Satisfiability Checking for the Coalgebraic μ-Calculus

Erfüllbarkeitsprüfung für den Koalgebraischen μ-Kalkül

Der Technischen Fakultät
der Friedrich-Alexander-Universität
Erlangen-Nürnberg
zur
Erlangung des Doktorgrades Dr.-Ing.
vorgelegt von
Daniel Hausmann
aus Heidelberg

Als Dissertation genehmigt
von der Technischen Fakultät
der Friedrich-Alexander-Universität Erlangen-Nürnberg

Tag der mündlichen Prüfung: 28. Juni 2018

Vorsitzender des Promotionsorgans: Prof. Dr.-Ing. Reinhard Lerch

Gutachter: Prof. Dr. Lutz Schröder
Prof. Dr. Alexander Kurz

Abstract: The coalgebraic μ-calculus is an expressive logic that generalizes the modal μ-calculus by interpreting formulas over *T-coalgebras* rather than Kripke structures. Due to the presence of fixpoint operators, the semantics of μ-calculi is rather involved and satisfiability checking for μ-calculus formulas typically necessitates the use of automata theoretic concepts, in particular the determinization of automata on infinite words. We introduce novel notions of automata in which all accepting runs are deterministic or linear, *from some point on*, and give novel determinization methods for such automata. In particular, we show that *limit-deterministic* parity automata with n states and k priorities can be determinized to parity automata of size $\mathcal{O}((nk)!)$ and with $\mathcal{O}(nk)$ priorities and that *limit-linear* Co-Büchi automata of size n can be determinized to Co-Büchi automata of size at most $n^2 \cdot 2^n$. We obtain satisfiability games for the coalgebraic μ-calculus by employing *tracking automata*, that is, parity automata that track fixpoint formulas through pre-tableaux. The tracking automata for *alternation-free* formulas are parity automata with just the priorities 0 and 1, i.e. Co-Büchi automata; tracking automata for *aconjunctive* formulas are limit-deterministic and tracking automata for *linear* formulas are limit-linear. We define *satisfiability games via determinization*, that is, satisfiability games for the coalgebraic μ-calculus that are constructed by determinizing tracking automata, and show that Éloïse wins such a game if and only if the corresponding formula is satisfiable. We also prove that for coalgebraic μ-calculus formulas of size n and with *alternation depth* k, the number of Abélard-nodes in the corresponding satisfiability game via determinization is bounded by $n^2 \cdot 2^n$ for linear formulas and by 3^n for alternation-free formulas, and is in $\mathcal{O}((nk)!)$ for aconjunctive formulas and in $\mathcal{O}(n!(nk)^{nk})$ for unrestricted formulas. As models are built over Abélard-nodes, we obtain corresponding bounds on the sizes of models for satisfiable formulas from the mentioned fragments. We also introduce *satisfiability games via focusing* that can be constructed without determinizing automata but only work for formulas that are both alternation-free and aconjunctive. We present an algorithm that can be instantiated to any of the presented games and solves the satisfiability problem of the (corresponding fragment of the) coalgebraic μ-calculus in ExpTime and *on-the-fly*. The algorithm has been implemented (see `https://www8.cs.fau.de/research:software:cool`) and it has been shown that both the asymptotically smaller games obtained by the new determinization procedures and on-the-fly solving allows for shorter runtimes in comparison with other satisfiability checking tools.

Zusammenfassung: Die Theorie der universellen Koalgebra stellt ein mathematisches Rahmenwerk zur formalen Behandlung von reaktiven Systemen in der Informatik dar. Die *Koalgebraische Modallogik* verwendet in diesem Rahmenwerk Koalgebren als logische Modelle und ermöglicht es somit, das Verhalten von allgemeinen reaktiven Systemen präzise zu Beschreiben. Der koalgebraische μ-Kalkül erweitert koalgebraische Modallogik um Fixpunktoperatoren mit deren Hilfe es möglich ist, differenzierte Aussagen über unendliches Verhalten von reaktiven Systemen zu treffen. Diese Arbeit behandelt das Erfüllbarkeitsproblem des koalgebraischen μ-Kalküls; hierzu werden sogenannte *Erfüllbarkeitsspiele* zunächst definiert und kommen sodann in einem generischen Algorithmus zur Anwendung um die Erfüllbarkeit von koalgebraischen μ-Kalkülformeln in ExpTime zu überprüfen. Die Konstruktion der Erfüllbarkeitsspiele determinisiert dabei Automaten über unendlichen Wörtern; zwei neuartige Algorithmen zur Determinisierung sogenannter *limes-linearer* und *limes-deterministischer* Automaten stellen ein zentrales Ergebnis der Arbeit dar. Für bestimmte Fragmente des koalgebraischen μ-Kalküls haben die resultierenden Erfüllbarkeitsspiele sodann eine asymptotisch kleinere Größe, als die mit bisherigen Determinisierungsmethoden erzeugbaren Erfüllbarkeitspiele.

Die zentralen theoretischen Ergebnisse dieser Dissertation wurden als wissenschaftliche Publikationen veröffentlicht [23,25,24].

Der entwickelte Erfüllbarkeitsprüfungsalgorithmus wurde als Erweiterung des Beweiswerkzeugs *Coalgebraic Ontology Logic Reasoner (COOL)* implementiert; der Quellcode, Testformeln, ein Formelgenerator sowie weiterführende Informationen sind unter `https://www8.cs.fau.de/research:software:cool` zu finden. Die Implementierung unterstützt das sogenannte *on-the-fly*-Lösen der jeweiligen Erfüllbarkeitspiele; somit kann die Erfüllbarkeit bzw. Unerfüllbarkeit einer Formel gegebenfalls erkannt werden, bevor das Spiel vollständig konstruiert wurde.

Table of Contents

– Für Sarah –

1 Introduction

In recent decades, the theory of universal coalgebra [43] has been established as a mathematically inspired area of research in theoretical computer science that forms the basis of a unified concept of formal reasoning about reactive systems. Coalgebraic modal logics [35,38,51] have emerged as suitable generic and expressive languages to describe the behaviour of coalgebras. The coalgebraic μ-calculus [7] is an even more expressive logic that is obtained by extending coalgebraic modal logic with fixpoint operators which enable statements about the infinite behaviour of coalgebras. This thesis contributes to the research on the satisfiability problem of this logic. Due to the rather involved structure of the semantics of the coalgebraic μ-calculus, automata theoretic, logical and coalgebraic concepts and methods have to combined to obtain correct and efficient decision procedures for this decision problem.

In this context, the central novel contributions of the work that is summarized in this thesis comprise

- the definition of *limit-deterministic* and *limit-linear* automata on infinite words,
- the conception of novel determinization methods for these types of automata,
- the definition of so-called *satisfiability games via tracking automata*, that is, generic satisfiability games for the coalgebraic μ-calculus that are constructed by determinizing *tracking automata* and can be instantiated to different fragments of the logic. For *aconjunctive* formulas [28], the involved determinization method for limit-deterministic parity automata uses partial permutations and results in asymptotically smaller parity games [24] than the Safra/Piterman [44,42] construction. For *alternation-free* formulas [23], Büchi games are obtained by using standard Co-Büchi automaton determinization. For *linear* formulas, Büchi games are obtained by using the introduced determinization procedure for limit-linear Co-Büchi automata.
- the definition of *satisfiability games via focusing* for coalgebraic μ-calculus formulas that are both alternation-free and aconjunctive. These games generalize the previously known focusing games for CTL [31].
- the presentation of a generic tableau-based algorithm that employs global caching to solve all types of satisfiability games that are presented in this work, thus deciding satisfiability of coalgebraic μ-calculus formulas in ExpTime (under mild assumptions); depending on the fragment to which the algorithm is instantiated, the algorithm can use the smallest appropriate satisfiability games. The algorithm supports on-the-fly solving of games and hence may finish satisfiability proofs or refutations early.

The central theoretical results of this work have been published [23,25,24].

The developed global caching algorithm has been implemented as part of the *Coalgebraic Ontology Logic Reasoner (COOL)*; the source code, test formulas, bench-

marking scripts, a formula generator, and more information on the tool can be found at `https://www8.cs.fau.de/research:software:cool`. The current version of the implementation supports various coalgebraic base logics and realizes satisfiability checking by means of the introduced satisfiability games via focusing for the alternation-free and aconjunctive fragment of the coalgebraic μ-calculus (using determinization of *limit-linear* Co-Büchi automata) and the introduced satisfiability games via tracking automata for the alternation-free and the aconjunctive fragments of the coalgebraic μ-calculus (using standard determinization of Co-Büchi automata and permutation determinization of limit-deterministic parity automata, respectively). The implementation supports on-the-fly solving for all currently implemented fragments and the practical effect of a) satisfiability games that are asymptotically smaller than the standard games used by other tools, b) on-the-fly solving and c) global caching has been shown in [25,24].

This thesis is structured as follows: We introduce several basic notions such as automata on finite words, finite satisfiability games for basic modal logic, fixpoints of functions, the relational μ-calculus, and T-coalgebras in Chapter 2. Then we recall the generalization of modal logic to the coalgebraic level of generality in Chapter 3 and introduce the central concept of *one-step rules* that enable automated reasoning in coalgebraic logic by means of sequent or tableau-methods; we also define finite satisfiability games that use one-step rules to decide the satisfiability of coalgebraic modal formulas in PSpace (under mild assumptions). In Chapter 4, we introduce automata on infinite words along with infinite games and present determinization methods for various types of automata on infinite words; the determinization methods for limit-deterministic parity automata and limit-linear Co-Büchi automata are novel. Furthermore, we give a discourse on the definition of regions in automata and games by relational μ-calculus formulas, recalling the close connection between model checking for the μ-calculus and some automata and game theoretic problems. Chapter 5 forms the core of this work and combines the coalgebraic notions and the automata theoretic procedures and results from the previous chapters to construct two kinds of generic satisfiability games for the coalgebraic μ-calculus. The construction of these satisfiability games can be instantiated to various fragments of the coalgebraic μ-calculus and then uses the appropriate determinization procedure (if necessary); depending on the syntactic structure of the input formula, this may yield games that are asymptotically smaller than the standard games. We also present the generic algorithm that can be instantiated to solve all introduced satisfiability games in ExpTime (under mild assumptions).

Acknowledgements: I would like to thank Lutz Schröder for his continuous and enduring support, and for encouraging me to make scientific discoveries. He made this work possible. Furthermore, I am thankful to Rajeev Goré, Dirk Pattinson, and Till Mossakowski for helpful discussions and to Horst Reichel and Markus Roggenbach for giving me direction.

I am grateful to my parents for planting the seed that now came to fruition.

2 Preliminaries and Notation

Throughout this work, we will make frequent use of the (bounded) powerset construction on sets which is denoted by $\mathcal{P}$ ($\mathcal{P}^\omega$) and assigns to each set U the set of all its subsets (of cardinality at most ω), i.e. $\mathcal{P}(U) = \{V \mid V \subseteq U\}$ ($\mathcal{P}^\omega(U) = \{V \mid V \subseteq U, |V| \leq \omega\}$, where $|V|$ denotes the cardinality of the set V). Given a set U, a binary relation $R \subseteq U \times U$ on U and an element $u \in U$, we furthermore define the set $R(u) = \{v \in U \mid (u, v) \in R\}$. The relation R is said to be *cyclic* if there is a sequence $u_0, \ldots, u_n$ of elements from U with $(u_i, u_{i+1}) \in R$ for all $0 \leq i < n$ and $u_0 = u_n$ and *acyclic* otherwise. Given two sets U and V, we define $U \setminus V = \{u \in U \mid u \notin V\}$.

We recall several basic notions and results about finite automata and games, fixpoints of functions, the standard μ-calculus and T-coalgebras.

2.1 Automata on Finite Words

Definition 2.1.1 (Finite and infinite words, languages). Given a finite *alphabet* Σ, i.e. a finite set of *letters* $a \in \Sigma$, the set Σ^* of *finite words* over Σ is defined by

$$\Sigma^* = \{w_0 w_1 \ldots w_n \mid n \in \mathbb{N}, w_i \in \Sigma\}.$$

The *length* $|w|$ of a finite word $w = w_0 w_1 \ldots w_n$ is just the number of letters it contains, i.e. $n + 1$. The set Σ^ω of *infinite words* over Σ is defined by

$$\Sigma^\omega = \{w : \mathbb{N} \to \Sigma\},$$

where $w(i)$ denotes the i-th letter in the word w. A *language* is just a set of words.

Definition 2.1.2 (Automata on finite words). A *finite automaton* is a tuple $\mathsf{A} = (V, \Sigma, \Delta, v_0, F)$, where V is a finite set of *states*, Σ is a finite *alphabet*, $\Delta \subseteq V \times \Sigma \times V$ a *transition relation*, v_0 is an initial state and $F \subseteq V$ a set of *accepting states*. If Δ is a function, then A is a *deterministic finite automaton* (DFA); otherwise, it is a *nondeterministic finite automaton* (NFA). For deterministic automata, we usually write δ instead of Δ. For $a \in \Sigma$, an *a-transition* (in A) is a tuple $(u, a, v) \in \Delta$. For $v \in V$, $U \subseteq V$ and $a \in \Sigma$, let

$$\Delta(v, a) := \{w \mid (v, a, w) \in \Delta\}$$

denote the set of states to which v has an a-transition and let

$$\Delta(U, a) := \bigcup_{v \in U} \Delta(v, a)$$

denote the set of states to which some state from U has an a-transition. Given a finite word $w = w_0 w_1 \dots w_n \in \Sigma^*$ of length $|w| = n+1$, a *finite run* of an automaton A on w is a sequence $\rho = v_0 v_1 \dots v_n \in V^{n+1}$ of states such that for all $0 \leq i < n$, $v_{i+1} \in \Delta(v_i, w_i)$; we denote the i-th state in ρ by $\rho(i) := v_i$ and the set of all finite runs of the automaton A on a word w that start at state $v \in V$ by $\mathsf{run}_f(\mathsf{A}, v, w)$ (or just by $\mathsf{run}_f(\mathsf{A}, w)$ if $v = v_0$). The length $|\rho|$ of a finite run ρ is just the number of states in ρ. The language $L(\mathsf{A})$ that is *recognized* or *accepted* by an automaton A on finite words is defined by

$$L(\mathsf{A}) := \{w \in \Sigma^* \mid \exists \rho \in \mathsf{run}_f(\mathsf{A}, w).\rho(|\rho|) \in F\},$$

i.e. by collecting those finite words for which there is a finite run of the automaton that starts at the initial state and ends in an accepting state.

The standard method to transform a NFA $\mathsf{A} = (V, \Sigma, \Delta, v_0, F)$ to an equivalent DFA is the *powerset construction*. Since several different states in A may be reached by reading a single word w, the determinized automaton can be seen to be at each point (i.e. after having read any number of letters) in a *set of states* rather than in a single state; this emulates the nondeterministic choices that can be taken in A.

Definition 2.1.3 (Powerset automaton). Let $\mathsf{A} = (V, \Sigma, \Delta, v_0, F)$ be an NFA. We define the deterministic automaton $\mathcal{P}(\mathsf{A}) = (\mathcal{P}(V), \Sigma, \delta, \{v_0\}, F')$ to have subsets of V (so-called *macrostates*) as states. A reachable macrostate contains all the states that are reachable in A from v_0 via some word w. The transition function $\delta : \mathcal{P}(V) \times \Sigma \to \mathcal{P}(V)$ is defined for $U \subseteq V$ and $a \in \Sigma$ by

$$\delta(U, a) = \Delta(U, a),$$

i.e. $\delta(U, a)$ consists of all states to which an a-transition exists that starts at some state from U. Intuitively, δ performs an a-transition on all states from U, i.e. traces all states from U through one a-step in A. Finally, we define

$$F' = \{U \subseteq V \mid U \cap F \neq \emptyset\},$$

i.e. a macrostate is accepting if and only if any of its states is accepting in A.

Fact 2.1.4. We have $L(\mathsf{A}) = L(\mathcal{P}(\mathsf{A}))$ and $|\mathcal{P}(V)| = 2^{|V|}$.

Thus deterministic and nondeterministic automata are equally expressive; it is a standard result that finite automata recognize *regular languages*, i.e. languages that are definable by *regular expressions*.

2.2 Fixpoints of Functions

Definition 2.2.1 (Fixpoints). Let U be a set and let $f : \mathcal{P}(U) \to \mathcal{P}(U)$ be a function. Then f is called *monotone w.r.t. set inclusion* if for all $V, W \subseteq U$, $V \subseteq W$ implies $f(V) \subseteq f(W)$. We define the sets of *prefixpoints*, *postfixpoints* and *fixpoints* of f by putting

$$\mathsf{PRE}(f) = \{V \subseteq U \mid f(V) \subseteq V\}$$
$$\mathsf{POST}(f) = \{V \subseteq U \mid V \subseteq f(V)\}$$
$$\mathsf{FIX}(f) = \{V \subseteq U \mid V = f(V)\}$$

If $\mathsf{FIX}(f)$ has a greatest (least) element, the we refer to it as the *greatest (least) fixpoint* of f and denote it by $\mathsf{GFP}(f)$ ($\mathsf{LFP}(f)$). We also define the n-fold application f to a set $V \subseteq U$ by putting

$$f^0(V) = V, \qquad f^{n+1}(V) = f(f^n(V)).$$

Greatest and least fixpoints of monotone functions can be characterized as follows:

Theorem 2.2.2 (Knaster-Tarski [54]). *Given a set U and a function $f : \mathcal{P}(U) \to \mathcal{P}(U)$ that is monotone w.r.t. set inclusion, we have*

$$\mathsf{LFP}(f) = \bigcap \mathsf{PRE}(f) \qquad \mathsf{GFP}(f) = \bigcup \mathsf{POST}(f)$$

Proof. We consider the least fixpoint case and note that the proof of the greatest fixpoint case is dual. Put $Q := \bigcap \mathsf{PRE}(f)$ so that we have $Q \subseteq Z$ for all $Z \in \mathsf{PRE}(f)$. By monotonicity of f, we have $f(Q) \subseteq f(Z)$ for all $Z \in \mathsf{PRE}(f)$ and hence by definition of prefixpoints $f(Q) \subseteq Z$ for all $Z \in \mathsf{PRE}(f)$. Thus $f(Q) \subseteq Q$, i.e. Q is a prefixpoint of f. It remains to show that Q is also a postfixpoint of f, i.e. that $Q \subseteq f(Q)$. Since $Q \subseteq Z$ for all $Z \in \mathsf{PRE}(f)$, it suffices to show that $f(Q) \in \mathsf{PRE}(f)$, i.e. that $f(f(Q)) \subseteq f(Q)$. But this follows by monotonicity from $f(Q) \subseteq Q$. Thus Q is a fixpoint of f. Let Q' be a fixpoint of f with $Q' \subseteq Q$. Then Q' is a prefixpoint of f and hence $Q = \bigcap \mathsf{PRE}(f) = \bigcap \mathsf{PRE}(f) \cap Q'$ and $Q \subseteq Q'$, i.e. $Q = Q'$. Thus Q is the least fixpoint of f. □

For *stabilizing* functions, we can also construct the extremal fixpoints by finite approximation (as in Kleene's fixpoint theorem):

Lemma 2.2.3. *Let $f : \mathcal{P}(U) \to \mathcal{P}(U)$ be a function that is monotone w.r.t. set inclusion. Further assume that there exist $n, m \in \mathbb{N}$ such that $f^n(\emptyset) = f(f^n(\emptyset))$ and $f^m(U) = f(f^m(U))$ (such functions are said to be* stabilizing*). Then we have*

$$\mathsf{LFP}(f) = f^n(\emptyset) \qquad \mathsf{GFP}(f) = f^m(U),$$

where n and m are the least numbers such that $f^n(\emptyset) = f(f^n(\emptyset))$ and $f^m(U) = f(f^m(U))$.

Proof. We consider the least fixpoint case; the proof of the greatest fixpoint case is dual. By Theorem 2.2.2, we have $\mathsf{LFP}(f) = \bigcap \mathsf{PRE}(f)$. Since $\mathsf{LFP}(f)$ is contained in each prefixpoint of f, $\mathsf{LFP}(f)$ is also contained in the fixpoint $f^n(\emptyset)$. For the other direction, we show $f^n(\emptyset) \subseteq \mathsf{LFP}(f) = \bigcap \mathsf{PRE}(f)$ by induction over n. If $n = 0$, then $f^0(\emptyset) = \emptyset \subseteq Z$ for all $Z \in \mathsf{PRE}(f)$. If $n > 0$, then let $Z \in \mathsf{PRE}(f)$ so that $f^{n-1}(\emptyset) \subseteq Z$ by the induction hypothesis. By monotonicity of f, we have $f^n(\emptyset) = f(f^{n-1}(\emptyset)) \subseteq f(Z)$. Since Z is a prefixpoint, $f(Z) \subseteq Z$ and hence $f^n(\emptyset) \subseteq Z$. □

We note that the construction of extremal fixpoints by approximation is applicable in particular to all monotone functions over finite sets: If U is a finite set and f a monotone function, then there obviously exist $m, n \in \mathbb{N}$ such that $f^n(\emptyset) = f^{n+1}(\emptyset)$ and $f^m(U) = f^{m+1}(U)$, namely $m = n = |U|$.

Definition 2.2.4. Let U and $V \subseteq U$ be two sets. The *complement of V in U* is defined by $\overline{V} = U \setminus V$. Let $f : \mathcal{P}(U) \to \mathcal{P}(U)$ and $g : \mathcal{P}(U) \to \mathcal{P}(U)$ be two functions. The function g is *complementary to f*, if for all $V \subseteq U$,

$$f(V) = \overline{g(\overline{V})}.$$

For all $V, W \subseteq U$, we have that $\overline{\overline{V}} = V$, that $V = W$ implies $\overline{V} = \overline{W}$ and that $V \subseteq W$ implies $\overline{V} \supseteq \overline{W}$.

Least and greatest fixpoints are dual concepts in the following sense:

Lemma 2.2.5. *Let $f : \mathcal{P}(U) \to \mathcal{P}(U)$ be monotone w.r.t. set inclusion and let g be complementary to f. Then*

$$\mathsf{LFP}(f) = \overline{\mathsf{GFP}(g)}$$
$$\mathsf{GFP}(f) = \overline{\mathsf{LFP}(g)}.$$

Proof. First note that since f is monotone and g is complementary to f, g is monotone as well. We consider only the least fixpoint case, the proof of the greatest fixpoint case is dual. We show that $\overline{\mathsf{GFP}(g)}$ is a fixpoint of f that is contained in each prefixpoint of f. Since g is complementary to f and since $\mathsf{GFP}(g)$ is a fixpoint of g, we have

$$f(\overline{\mathsf{GFP}(g)}) = \overline{g(\overline{\overline{\mathsf{GFP}(g)}})} = \overline{g(\mathsf{GFP}(g))} = \overline{\mathsf{GFP}(g)},$$

which shows that $\overline{\mathsf{GFP}(g)}$ is a fixpoint of f. Now let Z be a prefixpoint of f so that $f(Z) \subseteq Z$. Then

$$g(\overline{Z}) = \overline{f(\overline{\overline{Z}})} = \overline{f(Z)} \supseteq \overline{Z},$$

i.e. $\overline{Z}$ is a postfixpoint of g so that $\overline{Z} \subseteq \mathsf{GFP}(g)$. But then $Z \supseteq \overline{\mathsf{GFP}(g)}$, as required. □

Fixpoints operators can be nested: Consider a function $f : \mathcal{P}(U)^k \to \mathcal{P}(U)$ that maps k subsets of U to a single subset of U. The function f is said to be monotone w.r.t. set inclusion if for all $V_1, W_1, \ldots, V_k, W_k \subseteq U$, if $V_i \subseteq W_i$ for all $1 \leq i \leq k$, then $f(V_1, \ldots, V_k) \subseteq f(W_1, \ldots, W_k)$. It is convenient to make the arguments explicit when computing fixpoints of functions with several arguments. E.g. for $k = 1$, we may write $\mathsf{GFP}\ X_1.f(X_1)$ instead of $\mathsf{GFP}f$. As a corollary of Theorem 2.2.2, nested fixpoints of monotone k-ary functions $f : \mathcal{P}(U)^k \to \mathcal{P}(U)$ exist and are defined by

$$\eta_k X_k. \ldots \eta_1 X_1. f(X_1, \ldots, X_k),$$

where $\eta_i \in \{\mathsf{LFP}, \mathsf{GFP}\}$ for $1 \leq i \leq k$. Prominent examples of nested fixpoints include $\nu X_2.\, \mu X_1.\, f(X_1, X_2)$ and $\mu X_2.\, \nu X_1.\, f(X_1, X_2)$, where the former computes the set of those elements from U that satisfy a so-called *Büchi property* defined by f, i.e. elements which can be computed by relying on X_2 only finitely often or by relying on X_1 infinitely often; the latter fixpoint computes the set of elements from U that satisfy a *Co-Büchi property* defined by f, i.e. that can be computed by relying only on X_1 from some point on (for more details, see Section 4.3).

When considering stabilizing functions, elements of nested fixpoints can be annotated with *nested time-outs*, that is, sequences of natural numbers, assigning a time-out, i.e. an approximation number, to each individual fixpoint (for non-stabilizing but monotone functions, instead of natural numbers, ordinal numbers have to be used as approximation numbers in nested time-outs). In [14], a concept similar to nested time-outs is introduced in the more specific context of μ-calculus formulas and referred to as *signatures*.

Definition 2.2.6 (Time-outs for nested fixpoints). Let X be a *finite* set with $n = |X|$ and let $f : \mathcal{P}(X)^k \to \mathcal{P}(X)$ be a k-ary monotone (and by finiteness of X, stabilizing) function. Let $\overline{m} = (m_{k-1}, \ldots, m_{k-j})$ be a *time-out vector*, that is, a vector of $j \leq k$ natural numbers such that $m_i \leq n$ for all $k - j \leq i \leq k - 1$ and $m_i = n$ for all even $k - j \leq i \leq k - 1$. Also let

$$U = \eta_{k-1} X_{k-1} \ldots \eta_0 X_0 . f(X_0, \ldots, X_{k-1}) \subseteq X$$

where $\eta_i = \mathsf{LFP}$ if i is odd and $\eta_i = \mathsf{GFP}$ otherwise. We define the set $F_i(\overline{m})$ for $0 \leq i < k$ by

$$F_i(\overline{m}) = \begin{cases} \emptyset & \text{if } i \text{ is odd and } m_i = 0 \\ f(F_0(\overline{m}_0), \ldots, F_{k-1}(\overline{m}_{k-1})) & \text{if } i \text{ is odd and } m_i > 0 \\ G_i(\overline{m}) & \text{if } i \text{ is even,} \end{cases}$$

where

$$G_i(\overline{m}) = \eta_i X_i \ldots \eta_0 X_0 .\, f(X_0, \ldots, X_i, F_{i+1}(\overline{m}_{i+1}), \ldots, F_{k-1}(\overline{m}_{k-1})),$$

where for $i \geq k - j$,

$$\overline{m}_i = \begin{cases} (m_{k-1}, \ldots, m_i - 1) & \text{if } i \text{ is odd} \\ (m_{k-1}, \ldots, m_i) & \text{if } i \text{ is even} \end{cases}$$

and where for $i < k - j$, $\overline{m}_i$ is the vector $(m_{k-1}, \ldots, m_{k-j}, n, \ldots, n)$ of length $k - i$, that is, $\overline{m}$ extended with $i - j$ copies of n at the end. We abbreviate $f(F_0(\overline{m}_0), \ldots, F_{k-1}(\overline{m}_{k-1}))$ by $f^{\overline{m}}$ and given an element $x \in U$, we say that x has *(nested) time-out* $\overline{m}$ for the function f if $x \in f^{\overline{m}}$. We recall that the standard *lexicographic ordering* $\leq_l$ on vectors of natural numbers of lengths j' and k' is defined by putting $(m_1, \ldots, m_{j'}) \leq_l (m'_1, \ldots, m'_{k'})$ if

1. $j' \leq k'$ and $m_i = m'_i$ for $1 \leq i \leq j'$ or
2. there is a $1 \leq j \leq j'$ such that $m_j < m'_j$ and for all $1 \leq j' < j$, $m_{j'} = m'_{j'}$.

We observe that for all vectors $\overline{m}$ of length j and $1 \leq i \leq j$, $\overline{m}_i \leq_l \overline{m}$. The *least time-out* of $x \in U$ for f is the least (w.r.t. $\leq_l$) vector $\overline{m}$ such that $x \in f^{\overline{m}}$.

When considering time-outs for fixpoints of nesting depth k, two vectors $\overline{m} = (m_{k-1}, \ldots, m_{k-j})$ and $\overline{m}' = (m_{k-1}, \ldots, m_{k-j}, n, \ldots, n)$ with $1 \leq j < k$ hence are equivalent, i.e. we have $f^{\overline{m}} = f^{\overline{m}'}$; this is the case since $\overline{m}$ is padded to the right with copies of n whenever necessary during the computation of $f^{\overline{m}}$. In this sense, e.g. the *empty time-out vector* ϵ and the *maximal vector* $(n, \ldots, n)$ of length $|k-1|$ are equivalent, i.e. we have $f^{\epsilon} = f^{(n,\ldots,n)}$. Usually we omit trailing copies of n when writing time-out vectors.

We now show that the above indeed constitutes a definition, i.e. that nested time-outs exist for elements $x \in U$ of nested fixpoints of stabilizing functions. Items (1) and (2) of Lemma 3 in [14] state an essentially equivalent property for signatures for μ-calculus formulas.

Lemma 2.2.7. *Let X, f and U be as defined above. For each $x \in X$, we have $x \in U$ if and only if there is a time-out vector $\overline{m}$ such that $x \in f^{\overline{m}}$.*

Proof. We show $U = f^{\epsilon}$, noting that for all time-out vectors $\overline{m}$, $f^{\overline{m}} \subseteq f^{\epsilon}$. To this end we define, for $0 \leq i < k$ and $\overline{m}$, the set $U_i(\overline{m})$ by

$$U_i(\overline{m}) = \begin{cases} H_i(\overline{m}) & \text{if } i \text{ is odd} \\ G_i(\overline{m}) & \text{if } i \text{ is even,} \end{cases}$$

where

$$H_i(\overline{m}) = (\eta_{i-1} X_{i-1} \ldots \eta_0 X_0 . f(X_0, \ldots, X_i, U_{i+1}(\overline{m}_{i+1}), \ldots, U_{k-1}(\overline{m}_{k-1})))^{m_i}(\emptyset);$$

here, m_i is the i-th component of $\overline{m}$ if $|\overline{m}| \geq i$ and n otherwise. We first show that for all $0 \leq i < k$ and $\overline{m}$, $U_i(\overline{m}_i) = F_i(\overline{m}_i)$. If i is even, then $F_i(\overline{m}_i) = G_i(\overline{m}_i) = U_i(\overline{m}_i)$ and we are done. If i is odd, then we proceed by lexicographic induction over $\overline{m}_i$. We put $\overline{m}' = \overline{m}_i$ and have $U_i(\overline{m}') = H_i(\overline{m}') = (\eta_{i-1} X_{i-1} \ldots \eta_0 X_0 . f(X_0, \ldots, X_i, U_{i+1}(\overline{m}'_{i+1}), \ldots, U_{k-1}(\overline{m}'_{k-1})))^{m'_i}(\emptyset)$. If m'_i, i.e. the i-th component of $\overline{m}'$, is 0, then $H_i(\overline{m}') = \emptyset = F_i(\overline{m}')$. Otherwise we have $F_i(\overline{m}') = f(F_0(\overline{m}'_0), \ldots, F_{k-1}(\overline{m}'_{k-1})) = f(U_0(\overline{m}'_0), \ldots, U_{k-1}(\overline{m}'_{k-1})) = \nu X_0 . f(X_0, U_1(\overline{m}'_1), \ldots, U_{k-1}(\overline{m}'_{k-1})) = U_0(\overline{m}'_0) = U_0(\overline{m}')$, where the second equality is by the induction hypothesis for odd $0 < q < k$ since then $\overline{m}'_q <_l \overline{m}'$ and since $F_q(\overline{m}'_q) = U_q(\overline{m}'_q)$ as shown above for even q. We also have for all $0 < i < k$ that

$U_i(\overline{m}') = U_{i-1}(\overline{m}')$: If i is even, then

$$\begin{aligned} U_i(\overline{m}') &= G_i(\overline{m}') \\ &= \eta_i X_i. \ldots \eta_0 X_0. f(X_0, \ldots, X_i, F_{i+1}(\overline{m}'_{i+1}), \ldots, F_{k-1}(\overline{m}'_{k-1})) \\ &= \eta_{i-1} X_{i-1}. \ldots \eta_0 X_0. f(X_0, \ldots, X_{i-1}, U_i(\overline{m}'_i), F_{i+1}(\overline{m}'_{i+1}), \ldots, F_{k-1}(\overline{m}'_{k-1})) \\ &= (\eta_{i-2} X_{i-2}. \ldots \eta_0 X_0. f(X_0, \ldots, X_{i-1}, U_i(\overline{m}'_i), \ldots, U_{k-1}(\overline{m}'_{k-1})))^n(\emptyset) \\ &= H_{i-1}(\overline{m}') = U_{i-1}(\overline{m}'), \end{aligned}$$

where $\eta_{i-1} = \mu$, using that $U_q(\overline{m}'_q) = U_q(\overline{m}'_q)$ for $0 < q < k$. If is odd, then

$$\begin{aligned} U_i(\overline{m}') = H_i(\overline{m}') = (f_{i-1})^n(\emptyset) = f_{i-1}((f_{i-1})^{n-1}(\emptyset)) = f_{i-1}(H_i(\overline{m}'_i)) = f_{i-1}(U_i(\overline{m}'_i)) \\ = \eta_{i-1} X_{i-1}. \ldots \eta_0 X_0. f(X_0, \ldots, X_{i-1}, U_i(\overline{m}'_i), \ldots, U_{k-1}(\overline{m}'_{k-1})) \\ = G_{i-1}(\overline{m}') = U_{i-1}(\overline{m}'), \end{aligned}$$

where f_{i-1} is the function that maps the input set X_i to the set $\eta_{i-1} X_{i-1}. \ldots \eta_0 X_0. f(X_0, \ldots, X_i, U_{i+1}(\overline{m}'_{i+1}), \ldots, U_{k-1}(\overline{m}'_{k-1}))$ and the second equality is by Lemma 2.2.3, relying on the finiteness of X. Thus we have shown $F_i(\overline{m}') = U_0(\overline{m}') = U_i(\overline{m}')$. Then we have

$$U = U_{k-1}(\epsilon) = U_0(\epsilon) = f(U_0(\epsilon_0), \ldots, U_{k-1}(\epsilon_{k-1})) = f(F_0(\epsilon_0), \ldots, F_{k-1}(\epsilon_{k-1})) = f^\epsilon,$$

as required. □

Proofs concerning nested fixpoints typically involve the nesting of proofs by induction and proofs by *coinduction* (see e.g. [43]) where the lexicographic order on nested time-outs (as defined above) serves as termination measure for the induction part of the proofs. For nested fixpoints over finite sets, the correctness of this proof principle is based on Lemma 2.2.7 which guarantees the existence of termination measures for all elements of the respective fixpoints.

Example 2.2.8. Let $X = \{x, y, z\}$ with $n := |X| = 3$, let $f : (\mathcal{P}(X))^3 \to \mathcal{P}(X)$ be defined by

$$\begin{aligned} f(X_1, X_2, X_3) = \{u \in X \mid\ & u = y, \{x\} \subseteq X_1 \text{ or} \\ & u = x, \{y, z\} \cap X_2 \neq \emptyset \text{ or} \\ & u = z, \{x, y\} \subseteq X_3\} \end{aligned}$$

for $X_1, X_2, X_3 \subseteq X$ and let $U = \mu X_3.\nu X_2.\mu X_1.f(X_1, X_2, X_3)$. Intuitively, U is the winning region of player Éloïse in a *parity game* (see Definition 4.2.1 below) with set of nodes X where the nodes x, y and z have the priorities 2, 1 and 3, respectively, where x is an Éloïse-node with moves to y and z and where y and z are Abélard-nodes with moves to x and x and y, respectively. To show that $U = X$, it suffices by Lemma 2.2.7 to

show that each node in X has some nested time-out. First we show that x has time-outs $(1, n, n)$, i.e. that

$$\begin{aligned} x \in f^{(1,n,n)} &= f(F_1((1,n,n)_1), F_2((1,n,n)_2), F_3((1,n,n)_3)) \\ &= f(F_1(1,n,n-1), F_2(1,n), F_3(0)) \\ &= f(F_1(1,n,n-1), F_2(1,n), \emptyset) \end{aligned}$$

which is, by definition of f, the case if and only if $\{y,z\} \cap F_2(1,n) \neq \emptyset$. We do not have $z \in F_2(1,n)$ as $z \in F_2(1,n) = G_2(0,n) = \nu X_2.\mu X_1.f(X_1, X_2, \emptyset) = f(\mu X_1.f(X_1, F_2(1,n), \emptyset), F_2(1,n), \emptyset)$ by definition of f if and only if $\{x,y\} \subseteq \emptyset$. However, we have $y \in F_2(1,n) = f^{(1,n,n)} = f(F_1(1,n,n-1), F_2(1,n), \emptyset)$: This is by definition of f the case if and only if $x \in F_1(1,n,n-1)$ which itself is the case if and only if $\{y,z\} \cap X_2 \neq \emptyset$. Thus we have shown that no contradiction arises from y being contained in the greatest fixpoint $F_2(1,n)$; hence, y is by coinduction contained in $F_2(1,n)$. During the above derivation, we already have shown $y \in f^{(1,n,n)}$, i.e. that y has time-outs $(1,n,n)$ and that $z \notin F_2(1,n) = f^{(1,n,n)}$, i.e. that z does not have time-outs $(1,n,n)$. However, we have $z \in f^{(2,n,n)}$: This is the case if and only if $z \in f(F_1(2,n,n-1), F_2(2,n), F_3(1))$ which itself is the case if and only if $\{x,y\} \in F_3(1) = f(F_1(1,n,n-1), F_2(1,n), F_3(0)) = f^{(1,n,n)}$ which has been shown above. We observe that the unfolding of least fixpoints (i.e. the unravelling of clauses $F_i(\overline{m})$ with i odd) reduces nested time-outs. In particular, we have $(2,n,n) >_l (1,n,n)$, where $x, y \in f^{(1,n,n)} \not\ni z$ but $z \in f^{(2,n,n)}$. In terms of parity games, we have shown that player Éloïse wins all nodes in the game by means of the strategy that moves from x to y. As plays of parity games that contain n consequent occurrences of a fixed odd priority visit at least one node twice and hence contain a cycle that is lost by Éloïse, a time-out of n for an odd priority intuitively means that the respective priority may be visited at most $n-1$ times. Hence the mentioned strategy wins the nodes x and y without passing the node z at all (as $x, y \in f^{(1,n,n)}$) and the node z with passing z just once (as $z \in f^{(2,n,n)}$).

2.3 The Relational μ-Calculus

We briefly recall the definition of the μ-calculus [28,29]. We fix a set P of *propositions*, a set A of *actions*, and a set $\mathbf{V}$ of fixpoint variables. The set L_μ of μ-calculus formulas is the set of all formulas ϕ, ψ that can be constructed by the grammar

$$\psi, \phi ::= \bot \mid \top \mid p \mid \neg p \mid X \mid \psi \wedge \phi \mid \psi \vee \phi \mid \langle a \rangle \psi \mid [a]\psi \mid \mu X.\, \psi \mid \nu X.\, \psi$$

where $p \in P$, $a \in A$, and $X \in \mathbf{V}$; we write $|\psi|$ for the size of a formula ψ. Throughout this work, we use η to denote one of the fixpoint operators μ or ν and write $\overline{\eta}$ to denote the dual of η (i.e. $\overline{\mu} = \nu$, $\overline{\nu} = \mu$). We refer to formulas of the form $\eta X.\, \psi$ as *fixpoint literals*, to formulas of the form $\langle a \rangle \psi$ or $[a]\psi$ as *modal literals*, and to p, $\neg p$ as *propositional literals*. The operators μ and ν *bind* their variables, inducing a standard notion of *free variables*

in formulas. We refer to a variable that is bound by a least (greatest) fixpoint operator as μ-variable (ν-variable). We denote the set of free variables of a formula ψ by $\mathsf{FV}(\psi)$. We write $\psi \leq \phi$ ($\psi < \phi$) to indicate that ψ is a (proper) subformula of ϕ.

Formulas $\psi \in \mathsf{L}_\mu$ are evaluated over *Kripke structures* $\mathcal{K} = (W, (R_a)_{a\in A}, \pi)$, consisting of a set W of *states*, a family $(R_a)_{a\in A}$ of relations $R_a \subseteq W \times W$, and a valuation $\pi : P \to \mathcal{P}(W)$ of the propositions. Given an *interpretation* $i : \mathbf{V} \to \mathcal{P}(W)$ of the fixpoint variables, we define $[\![\psi]\!]_i \subseteq W$ by the obvious clauses for Boolean operators and propositions, $[\![X]\!]_i = i(X)$ for $X \in \mathbf{V}$, $[\![\langle a\rangle\psi]\!]_i = \{v \in W \mid R_a(v) \cap [\![\psi]\!]_i \neq \emptyset\}$, $[\![[a]\psi]\!]_i = \{v \in W \mid R_a(v) \subseteq [\![\psi]\!]_i\}$, $[\![\mu X.\,\psi]\!]_i = \mathsf{LFP}[\![\psi]\!]_i^X$ and $[\![\nu X.\,\psi]\!]_i = \mathsf{GFP}[\![\psi]\!]_i^X$, where $R_a(v) = \{w \in W \mid (v, w) \in R_a\}$ and $[\![\psi]\!]_i^X(V) = [\![\psi]\!]_{i[X\mapsto V]}$ for $V \subseteq W$. If ψ does not contain free variables, then $[\![\psi]\!]_i$ does not depend on i, so we just write $[\![\psi]\!]$. In finite models there is by Lemma 2.2.7 for all formulas ψ with $FV(\psi) = \{X_0, \ldots, X_{k-1}\}$ and all states $x \in [\![\eta_{k-1}X_{k-1}.\ldots.\eta_0X_0.\,\psi]\!]$ some time-out vector $\overline{m}$ such that x has time-outs $\overline{m}$ for the fixpoint literal $\eta_{k-1}X_{k-1}.\ldots.\eta_0X_0.\,\psi$.

A μ-calculus formula ψ is *satisfiable* if there is a Kripke structure $\mathcal{K} = (W, (R_a)_{a\in A}, \pi)$ and a state $v \in W$ such that $v \in [\![\psi]\!]$, and *unsatisfiable* otherwise. The formula ψ is *valid* if for all Kripke structures $\mathcal{K} = (W, (R_a)_{a\in A}, \pi)$ and all states $v \in W$, $v \in [\![\psi]\!]$. The *satisfiability problem* of the modal μ-calculus consists in deciding whether a given input formula is satisfiable; the problem is known to be ExpTime-complete [11].

2.4 Finite Games

We now define finite two-player games which essentially are a restriction of infinite games, such as *parity games*, to finite plays (see e.g. [21] for an introduction to infinite games). Formally, a *finite two-player game* $\mathsf{G} = (V, E)$ consists of a finite set of nodes V and an *acyclic* set of moves $E \subseteq V \times V$; the data G is also referred to as a *game arena*. Such games are played between the two players Éloïse and Abélard and the set of nodes V is partitioned accordingly as $V_\exists \cup V_\forall$, where the nodes from $V_\exists$ belong to Éloïse and the nodes from $V_\forall$ belong to Abélard. A *play* in a finite game G is a finite sequence $\rho = v_0v_1 \ldots v_n \in V^{n+1}$ of nodes such that $E(v_n) = \emptyset$ and for all $0 \leq i < n$, $(v_i, v_{i+1}) \in E$; we denote the i-th node in ρ by $\rho(i)$ and say that ρ *starts* at $\rho(0) = v_0$. Player Éloïse *wins* play ρ if $v_n \in V_\forall$ and *loses* it otherwise. Player Abélard wins a play if and only if Éloïse loses it. For $U \subseteq V$, a *(history-free) U-strategy* $s : U \to V$ selects a move $s(v)$ for each $v \in U$. We call $V_\exists$-strategies Éloïse-*strategies* and $V_\forall$-strategies Abélard-*strategies*. A play ρ of length $n+1$ *conforms* to a U-strategy s, if for all $0 \leq i < n$, $\rho(i) \in U$ implies $\rho(i+1) = s(i)$. An Éloïse-strategy s is a *winning strategy* for Éloïse at a node $v \in V$ if she wins every play that starts at v and conforms to s; then we say that Éloïse wins v with strategy s. We have an analogous notion of winning strategies for Abélard. Finite games are determined, i.e. every node $v \in V$ is won by exactly one of the two players. Thus we define the winning regions $\mathsf{win}_\exists$ and $\mathsf{win}_\forall$, i.e. the sets of nodes that are won by the respective player.

