19 Automata for Guarded Fixed Point Logics

Dietmar Berwanger and Achim Blumensath

Mathematische Grundlagen der Informatik
RWTH Aachen

19.1 Introduction

The guarded fixed point logics μGF and μCGF introduced in the previous chapter extend the guarded fragments of first-order logic GF and CGF on the one hand and the modal μ-calculus on the other hand. Thereby, the expressive power of the underlying formalisms is increased considerably. On transition systems, for instance, μGF already subsumes the μ-calculus with backwards modalities. Hence, the question arises, whether these logics are still manageable algorithmically. In this chapter we will study the complexity of their satisfiability problems.

As a consequence of the general criterion stated in Theorem 18.23, it follows that the satisfiability problems for μGF and μCGF are decidable. Yet, the argument does not allow us to derive precise complexity bounds for the decision problem. A lower bound can be obtained from the respective results for L_μ and GF. For L_μ the satisfiability problem is EXPTIME-complete [2], whereas for GF it is complete for 2EXPTIME [5]. However, if we consider formulae of bounded width, i.e., with a bounded number of variables, it becomes EXPTIME-complete as well.

Following Grädel and Walukiewicz [6,5] we will prove that even for μCGF, the strongest logic considered, the satisfiability problem is still in 2EXPTIME in the general case, and in EXPTIME for formulae of bounded width. In other words, the fixed point extensions of guarded logics are almost for free in terms of complexity of the satisfiability problem.

Given the expressive power of these logics, this result is rather surprising. For instance, in contrast to L_μ, already the weakest guarded fixed point logic μGF lacks the finite model property. An example of a formula with only infinite models was given in the previous chapter:

$$(\exists xy.Exy) \land (\forall xy.Exy)(\exists z.Eyz)[\mathrm{LFP}_{Z,z}(\forall y.Eyz)Zy](z).$$

A crucial model theoretic aspect of guarded logics is their (generalised) tree model property stated in Theorem 18.16. Informally, this asserts that models of guarded formulae can be represented as trees. In [8,4] Vardi and Grädel emphasise that the tree model property seems to be the key feature responsible for the good algorithmic behavior of modal logics because it makes them amenable to automata-theoretic techniques for solving satisfiability and model-checking problems. The generalised tree model property allows us to lift these techniques to guarded logics. In order to decide whether a given formula ψ is satisfiable one can construct two automata: the first one, called *model checking automaton*,

E. Grädel, W. Thomas, and T. Wilke (Eds.): Automata, Logics, and Infinite Games, LNCS 2500, pp. 343-355, 2002

takes an appropriate representation of a structure as input and accepts if and only if the structure satisfies ψ; the other automaton recognises the set of all appropriate representations. Then, the formula is satisfiable iff the product of these automata recognises a non-empty language.

This scheme outlines the plan of the present chapter. First, we introduce appropriate tree representations of structures in Section 19.2 together with a suitable automata model. For a better understanding we will proceed on two tracks. On the one hand, we define the *unravelling tree* of a structure. The nodes of this tree are associated to the guarded sets of the structure, in such a way that every node has all guarded sets represented in its successors. This rich encoding allows checking of the encoded model by rather simple automata. Moreover, the underlying technique was already discussed in the previous chapter, in the proof of Theorem 18.17. On the other hand, we introduce *decomposition trees* which, being more compact representations, require more sophisticated two-way automata for model checking.

Section 19.3 is dedicated to the construction of the model checking automaton. Starting from input structures encoded as unravelling trees, we define a one-way automaton which, when viewed as a two-way automaton, still recognises the same structures, but in a different encoding, as decomposition trees. At that point, the two tracks of our exposition converge.

Finally, Section 19.4 concludes the reduction by presenting an automaton which recognises valid decomposition trees. Putting all pieces together we are able to derive the desired complexity bounds for the satisfiability problem.

19.2 Requisites

19.2.1 Clique guarded fixed point formulae

When speaking of formulae we always mean μCGF-formulae as introduced in the previous chapter. To simplify our notation we will, however, omit the clique guards, i.e., instead of $(\exists \bar{x}.\mathrm{clique}(\bar{x}))\eta(\bar{x})$ we will write $\exists \bar{x}.\eta(\bar{x})$ and accordingly for universal formulae.

Furthermore, we will assume that all formulae are *well named* and in *negation normal form*, that is, fixed point variables are defined at most once and negation applies to atomic formulae only. Clearly, every μCGF-formula can be rewritten to meet these requirements.

A crucial parameter of a formula is its **width** which is defined as the greatest number of free variables occurring in a subformula. Equivalently, a formula has width k iff it can be transformed, by renaming of variables, so that it uses only k variables. In the following we will always assume that every formula of width k is written with the variables $\{x_0, x_1, \dots, x_{k-1}\}$.

