A

From LTL to Deterministic Automata–A Safraless Compositional Approach

JAVIER ESPARZA, Fakultät für Informatik, Technische Universität München, Germany
JAN KŘETÍNSKÝ, IST Austria
SALOMON SICKERT, Fakultät für Informatik, Technische Universität München, Germany

We present a new algorithm to construct a (generalized) deterministic Rabin automaton for an LTL formula φ. The automaton is the product of a co-Büchi automaton for φ and an array of Rabin automata, one for each $\mathbf{G}$-subformula of φ. The Rabin automaton for $\mathbf{G}\psi$ is in charge of recognizing whether $\mathbf{FG}\psi$ holds. This information is passed to the co-Büchi automaton that decides on acceptance. As opposed to standard procedures based on Safra's determinization, the states of all our automata have a clear logical structure, which allows for various optimizations. Experimental results show improvement in the sizes of the resulting automata compared to existing methods.

1. INTRODUCTION

Linear temporal logic (LTL) is the most popular language for the specification of properties of single computations of a program. The verification problem for LTL consists of deciding if all computations of a program satisfy a given LTL-formula formalizing a property. In the automata-theoretic approach to this problem [Vardi and Wolper 1986; 1994; Vardi 1999], the negation of the formula is translated into an ω-automaton, and the product of this automaton with the transition system describing the semantics of the program is analyzed. In particular, if this transition system—or some suitable abstraction of it—has a finite number of states, then the product can be exhaustively explored by a search algorithm, and the property can be checked automatically, at least in principle.

While the size of the ω-automaton can be exponential or even double-exponential in the length of the formula (depending on the kind of ω-automaton), typical formulae used in practice are either small, or belong to classes for which this blowup does not happen. However, since the transition system is often very large, generating small ω-automata is still crucial for the efficiency of the approach: Even a reduction of a few states in the ω-automaton can lead to a much larger reduction in the product.

For functional LTL verification (as opposed to the probabilistic verification discussed in the next paragraph), verification algorithms only require to transform the LTL formula into a non-deterministic ω-automaton, typically a Büchi or generalized Büchi automaton and, thanks to intense research in the last decade, the problem of generating small automata is well understood, e.g. [Gerth et al. 1995; Couvreur 1999; Gastin and Oddoux 2001]. Several tools implement a number of heuristic simplifications (of the formula, of intermediate automata generated during the translation, and of the final result), and generate Büchi automata of minimal or nearly minimal size for most common specifications, e.g. [Babiak et al. 2012; Duret-Lutz 2013]. An important factor for this success is the fact that the states of the automaton are LTL formulae, which allows one to use information about logical equivalence or implication between formulae to merge states.

The picture is still very different for quantitative LTL verification of probabilistic systems, i.e., for the problem of computing the probability with which an LTL property is satisfied, or deciding whether it exceeds a given bound. The standard approach to this problem requires to translate the LTL formula into a *deterministic* ω-automaton [Baier and Katoen 2008; Chatterjee et al. 2013], typically a deterministic Rabin automaton (DRA). Contrary to the functional case, up to 2012 there were no algorithms providing a direct translation, all algorithms available proceeded in two steps: first, the formula was translated into a non-deterministic Büchi automaton (NBA), and then Safra's construction [Safra 1988]—or improvements on it [Piterman 2006; Schewe 2009]—was applied to transform the NBA into a DRA. (Alternatively, the determinization step can be replaced by semi-determinization [Courcoubetis and Yannakakis 1988].) At the time of writing this paper this is also the default approach adopted in `PRISM` [Kwiatkowska et al. 2011], a leading probabilistic model checker, which reimplements the optimized Safra's construction of the `ltl2dstar` tool [Klein 2005]. While Safra's construction is a milestone of the theory of ω-automata, it is also difficult to implement (see e.g. [Kupferman 2012]). In particular, it is a monolithic construction that can be applied to any NBA, and therefore does not exploit the structure of LTL formulae.

In 2011 the second author initiated a research program for the design and implementation of a direct LTL-to-DRA translation that "bypasses" Safra's construction. As a first result, a translation for the LTL fragment containing only the temporal operators $\mathbf{F}$ and $\mathbf{G}$ was presented in [Křetínský and Esparza 2012]. The translation yields a deterministic generalized Rabin automaton (DGRA), which can then be degeneralized into a standard DRA. Alternatively, a verification algorithm was proposed in [Chatterjee et al. 2013] which does not require to degeneralize, and exhibits the same worst-case complexity. In both cases much smaller automata were obtained for many formulae. (For instance, while the standard approach translates a conjunction of three fairness constraints into an automaton with over a million states, the algorithm of [Křetínský and Esparza 2012] yields a DGRA with one single state (when acceptance is defined on transitions), and a DRA with 462 states.) Subsequently, the approach was extended to larger fragments of LTL containing the $\mathbf{X}$ operator and restricted appearances of $\mathbf{U}$ [Gaiser et al. 2012; Křetínský and Ledesma-Garza 2013]. However, a general algorithm remained elusive.

In this paper we present a novel approach able to handle full LTL. The approach is *compositional*: the DGRA is obtained as a parallel composition of automata running in lockstep[1]. More specifically, the automaton for a formula φ is the parallel composition of a co-Büchi automaton (a special case of DRA) and an array of DRAs, one for each $\mathbf{G}$-subformula of φ. Intuitively, the state of the co-Büchi automaton after reading a finite word corresponds to "the formula that remains to be fulfilled" (we say that the automaton *monitors* the remaining formula). For example, if $\varphi = (\neg a \wedge \mathbf{X}a) \vee \mathbf{XXG}a$, then the remaining formula after reading $\emptyset\{a\}$ is $\mathtt{tt}$, and after reading $\{a\}$ it is $\mathbf{XG}a$. In particular, if the automaton reaches the state $\mathtt{tt}$, it accepts.

If the co-Büchi automaton never reaches $\mathtt{tt}$, then it needs information from the DRAs to decide on acceptance. The DRA for a $\mathbf{G}$-subformula $\mathbf{G}\psi$ checks whether $\mathbf{G}\psi$ *eventually* holds, i.e., whether $\mathbf{FG}\psi$ holds. Like the co-Büchi automaton, the DRA also monitors the remaining formula, but only partially: more precisely, it does not monitor any $\mathbf{G}$-subformula of ψ, because other DRAs are responsible for them. For instance, if $\psi = a \wedge \mathbf{G}b \wedge \mathbf{G}c$, then the DRA for $\mathbf{G}\psi$ checks $\mathbf{FG}a$, and "delegates" checking $\mathbf{FG}b$ and $\mathbf{FG}c$ to other automata. Further, and crucially, the DRA for $\mathbf{G}\psi$ may also provide the information that not only $\mathbf{FG}\psi$, but a stronger formula $\mathbf{FG}(\psi \wedge \psi')$ holds. For example, the run of the DRA for $\mathbf{G}(a \vee \mathbf{X}c)$ on the word c^ω supplies the information that not only $\mathbf{FG}(a \vee \mathbf{X}c)$, but also the stronger formula $\mathbf{FG}((a \vee \mathbf{X}c) \wedge \mathbf{X}c) \equiv \mathbf{FGX}c$ holds.

The acceptance condition of the full parallel composition is a disjunction over all possible subsets $\mathcal{G}$ of $\mathbf{G}$-subformulae, and all possible sets of stronger formulae $\mathcal{F}$ that the DRAs can check together. Intuitively, the parallel composition accepts a word w by means of the disjunct for $\mathcal{G}$ and $\mathcal{F}$ when w satisfies $\mathbf{F}\mathcal{G}$ (meaning that w satisfies $\mathbf{FG}\psi$

[1]We could also speak of a product of automata, but the operational view behind the term parallel composition helps to convey the intuition.

for every $\mathbf{G}\psi \in \mathcal{G}$) and also $\mathbf{FG}\mathcal{F}$. The co-Büchi automaton is in charge of checking the conditional property that if w satisfies $\mathbf{FG}\mathcal{G}$ and $\mathbf{FG}\mathcal{F}$, then it also satisfies φ.

A previous version of our compositional algorithm appeared in [Esparza and Křetínský 2014]. Since the construction was involved and had a number of corner cases, the third author mechanically verified it in the Isabelle theorem prover. The exercise revealed that, as expected, some minor corrections were necessary, but also exposed a more serious bug requiring a substantial change in a lemma. An analysis revealed that the smallest to us known formula for which the construction of [Esparza and Křetínský 2014] would have produced a wrong result is $\mathbf{G}(\mathbf{X}a \vee \mathbf{GX}b)$, which has a high chance of surviving a large amount of testing.

Related work. There are many constructions translating LTL to NBA, e.g., [Gerth et al. 1995; Couvreur 1999; Daniele et al. 1999; Etessami and Holzmann 2000; Somenzi and Bloem 2000; Gastin and Oddoux 2001; Giannakopoulou and Lerda 2002; Fritz 2003; Babiak et al. 2012; Duret-Lutz 2013]. The one recommended by `ltl2dstar` and used in `PRISM` is `LTL2BA` [Gastin and Oddoux 2001]. The version of Safra's construction described in [Klein and Baier 2007], which includes a number of optimizations, has been implemented in `ltl2dstar` [Klein 2005], and re-implemented in `PRISM` [Kwiatkowska et al. 2011]. A comparison of LTL translators into deterministic ω-automata can be found in [Blahoudek et al. 2013].

Our compositional construction shares the idea of recursive use of automata with the construction that uses temporal testers. However, "testers are inherently non-deterministic" [Pnueli and Zaks 2008], whereas all our automata are deterministic.

Safra's construction can also be used as intermediate step to solve other translation problems, and bypassing it by means of "safraless approaches" has been the subject of several papers [Kupferman and Vardi 2005; Kupferman et al. 2006; Giampaolo et al. 2010].

Outline. The paper is organized as follows: After Section 2, which introduces basic definitions about LTL and ω-automata, the next four sections present LTL-to-DGRA constructions for increasingly general LTL fragments. As a warm-up, Section 3 considers the case of $\mathbf{G}$-free formulae. Section 4 considers the case of formulae $\mathbf{FG}\varphi$, where φ has no occurrence of $\mathbf{G}$. Loosely speaking, it gives the recipe to construct a single element of the array of DRAs, Section 5 constructs a DGRA for an arbitrary formula $\mathbf{FG}\varphi$ as an array of DRAs. Section 6 shows how to construct the co-Büchi automaton and the full parallel composition for an arbitrary formula. All four sections have the same structure. First, we obtain a logical characterization of the words that satisfy a formula of the corresponding fragment, and then derive the corresponding automaton from it.

The paper continues with Section 7, which describes some optimizations that reduce the number of states of the final DGRA, and the size of its acceptance condition. Section 8 contains some remarks about the worst-case complexity of our construction. Finally, Section 9 introduces `Rabinizer`, the tool implementing our construction, and presents a number of experimental results on different test suites of LTL formulae.

As mentioned above, the correctness proof of our construction has been mechanized using the Isabelle theorem prover. Section 10 shows how to access the mechanized proofs, and the relation between this paper and the formal proof. In particular, in the paper we sometimes omit cases in proofs by structural induction that do not provide special insight.

Finally, Section 11 presents our conclusions. Some technical proofs are presented in Appendix.

2. BASIC DEFINITIONS

We recall basic definitions of ω-automata and Linear Temporal Logic, and establish some notations.

In this paper, $\mathbb{N}$ denotes the set of natural numbers including zero. We say that a property holds for *almost every* $n \in \mathbb{N}$ if it holds for all but finitely many natural numbers.

2.1. Alphabets and words

An alphabet is any finite set Σ. The elements of Σ are called letters. A word is an infinite sequence of elements of Σ. The set of all words is denoted by Σ^ω. A finite word is a finite sequence of elements of Σ, and the set of all finite words is denoted by Σ^*.

The ith letter of a word $w \in \Sigma^\omega$ is denoted by $w[i]$, i.e. $w = w[0]w[1]\cdots$. Given $i, j \in \mathbb{N}$, we denote by w_{ij} the finite word $w[i]w[i+1]\cdots w[j-1]$ if $i < j$, and the empty word if $j \leq i$. We denote by w_i or sometimes $w_{i\infty}$ the suffix $w[i]w[i+1]\cdots$.

A (finite or infinite) set of words is called a *language*.

2.2. Linear Temporal Logic

Linear Temporal Logic (LTL) extends propositional logic with temporal operators.

2.2.1. Syntax and semantics

Definition 2.1 (*LTL Syntax*). Let Ap be a finite set of *atomic propositions*. The formulae of linear temporal logic (LTL) over Ap are given by the syntax

$$\varphi ::= \mid \mathbf{tt} \mid \mathbf{ff} \mid a \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{X}\varphi \mid \mathbf{F}\varphi \mid \mathbf{G}\varphi \mid \varphi\mathbf{U}\varphi$$

where $a \in Ap$.

Formulae are interpreted on words over the alphabet 2^{Ap}. That is, a letter is a subset of Ap .

Definition 2.2 (*LTL Semantics*). The satisfaction relation $\models$ between words and formulae is inductively defined as follows:

$$\begin{array}{lll@{\qquad}lll}
w \models \mathbf{tt} & & & w \models \mathbf{X}\varphi & \text{iff} & w_1 \models \varphi \\
w \not\models \mathbf{ff} & & & w \models \mathbf{F}\varphi & \text{iff} & \exists\, k \in \mathbb{N} : w_k \models \varphi \\
w \models a & \text{iff} & a \in w[0] & w \models \mathbf{G}\varphi & \text{iff} & \forall\, k \in \mathbb{N} : w_k \models \varphi \\
w \models \neg\varphi & \text{iff} & w \not\models \varphi & w \models \varphi\mathbf{U}\psi & \text{iff} & \exists\, k \in \mathbb{N} : w_k \models \psi \text{ and} \\
w \models \varphi \wedge \psi & \text{iff} & w \models \varphi \text{ and } w \models \psi & & & \forall\, 0 \leq j < k : w_j \models \varphi \\
w \models \varphi \vee \psi & \text{iff} & w \models \varphi \text{ or } w \models \psi & & &
\end{array}$$

Given two formulae ϕ, ψ, we say that ϕ *entails* ψ, denoted by $\phi \models \psi$, if $w \models \phi$ implies $w \models \psi$ for every $w \in (2^{Ap})^\omega$. We say that ϕ and ψ are *equivalent*, denoted by $\phi \equiv \psi$, if $\phi \models \psi$ and $\psi \models \phi$.

2.2.2. Negation normal-form. In LTL negations can be “pushed inwards”; for instance, we have $\neg\mathbf{FG}a \equiv \mathbf{G}\neg\mathbf{G}a \equiv \mathbf{GF}\neg a$. By pushing negations inwards until all negations appear only in front of atomic propositions, we obtain the negation normal form:

Definition 2.3 (*Negation normal form*). A formula of LTL is in *negation normal form* if it is given by the syntax:

$$\varphi ::= \mathbf{tt} \mid \mathbf{ff} \mid a \mid \neg a \mid \varphi \wedge \varphi \mid \varphi \vee \varphi \mid \mathbf{X}\varphi \mid \mathbf{F}\varphi \mid \mathbf{G}\varphi \mid \varphi\mathbf{U}\varphi$$

where $a \in Ap$.

PROPOSITION 2.4 (NORMAL FORM THEOREM). *Every formula of LTL is equivalent to a formula in negation normal form.*

PROOF. Exhaustive application of the following well-known rewrite rules (which replace a formula by an equivalent one) brings every formula in negation normal form: $\neg\mathbf{X}\varphi \rightsquigarrow \mathbf{X}\neg\phi$, $\neg\mathbf{F}\varphi \rightsquigarrow \mathbf{G}\neg\phi$, $\neg\mathbf{G}\varphi \rightsquigarrow \mathbf{F}\neg\phi$, $\neg(\varphi\mathbf{U}\psi) \rightsquigarrow (\neg\psi\mathbf{U}(\neg\varphi \wedge \neg\psi)) \vee \mathbf{G}\psi$. $\square$

Observe that, due to the last rule, the formula obtained by exhaustive rewriting can be exponentially longer than the original formula. However, if the formula is stored as a dag instead of a tree, then the dag of the formula in negation normal form is only linearly larger than the dag of the original formula.

In the rest of the paper we assume that formulae of LTL are in negation normal form, and speak of "a formula" instead of "a formula in negation normal form".

2.2.3. Propositional entailment, equivalence, and substitution. Loosely speaking, given two formulae φ and ψ, we say that φ propositionally entails ψ if $\varphi \models \psi$ can be proved using only propositional reasoning. So, for instance, $\mathbf{G}a$ propositionally implies $\mathbf{G}a \vee \mathbf{G}b$, but $\mathbf{G}a$ does not propositionally imply $\mathbf{F}a$.

Definition 2.5 (*Propositional implication and equivalence*). A formula of LTL is *proper* if it is not a conjunction or a disjunction (i.e., if the root of its syntax tree is not $\wedge$ or $\vee$). The set of proper formulae of LTL over Ap is denoted by $PF(Ap)$. A *propositional assignment*, or just an *assignment*, is a mapping $\mathcal{A}\colon PF(Ap) \to \{0,1\}$. Given $\varphi \in PF(Ap)$, we write $\mathcal{A} \models \varphi$ iff $\mathcal{A}(\varphi) = 1$, and extend the relation $\models_P$ to arbitrary formulae by:

$$\begin{array}{lll} \mathcal{A} \models_P \varphi \wedge \psi & \text{iff} & \mathcal{A} \models_P \varphi \text{ and } \mathcal{A} \models_P \psi \\ \mathcal{A} \models_P \varphi \vee \psi & \text{iff} & \mathcal{A} \models_P \varphi \text{ or } \mathcal{A} \models_P \psi \end{array}$$

We say that φ *propositionally entails* ψ, denoted by $\varphi \models_P \psi$, if $\mathcal{A} \models_P \varphi$ implies $\mathcal{A} \models_P \psi$ for every assignment $\mathcal{A}$. Finally, φ and ψ are *propositionally equivalent*, denoted by $\varphi \equiv_P \psi$, if $\varphi \models_P \psi$ and $\psi \models_P \varphi$. We denote by $[\varphi]_P$ the equivalence class of φ under the equivalence relation $\equiv_P$. (Observe that $\varphi \equiv_P \psi$ implies $\varphi \equiv \psi$ holds.)

Definition 2.6 (*Propositional substitution*). Let ψ, χ be formulae, and let Ψ be a set of proper LTL-formulae. The formula $\psi[\Psi/\chi]_P$ is inductively defined as follows:

— If $\psi = \psi_1 \wedge \psi_2$ then $\psi[\Psi/\chi]_P = \psi_1[\Psi/\chi]_P \wedge \psi_2[\Psi/\chi]_P$.
— If $\psi = \psi_1 \vee \psi_2$ then $\psi[\Psi/\chi]_P = \psi_1[\Psi/\chi]_P \vee \psi_2[\Psi/\chi]_P$.
— If ψ is a proper formula and $\psi \in \Psi$ then $\psi[\Psi/\chi]_P = \chi$, else $\psi[\Psi/\chi]_P = \psi$.

2.2.4. The After Function $af(\varphi, w)$. Given a formula φ and a finite word w, we define a formula $af(\varphi, w)$, read "φ after w". Intuitively, if a word ww' (where w is a finite word) satisfies φ, then $af(\varphi, w)$ is the formula that holds "after having read w", that is, the formula satisfied by w'. As shown in Proposition 2.10 below, the converse also holds: if w' satisfies $af(\varphi, w)$, then ww' satisfies φ.

Definition 2.7. Let φ be a formula and $\nu \in 2^{Ap}$. We define the formula $af(\varphi, \nu)$ as follows:

$$\begin{array}{lcl@{\qquad}lcl} af(\mathbf{tt}, \nu) & = & \mathbf{tt} & af(\varphi \wedge \psi, \nu) & = & af(\varphi, \nu) \wedge af(\psi, \nu) \\ af(\mathbf{ff}, \nu) & = & \mathbf{ff} & af(\varphi \vee \psi, \nu) & = & af(\varphi, \nu) \vee af(\psi, \nu) \\ af(a, \nu) & = & \begin{cases} \mathbf{tt} & \text{if } a \in \nu \\ \mathbf{ff} & \text{if } a \notin \nu \end{cases} & \begin{array}{l} af(\mathbf{X}\varphi, \nu) \\ af(\mathbf{G}\varphi, \nu) \end{array} & \begin{array}{c} = \\ = \end{array} & \begin{array}{l} \varphi \\ af(\varphi, \nu) \wedge \mathbf{G}\varphi \end{array} \\ af(\neg a, \nu) & = & \begin{cases} \mathbf{ff} & \text{if } a \in \nu \\ \mathbf{tt} & \text{if } a \notin \nu \end{cases} & \begin{array}{l} af(\mathbf{F}\varphi, \nu) \\ af(\varphi\mathbf{U}\psi, \nu) \end{array} & \begin{array}{c} = \\ = \end{array} & \begin{array}{l} af(\varphi, \nu) \vee \mathbf{F}\varphi \\ af(\psi, \nu) \vee (af(\varphi, \nu) \wedge \varphi\mathbf{U}\psi) \end{array} \end{array}$$

We extend the definition to finite words: $af(\varphi, \epsilon) = \varphi$; and $af(\varphi, \nu w) = af(af(\varphi, \nu), w)$ for every $\nu \in 2^{Ap}$ and every finite word w. Finally, we say that ψ is *reachable* from φ if $\psi = af(\varphi, w)$ for some finite word w.

Example 2.8. Let $Ap = \{a, b, c\}$ and $\varphi = a \vee (b \ \mathbf{U} \ c)$. We have $af(\varphi, \{a\}) = \mathbf{tt}$ $af(\varphi, \{b\}) = (b \ \mathbf{U} \ c)$, $af(\varphi, \{c\}) = \mathbf{tt}$, and $af(\varphi, \emptyset) = \mathbf{ff}$.

We collect a number of simple properties of af, proved in the Appendix.

LEMMA 2.9. *For every formula φ and every finite word $w \in (2^{Ap})^*$:*

(1) $af(\varphi, w)$ is a boolean combination of proper subformulae of φ.
(2) If $af(\varphi, w) = \mathbf{tt}$, then $af(\varphi, ww') = \mathbf{tt}$ for every $w' \in (2^{Ap})^*$, and analogously for $\mathbf{ff}$.
(3) If $\varphi_1 \equiv_P \varphi_2$, then $af(\varphi_1, w) \equiv_P af(\varphi_2, w)$.
(4) If φ has n proper subformulae, then the set of formulae reachable from φ has at most 2^{2^n} equivalence classes of formulae with respect to propositional equivalence.

Observe that, by Lemma 2.9(3), the function af can be lifted to equivalence classes of formulae w.r.t. propositional equivalence. Abusing language, we also denote this lifted function by af.

We now state the fundamental property of the After function, also proved in the Appendix: a word ww' satisfies a formula φ iff "after reading" w the "rest" of the word, i.e., the word w', satisfies $af(\varphi, w)$.

PROPOSITION 2.10. *Let φ be a formula, and let $ww' \in (2^{Ap})^\omega$ be an arbitrary word. Then $ww' \models \varphi$ iff $w' \models af(\varphi, w)$.*

2.3. Transition systems and ω-automata

A *deterministic transition system* (DTS) over an alphabet Σ is a tuple $\mathcal{T} = (Q, \Sigma, \delta, q_0)$ where Q is a set of states, Σ is an alphabet, $\delta \colon Q \times \Sigma \to Q$ is a *transition function*, and $q_0 \in Q$ is the *initial state*. If $\delta(q, a) = q'$ then we call the triple $t = (q, a, q')$ a *transition*, and say that q, a, and q' are the *source*, the *letter*, and the *target* of t. We denote by T the set of transitions of $\mathcal{T}$.

A *run* of $\mathcal{T}$ is an infinite sequence $\rho = t_0 t_1 \cdots$ of transitions such that the source of t_0 is the initial state q_0, and for every $i \geq 0$ the target of t_i is equal to the source of t_{i+1}. A transition t *occurs* in ρ if $t = t_i$ for some $i \geq 0$. A state q *occurs* in ρ if it is the source or target of some t_i. Given a word $w = a_0 a_1 \cdots \in \Sigma^\omega$, we denote by $\rho(w)$ the unique run $t_0 t_1 t_2 \cdots$ of $\mathcal{T}$ such that for every $i \geq 0$ the letter of t_i is a_i.

The *product* of two DTSs $\mathcal{T}_1 = (Q_1, \Sigma, \delta_1, q_{01})$ and $\mathcal{T}_2 = (Q_2, \Sigma, \delta_2, q_{02})$ is the DTS $\mathcal{T}_1 \times \mathcal{T}_2 = (Q, \Sigma, \delta, q_0)$, where $Q = Q_1 \times Q_2$, $\delta((q_1, q_2), a) = (\delta_1(q_1, a)\delta(q_2, a)$ for every $q_1 \in Q_1, q_2 \in Q_2, a \in \Sigma$, and $q_0 = (q_{01}, q_{02})$.

2.3.1. Acceptance conditions and ω-automata. A *state-based acceptance condition* for $\mathcal{T}$ is a positive boolean formula over the formal variables $V_Q = \{Inf(S), Fin(S) \mid S \subseteq Q\}$. Acceptance conditions are interpreted over runs. Given a run ρ of $\mathcal{T}$ and an acceptance condition α, we consider the truth assignment that sets the variable $Inf(S)$ to true iff ρ visits (some state of) S infinitely often, and sets $Fin(S)$ to true iff ρ visits (all states of) S finitely often. The run ρ satisfies α if this truth-assignment makes α true. The size of a condition α is its length as boolean formula.

A *transition-based acceptance condition* for $\mathcal{T}$ is defined exactly as a state-based acceptance condition, but replacing the set V_Q by the set $V_T = \{Inf(U), Fin(U) \mid U \subseteq T\}$. In this paper we use state-based or transition-based acceptance conditions, depending on what is more convenient. It is well-known that a state-based conditions can be transformed into an equivalent transition-based one (i.e., a condition satisfied by the same runs). It suffices to replace each occurrence of $Inf(S)$ by $Inf({}^\bullet S)$, where ${}^\bullet S$

denotes the set of transitions with target in S, and similarly for $Fin(S)$. Conversely, a transition-based condition can also be transformed into an equivalent state-based one by replicating the states. Given a DTS $\mathcal{T} = (Q, \Sigma, \delta, q_0)$ with a set T of transitions we construct the new DTS $\mathcal{T}'$ with states $\{q_0\} \cup T$, a transition (q_0, a, t) for every transition $t = (q_0, a, q)$ of T, and a transition (t, a, t') for every pair $t = (q_1, a, q_2)$ and $t' = (q_2, b, q_3)$ of transitions of T. Then, the condition over the transitions of $\mathcal{T}$ becomes an equivalent condition over the states of $\mathcal{T}'$.

A *deterministic ω-automaton* over Σ is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, \alpha)$, where (Q, Σ, δ, q_0) is a deterministic transition system and α is an acceptance condition. $\mathcal{A}$ *accepts* a word $w \in \Sigma^*$ if the run $\rho(w)$ satisfies α. The language of $\mathcal{A}$, denoted by $\mathsf{L}(\mathcal{A})$, is the set of words accepted by $\mathcal{A}$.

An acceptance condition α is a

— ***Büchi condition*** if $\alpha = Inf(S)$ for some $S \subseteq Q$.
— ***co-Büchi condition*** if $\alpha = Fin(S)$ for some $S \subseteq Q$.
— ***Rabin condition*** if $\alpha = \bigvee_{j=1}^{n} (Fin(F_j) \wedge Inf(I_j))$ for sets $F_1, I_1, \ldots, F_n, I_n \subseteq Q$. The pair $P_j = (F_j, I_j)$ is called a ***Rabin pair***.
— ***generalized Rabin condition*** if $\alpha = \bigvee_{j=1}^{n} (Fin(F_j) \wedge \bigwedge_{k=1}^{m_j} Inf(I_{jk}))$ for sets $F_1, \ldots, F_n, I_{11}, \ldots, I_{nm_n} \subseteq Q$.

Observe that Büchi and co-Büchi conditions are special cases of Rabin conditions. Further, every generalized Rabin condition can be transformed into an equivalent Rabin condition, which however may be exponentially longer. The generalized Rabin condition arises naturally when considering intersection of Rabin automata. Observe that we do not need to consider $\bigwedge_{k=1}^{\ell_j} Fin(F_{jk})$, but only $Fin(F_j)$, because $\bigwedge_{k=1}^{n_j} Fin(F_{jk})$ is equivalent to $Fin(\bigcup_{k=1}^{\ell_j} F_{jk})$.

A deterministic Büchi, co-Büchi, Rabin or generalized Rabin automaton is a deterministic ω-automaton with an acceptance condition of the corresponding kind. In the rest of the paper we shorten deterministic Rabin automaton to DRA, and the generalized version to DGRA.

The following results are well known.

