Guarded Logics: Algorithms and Bisimulation

Von der Fakultät für Mathematik, Informatik und Naturwissenschaften der Rheinisch-Westfälischen Technischen Hochschule Aachen zur Erlangung des akademischen Grades eines Doktors der Naturwissenschaften genehmigte Dissertation

vorgelegt von
Diplom-Informatiker Colin Hirsch
aus Wien, Österreich

Berichter:
Universitätsprofessor Dr. Erich Grädel
Universitätsprofessor Dr. Wolfgang Thomas

Tag der mündlichen Prüfung: 17.7.2002

Thank You

A.B., D.B., E.G., E.R., I.★., M.K., M.O., M.T., S.K., S.T., W.T., …

“Waterproof Cookies”

```
8.82oz / 250g margerine
8.82oz / 250g wheat flour
7.05oz / 200g sugar
3.53oz / 100g coconut rasps
1.76oz /  50g chocolate rasps
1.76oz /  50g candy sugar
      2-3     eggs
```

Separate the egg whites and whisk until solid. Make pastry out of other ingredients, including egg yolks. Finally add egg whites. Prepare baking tray and form cookies approx. 2 in. in diameter, 1/4 in. thick. Bake in oven at 392° F / 200° C until golden brown. Comments to `mail@cohi.at`.

Contents

Chapter 1

Introduction

For many practical applications of logic-based methods there is a requirement to balance expressive power against computational tractability. The aim can be to find the algorithmically least expensive framework that is able to express a class of properties. An alternative approach is to find the — ideally — most powerful logic with respect to syntactic flexibility and/or semantic discerning power inside of given complexity classes. This search can be conducted at various levels; where in one case PSPACE may already be considered as too hard for practice, others will be even happy to establish decidability.

Two traditional players found in fields like mathematical logic, database theory or artificial intelligence are first-order predicate logic on the dark, and modal logics on the sunny side of the decidability/undecidability border. Both identifying decidable sub-classes of first-order logic, and extending modal logic to larger, but nevertheless efficiently solvable languages has been a preeminent goal of research [9, 60, 27].

In particular the robust decidability of many modal style logics has spurred a whole flock of results that attempt to isolate and transfer the nice algorithmic properties of modal logics as high up the ladder as possible. One candidate to explain this nice modal behaviour is the restriction to structures with only unary and binary relations. Another explanation stems from the observation that large classes of modal logics can be embedded into 2-variable first-order logic. However, neither approach could explain for example why the powerful fixed-point extension to modal logics, the modal μ-calculus, is decidable.

It was later conjectured by Vardi, that the tree model property is the main reason for the decidability of many modal style logics [60]. The guarded fragment of first-order logic, introduced by Andréka, van Benthem, and Németi [1], was a successful attempt to transfer this key property of modal, temporal, and description logics to a larger fragment of predicate logic. As pointed out in [24], a generalisation of the tree model property indeed explains the similarly robust decidability of guarded logics. It can also be seen as a strong indication that guarded logics are an extension to the modal framework that retains the essence of modal logics. Besides decidability this includes the finite model property, invariance under an appropriate variant of bisimulation, and other nice model theoretic properties [1, 24]. This becomes even more evident when regarding the respective fixed-point extensions, which turned out to be decidable too [25].

These and further results, in particular with respect to complexity, of recent research into guarded logics indicate that the guarded fragment of first-order logic, and its relatives, provide an interesting domain of logics, with a good balance between expressive power and algorithmic manageability. The complexity of the satisfiability problem for many guarded logics is not higher than for mild extensions of basic modal logic, as long as the width of the vocabulary is fixed.

From a model theoretic perspective, modal logics — in a broad sense — are characterised by their invariance under bisimulation, i.e. they cannot distinguish between bisimilar structures. Moreover, invariance under bisimulation happens to be the root of the tree model property, which in turn paves the way towards the use of automata theory. The eminent role that bisimulation plays in the domain of modal logics is taken up by the similar, but much more wide-ranging and finer notion of equivalence induced by guarded bisimulation in the guarded domain. This correspondence has enabled various results about modal logics to be transferred up into the guarded world, including the characterisation of basic modal logic in terms of bisimulation invariant first-order logic. Characterisation theorems of this type have a strong tradition in the field. Besides bisimulation invariance, there is the related notion of bisimulation safety that generalises the same idea to the binary accessibility relations on labelled graphs.

The goal of this this work is to gain greater insight into the correspondence between the modal world and the guarded world, but also the previously less touted failure of this correspondence. Depending on the property under scrutiny, the transfer can be anything between blatantly obvious, conceptually simple but technically involved, or absolutely impossible. However, many of the hard cases can be partially repaired by selectively abandoning some of the additional features that guarded logic possesses over modal logic. Indeed the existence of many variations of so-called action guarded logics shows that guarded logics are not one, but a whole family of answers to the question of how modal style logics can be generalised.

Chapter 2 fixes the terminology of this work and repeats basic definitions and properties of predicate and modal logics, including the key notion of bisimulation and the related unravelling of graph structures. The notions of bisimulation invariance, and bisimulation safety are introduced. In Chapter 3, the same is done for guarded logics. A selection of previously unknown results about basic properties of guarded logics is shown along the way. The guarded relational algebra is introduced in the context of characterising the bisimulation safe fragment of first-order logic.

Theorem. *Every first-order formula that is safe for guarded bisimulation is equivalent to a guarded relational algebra term.*

Guarded second-order logic is defined and discussed, in particular concerning the relationship to monadic second-order logic on certain structures. Such a candidate in the guarded world for the position that monadic second-order logic occupies in the modal domain was previously unknown. Two other natural second-order extensions to the guarded fragment are shown to be equivalent to the guarded second-order logic, strengthening the claim on this position.

Theorem. *All variants of second-order logic with either or both of the first-order or second-order quantifiers restricted to guarded objects are equivalent.*

Chapter 4 makes precise the correspondence between modal and guarded bisimulations. A first analysis makes the comparison over modal logic's native structures, the class of transition systems, Kripke structures or labelled graphs. The greater expressive power of guarded logics is measured against modal bisimulation on so-called atomic expansions of graphs, that encode binary atomic types in new edge relations.

Theorem. *Guarded bisimulation on graphs is equivalent to modal bisimulation on atomic expansions.*

A game theoretic characterisation of guarded bisimulation through an appropriately restricted form of the Ehrenfeucht-Fraïssé game, producing a kind of guarded game, is given. This enables an encoding of arbitrary relational structures $\mathfrak{B}$ as graphs $\mathfrak{K}(\mathfrak{B})$ and trees $\mathfrak{T}(\mathfrak{B})$. A bisimulation closed class of first-order definable trees $\mathfrak{T}$ allows a converse transformation back into relational structures $\mathfrak{D}(\mathfrak{T})$. The interesting property is that guarded bisimulation equivalence on the original structures is retained in form of modal bisimulation on the graph encodings, and vice versa. This method is a powerful tool that will show up in several places, and in several more or less extended or restricted guises.

Theorem. *Two structures $\mathfrak{B}$ and $\mathfrak{B}'$ are guarded bisimilar iff $\mathfrak{T}(\mathfrak{B})$ and $\mathfrak{T}(\mathfrak{B}')$ are bisimilar.*

Two consistent trees $\mathfrak{T}$ and $\mathfrak{T}'$ are bisimilar iff $\mathfrak{D}(\mathfrak{T})$ and $\mathfrak{D}(\mathfrak{T}')$ are guarded bisimilar.

Chapter 5 is centred around a tableau algorithm for the guarded fragment. In modal logics, often in the form of description logics, tableau algorithms constitute a well studied technique of finding decidability proofs that, furthermore, are amiable to efficient implementation [46, 32, 18, 43, 42]. An introduction is given in form of an algorithm for modal logic, and modal logic with universal quantification. A more complex tableau algorithm for the guarded fragment is developed. A recent implementation of the algorithm suggests that the additional goal of finding a practically efficient decision procedure was reached with the given tableau algorithm [37].

Theorem. *The tableau algorithm is an effective (efficient) decision procedure for the guarded fragment.*

The tableau also serves as basis of an alternative, elementary proof of the finite model property of the guarded fragment, which was originally established building on strong model-theoretic results. Compared to infinite graph encodings of relational structures, the finite case possesses certain problems that are highlighted by the exhibited procedure and give a first insight into why certain modal features do not (easily) generalise.

Theorem. *The guarded fragment has the finite model property.*

Chapter 6 gives an overview on one take of action guarded logics. Several restrictive features of modal logics that were dropped by the guarded fragment are made selectively available, namely the separation of the vocabulary into action and state relations, and the directional nature and forgetfulness of actions. The characterisation theorems that compare the bisimulation invariant fragment of first-order logic to the appropriate action guarded logic are shown to hold true for a broad class of logics.

Theorem. *Every first-order formula that is invariant under a version of action guarded bisimulation is equivalent to a formula in the corresponding action guarded fragment.*

However for bisimulation safety the characterisation results for bisimulation invariant first-order logic break down due to the format of guarded quantification significantly changing certain behaviours of the corresponding bisimulation. The modal result can only be transferred to a severely restricted form of action guarded logic. Appropriate versions of guarded relational algebra and propositional dynamic logic are introduced.

Theorem. *Every first-order formula that is safe for the weak action guarded fragment with directed actions is equivalent to an iteration free program of the guarded propositional dynamic logic.*

Also considered is the decidability of extensions of guarded logics by means of counting quantifiers and transitivity statements. In both cases the precise border of decidability between basic modal logic and the full guarded fragment is established. The latter case implicitly takes up modal logics augmented with so-called frame conditions, of which transitivity is a common case. The method used in the main proof modifies the graph encodings of relational structures to include a larger class of guarded objects. Decidability and the tree model property for the largest decidable class of action guarded logics with transitive relations is established.

Theorem. *The guarded fragment with transitive guard relations is decidable and has the tree model property.*

Chapter 7 takes up the manipulation of graph encodings of structures in order to find canonical representatives for bisimulation classes of structures. This method has several applications, e.g. in database systems or complexity theory. Two successful versions of efficient canonisation algorithms are developed. For one version of action guarded logic, the method used for modal logic can be generalised by means of an appropriate graph encoding of structures.

Theorem. *Guarded logic on graphs and the weaker action guarded fragment with directed actions allow efficient canonisation procedures.*

For full guarded logics on graphs this is not feasible; although the structures are restricted to width 2, the quantifier pattern requires an elaborate construction of canonical models. For guarded bisimulation it is conjectured that there is no efficient canonisation procedure. More interesting than the result are the surrounding detailed observations. The problems encountered here are strongly related to the conceptual difficulties of recreating a finite model from a finite tableau.

Chapter 8 finally continues where Chapter 4 left off. Equivalent transformations from guarded second-order logic over relational structures to monadic second-order logic over their encodings, and in the other direction from the modal μ-calculus to guarded fixed-point logic are given. The main result is a lifting of the Janin-Walukiewicz theorem by means of a reduction of the guarded equivalent. This is the main motivation for the introduction of guarded second-order logic.

Theorem. *Every guarded bisimulation invariant formula of guarded second-order logic is equivalent to a formula of guarded fixed-point logic.*

Of the known characterisation theorems of this type, only the characterisation of basic modal logic is known to survive as a theorem of finite model theory too. We consider a variant of monadic second-order logic that speaks about the unravellings of graphs. The Janin-Walukiewicz method is modified to give an automata-theoretic proof of the finite model theory version of the bisimulation invariance characterisation theorem for this logic. Actually the following stronger result is established.

Theorem. *Every formula of monadic second-order logic that is bisimulation invariant on the unravelling of finite graphs is already bisimulation invariant on the unravellings of all graphs.*

This first step is a motivation for further research into related questions concerning larger classes of formulae. At this point, both whether this can be extended to full monadic second-order, and if the guarded version can then be deduced, must be left as open questions.

Chapter 2

Logical Background

The following recalls a basic set of definitions from predicate logic and model theory. A couple of general results that will be required later on are cited in the context of the corresponding definitions. This includes making explicit a method of sorting atomic types, which is necessary to create deterministic algorithms that would otherwise use arbitrary choice.

Several generally known notions are stated informally without further elaboration. The aim is to present the notational conventions used in this work, so that the experienced reader may recognise familiar concepts, even if they appear in a different than the habitual guise.

The remainder of this chapter recalls the basics of modal logics, including transition systems, basic modal logic, modal fixed-point logic and modal bisimulation. This includes numerous definitions and results whose generalisation to the guarded world we are concerned with later.

2.1 First and Second-Order Logic

We use lower-case letters $x, y, z, \ldots$ for first-order variables and upper-case $X, Y, Z, \ldots$ for second-order, or set variables. Tuples of variables, relations or elements are denoted by bold-face letters $\boldsymbol{x}, \boldsymbol{y}, \boldsymbol{a}, \boldsymbol{b}, \boldsymbol{X}, \boldsymbol{Y}, \ldots$.

For sets of relation symbols τ we interchangeably use the terms vocabulary, language and signature. In this work only purely relational vocabularies are considered and $\mathfrak{A}$, $\mathfrak{B}$ will typically refer to relational τ-structures.

If L is a logic, we write $\mathrm{L}[\tau]$ to indicate the class of L-formulae that only use relation symbols from τ. If φ is a formula we denote by $\tau(\varphi)$ the set of relations that syntactically occur in φ.

An *atomic formula*, short *atom*, is a formula of the form $R(\boldsymbol{x})$ or $x = y$. If $R \in \tau$, then $R(\boldsymbol{x})$ is also called a τ-atom. A *literal* is either an atom or a negated atom.

Syntax of FO**.**

(i) Every atomic formula is in FO.
(ii) FO is closed under Boolean combinations.
(iii) If φ is a FO-formula, then so is $\exists x\varphi$.

We write $\varphi(\boldsymbol{x})$ to denote that the free variables of φ are among $\boldsymbol{x}$, resp. write free(φ) for the actual set of free variables of φ. In logics closed under Boolean combinations the universal quantifier $\forall$ can be defined in terms of the existential quantifier and the well-known duality $\forall x\varphi \equiv \neg\exists x\neg\varphi$.

Syntax of SO**.**

(i) Every FO-formula is a SO-formula.
(ii) SO is closed under Boolean combinations.
(iii) If φ is a SO-formula, then $\exists X\varphi$ and $\exists x\varphi$ are SO-formulae.

We will write $\varphi(\boldsymbol{X}, \boldsymbol{x})$ to denote that the free variables of φ are among $\boldsymbol{X}$ and $\boldsymbol{x}$. If φ is a SO-formula where all occurring second-order variables have arity 1, then φ is also a *monadic second-order* formula .

In the context of bisimulation and bisimulation invariant properties we will often deal with formulae with exactly one free first-order variable.

Note. We usually assume all formulae with — first or second-order — variables to be *well named*, i.e. every single variable either occurs only free, or it is bound exactly once. For example the first-order formula $R(xy) \wedge \exists x\forall zE(zx)$ is not well named, because x occurs both free and bound. If the number of available variables is unlimited, every formula can be transformed into an equivalent well named version by renaming the bound variables.

Another normal form that is often convenient for algorithmic purposes imposes a restriction on where negation may occur in a formula.

Definition 2.1.1. A formula of any logic is in *negation normal form*, denoted NNF, if negation occurs only in front of atomic statements.

It is well known that every first or second-order formula is equivalent to a formula in negation normal form. In fact, all further logics considered in this work allow application of De Morgan's law and dualisation of quantifiers and fixed-point operators, and hence allow equivalent transformation of formulae into NNF.

2.2 Definitions and Conventions

We denote the power set of a set A by $\mathcal{P}(A)$, and write $\mathrm{ar}(R)$ for the arity of a relation R.

Restrictions, mostly of functions, are expressed by the $|$ syntax. If $f : A \to B$ is a function with domain $\mathrm{dom}(f) = A$ and range B, and $A' \subseteq A$, we write $f|_{A'}$ for the function $f' : A' \to B$ defined by $f'(a) = f(a)$ for every $a \in A'$.

The arity of a tuple $\boldsymbol{a}$ is denoted by $|\boldsymbol{a}|$. To express that $\boldsymbol{a}$ consists of elements from a set A, we write $\boldsymbol{a} \in A$ for short instead of $\boldsymbol{a} \in A^{|\boldsymbol{a}|}$. We also use tuples $\boldsymbol{a}$ as sets and write $a \in \boldsymbol{a}$ to denote that, if $\boldsymbol{a} = (a_1, \dots, a_n)$, $a = a_i$ for some $1 \le i \le n$, and, similarly, $X \subseteq \boldsymbol{a}$ if for all $x \in X$ also $x \in \boldsymbol{a}$.

If $f : D \to R$ is a function, we also use f as function $f : \mathcal{P}(D) \to \mathcal{P}(R)$ and $f : D^n \to R^n$, $n \in \mathbb{N}$ defined in the canonical way as $f(\{d_1, \dots, d_n\}) = \{f(d_1), \dots, f(d_n)\}$ and $f((d_1, \dots, d_n)) = (f(d_1), \dots, f(d_n))$, respectively.

For a formula φ of any logic we denote by $\mathrm{cl}(\varphi)$ the *closure* of φ, i.e. the set consisting of φ and all (syntactic) subformulae of φ.

Definition 2.2.1. Let $\mathcal{C}$ be a class of structures. A *global relation* of arity n on $\mathcal{C}$ is a function R that assigns every structure $\mathfrak{C} \in \mathcal{C}$ an n-ary relation $R^{\mathfrak{C}}$ such that if for $\mathfrak{A}, \mathfrak{B} \in \mathcal{C}$ there is an isomorphism $\pi : \mathfrak{A} \to \mathfrak{B}$, then $R^{\mathfrak{B}} = \pi \circ R^{\mathfrak{A}}$.

Every n-ary formula $\varphi(\boldsymbol{x})$ that belongs to a reasonable logic, in this case: a logic that does not distinguish isomorphic structures, gives rise to an n-ary global relation. The relation $\varphi^{\mathfrak{A}}$ on $\mathfrak{A}$ is simply defined as the set of all tuples $\boldsymbol{a}$ that satisfy φ, i.e. $\varphi^{\mathfrak{A}} = \{\boldsymbol{a} \in \mathfrak{A} \,:\, \mathfrak{A} \models \varphi(\boldsymbol{a})\}$.

Types and Theories

Definition 2.2.2. Let $\mathfrak{A}$ be a τ-structure, $\boldsymbol{a} \in \mathfrak{A}$ and $\tau' \subseteq \tau$. The *atomic* τ'*-type* of $\boldsymbol{a}$ in $\mathfrak{A}$, denoted $\mathrm{atp}_{\mathfrak{A},\tau'}(\boldsymbol{a})$, is the set of all τ'-literals $\alpha(\boldsymbol{x})$ where $\mathfrak{A} \models \alpha(\boldsymbol{a})$.

If $\tau' = \tau$ is clear from the context, we simply speak of an atomic type. The *equality type* of a tuple only allows literals built with $=$.

The common definition of types in model theory allows arbitrarily large sets of parameters. For our purpose we can restrict the definition to only allow finite parameter tuples.

Definition 2.2.3. Let L be a logic. A set of formulae $\Phi \subseteq \mathrm{L}$ is *consistent* if Φ is satisfiable, i.e. if there is a structure $\mathfrak{A}$ and tuple $\boldsymbol{a} \in \mathfrak{A}$ such that $\mathfrak{A} \models \varphi(\boldsymbol{a})$ for all $\varphi \in \Phi$. Further, Φ is *maximal consistent*, if there is no $\Psi \subseteq \mathrm{L}$ that properly contains Φ and is consistent.

Definition 2.2.4. An L-type is a maximal consistent set of L-formulae:

Let $\mathfrak{A}$ be a τ-structure, $\boldsymbol{a} \in \mathfrak{A}$ and L a logic. The L-*type* of $\boldsymbol{a}$ in $\mathfrak{A}$, denoted $\mathrm{tp}_{\mathfrak{A}}(\boldsymbol{a})$, is the set of all L-formulae $\varphi(\boldsymbol{x})$ where $\mathfrak{A} \models \varphi(\boldsymbol{a})$.

If $\boldsymbol{b} \in \mathfrak{A}$ is a further tuple, the L-*type with parameters* $\boldsymbol{b}$ of $\boldsymbol{a}$ in $\mathfrak{A}$, denoted $\mathrm{tp}_{\mathfrak{A}}(\boldsymbol{a}; \boldsymbol{b})$, is the set of all L-formulae $\varphi(\boldsymbol{x}, \boldsymbol{y})$ where $\mathfrak{A} \models \varphi(\boldsymbol{a}, \boldsymbol{b})$.

A structure $\mathfrak{A}$ *realises* a type p, if there is a tuple $\boldsymbol{a} \in \mathfrak{A}$ with $\mathrm{tp}_{\mathfrak{A}}(\boldsymbol{a}) = p$.

Definition 2.2.5. Let $\mathfrak{A}$ be a τ-structure and L a logic. The L-*theory* of $\mathfrak{A}$ is the L-type of the empty tuple in $\mathfrak{A}$.

Two structures are L-equivalent if they satisfy the same set of L-formulae, i.e. they share the same L-theories. In the line of the ubiquitous term of elementary equivalence, that is equivalence for first-order logic, we recall the following notion of an elementary extension.

Definition 2.2.6. Let $\mathfrak{A}$ be a τ-structure. A τ-structure $\mathfrak{B} \supseteq \mathfrak{A}$ is an *elementary extension* of $\mathfrak{A}$, denoted $\mathfrak{A} \prec \mathfrak{B}$, if $\mathfrak{A} \models \varphi(\boldsymbol{a}) \iff \mathfrak{B} \models \varphi(\boldsymbol{a})$ for every $\boldsymbol{a} \in \mathfrak{A}$ and every $\varphi \in FO[\tau]$.

Note that the above definition includes all τ-sentences; in particular the first-order theories of $\mathfrak{A}$ and $\mathfrak{B}$ are the same. We are interested in elementary extensions that satisfy a specific model-theoretic richness condition.

Definition 2.2.7. A relational structure $\mathfrak{A}$ is ω-saturated if every first-order type with finitely many parameters that is consistent with the first-order theory of $\mathfrak{A}$ is realised in $\mathfrak{A}$.

For a proof of the following theorem that will be required later see e.g. [38].

Theorem 2.2.8. *Every structure has an ω-saturated elementary extension.*

Orders on Atomic Types

Some of the constructive proofs in this work employ algorithms that can only be made deterministic, which will be necessary in some cases, in the presence of an order on types. This is a much weaker precondition than requiring an order on the structures themselves — an order on the vocabulary τ is sufficient to construct a possible required order on types.

At this point we handle the case of atomic types. Suppose that τ is finite and $\prec$ is a linear order on τ, and that the variables occurring in atomic types are from a fixed collection $X = \{x_1, x_2, x_3, \dots\}$. Define the function val that assigns every variable, every relation symbol, and the negation symbol $\neg$ an integer as follows.

1. $\mathrm{val}(\neg) = 0$.
2. $\mathrm{val}(x_i) = i$.
3. $\mathrm{val}(R) = -k$, if R is the k-th symbol in τ according to $\prec$.

We extend val to assign τ-atoms and literals a list of integers via

4. $\mathrm{val}(R(x_{i_1}, \dots, x_{i_j})) = \mathrm{val}(R), \mathrm{val}(x_{i_1}), \dots, \mathrm{val}(x_{i_j})$,
5. $\mathrm{val}(\neg R(x_{i_1}, \dots, x_{i_j})) = \mathrm{val}(\neg), \mathrm{val}(R), \mathrm{val}(x_{i_1}), \dots, \mathrm{val}(x_{i_j})$.

For sets of literals $\Gamma = \{\gamma_1, \dots, \gamma_k\}$ we purport that if Γ speaks about n elements, then the (free) variables of Γ are exactly $x_1, \dots, x_n$. If Γ is displayed such that $\mathrm{val}(\gamma_i) < \mathrm{val}(\gamma_j)$ for all $i < j$, then

6. $\mathrm{val}(\Gamma) = \mathrm{val}(\gamma_1), \dots, \mathrm{val}(\gamma_k)$.

The natural order on integers extends to these tuples via the usual lexicographical order, and yields the desired order on the set of all sets of τ-literals.

Note on (Finite) Model Theory

The model-theoretic properties considered in this work are mostly general statements made before the background of the class of all structures. At certain points we will be concerned with the corresponding statements in a finite model theory setting. In the sense of finite model theory, all statements are with respect to the class of finite models. Compared to the same result in classical model theory, a theorem in finite model theory has to get by with weaker preconditions, that only hold on finite models, but in return only has to prove that the statement holds on finite models too. This suggests that there is no a priori relationship between the classical and the finite model theory versions of the same theorem, however the finite case typically turns out to be harder.

2.3 Transition Systems and Bisimulation

Transition systems, labelled graphs or Kripke structures are three designations for structures whose universe is a set of states, nodes or possible worlds, labelled by unary predicates, and carrying binary labelled transition relations, or actions. We typically write $\mathfrak{K} = (V, (E_a)_{a\in A}, (P_b)_{b\in B})$ for a transition system with state set V based on a set B of atomic properties and a set A of actions. If $\mathfrak{K} \models E_a(v, w)$ for two nodes $v, w \in V$ we say that w is an E_a-successor, or short a-successor, of v, and likewise v is a predecessor of w. We regard trees as special transition systems.

Definition 2.3.1. A transition system $\mathfrak{T} = (V, (E_a)_{a\in A}, (P_b)_{b\in B})$ is a *tree* if the E_a are mutually disjoint and their union $E = \bigcup_{a\in A} E_a$ forms a directed tree in the graph theoretic sense.

Every tree has a root node with no predecessors. This corresponds to the definition used in computer science, whereas in mathematics a more liberal version that covers all graphs without cycles is usual.

In many applications, transition systems formalise some kind of dynamic behaviour. Considering only isomorphic or elementary equivalent transition systems as equivalent is therefore often too strict, since differing graphs can exhibit the same possible behaviours, e.g. in form of labelled paths. Several distinct attempts have been made in the past to capture an appropriately fine-grained notion of equivalence. Arguably the most important role is played by what is called bisimulation equivalence. Bisimulation between transition systems not only captures the natural notion of behavioural equivalence between processes, but also corresponds to the natural notion of Ehrenfeucht-Fraïssé equivalence associated with modal quantification.

Definition 2.3.2. A *bisimulation* between two transition systems

$$\mathfrak{K} = (V, (E_a)_{a\in A}, (P_b)_{b\in B}) \text{ and } \mathfrak{K}' = (V', (E'_a)_{a\in A}, (P'_b)_{b\in B})$$

is a non-empty relation $Z \subseteq V \times V'$, respecting the P_b in the sense that $v \in P_b$ iff $v' \in P'_b$, for all $b \in B$ and $(v, v') \in Z$, and satisfying the following back and forth conditions.

Forth. For all $(v, v') \in Z$, $a \in A$ and every w such that $(v, w) \in E_a$, there exists a w' such that $(v', w') \in E'_a$ and $(w, w') \in Z$.

Back. For all $(v, v') \in Z$, $a \in A$ and every w' such that $(v', w') \in E'_a$, there exists a w such that $(v, w) \in E_a$ and $(w, w') \in Z$.

A *total* bisimulation covers all nodes of either graph.

Two transition systems $\mathfrak{K}, \mathfrak{K}'$ with distinguished nodes $u \in \mathfrak{K}$ and $u' \in \mathfrak{K}'$ are *bisimilar*, denoted $\mathfrak{K}, u \sim \mathfrak{K}', u'$, if there is a bisimulation Z between them with $(u, u') \in Z$. We say that two trees are bisimilar if they are bisimilar at their roots λ and λ', and just write $\mathfrak{T} \sim \mathfrak{T}'$ for $\mathfrak{T}, \lambda \sim \mathfrak{T}', \lambda'$.

Remark. A crucial model theoretic feature of modal logics is the *tree model property*, the fact that every satisfiable formula is satisfiable in a tree. This is a well known and straightforward consequence of bisimulation invariance, since every transition system $(\mathfrak{K}, u)$ has a canonical bisimilar companion that is a tree.

This *unravelling* of $\mathfrak{K}$ from node u, denoted $\mathfrak{T}(\mathfrak{K}, u)$, is naturally induced on the set of all edge-labelled paths from u in $\mathfrak{K}$ with obvious interpretations of atomic propositions and accessibility relations. Since $\mathfrak{T}(\mathfrak{K}, u)$ is bisimilar to $(\mathfrak{K}, u)$, we can restrict attention to trees as long as bisimulation invariant properties are concerned. The advantage for all algorithmic issues in particular lies with the possibility of using tree automata, a rich and well-studied field, an example of which will be seen in Chapter 8.

Definition 2.3.3. The *unravelling* $\mathfrak{T}(\mathfrak{K}, u)$ of a transition system $\mathfrak{K}$ from node u is the tree of all paths through $\mathfrak{K}$ that start at u. More formally, given $\mathfrak{K} = (V, (E_a)_{a \in A}, (P_b)_{b \in B})$ with $u \in V$, the unravelling is $\mathfrak{T}(\mathfrak{K}, u) = (V^{\mathfrak{T}}, (E_a^{\mathfrak{T}})_{a \in A}, (P_b^{\mathfrak{T}})_{b \in B})$, where

(i) $V^{\mathfrak{T}}$ is the set of all sequences $\boldsymbol{v} = v_0 a_1 v_1 a_2 \cdots v_{r-1} a_r v_r$ where $v_i \in V$, $a_i \in A$ such that $v_0 = u$ and $(v_{i-1}, v_i) \in E_{a_i}$;

(ii) $P_b^{\mathfrak{T}}$ contains the sequences $v_0 a_1 v_1 a_2 \cdots v_{r-1} a_r v_r$ with $v_r \in P_b$;

(iii) $E_a^{\mathfrak{T}}$ contains the pairs $(v_0 a_1 v_1 \cdots v_{r-1}, v_0 a_1 v_1 \cdots v_{r-1} a_r v_r)$ in $V^{\mathfrak{T}} \times V^{\mathfrak{T}}$ where $(v_{r-1}, v_r) \in E_{a_r}$.