Since games are just Kripke structures with atoms indicating the ownership of nodes, the winning region $\mathsf{win}_\exists$ of Éloïse in G can be specified by the μ-calculus formula

$$\phi_\exists = \mu X.\ ((\forall \wedge \Box_E X) \vee (\exists \wedge \Diamond_E X)),$$

where $[\![\exists]\!] = V_\exists$, $[\![\forall]\!] = V_\forall$, $[\![\Diamond_E X]\!](X) = \{v \in V \mid E(v) \cap X \neq \emptyset\}$ and $[\![\Box_E X]\!](X) = \{v \in V \mid E(v) \subseteq X\}$. Thus $\mathsf{win}_\exists = [\![\phi_\exists]\!]$ is the set of nodes in which Éloïse can enforce that Abélard gets stuck, i.e. that for any sequence of moves, Éloïse always can move when an Éloïse-node is reached. Since all plays in finite games are finite, a least fixpoint suffices to express the property that no play gets stuck in an Éloïse-node. In Section 4.3.6 below, we extend the specification of regions in games by means of μ-calculus formulas to infinite games (such as parity games).

2.5 Finite Satisfiability Games for Basic Modal Logic

We now introduce finite satisfiability games for modal formulas, i.e. μ-calculus formulas which contain neither fixpoint operators nor fixpoint variables. For brevity we restrict our attention to formulas without atoms and with $|A| = 1$, i.e. with just a single pair of modalities $\Diamond$ and $\Box$, and call such formulas *basic modal formulas*.

Definition 2.5.1 (Finite satisfiability games). Let ψ be a basic modal formula and let $\mathsf{Cl}(\psi) = \{\phi \mid \phi \leq \psi\}$ denote the *closure* (i.e. the set of all subformulas) of ψ, noting $|\mathsf{Cl}(\psi)| \leq |\psi|$. We define the *finite satisfiability game* $\mathsf{G}(\psi) = (V, E)$ by putting $V = V_\exists \cup V_\forall$ where $V_\forall = \mathcal{P}(\mathsf{Cl}(\psi))$ and $V_\exists = \mathcal{P}(\mathsf{Cl}(\psi)) \times \{(\bot), (\wedge), (\vee), (\Diamond)\} \times \mathsf{Cl}(\psi)$ and by putting

$$\begin{aligned} E(U) = &\{(U, (\bot), \bot) \mid \bot \in U\} \cup \{(U, (\wedge), \psi_1 \wedge \psi_2) \mid \psi_1 \wedge \psi_2 \in U\} \cup \\ &\{(U, (\vee), \psi_1 \vee \psi_2) \mid \psi_1 \vee \psi_2 \in U\} \cup \{(U, (\Diamond), \Diamond\psi_1) \mid \Diamond\psi_1 \in U\}, \end{aligned}$$

for $U \in V_\forall$ and

$$E(U, R, \psi) = \begin{cases} \emptyset & \text{if } R = (\bot), \psi = \bot \\ \{(U \setminus \{\psi_1 \wedge \psi_2\}) \cup \{\psi_1, \psi_2\}\} & \text{if } R = (\wedge), \psi = \psi_1 \wedge \psi_2 \\ \{(U \setminus \{\psi_1 \vee \psi_2\}) \cup \{\psi_i\} \mid i \in \{1, 2\}\} & \text{if } R = (\vee), \psi = \psi_1 \vee \psi_2 \\ \{\{\psi \mid \Box\psi \in U\} \cup \{\phi_1\}\} & \text{if } R = (\Diamond), \psi = \Diamond\psi_1 \end{cases}$$

for $(U, R, \psi) \in V_\exists$.

Player Éloïse wins an Abélard-node U in $\mathsf{G}(\psi)$ if for any sequence of rules from $\{(\bot), (\wedge), (\vee), (\Diamond)\}$ that Abélard chooses to repeatedly apply to U, Éloïse always can choose a conclusion. In particular this means that Éloïse can avoid the situation that Abélard can apply the $(\bot)$-rule. Then Abélard cannot force plays to reach nodes $U \subseteq \mathsf{Cl}(\psi)$ with $\bot \in U$. Thus Éloïse wins the node $\{\psi\}$ in $\mathsf{G}(\psi)$ if and only if there is a standard tableau for ψ.

Fact 2.5.2. Let ψ be a basic modal formula and let $\mathsf{G}(\psi)$ be defined as above. Then Éloïse wins the node $\{\psi\}$ in $\mathsf{G}(\psi)$ if and only if ψ is satisfiable and we have $|W| \leq 5n \cdot 2^n \in 2^{\mathcal{O}(n)}$, where $n = |\psi|$.

Recalling the specification of winning regions of finite games, we have

Fact 2.5.3. A basic modal formula ψ is satisfiable if and only if $\{\psi\} \in [\![\phi_\exists]\!]$ in $\mathsf{G}(\psi)$.

Remark 2.5.4. Basic modal formulas ψ can be seen as finite *tracking automata* with single subformulas of ψ as states, where input words encode particular paths through the syntax tree of ψ (for more information, see Definition 5.2.21); in these finite tracking automata, conjunction states are nondeterministic and $\bot$ is the only accepting state. They can be determinized by means of the powerset construction for finite automata. The finite satisfiability games from Definition 2.5.1 are played (essentially) over the carrier of the determinized and then complemented tracking automaton. In Section 5.2.3 below we extend the construction of satisfiability games via tracking automata to make use of automata on *infinite* words. The resulting games can be used to decide the satisfiability of (coalgebraic) μ-calculus formulas.

2.6 *T*-Coalgebras

We now recall some basic notions from category theory (see e.g. [33]) and universal coalgebra (see e.g. [43]).

A *category* consists of a collection $\mathbf{C}$ of objects and a collection of morphisms (where the collection of all morphisms between two objects $A, B \in \mathbf{C}$ is denoted by $hom(A,B)_\mathbf{C}$) such that for each object $A \in \mathbf{C}$, there exists the identity morphism id_A and such that the composition of morphisms is associative, i.e. $(h \circ g) \circ f = h \circ (g \circ f)$. One important example of a category is **Set**, i.e. the category that has sets as objects and for which the collection $hom(A,B)$ consists of all functions from set A to set B.

Given two categories, a (covariant) *functor* $T : \mathbf{C} \to \mathbf{D}$ maps objects $A \in \mathbf{C}$ to objects $TA \in \mathbf{D}$ and morphisms $f \in hom(A,B)_\mathbf{C}$ to morphisms $Tf \in hom(TA,TB)_\mathbf{D}$ such that for each object $X \in \mathbf{C}$, $Tid_X = id_{TX}$ and for all morphisms $f : X \to Y$ and $g : Y \to Z$, $F(g \circ f) = F(g) \circ F(f)$. In the case that $\mathbf{C} = \mathbf{D}$, a functor $T : C \to C$ is referred to as an *endofunctor*. A *contravariant* functor $T : \mathbf{C} \to \mathbf{D}$ maps morphisms $f \in hom(A,B)_\mathbf{C}$ to morphisms $Tf \in hom(TB,TA)_\mathbf{D}$ while adhering to the condition that for all morphisms $f : X \to Y$ and $g : Y \to Z$, $F(g \circ f) = F(f) \circ F(g)$. A contravariant functor $T : \mathbf{C} \to \mathbf{D}$ may also be understood as a covariant functor $T : \mathbf{C}^{op} \to \mathbf{D}$ where $\mathbf{C}^{op}$ denotes the dual category of $\mathbf{C}$ (the category $\mathbf{C}^{op}$ is obtained from $\mathbf{C}$ by reversing the direction of all morphisms and reversing the order of composition of morphisms). The *composition* $T \circ T'$ of two compatible functors $T : \mathbf{C} \to \mathbf{D}$ and $T' : \mathbf{D} \to \mathbf{E}$ is defined by $(T \circ T')(A) = T(T'(A))$ for objects $A \in \mathbf{C}$ and by $(T \circ T')(f) = T(T'(f)) : (T \circ T')(A) \to (T \circ T')(B)$ for morphisms $f : A \to B$. As an example, we observe that the powerset construction extends naturally to a **Set**-endofunctor, the so-called *(covariant) powerset functor* with

$\mathcal{P}X = \mathcal{P}(X)$ for each set X and $(\mathcal{P}f)B = f[B] \subseteq Y$ for each function $f : X \to Y$ and each set $B \subseteq X$, where $f[B] = \{y \in Y \mid \exists b \in B.\, f(b) = y\}$ denotes the *image* of B under f. The *contravariant powerset functor* $\mathcal{Q}$ acts like $\mathcal{P}$ on sets but maps functions to the respective *preimage* function, that is, $(\mathcal{Q}f)C = f^{-1}[C] \subseteq X$ for $f : X \to Y$, $C \subseteq Y$, where $f^{-1}[C] = \{x \in X \mid f(x) \in C\}$ denotes the preimage of C under f.

Definition 2.6.1 (T-Coalgebras). Let $\mathbf{C}$ be a category, and let $T : \mathbf{C} \to \mathbf{C}$ be an endofunctor over $\mathbf{C}$. A *T-coalgebra* $\mathcal{A} = (A, \alpha)$ consists of an object A of $\mathbf{C}$ (the *carrier*) and a morphism $\alpha : A \to TA$ (the *transition map*, *observation function* or *coalgebra structure*). For $a \in A$, we refer to $\alpha(a)$ as the *$\mathcal{A}$-observation* of a. A *homomorphism* between two T-coalgebras $\mathcal{A} = (A, \alpha)$ and $\mathcal{B} = (B, \beta)$ is a morphism $h : A \to B$ such that $\beta \circ h = (Th) \circ \alpha$. A T-coalgebra $\mathcal{Z} = (Z, \zeta)$ is called *final* if for each T-coalgebra $\mathcal{A} = (A, \alpha)$, there exists a unique homomorphism $A \to Z$.

Intuitively, the transition map describes the successor states and observations of a state, organized in a data structure given by T. These data encode the observable *behaviour* of a system, and morphisms of coalgebras preserve this behaviour.

3 Satisfiability Checking for Coalgebraic Modal Logic

The notion of coalgebraic logic that is subject to our consideration in this chapter has been introduced and elaborated on in [38,51]: For a fixed functor T, coalgebraic formulas are interpreted over T-coalgebras and their modal semantics is defined by means of so-called *predicate liftings*. This approach allows for the definition of coalgebraic satisfiability games with ensuing generic complexity results (such as the PSPACE-completeness of the satisfiability problems of all rank-1 axiomatizable modal logics [50]). Another way to define logics for coalgebras – by means of so-called relation liftings – was introduced in [35]; we do not pursue this approach here.

3.1 Coalgebraic Modal Logic

We swiftly recall the established foundations of coalgebraic modal logic defined by means of *predicate liftings* [38,51].

Definition 3.1.1 (Predicate lifting). A *modal similarity type* is a set Λ of modal operators $\heartsuit$, each with a finite arity assigned to it. We write $\heartsuit_n$ to indicate that $\heartsuit \in \Lambda$ has arity n. Given a **Set**-endofunctor $T : \mathbf{Set} \to \mathbf{Set}$ and a modal operator $\heartsuit_n$, an n-ary *T-predicate lifting* $[\![\heartsuit]\!]$ is a *natural transformation*

$$[\![\heartsuit]\!] : \mathcal{Q}^n \to \mathcal{Q} \circ T^{op},$$

i.e. $[\![\heartsuit]\!]$ denotes a family of mappings $([\![\heartsuit]\!]_C : (\mathcal{P}C)^n \to \mathcal{P}TC)_{C \in \mathbf{Set}}$ such that the diagram

$$\begin{array}{ccc} (\mathcal{P}C)^n & \xrightarrow{[\![\heartsuit]\!]_C} & \mathcal{P}TC \\ {\scriptstyle (f^{-1})^n}\uparrow & & \uparrow{\scriptstyle Tf^{-1}} \\ (\mathcal{P}B)^n & \xrightarrow{[\![\heartsuit]\!]_B} & \mathcal{P}TB \end{array} \qquad \begin{array}{c} C \\ \downarrow{\scriptstyle f} \\ B \end{array}$$

commutes for each function $f : C \to B$, i.e. $[\![\heartsuit]\!]_C \circ (f^{-1})^n = Tf^{-1} \circ [\![\heartsuit]\!]_B$.

We fix a **Set**-endofunctor $T : \mathbf{Set} \to \mathbf{Set}$ and a modal similarity type Λ such that there is for each $\heartsuit_n \in \Lambda$ a dual modal operator $\overline{\heartsuit}_n \in \Lambda$ with the property that $\overline{\overline{\heartsuit}_n} = \heartsuit_n$. Furthermore, we assume that each $\heartsuit_n \in \Lambda$ comes along with an n-ary T-predicate lifting $[\![\heartsuit]\!]$ such that for all C and $B \subseteq C$, we have $[\![\heartsuit]\!]_C(B) = \overline{[\![\overline{\heartsuit}]\!]_C(\overline{B})}$.

Definition 3.1.2 (Coalgebraic formulas). The set $\mathcal{CML}(\Lambda)$ of *coalgebraic modal formulas* over Λ is defined by the grammar

$$\mathcal{CML}(\Lambda) \ni \psi_1, \ldots, \psi_n ::= \top \mid \bot \mid \psi_1 \wedge \psi_2 \mid \psi_1 \vee \psi_2 \mid \heartsuit(\psi_1, \ldots, \psi_n) \qquad \heartsuit_n \in \Lambda$$

The *size* $|\phi|$ of a coalgebraic formula $\phi \in \mathcal{CML}(\Lambda)$ is just its length over the alphabet $\{\top, \bot, \wedge, \vee\} \cup \Lambda$.

While coalgebraic formulas do not contain explicit negation, the dual of formulas can be defined as an abbreviated notion, as we will see.

Definition 3.1.3 (Coalgebraic satisfaction). Given a T-coalgebra $\mathcal{C} = (C, \xi)$, we refer to $\mathcal{C}$ as a *coalgebra model* (or *model* for short) and define the *satisfaction* of coalgebraic modal formulas $\phi \in \mathcal{CML}(\Lambda)$ by putting, for $x \in C$, $\mathcal{C}, x \models \phi$ if and only if $x \in [\![\phi]\!]_{\mathcal{C}}$ where $[\![\phi]\!]_{\mathcal{C}}$ is the *extension* (or *truth set*) of ϕ in $\mathcal{C}$ and is defined inductively by putting

$$[\![\top]\!]_{\mathcal{C}} = C \qquad [\![\bot]\!]_{\mathcal{C}} = \emptyset$$
$$[\![\psi_1 \wedge \psi_2]\!]_{\mathcal{C}} = [\![\psi_1]\!]_{\mathcal{C}} \cap [\![\psi_2]\!]_{\mathcal{C}} \qquad [\![\psi_1 \vee \psi_2]\!]_{\mathcal{C}} = [\![\psi_1]\!]_{\mathcal{C}} \cup [\![\psi_2]\!]_{\mathcal{C}}$$
$$[\![\heartsuit(\psi_1, \ldots, \psi_n)]\!]_{\mathcal{C}} = \xi^{-1}[[\![\heartsuit]\!]_C([\![\psi_1]\!]_{\mathcal{C}}, \ldots, [\![\psi_n]\!]_{\mathcal{C}})],$$

where $\heartsuit \in \Lambda$ is an n-ary modal operator and $\xi^{-1} : \mathcal{P}(TC) \to \mathcal{P}(C)$ denotes the *preimage function* for ξ and is defined by $\xi^{-1}[A] = \{x \mid \xi(x) \in A\}$ for $A \in \mathcal{P}(TC)$. We write $[\![\phi]\!]$ for $[\![\phi]\!]_{\mathcal{C}}$ if no confusion arises.

For $x \in C$ and $\psi \in \mathcal{CML}(\Lambda)$ we thus have $x \in [\![\heartsuit(\psi_1, \ldots, \psi_n)]\!]_{\mathcal{C}}$ if and only if $\xi(x) \in [\![\heartsuit]\!]_C([\![\psi_1]\!]_{\mathcal{C}}, \ldots, [\![\psi_n]\!]_{\mathcal{C}})$.

Definition 3.1.4. Given a coalgebraic modal formula ψ, we define its dual $\overline{\psi}$ inductively by putting

$$\overline{\top} = \bot \qquad \overline{\bot} = \top$$
$$\overline{\psi_1 \wedge \psi_2} = \overline{\psi_1} \vee \overline{\psi_2} \qquad \overline{\psi_1 \wedge \psi_2} = \overline{\psi_1} \wedge \overline{\psi_2}$$
$$\overline{\heartsuit(\psi_1, \ldots, \psi_n)} = \overline{\heartsuit}\,(\overline{\psi_1}, \ldots, \overline{\psi_n})$$

and then have $x \in [\![\overline{\psi}]\!]$ if and only if $x \notin [\![\psi]\!]$ (for (C, ξ) a T-coalgebra and $x \in C$).

Remark 3.1.5. Propositional atoms can be added to a coalgebraic modal logic by changing the functor T to $\mathcal{P}(P) \times T$ where P denotes a set of propositional atoms and by defining the semantics of atoms $p \in P$ by

$$[\![p]\!]_{\mathcal{C}} = \{x \in C \mid \xi(x) = (Q, y), p \in Q \subseteq P\}$$

where $\mathcal{C} = (C, \xi)$ is a $\mathcal{P}(P) \times T$-coalgebra.

Definition 3.1.6 (Satisfiability, validity). A coalgebraic formula ϕ is *valid* if for each T-coalgebra (C, ξ) and each state $x \in C$, $x \in [\![\phi]\!]$ and ϕ is *satisfiable* if there is a T-coalgebra (C, ξ) and a state $x \in C$, such that $x \in [\![\phi]\!]$.

Fact 3.1.7. A coalgebraic formula ϕ is satisfiable if and only if $\overline{\phi}$ is not valid.

For the most part, we restrict the scope of the technical developments in this work to unary modal operators, noting that the presented constructions and proofs naturally generalize to modal operators of arbitrary finite arity.

Example 3.1.8. We consider sets of predicate liftings for several functors (see e.g. [51]).

1. The least normal modal logic **K** (without atoms) is obtained by using Kripke frames, that is, coalgebras for the covariant powerset functor $\mathcal{P}$, as models. The according modal similarity type contains just the unary modal operators $\Box$ and $\Diamond$, i.e. we put $\Lambda = \{\Box_1, \Diamond_1\}$ and the modal semantics is obtained by defining the predicate liftings

$$[\![\Box]\!]_C(B) = \{A \in \mathcal{P}(C) \mid A \subseteq B\}$$
$$[\![\Diamond]\!]_C(B) = \{A \in \mathcal{P}(C) \mid A \cap B \neq \emptyset\}$$

   for sets B and C with $B \subseteq C$.
2. The modal logic **KD** uses the same modal operators $\Box$ and $\Diamond$ as **K** but is interpreted over *serial* Kripke frames. The coalgebraic semantics of **KD** thus can be defined by putting $T = \mathcal{P}^+$, where $\mathcal{P}^+$ is the *non-empty* powerset functor defined by $\mathcal{P}^+(C) = \{A \in \mathcal{P}(C) \mid A \neq \emptyset\}$ for sets C and $\mathcal{P}^+(f) = \mathcal{P}(f)$ for functions f. The according predicate liftings are defined by

$$[\![\Box]\!]_C(B) = \{A \in \mathcal{P}^+(C) \mid A \subseteq B\}$$
$$[\![\Diamond]\!]_C(B) = \{A \in \mathcal{P}^+(C) \mid A \cap B \neq \emptyset\}$$

   for sets B and C with $B \subseteq C$.
3. Classical modal logic [6] can be modelled by using *neighbourhood frames* [22] as models, that is, coalgebras $\mathcal{C} = (C, \xi)$ for the *neighbourhood functor* $\mathcal{N} = \mathcal{Q} \circ \mathcal{Q}$. We put $\Lambda = \{\Box_1, \Diamond_1\}$ and

$$[\![\Box]\!]_C(B) = \{\alpha \in \mathcal{N}(C) \mid \exists A \in \alpha.\, A \subseteq B\}$$
$$[\![\Diamond]\!]_C(B) = \{\alpha \in \mathcal{N}(C) \mid \forall A \in \alpha.\, B \cap A \neq \emptyset\}$$

   for sets B and C with $B \subseteq C$.
4. The coalgebraic semantics of monotone modal logic **M** [6] is obtained by putting $T = Up\mathcal{P}$ where $Up\mathcal{P}$ denotes the subfunctor of $\mathcal{N}$ that assigns the set of upwards closed subsets of $\mathcal{P}(C)$ to sets C so that $Up\mathcal{P}$-coalgebras $\mathcal{C} = (C, \xi)$ are *monotone neighbourhood frames* [22]. Again we have the modal operators $\Diamond$ and $\Box$ and define the predicate liftings

$$[\![\Box]\!]_C(B) = \{\alpha \in Up\mathcal{P}(C) \mid \exists A \in \alpha.\, A \subseteq B\}$$
$$[\![\Diamond]\!]_C(B) = \{\alpha \in Up\mathcal{P}(C) \mid \forall A \in \alpha.\, B \cap A \neq \emptyset\}$$

   for sets B and C with $B \subseteq C$.

5. Hennessy-Milner logic $\mathcal{HML}$ (also known as multi-modal **K**) may be obtained by using the functor $\mathcal{H}(C) = \mathcal{P}(A \times C)$ provided that A is a set of *actions*. Coalgebras for this functor are *labelled transition systems* with set of labels A. We have modal operators $\Box_a$ and $\Diamond_a$ for each $a \in A$ and their semantics is given by
$$[\![\Box_a]\!]_C(B) = \{S \in \mathcal{H}(C) \mid S_a \subseteq B\}$$
$$[\![\Diamond_a]\!]_C(B) = \{S \in \mathcal{H}(C) \mid S_a \cap B \neq \emptyset\}$$
for sets B and C with $B \subseteq C$, where $S_a = \{x \in C \mid (a, x) \in S\}$.
6. Pauly's coalition logic [41]: Let $N = \{1, \ldots, n\}$ denote a set of *agents*. *Coalitions* D are then subsets of N, i.e. $D \subseteq N$. We define the modal similarity type $\Lambda = \{[D]_1, \langle D\rangle_1 \mid D \subseteq N\}$, i.e. each coalition induces two unary modal operators. A suitable functor (modulo size issues) is given by the definition $\mathcal{G}(C) = \{(S_1, \ldots, S_n, f) \mid \emptyset \neq S_i \in \mathbf{Set}, f : \prod_{i\in N} S_i \to C\}$ so that $\mathcal{G}$ assigns to a set C tuples consisting of n non-empty sets S_i, so-called *strategies* and of an *outcome function* f. Coalgebras for this functor are in one-to-one correspondence to so-called *game frames* [41]. Each coalition D induces sets $S_D = \prod_{i\in D} S_i$ and $S_{\overline{D}} = \prod_{i\in \overline{D}} S_i$ such that for $s_D \in S_D$ and $s_{\overline{D}} \in S_{\overline{D}}$, $(s_D, s_{\overline{D}})$ is an element of $\prod_{i\in N} S_i$. This enables the definition of predicate liftings for $[D]$ and $\langle D\rangle$ by
$$[\![[D]]\!]_C(B) = \{(S_1, \ldots, S_n, f) \in \mathcal{G}(C) \mid \exists s_D \in S_D. \forall s_{\overline{D}} \in S_{\overline{D}}. f(s_D, s_{\overline{D}}) \in B\}$$
$$[\![\langle D\rangle]\!]_C(B) = \{(S_1, \ldots, S_n, f) \in \mathcal{G}(C) \mid \forall s_D \in S_D. \exists s_{\overline{D}} \in S_{\overline{D}}. f(s_D, s_{\overline{D}}) \in B\}$$
for sets B and C with $B \subseteq C$ and $D \subseteq N$. Formulas $[D]\phi$ then express that "coalition D can enforce ϕ" while formulas $\langle D\rangle\phi$ state that "coalition D cannot prevent ϕ".
7. Probabilistic modal logic [32,26]: We put $\Lambda = \{L_p, M_p \mid p \in [0, 1] \cap \mathbb{Q}\}$. The *finite distribution functor* $\mathcal{D}_\omega$ maps sets C to the set of probability distributions over C with finite support. $\mathcal{D}_\omega$-coalgebras are so-called *probabilistic type spaces* with finite branching degree and we define predicate liftings
$$[\![L_p]\!]_C(B) = \{P \in \mathcal{D}_\omega(C) \mid P(B) \geq p\}$$
$$[\![M_p]\!]_C(B) = \{P \in \mathcal{D}_\omega(C) \mid P(\overline{B}) < p\}$$
for sets C and $B \subseteq C$ and for $p \in [0, 1] \cap \mathbb{Q}$, where $P(B) = \sum_{x\in B} P(x)$. Formulas $L_p\phi$ express that "with probability at least p, ϕ holds in the next step".
8. Basic conditional logic **CK** [6]: We have two *binary* modal operators $\Rightarrow$ and $>$ and define the functor $\mathcal{CK}$ by $\mathcal{CK}(C) = \{f : \mathcal{Q}(C) \to \mathcal{P}(C)\}$. It maps a set C to the set of *selection functions* $f : \mathcal{P}(C) \to \mathcal{P}(C)$ so that $\mathcal{CK}$-coalgebras are standard *conditional frames*. The according binary predicate liftings are defined by
$$[\![\Rightarrow]\!]_C(A, B) = \{f \in \mathcal{CK}(C) \mid f(A) \subseteq B\}$$
$$[\![>]\!]_C(A, B) = \{f \in \mathcal{CK}(C) \mid f(A) \cap B \neq \emptyset\}$$
for sets A, B and C with $A \subseteq C$ and $B \subseteq C$. Formulas $\psi \Rightarrow \phi$ then state that "if ψ holds, then *usually* ϕ holds".

Further examples of basic coalgebraic logics may be found in [51] (e.g. majority logic, an extension of graded modal logic) and [52] (e.g. **CKCM**, **CKID**, and **CKCEM**, the extensions of **CK** with the *cautious monotonicity axiom*, the *identity axiom* and the *conditional excluded middle axiom*, respectively; *System S*). Furthermore, new coalgebraic logics may obtained by the modular combination of single coalgebraic logic *features* [8], [47].

3.1.1 One-step Rules

Initially, coalgebraic modal logic was limited to so-called *rank*-1 *logics*, that is, logics that can be axiomatized by formulas in which all modal operators have nesting depth exactly 1 [46]. It has since been extended to more general non-iterative logics [49] and, to some degree, to iterative logics, axiomatized by formulas with nested modalities [48]. In the rank-1 setting, it has been shown in [46] that all logics can be axiomatized by so-called *one-step rules* (ϕ/ψ), where ϕ is purely propositional and ψ is a clause over formulas of the form $\heartsuit(a_1, \ldots, a_n)$, where the a_i are propositional variables. In [39], *one-step sequent rules* have been used to obtain a generic sequent calculus for coalgebraic modal logics. The satisfiability games for coalgebraic (fixpoint) logics that we introduce in Sections 3.2 and 5.2 use tableau rules instead and hence rely on the dual concept of so-called *one-step tableau rules* [16] to encode the syntactic properties that characterize the satisfiability of coalgebraic formulas.

Definition 3.1.9 ((One-step) Tableau rules). Let B be a set of *rule variables*, let $\Lambda(B) = \{\heartsuit b \mid b \in B, \heartsuit \in \Lambda\}$ denote the set of *one-step formulas over* B and let $\mathsf{Prop}(B)$ denote the set of propositional formulas over B. A $\mathcal{CML}(\Lambda)$*-substitution* $\sigma : B \to \mathcal{CML}(\Lambda)$ is a function that maps rule variables to coalgebraic formulas. Then a *propositional tableau rule* $R = (\Gamma_0/\Sigma)$ consists of the *premise*, a set $\Gamma_0 \subseteq \mathsf{Prop}(B)$ of propositional formulas over B, and the *conclusion*, a set $\Sigma = \{\Gamma_1, \ldots, \Gamma_l\} \subseteq \mathcal{P}(B)$. A *one-step tableau rule* $R = (\Gamma_0/\Sigma)$ has a set $\Gamma_0 \subseteq \Lambda(B)$ of one-step formulas as premise and a set $\Sigma = \{\Gamma_1, \ldots, \Gamma_l\} \subseteq \mathcal{P}(B)$ as conclusion. Rules are *clean* if their premises mention every variable at most once.

Definition 3.1.10 (Rule applications). Let $\mathcal{R}$ be a set of tableau rules and let $R = (\Gamma_0/\Gamma_1, \ldots, \Gamma_l) \in \mathcal{R}$ be a tableau rule. A $\mathcal{CML}(\Lambda)$*-substitution* $\sigma : B \to \mathcal{CML}(\Lambda)$ induces the *rule application* $(\Gamma_0\sigma/\Gamma_1\sigma, \ldots, \Gamma_l\sigma)$ (sometimes denoted by just the data (R, σ)). Given a set of formulas $\Gamma \subseteq \mathcal{CML}(\Lambda)$, the rule application $(\Gamma_0\sigma/\Gamma_1\sigma, \ldots, \Gamma_l\sigma)$ *matches* Γ if $\Gamma_0\sigma \subseteq \Gamma$. For a set $\mathcal{R}$ of (propositional or one-step) tableau rules, we define the *list*

$$\mathcal{R}(\Gamma) = ((R_1, \sigma_1), \ldots, (R_n, \sigma_n))$$

of rule applications that match Γ, where $R_i = (\Gamma_0/\Gamma_1, \ldots, \Gamma_l) \in \mathcal{R}$ and $\Gamma_0\sigma_i \subseteq \Gamma$ for $1 \leq i \leq n$, and the set of conclusions of rule applications to Γ by

$$\mathsf{Cn}(\Gamma) = \{\{\Gamma_1\sigma, \ldots, \Gamma_l\sigma\} \mid ((\Gamma_0/\Gamma_1, \ldots, \Gamma_l), \sigma) \in \mathcal{R}(\Gamma)\}.$$

Definition 3.1.11 (Propositional tableau rules). We fix a set of rule variables B and define the set $\mathcal{R}_{prop}$ of *propositional tableau rules* as follows (for $a, b \in B$):

$$(\bot)\frac{\bot}{} \qquad (\wedge)\frac{a \wedge b}{a, b} \qquad (\vee)\frac{a \vee b}{a \qquad b}$$

In the context of sequent calculi [39] and tableau systems [16], the concept of one-step soundness and completeness relates provability and satisfiability, respectively, to the syntactic conditions that one-step rules impose. We define the tableau variants of one-step soundness and completeness.

Definition 3.1.12 (One-step tableau soundness / completeness). Let T be a **Set**-endofunctor and let C be a set. Given a set $\Gamma \subseteq B$ of rule variables, we define the extension $[\![\Gamma]\!]_C\tau$ of Γ under a $\mathcal{P}(C)$-substitution $\tau : B \to \mathcal{P}(C)$ by $[\![\Gamma]\!]_C\tau = \bigcap_{b\in\Gamma} \tau(b)$. The *one-step extension* of a set $\Gamma \subseteq \Lambda(B)$ of one-step formulas under τ is defined by $[\![\Gamma]\!]_{TC}\tau = \bigcap_{\heartsuit a\in\Gamma}[\![\heartsuit]\!]_C(\tau(a))$. For $\Gamma \subseteq \Lambda(B)$ we thus have $[\![\Gamma]\!]_{TC}\tau \subseteq TC$.

A set of one-step tableau rules $\mathcal{R}$ is *one-step tableau complete* if for all sets C, all $\mathcal{P}(C)$-substitutions $\tau : B \to \mathcal{P}(C)$ and all $R = (\Gamma_0/\Gamma_1, \ldots, \Gamma_l) \in \mathcal{R}$,

$$\text{if } [\![\Gamma_i]\!]_C\tau \neq \emptyset \text{ for some } 1 \leq i \leq l, \text{ then } [\![\Gamma_0]\!]_{TC}\tau \neq \emptyset.$$

A single one-step tableau rule $R = (\Gamma_0/\Gamma_1, \ldots, \Gamma_l)$ is *one-step tableau sound* if there is, for all sets C, all sets $\Gamma \subseteq \Lambda(B)$ of one-step formulas, all $\mathcal{P}(C)$-substitutions $\tau : B \to \mathcal{P}(C)$ with $[\![\Gamma]\!]_{TC}\tau \neq \emptyset$, and all renamings $\sigma : B \to B$ with $\Gamma_0\sigma \subseteq \Gamma$, some $1 \leq i \leq l$ such that $[\![\Gamma_i\sigma]\!]_C\tau \neq \emptyset$. A set of one-step tableau rules $\mathcal{R}$ is one-step tableau sound if each rule $R \in \mathcal{R}$ is one-step tableau sound.

Example 3.1.13. We consider one-step tableau sound and one-step tableau complete sets of tableau rules for some of the coalgebraic logics detailed in Example 3.1.8. The sets of one-step rules for **M**, **K**, **KD** and $\mathcal{HML}$ are as follows:

$$\textbf{(M)} \quad \frac{\Box a, \Diamond b}{a, b}$$

$$\textbf{(K)} \quad \frac{\Box b_1, \ldots, \Box b_n, \Diamond b}{b_1, \ldots, b_n, b}$$

$$\textbf{(KD)} \quad \frac{\Box b_1, \ldots, \Box b_n, \Diamond b}{b_1, \ldots, b_n, b} \qquad \frac{\Box b_1, \ldots, \Box b_n}{b_1, \ldots, b_n,}$$

$$(\mathcal{HML}) \quad \frac{\Box_a b_1, \ldots, \Box_a b_n, \Diamond_a b}{b_1, \ldots, b_n, b} \qquad a \in A$$

The rule scheme for probabilistic modal logic is more involved. For a given formula ϕ_i and $r_i \in \mathbb{Z}$ and for all $i \in I$ and $k \in \mathbb{Z}$ we define

$$\sum_{i \in I} r_i \phi_i \geq k \equiv \bigwedge_{J \subseteq I, r(J) < k} (\bigvee_{j \in J} \phi_j, \bigvee_{j \notin J} \neg \phi_j),$$

where $r(J) = \sum_{j \in J} r_j$. Furthermore, we use $\sum a_i \leq \sum b_j$ as an abbreviation for $\sum b_j - \sum a_i \geq 0$. The rule scheme for probabilistic modal logic then is

$$(\mathbf{P}) \quad \frac{\bigwedge_{i=0}^{n} \mathrm{sgn}(r_i) L_{p_i} a_i}{\sum_{i=0}^{n} r_i a_i \sqsupset \sum_{i=0}^{n} r_i p_i},$$

where $n \geq 0$, $r_0, \ldots, r_n \in \mathbb{Z} \setminus \{0\}$.

All sets of rules from this example (or the respective dual sets of sequent rules) have been shown to be one-step (sequent) sound and complete ([39,16,40]).

3.2 Satisfiability Games for Coalgebraic Modal Logic

We adapt the finite satisfiability games from Definition 2.5.1 to the more general setting of basic coalgebraic modal logic. To this end we fix a functor T, a modal similarity type Λ (along with a predicate lifting for each modal operator) and a one-step sound and one-step complete set of one-step tableau rules $\mathcal{R}_m$ and put $\mathcal{R} = \mathcal{R}_m \cup \mathcal{R}_{prop}$.

Definition 3.2.1 (Satisfiability games for coalgebraic modal logic). Let ψ be a coalgebraic modal formula. We define the closure of ψ by

$$\mathsf{Cl}(\psi) = \{\phi \in \mathcal{CML}(\Lambda) \mid \phi \text{ is a subformula of } \psi\},$$

noting $|\mathsf{Cl}(\psi)| \leq |\psi|$. Furthermore, we assume a set $\mathsf{code}(\psi)$ that contains for each rule application (R, σ) with $R \in \mathcal{R}$ and $\sigma : B \to \mathsf{Cl}(\psi)$ a code $\mathsf{code}(R, \sigma)$ (also denoted just by (R, σ), if no confusion arises) that identifies the rule application. The satisfiability game $\mathsf{G}(\psi) = (V, E)$ for ψ is a finite two-player game defined by putting $V = V_\exists \cup V_\forall$ where $V_\forall = \mathcal{P}(\mathsf{Cl}(\psi))$ and $V_\exists = \mathcal{P}(\mathsf{Cl}(\psi)) \times \mathsf{code}(\psi)$ and by putting

$$E(U) = \{(U, (R, \sigma)) \mid (R, \sigma) \in \mathcal{R}(U)\}$$

for $U \in V_\forall$ and

$$E(U, (R, \sigma)) = \{\Gamma_j \sigma \mid R = (\Gamma_0 / \Gamma_1, \ldots, \Gamma_l), 1 \leq i \leq l\}$$

for $(U, (R, \sigma)) \in V_\exists$.

Player Éloïse wins an Abélard-node U in $\mathsf{G}(\psi)$ if for any sequence of matching applications of rules from $\mathcal{R}$ that Abélard chooses to apply repeatedly, starting at U, Éloïse always can choose conclusion nodes. As the rule applications reduce the sizes of formulas, Abélard can only make finitely many choices and if Éloïse wins a play, then Abélard eventually cannot apply any more rules.

Lemma 3.2.2. *Let ψ be a coalgebraic modal formula and let $\mathsf{G}(\psi)$ be defined as above. Then Éloïse wins the node $\{\psi\}$ in $\mathsf{G}(\psi)$ if and only if ψ is satisfiable.*

Winning strategies for Éloïse in the $\mathsf{G}(\psi)$ essentially define tableaux for ψ and a model can be built over (the states of) tableaux. For now, we refrain from introducing the necessary notation and proving the Lemma in detail, as it is a direct consequence of the more general results that we prove in Chapter 5 below. In [39], an equivalent result (regarding the validity of formulas) was shown for a generic coalgebraic sequent calculus.

Again the winning region of Éloïse in $\mathsf{G}(\psi)$ can be specified by the μ-calculus formula

$$\phi_\exists = \mu X.\ ((\forall \wedge \Box_E X) \vee (\exists \wedge \Diamond_E X)).$$

Definition 3.2.3 (PSpace-tractability, ExpTime-tractability [39,51]). Let $\mathcal{R}$ be a set of one-step tableau rules over a set of rule variables B. Let P denote an alphabet and let $\mathsf{code} : \mathcal{R} \times \{\sigma : B \to \mathcal{CML}(\Lambda)\} \to P^*$ be a coding function, assigning a string $\mathsf{code}(R, \sigma)$ over P to every rule application (R, σ). Then $\mathcal{R}$ is PSpace-*tractable* (ExpTime-*tractable*) if there exists a polynomial function p such that for all sets $\Gamma \subseteq \mathcal{CML}(\Lambda)$ and all rule applications $(R, \sigma) \in \mathcal{R}(\Gamma)$, $|\mathsf{code}(R, \sigma)| \leq p(|\Gamma|)$ (where $|w|$ denotes the length of the string w) and if it can be decided in NP (ExpTime),

1. whether $c = \mathsf{code}(R, \sigma)$ for a given word $c \in P^*$ and
2. whether $(R, \sigma) \in \mathcal{R}(\Gamma)$ for a given rule application (R, σ).

Under the assumption of PSpace-tractability, the generic sequent calculus from [40] has been shown to yield a decision procedure that realizes validity checking for coalgebraic logics in PSpace. We recover this result for the satisfiability problem, using the satisfiability games that we defined in this section.

Theorem 3.2.4. *The satisfiability problem for coalgebraic modal logics with* PSpace*-tractable one-step sound and one-step complete sets of tableau rules is contained in* PSpace.

Proof. Let ψ be a coalgebraic modal formula and let $\mathcal{R}$ be a PSpace-tractable one-step sound and one-step complete set of tableau rules. The PSpace-tractability of $\mathcal{R}$ implies that the moves that Abélard or Éloïse have at a given node in the satisfiability game can be computed in NP. The set $\phi_\exists$ can thus be computed in time exponential in $n = |\psi|$; as runs in $\mathsf{G}(\psi)$ are of length at most n and it suffices by finiteness of plays to store a single play at a time when computing the winner of $\mathsf{G}(\psi)$, this computation can be implemented using space polynomial in n (at most n nodes have to be visited before a play ends and each node can be store in space polynomial in n). □

4 Automata on Infinite Words and Infinite Games

In this chapter, we introduce various types of automata on infinite words and infinite games (see e.g. [21] for an overview) along with determinization methods for several types of automata. Some of the determinization procedures are novel and rely on syntactic properties such as *limit-linearity* and *limit-determinism* of the input automata to yield asymptotically smaller determinized automata than e.g. Miyano/Hayashi construction [34] that works for unrestricted Co-Büchi automata or the Safra/Piterman construction [42,44] that works for unrestricted Büchi automata. In Section 4.3, we show that several algorithmic problems for automata and games (such as the *emptiness problem* for automata or computing the winning regions in two-player games) can be seen as model checking problems for the relational μ-calculus.

4.1 Automata on Infinite Words

First, we define deterministic and nondeterministic variants of several types of automata on infinite words.

Definition 4.1.1 (Automata on infinite words). A finite *Büchi automaton* is a finite automaton $\mathsf{A} = (V, \Sigma, \Delta, v_0, F)$ with set $F \subseteq \Delta$ of accepting *transitions* that runs on *infinite* words. We use accepting transitions instead of accepting states to obtain slightly smaller automata in several constructions; this is standard in recent work (see e.g. [12]) and automata with accepting transitions can be transformed to equivalent automata with accepting states with linear blow-up in the number of *priorities* (see below). Given an infinite word $w = w_0 w_1 \ldots \in \Sigma^\omega$, an *infinite run* of A on w is a sequence $\rho = v_0 v_1 \ldots \in V^\omega$ such that for all $i \geq 0$, $v_{i+1} \in \Delta(v_i, w_i)$; we denote the set of all infinite runs of the automaton A on a word w starting at a state v by $\mathsf{run}(\mathsf{A}, v, w)$ (or just by $\mathsf{run}(\mathsf{A}, w)$ if $v = v_0$). Given an infinite run $\rho \in \mathsf{run}(\mathsf{A}, w)$, we define the sequence $\mathsf{trans}(\rho)$ of its transitions by

$$(\mathsf{trans}(\rho))(i) = (\rho(i), w(i), \rho(i+1))$$

for $i \geq 0$. We define the projections $\pi_1, \pi_3 : \Delta \to V$, $\pi_2 : \Delta \to \Sigma$ by

$$\pi_1(v, a, w) = v \qquad \pi_2(v, a, w) = a \qquad \pi_3(v, a, w) = w$$

for $(v, a, w) \in \Delta$. The language $L(\mathsf{A})$ that is recognized by a Büchi automaton A is

$$L(\mathsf{A}) = \{w \in \Sigma^\omega \mid \exists \rho \in \mathsf{run}(\mathsf{A}, w).\ \mathsf{Inf}(\mathsf{trans}(\rho)) \cap F \neq \emptyset\},$$

where, for all sets C and all functions $f : \mathbb{N} \to C$,

$$\mathsf{Inf}(f) = \{c \in C \mid \forall i.\, \exists j \geq i.\, f(j) = c\}$$

denotes those elements from C that occur infinitely often in f; thus Büchi automata accept exactly those words for which there is a run that uses at least one accepting transition infinitely often. Similarly, *Co-Büchi automata* $\mathsf{A} = (V, \Sigma, \Delta, v_0, F)$ accept infinite words for which there is a run in which *only* accepting transitions are used infinitely often; formally, the Co-Büchi automaton A accepts the language

$$L(\mathsf{A}) = \{w \in \Sigma^\omega \mid \exists \rho \in \mathsf{run}(\mathsf{A}, w).\ \mathsf{Inf}(\mathsf{trans}(\rho)) \subseteq F\},$$

i.e. every accepting run uses only transitions from F from some point on. *Parity automata* $\mathsf{A} = (V, \Sigma, \Delta, v_0, \alpha)$ generalize both Büchi and Co-Büchi automata by allowing an acceptance function $\alpha : \Delta \to \{0, 1, \ldots, k-1\}$, where k is the number of *priorities* of the automaton; hence α assigns a priority $\alpha(t)$ to each transition $t \in \Delta$. The *index* $\mathsf{idx}(\mathsf{A}) = \max\{\alpha(t) \mid t \in \Delta\} + 1$ of a parity automaton A is the span of its priorities, e.g. the index of Co-Büchi automata with at least one non-accepting state is 2. The function α partitions Δ into disjoint sets

$$\Delta_i := \{t \in \Delta \mid \alpha(t) = i\},$$

for $0 \leq i < k$. We also define the sets

$$\Delta_{\leq i} = \{t \in \Delta \mid \alpha(t) \leq i\}.$$

The acceptance condition of parity automata requires the existence of a run for which the highest priority that occurs infinitely often is even; that is, for a parity automaton A, we put

$$L(\mathsf{A}) = \{w \in \Sigma^\omega \mid \exists \rho \in \mathsf{run}(\mathsf{A}, w).\ \max(\mathsf{Inf}(\alpha \circ \mathsf{trans}(\rho))) \text{ is even}\}.$$

Thus a parity automaton with the two priorities 0 and 1 is a Co-Büchi automaton (with $F = \Delta_0$, $\overline{F} = \Delta_1$) while a parity automaton with the two priorities 1 and 2 is a Büchi automaton (with $F = \Delta_2$, $\overline{F} = \Delta_1$).