19.2.2 Tree representations

In order to use tree automata for model checking and satisfiability we encode structures by trees. Recall that every subformula of a formula of width k can

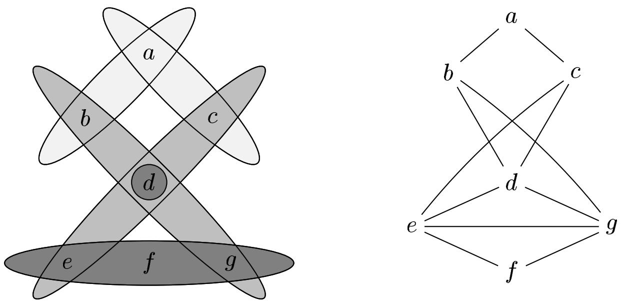

Fig. 19.1. A structure with relations of arity 1, 2, and 3 and its Gaifman graph

refer to at most k structure elements at the same time, which, moreover, have to be guarded. On account of this, we associate to a given structure $\mathfrak{A}$ a tree whose nodes are labelled by the substructures of $\mathfrak{A}$ induced by at most k guarded elements. In addition, the overlap of two adjacent nodes is stored in the label of their common edge.

Let us fix some notation for the remainder of this chapter. The set of guarded subsets of size at most k of a σ-structure $\mathfrak{A}$ is denoted by

$$\Gamma_k(\mathfrak{A}) := \{ K \subseteq A \mid K \text{ is } k\text{-clique-guarded in } \mathfrak{A} \}.$$

The substructures induced by these sets are mapped onto the fixed universe $[k] = \{0, \dots, k-1\}$ and then arranged to form a tree while keeping track of overlaps along the edges. Thus, the nodes of the resulting trees are labelled by the alphabet

$$\Sigma_k := \{ \mathfrak{C} \mid \mathfrak{C} \text{ is a } \sigma\text{-structure over a universe } C \subseteq [k] \}$$

while the edges are labelled by subsets of $[k]$. We call trees labelled by these alphabets shortly ***k*-type trees**. When we speak about a D-edge, we mean an edge labelled with $D \subseteq [k]$, and a D-neighbour or D-successor of a node is a neighbour respectively a successor along some D-edge.

Definition 19.1. For a given width k, the ***k*-unravelling tree** of a structure $\mathfrak{A}$ is the k-type tree T over the set of nodes $\Gamma_k(\mathfrak{A})^*$ labelled as follows:

(i) The root of T is labelled with the empty structure $(\emptyset, \sigma)$ and all outgoing edges are labelled with $\emptyset$.
(ii) Every node $v \in \Gamma_k(\mathfrak{A})^* K$ is labelled with an isomorphic copy $\mathfrak{C}$ of $\mathfrak{A}|_K$, the restriction of $\mathfrak{A}$ to $K \in \Gamma_k(\mathfrak{A})$.
(iii) If $\pi : \mathfrak{A}|_K \to \mathfrak{C}$ and $\pi' : \mathfrak{A}|'_K \to \mathfrak{C}'$ are isomorphisms labelling, respectively, a node $v \in \Gamma_k(\mathfrak{A})^* K$ and its successor $v' = vK'$, then π and π' agree on $K \cap K'$ and the edge (v, v') is labelled with $\pi(K \cap K')$.

Remark 19.2. It is easy to see that for every D-edge (v, v') of an unravelling tree T the following conditions hold:

(i) **Consistency:** the labels $\mathfrak{C}$ of v and $\mathfrak{C}'$ of v' agree on D, that is, $\mathfrak{C}|_D = \mathfrak{C}'|_D$.
(ii) **Completeness:** for any $H \subseteq [k]$ the H-successors of v and v' agree on $D \cap H$, i.e., there is a one-to-one map assigning to each H-successor w of v an H-successor w' of v' such that the labels of w and w' agree on $D \cap H$.

Generally, we call a k-type tree **consistent**, if it satisfies the first condition. Let us now look at the relationship between a tree representation and the encoded structure.

Definition 19.3. Given a consistent k-type tree T, consider the disjoint sum of its node labels,

$$\mathfrak{D} := \dot{\bigcup}\{ (\mathfrak{C}, v) \mid \mathfrak{C} \text{ is the label of } v \in T \}.$$

Let $\sim$ be the least equivalence relation on the universe of $\mathfrak{D}$ with

$$(i, v) \sim (i, v') \quad \text{if } v' \text{ is a successor of } v \text{ and } i \text{ is in the label of } (v, v').$$

Then, by consistency of T, $\sim$ is a congruence relation on $\mathfrak{D}$. We call the quotient $\mathfrak{D}/_\sim$ the structure **recovered** from T.

Definition 19.4. The ***k*-unravelling** $\mathfrak{A}^{(k)}$ of a structure $\mathfrak{A}$ is the structure recovered from the k-unravelling tree of $\mathfrak{A}$.

Since μCGF is invariant under guarded bisimulation (see [5]), it follows that sentences of width up to k cannot distinguish between a structure $\mathfrak{A}$ and its k-unravelling $\mathfrak{A}^{(k)}$.

Proposition 19.5. *Every structure $\mathfrak{A}$ is k-clique bisimilar to its k-unravelling $\mathfrak{A}^k$. That is, for all μCGF-sentences ψ of width at most k we have*

$$\mathfrak{A} \models \psi \quad \textit{iff} \quad \mathfrak{A}^{(k)} \models \psi.$$

If we recall the notion of tree decomposition of a structure introduced in the previous chapter we can easily establish the following connection.