The *natural projection* $\pi \colon \mathfrak{T}(\mathfrak{K}, u) \to \mathfrak{K}, u$ sends $\boldsymbol{v} = v_0 a_1 v_1 a_2 \cdots v_{r-1} a_r v_r \in V^{\mathfrak{T}}$ to its last node $\pi(\boldsymbol{v}) = v_r \in V$.

Note that a canonical bisimulation between $\mathfrak{T}(\mathfrak{K}, u)$ and $\mathfrak{K}, u$ is induced by the graph of the natural projection.

Trees are special as bisimilar companions to arbitrary transition systems also because bisimulation equivalence between trees comes with an additional feature. A *two-way bisimulation* is one that not only satisfies the back and forth conditions for the transition relations E_a themselves, but also for their inverses $E_a^{-1} = \{(u, v) : (v, u) \in E_a\}$. In general, two-way bisimulation is a much stronger notion than ordinary bisimulation. Not so, however, for trees in the strict sense, i.e. trees considered as independent structures, rather than substructures within a larger graph in which the root could have predecessor nodes.

Lemma 2.3.4. *Bisimilar trees are two-way bisimilar.*

Proof. Let $\mathfrak{T}$ and $\mathfrak{T}'$ be bisimilar trees with roots λ and λ', respectively. We inductively construct a two-way bisimulation $Z = \bigcup_{n<\omega} Z_n$ between $\mathfrak{T}$ and $\mathfrak{T}'$. The *depth* of a node is its distance from the root. Each Z_n will associate nodes of depth n in $\mathfrak{T}$ with nodes of depth n in $\mathfrak{T}'$. Let $Z_0 = \{(\lambda, \lambda')\}$. For $n > 0$, let $(v, v') \in Z_n$ if v and v' have depth n and

(1) $\mathfrak{T}, v \sim \mathfrak{T}', v'$ (with respect to usual (forward) bisimulation), and

(2) the edges to v and v' from their parent nodes u and u' have the same label, and $(u, u') \in Z_{n-1}$.

Obviously Z is a two-way bisimulation. □

A further property of arbitrary bisimulations is that if $Z : \mathfrak{K}, v \sim \mathfrak{K}', v'$ and $Z' : \mathfrak{K}, v \sim \mathfrak{K}', v'$ are bisimulations between $\mathfrak{K}, v$ and $\mathfrak{K}', v'$, then the union is a bisimulation $(Z \cup Z') : \mathfrak{K}, v \sim \mathfrak{K}', v'$ too.

From a different angle, $Z \subseteq V \times V'$ can be seen as the graph of a (partial) function $Z : V \to \mathcal{P}(V')$, or vice versa $Z^-1 : V' \to \mathcal{P}(V)$. When talking about the *image* and inverse image under Z, it is with respect to this functional interpretation of Z.

2.4 Basic Modal Logic

We describe propositional (multi-)modal logic ML for several transition relations, i.e. for reasoning about transition systems $\mathfrak{K} = (V, (E_a)_{a \in A}, (P_b)_{b \in B})$ where in general A may have more than one element. In the literature on modal logic, this system is sometimes called K_n (where $n = |A|$ is the number of actions or 'modalities').

Syntax of ML**.** The formulae of ML are defined by the following rules.

(i) Each atomic proposition P_b is a formula.

(ii) If ψ and φ are formulae of ML, then so are $(\psi \wedge \varphi)$, $(\psi \vee \varphi)$ and $\neg\psi$.

(iii) If ψ is a formula of ML and $a \in A$ is an action, then $\langle a \rangle\psi$ and $[a]\psi$ are formulae of ML.

If there is only one transition relation, $A = \{a\}$, one simply writes $\Box$ and $\Diamond$ for $[a]$ and $\langle a \rangle$, respectively.

Semantics of ML**.** Let ψ be a formula of ML, $\mathfrak{K} = (V, (E_a)_{a\in A}, (P_b)_{b\in B})$ a transition system, and $v \in V$ a state. We write $\mathfrak{K}, v \models \psi$ to denote that ψ holds in $\mathfrak{K}$ at state v.

In the case of atomic propositions, $\psi = P_b$, we have $\mathfrak{K}, v \models \varphi$ iff $v \in P_b$. Boolean connectives are treated in the natural way. Finally for the semantics of the modal operators we put

$$\begin{aligned} \mathfrak{K}, v \models \langle a \rangle\psi \quad &\text{iff} \quad \mathfrak{K}, w \models \psi \text{ for } \textit{some } w \text{ such that } (v, w) \in E_a, \\ \mathfrak{K}, v \models [a]\psi \quad &\text{iff} \quad \mathfrak{K}, w \models \psi \text{ for } \textit{all } w \text{ such that } (v, w) \in E_a. \end{aligned}$$

The standard translation from modal to first order logic takes a formula $\varphi \in$ ML and yields a formula $\varphi^\circ(x) \in$ FO with one free variable. By starting with free variable x, and alternating between x and y in the translation of the modal operators, one can verify that ML is contained in the two-variable fragment FO^2 of first-order logic. This is a much stronger restriction than the fact that modal logics only deal with relational structures of width (at most) two.

$$\begin{aligned} (P_b)^\circ(x) &= P_b(x) \\ (\neg\varphi)^\circ(x) &= \neg(\varphi^\circ(x)) \\ (\varphi_1 \wedge \varphi_2)^\circ(x) &= \varphi_1^\circ(x) \wedge \varphi_2^\circ(x) \\ (\langle a \rangle\varphi)^\circ(x) &= \exists y(E_a xy \wedge \varphi^\circ(y)) \end{aligned}$$

The equivalence is straightforward. We also consider the extensions of ML by an unrestricted universal quantification, and by backward edges. In ML + $\blacksquare$ there is an additional universal modality operator $\blacksquare$ that can be used in place of the usual $\Box$ and $\Diamond$. The semantics is intuitive,

$$\mathfrak{K}, v \models \blacksquare\psi \quad \text{iff} \quad \mathfrak{K}, w \models \psi \text{ for all } w \in \mathfrak{K}.$$

The addition of backward edges effectively makes predecessor nodes accessible as successors by adding, for every relation E_a, the inverse action E_a^- interpreted as $\{(v,w) : (w,v) \in E_a\}$. We denote by ML^- the extension of ML by the corresponding operations $\langle a^- \rangle$ and $[a^-]$.

2.5 The Modal μ-calculus

The propositional μ-calculus L_μ is propositional modal logic augmented with least and greatest fixed points. It was introduced in [45], and has in the meanwhile been established as *the* fundamental modal fixed point logic, subsuming almost all of the commonly used modal logics used in verification and reasoning about programs, in particular the temporal logics LTL, CTL, CTL*, dynamic logic PDL [33], and also many logics used in other areas of computer science, for instance description logics [17]. For more information on the μ-calculus, we refer to [3, 11] and the references there.

Syntax of L_μ. The μ-calculus extends propositional modal logic ML, including monadic relations, or propositional (second-order) variables $X, Y, \dots$, by the following rule for building fixed point formulae.

> If ψ is a formula in L_μ and X is a propositional variable that does not occur negatively in ψ, then $\mu X.\psi$ and $\nu X.\psi$ are L_μ formulae.

Semantics of L_μ. The semantics of the μ-calculus is given as follows. A formula $\psi(X)$ with propositional variable X defines on every transition system $\mathfrak{K}$ (with state set V, and with interpretations for other free second-order variables that ψ may have besides X) an operator $\psi^{\mathfrak{K}} : \mathcal{P}(V) \to \mathcal{P}(V)$ assigning to every set $X \subseteq V$ the set

$$\psi^{\mathfrak{K}}(X) = \{v \in V : (\mathfrak{K}, X), v \models \psi\}.$$

If X occurs only positively in ψ, $\psi^{\mathfrak{K}}$ is *monotone* for every $\mathfrak{K}$, and therefore has a least and a greatest fixed point. Now we put

$\mathfrak{K}, v \models \mu X.\psi$ iff v is an element of the least fixed point of the operator $\psi^{\mathfrak{K}}$,
$\mathfrak{K}, v \models \nu X.\psi$ iff v is an element of the greatest fixed point of $\psi^{\mathfrak{K}}$.

Remark. By the well known duality between least and greatest fixed points, $\nu X.\psi$ is equivalent to $\neg\mu X.\neg\psi[X/\neg X]$. Hence we could eliminate greatest fixed points. However, it will sometimes be more convenient to keep least and greatest fixed points and work with formulae in negation normal form, cf. Definition 2.1.1.

There is a variant of L_μ which admits systems of *simultaneous fixed points*. These do not increase the expressive power but sometimes allow for more straightforward formalisations. Here one associates with any tuple $\overline{\psi} = (\psi_1, \ldots, \psi_k)$ of formulae $\psi_i(\boldsymbol{X}) = \psi_i(X_1, \ldots, X_k)$, all positive in the X_i, a new formula that is a system of fixed-points

$$S \;=\; \mu X_1, \ldots, X_k. \left\{ \begin{array}{c} \psi_1(X_1, \ldots, X_k) \\ \vdots \\ \psi_k(X_1, \ldots, X_k) \end{array} \right\}.$$

The semantics of S is induced by the least fixed point of the monotone operator $S^{\mathfrak{K}}$ mapping $\boldsymbol{X}$ to $\boldsymbol{X}'$ where $X'_i = \{v \in V : (\mathfrak{K}, \boldsymbol{X}), v \models \psi_i\}$. By convention $\mathfrak{K}, v \models S$ iff v is an element of the first component of the least fixed point of the above operator. Similar conventions apply w.r.t. simultaneous greatest fixed point.

It is well known that simultaneous fixed points can be uniformly eliminated in favour of nested individual fixed points. This is based on a general fact concerning fixed points of monotone operators which is sometimes called the Bekic principle and which applies to least fixed point logics in general [3]. In its simplest form for the μ-calculus, it says that

$$\mu XY. \left\{ \begin{array}{c} \psi(X, Y) \\ \varphi(X, Y) \end{array} \right\} \;=\; \mu X. \psi(X, \mu Y. \varphi(X, Y)).$$

2.6 Invariance and Safety

In this section we present the notions of invariance and safety, two key players in the analysis of modal bisimulation. Informally, a formula, or a unary global relation, is invariant for bisimulation, if it does not distinguish between bisimilar states. One area of interest is the question of which classes of formulae can be used for invariant queries. The other take is that invariant relations can be added to a class of structures without changing the bisimulation equivalence. This second viewpoint is also central to bisimulation safety, which states that a binary global relation does not destroy bisimilarity when added as additional accessibility relation.

Definition 2.6.1. A formula ψ is *invariant for bisimulation* if whenever $\mathfrak{K}, v \sim \mathfrak{K}', v'$ and $\mathfrak{K}, v \models \psi$, then also $\mathfrak{K}', v' \models \psi$. A logic L is invariant for bisimulation if all L-formulae are invariant for bisimulation.

It is well known that all formulae of ML and L_μ are invariant under bisimulation (see e.g. [8, 3]). Furthermore there is a theorem by van Benthem that characterises the ML definable relations in terms of bisimulation invariant FO. Several standard proofs of this theorem can be found in the literature [40]. We will exhibit one of them since all following invariance characterisation theorems of this format for logics included in FO can be shown in exactly the same way. Note that the compactness theorem for first-order logic is used (implicitly) in various forms and generalising the method beyond first-order logic requires a different approach [40].

Definition 2.6.2. A transition system $\mathfrak{K}$ is ML-saturated if for every $v \in \mathfrak{K}$ and every modal type p:
If $\langle a \rangle p_0$ is consistent with the ML-theory of v in $\mathfrak{K}$ for every finite $p_0 \subseteq p$, then there is an a-successor w of v that realises p in $\mathfrak{K}$.

Clearly every ω-saturated transition system is ML-saturated — the requirement for all first-order types includes all modal types.

Lemma 2.6.3. *Let $\mathfrak{K}$ and $\mathfrak{K}'$ be* ML*-saturated transition systems. The modal equivalence relation between $\mathfrak{K}$ and $\mathfrak{K}'$ is a bisimulation.*

Proof. Consider two nodes $v \in \mathfrak{K}$, $v' \in \mathfrak{K}'$ such that v and v' are modally equivalent. Clearly this implies that they satisfy the same atomic properties. For the forth condition, consider an a-successor w of v and let p be the modal type of w. Since the modal theories of v and v' coincide, and $\langle a \rangle p$ is necessarily consistent with (the theory of) v, by ML-saturation of $\mathfrak{K}'$ there is an a-successor w' of v' that satisfies p. The back condition is shown similarly. □

Lemma 2.6.4. *A transition system $\mathfrak{K}$ is* ML*-saturated iff there is a total bisimulation between $\mathfrak{K}$ and some ω-saturated $\mathfrak{K}'$.*

Proof. Let $\mathfrak{K}$ be ML-saturated and by Theorem 2.2.8 let $\mathfrak{K}'$ be an ω-saturated elementary extension of $\mathfrak{K}$. By Lemma 2.6.3 the modal equivalence relation is a bisimulation between $\mathfrak{K}$ and $\mathfrak{K}'$. Since $\mathfrak{K}$ can be elementarily embedded into $\mathfrak{K}'$, this bisimulation is total.

Let $\mathfrak{K}'$ be ω-saturated and let $Z : \mathfrak{K} \sim \mathfrak{K}'$ be a total bisimulation. Let $v \in \mathfrak{K}$ and let p be a modal type. Suppose that $\langle a \rangle p$ is consistent with v. Then it is also consistent with any v' where $(v, v') \in Z$. Since $\mathfrak{K}'$ is, in particular, ML-saturated, there is an a-successor w' of v' that satisfies p. The bisimulation Z then yields an a-successor w of v in $\mathfrak{K}$ that satisfies p, too. □

Theorem 2.6.5 (van Benthem). *A unary global relation is* ML *definable iff it is* FO *definable and invariant for bisimulation.*

Proof. Bisimulation invariance for ML-formulae, which by the standard translation can be seen as FO-formulae, follows by straightforward induction.

For the converse direction, suppose that $\varphi(x) \in \mathrm{FO}$ is bisimulation invariant. Let $\Psi = \{\psi \in \mathrm{ML} \; : \; \varphi(x) \models \psi^\circ(x)\}$.

Let $\mathfrak{K}, v \models \Psi$ and suppose that $\mathfrak{K}$ was chosen ω-saturated. Let p be the ML-type of v in $\mathfrak{K}$ and note that $p \cup \{\varphi\}$ is consistent. For surely p is consistent, and if the union is not, then $\varphi \models \neg\pi_1 \vee \cdots \vee \neg\pi_n$ for some $\pi_i \in p$. Hence $\neg\pi_1 \vee \cdots \vee \neg\pi_n \in \Psi$, which means that $\neg\pi_1 \vee \cdots \vee \neg\pi_n$ holds at v, a contradiction to $\pi_i \in p$, the modal type of v.

Since $p \cup \{\varphi\}$ is consistent, there is an ω-saturated $\mathfrak{K}'$, and a $v' \in \mathfrak{K}'$ such that $p \cup \{\varphi\}$ holds at v'. Now v and v' both have the same modal type, and both are nodes of ML-saturated structures, so v and v' are bisimilar. Since φ is bisimulation invariant, φ holds at v, too.

In other words, because φ is invariant for bisimulation, we have $\Psi \models \varphi$. By compactness for FO there is a finite $\Psi_0 \subset \Psi$ such that $\Psi_0 \models \varphi(x)$, and $\varphi(x) \equiv \bigwedge \Psi_0$, i.e. $\bigwedge \Psi_0$ is the desired ML-formula. $\square$

A similar and highly non-trivial analogous characterisation is also known for the modal μ-calculus [44]. We do not repeat the proof, although certain techniques will be re-used in Chapter 8.

Theorem 2.6.6 (Janin, Walukiewicz). *A unary global relation is* L_μ*-definable iff it is* MSO*-definable and invariant under bisimulation.*

What bisimulation invariance is for unary global relations, bisimulation safety is for binary relations. The definition is analogous in the sense that adding a bisimulation safe, resp. invariant relation to a class of structures does not change the bisimulation equivalence among them.

Definition 2.6.7. A global relation R is *safe for bisimulation*, if for all transition systems $\mathfrak{K}$ and $\mathfrak{K}'$, any bisimulation $Z : \mathfrak{K} \sim \mathfrak{K}'$ is a bisimulation $Z : (\mathfrak{K}, R^{\mathfrak{K}}) \sim (\mathfrak{K}', R^{\mathfrak{K}'})$, too.

In order to characterise e.g. the bisimulation safe fragment of first-order logic, we first need a modal-style logic that is capable of defining binary relations. One candidate is the propositional dynamic logic PDL. Every PDL statement is either a formula, or a program, defining unary and binary global relations, respectively. PDL adds to ML the capability to inductively build new action relations using test, iteration, union and sequential composition. The iteration, a kind of Kleene star, is unbounded, and hence a very un-first-order construction. In the following we will mostly consider variations of PDL that do not allow iteration; following the nomenclature from regular languages we call the resulting fragments *star-free*.

Syntax of PDL**.**

(i) Every atomic proposition P_b is a formula.
(ii) The PDL-formulae are closed under Boolean combinations.
(iii) If π is a program, and φ is a formula, then $\langle\pi\rangle\varphi$ is a formula.

(i) Every atomic action E_a is a program.
(ii) If φ is a formula, then $\varphi?$ is a program.
(iii) If π is a program, then so is π^*.
(iv) If π and π' are programs, then so are $\pi \cup \pi'$ and $\pi; \pi'$.

The semantics of PDL is a straightforward extension of the semantics for basic modal logic.

Theorem 2.6.8 (Hollenberg). *A binary global relation is definable in star-free* PDL *iff it is first-order definable and safe for bisimulation.*

In Section 6.2.1 we will show how Hollenberg's proof can be pushed up to a certain action guarded logic, however in contrast to the invariance characterisation theorems this approach falls significantly short of providing a template that goes all the way up to full guarded bisimulation.

Although the notions of bisimulation invariance and bisimulation safety apply to — at least for modal-style bisimulation — relations of a different format, there is of course a connection between the two notions.

Proposition 2.6.9. *If $\varphi(x, y)$ is safe for bisimulation and P_a is an atomic proposition that does not occur in φ, then $\exists y(\varphi(x, y) \wedge P_a(y))$, which could be written as $\langle\varphi\rangle P_a$, is invariant for bisimulation.*

Chapter 3

Guarded Logics

This chapter is dedicated to the guarded world and gives definitions of the numerous guarded objects, including guarded first-order logic, guarded fixed-point logic, and introducing what we call guarded second-order logic. Taking up the lead from the modal world, we establish a basic set of properties for guarded logics, centred around the guarded equivalent of bisimulation equivalence. We motivate our version of guarded second-order logic by showing that several canonical versions fall together, and that the expressibility can be naturally described in terms of monadic second order logic on certain structures. We also propose a guarded variant of the relational algebra, as known from database theory, that will be used to characterise safe fragments of first-order logic here, and in Chapter 6.

3.1 Structures and Guards

Definition 3.1.1. Let $\mathfrak{B}$ be a relational structure with universe B and vocabulary τ.

(i) A set $X \subseteq B$ is *guarded* in $\mathfrak{B}$ if there exists a ground atom $\alpha(b_1, \dots, b_k)$ such that $\mathfrak{B} \models \alpha(b_1, \dots, b_k)$ and $X = \{b_1, \dots, b_k\}$.

(ii) A τ-structure $\mathfrak{A}$, in particular a substructure $\mathfrak{A} \subseteq \mathfrak{B}$, is *guarded* if its universe is a guarded set in $\mathfrak{A}$ (in $\mathfrak{B}$).

(iii) A tuple $(b_1, \dots, b_n) \in B^n$ is *guarded* in $\mathfrak{B}$ if $\{b_1, \dots, b_n\} \subseteq X$ for some guarded set $X \subseteq B$.

(iv) A tuple $(b_1, \dots, b_k) \in B^k$ is a *guarded list* in $\mathfrak{B}$ if its components are pairwise distinct and $\{b_1, \dots, b_k\}$ is a guarded set. We admit the empty list $\emptyset$ as a guarded list.

(v) A relation $X \subseteq B^n$ is *guarded* if it only consists of guarded tuples.

Note that a singleton set $X = \{b\}$ is always guarded by the atom $b = b$. Correspondingly, any monadic relation is a guarded relation according to (v). The cardinality of guarded sets in $\mathfrak{B}$ is bounded by the maximal arity of the relations in τ, the *width of* τ. Guarded tuples, however, can have any length. Guarded lists will be of technical interest later as succinct tuple representations of guarded sets; note that unlike guarded tuples they list a full guarded set, not just a subset of a guarded set.

Example 3.1.2. Let $\mathfrak{A}$ be the structure with universe $A = \{1, 2, 3, 4\}$ and the ternary relation $R = \{(1, 2, 3), (4, 2, 3)\}$. The substructure of $\mathfrak{A}$ induced by $\{2, 3, 4\}$ is a guarded structure, but $\mathfrak{A}$ itself is not. The tuples $(3, 2, 1)$ and $(2, 3, 4)$ are guarded tuples and guarded lists, $(1, 1)$ and $(4, 4, 3, 2)$ are guarded tuples but not guarded lists and $(4, 3, 2, 1)$ is neither.

In the context of a given structure $\mathfrak{A}$, we also distinguish tuples, lists, substructures etc. that are *maximal guarded*. A tuple $\boldsymbol{a} \in \mathfrak{A}$ is maximal guarded if no other guarded tuple $\boldsymbol{b} \in \mathfrak{A}$ is a superset of $\boldsymbol{a}$ when both are seen as unordered sets of elements. The same idea applies to the other kinds of guarded objects.

Sometimes it is necessary to specify the guarding relation more precisely. Against the background of a vocabulary τ, a guarded object may be guarded by any $R \in \tau$. For every $\tau' \subseteq \tau$ and $R \in \tau$ we use the terms τ'-guarded and R-guarded with the intuitive meaning that the guarding relation used in the guard statements is either an element of τ', or the specified R, respectively.

3.2 The Guarded Fragment

The guarded fragment extends modal logic to a richer fragment of first-order logic. Its characteristic feature is a relativised pattern of quantification, which generalises modal quantification. The formulae of GF in vocabulary τ are defined by the following rules.

Syntax of GF**.**

(i) All quantifier free first-order formulae are formulae of GF.

(ii) If ψ and φ are formulae of GF, then so are $(\psi \wedge \varphi)$, $(\psi \vee \varphi)$ and $\neg\psi$.

(iii) If $\psi(\boldsymbol{x},\boldsymbol{y})$ is a formula of GF and $\alpha(\boldsymbol{x},\boldsymbol{y})$ is a τ-atom such that all free variables of ψ do actually occur in α then $\exists\boldsymbol{y}\big(\alpha(\boldsymbol{x},\boldsymbol{y})\wedge\psi(\boldsymbol{x},\boldsymbol{y})\big)$ and $\forall\boldsymbol{y}\big(\alpha(\boldsymbol{x},\boldsymbol{y})\rightarrow\psi(\boldsymbol{x},\boldsymbol{y})\big)$ are formulae of GF.

The atoms α relativising GF quantifications are called *guards*. We shall often also use the more intuitive notation $(\forall\boldsymbol{y}\,.\,\alpha)\psi$ and $(\exists\boldsymbol{y}\,.\,\alpha)\psi$ as shorthand for correspondingly relativised first-order quantification. When writing $\alpha(\boldsymbol{x},\boldsymbol{y})$ for the guard and $\psi(\boldsymbol{x},\boldsymbol{y})$ for the kernel that is being quantified, we implicitly appeal to the convention that the free variables of ψ are *among* those displayed and that *all* variables displayed in the atom α actually occur in α.

Note that first-order quantification over a single free variable is always admissible in GF, since singletons are guarded: $\exists x\varphi(x)\equiv\exists x(x=x\wedge\varphi(x))$ is in GF if $\text{free}(\varphi)\subseteq\{x\}$.

Semantics of GF**.** The guarded fragment is a syntactic fragment of first-order logic, so the semantics of GF is the usual one for first-order formulae.

It is obvious that GF generalises ML. A typical modal formula $\varphi=[1]\langle 2\rangle P$, rendered in first-order terms as $\varphi^\circ(x)=\forall y\big(E_1xy\rightarrow\exists z(E_2yz\wedge Pz)\big)$, is clearly in GF. The following are some simple examples of GF formulae that are not in ML (even though in a language of transition systems).

$$\begin{aligned}\eta(x) &= \forall y(E_1yx\rightarrow(E_2xy\wedge y\neq x))\\ \psi &= \forall x\big(x=x\rightarrow\exists y(Exy\wedge\neg Eyx)\big)\end{aligned}$$

The formula $\eta(x)$ states that every E_1-predecessor y of x also is an E_2-successor and distinct from x, and ψ globally ensures that every node x has an E-successor that is not an E-predecessor too. The following technical example will also be of use later.

Example 3.2.1. The set of all guarded k-tuples in structures of vocabulary $\tau=\{R_1,\dots R_t\}$ is defined by the formula

$$\mathbb{G}(x_1,\dots,x_k)=\bigvee_{i=1}^{t}\exists\boldsymbol{y}\Big(R_i\boldsymbol{y}\wedge\bigwedge_{\ell=1}^{k}\bigvee_{j}x_\ell=y_j\Big).$$

Note that this formula is *not* in GF since the variables $x_1,\dots,x_k$ do not occur in the atoms $R_i(\boldsymbol{y})$. However it is not difficult to construct an equivalent guarded formula. For any complete equality type on $\{x_1,\dots,x_k\}$ specified by a quantifier-free formula $\eta(\boldsymbol{x})$ in the language of just $=$, let $\boldsymbol{x}^\eta$ be a subtuple of $\boldsymbol{x}$ comprising precisely one variable from each $=$-class specified by η. Let $\alpha(\boldsymbol{y}, \boldsymbol{x}^\eta)$ be a τ-atom in which all variables from $\boldsymbol{x}^\eta$ actually occur, and the $\boldsymbol{y}$ are fresh, i.e. disjoint from $\boldsymbol{x}$. Put $\xi_{\eta\alpha}(\boldsymbol{x}) = \eta(\boldsymbol{x}) \wedge \exists \boldsymbol{y} \alpha(\boldsymbol{y}, \boldsymbol{x}^\eta)$. For the degenerate case of η_0, specifying the equality type of a singleton tuple $x_1 = x_2 = \ldots = x_k$, we just put $\xi_0 = \eta_0(\boldsymbol{x})$. It is easily checked that the disjunction over all these formulae $\xi_{\eta\alpha}(\boldsymbol{x})$ is as desired.

Likewise, we note that quantifications of the form $\exists \boldsymbol{x}(\mathbb{G}(\boldsymbol{x}) \wedge \varphi(\boldsymbol{x}))$ and $\forall \boldsymbol{x}(\mathbb{G}(\boldsymbol{x}) \rightarrow \varphi(\boldsymbol{x}))$, though not strictly in GF, may equivalently be expressed in GF syntax. We shall sometimes use such liberalised formalisations for the sake of clarity and convenience.

A related issue is the question of whether the global relation defined by a formula is necessarily guarded. We formalise this notion as follows.

Definition 3.2.2. We say that a formula $\varphi(\boldsymbol{x})$ is *variable-guarded* if it implies guardedness of the tuple $\boldsymbol{x}$ of free variables, i.e., if $\varphi(\boldsymbol{x})$ is logically equivalent to $\varphi(\boldsymbol{x}) \wedge \mathbb{G}(\boldsymbol{x})$.

As GF imposes no restrictions at the quantifier free level, or on Boolean combinations, formulae of GF are not in general variable guarded. However, any formula of GF is equivalent to a Boolean combination of variable-guarded GF formulae. To see this, note that any GF formula is a Boolean combination of atomic statements, which are trivially variable-guarded, and formula whose top-level syntactic element is a guarded quantification – thereby providing guards for their respective free variables.

3.3 Guarded Fixed Point Logic

Guarded fixed point logic μGF as introduced in [31] is the natural extension of GF by means of least and greatest fixed points (or corresponding systems of simultaneous fixed points).

Syntax of μGF. Starting from GF, with second-order variables $X, Y, Z, \ldots$ that are treated like predicates in τ but may *not* be used in guards, we augment the syntax by the following rule for building fixed point formulae.

> Let X be a k-ary relation variable and let $\boldsymbol{x} = x_1, \ldots, x_k$ be a k-tuple of distinct variables. Further, let $\psi(X, \boldsymbol{x})$ be a formula in μGF with only positive occurrences of X, where X is not used in guards and where all free variables of ψ are among these x_i. Then $[\mathbf{lfp}\ X\boldsymbol{x} \,.\, \psi](\boldsymbol{x})$ and $[\mathbf{gfp}\ X\boldsymbol{x} \,.\, \psi](\boldsymbol{x})$ are formulae of μGF too.