PROPOSITION 2.11. *Given DRAs $\mathcal{R}_1$ and $\mathcal{R}_2$ recognizing languages L_1 and L_2, respectively, we can construct DRAs, denoted $\mathcal{R}_1 \cup \mathcal{R}_2$ and $\mathcal{R}_1 \cap R_2$, recognizing $L_1 \cup L_2$ and $L_1 \cap L_2$, respectively. Moreover, the transition system of both $\mathcal{R}_1 \cup \mathcal{R}_2$ and $\mathcal{R}_1 \cap R_2$ is the product of the transition systems of R_1 and R_2.*

PROPOSITION 2.12. *Let X be a finite set of indices, and let $\mathcal{R}_i = (Q, \Sigma, \delta, q_0, \alpha_i)$ be a family of DRAs, one for every index i belonging to some finite set I of indices, all of them with the same underlying transition system. Then $\mathcal{R}_\cup = (Q, \Sigma, \delta, q_0, \bigvee_{i \in I} \alpha_i)$ is a DRA recognizing $\bigcup_{i \in X} \mathsf{L}(\mathcal{R}_i)$, and $\mathcal{R}_\cap = (Q, \Sigma, \delta, q_0, \bigwedge_{i \in X} \alpha_i)$ is a generalized DRA recognizing $\bigcap_{i \in X} \mathsf{L}(\mathcal{R}_i)$.*

3. AUTOMATA FOR G-FREE FORMULAE

We present a translation of $\mathbf{G}$-free formulae (i.e., formulae without any occurrence of the $\mathbf{G}$-operator) into a deterministic ω-automaton with a very simple acceptance condition, which can be expressed both as a Büchi and a co-Büchi condition. The translation is by no means novel, but it serves as a warm-up for the next sections, which consider more general classes of formulae. Moreover, the section allows us to introduce the general scheme we use to design translations: first, we give a logical characterization theorem characterizing the words that satisfy a formula of the given class, and then we construct an automaton which accepts iff the condition of the characterization holds.

THEOREM 3.1 (LOGICAL CHARACTERIZATION THEOREM I). *Let φ be a* $\mathbf{G}$*-free formula and let w be a word. Then $w \models \varphi$ iff there exists $i > 0$ such that $af(\varphi, w_{0j}) \equiv_P$* $\mathbf{tt}$ *for every $j \geq i$.*

PROOF. By Lemma 2.9(2) it suffices to show that $w \models \varphi$ iff there exists $i > 0$ such that $af(\varphi, w_{0i}) \equiv_P \mathbf{tt}$. (In the rest of this proof we use Lemma 2.9(2) without explicitly mentioning it.)
($\Leftarrow$): Assume there exists $i > 0$ such that $af(\varphi, w_{0i}) \equiv_P \mathbf{tt}$. Then $w_i \models af(\varphi, w_{0i})$. By Proposition 2.10, we get $w = w_{0i}w_i \models \varphi$.
($\Rightarrow$): Assume $w \models \varphi$. We proceed by structural induction on φ. We only consider two representative cases.

— $\varphi = a$. Since $w \models \varphi$ we have $w = \nu w'$ for some word w' and for some $\nu \in Ap$ such that $a \in \nu$. By the definition of af we have $af(a, \nu) \equiv_P \mathbf{tt}$, and, since $\nu = w_{01}$, we get $af(\varphi, w_{01}) \equiv_P \mathbf{tt}$.
— $\varphi = \varphi_1 \mathbf{U} \varphi_2$. By the semantics of LTL there is $k \in \mathbb{N}$ such that $w_k \models \varphi_2$ and $w_\ell \models \varphi_1$ for every $0 \leq \ell < k$. By induction hypothesis there exists for every $0 \leq \ell < k$ an $i \geq \ell$ such that $af(\varphi_1, w_{\ell i}) \equiv_P \mathbf{tt}$ and there exists an $i \geq k$ such that $af(\varphi_2, w_{ki}) \equiv_P \mathbf{tt}$. Let j be the maximum of all those i's. We prove $af(\varphi_1 \mathbf{U} \varphi_2, w_{0j}) \equiv_P \mathbf{tt}$ via induction on k.
  — $k = 0$.

$$\begin{array}{lll} & af(\varphi_1 \mathbf{U} \varphi_2, w_{0j}) & \\ = & af(\varphi_2, w_{0j}) \vee (af(\varphi_1, w_{0j}) \wedge af(\varphi_1 \mathbf{U} \varphi_2, w_{1j})) & \text{(def. of } af\text{)} \\ \equiv_P & \mathbf{tt} \vee af(\varphi_1, w_{0j}) \wedge af(\varphi_1 \mathbf{U} \varphi_2, w_{1j})) & (af(\varphi_2, w_{kj}) \equiv_P \mathbf{tt}) \\ \equiv_P & \mathbf{tt} & \end{array}$$

  — $k > 0$.

$$\begin{array}{lll} & af(\varphi_1 \mathbf{U} \varphi_2, w_{0j}) & \\ = & af(\varphi_2, w_{0j}) \vee (af(\varphi_1, w_{0j}) \wedge af(\varphi_1 \mathbf{U} \varphi_2, w_{1j})) & \text{(def. of } af\text{)} \\ \equiv_P & af(\varphi_2, w_{0j}) \vee (\mathbf{tt} \wedge af(\varphi_1 \mathbf{U} \varphi_2, w_{1j})) & (af(\varphi_1, w_{0j}) \equiv_P \mathbf{tt}) \\ \equiv_P & af(\varphi_2, w_{0j}) \vee (\mathbf{tt} \wedge \mathbf{tt}) & \text{(ind. hyp.)} \\ \equiv_P & \mathbf{tt} & \end{array}$$

□

We derive from Theorem 3.1 a deterministic ω-automaton for a given $\mathbf{G}$-free formula φ. The states of the automaton are equivalence classes of formulae under propositional equivalence. The fundamental design idea is: after reading a finite word w, the current state of the automaton must be $af(\varphi, w_{0j})$. So we take the equivalence class of $af(\varphi, \epsilon) = \varphi$ as initial state, and the function af itself as transition function. By Theorem 3.1, a word satisfies φ iff its run in this automaton visits the state $[\mathbf{tt}]_P$. Since we have $af(\mathbf{tt}, \nu) = \mathbf{tt}$ for every $\nu \in 2^{Ap}$, the run visits $[\mathbf{tt}]_P$ iff it visits $[\mathbf{tt}]_P$ infinitely often, or if it visits all other states only finitely often. So we can take $F = \{[\mathbf{tt}]_P\}$ as Büchi condition.

Definition 3.2. Let φ be a $\mathbf{G}$-free formula. Let $Reach(\varphi)$ denote the set of equivalence classes of the formulae reachable from φ w.r.t. propositional equivalence. The *transition system of* φ is the deterministic transition system $\mathcal{T}(\varphi) = (Q, 2^{Ap}, q_0, \delta)$ where

- Q is the quotient of $Reach(\varphi)$ under propositional equivalence.
  (In other words, $[\psi]_P$ is a state of $\mathcal{T}(\varphi)$ iff $af(\varphi, w) = \psi$ for some finite word w.)
- $q_0 = [\varphi]_P$, the equivalence class of φ.
- $\delta([\psi]_P, \nu) = [af(\psi, \nu)]_P$ for every $[\psi]_P \in Q$ and every $\nu \in 2^{Ap}$.
  (I.e., there is a transition $[\varphi]_P \xrightarrow{\nu} [\psi]_P$ iff $af(\varphi, \nu) = \psi$.)

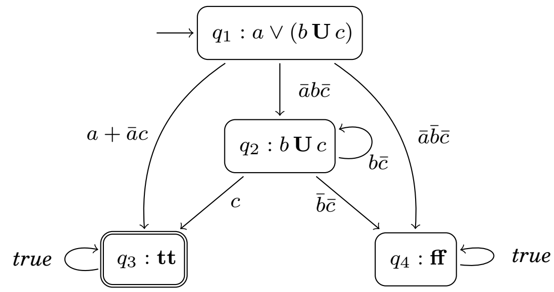


Fig. 1: Büchi (or co-Büchi) automaton for $a \vee (b\ \mathbf{U}\ c)$.

The Büchi automaton for φ is the tuple $\mathcal{B}(\varphi) = (Q, 2^{Ap}, q_0, \delta, F)$, where $F = \{[\mathbf{tt}]_P\}$. Observe that it can be also seen as a co-Büchi automaton with $F = Q \setminus \{[\mathbf{tt}]_P\}$.

Example 3.3. Figure 1 shows the automaton for the formula $\varphi = a \vee (b\ \mathbf{U}\ c)$. We assume $Ap = \{a, b, c\}$. The alphabet 2^{Ap} contains 8 elements, and so every state has 8 outgoing transitions. To avoid cluttering the figure, we use a boolean-function-like notation for transitions. For example, $q_2 \xrightarrow{c} q_3$ denotes that there is a transition from q_2 to q_3 for every subset of 2^{Ap} containing c. So, actually, $q_2 \xrightarrow{c} q_3$ stands for four different transitions. Similarly, $q_1 \xrightarrow{a+\bar{a}c} q_3$ means that there is a transition from q_1 to q_3 for each subset of 2^{Ap} that either contains a, or does not contain a and contains c.

THEOREM 3.4. *Let φ be a $\mathbf{G}$-free formula. Then $\mathsf{L}(\mathcal{B}(\varphi)) = \mathsf{L}(\varphi)$*

PROOF. Immediate consequence of Theorem 3.1 and the definition of $\mathcal{B}(\varphi)$. $\square$

4. DRAS FOR SIMPLE FG-FORMULAE

We introduce the main building block of our paper: a procedure to construct a DRA for formulae $\mathbf{FG}\varphi$ where φ is $\mathbf{G}$-free, i.e., contains no occurrence of $\mathbf{G}$. (Notice that even the formula $\mathbf{FG}a$ has no equivalent deterministic Büchi automaton.)

As in the previous section, we first characterize the words w satisfying a formula $\mathbf{FG}\varphi$ where φ is $\mathbf{G}$-free, and then show how to construct a DRA that accepts iff the condition of the characterization holds. However, in this section we divide this step into two parts. We first introduce an auxiliary automata model, called Mojmir automata[2], and show how to construct a Mojmir automaton recognizing $\mathsf{L}(\mathbf{FG}\varphi)$. (Mojmir automata are designed to make this construction intuitive and easy to grasp.) Then we show how to transform Mojmir automata into equivalent DRAs.

4.1. Logical characterization

The logical characterization of the words satisfying $\mathbf{FG}\varphi$ is an easy consequence of Theorem 3.1.

THEOREM 4.1 (LOGICAL CHARACTERIZATION THEOREM II). *Let $\mathbf{FG}\varphi$ be a formula such that φ is $\mathbf{G}$-free. Then $w \models \mathbf{FG}\varphi$ iff for almost every $i \in \mathbb{N}$ there exists $j \geq i$ such that $af(\varphi, w_{ij}) \equiv_P \mathbf{tt}$.*

PROOF. By the semantics of LTL, $w \models \mathbf{FG}\varphi$ iff $w_i \models \varphi$ for almost every $i \in \mathbb{N}$. By Theorem 3.1, $w \models \mathbf{FG}\varphi$ iff for almost every $i \in \mathbb{N}$ there exists $j \geq i$ such that $af(\varphi, w_{ij}) \equiv_P \mathbf{tt}$. $\square$

[2]Named in honour of Mojmír Křetínský, father of one of the authors

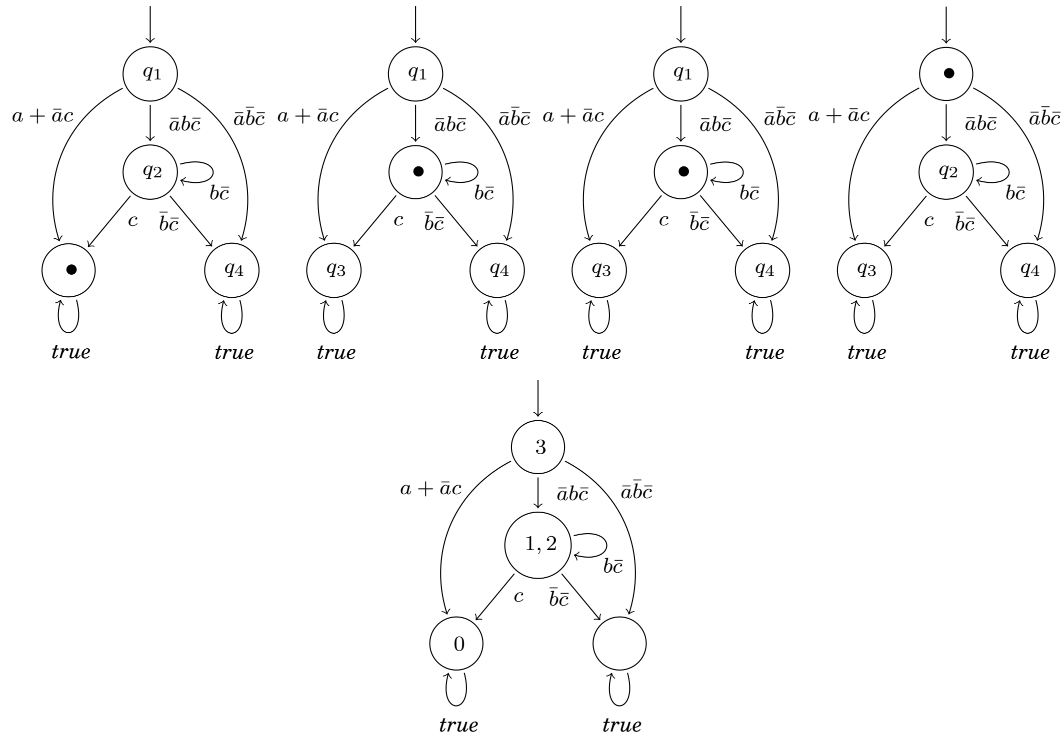
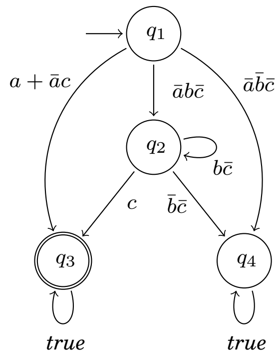


Fig. 2: The top row shows the first four elements of the array of co-Büchi automata for $\mathbf{FG}(a \vee (b\ \mathbf{U}\ c))$ after reading $abc\ \bar{a}b\bar{c}\ \bar{a}b\bar{c}$. At the bottom, the corresponding configuration of the Mojmir automaton.

4.2. Mojmir automata

By the definition of LTL, we have $w \models \mathbf{FG}\varphi$ iff $w_i \models \varphi$ for all but finitely many $i \geq 0$. Let A_φ be the deterministic co-Büchi automaton recognizing $\mathsf{L}(\varphi)$. From a mathematical point of view, we can recognize $\mathsf{L}(\mathbf{FG}\varphi)$ with the help of an infinite array of copies of A_φ. The ith automaton reads w_i, i.e., it skips the first $(i-1)$ letters of the input word, and then starts reading. Therefore, the i-th automaton accepts iff $w_i \models \varphi$. The array accepts iff almost every array element accepts. Figure 2 shows the first four elements of the array for the formula $\mathbf{FG}(a \vee (b\ \mathbf{U}\ c))$. The figure shows the state of the elements after reading $(abc)\ (\bar{a}b\bar{c})\ (\bar{a}b\bar{c})$. For example, the automaton on the left has read all three letters, and reached state q_3, graphically displayed by putting a token on the state, while the next one has only read the last two letters, and reached state q_2. The last automaton has not yet read any letter, and so it is currently in state q_1.

We now observe that the complete array can be replaced by one single automaton that handles all the tokens simultaneously. We call such an automaton a Mojmir automaton. The bottom part of Figure 2 shows the configuration of the Mojmir automaton corresponding to the array at the top. After reading $(abc)\ (\bar{a}b\bar{c})\ (\bar{a}b\bar{c})$, the automaton has created four tokens, labelled with their birthdates. Intuitively, when the automaton reads a letter it moves all tokens according to the transition function, and then puts a fresh token in the initial state, labelled with the position of the letter. Initially there is



	time				
token	0	1	2	3	$\cdots$
0	q_1	q_3	q_3	q_3	$\cdots$
1	$\perp$	q_1	q_2	q_2	$\cdots$
2	$\perp$	$\perp$	q_1	q_2	$\cdots$
3	$\perp$	$\perp$	$\perp$	q_1	$\cdots$
4	$\perp$	$\perp$	$\perp$	$\perp$	$\cdots$
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$

Fig. 3: Mojmir automaton for $\mathbf{FG}(a \vee (b \ \mathbf{U} \ c))$, and matrix representation of $run_w(token, time)$ for $w = abc\ \bar{a}b\bar{c}\ \bar{a}b\bar{c} \cdots$.

a unique token at the initial state, labelled by 0. The automaton accepts if almost every token eventually reaches an accepting state.

Definition 4.2. A Mojmir automaton is a tuple $\mathcal{M} = (Q, \Sigma, q_0, \delta, F)$, where (Q, Σ, q_0, δ) is a DTS and $F \subseteq Q$ is a set of *accepting* states satisfying $\delta(F, \nu) \subseteq F$ for every $\nu \in \Sigma$, i.e., states reachable from accepting states are also accepting.

The *run* of $\mathcal{M}$ over a word $w = w[0]w[1] \cdots \in \Sigma^\omega$ is the infinite sequence

$$(q_0^0)\ (q_0^1, q_1^1)\ (q_0^2, q_1^2, q_2^2)\ (q_0^3, q_1^3, q_2^3, q_3^3) \cdots$$

where

$$q_{token}^{time} = \begin{cases} q_0 & \text{if } token = time, \\ \delta\big(q_{token}^{time-1}, w[time-1]\big) & \text{if } token < time \end{cases}$$

The *position of a token at a time* in the run is given by the function $run_w \colon \mathbb{N} \times \mathbb{N} \to Q \cup \{\perp\}$, defined as follows:

$$run_w(token, time) = \begin{cases} q_{token}^{time} & \text{if } token \leq time \\ \perp & \text{if } token > time \end{cases}$$

For every time $t \in \mathbb{N}$, we denote by $conf_w(t)$ the function defined by

$$token \mapsto run_w(token, t))$$

We call $conf_w(t)$ the *configuration* of the run of $\mathcal{M}$ on w at time t. The run of $\mathcal{M}$ on w is accepting if for almost every $token \in \mathbb{N}$ there exists $time \in \mathbb{N}$ such that $run_w(token, time) \in F$.

Given a $\mathbf{G}$-free formula φ, the Mojmir automaton equivalent to $\mathbf{FG}\varphi$ has exactly the same syntactic structure as the Büchi automaton for φ: only the notions of run and acceptance are different.

Definition 4.3. Let φ be a $\mathbf{G}$-free formula. The Mojmir automaton for $\mathbf{FG}\varphi$ is $\mathcal{M}(\varphi) = (Reach(\varphi), 2^{Ap}, [\varphi]_P, af, \{[\mathbf{tt}]_P\})$.

Since $\mathcal{M}(\varphi)$ accepts iff almost every token eventually reaches an accepting state, $\mathcal{M}(\varphi)$ accepts a word w iff $w \models \mathbf{FG}\varphi$, and so we have:

THEOREM 4.4. *Let φ be a $\mathbf{G}$-free formula. Then* $\mathsf{L}(\mathcal{M}(\varphi)) = \mathsf{L}(\mathbf{FG}\varphi)$.

Example 4.5. Figure 3 shows the Mojmir automaton for $\mathbf{FG}(a \vee (b\ \mathbf{U}\ c))$ and the matrix representation of $run_w(token, time)$ for $w = abc\ \bar{a}b\bar{c}\ \bar{a}b\bar{c} \cdots$. The configurations of the run are given by the columns of the matrix. For instance, $conf_w(2)$ is the mapping $0 \mapsto q_3, 1 \mapsto q_2, 2 \mapsto q_1, \forall i \geq 3 : i \mapsto \bot$ given by the third column, indicating that after two steps the tokens $0, 1, 2$ are in states q_3, q_2, q_1, respectively, and other tokens do not exist yet.

In the rest of the section we show how to construct a deterministic Rabin automaton equivalent to a given Mojmir automaton. In Section 4.3 we define an abstraction that assigns to each configuration $conf_w(t)$ of a run an abstract object $sr_w(t)$, called a *state-ranking*. Since the run of $\mathcal{M}$ on a word w is completely characterized by the sequence of configurations $conf_w(0)\ conf_w(1)\ conf_w(2) \cdots$, the abstraction also abstracts a run into the infinite sequence of state-rankings $sr_w(0)\ sr_w(1)\ sr_w(2) \cdots$. Sections 4.4 and 4.5 show that the abstraction has the following properties:

(1) There is an easily computable function that given $sr_w(t)$ and $w[t+1]$ returns $sr_w(t+1)$. (Lemma 4.11)
(2) A run is accepting iff its corresponding abstract run satisfies a certain Rabin condition. (Definition 4.19)

Finally, Section 4.6 derives the deterministic Rabin automaton. As the reader can expect, the automaton will have the state-rankings as states, the function of (1) as transition function, and the condition of (2) as acceptance condition.

4.3. State-rankings

Intuitively, a state-ranking of a Mojmir automaton $\mathcal{M}$ is a ranking of the states of $\mathcal{M}$. Our state-rankings are allowed to be partial, that is, to leave some states unranked.

Definition 4.6. Let $\mathcal{M}$ be a Mojmir automaton with n states. A state-ranking of $\mathcal{M}$ is a partial injective function $sr\colon Q \to \{\mathbf{1}, \ldots, \mathbf{n}\}$, such that if the image of sr contains $\mathbf{i}$, then it also contains $\mathbf{j}$ for every $\mathbf{j} < \mathbf{i}$. When $sr(q)$ is undefined, we write $sr(q) = \bot$. The set of state-rankings of $\mathcal{M}$ is denoted by $\mathcal{SR}$.

The state-ranking $sr_w(t)$ associated to $conf_w(t)$ is the result of performing a sequence of abstraction steps, which we illustrate on an example. Consider a Mojmir automaton $\mathcal{M}$ with states $\{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}$. Assume that, after the first 8 steps of its run on some word, $\mathcal{M}$ has reached the following configuration, where for each state we give the set of tokens currently at that state:

$$\left(\begin{array}{ccccccc} q_0 & q_1 & q_2 & q_3 & q_4 & q_5 & q_6 \\ \{3,8\} & \{1,2\} & \emptyset & \{5,7\} & \{4\} & \{6\} & \{0\} \end{array} \right) \tag{1}$$

Assume further that states q_5, q_6 are *sinks*, meaning that $\delta(q_5, \nu) = q_5$ and $\delta(q_6, \nu) = q_6$ for every alphabet letter ν[3]. We start the abstraction process by discarding the information about tokens in sinks. We use the symbol $\bot$ to denote this, and obtain:

$$\left(\begin{array}{ccccccc} q_0 & q_1 & q_2 & q_3 & q_4 & q_5 & q_6 \\ \{3,8\} & \{1,2\} & \emptyset & \{5,7\} & \{4\} & \bot & \bot \end{array} \right)$$

We continue by keeping only the *oldest* token of each state (that is, the one with the smallest number). If the state is not populated by any token, again we just write $\bot$. We

[3]For technical reasons, we also decree that the initial state cannot be a sink.

obtain:

$$\begin{array}{ccccccccc} & q_0 & q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & \\ (& 3 & 1 & \perp & 5 & 4 & \perp & \perp &) \end{array}$$

We call tokens $3, 1, 5$ and 4 the *senior tokens* of the configuration, or just the *seniors*.

Since a run has infinitely many tokens, the number of possible abstract configurations of the automaton is still infinite. So we discard even more information. We throw away the identities of the senior tokens, and keep only their relative *seniority rank*: the oldest senior token has rank **1**, the second oldest rank **2**, etc. We obtain the *state-ranking*

$$\begin{array}{ccccccccc} & q_0 & q_1 & q_2 & q_3 & q_4 & q_5 & q_6 & \\ (& \mathbf{2} & \mathbf{1} & \perp & \mathbf{4} & \mathbf{3} & \perp & \perp &) \end{array}$$

It is useful to think of the set of tokens at a state as the partners of a partnership firm. The senior partner is the oldest token. The name of the firm is the rank of the senior partner. For instance, the firm **2** at state q_0 has tokens 3 and 8 as partners.

Let us formally define the rank $rk_w(\tau, t)$ of token τ at time t, and the state-ranking $sr_w(t)$ at time t.

Definition 4.7. Let $\mathcal{M} = (Q, \Sigma, q_0, \delta, F)$ be a Mojmir automaton with n states. A state $q \in Q$ is a *sink* if $q \neq q_0$ and $\delta(q, \nu) = q$ for every $\nu \in \Sigma$.

Let $w \in \Sigma^\omega$ be a word, and consider the run of $\mathcal{M}$ on w. Given two tokens $\tau, \tau' \in \mathbb{N}$, we say that τ is *older* than τ' if $\tau < \tau'$. The *senior of token τ at time $t > \tau$* is the oldest token τ' such that $run_w(\tau, t) = run_w(\tau', t)$. If a token is its own senior, then we call τ a *senior* (at time t).

The *rank* of token τ at time $t > \tau$, denoted by $rk_w(\tau, t)$, is defined as follows:

— If $run_w(\tau, t)$ is a sink, then $rk_w(\tau, t) = \perp$ (we say that τ is *unranked* at time t).
— If $run_w(\tau, t)$ is not a sink, then let s be the senior of token τ at time t. The rank $rk_w(\tau, t)$ is the number of senior tokens τ' such that $run_w(\tau', t)$ is not a sink and $\tau' \leq s$.

(Observe that $run_w(\tau, t) = run_w(\tau', t)$ implies that τ and τ' have the same seniors, and so that $rk_w(\tau, t) = rk_w(\tau', t)$; so all tokens at the same state get the same rank.)

Finally, the *state-ranking* at time t, denoted by $sr_w(t)$, is the mapping $Q \to \mathbb{N}$ that assigns to each state $q \in Q$ its state-ranking $sr_w(t, q) \in \{\mathbf{1}, \ldots, \mathbf{n}\}$, defined as follows:

- If q is a sink, then $sr_w(t, q) = \perp$.
- If q is not a sink and no token τ satisfies $run_w(\tau, t) = q$, then $sr_w(t, q) = \perp$.
- If q is not a sink and some token τ satisfies $run_w(\tau, t) = q$, then $sr_w(t, q) = rk_w(\tau, t)$.

Example 4.8. Consider for example token 7 in the configuration (1). The senior of 7 is 5. The seniors are $3, 1, 5, 4$. Since all seniors are at least as old as 5, the rank of token 7 is **4**. Since the configuration is the result of reading the first 8 letters of a word w, we have $rk_w(7, 8) = \mathbf{4}$.

While the birthdate of a token does not change along a run, its rank can change, and for two different reasons. Assume the current rank of a token τ is **4**. If the firm of rank, say, **3**, moves to a sink, then it "disappears", and the rank of τ is upgraded to **3**. If the token's firm merges with the firm of rank, say, **2**, the rank of τ is upgraded to **2**. In both cases, we observe that, as long as the token does not reach a sink, its rank can only improve (get older) along a run.

LEMMA 4.9. *Let $\mathcal{M} = (Q, \Sigma, q_0, \delta, F)$ be a Mojmir automaton and let $w \in \Sigma^\omega$ be a word. For every token $\tau \in \mathbb{N}$:*

- *if* $rk_w(\tau, t) = \bot$ *for some* $t \in \mathbb{N}$*, then* $rk_w(\tau, t') = \bot$ *for every* $t' \geq t$.
- *if* $t \leq t'$ *and* $rk_w(\tau, t), rk_w(\tau, t') \in \mathbb{N}$*, then* $rk_w(\tau, t) \geq rk_w(\tau, t')$.

PROOF. Follows easily from the definitions. $\square$

4.4. Computing the successor of a state-ranking

Recall that the run of a Mojmir automaton on a word w is completely determined by the sequence of configurations $conf_w(0)\, conf_w(1)\, conf_w(2) \cdots$. To this sequence corresponds a sequence $sr_w(0)\, sr_w(1), sr_w(2) \cdots$ of state-rankings. We show that $sr_w(t+1)$ can be directly computed from $sr_w(t)$ and the letter $w[t+1]$. More precisely, we define a function $nxt \colon \mathcal{SR} \times \Sigma \to \mathcal{SR}$ and show that it satisfies $nxt(sr_w(t), w[t+1]) = sr_w(t+1)$ for every time t.

