}}} = \mathsf{ESat}(\overleftarrow{\mathsf{G}}(\varphi \wedge \neg\psi) \vee \overleftarrow{\mathsf{F}}(\neg\varphi \wedge \neg\psi \wedge \overleftarrow{\mathsf{X}}\,\overleftarrow{\mathsf{G}}(\varphi \wedge \neg\psi)))$
- $L_{\overline{q},q} = [\overline{\varphi}\overline{\psi}]_{\equiv_{\mathfrak{A}_\mathsf{U}}} = \mathsf{ESat}(\overleftarrow{\mathsf{F}}(\neg\varphi \wedge \neg\psi \wedge \overleftarrow{\mathsf{X}}\,\overleftarrow{\mathsf{G}}(\varphi \wedge \neg\psi)))$
- $L_{q,\overline{q}} = [\psi]_{\equiv_{\mathfrak{A}_\mathsf{U}}} = \mathsf{ESat}(\overleftarrow{\mathsf{F}}(\psi \wedge \overleftarrow{\mathsf{X}}\,\overleftarrow{\mathsf{G}}(\varphi \wedge \neg\psi)))$
- $L_{q,q} = [\psi]_{\equiv_{\mathfrak{A}_\mathsf{U}}} \cup [\varphi\overline{\psi}]_{\equiv_{\mathfrak{A}_\mathsf{U}}} = \mathsf{ESat}(\overleftarrow{\mathsf{F}}(\psi \wedge \overleftarrow{\mathsf{X}}\,\overleftarrow{\mathsf{G}}(\varphi \wedge \neg\psi)) \vee \overleftarrow{\mathsf{G}}(\varphi \wedge \neg\psi))$

We obtain $[\varphi\ \mathsf{U}\ \psi] \leftrightarrow \mathcal{A}_\exists(\{q\}, q, q \leftrightarrow \psi \vee \varphi \wedge \mathsf{X}q, \mathsf{G1})$ according to our translations. Hence, $[\varphi\ \mathsf{U}\ \psi]$ is satisfied by all 'words' that start in q, i.e., by $L_{q,\overline{q}} \cup L_{q,q}$ (which is $L_{q,q}$), and therefore we finally conclude:

$$[\varphi\ \mathsf{U}\ \psi] \leftrightarrow \mathsf{G}\left(\overleftarrow{\mathsf{F}}(\psi \wedge \overleftarrow{\mathsf{X}}\,\overleftarrow{\mathsf{G}}(\varphi \wedge \neg\psi)) \vee \overleftarrow{\mathsf{G}}(\varphi \wedge \neg\psi)\right)$$

In [342], Maler and Pnueli describe another translation from temporally defined automata to temporal logic that is however based on the Krohn-Rhodes decomposition theorem.

5.5.3 Completeness of the Future Fragments

In the following, we consider the question whether the past operators are really required in the logics TL_κ, i.e., we will answer the question of whether there is for any TL_κ formula an equivalent one where no past operator occurs. In fact, we will see that the past operators can be eliminated and we will present an algorithm for the elimination of the past operators. The key to this algorithm is the so-called separation property.

In general, a temporal logic has the separation property, iff for any formula Φ of that logic, there is an equivalent formula of the form $\bigwedge_{i=1}^{n} \varphi_i \vee \psi_i$ so that no future operator occurs in φ_i and no past operator occurs in ψ_i. Hence, the future and past operators are separated in the formulas φ_i and ψ_i, respectively. As any temporal logic formula where only past operators occur is initially equivalent to a propositional one, it follows that if a temporal logic has the separation property, then its future time fragment is expressively complete.

The separation property can often be used to reason about the completeness of a logic. For example, in [202] the following fact is proved: Given that a temporal logic contains the formula $\mathsf{F}(a \wedge \overleftarrow{\mathsf{X}}\overleftarrow{\mathsf{G}}\, b)$, then this logic can express the formula $[a \,\underline{\mathsf{U}}\, b]$, because of the following equivalence[6]: $[a \,\underline{\mathsf{U}}\, b] \leftrightarrow_i \mathsf{F}(a \wedge \overleftarrow{\mathsf{X}}\overleftarrow{\mathsf{G}}\, b)$. As F and G can not express U (see [274] and Theorem 5.3 on page 289), it follows that the temporal logic with the monadic temporal operators X, G, F, $\overleftarrow{\mathsf{X}}$, $\underline{\overleftarrow{\mathsf{X}}}$, $\overleftarrow{\mathsf{G}}$, and $\overleftarrow{\mathsf{F}}$ does not have the separation property.

We will now show that the logics TL_κ for any of the mentioned classes κ have the separation property. We will show this by proving a couple of separation theorems that eliminate a particular nesting of a future operator in a past operator, or conversely the nesting of a past operator in a future time temporal operator. We use these theorems to rewrite a given TL_κ formula so that finally a separated formula is obtained. The crucial observation is that for any of these separation theorems, the right hand side belongs to TL_κ iff the left hand side belongs to TL_κ. Hence, the rewrite procedure preserves membership in the logics TL_κ, which implies that TL_κ has the separation property.

We now present a complete list of the required separation theorems. To reduce the number of cases that must be considered, we assume that the given formula is in negation normal form and that only the temporal operators $\overleftarrow{\mathsf{X}}$, $\underline{\overleftarrow{\mathsf{X}}}$, $\underline{\overleftarrow{\mathsf{U}}}$, $\overleftarrow{\mathsf{B}}$, X, $\underline{\mathsf{U}}$, and B are used. We have already seen that this is possible. Then, the following laws are sufficient:

Lemma 5.83 (Separation Laws for X). *The following equations are valid and can be used for separation of the* X *operator:*

- $\mathsf{X}[\alpha \wedge \overleftarrow{\mathsf{X}}\beta] \leftrightarrow \beta \wedge \mathsf{X}\alpha$
- $\mathsf{X}[\alpha \wedge \underline{\overleftarrow{\mathsf{X}}}\beta] \leftrightarrow \beta \wedge \mathsf{X}\alpha$
- $\mathsf{X}[\alpha \wedge [\beta \,\underline{\overleftarrow{\mathsf{U}}}\, \gamma]] \leftrightarrow \mathsf{X}[\alpha \wedge \gamma] \vee [\beta \,\underline{\overleftarrow{\mathsf{U}}}\, \gamma] \wedge \mathsf{X}[\alpha \wedge \beta]$
- $\mathsf{X}[\alpha \wedge [\beta \,\overleftarrow{\mathsf{B}}\, \gamma]] \leftrightarrow \mathsf{X}[\alpha \wedge \beta \wedge \neg\gamma] \vee [\beta \,\overleftarrow{\mathsf{B}}\, \gamma] \wedge \mathsf{X}[\alpha \wedge \neg\gamma]$
- $\mathsf{X}[\alpha \vee \overleftarrow{\mathsf{X}}\beta] \leftrightarrow \beta \vee \mathsf{X}\alpha$
- $\mathsf{X}[\alpha \vee \underline{\overleftarrow{\mathsf{X}}}\beta] \leftrightarrow \beta \vee \mathsf{X}\alpha$
- $\mathsf{X}[\alpha \vee [\beta \,\underline{\overleftarrow{\mathsf{U}}}\, \gamma]] \leftrightarrow \mathsf{X}[\alpha \vee \gamma] \vee [\beta \,\underline{\overleftarrow{\mathsf{U}}}\, \gamma] \wedge \mathsf{X}\beta$
- $\mathsf{X}[\alpha \vee [\beta \,\overleftarrow{\mathsf{B}}\, \gamma]] \leftrightarrow \mathsf{X}[\alpha \vee \neg\gamma] \wedge \Big([\beta \,\overleftarrow{\mathsf{B}}\, \gamma] \vee \mathsf{X}[\alpha \vee \beta]\Big)$

[6] This equation does however only hold at the initial point of time.

Lemma 5.84 (Separation Laws for $\underline{\mathsf{U}}$). *The following equations are valid and can be used for separation of the $\underline{\mathsf{U}}$ operator:*