The defined automata on infinite words are *existential* in the sense that they accept words for which an accepting run *exists*. However, sometimes it is convenient to use *universal automata*, that is, automata that accept words for which *all* runs are accepting. Furthermore, the defined PA are *maximal* in the sense that they accept runs in which the maximal priority that occurs infinitely often is even; the dual concept of *minimal priority automata* is obtained by the acceptance condition that the minimal priority that occurs infinitely often is even.

Definition 4.1.2. Given two states $v, w \in V$ and a letter $a \in \Sigma$, we refer to a transition (v, a, w) as an *a-transition* and to w as an *a-successor* of v. Given a state $v \in V$, sets of transitions $\beta, \gamma \subseteq \Delta$, a transition $t \in \Delta$ and a letter $a \in \Sigma$, we put

$$\mathsf{Id}_\gamma = \{(u, b, w) \in \gamma \mid w = u\} \qquad \gamma|_a = \{(u, b, w) \in \gamma \mid a = b\}$$
$$\gamma|_{v,a} = \{(u, b, w) \in \gamma \mid u = v, b = a\} \qquad \gamma|_{t,a} = \{(u, b, w) \in \gamma \mid u = \pi_3(t), b = a\}$$
$$\gamma|_t = \bigcup_{a \in \Sigma} \gamma|_{t,a} \qquad \gamma|_{\beta,a} = \bigcup_{t \in \beta} \gamma|_{t,a}$$

We use ϵ to denote the empty word. For a finite word $w = a_0 \dots a_n$, we recursively define $\gamma|_{\beta,\epsilon} = \beta$ and $\gamma|_{\beta,w} = \gamma|_{(\gamma|_{\beta,a_0}),a_1\dots a_n}$, the latter being the set of transitions that are reachable starting from transitions from β via the word w while using only transitions from γ. Furthermore, we define the set of transitions that are *reachable* from a set of transitions $\beta \subseteq \Delta$ by using transitions from $\gamma \subseteq \Delta$ as

$$\mathsf{reach}_\gamma(\beta) = \bigcup_{w \in \Sigma^*} \gamma|_{\beta,w}.$$

We have $\mathsf{reach}_\gamma(\beta) \subseteq \gamma$. If $\gamma = \Delta$, then we omit the subscripts. Given a function $f : \mathbb{N} \to C$ for some set C, we denote the sequence $f(0) \dots f(i)$ by $\mathsf{pre}(f, i)$ and the sequence $f(i+1)f(i+2)\dots$ by $\mathsf{post}(f, i)$. The *concatenation* $f; g$ of functions $f : \{0, \dots, n\} \to C$ and $g : \mathbb{N} \to C$ is defined by

$$(f; g)(i) = \begin{cases} f(i) & i \leq n \\ g(i) & i > n \end{cases}$$

Then we have for all $f : \mathbb{N} \to C$ and $i \in \mathbb{N}$ that $\mathsf{pre}(f, i); \mathsf{post}(f, i) = f$. These generic notions instantiate to infinite words $w : \mathbb{N} \to \Sigma$, runs $\sigma : \mathbb{N} \to V$ and sequences of transitions $\mathsf{trans}(\sigma) : \mathbb{N} \to \Delta$. A set $\gamma \subseteq \Delta$ of transitions is *deterministic* if for all $a \in \Sigma$, $\gamma|_a$ is a partial function. An automaton A on infinite words is *deterministic* if Δ is deterministic; the transition relation in deterministic automata hence is a partial function $\Delta : V \times \Sigma \rightharpoonup V$ (since such automata can be transformed to equivalent automata with total transition function, this definition suffices for purposes of determinization). For deterministic automata, we usually write δ instead of Δ.

We use the abbreviations NCBA, DCBA, NBA, DBA, NPA and DPA to denote the (non)deterministic variants of Co-Büchi, Büchi and parity automata, respectively. For Büchi automata, we assume w.l.o.g. that every accepting transition is part of a cycle (otherwise it cannot occur infinitely often in any run); for Co-Büchi automata we assume w.l.o.g. that every accepting transition is part of an F-cycle, that is, a cycle formed exclusively by transitions from F (otherwise the transition cannot be part of an accepting run). For general PA, we assume that for even l, every transition $t \in \Delta_l$ is part of a $\Delta_{\leq l}$-cycle and can be part of an accepting run. It is a standard result (see e.g. [21]) that NBA, DPA, and NPA all are equally expressive and that such automata recognize a language L if and only if L is an *ω-regular language*; NCBA, DCBA and DBA however all are strictly less expressive than NBA.

4.1.1 Limit Properties of Automata

We define the concepts of limit-determinism, limit-linearity and limit-stationarity of automata. In such automata, all accepting runs are deterministic, linear or stationary *from some point on.*

Limit-determinism of automata is defined as a semantic property.

Definition 4.1.3 (Limit-deterministic PA). A PA $\mathsf{A} = (V, \Sigma, \Delta, v_0, \alpha)$ is *limit-deterministic* if there is, for each word w and each accepting run $\rho \in \mathsf{A}(w)$, a number i such that for all $j \geq i$,

$$(\Delta_{\leq l})|_{\rho(j),w(j)} = \{(\mathsf{trans}(\rho))(j)\},$$

where $l = \max(\mathsf{Inf}(\alpha \circ \mathsf{trans}(\rho)))$ is even.

If A is a BA, then we have $\max(\mathsf{Inf}(\alpha \circ \mathsf{trans}(\rho))) = 2$ for every accepting run ρ; as $\Delta_{\leq 2} = \Delta$ in BA, the constraint in the above definition instantiates to requiring the existence of a number i such that for all $j \geq i$, $\Delta|_{\rho(j),w(j)} = \{(\mathsf{trans}(\rho))(j)\}$. For CBA we have $\max(\mathsf{Inf}(\alpha \circ \mathsf{trans}(\rho))) = 0$ for every accepting run ρ; as $\Delta_{\leq 0} = F$, the constraint requires a number i such that for all $j \geq i$, $F|_{\rho(j),w(j)} = \{(\mathsf{trans}(\rho))(j)\}$.

Definition 4.1.4 (Compartments). Given a PA $\mathsf{A} = (V, \Sigma, \Delta, u_0, \alpha)$ with k priorities, and an even number $l \leq k$, the *l-compartment* $C_l(t)$ of a transition $t \in \Delta_l$ is the set $\mathsf{reach}_{\Delta_{\leq l}}(t)$, that is, the set of transitions that are reachable from $\pi_3(t)$ without using transitions with priority higher than l. If l is irrelevant, we refer to l-compartments just as compartments. The *size* of a compartment C is just $|\pi_3[C]|$.

Note that the union of all l-compartments is $\mathsf{reach}_{\Delta_{\leq l}}(\Delta_l)$. By definition, every accepting run ρ of a parity automaton eventually stays inside one of the compartments of the automaton, say C, i.e. we have $\mathsf{Inf}(\mathsf{trans}(\rho)) \subseteq C$.

Fact 4.1.5. *Let $q \leq |V|$ be the number of states on which the largest compartment in a PA A is based. Then every accepting run of A visits at most q states infinitely often.*

Each compartment in the *tracking automata* that we define in Chapter 5 consists of formulas that belong to a single least fixpoint literal, having the intuition that accepting runs of these automata stay in the compartment for some least fixpoint from some point on, i.e. that the respective least fixpoint is unfolded infinitely often.

Compartments allow for a syntactic characterization of limit-determinism:

Fact 4.1.6. *A PA is limit-deterministic if and only if all its compartments are deterministic.*

Proof. Let A be a limit-deterministic PA with alphabet Σ and acceptance function α, let C be an l-compartment for some even l, let $v \in C$ and let $a \in \Sigma$. We have to show that $|\{u \mid (v, a, u) \in C\}| \leq 1$. As all transitions $(v, a, u) \in C$ are part of a $\Delta_{\leq l}$-cycle, there is a word w and an accepting run $\rho \in \mathsf{run}(\mathsf{A}, w))$ such that $\mathsf{trans}(\rho)$ contains (v, a, u) infinitely often. By limit-determinism of A, there is some i such that for all $j \geq i$, $(\Delta_{\leq l})|_{\rho(j),w(j)} = \{(\rho(j), w(j), \rho(j+1))\}$. Considering some position $j' \geq j$ with $\rho(j') = v$, $w(j') = a$, we have $|\{u \mid (v, a, u) \in C\}| \leq |(\Delta_{\leq l})|_{\rho(j'),w(j')}| = |\{(v, w(j'), \rho(j'+1))\}| =\leq 1$, as required. For the converse direction, let all compartments in A be deterministic, let w be a word and let $\rho \in \mathsf{run}(\mathsf{A}, w)$ be accepting so that $l = \max(\mathrm{Inf}(\alpha \circ \mathsf{trans}(\rho)))$ is even

and there is some position i such that for all $j \geq i$, we have $\alpha((\mathsf{trans}(\rho))(j)) \leq l$. From i on, ρ stays in the l-compartment $C := C_l(\mathsf{trans}(\rho(i)))$, i.e. we have that for all $j' \geq i$, $\mathsf{trans}(\rho(j')) \in C$. By assumption, C is deterministic so that we have for all $j \geq i$ that $|\{u \mid (\rho(j), w(j), u) \in C\}| \leq 1$ and hence $(\Delta_{\leq l})|_{\rho(j),w(j)} = \{\mathsf{trans}(\rho(j))\}$, as required. □

Fact 4.1.6 specializes to BA where we have $\Delta_0 = \emptyset$, $\Delta_{\leq 2} = \Delta$ and $\Delta_2 = F$. The union of all 0-compartments is the empty set and that of all 2-compartments is $\mathsf{reach}(F)$; thus a BA is limit-deterministic if and only if $\mathsf{reach}(F)$ is deterministic (see e.g. [12]). Such Büchi automata (with accepting *states* instead of transitions) are also called *semi-deterministic* [9]. For CBA, there is just one even priority 0 and we have $\Delta_0 = \Delta_{\leq 0} = F$; the union of all 0-compartments is $\mathsf{reach}_F(F) = F$. Thus a CBA is limit-deterministic if and only if F is deterministic.

Corollary 4.1.7. *It is decidable in polynomial time whether a given automaton is limit-deterministic.*

We define the following related semantic notion for Co-Büchi automata:

Definition 4.1.8 (Limit-linear CBA). Let $\mathsf{A} = (V, \Sigma, \Delta, v_0, F)$ be a CBA. A set $\gamma \subseteq \Delta$ of transitions is *linear* if for all $t \in \gamma$, $|\pi_3[\gamma|_t \setminus \mathsf{Id}_\gamma]| = 1$. A letter $a \in \Sigma$ is a *progressing letter* (in A) if there is no transition $(v, a, v) \in \Delta$. A *synchronizing state* is a state $v \in \pi_3[F]$ such that there is some progressing letter $a \in \Sigma$ such that v has an a-successor. A pair of letters $(a_1, a_2) \in \Sigma^2$ (referred to as *choice letters*) with $a_1 \neq a_2$ is a *choice pair* (in A) if there is exactly one state $v \in V$ that has an a_1- or a_2-successor and we have

$$\Delta(v, a_1) = \{u_1\} \qquad\qquad \Delta(v, a_2) = \{u_2\}$$

with $u_1 \neq v \neq u_2$, and for all $b \in \Sigma$ such that $a_1 \neq b \neq a_2$, there is no transition $(v, b, w) \in \Delta$ with $w \neq v$. For all transitions $t \in \Delta \setminus F$ and $t' = (\pi_3(t), a, v) \in F$, t' is an *entry transition* for its compartment $C(t')$. The CBA A is *limit-linear* if and only if F is linear, every compartment C of A contains at least one synchronizing state and at most one entry transition, denoted by t_C if it exists, and for all compartments C with entry transition and all transitions $(u, a, \pi_1(t_C)) \in C$ with $u \neq \pi_1(t_C)$, a is progressing, i.e. the entry transition of C can only be reached within C via a transition for a progressing letter (or via a trivial loop). Furthermore, we require that there is, for each $v \in \pi_3[F]$ and all non-progressing $a \in \Sigma$ such that a is not a choice letter, some transition $(v, a, w) \in F$. A *limit-stationary* CBA is a limit-linear CBA in which each compartment contains exactly one synchronizing state.

In the tracking automata defined in Section 5.2.2 below, individual compartments correspond to individual least fixpoint literals, the progressing letters are such letters that encode applications of modal rules, the choice letters are letters that encode applications of the disjunction rule, and entry transitions unfold fixpoint literals.

Definition 4.1.9 (Uniform and synchronizing acceptance). Let A be a limit-linear CBA with set of progressing letters P and set Q of pairs of choice letters. An infinite word w is *progressing* (in A) if it contains infinitely many progressing letters, i.e. there is, for all i, some $j \geq i$ such that $w(j) \in P$. The automaton A *admits progressing runs* if each of its compartments contains at least one synchronizing state. A progressing word $w \in \Sigma^\omega$ is *uniform* (in A) if for all $i < j$ such that $w(i)$ and $w(j)$ are progressing letters and the finite word $w(i+1) \dots w(j-1)$ contains no progressing letter, we have that for all choice pairs $(a_1, a_2) \in Q$, the word $w(i+1) \dots w(j-1)$ contains at most one of the letters a_1 or a_2. An infinite word w is *synchronizing* (in A) if for all i such that $w(i) \in P$, we have that for all runs $\rho \in \mathsf{run}(\mathsf{A}, w)$, $\rho(i)$ is a synchronizing state. Let $\tilde{\Sigma^\omega} \subseteq \Sigma^\omega$ denote the set of all uniform and synchronizing words. Given an automaton A, we define the uniform and synchronizing language that is accepted by A by

$$\tilde{L}(\mathsf{A}) = L(\mathsf{A}) \cap \tilde{\Sigma^\omega}.$$

Definition 4.1.10 (Circular permutation). Let U be a set. A bijection $f : U \to U$ on U is a *circular permutation* if there is, for all $x, y \in U$, a number $n \leq |U|$ such that $x = f^n(y)$. States from $\pi_3[C]$ for some compartment C in a limit-linear CBA are arranged linearly by the circular permutation $f_C : \pi_3[C] \to \pi_3[C]$ defined by $f_C(v) = w$, where w is *the* state with $v \neq w$ for which there is a transition $(v, a, w) \in F$. Given some compartment C such that $\pi_3[C]$ contains at least one synchronizing state and given some state $v \in \pi_3[C]$, we use $\mathsf{sync}(v)$ to denote *the* synchronizing state $w \in \pi_3[C]$ with the property that there is no $q' < q$ such that $f_C^{q'}(v)$ is a synchronizing state, where q is the least number with $f_C^q(v) = w$. In particular, if $\mathsf{sync}(v) = u$ for states $v, u \in V$, then there is some compartment C such that $v, u \in \pi_3[C]$. For a state $v \in V$ for which there is no compartment C such that $v \in \pi_3[C]$, we leave $\mathsf{sync}(v)$ undefined. We also define $\mathsf{nextsync}(v) = \mathsf{sync}(f(\mathsf{sync}(v)))$. Additionally, we assume a circular permutation on compartments, i.e. a suitable bijection $g : \{1, \dots, q\} \to \{1, \dots, q\}$, where the automaton at hand has q disjoint compartments $C_1, \dots, C_q$. Further we define a fixed function $\mathsf{nextcomp} : V \to V$ that maps, for all $1 \leq i \leq q$, each state $v \in \pi_3[C_i]$ to an *arbitrary* synchronizing state from $\pi_3[C_{g(i)}]$.

Thus the function $\mathsf{nextcomp}$ cycles through all compartments of a given CBA while the function $\mathsf{nextsync}$ cycles through the synchronizing states inside a single compartment.

Given an accepting run ρ of a limit-linear CBA on some word w, we have $\mathsf{Inf}(\rho) = \{v\}$ for some non-synchronizing state $v \in \pi_3[F]$ or $\mathsf{Inf}(\rho) = C$, where C is some compartment of the automaton. Furthermore, for a fixed compartment C in a limit-*stationary* automaton and a fixed word w, there are infinitely many numbers i such that for all runs ρ on w with $\mathsf{Inf}(\mathsf{trans}(\rho)) = C$, we have $\rho(i) = v$, where v is the synchronizing state of C. Then any two runs that accept w by using the whole compartment C are synchronized from some point on; in other words: in limit-linear or limit-stationary automata, runs that accept w by using the whole compartment C differ only on finite prefixes.

4.1.2 Determinization of Automata on Infinite Words

For automata on infinite words, determinization procedures are more involved and lead to asymptotically larger automata than the plain powerset construction for NFA. We now discuss several determinization methods for various types of automata and in particular show how limit-deterministic parity automata can be determinized through limit-deterministic Büchi automata. The determinization methods for limit-linear CBA and limit-deterministic PA are novel and the methods for unrestricted CBA [34], unrestricted BA [44,42] and unrestricted PA [27] are standard (but we prove a slighter lower bound on automata size for determinized unrestricted PA), while a method that determinizes limit-deterministic BA has recently been described in [12]. We give a different representation of this last method in terms of partial permutations and obtain a slightly lower bound on the size of determinized limit-deterministic BA than [12].

4.1.2.1 Determinizing Co-Büchi Automata For limit-linear CBA, it suffices to annotate macrostates from the powerset construction with single tracked states and a *time-out* (that is, a natural number) on the compartment to which the tracked state belongs.

Definition 4.1.11 (Determinization of limit-linear CBA). We assume a limit-linear CBA $\mathsf{A} = (V, \Sigma, \Delta, v_0, F)$ that admits progressing runs and in which each compartment has at most $q \leq |V|$ synchronizing states. By definition we have, for each $t \in F$, that $|\pi_3[F|_t \cap \mathsf{Id}]| = 1$. We recall that the function $\mathsf{nextcomp}$ allows us to cycle through all compartments of A while the function $\mathsf{nextsync}$ cycles through the synchronizing states in a single compartment. We put $\mathsf{s}(v) = \{u \in F \mid \mathsf{sync}(u) = \mathsf{nextsync}(v)\}$ and define the deterministic CBA $\mathsf{B} = (V', \Sigma, \delta, w_0, F')$ by putting

$$V' = \mathcal{P}(V) \times \{1, \ldots, q\} \times \pi_3[F]$$
$$F' = \{((U, m, v), a, (U', m', v')) \in \delta \mid m = m' \text{ and } C(v) = C(v')\}$$

and $w_0 = (\{v_0\}, q, v)$, for some synchronizing $v \in F$; furthermore, we put, for $(U, m, v) \in V'$ and $a \in \Sigma$,

$$\delta((U, m, v), a) = (\Delta(U, a), m, v)$$

if $a \notin P$ and

$$\delta((U, m, v), a) = \begin{cases} (\Delta(U, a), m, \mathsf{nextsync}(v)) & \text{if } F(U, a) \cap \mathsf{s}(v) \neq \emptyset \\ (\Delta(U, a), m - 1, (\mathsf{nextsync})^2(v)) & \text{if } F(U, a) \cap \mathsf{s}(v) = \emptyset, m > 1 \\ (\Delta(U, a), q, \mathsf{nextcomp}(v)) & \text{if } F(U, a) \cap \mathsf{s}(v) = \emptyset, m = 1 \end{cases}$$

if $a \in P$, where $F(U, a) = \{v \in V \mid \exists u \in U.\, (u, a, v) \in F\}$; we refer to transitions from $\overline{F'}$ (i.e. the latter two clauses in the above definition of $\delta((U, m, v), a)$ for $a \in P$) as *retracking steps*.

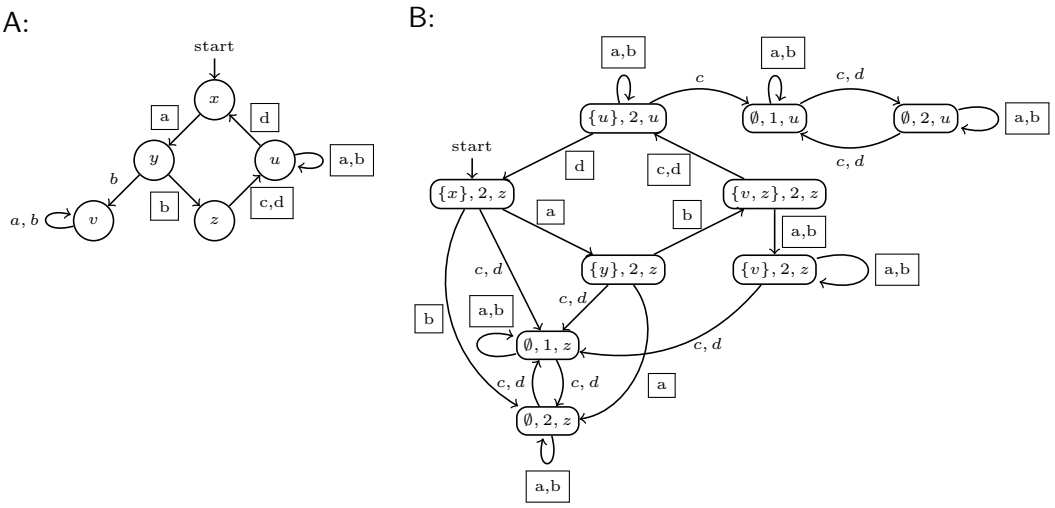
We define the *label* $l(u) \subseteq V$ of a macrostate $u = (U, m, v) \in V'$ by $l(u) = U$.

The determinized automaton thus tracks single synchronizing states $v \in \pi_3[F]$ along macrostates but also keeps a time-out on the number of retracking steps that are allowed to happen inside the compartment to which the tracked state belongs. The retracking steps are defined in such a way that if infinitely many retracking steps tack place, all runs from all compartments are eventually tracked. For any two synchronizing states $v, w \in C$ belonging to the same compartment C we have an asymmetric distance $d(v, w)$ such that $d \leq q \leq |C|$ and $\mathsf{nextsync}(w) = (\mathsf{nextsync})^d(v)$. This distance is invariant under simultaneous progressing transitions, i.e. for any two states $v, w \in \pi_3[F]$, $d(\mathsf{nextsync}(v), \mathsf{nextsync}(w)) = d(v, w)$. A refocusing step inside a compartment chooses $\mathsf{nextsync}^2(v) = \mathsf{nextsync}(\mathsf{nextsync}(v))$ as new tracked state and thus reduces the distance from the current tracked synchronizing state to any synchronizing state in the same compartment – except for $\mathsf{nextsync}(v)$ – by one. Thus q retracking steps inside a compartment suffice to ensure that every run through that compartment has been tracked at least once. Hence the time-out on compartments is initialized as q (where $q \geq |C|$ for any compartment C) and is reduced by each retracking step inside the compartment. When it reaches 1, all runs through the current compartment have been tracked; in any ensuing retracking step, a state from the next compartment is chosen as new tracked state and the time-out is set to q again. This combined retracking process ensures that if infinitely many retracking steps take place, then every uniform and progressing run is tracked and finished infinitely often.

Example 4.1.12. Let $\Sigma = \{a, b, c, d\}$, let c and d be progressing letters and let there be no choice letters. We determinize the limit-linear CBA A that is shown below to the DCBA B, depicting accepting transitions with rectangular boxes around their labels. All states in A have at most one successor that can be reached with an accepting transition (with the exception of trivial loops). Also no trivial loop is labelled with a progressing letter. There is just one 0-compartment, namely the set $C = \{(x, a, y), (y, b, z), (z, c, u), (u, a, u), (u, b, u), (u, d, x)\} = F$, and this compartment has the two synchronizing states z (which has a c-successor in C) and u (which has a d-successor in C); thus A permits progressing runs and we have $q = 2$. To improve the legibility of this example, there are, for some states in $\pi_3[F]$, non-choice letters for which the respective states do not have a transition in F; hence, A is formally not a limit-linear CBA. We note that A can be easily made limit-linear without changing the essence of this example by adding a trivial accepting loop at each state $\pi_3[F]$ for each non-choice letter.

We choose $f(x) = y$, $f(y) = z$, $f(z) = u$ and $f(u) = x$; thus $\mathsf{sync}(x) = \mathsf{sync}(y) = \mathsf{sync}(z) = z$ and $\mathsf{sync}(u) = u$ so that $\mathsf{nextsync}(z) = u$, $\mathsf{nextsync}(u) = z$. As there is just one compartment, we choose $\mathsf{nextcomp}(z) = \mathsf{nextcomp}(u) = z$ and pick $z \in \pi_3[F]$ as the initially tracked state in the determinized automaton, which then starts at the state $(\{x\}, 2, z)$. Moreover, we have $\tilde{L}(\mathsf{A}) = \tilde{L}(\mathsf{B}) = (ab(c + d)(a + b)^*d)^\omega$ since uniform words contain infinitely many progressing letters and hence cannot end on

$(a+b)^\omega$; in this example, there is no choice letter so that every progressing word is uniform.



Runs in the determinized automaton B start at the state $(\{x\}, 2, z)$ and initially attempt to track states that have z as synchronizing state. There is a a-transition from x to y and a b-transition from y to z in A so that we have an (accepting) a-transition from $(\{x\}, 2, z)$ to $(\{y\}, 2, z)$ and an (accepting) b-transition from $(\{y\}, 2, z)$ to $(\{v\}, 2, z)$ in B. From z there are accepting c- and d-transitions to u in A and both c and d are progressing letters so that we have accepting c- and d-transitions $(\{v\}, 2, z)$ to $(\{u\}, 2, u)$ in B; note that $\mathsf{nextsync}(z) = u$. The circle is closed by the accepting transition (u, d, x) in A and similarly by the accepting and progressing transition $((\{u\}, 2, u), d, (\{x\}, 2, z))$ in B. After having read the word ab, A can be in one of the states v or z and B is in the state $(\{v, z\}, 2, z)$. If the next read letter is a or b, then the ab-run through z in A does not continue and is finished. In B, the non-progressing letter b thus leads from $(\{v, z\}, 2, z)$ to $(\{v\}, 2, z)$. The state v has no c- or d-transition in A so that when letter c or d is read, $(\{v\}, 2, z)$ transitions via a non-accepting transition to $(\emptyset, 1, z)$ in B. Since c and d are progressing and there is no accepting c- or d-transition from $\{v\}$ to a state that has $\mathsf{nextsync}(z) = u$ as synchronizing state, the time-out for the compartment F is decreased from 2 to 1 and the transition indeed is non-accepting. Another progressing transition from $(\emptyset, 1, z)$ cannot decrease the time-out any more since it already is 1 so that a state from the next compartment is tracked. Since there is just one compartment in A and we chose $\mathsf{nextcomp}(z) = z$, $(\emptyset, 1, z)$ has non-accepting c- and d-transitions to $(\emptyset, 2, z)$. We note that all non-progressing transitions in B are accepting, which allows for additional accepting runs for non-progressing words in B; thus A and B are only equivalent w.r.t. uniform and synchronizing words.

Lemma 4.1.13. *Let* A, B, $n := |V|$ *and* $q \leq n$ *be as in Definition 4.1.11. Then* $\tilde{L}(\mathsf{A}) = \tilde{L}(\mathsf{B})$ *and* $|V'| \leq 2^n \cdot q \cdot o \in 2^{\mathcal{O}(n)}$*, where and* A *has* $o \leq |\pi_3[F]| \leq n$ *compartments.*

Proof. The bound on the size of V' follows immediately from the definition of V'.

Let $w \in \tilde{L}(\mathsf{A})$, i.e. let w be uniform (and hence progressing) and synchronizing and let there be an accepting run $\rho \in \mathsf{run}(\mathsf{A}, w)$. Then there is some i such that for all $j \geq i$, $(\mathsf{trans}(\rho))(j) \in F$. Since A is limit-linear, we have for all $j \geq i$ such that $w(j)$ is a progressing letter that $F|_{\rho(j),w(j)} = \{\rho(j+1)\}$. Let $\tau = \mathsf{run}(\mathsf{C}, w)$ and write, for $j \geq i$, $\tau(j) = (U_j, m_j, v_j)$, having $\rho(j) \in U_j$. We put $u_j = \mathsf{sync}(\rho(j))$ and distinguish three cases:

1. If $u_j = v_j$ for some $j \geq i$, then we have, that for all $j' \geq j$, $\tau(j'+1) = (\Delta(U_{j'}, w(j')), m_{j'}, u_{j'})$: Assume $v_{j'} = u_{j'}$. If $w(j')$ is not a progressing letter, then $v_{j'+1} = v_{j'} = u_{j'} = u_{j'+1}$. If $w(j')$ is a progressing letter, then $u_{j'+1} = \mathsf{nextsync}(u_{j'})$ and $v_{j'+1} = \mathsf{nextsync}(v_{j'}) = \mathsf{nextsync}(u_{j'})$. As $\rho(j') \in U_{j'}$ for all $j' \geq i$ and $\rho(j'+1) \in F(U_{j'}, w(j')) \cap \mathsf{s}(v_{j'})$ for all $j' \geq i$ such that $w(j')$ is a progressing letter, no retracking step takes place after position i, showing that τ is accepting.
2. If $u_j \neq v_j$, $v_j \in C(\rho(j))$ and $m_j > d(v_j, u_j)$ (where $d(v_j, u_j)$ denotes the least number m with $\mathsf{nextsync}^m(v_j) = u_j$), i.e. if both v_j and $\rho(j)$ belong to the same compartment and the time-out allows for reaching v_j before leaving the compartment, then we observe that for every retracking step at some position j' with $w(j')$ a progressing letter, we have $d(v_{j'}, u_{j'}) < d(\mathsf{nextsync}(\mathsf{nextsync}(v_{j'})), \mathsf{nextsync}(u_{j'}))$. By induction on m_j we see that eventually either no retracking step takes place (in which case τ is accepting) or a position j'' is reached where $d(v_{j''}, u_{j''}) = 0$, i.e. where $v_{j''} = u_{j''}$. Then we are in case 1.
3. If $u_j \neq v_j$, and $v_j \notin C(u_j)$ then v_j and u_j do not belong to the same compartment. If $u_j \neq v_j$, $v_j \in C(u_j)$ and $m_j \leq d(v_j, u_j)$, then the two states belong to the same compartment but the time-out m_j does not allow for reaching v_j before changing the compartment. Let $j' \geq j$ be the first position with $v_{j'} \notin C(u_{j'})$, noting that if no such position exists, then no retracking step occurs from some point on and τ is accepting. Let o denote the least number such that $(\mathsf{nextcomp})^o(u_{j'}) \in C(v_{j'})$, i.e. the number of times that the compartment has to be changed until a state from the compartment to which $v_{j'}$ belongs is tracked. Proceeding by induction over o, we have that either eventually no retracking step takes place, or we reach the situation $o = 0$. In the former case, τ is accepting, in the latter case, let j'' be the first position j'' with $j'' \geq j'$ and $u_{j''} \in C(v_{j''})$. As the focus just changed to the compartment of $v_{j''}$, $m_{j''} = |F|$ and either $v_{j''} = u_{j''}$ and we are in case 1, or $v_{j''} \neq u_{j''}$ but $u_{j''} \in C(v_{j''})$ and $m_{j''} > d(v_{j''}, u_{j''})$ and we are in case 2.

Conversely, let $w \in \tilde{L}(\mathsf{B})$, i.e. let w be a uniform (and hence progressing) and synchronizing word and let $\tau = \mathsf{run}(\mathsf{B}, w)$ be an accepting run. Then there is an i such that for all $j \geq i$ with $\tau(j) = (U_j, m_j, v_j)$, $m_{j+1} = m_j$ and $C(v_j) = C(v_{j+1})$. In other words: from i on, none of the two retracking steps takes place in τ. For $i \geq 0$, we have $U_i = \Delta(v_0, \mathsf{pre}(w, i))$, i.e. U_i contains exactly those states that are reachable in A via the first $i-1$ letters of the word

w. Let j_0 be the first position after i such that $w(j_0)$ is a progressing letter. Then we have $F(U_{j_0}, w(j_0)) \cap \mathsf{s}(v_{j_0}) = \{u_0\}$ for some u_0 and $\tau(j_0+1) = (U_{j_0+1}, m_{j_0+1}, \mathsf{nextsync}(v_{j_0}))$ with $m_{j_0+1} = m_{j_0}$. As $F(U_{j_0}, w(j_0)) \subseteq \Delta(U_{j_0}, w(j_0))$, we have $u_0 \in U_{j_0+1}$ and there is a finite run $\rho_0 \in \mathsf{run}_f(\mathsf{A}, \mathsf{pre}(w, j_0+1))$ with $\rho(j_0+1) = u_0$. Let j_1 denote the first position after j_0 such that $w(j_1)$ is a progressing letter. We have to show that there is a finite run in A on the word $w(j_0+1)\dots w(j_1)$ that starts at u_0 and uses only transitions from F. There is some u_1 such that $F(U_{j_1}, w(j_1)) \cap \mathsf{s}(v_{j_1}) = \{u_1\}$ where, i.e. U_{j_1} contains a state u'_{j_1} with $\mathsf{sync}(u'_{j_1}) = v_{j_1}$. By Lemma 4.1.14 below and since we have the F-transition $(v_{j_1}, w(j_1), u_1)$, there is a $w(j_0+1)\dots w(j_1)$-run ρ_1 that starts at u_0, ends at u_1 and only uses transitions from F. In this way we construct an infinite run $\rho = \rho_0; \rho_1; \dots$ of A on the word w. From i on, ρ uses only transitions from F and thus is accepting. □

Lemma 4.1.14. *Let A be a limit-linear CBA with progressing letters P and pairs of choice letters Q, let w be a uniform and synchronizing word and let $i < j$ be positions such that $w(i), w(j) \in P$ and for all $i < i' < j$, $w(i) \notin P$. Also let $\rho \in \mathsf{run}(\mathsf{A}, w)$ be a run such that $\rho(i) = v \in \pi_3[F]$ and $\mathsf{sync}(v) = u = \rho(j)$. Then there is a run $\tau \in \mathsf{run}(\mathsf{A}, w)$ such that $\tau(i) = v \in \pi_3[F]$, $\mathsf{sync}(v) = u = \tau(j)$ and $(\mathsf{trans}(\tau))(i') \in F$ for all $i < i' < j$.*

Proof. We start at v and inductively construct a run τ from v to u that reads the word $w(i+1)\dots w(j-1)$ and only uses transitions from F. We put $\tau(i) = \rho(i)$. Let $i < i' < j$. If $w(i')$ is not a choice letter then we have – since A is limit-linear – a transition $(\tau(i'-1), w(i'), v') \in F$ and we put $\tau(i') = v'$. If $w(i')$ is part of a choice pair (a_1, a_2), say $w(i') = a_1$, then $w(i+1)\dots w(j-1)$ does not contain a_2 by uniformity of w. Assume that $\tau(i'-1)$ does not have an a_1 successor in F. Then we have that for any position $i < j' < j$ such that $\rho(j'-1) = \tau(i'-1)$ and $\rho(j') \neq \rho(j'-1)$ (i.e. $(\mathsf{trans}(\rho))(j'-1)$ is not a trivial loop) that $w(j') = a$ so that $(\mathsf{trans}(\rho))(j'-1) \notin F$ and hence $\mathsf{sync}(\rho(j')) \neq u$: since A is a limit-linear CBA, no entry transition to the compartment of v (and hence no state u' with $\mathsf{sync}(u') = u$) can be reached by non-progressing transitions. This implies that there is no path from $\rho(j')$ for the word $w(j')\dots w(j-1)$ that reaches u, i.e. that $\rho(j) \neq u$, a contradiction. Since w is a synchronizing word and $w(j) \in P$, $\tau(j)$ is a synchronizing state and since τ only uses transitions from F between $\tau(i)$ and $\tau(j)$, we have $\tau(j) = \mathsf{sync}(\tau(i)) = u$, as required. □

Corollary 4.1.15. *Limit-linear Co-Büchi automata of size n, with o compartments and at most q synchronizing states per compartment can be determinized to DCBA of size at most $o \cdot q \cdot 2^n$. Limit-stationary Co-Büchi automata of size n and with o compartments can be determinized to DCBA of size at most $o \cdot 2^n$.*

In both cases the determinized automaton is equivalent to the original automaton only w.r.t. uniform and synchronizing acceptance, which however is sufficient for the constructions that are described in Chapter 5.

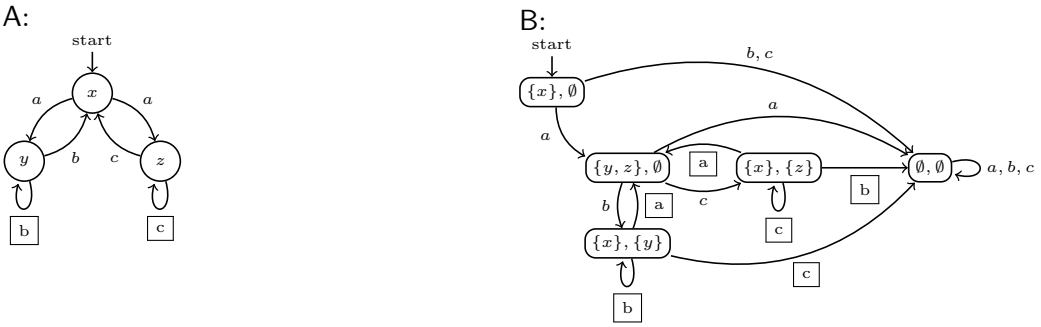
In unrestricted Co-Büchi automata, runs can branch (non-linearly and) nondeterministically through different transitions from F so that accepting runs for a single word can differ in their infinite part. Thus it is not sufficient to sequentially track single states; rather a *set* of states has to be tracked when determinizing unrestricted CBA. This method was first described by Miyano and Hayashi in [34]. When one of these tracked states is finished (i.e. no transition from that state under the next read letter is contained in F), then it is removed from the set of tracked states. When the set of tracked states is just the empty set, a retracking step takes place and the set of all current states that have been reached by an accepting transition by the last letter that was read is chosen as new set of tracked states. A run is accepting if it contains only finitely many retracking steps.

Definition 4.1.16 (Determinization of CBA). Let $\mathsf{A} = (V, \Sigma, \Delta, v_0, F)$ be a NCBA. We define the DCBA $\mathsf{B} = \{V', \Sigma, \delta, (\{v_0\}, \emptyset), F'\}$, where V' is the set of all functions $f : V \to \{0, 1, 2\}$ with the property that for all $v \in V$, $f(v) = 2$ implies $v \in F$. Given some $f \in V'$, we can see it as a macrostate; here, macrostates are pairs (U, W) of disjoint sets $U \subseteq V$, $W \subseteq F$ with $U = \{v \in V \mid f(v) = 1\}$ and $W = \{v \in F \mid f(v) = 2\}$, where W is the set of tracked states in f and U is the set of untracked states in f. We put $F' = \{((U, W), a, (U', W')) \in \delta \mid W \neq \emptyset\}$ and, for $(U, W) \in V'$ and $a \in \Sigma$,

$$\delta((U, W), a) = \begin{cases} ((\Delta(U, a) \cup \overline{F}(W, a)), F(W, a)) & \text{if } W \neq \emptyset \\ (\overline{F}(U, a), F(U, a)) & \text{if } W = \emptyset, \end{cases}$$

where $F(U, a) = \{v \in V \mid \exists u \in U.\, (u, a, v) \in F\}$ and $\overline{F}(U, a) = \{v \in V \mid \exists u \in U.\, (u, a, v) \in \overline{F}\}$. We refer to transitions from $\overline{F}$ as *retracking steps*. We define the *label* $l(v)$ of a macrostate $v = (U, W) \in V'$ by $l(v) = U \cup W$.

Example 4.1.17. Let $\Sigma = \{a, b, c\}$. We determinize the NCBA A that is shown below to the DCBA B, depicting accepting transitions with rectangular boxes around their labels. We have $L(\mathsf{A}) = L(\mathsf{B}) = (ab^+ + ac^+)^*(ab^\omega + ac^\omega)$, i.e. both automata accept words that start on a, do not contain bc, cb or aa and that end on b^ω or on c^ω.



Runs in the determinized automaton B start at the macrostate $(\{x\}, \emptyset)$ and initially

do not track any state. There are non-accepting a-transitions from x to y and to z in A so that we have an a-transition from $(\{x\}, \emptyset)$ to $(\{y, z\}, \emptyset)$ in B; this transition starts at a macrostate with empty set of tracked states and hence is non-accepting. Neither y nor z are reachable from x by an accepting a-transition, so the macrostate $(\{y, z\}, \emptyset)$ still has the empty set as set of tracked states. From y there is an accepting b-transition to y and a non-accepting b-transition to x in A; similarly, there is an accepting c-transition from z to z and a non-accepting c-transition from z to x. In B, there are corresponding non-accepting b- and c-transitions from $(\{y, z\}, \emptyset)$ to $(\{x\}, \{y\})$ and $(\{x\}, \{z\})$, respectively. All outgoing transitions from these two macrostates are accepting. The b-loop at $(\{x\}, \{y\})$ exists because of the b-loop at y in B that keeps on tracking y. For the letter a however, there is a transition from $(\{x\}, \{y\})$ to $(\{y, z\}, \emptyset)$, as the state y does not have any outgoing a-transition in A. Thus the states y and z can only be reached by a-transitions from $\{x, y\}$ via the non-accepting transitions that start at x in A. It is hence not possible to track the tracked state y from $\{x, y\}$ to $\{y, z\}$ via an accepting transition so that the set of tracked states in the macrostate $(\{y, z\}, \emptyset)$ in B is the empty set and any subsequent transition in B will be a (non-accepting) retracking transition. Similarly, an accepting c-loop exists at $(\{x\}, \{z\})$ in B and the state z cannot be tracked through an a-transition from $\{x, z\}$ to $\{y, z\}$ so that there is an a-transition from $(\{x\}, \{z\})$ to $(\{y, z\}, \emptyset)$. As a result, accepting runs in B eventually loop at $(\{x\}, \{y\})$ or at $(\{x\}, \{z\})$.

Lemma 4.1.18 ([34]). *Let* A *and* B *be defined as described above. Then* $L(\mathsf{A}) = L(\mathsf{B})$ *and* $|V'| \leq 2^{n-m}3^m \leq 3^n$ *where* $n = |V|$, $m = |F| \leq n$.

Proof. The bound on the size of B follows since for $f \in V'$, for every $v \in \overline{F}$, $f(v) \in \{0, 1\}$ and for every $v \in F$, $f(v) \in \{0, 1, 2\}$. Thus there are $2^{|\overline{F}|} = 2^{n-m}$ choices for the states from $\overline{F}$ and $3^{|F|} = 3^m$ choices for the states from F. Combining both bounds, we obtain that there are $2^{n-m}3^m$ different functions $f \in V'$.

To see that $L(\mathsf{A}) \subseteq L(\mathsf{B})$, let $w \in L(\mathsf{A})$, i.e. let there be an accepting run $\rho \in \mathsf{run}(\mathsf{A}, w)$. We have to show that the run $\tau = \mathsf{run}(\mathsf{B}, w)$ is accepting, that is, that there is a number i such that for all $j \geq i$, $(\mathsf{trans}(\tau))(j) \in F'$, i.e. $\tau(j) = (U_j, W_j)$ for some $U_j \subseteq V$, $\emptyset \neq W_j \subseteq F$. As ρ is accepting, there is a number i' such that for all $j' \geq i'$, $(\mathsf{trans}(\rho))(j') \in F$. For each such j', we have $\rho(j') \in U_{j'} \cup W_{j'}$ by construction. If $\rho(j') \in U_{j'}$ (i.e. $\rho(j')$ is not in the set of tracked states $W_{j'}$), then we distinguish two cases: If there is no $o \geq j'$ with $W_o = \emptyset$ (i.e. no retracking step takes place after j'), then τ is accepting and we are done. Otherwise, let $o \geq j'$ be a number with $W_o = \emptyset$, i.e. let a retracking step take place in the transition $(\mathsf{trans}(\tau))(o)$. We show that $\rho(o + 1)$ is contained in the new set of tracked states W_{o+1}: As $\rho(o) \in U(o)$, $\rho(o + 1) \in \Delta(U_o, w(o))$; furthermore, $(\mathsf{trans}(\rho))(o) \in F$ and hence $\rho(o + 1) \in W_{o+1} = F(U_o, w(o))$. It remains to show that for $j \geq i'$, $\rho(j) \in W_j$ implies $\rho(j + 1) \in W_{j+1}$, i.e. that whenever $\rho(j)$ is in the set of tracked states after at least i' steps, it remains in the set of tracked states forever. As $\rho(j) \in W_j$, $W_j \neq \emptyset$. Then $\rho(j + 1) \in F(W_j, w(j)) = W_{j+1}$, as required.