Let $sr_w(t)$ be the state-ranking

$$\begin{array}{c} \begin{array}{ccccccc} q_0 & q_1 & q_2 & q_3 & q_4 & q_5 & q_6 \end{array} \\ (\begin{array}{ccccccc} \mathbf{2} & \mathbf{1} & \bot & \mathbf{4} & \mathbf{3} & \bot & \bot \end{array}) \end{array}$$

Assume $w[t+1] = \nu$ for some $\nu \in \Sigma$, and assume further that

$$\delta(q_0, \nu) = q_5 \qquad \delta(q_1, \nu) = q_2 = \delta(q_3, \nu) \qquad \delta(q_4, \nu) = q_3$$

We obtain $sr_w(t+1)$ in four steps:

(i) Move all senior tokens according to δ.
The token of rank $\mathbf{2}$ at q_0 moves to the sink q_5 (recall that q_5 and q_6 are sinks) and "disappears". The tokens of ranks $\mathbf{1}$ and $\mathbf{4}$ move to state q_2. The token of rank $\mathbf{3}$ at q_4 moves to q_3. We obtain:

$$\begin{array}{c} \begin{array}{ccccccc} q_0 & q_1 & q_2 & q_3 & q_4 & q_5 & q_6 \end{array} \\ (\begin{array}{ccccccc} \bot & \bot & \{\mathbf{1}, \mathbf{4}\} & \mathbf{3} & \bot & \bot & \bot \end{array}) \end{array}$$

(ii) If a state holds more than one token, keep only the most senior token.
Only the token of rank $\mathbf{1}$ survives in q_2. Intuitively, the firms with rank $\mathbf{1}$ and $\mathbf{4}$ merge, and $\mathbf{1}$ becomes the senior partner.

$$\begin{array}{c} \begin{array}{ccccccc} q_0 & q_1 & q_2 & q_3 & q_4 & q_5 & q_6 \end{array} \\ (\begin{array}{ccccccc} \bot & \bot & \mathbf{1} & \mathbf{3} & \bot & \bot & \bot \end{array}) \end{array}$$

(iii) Recompute the seniority ranks of the remaining tokens.
The token of rank $\mathbf{3}$ is upgraded to rank $\mathbf{2}$.

$$\begin{array}{c} \begin{array}{ccccccc} q_0 & q_1 & q_2 & q_3 & q_4 & q_5 & q_6 \end{array} \\ (\begin{array}{ccccccc} \bot & \bot & \mathbf{1} & \mathbf{2} & \bot & \bot & \bot \end{array}) \end{array}$$

(iv) If there is no token on the initial state, add one with the next lowest seniority rank.
We add a token to q_0 of rank $\mathbf{3}$.

$$\begin{array}{c} \begin{array}{ccccccc} q_0 & q_1 & q_2 & q_3 & q_4 & q_5 & q_6 \end{array} \\ (\begin{array}{ccccccc} \mathbf{3} & \bot & \mathbf{1} & \mathbf{2} & \bot & \bot & \bot \end{array}) \end{array}$$

The corresponding formal definition is:

Definition 4.10. Let $\mathcal{M} = (Q, \Sigma, q_0, \delta, F)$ be a Mojmir automaton with n states and a set S of sinks. Let sr be a state-ranking of $\mathcal{M}$, and let $\nu \in \Sigma$. For every $q \in Q$, the set of ranks of sr that move to q under ν, denoted by $mvto(q)$, is given by:

$$mvto(q) = \begin{cases} \{sr(q') \mid sr(q') \neq \bot \wedge \delta(q', \nu) = q\} & \text{if } q \neq q_0 \\ \{sr(q') \mid sr(q') \neq \bot \wedge \delta(q', \nu) = q\} \cup \{\mathbf{n}\} & \text{if } q = q_0 \end{cases}$$

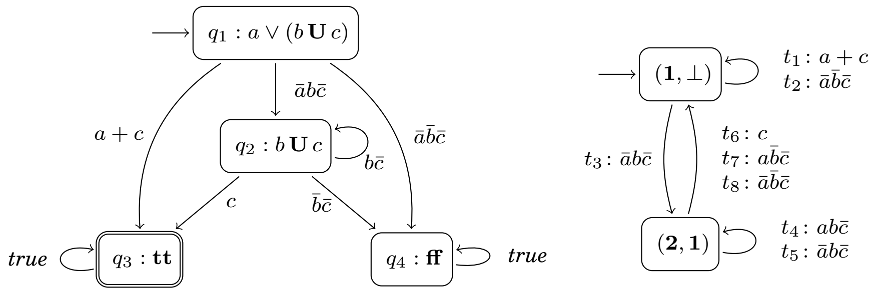


Fig. 4: A Mojmir automaton for $a \vee (b\ \mathbf{U}\ c)$ and its corresponding DRA.

The state-ranking $nxt(sr, \nu)$ is defined with $\min(\emptyset) = \infty$ by:

$$nxt(sr, \nu, q) = \begin{cases} |\{q' \in Q \setminus S \mid \min(mvto(q')) \leq \min(mvto(q))\}| & \text{if } q \notin S \text{ and } mvto(q) \neq \emptyset \\ \bot & \text{otherwise} \end{cases}$$

We get the following lemma.

LEMMA 4.11. *Let $\mathcal{M}$ be a Mojmir automaton and let w be a word. Then $sr_w(t+1) = nxt(sr_w(t), w[t+1])$ for every $t \geq 0$.*

PROOF. (Sketch.) The key observation for the proof is that $nxt(sr_w(t), w[t+1])$ computes for a state q the set of senior states q' at time $t+1$ and then takes the cardinality of this set as a value. This coincides with the definition of $sr_w(t+1)$. □

We already have all we need to define the states and transition function of the DRA equivalent to a given Mojmir automaton (although not the acceptance condition). The states of the Rabin automaton are the state-rankings, and the transition function is given by nxt.

Example 4.12. Figure 4 shows our running example on the left, and the states and transitions of its corresponding Rabin automaton on the right. Since states q_3 and q_4 are sinks, state rankings only rank states q_1 and q_2. The initial state-ranking is $(\mathbf{1}, \bot)$. The only other state-ranking reachable from it turns out to be $(\mathbf{2}, \mathbf{1})$.

4.5. Deciding acceptance of an abstract run

We define a Rabin acceptance condition that turns the transition system above into a DRA equivalent to the Mojmir automaton. We start by classifying the tokens of a run of the Mojmir automaton.

Definition 4.13. Let $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$ be a Mojmir automaton and let w be a word. A token $\tau \in \mathbb{N}$ of the run of $\mathcal{M}$ on w

— *squats* if it never reaches a sink (that is, if $run_w(\tau, t) \in Q \setminus S$ for every $t \in \mathbb{N}$);
— *fails* if it eventually reaches a non-accepting sink (that is, if there exists $t \in \mathbb{N}$ such that $run_w(\tau, t) \in S \setminus F$);
— *succeeds* if it eventually reaches an accepting state, sink or non-sink (that is, if there exists $t \in \mathbb{N}$ such that $run_w(\tau, t) \in F$).

Further, we say that a token *succeeds at rank* **i** if it has rank **i** immediately before entering the set of accepting states, i.e., if there is $t \in \mathbb{N}$ such that $run_w(\tau, t) \notin F \setminus \{q_0\}$, $run_w(\tau, t+1) \in F$, and $rk_w(\tau, t) = \mathbf{i}$.[4]

Observe that the three classes are not disjoint. More precisely, a token either fails, succeeds, or squats in non-accepting states. By definition, a Mojmir automaton accepts a word w if all but finitely many of the tokens generated during the run on w succeed (recall that tokens that reach an accepting state stay within the set of accepting states). So, given the abstract run of $\mathcal{M}$ on w, our task is to find a Rabin condition equivalent to "only finitely many tokens fail and only finitely many tokens squat in non-accepting states". The condition equivalent to "only finitely many tokens fail" is simple: since a token fails when it moves into a non-accepting sink, we stipulate that transitions moving tokens into non-accepting sinks can only occur finitely often.

Finding a condition equivalent to "only finitely many tokens squat in non-accepting states" is a bit more involved. Observe that, since a squatter τ never reaches a sink, it has a rank at every moment in time. So, if infinitely many tokens squat in non-accepting states, then, since they are all confined within $Q \setminus (S \cup F)$, infinitely many firm merges must take place in this set of states. This suggests the following definition:

Definition 4.14. Let $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$ be a Mojmir automaton and let w be a word. Let $\tau, \tau' \in \mathbb{N}$ be two tokens such that $\tau < \tau'$. We say that τ and τ' *merge* during the run of $\mathcal{M}$ on w if there is $t \in \mathbb{N}$ and a state $q \notin F$ such that $run_w(\tau, t) = q = run_w(\tau', t)$, and one of the two following conditions hold:

— $\tau' < t$ and $run_w(\tau, t-1) \neq run_w(\tau', t-1)$.
(Both tokens already existed at time $t-1$, and were at different states)
— $\tau' = t$.
(Token τ' is created at time t.)

Further, we say that the tokens merge *at rank* **i** if $rk_w(\tau, t) = \mathbf{i}$.

Notice the condition $q \notin F$ in the definition: we reserve the term "merge" for the merges occurring in non-accepting states.

If two tokens merge at some time t, then from that moment on they follow the same trajectory, and so we have:

LEMMA 4.15. *Let $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$ be a Mojmir automaton and let w be a word. Let $\tau, \tau' \in \mathbb{N}$ be two tokens that merge along the run of $\mathcal{M}$ on w. Then either both τ and τ' fail, or both succeed at the same rank, or both squat.*

PROOF. By the definition of *m*erge there is a time t_0 such that $run_w(\tau, t_0) = q \notin F$ and $run_w(\tau, t) = run_w(\tau', t)$ for all $t \geq t_0$. We proceed by case distinction and only consider two cases.

— τ fails. This means that the token τ moves at some point to a non-accepting sink and stays there forever. Let us call this time t'. Without loss of generality we assume that the *merge* happens outside the sinks S and we have $t' > t_0$. Hence we have $run_w(\tau', t') = run_w(\tau, t') = q_s$ and thus τ' also fails.
— τ succeeds at rank i. Thus the token τ moved at some time $t' > t_0$ from the non-accepting states to the accepting states with rank i. Since τ and τ' already merged and tokens that are in the same state have the same rank, also τ' succeeds with rank i.

[4]observe that in the special case $q_0 \in F$ (all states are accepting), the first move of each token is considered succeeding.

□

We can now formulate and prove the main theorem of the section, presenting conditions equivalent to “only finitely many tokens fail” (condition (1)), and “only finitely many tokens squat in non-accepting states” (condition (2)):

THEOREM 4.16. *Let $\mathcal{M} = (Q, \Sigma, \delta, q_0, F)$ be a Mojmir automaton and let w be a word. $\mathcal{M}$ accepts w if and only if the run of $\mathcal{M}$ on w satisfies the following two conditions:*

(1) Finitely many tokens fail.
(2) There is a rank $\mathbf{i}$ *such that*
(2.1) infinitely many tokens succeed at rank $\mathbf{i}$*, and*
(2.2) finitely many pairs of tokens merge at rank older than $\mathbf{i}$*, i.e. with a rank* $\mathbf{j} < \mathbf{i}$*.*

PROOF. ($\Rightarrow$): Assume $\mathcal{M}$ accepts w. Then almost every token of the run of $\mathcal{M}$ on w succeeds. Therefore, since no token can succeed and fail, (1) holds.

Let $\mathbf{i}$ be the smallest rank satisfying (2.1) (since almost all tokens succeed and the number of ranks are finite, such an $\mathbf{i}$ exists). We prove that $\mathbf{i}$ satisfies (2.2). Let M_i be the set of pairs (τ, τ') of tokens such that $\tau < \tau'$ and τ and τ' merge at rank older than $\mathbf{i}$. We prove that M_i is finite. By Lemma 4.15 either both τ and τ' succeed, or none succeeds. Let S_i be the set of pairs $(\tau, \tau') \in M_i$ such that both τ and τ' succeed. Since $\mathcal{M}$ accepts w, almost every token succeeds, and so $M_i \setminus S_i$ is finite.

It remains to prove that S_i is finite. By the definition of $\mathbf{i}$, it suffices to prove that for every $(\tau, \tau') \in S_i$ both τ and τ' succeed at a rank older than $\mathbf{i}$. Let t_0 be the time at which τ and τ' merge. By the definition of a merge, at time t_0 neither τ nor τ' have reached the set of accepting states. Since τ and τ' merge at rank older than $\mathbf{i}$ and two merged tokens always have the same rank, we have $rk_w(\tau', t_0) < \mathbf{i}$. Let $t_1 > t_0$ be the time at which both tokens enter the set of accepting states. By Lemma 4.9(2), we have $rk_w(\tau, t_1) < \mathbf{i}$ and $rk_w(\tau', t_1) < \mathbf{i}$, and so both τ and τ' succeed at a rank older than $\mathbf{i}$.

($\Leftarrow$): If $q_0 \in F$ then by the definition of Mojmir automata $\mathcal{M}$ accepts every word, and we are done. So assume $q_0 \notin F$.

By the definition of squatting, a token τ squats iff $rk_w(\tau, t) \in \mathbb{N}$ for every $t \geq \tau$. By Lemma 4.9, the rank of τ can only get older, and so there is a time t such that $rk_w(\tau, t) = rk_w(\tau, t')$ for every $t' \geq t$. We call this rank the *stable rank* of τ, denoted by $strk_w(\tau)$. The following lemma, proved in the Appendix, shows that all stable ranks are old.

LEMMA 4.17. *Let* $\mathbf{i}$ *be the rank of condition (2). If the rank of τ stabilizes, then* $strk_w(\tau) < \mathbf{i}$*.*

We now use the lemma to prove the result by contradiction. Assume $\mathcal{M}$ does not accept w. Then, infinitely many tokens do not succeed in the run of $\mathcal{M}$ on w. Since by (1) only finitely many tokens fail, infinitely many tokens squat in non-accepting states. By Lemma 4.17, their stable ranks are all older than $\mathbf{i}$. So there is a rank $\mathbf{j} < \mathbf{i}$ such that infinitely many tokens have stable rank $\mathbf{j}$. Let τ be one of these tokens, and let t be the time at which its rank stabilizes. All tokens born after t whose rank stabilize at $\mathbf{j}$ eventually merge with τ. Therefore, infinitely many pairs (τ, τ') merge at rank $\mathbf{i}$. But this contradicts our assumption that (2.2) holds. □

We conclude the section with a definition that will be important in Section 6.

Definition 4.18. Let $\mathcal{M}$ be a Mojmir automaton and let w be a word. We say that $\mathcal{M}$ accepts w *at rank* $\mathbf{i}$ if $\mathcal{M}$ accepts w and the rank of condition (2) in Theorem 4.16 is $\mathbf{i}$.

Note that a word can be accepted at several ranks. In Section 6.2 we will show that the ranks at which the automaton $\mathcal{M}(\varphi)$ of a formula φ accepts a word carry useful information.

4.6. From Mojmir automata to deterministic Rabin automata

From Theorem 4.16 we can easily derive a deterministic Rabin automaton equivalent to a given Mojmir automaton. More precisely, we show how to construct an automaton with a Rabin condition on transitions. Applying the construction of Section 2.3.1, this automaton can be transformed into one with a Rabin condition on states.

Definition 4.19. Let $\mathcal{M} = (Q, \Sigma, i, \delta, F)$ be a Mojmir automaton with a set S of sinks. The deterministic Rabin automaton $\mathcal{R}(\mathcal{M}) = (Q_{\mathcal{R}}, \Sigma, q_{0\mathcal{R}}, \delta_{\mathcal{R}}, \alpha_{\mathcal{R}})$ is defined as follows:

— $Q_{\mathcal{R}}$ is the set $\mathcal{SR}$ of state-rankings of $\mathcal{M}$;
— $q_{0\mathcal{R}}$ is the state-ranking satisfying $q_{0\mathcal{R}}(q_0) = \mathbf{1}$ and $q_{0\mathcal{R}}(q) = \bot$ for every $q \neq q_0$;
— $\delta_{\mathcal{R}}(sr, \nu) = nxt(sr, \nu)$ for every state-ranking sr and letter ν;
— $\alpha_{\mathcal{R}} = \bigvee_{i=1}^{|Q|} P_i$, where the ith Rabin pair is $P_i = (\mathit{fail} \cup \mathit{merge}(\mathbf{i}), \mathit{succeed}(\mathbf{i}))$, and the sets fail, $\mathit{merge}(\mathbf{i})$, and $\mathit{succeed}(\mathbf{i})$ are defined as follows. A transition $(sr, \nu, sr') \in \delta_{\mathcal{R}}$ belongs to
  — fail if there exists $q \in Q$ such that $sr(q) \in \mathbb{N}$ and $\delta(q, \nu) \in S \setminus F$.
  — $\mathit{succeed}(\mathbf{i})$ if there exists $q \notin F$ such that $sr(q) = \mathbf{i}$ and $\delta(q, \nu) \in F$, or $q_0 \in F$ and $sr(q_0) = \mathbf{i}$.[5]
  — $\mathit{merge}(\mathbf{i})$ if
    — there exists a state $q \in Q \setminus F$ and distinct states $q_1, q_2 \in Q$ such that $\delta(q_1, \nu) = q = \delta(q_2, \nu)$, $sr(q_1) < \mathbf{i}$, and $sr(q_2) \neq \bot$; or
    — $q_0 \notin F$, and there exists a state q such that $\delta(q, \nu) = q_0$ and $sr(q) < \mathbf{i}$.[6]

$\mathcal{R}(\mathcal{M})$ accepts a word w ***at rank*** $\mathbf{j}$ if P_j is an accepting pair on the run of $\mathcal{R}(\mathcal{M})$ on w.

Example 4.20. Let us determine the accepting pairs of the DRA on the right of Figure 4. We examine several representative cases.

— t_1 moves tokens from q_1 to the accepting sink q_3. Since $sr(q_1) = 1$, transition t_1 belongs to $\mathit{succeed}(\mathbf{1})$. Since we can safely ignore sinks (q_3, q_4) and states that are empty (q_2) for testing membership, we are done with t_1.
— t_2 takes tokens from the initial state and moves them to the non-accepting sink q_4. This matches the definition of fail, with $sr(q_1) \in \mathbb{N}$ and $\delta(q_1, \bar{a}\bar{b}\bar{c}) = q_4 \in S \setminus F$. Hence $t_2 \in \mathit{fail}$.
— t_3 moves tokens from q_1 to q_2. Since q_2 is neither a sink nor an accepting state, t_3 is not contained in fail or in any $\mathit{succeed}$ set. Moreover, since $sr(q_2) = \bot$, it does not belong to any merge set either.
— t_8 moves tokens from q_1 and q_2 to the non-accepting sink q_3. Hence $t_8 \in \mathit{fail}$. Moreover, the transition makes the firms from q_1 and q_2 to merge in q_3 with rank $sr(q_1) = 1$, and so t_8 is also contained in $\mathit{merge}(\mathbf{2})$.

Altogether we obtain

$$\mathit{fail} = \{t_2, t_7, t_8\} \quad \begin{array}{l} \mathit{merge}(\mathbf{1}) = \emptyset \\ \mathit{merge}(\mathbf{2}) = \{t_5, t_8\} \end{array} \quad \begin{array}{l} \mathit{succeed}(\mathbf{1}) = \{t_1, t_6\} \\ \mathit{succeed}(\mathbf{2}) = \{t_4, t_6, t_7\} \end{array}$$

[5]If q_0 is accepting then, by the definition of Mojmir automaton, all states reachable from q_0 are accepting. This condition covers the corner case in which no transition into an accepting state is possible, because all states are accepting state.
[6]In this case there is a merge between the firm at q and the token newly created on state q_0.

It is easy to see that the runs accepted by the pair P_1 are those that take t_2, t_7, t_8 only finitely often, and visit $(\mathbf{1}, \perp)$ infinitely often. They are accepted at rank $\mathbf{1}$. The runs accepted at rank $\mathbf{2}$ are those accepted by P_2 but not by P_1. They take $t_1, t_2, t_5, t_6, t_7, t_8$ finitely often, and so they are exactly the runs with a t_4^ω suffix.

LEMMA 4.21. *Let $\mathcal{M} = (Q, \Sigma, i, \delta, F)$ be a Mojmir automaton, and let $\mathcal{R}(\mathcal{M})$ be its corresponding Rabin automaton. For every word w, the sequence $conf_w(0) conf_w(1) \cdots$ is the run of $\mathcal{M}$ on w iff $sr_w(0) sr_w(1) \cdots$ is the run of $\mathcal{R}(\mathcal{M})$ on w.*

The Rabin condition of this automaton checks conditions (1) and (2) of Theorem 4.16. Consider a transition $conf_w(t) \xrightarrow{a} conf_w(t+1)$ between two configurations of $\mathcal{M}$ in which some token moves into a non-accepting sink. Then the transition $sr_w(t) \xrightarrow{a} sr_w(t+1)$ clearly belongs to the set $fail$, and vice versa. Similarly, transitions of $succeed(\mathbf{i})$ correspond to transitions of $\mathcal{M}$ that make some token succeed at rank $\mathbf{i}$, and transitions of $merge(\mathbf{i})$ correspond to transitions of $\mathcal{M}$ that merge two tokens at rank $\mathbf{i}$. So we obtain:

THEOREM 4.22. *Let $\mathcal{M}$ be a Mojmir automaton, and let $\mathcal{R}(\mathcal{M})$ be its corresponding Rabin automaton. Then $\mathsf{L}(\mathcal{M}) = \mathsf{L}(\mathcal{R}(\mathcal{M}))$. Moreover, for every $w \in \mathsf{L}(\mathcal{M})$ both $\mathcal{M}$ and $\mathcal{R}(\mathcal{M})$ accept w at the same ranks.*

5. DRAS FOR ARBITRARY $\mathbf{FG}$-FORMULAE

We show how to translate formulae of the form $\mathbf{FG}\varphi$ into DRAs. Thanks to the results of Section 4, it suffices to translate them into Mojmir automata. We show that the Mojmir automaton for a formula can be defined compositionally, as an intersection of Mojmir automata. The next proposition shows that Mojmir automata are closed under union and intersection (the proof can be found in the Appendix).

PROPOSITION 5.1. *Let $\mathcal{M}_1 = (Q_1, \Sigma, q_{01}, \delta_1, F_1)$ and $\mathcal{M}_2 = (Q_2, \Sigma, q_{02}, \delta_2, F_2)$. Let $Q = Q_1 \times Q_2$, let $q_0 = (q_{01}, q_{02})$, and let $\delta \colon Q \times \Sigma \to Q$ be the function given by $\delta(q_1, q_2, \nu) = (\delta_1(q_1, \nu), \delta_2(q_2, \nu))$ Then the tuples*

$$\begin{aligned} \mathcal{M}_1 \cap \mathcal{M}_2 &= \big(Q, \Sigma, q_0, \delta, F_1 \times F_2\big) \\ \mathcal{M}_1 \cup \mathcal{M}_2 &= \big(Q, \Sigma, q_0, \delta, (F_1 \times Q_2) \cup (Q_1 \times F_2)\big) \end{aligned}$$

are also Mojmir automata, and moreover $\mathsf{L}(\mathcal{M}_1 \cap \mathcal{M}_2) = \mathsf{L}(\mathcal{M}_1) \cap \mathsf{L}(\mathcal{M}_2)$ and $\mathsf{L}(\mathcal{M}_1 \cup \mathcal{M}_2) = \mathsf{L}(\mathcal{M}_1) \cup \mathsf{L}(\mathcal{M}_2)$.

5.1. A compositional construction: Intuition

We present the intuition behind the construction by means of an example. Consider the formula

$$\varphi = \mathbf{FG}(\mathbf{F}a \vee (\mathbf{G}(a \vee \mathbf{F}b) \wedge c)))$$

We use the abbreviations $\psi_2 = a \vee \mathbf{F}b$ and $\psi_1 = \mathbf{F}a \vee (\mathbf{G}\psi_2 \wedge c)$, and so we also refer to the formula as $\mathbf{FG}\psi_1$.

We cannot directly apply the construction of the last section because $\mathbf{FG}\psi_1$ contains the $\mathbf{G}$-subformula $\mathbf{G}\psi_2$. However, since ψ_2 does not contain any $\mathbf{G}$-subformula, we can construct a Mojmir automaton $\mathcal{M}(\psi_2)$ for $\mathbf{FG}\psi_2$. We use this fact to define the automaton $\mathcal{M}(\psi_1)$ as the union of two Mojmir automata: The first automaton recognizes all words satisfying $\mathbf{FG}\psi_1$ but not $\mathbf{FG}\psi_2$ (and perhaps some other words satisfying $\mathbf{FG}\psi_2$), while the second recognizes all words satisfying $\mathbf{FG}\psi_1$ and $\mathbf{FG}\psi_2$ (and perhaps some other words satisfying $\mathbf{FG}\psi_1$). Consider for example the words

$$w_1 = (a\bar{b}\bar{c}\;\bar{a}\bar{b}c)^\omega \qquad w_2 = (\bar{a}bc)^\omega \qquad w_3 = (\bar{a}\bar{b}c)^\omega$$

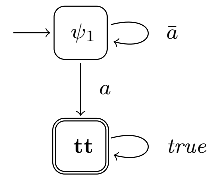


Fig. 5: Mojmir automaton for words satisfying $\mathbf{FG}\psi_1$ but not $\mathbf{FG}\psi_2$.

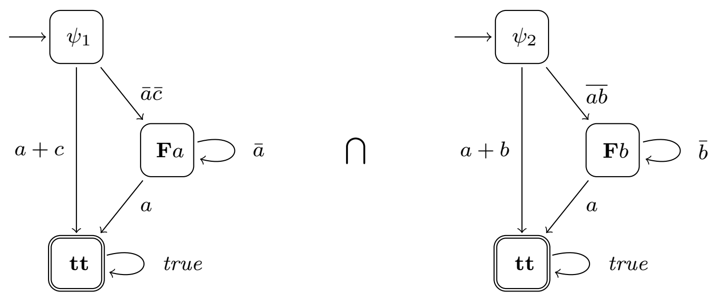


Fig. 6: The automata $\mathcal{M}(\psi_1, \{\psi_2\})$ and $\mathcal{M}(\psi_2)$.

We have $w_1 \models \mathbf{FG}\psi_1 \wedge \neg\mathbf{FG}\psi_2$, $w_2 \models \mathbf{FG}\psi_1 \wedge \mathbf{FG}\psi_2$ and $w_3 \not\models \mathbf{FG}\psi_1$. So both automata will reject w_3. Moreover, the first automaton will accept w_1, and the second w_2.

The first automaton, called $\mathcal{M}(\psi_1, \emptyset)$ in Section 5.2 below, is just the Mojmir automaton for the formula $\mathbf{FG}\psi_1[\mathbf{G}\psi_2/\mathbf{ff}]$, i.e., the result of substituting $\mathbf{G}\psi_2$ by $\mathbf{ff}$ in $\mathbf{FG}\psi_1$. It is easy to see that, since ψ_1 is in negation normal form, $\mathbf{FG}\psi_1[\mathbf{G}\psi_2/\mathbf{ff}]$ logically implies $\mathbf{FG}\psi_1$, and so every word accepted by $\mathcal{M}(\psi_1, \emptyset)$ satisfies $\mathbf{FG}\psi_1$. Moreover, observe that if a word w does not satisfy $\mathbf{FG}\psi_2$, then the formula $\mathbf{G}\psi_2$ is false for every suffix w_i of w, and so, intuitively, treating $\mathbf{FG}\psi_2$ as false still allows $\mathcal{M}(\varphi, \emptyset)$ to accept all words $\mathbf{FG}\psi_1$ but not $\mathbf{FG}\psi_2$. The automaton $\mathcal{M}(\varphi, \emptyset)$ that treats $\mathbf{G}\psi_2$ as $\mathbf{ff}$ is shown in Figure 5. To observe the effect of "treating $\mathbf{G}\psi_2$ as $\mathbf{ff}$", consider state ψ_1 and the letter $\bar{a}bc$. If we used the function af as transition relation, then we would obtain the transition $\psi_1 \xrightarrow{\bar{a}bc} \mathbf{F}a \vee (\mathbf{G}\psi_2 \wedge \mathbf{F}b)$. Instead, since $\mathbf{G}\psi_2$ is treated as $\mathbf{ff}$, we get $\psi_1 \xrightarrow{\bar{a}bc} \mathbf{F}a$.

The second automaton is the intersection of two Mojmir automata. The first one is $\mathcal{M}(\psi_2)$, the Mojmir automaton for ψ_2, which guarantees that the intersection only accepts words satisfying $\mathbf{FG}\psi_2$. The second one, which will be called $\mathcal{M}(\psi_1, \{\psi_2\})$ in Section 5.2, is intuitively in charge of checking that a word w satisfies $\mathbf{FG}\psi_1$ assuming that it satisfies $\mathbf{FG}\psi_2$. Both automata are shown in Figure 6. We choose $\mathcal{M}(\psi_1, \{\psi_2\})$ as the Mojmir automaton for $\mathbf{FG}\psi_1[\mathbf{G}\psi_2/\mathbf{tt}]$. At first sight, since $\mathbf{FG}\psi_2$ and $\mathbf{G}\psi_2$ are not equivalent, replacing $\mathbf{G}\psi_2$ by $\mathbf{tt}$ looks wrong. Let us see why it is correct. Since $\mathbf{G}\psi_2$ eventually holds, the assumption that $\mathbf{G}\psi_2$ is true can only be incorrect for a finite time, or, in other words, for a finite number of tokens. Now we observe that the acceptance condition of Mojmir automata is insensitive to the fate of a finite number of tokens: if almost every token eventually reaches the accepting states, then after changing the fate of a finite number of tokens this is still the case, and vice versa. So replacing $\mathbf{G}\psi_2$ by $\mathbf{tt}$ is correct after all.

Consider state ψ_1 of $\mathcal{M}(\psi_1, \{\psi_2\})$. If we used the function af as transition relation, then we would obtain the transition $\psi_1 \xrightarrow{\bar{a}c} \mathbf{F}a \vee \mathbf{G}\psi_2$. Since we handle $\mathbf{G}\psi_2$ as $\mathbf{tt}$, we get $\psi_1 \xrightarrow{\bar{a}c} \mathbf{tt}$ instead.