Proposition 19.6. *A k-type tree T is a tree decomposition of some structure $\mathfrak{A}$ iff the structure recovered from T is isomorphic to $\mathfrak{A}$.*

This relationship suggests tree decompositions as candidates for structure representations.

Definition 19.7. For a given width k, a ***k*-decomposition tree** of a structure $\mathfrak{A}$ is a k-type tree T where

(i) for every $K \in \Gamma_k(\mathfrak{A})$ there is a node labelled with (an copy of) $\mathfrak{A}|_K$;
(ii) the labels of any two nodes connected by a D-edge agree on D;

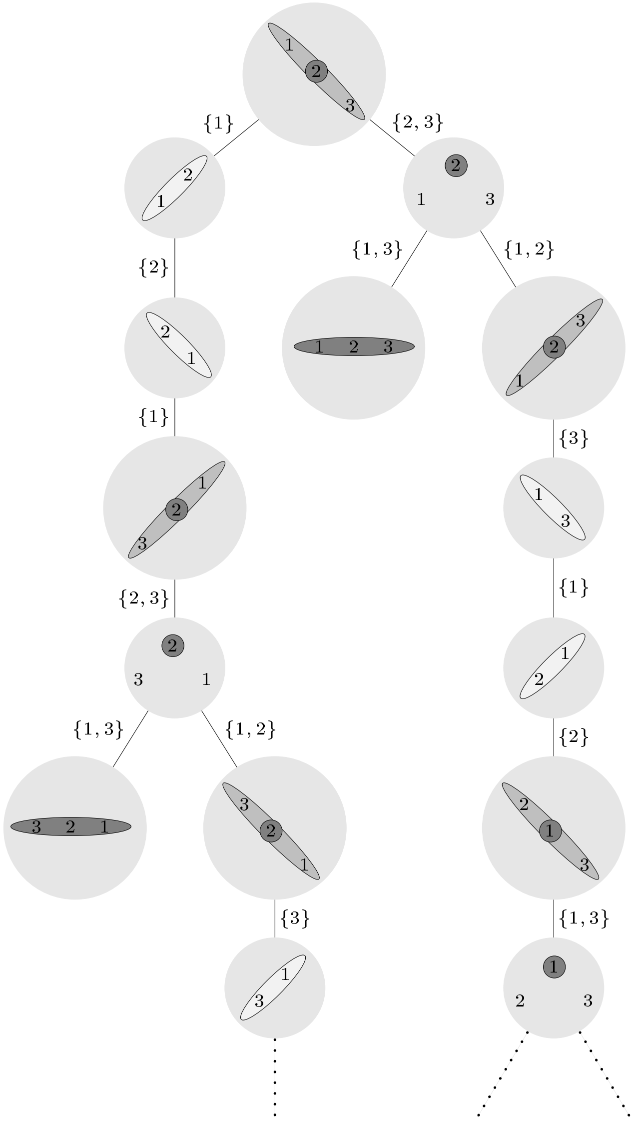

Fig. 19.2. A k-decomposition tree of the structure in Fig. 19.2.2

(iii) every node v is labelled with $\mathfrak{A}|_K$ for some $K \in \Gamma_k(\mathfrak{A})$ via an isomorphism π. Moreover, for each $K' \in \Gamma_k(\mathfrak{A})$ there is a node v' labelled with $\mathfrak{A}|_{K'}$, such that all edges on the path between v and v' include $\pi(K \cap K')$ in their labels.

Remark 19.8. (i) The k-unravelling tree of a structure is also a k-decomposition tree of that structure.

(ii) Each k-decomposition tree of a structure $\mathfrak{A}$ induces a subtree in the k-unravelling tree of $\mathfrak{A}$.

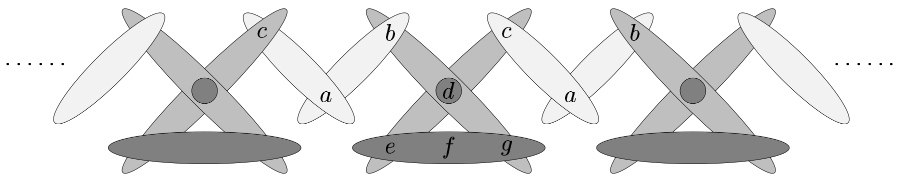

Fig. 19.3. The structure recovered from the decomposition tree in Fig. 19.2.2

It is an easy exercise to show that the process of k-decomposing a structure preserves its properties up to bisimulation, yielding a more compact representation than unravelling does.

Proposition 19.9. *Given a structure $\mathfrak{A}$, let $\mathfrak{A}'$ be the structure recovered from a k-decomposition tree of $\mathfrak{A}$. Then $\mathfrak{A}$ and $\mathfrak{A}'$ are clique-k-bisimilar.*

19.2.3 The automata model

We employ alternating automata that work on trees where nodes and edges are labelled.

Definition 19.10. An **alternating tree automaton** over a node alphabet Σ and an edge alphabet Δ is given by a tuple

$$\mathcal{A} = (Q, \Sigma, \Delta, \delta, q_{\mathrm{I}}, \Omega)$$

where $Q = Q_0 \cup Q_1$ is the set of existential and universal states, q_{I} designates the initial state, $\Omega : Q \to \omega$ is a parity condition and

$$\delta : Q \times \Sigma \to \mathscr{P}(\Delta \times Q)$$

is the transition function. The pairs $(d, q) \in \Delta \times Q$ are called transitions.