To see $L(\mathsf{B}) \subseteq L(\mathsf{A})$, let $w \in L(\mathsf{B})$, i.e. let $\tau = \mathsf{run}(\mathsf{B}, w)$ be accepting. We have to show that there is an accepting run $\rho \in \mathsf{run}(\mathsf{A}, w)$ for which there is a number i such that for all $j \geq i$, $(\mathsf{trans}(\rho))(j) \in F$. As τ is accepting, there is a number i' such that for all $j' \geq i'$, $(\mathsf{trans}(\tau))(j') \in F'$, i.e. $\tau(j') = (U_{j'}, W_{j'})$ with $W_{j'} \neq \emptyset$. Let $v_{i'} \in W_{i'}$ be a state with the property that for all $j' \geq i'$, there is a state $v_{j'} \in W_{j'} \cap \Delta(\{v_{i'}\}, w(i', j')) \neq \emptyset$, where $w(i', j')$ denotes the word $w(i'), w(i'+1), \ldots, w(j'-1), w(j')$; then $v_{j'}$ is a state that is always has a descendant in the set of tracked states $W_{j'}$ (i.e. from i' on, the set of tracked states always contains at least one state $v_{j'}$ that goes back to $v_{i'}$). As each set of tracked states $W_{j'}$ with $j' \geq i'$ consists of those states that are reachable in A from some state $v \in W_{i'}$ via the word $w(i'), w(i'+1), \ldots, w(j'-1), w(j')$ using only transitions from F, a state $v_{i'}$ with the described property must exist: otherwise there is, for every state $v \in W_{i'}$ a point $j_v \geq i'$ such that W_{j_v} contains no state that can be reached from v via the word $w(i'), w(i'+1), \ldots, w(j_v - 1), w(j_v)$ using only transitions from F. Let $j'' = \max\{j_v \mid v \in W_{i'}\}$ be the greatest such number; we have $W_{j''} = \emptyset$, a contradiction. Thus there is a run $\rho \in \mathsf{run}(\mathsf{A}, w)$ with $\rho(i') = v_{i'}$ and for all $j' > i'$, $\rho(j') = v_{j'}$. As for each such j', $v_{j'} \in W_{j'}$ has an $w(j')$-transition to the state $v_{j'+1}$, ρ is accepting. □

Corollary 4.1.19. *Co-Büchi automata of size n can be determinized to deterministic Co-Büchi automata of size at most 3^n.*

4.1.2.2 Determinizing Büchi Automata For the determinization of Büchi automata, it is necessary to annotate individual tracked states with additional information that allows to ensure that the corresponding run uses transitions from F infinitely often. It is convenient to think of this information as the *age* of the respective states, where a tracked state that was reached by an accepting transition in the last step is younger than any tracked state that was not reached by an accepting transition in the last step. In general, the age of any two different tracked states can be different, thus it is not possible to simply track a set of of states, as in Co-Büchi automata, and assign one joint age to the whole set. Instead, every tracked state requires its own age. As it turns out, the *relative age* of tracked states suffices to detect accepting runs, i.e. it is sufficient to know at any point and for any two tracked states, which of the two states is the younger one.

As a result, the determinization of Büchi automata necessitates a *permutation structure* on the set of tracked states, i.e. a structure that orders them according to their relative age. Making use of a trick due to Piterman [42], it is then possible to employ a parity condition to ensure the existence of a tracked state that is accepting infinitely often without ever being removed from the set of tracked states. This is achieved by using odd priorities $2i + 1$ to detect the removal of tracked states with relative age i and even priorities $2i$ to detect that the tracked state with relative age i is reached by an accepting transition. When the highest priority that occurs infinitely often is even, there is a tracked state that from some point on is tracked forever and that is reached by an accepting transition infinitely often.

For limit-deterministic Büchi automata, tracked states only have deterministic transitions. Thus the permutation structure described above suffices for their determinization; the underlying idea of the resulting determinization method for limit-deterministic Büchi automata has first been described in the context of controller synthesis for LTL [12]. Here we reformulate the construction in terms of partial permutations and obtain slightly smaller automata.

Definition 4.1.20 (Partial permutations). Given a set U of states, let $\mathsf{pperm}(U)$ denote the set of *partial permutations* over U, i.e. the set of non-repetitive lists $l = [v_1, \ldots, v_n]$ with $v_i \neq v_j$ for $i \neq j$ and $v_i \in U$, for all $1 \leq i \leq n$. We denote the i-th element in l by $l(i) = v_i$, the empty partial permutation by $[\,]$ and the length of a partial permutation l by $|l|$. For $v \in U$, we write $v \in l$ if $l = [v_1, \ldots, v_n]$ and there is some $1 \leq i \leq n$ such that $v_i = v$.

Definition 4.1.21 (Determinization of limit-deterministic BA). Fix a limit-deterministic BA $\mathsf{A} = (V, \Sigma, \delta, u_0, F)$, and put $Q = \pi_3[\mathsf{reach}(\pi_3[F])]$, $\overline{Q} = V \setminus Q$, $q = |Q|$. Define the DPA $\mathsf{B} = (W, \Sigma, \delta', w_0, \alpha)$ by putting $W = \mathcal{P}(\overline{Q}) \times \mathsf{pperm}(Q)$, $w_0 = (\{u_0\}, [\,])$ if $u_0 \in \overline{Q}$, $w_0 = (\emptyset, [u_0])$ if $u_0 \in Q$ and for $g = (U, l) \in W$ and $a \in \Sigma$, $\delta'(g, a) = h$, where $h = (\delta(U, a) \cap \overline{Q}, l')$ and where l' is constructed from $l = [v_1, \ldots, v_m]$ as follows:

1. Define a list t of length m over $Q \cup \{*\}$ (with $*$ representing undefinedness) in which $t(i) = w$ if $\delta(v_i, a) = \{w\}$, and $t(i) = *$ if $\delta(v_i, a) = \emptyset$.
2. For $j < k$ and $t(j) = t(k)$, put $t(k) = *$.
3. Remove undefined entries in t, formally: for each $1 \leq i \leq |t|$, if $t(i) = *$, then iteratively put $t(j) = t(j+1)$ for each $i \leq j \leq |t|$, starting at i.
4. For any $w \in \delta(U, a) \cap Q$ that does not occur in t, add w to the end of t. If there are several such w, the order in which they are added to t is irrelevant.
5. Put $l' = t$.

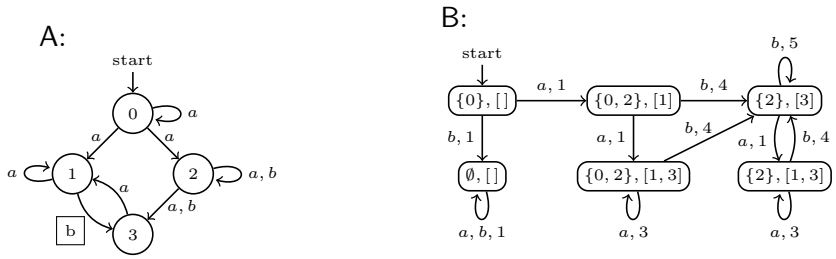
Temporarily, t may contain duplicate or undefined entries, but Steps 2. and 3. ensure that in the end, t is a partial permutation of length at most q. Let r (for 'removed') denote the lowest index i such that $t(i) = *$ after Step 2. Let a (for 'active') denote the lowest index i such that $(l(i), a, l'(i)) \in F$. If $r > |l'|$ and there is no i with $(l(i), a, l'(i)) \in F$, then put $\alpha(g, a, h) = 1$. Otherwise, put

$$\alpha(g, a, h) = \begin{cases} 2(q - r) + 3 & \text{if } r \leq a \\ 2(q - a) + 2 & \text{if } r > a. \end{cases}$$

We define the *label* $l(v)$ of a macrostate $v = (U, l) \in W$ by $l(v) = U \cup \{v \in V \mid v \in l\}$.

Example 4.1.22. Consider the limit-deterministic BA A depicted below and the determinized DPA B that is constructed from it by applying the method. We see by Lemma 4.1.6 that A is really limit-deterministic: we have $F = \{(1, b, 3)\}$, i.e. the

b-transition from state 1 to state 3 (depicted with a boxed transition label) is the only accepting transition; thus we have $Q = \mathsf{reach}(\pi_3[F]) = \{1, 3\}$ (so $\overline{Q} = \{0, 2\}$), and the states 1 and 3 are deterministic. Moreover, $L(\mathsf{A}) = L(\mathsf{B}) = a(a|b)^+(a^+b)^\omega$.



Notice that in B, there is a b-transition with priority 1 from the initial state to the sink state $(\emptyset, [\,])$ and an a-transition to $(\{0, 2\}, [1])$; as $1 \in Q$ but $(0, a, 1) \notin F$, this transition has priority 1. A further b-transition leads from 1 to 3 in A; in B, we have a b-transition from $(\{0, 2\}, [1])$ to $(\{2\}, [3])$ and since $(1, b, 3) \in F$, the first position in the permutation component is active during this transition so that the transition has priority 4. Yet another b-transition loops from $(\{2\}, [3])$ to $(\{2\}, [3])$. Since there is no b-transition starting at state 3, the first element in the permutation is removed in Step 1. of the construction. Since there is a b-transition from 2 to 3, it is added to the permutation again in Step 4. of the construction. Crucially, however, the priority of the transition is 5, since the first item of the permutation has been (temporarily) removed. The intuition is that the trace of 3 ends when the letter b is read; even though a new trace of 3 immediately starts, we do not consider it to be the same trace as the previous one. Thus the transition obtains priority 5 so that it may be used only finitely often in an accepting run of B, i.e. accepting runs contain an uninterrupted trace that visits state 3 infinitely often. Thus two or more consecutive b's can only occur finitely often in any accepted word.

Theorem 4.1.23. *Let* A *and* B *be as defined above and let* $n = |V|$*. We have* $L(\mathsf{A}) = L(\mathsf{B})$ *and* B *has at most* $2n + 1$ *priorities; for* $n \geq 4$*, we have* $|W| \leq n!e$*, where* e *denotes Euler's constant.*

Proof. Put $n = |W|$. The number of partial permutations over the set Q of size q is

$$|\mathsf{pperm}(Q)| \leq \sum_{i=0}^{q} \frac{q!}{(q-i)!} \leq q! \sum_{i=0}^{\infty} \frac{1}{i!} = q!e;$$

hence we have $|W| \leq 2^{n-q} \cdot q!e$. As $q \leq n$, we have that for $n > 3$, $2^{|n-q|} \cdot q! \leq n!$ so that the claimed bound follows. For all $t \in \delta'$, we have by definition of α that $1 \leq \alpha(t) \leq 2(q-1) + 3 = 2q + 1 \leq 2n + 1$, i.e. B has at most $2n + 1$ priorities. It remains to show that A and B are equivalent.

Let $w \in L(\mathsf{A})$, i.e. let there be an accepting run $\sigma \in \mathsf{run}(\mathsf{A}, w)$. By limit-determinism of A, there is an i such that for all $j \geq i$, $\delta|_{\sigma(j),w(j)} \cap Q = \{\mathsf{trans}(\sigma)(j)\}$. As σ is accepting,

$\mathsf{Inf}(\mathsf{trans}(\sigma)) \cap F \neq \emptyset$. To see that the run $\rho = \mathsf{run}(\mathsf{B}, w)$ is accepting, we have to show that the highest priority in $\mathsf{Inf}(\alpha \circ \mathsf{trans}(\rho))$ is even. For $j' \geq i$, we have $\rho(j') = (U_{j'}, l_{j'})$, and $Q \ni \sigma(j') = l_{j'}(k)$ for some k, i.e. from i on, the corresponding state from σ is contained in the permutation components of ρ. Since σ is infinite (and deterministic from i on), $\sigma(j')$ can be tracked forever so that the corresponding position in the permutation component never changes to $*$ in step 1. of the construction. If there is a position $m < k$ for which there is some $j'' \geq i$ such that $l_{j''}(m)$ changes to $*$ in step 1. of the construction, then k is changed to $k-1$, i.e. the considered state moves to the left in the permutation component; this can only happen finitely often without removing the considered state, thus eventually no position to the left of the considered state in the permutation changes to $*$; the considered state is stationary from then on. Now let o and k' be suitable numbers with $\sigma(o') = l_{o'}(k')$ for all $o' \geq o$; from o on, k' is the position that tracks the run σ and no position with index less than or equal to k' is removed from the permutation component after o and hence for all $q \geq o$, $r_q > k'$, where r_q is the lowest index that turns to $*$ during steps 1. and 2. of the transition $\mathsf{trans}(\rho)(q)$; i.e. all such transitions with odd priority have priority at most $2(n-k')+1$. As $\mathsf{Inf}(\mathsf{trans}(\sigma)) \cap F \neq \emptyset$, position k' is active infinitely often, i.e. there are infinitely many $q \geq o$ with $\mathsf{trans}(\sigma)(q) \in F$ and $\alpha(\mathsf{trans}(\rho)(q)) = 2(n-k')+2$. Thus ρ is accepting.

Conversely, let $w \in L(\mathsf{B})$, i.e. let the run $\rho = \mathsf{run}(\mathsf{B}, w)$ be accepting and let p be the highest priority in $\mathsf{Inf}(\alpha \circ \mathsf{trans}(\rho))$. Then p is even and there is some i such that for all $j \geq i$, $\alpha(\mathsf{trans}(\rho)(j)) \leq p$ and there are infinitely many $j \geq i$ with $\alpha(\mathsf{trans}(\rho)(j)) = p$. Let $\rho(j) = (U_j, l_j)$; abusing notation by interpreting l_j as a set, we observe $U_j \cup l_j \subseteq \delta(v_0, \mathsf{pre}(w, j))$, i.e. every state in U_j and l_j can be reached in A from v_0 via the word $\mathsf{pre}(w, j)$. We consider the position k with $2(q-k)+2 = p$. For all $j \geq i$ let r_j denote the lowest index that turns to $*$ in the transition $((U_j, l_j), w(j), (U_{j+1}, l_{j+1}))$; furthermore let a_j denote the lowest number such that $(l_j(a_j), w(j), l_j(a_j)) \in F$ (if no such number exists, then put $r_j = q+1$ and $a_j = q+1$, respectively). As for all $j \geq i$, $\alpha(\mathsf{trans}(\rho)(j)) \leq p = 2(q-k)+2$ we always have $r_j > k$ and $a_j \geq k$ since otherwise $\alpha(\mathsf{trans}(\rho)(j)) = 2(q-r_j)+3 \geq 2(q-k)+3$ or $\alpha(\mathsf{trans}(\rho)(j)) = 2(q-a_j)+2 > 2(q-k)+2$. Thus the k-th component is – from position i on – stationary in the permutations and infinitely often active. As $l_i(k) \in \delta(v_0, \mathsf{pre}(w, i))$, there is a finite run $\kappa \in \mathsf{run}_f(\mathsf{A}, \mathsf{pre}(w, i))$ with $|\kappa| = i+1$, $\kappa(0) = v_0$ and $\kappa(i) = l_i(k)$. By definition, $l_i(k) \in Q$ and since A is limit-deterministic, there is just a single run $\tau \in \mathsf{run}(\mathsf{A}, l_i(k), \mathsf{post}(w, i))$. We have $\tau \subseteq Q$ and for all $j \geq 0$, $\tau(j) = l_{i+j}(k)$, i.e. τ tracks the k-th element of the permutation components. Since there are infinitely many $j \geq i$ with $\alpha(\mathsf{trans}(\rho)(j)) = 2(q-k)+2$ and $a_j = k$, i.e. with the k-th element $l_j(k)$ in l_j active and hence with $\mathsf{trans}(\tau(j)) \in F$, we have $\mathsf{Inf}(\mathsf{trans}(\tau)) \cap F \neq \emptyset$. Thus $\kappa; \rho \in \mathsf{run}(\mathsf{A}, w)$ is accepting. □

Corollary 4.1.24. *Limit-deterministic Büchi automata of size n can be determinized to deterministic parity automata of size $\mathcal{O}(n!)$ and with $\mathcal{O}(n)$ priorities.*

For unrestricted Büchi automata, tracked states may transition non-deterministically. We say that two runs ρ_1 and ρ_2 of an automaton are *merged* at position i if $\rho_1(i) = \rho_2(i)$ and *split* otherwise. Runs in unrestricted Büchi automata may be merged and/or split infinitely often. Thus it is possible that an accepting and a non-accepting run both use some joint (non-accepting) transition infinitely often. Considering two runs ρ_1, ρ_2 that are merged at positions i and $j > i + 1$ but that are split at some position $i < k < j$, it is possible that $(\mathsf{trans}(\rho_1))(k) \in F$ but $(\mathsf{trans}(\rho_2))(k) \notin F$. The different runs from $\rho_1(i)$ to $\rho_1(j)$ have to be told apart and it is necessary to decide for each individual run whether it uses infinitely many accepting transitions. Assuming that we just track a set of states (ordered by their relative age), the two runs ρ_1 and ρ_2 are – at position j – both represented by the single state $\rho_1(j) = \rho_2(j)$, which does not hold the information that ρ_1 recently used an accepting transition while ρ_2 has not. Thus sets of states have to be tracked and annotated with additional information that allows for inferring whether the set of states has recently been accepting. In order to detect the accepting run ρ_1, the respective tracked set of states has to be accepting if it contains a state that is currently accepting. Then an accepting run is a sequence of macrostates, such that the sequence of tracked states allows for at least one run that uses a transition from F infinitely often. Here it is necessary to ensure that infinitely many of the accepting transitions belong to one individual run: It may be the case that the input word leads through two separate loops l_1, l_2 in the automaton consisting only of non-accepting transitions but having a nondeterministic accepting transition from loop l_1 to loop l_2. There is no single run that uses the accepting transition infinitely often but there are infinitely many runs that use the accepting transition (each run using it just once).

The Safra/Piterman construction [44,42] accounts for this fact by determinizing Büchi automata by means of so-called Safra trees, i.e. trees whose nodes are labelled with sets of states of the input automaton such that the label of a node is a proper superset of the union of all its children's labels. Additionally, the nodes are ordered by their relative age and the priorities of transitions between Safra trees determine the ages of the oldest nodes that are active and/or removed during this transition. In its original formulation, the Safra/Piterman construction adds new child nodes to the graph that are labelled with the accepting states in their parent's label. We observe that this step can be modified slightly – without affecting the correctness of the construction – by letting every state from the parent's label that is reached by an accepting transition receive its own separate child node; then the labels of newly created nodes always are singleton sets and in general the labels of nodes always belong to a single compartment.

Definition 4.1.25 (Compartmentalized Safra trees). Let $\mathsf{A} = (V, \Sigma, \delta, v_0, F)$ be a Büchi automaton with $n = |V|$ states, o different compartments with at most q states in each compartment. A *compartmentalized Safra tree* (W, p, m, l) over A is – similarly as in [42] – a tree with set $W = \{u_0, \ldots, u_{n-1}\}$ of nodes with $|W| = n$, with partial parent function $p : W \rightharpoonup W$ (which is undefined at the root), naming function $m : W \to \{0, \ldots, n-1\}$ and labelling function $l : W \to \mathcal{P}(V)$. The function m assigns

a name, that is, a number between 0 and $n-1$ (coding to relative age) to each node in a compartmentalized Safra tree while l assigns to each node $w \in W$ a set of states $l(w) \subseteq \pi_3[C] \subseteq V$, where C is some compartment of A; furthermore we require that the labels of child nodes are a proper subset of the parents label and that the labels of siblings do not intersect. The crucial difference to standard compact Safra trees is the condition that for each node $w \in W$, all states in the label of w belong to some individual compartment C. Given two nodes $v, w \in W$ with $m(v) < m(w)$ we say that v is *older* than w. Let $\mathsf{cst}(\mathsf{A})$ denote the set of all compartmentalized Safra trees over A.

It was shown in [42] that for any set of state V with $|V| = n$, there are $2(n^n) \cdot n! \in 2^{\mathcal{O}(n \log n)}$ *compact* Safra trees over V. We will construct a deterministic parity automaton so that – in contrast to compact Safra trees – information regarding the priorities is not part of compartmentalized Safra trees; instead the priorities are assigned to transitions. Furthermore, the restriction to compartmentalized Safra trees implies that the tree structure falls apart into o subtrees, each containing at most q states so that the bound on the number of compartmentalized Safra trees is $n^n o \cdot (q-1)!$ where $o \leq n$, $q \leq n$ (see Theorem 4.1.27).

Definition 4.1.26 (Determinization of NBA). Let $\mathsf{A} = (V, \Sigma, \Delta, v_0, F)$ be an NBA with $n = |V|$ states. We define the deterministic PA $\mathsf{B} = (\mathsf{cst}(\mathsf{A}), \Sigma, \delta, w_0, \alpha)$ by putting $w_0 = (\{u_0, \ldots, u_{n-1}\}, \emptyset, m_0, l_0)$ where $m_0(u_0) = 0$ and $l_0(u_0) = \{v_0\}$ (w_0 is the initial Safra tree with just one node u_0 with non-empty label $l_0(u_0) = \{v_0\}$ and with name $m_0(u_0) = 0$); For $t = (W, p, m, l) \in \mathsf{cst}(\mathsf{A})$ and $a \in \Sigma$ put $\delta(t, a) = t' = (W, p', m', l')$, where the components of t' are constructed from t as follows:

1. For every node $w \in W$ with label $l(w)$, put $l'(w) = \Delta(l(w), a)$, $p'(w) = p(w)$ and $m'(w) = m(w)$. This tracks the label of each node in the compartmentalized Safra tree through an a-transition in A while maintaining the tree and naming structures.
2. For every node $w \in W$ with $\emptyset \neq F(l(w), a) = \{v_1, \ldots, v_q\} \subseteq l'(w)$, $q > 0$, (i.e. for every node that has at least one state in its label that has an accepting a-transition), add q temporary nodes w_i to W, put

$$p'(w_i) = w \qquad\qquad l'(w_i) = \{v_i\},$$

   and assign the least unused name to w_i, i.e. put $m'(w_i) = q$, where q is the least number that is not assigned as name to some node with non-empty label. This step may temporarily increase the number of nodes in the tree to at most n^2; call this temporary set W'. Step 6. reduces the number of nodes and names to at most n again.
3. For every node $w \in W'$, remove all states from $l'(w)$ that are contained in the label of a sibling w' of w with $m(w') < m(w)$, $p'(w') = p'(w)$.
4. For every node $w \in W'$ with set of child nodes $D \subseteq W'$ such that $l'(w) = \bigcup_{d \in D} l'(d)$, put the label of each p'-descendant of w to $\emptyset$; call the node w *active during the a-transition from t to t'*.

5. Adjust the names of all remaining nodes: For each node $w \in W'$, let $rem(w)$ denote the number of those nodes whose labels have been empty in steps 4. or 5. and have a name less than $m'(w)$. Put $m'(w) = m'(w) - rem(w)$. Then there are at most n nodes with non-empty label: assign names up to n^2 to these nodes.
6. Remove each node v with $m'(v) \geq n$ from W', observing that for such v, $l'(v) = \emptyset$. Uses the remaining nodes $u_0, \dots u_{n-1}$ as new set W.

Let $b \in \Sigma$ and let t, t' be two compartmentalized Safra trees such that $(t, b, t') \in \delta$; let r (for *removed*) be the minimum of the names of all nodes whose label is empty in steps 4. and 5. when computing t' from t; if no node has an empty label, put $r = n$. Let a (for *active*) be the minimum of the names of all nodes that are active in step 4. when computing t' from t; if there is no active node, then put $a = n$. If $a = r = n$, then put $\alpha(t, b, t') = 1$; otherwise put

$$\alpha(t, b, t') = \begin{cases} 2(n-a) & \text{if } a < r \\ 2(n-r)+1 & \text{if } a \geq r. \end{cases}$$

We define the *label* $l(t)$ of a compartmentalized Safra tree $t \in \mathsf{cst}(\mathsf{A})$ with nodes $u_0, \dots, u_{n-1}$ by $l(t) = l(u_0)$, noting that for all $0 \leq i < n$, $l(u_i) \subseteq l(u_0)$ by construction.

Theorem 4.1.27 ([42]). *Let* A *and* B *be defined as described above. Then* $L(\mathsf{A}) = L(\mathsf{B})$, $|\mathsf{cst}(\mathsf{A})| \leq o(q-1)!n^n \leq n!n^n$ *and* B *has at most* $2n+1$ *priorities.*

Proof. Regarding the size of $|\mathsf{cst}(\mathsf{A})|$, we only consider the names of nodes and encode the labelling function as a function $f : V \to \{1, \dots, n\}$ where $f(v)$ denotes the name of the oldest node i such that the label of i and all ancestors of i contain v. There are n^n such functions. Given a labelling function f, the set of nodes W is partitioned into at most $o \leq n$ sets W_i of nodes such that for each W_i, the labels of all nodes from W_i are contained $\pi_3[C]$ for a single compartment C. Each such set W_i contains at most $q \leq n$ different nodes. There are o subtrees and each subtree can be encoded by a sequence of $q-1$ pointers, where the pointer for a node with name m points to the parent of the node which has name at most $m-1$. Thus there are, for each of the at most o compartments, at most $(q-1)!$ different subtrees. Putting everything together, we have $|\mathsf{cst}(\mathsf{A})| \leq o(q-1)!n^n$.

Let $w \in L(\mathsf{A})$, i.e. let there be an accepting run $\rho \in \mathsf{run}(\mathsf{A}, w)$. Then there is, for each i a $j \geq i$ such that $(\mathsf{trans}(\rho))(j) \in F$. Let $\tau = \mathsf{run}(\mathsf{B}, w)$ and for each i let $\tau(i) = (W_i, p_i, m_i, l_i)$ and let a_i and r_i denote the names of the oldest nodes that are active and removed, respectively, during the $w(i)$-transition from $\tau(i)$ to $\tau(i+1)$. We have $\rho(0) \in l(u_0)$. If the node u_0 is active infinitely often in τ, then we know – since the run ρ is infinite – that the label of u_0 is never the empty set and u_0 never changes its name. As u_0 is active infinitely often, there is for each i some $j \geq i$ with $a_j = 0$. As u_0 never has an empty label, we have $r_i > 0$ for all i. Thus $2n$ is the largest priority that occurs infinitely often in τ, i.e. τ is accepting. Otherwise let i_0 be the least number such that

u_0 is not active in $\tau(j)$ for all $j > i_0$. Take the first $j' > i_0$ such that $(\mathsf{trans}(\rho))(j') \in F$. Then in $\tau(j')$, u_0 has a new child node u_1 with $l(u_1) = \{\rho(j')\}$. As ρ is infinite, any node that is labelled with ρ at some point afterwards always has a non-empty label. Nodes with name less than $m_{j'}(u_1)$ can only finitely often have the empty set as label; similarly, the relevant state from ρ can only finitely often occur in an older sibling of u_1, in which case we move from u_1 to some older node. Let $u_{j''}$ be a node and let j'' the least number least number such that after j'', no node with name less than $m_{j''}(u_{j''})$ has an empty label, $\rho(j''') \in l_{j'''}(u_{j''})$ for all $j''' \geq j''$ and there is no $j''' \geq j''$ such that $\rho(j''')$ occurs in the label of some sibling u' of $u_{j''}$ with $m_{j'''}(u') < m_{j'''}(u_{j''})$. If u_1 is active infinitely often in τ, then τ is accepting. Otherwise, we repeat the above argumentation. As compartmentalized Safra trees contain at most n nodes, the argumentation has to be repeated at most n times, resulting in a chain $u_0, \ldots, u_{n'}$ of persisting nodes with $n' < n$ where u_{i+1} is a child of u_i and eventually no nodes older than u'_n are removed. Then $u_{n'}$ is active infinitely often without being removed. Thus $\max(\mathsf{Inf}(\alpha \circ \mathsf{trans}(\tau))) = 2(n - n')$, showing that τ is accepting and $w \in L(\mathsf{B})$.

Conversely, let $w \in L(\mathsf{B})$, i.e. let $\tau = \mathsf{run}(\mathsf{B}, w)$ be accepting. Let $\tau(i) = (W_i, p_i, m_i, l_i)$ and let $2e = \max(\mathsf{Inf}(\alpha \circ \mathsf{trans}(\tau)))$. Then from some point i on, no node with name less than or equal to e has an empty label and the node w_e with name e keeps this name forever and is active infinitely often. Between any two points $j' > j > i$ with $\alpha((\mathsf{trans}(\rho))(j)) = \alpha((\mathsf{trans}(\rho))(j')) = 2e$ and for all states $v \in l_{j'}(w_e)$, there is a state $u \in l_j(w_e)$ such that v is reachable from u in A by the word $w(j)w(j+1)\ldots w(j'-1)w(j')$ via a path that uses at least one transition from F; this is the case since w_e is active in the transition from $\tau(j'-1)$ to $\tau(j')$, meaning that union of the labels of its child nodes at j' is just $l_{j'}(w_e)$. As child nodes are only created for states that are reached by accepting transitions, there is for every $v \in l_{j'}(w_e)$ some $u \in l_j(w_e)$ and a path $u = u_j, u_{j+1} \ldots u_{j'-1}, u_{j'} = v$ such that $u_{i'+1} \in \Delta(u_{i'}, w(i'))$ for all $j \leq i' < j'$ and there is some $j \leq k \leq j'$ with $u_{k+1} \in F(u_k, w(k))$. We construct a tree with root v_0 in which there is a transition from v_0 to each state from $l_{j_1}(w_e)$, where $j_1 \geq i$ is the first position after i at which w_e is active. Then we inductively add nodes and edges as follows: let j_{m+1} be the first position after j_m at which w_e is active. Add a new node for each state $v \in l_{j_{m+1}}(w_e)$ and add an edge (u, v) for each state $u \in l_{j_m}(w_e)$ such that there is a path from u to v in A that uses an accepting transition; as described above, each newly added node has at least one predecessor. The constructed graph is infinite but finitely branching and by König's Lemma contains an infinite path. By construction, this path defines a run $\rho \in \mathsf{run}(A, w)$ that is accepting. □

Corollary 4.1.28. *Büchi automata of size n with o compartments and maximal compartment size q can be determinized to DPA of size $\mathcal{O}(o(q-1)!n^n)$.*

4.1.2.3 Determinizing Parity Automata It is well-known (e.g. Theorems 2 and 3 in [27]) that NPA of size n and with k priorities can be translated to equivalent NBA of size $\mathcal{O}(nk)$:

Definition 4.1.29. For a given PA $\mathsf{A} = (V, \Sigma, \Delta, u_0, \alpha)$ with $n = |V|$ and $k > 2$ priorities, we define the NBA $\mathsf{B} = (W, \Sigma, \Delta', u_0, F)$ by putting $W = V \cup (V \times \{0, \ldots, \lceil \frac{k-1}{2} \rceil\})$, and for $v \in W$ and $a \in \Sigma$,

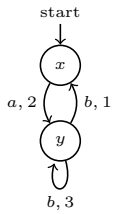
$$\Delta'(v,a) = \begin{cases} \{(w,m) \mid (v,a,w) \in \alpha_{2m}\} \cup \Delta(v,a) & \text{if } v \in V \\ \{(w,l) \mid (v',a,w) \in \alpha_{\leq 2l}\} & \text{if } v = (v',l) \notin V \end{cases}$$

Finally, we put $F = \{((v,l),a,(w,l)) \in \Delta' \mid (v,a,w) \in \alpha_{2l}\}$.

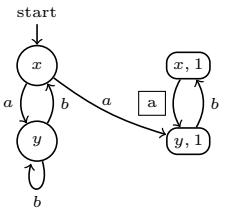
Every accepting run in the automaton B eventually has to assert that some even priority $2l$ is the highest priority that occurs from then on and that transitions with this priority are used infinitely often. Then the automaton moves to a copy of A with the carrier $V_{2l} = \{(v,l) \mid v \in V\}$ that represents the union of all $2l$-compartments in A. In this copy, transitions with priority greater than $2l$ are removed and transitions with priority $2l$ are accepting. This ensures that B accepts the same language as A (see Lemma 4.1.31). By definition of B, for all $l \leq \lceil \frac{k}{2} \rceil$ and each transition t with $\pi_3(t) = (v,l) \in V_{2l}$, $\pi_3[\mathsf{reach}(t)] \subseteq V_{2l}$ and $\pi_3[F] \subseteq \bigcup_{l \leq \lceil \frac{k}{2} \rceil} V_{2l}$. Thus B has at most $\lceil \frac{k}{2} \rceil$ compartments, each of size at most $o \leq n$, where o is the size of the largest compartment in A.

Example 4.1.30. Let $\Sigma = \{a, b\}$. We use the above construction to transform the PA A with three priorities that is shown below to the NBA B, depicting accepting transitions in B with rectangular boxes around their labels. We have $L(\mathsf{A}) = L(\mathsf{B}) = (ab^+)^*(ab)^\omega$, $n = 2$ and $k = 2$. We do not depict the states $(x,0)$ and $(y,0)$ from the construction since no accepting transition is reachable from these states so that they cannot be part of any accepting run.

A:



B:



Every accepting run in B has to eventually reach the state $(y,1)$ which corresponds to the assertion that the looping transition at b is not used any more. We observe that A is a limit-deterministic PA since $\{(x,a,y),(y,b,x)\}$ is the only 2-compartment in A and this compartment is deterministic. Furthermore, B is a limit-deterministic BA as $F = \{((x,1),a,(y,1))\}$ and $\mathsf{reach}(F) = \{((x,1),a,(y,1)),((y,1),b,(x,1))\}$ is deterministic. This is not a coincidence; in general, all 2-compartments V_{2l} in the obtained NBA correspond to a $2l$-compartment in the original NPA so that the transformation preserves limit-determinism (see Lemma 4.1.33 below).

Lemma 4.1.31. *Let* A, B, n *and* k *be as defined above. Then we have* $L(\mathsf{A}) = L(\mathsf{B})$ *and* $|W| \leq n(\lceil \frac{k}{2} \rceil + 1) \leq nk$.

Proof. The bound on the size of B follows trivially.

Let $w \in L(\mathsf{A})$, i.e. let there be a run $\rho \in \mathsf{run}(\mathsf{A}, w)$ with $\max(\mathsf{Inf}(\alpha \circ \mathsf{trans}(\rho)))$ even. Then there are i and l such that $\alpha(\mathsf{trans}(\rho)(i)) = 2l$, $\alpha(\mathsf{trans}(\rho)(j)) \leq 2l$ for all $j \geq i$ and there are infinitely many $j \geq i$ with $\alpha(\mathsf{trans}(\rho)(j)) = 2l$. Let $\sigma \in \mathsf{run}(\mathsf{B}, \mathsf{pre}(w, i))$ be a run with $\sigma(i) = \rho(i)$. We have $(\rho(i+1), l) \in \Delta'(\rho(i), w(i))$. As for all $j \geq i$, $\alpha(\mathsf{trans}(\rho)(j)) \leq 2l$, the run $\sigma;\kappa$ that is defined by $(\sigma;\kappa)(j') = \sigma(j')$ for $j' \leq i$ and by $(\sigma;\kappa)(j') = (\rho(j'), l)$ for $j > i$ is an infinite run of B on w. As there are infinitely many $j \geq i$ with $\alpha(\mathsf{trans}(\rho)(j)) = 2l$, $\mathsf{Inf}(\mathsf{trans}(\sigma;\kappa)) \cap F \neq \emptyset$. Thus $\sigma;\kappa \in \mathsf{run}(\mathsf{B}, w)$ is accepting.

Let $w \in L(\mathsf{B})$, i.e. let there be an accepting run $\rho \in \mathsf{run}(\mathsf{B}, w)$. Since ρ is accepting, there is a least number i such that $\rho(i) = (v, l)$ for some l and there are infinitely many $j \geq i$ with $(\mathsf{trans}(\rho))(j) \in F \cap V_l$, where $V_l = \{(v, l) \in W\}$. Write $(\mathsf{trans}(\rho))(j) = ((v(j), l), w(j), (v(j+1), l))$ for $j \geq i$. Since ρ is infinite, we have for all $j \geq i$ that $\alpha(v(j), w(j), v(j+1)) \leq 2l$. Also, there are infinitely many $j \geq i$ with $(\mathsf{trans}(\rho))(j) \in F$ and hence $\alpha(v(j), w(j), v(j+1)) = 2l$. We thus can define a run $\tau \in \mathsf{run}(\mathsf{A}, w)$ by putting $\tau(j) = \rho(j)$ for $j < i$ and $\tau(j) = v_j$ for $j \geq i$ where $\rho(j) = (v_j, l)$. As $\max(\mathsf{Inf}(\alpha \circ \mathsf{trans}(\tau))) = 2l$, τ is accepting. □

When using the improved Safra/Piterman construction described in the preceding section to determinize PA through NBA, the resulting Safra trees do not contain nodes with labels that contain states from different compartments in the original parity automata; thus the number of possible Safra trees is reduced. Indeed, for an input parity automaton with n states, k priorities, $o \leq n$ compartments and size of compartments at most $q \leq n$, the tree structure in the resulting Safra trees falls apart into at most $o \cdot \lceil \frac{k}{2} \rceil$ disjoint subtrees each of size at most q; the permutation component of the Safra trees however ranges over all $n + n\lceil \frac{k}{2} \rceil \in \mathcal{O}(nk)$ states in the Büchi automaton. By Corollary 4.1.28, the transformation yields a determinization procedure that determinizes NPA through NBA to DPA of size $\mathcal{O}(nk(q-1)!(nk)^{nk}) \in \mathcal{O}(n!(nk)^{nk+1})$ and with $2nk+1$ priorities.

Corollary 4.1.32. *Parity automata can be determinized to DPA of size* $\mathcal{O}(n!(nk)^{nk+1})$ *and with* $\mathcal{O}(nk)$ *priorities.*

In comparison, the original Safra/Piterman construction yields slightly larger DPA of size $\mathcal{O}((nk)!^2)$. As mentioned above, the described transformation from PA to BA also preserves limit-determinism:

Lemma 4.1.33. *If* A *and* B *are as in Definition 4.1.29, then limit-determinism of* A *implies limit-determinism of* B*.*

Proof. Let A be limit-deterministic; by Lemma 4.1.6 we have that for all $l \leq \lceil \frac{k}{2} \rceil$, all $2l$-compartments in A are deterministic. By Lemma 4.1.6 it suffices to show that $\mathsf{reach}(F)$ is deterministic. We observe that for each state $(v, l) \in \mathsf{reach}(F)$, $(v, l) \in V_{2l}$. As the $2l$-compartment of v in A is deterministic, $\mathsf{reach}(v, l)$ is deterministic by definition of Δ'. □

We thus obtain the following central result:

Corollary 4.1.34. *Limit-deterministic parity automata of size* n *with* k *priorities can be determinized to deterministic parity automata of size* $\mathcal{O}((nk)!)$ *and with* $\mathcal{O}(nk)$ *priorities.*

4.2 Infinite Games

We define the abstract notion of two-player games in which infinite plays are allowed (see e.g. [21] for an overview):

Definition 4.2.1 ((Co-)Büchi and parity games). A *parity game* is a tuple (V, E, α), where $V = V_\exists \cup V_\forall$, the set of *nodes*, is the disjoint union of the set of nodes $V_\exists$ for player Éloïse ($\exists$) and $V_\forall$ for player Abélard ($\forall$), respectively. Given a node $v \in V$, $E(v) \subseteq V$ is the set of *moves* for the player to whom v belongs. The function $\alpha : E \to \mathbb{N}$ assigns a *priority* $\alpha(v)$ to each *edge* $e \in E$. A *Büchi game* is a parity game (V, E, α) with just the two priorities 1 and 2, i.e. with $\alpha(e) \in \{1, 2\}$ for each $e \in E$. A *Co-Büchi game* is a parity game with just the two priorities 0 and 1. A *play* ρ is a sequence of moves of Abélard and Éloïse or, equivalently, a sequence of nodes of the game representing the nodes of the game through which the play progresses (with indices starting at 0 and $\rho(i+1) \in E(\rho(i))$); we require that a play is either infinite or ends in a node from which there are no moves. A *partial play* is defined in the same way but as a *finite* sequence of moves, with no requirements on the last node. The i-th node in a play ρ is referred to as $\rho(i)$. Finite plays are *won* by the player who does *not* own the last node in the play. An infinite play ρ is *won* by Éloïse if the highest priority that occurs infinitely often in ρ is even (formally: $\max(\mathsf{Inf}(\alpha \circ \mathsf{edge}(\rho)))$ is even, where $(\mathsf{edge}(\rho))(i) = (\rho(i), \rho(i+1)) \in E$); otherwise Éloïse *loses* the play ρ. Player Abélard wins a play if and only if Éloïse loses it.

An Éloïse- or Abélard-*strategy* is a partial function $s : V^*V_0 \rightharpoonup V$ or $t : V^*V_1 \rightharpoonup V$, respectively, that assigns a move to each sequence of nodes leading to a node (referred to as the *history* of the play) that belongs to the respective player. If the strategy depends only on the current node, it is said to be *history-free* or *positional*. A play $(v_0 \dots v_i)$ *conforms* to an $\exists$-strategy s if for all $j < i$ such that Éloïse owns v_j, $v_{j+1} = s(v_0 \dots v_j)$. We require a strategy to be defined on all partial plays that conform to it. An infinite play *conforms* to s if all its finite prefixes conform to s. We have similar definitions of

plays that conform to Abélard-strategies. Given a node v, a strategy s for Éloïse is a *winning strategy (for* Éloïse*) at* v if Éloïse wins all plays that start at v and conform to s; in this case we sometimes say shortly that s *wins* v. Parity games have positional winning strategies; formally: if there is a strategy with which a player wins some node, then there is a positional strategy with which the same player wins the node. Solving a parity game amounts to computing the *winning regions*, i.e. the sets of nodes at which Éloïse and Abélard have respective winning strategies. We note that the winning regions of the two players do not intersect; however, *partial* games (see Section 4.3.4) may have nodes in which neither player has a winning strategy.

4.3 Describing Regions in Automata and Games

Automata and games are Kripke structures with one transition relation Δ_a for each letter $a \in \Sigma$. Thus (multi-modal) μ-calculus formulas can be interpreted over automata and games. In fact,

– the non-emptiness problem for automata,
– the computation of winning regions of games and
– the verification of winning strategies for games

can all be seen as model checking problems for different μ-calculus formulas.

4.3.1 Fixpoint Formulas over Automata

Let $\mathsf{A} = (V, \Sigma, \Delta, v_0, \alpha)$ be an automaton with k priorities and put

$$\Delta^i(a, v) = \{w \in V \mid (v, a, w) \in \Delta, \alpha(v, a, w) = i\},$$

where $a \in \Sigma$, $v \in V$, $0 \leq i < k$. For all interpretations σ that map fixpoint variables to subsets of V and all $a \in \Sigma$, we define

$$[\![\Diamond_a^i X]\!]_\sigma = \{v \in V \mid \Delta^i(a, v) \cap \sigma(X) \neq \emptyset\} \qquad [\![\Box_a^i X]\!]_\sigma = \{v \in V \mid \Delta^i(a, v) \subseteq \sigma(X)\}$$
$$[\![\Diamond^i X]\!]_\sigma = \bigcup_{a \in \Sigma} [\![\Diamond_a^i X]\!]_\sigma \qquad [\![\Box^i X]\!]_\sigma = \bigcap_{a \in \Sigma} [\![\Box_a^i X]\!]_\sigma$$

Let ψ be a μ-calculus formula that uses only modal operators $\Diamond_a^i$, $\Box_a^i$, $\Diamond$ and $\Box$ that are directly applied to fixpoint variables. Then we define the extension $[\![\psi]\!]_\mathsf{A} \subseteq V$ of ψ in A in the standard way.

Definition 4.3.1 (Non-emptiness regions of CBA). For Co-Büchi automata, we have $\Delta_0 = F$ and $\Delta_1 = \overline{F}$. We define the formula

$$\phi_{\mathsf{CBA}} := \mu X_1 . \nu X_0 . \bigvee_{i \in \{0,1\}} \Diamond^i X_i,$$

expressing the existence of an infinite path that contains only finitely many non-accepting transitions. The *non-emptiness region* of the Co-Büchi automaton A is just $[\![\phi_{\mathsf{CBA}}]\!]_{\mathsf{A}}$ and consists of those states that accept a non-empty language, i.e. at least one word; the set $[\![\neg\phi_{\mathsf{CBA}}]\!]_{\mathsf{A}}$ is the *emptiness region* of A and consists of states that do not accept any word.

For any state $v \in [\![\phi_{\mathsf{CBA}}]\!]_A$, we can extract an accepting run ρ, i.e. an infinite path that starts at v and from some point on avoids transitions from $\overline{F}$ forever. Then a word w that is accepted by v can be read off from ρ. The non-emptiness region of CBA can also be specified by an *alternation-free* formula, e.g. we have

$$\phi_{\mathsf{CBA}} = \mu X.\, (\Diamond^1 X \vee \nu X.\, (\Diamond^0 X)).$$

Definition 4.3.2 (Non-emptiness regions of BA). For a Büchi automaton A, transitions with priority 2 are accepting, so we have $\Delta_2 = F$ and $\Delta_1 = \overline{F}$. The formula

$$\phi_{\mathsf{BA}} := \nu X_2.\mu X_1.\, \bigvee_{i\in\{1,2\}} \Diamond^i X_i$$

specifies the non-emptiness region of A.

