Decomposition Theorems and Model-Checking for the Modal μ-Calculus

Mikolaj Bojanczyk
University of Warsaw
bojan@mimuw.edu.pl

Christoph Dittmann and Stephan Kreutzer
Technical University Berlin
{christoph.dittmann, stephan.kreutzer}@tu-berlin.de

Categories and Subject Descriptors F.4.1 [*Mathematical Logic*]: Modal logic

General Terms Theory

Keywords mu-calculus, modal logic, decomposition theorems, fixed-parameter complexity, parity games, model checking

Abstract

We prove a general decomposition theorem for the modal μ-calculus L_μ in the spirit of Feferman and Vaught's theorem for disjoint unions. In particular, we show that if a structure (i.e., transition system) is composed of two substructures M_1 and M_2 plus edges from M_1 to M_2, then the formulas true at a node in M only depend on the formulas true in the respective substructures in a sense made precise below.

As a consequence we show that the model-checking problem for L_μ is fixed-parameter tractable (fpt) on classes of structures of bounded Kelly-width or bounded DAG-width. As far as we are aware, these are the first fpt results for L_μ which do not follow from embedding into monadic second-order logic.

1. Introduction

The modal μ-calculus L_μ, introduced by Dexter Kozen in 1983, is a well-known logic in the theory of verification that encompasses many other modal logics. Among others, propositional dynamic logic (PDL), linear time logic (LTL) and the full branching time logic (CTL*) have embeddings into L_μ. See e.g. [5] for a survey of the μ-calculus including these results.

It seems that L_μ strikes a good balance between expressivity and complexity. The computational complexity of the model-checking problem, i.e., the problem of checking whether a formula $\varphi \in L_\mu$ is true at a node v of a structure M (in this paper we use the term *structure* for *transition systems* or *Kripke structures*) is of particular interest~~, especially in the field of formal verification~~. The problem is polynomial-time reducible to the problem of determining the winner of a parity game, a certain kind of 2-player game played on directed graphs, and most approaches for analyzing the complexity of L_μ model-checking are based on parity games.

The problem of determining the winner of a parity game is in NP $\cap$ coNP, and in fact it is even in UP $\cap$ coUP [15]. Despite 30 years of research, the question whether parity games can be decided in polynomial time is a long-standing open problem in the theory of logics for verification.

As a precise analysis of the classical complexity of L_μ model-checking remains elusive, we study the problem within the framework of parameterized complexity theory [8, 10]. In particular, we aim at algorithms verifying whether a formula φ is true at a node v in a structure M in time $f(\varphi) \cdot |M|^c$, where f is a computable function from formulas into the positive integers and c is a constant independent of φ. Computational problems that can be solved in this way, i.e., in time $f(k) \cdot n^c$, where n is the size of input and k is a *parameter* of the input, a natural number such as length or quantifier-depth of a formula, are called *fixed-parameter tractable (fpt)* and the class of all fpt problems is denoted FPT.

The parameterized complexity of logics such as monadic second-order logic (MSO) or first-order logic (FO) has been well studied in the literature, especially in the context of algorithmic meta-theorems. See e.g. [12] for a recent survey. However, not much is known about the parameterized complexity of L_μ. As every L_μ-formula can be translated into an equivalent MSO formula, fpt results for MSO immediately imply fpt results for L_μ. As a consequence, L_μ is fpt on classes of structures of bounded clique-width [7], bi-rank-width [16] or tree-width [6]. However, besides these results that follow from embedding into MSO, we are not aware of any other tractable cases.

On the other hand, we know more about solving parity games on restricted classes. One of the first results in this direction was by Jan Obdrzálek [19], who showed that parity games of bounded tree-width can be solved in polynomial time. This result was later extended to bounded clique-width [20]. Since parity games are directed graphs, it is natural to look for graph measures taking the direction of edges into account. Such measures include directed path-width [1], DAG-width [2], Kelly-width [13], directed tree-width [14] and entanglement [3]. Classes of parity games for which any of these measures is bounded can be solved in polynomial time (see [2, 3, 13]), with the exception of directed tree-width. Solving parity games in polynomial time on directed tree-width is still an open problem.

CSL-LICS 2014, July 14–18, 2014, Vienna, Austria.

ACM 978-1-4503-2886-9. . . $15.00.
http://dx.doi.org/10.1145/2603088.2603144

A class of digraphs where the DAG- or Kelly-width is bounded also has bounded directed tree-width. DAG-width and Kelly-width are as yet uncomparable concepts. However, any class of digraphs of bounded directed path-width has bounded Kelly- and DAG-width, which implies polynomial time solvability of parity games of bounded directed path-width by the results cited above.

Our contributions. The aim of this paper is to develop the logical and algorithmic tools for proving fixed-parameter tractability of L_μ-model-checking on special classes of structures such as classes of bounded Kelly-width.

Such classes already contain natural and interesting examples of transition systems. However, we see our work also as a first step in a more general program of showing that L_μ-model-checking is fpt in general. For this, it is easily seen that it suffices to solve the problem on planar structures. We therefore aim, as a next step, to show that it is fpt on classes of planar structures of bounded directed tree-width. A general duality theorem [17] states that if the directed tree-width is high, then the structure contains a grid-like substructure. In the planar case, this yields a natural decomposition of the structure into smaller substructures which can possibly be exploited for solving L_μ-model-checking for structures of very high directed tree-width. The techniques we develop in this paper are a first step towards this goal and we believe that they will prove useful for classes of structures beyond bounded Kelly-width or bounded DAG-width.

Furthermore, besides the algorithmic applications, we believe that the decomposition theorems we establish below may be of independent interest.

Main contributions to logic of this paper. An important logical tool in the analysis of the parameterized complexity of model checking for FO or MSO are *decomposition theorems*, also referred to as Feferman-Vaught style theorems (see [18] for a comprehensive survey). Whereas for FO and MSO a range of such theorems are known, much less seems to be available for L_μ. In this paper we prove a general decomposition theorem for L_μ that allows us to compute the formulas true at a node in a structure from the formulas true at the nodes in some induced substructures. Our theorem is similar in spirit to the theorem by Feferman and Vaught on disjoint unions [9]. As far as we are aware, no such theorem was known for L_μ prior to our work.

The first step for such a theorem is finding a useful notion for the "depth" of a formula, so that up to equivalence there are only finitely many formulas up to a given depth, and that the types of the nodes in the full structure can be computed from the types of the nodes in some induced substructures. We propose the notion of *μ-depth* that satisfies both constraints.

In this paper we study the construction of a structure M from two structures M_1 and M_2 where M is defined as the union of M_1 and M_2 plus an arbitrary set of edges from M_1 to M_2. We call the pair (M_1, M_2) a *directed separation* of M and refer to the intersection $M_1 \cap M_2$ as the *interface*. See Definition 2.3 for details. Let (M_1, M_2) and (M_1, M_2') be two directed separations with interface X as defined above. Note that both have the same left-hand side M_1. For a given μ-depth δ, we define a notion of δ-equivalence on these separations. The main ingredient of δ-equivalence is that M_2 and M_2' realize the same L_μ-types up to μ-depth δ, when the interface nodes are indicated with special predicates. See Definition 2.4 for details.

Theorem 1.1 (Theorem 2.5) *Let δ be a μ-depth, and let $M = (M_1, M_2)$, $M' = (M_1, M_2')$ be δ-equivalent directed separations. Then for every node in M_1, the set of formulas of depth δ that it satisfies is the same in M and in M'.*

The theorem, apart from its purely logical appeal, also has applications for L_μ-model checking. The notion of equivalent structures (M_1, M_2) and (M_1, M_2') can also be read in the way that, given a huge structure (M_1, M_2), we can replace M_2 by a much smaller structure as long as it realizes the same types up to a certain depth. This will be the main tool in our algorithmic applications.

Applications to L_μ-model checking. Based on our decomposition theorems above, we show that L_μ-model checking is fpt on classes of structures of bounded Kelly-width or bounded DAG-width, provided a decomposition is given as part of the input.

Relation to other work. A natural idea for solving L_μ-model-checking on a class $\mathcal{C}$ of structures of bounded Kelly-width would be to reduce the problem to parity games and apply the polynomial-time algorithms for solving parity games of bounded Kelly-width. However, the degree of the polynomial-time algorithms for parity games in [2, 13] depends on the upper bound for the Kelly- or DAG-width of the games considered. By combining a structure of Kelly-width k and a formula φ into a parity game, the resulting game may have Kelly-width in the order of $k \cdot |\varphi|$. Hence, by translating into parity games we would not obtain fpt algorithms.

The polynomial-time algorithms for parity games developed in [2, 13, 19] all rely in some way on the concept of *borders*, *strategy profiles* and *interfaces* developed first in [19], the paper on parity games on bounded tree-width. Our results also make crucial use of these concepts. The main technical challenge we need to solve is that for our decomposition theorems we need these profiles to be definable in the μ-calculus in a uniform way, which was not necessary in the algorithmic papers on parity games.

Due to space constraints, we omit a few of the more technical proofs and refer the interested reader to the full version of this paper [4].

2. A Decomposition Theorem for L_μ

We use the usual definition of the modal μ-calculus L_μ, see for example in the comprehensive survey [5].