Semantics of μGF. The semantics of $[\mathbf{lfp}\ X\boldsymbol{x}\,.\,\psi](\boldsymbol{x})$ is the natural one associated with the least fixed point of the monotone operator $\psi^{\mathfrak{B}}$ sending $X \subseteq B^k$ to $\{\boldsymbol{b} \in B^k : \mathfrak{B} \models \psi(X, \boldsymbol{b})\}$. More precisely,

> $\mathfrak{B} \models [\mathbf{lfp}\ X\boldsymbol{x}\,.\,\psi](\boldsymbol{b})$ iff $\boldsymbol{b}$ is an element of the least fixed point of the operator $\psi^{\mathfrak{B}}$.
> Similarly for $[\mathbf{gfp}\ X\boldsymbol{x}\,.\,\psi](\boldsymbol{x})$ and the greatest fixed point of $\psi^{\mathfrak{B}}$.

One may also admit *simultaneous* least and greatest fixed points in μGF. Given formulae $\psi_1(X_1, \dots, X_k, \boldsymbol{x}_1), \dots, \psi_k(X_1, \dots, X_k, \boldsymbol{x}_k)$, with only positive occurrences of the relation variables $X_1, \dots, X_k$ and where the $\boldsymbol{x}_i$ contain all free first-order variables in ψ_i and match the arity of the X_i,

$$S = \left\{ \begin{array}{ccc} X_1\boldsymbol{x}_1 & = & \psi_1(X_1, \dots, X_k, \boldsymbol{x}_1) \\ & \vdots & \\ X_k\boldsymbol{x}_k & = & \psi_k(X_1, \dots, X_k, \boldsymbol{x}_k) \end{array} \right\}$$

is a *system of fixed point equations*, and $[\mathbf{lfp}\ X_i\,.\,S](\boldsymbol{x}_i)$ and $[\mathbf{gfp}\ X_i\,.\,S](\boldsymbol{x}_i)$ are formulae of μGF.

On every structure $\mathfrak{A}$, the system S defines a monotone operator $S^{\mathfrak{A}}$ on k-tuples of relations, with least fixed point $\mathbf{lfp}(S^{\mathfrak{A}}) = (X_1^{\mu}, \dots, X_k^{\mu})$ and greatest fixed point $\mathbf{gfp}(S^{\mathfrak{A}}) = (X_1^{\nu}, \dots, X_k^{\nu})$. Now $\mathfrak{A} \models [\mathbf{lfp}\ X_i\,.\,S](\boldsymbol{a})$ iff $\boldsymbol{a} \in X_i^{\mu}$ and $\mathfrak{A} \models [\mathbf{gfp}\ X_i\,.\,S](\boldsymbol{a})$ iff $\boldsymbol{a} \in X_i^{\nu}$.

Again, the Bekic principle implies that simultaneous fixed points can be eliminated in μGF.

It should be stressed that the presence of extra first-order parameters in fixed point operations as well as the use of second-order variables and fixed points as guards is disallowed in μGF. These restrictions are essential for keeping the semantics invariant under guarded bisimulation, cf. Definition 3.4.1. For instance, with the use of a first-order parameter (here x), one can define the transitive closure of a binary relation E by the formula

$$\varphi(x, y) = [\mathbf{lfp}\ Xy\,.\,E(xy) \vee (\exists z.E(zy))X(z)](y).$$

However, the transitive closure query is *not* invariant under guarded bisimulation and it is known that adding transitive closure to GF produces an undecidable logic [24]. Allowing fixed-point variables in guards would make full first-order quantification available. For instance,

$$\exists y\varphi(x, y) \equiv [\mathbf{gfp}\ Xxy\,.\,(\exists y.X(xy))\varphi(x, y)](x, x).$$

A seemingly more restrictive variant of guarded fixed point logic would only allow least and greatest fixed points over variable-guarded formulae in the sense of Definition 3.2.2 above, or fixed points like

$$[\mathbf{lfp}\ X\boldsymbol{x} \,.\, \varphi(X, \boldsymbol{x}) \wedge \mathbb{G}(\boldsymbol{x})](\boldsymbol{x}).$$

We refer to these as *strictly guarded fixed points*. Strictly guarded least and greatest fixed points can obviously only define guarded relations. A restriction of μGF to strictly guarded fixed points, however, does not diminish its expressive power. This claim is easily verified for sentences and variable-guarded formulae, see [31]. Formulae with arbitrary tuples of free first-order variables, however, require some extra attention.

Proposition 3.3.1. *Any formula of* μGF *is logically equivalent to one in which all fixed points are strictly guarded.*

Proof. Consider a least fixed point formula $\psi(\boldsymbol{x}) = [\mathbf{lfp}\ X\boldsymbol{x} \,.\, \varphi](\boldsymbol{x})$. Inductively we assume that all fixed points within φ are in strictly guarded form. Looking at X-atoms in $\varphi(X, \boldsymbol{x})$, we distinguish the following:

(i) X-atoms in the scope of guarded first-order quantification.
(ii) X-atoms in the scope of least or greatest fixed point operators.
(iii) X-atoms at quantifier free level, $X(\boldsymbol{z})$, $\boldsymbol{z} \subseteq \boldsymbol{x}$.

For occurrences of type (i) or (ii) we may replace X by its guarded part throughout the fixed-point process. The guarded part of X is the subset of X consisting of guarded tuples in X. For type (ii) occurrences that are not of type (i), this relies on the inductive assumption that fixed points within φ are strictly guarded. As far as guarded tuples are concerned, even an atom of type (iii) can only evaluate to true for a guarded instantiation of $\boldsymbol{x}$ in $\varphi(\boldsymbol{x})$ if it would also evaluate to true for the guarded part of X, rather than for the original X. Hence, inductively, we find that the guarded part ψ^{g} of the fixed point $[\mathbf{lfp}\ X\boldsymbol{x} \,.\, \varphi]$ is definable as a strictly guarded fixed point, by

$$\psi^{\mathrm{g}}(\boldsymbol{x}) = [\mathbf{lfp}\ X\boldsymbol{x} \,.\, \varphi(X, \boldsymbol{x}) \wedge \mathbb{G}(\boldsymbol{x})](\boldsymbol{x}).$$

Let $\varphi(\psi^{\mathrm{g}}, X, \boldsymbol{x})$ be the result of substituting this formula for all occurrences of X in φ apart from those of type (iii). As $\psi(\boldsymbol{x})$ implies $\psi^{\mathrm{g}}(\boldsymbol{x})$ and by monotonicity, we clearly have

$$[\mathbf{lfp}\ X\boldsymbol{x} \,.\, \varphi(X, \boldsymbol{x})](\boldsymbol{x}) \equiv [\mathbf{lfp}\ X\boldsymbol{x} \,.\, \varphi(\psi^{\mathrm{g}}, X, \boldsymbol{x})](\boldsymbol{x}).$$

Note that the only remaining free occurrences of X in $\varphi(\psi^{\mathrm{g}}, X, \boldsymbol{x})$ are at the quantifier free level. It follows that the fixed point formula

$$[\mathbf{lfp}\ X\boldsymbol{x} \,.\, \varphi(\psi^{\mathrm{g}}, X, \boldsymbol{x})](\boldsymbol{x})$$

is bounded. This means that the fixed point is attained within a uniformly bounded finite number of iterations, since there are only finitely many quantifier free types. By unravelling this finite number of iterations within GF we find that ψ is GF-definable from strictly guarded fixed points. □

Let us now consider the hierarchy of μGF formulae as defined by the alternation of greatest and least fixed point operators. This structural property of formulae is important in many practical applications: The expression complexity of the model-checking problem for μGF and L_μ formulae is exponential in this measure.

Several variants of a definition of alternation depth of a formula have been considered in the literature. The simple version counts syntactic alternations of least and greatest fixed-point operators. Further studies of L_μ have shown that counting only interdependent alternations captures the complexity of fixed-point alternation more precisely. The following informal definition of alternation depth applies to μGF as well as L_μ, and many other reasonable fixed-point logics.

Definition 3.3.2. Let φ be a fixed-point formula in NNF that is well named with respect to second-order variables. For every second-order variable X occurring in φ, find the smallest subformula $D_\varphi(X)$ in φ, the definition of X in φ, where X is bound. The *alternation level* of X, $\mathrm{al}(X)$, is defined as follows.

If $D_\varphi(X)$ has no free second-order variables, then $\mathrm{al}(X) = 1$.

If $D_\varphi(X)$ has free second-order variables $Y_1, \ldots, Y_n$, let k such that $\mathrm{al}(Y_k)$ is maximal among $\mathrm{al}(Y_1), \ldots, \mathrm{al}(Y_n)$. If X and Y_k are both bound by the same fixed-point operator, μ or ν, then $\mathrm{al}(X) = \mathrm{al}(Y_k)$. If the binding fixed-point operators differ, then $\mathrm{al}(X) = \mathrm{al}(Y_k) + 1$.

It can be shown that the above definition does not depend on the exact choice of k. The binding fixed-point operator is the same for all Y's with maximal alternation level. The *alternation depth* of a formula φ is the maximum of the alternation levels of the second-order variables of φ, or zero if φ does not contain any fixed-point operator.

For more precise definitions and proofs of the following results on the alternation hierarchy for the modal μ-calculus we refer to [2, 10, 47].

Theorem 3.3.3 (Bradfield, Lenzi).
The L_μ alternation-depth hierarchy is strict.

Theorem 3.3.4 (Arnold).
The L_μ *alternation-depth hierarchy is strict on binary trees.*

The proof given by Bradfield is much stronger and yields the following corollaries that are indifferent to which definition of alternation depth is used. By L_μ^- we denote the modal μ-calculus augmented by backward quantifiers, i.e. the fixed-point extension of ML^-.

Corollary 3.3.5. *The* L_μ^- *alternation-depth hierarchy is strict.*

Corollary 3.3.6. *The* $\mu\mathrm{GF}$ *alternation-depth hierarchy is strict.*

The finite model property of L_μ implies that the L_μ alternation-depth hierarchy is strict in the sense of finite model theory too. On the other hand it is well known [19] that the LFP hierarchy collapses in the finite — at the cost of increasing the arity of the fixed-point iterations. Since both L_μ^- and $\mu\mathrm{GF}$ do not have the finite model property, but also lack the possibility of increasing the width of fixed-point iterations, neither method can be used to establish or refute whether the hierarchy is strict for L_μ^- or $\mu\mathrm{GF}$ when restricted to finite models.

3.4 Guarded Bisimulation

A fundamental tool for analysing guarded logics are guarded bisimulations, introduced in [1].

Definition 3.4.1. A *guarded bisimulation* between two τ-structures $\mathfrak{A}$ and $\mathfrak{B}$ is a non-empty set I of finite partial isomorphisms $f : X \to Y$ from $\mathfrak{A}$ to $\mathfrak{B}$, such that the following back and forth conditions are satisfied.

Forth. For every $f : X \to Y$ in I and for every guarded set $X' \subseteq A$, there exists a partial isomorphism $g : X' \to Y'$ in I such that f and g agree on $X \cap X'$.

Back. For every $f : X \to Y$ in I and for every guarded set $Y' \subseteq B$, there exists a partial isomorphism $g : X' \to Y'$ in I such that f^{-1} and g^{-1} agree on $Y \cap Y'$.

Two τ-structures $\mathfrak{A}$ and $\mathfrak{B}$ are *guarded bisimilar*, also written as $\mathfrak{A} \sim_g \mathfrak{B}$, if there exists a guarded bisimulation between them. Two τ-structures with parameter tuples $\mathfrak{A}, \boldsymbol{a}$ and $\mathfrak{B}, \boldsymbol{b}$ are *guarded bisimilar* ($\mathfrak{A}, \boldsymbol{a} \sim_g \mathfrak{B}, \boldsymbol{b}$), if there is a guarded bisimulation I between $\mathfrak{A}$ and $\mathfrak{B}$ containing the partial isomorphism $f\colon \boldsymbol{a} \mapsto \boldsymbol{b}$.

Note that, with the possible exception of the one partial isomorphism dealing with a given pair of parameter tuples, one may restrict attention to systems of partial isomorphisms whose domain and range are guarded sets in $\mathfrak{A}$ and $\mathfrak{B}$, respectively. However it does not hurt if a guarded bisimulation contains additional partial isomorphisms with non-guarded domains and ranges.

We refer to the relation $\sim_g$ as *guarded bisimulation equivalence*.

Example 3.4.2. Let $\mathfrak{A}$ and $\mathfrak{B}$ be two structures as depicted in Figure 3.1, with two ternary relations $\triangle$ and $\blacktriangle$ and one binary relation $\wr$ (all assumed to be symmetric, as in hypergraph theory).

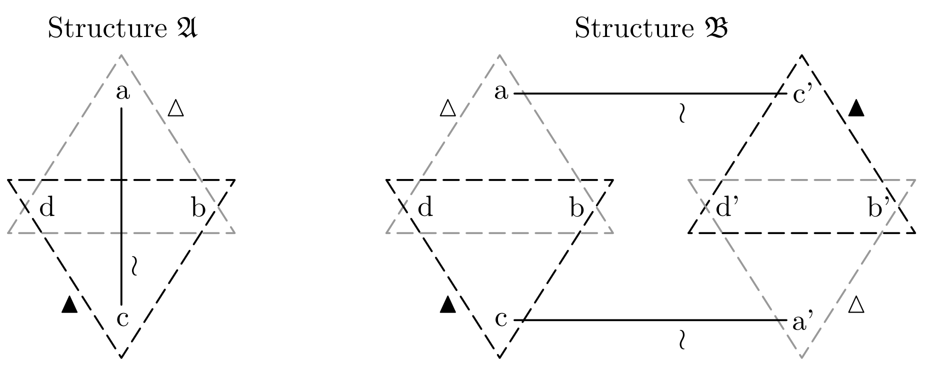

Figure 3.1: Sketches of $\mathfrak{A}$ and $\mathfrak{B}$.

A guarded bisimulation $I : \mathfrak{A} \sim_g \mathfrak{B}$ is given by $I = \{f_\triangle : abd \mapsto abd,\ f'_\triangle : abd \mapsto a'b'd'\ f_\blacktriangle : cbd \mapsto cbd,\ f'_\blacktriangle : cbd \mapsto c'b'd',\ f_\wr : ac \mapsto ac',\ f'_\wr : ac \mapsto a'c\}$, plus all subfunctions of the ones listed.

The reader is encouraged to check the details to get more familiar with the definition.

For guarded logics over arbitrary relational vocabularies there is obviously no way that the tree model property as found in modal logics will hold too. However there is a more general notion of tree-like structures that is perfectly matched to the guarded world.

Definition 3.4.3. A structure $\mathfrak{A}$ has *tree width* $k-1$ if there exists a tree $T = (V, E)$ and a function $F : V \to B^{\leq k}$ such that

(i) For every guarded set $X \subseteq B$ there is a $v \in V$ such that $X \subseteq F(v)$.

(ii) For every element $b \in B$ the set $\{v \in V \ : \ b \in F(v)\}$ is connected in T.

In Section 4.5 we will see that every structure of vocabulary τ is guarded bisimilar to a structure of tree-width $m - 1$, if m is the width of τ.

Definition 3.4.4. A logic L has the *generalised tree model property* iff there is a recursive function $f : \mathrm{L} \rightarrow \mathbb{N}$ such that for every $\varphi \in \mathrm{L}$, φ is satisfiable iff φ has a model of tree width at most $f(\varphi)$.

3.5 Invariance and Safety

Definition 3.5.1. A global relation R is *invariant* under guarded bisimulation if, whenever $\mathfrak{A}, \boldsymbol{a} \sim_g \mathfrak{B}, \boldsymbol{b}$ and $\boldsymbol{a} \in R^{\mathfrak{A}}$, then also $\boldsymbol{b} \in R^{\mathfrak{B}}$. A logic L is invariant under guarded bisimulation if all L-formulae are invariant under guarded bisimulation.

It suffices to look at variable-guarded formulae when checking a logic like GF or μGF for invariance under guarded bisimulation, as any formula is equivalent to a Boolean combination of variable-guarded formulae. The following is easily proved by syntactic induction, cf. also [1, 26].

Proposition 3.5.2. GF *and* μGF *are invariant under guarded bisimulation.*

Furthermore, the following characterisation of the bisimulation invariant fragment of FO in terms of GF has been obtained, matching the equivalent modal Theorem 2.6.5.

Theorem 3.5.3 (Andréka, van Benthem, Németi). *A property of relational structures is definable in the guarded fragment* GF *if, and only if, it is first-order definable and invariant under guarded bisimulation.*

The guarded version of Theorem 2.6.6, the characterisation theorem for fixed-point logics, is shown in Chapter 8. Next on our introductory agenda is safety in the guarded world.

Definition 3.5.4. A global relation R is *safe* for guarded bisimulation iff for all structures $\mathfrak{A}$ and $\mathfrak{B}$, any guarded bisimulation $I : \mathfrak{A} \sim_g \mathfrak{B}$ is a guarded bisimulation $I : (\mathfrak{A}, R^{\mathfrak{A}}) \sim_g (\mathfrak{B}, R^{\mathfrak{B}})$, too.

For GF the notions of safety and invariance nearly fall together. If, as some people may argue, the interpretation of guarded formulae is restricted to guarded tuples, then the two notions fully coincide.

Theorem 3.5.5. *A global relation R is safe for guarded bisimulation iff it is guarded and invariant for guarded bisimulation.*

Proof. Let $\mathfrak{A}$ be a relational structure and consider $R^{\mathfrak{A}}$. If R is safe for guarded bisimulation, then $R^{\mathfrak{A}}$ is a guarded relation. For assume to the contrary that $R^{\mathfrak{A}}$ contains some non-guarded tuple $\boldsymbol{a}$. Let $\mathfrak{B} \sim_g \mathfrak{A}$ and consider a guarded bisimulation $I : \mathfrak{B} \sim_g \mathfrak{A}$ where the domain and range of each $f \in I$ is guarded. Then I does not satisfy the forth condition for $(\mathfrak{A}, R^{\mathfrak{A}})$ and $(\mathfrak{B}, R^{\mathfrak{B}})$, independent of $R^{\mathfrak{B}}$: If we take the set $X_{\boldsymbol{a}} = \boldsymbol{a}$, which is guarded in $(\mathfrak{A}, R^{\mathfrak{A}})$, there is no $f \in I$ with domain $X_{\boldsymbol{a}}$.

Now since R is guarded, R is safe for guarded bisimulation iff for all $\mathfrak{A}$ and $\mathfrak{B}$, every guarded bisimulation $I : \mathfrak{A} \sim_g \mathfrak{B}$ also is a guarded bisimulation $I : (\mathfrak{A}, R^{\mathfrak{A}}) \sim_g (\mathfrak{B}, R^{\mathfrak{B}})$, which is the case iff for all $\mathfrak{A}$, $\mathfrak{B}$, and all $I : \mathfrak{A} \sim_g \mathfrak{B}$ and $f : \boldsymbol{a} \mapsto \boldsymbol{b} \in I$, we have $\boldsymbol{a} \in R^{\mathfrak{A}} \iff \boldsymbol{b} \in R^{\mathfrak{B}}$, which is the same as to say that R is invariant for guarded bisimulation. $\square$

Corollary 3.5.6. *A* FO *formula is safe for guarded bisimulation iff it is equivalent to a variable-guarded* GF *formula.*

As we have seen above, every safe relation is guarded. This can be interpreted as strength of guarded bisimulation, in the sense of high expressivity, or strong discerning capabilities — safety immediately breaks down for non-guarded relations. We can also highlight this property from a different angle.

Definition 3.5.7. If R is a global relation let $\tau' = \tau \cup \{R'\}$ for some new relation symbol R'. If $\mathfrak{A}$ is a τ-structure then the R-expansion of $\mathfrak{A}$, denoted $\mathfrak{A}^R$, is the τ'-expansion of $\mathfrak{A}$ where R' is interpreted as $R^{\mathfrak{A}}$.

We denote by $\mathrm{GF}^m_{\infty,\omega}$ the m-variable guarded fragment of infinitary first-order logic. The following recapitulates guarded bisimulation safety from a different angle.

Lemma 3.5.8. *A global relation R is safe for guarded bisimulation iff all guarded $\mathrm{GF}^m_{\infty\,\omega}$ formulae in vocabulary τ' are guarded τ-bisimulation invariant on the class of all R-expansions.*

Proof. If R is safe for guarded bisimulation, then R is guarded and any guarded τ-bisimulation for $\mathfrak{A}, \mathfrak{B}$ is a guarded τ'-bisimulation for the R-expansions $\mathfrak{A}^R, \mathfrak{B}^R$. Also $GF^m_{\infty\,\omega}[\tau']$ is invariant for guarded τ'-bisimulation.

Let R *not* be safe for guarded bisimulation. Then there are τ-structures $\mathfrak{A}, \mathfrak{B}$, a guarded τ-bisimulation $I : \mathfrak{A} \sim_g \mathfrak{B}$ and an $f : \boldsymbol{a} \mapsto \boldsymbol{b} \in I$ such that $\boldsymbol{a} \in R^{\mathfrak{A}}$ and $\boldsymbol{b} \notin R^{\mathfrak{B}}$. Let $\Phi = \bigwedge\{\varphi(\boldsymbol{x}) \in \mathrm{GF}[\tau] : \mathfrak{A} \models \varphi(\boldsymbol{a})\}$. Then $(\Phi \to R)(\boldsymbol{x})$ is not guarded τ-bisimulation invariant on the R-expansions. □

The definition of safety is rather strict, so the question arises whether an alternative definition can give greater freedom for, and in particular allow non-guarded tuples in, the construction of safe relations. Although it is not required, guarded bisimulations may in general contain partial isomorphisms for non-guarded tuples. This suggests the following more liberal definition.

A global relation R is safe *for guarded bisimulation iff for all structures $\mathfrak{A}$ and $\mathfrak{B}$ there is a guarded bisimulation $I : \mathfrak{A} \sim_g \mathfrak{B}$ that is a guarded bisimulation $I : (\mathfrak{A}, R^{\mathfrak{A}}) \sim_g (\mathfrak{B}, R^{\mathfrak{B}})$, too.*

This can be equivalently phrased by requiring that the maximal guarded bisimulation between $\mathfrak{A}$ and $\mathfrak{B}$, the union of all guarded bisimulations, also be a bisimulation for the expansions. However allowing safe relations of this kind in guards immediately gives undecidability, e.g. by axiomatising transitivity of a binary relation, which was shown to yield an undecidable fragment in [24]. See also Section 6.4 for further discussions on guarded logics with transitive relations.

Lemma 3.5.9. *Let $\rho(\boldsymbol{x})$ be a variable-guarded $\mathrm{GF}[\tau]$ formula. Then for all $\varphi \in \mathrm{GF}[\tau']$ there is a $\varphi_\rho \in \mathrm{GF}[\tau]$ such that $\mathfrak{A}, \rho^{\mathfrak{A}} \models \varphi \iff \mathfrak{A} \models \varphi_\rho$ for all τ-structures $\mathfrak{A}$.*

Proof. Since ρ is variable-guarded, ρ is equivalent to $\mathbb{G}_\tau(\boldsymbol{x}) \wedge \rho(\boldsymbol{x})$. We transform all subformulae of φ of the form

$$\exists \boldsymbol{y}(R'(\boldsymbol{z}, \boldsymbol{y}) \wedge \chi(\boldsymbol{z}, \boldsymbol{y})),$$

where R' is used as guard, by inserting the definition of ρ for R' and obtain

$$\exists \boldsymbol{y}(\mathbb{G}_\tau(\boldsymbol{z}, \boldsymbol{y}) \wedge \rho(\boldsymbol{z}, \boldsymbol{y}) \wedge \chi(\boldsymbol{z}, \boldsymbol{y})).$$

All R' at non-guard position can be directly replaced by ρ. Eliminating all occurrences of R' yields a formula that is equivalent to a $\mathrm{GF}[\tau]$ formula φ_ρ. Obviously φ_ρ has the desired properties. □

Corollary 3.5.10. *Let $\rho(\boldsymbol{x})$ be a variable-guarded $\mu\mathrm{GF}[\tau]$ formula. Then for all $\varphi \in \mu\mathrm{GF}[\tau']$ there is a $\varphi_\rho \in \mu\mathrm{GF}[\tau]$ such that $\mathfrak{A}, \rho^{\mathfrak{A}} \models \varphi \iff \mathfrak{A} \models \varphi_\rho$ for all τ-structures $\mathfrak{A}$.*

The proof for GF goes through in verbatim for μGF.

3.6 Guarded Relational Algebra

In relational database systems, first-order logic is often encountered in the more procedural form of relational algebra. We consider a slightly modified version of the standard relational algebra that is tuned to capture precisely the variable-guarded guarded fragment definable relations. All relations that are safe for guarded bisimulation and first-order definable are thus expressible in what we call the *guarded relational algebra* GRA.

Syntax of GRA.

1. U is a term of width 1.

2. Every $R \in \tau$ is a term of width $r = \text{width}(R)$.

3. If M, N are terms of width k, then so are $M \setminus N$, $M \cup N$ and $M \cap N$.

4. If M is a term of width k and $i, j \leq k$, then so is $\sigma_{i=j}(M)$.

5. If M is a term of width k and $n_1, \ldots, n_j \leq k$,
   then $\pi_{n_1,\ldots,n_j}(M)$ is a term of width j.

6. If M has width k, N width ℓ and S has width $k + \ell$,
   then $M \times_S N$ is a term of width $k + \ell$.

The semantics is defined straightforwardly following the lead from standard relational algebra. Given a τ-structure $\mathfrak{A}$, the interpretation $N^{\mathfrak{A}}$ of a GRA term N in $\mathfrak{A}$ is obtained inductively according to the following rules.

Semantics of GRA.

1. $U^{\mathfrak{A}}$ is the universe A.

2. $R^{\mathfrak{A}}$ is interpreted as itself.

3. The set-theoretic operations $\setminus$, $\cap$ and $\cup$ are interpreted as usual.

4. $\sigma_{i=j}(M)^{\mathfrak{A}} = \{\boldsymbol{a} \in M^{\mathfrak{A}} \ : \ a_i = a_j\}$.

5. $\pi_{n_1,\ldots,n_j}(M) = \{(a_{n_1}, \ldots, a_{n_j}) \ : \ \boldsymbol{a} \in M^{\mathfrak{A}}\}$.

6. $(M \times_S N)^{\mathfrak{A}} = (M^{\mathfrak{A}} \times N^{\mathfrak{A}}) \cap S^{\mathfrak{A}}$.

Analogous to the formula $\mathbb{G}(\boldsymbol{x})$, for every k there is a GRA term G_k that defines the set of all guarded tuples of width k. For $k = 1$, let $G_k = U$, and for $k > 1$

$$G_k = \bigcup\{\pi_{\boldsymbol{n}}(R) \ : \ \boldsymbol{n} \in \{(n_1, \dots, n_k) \ : \ 1 < n_i < \mathrm{ar}(R), 1 < i < k\}, R \in \tau\}.$$

Lemma 3.6.1. *A global relation R is definable by a* GRA *term iff it is definable by a variable-guarded* GF *formula.*

Proof. We give inductive translations from GF to GRA, and back. Proving correctness is a simple exercise in chasing definitions.

Claim. For every variable-guarded GF formula φ there is an equivalent GRA term N_φ.

- If $\varphi(x_1, \dots, x_k) = (x_i = x_j)$, then $N_\varphi = \sigma_{i=j}(G_k)$.
- If $\varphi(x_1, \dots, x_k) = R(x_{i_1}, \dots, x_{i_r})$, then $N_\varphi = \pi_{j_1,\dots,j_k}(R \times_G U)$ where $j_a = b$ for $i_b = a$ and $j_a = r + 1$ otherwise.
- If $\varphi(x_1, \dots, x_k) = \alpha(x_1, \dots, x_k) \vee \beta(x_1, \dots, x_k)$, then $N_\varphi = N_\alpha \cup N_\beta$.
- If $\varphi(x_1, \dots, x_k) = \neg\alpha(x_1, \dots, x_k)$, then $N_\varphi = G_k \setminus N_\alpha$.
- If $\varphi(x_1, \dots, x_k) = \exists x_{k+1}(R(x_1, \dots, x_{k+1}) \wedge \chi(x_1, \dots, x_{k+1}))$, then $N_\varphi = \pi_{1,\dots,k}(R \cap N_\chi)$.

Claim. For every GRA term N there is an equivalent variable-guarded GF formula ψ_N.

1. $\psi_U(x) = (x = x)$.
2. $\psi_R(\boldsymbol{x}) = R(x_1, \dots, x_r)$ for $R \in \tau$.
3. $\psi_{(M\setminus N)}(\boldsymbol{x}) = \psi_M(\boldsymbol{x}) \wedge \neg\psi_N(\boldsymbol{x})$.
4. $\psi_{(M\cup N)}(\boldsymbol{x}) = \psi_M(\boldsymbol{x}) \vee \psi_N(\boldsymbol{x})$.
5. $\psi_{(M\cap N)}(\boldsymbol{x}) = \psi_M(\boldsymbol{x}) \wedge \psi_N(\boldsymbol{x})$.
6. $\psi_{\sigma_{i=j}(M)}(\boldsymbol{x}) = \psi_M(\boldsymbol{x}) \wedge x_i = x_j$.
7. $\psi_{\pi_{n_1,\dots,n_j}(M)}(x_1, \dots, x_j) = \exists y_1 \cdots \exists y_m \, (\psi_M(y_1, \dots, y_m) \wedge \bigwedge\{x_i = y_{n_i} \ : \ i \leq j\})$.
8. $\psi_{M\times_S N}(\boldsymbol{x}, \boldsymbol{y}) = \psi_S(\boldsymbol{x}, \boldsymbol{y}) \wedge \psi_M(\boldsymbol{x}) \wedge \psi_N(\boldsymbol{y})$.