- $[(\alpha \wedge \beta)\ \underline{\mathsf{U}}\ \gamma] \leftrightarrow [\alpha\ \underline{\mathsf{U}}\ \gamma] \wedge [\beta\ \underline{\mathsf{U}}\ \gamma]$
- $[\alpha\ \underline{\mathsf{U}}\ (\beta \vee \gamma)] \leftrightarrow [\alpha\ \underline{\mathsf{U}}\ \beta] \vee [\alpha\ \underline{\mathsf{U}}\ \gamma]$
- $\left[\alpha\ \underline{\mathsf{U}}\ (\beta \wedge \overleftarrow{\mathsf{X}}\gamma)\right] \leftrightarrow \beta \wedge \overleftarrow{\mathsf{X}}\gamma \vee \alpha \wedge [(\mathsf{X}\alpha)\ \underline{\mathsf{U}}\ (\gamma \wedge \mathsf{X}\beta)]$
- $\left[\alpha\ \underline{\mathsf{U}}\ (\beta \wedge \overleftarrow{\underline{\mathsf{X}}}\gamma)\right] \leftrightarrow \beta \wedge \overleftarrow{\underline{\mathsf{X}}}\gamma \vee \alpha \wedge [(\mathsf{X}\alpha)\ \underline{\mathsf{U}}\ (\gamma \wedge \mathsf{X}\beta)]$
- $\left[\alpha\ \underline{\mathsf{U}}\ (\beta \wedge [\gamma\ \overleftarrow{\underline{\mathsf{U}}}\ \delta])\right] \leftrightarrow \left(\begin{array}{l}[\gamma\ \overleftarrow{\underline{\mathsf{U}}}\ \delta] \wedge [(\alpha \wedge \mathsf{X}\gamma)\ \underline{\mathsf{U}}\ \beta] \vee \\ [\alpha\ \underline{\mathsf{U}}\ (\delta \wedge [(\alpha \wedge \mathsf{X}\gamma)\ \underline{\mathsf{U}}\ \beta])]\end{array}\right)$
- $\left[\alpha\ \underline{\mathsf{U}}\ (\beta \wedge [\gamma\ \overleftarrow{\mathsf{B}}\ \delta])\right] \leftrightarrow \left(\begin{array}{l}[\gamma\ \overleftarrow{\mathsf{B}}\ \delta] \wedge [(\alpha \wedge \neg\mathsf{X}\delta)\ \underline{\mathsf{U}}\ \beta] \vee \\ [\alpha\ \underline{\mathsf{U}}\ (\gamma \wedge \neg\delta \wedge [(\alpha \wedge \neg\mathsf{X}\delta)\ \underline{\mathsf{U}}\ \beta])]\end{array}\right)$
- $\left[(\alpha \vee \overleftarrow{\mathsf{X}}\beta)\ \underline{\mathsf{U}}\ \gamma\right] \leftrightarrow \gamma \vee (\alpha \vee \overleftarrow{\mathsf{X}}\beta) \wedge [(\beta \vee \mathsf{X}\alpha)\ \underline{\mathsf{U}}\ (\mathsf{X}\gamma)]$
- $\left[(\alpha \vee \overleftarrow{\underline{\mathsf{X}}}\beta)\ \underline{\mathsf{U}}\ \gamma\right] \leftrightarrow \gamma \vee (\alpha \vee \overleftarrow{\underline{\mathsf{X}}}\beta) \wedge [(\beta \vee \mathsf{X}\alpha)\ \underline{\mathsf{U}}\ (\mathsf{X}\gamma)]$
- $\left[(\alpha \vee [\beta\ \overleftarrow{\underline{\mathsf{U}}}\ \gamma])\ \underline{\mathsf{U}}\ \delta\right] \leftrightarrow \left(\begin{array}{l}\left([\beta\ \overleftarrow{\underline{\mathsf{U}}}\ \gamma] \vee [\alpha\ \underline{\mathsf{U}}\ (\gamma \vee \delta)]\right) \wedge \\ [(\beta \vee \gamma \vee [\alpha\ \underline{\mathsf{U}}\ (\gamma \vee \delta)])\ \underline{\mathsf{U}}\ \delta]\end{array}\right)$
- $\left[(\alpha \vee [\beta\ \overleftarrow{\mathsf{B}}\ \gamma])\ \underline{\mathsf{U}}\ \delta\right] \leftrightarrow \left(\begin{array}{l}\left([\beta\ \overleftarrow{\mathsf{B}}\ \gamma] \vee [\alpha\ \underline{\mathsf{U}}\ (\beta \vee \delta)]\right) \wedge \\ [(\neg\gamma \vee \alpha \wedge \mathsf{X}\,[\alpha\ \underline{\mathsf{U}}\ (\beta \vee \delta)])\ \underline{\mathsf{U}}\ \delta]\end{array}\right)$

Finally, the following laws are derived by the equality $[\varphi\ \mathsf{B}\ \psi] \leftrightarrow \neg\,[(\neg\varphi)\ \underline{\mathsf{U}}\ \psi]$ for the separation of the B operator:

Lemma 5.85 (Separation Laws for B). *The following equations are valid and can be used for separation of the B operator:*

- $[(\alpha \vee \beta)\ \mathsf{B}\ \gamma] \leftrightarrow [\alpha\ \mathsf{B}\ \gamma] \vee [\beta\ \mathsf{B}\ \gamma]$
- $[\alpha\ \mathsf{B}\ (\beta \vee \gamma)] \leftrightarrow [\alpha\ \mathsf{B}\ \beta] \wedge [\alpha\ \mathsf{B}\ \gamma]$
- $\left[\alpha\ \mathsf{B}\ (\beta \wedge \overleftarrow{\mathsf{X}}\gamma)\right] \leftrightarrow \neg(\beta \wedge \overleftarrow{\mathsf{X}}\gamma) \wedge (\alpha \vee [(\mathsf{X}\alpha)\ \mathsf{B}\ (\gamma \wedge \mathsf{X}\beta)])$
- $\left[\alpha\ \mathsf{B}\ (\beta \wedge \overleftarrow{\underline{\mathsf{X}}}\gamma)\right] \leftrightarrow \neg(\beta \wedge \overleftarrow{\underline{\mathsf{X}}}\gamma) \wedge (\alpha \vee [(\mathsf{X}\alpha)\ \mathsf{B}\ (\gamma \wedge \mathsf{X}\beta)])$
- $\left[\alpha\ \mathsf{B}\ (\beta \wedge [\gamma\ \overleftarrow{\underline{\mathsf{U}}}\ \delta])\right] \leftrightarrow \left(\begin{array}{l}\left[[(\neg\gamma)\ \overleftarrow{\mathsf{B}}\ \delta] \vee [(\alpha \vee \neg\mathsf{X}\gamma)\ \mathsf{B}\ \beta]\right] \wedge \\ [\alpha\ \mathsf{B}\ (\delta \wedge [(\neg\alpha \wedge \mathsf{X}\gamma)\ \underline{\mathsf{U}}\ \beta])]\end{array}\right)$
- $\left[\alpha\ \mathsf{B}\ (\beta \wedge [\gamma\ \overleftarrow{\mathsf{B}}\ \delta])\right] \leftrightarrow \left(\begin{array}{l}\left[[(\neg\gamma)\ \overleftarrow{\underline{\mathsf{U}}}\ \delta] \vee [(\alpha \vee \mathsf{X}\delta)\ \mathsf{B}\ \beta]\right] \wedge \\ [\alpha\ \mathsf{B}\ (\gamma \wedge \neg\delta \wedge [(\neg\alpha \wedge \neg\mathsf{X}\delta)\ \underline{\mathsf{U}}\ \beta])]\end{array}\right)$
- $\left[(\alpha \wedge \overleftarrow{\mathsf{X}}\beta)\ \mathsf{B}\ \gamma\right] \leftrightarrow \neg\gamma \wedge \left(\alpha \wedge \overleftarrow{\mathsf{X}}\beta \vee [(\beta \wedge \mathsf{X}\alpha)\ \mathsf{B}\ (\mathsf{X}\gamma)]\right)$
- $\left[(\alpha \wedge \overleftarrow{\underline{\mathsf{X}}}\beta)\ \mathsf{B}\ \gamma\right] \leftrightarrow \neg\gamma \wedge \left(\alpha \wedge \overleftarrow{\underline{\mathsf{X}}}\beta \vee [(\beta \wedge \mathsf{X}\alpha)\ \mathsf{B}\ (\mathsf{X}\gamma)]\right)$
- $\left[(\alpha \wedge [\beta\ \overleftarrow{\underline{\mathsf{U}}}\ \gamma])\ \mathsf{B}\ \delta\right] \leftrightarrow \left(\begin{array}{l}[\beta\ \overleftarrow{\underline{\mathsf{U}}}\ \gamma] \wedge [\alpha\ \mathsf{B}\ (\neg\beta \vee \delta)] \vee \\ [(\gamma \wedge [(\mathsf{X}[\beta \wedge \neg\delta])\ \underline{\mathsf{U}}\ \alpha])\ \mathsf{B}\ \delta]\end{array}\right)$
- $\left[(\alpha \wedge [\beta\ \overleftarrow{\mathsf{B}}\ \gamma])\ \mathsf{B}\ \delta\right] \leftrightarrow \left(\begin{array}{l}[\beta\ \overleftarrow{\mathsf{B}}\ \gamma] \wedge [\alpha\ \mathsf{B}\ (\gamma \vee \delta)] \vee \\ [(\beta \wedge \neg\gamma \wedge [\alpha\ \mathsf{B}\ (\gamma \vee \delta)])\ \mathsf{B}\ \delta]\end{array}\right)$

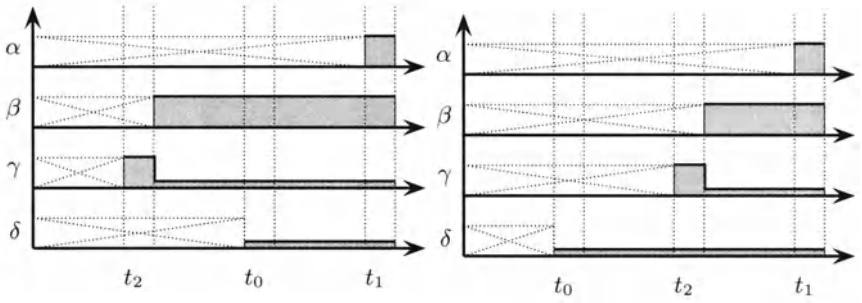


Fig. 5.22. Timing diagrams for the proof of the separation rule for B

The proofs of the above equations are all done in the same manner. For example, consider the separation law for $\Big[(\alpha \wedge [\beta \mathbin{\underline{\overleftarrow{\mathsf{U}}}} \gamma])\ \mathsf{B}\ \delta\Big]$. To prove the equivalence from left to right, we assume that $\Big[(\alpha \wedge [\beta \mathbin{\underline{\overleftarrow{\mathsf{U}}}} \gamma])\ \mathsf{B}\ \delta\Big]$ is evaluated along a path π at time t_0. Hence, we may assume that the event that is awaited, namely $\alpha \wedge [\beta \mathbin{\underline{\overleftarrow{\mathsf{U}}}} \gamma]$, holds at some time t_1 with $t_0 \leq t_1$, which means that there is another point of time t_2 with $t_2 \leq t_1$ where γ held the last time before t_1. Depending on whether $t_2 \leq t_0$ or $t_0 \leq t_2$ holds, we consider then two cases that are depicted in Figure 5.22. The timing diagram given on the left hand side corresponds with the first disjunct, where $[\beta \mathbin{\underline{\overleftarrow{\mathsf{U}}}} \gamma]$ describes the relationship of the signals in the past of t_0 and $[\alpha\ \mathsf{B}\ (\neg\beta \vee \delta)]$ describes the relationship of the signals in the future of t_0 (both formulas overlap at the present point of time). The diagram on the right hand side corresponds with the second disjunct, i.e., with the formula $[(\gamma \wedge [(\mathsf{X}[\beta \wedge \neg\delta]) \mathbin{\underline{\mathsf{U}}} \alpha])\ \mathsf{B}\ \delta]$.