We have thus constructed an automaton for $\mathbf{FG}(\mathbf{F}a \vee (\mathbf{G}(a \vee \mathbf{F}b) \wedge c))$. To handle formulae $\mathbf{FG}\psi$ where ψ has multiple $\mathbf{G}$-subformulae $\mathbf{G}\psi_1, \ldots, \mathbf{G}\psi_n$, possibly nested within each other, we generalize the procedure above, and construct an automaton $\mathcal{M}(\varphi, \mathcal{G})$ for each subset $\mathcal{G}$ of $\mathbf{G}$-subformulae. The automaton $\mathcal{M}(\varphi, \mathcal{G})$ accepts all words w such that $w \models \varphi$ and $w \models \mathbf{FG}\psi$ for every $\mathbf{G}\psi \in \mathcal{G}$. The automaton is an intersection of automata, one for each formula in $\mathcal{G}$. The automaton for $\mathbf{G}\psi_i$ handles the $\mathbf{G}$-subformulae of ψ_i that belong to $\mathcal{G}$ as $\mathtt{tt}$. Observe that circularity assumptions of the form "the automaton for $\mathbf{G}\psi_1$ assumes that $\mathbf{FG}\psi_2$ holds, and the automaton for $\mathbf{G}\psi_2$ assumes that that $\mathbf{FG}\psi_1$ holds" are not possible because no two formulae can be subformulae of each other.

The final point is to address the state-explosion problem. In the construction above, the final Mojmir automaton for a formula with $\mathbf{G}$-subformulae $\mathbf{G}\psi_1, \ldots, \mathbf{G}\psi_n$ is the union of 2^n Mojmir automata, and has an unacceptably large number of states. Fortunately, we can construct all these automata so that they have exactly the same states and transitions, and only differ on their set of accepting states. The idea is to construct $\mathcal{M}(\psi, \mathcal{G})$ using a different transition function. We replace af by another function $af_{\mathbf{G}}$ that behaves like af, except for $\mathbf{G}$ subformulae, where we set $af_{\mathbf{G}}(\mathbf{G}\psi, \nu) = \mathbf{G}\psi$ instead of $af(\mathbf{G}\psi, \nu) = \mathbf{G}\psi \wedge af(\psi, \nu)$. Intuitively, we leave the decision whether to handle $\mathbf{G}\psi$ as $\mathtt{tt}$ or $\mathtt{ff}$ "open". Then, for every set $\mathcal{G}$ we choose the accepting states appropriately: Since $\mathcal{M}(\varphi, \mathcal{G})$ assumes that all the formulae of $\mathcal{G}$ are true, we choose as accepting states those whose corresponding formulae are propositionally implied by $\mathcal{G}$.

In our example, both $\mathcal{M}(\psi_1, \emptyset)$ and $\mathcal{M}(\psi_1, \{\psi_2\})$ are the intersection of the two automata of Figure 7; they differ only in the accepting states. In the case of $\mathcal{M}(\psi_1, \emptyset)$, the left automaton treats $\mathbf{G}\psi_2$ as $\mathtt{ff}$, and the right automaton is redundant; therefore, the only accepting state of the left automaton is $\mathtt{tt}$, and all states of the right automaton are accepting. In the case of $\mathcal{M}(\psi_1, \{\psi_2\})$, the left automaton on the left treats $\mathbf{G}\psi_2$ as $\mathtt{tt}$, and the right automaton checks that $\mathbf{G}\psi_2$ holds; therefore, the accepting states of the left automaton are $\mathbf{F}a \vee \mathbf{GF}\psi_2$ and , and the only accepting state of the right automaton is $\mathtt{tt}$.

5.2. Logical Characterization

In order to formalize the notion of "handling a subformula $\mathbf{G}\psi$ as $\mathtt{tt}$" we introduce the following definition:

Definition 5.2. Let φ be a formula and $\nu \in 2^{Ap}$. The formula $af_{\mathbf{G}}(\varphi, \nu)$ is inductively defined as $af(\varphi, \nu)$, with only this difference:

$$af_{\mathbf{G}}(\mathbf{G}\varphi, \nu) = \mathbf{G}\varphi \qquad \text{(instead of } af(\mathbf{G}\varphi, \nu) = af(\varphi, \nu) \wedge \mathbf{G}\varphi).$$

We define $Reach_{\mathbf{G}}(\varphi) \;=\; \{[af_{\mathbf{G}}(\varphi, w)]_P \mid w \in (2^{Ap})^*\}$.

Example 5.3. Let $\varphi = \psi \mathbf{U} \neg a$, where $\psi = \mathbf{G}(a \wedge \mathbf{X}\neg a)$. We have

$$\begin{array}{lllll} af_{\mathbf{G}}(\varphi, \{a\}) & = & af_{\mathbf{G}}(\psi, \{a\}) \wedge \varphi & \equiv_p & \psi \wedge \varphi \\ af(\varphi, \{a\}) & = & af(\psi, \{a\}) \wedge \varphi & \equiv_p & \neg a \wedge \psi \wedge \varphi \end{array}$$

The logical characterization theorem will be an easy corollary of Lemma 5.5 below. Given an arbitrary formula φ and a word w, the lemma characterizes the set of $\mathbf{G}$-subformulae of φ that eventually hold at a word w, i.e., the subformulae $\mathbf{G}\psi$ such that $w \models \mathbf{FG}\psi$. If φ is of the form $\mathbf{FG}\psi$, then clearly $w \models \varphi$ iff the subformula $\mathbf{G}\psi$ belongs to this set.

Definition 5.4. Given a formula φ, we denote by $\mathbb{G}(\varphi)$ the set of $\mathbf{G}$-subformulae of φ, i.e., the subformulae of φ of the form $\mathbf{G}\psi$.

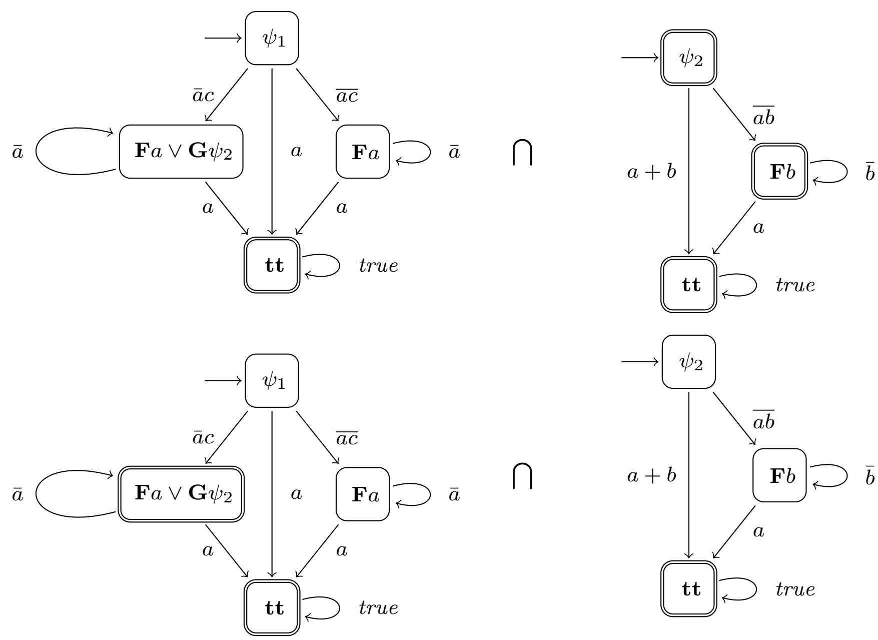


Fig. 7: Intersections with the same structure equivalent to $\mathcal{M}(\psi_1, \emptyset)$ and $\mathcal{M}(\psi_1, \{\psi_2\}) \cap \mathcal{M}(\phi_2)$.

Given a word w, we say that $\mathbf{G}\psi \in \mathbb{G}(\varphi)$ is *eventually true* in w if $w \models \mathbf{FG}\psi$. We denote the set of eventually true $\mathbf{G}$-subformulae of φ by $\mathcal{G}_w(\varphi)$.

A set of $\mathcal{G} \subseteq \mathbb{G}(\varphi)$ is *closed* for w if $\mathcal{G} \models_P af_{\mathbf{G}}(\psi, w_{ij})$ holds for almost all $i \in \mathbb{N}$, almost all $j \geq i$, and for every $\mathbf{G}\psi \in \mathcal{G}$.

The following lemma shows that eventually true $\mathbf{G}$-subformulae can be characterized using the closed sets.

LEMMA 5.5. *Let φ be a formula and let w be a word.*

— *Every set $\mathcal{G} \subseteq \mathbb{G}(\varphi)$ closed for w is included in $\mathcal{G}_w(\varphi)$.*
— *$\mathcal{G}_w(\varphi)$ is closed for w.*

THEOREM 5.6 (LOGICAL CHARACTERIZATION THEOREM III). *For every LTL formula $\mathbf{FG}\varphi$ and every word w: $w \models \mathbf{FG}\varphi$ iff there exists a closed set $\mathcal{G} \subseteq \mathbb{G}(\mathbf{FG}\varphi)$ containing $\mathbf{G}\varphi$.*

PROOF. ($\Rightarrow$): Assume $w \models \mathbf{FG}\varphi$. Then $\varphi \in \mathcal{G}_w(\mathbf{FG}\varphi)$ and by Lemma 5.5(2) $\mathcal{G}_w(\mathbf{FG}\varphi)$ is closed. So we can take $\mathcal{G} = \mathcal{G}_w(\varphi)$.
($\Leftarrow$): Assume some $\mathcal{G} \subseteq \mathbb{G}(\mathbf{FG}\varphi)$ containing $\mathbf{G}\varphi$ is closed. By Lemma 5.5(1) we have $\mathbf{G}\varphi \in \mathcal{G}_w(\mathbf{FG}\varphi)$, and so, by the definition of $\mathcal{G}_w(\mathbf{FG}\varphi)$, we get $w \models \mathbf{FG}\varphi$. $\square$

Let us see that the theorem indeed generalizes Theorem 3.1. If φ is a $\mathbf{G}$-free formula, then $\mathbb{G}(\mathbf{FG}\varphi) = \{\mathbf{G}\varphi\}$. So the only possible choice for $\mathcal{G}$ is $\mathcal{G} = \{\mathbf{G}\varphi\}$ and the only

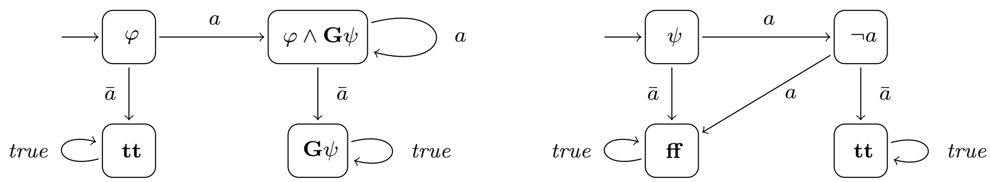


Fig. 8: Transition systems of the Mojmir automata for $\varphi = (\mathbf{G}\psi)\mathbf{U}\neg a$ and for $\psi = a \wedge \mathbf{X}\neg a$.

possible ψ is $\psi = \varphi$. Further, we have

$$\begin{aligned}
&\mathcal{G} \models_P \mathit{af}_{\mathbf{G}}(\psi, w_{ij}) \\
\text{iff } &\mathbf{G}\psi \models_P \mathit{af}_{\mathbf{G}}(\psi, w_{ij}) \\
\text{iff } &\emptyset \models_P \mathit{af}_{\mathbf{G}}(\psi, w_{ij}) && (\mathbf{G}\psi \text{ does not occur in } \mathit{af}_{\mathbf{G}}(\psi, w_{ij})) \\
\text{iff } &\mathit{af}_{\mathbf{G}}(\psi, w_{ij}) \equiv_P \mathbf{tt} \\
\text{iff } &\mathit{af}(\psi, w_{ij}) \equiv_P \mathbf{tt} && (\mathit{af}(\psi, w_{ij}) = \mathit{af}_{\mathbf{G}}(\psi, w_{ij} \text{ since } \varphi \text{ is } \mathbf{G}\text{-free})
\end{aligned}$$

So for a $\mathbf{G}$-free formula φ the theorem states that $w \models \mathbf{FG}\varphi$ iff $\mathit{af}(\varphi, w_{ij}) \equiv_P \mathbf{tt}$ for almost every $i \in \mathbb{N}$ and almost every $j \geq i$.

Let us construct a Mojmir automaton for $\mathbf{FG}\varphi$ from Theorem 5.6. The key is the following simple fact:

$$\begin{gathered}
\mathit{af}_{\mathbf{G}}(\varphi, w_{ij}) \equiv_P \mathbf{tt} \text{ holds for almost every } i \in \mathbb{N} \text{ and almost every } j \geq i \qquad (*) \\
\text{iff} \\
\text{for almost every } i \in \mathbb{N} \text{ there exists } j \geq i \text{ such that } \mathit{af}_{\mathbf{G}}(\varphi, w_{ij}) \equiv_P \mathbf{tt} \qquad (**)
\end{gathered}$$

For the proof, notice first that (*) implies (**); for the other direction recall that if $\mathit{af}_{\mathbf{G}}(\varphi, w_{ij}) \equiv_P \mathbf{tt}$ then $\mathit{af}_{\mathbf{G}}(\varphi, w_{ij'}) \equiv_P \mathbf{tt}$ for every $j' \geq j$.

Now, we observe that (**) has the form of the acceptance condition of a Mojmir automaton. Intuitively, we can reshape it into "for every token $i \in \mathbb{N}$ there exists a time $j \in \mathbb{N}$ such that $\mathit{af}_{\mathbf{G}}(\varphi, w_{ij}) \equiv_P \mathbf{tt}$". So we define:

Definition 5.7. Let φ be a formula and let $\mathcal{G} \subseteq \mathbb{G}(\varphi)$. The Mojmir automaton of φ with respect to $\mathcal{G}$ is $\mathcal{M}(\varphi, \mathcal{G}) = (\mathit{Reach}_{\mathbf{G}}(\varphi), \varphi, \mathit{af}_{\mathbf{G}}, F_{\mathcal{G}})$, where $F_{\mathcal{G}}$ is the set of formulae $\psi \in \mathit{Reach}_{\mathbf{G}}(\varphi)$ such that $\mathcal{G} \models_P \psi$.

As we announced earlier, only the set of accepting states of $\mathcal{M}(\varphi, \mathcal{G})$ depends on $\mathcal{G}$. The following lemma, proved in the Appendix, shows that $\mathcal{M}(\varphi, \mathcal{G})$ is indeed a Mojmir automaton, i.e., that states reachable from accepting states are also accepting.

LEMMA 5.8. *Let φ be a formula and let $\mathcal{G} \subseteq \mathbb{G}(\varphi)$. For every $\psi \in \mathit{Reach}_{\mathbf{G}}(\varphi)$ and every $\nu \in 2^{Ap}$, if $\mathcal{G} \models_P \psi$ then $\mathcal{G} \models_P \mathit{af}_{\mathbf{G}}(\psi, \nu)$.*

Example 5.9. Let $\varphi = (\mathbf{G}\psi)\mathbf{U}\neg a$, where $\psi = a \wedge \mathbf{X}\neg a$. We have $\mathbb{G}(\varphi) = \{\mathbf{G}\psi\}$, and so two automata $\mathcal{M}(\varphi, \emptyset)$ and $\mathcal{M}(\varphi, \{\mathbf{G}\psi\})$, whose common transition system is shown in Figure 8. We have one single automaton $\mathcal{M}(\psi, \emptyset)$, shown on the right of the figure. A formula ψ' is an accepting state of $\mathcal{M}(\psi, \emptyset)$ if $\mathbf{tt} \models_p \psi'$; and so the only accepting state of this automaton is $\mathbf{tt}$. The same holds for $\mathcal{M}(\varphi, \emptyset)$. On the other hand, ψ' is an accepting state of $\mathcal{M}(\varphi, \{\mathbf{G}\psi\})$ if $\mathbf{G}\psi \models \psi'$, and so both $\mathbf{G}\psi$ and $\mathbf{tt}$ are accepting states.

As a corollary of Lemma 5.5 and Definition 5.7 we obtain:

COROLLARY 5.10. *Let φ be a formula, w a word, and $\mathcal{G} \subseteq \mathbb{G}(\varphi)$.*

—*If for every $\mathbf{G}\psi \in \mathcal{G}$ we have $w \in \mathsf{L}(\mathcal{M}(\psi, \mathcal{G}))$, then for every $\mathbf{G}\psi \in \mathcal{G}$ we have $w \models \mathbf{FG}\psi$.*

— *If for every $\mathbf{G}\psi \in \mathcal{G}$ we have $w \models \mathbf{FG}\psi$, then for every $\mathbf{G}\psi \in \mathcal{G}_w(\varphi)$ we have $w \in \mathsf{L}(\mathcal{M}(\psi, \mathcal{G}_w(\varphi)))$.*

Moreover, as a particular case:

THEOREM 5.11. *Let $\mathbf{FG}\varphi$ be a formula and let w be a word. Then $w \models \mathbf{FG}\varphi$ iff there is $\mathcal{G} \subseteq \mathbb{G}(\mathbf{FG}\varphi)$ containing $\mathbf{G}\varphi$ such that $w \in \mathsf{L}(\mathcal{M}(\psi, \mathcal{G}))$ for every $\mathbf{G}\psi \in \mathcal{G}$.*

5.3. The Product Automaton

Theorem 5.11 allows us to construct a generalized Rabin automaton for an arbitrary $\mathbf{FG}$-formula $\mathbf{FG}\varphi$.

Definition 5.12. Let $\varphi = \mathbf{FG}\chi$ be a $\mathbf{FG}$-formula, and let $\mathbb{G}(\varphi)$ be the set of $\mathbf{G}$-subformulae of φ. For every formula $\mathbf{G}\psi \in \mathbb{G}(\varphi)$, let $\mathcal{R}(\psi, \mathcal{G}) = (Q_\psi, q_{0\psi}, \delta_\psi, Acc_\psi^{\mathcal{G}})$ be the Rabin automaton obtained by applying Definition 4.19 to the Mojmir automaton $\mathcal{M}(\psi, \mathcal{G})$. (Recall that Q_ψ, $q_{0\psi}$, and δ_ψ do not depend on $\mathcal{G}$.)

We define the generalized Rabin automaton automaton $\mathcal{R}(\varphi)$ as

$$\mathcal{R}(\varphi) = \left(\prod_{\mathbf{G}\psi \in \mathbb{G}(\varphi)} Q_\psi,\ 2^{Ap},\ \prod_{\mathbf{G}\psi \in \mathbb{G}(\varphi)} q_{0\psi},\ \prod_{\mathbf{G}\psi \in \mathbb{G}(\varphi)} \delta_\psi,\ \ Acc \right)$$

where the accepting condition Acc, which expresses "some $\mathcal{G} \subseteq \mathbb{G}(\varphi)$ containing $\mathbf{G}\psi$ is closed", is given by

$$Acc := \bigvee_{\{\mathcal{G} \subseteq \mathbb{G}(\varphi) \mid \mathbf{G}\chi \in \mathcal{G}\}} \ \bigwedge_{\mathbf{G}\psi \in \mathcal{G}} Acc_\psi^{\mathcal{G}}$$

Since each $Acc_\psi^{\mathcal{G}}$ is a Rabin condition, Acc is a generalized Rabin condition. $\mathcal{R}(\varphi)$ can be transformed into an equivalent Rabin automaton using the construction of Section 2.3.1. Notice however that, as shown in [Chatterjee et al. 2013], for many applications it is better to keep the generalized Rabin condition.

THEOREM 5.13. *Let φ be a $\mathbf{FG}$-formula and let w be a word. Then $w \models \varphi$ iff $w \in L(\mathcal{R}(\varphi))$.*

PROOF. Assume $\varphi = \mathbf{FG}\chi$. By the definition of its accepting condition, $\mathcal{R}(\varphi)$ accepts a word w iff there is a set $\mathcal{G} \subseteq \mathbb{G}(\varphi)$ containing $\mathbf{G}\chi$ such that $\mathcal{R}(\psi, \mathcal{G})$ accepts w for every $\mathbf{G}\psi \in \mathcal{G}$. By Theorem 4.22, this is the case iff $\mathcal{M}(\psi, \mathcal{G})$ accepts w for every $\mathbf{G}\psi \in \mathcal{G}$. By Theorem 5.11 this is the case iff $w \models \varphi$. $\square$

6. DRAS FOR ARBITRARY FORMULAE

In order to explain the last step of our procedure, let $Ap = \{a, b, c\}$ be a set of atomic propositions, and consider the formula $\varphi = b \vee \mathbf{XG}\psi$ over Ap, where $\psi = a \vee \mathbf{X}(b\mathbf{U}c)$. Following the ideas of the previous section, we try to construct an automaton for φ as the union of

(i) an automaton $\mathcal{M}(\varphi, \emptyset)$ accepting all words satisfying φ but not $\mathbf{FG}\psi$ (plus possibly other words satisfying φ), and
(ii) an automaton $\mathcal{M}(\varphi, \{\psi\})$ accepting all words satisfying φ and $\mathbf{FG}\psi$ (plus possibly other words satisfying φ).

By the same argument we gave in the previous section, for $\mathcal{M}(\varphi, \emptyset)$ we can take a Mojmir automaton accepting the words satisfying $\varphi[\mathbf{G}\psi/\mathbf{ff}] = b \vee \mathbf{X}\mathbf{ff} \equiv b$. We now try to construct $\mathcal{M}(\varphi, \{\psi\})$ as the intersection of two Mojmir automata: $\mathcal{M}(\psi)$, which

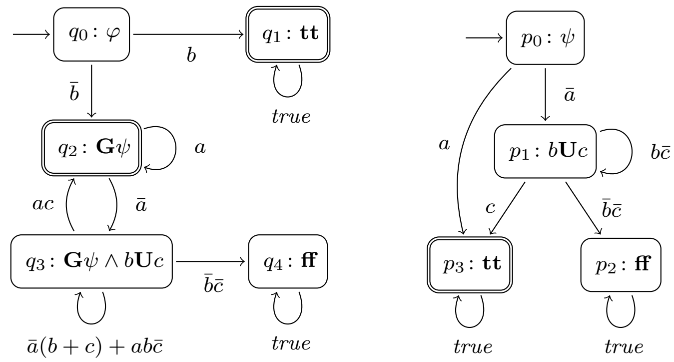


Fig. 9: Automata $\mathcal{B}(\varphi)$ and $\mathcal{M}(\psi)$ for $\varphi = b \vee \mathbf{XG}\psi$ and $\psi = a \vee \mathbf{X}(b\mathbf{U}c)$.

guarantees that the intersection only accepts words satisfying $\mathbf{FG}\psi$, and an automaton that accepts the words satisfying φ under the assumption that they satisfy $\mathbf{FG}\psi$. The automaton $\mathcal{M}(\psi)$ is shown on the right of Figure 9. But what can the other automaton be?

We consider the following idea. As transition system of the automaton we take $\mathcal{T}(\varphi)$ (see Definition 3.2). This guarantees that the state reached after reading a finite word w_{0i} is $af(\varphi, w_{0i})$. Further, we choose a co-Büchi accepting condition stating that the states $\varphi' \in Reach(\varphi)$ that occur infinitely often in the run satisfy $\mathbf{G}\psi \models_P \varphi'$. Then, an accepting run on a word w gets eventually trapped in states satisfying $\mathbf{G}\psi \models_P \varphi'$. So, since $\mathbf{G}\psi$ eventually holds, for a sufficiently large i we have $w_i \models af(\varphi, w_{0i})$, and so by Proposition 2.10 we have $w \models \varphi$.

Unfortunately, while this reasoning is sound, it is not complete. In our example, this idea leads to the automaton $\mathcal{B}(\varphi)$ shown on the left of Figure 9. Since we have $\mathbf{G}\psi \models_P \mathbf{tt}$ and $\mathbf{G}\psi \models_P \mathbf{G}\psi$, the accepting states of $\mathcal{B}(\varphi)$ are q_1 and q_2. Consider the word $w = \bar{a}\bar{b}\bar{c}\,(\bar{a}bc)^\omega$. We have $w \models \varphi$, but the run for w starts at q_0, moves to q_2, and then moves to q_3 and stays there forever. So w is rejected. The point is that neither $\mathcal{G} = \emptyset$ nor $\mathcal{G} = \{\mathbf{G}\psi\}$ satisfy $\mathcal{G} \models_P q_3$.

In the rest of the section we show that this second idea is, however, nearly correct. We construct a correct automaton with the same states and transitions as the one above, but with a modified accepting condition. For this we first interpret this failed attempt in logical terms.

6.1. Logical characterization theorem

Our failed attempt amounts to, given a word w, checking if there is a closed set $\mathcal{G}$ satisfying $\mathcal{G} \models_P af(\varphi, w_{0j})$ for almost every $j \in \mathbb{N}$. The following proposition summarizes our observation that this condition does not characterize the words satisfying φ.

PROPOSITION 6.1. *Let φ be a formula and w a word. If there exists a set $\mathcal{G} \subseteq \mathbb{G}(\varphi)$ such that (1) $\mathcal{G}$ is closed and (2) $\mathcal{G} \models_P af(\varphi, w_{0j})$ for almost every $j \in \mathbb{N}$, then $w \models \varphi$. However, the converse does not hold.*

PROOF. Assume such a $\mathcal{G}$ exists. Since $\mathcal{G}$ is closed, by Lemma 5.5(b) we have $w \models \mathbf{FG}\psi$ for every $\mathbf{G}\psi \in \mathcal{G}$, and so there exists an index $i \in \mathbb{N}$ such that $w_j \models \mathcal{G}$ for every $j \geq i$. By (2), we have $\mathcal{G} \models_P af(\varphi, w_{0j})$ for some $j \geq i$ and hence $w_j \models af(\varphi, w_{0j})$. Finally, by Proposition 2.10, $w \models \varphi$.

The converse does not hold due to the previous example where neither $\mathcal{G} = \emptyset$ nor $\mathcal{G} = \{\mathbf{G}\psi\}$ satisfy $\mathcal{G} \models_P af(\varphi, w_{0i})$. $\square$

In the rest of the section we weaken condition (2) of Proposition 6.1 so that the converse also holds, thus yielding a logical characterization theorem that generalizes Theorem 5.6. More precisely, our goal is to find an adequate formula $\mathcal{F}(\mathcal{G}, w_{0j})$ such that after replacing condition (2) by

(2*) $\mathcal{G} \wedge \mathcal{F}(\mathcal{G}, w_{0j}) \models_P \mathit{af}(\varphi, w_{0j})$ for almost every $j \in \mathbb{N}$.

both Proposition 6.1 and its converse hold. Observe that we replace $\mathcal{G}$ by the stronger formula $\mathcal{G} \wedge \mathcal{F}(\mathcal{G}, w_{0j})$, which makes the propositional implication easier to satisfy.

6.1.1. A first candidate for $\mathcal{F}(\mathcal{G}, w_{0j})$. The formula $\mathcal{F}(\mathcal{G}, w_{0j})$ should satisfy $w_j \models \mathcal{F}(\mathcal{G}, w_{0j})$ for almost every $j \in \mathbb{N}$, because then we can still prove that (1) and (2*) imply $w \models \varphi$ using the same proof as in Proposition 6.1. So we search for a formula satisfying this condition.

Let us examine the closure condition in more detail. Given $\mathbf{G}\psi \in \mathcal{G}$, it states that $\mathcal{G} \models_P \mathit{af}_{\mathbf{G}}(\psi, w_{ij})$ holds for almost all $i \in \mathbb{N}$ and for almost all $j \geq i$. So there is a smallest index i such that $\mathcal{G} \models_P \mathit{af}_{\mathbf{G}}(\psi, w_{ij})$ holds for almost every $j \geq i$. We give it a name, and define a first candidate for $\mathcal{F}(\mathcal{G}, w_{0j})$.

Definition 6.2. Let φ be a formula and let w be a word. Let $\mathcal{G} \subseteq \mathbb{G}(\varphi)$ be closed and let $\mathbf{G}\psi \in \mathcal{G}$. The *threshold* $\mathit{thr}_w(\psi, \mathcal{G})$ of ψ in $\mathcal{G}$ is the smallest index i such that $\mathcal{G} \models_P \mathit{af}_{\mathbf{G}}(\psi, w_{jk})$ holds for every $j \geq i$ and almost all $k \geq j$. Further, we define

$$\mathcal{F}_1(\psi, \mathcal{G}, w_{0j}) = \bigwedge_{i=\mathit{thr}_w(\psi,\mathcal{G})}^{j} \mathit{af}_{\mathbf{G}}(\psi, w_{ij})$$

$$\mathcal{F}_1(\mathcal{G}, w_{0j}) = \bigwedge_{\mathbf{G}\psi \in \mathcal{G}} \mathcal{F}_1(\psi, \mathcal{G}, w_{0j})$$

Recall that $w_{ij} = \epsilon$ if $i \geq j$ by definition. Since $\mathit{af}_{\mathbf{G}}(\psi, \epsilon) = \psi$, we can also define

$$\mathcal{F}_1(\psi, \mathcal{G}, w_{0j}) = \begin{cases} \psi & \text{if } j = 0 \\ \psi \wedge \bigwedge_{i=\mathit{thr}_w(\psi,\mathcal{G})}^{j-1} \mathit{af}_{\mathbf{G}}(\psi, w_{ij}) & \text{if } j > 0 \end{cases}$$

Example 6.3. Consider the formula $\varphi = b \vee \mathbf{XG}\psi$ and $\psi = a \vee \mathbf{X}(b\mathbf{U}c)$ and let $\mathcal{G} = \{\mathbf{G}\psi\}$. For $w = (a\bar{b}\bar{c})^\omega$ we have $\mathit{af}_{\mathbf{G}}(\psi, w_{ij}) = \mathbf{tt}$ for every $0 \leq i < j$. So $\mathcal{G}$ is closed for w. Further we have $\mathit{thr}_w(\psi, \mathcal{G}) = 0$, and so $\mathcal{F}_1(\psi, \mathcal{G}, w_{0j}) = \psi$ for every $j \geq 0$.