Note that the formula in 7. is not syntactically a guarded formula, however it can be transformed into a guarded formula similarly to how the assertion $\mathbb{G}(\boldsymbol{x})$ was turned into a guarded formula in Example 3.2.1. $\Box$

Corollary 3.6.2. *A first-order definable global relation is safe for guarded bisimulation iff it is definable by a* GRA *term.*

3.7 Guarded Second-Order Logic

We now introduce a guarded variant of second-order logic, denoted GSO. In our preferred definition of GSO we simply use the syntax of ordinary second-order logic, but restrict it semantically by the stipulation that all second-order quantifiers range just over *guarded relations* in the sense of Definition 3.1.1 (v), rather than over arbitrary relations. Two other possible variants turn out to be semantically equivalent.

Definition 3.7.1. *Guarded second-order logic* GSO is second-order logic with *guarded semantics* for the second-order quantifiers: all second order quantifiers range over guarded relations (i.e., collections of guarded tuples).

The definition is motivated by the idea that GSO should relate to GF and μGF in the same way as MSO relates to ML and L_μ. Since μGF has fixed-point definitions (i.e. second-order constructs) for relations of any arity, MSO itself is clearly not the right logic. On the other side, full second-order logic is far too powerful — no matter whether we consider invariance under bisimulation or under guarded bisimulation.

Proposition 3.7.2. *There exist second-order sentences that are invariant under guarded bisimulation and not equivalent to any sentence in* μGF *or* GSO*. Further, the satisfiability problem for (guarded) bisimulation-invariant second-order logic is undecidable.*

Proof. We first observe that bisimilarity is a second-order property. For each finite vocabulary, there exists second-order formulae $\sigma(x,y)$ and $\sigma_g(x,y)$ such that $\mathfrak{A} \models \sigma(a,a')$ iff $\mathfrak{A},a \sim \mathfrak{A},a'$, and $\mathfrak{A} \models \sigma_g(a,a')$ iff $\mathfrak{A},a \sim_g \mathfrak{A},a'$. The formula $\sigma(x,y)$ requires that there is a binary equivalence relation B that contains (x,y) and satisfies the definition of a bisimulation. The same atomic propositions are true at the elements of every pair in B, and the back and forth property holds:

$$\begin{aligned}
&\exists B\big[B(xy) \land \forall xy(B(xx) \land B(xy) \to B(yx) \land \forall z(Bxz \land Bzy \to Bxy))\\
&\land \bigwedge_{b\in B} \forall xy(B(xy) \to (P_b(x) \leftrightarrow P_b(y)))\\
&\land \bigwedge_{a\in A} \forall xyx'(B(xy) \land E_a(xx') \to \exists y'(E_a(yy') \land B(x'y')))\\
&\land \bigwedge_{a\in A} \forall xyy'(B(xy) \land E_a(yy') \to \exists x'(E_a(xx') \land B(x'y')))\big]
\end{aligned}$$

Similarly one obtains $\sigma_g(x, y)$ by checking the existence of a guarded bisimulation. We omit the details, and refer to Section 4.1 which shows how guarded bisimulation on graphs can be reduced to modal bisimulation.

Let ψ be the following second-order sentence on the vocabulary $\{E, \mathrm{red}\}$ of coloured graphs

$$\psi \quad = \quad \forall x\forall y\big(\mathrm{red}(x) \land \mathrm{red}(y) \to \sigma_g(x, y)\big),$$

stating that every pair of red nodes is guarded bisimilar. Indeed ψ is invariant under guarded bisimulation. Suppose that $\mathfrak{A} \models \psi$ and $\mathfrak{A} \sim_g \mathfrak{B}$. Let b, b' be red nodes in $\mathfrak{B}$. By the back-condition of guarded bisimulations, there exist red nodes a, a' in $\mathfrak{A}$ such that $\mathfrak{A}, a \sim_g \mathfrak{B}, b$ and $\mathfrak{A}, a' \sim_g \mathfrak{B}, b'$. As $\mathfrak{A}, a \sim \mathfrak{A}, a'$ it follows that $\mathfrak{B}, b \sim_g \mathfrak{B}, b'$. Hence $\mathfrak{B} \models \psi$. On trees, ψ is even invariant under bisimulation.

To show that ψ is not equivalent to any GSO-sentence it suffices to consider ψ on $\mathfrak{T} = (V, E, \mathrm{red})$ where (V, E) is the complete binary tree. Clearly, $\mathfrak{T} \models \psi$ if, and only if, all red nodes in $\mathfrak{T}$ are on the same level. But on trees GSO and MSO coincide, see Section 3.7.1 below for a more general result, and it is well-known that the equal level predicate on the binary tree is not MSO-definable [58].

For the second claim we assume that the reader is familiar with the method of proving undecidability via domino problems [9]. With each domino system $\mathcal{D}$ one associates a formula which is satisfiable if and only if $\mathcal{D}$ admits a tiling of the infinite grid $(\mathbb{N} \times \mathbb{N}, N, E)$. Here we have to do this in second-order logic in a way that respects bisimilarity. A domino system $\mathcal{D}$ is a triple (D, V, H) where D is the set of domino types, and V and H are the vertical and horizontal compatibility relations for D. We use $\sigma_g(x, y)$ to construct a second-order sentence $\psi_{\mathcal{D}}$ in vocabulary $\{N, E, \boldsymbol{D}\}$, N for "North", E for "East", and unary relations $\boldsymbol{D} = (P_D)_{D\in\mathcal{D}}$, one for each domino, that expresses the domino condition,

- every point has an N-successor and an E-successor,
- if $x \sim_g y$ then all N-successors of x are bisimilar to all N-successors of y; and similarly for E.
- for each point x, its N-E-successors are bisimilar to its E-N-successors,
- each point carries precisely one tile, and the horizontal and vertical adjacency conditions imposed by $\mathcal{D}$ are respected.

Then the formula $\psi_{\mathcal{D}}$ formalises the above as follows.

$$\begin{aligned}
&\forall x \exists y N(xy) \wedge \forall x \exists y E(xy)\\
&\wedge \forall xy(\sigma_g(x,y) \rightarrow \forall z z'(N(xz) \wedge N(yz') \rightarrow \sigma_g(z,z')))\\
&\wedge \forall xy(\sigma_g(x,y) \rightarrow \forall z z'(E(xz) \wedge E(yz') \rightarrow \sigma_g(z,z')))\\
&\wedge \forall xyzy'z'(N(xy) \wedge E(xz) \wedge E(xy') \wedge N(y'z') \rightarrow \sigma_g(z,z'))\\
&\wedge \forall x\big(\bigvee_{d \in D} P_d(x) \wedge \bigwedge_{d \in D} P_d(x) \rightarrow \bigwedge_{d' \neq d \in D} \neg P_{d'}(x)\big)\\
&\wedge \forall xy\big(E(xy) \rightarrow \bigwedge\{\neg(P_d(x) \wedge P_{d'}(y)) \ : \ d, d' \in D, (d,d') \notin H\}\big)\\
&\wedge \forall xy\big(N(xy) \rightarrow \bigwedge\{\neg(P_d(x) \wedge P_{d'}(y)) \ : \ d, d' \in D, (d,d') \notin V\}\big)
\end{aligned}$$

The models of $\psi_{\mathcal{D}}$ are precisely the transition systems that are bisimilar to an admissible tiling $(\mathbb{N} \times \mathbb{N}, N, E, \boldsymbol{D})$ of the infinite grid by $\mathcal{D}$. In particular $\psi_{\mathcal{D}}$ is invariant under bisimulation, and so the satisfiability problem for the class of these sentences is undecidable. □

We will show that the expressive power of GSO is between MSO and SO, and that GSO has the desired properties. Further, GSO defines a very robust level of expressiveness which is invariant under a number of changes in the actual formalisation and syntax. Indeed, we shall show that three natural candidates for a second-order guarded logic all have the same expressive power.

It should be stressed that for all considerations, guardedness (of sets, tuples, or relations) always refers to guardedness w.r.t. the underlying vocabulary; at no point will second-order variables be admitted as guards.

Theorem 3.7.3. *The following fragments of second-order logic are equally expressive:*

(1) The extension of GF *by full second-order quantification.*

(2) The extension of GF *by second-order quantification with guarded semantics.*

(3) Guarded second-order logic GSO.

Proof. Since (2) is more restrictive than either (1) or (3), it suffices to argue for translations from (1) and (3) into (2).

From (1) to (2). We show how to replace any second-order quantifier that ranges over arbitrary relations by second-order quantifiers that range over guarded relations. Let $\psi(X, \boldsymbol{x})$ be a formula according to (1), in which X is a k-ary second-order variable that does not occur as a guard, and consider the formula $\exists X \psi(X, \boldsymbol{x})$ in the sense of ordinary unconstrained semantics for the second-order quantification. Let the free first-order variables of ψ be $\boldsymbol{x} = (x_1, \ldots, x_n)$. We will modify ψ in a way that each atomic statement $X(\boldsymbol{y})$ is either eliminated or replaced by $X(\boldsymbol{y}) \wedge \mathbb{G}(\boldsymbol{y})$.

For any atom $X(\boldsymbol{y})$ in ψ within the scope of a first-order quantifier the definition of GF ensures that the variables $\boldsymbol{y}$ are included in the guard statement of that quantification[1]. These occurrences can therefore be replaced by $X(\boldsymbol{y}) \wedge \mathbb{G}(\boldsymbol{y})$ without changing the semantics of ψ.

On the other hand the quantifier-free (w.r.t. first-order quantification) part of ψ may depend on truth values of X for unguarded tuples, but with the restriction that the elements of these tuples are among the valuation of $\boldsymbol{x}$. This gives us a fixed finite number of unguarded tuples relevant to ψ. The truth value of X for these tuples can be explicitly enumerated. We formalise this as follows. Let $H = H(X, \boldsymbol{x})$ be the set of all $\{X\}$-structures with universe $\boldsymbol{x}$. Replace $\exists X \psi(X, \boldsymbol{x})$ by $\exists X \bigvee_{\mathfrak{H} \in H} \psi^{\mathfrak{H}}$, where $\psi^{\mathfrak{H}}$ is obtained from ψ by replacing all atoms $X(\boldsymbol{y})$ where $\boldsymbol{y} \subseteq \boldsymbol{x}$ with $(\mathbb{G}(\boldsymbol{y}) \wedge X(\boldsymbol{y})) \vee \neg\, \mathbb{G}(\boldsymbol{y})$, if $\mathfrak{H} \models X(\boldsymbol{y})$, and $(\mathbb{G}(\boldsymbol{y}) \wedge X(\boldsymbol{y}))$, otherwise. Remember the general assumption that all formulae are well named, whereby the free variables $\boldsymbol{x}$ are not quantified over in $\psi(\boldsymbol{x})$. Therefore the above replacement only covers atoms where all variables are free. The mixed cases are uncritical, since they are guarded.

The modified formula only contains occurrences of X that depend on the guarded part of X. They are guarded either explicitly through conjunction with $\mathbb{G}(\cdot)$, or implicitly, because they are in the scope of a guarded quantification. Any unrestricted valuation for X that satisfies $\psi(X, \boldsymbol{x})$ gives rise to an X' which is the guarded part of X, and an $\mathfrak{H}$ that coincides with X on

[1]More precisely, at least the guard statement of the *innermost* quantification above $X(\boldsymbol{y})$ includes all $\boldsymbol{y}$.

all atomic statements built with elements from $\boldsymbol{x}$, in a way that X' satisfies $\psi^{\mathfrak{H}}(X', \boldsymbol{x})$, and vice versa.

From (3) to (2). It suffices to show that unrestricted first-order quantification can be simulated by guarded (in fact: monadic) second-order quantification over GF. To this end, each element variable x is replaced by a set variable X, and we use the following rules for translating formulae $\varphi \in$ GSO to equivalent formulae $\varphi^{(2)}$ as required in (2). The transformation commutes with the Booleans and

$$\begin{aligned}
(x = y)^{(2)} &= \forall x(X(x) \leftrightarrow Y(x)) \\
(R\boldsymbol{x})^{(2)} &= (\exists \boldsymbol{x} \,.\, R(\boldsymbol{x})) \textstyle\bigwedge_i X_i x_i \\
(Z\boldsymbol{x})^{(2)} &= (\exists \boldsymbol{x} \,.\, \mathbb{G}(\boldsymbol{x})) \textstyle\bigwedge_i X_i(x_i) \wedge Z(\boldsymbol{x}) \\
(\exists x \varphi(x, \boldsymbol{y}))^{(2)} &= \exists X \big(\varphi(X, \boldsymbol{y}) \wedge \text{singleton}(X)\big)
\end{aligned}$$

where singleton(X) is a formula stating that X contains exactly one element,

$$\exists x X(x) \wedge \forall Y \big((\forall x (Y(x) \to X(x)) \to (\forall x\, \neg Y(x) \vee (\forall x\, Y(x) \leftrightarrow X(x)))\big).$$

Note that these translations, and in particular singleton(X), are in GF, since first-order quantification over a single free first-order variable x is always guarded (by $x = x$). $\square$

The semantic restriction of second-order quantification in GSO may alternatively be captured purely syntactically, for instance by only allowing occurrences of atoms $X(\boldsymbol{x})$ in conjunction with the GF-formula $\mathbb{G}(\boldsymbol{x})$, which says that $\boldsymbol{x}$ is a guarded tuple, thus effectively restricting X to its guarded part.

GSO includes full first-order logic. Hence GSO is undecidable and, unlike GF and μGF, not invariant under guarded bisimulation. Also note that, as singletons are always guarded, the monadic version of guarded second-order logic coincides with full MSO. Consequently, since MSO is strictly more expressive than FO, the same is true for GSO. Furthermore, we shall see in Lemma 3.7.5 below that GSO collapses to MSO over words.

Example 3.7.4. Hamiltonicity of graphs is GSO-definable as follows.

$$\begin{aligned}
\exists H \quad & \big(\forall x \forall y (H(xy) \to E(xy)) \;\wedge\; \forall x\, (\exists^{=1} y\, H(xy) \;\wedge\; \exists^{=1} y\, H(yx)) \wedge \\
& \forall X \, \big[\big(\exists x\, X(x) \wedge \forall x \forall y (H(xy) \wedge X(x) \to X(y))\big) \to \forall x\, X(x)\big]\big)
\end{aligned}$$

Evaluated on a graph $G = (V, E)$ the formula says that there exists an $H \subseteq E$ with unique successors and predecessors such that (V, H) is connected. This means that G has a Hamilton cycle.

As Hamiltonicity is known not to be expressible in MSO (see [19]), the example shows that GSO is more expressive than MSO. In fact, GSO lies strictly between MSO and SO.

Lemma 3.7.5. GSO *collapses to* MSO *over words.*

Proof. As usual we represent words $w = w_0 \cdots w_{n-1} \in B^*$ by word structures $\big(\{0, \dots, n-1\}, S, (P_b)_{b \in B}\big)$ where $S = \{(i, i+1) : i < n-1\}$ and P_b is the set of positions in the word carrying the letter b, i.e., $P_b = \{i < n : w_i = b\}$. The predicate of maximal arity in a word structure is the successor relation, so a guarded set is either a singleton or a set $\{i, i+1\}$. A guarded n-ary relation therefore contains only n-tuples $(a_1, \dots, a_n)$ such that $\{a_1, \dots, a_n\} \subseteq \{i, i+1\}$ for some i and can therefore be encoded by a sequence of monadic relations. For instance a guarded k-ary relation X can be represented by $\{X_u : u \in \{0,1\}^k\}$, where for each $u = u_0 \dots u_{k-1}$, $X_u = \{i < k-1 : (i + u_0, \dots, i + u_{k-1}) \in X\}$. Hence GSO is no more expressive than MSO over words, i.e. able to define exactly the regular languages. On the other hand full second-order logic is known to capture the polynomial-time hierarchy, which is of course much larger than the class of regular languages. □

Corollary 3.7.6. SO *is strictly more expressive than* GSO*.*

To summarise, we have the following hierarchy of logics.

Proposition 3.7.7. ML $\subsetneq$ GF $\subsetneq$ FO $\subsetneq$ MSO $\subsetneq$ GSO $\subsetneq$ SO.

Relationships between GSO and MSO are further explored in Section 3.7.1 below. In particular we shall see that a collapse, as exhibited over words in the proof of the previous Lemma, occurs over some important classes of graphs.

Unlike MSO, GSO is sufficiently expressive to capture μGF. By Proposition 3.3.1, we may w.l.o.g. consider μGF formulae whose fixed point applications are strictly guarded so that these fixed points are themselves guarded relations. It is clear that such fixed points are definable within GSO by means of the usual second-order characterisation of least and greatest fixed points: $[\mathbf{lfp}\, X\boldsymbol{x} \,.\, \varphi(X, \boldsymbol{x})](\boldsymbol{y}) \equiv \forall X \big[\big((\forall \boldsymbol{x}. \mathbb{G}(\boldsymbol{x}))(\varphi(X, \boldsymbol{x}) \to X\boldsymbol{x})\big) \to X\boldsymbol{y}\big]$.

It is also clear that GSO is strictly more powerful than μGF. For instance, as GSO includes all of MSO, it is neither decidable nor invariant under guarded bisimulation.

Proposition 3.7.8. μGF $\subsetneq$ GSO.

3.7.1 GSO **versus** MSO

We have seen that GSO is a strictly more expressive logic than MSO. However, GSO may nevertheless be viewed as a monadic logic, but over a different presentation of structures, the incidence graphs. We first explain the notion of the incidence graph of a relational structure and then prove the following natural characterisation result.

Theorem 3.7.9. *A property of relational structures is definable in* GSO *iff the corresponding property of their incidence graphs is definable in* MSO.

Incidence Graphs. Let $\tau = \{R_1, \dots, R_m\}$ be any finite relational vocabulary. With each τ-structure $\mathfrak{A} = (A, R_1, \dots, R_m)$ we associate a many-sorted structure $\mathfrak{A}^*$ with universe $A^* = (A \mathbin{\dot\cup} R_1 \mathbin{\dot\cup} \dots \mathbin{\dot\cup} R_m)$. To put it differently, $\mathfrak{A}^*$ contains for every $R \in \tau$ and for every tuple $\boldsymbol{a} \in R$ a new element $e[R\boldsymbol{a}]$. The relations of $\mathfrak{A}^*$ are binary incidence relations $\mathrm{Inc}_{R,j}$ for $R \in \tau$, $1 \le j \le \mathrm{ar}(R)$, recording identities between components of relational edges and base elements. More formally,

$$\begin{aligned} \mathfrak{A}^* &= \big(A^*, (\mathrm{Inc}_{R,j})_{R\in\tau, 1\le j\le \mathrm{ar}(R)}\big), \\ \text{where} \quad A^* &= A \mathbin{\dot\cup} \dot{\textstyle\bigcup}_{R\in\tau} \{e[R\boldsymbol{a}] \ : \ \boldsymbol{a} \in R\}, \\ \mathrm{Inc}_{R,j} &= \big\{(e, a) \in R \times A\colon \ \text{the } j\text{-th component of } e \text{ is } a\big\}. \end{aligned}$$

The vocabulary of $\mathfrak{A}^*$ is denoted τ^*.

First-order logic, monadic second-order logic, etc., on incidence graphs are assumed to be typed, that is, element variables and set variables always range over a particular sort. Hence quantifiers are of the form $(\exists\, x \in A)$ or $(\exists\, z \in R_i)$ for element variables and $(\exists\, X \subseteq A)$ or $(\exists\, Y \subseteq R_i)$ for monadic second-order variables.

Let now $X \subseteq A^k$ be any *guarded* relation on $\mathfrak{A}$. We represent X by a tuple $\boldsymbol{X}$ of monadic predicates $X_{R,\rho}$ on $\mathfrak{A}^*$ where $R \in \tau$ and $\rho : \{1, \dots, k\} \to \{1, \dots, \mathrm{ar}(R)\}$, and

$$X_{R,\rho} = \{e \in R : e = (e_1, \dots, e_s) \text{ such that } (e_{\rho(1)}, \dots, e_{\rho(k)}) \in X\}.$$

Conversely, every tuple $\boldsymbol{X}$ of monadic relations $X_{R,\rho} \subseteq R$ on $\mathfrak{A}^*$ represents a guarded k-ary relation

$$g(\boldsymbol{X}) = \bigcup_{R,\rho} \{(a_{\rho(1)}, \dots, a_{\rho(k)}) \in A^k : \boldsymbol{a} \in X_{R,\rho}\}$$

on $\mathfrak{A}$. Note that here we impose no consistency conditions on multiply represented tuples. We say that $\boldsymbol{X}$ is a representation of X if $g(\boldsymbol{X}) = X$. Every guarded relation on $\mathfrak{A}$ has a representation, but not necessarily a unique one.

The Translation. The following provides the proof for Theorem 3.7.9.

Proposition 3.7.10. (i) *For every formula $\psi(\boldsymbol{y}) \in \mathrm{GSO}(\tau)$ there exists a formula $\psi^*(\boldsymbol{y}) \in \mathrm{MSO}(\tau^*)$ such that for every τ-structure $\mathfrak{A}$ and every guarded tuple $\boldsymbol{a}$ in $\mathfrak{A}$*

$$\mathfrak{A} \models \psi(\boldsymbol{a}) \iff \mathfrak{A}^* \models \psi^*(\boldsymbol{a}).$$

(ii) *Conversely, for every formula $\psi(\boldsymbol{y}) \in \mathrm{MSO}(\tau^*)$, with variables $\boldsymbol{y}$ ranging over the universe-sort, there exists a formula $\psi^{\#}(\boldsymbol{y}) \in \mathrm{GSO}(\tau)$ such that for every τ-structure $\mathfrak{A}$ and every guarded tuple $\boldsymbol{a}$ in $\mathfrak{A}$*

$$\mathfrak{A}^* \models \psi(\boldsymbol{a}) \iff \mathfrak{A} \models \psi^{\#}(\boldsymbol{a}).$$

Proof. We first describe the translation from $\mathrm{GSO}(\tau)$ to $\mathrm{MSO}(\tau^*)$ required in (i).

(1) $(R\boldsymbol{y})^* = (\exists\, z \in R) \bigwedge_i \mathrm{Inc}_{R,i}(z, y_i)$.

(2) $(x = y)^* = (x = y)$.

(3) $(X\boldsymbol{y})^* = \bigvee_{R,\rho} (\exists\, z \in X_{R,\rho}) \bigwedge_i \mathrm{Inc}_{R,\rho(i)}(z, y_i)$.

(4) $(\neg\varphi)^* = \neg\varphi^*$ and $(\varphi \wedge \vartheta)^* = \varphi^* \wedge \vartheta^*$.

(5) $(\exists y \varphi)^* = (\exists\, y \in A)\varphi^*$.

(6) $(\exists X \varphi)^* = (\exists \boldsymbol{X} \,.\, \bigwedge_{R,\rho} X_{R,\rho} \subseteq R)\varphi^*$.

The converse translation from $\mathrm{MSO}(\tau^*)$ to $\mathrm{GSO}(\tau)$, as required for (ii), is defined as follows. Each variable z that ranges over a relational sort R (where R is k-ary) is replaced by a k-tuple $\boldsymbol{z}$ of variables. Similarly, each set variable $X \subseteq R$ is replaced by a k-ary relation symbol $X^{(k)}$.

(1) For $\varphi = \mathrm{Inc}_{R,j}(z, x)$, let $\varphi^{\#} = R\boldsymbol{z} \wedge z_j = x$.

(2) For a monadic second-order variable $X \subseteq A$, the atom Xy remains unchanged, whereas for $X \subseteq R$, Xz is replaced by $X^{(k)}\boldsymbol{z}$.

(3) Boolean combinations are translated in the obvious way.

(4) First-order quantification: $(\exists\, x \in A)\varphi$ is translated into $\exists x \varphi^{\#}$, whereas $(\exists\, z \in R)\varphi$ goes to $\exists \boldsymbol{z}(R\boldsymbol{z} \wedge \varphi^{\#})$.

(5) Second-order quantification: $(\exists\, X \subseteq A)\,\varphi$ corresponds with $\exists X \varphi^{\#}$, whereas $(\exists\, X \subseteq R)\varphi$ (with k-ary R) is replaced by a relativised quantification over the k-ary relation $X^{(k)}$, namely

$$\exists X^{(k)}(\forall \boldsymbol{y}(X^{(k)}\boldsymbol{y} \rightarrow R\boldsymbol{y}) \wedge \varphi^{\#}).$$

The correctness of both translations is again shown by straightforward induction. □

Incidence Graphs and Query Evaluation. The performance of a number of algorithms related to monadic second-order logic or to guarded logics depends strongly on the tree width of the input structure. Therefore it may often be interesting to work with incidence graphs rather than the usual relational structures. Indeed the tree width of $\mathfrak{A}^*$ is at most the tree width of $\mathfrak{A}$ plus one. On the other side, $\mathfrak{A}^*$ can have much smaller tree width than $\mathfrak{A}$ in cases where $\mathfrak{A}$ consists of 'tree-like' relations of high arity.

GSO **on Graphs.** Besides the usual variant of monadic second-order logic on graphs, Courcelle has studied a more powerful variant, called MSO_2, allowing quantification not only over sets of vertices but also over sets of edges (see e.g. [13] and the references there). It is not difficult to see that MSO_2 is actually equivalent to GSO on graphs. Courcelle's results show that, despite GSO being more expressive than MSO in general, there are a number of classes of graphs where the two logics are equivalent. This collapse occurs in particular over graphs of bounded degree, graphs of bounded tree width, planar graphs and graphs with an excluded minor. A general result covering and generalising all these graph classes has recently been proved in [14].

Theorem 3.7.11 (Courcelle). GSO *collapses to* MSO *on every class $\mathcal{C}$ of graphs that is closed under taking subgraphs and contains only k-sparse graphs, for some $k \in \mathbb{N}$, (i.e. $|E| \leq k|V|$ for every graph $G = (V, E) \in \mathcal{C}$).*

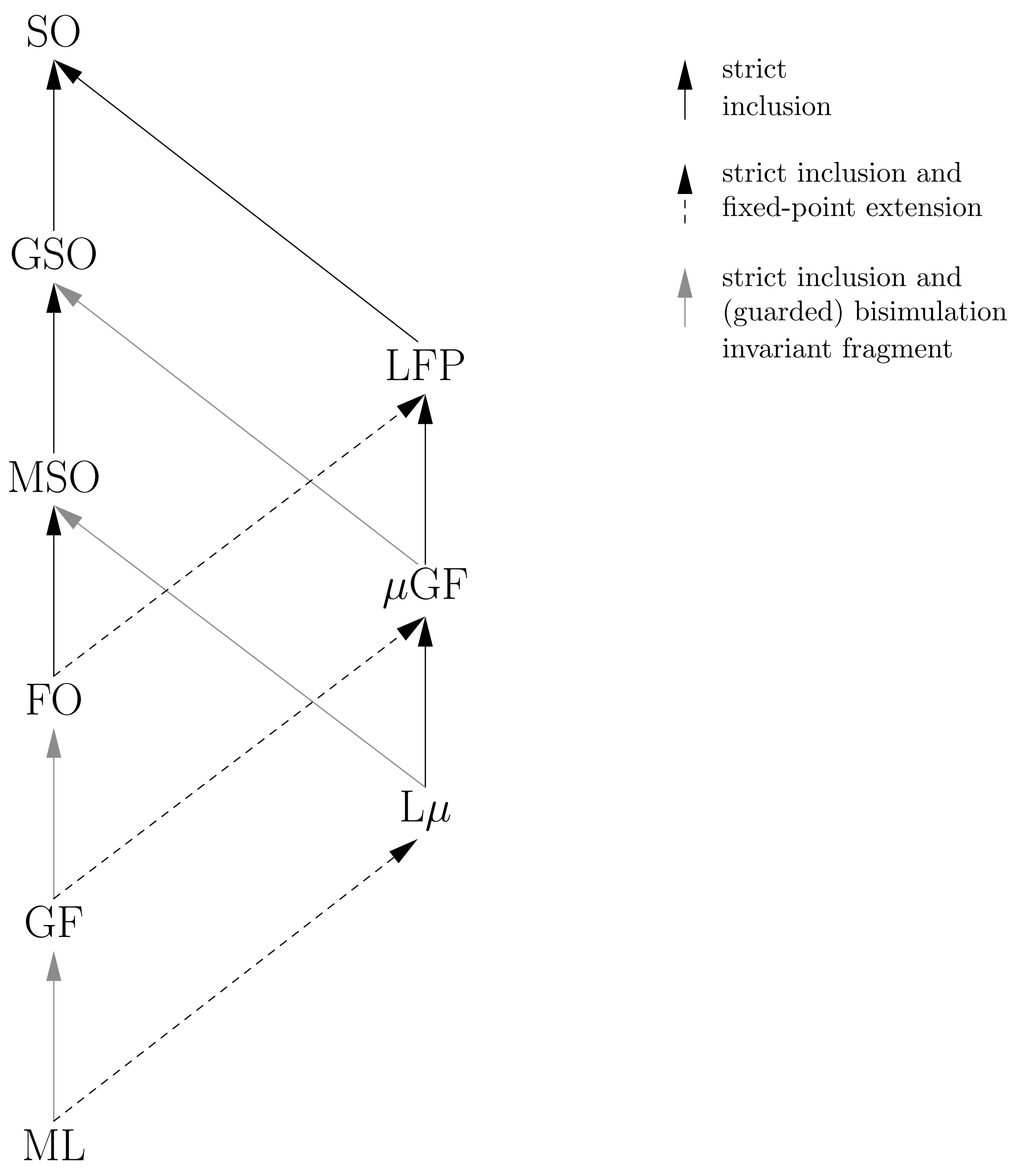
SO
GSO
LFP
MSO
μGF
FO
Lμ
GF
ML
strict inclusion
strict inclusion and fixed-point extension
strict inclusion and (guarded) bisimulation invariant fragment

Figure 3.2: Inclusion map of the major logics considered in this work.