For the other implication, we have to show that both disjuncts of the right hand side imply the left hand side. In both cases, this is easily seen by the timing diagrams in Figure 5.22.

Recall that we can replace any future time temporal operator by $\underline{\mathsf{U}}$ and X. Hence, the above laws allow us to eliminate all past operators that occur in the scope of any future operator. Actually, this is sufficient to prove that every temporal logic formula is initial equivalent to a temporal logic formula where only future operators occur. All we have to do is to eliminate the nestings of past operators in the future operators. The remaining occurrences of the past temporal operators are reduced to propositional logic according to their initial conditions.

To achieve the full separation property, we have to show that the converse is also true, i.e., that any occurrence of a future time temporal operator in the scope of a past operator can also be eliminated. We first consider the 'previous' operators $\overleftarrow{\mathsf{X}}$ and $\underline{\overleftarrow{\mathsf{X}}}$:

Lemma 5.86 (Separation Laws for $\overleftarrow{\mathsf{X}}$). *The following equations are valid and can be used for separation of the $\overleftarrow{\mathsf{X}}$ operator, where we abbreviated* $\mathsf{Init} := \overleftarrow{\mathsf{X}}0$*:*

- $\overleftarrow{\mathsf{X}}[\alpha \wedge \mathsf{X}\beta] \leftrightarrow \mathsf{Init} \vee \beta \wedge \overleftarrow{\mathsf{X}}\alpha$
- $\overleftarrow{\mathsf{X}}[\alpha \wedge [\beta \,\mathsf{U}\, \gamma]] \leftrightarrow \overleftarrow{\mathsf{X}}[\alpha \wedge \gamma] \vee [\beta \,\mathsf{U}\, \gamma] \wedge \overleftarrow{\mathsf{X}}[\alpha \wedge \beta]$
- $\overleftarrow{\mathsf{X}}[\alpha \wedge [\beta \,\mathsf{B}\, \gamma]] \leftrightarrow \overleftarrow{\mathsf{X}}[\alpha \wedge \beta \wedge \neg\gamma] \vee [\beta \,\mathsf{B}\, \gamma] \wedge \overleftarrow{\mathsf{X}}[\alpha \wedge \neg\gamma]$
- $\overleftarrow{\mathsf{X}}[\alpha \vee \mathsf{X}\beta] \leftrightarrow \beta \vee \overleftarrow{\mathsf{X}}\alpha$
- $\overleftarrow{\mathsf{X}}[\alpha \vee [\beta \,\mathsf{U}\, \gamma]] \leftrightarrow \overleftarrow{\mathsf{X}}[\alpha \vee \gamma] \vee [\beta \,\mathsf{U}\, \gamma] \wedge \overleftarrow{\mathsf{X}}\beta$
- $\overleftarrow{\mathsf{X}}[\alpha \vee [\beta \,\mathsf{B}\, \gamma]] \leftrightarrow \overleftarrow{\mathsf{X}}[\alpha \vee \neg\gamma] \wedge \Big([\beta \,\mathsf{B}\, \gamma] \vee \overleftarrow{\mathsf{X}}[\alpha \vee \beta]\Big)$

Similar theorems hold for the separation of the $\overleftarrow{\underline{\mathsf{X}}}$ operator:

- $\overleftarrow{\underline{\mathsf{X}}}[\alpha \wedge \mathsf{X}\beta] \leftrightarrow \neg\mathsf{Init} \wedge \beta \wedge \overleftarrow{\underline{\mathsf{X}}}\alpha$
- $\overleftarrow{\underline{\mathsf{X}}}[\alpha \wedge [\beta \,\mathsf{U}\, \gamma]] \leftrightarrow \overleftarrow{\underline{\mathsf{X}}}[\alpha \wedge \gamma] \vee [\beta \,\mathsf{U}\, \gamma] \wedge \overleftarrow{\underline{\mathsf{X}}}[\alpha \wedge \beta]$
- $\overleftarrow{\underline{\mathsf{X}}}[\alpha \wedge [\beta \,\mathsf{B}\, \gamma]] \leftrightarrow \overleftarrow{\underline{\mathsf{X}}}[\alpha \wedge \beta \wedge \neg\gamma] \vee [\beta \,\mathsf{B}\, \gamma] \wedge \overleftarrow{\underline{\mathsf{X}}}[\alpha \wedge \neg\gamma]$
- $\overleftarrow{\underline{\mathsf{X}}}[\alpha \vee \mathsf{X}\beta] \leftrightarrow \beta \vee \overleftarrow{\underline{\mathsf{X}}}\alpha$
- $\overleftarrow{\underline{\mathsf{X}}}[\alpha \vee [\beta \,\mathsf{U}\, \gamma]] \leftrightarrow \overleftarrow{\underline{\mathsf{X}}}[\alpha \vee \gamma] \vee [\beta \,\mathsf{U}\, \gamma] \wedge \overleftarrow{\underline{\mathsf{X}}}\beta$
- $\overleftarrow{\underline{\mathsf{X}}}[\alpha \vee [\beta \,\mathsf{B}\, \gamma]] \leftrightarrow \overleftarrow{\underline{\mathsf{X}}}[\alpha \vee \neg\gamma] \wedge \Big([\beta \,\mathsf{B}\, \gamma] \vee \overleftarrow{\underline{\mathsf{X}}}[\alpha \vee \beta]\Big)$

It is not surprising that the rules for $\overleftarrow{\mathsf{X}}$ and $\overleftarrow{\underline{\mathsf{X}}}$ are very similar and differ only at the initial point of time. Finally, we have to consider the cases where nestings of future operators occur in the scope of a binary temporal past operator, e.g. in the scope of a $[\ \overleftarrow{\mathsf{U}}\ ]$ operator.

Lemma 5.87 (Separation Laws for $[\ \overleftarrow{\mathsf{U}}\ ]$). *The following equations are valid and can be used for separation of the $[\ \overleftarrow{\mathsf{U}}\ ]$ operator:*

- $[(\alpha \wedge \beta)\ \overleftarrow{\mathsf{U}}\ \gamma] \leftrightarrow [\alpha\ \overleftarrow{\mathsf{U}}\ \gamma] \wedge [\beta\ \overleftarrow{\mathsf{U}}\ \gamma]$
- $[\alpha\ \overleftarrow{\mathsf{U}}\ (\beta \wedge \mathsf{X}\gamma)] \leftrightarrow \beta \wedge \mathsf{X}\gamma \vee [\alpha\ \overleftarrow{\mathsf{U}}\ (\alpha \wedge \gamma \wedge \overleftarrow{\underline{\mathsf{X}}}\beta)]$
- $[\alpha\ \overleftarrow{\mathsf{U}}\ (\beta \wedge [\gamma \,\mathsf{U}\, \delta])] \leftrightarrow \left(\begin{array}{l}[\gamma \,\mathsf{U}\, \delta] \wedge [(\alpha \wedge \overleftarrow{\mathsf{X}}\gamma)\ \overleftarrow{\mathsf{U}}\ \beta] \vee \\ [\alpha\ \overleftarrow{\mathsf{U}}\ (\delta \wedge [(\alpha \wedge \overleftarrow{\mathsf{X}}\gamma)\ \overleftarrow{\mathsf{U}}\ \beta])]\end{array}\right)$
- $[\alpha\ \overleftarrow{\mathsf{U}}\ (\beta \wedge [\gamma \,\mathsf{B}\, \delta])] \leftrightarrow \left(\begin{array}{l}[\gamma \,\mathsf{B}\, \delta] \wedge [(\alpha \wedge \neg\overleftarrow{\mathsf{X}}\delta)\ \overleftarrow{\mathsf{U}}\ \beta] \vee \\ [\alpha\ \overleftarrow{\mathsf{U}}\ (\gamma \wedge \neg\delta \wedge [(\alpha \wedge \neg\overleftarrow{\mathsf{X}}\delta)\ \overleftarrow{\mathsf{U}}\ \beta])]\end{array}\right)$
- $[\alpha\ \overleftarrow{\mathsf{U}}\ (\beta \vee \gamma)] \leftrightarrow [\alpha\ \overleftarrow{\mathsf{U}}\ \beta] \vee [\alpha\ \overleftarrow{\mathsf{U}}\ \gamma]$
- $[(\alpha \vee \mathsf{X}\beta)\ \overleftarrow{\mathsf{U}}\ \gamma] \leftrightarrow \gamma \vee (\alpha \vee \mathsf{X}\beta) \wedge [(\beta \vee \overleftarrow{\mathsf{X}}\alpha)\ \overleftarrow{\mathsf{U}}\ (\overleftarrow{\underline{\mathsf{X}}}\gamma)]$
- $[(\alpha \vee [\beta \,\mathsf{U}\, \gamma])\ \overleftarrow{\mathsf{U}}\ \delta] \leftrightarrow \left(\begin{array}{l}\Big[[\beta \,\mathsf{U}\, \gamma] \vee [(\delta \vee \overleftarrow{\mathsf{X}}\gamma)\ \overleftarrow{\mathsf{B}}\ (\neg\alpha \wedge \neg\delta)]\Big] \wedge \\ \Big[\Big(\beta \vee \gamma \vee [(\delta \vee \overleftarrow{\mathsf{X}}\gamma)\ \overleftarrow{\mathsf{B}}\ (\neg\alpha \wedge \neg\delta)]\Big)\ \overleftarrow{\mathsf{U}}\ \delta\Big]\end{array}\right)$
- $[(\alpha \vee [\beta \,\mathsf{B}\, \gamma])\ \overleftarrow{\mathsf{U}}\ \delta] \leftrightarrow \left(\begin{array}{l}\Big[[\beta \,\mathsf{B}\, \gamma] \vee [(\delta \vee \overleftarrow{\mathsf{X}}\beta)\ \overleftarrow{\mathsf{B}}\ (\neg\alpha \wedge \neg\delta)]\Big] \wedge \\ \Big[\Big(\neg\gamma \vee [(\delta \vee \overleftarrow{\mathsf{X}}\beta)\ \overleftarrow{\mathsf{B}}\ (\neg\alpha \wedge \neg\delta)]\Big)\ \overleftarrow{\mathsf{U}}\ \delta\Big]\end{array}\right)$