For $w = (\bar{a}\bar{b}c)^\omega$ we have $\mathit{af}_{\mathbf{G}}(\psi, w_{i(i+1)}) = b\mathbf{U}c$ for every $i \geq 0$, and $\mathit{af}_{\mathbf{G}}(\psi, w_{ij}) = \mathbf{tt}$ for every $j > i+1 \geq 1$. So $\mathcal{G}$ is closed for w. Further we have $\mathit{thr}_w(\psi, \mathcal{G}) = 0$, and so

$$\mathcal{F}_1(\psi, \mathcal{G}, w_{0j}) = \begin{cases} \psi & \text{if } j = 0 \\ \psi \wedge b\mathbf{U}c & \text{if } j > 0 \end{cases}$$

For $w = \bar{a}bc(ab\bar{c})^\omega$ we have $\mathit{af}_{\mathbf{G}}(\psi, w_{0j}) = b\mathbf{U}c$ for all $j > 0$ and $\mathit{af}_{\mathbf{G}}(\psi, w_{ij}) = \mathbf{tt}$ for all other pairs $j > i$. So $\mathcal{G}$ is closed for w. Further we have $\mathit{thr}_w(\psi, \mathcal{G}) = 1$, because $\mathbf{G}\psi \not\models_P \mathit{af}_{\mathbf{G}}(\psi, w_{0j}) = b\mathbf{U}c$ for all $j > 0$. So $\mathcal{F}_1(\psi, \mathcal{G}, w_{0j}) = \psi$ for every $j \geq 0$.

Let us prove that our first candidate indeed satisfies $w_j \models \mathcal{G} \wedge \mathcal{F}_1(\mathcal{G}, w_{0j})$ for almost every j.

LEMMA 6.4. *Let φ, w, $\mathcal{G}$ and $\mathbf{G}\psi$ as in Definition 6.2. Then $w_j \models \mathcal{G} \wedge \mathcal{F}_1(\mathcal{G}, w_{0j})$ for almost every $j \in \mathbb{N}$*

PROOF. By the semantics of LTL, there exists an index k such that for every $\mathbf{G}\psi \in \mathbb{G}(\varphi)$ either $w_k \models \mathbf{G}\psi$ or $w_k \not\models \mathbf{FG}\psi$ holds. We say that $\mathbb{G}(\varphi)$ *stabilizes* at k. By Theorem 5.6, we further have $w_k \models \mathcal{G}$. So $w_j \models \mathcal{G}$ for every $j \geq k$. We now show that $w_j \models$

$af_{\mathbf{G}}(\psi, w_{ij})$ holds for every $j \geq k$, every $\mathbf{G}\psi \in \mathcal{G}$, and every $i \geq thr_w(\psi, \mathcal{G})$, which concludes the proof. We consider two cases. If $\mathcal{G} \models_P af_{\mathbf{G}}(\psi, w_{ij})$ holds, then the claim follows from $w_k \models \mathcal{G}$. If $\mathcal{G} \not\models_P af_{\mathbf{G}}(\psi, w_{ij})$ then, since $i \geq thr_w(\psi, \mathcal{G})$, there exists $j' > j$ such that $\mathcal{G} \models_P af_{\mathbf{G}}(\psi, w_{ij'})$. Since $j' \geq k$, we have $w_{j'} \models \mathcal{G}$, and so $w_{j'} \models af_{\mathbf{G}}(\psi, w_{ij'}) = af_{\mathbf{G}}(af_{\mathbf{G}}(\psi, w_{ij}), w_{jj'})$. It remains to show that $w_{j'} \models af_{\mathbf{G}}(af_{\mathbf{G}}(\psi, w_{ij}), w_{jj'})$ implies $w_j \models af_{\mathbf{G}}(\psi, w_{ij})$. The proof is by structural induction on the structure of ψ. All cases are identical to those of Proposition 2.10, with the exception of $\psi = \mathbf{G}\psi'$. If $\psi = \mathbf{G}\psi'$ we have $af_{\mathbf{G}}(af_{\mathbf{G}}(\psi, w_{ij}), w_{jj'}) = af_{\mathbf{G}}(\psi, w_{ij}) = \mathbf{G}\psi'$, and so we have to prove that $w_{j'} \models \mathbf{G}\psi'$ implies $w_j \models \mathbf{G}\psi$. Since $j' > j$, this does not seem at first to be the case, but recall that we have $j' > j \geq k$ by hypothesis; since $\mathbb{G}(\varphi)$ stabilizes at k, the two suffixes $w_{j'}$ and w_j satisfy the same formulae of $\mathbb{G}(\varphi)$, and we are done. □

Unfortunately, our first candidate is not good enough for a logical characterization: we can find a formula φ and a word w such that $w \models \varphi$ but no set $\mathcal{G}$ satisfies conditions (1) and (2*).

Example 6.5. Let $\varphi = \mathbf{G}\psi$, where $\psi = \mathbf{X}a \vee \mathbf{G}b$, and $w = a^\omega$. We have $w \models \varphi$. The only non-empty set closed for w is $\mathcal{G} = \{\varphi\}$. However, for this $\mathcal{G}$ condition (2*) does not hold. Indeed, we have

$$\begin{array}{rcll} af_{\mathbf{G}}(\psi, w_{ij}) & = & a \vee \mathbf{G}b & \text{for every } j = i+1 \\ af_{\mathbf{G}}(\psi, w_{ij}) & = & \mathbf{tt} & \text{for every } j > i+1 \\ af(\varphi, w_{0j}) & = & \varphi \wedge a & \text{for every } j > i \geq 1 \end{array}$$

and so (2*) holds only if $\varphi \wedge (a \vee \mathbf{G}b) \models_P \varphi \wedge a$, which is not the case.

6.1.2. A second (and correct) candidate. Observe that, intuitively, if both (1) and (2*) hold, then w satisfies φ even if it does not satisfy any of the formulae of $\overline{\mathcal{G}} = \mathbb{G}(\varphi) \setminus \mathcal{G}$. Using this, we show that Lemma 6.4 still holds if we strengthen $\mathcal{F}_1(\mathcal{G}, w_{0i})$ by, loosely speaking, replacing occurrences of formulae of $\overline{\mathcal{G}}$ by $\mathbf{ff}$. Let us define this formula $\mathcal{F}(\mathcal{G}, w_{0i})$, our final candidate.

Definition 6.6. Let φ, w, $\mathcal{G}$, and $\mathbf{G}\psi$ as in Definition 6.2, and let $\overline{\mathcal{G}} = \mathbb{G}(\varphi) \setminus \mathcal{G}$. We define

$$\begin{aligned} \mathcal{F}(\psi, \mathcal{G}, w_{0j}) &= \mathcal{F}_1(\psi, \mathcal{G}, w_{0j})[\overline{\mathcal{G}}/\mathbf{ff}]_P \\ \mathcal{F}(\mathcal{G}, w_{0i}) &= \bigwedge_{\mathbf{G}\psi \in \mathcal{G}} \mathcal{F}(\psi, \mathcal{G}, w_{0i}) \end{aligned}$$

Example 6.7. In Example 6.5 we have $\mathcal{G} = \{\varphi\}$, hence $\overline{\mathcal{G}} = \{\mathbf{G}b\}$. So $\mathcal{F}(\psi, w_{0i}) = (a \vee \mathbf{G}b)[\{\mathbf{G}b\}/\mathbf{ff}]_P = a$, and now condition (2*) holds.
For the three words of Example 6.3 we have $\mathcal{G} = \mathbb{G}(\varphi)$, and so $\mathcal{F}(\psi, w_{0i}) = \mathcal{F}_1(\psi, w_{0i})$.

LEMMA 6.8. *Let φ, w, $\mathcal{G}$ and $\mathbf{G}\psi$ be as in Definition 6.2. Then $w_j \models \mathcal{G} \wedge \mathcal{F}(\mathcal{G}, w_{0i})$ for almost every $j \in \mathbb{N}$.*

PROOF. The proof is analogous to the proof of Lemma 6.4 and additionally relies on the following equivalence, which can proven by a straightforward induction on ψ.

$$\mathcal{G} \models_P af_{\mathbf{G}}(\psi[\overline{\mathcal{G}}/\mathbf{ff}]_P, w_{0i}) \quad \text{iff} \quad \mathcal{G} \models_P af_{\mathbf{G}}(\psi, w_{0i})[\overline{\mathcal{G}}/\mathbf{ff}]_P$$

□

We show that the new candidate indeed yields a logical characterization theorem.

THEOREM 6.9 (LOGICAL CHARACTERIZATION THEOREM IV). *Let φ be a formula and w a word. Then $w \models \varphi$ iff there exists $\mathcal{G} \subseteq \mathbb{G}(\varphi)$ satisfying (1) $\mathcal{G}$ is closed for w, and (2*) $\mathcal{G} \wedge \mathcal{F}(\mathcal{G}, w_{0i}) \models_P af(\varphi, w_{0i})$ for almost every $i \in \mathbb{N}$.*

PROOF. ($\Leftarrow$) By (1) and (2*), we have $w_j \models \mathcal{G} \wedge \mathcal{F}(\mathcal{G}, w_{0i})$ and $\mathcal{G} \wedge \mathcal{F}(\mathcal{G}, w_{0j}) \models_P af(\varphi, w_{0j})$ for almost every $j \in \mathbb{N}$, which implies $w_j \models af(\varphi, w_{0j})$ for almost every $j \in \mathbb{N}$, and therefore $w \models \varphi$.

($\Rightarrow$) Assume $w \models \varphi$. Let $\mathcal{G}_w$ be the set of all formulae $\mathbf{G}\psi \in \mathbb{G}\varphi$ such that $w \models \mathbf{FG}\psi$. Then by Lemma 5.5, $\mathcal{G}_w$ satisfies (1). For (2*), we first consider the special case in which $thr_w(\psi, \mathcal{G}) = 0$ holds for all $\mathbf{G}\psi \in \mathcal{G}_w$, that is, we not only have $w \models \mathbf{FG}\psi$ but even $w \models \mathbf{G}\psi$ for every $\psi \in \mathcal{G}_w$. Then, by the same reasoning as in the proof of Theorem 5.6, we obtain that $\mathcal{G}_\varphi \models_P af_{\mathbf{G}}(\varphi, w_{0j})$ holds for almost all $j \in \mathbb{N}$. So, after unfolding the definition of $\mathcal{F}(\mathcal{G}_\varphi, w_{0j})$, it remains to show that for almost all $j \in \mathbb{N}$:

$$af_{\mathbf{G}}(\varphi, w_{0j})[\overline{\mathcal{G}_w}/\mathbf{ff}]_P \;\wedge \bigwedge_{\mathbf{G}\psi \in \mathcal{G}_w} \left(\mathbf{G}\psi \wedge \bigwedge_{i=0}^{j} af_{\mathbf{G}}(\psi, w_{ij})[\overline{\mathcal{G}_w}/\mathbf{ff}]_P \right) \models_P af(\varphi, w_{0j})$$

which is proven by a straightforward induction on φ. We consider only two sample cases:

- $\varphi = a$. Since $\varphi = a$ does not have any $\mathbf{G}$-subformulae, the conjunction over all $\mathcal{G}_w$ on the left hand side is simply $\mathbf{tt}$ and also the propositional substitution has no effect. After simplification we obtain $af_{\mathbf{G}}(a, w_{0j}) \models_P af(a, w_{0j})$ which is true.
- $\varphi = \mathbf{G}\varphi'$. In the case $\mathbf{G}\varphi' \notin \mathcal{G}_w$, the left-hand side is propositionally equal to $\mathbf{ff}$ and hence the claim holds. Thus assume $\mathbf{G}\varphi' \in \mathcal{G}_w$. Let us now examine the right-hand side:

$$af(\mathbf{G}\varphi', w_{0i}) = \mathbf{G}\varphi' \wedge \bigwedge_{0=i}^{j} af(\varphi', w_{ij})$$

  Since $\mathbf{G}\varphi' \in \mathcal{G}_w$, the first conjunct is implied by the left-hand side. Let now $af(\varphi', w_{ij})$ be an arbitrary conjunct of the right-hand side. Then there is a matching $af_{\mathbf{G}}(\varphi', w_{ij})[\overline{\mathcal{G}_w}/\mathbf{ff}]_P$ on the left-hand side. We now apply the induction hypothesis on this pair and obtain that $af(\varphi', w_{ij})$ is propositionally entailed by the whole left-hand side. Applying this idea to all conjuncts yields the claim.

Let us now consider the general case. Let k be the maximum of $thr_w(\psi, \mathcal{G})$ for elements of $\mathcal{G}_w$. Then we have $w_k \models \mathbf{G}\psi$ for every $\psi \in \mathcal{G}_w$. Let $\varphi' = af(\varphi, w_{0k})$. By Proposition 2.10, we have $w_k \models \varphi'$, and we can apply the reasoning above to obtain: for almost every $i \in \mathbb{N}$: $\mathcal{G} \wedge \mathcal{F}_{w_k}(\mathcal{G}, w_{0ki}) \models_P af(\varphi', w_{ki})$. Since $\mathcal{F}(\mathcal{G}, w_{0(k+i)})$ contains all conjuncts of $\mathcal{F}_{w_k}(\mathcal{G}, w_{ki})$, after unfolding the definitions we finally obtain $\mathcal{G} \wedge \mathcal{F}(\mathcal{G}, w_{0i}) \models_P af(\varphi, w_{0i})$ for almost every $i \in \mathbb{N}$. $\square$

6.2. From the logical characterization to automata

As in the previous section, we transform the logical characterization into an automaton. For this, we show that $\mathcal{F}(\mathcal{G}, w_{0i})$ is closely related to the ranks at which the automata $\mathcal{M}(\psi, \mathcal{G})$ accept the word w. Loosely speaking, the fact that these automata accept tells us that the formulae of $\mathcal{G}$ eventually hold, and the ranks at which they accept allows us to determine the formula $\mathcal{F}(\mathcal{G}, w_{0i})$ for sufficiently large i. We need a preliminary definition.

Definition 6.10. Let $\mathcal{M}$ be a Mojmir automaton with set of states $Q_{\mathcal{M}}$, and let $sr\colon Q_{\mathcal{M}} \to \mathbb{N}$ be a state-ranking that assigns to each state $q \in Q_{\mathcal{M}}$ a rank $sr(q)$. For every $k \in \mathbb{N}$, we define

$$\mathcal{S}(sr, k) = \{q \in Q_{\mathcal{M}} \mid sr(q) \geq k\}$$

In words: $\mathcal{S}(sr, k)$ is the set of states that have rank at least k in the state-ranking sr.

Example 6.11. For a state-ranking

$$\begin{array}{c} \\ (\end{array}\begin{array}{ccccccc} q_0 & q_1 & q_2 & q_3 & q_4 & q_5 & q_6 \\ \mathbf{2} & \mathbf{1} & \perp & \mathbf{4} & \mathbf{3} & \perp & \perp \end{array}\begin{array}{c} \\ ) \end{array}$$

we have for example $\mathcal{S}(sr, 1) = \{q_0, q_1, q_3, q_4\}$, and $\mathcal{S}(sr, 3) = \{q_3, q_4\}$. For the bottom state of the DRA in Figure 4 (which is a state-ranking of the Mojmir automaton on the left of the figure) we get $\mathcal{S}(sr, 1) = \{a \vee (b\mathbf{U}c), b\mathbf{U}c\}$ and $\mathcal{S}(sr, 2) = \{a \vee (b\mathbf{U}c)\}$.

We can now state the theorem. Recall that the Mojmir automaton $\mathcal{M}(\psi, \mathcal{G})$ was defined in Definition 5.7, and that the states of its corresponding Rabin automaton $\mathcal{R}(\psi, \mathcal{G})$ are state-rankings for the states of the Mojmir $\mathcal{M}(\psi, \mathcal{G})$.

THEOREM 6.12. *Let $\mathcal{G} \subseteq \mathbb{G}(\varphi)$ be closed for w, and let $\mathbf{G}\psi \in \mathcal{G}$. For every $i \geq 0$, let $sr(i)$ be the state of $\mathcal{R}(\psi, \mathcal{G})$ reached after w_{0i} (in other words, $sr(i) = \delta_\psi(q_{0\psi}, w_{0i})$, where δ_ψ is the transition function of $\mathcal{R}(\psi, \mathcal{G})$). Finally, let* r *be the smallest rank at which $\mathcal{R}(\psi, \mathcal{G})$ accepts w. Then*

$$\mathcal{G} \wedge \mathcal{F}(\psi, \mathcal{G}, w_{0i}) \equiv_P \mathcal{G} \wedge \mathcal{S}(sr(i), \mathbf{r}) \quad \textit{for almost every } i \in \mathbb{N}.$$

Before proving the theorem, let us consider an example.

Example 6.13. Figure 10 shows the transition system $\mathcal{T}(\varphi)$, the Mojmir automaton $\mathcal{M}(\psi)$, and the DRA $\mathcal{R}(\psi)$ for the formula $\varphi = b \vee \mathbf{XG}\psi$ with $\psi = a \vee b\mathbf{U}c$ (*cf.* Figure 9). The state $(\mathbf{i}, \mathbf{j})$ of $\mathcal{R}(\psi)$ indicates that ψ has rank $\mathbf{i}$ and $b\mathbf{U}c$ has rank $\mathbf{j}$. We have

$$fail = \{t_3, t_8\} \qquad \begin{array}{ll} merge(\mathbf{1}) = \emptyset & succeed(\mathbf{1}) = \{t_1, t_5, t_7\} \\ merge(\mathbf{2}) = \{t_6\} & succeed(\mathbf{2}) = \{t_4, t_7, t_8\} \end{array}$$

We examine again the three words of Example 6.3.

Let $w = a^\omega$. The run of $\mathcal{R}(\psi)$ on w is t_1^ω, and so $\mathcal{R}(\psi)$ accepts w at rank $\mathbf{1}$. Recall that $\mathcal{F}(\psi, \mathcal{G}, w_{0i}) = \psi$ for every $i \geq 0$. So we have

$$\mathcal{G} \wedge \mathcal{F}(\mathcal{G}, w_{0i}) = \mathbf{G}\psi \wedge \psi \quad \text{for almost every } i \in \mathbb{N}$$

Further, since $\mathcal{S}(sr(i), 1)$ is the conjunction of the states q of $\mathcal{M}(\psi)$ such that $sr(w_{0i}, q) \geq \mathbf{1}$, and the run of $\mathcal{R}(\psi)$ on w only visits $(\mathbf{1}, \perp)$, we have $sr(i) = (\mathbf{1}, \perp)$ for every $i \geq 0$, and so $\mathcal{S}(sr(i), 1) = q_1 = \psi$. We get

$$\mathcal{G} \wedge \mathcal{S}(sr(i), 1) = \mathbf{G}\psi \wedge \psi \quad \text{for almost every } i \in \mathbb{N}$$

which is indeed propositionally equivalent to $\mathcal{G} \wedge \mathcal{F}(\mathcal{G}, w_{0i})$.

Let now $w = c^\omega$. The run of $\mathcal{R}(\psi)$ on w is $t_2 t_5^\omega$, and so $\mathcal{R}(\psi)$ accepts w at rank $\mathbf{1}$. But now we have $\mathcal{F}(\psi, \mathcal{G}, w_{0i}) \equiv_P \psi \wedge (b\mathbf{U}c)$ for every $i \geq 2$, and so

$$\mathcal{G} \wedge \mathcal{F}(\mathcal{G}, w_{0i}) = \mathbf{G}\psi \wedge \psi \wedge (b\mathbf{U}c) \quad \text{for almost every } i \in \mathbb{N}$$

Since the run of $\mathcal{R}(\psi)$ on w gets trapped in state $(\mathbf{2}, \mathbf{1})$, we have $\mathcal{S}(sr(i), 1) = \psi \wedge b\mathbf{U}c$ for almost every $i \geq 2$, and so

$$\mathcal{G} \wedge \mathcal{S}(sr(i), 1) = \mathbf{G}\psi \wedge \psi \wedge (b\mathbf{U}c) \quad \text{for almost every } i \in \mathbb{N}$$

Finally, let $w = \bar{a}bc\ ab\bar{c}^\omega$. The run of $\mathcal{R}(\psi)$ on w is $t_2 t_4^\omega$, and so $\mathcal{R}(\psi)$ accepts w at rank $\mathbf{2}$ and not at rank $\mathbf{1}$. We have $\mathcal{F}(\psi, \mathcal{G}, w_{0i}) = \psi$ for every $i \geq 1$, and so

$$\mathcal{G} \wedge \mathcal{F}(\mathcal{G}, w_{0i}) = \mathbf{G}\psi \wedge \psi \quad \text{for almost every } i \in \mathbb{N}$$

Further, since the run of $\mathcal{R}(\psi)$ on w gets trapped in state $(\mathbf{2}, \mathbf{1})$, we have $\mathcal{S}(sr(i), 2) = \psi$ for almost every $i \geq 0$, and so

$$\mathcal{G} \wedge \mathcal{S}(sr(i), 2) = \mathbf{G}\psi \wedge \psi \quad \text{for almost every } i \in \mathbb{N}$$

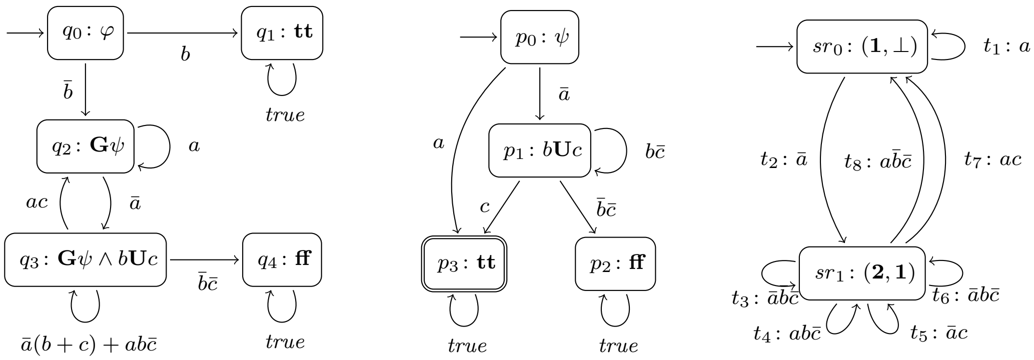


Fig. 10: Transition system $\mathcal{T}(\varphi)$ and automata $\mathcal{M}(\psi)$, and $\mathcal{R}(\psi)$ for $\varphi = b \vee \mathbf{XG}\psi$ and $\psi = a \vee \mathbf{X}(b\mathbf{U}c)$.

Before proving the theorem we have a closer look at the succeeding tokens of a Mojmir automaton. Assume that a Mojmir automaton accepts a word, and we are given the rank at which the word is accepted. The following lemma (proved in the Appendix) shows that from some moment on whether a token succeeds or not depends only on its birthdate, its current rank, and its current state. Most importantly, all young enough tokens will succeed.

LEMMA 6.14. *Let $\mathcal{M}(\psi, \mathcal{G})$ be the Mojmir automaton for a formula ψ. Assume $\mathcal{M}(\psi, \mathcal{G})$ accepts a word w at the smallest accepting rank* $\mathbf{r}$. *For almost every $t \in \mathbb{N}$ and for every token τ of the run of $\mathcal{M}(\psi, \mathcal{G})$ on w, the token succeeds iff*

(*1*) $\tau > t$*, or*
(*2*) $sr_w(t, run_w(\tau, t)) \geq \mathbf{r}$*, or*
(*3*) $run_w(\tau, t) \in F$*.*

The proof of the Theorem is based on the crucial insight that each $af_{\mathbf{G}}(\psi, w_{\tau t})$ precisely corresponds to the state that token τ occupies at time t.

PROOF OF THEOREM 6.12. Consider the run of $\mathcal{M}(\psi, \mathcal{G})$ on the word w. Let t be large enough so that

— every token τ succeeds iff one of the three conditions of Lemma 6.14 holds, and
— all tokens $\tau < thr_w(\psi, \mathcal{G})$ that succeed have already reached the set of accepting states of $\mathcal{M}(\psi, \mathcal{G})$.

Let $m \geq t$. We prove $\mathcal{G} \wedge \mathcal{F}(\psi, \mathcal{G}, w_{0m}) \equiv_P \mathcal{G} \wedge \mathcal{S}(sr(m), \mathbf{r})$.

($\Rightarrow$): $\mathcal{G} \wedge \mathcal{F}(\psi, \mathcal{G}, w_{0m}) \models_P \mathcal{G} \wedge \mathcal{S}(sr(m), \mathbf{r})$.
By definition we have $\mathcal{S}(sr(m), r) = \{q \in Q_{\mathcal{M}(\psi,\mathcal{G})} \mid sr_w(m, q) \geq \mathbf{r}\}$, and so it suffices to show that $\mathcal{G} \models_P q$ or $\mathcal{F}(\psi, \mathcal{G}, w_{0m}) \models_P q$ holds for every $q \in \mathcal{S}(sr(m), \mathbf{r})$. Assume $\mathcal{G} \not\models_P q$. We prove $\mathcal{F}(\psi, \mathcal{G}, w_{0m}) \models_P q$.

We position ourselves at time m: when we talk about the rank or the state of a token we mean its rank or state at time m. Since $sr_w(m, q) \geq \mathbf{r}$, in particular the state q is ranked, and so every token on state q has rank $sr_w(m, q)$. Let τ be any of these tokens. By our choice of t, and since $t \leq m$, all tokens with rank greater than or equal to $\mathbf{r}$ succeed. So τ succeeds. Moreover, since $\mathcal{G} \not\models_P q$, the state q is not an accepting state of $\mathcal{M}(\psi, \mathcal{G})$, and so τ has not succeeded yet. So τ will eventually reach the accepting states of $\mathcal{M}(\psi, \mathcal{G})$ in the future. Moreover, by our choice of t, all tokens born before

$thr_w(\psi, \mathcal{G})$ have already reached the accepting states. So we have $\tau \geq thr_w(\psi, \mathcal{G})$, and so, by the definition of $\mathcal{F}(\psi, \mathcal{G}, w_{0m})$, we get $\mathcal{F}(\psi, \mathcal{G}, w_{0m}) \models_P af_{\mathbf{G}}(\psi, w_{\tau m})$ (notice that $\tau < m$ because we assume that token τ was already born at time t). By the definition of the transition system of $\mathcal{M}(\psi, \mathcal{G})$, the equivalence class $[af_{\mathbf{G}}(\psi, w_{\tau m})]_P$ is precisely the state of $\mathcal{M}(\psi, \mathcal{G})$ reached by token τ at time m, that is, $q = [af_{\mathbf{G}}(\psi, w_{\tau m})]_P$. So $\mathcal{F}(\psi, \mathcal{G}, w_{0m}) \models_P q$.

($\Leftarrow$): $\mathcal{G} \wedge \mathcal{S}(sr(m), \mathbf{r}) \models_P \mathcal{G} \wedge \mathcal{F}(\psi, \mathcal{G}, w_{0m})$.

By the definition of $\mathcal{F}$ it suffices to show that the left-hand-side implies $af_{\mathbf{G}}(\psi, w_{im})$ for every $thr_w(\psi, \mathcal{G}) \leq i \leq m$. Without loss of generality we assume $\mathcal{G} \not\models af_{\mathbf{G}}(\psi, w_{im})$. Consider the token created at time i. Since it is created after time $thr_w(\psi, \mathcal{G})$, it will eventually reach the accepting states by the definition of the threshold and succeed. Furthermore, since $i \leq m$, one of the three conditions of Lemma 6.14 with $t = m$ and $\tau = i$ holds. Since i cannot satisfy conditions (1) or (3) ($\mathcal{G} \not\models af_{\mathbf{G}}(\psi, w_{im})$), it must satisfy condition (2). So the rank of the state $run_w(i, m)$ at time m is at least $\mathbf{r}$, and so it belongs to $\mathcal{S}(sr(m), \mathbf{r})$. But the state $run_w(i, m)$ is the state reached by token i at time m, and so it is equal to $[af_{\mathbf{G}}(\psi, w_{im})]_P$. So $\mathcal{G} \wedge \mathcal{S}(sr(m), \mathbf{r}) \models_P af_{\mathbf{G}}(\psi, w_{im})$. $\square$

6.3. The automaton $\mathcal{A}(\varphi)$: Informal definition

Let us first recall the structure of the DGRA $\mathcal{R}(\mathbf{FG}\psi)$ for a $\mathbf{FG}$-formula. It is the union of DGRAs $\mathcal{R}(\mathcal{G})$, one for each subset $\mathcal{G} \subseteq \mathbb{G}(\varphi)$ containing $\mathbf{G}\psi$. Given a set $\mathcal{G} = \{\mathbf{G}\psi_1, \ldots, \mathbf{G}\psi_n\}$ of $\mathbf{G}$-subformulae, $\mathcal{R}(\mathcal{G})$ accepts all words w satisfying φ and $\mathbf{FG}\psi_1, \ldots, \mathbf{FG}\psi_n$. It is defined as the intersection of the DRAs $\mathcal{R}(\psi_1, \mathcal{G}), \ldots, \mathcal{R}(\psi_n, \mathcal{G})$, which have all the same transition systems (i.e., the same states, transitions, and initial state), but differ on their accepting conditions. Recall that each $\mathcal{R}(\psi_i, \mathcal{G})$ can accept at different ranks (as many as the number of accepting pairs in $\mathcal{R}(\psi_i, \mathcal{G})$).