We define the behaviour of such automata by way of games.

Definition 19.11. Let $\mathcal{A} = (Q, \Sigma, \Delta, \delta, q_{\mathrm{I}}, \Omega)$ be an automaton and T an appropriately labelled tree. The game $\mathcal{G}(\mathcal{A}, T)$ associated to $\mathcal{A}$ and T is the parity game with positions $Q \times T$ and acceptance condition Ω played as follows.

Every play starts in state q_{I} at the root of T. Assume that the play reached some position (q, v) where the node v is labelled with c. If q belongs to Q_0, Player 0 can move to a position (q', v') if

(i) there is a transition $(d, q') \in \delta(q, c)$ and
(ii) v' is a d-successor of v.

The moves of Player 1 are defined analogously.

The language $L(\mathcal{A})$ accepted by a tree automaton $\mathcal{A}$ is the set of all trees T, such that Player 0 has a winning strategy in the game $\mathcal{G}(\mathcal{A}, T)$.

Usually, automata are defined as devices scanning their input only in one direction. However, for our purpose it is convenient to allow them to move backwards and remain still as well.

Definition 19.12. An **alternating two-way tree automaton** is given in the same way as a (one-way) alternating automaton,

$$\mathcal{A}^2 = (Q, \Sigma, \Delta, \delta, q_{\mathrm{I}}, \Omega)$$

where acceptance is defined in a different way. The game $\mathcal{G}(\mathcal{A}^2, T)$ associated to a two-way automaton $\mathcal{A}^2$ and a tree T is the parity game obtained as in Definition 19.11, but replacing rule (ii) with

(ii') either $v' = v$ or v' is a d-neighbour of v.

The language $L(\mathcal{A}^2)$ accepted by a two-way tree automaton $\mathcal{A}^2$ is the set of all trees T such that Player 0 has a winning strategy in the game $\mathcal{G}(\mathcal{A}^2, T)$.

19.3 Model Checking

The results presented in Chapter 14 and 10 reveal a close relationship between alternating automata and games on the one side, and logical formalisms on the other side. The automaton constructed in Section 10.3 for L_μ translates first-order operations into state transitions, while fixed point predicates are encoded as priorities.

In a similar way, we will construct automata for μCGF. But unlike L_μ, where a formula is evaluated at a single node of a transition system, a μCGF-formula with several free variables may involve several structure elements. Since these elements have to be clique-guarded, they appear together in the label of some node in the unravelling (or, decomposition) tree. To allow our automaton to access the structure in the node labels of the input tree, its states will contain two components: a subformula, and an assignment of the variables appearing free therein.

The *closure* $\mathrm{cl}(\psi)$ of a formula ψ is the set consisting of all subformulae of ψ together with the formulae true and false.

Definition 19.13. To any formula $\psi \in \mu$CGF of width k we associate the automaton $\mathcal{A}_\psi = (Q, \Sigma_k, \mathscr{P}([k]), \delta, q_{\mathrm{I}}, \Omega)$ over k-type trees where the state set

$$Q := \{\, (\varphi, \beta) \mid \varphi \in \mathrm{cl}(\psi) \text{ and } \beta : \{x_0 \dots x_{k-1}\} \to [k] \,\}$$

is partitioned into existential and universal states by

$$Q_0 := \{\, (\varphi, \beta) \mid \varphi = \text{false}, \text{ or } \varphi = \eta \vee \vartheta, \text{ or } \varphi = \exists \bar{y}.\eta \,\}, \text{ and}$$
$$Q_1 := Q \setminus Q_0.$$

The initial state is $q_{\mathrm{I}} = (\psi, \emptyset)$ where $\emptyset$ stands for the void assignment.

It remains to specify the transition function. To simplify our notation we use expressions $\langle H \rangle S$ with $H \subseteq [k]$ and $S \subseteq Q$ to denote the set of transitions $\{ D \subseteq [k] \mid H \subseteq D \} \times S$. In particular, when we refer to the universe of $\mathfrak{C}$ we write $\langle \cdot \rangle S$ instead of $\langle C \rangle S$. Furthermore, omitting parenthesis, we simply write $\delta(\varphi, \beta, \mathfrak{C})$ instead of $\delta((\varphi, \beta), \mathfrak{C})$.

(i) If $\varphi = \text{true}$ or $\varphi = \text{false}$ then $\delta(\varphi, \beta, \mathfrak{C}) = \emptyset$.