2.1 A Notion of Formula Depth for the μ-Calculus

Definition 2.1 Let $\overline{X} = (X_1, \ldots, X_n)$ be a finite sequence of fixpoint variables. A formula $\varphi \in L_\mu$ is called *consistent* with $\overline{X}$ if all fixpoint variables of φ (free and bound) are in the sequence, and in every subformula ψ of φ that binds a fixpoint variable X_i, only the variables $X_1, \ldots, X_i$ can appear freely in ψ. ⊣

A node x in the syntax tree of a formula that is consistent with $\overline{X}$ is called a *reboot* if the subformula in the node binds a fixpoint variable X_i such that no ancestor of x binds any of the fixpoint variables $\{X_1, \ldots, X_i\}$. The *$\overline{X}$-depth* of a formula is the biggest number of occurrences of operators from the set $\Box, \Diamond, \mu, \nu$ that can be found on a path in the syntax tree that does not visit reboot nodes. The $\overline{X}$-depth is undefined if the formula is not consistent with $\overline{X}$. Figure 1 shows a formula which has (X_1, X_2, X_3)-depth 2.

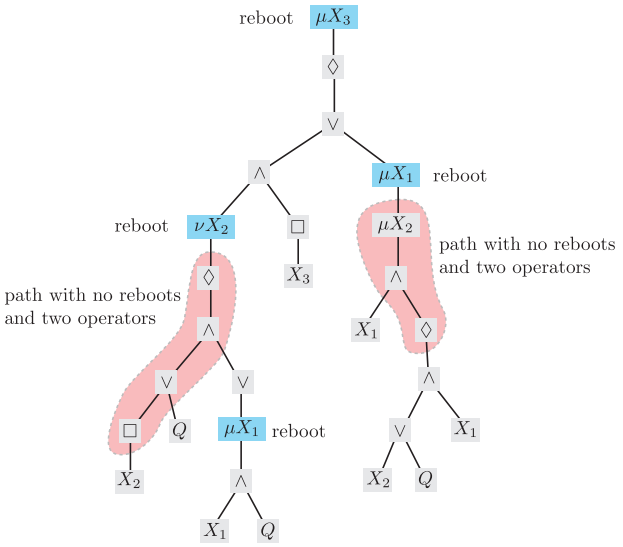

Figure 1. An example for reboots

The definition is designed so that $\mu X.\varphi$ and $\varphi[X/\mu X.\varphi]$ have the same $\overline{X}$-depth.

For a set $L \subseteq L_\mu$, define the *L-type* of a vertex in a structure to be the set of formulas from L that are true at the vertex. A *μ-depth* is a pair $\delta = (\overline{X}, d)$ where $\overline{X}$ is a sequence of fixpoint variables and d is a natural number. A formula is called *consistent with δ* if it is consistent with $\overline{X}$ and its $\overline{X}$-depth is at most d. The δ-type of a vertex in a structure is its L-type, with L being the set of all formulas consistent with δ. This information is finite thanks to the following lemma.

Lemma 2.2 *For every μ-depth δ and finite set of propositional variables, up to logical equivalence there are finitely many formulas in these propositional variables that are consistent with δ.*

Although the set in the statement of the above lemma is finite, its size is non-elementary with respect to δ.

2.2 Decompositions of Directed Separations

As promised in the introduction, we will prove a decomposition theorem for the union of two structures with a small intersection and some additional edges all going in the same direction. To formalize this, we introduce *directed separations*.

Definition 2.3 Let M be a σ-structure. A pair (M_1, M_2) of induced substructures is a *directed σ-separation of M with interface $\overline{X} = (x_1, \dots, x_k)$* if

- $V(M) = V(M_1) \cup V(M_2)$,
- $X = \{x_1, \dots, x_k\} = V(M_1) \cap V(M_2)$,
- and there are no edges from $M_2 \setminus X$ to $M_1 \setminus X$. ⊣

Abusing notation, we write $M = (M_1, M_2)$ to denote that (M_1, M_2) is a directed separation of M, and notationally we consider (M_1, M_2) to be interchangeable with M.

For some k, let $\overline{P} = P_1, \dots, P_k$ be a sequence of fresh proposition symbols. For a σ-structure M and a k-tuple $\overline{X} = (x_1, \dots, x_k) \in V(M)^k$, we define $\partial_{\overline{P}}(M, \overline{X})$ to be the $\sigma \cup P$-structure based on M such that P_i is true only at the node x_i. If the sequence $\overline{P} = P_1, \dots, P_k$ is longer than $\overline{X} = (x_1, \dots, x_l)$, then $\partial_{\overline{P}}(M, \overline{X})$ is defined the same except that P_i is always false for $i > l$.

Definition 2.4 Let (M_1, M_2), (M_1, M_2') be two directed separations with the same interface $\overline{X}$. Let $\overline{P}$ be a set of $|X|$ many proposition symbols and let $L \subseteq L_\mu[\sigma \cup P]$.

We call (M_1, M_2), (M_1, M_2') *L-equivalent* if

- for every vertex in X, its L-type is the same in $\partial_{\overline{P}}(M_2, \overline{X})$ and $\partial_{\overline{P}}(M_2', \overline{X})$, respectively, and
- for every edge (v, w) in (M_1, M_2) with $v \in M_1, w \in M_2$ there is an edge (v, w') in (M_1, M_2') with $w \in M_2'$ such that w and w' have the same L-types in $\partial_{\overline{P}}(M_2, \overline{X})$ and $\partial_{\overline{P}}(M_2', \overline{X})$, respectively, and vice versa. ⊣

If δ is a μ-depth, we say that two directed separations are *δ-equivalent* if they are L-equivalent with L being all formulas consistent with δ. Let us state our main theorem.

Theorem 2.5 *Let δ be a μ-depth, and let $M = (M_1, M_2)$, $M' = (M_1, M_2')$ be δ-equivalent directed separations. Then for every node in M_1, its δ-type is the same in M and M'.*

In fact, we will prove a more general version of Theorem 2.5, without limiting us to μ-depth. It turns out that there exists a suitable closure operator $\mathrm{CL}_P : 2^{L_\mu} \to 2^{L_\mu}$ that maps finite sets to finite sets such that the main theorem holds for $\mathrm{CL}_P(L)$-equivalent directed separations. In particular, we can choose $L = \{\varphi\}$ if we are only interested in the model checking problem for a fixed formula φ consistent with δ. Then $\mathrm{CL}_P(\{\varphi\})$ will be significantly smaller than the set of all δ-consistent formulas.

3. Proof of the Decomposition Theorem

Definition 3.1 For $\varphi \in L_\mu$, let $\mathrm{sub}(\varphi)$ be the set of all indexed subformulas without formulas of the form X for fixpoint variables X. That is,

$$\mathrm{sub}(\varphi) := \{(\psi, i) \mid \psi \text{ is a subformula of } \varphi \text{ at position } i \text{ in the string } \varphi \text{ and } \psi \text{ is not a variable}\}.$$

Let $\mathrm{sub}^+(\varphi) = \mathrm{sub}(\varphi) \setminus \{(\varphi, 0)\}$ be the set of proper subformulas.

For an occurrence of a fixpoint variable X in a formula φ, its *definition in φ* is the enclosing fixpoint $(\mu X.\psi, i) \in \mathrm{sub}(\varphi)$ (or $(\nu X.\psi, i) \in \mathrm{sub}(\varphi)$) where this occurrence of X is quantified. For a formula $(\psi, i) \in \mathrm{sub}(\varphi)$, let $\mathrm{closure}_\varphi(\psi, i) = (\psi', i)$ be such that ψ' is the formula ψ with all free variables replaced by their definitions until there are no more free variables.

Define $\mathrm{CL}(\varphi) := \{\mathrm{closure}_\varphi(\psi, i) \mid (\psi, i) \in \mathrm{sub}(\varphi)\}$ and $\mathrm{CL}^+(\varphi) = \mathrm{CL}(\varphi) \setminus \{(\varphi, 0)\}$. ⊣

We will often not distinguish formulas in $\mathrm{sub}(\varphi)$ and $\mathrm{CL}(\varphi)$ and instead identify them via the obvious bijection that preserves the second component.

We will usually write $\psi \in \mathrm{sub}(\varphi)$ instead of $(\psi, i) \in \mathrm{sub}(\varphi)$ if there is no confusion. We only need the index i in order to distinguish identically looking subformulas.

Even though two subformulas may look identical, they could be in the scope of different fixpoint operators. A few paragraphs below we will introduce a closure operation called PT_P that modifies different subformulas in different ways in order to distinguish between these cases. For this reason we need to keep track of the positions of the subformulas.

In the rest of the paper, whenever we mention an element of $\mathrm{sub}(\varphi)$ or $\mathrm{CL}(\varphi)$, the reader should assume that it also contains the position of the subformula in φ.

Lemma 3.2 *For all $\varphi \in L_\mu$, the set*

$$\{\psi \mid (\psi, i) \in \mathrm{CL}(\varphi) \text{ for some } i\}$$

is equal to the usual definition of the Fischer-Ladner closure of φ (see e.g., [22, Definition 4.1]).

For a set of formulas $L \subseteq L_\mu$, define $\mathrm{CL}(L) := \{\psi \mid \varphi \in L, (\psi, i) \in \mathrm{CL}(\varphi)\}$. In this set we do not need the index i, different from $\mathrm{CL}(\varphi)$.

Let $P = \{P_1, \ldots, P_k\}$ be a set of proposition symbols disjoint from σ.