Given an arbitrary formula φ, we also define its DGRA $\mathcal{A}(\varphi)$ as a union of DGRAs. However, the union now contains an element $\mathcal{R}(\mathcal{G}, \vec{\mathbf{r}})$ for every set $\mathcal{G} = \{\mathbf{G}\psi_1, \ldots, \mathbf{G}\psi_n\} \subseteq \mathbb{G}(\varphi)$, **and** for each possible vector $\vec{\mathbf{r}} = (\mathbf{r}_1, \ldots, \mathbf{r}_n)$ of accepting ranks of $\mathcal{R}(\psi_1, \mathcal{G}), \ldots, \mathcal{R}(\psi_n, \mathcal{G})$. For example, if $n = 2$ and $\mathcal{R}(\psi_1, \mathcal{G})$ and $\mathcal{R}(\psi_2, \mathcal{G})$ have 3 and 2 accepting pairs, respectively, then instead of one single DGRA $\mathcal{R}(\mathcal{G})$ we have six DGRAs $\mathcal{R}(\mathcal{G}, (\mathbf{1}, \mathbf{1})), \ldots, \mathcal{R}(\mathcal{G}, (\mathbf{3}, \mathbf{2}))$.

The transition system of $\mathcal{R}(\mathcal{G}, \vec{\mathbf{r}})$ is the product of the transition system $\mathcal{T}(\varphi)$ and the transition system of $\mathcal{R}(\mathcal{G})$. Recall that $\mathcal{T}(\varphi)$ has $Reach(\varphi)$ as set of states, and af as transition function. Since, in turn, the transition system of $\mathcal{R}(\mathcal{G})$ is the product of the transition systems of $\mathcal{R}(\psi_1, \mathcal{G}), \ldots, \mathcal{R}(\psi_n, \mathcal{G})$, a state of $\mathcal{R}(\mathcal{G})$ is a tuple $(sr_1, \ldots, sr_n)$, where sr_i is a state-ranking of the formulae of $Reach_{\mathbf{G}}(\psi_i)$, and a state of $\mathcal{R}(\mathcal{G}, \vec{\mathbf{r}})$ is a tuple $(\chi, sr_1, \ldots, sr_n))$, where $\chi \in Reach(\varphi)$.

It remains to describe the accepting condition of $\mathcal{R}(\mathcal{G}, \vec{\mathbf{r}})$. We say that $\mathcal{R}(\mathcal{G})$ accepts at rank-vector $\vec{\mathbf{r}} = (\mathbf{r}_1, \ldots, \mathbf{r}_n)$ if each $\mathcal{R}(\psi_i, \mathcal{G})$ accepts at rank $\mathbf{r}_i$. Our goal is to design the accepting condition as a conjunction of two conditions guaranteeing that:

(i) $\mathcal{G}$ is closed (which implies that $\mathcal{R}(\mathcal{G})$ accepts), and moreover $\mathcal{R}(\mathcal{G})$ accepts at rank-vector $\vec{\mathbf{r}}$, and

(ii) $\mathcal{R}(\mathcal{G}, \vec{\mathbf{r}})$ eventually stays within states $(\chi, sr_1, \ldots, sr_n)$ satisfying

$$\mathcal{G} \wedge \mathcal{S}(sr_1, r_1) \wedge \cdots \wedge \mathcal{S}(sr_n, r_n) \models_P \chi$$

In particular, (i) checks condition (1) of the logical characterization theorem, Theorem 6.9. Let us now see that (ii) checks condition (2*). By definition, the formula χ reached after reading a finite prefix w_{0i} of a word w is the formula $af(\varphi, w_{0i})$. Therefore, (ii) is equivalent to

$$\mathcal{G} \wedge \mathcal{S}(sr_1(w_{0i}, r_1)) \wedge \cdots \wedge \mathcal{S}(sr_n(w_{0i}, r_n)) \models_P af(\varphi, w_{0i}) \quad \text{for almost every } i \in \mathbb{N}$$

which by Theorem 6.12 is equivalent to

$$\mathcal{G} \wedge \mathcal{F}(\psi, \mathcal{G}, w_{0i}) \models_P af(\varphi, w_{0i}) \quad \text{for almost every } i \in \mathbb{N}$$

and so to condition (2*) of the logical characterization theorem.

We still have to express (i) and (ii) as generalized Rabin conditions. Condition (i) is a conjunction of conditions expressing that $\mathcal{R}(\psi_i, \mathcal{G})$ accepts at rank $\mathbf{r}_i$ for every $1 \leq i \leq n$. Let $P_1 \vee \cdots \vee P_n$ be the accepting condition of $\mathcal{R}(\psi_i, \mathcal{G})$. Recall that $\mathcal{R}(\psi_i, \mathcal{G})$ accepts at rank $\mathbf{r}_i$ if it accepts with the Rabin pair P_{r_i}. $P_{r_i} \vee P_{r_i+1} \vee \cdots \vee P_n$. Further, condition (ii) is a co-Büchi condition, which is a special case of a Rabin condition. So the conjunction of (i) and (ii) is a conjunction of Rabin conditions, and so a generalized Rabin condition.

Observe that condition (i) can be decomposed into a conjunction of conditions, each of which concerns only one of the automata in the product. On the contrary, condition (ii) involves all components of the product, and cannot be decomposed.

As in the case of **FG**-formulae, it remains to deal with the state-explosion problem. Recall that, when we introduced the automata $\mathcal{R}(\psi, \mathcal{G})$, we observed that they can all be constructed so that they all have the same transition system, and therefore the intersection $\mathcal{R}(\mathcal{G})$ has the same transition system as well. Since $\mathcal{R}(\mathcal{G})$ and $\mathcal{R}(\mathcal{G}, \vec{\mathbf{r}})$ have the same transition system, the same happens now.

6.4. The automaton $\mathcal{A}(\varphi)$: Formal definition

We conclude the section by giving a precise definition of the automaton $\mathcal{A}(\varphi)$.

Definition 6.15. Let φ be an arbitrary formula, and let $\mathbb{G}(\varphi) = \{\mathbf{G}\psi_1, \ldots, \mathbf{G}\psi_n\}$ be the set of **G**-subformulae of φ. For every formula $\mathbf{G}\psi_i \in \mathbb{G}(\varphi)$, let $\mathcal{R}(\psi_i, \mathcal{G}) = (Q_i, 2^{Ap}, q_{0i}, \delta_i, Acc_i^{\mathcal{G}})$ be the DRA obtained by applying Definition 4.19 to the Mojmir automaton $\mathcal{M}(\psi_i, \mathcal{G})$. Recall that a state of Q_i is a state-ranking of the states of $\mathcal{M}(\psi_i, \mathcal{G})$. We use sr_i to denote a state-ranking of Q_i.

The DGRA $\mathcal{A}(\varphi) = (Q_\varphi, 2^{Ap}, q_{0\varphi}, \delta_\varphi, Acc_\varphi)$ is defined as follows:

- $Q_\varphi = Reach(\varphi) \times Q_1 \times \cdots \times Q_n$.
- $q_{0\varphi} = (\varphi, q_{01}, \ldots, q_{0n})$.
- $\delta_\varphi((\chi, sr_1, \ldots, sr_n), a) = (af(\chi, a), \delta_1(sr_1, a), \ldots, \delta_n(sr_n, a))$.
- Acc_φ is a disjunction containing a disjunct $Acc_{\vec{\mathbf{r}}}^{\mathcal{G}}$ for each pair $(\mathcal{G}, \vec{\mathbf{r}})$, where $\mathcal{G} \subseteq \mathbb{G}(\varphi)$ and $\vec{\mathbf{r}}$ is a mapping assigning to each $\psi \in \mathcal{G}$ a rank, i.e., a number between **1** and the number of Rabin pairs of $\mathcal{R}(\psi, \mathcal{G})$; each $Acc_{\vec{\mathbf{r}}}^{\mathcal{G}}$ is then of the form

$$M_{\vec{\mathbf{r}}}^{\mathcal{G}} \wedge \bigwedge_{\mathbf{G}\psi \in \mathcal{G}} Acc_{\vec{\mathbf{r}}}^{\mathcal{G}}(\psi)$$

where $Acc_{\vec{\mathbf{r}}}^{\mathcal{G}}(\psi)$ denotes the Rabin pair of $\mathcal{R}(\psi, \mathcal{G})$ with number $\vec{\mathbf{r}}(\psi)$, and $M_{\vec{\mathbf{r}}}^{\mathcal{G}}$ says that transitions taken infinitely often by $\mathcal{A}(\varphi)$ must lead into the following set:

$$\{(\chi, sr_1, \ldots, sr_n) \in Q_\varphi \mid \mathcal{G} \wedge \bigwedge_{\mathbf{G}\psi_i \in \mathcal{G}} \mathcal{S}(sr_i, \vec{\mathbf{r}}(\psi_i)) \models_P \chi\} \; .$$

Observe that $M_{\mathbf{r}}^{\mathcal{G}}$ can be phrased as a co-Büchi condition on transitions. Therefore, the whole condition Acc_φ is a generalized Rabin condition.

Example 6.16. Recall Example 6.13 illustrated in Figure 10. The states of $\mathcal{A}(\varphi)$ are pairs (χ, sr), where χ is a state of $\mathcal{T}(\varphi)$ (on the left of the figure) and sr is a state of $\mathcal{R}(\psi)$ (on the right). Rank vectors have only one component, and so we write $\mathbf{r}$ instead of $\vec{\mathbf{r}}$. Since $\mathcal{R}(\psi)$ has two Rabin pairs, we have $\mathbf{r} = \mathbf{1}$ or $\mathbf{r} = \mathbf{2}$.

For $\mathcal{G} = \emptyset$ we have $Acc^{\emptyset}_{\mathbf{r}} = M^{\emptyset}_{\mathbf{r}}$, and, independently of $\mathbf{r}$, condition $M^{\emptyset}_{\mathbf{r}}$ requests that $\mathcal{A}(\varphi)$ eventually stays in states (χ, sr) satisfying $\mathbf{tt} \models \chi$, and so in the set $\{ (q_1, sr_0), (q_1, sr_1) \}$.

For $\mathcal{G} = \{\psi\}$ we have $Acc^{\psi}_{\mathbf{r}} = M^{\psi}_{\mathbf{r}} \wedge Acc^{\psi}_{\mathbf{r}}$. Condition $Acc^{\psi}_{\mathbf{r}}$ states that $\mathcal{R}(\psi)$ must accept using the pair $P(\mathbf{r})$. Let us now examine $M^{\psi}_{\mathbf{1}}$ and $M^{\psi}_{\mathbf{2}}$, starting with the latter.

$M^{\psi}_{\mathbf{2}}$ requests that $\mathcal{A}(\varphi)$ eventually stays in states (χ, sr) satisfying $\mathbf{G}\psi \wedge \mathcal{S}(sr, \mathbf{2}) \models_P \chi$. Since $\mathcal{S}(sr_0, \mathbf{2}) = \mathbf{tt}$ and $\mathcal{S}(sr_1, \mathbf{2}) = \psi$ (see the Mojmir automaton in the middle of the figure), $\mathcal{A}(\varphi)$ must eventually stays in states (χ, sr_0) satisfying $\mathbf{G}\psi \models_P \chi$ or states (χ, sr_1) satisfying $\mathbf{G}\psi \wedge \psi \models_P \chi$, and so in the states $\{q_1, q_2\} \times \{sr_0, sr_1\}$.

$M^{\psi}_{\mathbf{1}}$ requests that $\mathcal{A}(\varphi)$ eventually stays in states (χ, sr) satisfying $\mathbf{G}\psi \wedge \mathcal{S}(sr, \mathbf{1}) \models_P \chi$. Since $\mathcal{S}(sr_1, \mathbf{1}) = \{p_0, p_1\} = \psi \wedge (b\mathbf{U}c)$, we have $\mathbf{G}\psi \wedge \mathcal{S}(sr_1, \mathbf{1}) \models_P \chi$ for $\chi = \mathbf{G}\psi \wedge (b\mathbf{U}c)$, the formula of state q_3. So $\mathcal{A}(\varphi)$ must eventually stay in the set $(\{q_1, q_2\} \times \{sr_0, sr_1\}) \cup \{(q_3, sr_1)\}$.

We now proceed to our final result.

THEOREM 6.17. *For any LTL formula φ, $\mathsf{L}(\mathcal{A}(\varphi)) = \mathsf{L}(\varphi)$.*

PROOF. ($\Rightarrow$) By Theorem 6.9 we only need to prove that if $\mathcal{A}(\varphi)$ accepts w with $\mathcal{G} \subseteq \mathbb{G}(\varphi)$ and rank vector $\vec{\mathbf{r}}$, then (1) $\mathcal{G}$ is closed for w and (2*) $\mathcal{G} \wedge \mathcal{F}(\mathcal{G}, w_{0i}) \models_P af(\varphi, w_{0i})$ holds for almost every $i \in \mathbb{N}$. By construction $\mathcal{A}(\varphi)$ only accepts with closed $\mathcal{G}$'s and thus (1) holds. For (2*) we observe that $\mathcal{A}(\varphi)$ also accepts w with the rank vector $\vec{\mathbf{r}}^*$ that maps every element of $\mathcal{G}$ to the smallest accepting rank for w. So we obtain from $M^{\mathcal{G}}_{\vec{\mathbf{r}}^*}$:

$$\mathcal{G} \wedge \bigwedge_{\mathbf{G}\psi_i \in \mathcal{G}} \mathcal{S}(sr_i, \vec{\mathbf{r}}^*(\psi_i)) \models_P af(\varphi, w_{0i})$$

By Theorem 6.12 we have $\mathcal{G} \wedge \mathcal{S}(sr_i, \vec{\mathbf{r}}) \models_P \mathcal{G} \wedge \mathcal{F}(\mathcal{G}, w_{0i})$ for almost every $i \in \mathbb{N}$, and so we conclude that property (2*) holds.
($\Leftarrow$): Let $\mathcal{G} \subseteq \mathbb{G}(\varphi)$ be a set satisfying the conditions of Theorem 6.9, and let $\vec{\mathbf{r}}$ be the rank vector that maps every element of $\mathcal{G}$ to the corresponding smallest accepting rank. We now prove that $\mathcal{A}(\varphi)$ accepts w with $Acc^{\mathcal{G}}_{\vec{\mathbf{r}}}$. Since $\mathcal{G}$ is closed for w, the Rabin pairs $Acc^{\mathcal{G}}_{\vec{\mathbf{r}}}(\psi)$ are accepting for all $\mathbf{G}\psi \in \mathcal{G}$. Hence it remains to show that also $M^{\mathcal{G}}_{\vec{\mathbf{r}}}$ is accepting. For this we use the other direction of Theorem 6.12, i.e., that $\mathcal{G} \wedge \mathcal{F}(\mathcal{G}, w_{0i}) \models_P \mathcal{G} \wedge \mathcal{S}(sr_i, \vec{\mathbf{r}})$ for almost every $i \in \mathbb{N}$. $\square$

7. OPTIMIZATIONS

The construction described in the previous sections can be optimized in a number of ways. In fact, we have already presented an important optimization: the fact that sink states are not ranked. It is possible to handle sinks just as any other state, but this leads to much larger Rabin automata. Even the toy examples of the paper would then be too large to be drawn.

We describe further optimizations reducing the number of the states or the size of the accepting condition of the automata. Some, but not all, have been mechanically proven.

7.1. Reducing the state space

The first obvious reduction is to construct only the states reachable from the initial states. Further, we merge equivalent states in several ways. Interestingly, this happens based on the formulae that label the states, and not on the graph structure of the automaton, as is the case for, e.g., simulation-based reductions.

(1) Unfolding formulae.
Let the one-step unfolding $\mathfrak{Unf}$ of a formula be inductively defined by the following

rules:

$$\begin{array}{rlrl}
\mathfrak{Unf}(a) & = a & \mathfrak{Unf}(\mathbf{X}\varphi) & = \mathbf{X}\varphi \\
\mathfrak{Unf}(\neg a) & = \neg a & \mathfrak{Unf}(\mathbf{F}\varphi) & = \mathfrak{Unf}(\varphi) \vee \mathbf{F}\varphi \\
\mathfrak{Unf}(\varphi \wedge \psi) & = \mathfrak{Unf}(\varphi) \wedge \mathfrak{Unf}(\psi) & \mathfrak{Unf}(\mathbf{G}\varphi) & = \mathfrak{Unf}(\varphi) \wedge \mathbf{G}\varphi \\
\mathfrak{Unf}(\varphi \vee \psi) & = \mathfrak{Unf}(\varphi) \vee \mathfrak{Unf}(\psi) & \mathfrak{Unf}(\varphi\mathbf{U}\psi) & = \mathfrak{Unf}(\psi) \vee (\mathfrak{Unf}(\varphi) \wedge (\varphi\mathbf{U}\psi))
\end{array}$$

The optimization consists of always using unfolded formulae as states. Note that $af(\mathfrak{Unf}(\varphi), \cdot) = af(\varphi, \cdot)$ since af is $\mathfrak{Unf}$ followed by plugging in the valuation read. Therefore, the only change in the transition system of the automaton is to merge states labelled by $\varphi_1 \neq \varphi_2$ such that $\mathfrak{Unf}(\varphi_1) = \mathfrak{Unf}(\varphi_2)$. This is an efficient way to under-approximate LTL equivalence by propositional equivalence, which is also easier to check (PSPACE vs. NP), e.g. using BDDs. As a simple example, the optimized automaton for $\mathbf{FG}a$ has one state, instead of two states, as illustrated in Fig. 11.

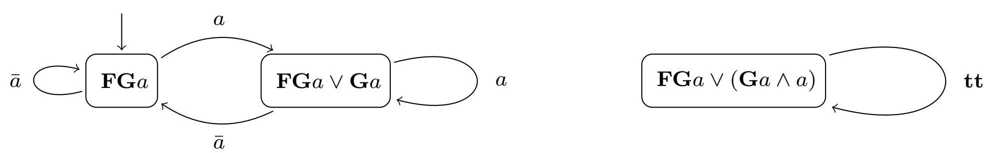


Fig. 11: Original and optimized co-Büchi automata for $\mathbf{FG}a$

(2) Different initial states for DRAs.
Since no finite prefix influences acceptance of Rabin automata for $\mathbf{FG}$-formulae, introducing arbitrary initial states for them does not change the accepted language. Therefore, instead of using "transient" states, which cannot be visited once left, we try to use states that are reachable even after reading some prefixes. For instance, consider the formula $\mathbf{GF}((a \wedge \mathbf{XX}a) \vee (\neg a \wedge \mathbf{XX}\neg a))$. The automaton corresponds to a buffer keeping track of several last letters read. Without the optimization, we start with an empty buffer; such an initial state of the Rabin automaton has only a single token in the initial state of the Mojmir automaton. Then we read a letter and move to a buffer filled with either a or $\bar{a}$. In the next step, we move to a buffer with two letters and from that point switch only among the two-letter buffers. The total size is thus $2^0 + 2^1 + 2^2 = 7$. However, if we start with an already full buffer (filled with whatever letters), the acceptance is not affected, but the reachable state space is only of size $2^2 = 4$.

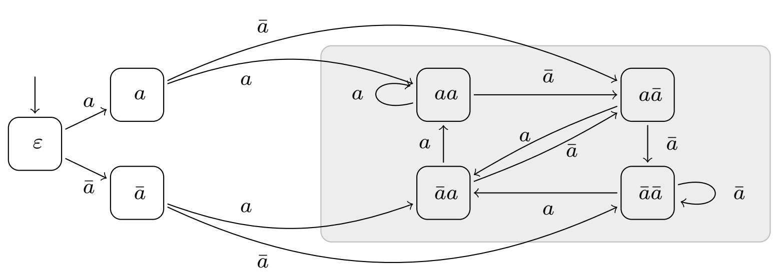


Fig. 12: A co-Büchi automaton for $\mathbf{GF}((a \wedge \mathbf{XX}a) \vee (\neg a \wedge \mathbf{XX}\neg a))$ and the optimized automaton inside the grey area (with an arbitrary initial state)

(3) Irrelevant DRAs.
Recall that a state of our parallel composition is an array of formulae, one corresponding to the current state of the co-Büchi automaton, and the others to the states of the DRAs. We say that a DRA is irrelevant at a state if its corresponding $\mathbf{G}$-formula either does not appear inside the current formula of the co-Büchi automaton, or it only appears in conjunction with another formula without any occurrence of $\mathbf{G}$. For instance, after reading a in $a \wedge \mathbf{F}b \wedge \mathbf{FG}c \vee \neg a \wedge \mathbf{FG}d$, the co-Büchi automaton reaches the state $\mathbf{F}b \wedge \mathbf{FG}c$, where the DRA for the formula d is irrelevant. Consider now $\mathbf{F}b \wedge \mathbf{FG}c$. At this state the DRA for c is irrelevant, due to the conjunction with $\mathbf{F}b$. Intuitively, the co-Büchi automaton waits for a b, and only after that it is important to monitor the satisfaction of $\mathbf{FG}c$. Indeed, postponing the monitoring by finite time does not affect acceptance, similarly to the previous optimization. Moreover, if b never holds, then it is unnecessary to check satisfaction of $\mathbf{FG}c$.

7.2. Reducing the acceptance condition

All disjuncts of a generalized Rabin condition are of the form $(F, \bigwedge_{k\in K} I_k)$, which we call a *generalized pair*. We consider a transition-based condition and denote the set of all transitions by T. We remove generalized pairs that cannot be satisfied, as well those whose satisfaction implies satisfaction of another pair. In order to detect such pairs, we first simplify them. The optimizations are performed to exhaustion in the following order.

(1) Remove every generalized pair $(F, \mathcal{I})$ such that $F = T$.
Such pairs never accept, since the whole T cannot be avoided.
(2) Replace every generalized pair $(F, \mathcal{I} \wedge I)$ such that $I \cup F = T$ by $(F, \mathcal{I})$.
If F is visited only finitely often then $T \setminus F \subseteq I$ is visited infinitely often.
(3) Replace every generalized pair $(F, \bigwedge_{k\in K} I_k)$ by $(F, \bigwedge_{k\in K} I_k \setminus F)$.
Visiting F infinitely often excludes acceptance.
(4) Remove every generalized pair $(F, \mathcal{I} \wedge \emptyset)$.
The empty set cannot be visited (infinitely often).
(5) Replace every generalized pair $(F, \mathcal{I} \wedge I \wedge J)$ such that $I \subseteq J$ by $(F, \mathcal{I} \wedge I)$.
If I is visited infinitely often then so is J.
(6) Remove every generalized pair $(F, \bigwedge_{k\in K} I_k)$ for which there exists $(F', \bigwedge_{k'\in K'} I'_{k'})$ such that $F' \subseteq F$, and for each $k' \in K'$ there is $k \in K$ such that $I_k \subseteq I'_{k'}$.
A run accepted by the unprimed pair is also accepted by the primed pair.

For example, consider the formula $(\mathbf{GF}(a \wedge \mathbf{X}b) \vee \mathbf{FG}(b \vee \mathbf{X}\neg a)) \wedge (\mathbf{GF}(b \wedge \mathbf{X}c) \vee \mathbf{FG}(!c \vee \mathbf{X}a)) \wedge (\mathbf{GF}(b \wedge \mathbf{XX}a) \vee \mathbf{FG}(\neg c \vee X\neg b))$. We start with 4568 pairs and after each phase we are left with 4052, 3715, 1997, 131, 122, and finally 12 pairs, respectively.

8. COMPLEXITY BOUNDS

Before discussing the implementation of our construction and experimental results we briefly discuss the worst-case complexity and compare it with that of Safra-based constructions.

Recall first that the smallest DRA for an LTL formula of length n may have $\Theta(2^{2^n})$ states. This is the case even for the fragment of LTL containing only conjunction, disjunction and the $\mathbf{F}$ operator [Alur and Torre 2004, Theorem 3.8]. Indeed, this paper shows that all DRAs for the formula

$$\mathbf{F} \bigwedge_{i=1}^{n} (a_i \vee \mathbf{F}b_i)$$

have a double exponential number of states (in n).

Our construction almost matches this lower bound. Given a formula φ, the set of states of our co-Büchi automaton is $Reach(\varphi)$, and the set of states of our DRAs are $Reach_{\mathbf{G}}(\psi)$ for subformulae ψ of φ. By Lemma 2.9, if φ has n proper subformulae then both $Reach(\varphi)$ and $Reach_{\mathbf{G}}(\psi)$ have size at most 2^{2^n}, and so the number of states of their product is at most

$$(2^{2^n})^n = 2^{2^{n+\log n}}$$

Further, each pair corresponds to DRAs accepting at one of less than 2^n ranks, or not accepting at all. Altogether, there are at most $(2^n)^n = 2^{n^2}$ pairs.

The upper bound in the number of states essentially coincides with those for LTL-to-DRA translations based on Safra's construction. These translations first transform φ into a NBA of size $\mathcal{O}(2^n)$, and then apply Safra's construction, which runs in $m^{\mathcal{O}(m)}$ time and space, for an automaton of size m [Safra 1988]. The overall complexity is thus

$$2^{n \cdot \mathcal{O}(2^n)} = 2^{\mathcal{O}(2^{n+\log n})}$$

The number of Rabin pairs of Safra-based translations is at most $\mathcal{O}(m) = \mathcal{O}(2^n)$. We leave the question whether a modification of our construction can match this bound (or whether our 2^{n^2} upper bound is tight) open.

Consider now the LTL fragment with syntax

$$\lambda ::= \lambda \wedge \lambda \mid \lambda \vee \lambda \mid \mathbf{GF}a \mid \mathbf{FG}a$$

where $a \in Ap$ (or more generally, a is a boolean combination of Ap). This fragment contains many interesting fairness formulae, like those of the family $\bigwedge_{i=1}^{n} (\mathbf{GF}\ a_i \rightarrow \mathbf{GF}b_i)$. Our construction yields DGRAs with only one single state, provided we use the unfolding optimization presented in Section 7. Indeed, a simple induction shows that for every formula φ in the fragment and for every $\nu \in 2^{Ap}$, we have $\mathfrak{Unf}(af(\varphi,\nu)) \equiv_P \mathfrak{Unf}(\varphi)$. Therefore, if we take $\mathfrak{Unf}(\varphi)$ as the initial state, the co-Büchi automaton only has one reachable state. By a similar argument, replacing af by $af_{\mathbf{G}}$, the Mojmir automaton $\mathcal{M}(\psi)$ for a $\mathbf{G}$-subformula $\mathbf{G}\psi$ also has one single state, and the same holds for its corresponding Rabin automaton. Since every component of the parallel composition only has one state, the same holds for the parallel composition itself. Note that without the unfolding optimization the co-Büchi automaton for $\bigwedge_{i=1}^{n} \mathbf{FG}a_i$ would have 2^n states.

9. IMPLEMENTATION AND EXPERIMENTAL RESULTS

9.1. Implementation

The construction is implemented in a tool `Rabinizer 3`, which was reported on in [Komárková and Křetínský 2014]. It is written in Java and uses JavaBDD to work with formulae as Boolean functions. Furthermore, in order to optimize the construction time, we have implemented a new version `3.1` of the tool.[7] It uses BDDs also for labelling edges in automata and explores the state space in this more symbolic way rather than examining successors for each valuation separately.

The implementation allows to choose between the mechanically proved construction and switching on any subset of the described optimizations. Furthermore, apart from producing the resulting transition-based generalized Rabin automata, it can also convert the result to state-based automata as well as degeneralize them into Rabin automata.

[7]http://www7.in.tum.de/~kretinsk/rabinizer3.html

Finally, there is a choice of output formats: dot format, useful for graphical representation, e.g. by `dotty` or `Graphviz`; and the HOA (Hanoi omega-automata) format, the new standard [Babiak et al. 2015], nowadays implemented by other translators as well as `PRISM`. This allows for linking `Rabinizer` to `PRISM`, resulting in a significantly faster probabilistic LTL model checker, see [Chatterjee et al. 2013; Komárková and Křetínský 2014].

9.2. Experimental results

We compare the performance of the following tools and methods:

- (L*) `ltl2dstar` [Klein 2005] implements and optimizes [Klein and Baier 2007] Safra's construction [Safra 1988]. It uses `LTL2BA` [Gastin and Oddoux 2001] to obtain the non-deterministic Büchi automata (NBA) first. Other translators to NBA may also be used, such as `Spot` [Duret-Lutz 2013] or `LTL3BA` [Babiak et al. 2012] and in some cases may yield better results (see [Blahoudek et al. 2013] for comparison thereof), but `LTL2BA` is recommended by `ltl2dstar` and is used this way in `PRISM` [Kwiatkowska et al. 2011].
- (R1/2) `Rabinizer` [Gaiser et al. 2012] and `Rabinizer 2` [Křetínský and Ledesma-Garza 2013] implement a direct construction based on [Křetínský and Esparza 2012] for fragments $\text{LTL}(\mathbf{F},\mathbf{G})$ and $\text{LTL}_{\setminus\mathbf{GU}}$[8], respectively. The latter tool is applied here only on formulae not in $\text{LTL}(\mathbf{F},\mathbf{G})$.
- (L3) `LTL3DRA` [Babiak et al. 2013] implements a construction via alternating automata, which is "inspired by [Křetínský and Esparza 2012]" (quoted from [Babiak et al. 2013]) and performs several optimizations.
- (R3) `Rabinizer 3.1` performs our new construction. Unless specified otherwise we employ the previously described optimizations. Notice that we produce a state space with a logical structure, which permits many further optimizations; for instance, one could incorporate the suspension optimization of `LTL3BA` [Babiak et al. 2013].