(ii) If φ is a σ-atom or a negated σ-atom then

$$\delta(\varphi, \beta, \mathfrak{C}) = \begin{cases} \langle \cdot \rangle \{(\text{true}, \emptyset)\} & \text{if } \mathfrak{C}, \beta \models \varphi, \\ \langle \cdot \rangle \{(\text{false}, \emptyset)\} & \text{if } \mathfrak{C}, \beta \not\models \varphi. \end{cases}$$

(iii) If $\varphi = \eta \wedge \vartheta$ or $\varphi = \eta \vee \vartheta$ then

$$\delta(\varphi, \beta, \mathfrak{C}) = \langle \cdot \rangle \{(\eta, \beta), (\vartheta, \beta)\}.$$

(iv) If $\varphi(\bar{x}) = \mathrm{FP}_{T\bar{y}}(\eta)(\bar{x})$ then

$$\delta(\varphi, \beta, \mathfrak{C}) = \langle \cdot \rangle \{(\eta, \beta)\}.$$

(v) If $\varphi(\bar{x}) = T\bar{x}$ and $\mathrm{FP}_{T\bar{y}}(\eta)(\bar{x})$ is the unique definition of T in ψ then

$$\delta(\varphi, \beta, \mathfrak{C}) = \langle \cdot \rangle \{(\eta, \beta)\}.$$

(vi) If $\varphi(\bar{x}) = \exists \bar{y}.\eta(\bar{x}, \bar{y})$ or $\varphi(x) = \forall \bar{y}.\eta(\bar{x}, \bar{y})$ then

$$\delta(\varphi, \beta, \mathfrak{C}) = \langle \cdot \rangle \{ (\eta, \beta') \mid \beta'|_{\bar{x}} = \beta|_{\bar{x}} \} \cup \langle \beta(\bar{x}) \rangle \{(\varphi, \beta)\}.$$

Finally, if the fixed point variables of ψ occur in the order $Z_1, \dots, Z_n$ the parity condition is given by

$$\Omega(\varphi, \beta) := \begin{cases} 2i & \varphi = Z_i\bar{x} \text{ and } Z_i \text{ is a GFP-variable,} \\ 2i+1 & \varphi = Z_i\bar{x} \text{ and } Z_i \text{ is an LFP-variable,} \\ 2n+4 & \varphi = \forall \bar{y}.\eta, \\ 2n+3 & \varphi = \exists \bar{y}.\eta, \\ 2n+2 & \text{otherwise.} \end{cases}$$

The automaton works in a similar way as the L_μ-automata defined in Section 10.3: disjunctions are decomposed by Player 0, conjunctions by Player 1 and fixed points are regenerated. Atomic statements are verified locally and terminate the run. Acceptance of infinite runs is determined by the priority function which reflects the nesting and type of fixed point definitions. Note that, except when dealing with quantifiers, the automaton changes only the formula component of its states, while the variable assignment remains the same. Moreover, the $\langle \cdot \rangle$-transitions allow to move only to successors that retain the structure information of the current node.

To understand the handling of quantification, consider, e.g., an existential formula $\varphi(\bar{x}) = \exists \bar{y}.\eta(\bar{x}, \bar{y})$. Player 0 may use a transition from $\langle\beta(\bar{x})\rangle\{(\varphi, \beta)\}$ to proceed to a successor that retains the structure living on elements currently assigned to the free variables $\bar{x}$. In this way, he can reassign the quantified variables $\bar{y}$ to elements of the chosen successor. After such a move, the formula in the new state is still φ and Player 0 is again in turn to move. But, as existential formulae have odd priority, he can reiterate these moves only finitely many times and must then take a transition of the form $\langle\cdot\rangle\{\, (\eta, \beta') \mid \beta'|_{\bar{x}} = \beta|_{\bar{x}} \,\}$.

Given an input tree that k-unravels a structure $\mathfrak{A}$, the structures labelling the nodes are all induced by k-cliques in $\mathfrak{A}$. Moreover, from each node (a copy of) every other k-clique of $\mathfrak{A}$ is accessible within one move.

It remains to prove that our construction is correct, that is, that we can use the automaton defined above to solve the model checking problem for μCGF.

Proposition 19.14. *Given a formula ψ of width k and a structure $\mathfrak{A}$, the automaton $\mathcal{A}_\psi$ accepts the k-unravelling tree of $\mathfrak{A}$ iff $\mathfrak{A} \models \psi$.*

Proof. It is convenient to argue in terms of games. Model checking games for μCGF were introduced in [1] as a generalisation of the model checking games for L_μ. Although defined for finite structures, the extension of these games to the transfinite case is straightforward.

Let T be the k-unravelling tree of the structure $\mathfrak{A}$. We will show that the game which determines acceptance of T by the automaton $\mathcal{A}_\psi$ is essentially the model checking game associated to $\mathfrak{A}$ and ψ.

Let $\mathcal{G}$ be the acceptance game $\mathcal{G}(\mathcal{A}_\psi, T)$. We can simplify this game by collapsing positions which share the same formula and map its free variables to the same part of the structure.

Recall that any node v of T is labelled via some isomorphism π. Furthermore, at every position (φ, β, v) in a play of $\mathcal{G}$, the image of π includes the image of the assignment β. Thus, we can define a mapping from the positions of $\mathcal{G}$ to $\{\, (\varphi, \chi) \mid \varphi \in \mathrm{cl}(\psi) \text{ and } \chi : \{x_0 \dots x_{k-1}\} \to A \,\}$ as follows:

$$\widehat{\cdot} : (\varphi, \beta, v) \mapsto (\varphi, \pi^{-1} \circ \beta).$$

By the construction of $\mathcal{G}$, we can easily verify that this mapping induces a congruence relation $\widehat{\sim}$ among the positions of $\mathcal{G}$,

$$(\varphi, \beta, v) \mathrel{\widehat{\sim}} (\varphi, \beta, v') \quad \text{iff} \quad \widehat{(\varphi, \beta, v)} = \widehat{(\varphi, \beta, v')},$$

which is also a bisimulation on $\mathcal{G}$.