It is not surprising that the above equations for the separation of $[\ \underline{\overleftarrow{\mathsf{U}}}\ ]$ operator are similar to those of $[\ \underline{\mathsf{U}}\ ]$ that have been given in Lemma 5.84. Moreover, negating both the right hand and the right hand sides give separation laws for the $[\ \overleftarrow{\mathsf{B}}\ ]$ operator which are analogous to the rules in Lemma 5.85.

Note that there is a plethora of ways how one temporal operator can be expressed by another one. Therefore, there are many ways to formulate the above separation laws. For example, consider the following laws to generate further ones (that also have analogous past versions):

- $[e\ \mathsf{B}\ x] \leftrightarrow \neg\,[(\neg e)\ \underline{\mathsf{U}}\ x]$
- $[e\ \mathsf{B}\ x] \leftrightarrow \neg x \wedge [(\neg \mathsf{X} x)\ \mathsf{U}\ e]$
- $[e\ \mathsf{B}\ x] \leftrightarrow [(\neg x)\ \mathsf{U}\ (e \wedge \neg x)])$
- $[x\ \mathsf{U}\ e] \leftrightarrow \neg\,[(\neg x)\ \underline{\mathsf{B}}\ e]$
- $[x\ \mathsf{U}\ e] \leftrightarrow e \vee [(\mathsf{X} e)\ \mathsf{B}\ (\neg x)]$
- $[x\ \mathsf{U}\ e] \leftrightarrow [e\ \mathsf{B}\ (\neg e \wedge \neg x)])$

The above lemmas now give enough rules to separate the past and future temporal operators in any LTL_p formula. We remark here, that the rules of Lemma 5.84 for the separation of the $[\ \underline{\mathsf{U}}\ ]$ operator remain true, if we replace any occurrence of $[\ \underline{\mathsf{U}}\ ]$ by $[\ \mathsf{U}\ ]$. It is moreover remarkable that for any class $\kappa \in \{\mathsf{G}, \mathsf{F}, \mathsf{Prefix}, \mathsf{GF}, \mathsf{FG}, \mathsf{Streett}\}$, all of the rules preserve membership in the logic TL_κ, which means that we now have the following theorem:

Theorem 5.88 (Expressive Completeness of the Future Fragments of TL_κ). *For every $\kappa \in \{\mathsf{G}, \mathsf{F}, \mathsf{Prefix}, \mathsf{GF}, \mathsf{FG}, \mathsf{Streett}\}$, the logic TL_κ has the separation property, i.e., it can be written in the form $\bigvee_{i=1}^{n} \Phi_i^{<} \wedge \Phi_i^{=} \wedge \Phi_i^{>}$ with formulas $\Phi_i^{<}$, $\Phi_i^{=}$, $\Phi_i^{>}$ of the same class $\Phi \in \mathsf{TL}_\kappa$. Hence, for any formula $\Phi \in \mathsf{TL}_\kappa$, there is a formula $\Psi \in \mathsf{TL}_\kappa$ where no past operator occurs which is initially equivalent to Φ. In particular, LTL_p has the separation property.*

Recall that initially equivalent means that for any structure $\mathcal{K}$ and any path π through that structure, we have $(\mathcal{K}, \pi, 0) \models \Phi$ iff $(\mathcal{K}, \pi, 0) \models \Psi$. This is the same as stating that the formula $\mathsf{G}[\Phi_{\mathsf{init}} \rightarrow (\Phi \leftrightarrow \Psi)]$ is valid, where $\Phi_{\mathsf{init}} := \overleftarrow{\mathsf{X}}\,0$. Clearly, the past time fragment of TL_κ can not be as expressive as the entire logic TL_κ, since any formula that does not have future time operators is initially equivalent to a propositional formula.

The separation property allows us to construct for every LTL_p formula φ a LTL_p formula without past operators that is initially equivalent. For this reason, past operators are often viewed to be unnecessary. However, separating the formula into past, present, and future parts yields an exponential blow-up in the formula size. In [355, 443], it has been shown that this is unavoidable (compare this with the construction on page 451:

Theorem 5.89 (Succinctness of LTL_p with Past Operators). *There are LTL_p formulas (with past operators) of length n, such that all initial equivalent temporal logic formulas without past operators are of size $\Omega(2^n)$.*

Proof. The mentioned LTL_p formulas (with past operators) of length n are the following ones, where $\Phi_{\mathsf{init}} :\equiv \underline{\overleftarrow{\mathsf{X}}}\,0$:

$$\mathsf{G}\left[\left(\bigwedge_{i=1}^{n}\left(x_i \leftrightarrow \overleftarrow{\mathsf{F}}\left(x_i \wedge \Phi_{\mathsf{init}}\right)\right)\right) \rightarrow \left(y \leftrightarrow \overleftarrow{\mathsf{F}}\left(y \wedge \Phi_{\mathsf{init}}\right)\right)\right]$$

The formula memorizes the values of the variables $x_1,\ldots,x_n$, and y at the initial point of time. For every point of time, it is then required that whenever the variables $x_1,\ldots,x_n$ have the value they had at the initial point of time, also y must have its initial value.

As past operators can look back, and can identify the initial point of time, e.g., with the formula Φ_{init} above, it is possible to refer to the initial values at any point of time. Without past operators, however, this is not possible. Instead, we have to make a case distinction depending on the initial values that yields 2^{n+1} different cases. □

For this reason, there are properties that can be expressed exponentially more succinct in with the help of past operators. Adding past does neither increase the expressive power, nor the complexity of the translation and decision procedures. For this reason, the use of past operators in temporal logics is not necessary, but recommended.

5.6 Complexities of the Model Checking Problems

In the previous sections, we have considered several temporal logics that are all subsets of CTL*. We have already discussed the expressiveness of these logics, and we will now consider the complexity classes of the corresponding model checking problems. Clearly, the more expressive a logic is, the more complex is its model checking problem. However, this is not necessarily so: there are logics with different expressivenesses, but with the same complexity. For example, model checking of LTL with or without past operators is PSPACE-complete as well as CTL* model checking. This is remarkable since CTL* is more expressive than LTL, and although LTL with past operators has the same expressiveness as LTL without past operators, it is exponentially more succinct. Moreover, LeftCTL* is not PSPACE-complete, but only Δ_2^p-complete. Finally, LeftCTL* and CTL are equal expressive, but model checking of CTL is only P-complete. Considering the satisfiabilty problem, it turns out that the sat-problem of LeftCTL* is 2EXPTIME-complete, while the sat-problem of CTL is only 2EXPTIME-complete. This explains why an exponential blow-up in the translation from LeftCTL* to CTL can not be avoided. This has been shown initially by Wilke [513]. Adler and Immerman have improved the lower bound of this translation in 2001 to even $\Omega(|\varphi|!)$. Johannsen and Lange [266] have recently proved that the sat-problem of CTL$^+$ is 2EXPTIME-complete and gave therefore another explanation.