For a formula $\varphi \in L_\mu[\sigma \cup P]$ and $\varphi' \in L_\mu[\sigma \cup P]$, we call φ' a *priority tracking variant of* φ if φ' is syntactically derived from φ by applying the following operation for each subformula ψ of the form $\psi = \Diamond\chi$ or $\psi = \Box\chi$.

1. If $\psi = \Diamond\chi$, then pick a set $Q \subseteq P$ and replace the subformula ψ by
$$\Big(\Big(\bigvee_{R \in Q} R\Big) \vee \Diamond\chi\Big).$$
2. If $\psi = \Box\chi$, then pick a set $Q \subseteq P$ and replace the subformula ψ by
$$\Big(\Big(\bigwedge_{R \in Q} \neg R\Big) \wedge \Box\chi\Big).$$

We denote the set of all priority tracking variants of φ with respect to P by $\mathrm{PT}_P(\varphi)$. Note that $\mathrm{PT}_P(\varphi)$ is finite because φ has a finite number of subformulas and P is a finite set. Similar to CL, we define $\mathrm{PT}_P(L)$ for sets of formulas $L \subseteq L_\mu[\sigma \cup P]$ as $\mathrm{PT}_P(L) := \bigcup_{\varphi \in L} \mathrm{PT}_P(\varphi)$.

Definition 3.3 Let $\mathrm{CL}_P(L) := \mathrm{PT}_P(\mathrm{CL}(L))$. ⊣

Lemma 3.4 $\mathrm{CL}_P(\mathrm{CL}_P(L)) = \mathrm{CL}_P(L)$ *for all* $L \subseteq L_\mu[\sigma \cup P]$.

Definition 3.5 For a structure M, v, a k-tupel $\overline{X} \in V(M)^k$ and a set $L \subseteq L_\mu[\sigma \cup P]$ where $\overline{P}$ is sequence of at least k many proposition symbols, we define the $(L, \overline{P})$*-type of v in $M, \overline{X}$* as

$$\mathrm{tp}_{L,\overline{P}}(M, v, \overline{X}) := \big\{\varphi \in \mathrm{CL}_P(L) \;\big|\; \partial_{\overline{P}}(M, \overline{X}), v \models \varphi\big\}.$$

We also define the set of $(L, \overline{P})$-types realized in a structure,

$$\mathcal{T}_{L,\overline{P}}(M, \overline{X}) := \big\{\mathrm{tp}_{L,\overline{P}}(M, v, \overline{X}) \;\big|\; v \in V(M)\big\}.$$

Finally, let $\mathcal{T}_L(\overline{P}) := 2^{\mathrm{CL}_P(L)}$ be the set of all candidates for $(L, \overline{P})$-types. ⊣

Using the new terminology, let us restate Theorem 2.5 in these more general terms.

Theorem 3.6 *Let $\overline{P}$ be a sequence of proposition symbols disjoint from σ, $L \subseteq L_\mu[\sigma \cup P]$ and let (M_1, M_2), (M_1, M_2') be $\mathrm{CL}_P(L)$-equivalent directed σ-separations with interface $\overline{X}$.*

Then for all $v \in M_1$, we have

$$\mathrm{tp}_{L,\overline{P}}((M_1, M_2), v, \overline{X}) = \mathrm{tp}_{L,\overline{P}}((M_1, M_2'), v, \overline{X}).$$

It is not difficult to show that if all formulas in L are consistent with a μ-depth δ, then the same is true for $\mathrm{CL}_P(L)$ (this is Lemma 3.8). Therefore, δ-equivalence implies $\mathrm{CL}_P(L)$-equivalence, and thus Theorem 2.5 follows from Theorem 3.6. We will also use a different and slightly stronger way of stating Theorem 3.6, stated below.

Theorem 3.7 *Let $\overline{P}$, $\overline{Q}$ be sequences of proposition symbols such that $\sigma \cap P = \sigma \cap Q = P \cap Q = \emptyset$.*

Let $L \subseteq L_\mu[\sigma \cup P]$ and M be a structure with a directed σ-separation (M_1, M_2) with interface $\overline{X}$. Let $\overline{Y} \in V(M_1)^{|Q|}$ be a tuple.

For all $v \in M_1$, the set $\mathrm{tp}_{L,\overline{Q}}(M, v, \overline{Y})$ depends only on

- *M_1 and $\overline{Q}$ and*
- *$\big\{(x_i, \mathrm{tp}_{L,\overline{P}}(M_2, x_i, \overline{X})) \;\big|\; x_i \in X\big\}$ and*
- *$\big\{(v, \mathrm{tp}_{L,\overline{P}}(M_2, w, \overline{X})) \mid (v, w) \in E(M) \cap (M_1 \times M_2)\big\}$.*

Provided L is finite, $\mathrm{tp}_{L,\overline{Q}}(M, v, \overline{Y})$ can be computed from these sets.

Furthermore, for every $w \in M_2$, the set $\mathrm{tp}_{L,\overline{Q}}(M, w, \overline{Y})$ depends only on the above sets and on $\mathrm{tp}_{L,\overline{P}}(M_2, w, \overline{X})$ and can be computed from these sets if L is finite.

3.1 Parity Games

To prove the decomposition theorems, we want to use the model checking game of the modal μ-calculus. Instead of replacing a substructure by a different substructure preserving the types in the whole structure, we replace a subgame by a different subgame preserving the winner in the whole game.

For this, we first need parity games, strategies and the model checking game. These are all well-known concepts in the literature, see for example [11], so we skip over the exact definitions.

The winner of a parity game from a given node is always determined. However, in order to replace subgames by different subgames preserving the winner in the whole game, we need a more subtle analysis of the subgame than just its winner.

We call the intersection between a subgame and the rest of the game its *interface*. For the more subtle analysis, we look at *partial* strategies, which may be undefined on some nodes of the interface. If a partial strategy is undefined on some node, the player indicates that she would like to leave the subgame. These strategies can be partially ordered by their *profiles*, that is, the set of interface nodes that are possibly reachable by Player $\Box$, together with the worst priority that Player $\Box$ can enforce.

All this culminates in a proof that the feasibility of profiles of strategies is in fact definable in L_μ. The formulas that define profiles in a partial model checking game of φ will all be in $\mathrm{CL}_P(\{\varphi\})$, so this proves that $\mathrm{tp}_{\{\varphi\},\overline{P}}(M, v, \overline{X})$ determines the set of possible profiles, which we will use to define a specific parity game.

Let $\overline{Z} = (Z_1, \ldots, Z_n)$ be a finite sequence of fixpoint variables. Recall the definition of a formula consistent with $\overline{Z}$ (Definition 2.1 on page 2). We strengthen this definition in the sense that every Z_i is either bound only in μ-subformulas or only in ν-subformulas. Let $(p_1, \ldots, p_n)$ be a strictly increasing sequence of natural numbers such that p_i is odd if and only if Z_i is only bound in μ-subformulas.

Let $\varphi \in L_\mu$ be consistent with $\overline{Z}$ and $\mu Z_i.\psi \in \mathrm{sub}(\varphi)$. We write $\overset{p_i}{\mu} Z_i.\psi$ to indicate that Z_i gets the priority p_i in the

model checking game that we will define shortly (similarly for ν). We call a formula with numbers over their fixpoint operators an *annotated* formula. In this section it does not affect the results if the sequences are infinite.

From now on, let us fix a sequence $\overline{Z}$ and a corresponding priority sequence $(p_1, \ldots, p_n)$. All formulas in the rest of this section should be consistent with $\overline{Z}$ and annotated with the p_i, even if we do not mention this explicitly. For example, a formula $\nu Y.\Diamond(\mu X.\nu Y.\Diamond X \vee \Diamond Y) \vee \Diamond Y$ consistent with (X, Y) under the priority sequence $(1, 2)$ would be labelled as $\overset{2}{\nu} Y.\Diamond(\overset{1}{\mu} X.\overset{2}{\nu} Y.\Diamond X \vee \Diamond Y) \vee \Diamond Y$. Note that it cannot be labelled $\overset{2}{\nu} Y.\Diamond(\overset{3}{\mu} X.\overset{4}{\nu} Y.\Diamond X \vee \Diamond Y) \vee \Diamond Y$, even though these priorities would work in the model-checking game. However, they violate the sequence (X, Y) and the priority sequence $(1, 2)$.

Note that the first formula is an element of $\mathrm{CL}(\overset{1}{\mu} X.\ \overset{2}{\nu} Y.\Diamond X \vee \Diamond Y)$. This holds true in general.

Lemma 3.8 *Let $\overline{Z} = (Z_1, \ldots, Z_n)$, $\varphi \in L_\mu$ be consistent with $\overline{Z}$ and $\psi \in \mathrm{CL}(\varphi)$. Then ψ is consistent with $\overline{Z}$.*

Definition 3.9 A *parity game* $P = (V, V_\Diamond, E, \omega)$ is a directed graph (V, E) with $V_\Diamond \subseteq V$ and a function $\omega : V \to \mathbb{N}$ mapping nodes to *priorities*. ⊣

An infinite path is winning for Player $\Diamond$ iff the minimum priority on this path is even. Strategies are partial functions $\pi : V(P)^+ \to V(P)$ and π-conforming paths are defined in the usual way.