Chapter 4

Structures, Games and Graphs

Guarded bisimulation is for the guarded world what bisimulation is for the modal setting. This correspondence is put on a precise footing in a framework of model theoretic translations.

We associate relational structures with transition systems and vice versa, in such a way that guarded bisimulation classes of relational structures correspond to bisimulation classes of graphs, and often trees. This will pave the way to go back and forth between the guarded world (relational structures, guarded bisimulation equivalence, guarded logics) and the modal world (transition systems and trees, bisimulation, modal logics).

The relevant back and forth translations from structures to trees and back from trees to structures will first be presented here and in Section 8.1 for the parameter free case which will support the proof of this theorem for sentences. This case is conceptually and notationally easier. As we shall indicate in Section 8.2, the treatment given here extends easily to cover also the case of variable-guarded formulae.

4.1 Guarded Logics on Graphs

Guarded logics extend modal logics to a general relational setting. Nevertheless the question arises whether, and to what extent, guarded logics are also more powerful on the structures on which modal logics operate, i.e. on transition systems or, equivalently, vertex and edge labelled graphs. Since

ML, L_μ, and, more generally, standard modal bisimulation equivalence can only "see" the connected component of the regarded node(s), we restrict attention to *connected* graphs.

Atomic Expansions of Graphs. Consider graphs given in our usual form $\mathfrak{G} = (V, (E_a)_{a\in A}, (P_b)_{b\in B})$ in a vocabulary τ consisting of binary relations E_a and unary relations P_b. Let

$$\Phi = \{(\neg)E_axy \ : \ a \in A\} \cup \{(\neg)E_ayx \ : \ a \in A\}.$$

The set T of *edge types* consists of all maximal consistent subsets of Φ that contain at least one positive literal. Note that the purely negative type of non-connected nodes is explicitly excluded. The edge type *realised* by a guarded pair of nodes u, v, $u \neq v$ of $\mathfrak{G}$ is

$$t(u, v) = \{\varphi(x, y) \in \Phi \ : \ \mathfrak{G} \models \varphi(u, v)\}.$$

Definition 4.1.1. The *atomic expansion* $\mathfrak{G}^+$ of the graph $\mathfrak{G}$ is the expansion of $\mathfrak{G}$ by new edge relations F_t for every edge type $t \in T$ and new unary relations Q_a for each $a \in A$. These are interpreted as

$$\begin{aligned} F_t &= \{(u, v) \ : \ u \neq v \text{ and } (u, v) \text{ realises } t\}, \\ Q_a &= \{u \ : \ \mathfrak{G} \models E_auu\}. \end{aligned}$$

Recall that we call a relational structure *connected* if its Gaifman graph is. That is $\mathfrak{G} = (V, (E_a)_{a\in A}, (P_b)_{b\in B})$ is connected if the undirected graph (V, E) with $E = \bigcup_{a\in A}(E_a \cup E_a^{-1})$ is connected in the usual sense. We first observe that guarded bisimulation on connected graphs is equivalent to ordinary bisimulation on the corresponding atomic expansions.

Proposition 4.1.2. *Let $\mathfrak{G}, \mathfrak{H}$ be connected graphs with nodes $g \in \mathfrak{G}, \mathfrak{H} \in H$. The following are equivalent.*

(i) $\mathfrak{G}, g \sim_g \mathfrak{H}, h$.
(ii) $\mathfrak{G}^+, g \sim_g \mathfrak{H}^+, h$.
(iii) $\mathfrak{G}^+, g \sim \mathfrak{H}^+, h$.

Proof. (i) $\Leftrightarrow$ (ii). Note that a set is guarded in a graph $\mathfrak{G}$ iff it is guarded in the atomic expansion $\mathfrak{G}^+$. It is immediate that any partial bijection $f : \mathfrak{G} \to \mathfrak{H}$ is a partial isomorphism for $\mathfrak{G}$ and $\mathfrak{H}$ iff it is one for $\mathfrak{G}^+$ and $\mathfrak{H}^+$. Consequently, a set of partial isomorphisms I is a guarded bisimulation between $\mathfrak{G}$ and $\mathfrak{H}$ iff it is one for $\mathfrak{G}^+$ and $\mathfrak{H}^+$.

(iii) $\Rightarrow$ (ii). Let $Z : \mathfrak{G}^+ \sim \mathfrak{H}^+$ be a bisimulation with $(g, h) \in Z$. Since $\mathfrak{G}^+$ and $\mathfrak{H}^+$ are connected, Z is in fact a total bisimulation.

Let I be the following collection of partial bijections (of sizes 1 or 2) from $\mathfrak{G}^+$ to $\mathfrak{H}^+$: $f = \{(x_1, y_1)\} \in I$ for all $(x_1, y_1) \in Z$; $f = \{(x_1, y_1), (x_2, y_2)\} \in I$ for all $(x_1, y_1), (x_2, y_2) \in Z$ such that $(x_1, x_2), (y_1, y_2) \in F_t$ for the same $t \in T$.

It is easily checked that these maps are in fact partial isomorphisms between $\mathfrak{G}^+$ and $\mathfrak{H}^+$. By assumption we also have $\{(g, h)\} \in I$.

Claim. I is a guarded bisimulation between $\mathfrak{G}^+$ and $\mathfrak{H}^+$.

We have to show that I satisfies the back and forth conditions. By symmetry it suffices to consider the forth property.

Let $f : X \to Y \in I$. We distinguish cases, according to whether the newly chosen guarded set X' has size one or two, and whether X and X' are disjoint. The case $X = X'$ is trivial.

Suppose $X' = \{x'\}$ is a singleton. We need to find an appropriate $y' \in \mathfrak{H}^+$ such that $f' : x' \mapsto y'$ is a partial isomorphism in I, compatible with f. If $x' \notin X$, we find a y' such that $(x', y') \in Z$, by totality of Z. Then $f' = \{(x', y')\} \in I$ is as desired. If $x' \in X$ we let f' be the restriction of f to X'.

Now let $X' = \{x'_1, x'_2\}$ contain two elements, of which one could occur in X too. We assume x'_1 to be that element (if any), and let $X'' = \{x'_1\}$. As above we argue that there is a y'_1 such that $f'' = \{(x'_1, y'_1)\} \in I$. X' is guarded in $\mathfrak{G}^+$, so there is a $t \in T$ such that $(x'_1, x'_2) \in F_t$. The forth property of the bisimulation Z yields a y'_2 that is connected to y'_1 via an F_t edge and $(x'_2, y'_2) \in Z$. It follows that $f = \{(x'_1, y'_1), (x'_2, y'_2)\} \in I$ is as required.

Note that for any pair of adjacent vertices $x \neq y$ there is a unique $t \in T$ such that $(x, y) \in F_t$. Clearly this t also determines the unique $t^{-1} \in T$ for which $(y, x) \in F_{t^{-1}}$. The chosen format is such that it is sufficient to consider only one of the edges involved, F_t or $F_{t^{-1}}$, between x and y without missing any atomic information.

(ii) $\Rightarrow$ (iii). Let $I : \mathfrak{G} \sim_g \mathfrak{H}$ be a guarded bisimulation connecting g to h. Define Z as the union of all graphs of the functions in I.

Claim. Z is a total bisimulation and $(g, h) \in Z$.

It is immediate that Z is total, that $(g, h) \in Z$ and that the atomic types of any $(x, y) \in Z$ match as required. We show that Z satisfies the back and forth conditions.

Let $(x, y) \in Z$ and let $(x, x') \in F_t$ for some $t \in T$. Since $(x, y) \in Z$, the function sending x to y already was a partial isomorphism in I. Further, as (x, x') is guarded there is an $f \in I$ with domain $\{x, x'\}$ that also sends x to y. For the image of y under f, denoted y', we have that $(y, y') \in Z$ by definition.

The case of E_a edges and the back condition are shown similarly. $\square$

Note that the assumption that $\mathfrak{G}$ and $\mathfrak{H}$ are connected can be dropped if we require the modal bisimulation to be total. The proof of Proposition 4.1.2 then effectively establishes the following.

Corollary 4.1.3. *Let $\mathfrak{G}, \mathfrak{H}$ be graphs. The following are equivalent.*

(i) $\mathfrak{G} \sim_g \mathfrak{H}$.
(ii) $\mathfrak{G}^+ \sim_g \mathfrak{H}^+$.
(iii) $\mathfrak{G}^+ \sim \mathfrak{H}^+$ *via some total bisimulation.*

We next show that guarded logics on connected graphs essentially reduce to modal logics on the corresponding atomic expansions. This gives the answer to the question mentioned at the beginning of this section. On graphs, guarded logics extend modal logic precisely by the power to use all quantifier-free definable guarded binary predicates. For GF, the corresponding modal logic is not ML itself, but ML with the universal modality, i.e. ML + $\blacksquare$ as introduced in Section 2.4. For μGF we show the analogous statement for the fragment that allows only monadic fixed points, which we denote by $\mu\text{GF}_{\text{mon}}$. The question of whether this can be generalised to full μGF is open.

The following Lemma is a direct consequence of the syntax of $\mu\text{GF}_{\text{mon}}$ formulae.

Lemma 4.1.4. *Every formula $\varphi(x, y) \in \mu\text{GF}_{mon}$ is (on graphs) equivalent to a Boolean combination of quantifier free formulae and formulae in at most one free variable.*

We will present formulae with two free variables in a disjunctive normal form

$$\varphi(x, y) = \bigvee_i \big(\varphi_i^x(x) \wedge \varphi_i^y(y) \wedge \varphi_i^{xy}(x, y)\big)$$

where φ_i^{xy} contains only atoms whose free variables are exactly x and y, and $\varphi_i^{xy} \models x \neq y$.

Proposition 4.1.5. *A monadic query on connected graphs is definable in* GF *if, and only if, it is definable in* ML $+ \blacksquare$ *over the corresponding atomic expansions. Similarly, a monadic query on connected graphs is definable in* $\mu\mathrm{GF}_{mon}$ *if, and only if, it is definable in the modal μ-calculus on the corresponding atomic expansions.*

Proof. Let $\varphi(x) \in \mathrm{GF}[\tau]$ resp. $\varphi(x) \in \mu\mathrm{GF}_{\mathrm{mon}}$.

Note that any subformula of φ with two free variables necessarily occurs in the scope of a quantifier. This observation and Lemma 4.1.4 enable an explicit transformation.

Both translations require the following inductive rules. The modal quantifiers $[t]$ and $\langle t\rangle$ w.r.t. the new edge relations F_t are sufficient to simulate local guarded quantification[1].

$$\begin{array}{rcl}
(\neg\psi)^+ &=& \neg(\psi^+)\\
(\psi_1 \wedge \psi_2)^+ &=& \psi_1^+ \wedge \psi_2^+\\
(\psi_1 \vee \psi_2)^+ &=& \psi_1^+ \vee \psi_2^+\\
(P_b(y))^+ &=& P_b\\
(E_a(y,y))^+ &=& Q_a\\
(\forall y.\psi(xy))^+ &=& \bigvee\limits_i \big((\psi_i^x(x))^+ \wedge \bigwedge_{\{t\in T\,:\,t\models\psi_i^{xy}\}}\big([t](\psi_i^y(y))^+\big)\big)
\end{array}$$

We do not assume negation normal form and assume that the $\neg$ operator is used to convert between $\exists$ and $\forall$. The translation from GF into ML $+ \blacksquare$ is completed with the following rules.

$$\begin{array}{rcl}
(\forall x.\psi(x))^+ &=& \blacksquare((\psi(x))^+)\\
(\forall xy.\psi(xy))^+ &=& \blacksquare(\forall y.\psi(xy))^+
\end{array}$$

The translation from μGF into L_μ uses a fixed-point iteration to quantify over all nodes. Connectedness of the given graphs is necessary for capturing the universal modality in L_μ.

$$\begin{array}{rcl}
(X(x))^+ &=& X\\
([\mathbf{lfp}\ Xx\,.\,\psi](x))^+ &=& \mu X.(\psi(X,x)^+)\\
(\exists x.\psi(x))^+ &=& \mu X.\big(\psi(x)^+ \vee \bigvee_{t\in T}\langle t\rangle X\big)\\
(\exists xy.\psi(xy))^+ &=& \mu X.\big((\forall y.\psi(xy))^+ \vee \bigvee_{t\in T}\langle t\rangle X\big)
\end{array}$$

[1]We abbreviate guarded quantifications $(\exists y.\alpha(xy))\varphi(xy)$ as $\exists y.\psi(xy)$, i.e. $\psi(xy) = \alpha(xy) \wedge \varphi(xy)$, etc.

By a straightforward induction one shows that for all τ-graphs G and all nodes g, $\mathfrak{G} \models \varphi(g)$ iff $\mathfrak{G}^+, g \models \varphi^+$. For the other direction it is immediate that any monadic query in $\mathrm{ML} + \blacksquare$ resp. L_μ on atomic expansions of graphs can be converted into a GF resp. $\mu\mathrm{GF}_{\mathrm{mon}}$ query on the original graphs. $\Box$

4.2 A Normal Form for Guarded Quantification

It will be convenient for our intended translation from the guarded world into the modal world to adopt a normalisation for guarded quantification resembling the first-order transcription of modal quantification that uses only two first-order variables, x and y. Recall that the crucial feature of that translation is how x and y are used in alternating fashion to capture the semantics of nested modal quantification, as in

$$[\Box\Diamond\Box P](x) \quad = \quad \forall y\big(Exy \to \exists x\big(Eyx \wedge \forall y(Exy \to Py)\big)\big).$$

Corresponding to the state variables x or y in ML we essentially want to use variable tuples for guarded lists $\boldsymbol{x}$ and $\boldsymbol{y}$ in GF. Corresponding to the transition relations Exy between states we record the overlap between components of $\boldsymbol{y}$ and components of $\boldsymbol{x}$ in the form of identities $y_i = x_j$.

Let $X = \{x_1, x_2, \ldots\}$ and $Y = \{y_1, y_2, \ldots\}$ be two disjoint sets of variables.

Definition 4.2.1. Let GF_X and GF_Y be inductively defined as follows.

(i) Every relational atomic formula α with $\mathrm{free}(\alpha) \subseteq X$ belongs to GF_X; every relational atomic formula α with $\mathrm{free}(\alpha) \subseteq Y$ belongs to GF_Y.

(ii) A Boolean combination of formulae in GF_X also belongs to GF_X; similarly for GF_Y.

(iii) Let $m, n \in \mathbb{N}$, $\boldsymbol{y} = (y_1, \ldots, y_n)$, and let ρ be any partial 1-1 map from $\{1, \ldots, n\}$ into $\{1, \ldots, m\}$. Then, for every guard $\alpha(y_1, \ldots, y_n)$ and $\varphi(y_1, \ldots, y_n) \in \mathrm{GF}_Y$ the following formula is in GF_X:

$$\big(\exists \boldsymbol{y}.\ \textstyle\bigwedge_{\rho(i)=j} y_i = x_j \wedge \bigwedge_{i \neq j} y_i \neq y_j \wedge \alpha(\boldsymbol{y})\big)\varphi(\boldsymbol{y}).$$

Interchanging the roles of X- and Y-variables we obtain formulae in GF_Y. Let $\mathrm{GF}_0 = \mathrm{GF}_X \dot\cup \mathrm{GF}_Y$ and let GSO_0 be the extension of GF_0 by second-order quantification over guarded relations.

In the sequel, we let $\exists^{\neq}\boldsymbol{y}\ldots$ be an abbreviation for $\exists \boldsymbol{y}\big(\bigwedge_{i\neq j} y_i \neq y_j \wedge \ldots\big)$ and denote relativised quantifications of the type

$$(\exists \boldsymbol{y}.\ \bigwedge_{\rho(i)=j} y_i = x_j \wedge \bigwedge_{i\neq j} y_i \neq y_j \wedge \alpha(\boldsymbol{y}))\varphi(\boldsymbol{y})$$

as used in GF_0 as

$$(\exists^{\neq}\boldsymbol{y}\,.\,\rho(\boldsymbol{x},\boldsymbol{y}) \wedge \alpha(\boldsymbol{y}))\varphi(\boldsymbol{y}).$$

Note that $(\exists^{\neq}\boldsymbol{y}\,.\,\rho(\boldsymbol{x},\boldsymbol{y}) \wedge \alpha(\boldsymbol{y}))\varphi(\boldsymbol{y})$ says that we can pass from a current tuple instantiating $\boldsymbol{x}$ to a guarded list $\boldsymbol{y} = (y_1,\ldots,y_n)$ which extends the subtuple $(x_{\rho(i)}) = (y_i)$ in such a manner that the $\boldsymbol{y}$ satisfy the guard $\alpha(\boldsymbol{y})$ and the formula $\varphi(\boldsymbol{y})$.

It should be noted that the formulae in GF_X and GF_Y are syntactically *not* in GF. It is clear, however, that these formulae are logically equivalent to guarded ones.

Proposition 4.2.2. *Every sentence in* GF *is equivalent to a sentence in* GF_0.

Proof. For an inductive argument, we consider formulae of GF and show the following.

Let $\varphi(\boldsymbol{z})$ be any formula in GF with free variables among $\boldsymbol{z}$. Let $\eta(\boldsymbol{z})$ be any complete quantifier free equality type on $\boldsymbol{z}$, and let $\boldsymbol{z}^\eta$ be an n-tuple of variables from $\boldsymbol{z}$ such that every $=$-class of η is represented exactly once. Then there is a formula $\varphi^\eta_X(\boldsymbol{x}) \in \mathrm{GF}_X$ such that

$$\eta(\boldsymbol{z}) \wedge \boldsymbol{x} = \boldsymbol{z}^\eta \models \varphi(\boldsymbol{z}) \leftrightarrow \varphi^\eta_X(\boldsymbol{x}),$$

and, similarly, a corresponding $\varphi^\eta_Y(\boldsymbol{y}) \in \mathrm{GF}_Y$.

The atomic and Boolean cases are trivial and it just remains to deal with guarded quantification. So let $\varphi(\boldsymbol{z})$ be of the form

$$\varphi(\boldsymbol{z}) = \big(\exists \boldsymbol{w}.\alpha(\boldsymbol{z},\boldsymbol{w})\big)\psi(\boldsymbol{z},\boldsymbol{w}).$$

For the given equality type $\eta = \eta(\boldsymbol{z})$, consider all equality types $\eta'(\boldsymbol{z},\boldsymbol{w})$ that are compatible with the given η. Clearly, $\varphi \wedge \eta$ is equivalent to a disjunction over all such η' of the form

$$\varphi(\boldsymbol{z}) \wedge \eta(\boldsymbol{z}) = \textstyle\bigvee_{\eta' \supseteq \eta}\big(\exists \boldsymbol{w}.\alpha(\boldsymbol{z},\boldsymbol{w})\big)\big(\eta'(\boldsymbol{z},\boldsymbol{w}) \wedge \psi(\boldsymbol{z},\boldsymbol{w})\big).$$

We may therefore consider just one such disjunct, for a fixed $\eta'(\boldsymbol{z},\boldsymbol{w})$. Let $(\boldsymbol{z},\boldsymbol{w})^{\eta'} = (u_1,\ldots,u_m) = \overline{u}$ be a tuple of representatives according to η', and let $\boldsymbol{y} = (y_1,\ldots,y_m)$.

According to the inductive hypothesis, there is some formula $\psi^{\eta'}_Y(y_1,\ldots,y_m)$ in GF_Y such that $\eta'(\boldsymbol{z},\boldsymbol{w}) \wedge \boldsymbol{y} = \overline{u} \models \psi(\boldsymbol{z},\boldsymbol{w}) \leftrightarrow \psi^{\eta'}_Y(\boldsymbol{y})$.

Let $\alpha'(\boldsymbol{y})$ be obtained from $\alpha(\boldsymbol{z},\boldsymbol{w})$ by replacing by y_i each variable in $\alpha(\boldsymbol{z},\boldsymbol{w})$ that is identified with u_i by η'. If $\boldsymbol{z} = (z_1,\dots,z_k)$, let ρ be the partial 1-1 map defined by $\rho(i) = j$ if η' identifies u_i with z_j. Then

$$\big(\exists \boldsymbol{w}.\alpha(\boldsymbol{z},\boldsymbol{w})\big)\big(\eta'(\boldsymbol{z},\boldsymbol{w}) \wedge \psi(\boldsymbol{z},\boldsymbol{w})\big) \equiv \big(\exists \boldsymbol{y}. \textstyle\bigwedge_{\rho(i)=j} y_i = z_j \wedge \alpha'(\boldsymbol{y})\big)\psi_Y^{\eta'}(\boldsymbol{y}).$$

It follows that, after substituting X-variables for $\boldsymbol{z}$, the formula

$$(\exists^{\neq} \boldsymbol{y}.\rho(\boldsymbol{z},\boldsymbol{y}) \wedge \alpha')\psi_Y^{\eta'}(\boldsymbol{y})$$

is in GF_X and satisfies the inductive claim. □

Recall that GSO_0 is the extension of GF_0 by second-order quantification over guarded relations. The proof of the proposition clearly extends to cover GSO_0, and most other conceivable guarded logics.

Corollary 4.2.3. *Every sentence in* GSO *is equivalent to a sentence in* GSO_0.

4.3 The Guarded Ehrenfeucht-Fraïssé Game

Guarded bisimulation equivalence may be described in terms of a natural Ehrenfeucht-Fraïssé game associated with GF. We indicate the characteristic features of the *guarded game*, as outlined in [1], and introduce a restricted or standardised version of the game that is closely related to the GF_0 normal form of Section 4.2.

The standard game for GF and guarded bisimulation equivalence is analogous to the usual pebble games for bounded-variable logics. The purpose of the finite ℓ-round game is to determine whether two structures $\mathfrak{A}$ and $\mathfrak{B}$ are indistinguishable in GF up to guarded quantifications of depth ℓ. The infinite game, which we are mainly interested in, allows us to determine whether $\mathfrak{A}$ and $\mathfrak{B}$ are guarded bisimulation equivalent, or indistinguishable in the infinitary variant of GF.

There are two players, Adam and Eve. Adam tries to show that $\mathfrak{A}$ and $\mathfrak{B}$ are inequivalent, while Eve attempts to show that they are equivalent. The task for Eve is to mimic moves that Adam makes in one of the structures with corresponding moves in the opposite structure.

During the game, pebbles can be placed on elements of the respective structures. The crucial restriction for the guarded game is that the currently marked elements in either structure must form a guarded set. After each round of the game both structures will carry pebbles with corresponding labels, so that the pebble placement induces a correspondence $\boldsymbol{a} \mapsto \boldsymbol{b}$, where $a \in A$ is associated to $b \in B$ if a and b are the positions of corresponding pebbles. Initially, i.e., before the first round, neither $\mathfrak{A}$ nor $\mathfrak{B}$ carry pebbles.

In each new round, Adam chooses one of the structures, possibly removes some of the pebbles, and then places some pebbles (from among those just removed or those not on the structure at the start of this round) on elements of this structure. The only constraint for Adam is that after this move all pebbles on this structure again form a guarded set. Eve answers by removing, from the other structure, precisely those pebbles corresponding to those first taken off in Adam's move, and then placing pebbles corresponding to those placed by Adam. Eve is required to make her choices such that the new correspondence $\boldsymbol{a} \mapsto \boldsymbol{b}$ between pebbled elements is a partial isomorphism. Note that in particular this also forces her to place her pebbles so that they form a guarded set. If no such move is available, then Eve loses and the game is over.

The whole game can be played with a fixed finite number of rounds, or as an infinite game. In the ℓ-round game, Eve wins a play if she was able to respond through all ℓ rounds; Adam only wins if Eve got stuck before completion of the last round. In the infinite game, an infinite play in which Eve can always respond to the next move of Adam, is won by Eve; Adam only wins if Eve gets stuck after some finite number of rounds.

The classical results on Ehrenfeucht-Fraïssé games (and their proofs) obviously translate to this guarded setting. Eve has a winning strategy in the ℓ-round game if $\mathfrak{A}$ and $\mathfrak{B}$ cannot be distinguished by any sentence of GF with nesting depth ℓ for guarded quantification; and Eve has a winning strategy in the infinite game if $\mathfrak{A}$ and $\mathfrak{B}$ cannot be distinguished by any sentence of GF_∞, infinitary guarded logic, or the guarded fragment of $\mathrm{L}_{\infty\omega}$. The latter, of course, is also equivalent to $\mathfrak{A} \sim_g \mathfrak{B}$. We shall here only use this characterisation of guarded bisimulation equivalence by means of the infinite game.

Proposition 4.3.1. Eve *has a winning strategy in the infinite guarded Ehrenfeucht-Fraïssé game on* $\mathfrak{A}$ *and* $\mathfrak{B}$ *iff* $\mathfrak{A} \sim_g \mathfrak{B}$.

This equivalence is straightforward, but becomes even more immediate if we look at a slight variation of the game. This variation – although equivalent to the standard version described above – is a closer analogy to the quantification pattern of GF_0, whereas the standard game is modelled after the standard quantification pattern in GF.

Let m be the width of the vocabulary τ, i.e., the maximal arity of relations in τ, which is also a bound on the size of guarded sets in τ-structures. It is not hard to see that the number of pebbles in each structure can w.l.o.g. be restricted to m. In fact, the constraint in terms of guarded sets means that whenever more than m pebbles are in the game some elements must be multiply pebbled. Whenever several pebbles are placed on the same elements, the multiplicities and pebbles concerned must be the same in both structures since the pebbles induce a partial isomorphism. It is then clear that we could w.l.o.g. force Adam to avoid multiple pebbling. One verifies inductively that if Adam can force a win in the old game then this is still true under this additional constraint. Together with the constraint that the set of pebbled elements always be a guarded set, this precisely means that we require the pebbled tuples to be guarded lists in the sense of Definition 3.1.1.

The analogy with quantification in GF_0 is now obvious:

- Quantification is over tuples that are guarded lists (with explicit mention of the ground atom that guards the component set); this corresponds to a new pebble placement.
- Constraints in terms of identities between newly quantified elements (quantified variables) and the previous elements (free variables) are described by a partial 1-1 map; this corresponds to the choice of pebbles that remain fixed in place during Adam's move.

With regard to the second correspondence it should be pointed out that both in the game and in GF_0 quantification nothing prevents Eve to put other pebbles back into positions that were just previously pebbled; in that sense Adam and GF can explicitly force some identifications, but not forbid others.

We thus obtain the following restricted version of the guarded game, which, as we have argued, is equivalent to the more liberal standard version.

Let $\mathfrak{A}$ and $\mathfrak{B}$ be τ-structures, τ finite and relational of width m. The restricted guarded game on $\mathfrak{A}$ and $\mathfrak{B}$ is played by two players, Adam and Eve, who take turns to position and relocate pebbles from two sets of m pebbles labelled $1, \dots, m$, one set for each structure. The rules of the game are such that after each round the groups of pebbles positioned in $\mathfrak{A}$ and $\mathfrak{B}$, respectively, label guarded lists $\boldsymbol{a}$ and $\boldsymbol{b}$ in such a way that the correspondence $\boldsymbol{a} \mapsto \boldsymbol{b}$ is a partial isomorphism. In each round, Adam has the choice in which of the two structures to play. In the chosen structure, Adam may remove some of the pebbles currently placed on that structure; Eve removes the corresponding pebbles in the opposite structure. Then Adam may place any number of pebbles back onto elements of the chosen structure in such a way that the overall tuple of pebbled positions again forms a guarded list; Eve must place corresponding pebbles onto elements of the opposite structure in such a way that the induced correspondence between the pebbled elements in $\mathfrak{A}$ and $\mathfrak{B}$ is a partial isomorphism. Eve loses if no such response is available. The game is infinite unless Eve loses in this way. Eve wins a play if she never gets stuck. We obtain the following correspondence between games and bisimulations.

Proposition 4.3.2. Eve *has a winning strategy in the restricted guarded game on* $\mathfrak{A}$ *and* $\mathfrak{B}$ *iff* $\mathfrak{A} \sim_g \mathfrak{B}$.

Proof. A guarded bisimulation in the sense of Definition 3.4.1 is nothing other than a formalisation of a (non-deterministic) winning strategy for Eve, and a winning strategy gives rise to a collection of partial isomorphisms that is a guarded bisimulation.