Consider now the (strong homomorphic) image $\widehat{\mathcal{G}}$ of $\mathcal{G}$ under $\widehat{\cdot}$. On the one hand, $\mathcal{G}$ and $\widehat{\mathcal{G}}$ are bisimilar via $\widehat{\cdot}$ and, consequently, the same player has a winning strategy in both plays. On the other hand, $\widehat{\mathcal{G}}$ is almost the model checking game $\mathcal{G}' = \mathcal{G}(\mathfrak{A}, \psi)$ as defined in [1]. The only difference arises at positions (φ, χ) where φ is an existential or universal formula, say $\varphi = \exists \bar{y} \eta(\bar{x}, \bar{y})$. Then, the model checking game allows moves to (η, χ') with χ' such that

(i) χ and χ' agree on the values of $\bar{x}$ and

(ii) $\mathfrak{A}, \chi' \models \text{clique}(\bar{x}, \bar{y})$,

whereas in $\widehat{\mathcal{G}}$ the legal moves go either to (φ, χ') with χ' as above, or to (η, χ).

Nevertheless, we will show that the same player wins both $\mathcal{G}'$ and $\widehat{\mathcal{G}}$. If Player 0 has a winning strategy in the model checking game $\mathcal{G}'$, he can also play this strategy in $\widehat{\mathcal{G}}$, as long as no existential formula is met. Otherwise, at positions (φ, χ) as above, he can imitate the move to the position (η, χ') he would perform in $\mathcal{G}'$ by taking two steps:

(i) move to (φ, χ'); this is possible since, for every χ' agreeing with χ on the free variables of φ, the position (φ, χ') is reachable from (φ, χ) in one step.
(ii) At (φ, χ') it's still Player 0 turn: move to (η, χ').

Towards a contradiction, let us assume that Player 1 wins this play. Then, after any universal formula $\varphi = \forall \bar{y} \eta(\bar{x}, \bar{y})$ occurring in the play, there can follow only finitely many positions with φ until Player 1 chooses some position (η, χ'); otherwise he would lose with the highest even priority. But then, Player 1 also wins by choosing (φ, χ') right from position (φ, χ) and proceeding with (η, χ'). However, these two moves translate into one move in the corresponding play of $\mathcal{G}'$ which leads Player 1 to a win in $\mathcal{G}$ despite Player 0's winning strategy, which is not possible. This concludes our proof that a player has a winning strategy in the model checking game iff he has one in the acceptance game.

The correctness of our construction relies on the fact that the input trees are complete in the sense of Remark 19.2 (ii). That is, if the current node is labelled by a k-clique of the represented structure, then every other k-clique appears in the label of some successor node. Unfortunately, it is very hard to check whether a given tree satisfies this property. By letting $\mathcal{A}_\psi$ run as a two-way automaton $\mathcal{A}^2_\psi$, we can relax this requirement and claim instead that every k-clique shall be reachable via a finite path from the current node.

Proposition 19.15. *Given a formula ψ of width k and a structure $\mathfrak{A}$, let T be a k-decomposition tree of $\mathfrak{A}$. Then the automaton $\mathcal{A}^2_\psi$ accepts T iff $\mathfrak{A} \models \psi$.*

Proof. The idea is to show that $\mathcal{A}^2_\psi$ runs on T in a similar way as its one-way variant does on the k-unravelling tree T' of $\mathfrak{A}$. Towards this we will transform the acceptance game $\mathcal{G}(\mathcal{A}^2_\psi, T)$ by introducing shortcuts into a game which is bisimilar to the acceptance game $\mathcal{G}(\mathcal{A}, T')$ of the one-way automaton.

Let $\mathcal{G}^*$ be the least game extending $\mathcal{G} := \mathcal{G}(\mathcal{A}^2_\psi, T)$ by new transitions in such a way that, whenever there are two transitions

$$(\varphi, \beta, v) \to (\varphi, \beta, v') \to (\varphi, \beta, v'')$$

in $\mathcal{G}^*$, the shortcut $(\varphi, \beta, v) \to (\varphi, \beta, v'')$ is also a transition in $\mathcal{G}^*$.

Observe that the new transitions just shortcut a sequence of steps in the original game, all performed by the same player. To see that this does not change the winning partitions, assume, towards a contradiction, that Player 1 has a winning strategy for $\mathcal{G}^*$ while Player 0 has one for $\mathcal{G}$. All moves in $\mathcal{G}$ are still available in $\mathcal{G}^*$, so Player 0 can apply his winning strategy for $\mathcal{G}$ in the play π of $\mathcal{G}^*$ against the winning strategy of Player 1. Let us now look at the play in $\mathcal{G}$ in which both players move like in π except at positions (φ, β, v) where Player 1 used a shortcut to, say (φ, β, v'), for φ a universal formula. At that point, Player 1 can move step by step via finitely many positions (φ, β, w) along the path leading to the destination of the shortcut. From there, the play proceeds like in π. Clearly, Player 1 wins this play in $\mathcal{G}$ in contradiction to our assumption on Player 0's winning strategy.