We write $\mathrm{P}[M, \varphi] = (V, V_\Diamond, E, \omega)$ for the usual model checking game of M and φ, with $V(\mathrm{P}[M, \varphi]) := M \times \mathrm{CL}(\varphi)$ and where a node (v, ψ) has the priority

$$\omega(v, \psi) := \begin{cases} p & \text{if } \psi = \overset{p}{\mu} X.\chi \text{ or } \psi = \overset{p}{\nu} X.\chi \text{ for some } \chi \\ p_{\max} & \text{otherwise,} \end{cases}$$

where $p_{\max}$ is the maximum priority.

Lemma 3.10 *For a structure M, v, a formula $\varphi \in L_\mu$ and $\psi \in \mathrm{CL}(\varphi)$, Player $\Diamond$ wins $(\mathrm{P}[M, \varphi], (v, \psi))$ iff $M, v \models \psi$.*

3.2 Profiles and Types

In the previous section we considered parity games, (positional) strategies and the model checking game. We now generalize these definition to partial games and partial strategies. This is necessary so we can analyze the effect of replacing a subgame by a different, but in some sense similar subgame.

Definition 3.11 A *partial parity game* is a parity game P with a subset $U \subseteq V(P)$ called the *interface*. ⊣

The game is played the same way as a parity game, except that upon reaching an interface $\Diamond$-node, Player $\Diamond$ may choose to end the play and win immediately. Therefore, a *partial strategy* for Player $\Diamond$ is defined the same way as in a non-partial parity game, except that the partial strategy may be undefined on plays that end in an interface $\Diamond$-node.

A partial strategy π is called *winning* if for every strategy of the opponent, the resulting play either visits an interface node where π is undefined or satisfies the parity condition. The exact formal definitions can be found in the full version of this paper [4].

Definition 3.12 Let $\varphi \in L_\mu[\sigma]$, M be a σ-structure and $X \subseteq V(M)$. The game $\mathrm{P}[X, M, \varphi]$ is the partial parity game defined as $\mathrm{P}[M, \varphi]$ with interface $\{(v, \psi) \in X \times \mathrm{CL}(\varphi) \mid \psi \text{ starts with } \Diamond \text{ or } \Box\}$. We will usually write $\mathrm{P}[M, \varphi]$ for this game if X is clear from the context. ⊣

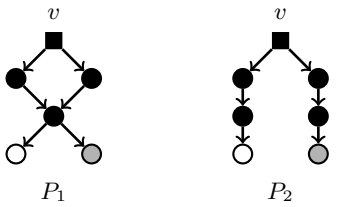

Figure 2.

We emphasize again that $\mathrm{P}[X, M, \varphi]$ and $\mathrm{P}[M, \varphi,]$ are exactly the same game, only viewed from two different angles.

Definition 3.13 Let P be a partial parity game with interface U. We define

$$\text{strategy-targets}(P) := \{(u, p) \mid u \in U,\ p \text{ a priority of } P\}$$
$$\text{profiles}(P) := \{y \subseteq \text{strategy-targets}(P) \mid \text{for all } u \text{ there is at most one } p \text{ with } (u, p) \in y\}.$$ ⊣

Definition 3.14 Let $\sqsubseteq$ be the *reward ordering* on priorities. That is, $p \sqsubseteq p'$ if p is better for Player $\Diamond$ than p'. Formally, $p \sqsubseteq p'$ is true if and only if

- p is even and p' is odd or
- both p and p' are even and $p \leq p'$ or
- both p and p' are odd and $p \geq p'$. ⊣

Definition 3.15 Let P be a partial parity game with interface U, $v_1 \in V(P)$ and let π be a partial winning strategy for (P, v_1). We define

$$\text{preprofile}(\pi, v_1) := \{(v_n, \min_{1 \leq i \leq n} \omega(v_i)) \mid n > 1,\ (v_1, \ldots, v_n) \text{ is a path with } v_n \in U \text{ and } (v_1, \ldots, v_n) \notin \text{dom}(\pi)\}$$
$$\text{profile}(\pi, v_1) := \{(u, p) \mid p \text{ is } \sqsubseteq\text{-maximal such that } (u, p) \in \text{preprofile}(\pi, v_1)\}.$$

The min is taken with respect to the usual ordering $\leq$.

We say that a profile $y \in \text{profiles}(P)$ is *possible* on (P, v_1) if there exists a π such that $y = \text{profile}(\pi, v_1)$. ⊣

Definition 3.16 Let $y, y' \in \text{profiles}(P)$. We say that y *is at least as good as* y' iff for every $(u, p) \in y$, there is a $(u, p') \in y'$ with $p \sqsubseteq p'$. We denote this as $y \sqsubseteq y'$. ⊣

As an example, consider the two parity games given in Figure 2 with interface nodes ○, ●. For simplicity, we assume that all nodes in these parity games have priority 0. Then the profile $\{(○, 0)\}$ is possible on (P_1, v) but not on (P_2, v). On the other hand, the profile $\{(○, 0), (●, 0)\}$ is possible on both (P_1, v) and (P_2, v). Note that on (P_1, v), the last profile is only possible with a non-positional strategy. However, the need for a non-positional strategy here is of course somewhat artificial because Player $\Diamond$ must deliberately avoid a decision where she could simply make one.

As one might expect, every partial strategy can be converted into a positional partial strategy at least as good as the original strategy.

Lemma 3.17 *Let $P = (V, V_\Diamond, E, \omega)$ be a partial parity game with interface U, $v \in V$ and π be a partial strategy for (P, v). Then there exists a positional partial strategy ρ such that* $\text{profile}(\rho, v) \sqsubseteq \text{profile}(\pi, v)$.

The proof is a reduction to the positional determinacy of (non-partial) parity games.

Definition 3.18 The type of a node $v \in V(P)$ is the set of optimal profiles.

$$\begin{aligned}\text{ptype}_P(v) := \{&\text{profile}(\pi, v) \mid \\ &\pi \text{ is a partial winning strategy for } (P, v) \text{ and} \\ &\text{there is no partial winning strategy } \pi' \text{ such that} \\ &\quad \text{profile}(\pi', v) \sqsubset \text{profile}(\pi, v)\}. \end{aligned}$$ ⊣

By Lemma 3.17, the strategies occurring in the above definition can be chosen to be positional.

Next, we define the notion of a parity game *simulating* another parity game. A game simulates another game if it behaves in the same way when viewed from the outside. For every node in the old game there must be a node in the new game that has the same type. Internally the games could be quite different, and in fact the new game could have a very different number of nodes than the old game.

Our goal is to find small games that simulate large games.

Definition 3.19 Let P, P' be partial parity games with the same interface U.

The game P' *simulates* P if there is a map $f : V(P) \to V(P')$ such that $f(u) = u$ for all $u \in U$ and for every node $v \in V(P)$, $\text{ptype}_P(v) = \text{ptype}_{P'}(f(v))$. ⊣

Whenever we have a game P with an induced subgame Q with no edges going from Q to the rest of P except via the interface of Q, we can replace Q in P by one of its simulations without the rest of P noticing.

Lemma 3.20 (Simulation Lemma) *Let P, Q be parity games such that Q is an induced subgame of P with interface U and with no edges from $Q \setminus U$ to $P \setminus Q$. Let Q' be a partial parity game with interface U which simulates Q via the function $f : V(Q) \to V(Q')$. Extend f to $V(P)$ by letting $f(v) = v$ for all $v \in V(P) \setminus V(Q)$.*

Define P' as the parity game where the induced subgame Q has been replaced by Q' and edges pointing to nodes $v \in V(Q)$ now point to $f(v) \in V(Q')$.

Then for all $v \in V(P)$, Player $\Diamond$ wins (P, v) iff Player $\Diamond$ wins $(P', f(v))$.

Proof. Translation of strategies. Because the types agree, neither player can be worse off in one game. □

3.3 Definable Profiles

In the next step, we would like to encode a profile in a formula. Given a profile y in a model checking game and a starting point $x = (x', \psi)$, we would like to define a formula ψ^y with the property that ψ^y is true on the node x' in the structure if and only if the profile y is possible on (P, x). However, we do not know how to do this.

Hence we weaken the restriction and want ψ^y to be true iff a profile $y' \sqsubseteq y$ is possible. This is enough for our purposes because the type of x only cares about $\sqsubseteq$-minimal profiles. This formula turns out to be definable. Using a suitable definition of ψ^y, we get the following theorem.

Theorem 3.21 *Let $\overline{P}$ be a sequence of proposition symbols disjoint from σ. Let $\varphi \in L_\mu[\sigma \cup P]$, M, v be a σ-structure and $\overline{X}$ be a sequence of nodes of M. For $\psi \in \text{CL}(\varphi)$, $y \in \text{profiles}(\text{P}[M, \varphi])$, it holds that $M, v \models \psi^y$ iff there is a positional partial winning strategy π for $(\text{P}[M, \varphi], (v, \psi))$ such that $\text{profile}(\pi, (v, \psi)) \sqsubseteq y$.*

Corollary 3.22 *Let $\overline{P}$ be a sequence of proposition symbols disjoint from σ. Let $\varphi \in L_\mu[\sigma \cup P]$, M, v be a σ-structure, $\overline{X} \in V(M)^{|P|}$ and $\psi \in \text{CL}(\varphi)$. Then*

$$\text{ptype}_{\text{P}[M,\varphi]}((v, \psi)) = \Big\{ y \in \text{profiles}(\text{P}[M, \varphi]) \;\Big|\; M, v \models \psi^y \text{ and there is no } y' \sqsubset y \text{ with } M, v \models \psi^{y'} \Big\}.$$

That is, $\text{tp}_{\{\varphi\}, \overline{P}}(M, v, \overline{X})$ *determines* $\text{ptype}_{\text{P}[M,\varphi]}((v, \psi))$.