In the following, we consider therefore some important complexity classes of model-checking problems. To this end, recall that for any function $f : \mathbb{N} \rightarrow \mathbb{N}$, the sets DTime($f(n)$)/NTime($f(n)$) are the sets of problems of size n that

can be solved with a deterministic/nondeterministic Turing machine in time $O(f(n))$. Analogously, $\mathsf{DSpace}(f(n))/\mathsf{NSpace}(f(n))$ are the sets of problems of size n that can be solved with a deterministic/nondeterministic Turing machine with space $O(f(n))$. Using this notation, we moreover have (see also page 427):

- $\mathsf{P} := \bigcup_{k\in\mathbb{N}} \mathsf{DTime}(n^k)$
- $\mathsf{NP} := \bigcup_{k\in\mathbb{N}} \mathsf{NTime}(n^k)$
- $\mathsf{PSPACE} := \bigcup_{k\in\mathbb{N}} \mathsf{DSpace}(n^k) = \bigcup_{k\in\mathbb{N}} \mathsf{NSpace}(n^k)$
- $\mathsf{EXPTIME} := \bigcup_{k\in\mathbb{N}} \mathsf{DTime}(2^{kn})$
- $\mathsf{2EXPTIME} := \bigcup_{k\in\mathbb{N}} \mathsf{DTime}(2^{2^{kn}})$

Clearly, we have $\mathsf{P} \subseteq \mathsf{NP} \subseteq \mathsf{PSPACE} \subseteq \mathsf{EXPTIME} \subseteq \mathsf{2EXPTIME}$. Besides membership in one of these classes, which gives us an *upper bound* of the complexity, one also considers *lower bounds* that are given by hardness results. If both bounds match, we even have a *completeness* result of the problem. The hardness w.r.t. a class $\mathcal{C}$-complete is shown by reduction of an arbitrary problem in $\mathcal{C}$-complete to the considered problem. The reduction must also be in class $\mathcal{C}$. Hence, due to this reductions, solving one problem in $\mathcal{C}$-complete gives also algorithms for all of the other problems in that class.

We now start with establishing a lower bound for some model checking problems by a reduction of 3SAT to a model checking problem [103]. In particular, we prove that the model checking problem $(\mathcal{K}, s) \models \mathsf{E}\bigwedge_{i=1}^{n} \mathsf{F}a_i$ is NP-hard, i.e., every NP-complete problem can be reduced to a model checking problem of the form $(\mathcal{K}, s) \models \mathsf{E}\bigwedge_{i=1}^{n} \mathsf{F}a_i$ in polynomial time [103, 456]:

Theorem 5.90. *The model checking problem* $(\mathcal{K}, s) \models \mathsf{E}\bigwedge_{i=1}^{n} \mathsf{F}a_i$ *is* NP*-complete (with variables* a_i*). Hence, the model checking problems for* LeftCTL* *and* CTL$^+$ *are both* NP*-hard and* coNP*-hard.*

Proof. The proof is obtained by reducing an arbitrary NP-complete problem in polynomial time to a model checking problem of the form $(\mathcal{K}, s) \models \mathsf{E}\bigwedge_{i=1}^{n} \mathsf{F}a_i$. As the latter formula belongs to CTL$^+$, and also to LeftCTL*, and as these logics are closed under complement, the above result follows.

We use 3SAT [204] for the reduction: consider an arbitrary propositional formula in conjunctive normal form over the variables $\{x_1, \ldots, x_n\}$, where each minterm consists of exactly three literals, i.e., $\Phi \equiv \bigwedge_{i=1}^{m}(a_i \vee b_i \vee c_i)$ (this means each a_i, b_i, c_i is a variable or a negated variable x_k). We define a Kripke structure $\mathcal{K}_\Phi = (\mathcal{I}_\Phi, \mathcal{S}_\Phi, \mathcal{R}_\Phi, \mathcal{L}_\Phi)$ as follows: we use m variables $\vartheta_1, \ldots, \vartheta_m$, one for each minterm of Φ. The set of states $\mathcal{S}_\Phi$ is given as $\mathcal{S}_\Phi := \{p_i \mid 0 \leq i \leq n\} \cup \{q_{i,1} \mid 0 \leq i \leq n\} \cup \{q_{i,2} \mid 0 \leq i \leq n\}$, the only initial state is p_0 and the labeling function is given as $\mathcal{L}_\Phi(p_i) := \{\}$, $\mathcal{L}_\Phi(q_{i,1}) := \{\vartheta_j \mid x_i \in \{a_j, b_j, c_j\}\}$, and $\mathcal{L}_\Phi(q_{i,2}) := \{\vartheta_j \mid \neg x_i \in \{a_j, b_j, c_j\}\}$. This means that each state $q_{i,1}$ satisfies all minterms ϑ_j that have x_i as one of its literals and $q_{i,2}$ satisfies all minterms ϑ_j that have $\neg x_i$ as one of its literals. For this reason, the state $q_{i,1}$

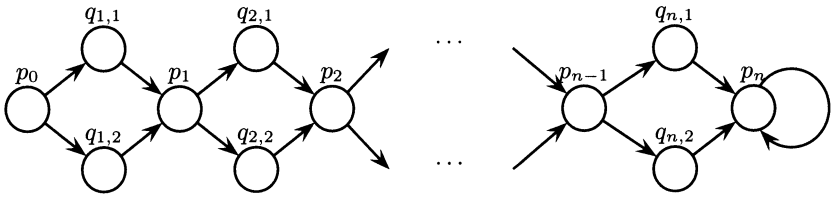


Fig. 5.23. Reducing 3SAT to CTL$^+$ / LeftCTL* model checking

is associated with the variable x_i, while $q_{i,2}$ is associated with the negation of x_i. The transition relation $\mathcal{R}_\Phi$ is given in Figure 5.23.

Clearly, for every assignment of truth values to $\{x_1, \ldots, x_n\}$, we have a unique path through the structure $\mathcal{K}_\Phi$, since each of the 2^n paths from p_0 to p_n corresponds uniquely to such an assignment. Moreover, Φ is satisfiable iff there is a path from p_0 to p_n such that $\bigwedge_{j=1}^{m} \mathsf{F}\vartheta_j$ holds on that path. Consequently, Φ is satisfiable iff $(\mathcal{K}_\Phi, p_0) \models \mathsf{E}\bigwedge_{j=1}^{m} \mathsf{F}\vartheta_j$ holds. Finally, note that $\mathcal{K}_\Phi$ has $3n+1$ states and $4n+1$ transitions, so that the model checking problem $(\mathcal{K}_\Phi, p_0) \models \mathsf{E}\bigwedge_{j=1}^{m} \mathsf{F}\vartheta_j$ is of size $O(n)$.

Membership in NP is easily seen: The nondeterministic algorithm guesses a path through $\mathcal{K}$ and checks if every variable a_i occurs on this path. □

In [443, 456] a nondeterministic model checking algorithm has been given to decide the first of these model checking problems. We will now extend this result to arbitrary LeftCTL* model checking problems of the form $\mathsf{E}\varphi$, where φ must not contain a path quantifier. Since $\mathsf{E}\bigwedge_{i=1}^{m} \mathsf{F}\vartheta_i$ is of this form, NP-hardness is already shown. The NP-completeness follows by a nondeterministic polynomial-time reduction to CTL. Our algorithm therefore guesses a permutation π of the numbers $\{1, \ldots, n\}$ and constructs then the following CTL formula:

$$\mathsf{EF}(\vartheta_{\pi(1)} \wedge \mathsf{EF}(\vartheta_{\pi(2)} \wedge \ldots \mathsf{EF}(\vartheta_{\pi(n-1)} \wedge \mathsf{EF}\vartheta_{\pi(n)}) \ldots))$$

The CTL model checking algorithm solves this remaining problem in polynomial time.

In the same way, we obtain a nondeterministic algorithm for checking LeftCTL* formulas $\mathsf{E}\varphi$ where φ contains no path quantifiers. We assume that the formula is in negation normal form and is based on the temporal operators U, $\underline{\mathsf{U}}$, and X. We reduce it by a nondeterministic algorithm to a CTL formula that implies $\mathsf{E}\varphi$. The code is given in Figure 5.24, where only the case of conjunctions is interesting. Whenever we encounter a disjunction in the formula, we nondeterministically choose one of the disjuncts. After this it follows that conjunctions are only conjunctions of formulas of the form $[\alpha_i \,\underline{\mathsf{U}}\, \beta_i]$, $[\gamma_i \,\mathsf{U}\, \delta_i]$, $\mathsf{X}\varphi_i$ and state formulas η_i. We therefore sort these conjuncts by their top-level operator.

After this, we guess a permutation to combine the $[\alpha_i \,\underline{\mathsf{U}}\, \beta_i]$ and $[\gamma_i \,\mathsf{U}\, \delta_i]$ subformulas according to that permutation, similar to the above reduction of

$\bigwedge_{i=1}^{m} \mathsf{F}\vartheta_i$ (see Lemma 5.18 on page 307). We thereby obtain a formula $[\alpha \underline{\mathsf{U}} \beta]$ or $[\alpha \mathsf{U} \beta]$ of a size linear to the conjunction of the $[\alpha_i \underline{\mathsf{U}} \beta_i]$ and $[\gamma_i \mathsf{U} \delta_i]$ subformulas. We only consider the strong-until case in the following, the weak-until case is obviously done in the same manner. Hence, our entire formula for the conjunct is now $[\alpha \underline{\mathsf{U}} \beta] \wedge \mathsf{X}\xi \wedge \eta$, where we combined also the X-subformulas to a formula $\mathsf{X}\xi$ and the state formulas to a single state formula η. Applying the recursion law $[\alpha \underline{\mathsf{U}} \beta] = \beta \vee \alpha \wedge \mathsf{X}[\alpha \underline{\mathsf{U}} \beta]$ shows that our nondeterministic algorithm can choose one more between $\eta \wedge \beta \wedge \mathsf{X}\xi$ and $\eta \wedge \alpha \wedge \mathsf{X}(\xi \wedge [\alpha \underline{\mathsf{U}} \beta])$.