The mapping $\widehat{\cdot}$ which was defined in the proof of Proposition 19.14 can be applied to the positions of $\mathcal{G}^*$. It induces a congruence relation on $\mathcal{G}^*$ and, as such, a bisimulation between $\mathcal{G}^*$ and its strong homomorphic image $\widehat{\mathcal{G}}^*$. This image is precisely the game $\widehat{\mathcal{G}}(\mathfrak{A}_\psi, T')$ which is bisimilar to $\mathcal{G}(\mathfrak{A}_\psi, T')$.

Accordingly, the automaton $\mathcal{A}^2_\psi$ accepts the k-decomposition tree T iff $\mathcal{A}_\psi$ accepts the k-unravelling tree T'.

19.4 Satisfiability

The model checking automata introduced above operate correctly on inputs which represent structures. But in order to solve the satisfiability problem this does not suffice. We need to make sure that all inputs which do not represent structures are rejected.

Checking representation validity. From a given a k-type tree T, we can recover a structure according to Definition 19.3, only if T is consistent, that is, if every node agrees with its D-neighbours on the D-part of its label.

Provided T is consistent, let $\mathfrak{A}$ be the recovered structure. Now T is a k-decomposition tree of $\mathfrak{A}$ iff every node label of T induces a clique in $\mathfrak{A}$. This is crucial, since the model-checking automaton assumes that all elements appearing in the label of its input represent clique-guarded elements of the structure. Another way to formulate this condition is: For every node v and every pair of elements $\{i, j\} \subseteq [k]$ in its label, there is a node v' in which i and j are guarded by an atom and all edges on the path between v and v' include $\{i, j\}$ in their labels.

Now, we build an automaton that checks the above two conditions.

Definition 19.16. For every width k, we construct a two-way automaton $\mathcal{A}^2_k = (Q, \Sigma_k, \mathscr{P}([k]), \delta, \text{check}, \Omega)$ over k-type trees whose set of states is partitioned into

$$Q_0 = \{\text{false}\} \cup [k]^2 \quad \text{and}$$
$$Q_1 = \{\text{true}, \text{check}\} \cup \{\, R\bar{a} \mid R \in \sigma,\ \bar{a} \subseteq [k]\,\}.$$

In state check the automaton allows Player 1 to move freely on the input tree to reach a node where either the consistency or the guardedness condition may be violated. At that event, state $R\bar{a}$ records the loss of the atom $R\bar{a}$ along an edge that preserves $\bar{a}$ while the states $(i,j) \in [k]^2$ indicate the search for witnesses to the guardedness of i and j. The transitions are as follows.

$$\begin{aligned}\delta(\text{check}, \mathfrak{C}) = \langle\emptyset\rangle\{\text{check}\} \cup \bigcup\{\, \langle\bar{a}\rangle\{R\bar{a}\} \mid \mathfrak{C} \models R\bar{a} \,\} \\ \cup \langle\cdot\rangle\{\, (i,j) \mid \mathfrak{C} \not\models \text{clique}(i,j) \,\}.\end{aligned}$$

At a node where the elements i and j are not guarded, Player 1 can challenge his opponent to find a node where $\{i,j\}$ appear guarded, along a path where these elements persist in the edge labels.

$$\delta((i,j), \mathfrak{C}) = \begin{cases} \langle\cdot\rangle\{\text{true}\} & \mathfrak{C} \models \text{clique}(i,j), \\ \langle\{i,j\}\rangle\{(i,j)\} & \text{otherwise.} \end{cases}$$

Also, Player 1 may pick a currently valid atomic fact to check whether it is indeed preserved along the edges that contain all involved elements in their label.

$$\delta(R\bar{a}, \mathfrak{C}) = \begin{cases} \langle\cdot\rangle\{\text{true}\} & \mathfrak{C} \models R\bar{a}, \\ \langle\cdot\rangle\{\text{false}\} & \text{otherwise.} \end{cases}$$

If the player agree on a local test, the run is finite: $\delta(\text{true}) = \delta(\text{false}) = \emptyset$.

On an infinite run, the automaton assumes forever either the state check or some state (i,j). Since in the first case Player 0 should win, we set $\Omega(\text{check}) = 0$. In the second case, instead, Player 0 should lose, because he does not provide a witness to the guardedness of i and j after a finite number of steps. To enforce that, we set $\Omega((i,j)) = 1$ for all $(i,j) \in [k]^2$.

It is easy to see that the above checks ensure the consistency and the guardedness of the input tree.

Lemma 19.17. *The automaton $\mathcal{A}_k^2$ recognises the set of k-decomposition trees of all σ-structures.*

Reduction to the emptiness problem. After having constructed an automaton $\mathcal{A}_\psi^2$ for the model checking of a tree representation and an automaton $\mathcal{A}_k^2$ to check the validity of the input tree, we can build the product automaton $\mathcal{B}_\psi^2 := \mathcal{A}_\psi^2 \times \mathcal{A}_k^2$ which recognises precisely the set of k-decomposition trees of all models of ψ. In this way, the satisfiability problem for ψ is reduced to the emptiness problem for $\mathcal{B}_\psi^2$.