The formula states the existence of some accepting run by expressing the existence of an infinite path on which infinitely many accepting transitions are passed. For any state $v \in [\![\phi_{\mathsf{BA}}]\!]_A$, we can extract an accepting run ρ, i.e. an infinite path that starts at v and visits some state from F infinitely often. We note that ϕ_{BA} is not equivalent to any alternation-free formula.

Definition 4.3.3 (Non-emptiness regions of PA). Let $\mathsf{A} = (V, \Sigma, \Delta, v_0, \alpha)$ be a parity automaton. The non-emptiness region of parity automata can be specified by the formula

$$\phi_{\mathsf{PA}} := \eta_{k-1} X_{k-1}.\, \ldots \eta_1 X_1.\eta_0 X_0.\, \bigvee_{0\le i<k} \Diamond^i X_i,$$

where for $0 \le i < k$, η_i is ν if i is even and μ if i is odd, that is, the k fixpoint operators in the formula are alternating between least and greatest fixpoints, where the innermost fixpoint operator is a greatest fixpoint operator.

The formula expresses the existence of an infinite path on which the highest priority that occurs infinitely often is even. Hence $[\![\phi_{\mathsf{PA}}]\!]_{\mathsf{A}}$ is just the set of states that accept a non-empty language. For any state $v \in [\![\phi_{\mathsf{PA}}]\!]_A$, we can extract an accepting run ρ, i.e. an infinite path that starts at v and uses some transition with even priority i infinitely often but uses all transitions with priority greater than i only finitely often.

We observe that ϕ_{PA} is a formula with alternation depth k. However, like all region formulas for the automata that we have considered, the non-emptiness formula for PA

utilizes only existential modal quantification and no universal modal quantification. The non-emptiness region of PA thus can equivalently be specified by a formula with alternation depth just 2. This mirrors the transformation from PA to equivalent NBA as defined in Definition 4.1.29. In the same way in which NBA are as expressive as NPA (both can recognize all ω-regular languages), the box-free μ-calculus restricted to alternation-depth 2 is as expressive as the box-free μ-calculus with arbitrary alternation-depth.

Fact 4.3.4. We have the following logical equivalence:

$$\phi_{\mathsf{PA}} = \bigvee_{0 \leq i < k, i \text{ even}} (\mu X_3.\nu X_2.\mu X_1.\Diamond^i X_2 \vee \bigvee_{j<i} \Diamond^j X_1 \vee \bigvee_{j>i} \Diamond^j X_3)$$

This formula has alternation-depth 3. The outermost least fixpoint can be replaced by an initial EF operator, so we even obtain the equivalence

$$\phi_{\mathsf{PA}} = \mathsf{EF} \bigvee_{0 \leq i < k,\, i \text{ even}} (\nu X_2.\mu X_1.\ \Diamond^i X_2 \vee \bigvee_{j<i} \Diamond^j X_1).$$

Definition 4.3.5. Dually, the set of nodes that are are non-empty in a *minimal priority* parity automaton, can be specified by the formula

$$\phi_{\mathsf{PA}_m} = \mathsf{EF} \bigvee_{0 \leq i < k,\, i \text{ even}} (\nu X_2.\mu X_1.\ \Diamond^i X_2 \vee \bigvee_{j>i} \Diamond^j X_1).$$

4.3.2 Fixpoint Formulas over Games

Like automata, games (V, E, α) can be seen as Kripke structures (V, E, π) where π evaluates atoms that indicate ownership of nodes. A problem that is related to game solving is *strategy checking*: to check whether some given strategy s wins a node v in a game A, it suffices to compute the non-emptiness region of the universal automaton (recall Definition 4.1.1) by imposing strategy s on the game; this non-emptiness region contains v if and only if s wins v. The adversarial nature of the two players Éloïse and Abélard complicates the specification of the acceptance region of games since the relevant formulas now contain boxes *and* diamonds.

Definition 4.3.6 (Winning regions of games). Let (V, E, α) be a parity game with k priorities. We put $\pi(\exists) = V_\exists$, $\pi(\forall) = V_\forall$ and

$$E^i(v) = \{w \in V \mid (v, w) \in E, \alpha(v, w) = i\},$$

where $v \in V$, $0 \leq i < k$. For all interpretations σ that map fixpoint variables to subsets of V and all $0 \leq i < k$, we put

$$[\![\Diamond^i X]\!]_\sigma = \{v \in V \mid E^i(v) \cap \sigma(X) \neq \emptyset\} \qquad [\![\Box^i X]\!]_\sigma = \{v \in V \mid E^i(v) \subseteq \sigma(X)\}$$

The *winning regions* of Éloïse in Co-Büchi, Büchi and parity games are specified by the formulas

$$\phi_{\mathsf{CBG}} := \mu X_1.\nu X_0. \bigvee_{i\in\{0,1\}} ((\exists \wedge \Diamond^i X_i) \vee (\forall \wedge \Box^i X_i))$$
$$\phi_{\mathsf{BG}} := \nu X_2.\mu X_1. \bigvee_{i\in\{1,2\}} ((\exists \wedge \Diamond^i X_i) \vee (\forall \wedge \Box^i X_i))$$
$$\phi_{\mathsf{PG}} := \eta X_{k-1}.\ldots\mu X_1.\nu X_0. \bigvee_{0\leq i<k} ((\exists \wedge \Diamond^i X_i) \vee (\forall \wedge \Box^i X_i))$$

The strictness of the alternation hierarchy of the μ-calculus [3] implies in particular that ϕ_{PG} cannot be expressed by a μ-calculus formula with alternation-depth less than k (and similarly that ϕ_{CBG} and ϕ_{BG} cannot be expressed as alternation-free formulas).

Definition 4.3.7 (Game time-outs). The nested time-outs from Definition 2.2.6 can be instantiated to parity games by defining the function $f : \mathcal{P}(V)^k \to \mathcal{P}(V)$ that is used in the definition by

$$f(V_0,\ldots,V_{k-1}) = [\![\bigvee_{0\leq i<k} ((\exists \wedge \Diamond^i X_i) \vee (\forall \wedge \Box^i X_i))]\!]_\sigma$$

where $V_i \subseteq V$, $\sigma(X_i) = V_i$ for $0 \leq i < k$. Then we say that a node $v \in V$ has *game time-outs* $\overline{m}$ if $v \in f^{\overline{m}}$; we write $x \in \mathsf{gto}(\overline{m})$ if x has game time-outs $\overline{m}$.

Given a node v with game time-outs $\overline{m} = (m_{k-1},\ldots,m_0)$, Éloïse has a strategy that ensures that in every play that starts at v, each odd priority $2i+1$ is visited at most $m_{2i+1}-1$ times before a priority greater than $2i+1$ is visited. By Lemma 2.2.7, Éloïse wins a node $v \in V$ if and only if there is some vector $\overline{m}$ such that v has game time-outs $\overline{m}$.

Definition 4.3.8 (Strategy checking). Fixing a strategy s for one of the two players turns a game into a universal automaton. If the strategy is positional, then the resulting automaton has finitely many states. Thus strategy checking, i.e. the computation of the set of nodes that are won by some positional strategy, is the same as computing the non-emptiness region of the according universal automaton.

Let (V, E, α) be a game. Without loss of generality, let $s : V_\exists \to V$ be a positional Éloïse-strategy and define

$$[\![\Box^i_s X]\!]_\sigma = \{v \in V_\forall \mid E^i(v) \subseteq \sigma(X)\} \cup \{v \in V_\exists \mid s(v) \in E^i(v) \cap \sigma(X)\},$$

where X is a fixpoint variable; notice that $\Box^i_s$ acts like standard $\Box^i$ on Abélard-nodes but imposes the strategy s on Éloïse-nodes, which then have at most one E^i-successor (for $0 \leq i < k$). The regions that Éloïse wins with strategy s in Co-Büchi, Büchi and parity

games are then defined by the formulas

$$\phi_{\mathsf{CBG}}(s) := \mu X_1.\nu X_0. \bigwedge_{i\in\{0,1\}} \Box_s^i X_i$$
$$\phi_{\mathsf{BG}}(s) := \nu X_2.\mu X_1. \bigwedge_{i\in\{1,2\}} \Box_s^i X_i$$
$$\phi_{\mathsf{PG}}(s) := \eta_{k-1}X_{k-1}.\ldots.\eta_1 X_1.\eta_0 X_0. \bigwedge_{0\leq i<k} \Box_s^i X_i$$

As mentioned above, these formulas define the acceptance regions of *universal* Co-Büchi, Büchi and parity *automata* and the latter two can thus can be simplified as follows:

$$\begin{aligned}
\phi_{\mathsf{BG}}(s) &= \nu X.\,(\mu X.\,\Box_s^1 X) \wedge \Box_s^1 X \wedge \Box_s^2 X \\
\phi_{\mathsf{PG}}(s) &= \eta_{k-1}X_{k-1}.\ldots.\eta_1 X_1.\eta_0 X_0. \bigwedge_{0\leq i<k} \Box_s^i X_i \\
&= \neg\overline{\eta_{k-1}}X_{k-1}.\ldots.\overline{\eta_1}X_1.\overline{\eta_0}X_0. \bigvee_{0\leq i<k} \Diamond_s^i X_i \\
&= \neg \bigvee_{0<i<k,i \text{ odd}} \mathsf{EF}_s(\nu X_2.\mu X_1.\,(\Diamond_s^i X_2 \vee \bigvee_{j<i} \Diamond_s^j X_1)) \\
&= \bigwedge_{0<i<k,i \text{ odd}} \mathsf{AG}_s(\mu X_1.\nu X_0.\,(\Box_s^i X_1 \wedge \bigwedge_{j<i} \Box_s^j X_0)),
\end{aligned}$$

where, for $0 \leq i < k$, η_i is ν if i is even and μ otherwise and where $\overline{\eta_i}$ is μ if i is even and ν otherwise; furthermore, $\mathsf{EF}_s\psi = \mu X.\,(\psi \vee \bigvee_{0\leq i<k} \Diamond_s^i X)$ and $\mathsf{AF}_s\psi = \nu X.\,(\psi \wedge \bigwedge_{0\leq i<k} \Box_s^i X)$. Notice that universal Büchi automata are the duals of existential Co-Büchi automata so that in contrast to the situation for existential automata, the acceptance region of universal *Büchi* automata can be specified by an alternation-free formula. However, $\mu X_1.\nu X_0.\, \bigvee_{i\in\{0,1\}}(\alpha(i) \wedge \Box_s X_i)$ cannot be expressed by an alternation-free formula and hence the region that strategy s wins in a *Co-Büchi* game cannot be defined by an alternation-free formula. A universal parity automaton is the complement of an existential parity automaton in which all priorities have been increased by one. Using Fact 4.3.4, we can transform this existential NPA to an NBA that has a universal Co-Büchi automaton as its dual. The acceptance region of this universal Co-Büchi automaton can be specified by the resulting formula that has alternation depth 2.

By definition, plays ρ of parity games are won by Éloïse if and only if the highest priority that is passed infinitely often in ρ is even, i.e. if $\max(\mathsf{Inf}(\alpha \circ \mathsf{trans}(\rho)))$ is even; in this sense, parity games as defined above are *maximal priority* parity games. Sometimes it is convenient to use *minimal priority* parity games in which Éloïse wins plays ρ for which the *least* priority that is passed infinitely often is even, i.e. if $\min(\mathsf{Inf}(\alpha \circ \mathsf{trans}(\rho)))$ is even. Regions in such parity games can be defined similarly as the regions in maximal

priority parity games, but with reversed order of priorities, i.e. with

$$E^i(v) = \{w \in V \mid (v,w) \in E, \alpha(v,w) = j - i\},$$

where $j = 2\lceil \frac{\mathsf{id}\times(\alpha)}{2} \rceil$.

4.3.3 Algorithmic Consequences

Given a Kripke structure of size n, the extension of a μ-calculus formula of size $\mathcal{O}(n)$ and with alternation-depth k can be computed in time $n^{\mathcal{O}(k)}$. Thus the results from the previous sections immediately imply the following standard results (see e.g. [21]).

Corollary 4.3.9. *The following problems are in* PTIME*:*

- *the non-emptiness problems of Co-Büchi, Büchi and parity automata,*
- *computing the winning regions of Co-Büchi and Büchi games,*
- *the strategy checking problems of Co-Büchi, Büchi and parity games.*

Solving parity games is in $\mathrm{NP} \cap \mathrm{Co{-}NP}$.

Solving parity games has been shown to even be contained in $\mathrm{UP} \cap \mathrm{Co{-}UP}$ (see e.g. [21]) and recently an algorithm has been introduced that solves parity games in quasipolynomial time [5,13].

4.3.4 Partial Automata and Partial Games

A *partial* (or *incomplete*) automaton $\mathsf{A} = (V, U, \Sigma, \Delta, v_0, \alpha)$ with k priorities is just an automaton $(V, \Sigma, \Delta, v_0, \alpha)$ together with a set U of *unexpanded* states such that for all $a \in \Sigma$ and $u \in U$, $\Delta(u,a)$ is undefined. Like in complete (i.e. standard) automata, the partial automaton A accepts a word w if there is an accepting run $\rho \in \mathsf{run}(\mathsf{A}, w)$. But if there is no such run, it is not necessarily the case that the word is not accepted by A. This is the case since unexpanded nodes from U can be expanded (i.e. be replaced with a sub-automaton, possibly using edges that lead back to V) and may then be part of an accepting run. The word w is only classified to be not accepted by A if $\mathsf{run}(\mathsf{A}, w)$ contains no accepting run and also no finite run that ends in a state from U; then there is no way to expand an unexpanded node such that the resulting automaton accepts w. This means that for each word w, incomplete automata may accept w, refuse w or neither. For the specification of regions of partial automata by logical formulas, we recall that $\overline{U}$ denotes the complement of U in V, i.e. the set of expanded states and put, for all fixpoint variables X, all $a \in \Sigma$, all $0 \leq i < k$ and all interpretations σ that map fixpoint variables to subsets of V,

$$[\![\Diamond_a^i X]\!]_\sigma = \{v \in V \mid \Delta_i(v,a) \cap \sigma(X) \cap \overline{U} \neq \emptyset\} \quad [\![\Box_a^i X]\!]_\sigma = \{v \in V \mid \Delta_i(v,a) \subseteq \sigma(X) \cap \overline{U}\}$$

and

$$[\![\Diamond^i X]\!]_\sigma = \bigvee_{a \in \Sigma} [\![\Diamond_a^i X]\!]_\sigma \qquad\qquad [\![\Box^i X]\!]_\sigma = \bigwedge_{a \in \Sigma} [\![\Box_a^i X]\!]_\sigma$$

Then in general, $[\![\neg\Box^i \psi]\!] \neq [\![\Diamond^i \neg\psi]\!]$ since it may be the case that there is some unexpanded i-successor, but no *expanded* i-successor that satisfies $\neg\psi$.

Similar definitions lead to the notion of *partial games* and corresponding modal operators that quantify over moves.

When computing regions of automata or games, the extension of the (negated) relevant formula contains exactly those states or nodes that are already known to be in the respective region. This means that the resulting algorithms are directly suitable for computing the respective regions in *partial* automata or games as well (a fact that has previously been noted in e.g. [17]). In particular, this enables on-the-fly game solving in which a (potentially large) game is constructed move by move and may possibly be solved before it has been fully constructed.

5 Satisfiability Checking for the Coalgebraic μ-Calculus

We now equip coalgebraic modal logic as introduced in Chapter 3 with fixpoint operators and obtain the *coalgebraic μ-calculus* which was introduced in [7] after previous results in the direction of coalgebraic fixpoint logic and automata for the relation lifting approach (cf. [35]) were obtained by Venema [55] and Kupke and Venema [30]. Fixpoint operators are introduced in the standard way, that is, by extending the syntax of coalgebraic modal logic with fixpoint variables and fixpoint operators and by interpreting the resulting formulas as functions; monotonicity of the involved predicate liftings and propositional operators then guarantees the existence of extremal fixpoints of these functions, allowing us to interpret the fixpoint operators.
Our primary interest will be in the satisfiability problem of the coalgebraic μ-calculus; this problem has been shown to be in ExpTime (under mild assumptions) in [7]. Least fixpoint operators require particular attention when checking satisfiability of formulas as they need to be satisfied after a *finite* number of unfolding steps whereas for the satisfaction of greatest fixpoint operators, infinite unfolding poses no problem. Unfolding of fixpoint formulas gives rise to new formulas that in turn evolve in the course of a satisfiability proof. We use the notion of *deferrals* in the upcoming technical development to denote formulas that fixpoint formulas may evolve to. We will detail decision procedures for various fragments of the coalgebraic μ-calculus; these procedures essentially construct and solve *satisfiability games* that track deferrals through pre-tableaux, checking for their eventual satisfaction, thus ruling out reliance on least fixpoints that are never finished.

5.1 The Coalgebraic μ-Calculus

We add *fixpoint variables* to the syntax of coalgebraic modal logic and require that the obtained formulas are monotone w.r.t. set inclusion when interpreted as functions. E.g. the formula $\phi = \top \wedge \heartsuit(X \vee \overline{\heartsuit}Y)$ can be seen as a function taking two sets of states A and B as input and returning $[\![\phi]\!]_{[X \mapsto A, Y \mapsto B]}$ as result, that is, the extension of ϕ where X is interpreted as A and Y is interpreted as B.

Definition 5.1.1 (Coalgebraic μ-calculus, syntax). We fix a set $\mathbf{V}$ of fixpoint variables and a similarity type Λ. The set of *coalgebraic μ-calculus formulas* $C\mu(\Lambda, \mathbf{V})$ (or just $C\mu$) over Λ and $\mathbf{V}$ is defined by the following grammar:

$$\mathcal{C}\mu(\Lambda, \mathbf{V}) \ni \psi_1, \psi_2 ::= \top \mid \bot \mid X \mid \psi_1 \wedge \psi_2 \mid \psi_1 \vee \psi_2 \mid \heartsuit\psi_1 \mid \mu X.\, \psi_1 \mid \nu X.\, \psi_1,$$

where $\heartsuit \in \Lambda$ and $X \in \mathbf{V}$. The *size* $|\psi|$ of a formula $\psi \in \mathcal{C}\mu$ is just its length over the alphabet $\{\top, \bot, \vee, \wedge\} \cup \Lambda \cup \mathbf{V} \cup \{\nu X., \mu X. \mid X \in \mathbf{V}\}$.

Throughout this section, we use η to denote one of the fixpoint operators μ or ν. We refer to formulas of the form $\eta X.\, \psi$ as *fixpoint literals* and to formulas of the form

$\heartsuit\psi$ as *modal literals*. The operators μ and ν *bind* their variables, inducing a standard notion of *free variables* in formulas. We denote the set of *free variables* of a formula ψ, i.e. the set of those fixpoint variables that are free in ψ, by $\mathsf{FV}(\psi)$. Similarly, the set $\mathsf{BV}(\psi)$ of *bound variables* of a formula ψ is the set of fixpoint variables X for which ψ contains some subformula $\eta X.\,\phi$. A formula ψ with $\mathsf{FV}(\psi) = \emptyset$ is *closed* and a formula that is not closed is *open*. We say that ϕ *occurs free* in ψ if ϕ occurs as a subformula in ψ that is not in the scope of any fixpoint operator. Given a formula $\mu X.\,\psi$, a free occurrence of the fixpoint variable X within ψ is referred to as a *μ-variable*; similarly, for $\nu X.\,\psi$, free occurrences of X in ψ are *ν-variables*. We write $\psi \leq \phi$ ($\psi < \phi$) to indicate that ψ is a (proper) subformula of ϕ. We refer to $\mu X_1.\nu X_0.$ as the *Co-Büchi operator* and to $\nu X_2.\mu X_1.$ as the *Büchi operator*; they are to be applied to formulas $\phi(X_0, X_1)$ and $\phi(X_1, X_2)$ with exactly two free variables X_0, X_1 and X_1, X_2, resulting in *Co-Büchi formulas* $\mu X_1.\nu X_0.\phi(X_0, X_1)$ and *Büchi formulas* $\nu X_2.\mu X_1.\phi(X_1, X_2)$, respectively. The *parity operator* $\eta X_{k-1}. \ldots \mu X_1.\nu X_0.$ is applied to arguments $\phi(X_0, X_1, \ldots, X_{k-1})$ with k free variables and constructs the *parity formula* $\eta X_{k-1}. \ldots \mu X_1.\nu X_0.\ \phi(X_0, X_1, \ldots, X_{k-1})$; as usual, η is ν if $k-1$ is even, and μ otherwise. Throughout, we *restrict to formulas that are guarded*, i.e. have at least one modal operator between any occurrence of a variable X and an enclosing binder ηX. (This is standard although possibly not without loss of generality [14].) Formulas are *clean* if all fixpoint variables they use are distinct, and *irredundant* if $X \in \mathsf{FV}(\psi)$ for all subformulas $\eta X.\,\psi$.

Formulas are evaluated over T-coalgebras $\mathcal{C} = (W, \xi)$, consisting of a set W of states and a coalgebra structure $\xi : W \to T(W)$. We assume that each $\heartsuit \in \Lambda$ comes with a T-predicate lifting $[\![\heartsuit]\!]$ that is monotone w.r.t. set inclusion, i.e. we require that for all sets U and all $A, B \subseteq U$,

$$A \subseteq B \text{ implies } [\![\heartsuit]\!]_U A \subseteq [\![\heartsuit]\!]_U B.$$

Furthermore, we require, as before, that Λ contains, for each $\heartsuit \in \Lambda$, a dual modal operator $\overline{\heartsuit}$ such that for all U and $A \subseteq U$, we have $\overline{[\![\heartsuit]\!]_U}(A) = [\![\heartsuit]\!]_U(\overline{A})$, recalling that $\overline{A} = \{x \in U \mid x \notin A\}$ is the complement of A in U; finally, we require $\overline{\overline{\heartsuit}} = \heartsuit$.

Definition 5.1.2 (Coalgebraic μ-calculus, semantics). Given a T-coalgebra $\mathcal{C} = (C, \xi)$, an *interpretation* (of the fixpoint variables) is a partial mapping $i : \mathbf{V} \rightharpoonup \mathcal{P}(C)$, assigning a set of states from C to each fixpoint variable $X \in \mathsf{dom}(i)$. The extension $[\![\phi]\!]_i$ of a coalgebraic fixpoint formula ϕ w.r.t. i (where $\mathsf{FV}(\phi) \subseteq \mathsf{dom}(i)$) is defined recursively:

$$\begin{aligned}
[\![\bot]\!]_i &= \emptyset & [\![\top]\!]_i &= C \\
[\![X]\!]_i &= i(X) & [\![\heartsuit\psi]\!]_i &= \xi^{-1}[[\![\heartsuit]\!]_C [\![\psi]\!]_i] \\
[\![\psi_1 \wedge \psi_2]\!]_i &= [\![\psi_1]\!]_i \cap [\![\psi_2]\!]_i & [\![\psi_1 \vee \psi_2]\!]_i &= [\![\psi_1]\!]_i \cup [\![\psi_2]\!]_i \\
[\![\mu X.\,\psi]\!]_i &= \mathsf{LFP}[\![\psi]\!]_i^X & [\![\nu X.\,\psi]\!]_i &= \mathsf{GFP}[\![\psi]\!]_i^X,
\end{aligned}$$

where $[\![\psi]\!]_i^X(A) = [\![\psi]\!]_{i[X \mapsto A]}$ for $A \subseteq C$; here, $i[X \mapsto A](X) = A$ and $i[X \mapsto A](Y) = i(Y)$ for $X \neq Y$. Recall from Definition 2.2.1 that LFP and GFP compute the least and the

greatest fixpoint of the argument function, respectively. Since the predicate liftings for Λ are assumed to be monotone w.r.t. set inclusion, we have that for all coalgebraic formulas ψ that use modal operators from Λ and all functions i that map fixpoint variables to subsets of C, the function $[\![\psi]\!]_i^X$ that maps sets A to $[\![\phi]\!]_{i[X\mapsto A]}$ is monotone w.r.t. set inclusion so that the above indeed constitutes a definition. Given a closed coalgebraic fixpoint formula ϕ and a state $x \in C$, we write $\mathcal{C}, x \models \phi$ for $x \in [\![\phi]\!]_\epsilon$, where ϵ denotes the empty function. If ψ is closed, then $[\![\psi]\!]_i$ does not depend on i, so we just write $[\![\psi]\!]$. A coalgebraic μ-calculus formula ψ is *satisfiable* if there is a coalgebra $\mathcal{C} = (C, \xi)$ and a state $x \in C$ such that $\mathcal{C}, x \models \psi$. W.l.o.g., we restrict our attention to clean and irredundant formulas, when considering the satisfiability of formulas.

Example 5.1.3. We revisit Example 3.1.8 and now consider the logics that we obtain when adding fixpoint operators to those coalgebraic logics from that example that have monotone predicate liftings.

1. Extending the logic **K** of Kripke frames with fixpoint operators yields the standard uni-modal μ-calculus (also referred to as the *relational μ-calculus*), containing – as a simple fragment – the logic CTL with operators $\mathsf{A}\,\phi\,\mathsf{U}\,\psi = \mu X.\,(\psi \vee (\phi \wedge \Box X))$ and $\mathsf{E}\,\phi\,\mathsf{U}\,\psi = \mu X.\,(\psi \vee (\phi \wedge \Diamond X))$ stating that "on all paths, ϕ holds until ψ holds" and "there is a path on which ϕ holds until ψ holds", respectively.
2. The *serial μ-calculus* is obtained by extending the serial logic **KD** with fixpoint operators. However, seriality can also be expressed in the standard μ-calculus by means of the formula $\mathsf{AG}\Diamond\top = \nu X.\,(\Diamond\top \wedge \Box X)$.
3. The predicate liftings for classical modal logic **E** from Example 3.1.8 are not monotone, so we do not obtain a corresponding μ-calculus.
4. Extending serial monotone modal logic with fixpoint operators yields the *serial monotone μ-calculus*, containing Parikh's game logic [37] as a fragment.
5. The standard *(multi-)modal μ-calculus* is built over Hennessy-Milner logic $\mathcal{HML}$ and contains formulas such as e.g. $\nu X.\,\mu X.\,(\Diamond_a Y \vee \Diamond_b X)$, stating that there is a path containing infinitely many a-transitions and only b-transitions between any two a-transitions.
6. Extending Pauly's coalition logic with fixpoint operators, we obtain the *alternating-time μ-calculus* (AMC) [45] which contains alternating-time temporal logic (ATL) [1] as a fragment, where in the latter, coalitions are formed by *finite* sets of agents.
7. The *probabilistic μ-calculus* [7] is obtained by adding fixpoint operators to probabilistic modal logic. It contains e.g. modal formulas $\mu X.(\phi \vee L_p X)$ stating that there is a path with probability p at each step that eventually reaches a state that satisfies ϕ.
8. While the predicate liftings of the conditional logics mentioned in Example 3.1.8 are not monotone and we do not directly obtain corresponding μ-calculi, the monotonicity of formulas can be ensured by requiring that fixpoint variables occur only in the second arguments of conditional modal operators (allowing formulas such as e.g. $\mu X.\,p \Rightarrow X$).

Since we use naîve substitution (instead of capture avoiding substitution) in Definition 5.1.2, we need the following lemma to be able to show that the process of unfolding fixpoint literals preserves the extension of formulas.

Lemma 5.1.4 (Substitution). *If* $\mathsf{BV}(\psi) \cap \mathsf{FV}(\phi) = \emptyset$*, then*

$$[\![\psi]\!]_i^X [\![\phi]\!]_i = [\![\psi[X \mapsto \phi]]\!]_i.$$

Proof. The proof is by induction over ψ. If ψ is closed, then $[\![\psi]\!]_i^X [\![\phi]\!]_i = [\![\psi]\!] = [\![\psi[X \mapsto \phi]]\!]_i$. Otherwise, if $\psi = X$, then $[\![X]\!]_i^X [\![\phi]\!]_i = [\![\phi]\!]_i = [\![X[X \mapsto \phi]]\!]_i$. If $\psi = Y \neq X$, then $[\![Y]\!]_i^X [\![\phi]\!]_i = [\![Y]\!]_i = [\![Y[X \mapsto \phi]]\!]_i$. The cases for disjunction, conjunction and modal operators are straightforward. If $\psi = \eta X.\, \theta$, then $[\![\eta X.\, \theta]\!]_i^X [\![\phi]\!]_i = [\![\eta X.\, \theta]\!]_i = [\![(\eta X.\, \theta)[X \mapsto \phi]]\!]_i$. If $\psi = \eta Y.\, \theta$ for $Y \neq X$, then $[\![\eta Y.\, \theta]\!]_i^X [\![\phi]\!]_i = \eta [\![\theta]\!]^Y_{i[X \mapsto [\![\phi]\!]_i]} = \eta [\![\theta[X \mapsto \phi]]\!]_i^Y = [\![(\eta Y.\, (\theta[X \mapsto \phi]))]\!]_i = [\![(\eta Y.\, \theta)[X \mapsto \phi]]\!]_i$, where the second equality holds since for all A,

$$\begin{aligned}
[\![\theta]\!]^Y_{i[X \mapsto [\![\phi]\!]_i]}(A) &= [\![\theta]\!]_{i[X \mapsto [\![\phi]\!]_i][Y \mapsto A]} \\
&= [\![\theta]\!]_{i[Y \mapsto A][X \mapsto [\![\phi]\!]_i]} \\
&= [\![\theta]\!]^X_{i[Y \mapsto A]} [\![\phi]\!]_i \\
&= [\![\theta]\!]^X_{i[Y \mapsto A]} [\![\phi]\!]_{i[Y \mapsto A]} \\
&= [\![\theta[X \mapsto \phi]]\!]_{i[Y \mapsto A]} \\
&= [\![\theta[X \mapsto \phi]]\!]_i^Y (A),
\end{aligned}$$

where the second equality holds since $X \neq Y$, the fourth equality holds since by assumption, $Y \notin \mathsf{FV}(\phi)$ and the fifth equality is by the induction hypothesis. □

Lemma 5.1.5. *If* $\mathsf{BV}(\psi) \cap \mathsf{FV}(\eta X.\, \psi) = \emptyset$*, then*

$$[\![\eta X.\, \psi]\!]_i = [\![\psi[X \mapsto \eta X.\, \psi]]\!]_i.$$

Proof. We show the proof for the case $\eta = \mu$; the proof for the other case is analogous. By Lemma 5.1.4 and since $\mathsf{LFP}[\![\psi]\!]_i^X$ is a fixpoint, we have

$$[\![\mu X.\, \psi]\!]_i = \mathsf{LFP}[\![\psi]\!]_i^X = [\![\psi]\!]_i^X(\mathsf{LFP}[\![\psi]\!]_i^X) = [\![\psi]\!]_i^X [\![\mu X.\, \psi]\!]_i = [\![\psi[X \mapsto \eta X.\, \psi]]\!]_i$$

□

In clean formulas, fixpoint variables are bound at most once, so we have $\mathsf{BV}(\psi) \cap \mathsf{FV}(\eta X.\, \psi) = \emptyset$ for all subformulas $\eta X.\, \psi$.

Fact 5.1.6 (Syntactic substitution). *If* $(\{X\} \cup \mathsf{BV}(\psi)) \cap \mathsf{dom}(\sigma) = \emptyset$ *and for each* $Y \in \mathsf{FV}(\psi)$*,* $(\{X\} \cup \mathsf{BV}(\psi)) \cap \mathsf{FV}(\sigma(Y)) = \emptyset$*, then*

$$(\psi\sigma)[X \mapsto (\phi\sigma)] = (\psi[X \mapsto \phi])\sigma.$$

Proof. The proof is by induction over ψ. If $\psi = X$, then note that by assumption $X \notin \mathsf{dom}(\sigma)$ so that $(X\sigma)[X \mapsto (\phi\sigma)] = X[X \mapsto \phi\sigma] = \phi\sigma = (X[X \mapsto \phi])\sigma$. If $\psi = Y \neq X$, then we have by assumption $X \notin \mathsf{FV}(\sigma(Y))$ so that $(Y\sigma)[X \mapsto (\phi\sigma)] = \sigma(Y)[X \mapsto \phi\sigma] = \sigma(Y) = Y\sigma = (Y[X \mapsto \phi])\sigma$. The cases for conjunction, disjunction and modal operators are straightforward. If $\psi = \eta X.\, \psi$, then $((\eta X.\, \psi)\sigma)[X \mapsto (\phi\sigma)] = (\eta X.\, \psi)\sigma = ((\eta X.\, \psi)[X \mapsto \phi])\sigma$. If $\psi = \eta Y.\, \psi$ for $X \neq Y$, then we have by assumption that $Y \notin \mathsf{dom}(\sigma)$ and for any $Z \in \mathsf{FV}(\psi)$, $Y \notin \mathsf{FV}(\sigma(Z))$ so that $((\eta Y.\, \psi)\sigma)[X \mapsto (\phi\sigma)] = \eta Y.\, (\psi\sigma)[X \mapsto (\phi\sigma)] = \eta Y.\, (\psi\sigma[X \mapsto (\phi\sigma)]) = \eta Y.\, ((\psi[X \mapsto \phi])\sigma) = (\eta Y.\, (\psi[X \mapsto \phi]))\sigma = ((\eta Y.\, \psi)[X \mapsto \phi])\sigma$, where the third equality is by the induction hypothesis. □

Definition 5.1.7. We define several fragments of the coalgebraic μ-calculus that are obtained by imposing various constraints on the syntactic structure of formulas.

1. The *depth-1 linear* (or *depth-1 single-use*) *fragment* $C\mu_{\mathsf{lin1}}$ is obtained by requiring that in each least fixpoint formula $\mu X.\, \psi$, X has exactly one free occurrence in ψ and this occurrence is of the shape $\heartsuit X$ for some $\heartsuit \in \Lambda$, where, $\heartsuit X$ is not in the scope of further modal operators within ψ. Notice that important logics such as CTL, ATL and PLTL are covered by this fragment.
2. The *linear* (or *single-use*) *fragment* $C\mu_{\mathsf{lin}}$ is obtained by requiring that for every least fixpoint formula $\mu X.\, \psi$, X has exactly one free occurrence in ψ and this occurrence is of the shape $\heartsuit X$ for some $\heartsuit \in \Lambda$. Hence every depth-1 linear formula is linear.
3. The *flat* (or *single-variable*) *fragment* $C\mu_{\mathsf{flat}}$ of the coalgebraic μ-calculus [53,23] is obtained by requiring that in each least fixpoint formula $\mu X.\, \psi$, the only free fixpoint variable in ψ is X. Hence every linear formula is flat.
4. The *alternation-free fragment* $C\mu_{\mathsf{af}}$ of the coalgebraic μ-calculus [25] is obtained by prohibiting formulas in which some subformula contains both a free ν-variable and a free μ-variable. E.g. $\mu X.\, \mu Y.\, (\Box X \wedge \Diamond Y \wedge \nu Z.\, \Diamond Z)$ is alternation-free but $\nu Z.\, \mu X.\, (\Box X \wedge \nu Y.\, (\Diamond Y \wedge \Diamond Z))$ is not. Every flat formula is alternation-free.
5. The *aconjunctive fragment* (introduced for the relational μ-calculus in [28]) $C\mu_{\mathsf{ac}}$ is obtained by requiring that for every conjunction $\psi \wedge \phi$ that occurs as a subformula of fixpoint literals, at most one of the conjuncts ψ, ϕ contains an *active* μ-variable. Here, an active μ-variable is a fixpoint variable that occurs free in $\psi \wedge \phi$ and either is a μ-variable or is a ν-variable X that occurs free in $\psi \wedge \phi$ and can be transformed to a formula that contains a free μ-variable by replacing X with its binding fixpoint literal and then repeatedly replacing free ν-variables in the resulting formula with their binding fixpoint literals.
6. The *alternation-free aconjunctive fragment* $C\mu_{\mathsf{ac,af}}$ is obtained by requiring both, alternation-freeness and aconjunctivity. Linear formulas are alternation-free and aconjunctive.
7. When no syntactic restrictions are imposed, we obtain the *full coalgebraic μ-calculus* $C\mu$.

The following diagram shows the relation between the various fragments of the coalgebraic μ-calculus:

$$\begin{array}{ccccccc} & & C\mu_{\mathsf{flat}} & \subsetneq & C\mu_{\mathsf{af}} & \subsetneq & C\mu \\ & & \cup\!\!\!\!\!| & & \cup\!\!\!\!\!| & & \cup\!\!\!\!\!| \\ C\mu_{\mathsf{lin1}} & \subsetneq & C\mu_{\mathsf{lin}} & \subsetneq & C\mu_{\mathsf{ac,af}} & \subsetneq & C\mu_{\mathsf{ac}} \end{array}$$

5.2 Satisfiability Games for the Coalgebraic μ-Calculus

When deciding the satisfiability of a coalgebraic fixpoint formula, it is necessary to construct a model for or a refutation of the formula. This can be achieved by checking whether it is possible to construct a structure (labelled with subsets of the *Fischer-Ladner closure* – see Definition 5.2.8 or [29] – of the input formula) such that the label of some node in the structure contains the input formula and such that all least fixpoint literals from the labels of nodes are satisfied after a finite number of unfolding steps. In the relational setting, such structures have been referred to as *tableaux* [14]; here we generalize this concept to the coalgebraic setting and refer to such structures as *timed-out tableaux*; relying on the time-outs, models can be extracted from timed-out tableaux. Thus it is sufficient if successful runs of satisfiability checking algorithms guarantee the existence of a timed-out tableau for the input formula. However, timed-out tableaux cannot be extracted directly from *pre-tableaux* (that is, labelled graphs that are obtained by repeatedly applying the standard tableau rules (including fixpoint unfolding rules) to labels until every node is expanded). This is the case since pre-tableaux may contain so-called *bad branches* [14], that is, cycles in which satisfaction of least fixpoint literals is postponed indefinitely.

Remark 5.2.1 (No filtration). Related to this is the observation that timed-out tableaux may contain different nodes with different time-out information but identical labels. It was shown e.g. in [10] that in the construction of a model for a given satisfiable fixpoint formula in the relational setting, the standard method of filtering ([2]) pre-tableaux (as supplied by any successful run of a standard satisfiability checking algorithm) through the Fischer-Ladner closure and then imposing *coherent* successor structures (in which modal operators are satisfied) on the obtained carrier sets fails. There is no way to ensure the satisfaction of least fixpoint literals since the filtration process may identify nodes with identical labels but different time-out information; hence the resulting filtered structures may have loops in which some least fixpoint literal is not satisfied.

Thus satisfiability checking procedures for the *relational* μ-calculus, such as the one described in [14] typically employ automata theoretic methods to detect and avoid the mentioned bad branches. To this end, *tracking automata* [14] are used to nondeterministically track single formulas through pre-tableaux and recognize the bad branches. As branches are infinite paths, these tracking automata are automata on infinite words (where letters identify single rule applications and choices of conclusion nodes in pre-tableaux); the acceptance conditions of tracking automata are Co-Büchi condition for

alternation-free logics and Büchi conditions for logics that allow alternation of least and greatest fixpoints. Since we want to obtain timed-out tableaux in which no branch contains *any* unsatisfied fixpoint formula, we have to track formulas through pre-tableaux simultaneously. This can be achieved by first determinizing and then complementing the tracking automata. The resulting automata then recognize the *good branches* in pre-tableaux, i.e. branches that do not contain unsatisfied least fixpoint formulas; these good branches may then be used to construct timed-out tableaux. Thus satisfiability games over the carriers of the determinized and complemented tracking automata can be used to decide whether timed-out tableaux, i.e. pre-tableaux in which all branches are good branches, exist.

Interestingly, the construction of satisfiability games via automata works in (almost) the same way for the relational and the *coalgebraic* μ-calculus. That is, the structure of satisfiability proofs for μ-calculi appears to be largely independent of the concrete modal logics over which the calculi are built.

We now give a short overview of the game methods that we obtain for the various fragments of the coalgebraic μ-calculus from Definition 5.1.7. Here we make use of the automata theoretic concepts, constructions and results from Chapter 4. For a detailed definition of the satisfiability games, see Sections 5.2.3 and 5.2.3 below. In the following, we let n and k denote the size and the *alternation-depth* (see Definition 5.2.9 below) of input formulas, respectively.

1. In the depth-1 fragment $C\mu_{\mathsf{lin1}}$, bad branches can be recognized by limit-stationary CBA of size n. By Lemma 4.1.13, these automata can be determinized to DBA of size at most $n \cdot 2^n$; models have size at most $n \cdot 2^n$. In the relational case, the resulting Co-Büchi games can be solved in time $2^{\mathcal{O}(n)}$.
2. In the flat and alternation-free fragments $C\mu_{\mathsf{flat}}$ and $C\mu_{\mathsf{af}}$, respectively, we need NCBA to recognize bad branches in pre-tableaux. By Lemma 4.1.18, these automata can be determinized to DBA of size at most 3^n; models have size at most 3^n. In the relational case the resulting Co-Büchi games can again be solved in time $2^{\mathcal{O}(n)}$.
3. In the alternation-free and aconjunctive fragment $C\mu_{\mathsf{ac,af}}$, we can directly use the method from the previous item to decide satisfiability and obtain models of size 3^n. However, it is also possible to define smaller satisfiability games via *focusing*: instead of determinizing tracking automata, we can rely on aconjunctivity to directly define DCBA that track *single* formulas. Then it is possible to define satisfiability games that focus single formulas but in which Abélard is allowed to do the *refocusing*, i.e. to choose any least fixpoint formula as new focus whenever the previous focus has been finished. The nodes in the resulting Co-Büchi games are sets of formulas annotated with single formulas; hence they are of size at most $n \cdot 2^n$. However, the construction of timed-out tableaux from winning strategies in the these games is more involved than for standard games as there may be two (or more) refocusing moves at a node in the game that correspond to the same application of a tableau rule. Thus the

obtained models are of size at most $n \cdot 3^n$, which means that they are unnecessarily large.

4. In the linear fragment $C\mu_{\mathsf{lin}}$, bad branches can be recognized by limit-linear CBA of size n. By Lemma 4.1.13, these automata can be determinized to DBA of size at most $n^2 \cdot 2^n$; models have size at most $n^2 \cdot 2^n$. In the relational case, the resulting Co-Büchi games can be solved in time $2^{\mathcal{O}(n)}$. Alternatively, one also can use the satisfiability games via focusing of size at most $n \cdot 2^n$ as described in item 4.
5. In the aconjunctive fragment $C\mu_{\mathsf{ac}}$, bad branches can be recognized by limit-deterministic PA. We use Lemmas 4.1.31 and 4.1.33 to translate these tracking automata to equivalent limit-deterministic BA of size at most nk; then we use the permutation determinization method and the accompanying Theorem 4.1.23 to obtain equivalent DPA. We refer to the resulting satisfiability games as *permutation games*. The DPA are of size at most $e \cdot (nk)!$ and have $nk + 1$ priorities; satisfiable (weakly) aconjunctive formulas have models of size at most $e \cdot (nk)!$. In the relational case, the games can be solved in time $\mathcal{O}((nk)!^{nk+1})$.
6. In the full coalgebraic μ-calculus $C\mu$, bad branches can be recognized by general NPA. We use Lemma 4.1.31 to translate them to equivalent NBA of size at most nk; then we use the compartmentalized Safra/Piterman construction and Theorem 4.1.27 to obtain equivalent DPA. The resulting DPA are of size at most $n!(nk)^{nk+1}$ and have $nk+1$ priorities; satisfiable coalgebraic fixpoint formulas have models of size at most $n!(nk)^{nk+1}$. In the relational case, the games can be solved in time $\mathcal{O}((n!(nk)^{nk+1})^{nk+1})$.

As these game methods construct satisfiability games by determinizing automata, the size of the resulting games is exponential in the size of the input formulas. While the search space grows exponentially with the input formula's sizes, actual models or refutations may be reasonably small. Thus it is desirable to make the methods work *on-the-fly*; by this we mean that the determinized automata and the corresponding games are constructed step by step and that the resulting partial satisfiability games (c.f. Section 4.3.4) can be solved at any point. To this end, we will now introduce a so-called *global caching algorithm* that enables on-the-fly construction and solving of satisfiability games for (fragments of) the coalgebraic μ-calculus.