Before we can explain ψ^y, we need one more definition.

Definition 3.23 For an annotated $\varphi \in L_\mu[\sigma]$, $\psi \in \text{CL}(\varphi)$ and $\chi \in \text{sub}(\psi)$, let $\text{prio}_\varphi(\psi \rightsquigarrow \chi)$ be the minimum priority of all fixpoint operators that enclose χ in ψ. ⊣

Definition 3.24 Let $\overline{P} = (P_1, \dots, P_k)$ be a sequence of proposition symbols disjoint from σ. Let $\varphi \in L_\mu[\sigma \cup P]$ be a formula, M be a σ-structure and $X = (x_1, \dots, x_k) \in V(M)^k$. Let $\psi \in \text{CL}(\varphi)$ and $y \in \text{profiles}(\text{P}[M, \varphi])$. For every $\psi' \in \text{sub}^+(\psi)$, there is a formula $\varphi' \in \text{CL}(\varphi)$ corresponding to ψ'. We inductively define an operation $\cdot^y$ over the structure of ψ'.

$$\begin{aligned} V^y &:= V, \quad (\neg V)^y := \neg V && \text{for prop. or var. } V \\ (\chi * \chi')^y &:= (\chi^y) * (\chi'^y) && \text{for } * \in \{\vee, \wedge\} \\ (\alpha X.\chi)^y &:= \alpha X.(\chi^y) && \text{for } \alpha \in \{\mu, \nu\} \\ (\Diamond\chi)^y &:= \Big(\Big(\bigvee_{i \in N} P_i \Big) \vee \Diamond(\chi^y) \Big) \\ (\Box\chi)^y &:= \Big(\Big(\bigwedge_{i \in N} \neg P_i \Big) \wedge \Box(\chi^y) \Big) \end{aligned}$$

In the case $\Diamond\chi$, we use

$$\begin{aligned} N := \{&1 \le i \le k \mid \\ &((x_i, \varphi'), p') \in y \text{ for some } p' \sqsupseteq \text{prio}_\varphi(\psi \rightsquigarrow \Diamond\chi)\}. \end{aligned}$$

In the case $\Box\chi$, we use

$$\begin{aligned} N := \{&1 \le i \le k \mid \\ &((x_i, \varphi'), p') \notin y \text{ for all } p' \sqsubset \text{prio}_\varphi(\psi \rightsquigarrow \Box\chi)\}. \end{aligned}$$

In both cases, $\varphi' \in \text{CL}(\varphi)$ is the formula corresponding to $\Diamond\chi$ or $\Box\chi$, respectively. ⊣

The motivation behind this seemingly quite arbitrary definition is that if a profile says we can reach $(x_i, \Diamond\chi)$ with the worst priority p', and the actual priority we have is at least as good as p', we are allowed to take the shortcut and leave the game. That is why we add X_i to the disjunction in this case. Of course, we need to pay close attention to the games that are involved, because $(x_i, \Diamond\chi)$ is not a node in $\text{P}[M, \varphi]$ and y is not a profile of $\text{P}[M, \psi]$. However, this is not a problem because every $\Diamond\chi$ corresponds to a unique $\varphi' \in \text{CL}(\varphi)$, and the game $\text{P}[M, \psi]$ is a partial unfolding of the $\text{P}[M, \varphi]$. This means that every strategy on one of these games is also a strategy on the other game, although not necessarily positional.

Dually, in the case $\Box\chi$, if the actual priority is worse than what the profile wants, we must make sure that $(x_i, \Box\chi)$ is not reached, so we add $\neg X_i$ with a conjunction.

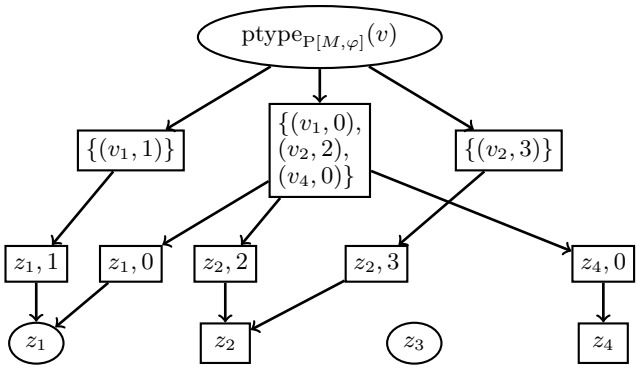

Figure 3. A part of P^φ

A formal statement of this explanation is Theorem 3.21. Due to space constraints, the full proof of this theorem can be found in the full version of this paper [4].

3.4 A Small Parity Game

With Theorem 3.21 at our hands, we can now define a partial parity game simulating the model checking game that only depends on the types of some nodes in the original structure. The parity game consists of four layers of nodes.

1. One layer of $\diamond$-nodes, one for each type, where Player $\diamond$ can choose a profile.
2. Then one layer of $\Box$-nodes, one for each profile, where Player $\Box$ can choose one of the allowed paths.
3. Then a layer of nodes with out-degree 1 to ensure the priorities match the chosen path.
4. Finally a layer representing the interface.

The edges only point from one layer to the next or from the last layer back to the first layer. Formally, let M be a structure and $X = \{x_1, \dots, x_k\} \subseteq V(M)$. Let $\varphi \in L_\mu$. First, we define the layers described above.

$$V_1 := 2^{\text{profiles}(\mathrm{P}[M,\varphi])} \qquad V_3 := \text{strategy-targets}(\mathrm{P}[M,\varphi])$$
$$V_2 := \text{profiles}(\mathrm{P}[M,\varphi]) \quad V_4 := X \times \mathrm{CL}(\varphi).$$

Next, we define the game $P^\varphi = (V, V_\diamond, E, \omega)$ with interface V_4 depending only on φ and the sets $\mathrm{tp}_{\{\varphi\},\overline{P}}(M, x_i, \overline{X})$, but not on M.

$$V := V_1 \cup V_2 \cup V_3 \cup V_4 \qquad E := E_1 \cup E_2 \cup E_3 \cup E_4$$
$$V_\diamond := V_1 \cup \{(x_i, \psi) \in V_4 \mid \psi \text{ starts with a } \diamond\}$$
$$\omega(v) := \begin{cases} p & \text{for } v = (x_i, \psi, p) \in V_3 \\ p' & \text{otherwise,} \end{cases}$$

where p' is the maximum priority of φ.

For the set of edges, we connect the nodes according to the subset relation and the nodes from V_4 back to their types.

$$E_1 := \{(x, y) \in V_1 \times V_2 \mid y \in x\}$$
$$E_2 := \{(x, y) \in V_2 \times V_3 \mid y \in x\}$$
$$E_3 := \left\{\big((x_i, \psi, p), (x_i', \psi')\big) \in V_3 \times V_4 \;\middle|\; (x_i, \psi) = (x_i', \psi')\right\}$$
$$E_4 := \left\{(x, t) \in V_4 \times V_1 \;\middle|\; t = \text{ptype}_{\mathrm{P}[M,\varphi]}(x)\right\}.$$

Note that E_4 is determined by the sets $\mathrm{tp}_{\{\varphi\},\overline{P}}(M, x, \overline{X})$ by Corollary 3.22.

To illustrate this construction, assume that $\mathrm{P}[M,\varphi]$ has the interface $\{z_1, \dots, z_4\} \in X \times \mathrm{CL}(\varphi)$ and a node $v \in \mathrm{P}[M,\varphi]$ with $\text{ptype}_{\mathrm{P}[M,\varphi]}(v) = \big\{\{(z_1,1)\}, \{(z_1,0),(z_2,2),(z_4,0)\}, \{(z_2,3)\}\big\}$. Figure 3 illustrates a part that could occur in the game P^φ. In the full game P^φ, we would also add the edges $(z_i, \text{ptype}_{\mathrm{P}[M,\varphi]}(z_i)) \in V_4 \times V_1$. In the node $\text{ptype}_{\mathrm{P}[M,\varphi]}(v)$, Player $\diamond$ can choose one of the possible profiles. This corresponds to Player $\diamond$ fixing a strategy π. After fixing her strategy, Player $\Box$ can choose a path through the game conforming to this strategy. The profile tells us exactly what the worst possible paths are, and the layer V_3 makes sure that the correct priority is visited.

The goal of this construction is to get a game such that the type of a node labeled $\text{ptype}_{\mathrm{P}[M,\varphi]}(v)$ is exactly $\text{ptype}_{\mathrm{P}[M,\varphi]}(v)$. This leads to the main theorem of this subsection.

Theorem 3.25 *For a formula $\varphi \in L_\mu$, a structure M and $X \subseteq V(M)$, the game P^φ simulates $\mathrm{P}[M,\varphi]$.*

Proof. For every node $u \in X \times \mathrm{CL}(\varphi)$, define $f(u) = u$. For the remaining nodes $v \in V(\mathrm{P}[M,\varphi]) \setminus (X \times \mathrm{CL}(\varphi))$, define $f(v) = \text{ptype}_{\mathrm{P}[M,\varphi]}(v) \in V_\diamond(P^\varphi)$.