For L* and R1/2 we produce DRAs (although `Rabinizer 2` can also produce DGRAs) with state-based acceptance conditions. For L3 and R3 we produce DGRAs with transition-based acceptance conditions (tDGRAs), which can be directly used for probabilistic model checking without any blow-up [Chatterjee et al. 2013]. Inapplicability of a tool to a formula is denoted in tables by $-$. All automata in this section were constructed within a few seconds, with the exception of the larger automata generated by `ltl2dstar`: it took several minutes for automata over ten thousand states and hours for hundreds of thousands of states. The automaton for $\bigwedge_{i=1}^{3}(\mathbf{GF}a_i \rightarrow \mathbf{GF}b_i)$ took even more than a day and "?" denotes a time-out after one day.

Table I shows formulae of the $\text{LTL}(\mathbf{F},\mathbf{G})$ fragment. The upper part comes from BEEM (BEnchmarks for Explicit Model checkers)[Pelánek 2007], the lower one from [Somenzi and Bloem 2000] on which `ltl2dstar` was originally tested [Klein and Baier 2006]. There are overlaps between the two sets. Note that the formula $(\mathbf{FF}a \wedge \mathbf{G}\neg a) \vee (\mathbf{GG}\neg a \wedge \mathbf{F}a)$ is a contradiction. All the formulae were used already in [Křetínský and Esparza 2012; Babiak et al. 2013]. Although more general, our method usually achieves the same results as the optimized `LTL3DRA`, outperforming the first two approaches.

Table II shows formulae of $\text{LTL}_{\setminus\mathbf{GU}}$ used in [Křetínský and Ledesma-Garza 2013]. The first part comes mostly from the same sources and [Etessami and Holzmann 2000]. The second part is considered in [Křetínský and Ledesma-Garza 2013] in order to demonstrate the difficulties of the standard approach to handle

[8] $\text{LTL}_{\setminus\mathbf{GU}}$ was introduced in [Křetínský and Ledesma-Garza 2013] and disallows occurrences of $\mathbf{U}$ in the scope of $\mathbf{G}$.

Table I: Experimental results on LTL($\mathbf{F}$,$\mathbf{G}$)-fragment

Formula	L*	R1	L3	R3
$\mathbf{G}(a \vee \mathbf{F}b)$	4	4	2	2
$\mathbf{FG}a \vee \mathbf{FG}b \vee \mathbf{GF}c$	8	8	1	1
$\mathbf{F}(a \vee b)$	2	2	2	2
$\mathbf{GF}(a \vee b)$	2	2	1	1
$\mathbf{G}(a \vee b \vee c)$	3	2	2	2
$\mathbf{G}(a \vee \mathbf{F}(b \vee c))$	4	4	2	2
$\mathbf{F}a \vee \mathbf{G}b$	4	3	3	3
$\mathbf{G}(a \vee \mathbf{F}(b \wedge c))$	4	4	2	2
$(\mathbf{FG}a \vee \mathbf{GF}b)$	4	4	1	1
$\mathbf{GF}(a \vee b) \wedge \mathbf{GF}(b \vee c)$	7	3	1	1
$(\mathbf{FF}a \wedge \mathbf{G}\neg a) \vee (\mathbf{GG}\neg a \wedge \mathbf{F}a)$	1	0	1	2
$(\mathbf{GF}a) \wedge \mathbf{FG}b$	3	3	1	1
$(\mathbf{GF}a \wedge \mathbf{FG}b) \vee (\mathbf{FG}\neg a \wedge \mathbf{GF}\neg b)$	5	4	1	1
$\mathbf{FG}a \wedge \mathbf{GF}a$	2	2	1	1
$\mathbf{G}(\mathbf{F}a \wedge \mathbf{F}b)$	5	3	1	3
$\mathbf{F}a \wedge \mathbf{F}\neg a$	4	4	4	4
$(\mathbf{G}(b \vee \mathbf{GF}a) \wedge \mathbf{G}(c \vee \mathbf{GF}\neg a)) \vee \mathbf{G}b \vee \mathbf{G}c$	13	18	4	4
$(\mathbf{G}(b \vee \mathbf{FG}a) \wedge \mathbf{G}(c \vee \mathbf{FG}\neg a)) \vee \mathbf{G}b \vee \mathbf{G}c$	14	6	4	4
$(\mathbf{F}(b \wedge \mathbf{FG}a) \vee \mathbf{F}(c \wedge \mathbf{FG}\neg a)) \wedge \mathbf{F}b \wedge \mathbf{F}c$	7	5	4	4
$(\mathbf{F}(b \wedge \mathbf{GF}a) \vee \mathbf{F}(c \wedge \mathbf{GF}\neg a)) \wedge \mathbf{F}b \wedge \mathbf{F}c$	7	5	4	4

(1) many $\mathbf{X}$ operators inside the scope of other temporal operators, especially $\mathbf{U}$, where the DRAs are already quite complex, and
(2) conjunctions of liveness properties where the efficiency of generalized Rabin acceptance condition may be fully exploited.

Table II: Experimental results on $\mathrm{LTL}_{\setminus \mathbf{GU}}$ -fragment

Formula	L*	R2	L3	R3
$(\mathbf{F}p)\mathbf{U}(\mathbf{G}q)$	4	3	2	2
$(\mathbf{G}p)\mathbf{U}q$	5	5	5	5
$(p \vee q)\mathbf{U}p \vee \mathbf{G}q$	4	3	3	3
$\mathbf{G}(!p \vee \mathbf{F}q) \wedge ((\mathbf{X}p)\mathbf{U}q \vee \mathbf{X}((!p \vee !q)\mathbf{U}!p \vee \mathbf{G}(!p \vee !q)))$	19	8	–	5
$\mathbf{G}(q \vee \mathbf{XG}p) \wedge \mathbf{G}(r \vee \mathbf{XG}!p)$	5	14	4	4
$(\mathbf{X}(\mathbf{G}r \vee r\mathbf{U}(r \wedge s\mathbf{U}p)))\mathbf{U}(\mathbf{G}r \vee r\mathbf{U}(r \wedge s))$	18	9	8	8
$p\mathbf{U}(q \wedge \mathbf{X}(r \wedge (\mathbf{F}(s \wedge \mathbf{X}(\mathbf{F}(t \wedge \mathbf{X}(\mathbf{F}(u \wedge \mathbf{XF}v))))))))$	9	13	13	13
$(\mathbf{GF}(a \wedge \mathbf{XX}b) \vee \mathbf{FG}b) \wedge \mathbf{FG}(c \vee (\mathbf{X}a \wedge \mathbf{XX}b))$	353	73	–	12
$\mathbf{GF}(\mathbf{XXX}a \wedge \mathbf{XXXX}b) \wedge \mathbf{GF}(b \vee \mathbf{X}c) \wedge \mathbf{GF}(c \wedge \mathbf{XX}a)$	2 127	169	–	16
$(\mathbf{GF}a \vee \mathbf{FG}b) \wedge (\mathbf{GF}c \vee \mathbf{FG}(d \vee \mathbf{X}e))$	18 176	80	–	2
$(\mathbf{GF}(a \wedge \mathbf{XX}c) \vee \mathbf{FG}b) \wedge (\mathbf{GF}c \vee \mathbf{FG}(d \vee \mathbf{X}a \wedge \mathbf{XX}b))$	?	142	–	12
$a\mathbf{U}b \wedge (\mathbf{GF}a \vee \mathbf{FG}b) \wedge (\mathbf{GF}c \vee \mathbf{FG}d) \vee$ $\vee a\mathbf{U}c \wedge (\mathbf{GF}a \vee \mathbf{FG}d) \wedge (\mathbf{GF}c \vee \mathbf{FG}b)$	640 771	210	8	7

Table III contains formulae of the general LTL. The first part contains two randomly picked formulae illustrating the same two phenomena as in the previous table now on general LTL formulae. The second part contains two examples of formulae from a network monitoring project LIBEROUTER[9]. The third part contains five more complex formulae from SPEC PATTERN [Dwyer et al. 1999][10] and express the following "after Q until R" properties:

[9]https://www.liberouter.org/
[10]Spec Patterns: Property Pattern Mappings for LTL. http://patterns.projects.cis.ksu.edu/documentation/patterns/ltl.shtml

φ_{35} : $\mathbf{G}(!q \vee (\mathbf{G}p \vee (!p\mathbf{U}(r \vee (s \wedge !p \wedge \mathbf{X}(!p\mathbf{U}t))))))$
φ_{40} : $\mathbf{G}(!q \vee (((!s \vee r) \vee \mathbf{X}(\mathbf{G}(!t \vee r) \vee !r\mathbf{U}(r \wedge (!t \vee r))))\mathbf{U}(r \vee p) \vee \mathbf{G}((!s \vee \mathbf{XG}!t))))$
φ_{45} : $\mathbf{G}(!q \vee (!s \vee \mathbf{X}(\mathbf{G}!t \vee !r\mathbf{U}(r \wedge !t)) \vee \mathbf{X}(!r\mathbf{U}(r \wedge \mathbf{F}p)))\mathbf{U}(r \vee \mathbf{G}(!s \vee \mathbf{X}(\mathbf{G}!t \vee !r\mathbf{U}(r \wedge !t)) \vee \mathbf{X}(!r\mathbf{U}(t \wedge \mathbf{F}p)))))$
φ_{50} : $\mathbf{G}(!q \vee (!p \vee (!r\mathbf{U}(s \wedge !r \wedge \mathbf{X}(!r\mathbf{U}t))))\mathbf{U}(r \vee \mathbf{G}(!p \vee (s \wedge \mathbf{XF}t))))$
φ_{55} : $\mathbf{G}(!q \vee (!p \vee (!r\mathbf{U}(s \wedge !r \wedge !z \wedge \mathbf{X}((!r \wedge !z)\mathbf{U}t))))\mathbf{U}(r \vee \mathbf{G}(!p \vee (s \wedge !z \wedge \mathbf{X}(!z\mathbf{U}t)))))$

Here we also compare unoptimized and optimized versions of our construction.

Table III: Experimental results on general LTL

Formula	L*	R1/2	L3	R3-unopt.	R3-opt.
$\mathbf{FG}((a \wedge \mathbf{XX}b \wedge \mathbf{GF}b)\mathbf{U}(\mathbf{G}(\mathbf{XX}!c \vee \mathbf{XX}(a \wedge b))))$	2 053	–	–	9	3
$\mathbf{G}(\mathbf{F}!a \wedge \mathbf{F}(b \wedge \mathbf{X}!c) \wedge \mathbf{GF}(a\mathbf{U}d)) \wedge \mathbf{GF}((\mathbf{X}d)\mathbf{U}(b \vee \mathbf{G}c))$	283	–	–	25	7
$\mathbf{G}(((!p1)) \wedge (p2\mathbf{U}((!p2)\mathbf{U}((!p3) \vee p4))))$	7	–	–	6	4
$\mathbf{G}(((p1) \wedge \mathbf{X}!p1) \vee \mathbf{X}(p1\mathbf{U}(((!p2) \wedge p1) \wedge$ $\mathbf{X}(p2 \wedge p1 \wedge (p1\mathbf{U}(((!p2) \wedge p1) \wedge \mathbf{X}(p2 \wedge p1)))))))$	8	–	–	12	9
φ_{35} : 2 cause-1 effect precedence chain	6	–	–	9	6
φ_{40} : 1 cause-2 effect precedence chain	314	–	–	16	16
φ_{45} : 2 stimulus-1 response chain	1450	–	–	81	68
φ_{50} : 1 stimulus-2 response chain	28	–	–	36	21
φ_{55} : 1-2 response chain constrained by a single proposition	28	–	–	36	21

9.3. Advantages and limits of the approach

In this section, we focus on formulae with extremely complex acceptance conditions. This is caused by combinations of “infinitary” behaviour, whose satisfaction does not depend on any finite prefix of the word. A typical example is the “fairness”-fragment given by λ of Section 8. In this case, our DGRA have only one state. While DRA need to remember the last letter read, the transition-based acceptance together with the generalized acceptance condition allow transition-based DGRA not to remember anything. Such formulae are the most difficult for our as well as the traditional determinization approach.

In Table IV, the first two parts were used in [Křetínský and Esparza 2012; Babiak et al. 2013] and the third part in [Babiak et al. 2013]. The first part focuses on properties with fairness-like constraints.

Table IV: Experimental results on “fairness”-fragment given by λ of Section 8

Formula	L*	R1	L3	R3-unopt.	R3-opt.
$(\mathbf{FG}a \vee \mathbf{GF}b)$	4	4	1	4	1
$(\mathbf{FG}a \vee \mathbf{GF}b) \wedge (\mathbf{FG}c \vee \mathbf{GF}d)$	11 324	18	1	16	1
$\bigwedge_{i=1}^{3}(\mathbf{FG}a_i \vee \mathbf{GF}b_i)$	1 304 706	462	1	64	1
$\mathbf{GF}(\mathbf{F}a \vee \mathbf{GF}b \vee \mathbf{FG}(a \vee b))$	14	4	1	3	1
$\mathbf{FG}(\mathbf{F}a \vee \mathbf{GF}b \vee \mathbf{FG}(a \vee b))$	145	4	1	4	1
$\mathbf{FG}(\mathbf{F}a \vee \mathbf{GF}b \vee \mathbf{FG}(a \vee b) \vee \mathbf{FG}b)$	181	4	1	4	1
$(\mathbf{GF}a \vee \mathbf{FG}b)$	4	4	1	4	1
$(\mathbf{GF}a \vee \mathbf{FG}b) \wedge (\mathbf{GF}b \vee \mathbf{FG}c)$	572	11	1	9	1
$(\mathbf{GF}a \vee \mathbf{FG}b) \wedge (\mathbf{GF}b \vee \mathbf{FG}c) \wedge (\mathbf{GF}c \vee \mathbf{FG}d)$	290 046	52	1	17	1
$(\mathbf{GF}a \vee \mathbf{FG}b) \wedge (\mathbf{GF}b \vee \mathbf{FG}c) \wedge (\mathbf{GF}c \vee \mathbf{FG}d) \wedge (\mathbf{GF}d \vee \mathbf{FG}h)$	?	1288	1	33	1

Table V shows it is very beneficial to use the generalized Rabin acceptance. Furthermore, using transition-based acceptance even more states are saved.

Table V: Experimental comparisons of acceptance conditions. We display number of states and acceptance pairs for `ltl2dstar` and `Rabinizer 3` producing different types of automata, all with the same number of pairs. Here $\psi_1 = \mathbf{FG}(((a \wedge \mathbf{XX}b) \wedge \mathbf{GF}b)\mathbf{UG}(\mathbf{XX}!c \vee \mathbf{XX}(a \wedge b)))$ and $\psi_2 = \mathbf{G}(!q \vee (((!s \vee r) \vee \mathbf{X}(\mathbf{G}(!t \vee r) \vee !r\mathbf{U}(r \wedge (!t \vee r))))\mathbf{U}(r \vee p) \vee \mathbf{G}((!s \vee \mathbf{XG}!t))))$, the latter being φ_{40} "1 cause-2 effect precedence chain" of SPEC PATTERNS

Formula	`ltl2dstar`		`Rabinizer 3`			
	DRA states	pairs	DRA st.	DGRA st.	tDGRA st.	pairs
$\mathbf{FG}a \vee \mathbf{GF}b$	4	2	4	4	1	2
$(\mathbf{FG}a \vee \mathbf{GF}b) \wedge (\mathbf{FG}c \vee \mathbf{GF}d)$	11324	8	21	16	1	4
$\bigwedge_{i=1}^{3}(\mathbf{GF}a_i \to \mathbf{GF}b_i)$	1 304 706	10	511	64	1	8
$\bigwedge_{i=1}^{3}(\mathbf{GF}a_i \to \mathbf{GF}a_{i+1})$	153 558	8	58	17	1	8
ψ_1	40	4	4	4	3	1
ψ_2	314	7	21	21	16	4

However, when the the automata are used for model checking, transition-based acceptance does not improve the results so much. Indeed, although state-based DGRA are larger than their transition-based counterpart tDGRA, the respective product is not much larger (often not at all), see Table VI. For instance, consider the case when the only extra information that DGRA carries in states, compared to tDGRA, is the labelling of the last transition taken. Then this information is absorbed in the product, as the system's states carry their labelling anyway. Therefore, in this relatively frequent case for simpler formulae (like the one in Table VI), there is no difference in sizes of products with DGRA and tDGRA.

Table VI: Model checking Pnueli-Zuck mutex protocol with 5 processes (altogether $m = 308\,800$ states) from the benchmark set [Kwiatkowska et al. 2011] for the property that either all processes 1-4 enter the critical section infinitely often, or process 5 asks to enter it only finitely often

	L* DRA	R3 DRA	R3 DGRA	R3 tDGRA
Automaton size (and nr. of pairs)	196 (5)	11 (2)	33 (2)	1 (2)
Product size	13 826 588	1 100 608	308 800	308 800
"Effective" size of automaton = Product size/m	44.78	3.56	1	1

Further, notice that the DGRA in Table VI is larger than the DRA obtained by degeneralization of tDGRA and subsequent transformation to a state-based automaton. However, the product with the DGRA is of the size of the original system, while for DRA it is larger! This demonstrates the superiority of generalized Rabin automata over standard Rabin automata with respect to the product size and thus also computation time, which is superlinear in the size.

Finally, Table VII compares the running times for the discussed fairness-fragment.

10. FORMALIZATION IN ISABELLE

We have mechanically verified the proof of correctness of our construction using the Isabelle theorem prover[11], which provides a rich library of formalised mathematics and convenient support for proof development. A detailed introduction can be found in

[11] https://isabelle.in.tum.de/

Table VII: Running times for constructing an automaton and its acceptance condition for fairness constraints $\bigwedge_{i=1}^{k}(\mathbf{FG}a_i \vee \mathbf{GF}b_i)$ for different k. Times are given in seconds with time-out (blank space) after one hour. The experiments were run on an Intel Core i7 with 8 GB memory. Here we also compare to `Rabinizer 3` of [Komárková and Křetínský 2014], denoted by R3.0, where all transitions are handled separately, as opposed to a symbolic encoding into edges of `Rabinizer 3.1`, denoted by R3.1

k	L*	R1	L3	R3.0	R3.1-unopt.	R3.1-opt.
1	0.15	0.10	0.01	0.04	0.12	0.12
2	4.3	0.19	0.01	0.08	0.29	0.14
3		5.7	0.03	0.38	2.1	0.24
4			0.19	3.8	22	0.54
5			1.9	105	640	1.2
6			25			4.1
7			350			17
8						86
9						670
10						

[Nipkow et al. 2002]. Similar work was pioneered by the CAVA project[12], which already verified a range of automata-theoretic algorithms [Esparza et al. 2013]. In fact some of the theories developed in the context of CAVA project are also reused in our work. The formalization was carried out by one of us, and constituted his Master's thesis. The formal proof can be found at [Sickert 2015], and consists of around 11000 lines.

10.1. Relation between formalisation and the content of this paper

The formalization is split into several "theories". A theory is just a collection of definitions and results, which can reuse results from other theories. Our theories are listed in Table VIII.

Table VIII: Important theories and their content.

`LTL.thy`	Syntax and semantics of LTL.
`af.thy`	The af and $af_{\mathbf{G}}$ functions and their properties.
`Logical_Characterization.thy`	The logical characterization theorems.
`Mojmir.thy`	Mojmir automata.
`Rabin.thy`	(Generalised) Rabin automata.
`Mojmir_Rabin.thy`	Translation from Mojmir to Rabin automata.
`LTL_Rabin.thy`	Translation from LTL to tGDRA.
`LTL_Rabin_Unfold_Opt.thy`	Unfold optimisation of the general translation.

For the main definitions, lemmas, and theorems of this paper, Table IX shows their corresponding name and location in the formalized theories. With the help of this table, interested readers can establish the correspondence between our results and their formal versions. For example, we reproduce here Theorem 5.6 next to the formal version in the mechanized proof:

THEOREM 10.1 (LOGICAL CHARACTERIZATION THEOREM III). *For every LTL formula $\mathbf{FG}\varphi$ and every word w: $w \models \mathbf{FG}\varphi$ iff there exists a closed set $\mathcal{G} \subseteq \mathbb{G}(\mathbf{FG}\varphi)$ containing $\mathbf{G}\varphi$.*

[12] https://cava.in.tum.de/

```
theorem ltl_FG_logical_characterization:
  "w ⊨ FGφ ⟷ (∃𝒢 ⊆ G(FGφ). Gφ ∈ 𝒢 ∧ closed 𝒢 w)"
  (is "?lhs ⟷ ?rhs")
proof
  assume ?lhs
  hence "Gφ ∈ 𝒢_FG(FGφ) w" and "𝒢_FG(FGφ) w ⊆ G(FGφ)"
    unfolding 𝒢_FG_alt_def by auto
  thus ?rhs
    using closed_𝒢_FG by metis
qed (blast intro: closed_FG)
```

Note that there are several differences between the formulation of the theorem in the paper and in the formalized theories.

- — Unbounded variables such as `w` and φ are implicitly universally quantified.
- — The type system automatically deduces the types of `w`, which is an ω-word, and φ, which is an LTL formula, using the signature of the operator $\models$. Thus the type annotations are omitted.
- — Since we cannot use the whole range of mathematical symbols and notation due to technical constraints, alternative notation is used. In this instance $\mathbb{G}$ is replaced by $\mathbf{G}$, and $\mathcal{G}_w(\varphi)$ by $\mathcal{G}_{\mathrm{FG}}\ \varphi\ w$.

The theorem declaration is then followed by the proof body, which is written in the proof language Isar. In every proof step facts are established using the keywords `have`, `hence`, `show`, and `thus`. These claims then have to be proven using a proof method, such as `blast`, `metis`, and `auto`. Furthermore, we can pass additional facts to these methods using parameters such as `intro`, `dest` or via the `using` keyword. All remaining proof goals, in this case that the right hand side implies the left, are proven with the method behind `qed`. A detailed explanation of the language is given in [Wenzel 2007], while the whole specification can be found [Wenzel 2014].

Note that some definitions and claims, like for instance Proposition 2.4 and Theorem 4.1, have no counterpart in the formalisation, as they only illustrate different aspects of the construction, but are not an essential part of it. In the first case, we directly define LTL in negation normal form and do not include a translation method, while in the second case the theorem is just a special case of Theorem 5.6 and thus left out.

10.2. Merits of the Mechanization

While the effort invested in the mechanization of the proof has been very considerable (about 8 person-months of a master student who had taken an introductory course on Isabelle), it has helped to identify several bugs in the construction we presented in [Esparza and Křetínský 2014], the conference paper preceding this one. All but one concerned corner cases that were arguably not very relevant. For example, the translation from a Mojmir to a Rabin automaton was incorrect for the case in which the Mojmir automaton has one single state, which is at the same time an accepting state. However, one bug was more serious. Lemma C of our conference paper was wrong, due to a mistake in the proof. The proof was carried out by induction over the structure of LTL formulae. Since our attempts at mechanizing the proof obviously failed, we repeatedly tried to correct the argument by nesting induction proofs. This process eventually lead to the smallest to us known formula for which the lemma fails: $\mathbf{G}(\mathbf{X}a \vee \mathbf{G}\mathbf{X}b)$. Observe that the formula is already long enough to have a good chance of surviving random testing. Moreover, testing can only be performed with respect to another tool producing DRAs from formulae, which could itself have a bug, and the test requires to check equivalence of deterministic Rabin automata, which is a complicated task. Finally, we

Table IX: Location of definitions, lemmas and theorems.

Def. 2.2	`LTL.thy`	`ltl_semantics`
Def. 2.5	`LTL.thy`	`ltl_prop_entailment`
Def. 2.7	`af.thy`	`af_letter`, `af`
Lem. 2.9	`af.thy`	`af_nested_propos`, `af_simps`, `af_respectfulness`
Prop. 2.10	`af.thy`	`af_ltl_continuation`
Thm. 3.1	`Logical_Characterization.thy`	`ltl_implies_provable`
Lem. 4.9	`Mojmir.thy`	`rank_None_Suc`, `rank_monotonic`
Lem. 4.11	`Mojmir.thy`	`state_rank_step`
Lem. 4.15	`Mojmir.thy`	`token_succeeds_run_merge`, `token_squats_run_merge`
Lem. 4.16	`Mojmir.thy`	`mojmir_accept_iff_token_set_accept`
Lem. 4.17	`Mojmir.thy`	`stable_rank_bounded`
Thm. 4.22	`Mojmir_Rabin.thy`	`mojmir_accept_iff_rabin_accept`
Def. 5.2	`af.thy`	`af_G_letter`, $\mathtt{af}_{\mathtt{G}}$
Lem. 5.5	`Logical_Characterization.thy`	$\mathtt{closed\_}\mathcal{G}_{\mathtt{FG}}$, `closed_FG`
Thm. 5.6	`Logical_Characterization.thy`	`ltl_FG_logical_characterization`
Lem. 5.8	`af.thy`	$\mathtt{af}_{\mathtt{G}}\mathtt{\_sat\_core}$
Thm. 5.13	`LTL_Rabin.thy`	`ltl_FG_to_generalised_rabin_correct`
Lem. 6.8	`Logical_Characterization.thy`	`almost_all_suffixes_model_F`
Thm. 6.9	`Logical_Characterization.thy`	`ltl_logical_characterization`
Thm. 6.12	`LTL_Rabin.thy`	`F_eq_S`
Lem. 6.14	`Mojmir.thy`	`token_accepting_rank`
Thm. 6.17	`LTL_Rabin.thy`	`ltl_to_generalised_rabin_correct`

do not know of any reasonable way of certifying an LTL to DRA translation, that is, of making the tool produce a certificate of correctness that can be checked by independent means.

After these experiences, we consider automata-theoretic constructions used in model-checking tools an area in which mechanized proofs are highly desirable, if not necessary. Many of the constructions are very clever and involved. Moreover, while they often rely on relatively simple intuitions, their correctness proofs often involve detailed case analyses. Since the constructions become part of model-checkers, which for the most part are used to find bugs in other systems, bugs in the construction itself can have a multiplying effect. Finally, as mentioned above, there is no simple way to test the tools, since there is no independent way of checking that the output of the construction is correct.

11. CONCLUSIONS

We have presented the first direct translation from LTL formulae to deterministic Rabin automata able to handle arbitrary formulae. The construction generalizes previous ones for LTL fragments [Křetínský and Esparza 2012; Křetínský and Ledesma-Garza 2013]. A mechanized proof also discovered a bug in the original construction [Esparza and Křetínský 2014]. Given φ, we compute (1) a transition system for φ, automata for each **G**-subformula of φ, and their parallel composition, and (2) the acceptance condition: we first guess a set of **G**-subformulae that are true (this yields the accepting states of automata for **G**-subformulae), and then guess the ranks (this yields the information for a co-Büchi acceptance condition of the whole product).

The compositional approach opens the door to many possible optimizations. Since the automata for **G**-subformulae are typically very small, we can aggressively try to optimize them, knowing that each reduced state in one potentially leads to large savings in the final number of states of the product. So far we have only implemented a few simple optimizations, and we think there is still much room for improvement.

We have conducted a detailed experimental comparison. Our construction outperforms two-step approaches that first translate the formula into a Büchi automaton and then apply Safra's construction. Moreover, despite handling full LTL, it is at least as efficient as previous constructions for fragments. Finally, we produce a (often much smaller) generalized Rabin automaton, which can be directly used for verification, without further translation into a standard Rabin automaton.

REFERENCES

Rajeev Alur and Salvatore La Torre. 2004. Deterministic generators and games for LTL fragments. *ACM Trans. Comput. Log.* 5, 1 (2004), 1–25.

Tomáš Babiak, Thomas Badie, Alexandre Duret-Lutz, Mojmír Křetínský, and Jan Strejček. 2013. Compositional Approach to Suspension and Other Improvements to LTL Translation. In *SPIN*. 81–98.

Tomáš Babiak, František Blahoudek, Alexandre Duret-Lutz, Joachim Klein, Jan Křetínský, David Müller, David Parker, and Jan Strejček. 2015. The Hanoi Omega-Automata Format. In *CAV*. To appear.

Tomáš Babiak, František Blahoudek, Mojmír Křetínský, and Jan Strejček. 2013. Effective Translation of LTL to Deterministic Rabin Automata: Beyond the (F, G)-Fragment. In *ATVA*. 24–39.

Tomáš Babiak, Mojmír Křetínský, Vojtěch Rehák, and Jan Strejček. 2012. LTL to Büchi Automata Translation: Fast and More Deterministic. In *TACAS*. 95–109.

Christel Baier and Joost-Pieter Katoen. 2008. *Principles of model checking*. MIT Press. I–XVII, 1–975 pages.

František Blahoudek, Mojmír Křetínský, and Jan Strejček. 2013. Comparison of LTL to Deterministic Rabin Automata Translators. In *LPAR*. 164–172.

Krishnendu Chatterjee, Andreas Gaiser, and Jan Křetínský. 2013. Automata with Generalized Rabin Pairs for Probabilistic Model Checking and LTL Synthesis. In *CAV*. 559–575.

Costas Courcoubetis and Mihalis Yannakakis. 1988. Verifying Temporal Properties of Finite-State Probabilistic Programs. In *FOCS*. 338–345.

Jean-Michel Couvreur. 1999. On-the-Fly Verification of Linear Temporal Logic. In *World Congress on Formal Methods*. 253–271.

Marco Daniele, Fausto Giunchiglia, and Moshe Y. Vardi. 1999. Improved Automata Generation for Linear Temporal Logic. In *CAV*. 249–260.