Our nondeterministic algorithm given in Figure 5.24 is therefore able to compute a linear-sized CTL formula that implies $\mathsf{E}\varphi$. We therefore have the following result:

Theorem 5.91 (Complexity of LeftCTL* Model Checking). *Model checking* LeftCTL* *formulas of the form* $\mathsf{E}\varphi$*, where* φ *is quantifier-free is* NP*-complete. Analogously, checking* LeftCTL* *formulas of the form* $\mathsf{A}\varphi$*, where* φ *is quantifier-free is* coNP*-complete.*

This does not however imply that LeftCTL* model checking is NP-complete. If we could find a NP-algorithm for solving the general LeftCTL* model checking problem, this would mean that the problem is NP-complete. Furthermore, as it is closed under negation, the problem would also be coNP-complete. This would answer the so-far open question of whether the complexity classes NP-complete and coNP-complete are the same. Hence, it turns out that the construction of a NP model checking algorithm for LeftCTL* seems to be a basic problem of todays computer science.

The precise complexity class of LeftCTL$^+$ and LeftCTL* model checking is Δ_2^p-complete. Δ_2^p is the set of problems that can be solved in polynomial time by a deterministic Turing machine that is allowed to delegate subproblems to a nondeterministic Turing machine that must solve these subproblems in polynomial time. Hence, a problem in Δ_2^p contains polynomially many problems of NP. It is easily seen that LeftCTL* model checking can be solved by a Δ_2^p-algorithm by computing the PQNF normal form of page 85 (see also [103]). The hardness of the problem, whch closes the gap between the lower bounds NP and coNP and the upper bound Δ_2^p, has been recently shown in [314] (see also [443]):

Theorem 5.92. *The model checking problems for* LeftCTL* *and* CTL$^+$ *are* Δ_2^p*-complete.*

We will now show that the model checking problems for LTL and CTL*, are more complex, namely PSPACE-complete.

Theorem 5.93. *The model checking problems for* CTL* *and* LTL *are* PSPACE*-complete.*

```
function Conjuncts(Φ)
(* all disjunctions in Φ are under a temporal operator or form a state formula *)
  case Φ of
    φ ∧ ψ     : return Conjuncts(φ) ∪ Conjuncts(ψ);
    else      : return {Φ};
  end case
end function

function CombineU(i, π, I, C_U)
  if i = 0 then return 1
  else
    η := CombineU(i − 1, π, I \ {π(i)}, C_U);
    return [(⋀_{ι∈I} α_ι) U (β_{π(i)} ∧ η)]
  end
end function

function SelectCTL(Φ)
(* Φ ∈ TL_PE is in negation normal form *)
  case Φ of
    is_prop(Φ): return Φ;
    ¬φ        : return Φ;
    φ ∨ ψ     : α := choose {φ, ψ};
                return SelectCTL(α);
    φ ∧ ψ     : φ' := SelectCTL(φ); ψ' := SelectCTL(ψ);
                C := Conjuncts(φ' ∧ ψ');
                C_U := getU(C); C_X := getX(C); C_State := C \ (C_U ∪ C_X);
                π_U := choose Perm({1, ..., |C_U|});
                Φ_U := CombineU(|C_U|, π_U, {1, ..., |C_U|}, C_U);
                Xξ ≡ CombineX(C_X); η := ⋀_{φ∈C_State} φ;
                case Φ_U of
                    [α U β] : Ψ := choose {η ∧ β ∧ Xξ, η ∧ α ∧ X(ξ ∧ [α U β])};
                    [α U β] : Ψ := choose {η ∧ β ∧ Xξ, η ∧ α ∧ X(ξ ∧ [α U β])};
                end;
                return SelectCTL(Ψ);
    Xφ        : return X(SelectCTLφ);
    [φ U ψ]   : ψ' := SelectCTL(ψ);
                return [φ U ψ'];
    [φ U ψ]   : ψ' := SelectCTL(ψ);
                return [φ U ψ'];
  end case
end function
```

Fig. 5.24. Nondeterministic reduction from $\mathsf{TL}_{\mathsf{PE}}$ to CTL

To prove PSPACE-completeness, we need to show that (i) the problem can be solved with a nondeterministic algorithm that runs with polynomial space requirements and (ii) that each PSPACE-complete problem can be reduced in polynomial time to a corresponding model checking problem.

Proof. Proving membership in PSPACE for LTL model checking is easy. Simply guess a path through the structure and check that the given formula holds on that path. This can be easily implemented with a nondeterministic machine.

To prove PSPACE-hardness, we give a polynomial reduction of an arbitrary PSPACE-complete problem to a model checking problem. For this reason, we consider a Turing machine $P = (\Sigma, Q, q_0, F, \Delta)$ where Σ is the alphabet, Q the set of states of P, $q_0 \in Q$ is the initial state and F are the final states. $\Delta \subseteq Q \times \Sigma \times \{-1, 0, 1\} \times Q \times \Sigma$ is the transition relation of P. A tuple $(q_i, a_i, m, q_j, a_j) \in \Delta$ has therefore the following meaning: if P is in state q_i and reads the input a_i, then it will write a_j, and will move its head by m and switches then to state q_j.

Let now P be an arbitrary Turing machine that runs for an input of length n with at most $S(n)$ memory cells such that $S(n)$ is bound by a polynomial in n. It is clear that the runtime $T(n)$ is then bound by an exponential, i.e., $T(n) \in O(2^{cn})$ for some $c \in \mathbb{N}$. We construct the computation table of P as follows:

	1	2	...	n	$n+1$	...	$S(n)$
0	(a_1, q_a)	a_2	...	a_n	β	...	β
$\vdots$	$\vdots$	$\vdots$	...	$\vdots$	$\vdots$	...	$\vdots$
$T(n)$	(β, q_e)	β	...	β	β	...	β

Each row in the above table corresponds to the description of an intermediate configuration of the Turing machine P. In particular, each row directly shows the contents of the tape, and of course, there is exactly one cell that is in $\Sigma \times Q$. This cell indicates the current position of the head of the Turing machine. Hence, the entries of the table are members of the set $\Sigma' := \Sigma \cup (\Sigma \times Q)$. The computation starts at time 0 with the tape $a_1 \ldots a_n \beta \ldots \beta$, where β is the blank symbol and $a_1 \ldots a_n$ the input of length n that we consider. The input is accepted iff there is a computation of the machine that yields in the empty tape (filled with blank symbols β) such that the machine is in a final state $q_e \in F$.

Now, we consider the structure $\mathcal{K}_P = (\mathcal{I}_P, \mathcal{S}_P, \mathcal{R}_P, \mathcal{L}_P)$ as given in Figure 5.25, where $s := S(n)$ and $\delta = |\Sigma \cup (\Sigma \times Q)|$. The structure has the states $\mathcal{S} = \{p_i \mid 0 \leq i \leq s\} \cup \{q_{i,j} \mid 1 \leq i \leq s, 1 \leq j \leq \delta\}$, the initial states are $\mathcal{I}_P := \{p_0\}$ and the transition relation $\mathcal{R}_P$ is as given in Figure 5.25, i.e. $(p_i, q_{i+1,j}) \in \mathcal{R}_P$, $(q_{i,j}, p_i) \in \mathcal{R}_P$ and $(p_s, p_0) \in \mathcal{R}$. We enumerate the elements of $\Sigma \cup (\Sigma \times Q)$ such that $\Sigma \cup (\Sigma \times Q) := \{\sigma_1, \ldots, \sigma_\delta\}$ and $\Sigma = \{\sigma_1, \ldots, \sigma_d\}$. The variables of our temporal logic are $V_\Sigma := \{x_\sigma \mid \sigma \in \Sigma \cup (\Sigma \times Q)\} \cup \{y_i \mid 0 \leq i \leq s\}$, and the labeling function is given as $\mathcal{L}_P(p_i) := \{y_i\}$ and $\mathcal{L}_P(q_{i,j}) := \{x_j\}$.