More explicitly, suppose that Eve has a winning strategy. Let I be the collection of all partial isomorphisms $f : \boldsymbol{a} \to \boldsymbol{b}$ obtained as

- $\boldsymbol{b} \in \mathfrak{B}$ is Eve's answer to Adam placing the pebbles on the guarded $\boldsymbol{a} \in \mathfrak{A}$,
- and the same for reversed roles of $\mathfrak{A}, \boldsymbol{a}$ and $\mathfrak{B}, \boldsymbol{b}$,

where Eve plays according to the given winning strategy. Using the definitions it is easy to see that the requirement for Eve to answer Adam's moves corresponds precisely to the back and forth condition of a guarded bisimulation. For example, let $f : \boldsymbol{a} \to \boldsymbol{b} \in I$ and suppose that $\boldsymbol{a}'\boldsymbol{c} \in \mathfrak{A}$ is guarded and $\boldsymbol{a}' \subseteq \boldsymbol{a}$, $\boldsymbol{c} \cap \boldsymbol{a} = \emptyset$. We set up the game on $\mathfrak{A}$ and $\mathfrak{B}$ with currently pebbled elements $\boldsymbol{a}$ and $\boldsymbol{b}$. Pretend that Adam removed the pebbles corresponding to $\boldsymbol{a} \setminus \boldsymbol{a}'$ from $\mathfrak{A}$, and placed some pebbles on the elements of $\boldsymbol{c}$. The winning strategy for Eve yields a tuple $\boldsymbol{d} \in \mathfrak{B}$ such that $\boldsymbol{a}'\boldsymbol{c} \mapsto \boldsymbol{d}$ is a partial isomorphism f' that, since the pebbles corresponding to $\boldsymbol{a}'$ on $\mathfrak{B}$ were not touched, coincides with f on $\boldsymbol{a}'$, as desired.

Similarly, a guarded bisimulation I gives rise to a strategy for Eve. □

We now use the intuition behind the game to introduce the following two structural transformations. One which abstracts from a given τ-structure

$\mathfrak{B}$ a *tree representation* $\mathfrak{T}(\mathfrak{B})$, which fully describes, as a transition system, the behaviour of $\mathfrak{B}$ in the guarded game, and thus characterises $\mathfrak{B}$ up to guarded bisimulation equivalence. Another one which, conversely, associates with a tree $\mathfrak{T}$ a τ-structure $\mathfrak{D}(\mathfrak{T})$, such that the game behaviour specified by $\mathfrak{T}$ is realised in the guarded game on $\mathfrak{D}(\mathfrak{T})$.

The guiding idea behind these transformations is that guarded bisimulations at the level of the τ-structures lift to ordinary bisimulations at the level of the abstracted transition systems. In particular, $\mathfrak{B}^* = \mathfrak{D}(\mathfrak{T}(\mathfrak{B})) \sim_g \mathfrak{B}$ will be a tree-like variant of $\mathfrak{B}$, a structure of bounded tree width as per Definition 3.4.3. This intuitively corresponds to a generalisation of the modal unravelling shown in Definition 2.3.3 to a *guarded unravelling* of $\mathfrak{B}$, as considered e.g. in [1] and [26].

4.4 From Structures to Trees

Recall from Definition 3.1.1 that a guarded list in $\mathfrak{B}$ is a tuple $(b_1, \dots, b_k)$ of *distinct* elements such that $\{b_1, \dots, b_k\}$ is a guarded set. We regard such guarded lists as descriptions of positions over $\mathfrak{B}$ in the restricted guarded game. The behaviour of $\mathfrak{B}$ w.r.t. the game can firstly be described as a game graph in the form of a transition system $\mathfrak{K}(\mathfrak{B})$. The states of the game graph are the game positions formalised as guarded lists over $\mathfrak{B}$. Guarded lists are really used as enumerations of guarded sets, and every guarded set of size k can thus be represented by $k!$ many different guarded lists. As there is in general no canonical way to single out any particular choice, we symmetrise the description of the game positions so as to include all these variant descriptions of the same guarded set, which are related by permutations that correspond to relabellings of the pebbles.

One could introduce a kind of ε-move in the restricted guarded game in which the current pebbles are permuted, according to some choice of a permutation made by Adam, which then has to be copied identically by Eve. Of course such moves would be a vacuous addition to the game, as far as winning or losing the game is concerned. For the resulting game graph, however, they allow for a locally more symmetric description.

The information to be recorded in each node $v = (b_1, \dots, b_k)$ of the game graph is precisely the isomorphism type of the guarded substructure induced on $\{b_1, \dots, b_k\}$ (cf. Definition 3.1.1 (ii)). The transition relations of the game graph describe possible moves and record the constraints imposed on a move from $\boldsymbol{c} = (c_1, \ldots, c_l)$ to $\boldsymbol{d} = (d_1, \ldots, d_k)$ by some choice of elements that have to remain fixed (according to Adam's choice).

With the finite relational vocabulary τ of width m, associate the following vocabulary $\tilde{\tau}$ for the associated transition systems and trees. Let S be the set of all guarded τ-structures $\mathfrak{A}$ with universes $\{1, \ldots, k\}$, $1 \leq k \leq m$, plus the empty structure $\mathfrak{A} = \emptyset$ for $k = 0$. Let F be the set of all partial 1-1 maps ρ from $\{1, \ldots, m\}$ to $\{1, \ldots, m\}$. The associated vocabulary $\tilde{\tau}$ has predicates

$P_{\mathfrak{A}}$	(monadic)	for $\mathfrak{A} \in S$,
E_ρ	(binary)	for $\rho \in F$.

If $\mathfrak{B}$ is a τ-structure, we write $G(\mathfrak{B})$ for the set of guarded lists over $\mathfrak{B}$,

$$G(\mathfrak{B}) = \big\{\boldsymbol{b} \ : \ \boldsymbol{b} \text{ a guarded list in } \mathfrak{B}\big\},$$

where we recall from Definition 3.1.1 that this includes the empty list $\emptyset$.

Definition 4.4.1. The transition system $\mathfrak{K}(\mathfrak{B})$ is the following $\tilde{\tau}$-structure

$$\begin{aligned} \mathfrak{K}(\mathfrak{B}) &= \big(G(\mathfrak{B}), (E_\rho)_{\rho \in F}, (P_{\mathfrak{A}})_{\mathfrak{A} \in S}\big) \\ E_\rho &= \big\{((c_1, \ldots, c_l), (d_1, \ldots, d_k)) \ : \ d_j = c_{\rho(j)} \text{ for all } j \in \mathrm{dom}(\rho)\big\} \\ P_{\mathfrak{A}} &= \big\{(b_1, \ldots, b_k) \ : \ \mathfrak{B}|_{\{b_1, \ldots, b_k\}} \simeq \mathfrak{A} \text{ via } b_i \mapsto i\big\}. \end{aligned}$$

Note that we have multiple transitions between nodes; in particular $E_{\rho'} \subseteq E_\rho$ for all $\rho \subseteq \rho'$.

The transition relations of $\mathfrak{K}(\mathfrak{B})$ reflect possible moves over $\mathfrak{B}$ in the restricted guarded game. The choice of the E_ρ and the $P_{\mathfrak{A}}$ is such that bisimulations between two transition systems $\mathfrak{K}(\mathfrak{B})$ and $\mathfrak{K}(\mathfrak{B}')$ capture winning strategies for Eve in the restricted game on $\mathfrak{B}$ and $\mathfrak{B}'$. To see this, consider a game position with pebbles placed on $\boldsymbol{b}$ in $\mathfrak{B}$ and on $\boldsymbol{b}'$ in $\mathfrak{B}'$. If $\mathfrak{K}(\mathfrak{B}), \boldsymbol{b} \sim \mathfrak{K}(\mathfrak{B}'), \boldsymbol{b}'$, then $\boldsymbol{b} \mapsto \boldsymbol{b}'$ must be a partial isomorphism since $\boldsymbol{b}$ and $\boldsymbol{b}'$ are labelled by the same $P_{\mathfrak{A}}$.

Now consider a move in the game in which Adam plays in $\mathfrak{B}$ say, from some position $\boldsymbol{c} = (c_1, \ldots, c_l)$ to a new position $\boldsymbol{d} = (d_1, \ldots, d_k)$ over $\mathfrak{B}$. Up to possible permutations in the enumeration of these guarded lists, or a consistent relabelling of pebbles over both structures, this relocation can have been effected by leaving put any subset of pebbles that mark elements in $\{c_1, \ldots, c_l\} \cap \{d_1, \ldots, d_k\}$.

Suppose Adam left put pebbles on positions $\boldsymbol{c}_0 \subseteq \boldsymbol{c}$. Then there is some partial 1-1 map ρ from $\{1, \ldots, k\}$ to $\{1, \ldots, l\}$ such that $\boldsymbol{c}_0 = (d_{\rho(j)})_{j \in \mathrm{dom}(\rho)}$.

It follows that $(\boldsymbol{c},\boldsymbol{d}) \in E_\rho$ in $\mathfrak{K}(\mathfrak{B})$. If now $\mathfrak{K}(\mathfrak{B}),\boldsymbol{c} \sim \mathfrak{K}(\mathfrak{B}'),\boldsymbol{c}'$, then an application of the forth property yields a position $\boldsymbol{d}'$ over $\mathfrak{B}'$ for which $(\boldsymbol{c}',\boldsymbol{d}') \in E_\rho$ in $\mathfrak{K}(\mathfrak{B}')$ and $\mathfrak{K}(\mathfrak{B}),\boldsymbol{d} \sim \mathfrak{K}(\mathfrak{B}'),\boldsymbol{d}'$. It follows that Eve can make a move to $\boldsymbol{d}'$ in $\mathfrak{B}'$, which satisfies both the constraints regarding pebbles that were fixed by I, and the requirement that the resulting correspondence $\boldsymbol{d} \mapsto \boldsymbol{d}'$ is again a partial isomorphism. Inductively we thus obtain the following.

Lemma 4.4.2. *For τ-structures $\mathfrak{B}$ and $\mathfrak{B}'$: $\mathfrak{B} \sim_g \mathfrak{B}'$ iff $\mathfrak{K}(\mathfrak{B}) \sim \mathfrak{K}(\mathfrak{B}')$.*

The desired tree representation $\mathfrak{T}(\mathfrak{B})$ of (the game on) $\mathfrak{B}$ is now obtained as an unravelling of the transition system $\mathfrak{K}(\mathfrak{B})$ from the empty list. Note that the empty list $\emptyset \in G(\mathfrak{B})$ corresponds to the start position of the game, with no pebbles placed. Compare Definition 2.3.3 for unravellings.

The set V of nodes of $\mathfrak{T}(\mathfrak{B})$ is the set of all sequences $g_0\rho_1g_1\rho_2g_2\dots g_n$ where all $g_i \in G(\mathfrak{B})$, with $g_0 = \emptyset$ (the empty guarded list) and for all $g_i\rho_{i+1}g_{i+1}$ we have

(i) $\rho_{i+1} \subseteq \{1,\dots,|g_{i+1}|\} \times \{1,\dots,|g_i|\}$,

(ii) for all $j \in \mathrm{dom}(\rho)$: the j-th element at g_{i+1} is the same as the $\rho(j)$-th element at g_i.

According to Definition 2.3.3, the natural projection $\pi : \mathfrak{T}(\mathfrak{B}) \to \mathfrak{K}(\mathfrak{B})$ maps a node $v = g_0\dots g_n$ to the guarded list $g_n = (b_1,\dots,b_k)$ in $\mathfrak{B}$ in which the sequence v terminates.

We associate some relevant information in $\pi(v)$ with v itself. In particular we let $|v| = |\pi(v)| = k$ (the size of v) and, if $\pi(v) = (b_1,\dots,b_k)$, we denote as $\mathfrak{A}_v$ the unique $\mathfrak{A} \in S$ that is isomorphic with $\mathfrak{B}|_{\{b_1,\dots,b_k\}}$ via $i \mapsto b_i$.

Definition 4.4.3. For a τ-structure $\mathfrak{B}$ we let $\mathfrak{T}(\mathfrak{B})$ be the unravelling of the transition system $\mathfrak{K}(\mathfrak{B})$ from the empty list.

$$\begin{aligned}
\mathfrak{T}(\mathfrak{B}) &= \big(V, (P_{\mathfrak{A}})_{\mathfrak{A}\in S}, (E_\rho)_{\rho\in F}\big)\\
P_{\mathfrak{A}} &= \big\{v \in V : \mathfrak{A}_v = \mathfrak{A}\big\}\\
E_\rho &= \big\{(u,v) \in V^2 \colon v = u\rho g\big\}
\end{aligned}$$

Example. Let $\mathfrak{B}$ be as in Figure 3.1. We intuitively describe $\mathfrak{K}(\mathfrak{B})$ and $\mathfrak{T}(\mathfrak{B})$. For illustrative purposes we use the elements of $\mathfrak{B}$ in the universes of the $\mathfrak{A}_v$, instead of initial segments of the natural numbers as required in the definition.

$\mathfrak{K}(\mathfrak{B})$ is an (edge and node labelled) graph that contains one node for each guarded list in $\mathfrak{B}$. This includes (a, b, d), (b, d, a) and (a, c'), but not (a, c). Any two (not necessarily distinct) nodes are connected by an $E_\emptyset$-edge, i.e. an E_ρ-edge for the empty ρ. If the intersection of the universes of the two nodes is non-empty, there are further connecting edges for corresponding non-empty ρ. E.g. from (a, c) to (c, a) there is an E_ρ-edge for all ρ that are a subset of $\{2 \mapsto 1, 1 \mapsto 2\}$, and from (a, b, d) to (d, c, b) when ρ is a subset of $\{1 \mapsto 3, 3 \mapsto 2\}$ (the first element at the destination node (d, c, b) is the same as the third element at the source node (a, b, d) and the third element at the destination node is the same as the second one at the source node).

$\mathfrak{T}(\mathfrak{B})$ is the unravelling of $\mathfrak{K}(\mathfrak{B})$. The structure $\mathfrak{A}_\lambda$ represented at the root is the empty structure. Inductively, the set of successors of each node contains (possibly several) copies of each non-empty guarded list of $\mathfrak{B}$, i.e. (possibly several) copies of each node of $\mathfrak{K}(\mathfrak{B})$. Several successors obtained from the same guarded list are used to disentangle multiple edges as occur in $\mathfrak{K}(\mathfrak{B})$. E.g. a node v in $\mathfrak{T}(\mathfrak{B})$ with $A_v = (a, b, d)$ has one successor node with universe (d, c, b) for each ρ that is a subset of $\{1 \mapsto 3, 3 \mapsto 2\}$ (the same set as above) and each of these successors is connected via exactly one of the E_ρ.

A major difference between how $\mathfrak{K}(\mathfrak{B})$ and $\mathfrak{T}(\mathfrak{B})$ represent $\mathfrak{B}$ is that in $\mathfrak{K}(\mathfrak{B})$ each element of $\mathfrak{B}$ only occurs once. That is, for any two guarded lists that share an element, say b, there is an E_ρ-edge between them such that ρ explicitly connects the two b's. $\mathfrak{T}(\mathfrak{B})$ behaves very differently. Consider a node v where the universe of the structure represented at v is (a, c') and its successors v' with (a', c) and v'' with (a, c'). Since the universes of v and v', resp. v' and v'' do not share common elements (of $\mathfrak{B}$), there are only $E_\emptyset$-edges from v to v' and v' to v''. After the normalisation step, that makes every $\mathfrak{A}_v$ have universe $\{1, \ldots, |\mathfrak{A}_v|\}$, the information that the a at v and the a at v'' were actually the same element, is lost. This means that each element of $\mathfrak{B}$ will have multiple copies in the τ-structure that is in turn obtained from $\mathfrak{T}(\mathfrak{B})$, as explained in the following section.

The following is then a direct consequence of Lemma 4.4.2, and the fact that unravellings are bisimilar companions.

Corollary 4.4.4. *For τ-structures $\mathfrak{B}$ and $\mathfrak{B}'$: $\mathfrak{B} \sim_g \mathfrak{B}'$ iff $\mathfrak{T}(\mathfrak{B}) \sim \mathfrak{T}(\mathfrak{B}')$.*

4.5 From Trees to Structures

Conversely, we would like to associate with a tree $\mathfrak{T}$ of type $\tilde{\tau}$ a τ-structure $\mathfrak{D}$ for which $\mathfrak{T}(\mathfrak{D}) \sim \mathfrak{T}$. This is clearly not possible for arbitrary $\mathfrak{T}$. We collect some suitable conditions, which are obviously satisfied by every tree $\mathfrak{T}(\mathfrak{B})$, in the following definition.

Definition 4.5.1. Let $\mathfrak{T}$ be any $\tilde{\tau}$-tree. We call $\mathfrak{T}$ *consistent* if the following are satisfied.

(a) For each node v there is a unique $\mathfrak{A} \in S$ such that $\mathfrak{T} \models P_{\mathfrak{A}}(v)$, denoted $\mathfrak{A}_v$; $\mathfrak{A}_\lambda = \emptyset$, and $\mathfrak{A}_u \neq \emptyset$ for all $u \neq \lambda$.

(b) If $(u, v) \in E_\rho$ then ρ is a partial isomorphism between $\mathfrak{A}_v$ and $\mathfrak{A}_u$.

(c) If $(u, v) \in E_\rho$, then for every isomorphic embedding $\pi : \mathfrak{A} \to \mathfrak{A}_v$ from some structure $\mathfrak{A} \in S$ to $\mathfrak{A}_v$ there exists a node w such that $\mathfrak{A}_w = \mathfrak{A}$ and $(u, w) \in E_{\rho\circ\pi}$.

The local richness conditions expressed in (c) is technically useful as it ensures that all enumerations of every guarded (sub)set are locally available as guarded lists. Note that the $\mathfrak{T}(\mathfrak{B})$ satisfy much stronger homogeneity properties. For any two nodes v, w in $\mathfrak{T}(\mathcal{B})$ there is a canonical isomorphism between the two forests consisting of the trees rooted at the successor nodes of v and w, respectively. This property cannot be captured in MSO. MSO-definability and closure under bisimulation of the underlying class will be technically essential in the sequel. The conditions (a)—(c) are of course first-order expressible.

Lemma 4.5.2. *Within the class of $\tilde{\tau}$-trees, the class of consistent trees is bisimulation invariant and first-order definable.*

This is obvious from the definition. The important fact, however, is that these simple first-order expressible consistency conditions are sufficiently restrictive to allow us to (re)construct from every consistent $\tilde{\tau}$-tree an associated τ-structure. And in the special case that the $\tilde{\tau}$-tree comes as $\mathfrak{T}(\mathfrak{B})$ for some τ-structure $\mathfrak{B}$, whose guarded behaviour is described in $\mathfrak{T}(\mathfrak{B})$, we actually recover this behaviour: the resulting τ-structure will be guarded bisimulation equivalent to the original one.

The idea is to start from the $\mathfrak{A}_v$ that are represented in the nodes v of $\mathfrak{T}$, and to patch these local isomorphism types together in such a way that the moves in the guarded bisimulation game recorded in $\mathfrak{T}$ are realised in the resulting structure. To this end we need to identify elements in neighbouring $\mathfrak{A}_u$ and

$\mathfrak{A}_v$, attached to nodes u and v, that are linked by E_ρ, in accordance with the partial isomorphism ρ. In other words, we let $\mathfrak{A}_u$ and $\mathfrak{A}_v$ overlap according to identifications prescribed in E_ρ. Technically we take the disjoint union of the $\mathfrak{A}_v$ and form the quotient w.r.t. equalities imposed by overlaps according to the E_ρ. We describe this process more formally in the following.

Let $\mathfrak{T} = (V, (P_\mathfrak{A})_{\mathfrak{A}\in S}, (E_\rho)_{\rho\in F})$ be a consistent $\tilde{\tau}$-tree. We find the associated τ-structure $\mathfrak{D}(\mathfrak{T})$ as follows. Let $U = \{(v,i) : v \in V, i \in \mathfrak{A}_v\}$ be the disjoint union of the $\mathfrak{A}_v$.

Let $\approx$ be the reflexive, symmetric transitive closure of the following relation $\approx_0$ on U:

$$(v,i) \approx_0 (w,j) \quad \Leftrightarrow \quad (v,w) \in E_\rho \text{ and } \rho(j) = i \quad \text{for some } \rho \in F.$$

The universe of $\mathfrak{D}(\mathfrak{T})$ is $D = U/_\approx$, the set of $\approx$-equivalence classes $[v,i]$ in U. We say that an equivalence class $d = [v,i]$ *lives at node* u, or that d is *represented* in that node u, if $[v,i] = [u,j]$ for some j.

Note that $\approx_0$ describes the local identification pattern imposed by single E_ρ-edges. Intuitively, it is necessary to take its transitive closure because paths along consecutive different E_ρ-edges in $\mathfrak{T}$ can lead to identifications involving one and the same element, if this element happens to be passed through the successive overlaps along that path. It is important to note that we are thus dealing with non-local phenomena, insofar as nodes that are far apart in $\mathfrak{T}$ may represent the same element. It is clear from the definition of $\approx_0$, however, that the set of nodes of $\mathfrak{T}$ that represent some fixed equivalence class d form a connected subgraph, i.e., a subtree. If an equivalence class d lives at nodes u and v in $\mathfrak{T}$ then it must also live at every node along the (unique) shortest path connecting u to v in $\mathfrak{T}$.

It follows from consistency condition (b), that we may consistently interpret every k-ary predicate R in τ over D by putting $\boldsymbol{d} = ([v,i_1],\dots,[v,i_k]) \in R$ if $\mathfrak{A}_v \models R(i_1,\dots,i_k)$.

For further reference we sum up the construction in the following definition.

Definition 4.5.3. For any $\tilde{\tau}$-tree $\mathfrak{T} = (V, (P_\mathfrak{A})_{\mathfrak{A}\in S}, (E_\rho)_{\rho\in F})$ satisfying the consistency conditions (a) and (b) from Definition 4.5.1 let $\mathfrak{D}(\mathfrak{T}) = (D, (R)_{R\in\tau})$ be the τ-structure on universe $D = U/\approx$, where

$$U = \dot{\bigcup}_{v\in V} A_v = \big\{(v,i)\colon v \in V, i \in \mathfrak{A}_v\big\}$$

and $\approx$ is as defined above, with interpretations of $R \in \tau$ according to

$$R^{\mathfrak{D}(\mathfrak{T})} = \big\{\boldsymbol{d} = ([v,i_1],\dots,[v,i_k])\colon (i_1,\dots,i_k) \in R^{\mathfrak{A}_v}\big\}.$$

Note that $\boldsymbol{d} = (d_1, \ldots, d_k)$ is a guarded list in $\mathfrak{D}(\mathfrak{T})$ if, and only if, $\boldsymbol{d} = ([v, 1], \ldots, [v, k])$ for some node $v \in \mathfrak{T}$ of size $|v| = k$. In this case we say that v *represents the guarded list* $\boldsymbol{d}$. Note that the components of this list can be represented at other nodes too, including all nodes that represent permutations of the same guarded list, and possibly also at nodes representing guarded super-lists.

Lemma 4.5.4. *For all τ-structures $\mathfrak{B}$: $\mathfrak{D}(\mathfrak{T}(\mathfrak{B})) \sim_g \mathfrak{B}$.*

Proof. Let $\mathfrak{T} = \mathfrak{T}(\mathfrak{B})$, $\mathfrak{K} = \mathfrak{K}(\mathfrak{B})$ and $\mathfrak{D} = \mathfrak{D}(\mathfrak{T})$. Recall that $\mathfrak{T}$ is the unravelling of $\mathfrak{K}$, and that the natural projection $\pi\colon \mathfrak{T} \to \mathfrak{K}$ associates to any node v of $\mathfrak{T}$ the node $\pi(v)$ in $\mathfrak{K}$, which is the last node in the path that gives rise to v. Further recall that $\pi(v)$, being a node in $\mathfrak{K}$, is by definition a guarded list in $\mathfrak{B}$, $\pi(v) \in G(\mathfrak{B})$.

Let $\boldsymbol{d} = (d_1, \ldots, d_k)$ be a guarded list in $\mathfrak{D}$, v any node of $\mathfrak{T}$ representing this guarded list, $\boldsymbol{d} = ([v, 1], \ldots, [v, k])$. Let $\pi(v) = (b_1, \ldots, b_k) \in G(\mathfrak{B})$. We let $f_{\boldsymbol{d},v}$ be the partial 1-1 map from $\mathfrak{D}$ to $\mathfrak{B}$ that maps $\boldsymbol{d}$ to the guarded list $\pi(v)$ in $\mathfrak{B}$:

$$f_{\boldsymbol{d},v}\colon d_i \mapsto b_i, \quad i = 1, \ldots, k.$$

Note that as a map $f_{\boldsymbol{d},v}$ only depends on the guarded set $\{d_1, \ldots, d_k\}$ and not on the order of components or on the chosen representative v. If v and v' represent lists that enumerate the same guarded set in $\mathfrak{D}$, they are linked in $\mathfrak{T}$ by a path along which all components always live together. The corresponding path in $\mathfrak{K}$ similarly links $\pi(v)$ to $\pi(v')$, with matching permutations along that path.

We claim that the set of all these $f_{\boldsymbol{d},v}$ is a guarded bisimulation between $\mathfrak{D}$ and $\mathfrak{B}$. It is obvious from the construction that the $f_{\boldsymbol{d},v}$ are partial isomorphisms whose domains are guarded sets. Consider $f_{\boldsymbol{d},v} : \boldsymbol{d} \mapsto \pi(v)$. Its domain is enumerated by the guarded list $\boldsymbol{d} = (d_1, \ldots, d_l)$, represented at some node v in $\mathfrak{T}$; its image is enumerated by the guarded list $\pi(v) = (b_1, \ldots, b_l)$ of $\mathfrak{B}$. It remains to check the back and forth conditions.

For the forth property consider any other guarded set in $\mathfrak{D}$, enumerated as a guarded list $\boldsymbol{d}'$ in $\mathfrak{D}$, which is represented by some node v' in $\mathfrak{T}$. Let $\boldsymbol{c}$ be the tuple of common components in $\boldsymbol{d}$ and $\boldsymbol{d}'$. The tuple $\boldsymbol{c}$ lives along the unique shortest path from v to v' in $\mathfrak{T}$. Projecting this path down to $\mathfrak{K}$ and $\mathfrak{B}$ we see that $f_{\boldsymbol{d}',v'}$ and $f_{\boldsymbol{d},v}$ agree on their common domain.

For the back property consider any other guarded set in $\mathfrak{B}$, enumerated as a guarded list $\boldsymbol{b}' = (b'_1, \ldots, b'_k)$. Consider both $\boldsymbol{b}'$ and $\boldsymbol{b} = \pi(v) = (b_1, \ldots, b_k)$

as nodes of $\mathfrak{K}$ and note that $\mathfrak{T}, v \sim \mathfrak{K}, \boldsymbol{b}$. Let $\rho = \{(i,j) \colon b'_i = b_j\}$. It follows that $(\boldsymbol{b}, \boldsymbol{b}') \in E_\rho$. As $\mathfrak{T}, v \sim \mathfrak{K}, \boldsymbol{b}$, there is a node v' in $\mathfrak{T}$ such that $(v, v') \in E_\rho$ and $\mathfrak{T}, v' \sim \mathfrak{K}, \boldsymbol{b}'$. This node v' represents a guarded list $\boldsymbol{d}'$ of $\mathfrak{D}$ such that $f_{\boldsymbol{d}',v'} \colon \boldsymbol{d}' \mapsto \boldsymbol{b}'$. As E_ρ induces an identification of elements between $\boldsymbol{b}'$ and $\boldsymbol{b}$ according to ρ, it is clear that the inverses of $f_{\boldsymbol{d}',v'}$ and $f_{\boldsymbol{d},v}$ agree on their common domain. □

Remark. $\mathfrak{B}^* = \mathfrak{D}(\mathfrak{T}(\mathfrak{B}))$ may be regarded as a *guarded unravelling* of $\mathfrak{B}$, analogous to the standard unravelling of transition systems. Indeed, the resulting guarded bisimilar structure $\mathfrak{B}^*$ is also tree-like in that it has a tree decomposition of width $m - 1$, where m is the width of τ. The naked tree $\mathfrak{T}(\mathfrak{B})$, stripped of its labels and regarded as a directed graph, together with the natural association of a node v of $\mathfrak{T}(\mathfrak{B})$ with the guarded set it represents in $\mathfrak{B}^*$, induces a tree decomposition of $\mathfrak{B}^*$ in the usual sense. Tree unravellings, and the corresponding generalised tree model property for GF and some of its relatives have been put to use e.g. in [1] and [26].

The following proposition extends the intuition that the bisimulation type of $\mathfrak{T}(\mathfrak{B})$ captures the guarded bisimulation type of $\mathfrak{B}$ to the setting of all consistent trees. The proof is via a canonical lift of tree bisimulations.

Proposition 4.5.5. *For any consistent $\tilde{\tau}$-trees $\mathfrak{T}$ and $\mathfrak{T}'$: If $\mathfrak{T} \sim \mathfrak{T}'$, then $\mathfrak{D}(\mathfrak{T}) \sim_g \mathfrak{D}(\mathfrak{T}')$.*

Proof. Let $Z \subseteq V \times V'$ be a two-way bisimulation between $\mathfrak{T}$ and $\mathfrak{T}'$ according to Lemma 2.3.4. Recall in this context that, according to our convention in Definition 2.3.1, trees have unique edge labels.