5.2.1 A Global Caching Algorithm

The term *global caching* describes a family of single-pass tableau algorithms [18,20] that build graph-shaped pre-tableaux in so-called *expansion* steps, with no label ever generated twice, and attempt to terminate before the pre-tableaux are completely expanded by means of intermediate *propagation* of satisfiability and/or unsatisfiability through partially expanded pre-tableaux. In terms of games, the propagation step in global caching algorithms corresponds to solving partial games. Global caching offers room for heuristic optimization, regarding standard tableau optimizations as well as the order in which expansion and propagation steps are triggered, and has been shown

to perform competitively in practice; see [20] for an evaluation of heuristics in global caching for the description logic $\mathcal{ALCI}$. A global caching algorithm for PDL has been described by Goré and Widmann [19]; finding an optimal global caching algorithm even for CTL has been named as an open problem as late as 2014 [15] (a non-optimal, doubly exponential algorithm is known [15]). Here we provide optimal, that is single exponential, global caching algorithms for (fragments of) the coalgebraic μ-calculus.

We proceed to describe the general concepts and notions of global caching algorithms. First off, we need some syntactic notions regarding decomposition of fixpoint formulas. Recall that unfolding a fixpoint literal $\chi := \eta X.\, \psi$ results in the formula $\psi[X \mapsto \chi]$; eventual satisfaction of this formula potentially depends on the eventual satisfaction of all those of its subformulas that contain χ. We represent these subformulas in decomposed form as *deferrals* [25] of the shape (α, σ) where σ is a sequence of substitutions that replace fixpoint variables with fixpoint literals and where α is an open subformula that occurs free in the fixpoint literal from the first substitution in σ. Before we formally define deferrals, we consider an example.

Example 5.2.2. Consider the three nested fixpoints

$$\chi_1 := \mu X.\, \psi_1 \qquad \chi_2 := \nu Y.\psi_2 \qquad \chi_3 := \mu Z.\psi_3$$

where

$$\psi_1 = p \vee \Box\chi_2 \qquad \psi_2 = q \wedge \Diamond\chi_3 \qquad \psi_3 = r \to \Diamond(X \wedge (Z \vee Y)).$$

Unfolding χ_1 results in the formula $\psi_1[X \mapsto \chi_1]$. This formula contains the greatest fixpoint literal $\chi_2[X \mapsto \chi_1] = (\nu Y.\psi_2)[X \mapsto \chi_1] = \nu Y.(\psi_2[X \mapsto \chi_1])$ as a subformula, which in turn unfolds to $(\psi_2[X \mapsto \chi_1])[Y \mapsto \nu Y.(\psi_2[X \mapsto \chi_1])] = (\psi_2[Y \mapsto \chi_2])[X \mapsto \chi_1]$. Finally $(\psi_2[Y \mapsto \chi_2])[X \mapsto \chi_1]$ contains $(\chi_3[Y \mapsto \chi_2])[X \mapsto \chi_1] = ((\mu Z.\psi_3)[Y \mapsto \chi_2])[X \mapsto \chi_1] = (\mu Z.(\psi_3[Y \mapsto \chi_2]))[X \mapsto \chi_1] = \mu Z.((\psi_3[Y \mapsto \chi_2])[X \mapsto \chi_1])$ as a subformula that in turn unfolds to $((\psi_3[Z \mapsto \chi_3])[Y \mapsto \chi_2])[X \mapsto \chi_1]$. All the mentioned formulas may be relevant for the satisfaction of χ_1 at some state. We will define deferrals in such a way that

$$(\psi_1, [X \mapsto \chi_1])$$
$$(\psi_2, [Y \mapsto (\chi_2[X \mapsto \chi_1]), X \mapsto \chi_1])$$
$$(\psi_3, [Z \mapsto ((\chi_3[Y \mapsto \chi_2])[X \mapsto \chi_1]), Y \mapsto (\chi_2[X \mapsto \chi_1]), X \mapsto \chi_1])$$

are χ_1-deferrals (since they all belong to the outermost fixpoint χ_1) that induce the above-mentioned formulas.

Definition 5.2.3 (Sequential unfolding). Given two substitutions σ and κ, we define their composite $\sigma;\kappa$ as $\sigma;\kappa(X) = (\sigma(X))\kappa$ for all $X \in \mathbf{V}$.

Given fixpoint literals $\chi_i = \eta X_i.\, \psi_i$, $i = 1, \ldots, n$, we say that a substitution $\sigma = [X_1 \mapsto \chi_1]; \ldots; [X_n \mapsto \chi_n]$ *sequentially unfolds* χ_n if $\chi_i <_f \chi_{i+1}$ for all $1 \leq i < n$, where

we write $\psi <_f \eta X.\,\phi$ if $\psi \leq \phi$ and ψ is open and occurs free in ϕ (i.e. σ unfolds a nested sequence of fixpoints in χ_n innermost-first). We define $|[X_1 \mapsto e_1]; \ldots; [X_n \mapsto e_n]| = n$. We say that a formula χ is *irreducible* if for every substitution $[X_1 \mapsto \chi_1]; \ldots; [X_n \mapsto \chi_n]$ that sequentially unfolds χ_n, we have that $\chi = \chi_1([X_2 \mapsto \chi_2]; \ldots; [X_n \mapsto \chi_n])$ implies $n = 1$ (i.e. $\chi = \chi_1$). An *eventuality* is an irreducible closed least fixpoint literal. Given a formula α and a substitution $\sigma = [X_1 \mapsto e_1]; \ldots; [X_n \mapsto e_n]$, the pair (α, σ) *induces* the formula $\alpha\sigma$.

Note that sequential unfolding is not stable under α-equivalence, i.e. if $[X_1 \mapsto \chi_1]; \ldots; [X_i \mapsto \chi_i]; \ldots; [X_n \mapsto \chi_n]$ sequentially unfolds χ_n, then $[X_1 \mapsto \chi_1]; \ldots; [X_i \mapsto \chi_i']; \ldots; [X_n \mapsto \chi_n]$ does in general not sequentially unfold χ_n, where χ_i' is obtained from χ_i by renaming bound fixpoint variables.

A fixpoint literal is irreducible if it is not an unfolding $\psi[X \mapsto \eta X.\,\psi]$ of a fixpoint literal $\eta X.\,\psi$. In particular we have

Lemma 5.2.4. *Every clean irredundant fixpoint literal is irreducible.*

Proof. Let χ be a clean irredundant fixpoint literal. If χ is of the shape $\chi = \chi_1([X_2 \mapsto \chi_2]; \ldots; [X_n \mapsto \chi_n])$ where $[X_2 \mapsto \chi_2]; \ldots; [X_n \mapsto \chi_n]$ sequentially unfolds χ_n and where $\chi_1 = \eta X_1.\,\psi_1$, then $\chi_1 \leq \chi_n$ by Lemma 5.2.7 below, and since χ is clean and hence binds the fixpoint variable X_1 just once, we have $n = 1$. □

Recall that we assume cleanness for target formulas and hence all closed fixpoint operators in target formulas are irreducible.

Definition 5.2.5 (Deferrals). A formula ψ *belongs* to a closed irreducible fixpoint θ_n, or is a θ_n*-deferral*, if $\psi = \alpha\sigma$ for some substitution $\sigma = [X_1 \mapsto \theta_1]; \ldots; [X_n \mapsto \theta_n]$ that sequentially unfolds θ_n and some $\alpha <_f \theta_1$. We denote the set of θ_n-deferrals by $\mathsf{dfr}(\theta_n)$. Given a deferral $\alpha\sigma$, we refer to α and σ as the *base* and the *sequence* of the deferral, respectively.

Example 5.2.6. The substitution $\sigma = [Y \mapsto \mu Y.\,(\Box X \wedge \Diamond\Diamond Y)]; [X \mapsto \theta]$ sequentially unfolds the eventuality $\theta = \mu X.\,\mu Y.\,(\Box X \wedge \Diamond\Diamond Y)$, and $(\Diamond Y)\sigma = \Diamond\mu Y.\,(\Box\theta \wedge \Diamond\Diamond Y)$ is a θ-deferral.

The formula $\phi = \nu X.\,(p \vee \Box\psi)$ with $\psi = \mu Y.\,(X \wedge \Diamond Y)$ is irreducible and closed. For an example of a reducible fixpoint literal, consider the least fixpoint literal

$$\begin{aligned}\psi[X \mapsto \phi] &= \mu Y.\,(X \wedge \Diamond Y)[X \mapsto \phi] \\ &= \mu Y.\,(\nu X.\,(p \vee \Box(\mu Y.\,(X \wedge \Diamond Y))) \wedge \Diamond Y.\end{aligned}$$

Notice how $\psi[X \mapsto \phi]$ binds Y twice and is induced by the pair $(\psi, [X \mapsto \phi])$, where $[Y \mapsto \psi]; [X \mapsto \phi]$ sequentially unfolds ϕ.

In the following we will consider all deferrals to be in *decomposed form*, i.e. given a formula ψ that belongs to some closed irreducible fixpoint literal θ, so that $\psi = \alpha\sigma$ for

appropriate α and σ according to Definition 5.2.5, we equivalently represent ψ by the pair (α, σ). This will allow us to directly refer to the base α and the sequence σ of a deferral.

Lemma 5.2.7. *Each formula ψ belongs to at most one closed irreducible fixpoint literal θ, and then $\theta \leq \psi$.*

Proof. For the first part of the claim, we show that if (α, σ) is a θ_1-deferral, (β, κ) is a θ_2-deferral and $\alpha\sigma = \beta\kappa$, then $\theta_1 = \theta_2$. To see this, let $\alpha\sigma = \psi$. We show that $\theta_2 \leq \theta_1$, the other direction is symmetric. We note that by the second part of the claim, proven independently below, $\theta_2 \leq \psi$. If $\theta_2 \leq \alpha$, then $\theta_2 < \theta_1$ and hence $\theta_2 \leq \theta_1$, as required. If $\theta_2 \not\leq \alpha$, then let $\theta_2 = \mu Y.\, \phi$ and $\sigma = [X_1 \mapsto \chi_1]; \ldots; [X_n \mapsto \chi_n]$ where $\chi_n = \theta_1$. Since $\theta_2 \leq \psi = \alpha\sigma$ but $\theta_2 \not\leq \alpha$, we are in one of the following two cases: a) There is a variable $X \in \mathsf{FV}(\alpha)$ with $\theta_2 \leq X\sigma$ in which case – since θ_2 is irreducible – $\theta_2 \leq \chi_i \leq \theta_1$ for some $1 \leq i \leq n$: otherwise there is some $\chi_j = \mu Y.\phi_1$ such that $\mu Y.\phi_1([X_{j+1} \mapsto \chi_{j+1}]; \ldots; [X_n \mapsto \chi_n]) = \theta_2$, which is a contradiction to θ_2 being irreducible; b) The formula α contains a fixpoint literal $\mu Y.\, \phi_1$ with $\phi_1\sigma = \phi$. But then $\theta_2 = (\mu Y.\, \phi_1)\sigma$ and $(\mu Y.\, \phi_1, \sigma)$ is a sequence over χ_n which is a contradiction to θ_2 being irreducible.

The proof of the second part of the claim is by lexicographic induction over $(|\sigma|, \alpha)$, distinguishing cases for α. The interesting case is the fixpoint variable case, i.e. $\alpha = Y$ for some Y. If $|\sigma| = 1$, then we have that $\sigma = [Y \mapsto \theta]$ and hence $Y\sigma = \theta$. If $|\sigma| > 1$, then we have $Y\sigma = \chi\kappa$ where χ is the result of applying the first substitution from σ that touches Y to Y and where κ consists of the remaining substitutions from σ. We have $|\kappa| < |\sigma|$ and (χ, κ) is a θ-deferral so that the induction hypothesis finishes the proof. □

The requirement that deferrals are built over irreducible closed fixpoint literals implies that each deferral ψ belongs to at most one irreducible fixpoint literal θ.

Definition 5.2.8 (Fischer-Ladner closure). Given a formula ψ, the *Fischer-Ladner closure* [29] $\mathsf{FL}(\psi)$ of ψ is defined to be the least set of formulas that contains ψ and adheres to the following closure conditions:

$$\begin{aligned}
\phi_1 \wedge \phi_2 \in \mathsf{FL}(\psi) &\text{ implies } \phi_1, \phi_2 \in \mathsf{FL}(\psi),\\
\phi_1 \vee \phi_2 \in \mathsf{FL}(\psi) &\text{ implies } \phi_1, \phi_2 \in \mathsf{FL}(\psi),\\
\heartsuit\phi \in \mathsf{FL}(\psi) &\text{ implies } \phi \in \mathsf{FL}(\psi),\\
\eta X.\, \phi \in \mathsf{FL}(\psi) &\text{ implies } \phi[X \mapsto \eta X.\, \phi] \in \mathsf{FL}(\psi).
\end{aligned}$$

We observe that for all formulas ψ, $|\mathsf{FL}(\psi)| \leq |\psi|$.

Definition 5.2.9 (Alternation level, alternation depth). The *alternation level* $\mathsf{al}(\phi\sigma) := \mathsf{al}(\sigma)$ of a deferral (ϕ, σ) with ϕ not a fixpoint literal is defined inductively over

$|\sigma|$, where $\mathsf{al}(\epsilon) = \mathsf{al}(\epsilon)_\mu = \mathsf{al}(\epsilon)_\nu = 0$,

$$\mathsf{al}(\sigma; [X \mapsto \eta X.\, \psi]) = \begin{cases} \mathsf{al}(\sigma)_\mu + 1 & \text{if } \eta = \mu \\ \mathsf{al}(\sigma)_\nu & \text{otherwise} \end{cases}$$

and

$$\mathsf{al}(\sigma; [X \mapsto \eta X.\, \psi])_\mu = \begin{cases} \mathsf{al}(\sigma)_\mu & \text{if } \eta = \mu \\ \mathsf{al}(\sigma)_\nu + 1 & \text{otherwise} \end{cases}$$

$$\mathsf{al}(\sigma; [X \mapsto \eta X.\, \psi])_\nu = \begin{cases} \mathsf{al}(\sigma)_\nu & \text{if } \eta = \nu \\ \mathsf{al}(\sigma)_\mu + 1 & \text{otherwise} \end{cases}$$

Notice that this definition assigns greater numbers to deferrals that occur at higher nesting depth in the formula to which they belong, i.e. with more alternation inside their sequence σ. For a deferral (ϕ, σ) with $\phi = \eta X.\, \psi$, we put $\mathsf{al}(\phi\sigma) = \mathsf{al}(X([X \mapsto \phi]; \sigma))$. Given a formula ϕ, let $q(\phi) = \max\{\mathsf{al}(\delta) \mid \delta \in \mathsf{FL}(\phi) \text{ and } \delta \text{ is a deferral}\}$. The *alternation depth* $\mathsf{ad}(\phi)$ of ϕ is defined as $q(\phi)$ if $q(\phi)$ is even and as $q(\phi) + 1$ if $q(\phi)$ is odd.

An alternative definition of alternation-depth from [36] counts *all* occurrences of alternation of least and greatest fixpoint literals. The function ad defined above however only takes dependent nestings of least and greastest fixpoint literals into account, where we say that two nested fixpoint literals are dependent if the inner fixpoint literal mentions the fixpoint variable that is bound by the outer fixpoint literal as a free variable. Thus we have that for all (relational) formulas ϕ, $\mathsf{ad}(\phi) \leq z$ where z is the value that the definition of alternation depth in [36] assigns to ϕ.

We now fix a closed, irreducible input formula ψ_0, let $n = |\psi_0|$ and denote the Fischer-Ladner closure of ψ_0 by $\mathsf{FL} := \mathsf{FL}(\psi_0)$; then $|\mathsf{FL}| \leq n$. Let $k = \mathsf{ad}(\psi_0)$ be the alternation-depth of ψ_0. Let $\mathbf{N} = \mathcal{P}(\mathsf{FL})$ be the set of all *nodes* and $\mathbf{S} \subseteq \mathbf{N}$ the set of all *state nodes*, i.e. nodes that contain only $\top$ and modal literals; so $|\mathbf{S}| \leq |\mathbf{N}| \leq 2^n$. We assume a fixed set of *focused nodes* $\mathbf{C}$ with labelling function $l : \mathbf{C} \to \mathbf{N}$. The upcoming global caching algorithm is parametrized by $\mathbf{C}$ and l, and these data have to be supplied to obtain a concrete instance of the algorithm. For a set $G \subseteq \mathbf{N}$, we define the restriction of $\mathbf{C}$ to nodes with labels from G:

$$\mathbf{C}_G = \{v \in \mathbf{C} \mid l(v) \in G\}$$

The intuition behind this definition is that $\mathbf{C}_G$ contains all focused nodes that correspond to some node in a partially expanded tableau with set of nodes G.

Global caching algorithms incrementally build sets of nodes but perform fixpoint computations on $\mathcal{P}(\mathbf{C})$, essentially computing winning regions of the corresponding satisfiability games on-the-fly.

The upcoming global caching algorithm will use the following standard *non-modal tableau rules* in the expansion step. We adapt the notion of tableau rules from Definition 3.1.11 to accommodate the unfolding of fixpoints. To this end, we fix a set B of rule variables. Each rule consists of one *premise* (i.e. a formula built over variables from B using only the operators $\vee$, $\wedge$ and $\eta X.$) and a set of *conclusions* (i.e. a set of sets of variables from B and constructs of the shape $a[X \mapsto \eta X.\, a]$ for $a \in B$) and the rules will be interpreted AND-OR style, i.e. to show satisfiability of a set of formulas Γ, it will be necessary to show that *every* rule application that matches Γ has *some* conclusion that is satisfiable.

$$(\bot)\frac{\bot}{-} \qquad (\wedge)\frac{a \wedge b}{a, b} \qquad (\vee)\frac{a \vee b}{a \qquad b} \qquad (\vee)\frac{\eta X.\, a}{a[X \mapsto \eta X.\, a]}$$

We refer to the set of the non-modal rules by $\mathcal{R}_p$ and additionally assume a one-step sound and one-step complete set of clean modal tableau rules $\mathcal{R}_m$ (recall Definition 3.1.12) and put $\mathcal{R} = \mathcal{R}_p \cup \mathcal{R}_m$. We usually write rules with premise Γ and conclusion $\Sigma = \Gamma_1, \ldots, \Gamma_n$ in sequential form, i.e. as (Γ / Σ).

Definition 5.2.10 (Matching rule applications, conclusions). Let ψ be a formula. Given a set of formulas $\Gamma \subseteq \mathsf{FL}(\psi)$, let $\mathcal{R}(\Gamma) = ((R_1, \sigma_1), \ldots, (R_n, \sigma_n))$ denote the *list* of all applications of rules from $\mathcal{R}$ that *match* Γ, i.e. all pairs (R_i, σ_i) with $R_i = (\Gamma_0 / \Gamma_1, \ldots, \Gamma_l) \in \mathcal{R}$ and $\Gamma_0 \sigma_i \subseteq \Gamma$, where σ_i maps rule variables to formulas from $\mathsf{FL}(\psi)$. Here, $(a[X \mapsto \eta X.\, a])\sigma = \sigma(a)[X \mapsto \eta X.\, \sigma(a)]$ for rule variables $a \in B$. For a set $\mathcal{S}$ of tableau rules, the set of *conclusions* of Γ under $\mathcal{S}$ is

$$\mathsf{Cn}(\mathcal{S}, \Gamma) = \{\{\Gamma_1\sigma, \ldots, \Gamma_n\sigma\} \in \mathcal{P}(\mathcal{P}(\mathsf{FL}(\psi))) \mid (\Gamma_0 / \Gamma_1 \ldots \Gamma_n) \in \mathcal{S}, \Gamma_0\sigma \subseteq \Gamma\}.$$

We define $\mathsf{Cn}(\Gamma)$ as $\mathsf{Cn}(\mathcal{R}_m, \Gamma)$ if Γ is a state node and as $\mathsf{Cn}(\mathcal{R}_p, \Gamma)$ otherwise. A set $N \subseteq \mathbf{N}$ of nodes is *fully expanded* if for each $\Gamma \in N$, $\bigcup \mathsf{Cn}(\Gamma) \subseteq N$.

Definition 5.2.11 (Formula tracking). We say that a formula ψ is *principal* in the application (R, σ) of a rule $R = (\Gamma_0 / \Gamma_1, \ldots, \Gamma_l) \in \mathcal{R}$ if $\psi \in \Gamma_0\sigma$. To track single formulas through rule applications, we define a function $\mathsf{Tr} : \mathsf{FL} \times \mathsf{code}(\psi_0) \to \mathcal{P}(\mathsf{FL})$; here, $\mathsf{code}(\psi_0)$ is assumed to contain encodings of all tuples (R, σ, i) where (R, σ) is a rule application and $i \leq l$ identifies the i-th conclusion of (R, σ), i.e. $\Gamma_i\sigma$. For a given rule application (R, σ), we also define the set

$$\mathsf{code}(R, \sigma) = \{\mathsf{code}(R, \sigma, j) \in \mathsf{code}(\psi_0)\}.$$

Given a tuple (R, σ, i) with $R = (\Gamma_0 / \Gamma_1, \ldots, \Gamma_l)$, $i \leq l$, we define $\mathsf{Tr}(\psi, \mathsf{code}(R, \sigma, i)) = \emptyset$ if $R \in \mathcal{R}_m$ and ψ is not principal in (R, σ); otherwise, if $R \in \mathcal{R}_p$ and ψ is not principal

in (R,σ), then we put $\mathsf{Tr}(\psi,\mathsf{code}(R,\sigma,i)) = \{\psi\}$. If ψ is principal in (R,σ), then we put

$$\mathsf{Tr}(\psi,\mathsf{code}(R,\sigma,i)) = \begin{cases} \emptyset & \text{if } \psi = \bot \text{ and } R = (\bot) \\ \{\psi_1,\psi_2\} & \text{if } \psi = \psi_1 \wedge \psi_2 \text{ and } R = (\wedge) \\ \{\psi_i\} & \text{if } \psi = \psi_1 \vee \psi_2 \text{ and } R = (\vee) \\ \{\psi_1[X \mapsto \psi]\} & \text{if } \psi = \eta X.\,\psi_1 \text{ and } R = (\eta) \\ \{\psi_1\} & \text{if } \psi = \heartsuit\psi_1,\ R \in \mathcal{R}_m \text{ and} \\ & \exists b \in \Gamma_i.\heartsuit b \in \Gamma_0, \sigma(b) = \psi_1 \end{cases}$$

As there are up to the equivalence of conclusions only finitely many rule applications to subsets of the Fischer-Ladner closure and since each rule application has finitely many conclusions, we can always choose $\mathsf{code}(\psi_0)$ in the above definition to be a finite set. In the relational case, we have just six rules (the five non-modal rules plus one modal rule), each rule has at most two conclusions and rule applications can be identified by designating a single formula that is principal to the rule application; hence we can put $\mathsf{code}(\psi_0) = \{1,\dots,5\} \times \mathsf{FL} \times \{1,2\}$ so that we have $|\mathsf{code}(\psi_0)| \leq 10n \in \mathcal{O}(n)$. In the coalgebraic case, we typically assume ExpTime-*tractability* of $\mathcal{R}_m$ (recall Definition 3.2.3), which guarantees that we can choose a set $\mathsf{code}(\psi_0)$ with $|\mathsf{code}(\psi_0)| \leq 2^{p(n)}$ where p is a polynomial function; this is a crucial property for obtaining an ExpTime satisfiability global caching algorithm in the following.

Definition 5.2.12 (Tracking function). We parametrize our algorithm with a number $k \in \mathbb{N}$ and a prioritized function $\mathsf{track} : \mathbf{C} \times \mathsf{code}(\psi_0) \times \{0,\dots,k-1\} \to \mathcal{P}(\mathbf{C})$, that tracks focused nodes through rule applications so that for all focused nodes $v \in \mathbf{C}$ and all codes $\mathsf{code}(R,\sigma,j) \in \mathsf{code}(\psi_0)$ with $(R,\sigma) \in \mathcal{R}(l(v))$, $\mathsf{track}(v,\mathsf{code}(R,\sigma,j),i)$ denotes the set of focused nodes that can result when tracking the focused node v through the rule application (R,σ) that matches $l(v)$ and by choosing the j-th conclusion of that application, with the requirement that the respective transition has priority i.

We next introduce the functions underlying the fixpoint computations for propagation of satisfiability and unsatisfiability.

We assume that a number k of priorities is given, where for $0 \leq i < k$, transitions from v to some $w \in \mathsf{track}(v,\mathsf{code}(R,\sigma,j),i)$ are said to *have* priority i.

Definition 5.2.13 (Proof transitionals). For a set $C \subseteq \mathbf{C}$ of focused nodes, we define the functions $f : (\mathcal{P}(C))^k \to \mathcal{P}(C)$ and $g : (\mathcal{P}(C))^k \to \mathcal{P}(C)$ by

$$f(\mathbf{X}) = \{v \in C \mid \forall (R,\sigma) \in \mathcal{R}(l(v)).\,\exists c \in \mathsf{code}(R,\sigma).\forall 0 \leq i < k.\,\mathsf{track}(v,c,i) \subseteq X_i\}$$
$$g(\mathbf{X}) = \{v \in C \mid \exists (R,\sigma) \in \mathcal{R}(l(v)).\,\forall c \in \mathsf{code}(R,\sigma).\exists 0 \leq i < k.\,\mathsf{track}(v,c,i) \cap X_i \neq \emptyset\}$$

for $\mathbf{X} = (X_0,\dots,X_{k-1}) \in (\mathcal{P}(C))^k$. We refer to C as the *base set* of f and g.

That is, a focused node $v \in C$ is contained in $f(X_0, \ldots, X_{k-1})$ if there is for each rule matching $l(v)$ a conclusion indexed by j such that for all priorities $0 \leq i < k$, $\mathsf{track}(v, \mathsf{code}(R, \sigma, j), i) \subseteq X_i$, where $\mathsf{track}(v, \mathsf{code}(R, \sigma, j), i)$ denotes the set of all focused nodes to which v can evolve with priority i by the respective rule application and by choosing the j-th conclusion. Typically, there is just one i such that $\mathsf{track}(v, \mathsf{code}(R, \sigma, j), i) \neq \emptyset$ and for this i, $\mathsf{track}(v, \mathsf{code}(R, \sigma, j), i)$ consists of just a single focused node; an exception to this is the situation that a nondeterministic refocusing step, governed by Abélard, takes place (see Section 5.2.4 below).

Lemma 5.2.14. *The proof transitionals are monotone w.r.t. set inclusion, i.e. if $X_i \subseteq Y_i$ for all $0 \leq i \leq k-1$, then $f(\mathbf{X}) \subseteq f(\mathbf{Y})$ and $g(\mathbf{X}) \subseteq g(\mathbf{Y})$, where $\mathbf{X} = (X_0, \ldots, X_{k-1})$ and $\mathbf{Y} = (Y_0, \ldots, Y_{k-1})$.*

Proof. We have

$$\begin{aligned} f(\mathbf{X}) =& \{v \in C \mid \forall (R,\sigma) \in \mathcal{R}(l(v)).\, \exists c \in \mathsf{code}(R,\sigma). \forall 0 \leq i < k.\, \mathsf{track}(v,c,i) \subseteq X_i\} \\ \subseteq& \{v \in C \mid \forall (R,\sigma) \in \mathcal{R}(l(v)).\, \exists c \in \mathsf{code}(R,\sigma). \forall 0 \leq i < k.\, \mathsf{track}(v,c,i) \subseteq Y_i\} \\ =& f(\mathbf{Y}) \end{aligned}$$

where the inclusion holds since for all $0 \leq i < k$, we have $X_i \subseteq Y_i$ by assumption. □

Definition 5.2.15 (Propagation). For $G \subseteq \mathbf{N}$, we define $E_G, A_G \subseteq \mathbf{C}_G$ as

$$\begin{aligned} E_G &= \eta_{k-1} X_{k-1} \ldots \eta_1 X_1 . \eta_0 X_0 .\, f(X_0, \ldots, X_{k-1}) \quad \text{and} \\ A_G &= \overline{\eta_{k-1}} X_{k-1} \ldots \overline{\eta_1} X_1 . \overline{\eta_0} X_0 .\, g(X_0, \ldots, X_{k-1}), \end{aligned}$$

where $\eta_i = \mathsf{LFP}$ if i is odd and $\eta_i = \mathsf{GFP}$ otherwise, where $\overline{\eta_i} = \mathsf{GFP}$ if i is odd and $\overline{\eta_i} = \mathsf{LFP}$ otherwise, and where $\mathbf{C}_G$ is the base set of f and g.

Notice that in terms of the satisfiability games described in the overview in the beginning of Section 5.2 above, the computation of E_G and A_G corresponds exactly to solving an incomplete parity game in which priorities $0 \leq i < k$ are assigned to *transitions* and that has $\mathbf{C}_G$ as set of nodes (recall Definition 4.3.6). Player Abélard chooses matching rule applications, player Éloïse chooses conclusions to these rule applications, and in satisfiability games via focusing (see Section 5.2.4 below), Abélard additionally chooses a new focus whenever a refocusing step takes place. The set E_G contains nodes v for which player Éloïse has a strategy to enforce – for each infinite play starting at v – the parity condition that the highest priority that is visited infinitely often is even; similarly A_G is the winning region of player Abélard in the corresponding game, i.e. contains the nodes for which player Abélard has a strategy to enforce an infinite play in which the highest priority that is passed infinitely often is odd; finite plays (i.e. plays that get stuck in some node) are lost by the player who cannot move.

Lemma 5.2.16. *If $G' \subseteq G$, then $E_{G'} \subseteq E_G$ and $A_{G'} \subseteq A_G$.*

Proof. Let $G' \subseteq G$. We show $E_{G'} \subseteq E_G$, the proof of $A_{G'} \subseteq A_G$ is analogous. We denote by f_C the proof transitional with base set $C \subseteq G$ and have that for all $X_0, \dots, X_{k-1} \subseteq G'$,

$$f_{G'}(X_0, \dots, X_{k-1}) \subseteq f_G(X_0, \dots, X_{k-1}).$$

From this we obtain by induction over Kleene numbers and monotonicity of f that for all $X_1, \dots, X_{k-1} \subseteq G'$,

$$\mathsf{GFP}X_0.\, f_{G'}(X_0, \dots, X_{k-1}) \subseteq \mathsf{GFP}X_0.\, f_G(X_0, \dots, X_{k-1}),$$

which in turn implies by induction over Kleene numbers and monotonicity of f that for all $X_2, \dots, X_{k-1} \subseteq G'$,

$$\mathsf{LFP}X_1.\, \mathsf{GFP}X_0.\, \hat{f}_{G'}(X_0, \dots, X_{k-1}) \subseteq \mathsf{LFP}X_1.\, \mathsf{GFP}X_0.\, \hat{f}_G(X_0, \dots, X_{k-1}).$$

We repeat this argumentation to eventually obtain $E_{G'} \subseteq E_G$. □

Lemma 5.2.17. *Let $G \subseteq \mathbf{N}$ be fully expanded and let $C \subseteq \mathbf{C}_G$ be the base set of f and g. For all sets $X_1, \dots, X_{k-1} \subseteq C$,*

$$f(X_0, \dots, X_{k-1}) = \overline{g(\overline{X_0}, \dots, \overline{X_{k-1}})},$$

where for each $Z \subseteq C$, $\overline{Z}$ denotes the complement of Z in C.

Proof. The inclusion "$\subseteq$" is immediate. For the inclusion "$\supseteq$", let $v \in \overline{g(\overline{X}_0, \dots, \overline{X}_{k-1})}$ so that it is not the case that there is $(R, \sigma) \in \mathcal{R}(l(v))$ such that for each $c \in \mathsf{code}(R, \sigma)$ there is $0 \leq i < k$ such that $\mathsf{track}(v, c, i) \cap \overline{X_i} \neq \emptyset$, where, crucially, $\overline{X_i}$ is the complement of X_i in C, and not in $\mathbf{C}$. However, since G is fully expanded, we have $\mathsf{track}(v, c, i) \subseteq C$, which implies that for all $(R, \sigma) \in \mathcal{R}(l(v))$ there is $c \in \mathsf{code}(R, \sigma)$ such that for all $0 \leq i < k$ we have $\mathsf{track}(v, c, i) \subseteq X_i$, i.e. that $v \in f(X_0, \dots, X_{k-1})$. □

Lemma 5.2.18. *Let $G \subseteq \mathbf{N}$ be fully expanded. Then $E_G = \overline{A_G}$.*

Proof. By Lemma 5.2.17, for all $X_0, \dots, X_{k-1} \subseteq \mathbf{C}_G$, $f(X_0, \dots, X_{k-1}) = \overline{g(\overline{X_0}, \dots, \overline{X_{k-1}})}$ so that for fixed $X_1, \dots, X_{k-1} \subseteq \mathbf{C}_G$, $(X_0 \mapsto f(X_0, X_1 \dots, X_{k-1}))$ and $(X_0 \mapsto g(X_0, \overline{X_1} \dots, \overline{X_{k-1}}))$ are complementary (and monotone) functions. We also know that for complementary monotone functions f' and g', $\mathsf{LFP}f' = \overline{\mathsf{GFP}g'}$. Thus we obtain that for all $X_1, \dots, X_{k-1} \subseteq \mathbf{C}_G$,

$$\mathsf{LFP}X_0.\, f(X_0, X_1, \dots, X_{k-1}) = \overline{\mathsf{GFP}X_0.\, g(X_0, \overline{X_1}, \dots, \overline{X_{k-1}})},$$

showing that for all $X_2, \dots, X_{k-1} \subseteq \mathbf{C}_G$, $(X_1 \mapsto \mu X_0.\, f(X_0, X_1, X_2, \dots, X_{k-1}))$ and $(X_1 \mapsto \nu X_0.\, g(X_0, X_1, \overline{X_2}, \dots, \overline{X_{k-1}}))$ are complementary functions, which implies that

$$\mathsf{GFP}X_1.\mathsf{LFP}X_0.\, f(X_0, X_1, X_2, \dots, X_{k-1}) = \overline{\mathsf{LFP}X_1.\mathsf{GFP}X_0.\, g(X_0, X_1, \overline{X_2}, \dots, \overline{X_{k-1}})};$$

repeat this argumentation $k-2$ further times to obtain $E_G = \overline{A_G}$. □

Now we are ready to present the generic global caching algorithm. To obtain a concrete instance of the algorithm, the following data has to be provided:

1. the set of focused nodes $\mathbf{C}$ for a fixed input formula ψ_0 together with a labelling function l that returns the labels $l(v) \subseteq \mathsf{FL}(\psi_0)$ of focused nodes $v \in \mathbf{C}$;
2. a number k of priorities;
3. a tracking function $\mathsf{track} : \mathbf{C} \times \mathsf{code}(\psi_0) \times \{0, \ldots, k-1\} \to \mathcal{P}(\mathbf{C})$ that tracks focused nodes through prioritized rule applications;
4. an initial focused node v_0 that stands for the input formula ψ_0.

The algorithm constructs a partial pre-tableau (see Definition 5.2.22 below), maintaining sets $G, U \subseteq \mathbf{N}$ of *expanded* and *unexpanded* nodes, respectively. It computes $E_G, A_G \subseteq \mathbf{C}_G$ in the optional propagation steps; as these sets grow monotonically, they can be computed incrementally.

Algorithm 5.2.19 (Global caching). Decide satisfiability of a closed formula ψ_0.

1. (Initialization) Put $G := \emptyset$, $\Gamma_0 := \{\psi_0\}$, $U := \{\Gamma_0\}$.
2. (Expansion) Pick $u \in U$ and put $G := G \cup \{u\}$, $U := (U \setminus \{u\}) \cup (\bigcup \mathsf{Cn}(u) \setminus G)$.
3. (Intermediate propagation) Optional: Compute E_G and/or A_G. If $v_0 \in E_G$, then return 'satisfiable'. If $v_0 \in A_G$, then return 'unsatisfiable'.
4. If $U \neq \emptyset$, continue with Step 2.
5. (Final propagation) Compute E_G. If $v_0 \in E_G$, then return 'satisfiable', else 'unsatisfiable'.

Note that in Step 5, G is fully expanded. For purposes of the soundness proof, we note an immediate consequence of Lemmas 5.2.16 and 5.2.18:

Lemma 5.2.20. *If some run of the algorithm without intermediate propagation steps is successful on input ψ_0, then all runs on input ψ_0 are successful.*

Proof. Let G denote the set of nodes that is created by the algorithm without intermediate propagation – i.e. without step 3) – and notice that G is fully expanded. Let $v_0 \in E_G$ and let G_p be the set of nodes created by any run of the algorithm (possibly involving intermediate propagation). We note that $G_p \subseteq G$ so that Lemma 5.2.16 tells us that $A_{G_p} \subseteq A_G$. As G is fully expanded, Lemma 5.2.18 states that $A_G = \overline{E_G}$. As $v_0 \in E_G$, $v_0 \notin A_{G_p} \subseteq A_G = \overline{E_G}$, as required. □

Definitions of various instances of this algorithm to concrete fragments of the coalgebraic μ-calculus can be found in the ensuing sections, together with the generic soundness and completeness proofs.

5.2.2 Timed-out Tableaux

We continue to introduce several technical concepts and lemmas that will be used to show the soundness and completeness of the various instances of the introduced algorithm.

Definition 5.2.21 (Tracking automata). The *tracking automaton* for ψ_0 is the *minimal priority* PA (recall Definition 4.3.5) $\mathsf{N}(\psi_0) = (V, \Sigma, \Delta, \psi_0, \alpha)$ where $V = \mathsf{FL}$, $\Sigma = \mathsf{code}(\psi_0)$; the priority function is defined – for $t = (\psi, a, \psi') \in \Delta$ – by $\alpha(\psi, a, \psi') = \mathsf{al}(\psi') + 1$ if $\psi = \eta X.\, \psi_1$ and $\psi' = \psi_1[X \mapsto \eta X.\, \psi_1]$ and otherwise by $\alpha(\psi) = \mathsf{ad}(\psi) + 1$ if $\min(\mathsf{al}(\psi), \mathsf{al}(\psi')) > 0$ and by $\alpha(\psi) = 1$ if $\min(\mathsf{al}(\psi), \mathsf{al}(\psi')) = 0$; the transition relation is defined by $\Delta = \mathsf{Tr}$. Letters $\mathsf{code}(R, \sigma, j) \in \Sigma$ identify applications (R, σ) of tableau rules R and the j-th conclusions of (R, σ).

Given a formula $\psi \in V$, a letter $\mathsf{code}(R, \sigma, j) \in \mathsf{code}(\psi_0)$ and some priority i, $\Delta_i(\psi, \mathsf{code}(R, \sigma, j))$ is the set of formulas to which v can be tracked through the rule application (R, σ) and the choice of the j-th conclusion such that the according transformation of the formula has priority i. Infinite words over Σ encode infinite sequences of rule applications and the automaton $\mathsf{N}(\psi_0)$ accepts those words over Σ that encode a sequence of rule applications for which there is a way to track a formula such that the outermost fixpoint that is unfolded infinitely often is a least fixpoint; such sequences are called *bad sequences*.

Definition 5.2.22 (Pre-tableaux). A labelled set C with labelling function $l : C \to \mathcal{P}(\mathsf{FL})$ such that $\bot \notin l(x)$ for all $x \in C$ is a *pre-tableau set* (for ψ_0) if for each $x \in C$ and all $(R, \sigma) \in \mathcal{R}(l(x))$ with $R = (\Gamma_0/\Gamma_1, \ldots, \Gamma_l)$, there are $1 \leq j \leq l$ and $y \in C$ such that $\Gamma_j \sigma \subseteq l(y)$. Given a function $L : C \to \mathsf{List}(C)$, the pair (C, L) is a *pre-tableau* (for ψ_0) if for each $x \in C$ and all $(R, \sigma) \in \mathcal{R}(l(x))$, there are $1 \leq j \leq n$ and $y \in L(x)$ such that $\Gamma_j \sigma \subseteq l(y)$; we point out that $L(x) \in \mathsf{List}(C)$ is a *list* of elements of C. An Éloïse*-determined pre-tableau* (for ψ_0) is a pre-tableau (C, L) such that for each $x \in C$ and writing $\mathcal{R}(l(x)) = ((R_1, \sigma_1), \ldots, (R_n, \sigma_n))$, we have $L(x) = (y_1, \ldots, y_n)$ and for each $1 \leq i \leq n$ with $R_i = (\Gamma_0/\Gamma_1, \ldots, \Gamma_l)$, there is $1 \leq j_i \leq l$ such that $\Gamma_{j_i} \subseteq l(y_i)$; furthermore, we require a function $c : C \times \mathbb{N} \to \mathbb{N}$ that records the choice of conclusion for each rule application to a node, i.e. with $c(x, i) = j_i$ where x, i and j_i are as defined above.

In the definition of Éloïse-determined pre-tableaux, it is crucial that for each node $x \in C$, rule applications $(R_i, \sigma_i) \in \mathcal{R}(l(x))$ and successors $y_i \in L(x)$ are matched by their index, i.e. that we can pick the i-th rule application and the node that corresponds to this rule application and also know for each node $y_i \in L(x)$ that it is obtained from x by the rule application (R_i, σ_i) (and choice of conclusion $c(x, i)$).

It will be convenient to use so-called *timed-out tableaux* as a stepping stone between models and winning regions in satisfiability games. Timed-out tableaux are Éloïse-determined pre-tableaux in which no infinite path contains an unsatisfied least fixpoint formula; to make this property formal, we introduce the notion of tableau automata, which

use tracking automata to nondeterministically track formulas through Éloïse-determined pre-tableaux:

Definition 5.2.23 (Tableau automata). Let (C, L) be an Éloïse-determined pre-tableau for ψ_0 and let $w_0 \in C$ be a node with $\psi_0 \in l(w_0)$. Let Δ be the transition function and α the priority function of the nondeterministic tracking automaton $\mathsf{N}(\psi_0)$ for ψ_0 (see Definition 5.2.21). The *tableau automaton* $\mathsf{A}(C, L) = (V', \Sigma, \Delta', v_0, \beta)$ is the *minimal priority* PA that is defined by $V' = \{(w, \psi) \in C \times \mathsf{FL} \mid \psi \in l(w)\}$, $\Sigma = \{*\}$, $v_0 = (w_0, \psi_0)$ and

$$\Delta'((w, \psi), *) = \{(w_i, \phi) \mid w_i \in L(w), \phi \in \Delta(\psi, \mathsf{code}(R_i, \sigma_i, j_i))\},$$

where $(w, \psi) \in V'$, $\mathcal{R}(l(w)) = ((R_1, \sigma_1), \ldots, (R_n, \sigma_n))$, $L(w) = (w_1, \ldots, w_n)$, $c(w, i) = j_i$ and $\mathsf{code}(R_i, \sigma_i, j_i)$ encodes the i-th rule application and the choice of the j_i-th conclusion that leads to the node w_i; finally, we put $\beta((w, \psi), *, (w_i, \phi)) = \alpha(\psi, \mathsf{code}(R_i, \sigma_i, j_i), \phi)$ where $\mathsf{code}(R_i, \sigma_i, j_i)$ is the letter that encodes the i-th rule application and the choice of the j_i-th conclusion that leads from w to the node w_i such that $\phi \in \Delta(\psi, \mathsf{code}(R_i, \sigma_i, j))$. By invertibility of non-modal rules, we assume that just a single fixed rule is applied to each unsaturated node.

By construction, a state (w, ψ) in $\mathsf{A}(C, L)$ is empty (i.e. does not accept the only possible word $*^\omega$) if and only if no infinite L-path that starts at w contains an infinite trace of ψ in which a least fixpoint literal is unfolded infinitely often while all superincumbent greatest fixpoint literals are unfolded only finitely often.

Recall that by Definition 4.3.5, a state $(v, \psi) \in V'$ is empty if and only if it satisfies the formula

$$\neg\phi_{\mathsf{PA}_m} = \bigwedge_{1<i<k,\, i \text{ even}} \mathsf{AG}\ \phi_i, \text{ where} \qquad \phi_i = \mu X_1.\nu X_0.\ \Box^i X_1 \wedge \bigwedge_{j>i} \Box^j X_0,$$

which is, after transformation to an equivalent formula with alternation depth k (as described in Definition 4.3.3), by Lemma 2.2.7 the case if and only if there is some time-out vector $\overline{m}$ such that v has nested time-outs $\overline{m}$ (for $\neg\phi_{\mathsf{PA}_m}$).

Definition 5.2.24. Let $\overline{m}$ be a time-out vector, let ψ be a formula and let (C, L) be an Éloïse-determined pre-tableau. Recalling Definition 2.2.6, we define $\mathsf{to}(\psi, \overline{m}) \subseteq C$ to be the set of nodes $v \in C$ with $\psi \in l(v)$ such that (v, ψ) has nested time-outs $\overline{m}$ for $\neg\phi_{\mathsf{PA}_m}$ in the tableau automaton. For all $v \in \mathsf{to}(\psi, \overline{m})$ there is no accepting run in the tableau automaton that starts at (v, ψ).

Definition 5.2.25 (Timed-out tableaux). A *timed-out tableau* for ψ_0 is a finite Éloïse-determined pre-tableau (C, L) for ψ_0 with the additional property that the non-emptiness region of the tableau automaton $\mathsf{A}(C, L)$ is the empty set.