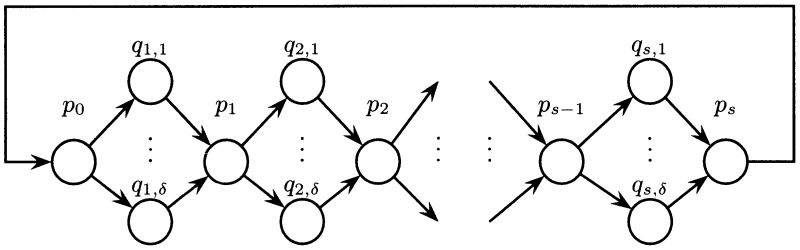


Fig. 5.25. Kripke structure for proving PSPACE-completeness

The aim of the construction of $\mathcal{K}_P$ is to model configurations of P by paths from p_0 to p_s. However, not each path p_0 to p_s corresponds to a configuration, as it is required that there is exactly one state $q_{\imath_0,\jmath_0}$ on the path such that $\sigma_{\jmath_0} \in \Sigma \times Q$ while for all other states $q_{\imath,\jmath}$ on the path $\sigma_\jmath \in \Sigma$ holds. Each path that satisfies the following formula IsConfig corresponds to a configuration of a Turing machine P:

$$\mathsf{IsHead} := \bigvee_{\imath=d+1}^{\delta} x_\imath \qquad \mathsf{OneHead} := \bigvee_{\imath=1}^{s} \mathsf{X}^{2\imath-1}\mathsf{IsHead}$$

$$\mathsf{AtmostOneHead} := \bigvee_{\imath=1}^{s} \mathsf{X}^{2\imath-1}\mathsf{IsHead} \rightarrow \neg \bigvee_{\jmath=1,\jmath\neq\imath}^{s} \mathsf{X}^{2\jmath-1}\mathsf{IsHead}$$

$$\mathsf{IsConfig} := \mathsf{OneHead} \wedge \mathsf{AtmostOneHead}$$

The length of the formula IsHead grows linearly in δ, and the lengths of OneHead and AtmostOneHead grow with $O(s^2)$ and $O(s^3)$, respectively. The initial and final configuration of a computation of P can also be easily specified with a temporal formula using only X operators. The following formula $\mathsf{isInitial}_a$ holds on a path starting in p_0 iff the paths corresponds to the initial configuration of P with the input $a := a_1 \ldots a_n$ and the formula isFinal describes the final configuration of P (where only the s relevant space cells are considered):

$$\mathsf{isInitial}_a := \mathsf{X}x_{(a_1,q_a)} \wedge \left(\bigwedge_{i=2}^{n} \mathsf{X}^{2\imath-1}x_{a_\imath}\right) \wedge \left(\bigwedge_{i=n+1}^{s} \mathsf{X}^{2\imath-1}x_\beta\right)$$

$$\mathsf{isFinal} := \bigvee_{q_e\in F} \mathsf{X}x_{(\beta,q_e)} \wedge \left(\bigwedge_{\imath=2}^{s} \mathsf{X}^{2i-1}x_\beta\right)$$

It is easy to see that the lengths of both formulas are again polynomial. Also, one can easily define a formula IsSucc that holds exactly on those paths π starting in p_0 where the part $\pi^{(1)}, \pi^{(3)}, \ldots, \pi^{(2s-1)}$ is a configuration such that $\pi^{(2s+2)}, \pi^{(2s+4)}, \ldots, \pi^{(4s)}$ is a succeeding configuration according to the transition function Δ of the Turing machine. Using IsSucc, the following formula states that a path π corresponds to a legal computation of P:

$$\mathsf{Run}_P := \mathsf{isInitial}_a \wedge [(y_0 \rightarrow \mathsf{IsSucc}) \,\underline{\mathsf{U}}\, \mathsf{isFinal}]$$

Clearly, the length of Run_P also grows polynomially in s and Run_P holds on a path π through $\mathcal{K}_P$ iff this path starts with an initial configuration for P with input a, and finally yields in the final configuration while all intermediate path sequences starting in p_0 describe succeeding configurations according to P. Hence, we have

$$(\mathcal{K}_P, p_0) \models \mathsf{ERun}_P \text{ iff } P \text{ accepts } a$$

Therefore, the problem to decide whether P accepts a or not can be transformed in polynomial time in $s = S(n)$ and therefore in polynomial time in n to a LTL model checking problem of polynomial size in n. □

Note that Run_P as given in the proof can neither be expressed in CTL, nor with the subset of LTL where only the operators X, G, and F are allowed. In the original proof given in [456], it is also shown how the formulas required for the reduction can be described with only the $[\cdot \,\underline{\mathsf{U}}\, \cdot]$ operator. Moreover, it is not possible to express the above property in LeftCTL*. However, we immediately obtain the following result:

Theorem 5.94 (Complexity of AFCTL*). *The model checking problem of* AFCTL* *is* PSPACE*-complete.*

The upper bound, i.e., the existence of a PSPACE-algorithm for model checking AFCTL* follows from the PSPACE-algorithm for CTL* model checking. The hardness can be proved as shown above, since the formulas used there are also AFCTL* formulas.

5.7 Reductions by Simulation and Bisimulation Relations

In Section 3.8.1, we have already proved that μ-calculus formulas (without past modalities) can not distinguish between bisimilar states of a structure. For this reason, we can reduce the complexity of a μ-calculus model checking problem by computing the quotient structure with the largest bisimulation relation. As CTL* can be translated to the μ-calculus, it is also not possible to distinguish bisimilar states by CTL* formulas, and for this reason, the same reduction of the model checking problem via bisimulation relations is possible.

The converse relation needs however some care: In fact, in case of the μ-calculus, even fixpoint-free formulas are powerful enough to distinguish states that are not bisimilar. However, the same does not hold for temporal logic, when deadend states appear in the Kripke structure. For example, consider the two Kripke structures $\mathcal{K}_1$ and $\mathcal{K}_2$ given in Figure 5.26. The states s_0 and t_0 can be distinguished by the μ-calculus formula $\Diamond\Box 0$, since we have

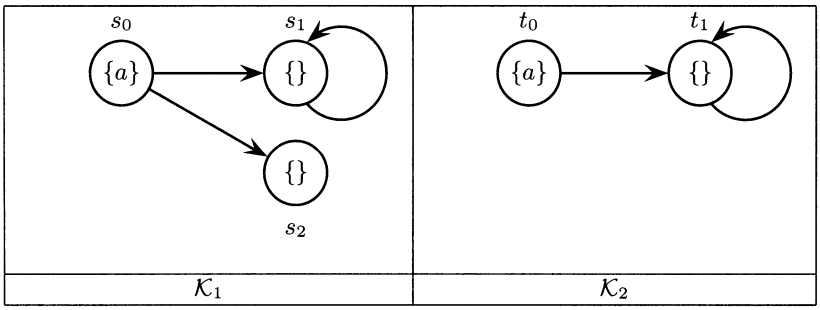


Fig. 5.26. Non-bisimilar structures that can not be distinguished by CTL*

$(\mathcal{K}_1, s_0) \models \Diamond\Box 0$, but $(\mathcal{K}_2, t_0) \not\models \Diamond\Box 0$ (note that $\Box 0$ holds in a state iff this state has no successors). As two states are bisimilar if and only if they satisfy the same μ-calculus formulas, it follows that s_0 and t_0 are not bisimilar.

However, s_0 and t_0 satisfy the same CTL* formulas. *The problem is that temporal logics ignore finite paths*, which is the reason why some authors set up the restriction that every state of a Kripke structure should have a successor state. Only with this restriction, the translation of $\mathsf{EF}\varphi$ is $\mu x.\varphi \vee \Diamond x$, otherwise, we need the slightly more complex version $\mu x.\varphi \wedge (\nu y.\Diamond y) \vee \Diamond x$.

This is due to the definition of the path quantifiers E and A, which consider only infinite paths. Changing this definition causes trouble in the relationship between ω-automata and temporal logic. The deadend states are the only problem, since once we remove them, the desired relationship between bisimulation and satisfiability of the same CTL* formulas becomes valid: The restricted structures that obtained by removing the deadend states, i.e., the states in $[\![\mu x.\Box x]\!]_{\mathcal{K}}$ (cf. pages 81 and 106), are bisimilar iff they satisfy the same CTL* formulas. The same can even be stated for CTL:

Theorem 5.95 (Bisimulation Equivalence for CTL* and CTL). *Given two structures $\mathcal{K}_1$ and $\mathcal{K}_2$ over a the variables V_Σ. Compute $\mathcal{S}^{(1)}_{\mathsf{inf}} = [\![\nu x.\Diamond x]\!]_{\mathcal{K}_1}$ and $\mathcal{S}^{(2)}_{\mathsf{inf}} = [\![\nu x.\Diamond x]\!]_{\mathcal{K}_2}$, and let $\mathcal{K}_1^{\mathsf{inf}}$ and $\mathcal{K}_2^{\mathsf{inf}}$ be the restrictions of $\mathcal{K}_1$ and $\mathcal{K}_2$ to $\mathcal{S}^{(1)}_{\mathsf{inf}}$ and $\mathcal{S}^{(2)}_{\mathsf{inf}}$, respectively. Moreover, assume that $\approx$ is the largest bisimulation relation between $\mathcal{K}_1^{\mathsf{inf}}$ and $\mathcal{K}_2^{\mathsf{inf}}$. Then, the following holds:*

- $s_1 \approx s_2 \Leftrightarrow \forall \varphi \in \mathsf{CTL}.(\mathcal{K}_1, s_1) \models \varphi \Leftrightarrow (\mathcal{K}_2, s_2) \models \varphi$
- $s_1 \approx s_2 \Leftrightarrow \forall \varphi \in \mathsf{CTL}^*.(\mathcal{K}_1, s_1) \models \varphi \Leftrightarrow (\mathcal{K}_2, s_2) \models \varphi$

This result has been originally proved in [72]. The proof of the above theorem is similar to the proof of Theorem 3.44 on page 170, except that $\Diamond$ and $\Box$ are replaced with EX and AX in the construction of the formulas $\Phi_i(s_j)$.

On page 173, we have also established the relationship between simulation relations and the existential or the universal fragment of the μ-calculus.