For each pair $(v, v') \in Z$ let $f_{vv'}$ be the function that maps the guarded list represented by v to that represented by v'. Clearly, $f_{vv'}$ is a partial isomorphism. This is immediate from $\mathfrak{A}_v = \mathfrak{A}_{v'}$, which in turn is just a consequence of bisimilarity between v and v' at the level of basic propositions: $v' \in P_{\mathfrak{A}}$ iff $v \in P_{\mathfrak{A}}$ for all $\mathfrak{A} \in S$. Note also that for $(u, v) \in E_\rho$ and $(u', v') \in E'_\rho$ the maps $f_{uu'}$ and $f_{vv'}$ agree on their common domain: this follows from the construction of $\mathfrak{D}(\mathfrak{T})$ and $\mathfrak{D}(\mathfrak{T}')$, because elements represented at u and v are identified in $\mathfrak{D}(\mathfrak{T})$ via ρ in exactly the same way as the corresponding elements at u' and v' are identified in $\mathfrak{D}(\mathfrak{T}')$.

We claim that the set of all $f_{vv'}$, for $(v, v') \in Z$, is a guarded bisimulation between $\mathfrak{D}(\mathfrak{T})$ and $\mathfrak{D}(\mathfrak{T}')$.

As the situation is entirely symmetric, it suffices to argue, for instance, for the forth condition. Let $\boldsymbol{c}$ and $\boldsymbol{d}$ be guarded lists in $\mathfrak{D}(\mathfrak{T})$, represented by u and v in $\mathfrak{T}$, respectively. Let $(u, u') \in Z$, $f_{uu'} : \boldsymbol{c} \mapsto \boldsymbol{c}'$. We need to find $v' \in \mathfrak{T}'$ such that $(v, v') \in Z$ and such that $f_{vv'}$ agrees with $f_{uu'}$ on their common domain. Let X be the set of common elements in $\boldsymbol{c}$ and $\boldsymbol{d}$. Consider the unique shortest path from u to v in $\mathfrak{T}$. As Z is a two-way bisimulation, this path gives rise to a bisimilar path from u' to some v' in $\mathfrak{T}'$. Now $f_{vv'}$ is as desired: the elements of X live at all nodes on the path from u to v, whence all the intermediate mappings $f_{ww'}$ along the path, and in particular $f_{vv'}$ respect $f_{uu'}|_X$. $\square$

Remark. It is straightforward to extend all the notions and observations of this section to the case of structures $\mathfrak{B}$ with a distinguished parameter tuple $\boldsymbol{b}$ that is a guarded list in $\mathfrak{B}$. The tree $\mathfrak{T}(\mathfrak{B}, \boldsymbol{b})$, for instance, would be rooted in the guarded list $\boldsymbol{b}$ rather than in the empty list, and could alternatively be described as the unravelling of $\mathfrak{K}(\mathfrak{B})$ from $\boldsymbol{b}$. In the opposite direction, i.e. for the passage from a consistent $\tilde{\tau}$-tree to a τ-structure, one can clearly extract a guarded list from the root as the distinguished parameter tuple for $\mathfrak{D}(\mathfrak{T})$. All results, and in particular Lemma 4.5.4 and Proposition 4.5.5, go through trivially with a distinguished guarded list of parameters.

Chapter 5

Tableaux and Finite Models

In this chapter we show how the methods employed for modal logics can be generalised to a tableau algorithm for the guarded fragment. As major spin-off we obtain an alternative, elementary proof of the finite model property for the guarded fragment. However, retrieving a finite model from a tableau is significantly more involved than in the case of modal logic.

The version of the tableau algorithm presented here is specifically tuned towards generating finite models. An alternative approach, shown to work for the much more expressive clique guarded fragment, is presented in [35]. There, the output of the tableau algorithm is used as template for an infinite model of bounded tree width, giving an alternative proof of the generalised tree-model property for both the guarded and clique guarded fragments. The question of whether there is a tableau-based proof of the finite model property for the clique guarded fragment is still open.

For starters, a tableau algorithm for modal logic will be presented. The reader familiar with tableau algorithms can skip to Section 5.2.

5.1 Tableau Algorithm for ML

Definition 5.1.1. Let $\psi \in \mathrm{ML}$ be a formula in NNF.
A *completion tree* $\mathcal{T} = (V, (E_a)_{a \in A}, \Delta)$ for ψ consists of

- a rooted tree $(V, (E_a)_{a \in A})$
- a function $\Delta : V \to \mathrm{cl}(\psi)$

A completion tree $\mathcal{T}$ is called *complete*, if none of the rules from Figure 5.1

can be applied. If a Δ-label of $\mathcal{T}$ contains contradictory atomic statements P_b and $\neg P_b$, we say that $\mathcal{T}$ contains a *clash*. Else $\mathcal{T}$ is *clash-free*. A completion tree that is both clash-free and complete is a *tableau*.

Given a formula $\psi \in$ ML, the tableau algorithm starts by creating a root node λ and initialising $\Delta(v)$ to $\{\psi\}$. Then the rules from Figure 5.1 are applied until either a clash occurs, producing output "ψ `unsatisfiable`", or a tableau is reached, in which case "ψ `satisfiable`" is output.

$$
\begin{array}{lll}
\mathrm{R}_\wedge: & \text{if} & \varphi \wedge \vartheta \in \Delta(v) \text{ and } \{\varphi, \vartheta\} \not\subseteq \Delta(v) \\
& \text{then} & \Delta(v) = \Delta(v) \cup \{\varphi, \vartheta\} \\
\\
\mathrm{R}_\vee: & \text{if} & \varphi \vee \vartheta \in \Delta(v) \text{ and } \{\varphi, \vartheta\} \cap \Delta(v) = \emptyset \\
& \text{then} & \Delta(v) = \Delta(v) \cup \{\chi\} \text{ for } \chi \in \{\varphi, \vartheta\} \\
\\
\mathrm{R}_\Box: & \text{if} & [a]\varphi \in \Delta(v) \text{ and } \varphi \notin \Delta(w) \\
& & \text{for some } a\text{-successor } w \text{ of } v \\
& \text{then} & \Delta(w) = \Delta(w) \cup \{\varphi\} \\
\\
\mathrm{R}_\Diamond: & \text{if} & \langle a \rangle\varphi \in \Delta(v) \text{ and there is no } a\text{-successor } w \text{ of } v \\
& & \text{with } \varphi \in \Delta(w) \\
& \text{then} & \text{create an } a\text{-successor } w \text{ of } v \text{ and let } \Delta(w) = \{\varphi\}
\end{array}
$$

Figure 5.1: The Completion Rules for ML

Proposition 5.1.2. *The tableau algorithm is a (non-deterministic) decision procedure for* ML*-satisfiability.*

The proof typically consists of three parts.

1. Every sequence of rule applications terminates after a finite number of steps (*Termination*).
2. If ψ is satisfiable, then the rules can be applied to generate a tableau for ψ (*Completeness*).
3. If the algorithm constructs a tableau for ψ, then ψ is satisfiable (*Soundness* or *Correctness*).

We give a sketch of the proof for the ML algorithm. For termination, observe that each application of the $\mathrm{R}_\Diamond$- or $\mathrm{R}_\Box$-rule decreases the modal depth of the formula that is propagated. This gives a bound on the depth of the completion tree. The number of successor nodes generated for any node by the $\mathrm{R}_\Diamond$-rule is bounded by the number of $\Diamond$-subformulae of ψ. And a finite number of rule applications suffices to decompose the formulae in the Δ-labels. These observations can be combined to one global measure that shows that the interference between the decomposing rules and the propagation of formulae does not create the possibility of an infinite chain of rule applications.

Towards completeness, if ψ is satisfiable and $\mathfrak{K}, k \models \psi$, one uses a function $g : V \to \mathfrak{K}$ to track which node of $\mathfrak{K}$ gave rise to a node in $\mathcal{T}$, with $g(\lambda) = k$. This can be done in a way that $\varphi \in \Delta(v)$ implies $\mathfrak{K}, g(v) \models \varphi$ for all $v \in V$ is an invariant across rule application. Then use $\mathfrak{K}$ to steer the algorithm to produce a tableau. E.g. when applying the $\mathrm{R}_\vee$-rule at v to $\varphi \vee \vartheta \in \Delta(v)$, choose one of φ and ϑ according to which of them holds at $g(v)$. Or when creating a new node w as successor to v because of a formula $\Diamond\chi \in \Delta(v)$, let $g(w)$ be a successor of $g(v)$ that satisfies χ. Since the occurrence of both P_b and $\neg P_b$ would contradict the invariant for g, the resulting tableau is necessarily clash-free.

Towards correctness, each tableau for ψ can be made into a model for ψ by defining the unary predicates $(P_b)_{b\in B}$. For each node $v \in V$, let $v \in P_b$ iff $P_b \in \Delta(v)$. This yields a structure $\mathfrak{T} = (V, (E_a)_{a\in A}, (P_b)_{b\in B})$. Then, by straightforward induction, one obtains $\mathfrak{T}, v \models \varphi$ for all $\varphi \in \Delta(v)$.

We next take a look at the ramifications for the tableau algorithm when we add universal quantification $\blacksquare$ to ML. Intuitively there is not much margin for choice other than to propagate the φ in $\blacksquare\varphi$ to all nodes. A new completion rule $\mathrm{R}_\blacksquare$ is given in Figure 5.2, for the definition of *blocked* see below. If we keep the other rules unchanged, we can not guarantee termination of the algorithm.

Example. Consider the formula $\psi = \blacksquare\Diamond P$, and for the moment forget that it holds in the empty transition system. In the beginning, the tableau algorithm creates the root λ, and after applying the $\mathrm{R}_\blacksquare$-rule we get $\Delta(\lambda) = \{\blacksquare\Diamond P, \Diamond P\}$. Application of the $\mathrm{R}_\Diamond$-rule creates a new node v with $\Delta(v) = \{P\}$. Next the $\mathrm{R}_\blacksquare$-rule adds $\Diamond P$ to $\Delta(v)$. The last two steps are repeated ad infinitum, creating an infinite chain of nodes at which P and $\Diamond P$ hold.

The solution lies in the finite bound on the number of different Δ-labels. Every chain of nodes that grows beyond a certain point will contain pairs of nodes that locally look alike. The algorithm needs to be modified to recognise these situations where a node is *blocked* by an ancestor of the same local type.

$\mathrm{R}_\diamond$:	if	$\langle a\rangle\varphi \in \Delta(v)$ and there is no a-successor w of v with $\varphi \in \Delta(w)$ and v is not blocked
	then	create an a-successor w of v and let $\Delta(w) = \{\varphi\}$
$\mathrm{R}_\blacksquare$:	if	$\blacksquare\varphi \in \Delta(v)$ and $\varphi \notin \Delta(w)$ for some $w \in V$
	then	$\Delta(w) = \Delta(w) \cup \{\varphi\}$

Figure 5.2: Additional/Changed Rules for ML + $\blacksquare$

With the correct definition of blocking, and an appropriately adapted tableau algorithm, one can again show the following proposition. We omit the proof and continue with the tableau for the guarded fragment, where the methods are exhibited in greater detail.

Proposition 5.1.3. *The tableau algorithm is a (non-deterministic) decision procedure for* ML+$\blacksquare$*-satisfiability.*

5.2 Tableau Algorithm for GF

Let φ be a formula and K a set of constants. In the following an alternative notion of the closure of φ is needed, where constants from K are substituted for the free variables of φ.

Definition 5.2.1. The *closure of φ relative to K*, $\mathrm{cl}(\varphi, K)$, is defined as $\mathrm{cl}(\varphi, K) = \{\chi(\boldsymbol{c}) \ : \ \chi \in \mathrm{cl}(\varphi), \boldsymbol{c} \subseteq K\}$.

Definition 5.2.2. Let K be a countable set of constants. Let $\psi \in \mathrm{GF}$ be a closed formula in NNF.
A *completion tree* $\mathcal{T} = (V, E, C, \Delta, N, B, I)$ for ψ consists of

- a rooted tree (V, E),
- a function $C : V \to \mathcal{P}(K)$ assigning each node a set of constants of size at most $\mathrm{width}(\psi)$,
- a function $\Delta : V \to \mathit{cl}(\psi, C(v))$,
- a function $N : V \to \mathbb{N}$ mapping each node to a *distinct* natural number, with the additional property that, if v is an ancestor of w, then $N(v) < N(w)$,
- an explicit blocking relation B that will be explained below,

- an indexing function $I : \bigcup\{C(v) \,:\, v \in V\} \to \{1, \ldots, \text{width}(\psi)\}$, with the additional property that $I|_{C(v)}$ is 1-1 for every $v \in V$.

A constant $c \in K$ is called *shared* between two nodes $v_1, v_2 \in V$, if $c \in C(v_1) \cap C(v_2)$, and $c \in C(w)$ for all nodes w on the (unique, undirected, possibly empty) shortest path connecting v_1 to v_2.

A node $v \in V$ is called *directly blocked* by a node $w \in V$, if $w \neq v$ is a node such that $N(w) < N(v)$, and there is an injective mapping π from $C(v)$ into $C(w)$ such that, for all constants $c \in C(v)$ that are shared between v and w, $\pi(c) = c$, $\pi(\Delta(v)) = \Delta(w)|_{\pi(C(v)^*)}$, and $I(c) = I(\pi(c))$ for all $c \in C(v)$. The definition of blocking is recursive. This does not cause any problems because the status of a node v only depends on its own label, and the status of nodes w with $N(w) < N(v)$. At the latest, the recursion terminates at the root node, where the N-value is minimal.

For each directly blocked node v the relation B contains (w, v) for one (of the possibly many) nodes w that directly block v. In this case we say that v is *explicitly blocked* by w.

A node is called *blocked* if it is directly blocked or if its predecessor is blocked.

A completion tree $\mathcal{T}$ *contains a clash* if there is a node $v \in V$ such that

- for a constant $c \in C(v)$, $c \neq c \in \Delta(v)$, or
- there is an atomic formula α and a tuple of constants $\boldsymbol{a} \subseteq C(v)$ such that $\{\alpha(\boldsymbol{a}), \neg\alpha(\boldsymbol{a})\} \subseteq \Delta(v)$.

Otherwise, $\mathcal{T}$ is called *clash-free*. A completion tree $\mathcal{T}$ is called *complete* if none of the *completion rules* given in Figure 5.3 can be applied to $\mathcal{T}$. A complete and clash-free completion tree for ψ is called a *tableau* for ψ.

To test ψ for satisfiability, the tableau algorithm creates an initial tree with only a single node v_0, $\Delta(v_0) = \{\psi\}$ and $C(v_0) = \emptyset$. Then, alternately, the rules from Figure 5.3 are applied and B updated so that it adheres to the definition, until either a clash occurs, producing output "ψ `unsatisfiable`", or the tree is complete, in which case "ψ `satisfiable`" is output.

Theorem 5.2.3. *The tableau algorithm is a (non-deterministic) decision procedure for* GF*-satisfiability.*

Proof. This is an immediate consequence of the following facts established in the subsequent sections. Termination is shown in Lemma 5.2.5,

$\mathrm{R}_\wedge$:	if	$\varphi \wedge \vartheta \in \Delta(v)$ and $\{\varphi, \vartheta\} \not\subseteq \Delta(v)$
	then	$\Delta(v) = \Delta(v) \cup \{\varphi, \vartheta\}$
$\mathrm{R}_\vee$:	if	$\varphi \vee \vartheta \in \Delta(v)$ and $\{\varphi, \vartheta\} \cap \Delta(v) = \emptyset$
	then	$\Delta(v) = \Delta(v) \cup \{\chi\}$ for $\chi \in \{\varphi, \vartheta\}$
$\mathrm{R}_=$:	if	$a = b \in \Delta(v)$ and $a \neq b$
	then	$C(w) = (C(w) \setminus \{a\}) \cup \{b\}$ and $\Delta(w) = \Delta(w)[a \mapsto b]$ for all w that share a with v
$\mathrm{R}_\forall$:	if	$(\forall \boldsymbol{y}.\alpha(\boldsymbol{a}, \boldsymbol{y}))\varphi(\boldsymbol{a}, \boldsymbol{y}) \in \Delta(v)$ and $\alpha(\boldsymbol{a}, \boldsymbol{b}) \in \Delta(v), \varphi(\boldsymbol{a}, \boldsymbol{b}) \notin \Delta(v)$ for some $\boldsymbol{b} \subseteq C(v)$
	then	$\Delta(v) = \Delta(v) \cup \{\varphi(\boldsymbol{a}, \boldsymbol{b})\}$
$\mathrm{R}_\exists$:	if	$(\exists \boldsymbol{y}.\alpha(\boldsymbol{a}, \boldsymbol{y}))\varphi(\boldsymbol{a}, \boldsymbol{y}) \in \Delta(v)$ and for every $\boldsymbol{b} \subseteq C(v)$, $\{\alpha(\boldsymbol{a}, \boldsymbol{b}), \varphi(\boldsymbol{a}, \boldsymbol{b})\} \not\subseteq \Delta(v)$ and there is no child w of v with $\{\alpha(\boldsymbol{a}, \boldsymbol{b}), \varphi(\boldsymbol{a}, \boldsymbol{b})\} \subseteq \Delta(w)$ for some $\boldsymbol{b} \subseteq C(w)$ and v is not blocked
	then	let $\boldsymbol{b}$ be a sequence of distinct and fresh constants that match the length of $\boldsymbol{y}$, create a child w of v with $C(w) = \boldsymbol{a} \cup \boldsymbol{b}$ and $\Delta(w) = \{\alpha(\boldsymbol{a}, \boldsymbol{b}), \varphi(\boldsymbol{a}, \boldsymbol{b})\}$, let $N(w) = 1 + \max\{N(v) \ : \ v \in V \setminus \{w\}\}$ and arbitrarily extend I according to Definition 5.2.2.
$\mathrm{R}_\updownarrow$:	if	$\alpha(\boldsymbol{a}) \in \Delta(v), \alpha$ atomic, not an equality, w is a neighbour of v with $\boldsymbol{a} \subseteq C(w)$, and $\alpha(\boldsymbol{a}) \notin \Delta(w)$
	then	$\Delta(w) = \Delta(w) \cup \{\alpha(\boldsymbol{a})\}$
$\mathrm{R}_{\updownarrow\forall}$:	if	$\varphi(\boldsymbol{a}) \in \Delta(v), \varphi(\boldsymbol{a})$ is univ. quantified, and w is a neighbour of v with $\boldsymbol{a} \subseteq C(w)$ and $\varphi(\boldsymbol{a}) \notin \Delta(w)$
	then	$\Delta(w) = \Delta(w) \cup \{\varphi(\boldsymbol{a})\}$

Figure 5.3: The Completion Rules for GF

completeness in Lemma 5.2.6 and correctness via explicit construction of a finite model is shown in Lemma 5.2.11. □

5.2.1 Termination

The following technical lemma is a consequence of the completion rules and the blocking condition.

Lemma 5.2.4. *Let $\psi \in$ GF be a sentence in* NNF*with* $|\psi| = n$, $\mathrm{width}(\psi) = m$, *and $\mathcal{T}$ a completion tree generated for ψ by application of the rules in Figure 5.3. For every node $v \in \mathcal{T}$,*

1. $|C(v)| \leq m$
2. $|\Delta(v)| \leq n \times m^m$
3. *Every path of length $\ell > m! \times 2^{n \times m^m}$ in $\mathcal{T}$ contains a blocked node.*

Proof. 1. Nodes are only generated when initialising the tree, and by the $\mathrm{R}_\exists$-rule. The $\mathrm{R}_=$-rule may later remove some of the constants living at a node, but no constants are ever added to a $C(v)$ once v has been generated.

When triggered by the formula $(\exists \boldsymbol{y}.\alpha(\boldsymbol{a}, \boldsymbol{y}))\varphi(\boldsymbol{a}, \boldsymbol{y})$, the $\mathrm{R}_\exists$-rule initialises $C(w)$ such that it contains $\boldsymbol{a}$ and another constant for every variable in $\boldsymbol{x}$ and $\boldsymbol{y}$. Hence,

$$|C(w)| \leq |\boldsymbol{a} \cup \boldsymbol{y} \cup \boldsymbol{z}| \leq |\,\mathrm{free}(\alpha)| \leq \mathrm{width}(\psi).$$

2. The set $\Delta(v)$ is a subset of $cl(\psi, C(v))$, for which $|cl(\psi, C(v))| \leq n \times m^m$ holds because there are at most n formulae in $cl(\psi)$, each of which has at most m free variables. There are at most $|C(v)|^m$ distinct sequences of length m with constants from $C(v)$.

3. Let $u_1, \dots, u_\ell$ be $\ell > m! \times 2^{n \times m^m}$ distinct nodes that form a path; w.l.o.g. u_1 is the root of $\mathcal{T}$. For every u_i, we construct an injective mapping $\pi_i : C(v_i) \to \{1, \dots m\}$ such that, if a constant a is shared between two nodes u_i, u_j, then $\pi_i(a) = \pi_j(a)$.

By induction on i we define an injective mapping $\nu_i : C(u_i) \to \{1, \dots, m\}$ for every $i \in \{1, \dots, \ell\}$ as follows. For ν_1 we pick an arbitrary injective function from $C(u_1)$ to $\{1, \dots, m\}$. For ν_i we choose an arbitrary injective function such that $\nu_i(a) = \nu_{i-1}(a)$ for all $a \in C(u_i) \cap C(u_{i-1})$.

All mappings ν_i are injective. For any constant a the set $V_a = \{v \in V \ : \ a \in C(v)\}$ induces a subtree of $\mathcal{T}$. If $u_i, u_j \in V_a$ are neighbours, the definition above ensures $\nu_i(a) = \nu_j(a)$. By induction over the length of the shortest connecting path we obtain the same for arbitrary $u_i, u_j \in V_a$.

For every node v_i there is a j_i such that $v_i = u_{j_i}$ and we set $\pi_i = \nu_{j_i}$.

There are at most $2^{n \times m^m}$ distinct subsets of $cl(\psi, \{1, \dots, m\})$, and at most $m!$ different indexings of the occurring constants. Hence, there must be two nodes u_i, u_j such that $\pi_i(\Delta(u_i)) = \pi_j(\Delta(u_j))$, $N(v_i) < N(v_j)$, and π_i, π_j

are compatible with I. This implies that either u_j is blocked by u_i via $\pi = \pi_i^{-1} \circ \pi_j$, or both are blocked by a common third node w. Note that for π to be well-defined, π_i must be injective. By construction, π preserves shared constants. Since $\pi_i(\Delta(u_i)) = \pi_j(\Delta(u_j))$, $\pi(\Delta(u_j)) = \Delta(u_i)|_{\pi(C(u_j))}$ holds. □

Lemma 5.2.5. *Let* $\psi \in \mathrm{GF}$ *be a sentence in* NNF*. Any sequence of rule applications of the tableau algorithm starting from the initial tree terminates.*

Proof. For any completion tree $\mathcal{T}$ generated by the tableau algorithm, we define $\|\cdot\| : V \mapsto \mathbb{N}^3$ by

$$\|v\| = (|C(v)|, \quad n \times (m+1)^m - |\Delta(v)|, \\ |\{\varphi \in \Delta(v) : \varphi \text{ triggers the } \mathrm{R}_\exists\text{-rule for } v\}|).$$

The lexicographical order $\prec$ on $\mathbb{N}^3$ is well-founded, i.e. it has no infinite decreasing chains. Any rule application decreases $\|v\|$ w.r.t. $\prec$ for at least one node v, and never increases $\|v\|$ w.r.t. $\prec$ for an existing node v. However it may create new successors, one at a time. Since $\prec$ is well-founded, there can only be a finite number of applications of rules to every node in $\mathcal{T}$, and hence a finite number of successors to each node. An infinite sequence of rule applications would necessarily generate a tree of infinite depth.

Yet, as a corollary of Lemma 5.2.4, we have that the depth of $\mathcal{T}$ is bounded by $2^{n \times m^m}$. □

5.2.2 Completeness

Lemma 5.2.6. *Let* $\psi \in \mathrm{GF}$ *be a sentence in* NNF*. If* ψ *is satisfiable, then there is a sequence of rule applications starting from the initial tree that yields a tableau.*

Proof. Since ψ is satisfiable, there is a model $\mathfrak{A}$ of ψ. We will use $\mathfrak{A}$ to guide the application of the non-deterministic $\mathrm{R}_\vee$-rule. For this we incrementally define a function $g : \bigcup\{C(v) \ : \ v \in V\} \to A$ such that $\mathfrak{A} \models g(\Delta(v))$ for all $v \in V$. We refer to this property by (§).

Claim 5.2.7. *If, for a completion tree* $\mathcal{T}$*, there exists a function* g*, such that* (§) *holds and a rule is applicable to* $\mathcal{T}$*, then it can be applied in a way that maintains* (§).

- If the $\mathrm{R}_\wedge$-rule is applicable to a node $v \in V$ with $\varphi \wedge \vartheta \in \Delta(v)$ then, due to (§), $\mathfrak{A} \models g(\varphi \wedge \vartheta)$ and hence $\mathfrak{A} \models \{g(\varphi), g(\vartheta)\}$. So the $\mathrm{R}_\wedge$-rule can be applied to v without violating (§).

- If the $\mathrm{R}_\vee$-rule is applicable to a node $v \in V$ with $\varphi \vee \vartheta \in \Delta(v)$ then, due to (§), $\mathfrak{A} \models g(\varphi \vee \vartheta)$ and $\mathfrak{A} \models g(\chi)$ for a $\chi \in \{\varphi, \vartheta\}$. Hence, the $\mathrm{R}_\vee$-rule can be applied to v without violating (§).

- If the $\mathrm{R}_=$-rule is applicable to $v \in V$ with $a = b \in \Delta(v)$, then, since $\mathfrak{A} \models g(a) = g(b)$, $g(a) = g(b)$ must hold. Hence, for every node w that shares a with v, $g(\Delta(w)) = g(\Delta(w)[a \mapsto b])$ and the rule can be applied without violating (§).

- If the $\mathrm{R}_\forall$-rule is applicable to $v \in V$ with $(\forall \boldsymbol{y}.\alpha(\boldsymbol{a}, \boldsymbol{y}))\varphi(\boldsymbol{a}, \boldsymbol{y}) \in \Delta(v)$ and $\boldsymbol{b} \subseteq C(v)$ with $\alpha(\boldsymbol{a}, \boldsymbol{b}) \in \Delta(v)$, then from (§) we get that $\mathfrak{A} \models \alpha(g(\boldsymbol{a}), g(\boldsymbol{b}))$ and $\mathfrak{A} \models (\forall \boldsymbol{y}.\alpha(g(\boldsymbol{a}), \boldsymbol{y})) \rightarrow \varphi(g(\boldsymbol{a}), \boldsymbol{y})$, which implies $\mathfrak{A} \models \varphi(g(\boldsymbol{a}), g(\boldsymbol{b}))$. Hence $\varphi(\boldsymbol{a}, \boldsymbol{b})$ can be added to $\Delta(v)$ without violating (§).

- If the $\mathrm{R}_\exists$-rule is applicable to $v \in V$ with $(\exists \boldsymbol{y}.\alpha(\boldsymbol{a}, \boldsymbol{y}))\varphi(\boldsymbol{a}, \boldsymbol{y})$, then this implies $\mathfrak{A} \models g((\exists \boldsymbol{y}.\alpha(\boldsymbol{a}, \boldsymbol{y}))\varphi(\boldsymbol{a}, \boldsymbol{y}))$. Hence, there is a sequence $\boldsymbol{b}' \subseteq A$ of elements such that $\mathfrak{A} \models \{\alpha(g(\boldsymbol{a}), \boldsymbol{b}'), \varphi(g(\boldsymbol{a}), \boldsymbol{b}')\}$. Extend g such that $g(\boldsymbol{b}) = \boldsymbol{b}'$, so $\mathfrak{A} \models \{g(\alpha(\boldsymbol{a}, \boldsymbol{b}), g(\varphi(\boldsymbol{a}, \boldsymbol{b}))\}$. Note that this might involve setting $g(b_1) = g(b_2)$ for some $b_1, b_2 \in \boldsymbol{b}$. With this construction the resulting extended completion-tree $\mathcal{T}$ and extended function g again satisfy (§).

- If the $\mathrm{R}_\updownarrow$-rule is applicable to $v \in V$ with $\alpha(\boldsymbol{a}) \in \Delta(v)$ and a neighbour w with $\boldsymbol{a} \subseteq C(w)$, then it adds $\alpha(\boldsymbol{a})$ to $\Delta(w)$. From (§) we get that $\mathfrak{A} \models \alpha(g(\boldsymbol{a}))$. Hence, adding $\alpha(\boldsymbol{a})$ to $\Delta(w)$ does not violate (§).

- If the $\mathrm{R}_{\updownarrow\forall}$-rule is applicable to a node $v \in V$ with a universally quantified formula $\varphi(\boldsymbol{a}) \in \Delta(v)$ and a neighbour w which shares $\boldsymbol{a}$ with v, (§) yields $\mathfrak{A} \models \varphi(g(\boldsymbol{a}))$. Hence adding $\varphi(\boldsymbol{a})$ to $\Delta(w)$ does not violate (§).

Claim 5.2.8. *A completion-tree $\mathcal{T}$ for which a function g exists such that* (§) *holds is clash free.*

Assume that $\mathcal{T}$ contains a clash, namely, there is a node $v \in V$ such that either $a \neq a \in V(v)$—implying $\mathfrak{A} \models g(a) \neq g(a)$—, or that there is a sequence $\boldsymbol{a} \subseteq C(v)$, and an atomic formula α such that $\{\alpha(\boldsymbol{a}), \neg\alpha(\boldsymbol{a})\} \subseteq \Delta(v)$. From (§), $\mathfrak{A} \models \{\beta(g(\boldsymbol{a})), \neg\beta(g(\boldsymbol{a}))\}$ would follow, a contradiction.