All we have to do now is to show that $\text{ptype}_{\mathrm{P}[M,\varphi]}(v) = \text{ptype}_{P^\varphi}(f(v))$ for all $v \in V(\mathrm{P}[M,\varphi])$. First we show $\subseteq$.

Let π be a positional partial winning strategy for $(\mathrm{P}[M,\varphi], (v,\psi))$. We want to construct a positional partial winning strategy π' for $(P^\varphi, f((v,\psi)))$ such that $\text{profile}(\pi, (v,\psi)) = \text{profile}(\pi', f((v,\psi)))$.

For every node $(v,\psi) \in \mathrm{P}[M,\varphi]$, define

$$\pi'(\text{ptype}_{\mathrm{P}[M,\varphi]}((v,\psi))) := \text{profile}(\pi, (v,\psi)).$$

For $(x_i,\psi) \in V_\diamond(P^\varphi)$, if $(x_i,\psi) \in \text{dom}(\pi)$, then we define $\pi'((x_i,\psi)) = \text{ptype}_{\mathrm{P}[M,\varphi]}(x_i)$. Otherwise, leave $\pi'((x_i,\psi))$ undefined.

We claim that π' is a partial winning strategy on $(P^\varphi, (v,\psi))$. By Theorem 3.21, for all $(x_i,\chi) \in X \times \mathrm{CL}(\varphi)$ it holds that $M, x_i \models \chi^{\text{profile}(\pi,(x_i,\chi))}$. So the unique edge leaving from (x_i,ψ) in P^φ goes to some node y with $\text{profile}(\pi,(x_i,\psi)) \in y$.

Inductively it follows that every π'-conforming path in P^φ corresponds to a π-conforming path in $\mathrm{P}[M,\varphi]$ and vice versa. So π' is a partial winning strategy with $\text{profile}(\pi,(v,\psi)) = \text{profile}(\pi', f((v,\psi)))$.

It remains to show the other direction $\text{ptype}_{\mathrm{P}[M,\varphi]}(v) \supseteq \text{ptype}_{P^\varphi}(f(v))$.

Let π' be a positional partial winning strategy for $(P^\varphi, f((v,\psi)))$. We want to construct a partial winning strategy π for $(\mathrm{P}[M,\varphi], (v,\psi))$ such that $\text{profile}(\pi,(v,\psi)) = \text{profile}(\pi', f((v,\psi)))$.

By Theorem 3.21 and some technical work, we can show that there is a π such that $\text{profile}(\pi,(v,\psi)) \sqsubseteq \text{profile}(\pi', f((v,\psi)))$. As we saw when proving the other direction, we can construct from π a partial winning strategy π'' for $(P^\varphi, f((v,\psi)))$ such that $\text{profile}(\pi,(v,\psi)) = \text{profile}(\pi'', f((v,\psi)))$. From the definition of ptype() it follows that $\text{profile}(\pi,(v,\psi)) = \text{profile}(\pi',(v,\psi))$. □

3.5 Proof of the Decomposition Theorem

With Theorem 3.25, we finally have the necessary tool to conclude the proof of the decomposition theorems from page 4.

Proof of Theorem 3.6. Fix some $\varphi \in \mathrm{CL}_P(L)$. Consider the model checking game $\mathrm{P}[M,\varphi]$ and the induced subgame

$\mathrm{P}[M_2, \varphi]$ with interface U. We can assume that $V(\mathrm{P}[M_2, \varphi]) \cap V(\mathrm{P}[M, \varphi]) = U$ by duplicating some nodes as necessary.

The game $\mathrm{P}[M_2, \varphi]$ is simulated by P^φ, constructed as described in Theorem 3.25. By Lemma 3.20, we can replace $\mathrm{P}[M_2, \varphi]$ by P^φ (by properly adapting the edges) without changing the winner on (v, φ). Since the construction of P^φ only depends on the types of the nodes in X, we will get the same game P^φ if we start the construction with M_2'.

Let (v, w) be an edge from $M_1 \setminus X$ to $M_2 \setminus X$ and let $w' \in M_2'$ be the node chosen as the replacement for w. Because $\mathrm{tp}_{\{\varphi\},\overline{P}}(M_2, w, \overline{X})$ determines $\mathrm{ptype}_{\mathrm{P}[M_2,\varphi]}((w, \varphi))$ by Corollary 3.22 and we have

$$\mathrm{tp}_{\{\varphi\},\overline{P}}(M_2, w, \overline{X}) \subseteq \mathrm{tp}_{L,\overline{P}}(M_2, w, \overline{X}),$$

it follows that

$$\mathrm{ptype}_{\mathrm{P}[M_2,\varphi]}((w, \varphi)) = \mathrm{ptype}_{\mathrm{P}[M_2',\varphi]}((w', \varphi)).$$

So in the simulation, the edge will point to the same node no matter if we started with M_2 or M_2'. □

Proof of Theorem 3.7. The first part is essentially a different way of stating Theorem 3.6 which follows immediately with the same argument as in the previous proof.

Note that we may assume without loss of generality that $X \cap Y = \emptyset$. If this is not the case, then we have $x_i = y_j$ for some $x_i \in X, y_j \in Y$ and the propositional variables $X_i \in P$ and $Y_j \in Q$ will be interchangeable.

Set $L' := \mathrm{CL}_Q(L)$. Theorem 3.6 states that $\mathrm{tp}_{L',\emptyset}(M, v, \emptyset)$ is invariant under $\mathrm{CL}_\emptyset(L')$-equivalent directed separations for all $v \in M_1$. All we need to show is that the requirements listed in Theorem 3.7 specify the directed separation (M_1, M_2) up to $\mathrm{CL}_\emptyset(L')$-equivalence.

For all nodes $w \in M_2$, the set $\mathrm{tp}_{L',\overline{P}}(M_2, w, \overline{X})$ can be computed from $\mathrm{tp}_{L,\overline{P}}(M_2, w, \overline{X})$; a propositional variable $Y_i \in Q$ corresponding to a node $y_i \in Y$ is always false in M_2. From this we can easily compute $\mathrm{tp}_{L',\emptyset}(M_2, w, \emptyset)$ by forgetting about $\overline{P}$.

The computability in the above argument follows from the observation that all sets involved are finite in size and the model checking for L_μ is decidable.

For the second part, let $\varphi \in \mathrm{CL}_Q(L)$. We want to decide whether $M, w \models \varphi$. By the first part, we already know the sets $\mathrm{tp}_{L,\overline{Q}}(M, x_i, \overline{Y})$ for all $x_i \in X$. Consider the model checking game $\mathrm{P}[M, \varphi]$. In this game, the nodes of the form (v, Y_i) with $v \in M_1$ are always losing because $Y_i \in Q$ is never true in M_2. It follows that the subgame $\mathrm{P}[M_2, \varphi]$ is isomorphic to $\mathrm{P}[M_2, \varphi']$, where φ' is constructed from φ by replacing all $Y_i \in Q$ by $\bot$. Note that $\varphi' \in \mathrm{CL}_P(L)$, so we know all optimal partial strategies for $(\mathrm{P}[M_2, \varphi'], (w, \varphi'))$ because we know $\mathrm{tp}_{L,\overline{P}}(M_2, w, \overline{X})$. It follows that the winner is determined by the remaining sets given in the theorem. □

4. FPT Algorithms for L_μ Model Checking

In this section we derive two algorithmic applications of Theorem 3.7. More precisely, we show that L_μ-model-checking is fixed-parameter tractable on any class of structures of bounded Kelly-width or bounded DAG-width.

Before proving our results, we develop some algorithmic concepts common to both proofs. We first need an algorithmic version of L-equivalence.

In the following, let σ be a signature, $\overline{P}$ be a sequence of propositional symbols of the appropriate length disjoint from σ and let $L \subseteq L_\mu[\sigma \cup P]$.

4.1 Weak Separations

Definition 4.1 Let M be a σ-structure. A pair (M_1, M_2) of induced substructures is a *weak* directed σ-separation of M with interface $\overline{X} = (x_1, \ldots, x_k)$ if

- $V(M) = V(M_1) \cup V(M_2)$,
- $X = \{x_1, \ldots, x_k\} \subseteq V(M_1) \cap V(M_2)$,
- there are no edges from $M_2 \setminus (V(M_1) \cap V(M_2))$ to $M_1 \setminus (V(M_1) \cap V(M_2))$,
- there are no edges from $(V(M_1) \cap V(M_2)) \setminus X$ to $V(M_1) \setminus V(M_2)$. ⊣

Clearly, every directed separation is a weak directed separation. Weak separations can be transformed into proper separations by duplicating the nodes outside of the interface X. This gives us the following theorem.

Theorem 4.2 *Let (M_1, M_2) be a weak directed separation of M with interface $\overline{X}$. Then there exists a structure M' and a directed separation (M_1', M_2') of M' with the same interface $\overline{X}$ and isomorphisms $\pi_1 : M_1 \to M_1'$, $\pi_2 : M_2 \to M_2'$ which are the identity on $\overline{X}$ such that*

$$\mathrm{tp}_{L,\overline{P}}(M, v, \overline{X}) = \mathrm{tp}_{L,\overline{P}}(M', \pi_i(v), \overline{X})$$

for all $i \in \{1, 2\}$ and $v \in V(M_i)$.