Alexandre Duret-Lutz. 2013. Manipulating LTL Formulas Using Spot 1.0. In *ATVA*. 442–445.

Matthew B. Dwyer, George S. Avrunin, and James C. Corbett. 1999. Patterns in Property Specifications for Finite-State Verification. In *ICSE*. 411–420.

Javier Esparza and Jan Křetínský. 2014. From LTL to Deterministic Automata: A Safraless Compositional Approach. In *CAV*. 192–208.

Javier Esparza, Peter Lammich, René Neumann, Tobias Nipkow, Alexander Schimpf, and Jan-Georg Smaus. 2013. A Fully Verified Executable LTL Model Checker. In *CAV*. 463–478.

Kousha Etessami and Gerard J. Holzmann. 2000. Optimizing Büchi Automata. In *CONCUR*. 153–167.

Carsten Fritz. 2003. Constructing Büchi Automata from Linear Temporal Logic Using Simulation Relations for Alternating Büchi Automata. In *CIAA*. 35–48.

Andreas Gaiser, Jan Křetínský, and Javier Esparza. 2012. Rabinizer: Small Deterministic Automata for LTL(F,G). In *ATVA*. 72–76.

Paul Gastin and Denis Oddoux. 2001. Fast LTL to Büchi Automata Translation. In *CAV (LNCS)*, Vol. 2102. Springer, 53–65. Tool accessible at http://www.lsv.ens-cachan.fr/∼gastin/ltl2ba/.

Rob Gerth, Doron Peled, Moshe Y. Vardi, and Pierre Wolper. 1995. Simple on-the-fly automatic verification of linear temporal logic. In *Protocol Specification, Testing and Verification XV, Proceedings of the Fifteenth IFIP WG6.1 International Symposium on Protocol Specification, Testing and Verification*. 3–18.

Barbara Di Giampaolo, Gilles Geeraerts, Jean-François Raskin, and Nathalie Sznajder. 2010. Safraless Procedures for Timed Specifications. In *FORMATS*. 2–22.

Dimitra Giannakopoulou and Flavio Lerda. 2002. From States to Transitions: Improving Translation of LTL Formulae to Büchi Automata. In *FORTE*. 308–326.

Joachim Klein. 2005. *Linear time logic and deterministic omega-automata*. Master's thesis. Rheinische Friedrich-Wilhelms Universität Bonn. The tool *ltl2dstar - LTL to deterministic Streett and Rabin automata* is available at http://www.ltl2dstar.de/.

Joachim Klein and Christel Baier. 2006. Experiments with deterministic ω-automata for formulas of linear temporal logic. *Theor. Comput. Sci.* 363, 2 (2006), 182–195.

Joachim Klein and Christel Baier. 2007. On-the-Fly Stuttering in the Construction of Deterministic ω-Automata. In *CIAA (LNCS)*, Vol. 4783. Springer, 51–61.

Zuzana Komárková and Jan Křetínský. 2014. Rabinizer 3: Safraless Translation of LTL to Small Deterministic Automata. In *ATVA*. 235–241.

Jan Křetínský and Ruslán Ledesma-Garza. 2013. Rabinizer 2: Small Deterministic Automata for LTL$\setminus$GU. In *ATVA*. 446–450.

Orna Kupferman. 2012. Recent Challenges and Ideas in Temporal Synthesis. In *SOFSEM (LNCS)*, Vol. 7147. Springer, 88–98.

Orna Kupferman, Nir Piterman, and Moshe Y. Vardi. 2006. Safraless Compositional Synthesis. In *CAV*. 31–44.

Orna Kupferman and Moshe Y. Vardi. 2005. Safraless Decision Procedures. In *FOCS*. 531–542.

Jan Křetínský and Javier Esparza. 2012. Deterministic Automata for the (F,G)-Fragment of LTL. In *CAV*. 7–22.

Marta Z. Kwiatkowska, Gethin Norman, and David Parker. 2011. PRISM 4.0: Verification of Probabilistic Real-time Systems. In *CAV*. 585–591.

Tobias Nipkow, Lawrence C. Paulson, and Makarius Wenzel. 2002. *Isabelle/HOL: A Proof Assistant for Higher-Order Logic*. Springer.

Radek Pelánek. 2007. BEEM: Benchmarks for Explicit Model Checkers. In *Proc. of SPIN Workshop (LNCS)*, Vol. 4595. Springer, 263–267.

Nir Piterman. 2006. From Nondeterministic Büchi and Streett Automata to Deterministic Parity Automata. In *LICS*. 255–264.

Amir Pnueli and Aleksandr Zaks. 2008. On the Merits of Temporal Testers. In *25 Years of Model Checking - History, Achievements, Perspectives*. 172–195.

Shmuel Safra. 1988. On the Complexity of ω-Automata. In *FOCS*. IEEE Computer Society, 319–327.

Sven Schewe. 2009. Tighter Bounds for the Determinisation of Büchi Automata. In *FOSSACS*. 167–181.

Salomon Sickert. 2015. Converting Linear Temporal Logic to Deterministic (Generalised) Rabin Automata. *Archive of Formal Proofs* (Sept. 2015). http://afp.sf.net/entries/LTL_to_DRA.shtml, Formal proof development.

Fabio Somenzi and Roderick Bloem. 2000. Efficient Büchi Automata from LTL Formulae. In *CAV (LNCS)*, Vol. 1855. Springer, 248–263.

Moshe Y. Vardi. 1999. Probabilistic Linear-Time Model Checking: An Overview of the Automata-Theoretic Approach. In *Formal Methods for Real-Time and Probabilistic Systems, 5th International AMAST Workshop*. 265–276.

Moshe Y. Vardi and Pierre Wolper. 1986. An Automata-Theoretic Approach to Automatic Program Verification (Preliminary Report). In *LICS*. 332–344.

Moshe Y. Vardi and Pierre Wolper. 1994. Reasoning About Infinite Computations. *Inf. Comput.* 115, 1 (1994), 1–37.

Makarius Wenzel. 2007. Isabelle/Isar—a generic framework for human-readable proof documents. *From Insight to Proof—Festschrift in Honour of Andrzej Trybulec* 10, 23 (2007), 277–298.

Makarius Wenzel. 2014. The Isabelle/Isar Reference Manual. (2014). https://isabelle.in.tum.de/doc/isar-ref.pdf [Online; accessed 19-May-2015].

Lemma 2.9. For every formula φ and every finite word $w \in (2^{Ap})^*$:

(1) $af(\varphi, w)$ is a boolean combination of proper subformulae of φ.
(2) If $af(\varphi, w) = \mathbf{tt}$, then $af(\varphi, ww') = \mathbf{tt}$ for every $w' \in (2^{Ap})^*$, and analogously for $\mathbf{ff}$.
(3) If $\varphi_1 \equiv_P \varphi_2$, then $af(\varphi_1, w) \equiv_P af(\varphi_2, w)$.
(4) If φ has n proper subformulae, then the set of formulae reachable from φ has at most 2^{2^n} equivalence classes of formulae with respect to propositional equivalence.

PROOF. (1) By structural induction on φ.
(2) Follows immediately from $af(\mathbf{tt}, \nu) = \mathbf{tt}$ and $af(\mathbf{ff}, \nu) = \mathbf{ff}$.
(3) By (1) every formula φ is a positive boolean combination of proper formulae. Since af distributes over $\wedge$ and $\vee$, the formula $af(\varphi, \nu)$ is obtained by applying a simultaneous substitution to the proper formulae. (For example, a proper formula $\mathbf{G}\psi$ is substituted by $af(\psi, \nu) \wedge \mathbf{G}\psi$.) Let $\varphi[S]$ be the result of the substitution.
Consider two equivalent formulae $\varphi_1 \equiv_P \varphi_2$. Since we apply the same substitution to both sides, the substitution lemma of propositional logic guarantees $\varphi_1[S] \equiv_P \varphi_2[S]$. So $af(\varphi_1, \nu) \equiv_P af(\varphi_2, \nu)$ for a letter ν. The general case $af(\varphi_1, w) \equiv_P af(\varphi_2, w)$ follows by induction on the length of w.
(4) Follows from (1) and the fact that there are 2^{2^n} equivalence classes of boolean formulae with n variables. $\square$

Proposition 2.10. Let φ be a formula, and let $ww' \in (2^{Ap})^\omega$ be an arbitrary word. Then $ww' \models \varphi$ iff $w' \models af(\varphi, w)$.

PROOF. First we prove the property when w is a single letter ν:

$$\nu w' \models \varphi \quad \text{iff} \quad w' \models af(\varphi, \nu) \tag{2}$$

We prove (2) by structural induction on φ. We only consider two representative cases.

- $\varphi = a$. Then

$$\begin{array}{llllr} & \nu w' \models a & & \nu w' \not\models a & \\ \text{hence} & a \in \nu & \text{hence} & a \notin \nu & \text{(semantics of LTL)} \\ \text{hence} & af(a, \nu) = \mathbf{tt} & \text{hence} & af(a, \nu) = \mathbf{ff} & \text{(def. of } af\text{)} \\ \text{hence} & w' \models af(a, \nu) & \text{hence} & w' \not\models af(a, \nu) & \end{array}$$

- $\varphi = \mathbf{F}\varphi'$. Then

$$\begin{array}{llr} & \nu w' \models \mathbf{F}\varphi' & \\ \text{iff} & \nu w' \models (\mathbf{XF}\varphi') \vee \varphi' & (\mathbf{F}\varphi' \equiv \mathbf{XF}\varphi' \vee \varphi') \\ \text{iff} & \big(w' \models \mathbf{F}\varphi'\big) \vee \big(\nu w' \models \varphi'\big) & \text{(semantics of LTL)} \\ \text{iff} & \big(w' \models \mathbf{F}\varphi'\big) \vee \big(w' \models af(\varphi', \nu)\big) & \text{(ind. hyp.)} \\ \text{iff} & w' \models \mathbf{F}\varphi' \vee af(\varphi', \nu) & \text{(def. of } af\text{)} \\ \text{iff} & w' \models af(\mathbf{F}\varphi', \nu) & \text{(def. of } af\text{)} \end{array}$$

Now we prove the property for every word w by induction on the length of w. If $w = \epsilon$ then $af(\varphi, w) = \varphi$, and so $ww' \models \varphi$ iff $w' \models \varphi$ iff $w' \models af(\varphi, w)$. If $w = \nu w''$ for some $\nu \in 2^{Ap}$, then we have

$$\begin{array}{lll} & w' \models af(\varphi, w) & \\ \text{iff} & w' \models af(\varphi, \nu w'') & \\ \text{iff} & w' \models af(af(\varphi, \nu), w'') & \text{(def. of } af\text{)} \\ \text{iff} & w''w' \models af(\varphi, \nu) & \text{(ind. hyp.)} \\ \text{iff} & \nu w''w' \models \varphi & (2) \\ \text{iff} & ww' \models \varphi & \end{array}$$

□

Lemma 4.17. Let $\mathbf{i}$ be the rank of condition (2) in Theorem 4.16. If the rank of τ stabilizes, then $strk_w(\tau) < \mathbf{i}$.

PROOF. We first prove the following two claims, where $\mathbf{i}$ is the rank of condition (2):

(a) If τ succeeds at rank $\mathbf{i}$, then $strk_w(\tau) < \mathbf{i}$.
Since τ has rank $\mathbf{i}$ when it reaches the accepting states, we clearly have $strk_w(\tau) \leq \mathbf{i}$. We show $strk_w(\tau) < \mathbf{i}$. Assume the contrary. With the previous observation, we have $strk_w(\tau) = \mathbf{i}$. Let t be some time at which τ has already entered the accepting states, and its rank has stabilized. By (2.1), some token τ' born after time t (i.e., $\tau' > t$) also succeeds at rank $\mathbf{i}$. Let $t' \geq t$ be the time immediately before τ' enters the accepting states. Then we have $rk_w(\tau, t') = \mathbf{i}$, because at time t' token τ has already stabilized, and $rk_w(\tau', t') = \mathbf{i}$ by definition. But at time t' token τ is in some accepting state, while τ' is not. So we have two tokens in different states with the same rank, contradicting the definition of rank.

(b) If $rk_w(\tau, t) \leq rk_w(\tau', t) = strk_w(\tau') \in \mathbb{N}$, then $rk_w(\tau, t) = strk_w(\tau)$.
(If a token has reached its stable rank at some time t, then so have all tokens of older rank.)
Assume $rk_w(\tau, t) \neq strk_w(\tau)$. Then at some time $t' > t$ the rank of τ either becomes $\perp$ (because τ reaches a sink) or improves (because τ's firm merges with a firm of older rank). In both cases, the rank of $rk_w(\tau', t)$ also improves (because the rank of τ becomes vacant), contradicting the assumption that at time t token τ has already reached its stable rank.

Assume now that the rank of τ stabilizes but $strk_w(\tau) \geq \mathbf{i}$. By (2.1), some token τ' born after the rank of τ stabilizes succeeds at rank $\mathbf{i}$. Since $q_0 \notin F$, this token eventually enters the accepting states. Let t be the time immediately before τ' enters the accepting states. We have $rk_w(\tau', t) = \mathbf{i}$. Since $strk_w(\tau) \geq \mathbf{i}$, we have $rk_w(\tau, t) \geq \mathbf{i} = rk_w(\tau', t)$. By (b) (with the roles of τ and τ reversed), we get $rk_w(\tau', t) = strk_w(\tau')$, and so $strk_w(\tau') = \mathbf{i}$. But, since τ' succeeds at rank $\mathbf{i}$, this contradicts (a). □

Proposition 5.1. Let $\mathcal{M}_1 = (Q_1, \Sigma, q_{01}, \delta_1, F_1)$ and $\mathcal{M}_2 = (Q_2, \Sigma, q_{02}, \delta_2, F_2)$. Let $Q = Q_1 \times Q_2$, let $q_0 = (q_{01}, q_{02})$, and let $\delta \colon Q \times \Sigma \to Q$ be the function given by $\delta(q_1, q_2, \nu) = (\delta_1(q_1, \nu), \delta_2(q_2, \nu))$ Then the tuples

$$\begin{aligned} \mathcal{M}_1 \cap \mathcal{M}_2 &= \big(Q, \Sigma, q_0, \delta, F_1 \times F_2\big) \\ \mathcal{M}_1 \cup \mathcal{M}_2 &= \big(Q, \Sigma, q_0, \delta, (F_1 \times Q_2) \cup (Q_1 \times F_2)\big) \end{aligned}$$

are also Mojmir automata, and moreover $\mathsf{L}(\mathcal{M}_1 \cap \mathcal{M}_2) = \mathsf{L}(K_1) \cap \mathsf{L}(K_2)$ and $\mathsf{L}(\mathcal{M}_1 \cup \mathcal{M}_2) = \mathsf{L}(K_1) \cup \mathsf{L}(K_2)$.

PROOF. We have to show that states reachable from an accepting state of $\mathcal{M}_1 \cap \mathcal{M}_2$ or $\mathcal{M}_1 \cup \mathcal{M}_2$ are again accepting. If (q_1, q_2) is an accepting state of $\mathcal{M}_1 \cap \mathcal{M}_2$ or $\mathcal{M}_1 \cup \mathcal{M}_2$, then by definition $\delta((q_1, q_2), \nu) = (\delta_1(q_1, \nu), \delta_2(q_2, \nu))$.

- If $(q_1, q_2) \in F_1 \times F_2$, then, since $\mathcal{M}_1$ and $\mathcal{M}_2$ are $\mathcal{M}$ automata, we have $\delta_1(q_1, \nu) \in F_1$ and $\delta_2(q_2, \nu) \in F_2$, and so $\delta((q_1, q_2), \nu) \in F_1 \times F_2$.
- If $(q_1, q_2) \in (F_1 \times Q_2) \cup (Q_1 \times F_2)$, then, since $\mathcal{M}_1$ and $\mathcal{M}_2$ are $\mathcal{M}$ automata, we have $\delta(q_1, \nu) \in F_1$ or $\delta(q_2, \nu) \in F_2$, and so $\delta((q_1, q_2), \nu) \in (F_1 \times Q_2) \cup (Q_1 \times F_2)$.

We now prove $\mathsf{L}(\mathcal{M}_1 \cap \mathcal{M}_2) = \mathsf{L}(K_1) \cap \mathsf{L}(K_2)$ and $\mathsf{L}(\mathcal{M}_1 \cup \mathcal{M}_2) = \mathsf{L}(K_1) \cup \mathsf{L}(K_2)$. Since $\mathcal{M}_1 \cap \mathcal{M}_2$ and $\mathcal{M}_1 \cup \mathcal{M}_2$ only differ in their accepting states, they have the same function $run_w(\tau, t)$ describing the position of token τ at time t. Moreover, by the definition of q_0 and δ we easily get

$$run_w(\tau, t) = \big(run1_w(\tau, t), run2_w(\tau, t)\big)$$

where $run1$ and $run2$ are the corresponding functions for $\mathcal{M}_1$ and $\mathcal{M}_2$. So we have

(a) Token τ of $\mathcal{M}_1 \cap \mathcal{M}_2$ eventually reaches $F_1 \times F_2$ iff the token τ of $\mathcal{M}_1$ eventually reaches F_1 and the token τ of $\mathcal{M}_2$ eventually reaches F_2.
(b) Token τ of $\mathcal{M}_1 \cup \mathcal{M}_2$ eventually reaches $(F_1 \times Q_2) \cup (Q_1 \times F_2)$ iff the token τ of $\mathcal{M}_1$ eventually reaches F_1, or the token τ of $\mathcal{M}_2$ eventually reach F_2.

By (a), almost every token of $\mathcal{M}_1 \cap \mathcal{M}_2$ eventually reaches $F_1 \times F_2$ iff almost every token of $\mathcal{M}_1$ eventually reaches F_1, and almost every token of $\mathcal{M}_2$ eventually reaches F_2. So $\mathsf{L}(\mathcal{M}_1 \cap \mathcal{M}_2) = \mathsf{L}(K_1) \cap \mathsf{L}(K_2)$. By (b), almost every token of $\mathcal{M}_1 \cap \mathcal{M}_2$ eventually reaches $(F_1 \times Q_2) \cup (Q_1 \times F_2)$ iff almost every token of $\mathcal{M}_1$ eventually reaches F_1, or almost every token of $\mathcal{M}_2$ eventually reaches F_2. So $\mathsf{L}(\mathcal{M}_1 \cup \mathcal{M}_2) = \mathsf{L}(K_1) \cup \mathsf{L}(K_2)$ $\square$

Lemma 5.5. Let φ be a formula and let w be a word.

(a) Every set $\mathcal{G} \subseteq \mathbb{G}\varphi$ closed for w is included in $\mathcal{G}_w(\varphi)$.
(b) $\mathcal{G}_w(\varphi)$ is closed for w.

PROOF. (a): Given $\mathcal{G} \subseteq \mathbb{G}\varphi$, we inductively assign to every $\mathbf{G}\psi \in \mathcal{G}$ an index as follows. If ψ has no $\mathbf{G}$-subformulae, then $\mathbf{G}\psi$ has index 0; if ψ has $\mathbf{G}$-subformulae, then its index is the maximum of the indices of its subformulae plus 1.

Assume $\mathcal{G} \subseteq \mathbb{G}(\varphi)$ is closed for w, and let $\mathbf{G}\psi \in \mathcal{G}$. We prove $w \models \mathbf{FG}\psi$ by induction on the index n of $\mathbf{G}\psi$.

— $n = 0$. Since $\mathcal{G}$ is closed, we have $\mathcal{G} \models_P af_{\mathbf{G}}(\psi, w_{ij})$ for almost every $i \in \mathbb{N}$ and almost every $j \geq i$. Let $j > i$ be such that $\mathcal{G} \models_P af_{\mathbf{G}}(\psi, w_{ij})$ holds. Since ψ has no $\mathbf{G}$-subformulae (because $n = 0$), the formulae of $\mathcal{G}$ occur neither in ψ nor, by the definition of $af_{\mathbf{G}}$, in $af_{\mathbf{G}}(\psi, w_{ij})$. So we get $\emptyset \models_P af_{\mathbf{G}}(\psi, w_{ij})$, which implies $af_{\mathbf{G}}(\psi, w_{ij}) \equiv_P \mathbf{tt}$. Moreover, since ψ has no subformulae and $af_{\mathbf{G}}$ and af only differ on $\mathbf{G}$-formulae, we have $af_{\mathbf{G}}(\psi, w_{ij}) = af(\psi, w_{ij})$. So we finally obtain $af(\psi, w_{ij}) \equiv_P \mathbf{tt}$ for almost every $i \in \mathbb{N}$ and almost every $j \geq i$. Apply now Theorem 4.1.
— $n > 0$. Let $\mathcal{G}'$ be the set of formulae of $\mathcal{G}$ that are subformulae of ψ. For every $\mathbf{G}\psi' \in \mathcal{G}'$ the index of $\mathbf{G}\psi'$ is at most $n-1$ and so, by induction hypothesis, we have $w \models \mathbf{FG}\psi'$. So there exists k_1 such that $w_i \models \mathcal{G}'$ for every $i \geq k_1$.
Moreover, since $\mathcal{G}$ is closed, we have $\mathcal{G} \models_P af_{\mathbf{G}}(\psi, w_{ij})$ for almost every $i \in \mathbb{N}$ and almost every $j \geq i$. Further, since the formulae of $\mathcal{G} \setminus \mathcal{G}'$ do not appear in any $af_{\mathbf{G}}(\psi, w_{ij})$, there exists k_2 such that $\mathcal{G}' \models_P af_{\mathbf{G}}(\psi, w_{ij})$ for every $i \geq k_2$ and almost every $j \geq i$.
Taking $k = \max\{k_1, k_2\}$, we obtain:
(i) $w_i \models \mathcal{G}'$ for every $i \geq k$, and
(ii) $\mathcal{G}' \models_P af_{\mathbf{G}}(\psi, w_{ij})$ for every $i \geq k$ and almost every $j \geq i$.

We show that (i) and (ii) imply $w_i \models \psi$ for almost every $i \geq k$. We proceed by an structural induction on ψ, very similar to the one in the proof of Proposition 2.10, except for the case $\psi = \mathbf{G}\psi'$. We omit some cases, and only sketch the proof of others.

— $\psi = a$. Let $i \geq k$ such that (i) holds. By (ii) we have $\mathcal{G}' \models_P af_{\mathbf{G}}(a, w_{ij})$ for almost every $j \geq i$, and so $af_{\mathbf{G}}(a, w_{ij}) = \mathbf{tt}$ for almost every $j \geq i$. But $af_{\mathbf{G}}(a, w_{ij}) = \mathbf{tt}$ implies $w_{i(i+1)} = a$, and so $w_i \models a$.

— $\psi = \psi_1 \wedge \psi_2$ and $\psi = \psi_1 \vee \psi_2$. Both cases follow immediately from the induction hypothesis.

— $\psi = \mathbf{G}\psi'$. By the definition of $af_{\mathbf{G}}$, we have $af_{\mathbf{G}}(\psi, w_{ij}) = \mathbf{G}\psi' = \psi$ for every $j \geq i$. So, by (ii), we have $\mathcal{G}' \models_P \psi$ which, together with (i), implies $w_i \models \psi$ for every $i \geq k$.

(b): We first prove a preliminary result: if $w \models \varphi$, then $\mathcal{G}_w(\varphi) \models af_{\mathbf{G}}(\varphi, w_{0i})$ for almost every $i \in \mathbb{N}$. The proof is very similar to that of Theorem 3.1. It suffices to say that we proceed by structural induction on φ, using the same arguments as in Theorem 3.1, with two minor adjustments:

— $af_{\mathbf{G}}(\varphi, w_{0i}) \equiv_P \mathbf{tt}$ is replaced by $\mathcal{G}_w(\varphi) \models af_{\mathbf{G}}(\varphi, w_{0i})$.

— The $\mathbf{G}$-case, i.e., $\varphi = \mathbf{G}\varphi'$, is proved differently. It follows immediately from the fact that, since $w \models \mathbf{G}\varphi'$ by assumption, we have $\mathbf{G}\varphi' \in \mathcal{G}_w(\mathbf{G}\varphi')$.

Now we proceed to prove (b), also by structural induction on φ. If φ is not a $\mathbf{G}$-formula, then the result follows either directly from the definitions or directly from the induction hypothesis. So consider the case $\varphi = \mathbf{G}\varphi'$. By definition we have $\mathcal{G}_w(\varphi') \subseteq \mathcal{G}_w(\varphi)$, and by induction hypothesis $\mathcal{G}_w(\varphi')$ is closed. If $w \not\models \mathbf{FG}\varphi'$ then $\mathcal{G}_w(\varphi') = \mathcal{G}_w(\varphi)$, and so $\mathcal{G}_w(\varphi)$ is closed. If $w \models \mathbf{FG}\varphi'$ then $\mathcal{G}_w(\varphi) = \mathcal{G}_w(\varphi') \cup \{\mathbf{G}\varphi'\}$. Since $\mathcal{G}_w(\varphi')$ is closed, we have $\mathcal{G}_w(\varphi') \models_P af_{\mathbf{G}}(\psi, w_{ij})$ for almost every $i \in \mathbb{N}$, almost every $j \geq i$, and for every $\mathbf{G}\psi \in \mathcal{G}_w(\varphi')$. So it suffices to show $\mathcal{G}_w(\varphi) \models_P af_{\mathbf{G}}(\varphi', w_{ij})$ for almost all every $i \in \mathbb{N}$ and almost every $j \geq i$. Since $w \models \mathbf{FG}\varphi'$, we have $w_i \models \varphi'$ for almost all $i \in \mathbb{N}$. Applying the preliminary result above to every w_i, we obtain $\mathcal{G}_w(\varphi') \models_P af_{\mathbf{G}}(\varphi', w_{ij})$ for almost every $i \in \mathbb{N}$ and almost every $j \geq i$, and we are done. □

Lemma 5.8. Let φ be a formula and let $\mathcal{G} \subseteq \mathbb{G}(\varphi)$. For every $\psi \in Reach_{\mathbf{G}}(\varphi)$ and every $\nu \in 2^{Ap}$, if $\mathcal{G} \models_P \psi$ then $\mathcal{G} \models_P af_{\mathbf{G}}(\psi, \nu)$.

PROOF. We proceed by induction on the structure of ψ. Since $\mathcal{G} \models_P \psi$, by the definition of propositional implication, the formula ψ must be either $\mathbf{tt}$, a conjunction, a disjunction, or a $\mathbf{G}$-formula. If $\psi = \mathbf{tt}$ then $af_{\mathbf{G}}(\psi, \nu) = \mathbf{tt}$ and we are done. If $\psi = \psi_1 \wedge \psi_2$ then $af_{\mathbf{G}}(\psi, \nu) = af_{\mathbf{G}}(\psi_1, \nu) \wedge af_{\mathbf{G}}(\psi_2, \nu)$ and $\mathcal{G} \models_P af_{\mathbf{G}}(\psi, \nu)$ follows immediately from the induction hypothesis. The case $\psi = \psi_1 \vee \psi_2$ is analogous. Finally, if $\psi = \mathbf{G}\psi'$ for some formula ψ' then $af_{\mathbf{G}}(\mathbf{G}\psi') = \mathbf{G}\psi'$, and we are done. □

Lemma 6.14. Let $\mathcal{M}(\psi, \mathcal{G})$ be the Mojmir automaton for a formula ψ. Assume $\mathcal{M}(\psi, \mathcal{G})$ accepts a word w at the smallest accepting rank $\mathbf{r}$. For almost every $t \in \mathbb{N}$ and for every token τ of the run of $\mathcal{M}(\psi, \mathcal{G})$ on w, the token succeeds iff

(1) $\tau > t$, or
(2) $sr_w(run_w(\tau, t), t) \geq r$, or
(3) $run_w(\tau, t) \in F$.

PROOF. Consider the accepting run of $\mathcal{M}(\psi, \mathcal{G})$ on w. Let k' be large enough such that at time $t' \geq k'$: all tokens τ born after k' eventually succeed; the finitely many tokens that fail have already reached a sink; and the finitely many tokens that succeed with rank smaller than r have already already reached an accepting state. Notice that such a k' only exists for the smallest accepting rank, since infinitely many tokens enter the accepting states with this rank and for all larger accepting ranks this constant does not exist. Furthermore let $k \geq k'$ be large enough so that all squatting tokens born before or at time k' have already reached their stable rank at time k. We show that the lemma holds for every $t \geq k$.

Let τ be an arbitrary token.

- Assume τ succeeds. We show that if (1) and (3) do not hold, then (2) holds. By (3), τ has not yet reached the accepting states. By our choice of k', by the time τ enters the accepting states it will have rank r or larger. Since the rank of a token can only decrease, its current rank is also equal to the accepting rank r or larger. So $sr_w(run_w(\tau, t), t) \geq r$.
- Assume (1), (2), or (3) hold. If (3) holds, then τ succeeds by the definition of success. If (1) holds, then τ succeeds by our choice of k'. Assume now that (2) holds. We show that (2) neither fails nor squats outside the accepting states, and so necessarily succeeds. Since τ has a rank at time t, it is not in a sink, and so, by our choice of k', the token does not fail. To show that τ does not squat outside the accepting states, we recall part (c) in the proof of Theorem 4.16: the stable rank of a token is bounded from above by accepting ranks, thus also by the smallest. So, by (2), the rank of τ has not stabilized yet, and therefore, by our choice of k, it does not squat outside the accepting states.

□