The above theorem can also be stated for simulation relations and the existential/universal fragments of ACTL* and ACTL:

$$
\begin{array}{lll}
\mathsf{ACTL}^* : & S ::= & S_A \\
& S_A ::= & V_\Sigma \mid \neg S_E \mid S_A \wedge S_A \mid S_A \vee S_A \mid \mathsf{A} P_A \\
& S_E ::= & V_\Sigma \mid \neg S_A \mid S_E \wedge S_E \mid S_E \vee S_E \mid \mathsf{E} P_E \\
& P_E ::= & S_E \mid \neg P_A \mid P_E \wedge P_E \mid P_E \vee P_E \mid \mathsf{X} P_E \mid \mathsf{G} P_E \mid \mathsf{F} P_E \\
& & \mid [P_E \mathsf{W} P_E] \mid [P_E \mathsf{U} P_E] \mid [P_E \mathsf{B} P_A] \\
& & \mid [P_E \underline{\mathsf{W}} P_E] \mid [P_E \underline{\mathsf{U}} P_E] \mid [P_E \underline{\mathsf{B}} P_A] \\
& P_A ::= & S_A \mid \neg P_E \mid P_A \wedge P_A \mid P_A \vee P_A \mid \mathsf{X} P_A \mid \mathsf{G} P_A \mid \mathsf{F} P_A \\
& & \mid [P_A \mathsf{W} P_A] \mid [P_A \mathsf{U} P_A] \mid [P_A \mathsf{B} P_E] \\
& & \mid [P_A \underline{\mathsf{W}} P_A] \mid [P_A \underline{\mathsf{U}} P_A] \mid [P_A \underline{\mathsf{B}} P_E] \\
\mathsf{ACTL} : & S ::= & S_A \\
& S_A ::= & V_\Sigma \mid \neg S_E \mid S_A \wedge S_A \mid S_A \vee S_A \mid \mathsf{A} P_A \\
& S_E ::= & V_\Sigma \mid \neg S_A \mid S_E \wedge S_E \mid S_E \vee S_E \mid \mathsf{E} P_E \\
& P_E ::= & \mathsf{X} S_E \mid \mathsf{G} S_E \mid \mathsf{F} S_E \\
& & \mid [S_E \mathsf{W} S_E] \mid [S_E \mathsf{U} S_E] \mid [S_E \mathsf{B} S_A] \\
& & \mid [S_E \underline{\mathsf{W}} S_E] \mid [S_E \underline{\mathsf{U}} S_E] \mid [S_E \underline{\mathsf{B}} S_A] \\
& P_A ::= & \mathsf{X} S_A \mid \mathsf{G} S_A \mid \mathsf{F} S_A \\
& & \mid [S_A \mathsf{W} S_A] \mid [S_A \mathsf{U} S_A] \mid [S_A \mathsf{B} S_E] \\
& & \mid [S_A \underline{\mathsf{W}} S_A] \mid [S_A \underline{\mathsf{U}} S_A] \mid [S_A \underline{\mathsf{B}} S_E]
\end{array}
$$

Fig. 5.27. Definition of the temporal logics ACTL* and ACTL

Definition 5.96 (ACTL*). *Given a finite set of variables V_Σ, the grammars given in Figure 5.27 define the sublanguages* ACTL* *and* ACTL *of* CTL*.

ACTL* is the set of CTL* formulas where each occurrence of a path quantifier A is positive and each occurrence of a path quantifier E is negative. Analogously, ACTL is the subset of CTL where the occurrences of all A and E path quantifiers are positive and negative, respectively. Note that LTL $\subseteq$ ACTL* holds, but the other direction does of course, not hold.

Theorem 5.97 (Simulation Reduction for ACTL). *Given two structures $\mathcal{K}_1$ and $\mathcal{K}_2$ over a the variables V_Σ. Compute $\mathcal{S}^{(1)}_{\mathsf{inf}} = [\![\nu x.\Diamond x]\!]_{\mathcal{K}_1}$ and $\mathcal{S}^{(2)}_{\mathsf{inf}} = [\![\nu x.\Diamond x]\!]_{\mathcal{K}_2}$, and let $\mathcal{K}_1^{\mathsf{inf}}$ and $\mathcal{K}_2^{\mathsf{inf}}$ be the restrictions of $\mathcal{K}_1$ and $\mathcal{K}_2$ to $\mathcal{S}^{(1)}_{\mathsf{inf}}$ and $\mathcal{S}^{(2)}_{\mathsf{inf}}$, respectively. Moreover, assume that ζ is the largest simulation relation between $\mathcal{K}_1^{\mathsf{inf}}$ and $\mathcal{K}_2^{\mathsf{inf}}$. Then, the following holds:*

- $(s_1, s_2) \in \zeta \Leftrightarrow \forall \varphi \in \mathsf{ACTL}.(\mathcal{K}_2, s_2) \models \varphi \Rightarrow (\mathcal{K}_1, s_1) \models \varphi$
- $(s_1, s_2) \in \zeta \Leftrightarrow \forall \varphi \in \mathsf{ACTL}^*.(\mathcal{K}_2, s_2) \models \varphi \Rightarrow (\mathcal{K}_1, s_1) \models \varphi$

Proof. We have already proved the direction from left to right, since ACTL formulas can be reduced to $\mathcal{L}^{\mathsf{A}}_\mu$ (and negations of ACTL to $\mathcal{L}^{\mathsf{E}}_\mu$). For the other direction, we construct for all states $s_1 \in \mathcal{S}_1$, $s_2 \in \mathcal{S}_2$ with $(s_1, s_2) \notin \zeta$ a ACTL formula that holds in s_2, but not in s_1. To this end, we construct the following formulas $\Phi_i(s_1)$ for a given state $s_1 \in \mathcal{S}_1$:

- $\Phi_0(s_1) = \bigwedge_{x \in \mathcal{L}_1(s_1)} x \wedge \bigwedge_{x \in V_\Sigma \setminus \mathcal{L}_1(s_1)} \neg x$
- $\Phi_{i+1}(s_1) = \Phi_i(s_1) \wedge \bigwedge_{(s_1,s_1') \in \mathcal{R}_1} \mathsf{EX}\Phi_i(s_1')$

It is obvious that $\Phi_i(s_1)$ is the negation of a ACTL formula and that $(\mathcal{K}_1, s_1) \models \Phi_i(s_1)$ holds for all i. Recall now that the following sequence of relations converges to the largest simulation relation (see Lemma 2.15 on page 56):

- $(s_1, s_2) \in \mathcal{H}_0 :\Leftrightarrow \mathcal{L}_1(s_1) = \mathcal{L}_2(s_2)$
- $(s_1, s_2) \in \mathcal{H}_{i+1} :\Leftrightarrow \left(\begin{array}{l} (s_1, s_2) \in \mathcal{H}_i \wedge \\ \forall s_1' \in \mathcal{S}_1.(s_1, s_1') \in \mathcal{R}_1 \rightarrow \\ \qquad \exists s_2' \in \mathcal{S}_2.(s_2, s_2') \in \mathcal{R}_2 \wedge (s_1', s_2') \in \mathcal{H}_i \end{array} \right)$

We have already seen that there is a $n \in \mathbb{N}$ such that $\zeta = \mathcal{H}_n$. We show now by induction on i that $(\mathcal{K}_2, s_2) \models \Phi_i(s_1)$ holds if and only if $(s_1, s_2) \in \mathcal{H}_i$.

i=0: $(\mathcal{K}_2, s_2) \models \Phi_0(s_1) \Leftrightarrow \mathcal{L}_1(s_1) = \mathcal{L}_2(s_2) \Leftrightarrow (s_1, s_2) \in \mathcal{H}_0$

i>0: $(\mathcal{K}_2, s_2) \models \Phi_{i+1}(s_1)$
$\Leftrightarrow (\mathcal{K}_2, s_2) \models \Phi_i(s_1) \wedge \bigwedge_{(s_1,s_1') \in \mathcal{R}_1} \mathsf{EX}\Phi_i(s_1')$
$\Leftrightarrow (\mathcal{K}_2, s_2) \models \Phi_i(s_1) \wedge$
$\qquad \forall s_1' \in \mathcal{S}_1.(s_1, s_1') \in \mathcal{R}_1 \rightarrow (\mathcal{K}_2, s_2) \models \mathsf{EX}\Phi_i(s_1')$
$\Leftrightarrow (s_1, s_2) \in \mathcal{H}_i \wedge$
$\quad \forall s_1' \in \mathcal{S}_1.(s_1, s_1') \in \mathcal{R}_1 \rightarrow \exists s_2' \in \mathcal{S}_2.(s_2, s_2') \in \mathcal{R}_2 \wedge (s_1', s_2') \in \mathcal{H}_i$
$\Leftrightarrow (s_1, s_2) \in \mathcal{H}_{i+1}$

Note that due to the assumption that there are no deadend states, we have $\Diamond\varphi = \mathsf{EX}\varphi$. Moreover, note that for all i, $\neg\Phi_i(s_1)$ is a ACTL formula with $(\mathcal{K}_1, s_1) \not\models \neg\Phi_i(s_1)$. Finally, it follows that whenever there are two states $(s_1, s_2) \notin \zeta$, then there is a $n \in \mathbb{N}$ such that $(s_1, s_2) \notin \mathcal{H}_n$, and therefore by the above result, that $(\mathcal{K}_2, s_2) \models \neg\Phi_n(s_1)$ holds. Choosing n large enough (so that $\mathcal{H}_n = \zeta$ holds) shows then the result. □

Simulation relations are interesting means to reduce the complexity of a model checking problem. According to the above result, the greatest simulation relation is the coarsest abstraction that preserves all universal temporal properties. Constructions of abstract structures by simulation relation are given in Appendix C. Moreover, due its local definition, the greatest simulation relation can be computed in polynomial time, in contrast to trace containment. Extensions to fairness restrictions are considered in [26], and different variants are compared in [242].