In a timed-out tableau with set of nodes C, we have that for all states $(v, \psi) \in V'$ in the according tableau automaton $\mathsf{A}(C, L)$, no run in $\mathsf{run}(\mathsf{A}(C, L), (v, \psi), *^\omega)$ is accepting. Given a timed-out tableau (C, L) and a pair (v, ψ) with $v \in C$, $\psi \in l(v)$, the timed-out property ensures that on any L-path that starts at v, in any possible trace of ψ along this path, the lowest alternation-level at which the traced formula is unfolded infinitely often is even (i.e. that the outermost fixpoint that is unfolded infinitely often is a greatest fixpoint). Thus a timed-out tableau is an Éloïse-determined pre-tableau that does not contain bad branches (see Section 5.2.3 below for more details).

As we will now show, a formula is satisfiable if and only if a timed-out tableau exists for it. To show the forward direction of this equivalence, we first introduce several concepts that will allow us to construct timed-out tableaux from models.

Definition 5.2.26. For a formula ψ and an interpretation σ, we define $\psi^X_\sigma(\phi) = \psi(\sigma[X \mapsto \phi])$, $(\psi^X_\sigma)^0(\phi) = \phi$ and $(\psi^X_\sigma)^{n+1}(\phi) = \psi^X_\sigma((\psi^X_\sigma)^n(\phi))$. We say that a coalgebra $\mathcal{C}$ is *stabilizing* if for each state x in $\mathcal{C}$, for each formula $\mu X.\, \psi$, each interpretation σ that is defined on variables from $\mathsf{FV}(\mu X.\, \psi)$, all formulas ϕ and all $Y \in \mathsf{FV}(\phi)$, if $\mathcal{C}, x \models \phi[Y \mapsto \mu X.\, \psi\sigma]$, then there is $0 \leq n \in \mathbb{N}$ such that $\mathcal{C}, x \models \phi[Y \mapsto (\psi^X_\sigma)^n(\bot)]$.

We note that fixpoints over finite sets stabilize after finitely many approximation steps so that finite coalgebras are stabilizing. Importing the finite model property (without requiring a bound on model size) for the coalgebraic μ-calculus from [7], we can thus w.l.o.g. restrict our attention to stabilizing coalgebras.

Definition 5.2.27 (Unfolding time-outs). Let $\mathcal{C} = (C, \xi)$ be a coalgebra, let (ψ, σ) be a deferral with $\sigma = [X_j \mapsto \chi_j]; \ldots; [X_0 \mapsto \chi_0]$, where $\chi_i = \eta_i X_i.\, \psi_i$. Also let $\overline{m} = (m_0, \ldots, m_j)$ be a vector of length $|\sigma|$ with $m_i \leq n$ for all $0 \leq i \leq j$ and $m_i = n$ for all $0 \leq i \leq j$ with $\eta_i = \nu$. Finally let $x \in C$ be a state with $x \in [\![\psi\sigma]\!]$. Then we say that (ψ, σ) *has unfolding time-outs* $\overline{m}$ at x if $x \in [\![\psi]\!]_{i_{\sigma(\overline{m})}}$ where $i_{\sigma(\overline{m})}$ is defined inductively by putting $i_{\epsilon(\overline{m})} = \epsilon$ in the base case (where ϵ denotes the empty substitution), and by putting

$$i_{\sigma(\overline{m})} = i_{\kappa(\overline{o})}[X_j \mapsto ([\![\psi_j]\!]^{X_j}_{i_{\kappa(\overline{o})}})^{m_j}(\emptyset)],$$

if $\eta_j = \mu$, and

$$i_{\sigma(\overline{m})} = i_{\kappa(\overline{o})}[X_j \mapsto [\![\nu X_j.\, \psi_j]\!]_{i_{\kappa(\overline{o})}}],$$

if $\eta_j = \nu$; here $\overline{o} = (m_0, \ldots, m_{j-1})$ and $\kappa = [X_{j-1} \mapsto \chi_{j-1}]; \ldots; [X_0 \mapsto \chi_0]$. Thus the first component of time-out vectors is the most significant component and corresponds to the outermost fixpoint χ_0; this is in contrast to sequences of deferrals in which the last substitution corresponds to the outermost fixpoint.

Lemma 5.2.28. *Let $\mathcal{C} = (C, \xi)$ be a stabilizing coalgebra. For all states $x \in C$ and all deferrals (ψ, σ) with $|\sigma| = j$ and $x \models \psi\sigma$, there is a least (by lexicographic ordering) time-out vector $(m_0, \ldots, m_j)$ such that (ψ, σ) has unfolding time-outs $(m_0, \ldots, m_j)$ at x.*

Proof. By stabilization, (ψ, σ) has unfolding time-outs $(|C|, \ldots, |C|)$ at x. By the well-ordering $\leq_l$ of time-out vectors, there is a least time-out vector $(m_0, \ldots, m_j)$ such that (ψ, σ) has unfolding time-outs $(m_0, \ldots, m_j)$ at x □

If ψ is not a fixpoint literal, then we denote the least (by left-to-right lexicographic ordering) vector $\overline{m}$ such that (ψ, σ) has unfolding time-outs $\overline{m}$ at x by $\mathsf{uto}((\psi, \sigma), x)$ (uto for *unfolding time-out*). If ψ is a fixpoint literal $\eta X.\, \phi$, then we put $\mathsf{uto}((\psi, \sigma), x) = \mathsf{uto}((X, [X \mapsto \phi]; \sigma), x)$.

Fact 5.2.29. Let $x \models (\mu X.\, \psi)\sigma$. Then

$$\mathsf{uto}((\mu X.\psi, \sigma), x) > \mathsf{uto}((\psi, [X \mapsto \mu X.\, \psi]; \sigma), x),$$

i.e. the unfolding of least fixpoints reduces unfolding time-outs.

Proof. Assume $\mathsf{uto}((\mu X.\psi, \sigma), x) = \mathsf{uto}((X, [X \mapsto \mu X.\psi]; \sigma), x) = (m_0, \ldots, m_j)$. Then $m_j > 0$, and since

$$\begin{aligned}[\![X]\!]_{i_\sigma(m_0,\ldots,m_j)} &= ([\![\psi]\!]^X_{i_\sigma(m_0,\ldots,m_{j-1})})^{m_j}(\emptyset) \\ &= [\![\psi]\!]^X_{i_\sigma(m_0,\ldots,m_{j-1})}(([\![\psi]\!]^X_{i_\sigma(m_0,\ldots,m_{j-1})})^{m_j-1}(\emptyset)),\end{aligned}$$

we have $\mathsf{uto}((\psi, [X \mapsto \mu X.\, \psi]; \sigma), x) = (m_0, \ldots, m_j - 1) < (m_0, \ldots, m_j)$. □

Lemma 5.2.30. *If ψ_0 is satisfiable, then there is a timed-out tableau for ψ_0.*

Proof. Let ψ_0 be a satisfiable formula, i.e. let there be a finite, stabilizing model $\mathcal{C} = (C, \xi)$ and a state $x \in C$ with $\mathcal{C}, x \models \psi_0$. We define the labelling function $l : C \to \mathcal{P}(\mathsf{FL})$ by putting $l(x) = \{\psi \in \mathsf{FL} \mid \mathcal{C}, x \models \psi\}$ for $x \in C$. Furthermore, we define the tableau structure $L : C \to \mathsf{List}(C)$ and the choice function $c : C \times \mathbb{N} \to \mathbb{N}$. So let $x \in C$ be a state with $\mathcal{R}(l(x)) = ((R_1, \sigma_1), \ldots, (R_n, \sigma_n))$. For each $1 \leq i \leq n$, we pick a state y_i and a number $c(x, i)$ as follows. If the i-th rule application is an application of a non-modal rule, then put $y_i = x$ and if the applied rule is not the disjunction rule, then put $c(x, i) = 1$, otherwise let the principal disjunction $\psi_1 \vee \psi_2$ and let $\overline{m} := \mathsf{uto}(\psi_1 \vee \psi_2, x)$ so that there is a $j \in \{1, 2\}$ such that $\mathsf{uto}(\psi_j, x) = \overline{m}$ and put $c(x, i) = j$. If the i-th rule application is an application of a modal rule $R_i = (\Gamma_0/\Gamma_1, \ldots, \Gamma_l)$, then we have, for each $\heartsuit\psi \in l(x)$, a time-out vector $\overline{m} = \mathsf{uto}(\heartsuit\psi, x)$. We put $B_{\heartsuit\psi} = [\![\psi]\!]_{i_\sigma(\overline{m})}$ and since $\heartsuit\psi$ has time-outs $\overline{m}$ at x, we have $x \in [\![\heartsuit\psi]\!]_{i_\sigma(\overline{m})}$ and hence $\xi(x) \in [\![\heartsuit]\!]B_{\heartsuit\psi}$. By Lemma 5.2.31 below, there are $1 \leq j \leq l$ and $y \in C$ such that $\Gamma_j\sigma \subseteq l(y)$ and for each $\heartsuit\psi \in l(x)$ with $\psi \in \mathsf{Tr}(\heartsuit\psi, \mathsf{code}(R_i, \sigma_i, j))$, we have $y \in B_{\heartsuit\psi} = [\![\psi]\!]_{i_\sigma(\overline{m})}$ and hence $\mathsf{uto}(\psi, y) \leq \overline{m} = \mathsf{uto}(\heartsuit\psi, x)$. Choose $y_i = y$ and $c(x, i) = j$. We define $L(x) = (y_1, \ldots, y_n)$. By construction, the structure (C, L) together with the function c is a finite Éloïse-determined pre-tableau. It remains to show that (C, L) also has the required time-out property, i.e. that the non-emptiness region of the tableau automaton $\mathsf{A}(C, L)$

is the empty set. Recall from Definition 4.3.5 that the *emptiness* region of the *minimal priority* parity automaton $\mathsf{A}(C,L) = (V', \Sigma, \Delta', v_0, \alpha)$ with priorities 1 to $k = \mathsf{ad}(\psi_0) + 1$ is defined by the formula

$$\neg\phi_{\mathsf{PA}_m} = \bigwedge_{1<i<k,\, i \text{ even}} \mathsf{AG}\ \phi_i, \text{ where} \qquad \phi_i = \mu X_1.\nu X_0.\ \Box^i X_1 \wedge \bigwedge_{j>i} \Box^j X_0.$$

Thus it suffices to show $V' = [\![\neg\phi_{\mathsf{PA}_m}]\!]$. As $V' \subseteq [\![\theta]\!]$ implies $V' \subseteq [\![\mathsf{AG}\,\theta]\!]$ for any formula θ, we have that if $V' \subseteq [\![\phi_i]\!]$ for all even $1 < i < k$, then $V' \subseteq [\![\neg\phi_{\mathsf{PA}_m}]\!]$. Hence we let $1 < i < k$ be even and show

$$\begin{aligned} V' \subseteq [\![\phi_i]\!] &= [\![\mu X_1.\nu X_0.\ \Box^i X_1 \wedge \bigwedge_{j>i} \Box^j X_0]\!] \\ &= [\![\nu X_0.\ \Box^i X_1 \wedge \bigwedge_{j>i} \Box^j X_0]\!]^{X_1}([\![\phi_i]\!]) \\ &= [\![\nu X_0.\ \Box^i \phi_i \wedge \bigwedge_{j>i} \Box^j X_0]\!] =: U \end{aligned}$$

To show this inclusion, we let $\mathsf{uto}(\overline{m})$ denote the set of pairs $(x,\psi) \in V'$ such that $\psi = \alpha\sigma$ where (α,σ) is a deferral with $x \in [\![\psi]\!]_{i_{\sigma(\overline{m})}}$, or such that ψ is not a deferral. We claim that for all time-out vectors $\overline{m} = (m_0, \ldots, m_i)$,

$$\mathsf{uto}(\overline{m}) \subseteq U,$$

from which the required inclusion then follows, since for each $(x,\psi) \in V'$ with ψ a deferral, there is by Lemma 5.2.28 some $\overline{m}$ with $\mathsf{uto}(\psi, x) = \overline{m}$ and $(x,\psi) \in \mathsf{uto}(\overline{m})$. To prove the claim, we proceed by lexicographic induction over $\overline{m}$. We show inclusion in the greatest fixpoint by coinduction, i.e. we show that $\mathsf{uto}(\overline{m})$ is a postfixpoint of the function $[\![\Box^i\phi_i \wedge \bigwedge_{j>i} \Box^j X_0]\!]^{X_0}$, i.e. that

$$\mathsf{uto}(\overline{m}) \subseteq [\![\Box^i\phi_i \wedge \bigwedge_{j>i} \Box^j X_0]\!]^{X_0}(\mathsf{uto}(\overline{m})).$$

So let $(x,\psi) \in \mathsf{uto}(\overline{m})$ and $(y,\psi') \in \Delta'_j((x,\psi),*)$ for $j \geq i$.

If $j > i$, then we have to show that $(y,\psi') \in \mathsf{uto}(\overline{m})$. We distinguish upon the shapes of ψ and ψ'. If $\psi = \psi'$, then ψ is not principal in the according rule application and the applied rule is a non-modal rule; then $(x,\psi) = (y,\psi') \in \mathsf{uto}(\overline{m})$. If $\psi = \psi_1 \wedge \psi_2$ and $\psi' = \psi_q$ for $q \in \{1,2\}$, then $(x,\psi_q) \in \mathsf{uto}(\overline{m})$ follows from $(x,\psi_1 \wedge \psi_2) \in \mathsf{uto}(\overline{m})$. If $\psi = \psi_1 \vee \psi_2$ and $\psi' = \psi_q$ for $q \in \{1,2\}$ with $c(x,1) = q$ (i.e. Éloïse chooses the disjunct ψ_q), then $(x,\psi_q) \in \mathsf{uto}(\overline{m})$ since c by construction chooses a disjunct that has time-outs $\overline{m}$ at x. If $\psi = \eta X.\,\psi_1$ and $\psi' = \psi_1[X \mapsto \psi]$, then $\mathsf{uto}(\psi_1[X \mapsto \psi], x) \leq \mathsf{uto}(\eta X.\,\psi_1, x)$ (by Fact 5.2.29, if $\eta = \mu$ and since the unfolding of greatest fixpoints does not change unfolding time-outs at all) and hence $(x,\psi_1[X \mapsto \psi]) \in \mathsf{uto}(\overline{m})$. If $\psi = \heartsuit\psi_1$ and $\psi' = \psi_1$, then $\mathsf{uto}(\psi_1, y) \leq \mathsf{uto}(\heartsuit\psi_1, x)$ by definition of L and c above and hence $(y,\psi_1) \in \mathsf{uto}(\overline{m})$.

If $j = i$, then we have to show that $(y, \psi') \in [\![\phi_i]\!] = U$. Since i is even, ψ is a least fixpoint literal $\mu X.\, \phi\sigma$ with $\mathsf{al}(\mu X.\, \phi\sigma) = i - 1$ and we have $\psi = (\phi[X \mapsto \mu X.\, \phi])\sigma$. Let $p = |\sigma| + 2$. If $\overline{m}_p = 0$, then ψ has time-out 0 at x for the fixpoint $\mu X.\, \phi\sigma$, which yields $\mathcal{C}, x \models (X[X \mapsto \phi^0(\bot)])\sigma$ and hence $\mathcal{C}, x \models \bot$, a contradiction. If $\overline{m}_p > 0$, then by Fact 5.2.29, $\mathsf{uto}(\mu X.\, \phi, x) = \mathsf{uto}(X[X \mapsto \mu X.\, \phi], x) < \mathsf{uto}(\phi[X \mapsto \mu X.\, \phi], x)$, so that $(x, \mu X.\, \phi) \in \mathsf{uto}(\overline{m})$ implies $(x, \phi[X \mapsto \mu X.\phi]) \in \mathsf{uto}(\overline{m}')$ with $\overline{m}' < \overline{m}$. By the induction hypothesis, we have $(x, \phi[X \mapsto \mu X.\phi]) \in U$, as required. □

Lemma 5.2.31. *Let $\mathcal{C} = (C, \xi)$ be a coalgebra with labelling function $l : C \to \mathcal{P}(\mathsf{FL})$ such that for all $x \in C$, $l(x) = \{\psi \in \mathsf{FL} \mid \mathcal{C}, x \models \psi\}$. Also let $x \in C$ be a state. For each $\heartsuit\psi \in l(x)$, let $B_{\heartsuit\psi} \subseteq [\![\psi]\!]$ be a set with $\xi(x) \in [\![\heartsuit]\!]B_{\heartsuit\psi}$. Let $(\Gamma_0\sigma/\Gamma_i\sigma, \ldots, \Gamma_n\sigma)$ be an application of a modal rule R such that $\Gamma_0\sigma \subseteq l(x)$. Then there exist $1 \leq j \leq n$ and a state $y \in C$ with $\Gamma_j\sigma \subseteq l(y)$ such that for all $\heartsuit\psi \in l(x)$, if $\psi \in \mathsf{Tr}(\heartsuit\psi, \mathsf{code}(R, \sigma, j))$, then $y \in B_{\heartsuit\psi}$.*

Proof. Define a renaming κ by $\kappa(b) = a_{\heartsuit\sigma(b)}$ for $\heartsuit b \in \Gamma_0$ (recall that premises of rules are *clean*, i.e. mention every variable at most once). Put

$$\theta = \{\heartsuit a_{\heartsuit\psi} \mid \heartsuit\psi \in l(v)\}.$$

Notice that $x \in [\![\theta]\!]_{TC}\tau \neq \emptyset$ for $\tau(a_{\heartsuit\psi}) = B_{\heartsuit\psi}$. Also, $\Gamma_0\kappa \subseteq \theta$ so that by one-step soundness, we have an i such that $[\![\Gamma_i\kappa]\!]_C\tau \neq \emptyset$. Let $y \in [\![\Gamma_i\kappa]\!]_C\tau$. For each $b \in \Gamma_i$, we have $\heartsuit b \in \Gamma_0$ for some $\heartsuit$ (since every variable that occurs in a rule conclusion also occurs in the premise), and hence $y \in B_{\heartsuit\sigma(b)} \subseteq [\![\sigma(b)]\!]$. Thus $\Gamma_i\sigma \subseteq l(y)$. Now let $\heartsuit\psi \in v$ and $\psi \in \mathsf{Tr}(\heartsuit\psi, \mathsf{code}(R, \sigma, j))$. Then there is a $b \in \Gamma_i$ with $\heartsuit b \in \Gamma_0$ and $\sigma(b) = \psi$. But then $\kappa(b) = a_{\heartsuit\psi}$, $\tau(a_{\heartsuit\psi}) = B_{\heartsuit\psi}$ and hence $y \in B_{\heartsuit\psi}$. □

We will now show the converse direction, i.e. that the existence of a timed-out tableau for a formula ψ_0 implies the existence of a model for ψ_0.

Definition 5.2.32 (Propositional entailment). For a finite set Ψ of formulas, we write $\bigwedge\Psi$ for the conjunction of the elements of Ψ. We say that Ψ *propositionally entails* a formula ϕ (written $\Psi \vdash_{PL} \phi$) if $\bigwedge\Psi \to \phi$ is a propositional tautology, where modal literals are treated as propositional atoms and fixpoint literals $\eta X.\, \phi$ are unfolded to $\phi[X \mapsto \eta X.\, \phi]$ (recall that fixpoint operators are guarded). A finite set of formulas Ψ *propositionally entails* a finite set Φ of formulas (written $\Psi \vdash_{PL} \Phi$) if $\Psi \vdash_{PL} \bigwedge\Phi$.

Definition 5.2.33 (Pseudo-extension). The *pseudo-extension* $\widehat{[\![\phi]\!]}_W$ of a formula ϕ in a set W of labelled nodes with labelling function $l : W \to \mathcal{P}(\mathsf{FL})$ is

$$\widehat{[\![\phi]\!]}_W = \{v \in W \mid l(v) \vdash_{PL} \phi\}.$$

We omit the index if no confusion arises and extend this notion to sets Ψ of formulas by putting $\widehat{[\![\Psi]\!]} = \bigcap_{\psi\in\Psi} \widehat{[\![\psi]\!]}$.

Definition 5.2.34. We denote by $u_f(\phi)$ and $u_p(\phi)$ the numbers of unguarded occurrences of fixpoint and propositional operators in ϕ, respectively; we extend this notation to sets of formulas by putting $u_f(\Gamma) = \sum_{\phi\in\Gamma} u_f(\phi)$ and $u_p(\Gamma) = \sum_{\phi\in\Gamma} u_p(\phi)$.

Definition 5.2.35. Given two sets C, W with $C \subseteq W$, we say that a function $L : W \to \mathsf{List}(W)$ is *loop-free functional in* $\overline{C}$ if for all $u \in \overline{C}$, $|L(u)| = 1$ and some node $c \in C$ is reachable by L from u.

If L is loop-free functional in $\overline{C}$, then there is, for all $v \in C$ and $u \in L(v)$, exactly one $v' \in C$ – which we denote by $\lceil u \rceil$ – such that there is an L-path $u = u_0 L u_1 L \dots L u_m = v'$, where all u_i except u_m are not contained in C; notice that we allow $m = 0$.

Definition 5.2.36 (Strong coherence). Let (W, L) be an Éloïse-determined pre-tableau with labelling function l and let $C \subseteq W$ be a set such that L is loop-free functional in $\overline{C}$. Also, let $\mathcal{C} = (C, \xi)$ be a T-coalgebra and let $v \in C$. For a state $v \in C$ with $\mathcal{R}(l(v)) = ((R_1, \sigma_1), \dots, (R_n, \sigma_n))$ and $L(v) = (w_1, \dots, w_n)$, we define

$$\mathsf{rec}(\phi, \heartsuit\phi, v) = \{\lceil w_i \rceil \in C \mid \phi \in \mathsf{Tr}(\heartsuit\phi, \mathsf{code}(R_i, \sigma_i, c(v, i)))\}.$$

The coalgebra structure ξ is *strongly coherent* at v if for all formulas $\heartsuit\phi \in \mathsf{FL}$,

$$v \in \widehat{[\![\heartsuit\phi]\!]}_C \text{ implies } \xi(v) \in [\![\heartsuit]\!](\widehat{[\![\phi]\!]}_C \cap \mathsf{rec}(\phi, \heartsuit\phi, v)).$$

The whole coalgebra $\mathcal{C}$ is strongly coherent if ξ is strongly coherent at all states $v \in C$; then we have that for all formulas $\heartsuit\phi \in \mathsf{FL}$,

$$\widehat{[\![\heartsuit\phi]\!]}_C \subseteq \xi^{-1}[[\![\heartsuit]\!]\widehat{[\![\phi]\!]}_C],$$

i.e. that for all $v \in C$,

$$v \in \widehat{[\![\heartsuit\phi]\!]}_C \text{ implies } \xi(v) \in [\![\heartsuit]\!](\widehat{[\![\phi]\!]}_C).$$

Given an Éloïse-determined pre-tableau (W, L), a set $C \subseteq W$ that is loop-free functional in $\overline{C}$ and a strongly coherent coalgebra (C, ξ), we have $\mathsf{rec}(\phi, \heartsuit\phi, v) \subseteq L|_C(v)$ for all $v \in C$ and $\heartsuit\phi \in l(v)$, where $L|_C = \{(v, \lceil w \rceil) \mid w \in L(v)\}$.

Lemma 5.2.37 (Coalgebra existence). *Let (W, L) be an Éloïse-determined pre-tableau and let $C \subseteq W$ be a set such that L is loop-free functional in $\overline{C}$. Then there is a strongly coherent coalgebra over C.*

Proof. Let $x \in C$ and put $\theta = \{\heartsuit a_{\heartsuit\alpha} \mid \heartsuit\alpha \in l(x)\}$ and $\tau(a_{\heartsuit\alpha}) = \widehat{[\![\alpha]\!]}_C \cap \mathsf{rec}(\alpha, \heartsuit\alpha, x)$. Then we have $[\![\theta]\!]_{TW}\tau \subseteq [\![\heartsuit]\!](\widehat{[\![\alpha]\!]}_C \cap \mathsf{rec}(\alpha, \heartsuit\alpha, x))$ for each $\heartsuit\alpha \in l(x)$; thus it suffices to show that $[\![\theta]\!]_{TW}\tau \neq \emptyset$. By one-step tableau completeness of $\mathcal{R}_m$, this is the case if there is, for each rule $R = (\Gamma_0/\Gamma_1, \dots, \Gamma_n) \in \mathcal{R}_m$ and substitution σ such that $\Gamma_0\sigma \subseteq \theta$, a $1 \leq j \leq n$ such that $[\![\Gamma_j\sigma]\!]_W\tau \neq \emptyset$. Let $\mathcal{R}(x) = ((R_1, \sigma_1), \dots, (R_n, \sigma_n))$ and $L(x) =$

$(w_1, \dots, w_n)$. Let $(\Gamma_0\sigma/\Gamma_1\sigma, \dots, \Gamma_n\sigma)$ be an application of R with $\Gamma_0\sigma \subseteq \theta$. We define $\kappa(b) = \alpha$ if $\sigma(b) = a_{\heartsuit\alpha}$ and note that (R, κ) induces a rule instance $(\Gamma_0\kappa/\Gamma_1\kappa, \dots, \Gamma_n\kappa)$ with $\Gamma_0\kappa \subseteq l(x)$. Let i be the number with $(R, \kappa) = (R_i, \sigma_i)$. As (W, L) is an Éloïse-determined pre-tableau that is loop-free functional in $\overline{C}$, we have $\lceil w_i \rceil \in \widehat{[\![l(w_i)]\!]_C}$ and $l(w_i) = \Gamma_j\kappa$, where $j = c(x, i)$. Put $y = \lceil w_i \rceil$. We have $[\![\Gamma_j\sigma]\!]_W\tau = \bigcap_{a_{\heartsuit\alpha} \in \Gamma_j\sigma} \tau(a_{\heartsuit\alpha}) = \bigcap_{a_{\heartsuit\alpha} \in \Gamma_j\sigma} (\widehat{[\![\alpha]\!]_C} \cap \mathsf{rec}(\alpha, \heartsuit\alpha, x))$ and for each $a_{\heartsuit\alpha} \in \Gamma_j\sigma$, we have $\alpha \in \Gamma_j\kappa = l(w_i)$, $y \in \widehat{[\![\alpha]\!]_C}$ and $y \in \mathsf{rec}(\alpha, \heartsuit\alpha, x)$; the latter follows since $\lceil w_i \rceil = y$, $l(w_i) = \Gamma_j\kappa$ and – as we have $\alpha \in \Gamma_j\kappa$ – $\alpha \in \mathsf{Tr}(\heartsuit\alpha, \mathsf{code}(R, \kappa, j))$. Hence we have $y \in \widehat{[\![\alpha]\!]_C} \cap \mathsf{rec}(\alpha, \heartsuit\alpha, v)$ for each $a_{\heartsuit\alpha} \in \Gamma_j\sigma$ so that $y \in [\![\Gamma_j\sigma]\!]_W\tau \neq \emptyset$ and hence $[\![\theta]\!]_{TW}\tau \neq \emptyset$. Put $\xi(x) = t$ for some $t \in [\![\theta]\!]_{TW}\tau$. □

Definition 5.2.38. Let (W, L) be a timed-out tableau. *Saturated nodes* (also referred to as *state nodes*) are nodes $v \in W$ with $\bot \notin l(v)$ and with the property that for all formulas $\psi_1 \wedge \psi_2 \in l(v)$, $\psi_1 \vee \psi_2 \in l(v)$ or $\eta X.\, \psi_3 \in l(v)$, $l(v)$ contains ψ_1 and/or ψ_2 or $\psi_3[X \mapsto \eta X.\, \psi_3]$, respectively. By commutation of the non-modal rules, we can fix an order on the rule applications to non-state nodes and apply just a single non-modal rule to such states; furthermore, we apply modal rules only to saturated nodes. Let L' denote the relation that is obtained from L in this way. As (W, L) is Éloïse-determined, L' is loop-free functional in the set of unsaturated nodes and we can define, for each node $v \in W$, *the* next node $\lceil v \rceil$ in (W, L) that is saturated as there is exactly one sequence $v = v_1\, L\, v_2 \dots v_{m-1}\, L\, v_m = \lceil v \rceil$ for which v_1 to v_{m-1} are unsaturated nodes (note that if v is saturated, then $m = 0$ and $\lceil v \rceil = v$).

Lemma 5.2.39. *If there is a timed-out tableau (W, L) for ψ_0, then ψ_0 is satisfiable in a model of size at most $|W|$.*

Proof. Let (W, L) be a timed-out tableau for ψ_0, that is, an Éloïse-determined pre-tableau for ψ_0 such that the non-emptiness region of the tableau automaton $\mathsf{A}(W, L)$ is the empty set. As described in Definition 5.2.38, we assume that the relation L is loop-free functional at unsaturated nodes. As carrier of the model, we take the set $C \subseteq W$ of saturated nodes from W. As (W, L) is an Éloïse-determined pre-tableau that is loop-free functional in $\overline{C}$, there is by Lemma 5.2.37 a strongly coherent coalgebra $\mathcal{C} = (C, \xi)$. By the Truth Lemma 5.2.45 below, we have that for all $\psi \in \mathsf{FL}$, $\widehat{[\![\psi]\!]} \subseteq [\![\psi]\!]$ in $\mathcal{C}$, i.e. $\mathcal{C}$ is a model for ψ_0. Also, $|C| \leq |W|$. □

It remains to prove the Truth Lemma; to show the satisfaction of least fixpoints in $\mathcal{C}$, we put the time-out information (recall Definition 5.2.26) from $\mathsf{A}(W, L)$ to use.

For the rest of the section, we fix a timed-out tableau (W, L) that is loop-free functional at unsaturated nodes and a strongly coherent coalgebra $\mathcal{C} = (C, \xi)$ with $C \subseteq W$ over the saturated nodes from W, as described above.

Definition 5.2.40. Let $v \in C$ be a saturated node, let ψ be a formula with $v \in \widehat{[\![\psi]\!]}$ and let $\overline{m}$ be a time-out vector. If there is a $w \in W$ with $\psi \in l(w)$ such that $\lceil w \rceil = v$ and $w \in \mathsf{to}(\psi, \overline{m})$, then we say that v has (not necessarily minimal) time-outs $\overline{m}$ for ψ and define $\mathsf{uto}(\psi, \overline{m}) \subseteq C$ as

$$\mathsf{uto}(\psi, \overline{m}) = \{v \in C \mid v \text{ has time-outs } \overline{m} \text{ for } \psi\}.$$

Definition 5.2.41. Let (ψ, σ) be a deferral with $\sigma = [X_j \mapsto \chi_j]; \ldots; [X_0 \mapsto \chi_0]$, where $\chi_i = \eta_i X_i.\, \psi_i$. Also let $\overline{m} = (m_{k-1}, \ldots, m_q)$ be a time-out vector with $m_i \leq n$ for all $q \leq i < k$. We define the interpretation $i_{(\sigma,\overline{m})} : \mathbf{V} \to \mathcal{P}(C)$ by lexicographic induction over $\overline{m}$. For each $X_q \in \mathbf{V}$ with $\sigma(X_q) = \chi_q\kappa$, where $\kappa = [X_{q-1} \mapsto \chi_{q-1}]; \ldots; [X_0 \mapsto \chi_0]$, if $\eta_q = \mu$ and $m_{\mathsf{al}(\chi_q\kappa)} = 0$, then put $i_{(\sigma,\overline{m})}(X_q) = \emptyset$; if $\eta_q = \mu$ and $m_{\mathsf{al}(\chi_q\kappa)+1} > 0$, then put $i_{(\sigma,\overline{m})}(X_q) = [\![\psi_q]\!]_{i_{([X_q \mapsto \chi_q];\kappa,\overline{o})}}$ where $\overline{o} = (m_{k-1}, \ldots, m_{\mathsf{al}(\chi_q\kappa)+1} - 1) < \overline{m}$ and if $\eta_q = \nu$, then put $i_{(\sigma,\overline{m})}(X_q) = \mathsf{uto}(\chi_q\kappa, \overline{m}) \cap \widehat{[\![\chi_q\kappa]\!]}$. Also put $[\![\psi]\!]\sigma_{\overline{m}} = [\![\psi]\!]_{i_{(\sigma,\overline{m})}}$.

Definition 5.2.42. A formula ϕ is *closed-respected* if $\widehat{[\![\eta X.\, \psi]\!]} \subseteq [\![\eta X.\, \psi]\!]$ for each closed fixpoint literal $\eta X.\, \psi < \phi$. A formula ϕ is *respected* if $\widehat{[\![\phi]\!]} \subseteq [\![\phi]\!]$.

Lemma 5.2.43. *Let (ψ, σ) be a deferral with $\sigma = [X_1 \mapsto \chi_1]; \ldots; [X_n \mapsto \chi_n]$, let $\psi\sigma$ be closed-respected and let $\overline{m}$ be a time-out vector. Then*

$$\mathsf{uto}(\psi\sigma, \overline{m}) \cap \widehat{[\![\psi\sigma]\!]} \subseteq [\![\psi]\!]\sigma_{\overline{m}}.$$

Proof. We proceed by lexicographic induction over $(\overline{m}, \psi)$, where the first component $\overline{m} = (m_{k-1}, \ldots, m_q)$ is relevant for the least fixpoint case. We distinguish upon the shape of ψ: The cases $\psi = \bot$ and $\psi = \top$ cannot occur since bases of deferrals contain a free fixpoint variable by definition. If $\psi = X_i$, then $\widehat{[\![X_i\sigma]\!]} = \widehat{[\![\sigma(X_i)]\!]} = \widehat{[\![\chi_i\kappa]\!]} = [\![\psi_i([X_i \mapsto \chi_i]; \kappa)]\!]$ for $\kappa = [X_{i+1} \mapsto \chi_{i+1}]; \ldots; [X_n \mapsto \chi_n]$, $\chi_i = \eta_i X_i.\, \psi_i$. If $\eta_i = \mu$, then we note that the base case that $m_{\mathsf{al}(\chi_i\kappa)} = 0$ cannot occur since $\mathcal{C}$ is built over a timed-out tableau. Hence we have $[\![X_i]\!]\sigma_{\overline{m}} = [\![\psi_i]\!]([X_i \mapsto \chi_i]; \kappa)_{\overline{m}'}$, where $\overline{m}' = (m_0, \ldots, m_{\mathsf{al}(\chi_i\kappa)} - 1) < \overline{m}$ and we have $\mathsf{uto}(\psi\sigma, \overline{m}) \subseteq \mathsf{uto}(\psi_i([X_i \mapsto \chi_i]; \kappa), \overline{m}')$; we are done by the induction hypothesis. If $\eta_i = \nu$, then $[\![X_i]\!]\sigma_{\overline{m}} = \mathsf{uto}(\chi_i\kappa, \overline{m}) \cap \widehat{[\![\chi_i\kappa]\!]}$ and we are done. If $\psi = \psi_1 \wedge \psi_2$, then

$$\mathsf{uto}((\psi_1 \wedge \psi_2)\sigma, \overline{m}) \cap \widehat{[\![(\psi_1 \wedge \psi_2)\sigma]\!]} = \mathsf{uto}(\psi_1\sigma \wedge \psi_2\sigma, \overline{m}) \cap \widehat{[\![\psi_1\sigma]\!]} \cap \widehat{[\![\psi_2\sigma]\!]}$$

and

$$[\![\psi_1 \wedge \psi_2]\!]\sigma_{\overline{m}} = [\![\psi_1]\!]\sigma_{\overline{m}} \cap [\![\psi_2]\!]\sigma_{\overline{m}}.$$

Let $v \in \mathsf{uto}(\psi_1\sigma \wedge \psi_2\sigma, \overline{m}) \cap \widehat{[\![\psi_i\sigma]\!]}$ for $i \in \{1, 2\}$. If ψ_i is closed, then $\widehat{[\![\psi_i\sigma]\!]} = \widehat{[\![\psi_i]\!]} \subseteq [\![\psi_i]\!] = [\![\psi_i]\!]\sigma_{\overline{m}}$ since $\psi\sigma$ is closed respected and hence ψ_i is respected. If ψ_i is open, then

we have $v \in \mathsf{uto}(\psi_i\sigma, \overline{m})$ and (ψ_i, σ) is a deferral so that the induction hypothesis finishes the case. If $\psi = \psi_1 \vee \psi_2$, then we have

$$\mathsf{uto}((\psi_1 \vee \psi_2)\sigma, \overline{m}) \cap \widehat{[\![(\psi_1 \vee \psi_2)\sigma]\!]} = \mathsf{uto}(\psi_1\sigma \vee \psi_2\sigma, \overline{m}) \cap (\widehat{[\![\psi_1\sigma]\!]} \cup \widehat{[\![\psi_2\sigma]\!]})$$

and

$$[\![\psi_1 \vee \psi_2]\!]\sigma_{\overline{m}} = [\![\psi_1]\!]\sigma_{\overline{m}} \cup [\![\psi_2]\!]\sigma_{\overline{m}}.$$

Let $v \in \mathsf{uto}(\psi_1\sigma \vee \psi_2\sigma, \overline{m}) \cap (\widehat{[\![\psi_1\sigma]\!]} \cup \widehat{[\![\psi_2\sigma]\!]})$. If ψ_i is closed for some $i \in \{1, 2\}$, then $\widehat{[\![\psi_i\sigma]\!]} = \widehat{[\![\psi_i]\!]} \subseteq [\![\psi_i]\!] = [\![\psi_i]\!]\sigma_{\overline{m}}$ since $\psi\sigma$ is closed respected and hence ψ_i is respected, and we are done; otherwise, since (W, L) is a timed-out Éloïse-determined tableau, a time-out respecting disjunct is chosen by Éloïse when tracking $\psi_1\sigma \vee \psi_2\sigma$ through the disjunction rule, i.e. we have $v \in \mathsf{uto}(\psi_j\sigma, \overline{m}) \cap \widehat{[\![\psi_j\sigma]\!]}$ for $c(v', 1) = j$, where $v' \in \mathsf{to}(\psi\sigma, \overline{m})$ is a witness node for $v \in \mathsf{uto}(\psi\sigma, \overline{m})$ to which the disjunction rule is applied. As (ψ_j, σ) is a deferral, the induction hypothesis finishes the case. If $\psi = \heartsuit\psi_1$, then

$$\begin{aligned}
\mathsf{uto}(\psi\sigma, \overline{m}) \cap \widehat{[\![(\heartsuit\psi_1)\sigma]\!]} &\subseteq \{v \in \mathsf{uto}(\psi\sigma, \overline{m}) \mid \xi(v) \in [\![\heartsuit]\!](\widehat{[\![\psi_1\sigma]\!]} \cap \mathsf{rec}(\heartsuit\psi_1\sigma, \psi_1\sigma, v)\} \\
&\subseteq \{v \in \mathsf{uto}(\psi\sigma, \overline{m}) \mid \xi(v) \in [\![\heartsuit]\!](\widehat{[\![\psi_1\sigma]\!]} \cap \mathsf{uto}(\psi_1\sigma, \overline{m}))\} \\
&\subseteq \xi^{-1}[[\![\heartsuit]\!](\widehat{[\![\psi_1\sigma]\!]} \cap \mathsf{uto}(\psi_1\sigma, \overline{m}))] \\
&\subseteq \xi^{-1}[[\![\heartsuit]\!]([\![\psi_1]\!]\sigma_{\overline{m}})] \\
&= [\![\heartsuit\psi_1]\!]\sigma_{\overline{m}},
\end{aligned}$$

where the first inclusion holds by strong coherence, the second inclusion holds since if v has time-outs $\overline{m}$ for $\heartsuit\psi_1\sigma$, then every state from $\mathsf{rec}(\heartsuit\psi_1\sigma, \psi_1\sigma, v)$ has time-outs $\overline{m}$ for $\psi_1\sigma$ and the fourth inclusion follows by monotonicity of $[\![\heartsuit]\!]$ from the induction hypothesis. If $\psi = \nu X_j.\, \psi_j$, then

$$\begin{aligned}
\mathsf{uto}(\nu X_j.\, \psi_j\sigma, \overline{m}) \cap \widehat{[\![(\nu X_j.\, \psi_j)\sigma]\!]} &= \mathsf{uto}(\psi_j\kappa, \overline{m}) \cap \widehat{[\![\psi_j\kappa]\!]} \\
&\subseteq [\![\psi_j]\!]\kappa_{\overline{m}} \\
&= [\![\psi_j]\!]_{i_{(\sigma,\overline{m})}[X_j \mapsto \mathsf{uto}(\psi\sigma,\overline{m}) \cap \widehat{[\![\psi\sigma]\!]}]} \\
&= [\![\psi_j]\!]^{X_j}_{i_{(\sigma,\overline{m})}}(\mathsf{uto}(\psi\sigma, \overline{m}) \cap \widehat{[\![\psi\sigma]\!]}),
\end{aligned}$$

where $\kappa = [X_j \mapsto \psi]; \sigma$ and the inclusion is by the induction hypothesis, showing by coinduction that $\mathsf{uto}(\nu X_j.\, \psi_j\sigma, \overline{m}) \cap \widehat{[\![(\nu X_j.\, \psi_j)\sigma]\!]} \subseteq [\![\nu X_j.\, \psi_j]\!]\sigma_{\overline{m}}$, as required. If $\psi = \mu X_j.\, \psi_j$, then we have

$$\widehat{[\![(\mu X_j.\, \psi_j)\sigma]\!]} = \widehat{[\![(\psi_j[X_j \mapsto \psi])\sigma]\!]} = \widehat{[\![\psi_j\kappa]\!]}$$

and

$$[\![\mu X_j.\, \psi_j]\!]\sigma_{\overline{m}} = [\![\psi_j]\!]\kappa_{\overline{m}'},$$

where $\kappa = [X_j \mapsto \psi];\sigma$ and $\overline{m}'$ is some time-out vector with $\overline{m}' \leq \overline{m}$. As (ψ_j, κ) is a deferral and since $\mathcal{C}$ is constructed over a timed-out tableau, we have $\mathsf{uto}(\psi\sigma, \overline{m}) \subseteq \mathsf{uto}(\psi_j\kappa, \overline{m}')$. By the induction hypothesis and since $[\![\mu X_j.\, \psi_j]\!]\sigma_{\overline{m}} = [\![\psi_j]\!]\kappa_{\overline{m}'}$, we have $\mathsf{uto}(\psi_j\kappa, \overline{m}') \cap \widehat{[\![\psi_j\kappa]\!]} \subseteq [\![\psi_j]\!]\kappa_{\overline{m}'}$, which finishes the case since $[\![\mu X_j.\, \psi_j]\!]\sigma_{\overline{m}} = [\![\psi_j]\!]\kappa_{\overline{m}'}$. □

Lemma 5.2.44. *All closed fixpoint literals are respected.*

Proof. Let $\eta X.\, \phi$ be a closed fixpoint literal and notice that $(\eta X.\, \phi, \epsilon)$ is a deferral. We proceed by induction over the depth n of nesting of closed fixpoint literals in $\eta X.\, \phi$. If $n = 1$, then ψ contains no closed fixpoint literals and $\eta X.\, \phi$ hence is closed-respected. If $n > 1$, then any closed fixpoint literal $\eta' Y.\phi' \leq \phi$ has a depth of nesting of closed fixpoint literals less than n and is respected by the induction hypothesis so that $\eta X.\, \phi$ is closed-respected. In both cases, Lemma 5.2.43 finishes the proof with $\psi = \eta X.\, \phi$ and $\sigma = \epsilon$ (i.e. the empty sequence) and $\overline{m} = \epsilon$ (i.e. the maximal time-out vector): since $\mathcal{C}$ is built over a timed-out tableau, we have $\widehat{[\![\psi\epsilon]\!]} \subseteq \mathsf{uto}((\psi\epsilon), \epsilon)$ and hence $\widehat{[\![\psi]\!]} = \widehat{[\![\psi\epsilon]\!]} = \mathsf{uto}((\psi\epsilon), \epsilon) \cap \widehat{[\![\psi\epsilon]\!]} \subseteq [\![\psi]\!]\epsilon = [\![\psi]\!]$ where the inclusion is by Lemma 5.2.43.□

Lemma 5.2.45 (Truth). *In the coalgebra $\mathcal{C}$, we have $\widehat{[\![\psi]\!]} \subseteq [\![\psi]\!]$ for all $\psi \in \mathsf{FL}$.*

Proof. We proceed by induction over ψ. If $\psi = \bot$ or $\psi = \top$, then $\widehat{[\![\psi]\!]} = [\![\psi]\!]$ by definition. For the propositional connectives, the inductive step is straightforward. If $\psi = \heartsuit\psi_1$, then $\widehat{[\![\heartsuit\psi_1]\!]} \subseteq \xi^{-1}[[\![\heartsuit]\!]\widehat{[\![\psi_1]\!]}] \subseteq \xi^{-1}[[\![\heartsuit]\!][\![\psi_1]\!]] = [\![\heartsuit\psi_1]\!]$, where the first inclusion holds by strong coherence of $\mathcal{C}$ and the second inclusion follows by monotonicity of $[\![\heartsuit]\!]$ from the induction hypothesis. If $\psi = \eta X.\, \psi_1$, then ψ is a closed fixpoint literal and Lemma 5.2.44 finishes the case. □

Corollary 5.2.46. *The formula ψ_0 is satisfiable if and only if a timed-out tableau for ψ_0 exists.*

5.2.3 Satisfiability Games via Determinization

As we have seen, the existence of timed-out tableaux coincides with the existence of models; to decide the satisfiability of fixpoint formulas, it thus suffices to construct and solve suitable satisfiability games that are won by player Éloïse if and only if a timed-out tableau exists for the respective input formula. The standard approach to obtain such satisfiability games is to view formulas as alternating tracking automata where Abélard chooses rule applications and Éloïse chooses conclusions of rule applications; a combination of two alternating moves induces a single transition in the non-alternating tracking automaton $\mathsf{N}(\psi_0)$ from Definition 5.2.21. In this set-up, a formula is satisfiable if and only if Éloïse can repeatedly choose – for any matching rule application that Abélard chooses – an according conclusion node such that Abélard cannot enforce a bad branch; then Éloïse has a strategy

that turns the alternating tracking automaton into a universal automaton in which all branches are good, i.e. into a timed-out tableau. We refer to these standard satisfiability games as *satisfiability games via determinization* for the reason that either the left or the right conjunct can be tracked through the application of the conjunction rule ($\wedge$) to some conjunction so that the alternating tracking automaton is nondeterministic at Abélard nodes. To see the alternating tracking automaton as a game however, every rule application or choice of a conclusion in the pre-tableau has to correspond to exactly one transition in the alternating tracking automaton, that is, the alternating tracking automaton has to be deterministic w.r.t. rule applications. Considering some input formula ψ_0, we thus determinize the tracking automaton $\mathsf{N}(\psi_0)$, obtaining the equivalent DPA $\mathsf{D}(\psi_0) = (W, \Sigma, \delta', v_0, \alpha')$ with labelling function $l : W \to \mathcal{P}(\mathsf{FL}(\psi_0))$; depending on the properties of $\mathsf{N}(\psi_0)$ we may use Safra/Piterman-style determinization or one of the other determinization methods described in Chapter 4 above. Then we remove all transitions that do not correspond to matching applications of tableau rules from the determinized tracking automaton, i.e. we put

$$\delta = \{(v, \mathsf{code}(R, \sigma, j), w) \in \delta' \mid (R, \sigma) \in \mathcal{R}(l(v))\}.$$

The set δ thus retains just those transitions from δ' that correspond to *matching* applications of tableau rules; we define the automaton $\mathsf{D}'(\psi_0) = (W, \Sigma, \delta, v_0, \alpha')$.