Proof. For $i \in \{1, 2\}$, define π_i and M' as

$$\pi_i(v) := \begin{cases} v & \text{if } v \in X \\ (i, v) & \text{if } v \notin X \end{cases}$$
$$V(M') := \pi_1(V(M_1)) \cup \pi_2(V(M_2))$$
$$E(M') := E_1 \cup E_2 \cup E_3,$$

where, for $i \in \{1, 2\}$,

$$E_i := \{(\pi_i(v), \pi_i(w)) \mid (v, w) \in E(M_i)\}$$
$$E_3 := \{(\pi_1(v), \pi_2(w)) \mid (v, w) \in E(M) \cap (V(M_1) \times V(M_2))\}.$$

The substructures M_1', M_2' of M' are induced by the sets

$$V(M_i') := \pi_i(V(M_i)).$$

Clearly, π_i is an isomorphism between M_i and M_i' and the identity on $\overline{X}$. We also have that (M_1', M_2') is a directed separation of M'.

It is easy to verify that the colored structures $\partial_{\overline{P}}(M, \overline{X})$ and $\partial_{\overline{P}}(M', \overline{X})$ are bisimilar. Bisimilarity of these structures implies

$$\mathrm{tp}_{L,\overline{P}}(M, v, \overline{X}) = \mathrm{tp}_{L,\overline{P}}(M', \pi_i(v), \overline{X})$$

for all $i \in \{1, 2\}$ and $v \in V(M_i)$. □

Having isomorphisms means that

$$\mathrm{tp}_{L,\overline{P}}(M_i, v, \overline{X}) = \mathrm{tp}_{L,\overline{P}}(M_i', \pi_i(v), \overline{X})$$

for all $v \in V(M_i)$.

This and the previous theorem imply that Theorem 3.6 and with appropriate wording also Theorem 3.7 hold for weak directed separations as well. Let us restate the last theorem in its more general form.

Theorem 4.3 (Corollary of theorems 3.7 and 4.2) *Let $\overline{P}$, $\overline{Q}$ be sequences of proposition symbols such that $\sigma \cap P = \sigma \cap Q = P \cap Q = \emptyset$.*

Let $L \subseteq L_\mu[\sigma \cup P]$ and M be a structure with a weak directed σ-separation (M_1, M_2) with interface $\overline{X}$. Let $\overline{Y} \in ((V(M_1) \setminus V(M_2)) \cup X)^{|Q|}$ be a tuple.

For all $v \in M_1$, the set $\mathrm{tp}_{L,\overline{Q}}(M, v, \overline{Y})$ depends only on

- *M_1 and $\overline{Q}$ and*
- *$\left\{(x_i, \mathrm{tp}_{L,\overline{P}}(M_2, x_i, \overline{X})) \;\middle|\; x_i \in X\right\}$ and*
- *$\left\{(v, \mathrm{tp}_{L,\overline{P}}(M_2, w, \overline{X})) \mid (v, w) \in E(M) \cap (M_1 \times M_2)\right\}$.*

Provided L is finite, $\mathrm{tp}_{L,\overline{Q}}(M, v, \overline{Y})$ can be computed from these sets.

Furthermore, for every $w \in M_2$, the set $\mathrm{tp}_{L,\overline{Q}}(M, w, \overline{Y})$ depends only on the above sets and on $\mathrm{tp}_{L,\overline{P}}(M_2, w, \overline{X})$ and can be computed from these sets if L is finite.

The only difference of this statement to Theorem 3.7 is that we only require a weak separation and that the tuple $\overline{Y}$ should not contain a node $v \in V(M_1) \cap V(M_2)$ which is not part of the interface. This last requirement is necessary because otherwise we would have a color in M_2 where there was none before, and the types of M_2 with respect to $\overline{X}$ do not carry this information.

4.2 Kelly-Width

First we consider Kelly-width. We follow the notation and definitions given in [13]. For a directed acyclic graph (DAG), we write $\preceq$ for the reflexive and transitive closure of the edge relation.

Let G be a digraph. A set $W \subseteq V(G)$ *guards* $X \subseteq V(G)$ if $W \cap X = \emptyset$ and for all $(u, v) \in E(G)$ with $u \in X$, we have $v \in X \cup W$. For any set $W \subseteq V(G)$ we write $\mathrm{guard}(W)$ for the minimal set $U \subseteq V(G)$ guarding W.

Definition 4.4 A *Kelly decomposition* of a digraph G is a triple $\mathcal{D} := (D, \beta, \gamma)$, where $\beta, \gamma : V(D) \to 2^{V(G)}$ such that

- D is a DAG and $(\beta(t))_{t \in V(D)}$ partitions $V(G)$,
- for all $t \in V(D)$, $\gamma(t)$ guards $\mathcal{B}^\downarrow_t := \bigcup_{t' \succeq t} \beta(t')$ and
- for all $s \in V(D)$ there is a linear order $\leq_t$ on its children so that the children can be ordered as $t_1, \ldots, t_p$ such that for all $1 \leq i \leq p$, $\gamma(t_i) \subseteq \beta(s) \cup \gamma(s) \cup \bigcup_{j<i} \mathcal{B}^\downarrow_{t_j}$. Similarly, there is a linear order on the roots such that $\gamma(r_i) \subseteq \bigcup_{j<i} \mathcal{B}^\downarrow_{r_j}$.

The *width* of $\mathcal{D}$ is $\max\{|\beta(t) \cup \gamma(t)| \mid t \in V(D)\}$. The *Kelly-width* of G is the minimal width of any of its Kelly decompositions. ⊣

Theorem 4.5 *There exists an algorithm that solves the L_μ model checking problem in time $O(f(k + |\varphi|) \cdot n^c)$ for some computable function f and some constant c, where k is the Kelly-width and n the size of the input structure, provided a Kelly decomposition of width at most k is given as part of the input.*

Let G be a structure of Kelly-width k and $v \in V(G)$. It is easily seen that, by increasing the Kelly-width by one, we can always take a Kelly decomposition of G of width $\leq k+1$ which has only one root and this root contains v. We call such a Kelly decomposition *rooted at* v.

Proof of Theorem 4.5. Let G, v be a structure and $\overline{P}$ be a sequence of k fresh proposition symbols. We pick an arbitrary linear order of $V(G)$ in order to define interfaces consistently.

Let $\mathcal{D} = (D, \beta, \gamma)$ be a Kelly decomposition of width k of G rooted at v and $\varphi \in L_\mu$. We set $L := \{\varphi\}$.

Let us introduce the abbreviation

$$\mathcal{T}(A, B) := \left\{(v, \mathrm{tp}_{L,\overline{P}}(A, v, B)) \mid v \in A\right\}.$$

We will inductively compute the types $\mathcal{T}(\mathcal{B}^\downarrow_t \cup \gamma(t), \gamma(t))$ for all $t \in V(D)$. For the leaves, these sets can be computed by brute force. Let $t \in V(D)$ be a node with children $s_1, \ldots, s_l$ and assume that we already know the above types for all s_i.

Let

$$\delta_i := \bigcup_{j \leq i} (\gamma(s_j) \cap (\beta(t) \cup \gamma(t)))$$
$$\delta'_i := \delta_i \cup \bigcup_{j \leq i} \mathcal{B}^\downarrow_{s_j}.$$

We inductively compute the types $\mathcal{T}(\delta'_i, \delta_i)$. For $i = 1$ we already know these types by assumption. Assume $i > 1$.

We want to construct weak directed separations. Note that by assumption we know $\mathcal{T}(\mathcal{B}^\downarrow_{s_i} \cup \gamma(s_i), \gamma(s_i))$. We now first compute

$$\mathcal{T}(\mathcal{B}^\downarrow_{s_i} \cup \gamma(s_i) \cup \delta_{i-1}, \gamma(s_i) \cup \delta_{i-1}).$$

This is possible because $(\delta_{i-1}, \mathcal{B}^\downarrow_{s_i} \cup \gamma(s_i))$ is a directed separation with interface $\delta_{i-1} \cap \gamma(s_i)$.

Next, we observe that $(\mathcal{B}^\downarrow_{s_i} \cup \gamma(s_i) \cup \delta_{i-1}, \delta'_{i-1})$ is a weak directed separation with interface $\delta_{i-1} \cup (\gamma(s_i) \cap \delta_{i-1})$. Thus Theorem 4.3 allows us to compute $\mathcal{T}(\delta'_i, \delta_i)$.

After the last step we still need to compute $\mathcal{T}(\mathcal{B}^\downarrow_t \cup \gamma(t), \gamma(t))$ for the parent t. The pair $(\beta(t) \cup \gamma(t), \delta'_l)$ is a directed separation with interface δ_l, which is the final piece to the proof.

The runtime of this algorithm is $O(f(k + |\varphi|) \cdot n^3)$ for a function f because $|V(D)| \leq |V(M)|$, and we consider every element $t \in V(D)$ at most once. Every computation of $\mathcal{T}(\mathcal{B}^\downarrow_t \cup \gamma(t), \gamma(t))$ requires time at most linear in $V(D)$ because t has at most that many successors and at most quadratic in $V(M)$ because all sets involved are of size linear in $V(M)$. □

4.3 DAG-width

Definition 4.6 A *DAG decomposition* ([2]) of a digraph G is a pair $\mathcal{D} := (D, (X_d)_{d \in D})$ such that

- D is a DAG,
- $\bigcup_{d \in D} X_d = V(G)$,
- For all $d \preceq d' \preceq d''$, $X_d \cap X_{d''} \subseteq X_{d'}$,
- for all edges $(d, d') \in E(D)$, $X_d \cap X_{d'}$ guards $X_{\succeq d'} \cap X_d$, where $X_{\succeq d'} = \bigcup_{d' \preceq d''} X_{d''}$.