Proposition 19.18. *A μCGF formula ψ is satisfiable iff $L(\mathcal{B}_\psi^2) \neq \emptyset$.*

Emptiness of two-way automata. In order to establish the complexity of the emptiness problem for our automata model we will reduce it to the two-way automata introduced by Vardi [9] defined for input trees of bounded branching degree.

Lemma 19.19. *Any two-way automaton recognising a non-empty language, accepts some tree whose degree is bounded by the size of its state set.*

Proof. Let $\mathcal{A}^2$ be a two-way automaton accepting some input tree T. Hence, Player 0 has a winning strategy in the parity game $\mathcal{G}(\mathcal{A}^2, T)$ and, by [3,7], even a memoryless one: $f : Q_0 \times T \to Q \times T$. For any node $v \in T$, let $S(v)$ be the set of nodes targeted by f at some position with v:

$$S(v) := \{\, v' \in T \mid f(q, v) \in Q \times \{v'\} \text{ for some } q \in Q_0 \,\}.$$

Consider now the tree T' obtained from T by discarding at every node v those successors which are not in $S(v)$. Since $|S(v)| \leq |Q_0|$, this yields a tree of branching degree bounded by $|Q_0| \leq |Q|$. Moreover, f is still a winning strategy in $\mathcal{G}(\mathcal{A}^2, T')$. In other words, the automaton $\mathcal{A}^2$ also accepts T'.

19.5 Complexity

Since Vardi's automata work on trees with unlabelled edges, we have to remove the edge labels and place them into their target node. Then, our automaton has to verify the validity of taken transitions, thus, requiring a blow-up of its state set by the size of the edge alphabet. Taking into account this modification, we can transfer the complexity of the emptiness test of Vardi's automata to our model.

Theorem 19.20. *The emptiness of a two-way alternating automaton with s states and t edge symbols can be decided in time $2^{O((st)^2)}$.*

For the computations in the remainder of this chapter, let us fix a formula ψ of size n and width k. Note that, the number of states of the automata $\mathcal{A}^2_\psi$ and $\mathcal{A}^2_k$ is bounded by $O(n \cdot k^k)$. Accordingly, their product $\mathcal{B}^2_\psi$ has at most $O(n^2 k^{2k})$ states. From Lemma 19.19 we can now infer a stronger variant of the tree model property for μCGF.

Proposition 19.21. *Any satisfiable μCGF-formula of width k has a model with a tree decomposition of width at most $k-1$ and branching degree bounded by $O(n^2 k^{2k})$.*

By Theorem 19.20, the reduction of the satisfiability problem for ψ to the emptiness of $\mathcal{B}^2_\psi$ yields the following complexity bounds:

$$2^{O((n \cdot k^k)^4)} = 2^{O(n^4 2^{4k \log k})} = 2^{2^{O(n)}}.$$

When k is bounded by a constant, the above expression boils down to $2^{O(n^4)}$.

Since the complexity results on CGF quoted in the introduction of this chapter imply hardness of this bounds, we can state:

Theorem 19.22. *The satisfiability problem for μCGF is 2*EXPTIME*-complete in the general case. For clique guarded fixed point sentences of bounded width it is* EXPTIME*-complete.*

Literature

1. D. Berwanger and E. Grädel, *Games and model checking for guarded logics*, in Proceedings of the 8th International Conference on Logic for Programming, Artificial Intelligence and Reasoning, LPAR 2001, vol. 2250 of Lecture Notes in Artificial Intelligence, Springer-Verlag, 2001, pp. 70–84.
2. E. A. Emerson and C. S. Jutla, *The complexity of tree automata and logics of programs (exteded abstract)*, in Proceedings of the 29th Annual Symposium on Foundations of Computer Science, FoCS '88, IEEE Computer Society Press, 1988, pp. 328–337.
3. ———, *Tree automata, mu-calculus and determinacy (extended abstract)*, in Proceedings of the 32nd Annual Symposium on Foundations of Computer Science, FoCS '91, IEEE Computer Society Press, 1991, pp. 368–377.
4. E. Grädel, *On the restrainning power of guards*, Journal of Symbolic Logic, 64 (1999), pp. 1719–1742.
5. ———, *Guarded fixed point logics and the monadic theory of countable trees*, Theoretical Computer Science, 288 (2002), pp. 129 – 152.
6. E. Grädel and I. Walukiewicz, *Guarded fixed point logic*, in Proceedings of the 4th Annual IEEE Symposium on Logic in Computer Science, LICS '99, IEEE Computer Society Press, 1999, pp. 45–54.
7. A. W. Mostowski, *Games with forbidden positions*, Tech. Rep. 78, Instytut Matematyki, Uniwersytet Gdański, Poland, 1991.
8. M. Y. Vardi, *Why is modal logic so robustly decidable?*, in Descriptive Complexity and Finite Models: Proceedings of a DIMACS Workshop, vol. 31, American Mathematical Society, 1996, pp. 149–184.
9. ———, *Reasoning about the past with two-way automata.*, in Proceedings of the 25th International Colloquium on Automata, Languages and Programming, ICALP '98, vol. 1443 of Lecture Notes in Computer Science, Springer-Verlag, 1998, pp. 628–641.