Given a pre-tableau (W, L) for ψ_0, let $w \in \Sigma^\omega$ be an infinite word that encodes an infinite L-path $p = x_0, x_1, \ldots \in W^\omega$ through (W, L), where $l(x_0) = \{\psi_0\}$; for $j \geq 0$, $w(j) = (R_j, \sigma_j, i_j)$ then encodes an application of rule $R_j = (\Gamma_0/\Gamma_1, \ldots, \Gamma_l)$ to $l(x_j)$, i.e. with $\Gamma_0\sigma_j \subseteq l(x_j)$, $\Gamma_{i_j}\sigma_j = l(x_{j+1})$ and $x_{j+1} \in L(x_j)$. As $\mathsf{D}'(\psi_0)$ is equivalent to $\mathsf{N}(\psi_0)$ when we restrict our attention to words that encode some path through some pre-tableau, $w \in \mathsf{D}'(\psi_0)$ if and only if there is an infinite sequence $\Psi = \psi_0, \psi_1, \ldots$ of formulas with $\psi_{i+1} \in \mathsf{Tr}(\psi_i, w(i))$ such that $\min(\mathsf{Inf}(\alpha \circ \mathsf{trans}(\Psi)))$ is odd and Ψ is a trace of ψ_0 through some branch of some pre-tableau. In other words, $\mathsf{D}'(\psi_0)$ accepts exactly those words that encode some bad branch.

Since we are interested in detecting good branches and since $\mathsf{D}'(\psi_0)$ is deterministic, we complement $\mathsf{D}'(\psi_0)$ by increasing each priority by 1 and obtain the DPA $\mathsf{E}(\psi_0) = (W, \Sigma, \delta, v_0, \beta)$ with $\beta(v) = \alpha'(v) + 1$ for $v \in W$; this automaton accepts exactly those words that encode only infinite paths through pre-tableaux for ψ_0 on which for each formula that can be tracked forever, the alternation-level of the outermost fixpoint that is unfolded infinitely often is even. Such paths are called *good branches*.

Definition 5.2.47 (Satisfiability games via determinization). Let ψ_0 be a coalgebraic μ-calculus formula and let $\mathsf{E}(\psi_0) = (W, \Sigma, \delta, v_0, \beta)$ be the corresponding determinized tracking automaton. We define the *satisfiability game via determinization* (for ψ_0) as a parity game $\mathsf{G}(\psi_0) = (U, E, \gamma)$ by

$$U = W \cup (W \times \mathsf{code}(\psi_0)),$$

where nodes $w \in W$ belong to Abélard and nodes $(w, (R, \sigma)) \in V \times \mathsf{code}(\psi_0)$ belong to Éloïse. For Abélard-nodes w, we put

$$E(w) = \{(w, \mathsf{code}(R, \sigma)) \mid (R, \sigma) \in \mathcal{R}(l(w))\}$$

and for Éloïse-nodes $(w, \mathsf{code}(R, \sigma))$ with $R = (\Gamma_0/\Gamma_1, \ldots, \Gamma_l)$, we put

$$E(w, \mathsf{code}(R, \sigma)) = \{w' \mid \delta(w, (R, \sigma, j)) = \{w'\} \text{ where } 0 \leq j \leq l\}.$$

For Éloïse-moves $t = ((w, \mathsf{code}(R, \sigma)), w') \in E$ with $R = (\Gamma_0/\Gamma_1, \ldots, \Gamma_l), 0 \leq j \leq l$ and $\{w'\} = \delta(w, (R, \sigma, j))$, we put $\gamma(t) = \beta(w, (R, \sigma, j), w')$ and for Abélard-moves $t = (w, (w, \mathsf{code}(R, \sigma))) \in E$, we put $\gamma(t) = 0$.

Theorem 5.2.48. *Player Éloïse wins v_0 in $\mathsf{G}(\psi_0)$ if and only if there is a pre-tableau for ψ_0.*

Proof. Let $\mathsf{win} = [\![\phi_{\mathsf{G}(\psi_0)}]\!]$ denote the winning region of Éloïse in G (recalling Definition 4.3.6) and let $v_0 \in \mathsf{win}$. For each Abélard-node $v \in \mathsf{win}$, there is some $\overline{m}$ such that v has nested time-outs $\overline{m}$ w.r.t. $\mathsf{G}(\psi_0)$, which means that for each Abélard-move, i.e. for each rule application $(R, \sigma) \in \mathcal{R}(l(v))$, there is an Éloïse-move that chooses a number j such that $\delta(v, \mathsf{code}(R, \sigma, j)) = \{w\}$ where $w \in \mathsf{win}$ and w has game time-outs $\overline{m}'$ for some $\overline{m}'$ with the following properties: let $\gamma((v, \mathsf{code}(R, \sigma, j)), w) = p$. If p is odd, then $\overline{m}'(p) < \overline{m}(p)$ and $\overline{m}' < \overline{m}$. If p is even, then $\overline{m}'(p) = \overline{m}(p)$ and $\overline{m}'|_p \leq \overline{m}$, where $\overline{m}'|_p$ denotes the first p components of $\overline{m}'$. We construct a timed-out tableau as follows: for each node $v \in \mathsf{win}$ with $\mathcal{R}(l(v)) = ((R_1, \sigma_1), \ldots, (R_n, \sigma_n))$ and all $1 \leq i \leq n$, we chose a node w_i in a game time-outs respecting manner, as detailed above and put $L(v) = (w_1, \ldots, w_n)$. This results in a structure (win, L) where for each $v \in \mathsf{win}$ with $\mathcal{R}(l(v)) = ((R_1, \sigma_1), \ldots, (R_n, \sigma_n))$ we have $L(v) = (w_1, \ldots, w_n)$ and for each $(R_i, \sigma_i) \in \mathcal{R}(l(v))$, there is $1 \leq j \leq l$ such that $\Gamma_j \sigma_i \subseteq l(w_i)$ and $\delta(v, \mathsf{code}(R_i, \sigma_i, j)) = \{w_i\}$; we record the choices of conclusions by means of a function c by putting $c(v, i) = j$, where i and j are as described above. Thus (win, L) is an Éloïse-determined pre-tableau. It remains to see that the non-emptiness region of the tableau automaton $\mathsf{A}(\mathsf{win}, L)$ is the empty set, i.e. that for each pair (v, ψ) with $v \in \mathsf{win}$, $\psi \in l(v)$, no run of $\mathsf{A}(E, L)$ that starts at (v, ψ) is accepting. Let (v, ψ) be such a pair and let $\rho \in \mathsf{run}(\mathsf{A}(E, L), (v, \psi), *^\omega)$ be such a run of the tableau automaton. Then ρ corresponds to one way to trace ψ along one particular branch $\pi_1 \circ \rho$ of the Éloïse-determined pre-tableau (win, L); let $w \in \Sigma^\omega$ be the word that encodes this branch. The automaton $\mathsf{E}(\psi_0) = (W, \Sigma, \delta, v_0, \beta)$ accepts any word that encodes some branch if and only if the word does not encode a bad branch. Since L respects the game time-outs for the priority function γ as described above and the game is built over $\mathsf{E}(\psi_0)$, we know that w does not encode a bad branch. Thus for any trace of ψ along the branch encoded by w, and in particular for the trace $\psi_2 \circ \rho$, the outermost fixpoint to which the traced formula evolves infinitely often is a greatest fixpoint. Hence ρ is accepting.

For the backwards direction, we have to show that if there is some timed-out tableau (W, L) for ψ_0, then $v_0 \in \mathsf{win}$. The timed-out tableau is Éloïse-determined and hence

comes with a function $c : W \times \mathbb{N} \to \mathbb{N}$ that is, as it turns out, a winning strategy for Éloïse in the game: for all $v \in W$ with $\mathcal{R}(l(v)) = ((R_1, \sigma_1), \ldots, (R_n, \sigma_n))$, $c(v, i)$ denotes the conclusion that Éloïse chooses if Abélard chooses the rule application (R_i, σ_i). Since the emptiness-region of the tableau automaton $\mathsf{A}(W, L)$ is the empty set, every play that conforms to the strategy c corresponds to a good branch; thus Éloïse wins every play that conforms to c, i.e. $v_0 \in \mathsf{win}$. □

Definition 5.2.49 (Global caching via determinization). We define the corresponding instance of the generic global caching algorithm that solves the satisfiability games from Definition 5.2.47 on-the-fly. Again we make use of the determinized tracking automaton $\mathsf{E}(\psi_0) = (W, \Sigma, \delta, v_0, \beta)$. As set of focused nodes, we take $\mathbf{C} = W$, the global caching algorithm uses propagation with $\mathsf{idx}(\beta)$ many priorities and we take δ as tracking function, i.e. for $v \in W$ and $a \in \Sigma = \mathsf{code}(\psi_0)$, we put $\mathsf{track}(v, a, i) = \delta_i(v, a)$. The labelling function $l : W \to \mathcal{P}(\mathsf{FL}(\psi_0))$ assigns to each macrostate in W the according label in the determinized tracking automaton. The initial focused node is just v_0.

By Definition 5.2.21, the automaton $\mathsf{N}(\psi_0)$ is an NPA with at most $n = |\psi_0|$ states and at most $k = \mathsf{ad}(\psi_0)$ priorities and can be transformed by Lemma 4.1.31 to an NBA with blow-up linear in k. Additionally, we have the following:

Lemma 5.2.50. *Let $\mathsf{N}(\psi_0)$ be the tracking automaton for ψ_0. Then we have:*

1. *If ψ_0 is alternation-free, then $\mathsf{N}(\psi_0)$ is an NCBA.*
2. *If ψ_0 is linear, then $\mathsf{N}(\psi_0)$ is a limit-linear CBA with at most $|\psi_0|$ synchronizing states in each compartment.*
3. *If ψ_0 is depth-1 linear, then $\mathsf{N}(\psi_0)$ is a limit-stationary CBA.*
4. *If ψ_0 is aconjunctive, then $\mathsf{N}(\psi_0)$ is a limit-deterministic PA.*

Proof. 1. If ψ_0 is alternation-free, then ψ_0 contains no dependent nesting of least and greatest fixpoints and hence we have $\mathsf{ad}(\psi_0) \leq 1$. Thus the tracking automaton has $k \leq 1$ priorities so that $\mathsf{N}(\psi_0) = (V, \Sigma, \Delta, \psi_0, \alpha)$ is a NCBA with

$$\alpha_2 = F = \{(\psi, a, \psi') \in \Delta \mid \mathsf{al}(\psi) = \mathsf{al}(\psi') = 1\}$$

Thus F consists of transitions that transform deferrals to deferrals and $\mathsf{N}(\psi_0)$ accepts any word that encodes a sequence of rule application through which some deferral can be tracked indefinitely.

2. If ψ_0 is linear, then all conjunctions and disjunctions contain at most one fixpoint variable so that every deferral in $\mathsf{FL}(\psi_0)$ has exactly one deferral as a direct subformula. As linear formulas are alternation-free, $\mathsf{N}(\psi_0)$ is a CBA and accepting transitions $(\psi, a, \psi') \in F$ transform deferrals to deferrals and compartments $C(\chi)$ are the sets of deferrals for closed irreducible fixpoints χ. For all $t \in F$ we have that $\pi_3(t) = \psi$ is a deferral and there is at most one ψ' such that there is $a \in \Sigma$ such that $(\psi, a, \psi') \in F$.

Thus we have $|\pi_3[F|_t \setminus \mathsf{Id}_F]| \leq 1$, i.e. F is linear. As set of progressing letters P, we have the set of letters $a = \mathsf{code}(R, \sigma, i) \in \Sigma$ for which R is a modal rule. As all modal rules remove one layer of modal operators, they leave no formula unchanged and there is no transition $(\psi, a, \psi) \in \Delta$ for any ψ. Modal literals $\heartsuit\psi$ then are synchronizing states as they can be matched by some modal rule and thus have at least one a-successor for some $a \in P$. The number of synchronizing states is bounded by $|\psi_0|$ and by guardedness, each compartment in $\mathsf{N}(\psi_0)$ contains at least one modal literal, i.e. one synchronizing state. As set of choice pairs Q, we have all pairs $((\vee), \sigma, 1), ((\vee), \sigma, 2)) \in \Sigma^2$ where σ is defined on $\{a, b\}$. Uniform words thus contain infinitely progressing (i.e. modal) steps and for any two applications of the disjunction rule to a principal disjunction $\psi_1 \vee \psi_2$ that occur between two consecutive modal steps, the word chooses the same disjunct ψ_i for $i \in \{1, 2\}$. All compartments $C(\chi)$ with $\chi = \mu X.\, \psi$ have the single entry transition $(\chi, (\mu, \sigma, 1), \psi[X \mapsto \chi]) \in F$ where $\sigma(a) = \psi$ and as all fixpoint variables in ψ_0 occur in the shape $\heartsuit X$ for some $\heartsuit \in \Lambda$, this transition can only be reached from within $C(\chi)$ by a modal rule that progresses from $\heartsuit X$ to X. For each deferral $\psi \in \pi_3[F]$ and each non-progressing letter $a = \mathsf{code}(R, \sigma, i)$ that is not a choice letter, ψ is not principle in the rule application encoded by a and we have a transition $(\psi, a, \psi) \in F$ or ψ is principle in the rule application and $R \in \{(\wedge), (\eta)\}$ and the conclusion of the rule application contains at least one deferral ψ' so that we have $(\psi, a, \psi') \in F$, as required. Hence $\mathsf{N}(\psi_0)$ indeed is a limit-linear CBA.

3. If ψ_0 is depth-1 linear, then $\mathsf{N}(\psi_0)$ is limit-linear by the previous item. It remains to show that every compartment in $\mathsf{N}(\psi_0)$ has just one synchronizing state. Since ψ_0 is depth-1 linear, the fixpoint variable occurs at modal nesting depth exactly 1 in each closed irreducible fixpoint χ so that $C(\chi) = \mathsf{dfr}(\chi)$ contains exactly one modal literal and $\mathsf{N}(\psi_0)$ is limit-stationary.
4. We have to show that all compartments in $\mathsf{N}(\psi_0)$ are deterministic. As non-determinism can only occur at conjunctions in $\mathsf{N}(\psi_0)$, it suffices to realize that by aconjunctivity, for each deferral of the shape $\psi_1 \wedge \psi_2$ only one of the conjuncts, say ψ_1, is a deferral that has to be tracked. Thus we put $\alpha(\psi_1 \wedge \psi_2, ((\wedge), \sigma, 1), \psi_2) = 0$ and note that this does not change the language that $\mathsf{N}(\psi_0)$ accepts. Then $\mathsf{N}(\psi_0)$ is limit-deterministic.

□

We sum up the results of this section:

Corollary 5.2.51. *Let ψ_0 be a coalgebraic μ-calculus formula with $n = |\psi_0|$ and $k = \mathsf{ad}(\psi_0)$. Then we have:*

1. *If ψ_0 is alternation-free, then the satisfiability of ψ_0 can be decided by solving a Büchi game; if ψ_0 is satisfiable, then it has a model of size at most 3^n. In the relational case, the game can be solved in time $2^{\mathcal{O}(n)}$.*

2. *If ψ_0 is depth-1 linear, then the satisfiability of ψ_0 can be decided by solving a Büchi game; if ψ_0 is satisfiable, then it has a model of size at most $n \cdot 2^n$. In the relational case, the game can be solved in time $2^{\mathcal{O}(n)}$.*
3. *If ψ_0 is linear, then the satisfiability of ψ_0 can be decided by solving a Büchi game; if ψ_0 is satisfiable, then it has a model of size at most $n^2 \cdot 2^n$. Again the game can be solved in time $2^{\mathcal{O}(n)}$ in the relational case.*
4. *If ψ_0 is aconjunctive, then the satisfiability of ψ_0 can be decided by solving a parity game with nk priorities; if ψ_0 is satisfiable, then it has a model of size at most $e(nk)!$. In the relational case, the game is of size $\mathcal{O}(nk)!$ and can be solved in time $\mathcal{O}((nk)!^{nk})$.*
5. *The satisfiability of an unrestricted coalgebraic μ-calculus formula ψ_0 can be decided by solving a parity game with nk priorities; if ψ_0 is satisfiable, then it has a model of size at most $n!(nk)^{nk}$. In the relational case, this game is of size $\mathcal{O}(n!(nk)^{nk})$ and can be solved in time $\mathcal{O}((n!(nk)^{nk})^{nk})$.*

Proof. For the items 1. to 3., the formula ψ_0 is alternation-free so that $\mathsf{N}(\psi_0)$ is a CBA and can be determinized by Lemma 4.1.18 to a DCBA; hence the satisfiability game from Definition 5.2.47 is a Büchi game. By the results from Section 4.3, Büchi games of size m can be solved in time $\mathcal{O}(m^3)$. For the remaining cases, $\mathsf{N}(\psi_0)$ is a PA and can by Theorem 4.1.27 be determinized to a DPA so that the corresponding satisfiability game is a parity game. By the results from Section 4.3, parity games of size m with p priorities can be solved in time $\mathcal{O}(m^p)$. A recent algorithm brings this bound down to time quasipolynomial in p [5]. Winning strategies in all these games define timed-out tableaux with the set of Abélard-nodes that Éloïse wins as carrier sets; models exists over these carrier sets so that the number of Abélard-nodes in the games yields an upper bound on model sizes. It remains to verify the stated bounds on the number of Abélard-nodes in the respective games. The Abélard-nodes are just the states of the determinized and complemented tracking automaton. Making use of Lemma 5.2.50, we observe:

1. If ψ_0 is alternation-free, then $\mathsf{N}(\psi_0)$ is an NCBA of size at most n. By Lemma 4.1.18, this automaton can be determinized and complemented to a DBA of size at most 3^n.
2. If ψ_0 is depth-1 linear, then $\mathsf{N}(\psi_0)$ is a limit-stationary CBA of size at most n. By Lemma 4.1.13, this automaton can be determinized and complemented to a DBA of size at most $n \cdot 2^n$. We note that the determinized DCBA is equivalent to $\mathsf{N}(\psi_0)$ only up to uniform and synchronizing acceptance. However, by guardedness, all infinite branches through pre-tableaux contain infinitely many applications of modal rules so that all words that encode some branch in a pre-tableau are progressing. We also restrict our attention to uniform words since bad branches can always be encoded by a uniform word, i.e. a word that makes uniform choices whenever a single disjunction $\psi_1 \vee \psi_2$ is transformed twice (or more often) between any two consecutive applications of modal rules. Similarly, we can restrict our attention – along the lines of Definition 5.2.38 – to synchronizing words, that is, words that encode *saturating branches* (i.e. branches in which modal rules are only applied to saturated nodes). Formally, we accommodate for this restriction to uniform and synchronizing runs

by restricting rule applications in the satisfiability games via determinization and in pre-tableaux so that disjunctions are treated uniformly and modal rules are only applied to saturated nodes, as described above.

3. If ψ_0 is linear, then $\mathsf{N}(\psi_0)$ is a limit-linear CBA of size at most n and with at most n synchronizing states in each compartment. Using Lemma 4.1.13, this automaton can be determinized and complemented to a DBA of size at most $n^2 \cdot 2^n$. As in the previous case, the determinized DCBA is equivalent to $\mathsf{N}(\psi_0)$ only for uniform and synchronizing words, which however suffices for our purposes.
4. If ψ_0 is aconjunctive, then $\mathsf{N}(\psi_0)$ is a limit-deterministic PA of size at most n and with at most k priorities. By Theorem 4.1.23, this automaton can be determinized and complemented to a DPA of size at most $e(nk)!$ and with at most nk priorities.
5. If ψ_0 is a full μ-calculus formula, then $\mathsf{N}(\psi_0)$ is a PA of size at most n and with at most k priorities. By Theorem 4.1.27, this automaton can be determinized and complemented to a DPA of size at most $n!(nk)^{nk}$ and with at most nk priorities.

□

The instances of Algorithm 5.2.19 that result from Definition 5.2.49 solve these games *on-the-fly*, where the propagation step in the algorithm is equivalent to solving the partial satisfiability game by means of the fixpoint iteration algorithm as described in [4].

Corollary 5.2.52. *If the employed set of one-step tableau rules is* ExpTime*-tractable, then the instantiation of Algorithm 5.2.19 to satisfiability games via determinization decides the satisfiability of formulas from the respective fragments of the coalgebraic μ-calculus in* ExpTime*.*

Proof. The ExpTime-tractability of the employed set of rules implies that the set of possible moves for players Éloïse and Abélard at some given node in the automaton game for each formula ψ_0 with $n = |\psi_0|$ can be computed in time exponential in n so that the winning regions in the game can also be computed in time exponential in n. As the algorithm constructs the according satisfiability game for ψ and while attempting to solve the game at most exponentially often, the algorithm runs in time exponential in n as well. □

5.2.4 Satisfiability Games via Focusing

For alternation-free formulas that additionally are aconjunctive, smaller games – so-called *satisfiability games via focusing* – can be defined by allowing Abélard to perform finitely many additional *focusing* moves in each play; then it suffices to *focus* and then track one formula at a time. For satisfiability games via focusing that are won by Éloïse, timed-out tableaux can be constructed by serially focusing and finishing all least fixpoints on a given path. This approach generalizes the previously known focusing games for CTL [31] both to the coalgebraic level of generality and to the alternation-free and aconjunctive fragment.

Definition 5.2.53 (Satisfiability games via focusing). Let ψ_0 be an alternation-free and aconjunctive coalgebraic μ-calculus formula. We define the Büchi game $\mathsf{G}_f(\psi_0) = (W, E, F)$ by

$$W = V \cup (V \times \mathsf{code}(\psi_0)),$$

where $V = (\mathcal{P}(\mathsf{FL}) \times (\mathsf{dfr}(\mathsf{FL}) \cup \{*\}))$, where $\mathsf{dfr}(\mathsf{FL}) \subseteq \mathsf{FL}$ denotes the set of all deferrals from FL and where nodes $(U, v) \in V$ belong to Abélard and nodes $(U, v, (R, \sigma)) \in V \times \mathsf{code}(\psi_0)$ belong to Éloïse; we use $*$ to denote a finished focus. For Abélard-nodes (U, v), we put

$$\begin{aligned} E(U, v) = &\{(U, v, \mathsf{code}(R, \sigma))) \mid (R, \sigma) \in \mathcal{R}(U)\} \cup \\ &\{(U, w) \mid v = *,\ w \in U \text{ is a deferral}\} \end{aligned}$$

and for Éloïse-nodes $(U, v, \mathsf{code}(R, \sigma))$, we put

$$E(U, v, \mathsf{code}(R, \sigma)) = \{(\Gamma_j \sigma, \mathsf{tr}(v, \mathsf{code}(R, \sigma, j))) \mid R = (\Gamma_0 / \Gamma_1, \ldots, \Gamma_l), 0 \leq j \leq l\}$$

where

$$\mathsf{tr}(v, \mathsf{code}(R, \sigma, j)) = \begin{cases} * & \text{if } v = * \text{ or } \mathsf{Tr}(v, \mathsf{code}(R, \sigma, j)) \cap \mathsf{dfr}(\mathsf{FL}) = \emptyset \\ w & \text{if } v \neq * \text{ and } \mathsf{Tr}(v, \mathsf{code}(R, \sigma, j)) \cap \mathsf{dfr}(\mathsf{FL}) = \{w\}, \end{cases}$$

noting that by aconjunctivity, we always have $|\mathsf{Tr}(v, \mathsf{code}(R, \sigma, j)) \cap \mathsf{dfr}(\mathsf{FL})| \leq 1$. Finally, we put

$$F = \{((U, v, \mathsf{code}(R, \sigma)), (\Gamma_j \sigma, w)) \in E \mid w = *\}.$$

Theorem 5.2.54. *Let ψ_0 be an alternation-free and aconjunctive formula, let $n = |\psi_0|$ and let $\mathsf{G}_f(\psi_0)$ be the satisfiability game via focusing for ψ_0 as defined above. Then $|\mathsf{code}(\psi_0)| \in 2^{\mathcal{O}(n)}$ implies $|W| \in 2^{\mathcal{O}(n)}$. Also Éloïse wins the node $(\{\psi_0\}, *)$ in $\mathsf{G}_f(\psi_0)$ if and only if there is a timed-out tableau of size at most $n \cdot 3^n$ for ψ_0.*

Proof. Let Éloïse win the node $(\{\psi_0\}, *)$. Then she has a winning strategy $s : V_\exists \to V_\forall$. We construct a timed-out tableau over the set

$$V = \{(U, W, v) \in V' \times (\mathsf{dfr}(\mathsf{FL}) \cup \{*\}) \mid \text{Éloïse wins node } (U \cup W, v) \text{ in } \mathsf{G}_f(\psi_0) \text{ with } s\},$$

where V' is the set of all functions $f : \mathsf{FL} \to \{0, 1, 2\}$, as introduced in Definition 4.1.16, noting that $|V| \leq 3^n \cdot n$, as required. We have $(\{\psi_0\}, \emptyset, *) \in V$. We apply the strategy s to the game $\mathsf{G}_f(\psi_0)$ and obtain an Éloïse-determined tableau where additionally to selecting rule applications, Abélard has refocusing moves at any node with empty focus $*$. Since s is a winning strategy, all paths in this pre-tableau eventually contain an Éloïse-move $((U, v, \mathsf{code}(R, \sigma)), (\Gamma_j \sigma, w))$ where $\mathsf{Tr}(v, \mathsf{code}(R, \sigma, j)) \cap \mathsf{dfr}(\mathsf{FL}) = \emptyset$, i.e.

the deferral v is finished. We construct a structure over V by storing all deferrals in a given macrostate in a set of tracked formulas and then serially finish individual deferrals from this set. By aconjunctivity, the set of tracked formulas never grows; as individual deferrals are finished one by one, we eventually arrive at a node with the empty set as set of tracked formulas. At this point, we retrack all deferrals in the current macrostate and repeat the described procedure. This guarantees that every deferral is eventually finished, i.e. that the constructed structure indeed is a timed-out tableau. Formally, we construct an Éloïse-determined pre-tableau. For $(U, W, v) \in V$ with $\mathcal{R}(U \cup W) = (R_1, \sigma_1), \ldots, (R_n, \sigma_n)$ and for $1 \leq i \leq n$, we choose a node y_i as follows, recalling that Δ is the transition relation of the nondeterministic tracking automaton $\mathsf{N}(\psi_0)$: If $v \in W$ and $v \neq *$, then we choose $y_i = (\Delta_1(U \cup W, (R, \sigma, j)), \Delta_0(W, (R, \sigma, j)), w)$, where $s(U \cup W, v, (R_i, \sigma_i)) = (\Gamma_j, w)$, noting that if $w \neq *$, then $\Delta_0(v, (R, \sigma, j)) = \{w\}$. If $v \notin W$ or $v = *$ but $W \neq \emptyset$, then we choose $y_i = (\Delta_0(U \cup W, (R, \sigma, j)), \Delta_0(W, (R, \sigma, j)), w)$, where $s(U \cup W, v', (R_i, \sigma_i)) = (\Gamma_j, w)$ and where v' is *some* deferral from W. If $W = \emptyset$, then we choose $y_i = (\Delta_1(U, (R, \sigma, j)), \Delta_0(U, (R, \sigma, j)), *)$, where $s(U, *, (R_i, \sigma_i)) = (\Gamma_j, *)$. Furthermore, we put $c((U, W, v), i) = j$ and $L(U, W, v) = (y_1, \ldots, y_n)$. Then (V, L) together with the choice function c is an Éloïse-determined pre-tableau. It remains to show that every branch in the tableau automaton $\mathsf{A}(V, L)$ is accepting. So let $((U, W, v), \psi)$ be a state in the tableau automaton. We distinguish three cases: 1) If $\psi = v$, then the deferral ψ is eventually finished on all paths through $\mathsf{A}(V, L)$ that start at $((U, W, v), \psi)$. 2) If $\psi \neq v$ but $\psi \in W$, then we proceed by induction over $m = |W|$. On every path through $\mathsf{A}(V, L)$ that starts at $((U, W, v), \psi)$, a node $((U', W', *), \psi')$ where the deferral v is finished is reached eventually. In the next step, some deferral from W' is chosen as new focus and we have $|W'| < m$. Eventually, ψ is finished or we arrive, by induction, at the situation that $\psi = v$, at the latest when $\psi \in W$, $v \in W$ and $|W| = 1$. Then we proceed as in the first case. 3) If $\psi \notin W$, then we see by the argumentation from the second case above that every path through $\mathsf{A}(V, L)$ that starts at $((U, W, v), \psi)$ eventually reaches a node $((U', \emptyset, v'), \psi')$. If the deferral ψ has not been finished up to this point, then we have – assuming $L(U', \emptyset, v') = (y_1, \ldots, y_n)$ – that for each $(R_i, \sigma_i) \in \mathcal{R}(U', \emptyset, v')$, $y_1 = (U'', W'', \psi'')$, where the deferral at hand is contained in W''. Proceed as in the first or second case.

For the converse direction, let (V, L) be a timed-out tableau for ψ_0. Then the choice function c is an Éloïse-strategy that wins the node $(\{\psi_0\}, *)$ in $\mathsf{G}_f(\psi_0)$. □

Corollary 5.2.55. *Let ψ_0 be an aconjunctive and alternation-free coalgebraic μ-calculus formula with $n = |\psi_0|$. Then the satisfiability of ψ_0 can be decided by solving a Büchi game; if ψ_0 is satisfiable, then it has a model of size at most $n \cdot 3^n$. In the relational case, the game can be solved in time $2^{\mathcal{O}(n)}$.*

We define the instance of the generic global caching algorithm that solves the satisfiability games from Definition 5.2.53 on-the-fly:

Definition 5.2.56 (Global caching via focusing). As set of focused nodes, we take $\mathbf{C} = V$; the global caching algorithm uses propagation with just the priorities 1 and 2. For $(U, v) \in V$ and $a = \mathsf{code}(R, \sigma, j) \in \Sigma$, we put

$$\mathsf{track}((U,v),a,1) = \begin{cases} \emptyset & \text{if } v = * \\ \{(\Gamma_j\sigma, \psi) \mid \psi \in (\mathsf{Tr}(v,a) \cap \mathsf{dfr}(\mathsf{FL})) \cup \{*\}\} & \text{if } v \neq * \end{cases}$$

$$\mathsf{track}((U,v),a,2) = \begin{cases} \emptyset & \text{if } v \neq * \\ \{(\Gamma_j\sigma, \psi) \mid \psi \in \mathsf{dfr}(\mathsf{FL})\} & \text{if } v = * \end{cases}$$

The labelling function $l : V \to \mathcal{P}(\mathsf{FL}(\psi_0))$ maps (U, v) to U. The initial focused node is just $(\{\psi_0\}, *)$.

Corollary 5.2.57. *If the employed set of one-step rules is* ExpTime*-tractable, then instantiation of Algorithm 5.2.19 to satisfiability games via focusing decides the satisfiability problem of the alternation-free and aconjunctive coalgebraic μ-calculus in* ExpTime*.*

We note that while the satisfiability games via focusing are asymptotically smaller than the satisfiability games via determinization even for the linear μ-calculus, due to the way in which deferrals are finished serially in the timed-out tableau construction, the resulting models are asymptotically larger than the models resulting from the satisfiability games via determinization for alternation-free formulas. To decide just the satisfiability of aconjunctive and alternation-free formulas, satisfiability games via focusing should be used; if concrete models for satisfiable formulas (or refutations for unsatisfiable formulas) are also desired, then satisfiability games via determinization should be used.

6 Conclusion

We have given a detailed discourse on the method of constructing different kinds of satisfiability games for the coalgebraic μ-calculus with the help of tracking automata. We have also presented novel procedures that determinize so-called limit-deterministic and limit-linear automata on infinite words. These procedures lead to asymptotically smaller determinized automata than the Safra/Piterman construction (that determinizes unrestricted Büchi automata) and the Miyano/Hayashi construction (that determinizes unrestricted Co-Büchi automata), respectively. Since the presented satisfiability games via determinization are constructed over the carrier sets of determinized automata, the new determinization methods lead to asymptotically smaller satisfiability games for formulas whose tracking automata are limit-deterministic or limit-linear. We have also presented a generic global caching algorithm that solves the introduced satisfiability games on-the-fly. A prototypical implementation of this algorithm shows promising initial results [25,24].

References

1. Rajeev Alur, Thomas Henzinger, and Orna Kupferman. Alternating-time temporal logic. *Journal of the ACM*, 49:672–713, 2002.
2. Patrick Blackburn, Maarten de Rijke, and Yde Venema. *Modal Logic*. CUP, 2001.
3. Julian C. Bradfield. The modal μ-calculus alternation hierarchy is strict. *Theoretical Computer Science*, 195(2):133–153, 1998.
4. Florian Bruse, Michael Falk, and Martin Lange. The fixpoint-iteration algorithm for parity games. In *Games, Automata, Logics and Formal Verification, GandALF 2014*, volume 161 of *EPTCS*, pages 116–130. Open Publishing Association, 2014.
5. Cristian Calude, Sanjay Jain, Bakhadyr Khoussainov, Wei Li, and Frank Stephan. Deciding parity games in quasipolynomial time. In *Theory of Computing, STOC 2017*, pages 252–263. ACM, 2017.
6. Brian Chellas. *Modal Logic*. Cambridge University Press, 1980.
7. Corina Cîrstea, Clemens Kupke, and Dirk Pattinson. EXPTIME tableaux for the coalgebraic μ-calculus. In *Computer Science Logic, CSL 2009*, volume 5771 of *LNCS*, pages 179–193. Springer, 2009.
8. Corina Cîrstea and Dirk Pattinson. Modular construction of modal logics. *Theoretical Computer Science*, 388(1-3):83–108, 2007.
9. Costas Courcoubetis and Mihalis Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, 1995.
10. Allen Emerson and Joseph Halpern. Decision procedures and expressiveness in the temporal logic of branching time. *Journal of Computer and System Sciences*, 30:1–24, 1985.
11. Allen Emerson and Charanjit Jutla. The complexity of tree automata and logics of programs. *SIAM Journal on Computing*, 29(1):132–158, 1999.
12. Javier Esparza, Jan Kretínský, Jean-François Raskin, and Salomon Sickert. From LTL and limit-deterministic Büchi automata to deterministic parity automata. In *Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2017*, volume 10205 of *LNCS*, pages 426–442. Springer, 2017.
13. John Fearnley, Sanjay Jain, Sven Schewe, Frank Stephan, and Dominik Wojtczak. An ordered approach to solving parity games in quasi polynomial time and quasi linear space. In *SPIN Symposium on Model Checking of Software, 2017*, pages 112–121. ACM, 2017.
14. Oliver Friedmann and Martin Lange. Deciding the unguarded modal μ-calculus. *Journal of Applied Non-Classical Logics*, 23:353–371, 2013.
15. Rajeev Goré. And-Or tableaux for fixpoint logics with converse: LTL, CTL, PDL and CPDL. In *Automated Reasoning, IJCAR 2014*, volume 8562 of *LNCS*, pages 26–45. Springer, 2014.
16. Rajeev Goré, Clemens Kupke, and Dirk Pattinson. Optimal tableau algorithms for coalgebraic logics. In *Tools and Algorithms for the Construction and Analysis Of Systems, TACAS 2010*, volume 6015 of *LNCS*, pages 114–128. Springer, 2010.
17. Rajeev Goré, Clemens Kupke, Dirk Pattinson, and Lutz Schröder. Global caching for coalgebraic description logics. In *Automated Reasoning, IJCAR 2010*, volume 6173 of *LNCS*, pages 46–60. Springer, 2010.
18. Rajeev Goré and Linh Anh Nguyen. EXPTIME tableaux for ALC using sound global caching. *Journal of Automated Reasoning*, 50:355–381, 2013.
19. Rajeev Goré and Florian Widmann. An optimal on-the-fly tableau-based decision procedure for PDL-satisfiability. In *Automated Deduction, CADE 2009*, volume 5663 of *LNCS*, pages 437–452. Springer, 2009.
20. Rajeev Goré and Florian Widmann. Sound global state caching for ALC with inverse roles. In *Automated Reasoning with Analytic Tableaux and Related Methods, TABLEAUX 2009*, volume 5607 of *LNCS*, pages 205–219. Springer, 2009.
21. Erich Grädel, Wolfgang Thomas, and Thomas Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *LNCS*. Springer, 2002.

22. Helle Hansen and Clemens Kupke. A coalgebraic perspective on monotone modal logic. In *Coalgebraic Methods in Computer Science, CMCS 2004*, volume 106 of *ENTCS*, pages 121–143. Elsevier, 2004.
23. Daniel Hausmann and Lutz Schröder. Global caching for the flat coalgebraic μ-calculus. In *Temporal Representation and Reasoning, TIME 2015*, pages 121–143. IEEE Computer Society, 2015.
24. Daniel Hausmann, Lutz Schröder, and Hans-Peter Deifel. Permutation games for the weakly aconjunctive μ-calculus. In *Tools and Algorithms for the Construction and Analysis of Systems, TACAS 2018*, volume 10806 of *LNCS*, pages 361–378. Springer, 2018.
25. Daniel Hausmann, Lutz Schröder, and Christoph Egger. Global caching for the alternation-free coalgebraic μ-calculus. In *Concurrency Theory, CONCUR 2016*, volume 59 of *LIPIcs*, pages 34:1–34:15. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2016.
26. Aviad Heifetz and Philippe Mongin. Probability logic for type spaces. *Games and Economic Behavior*, 35(1):31–53, 2001.
27. Valerie King, Orna Kupferman, and Moshe Vardi. On the complexity of parity word automata. In *Foundations of Software Science and Computation Structures, FoSSaCS 2001*, volume 2030 of *LNCS*, pages 276–286. Springer, 2001.
28. Dexter Kozen. Results on the propositional μ-calculus. *Theoretical Computer Science*, 27:333–354, 1983.
29. Dexter Kozen. A finite model theorem for the propositional μ-calculus. *Studia Logica*, 47:233–241, 1988.
30. Clemens Kupke and Yde Venema. Coalgebraic automata theory: Basic results. *Logical Methods in Computer Science*, 4(4:10):1–43, 2008.
31. Martin Lange and Colin Stirling. Focus games for satisfiability and completeness of temporal logic. In *Logic in Computer Science, LICS 2001*, pages 357–365. IEEE Computer Society, 2001.
32. Kim Larsen and Arne Skou. Bisimulation through probabilistic testing. *Information and Computation*, 94(1):1–28, September 1991.
33. Saunders MacLane. *Categories for the Working Mathematician*. Springer, 1971.
34. Satoru Miyano and Takeshi Hayashi. Alternating finite automata on ω-words. *Theoretical Computer Science*, 32:321–330, 1984.
35. Lawrence Moss. Coalgebraic logic. *Annals of Pure and Applied Logic*, 96(1):277–317, 1999.
36. Damian Niwinski. On fixed-point clones. In *Automata, Languages and Programming, ICALP 1986*, volume 226 of *LNCS*, pages 464–473. Springer, 1986.
37. Rohit Parikh. Propositional game logic. In *Foundations of Computer Science, FOCS 1983*, pages 195–200. IEEE Computer Society, 1983.
38. Dirk Pattinson. Expressive logics for coalgebras via terminal sequence induction. *Notre Dame Journal of Formal Logic*, 45:19–33, 2004.
39. Dirk Pattinson and Lutz Schröder. Admissibility of cut in coalgebraic logics. In *Coalgebraic Methods in Computer Science, CMCS 2008*, volume 203 of *ENTCS*, pages 221–241. Elsevier, 2008.
40. Dirk Pattinson and Lutz Schröder. Generic modal cut elimination applied to conditional logics. In *Automated Reasoning with Analytic Tableaux and Related Methods, TABLEAUX 2009*, volume 5607 of *LNCS*, pages 280–294. Springer, 2009.
41. Marc Pauly. A modal logic for coalitional power in games. *Journal of Logic and Computation*, 12:149–166, 2002.
42. Nir Piterman. From nondeterministic Büchi and Streett automata to deterministic parity automata. *Logical Methods in Computer Science*, 3(3):255–264, 2007.
43. Jan Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000.
44. Shmuel Safra. On the complexity of ω-automata. In *Foundations of Computer Science, FOCS 1988*, pages 319–327. IEEE Computer Society, 1988.
45. Sven Schewe. *Synthesis of distributed systems*. PhD thesis, Universität des Saarlands, 2008.
46. Lutz Schröder. Expressivity of coalgebraic modal logic: The limits and beyond. *Theoretical Computer Science*, 390(2):230–247, 2008.

47. Lutz Schröder and Dirk Pattinson. Modular algorithms for heterogeneous modal logics. In *Automata, Languages and Programming, ICALP 07*, volume 4596 of *LNCS*, pages 459–471. Springer, 2007.
48. Lutz Schröder and Dirk Pattinson. How many toes do I have? Parthood and number restrictions in description logics. In *Principles of Knowledge Representation and Reasoning, KR 2008*, pages 307–317. AAAI Press, 2008.
49. Lutz Schröder and Dirk Pattinson. Shallow models for non-iterative modal logics. In *Advances in Artificial Intelligence, KI 2008*, volume 5243 of *LNAI*, pages 324–331. Springer, 2008.
50. Lutz Schröder and Dirk Pattinson. PSPACE bounds for rank-1 modal logics. *ACM Transactions on Computational Logic*, 10(2):13:1–13:33, 2009.
51. Lutz Schröder and Dirk Pattinson. Strong completeness of coalgebraic modal logics. In *Theoretical Aspects of Computer Science, STACS 09*, volume 3 of *LIPIcs*, pages 673–684. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, 2009.
52. Lutz Schröder, Dirk Pattinson, and Daniel Hausmann. Optimal tableaux for conditional logics with cautious monotonicity. In *Artificial Intelligence, ECAI 2010*, volume 215 of *FAIA*, pages 707–712. IOS Press, 2010.
53. Lutz Schröder and Yde Venema. Completeness of flat coalgebraic fixpoint logics. *ACM Transactions on Computational Logic*, 19(1):4:1–4:34, 2018.
54. Alfred Tarski. A lattice-theoretical fixpoint theorem and its applications. *Pacific Journal of Mathematics*, 5(2):285–309, 1955.
55. Yde Venema. Automata and fixed point logics for coalgebras. In *Coalgebraic Methods in Computer Science, CMCS 2004*, volume 106 of *ENTCS*, pages 355–375. Elsevier, 2004.