The *width* of $\mathcal{D}$ is $\max\{|X_d| \mid d \in V(D)\}$. The *DAG-width* of G is the minimal width of any of its DAG decompositions. ⊣

Theorem 4.7 *There exists an algorithm that solves the L_μ model checking problem in time $O(f(k + |\varphi|) \cdot n^c)$ for some computable function f and some constant c, where k is the DAG-width and n the size of the input structure, provided a DAG decomposition of width at most k is given as part of the input.*

Proof. Let G, v_0 be a structure and $(D, (X_d)_{d \in V(D)})$ be a nice DAG decomposition of G. That means (see [2])

1. D has a unique source.
2. Every $d \in V(D)$ has at most two successors.
3. For $d_0, d_1, d_2 \in V(D)$, if d_1, d_2 are two successors of d_0, then $X_{d_0} = X_{d_1} = X_{d_2}$.
4. For $d_0, d_1 \in V(D)$, if d_1 is the unique successor of d_0, then $|(X_{d_0} \setminus X_{d_1}) \cup (X_{d_1} \setminus X_{d_0})| = 1$.

We set $L := \{\varphi\}$. As in the proof for bounded Kelly width, we fix an arbitrary linear order $<$ on $V(G)$ so that we can consistently map nodes to the proposition symbols P_i occurring in the types.

During the run of the algorithm, we fill a table $\mathcal{T}$ with indices from the set $\{(v, d) \in V(G) \times V(D) \mid v \in X_{\succeq d}\}$ and entries that are elements of $\mathcal{T}_L(\overline{P})$. We will write to every index in this table at most once during the run, and we will always make sure to write $\mathcal{T}(v, d) = \mathrm{tp}_{L,\overline{P}}(G[X_{\succeq d}], v, X_d)$. If d is the root of D, then $\mathcal{T}(v_0, d)$ will answer the model checking problem $G, v_0 \models \varphi$.

Clearly, we can fill in all values for the leaves d immediately by computing them directly.

Let $d \in V(D)$. If d has two successors d_0, d_1, then we have $X_d = X_{d_0} = X_{d_1}$. Then $(G[X_{\succeq d_0}], G[X_{\succeq d_1}])$ is a weak directed separation with interface X_d. Because we already know $\mathrm{tp}_{L,\overline{P}}(G[X_{\succeq d_i}], v, X_d)$ for all v and $i \in \{0, 1\}$, Theorem 4.3 allows us to compute the types $\mathrm{tp}_{L,\overline{P}}(G[X_{\succeq d}], v, X_d)$.

The other case is that d has a unique successor d_0. Let $X_d = \{v_1, \ldots, v_k\}$ be ordered by the global linear order $<$. If $X_{d_0} \setminus X_d = \{v_i\}$, then for all $v \in X_{\succeq X_d}$ we set

$$\mathcal{T}(v, d) = \{\mathrm{shrink}_i(\psi) \mid \psi \in \mathcal{T}(v, d_0) \text{ and } P_i \text{ does not occur in } \psi\},$$

where $\mathrm{shrink}_i(\psi)$ is a function defined inductively over the structure of formulas with the base case

$$\mathrm{shrink}_i(P_j) := \begin{cases} P_j & \text{if } j < i \\ P_{j-1} & \text{if } j > i. \end{cases}$$

In other words, $\mathrm{shrink}_i(\psi)$ is the formula ψ with all P_j with $j > i$ replaced by P_{j-1} in order to not leave a hole. It is easy to check that we have

$$\{\mathrm{shrink}_i(\psi) \mid \psi \in \mathcal{T}(v, d_0) \text{ and } P_i \text{ does not occur in } \psi\} = \mathrm{tp}_{L,\overline{P}}(G[X_{\succeq d}], v, X_d).$$

The last case is $X_d \setminus X_{d_0} = \{v_i\}$. Because $X_d \cap X_{d_0}$ guards $X_{\succeq d_0} \setminus X_d$, all edges $(v, v_i) \in G[X_{\succeq d}]$ satisfy $v \in X_d$.

This means we have in fact a directed separation $(G[X_d], G[X_{\succeq d_0}])$ with interface X_{d_0}. We know $G[X_d]$ (its size is small), and we know the types $\mathrm{tp}_{L,\overline{P}}(G[X_{\succeq d_0}], v, X_{d_0})$ for all $v \in X_{\succeq X_{d_0}}$.

By Theorem 3.7, this is all the information we need to compute $\mathrm{tp}_{L,\overline{P}}(G[X_{\succeq d}], v, X_d)$ for all $v \in X_{\succeq X_d}$, which completes the algorithm and the proof. □

5. Conclusion

We proved a decomposition theorem for the modal μ-calculus. This theorem, interesting already all by itself, further allowed us to prove fixed-parameter tractability results for the L_μ model checking problem on classes of bounded Kelly-width or bounded DAG-width.

Open questions arise from the diverse number of decompositions for directed graphs. In particular, we think it could be promising to analyze D-width [21] and directed tree-width [14].

References

[1] J. Barát. Directed path-width and monotonicity in digraph searching. *Graphs and Combinatorics*, 22(2):161–172, 2006.

[2] D. Berwanger, A. Dawar, P. Hunter, S. Kreutzer, and J. Obdrzálek. The dag-width of directed graphs. *J. Comb. Theory, Ser. B*, 102(4):900–923, 2012.

[3] D. Berwanger, E. Grädel, Ł. Kaiser, and R. Rabinovich. Entanglement and the Complexity of Directed Graphs. *Theoretical Computer Science*, 463(0):2–25, 2012. ISSN 0304-3975. . URL `http://logic.rwth-aachen.de/~rabinovich/entangle.pdf`.

[4] M. Bojanczyk, C. Dittmann, and S. Kreutzer. Decomposition theorems and model-checking for the modal μ-calculus. arXiv:????.????, 2014.

[5] J. Bradfield and C. Stirling. Modal mu-calculi. In P. Blackburn, J. V. Benthem, and F. Wolter, editors, *Handbook of Modal Logic*, volume 3 of *Studies in Logic and Practical Reasoning*, pages 721 – 756. Elsevier, 2007. . URL `http://www.sciencedirect.com/science/article/pii/S1570246407800152`.

[6] B. Courcelle. Graph rewriting: An algebraic and logic approach. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume 2, pages 194 – 242. Elsevier, 1990.

[7] B. Courcelle, J. A. Makowsky, and U. Rotics. Linear time solvable optimization problems on graphs of bounded clique-width. *Theory of Computing Systems*, 33(2):125–150, 2000.

[8] R. G. Downey and M. R. Fellows. *Parameterized Complexity*. Springer, 1998.

[9] S. Feferman and R. L. Vaught. The first-order properties of algebraic systems. *Fundamenta Mathematicae*, 47:57–103, 1959.

[10] J. Flum and M. Grohe. *Parameterized Complexity Theory*. Springer, 2006. ISBN 3-54-029952-1.

[11] E. Grädel, W. Thomas, and T. Wilke, editors. *Automata, Logics, and Infinite Games: A Guide to Current Research*, volume 2500 of *LNCS*, 2002. Springer. ISBN 3-540-00388-6.

[12] M. Grohe and S. Kreutzer. Methods for algorithmic meta-theorems. *Contemporary Mathematics*, 588, American Mathematical Society 2011.

[13] P. Hunter and S. Kreutzer. Digraph measures: Kelly decompositions, games, and orderings. *Theor. Comput. Sci.*, 399 (3):206–219, 2008.

[14] T. Johnson, N. Robertson, P. D. Seymour, and R. Thomas. Directed tree-width. *J. Comb. Theory Ser. B*, 82(1):138–154, May 2001. ISSN 0095-8956. .

[15] M. Jurdziński. Deciding the winner in parity games is in UP ∩ co-UP. *Inf. Process. Lett.*, 68(3):119–124, 1998.

[16] M. M. Kanté. The rank-width of directed graphs. *CoRR*, abs/0709.1433, 2007.

[17] K. Kawarabayashi and S. Kreutzer. An excluded grid theorem for digraphs with forbidden minors. In *ACM/SIAM Symposium on Discrete Algorithms (SODA)*, 2014.

[18] J. A. Makowsky. Algorithmic uses of the feferman-vaught theorem. *Ann. Pure Appl. Logic*, 126(1-3):159–213, 2004.

[19] J. Obdrzálek. Fast mu-calculus model checking when tree-width is bounded. In *CAV*, pages 80–92, 2003.

[20] J. Obdrzálek. Clique-width and parity games. In *Computer Science Logic (CSL)*, pages 54–68, 2007.

[21] M. A. Safari. D-width: A more natural measure for directed tree width. In J. Jedrzejowicz and A. Szepietowski, editors, *MFCS*, volume 3618 of *Lecture Notes in Computer Science*, pages 745–756. Springer, 2005. ISBN 3-540-28702-7.

[22] R. S. Streett and E. A. Emerson. An automata theoretic decision procedure for the propositional mu-calculus. *Inf. Comput.*, 81(3):249–264, 1989.