These claims yield Lemma 5.2.6 as follows. Let $\mathcal{T}$ be a tableau for ψ. Since $\mathfrak{A} \models \psi$, (§) is satisfied for the initial tree together with the function g mapping a_0 to an arbitrary element of the universe of $\mathfrak{A}$. By Lemma 5.2.5, any sequence of applications is finite, and from Claim 5.2.7 we get that there is a sequence of rule-applications that maintains (§). By Claim 5.2.8, this sequence results in a tableau. □

Lemma 5.2.6 involves two different kinds of non-determinism, namely how to extend I to new constants as well as the choice of which rule to apply to which constraint (as several rules might be applicable simultaneously), and which disjunct to choose in an application of the $\mathrm{R}_\vee$-rule. While the latter choice is *don't-know* non-deterministic, i.e., for a satisfiable formula only certain choices will lead to the discovery of a tableau, the former choice is *don't-care* non-deterministic. This means that arbitrary choices of which rule to apply next and how to extend I will lead to the discovery of a tableau for a satisfiable formula. For an implementation of the tableau algorithm this has the following consequences. Exhaustive search is necessary to deal with all possible expansions of the $\mathrm{R}_\vee$-rule, but arbitrary strategies of choosing which rule to apply next, and where to apply it, will lead to a correct implementation—although the efficiency of the implementation will strongly depend on a sophisticated strategy.

5.2.3 Correctness

In order to prove the correctness of the tableau algorithm we have to show that the existence of a tableau for ψ implies satisfiability of ψ. To this purpose, we construct an, indeed finite, model from a tableau. An alternative correctness proof is given in [59], providing a further proof of the generalised tree model property.

In the following, let $\psi \in \mathrm{GF}$ and let $\mathcal{T} = (V, E, C, \Delta, N, B, I)$ be a tableau for ψ. W.l.o.g. we assume, for every node $v \in V$ and every $a \in C(v)$, that $a = a \in \Delta(v)$. Further, whenever a new node w is created we assume all constants in $C(w)$ that are not propagated from the parent node to be chosen fresh, i.e. not occuring in the C-label of any other node of the completion tree.

Let $C(V) = \bigcup\{C(v) \ : \ v \in V, \ v \text{ not blocked}\}$ and define the equivalence relation $\sim$ on $C(V)$ as the reflexive and transitive closure of the set of all pairs of constants (c, d), where $c \in C(u)$ and $d \in C(v)$ for some $(u, v) \in B$, and it is the case that the function π that verifies that u blocks v maps d to

c.

We also use $\sim$ as an operator that maps a constant a to its $\sim$-class $\tilde{a}$. For tuples of constants $\boldsymbol{a}$, this operation is performed component-wise. We say that $\tilde{\boldsymbol{a}} \subseteq C(v)$, if for each $a \in \boldsymbol{a}$ we have $\tilde{a} \cap C(v) \neq \emptyset$.

Definition 5.2.9. Let $v, w \in V$ and $a \in C(v)$, $b \in C(w)$. An *(a, b)-path* in $\mathcal{T}$ is a sequence $(s_1, c_1), \ldots, (s_k, c_k)$ in $V \times C(V)$ such that $c_1 = a$, $c_k = b$ and for all $1 \leq i < k$ we have $c_i \in C(s_i)$ and one of the following holds.

1. $(s_i, s_{i+1}) \in E$ and $c_i = c_{i+1}$
2. $(s_i, s_{i+1}) \in B$ and $\pi(c_{i+1}) = c_i$
3. 1. and 2. for reversed roles of i and $i + 1$.

Intuitively, an (a, b)-path verifies $a \sim b$. Since each single explicit blocking preserves the I-index, it follows that $I(a) = I(b)$ whenever there is an (a, b)-path in $\mathcal{T}$.

The general idea in the construction of a model from a tableau is to use $C(V)/\sim$ for the universe, and define the relations according to the atomic information in the nodes. In general, there may be problematic situations in a tableau that prevents this construction from being well defined, what we call *dormant clashes*.

Definition 5.2.10. Two distinct nodes $v, w \in V$, two tuples of constants $\boldsymbol{a}, \boldsymbol{b}$ and an atomic statement β form a *dormant clash* $(v, w, \boldsymbol{a}, \boldsymbol{b}, \beta)$ in $\mathcal{T}$, if $\boldsymbol{a} \in C(v)$, $\boldsymbol{b} \in C(w)$ and it is the case that $\boldsymbol{a} \neq \boldsymbol{b}$, but $\boldsymbol{a} \sim \boldsymbol{b}$ and $\beta(\boldsymbol{a}) \in \Delta(v)$ and $\beta(\boldsymbol{b}) \notin \Delta(w)$.

A dormant clash $(v, w, \boldsymbol{a}, \boldsymbol{b}, \beta)$ can occur whenever there are (a_i, b_i)-paths for all $i \leq |\boldsymbol{a}| = |\boldsymbol{b}|$, but there is not (at least) one path from v to w along which all of these constants are propagated simultaneously. Since β indeed makes use of *all* constants from $\boldsymbol{a}$, only such a path would ensure the complete atomic information about $\boldsymbol{a}$ at v to match that of $\boldsymbol{b}$ at w.

Given a tableau $\mathcal{T}$, we isolate a set of *critical edges* $S = S(\mathcal{T})$, a subset of $E \cup B$ defined as follows.

Let $D = (v, w, \boldsymbol{a}, \boldsymbol{b}, \beta)$ be a dormant clash. Let $V_D \subseteq V$ be the set of nodes that contains v, and all nodes u where there are (possibly several) $E \cup B$-paths p from v to each u such that this p is an (a_i, c_i)-path for all $a_i \in \boldsymbol{a}$ and some appropriate $c_i \in C(u)$. That is, V_D is a subset of the nodes that contain tuple of constants $\sim$-equivalent to $\boldsymbol{a}$. The existence of a single path from v to each $u \in V_D$ that propagates a whole set of constants $\sim$-equivalent to $\boldsymbol{a}$ implies that the complete atomic information about these constants is shared between v and all $u \in V_D$. More precisely, for every $u \in V_D$ there is a $\boldsymbol{c} \subseteq C(u)$, $\boldsymbol{c} \sim \boldsymbol{a}$ and for every atomic statement γ, $\gamma(\boldsymbol{a}) \in \Delta(v) \iff \gamma(\boldsymbol{c}) \in \Delta(u)$. This shows that $w \notin V_D$, since $\beta(\boldsymbol{a}) \in \Delta(v)$, but $\beta(\boldsymbol{b}) \notin \Delta(w)$.

Next, we choose an index $t = t_D \leq |\boldsymbol{a}| = |\boldsymbol{b}|$ and let S_D be the set of $E \cup B$-edges that (a) go from V_D to $V \setminus V_D$, or vice versa, and (b) propagate a constant that is $\sim$-equivalent to a_t. Let S be the union of the sets S_D for all dormant clashes D of $\mathcal{T}$.

By making (finitely) many isomorphic copies of all subtrees of the tableau below the root, it is possible to redirect the critical edges in a kind of hypercube construction that eliminates the dormant clashes.

Lemma 5.2.11. *If there is a finite tableau $\mathcal{T}$ for ψ, then there is also a finite tableau $\mathcal{T}'$ for ψ that does not contain dormant clashes.*

Proof. Observe that no B-edge can be incident to the root of $\mathcal{T}$, since all other nodes contain at least one constant in their C-label.

Let λ be the root of $\mathcal{T}$ and let $n = |S|$. We make an enlarged version $\mathcal{T}'$ of $\mathcal{T}$ where all subtrees with roots that are direct E-successors of λ are replaced by 2^n isomorphic copies. The formal definition of $\mathcal{T}'$ is

- $V' = \{\lambda\} \cup \{v^i \;:\; v \in V \setminus \{\lambda\}\}$,
- $E' = \{(\lambda, v^i) \;:\; (\lambda, v) \in E\} \cup \{(v^i, w^i) \;:\; (v, w) \in E\}$,
- $C'(v^i) = \{a^i \;:\; a \in C(v)\}$, $C'(\lambda) = \emptyset$,
- $\Delta'(v^i) = \{\varphi(\boldsymbol{a}^i) \;:\; \varphi(\boldsymbol{a}) \in \Delta(v)\}$, $\Delta'(\lambda) = \Delta(\lambda)$,
- $N'(v^i) = 2^n \cdot N(v) + i$, $N'(\lambda) = 0$,
- $B' = \{(v^i, w^i) \;:\; (v, w) \in B\}$,
- $I'(v^i) = I(v)$,

for all $0 \leq i < 2^n$.

This first step also creates 2^n copies of our original critical edges, namely all (r^i, s^i) for which $(r, s) \in S$. We now *modify* E' and B' as follows. Let $\{(r_k, s_k) \;:\; 0 \leq k < n\}$ be an enumerated version of S.

For all $0 \leq k < n$ and all $0 \leq \ell, j < 2^n$, if the binary representations of ℓ and j differ exactly at the k-th position, we modify $\mathcal{T}'$ as follows:

- If (r_k, s_k) is a B-edge, delete (r_k^ℓ, s_k^ℓ) and (r_k^j, s_k^j) from B' and add (r_k^ℓ, s_k^j) and (r_k^j, s_k^ℓ) instead.
- If (r_k, s_k) is an E-edge, delete (r_k^ℓ, s_k^ℓ) and (r_k^j, s_k^j) from E' and add (r_k^ℓ, s_k^j) and (r_k^j, s_k^ℓ) instead. Then, at all nodes in the subtrees with root s_k^j or s_k^ℓ, rename the constants occurring in the C and Δ-labels from c^ℓ to c^j, resp. vice versa.

In the second case the π-functions associated with the B'-edges are canonically modified to take care of the renaming of constants, e.g. if some π belongs to a B'-edge (u^j, v^j) and $\pi(c^j) = d^j$, then π is changed to send c^ℓ, which now replaces c^j at v^j, to d^j, instead.

This construction achieves the following. If $i \neq j$ there are either exactly two copies of an S-edge connecting T^i and T^j, the case when the binary representations of i and j differ at exactly one position, and no connecting edges in all other cases. Unfortunately the correlation between the names of constants and nodes, that a constant c^i necessarily lives at a node v^i, is destroyed.

Let T^i be $\mathcal{T}'$ restricted to $\{v^i \ : \ v \in V\} \cup \{\lambda\}$, which is the i-th copy of the original $\mathcal{T}$.

Claim. $\mathcal{T}'$ is a complete and clash-free finite completion tree for ψ.

By construction $\mathcal{T}'$ is finite and all labelling functions satisfy the definition of a completion tree. Most of the labels, including the Δ-labels, are isomorphic to some label in $\mathcal{T}$. This in particular shows that $\mathcal{T}'$ is clash-free. Further, for every node w that is blocked in $\mathcal{T}$, the copies w^i in $\mathcal{T}'$ are blocked, too, and if w was explicitly blocked by v in $\mathcal{T}$ then w^i is explicitly blocked by some copy v^j of v. Hence $\mathcal{T}'$ is complete.

Claim. $\mathcal{T}'$ contains no dormant clash.

In the following sense the construction of $\mathcal{T}'$ does not introduce new dormant clashes, although it creates multiple copies of the ones already existing in $\mathcal{T}$. If $D' = (v^{i_1}, w^{j_1}, \boldsymbol{a}^{i_2}, \boldsymbol{b}^{j_2}, \beta)$ is a dormant clash in $\mathcal{T}'$, it follows that $\boldsymbol{a} \neq \boldsymbol{b}$. For consider the following. The π-functions used to verify blockings are compatible with I, so no two distinct constants that co-exist in the C-label of one node are $\sim$-equivalent. Consequently, if we suppose $\boldsymbol{a} = \boldsymbol{b}$, the atomic information for $\boldsymbol{a}^i$ and $\boldsymbol{b}^j$ is necessarily the same, and D' can not be a dormant clash. This also means that $v \neq w$, since $v = w$ and $\boldsymbol{a} \sim \boldsymbol{b}$ would imply $\boldsymbol{a} = \boldsymbol{b}$.

A crucial observation is that the (canonical) projection of any (a^i, b^j)-path in $\mathcal{T}'$ to $\mathcal{T}$, by mapping nodes v^x to v and constants c^y to c, is an (a, b)-path in $\mathcal{T}$. With our precondition $\boldsymbol{a}^{i_2} \sim \boldsymbol{b}^{j_2}$, we get $\boldsymbol{a} \sim \boldsymbol{b}$ in $\mathcal{T}$. Moreover, since the Δ-labels at v^{i_1} and w^{j_1} are copies of the Δ-labels of v and w, $D = (v, w, \boldsymbol{a}, \boldsymbol{b}, \beta)$ already was a dormant clash in $\mathcal{T}$.

For each dormant clash D in $\mathcal{T}$ let $S'_D \subseteq E \cup B'$ be the set of all copies of the original critical edges S_D in $\mathcal{T}$, and let S' be the union of all S'_D. Then, if D' is a dormant clash in $\mathcal{T}'$ and D the clash in $\mathcal{T}$ that it is a copy of, $S'_{D'}$ is a subset of S'_D.

Let $D' = (v^{i_1}, w^{j_1}, \boldsymbol{a}^{i_2}, \boldsymbol{b}^{j_2}, \beta)$ be a dormant clash in $\mathcal{T}'$. We distinguish two cases according to whether $\text{bin}(i_1)$ and $\text{bin}(j_1)$ differ at bit positions associated with edges in S_D.

First assume that there is a difference at an S_D-position and choose one of the corresponding edges $\varepsilon \in S_D$. Every path from v^{i_1} to w^{j_1} (that does not pass through the root λ), in particular any (a^{i_2}, b^{j_2})-path for constants $a^{i_2} \in C'(v^{i_1})$ and $b^{j_2} \in C'(w^{j_1})$, has to pass through an odd number of copies of ε. Since S_D was chosen to contain edges that do not propagate the complete tuple of constants involved in the dormant clash D, the above is the same as to say that there is at least one $a_t^{i_2} \in \boldsymbol{a}^{i_2}$ that is not propagated from v^{i_1} to w^{j_1}. Else, for all $a_t^{i_2} \in \boldsymbol{a}$, we could find $(a_t^{i_2}, b_t^{j_2})$-paths containing a copy of ε. Their canonical projections to $\mathcal{T}$ would be (a_k, b_k)-paths using ε itself, so ε propagates a complete set of constants $\sim$-equivalent to $\boldsymbol{a}$. This is a contradiction to $\varepsilon \in S_D$ because of the choice of S_D. We conclude that $\boldsymbol{a}^{i_2} \not\sim \boldsymbol{b}^{j_2}$, a contradiction to D' being a dormant clash.

Next let i and j only differ at bit positions *not* associated with S_D and let $t = t_D$. If $a_t^{i_2} \sim b_t^{j_2}$, there is an $(a_t^{i_2}, b_t^{j_2})$-path p' that starts at v^{i_1} and ends at w^{j_1}. (In fact, any $(a_t^{i_2}, b_t^{j_2})$-path can be extended in such a way since, for any constant c^x, the set of nodes where c^x occurs is necessarily E'-connected.)

Let p be the canonical projection of p' to $\mathcal{T}$. Then p is an (a_t, b_t)-path from v to w. As i and j do not differ at S_D associated bit-positions, p' contains an even number of S'_D-edges, and the projection p contains an even number of S_D-edges. Since $v \in V_D$, $w \notin V_D$ and p propagates a constant $\sim$-equivalent to a_t, p has to use an S_D-edge for going from V_D to $V \setminus V_D$. Actually, each time p switches between V_D and $V \setminus V_D$ it has to be done via an S_D-edge, otherwise p could not propagate a constant $\sim$-equivalent to a_t. Now the total number of S_D-edges along p is even, so each time p leaves V_D it has to come back again later. This zig-zagging shows that p either does not end at $w \in V \setminus V_D$, or that p does not propagate a_t to w. Either way is a contradiction to the assumption that D' is a dormant clash and we can start with an $(a_t^{i_2}, b_t^{j_2})$-path p'. $\square$

Lemma 5.2.12. *Let $\psi \in$ GF and let $\mathcal{T}$ be a tableau for ψ. Then ψ is (finitely) satisfiable.*

Proof. According to Lemma 5.2.11 we assume that $\mathcal{T} = (V, E, C, \Delta, N, B, I)$ is a tableau for ψ that does not contain critical edges.

Towards the finite satisfiability we construct a finite structure $\mathfrak{A} = \mathfrak{A}(\mathcal{T})$ with universe $A = C(V)/{\sim}$. For each relation $R \in \tau$ and each tuple $\boldsymbol{a} \in A$ of matching arity let $\boldsymbol{a} \in R^{\mathfrak{A}}$ iff there is a node $v \in V$ and a tuple of constants $\boldsymbol{b} \in C(v)$ such that all $b_i \sim a_i$ and $R\boldsymbol{b} \in \Delta(v)$. Note that with $\mathrm{R}_{\updownarrow}$ and the non-existence of dormant clashes, this is the case iff the same holds true independent of the specific choice of $\boldsymbol{b}$ or v. Hence $\mathfrak{A}$ is well defined.

Claim. $\mathfrak{A} \models \psi$.

This is implied by the stronger statement that for every closed formula φ using constants from $\boldsymbol{a}$ that appears in the Δ-label of some unblocked node v of $\mathcal{T}$, $\varphi[\boldsymbol{a} \mapsto \tilde{\boldsymbol{a}}]$ holds in $\mathfrak{A}$. Again φ is assumed to be in NNF.

- For equality statements this is immediate. The $\mathrm{R}_=$-rule makes sure that distinct constants occurring at a common node have distinct $\sim$-classes. For inequality statements, assume that $a \neq b \in \Delta(v)$, but $a \sim b$ for two distinct constants a, b and some node v. However $I(a) \neq I(b)$, since a and b occur in a C-label together. This is a contradiction to (a, b)-paths preserving I-indices.

- For an atomic sentence $R\boldsymbol{a}$, we get $\mathfrak{A} \models R\tilde{\boldsymbol{a}}$ immediately from the construction of $\mathfrak{A}$. In case of a negated atomic sentence, assume $\varphi(\boldsymbol{a}) = \neg R\boldsymbol{a} \in \Delta(v)$ but $\mathfrak{A} \models R\tilde{\boldsymbol{a}}$. This implies the existence of a (dormant) clash in $\mathcal{T}$, a contradiction.

- For positive Boolean combinations the argument is immediate.

- Let $\varphi(\boldsymbol{a}) = (\exists \boldsymbol{y}.\beta(\boldsymbol{a}, \boldsymbol{y}))\eta(\boldsymbol{a}, \boldsymbol{y})$. If, for some $\boldsymbol{b} \in C(v)$, it holds that $\beta(\boldsymbol{a}, \boldsymbol{b}), \eta(\boldsymbol{a}, \boldsymbol{b}) \in \Delta(v)$, we note that $\mathfrak{A} \models \eta(\tilde{\boldsymbol{a}}, \tilde{\boldsymbol{b}})$ and $\mathfrak{A} \models \beta(\tilde{\boldsymbol{a}}, \tilde{\boldsymbol{b}})$ by induction hypothesis for β and η.

If there is no $\boldsymbol{b} \in C(v)$ with $\beta(\boldsymbol{a}, \boldsymbol{b}), \eta(\boldsymbol{a}, \boldsymbol{b}) \in \Delta(v)$, then application of the R$_\exists$-Rule yields a successor node w of v with constants $\boldsymbol{b} \in C(w)$ such that $\beta(\boldsymbol{a}, \boldsymbol{b}), \eta(\boldsymbol{a}, \boldsymbol{b}) \in \Delta(w)$. If w is not blocked, the claim again follows by induction hypothesis for β and η.

If, however, w is blocked, consider the node u with $(u, w) \in B$ and the injection $\pi : C(w) \to C(u)$. Then $\beta(\pi(\boldsymbol{a}), \pi(\boldsymbol{b}))$ and $\eta(\pi(\boldsymbol{a}), \pi(\boldsymbol{b}))$ are in the Δ-label of u. Since all pairs of constants (a, a') where $a' = \pi(a)$ are in the same $\sim$-class, it follows by induction that $\mathfrak{A} \models \beta(\tilde{\boldsymbol{a}}, \tilde{\boldsymbol{b}}) \wedge \eta(\tilde{\boldsymbol{a}}, \tilde{\boldsymbol{b}})$, and hence $\varphi(\tilde{\boldsymbol{a}})$ holds in $\mathfrak{A}$.

- Finally let $\varphi(\boldsymbol{a}) = (\forall \boldsymbol{y}.\beta(\boldsymbol{a}, \boldsymbol{y}))\eta(\boldsymbol{a}, \boldsymbol{y})$. Assume that there is a tuple $\boldsymbol{b}$ such that $\mathfrak{A} \models \beta(\tilde{\boldsymbol{a}}, \tilde{\boldsymbol{b}})$. Then there is a node w where $\tilde{\boldsymbol{a}} \cup \tilde{\boldsymbol{b}} \subseteq C(w)$, i.e., there are tuples $\boldsymbol{a}', \boldsymbol{b}' \subseteq C(w)$ with $\boldsymbol{a}' \sim \boldsymbol{a}$ and $\boldsymbol{b}' \sim \boldsymbol{b}$. Moreover, $\beta(\boldsymbol{a}', \boldsymbol{b}') \in \Delta(w)$ and $\varphi(\boldsymbol{a}') \in \Delta(w)$. Hence, the R$_\forall$-rule is applicable for $\varphi(\boldsymbol{a}')$ at w and must have been applied because $\mathcal{T}$ is complete. This gives us $\eta(\boldsymbol{a}', \boldsymbol{b}') \in \Delta(v)$, which, by induction, yields $\mathfrak{A} \models \eta(\tilde{\boldsymbol{a}}', \tilde{\boldsymbol{b}}')$ and hence $\mathfrak{A} \models \eta(\tilde{\boldsymbol{a}}, \tilde{\boldsymbol{b}})$.

This completes the proof of Lemma 5.2.12. □

Corollary 5.2.13. GF *has the finite model property.*

Proof. If $\psi \in \mathrm{GF}[\tau]$ is satisfiable, then, by Lemma 5.2.6, it has a finite tableau. As shown in the proof of Lemma 5.2.12, such a tableau induces a finite model for ψ. □

Chapter 6

Action Guarded Logics

At this point we take a closer look at the area between modal and guarded logics. Our main focus is on the class of action guarded logics, where the vocabulary will be split into two disjoint sets, the action predicates and the state predicates. Action predicates are only allowed in guard positions, whereas state predicates may only occur as non-guards. It turns out that there is a whole flock of design choices for action guarded logics between ML and GF. We will consider some properties of ML that no longer hold true for GF and identify the exact borders on a map of action guarded logics. The study of bisimulation safety is rounded of by identifying an action guarded fragment that exhibits a behaviour similar to the modal case. Further studies, implicitly or explicitly concerned with our take on action guarded logics, can be found e.g. in [6, 22, 48].

6.1 The Playground

The aim of guarded logics was to capture the good behavioural properties of modal logic by keeping the guarded quantification pattern, and dropping all other restrictions, most prominently the restriction to structures of width at most two. The freedom to use relations of arbitrary arity is one of the central points of the greater flexibility that guarded logics posses over modal logics, and will therefore not be touched. Otherwise we typically end up in 2-variable logic, which has been extensively studied previously, cf. [28, 29] for more information and references.

Already the initial publication on the guarded fragment [1] contained three variants of quantifier patterns.

- $(\exists \boldsymbol{y}.R(\boldsymbol{x},\boldsymbol{y}))\varphi(\boldsymbol{y})$ for GF_1
- $(\exists \boldsymbol{y}.R(\boldsymbol{x},\boldsymbol{y}))\varphi(\boldsymbol{x},\boldsymbol{y})$ for GF_2
- $(\exists \boldsymbol{y}.R(\boldsymbol{x},\boldsymbol{y}))\varphi(\boldsymbol{x},\boldsymbol{y},\boldsymbol{z})$ for GF_3

The 2nd variant, GF_2, is what became to be known as the guarded fragment GF. The 3rd variant, GF_3, was shown to be undecidable, and is therefore unsuitable for a main role in guarded logics. The 1st variant, GF_1, has not attracted much attention for the simple reason that many desired properties hold for the much larger GF_2. On the other hand, research in [48] has shown that restriction to GF_1 is necessary for transferring certain complexity results from modal logics, in this case in the shape of description logics, to the guarded world.

More concretely, compare the guarded fragment, where any quantification $(\exists \boldsymbol{x}'.R(\boldsymbol{x}))\varphi(\boldsymbol{x})$ is valid as long as $\boldsymbol{x}' \subseteq \boldsymbol{x}$, and modal logic, where the quantifier pattern is $(\exists y.E(x,y))\varphi(y)$. We see that GF_1 more closely resembles the modal quantifier pattern than GF. This is motivation enough to let the question of whether a logic allows the re-use of old elements in the body of a quantification be one of our three main distinguishing features.

The naming feature of action guarded logics is the partitioning of the vocabulary τ into an action vocabulary τ_a and a state vocabulary τ_s. An action guarded quantification $(\exists \boldsymbol{x}.R(\boldsymbol{x}))$ is required to use an $R \in \tau_a$ as guard, whereas only predicates from τ_s are allowed in non-guard positions. Compare this to modal logics, where similarly binary predicates occur only in modalities, i.e. as guards of quantifications, and unary predicates occur only independent of quantifications.

Finally, note that modal logic equips guard relations with a sense of direction. An edge relation E permits modal operators that translate into $(\forall y.Exy)$ and $(\exists y.Exy)$, but not the inverse $(\forall y.E(yx))$. We generalise this aspect by optionally regarding an n-ary action relation as $(i;n-i)$-ary. This fixes the first i positions as input or source elements, and the last $n-i$ positions of E-guarded tuples as output or destination elements. In this sense, the binary relations in modal logic are action predicates of arity $(1;1)$. For example, a guard relation R of arity $(2;2)$ can be used as $(\forall yz.R(uv.yz))$, but not as $(\forall xyz.R(xy.uz))$, as is possible in GF.

All in all there is a whole flock of discerning properties, some of which can be turned on and off independently, that break down the step from ML to GF. Freely combining these parameters leads to an impressive number of different logics. Not considered here are variants of guarded logic where the use of equality is restricted. Some aspects of restricting the vocabulary to that of labelled graphs are discussed in Sections 4.1 and 7.2.

Definitions. For the following definitions τ, τ_s and τ_a are relational vocabularies. For the logics with directional actions, the action vocabulary τ_a consists of relations with split arity $(i;j)$, where $i, j \geq 1$. The state vocabulary τ_s contains relations with normal arities i. For all guard atoms α the notation $\alpha(\boldsymbol{x}, \boldsymbol{y})$ means that α contains *all* variables from $\boldsymbol{x}$ and $\boldsymbol{y}$. We consider the following combinations.

Syntax of GF_1**.** (Disallow re-use of old elements.)

(i) If $R \in \tau \cup \{=\}$, then $R(\boldsymbol{x})$ is in GF_1.
(ii) GF_1 is closed under Boolean combinations.
(iii) If $\psi(\boldsymbol{y})$ is a formula of $\mathrm{GF}_1$and $G(\boldsymbol{x}, \boldsymbol{y})$ is a τ-atom, then $\exists \boldsymbol{y}\big(G(\boldsymbol{x}, \boldsymbol{y}) \wedge \psi(\boldsymbol{y})\big)$ is in GF_1.

Syntax of SGF**.** (Split action/state vocabulary.)

(i) If $R \in \tau_s \cup \{=\}$, then $R(\boldsymbol{x})$ is in SGF.
(ii) SGF is closed under Boolean combinations.
(iii) If $\psi(\boldsymbol{x}, \boldsymbol{y})$ is a formula of SGF and $G(\boldsymbol{x}, \boldsymbol{y})$ is a τ_a-atom, then $\exists \boldsymbol{y}\big(G(\boldsymbol{x}, \boldsymbol{y}) \wedge \psi(\boldsymbol{x}, \boldsymbol{y})\big)$ is in SGF.

Syntax of SGF_1**.** (Split action/state vocabulary; disallow re-use.)

(i) If $R \in \tau_s \cup \{=\}$, then $R(\boldsymbol{x})$ is in SGF_1.
(ii) SGF_1 is closed under Boolean combinations.
(iii) If $\psi(\boldsymbol{x}, \boldsymbol{y})$ is a formula of SGF_1 and $\alpha(\boldsymbol{x}, \boldsymbol{y})$ is a τ_a-atom, then $\exists \boldsymbol{y}\big(G(\boldsymbol{x}, \boldsymbol{y}) \wedge \psi(\boldsymbol{y})\big)$ is in SGF_1.

Syntax of AGF**.** (Split action/state vocabulary with directed actions.)

(i) If $R \in \tau_s \cup \{=\}$, then $R(\boldsymbol{x})$ is in AGF.
(ii) AGF is closed under Boolean operations.
(iii) If $G \in \tau_a$, $\varphi(\boldsymbol{x}, \boldsymbol{y}) \in$ AGF, G is of arity $(|\boldsymbol{x}|; |\boldsymbol{y}|)$ and $\boldsymbol{y} \cap \boldsymbol{x} = \emptyset$, then $\exists \boldsymbol{y}(G(\boldsymbol{x}; \boldsymbol{y}) \wedge \varphi(\boldsymbol{x}, \boldsymbol{y}))$ is in AGF.

Syntax of AGF_1**.** (Split action/state vocabulary with directed actions; disallow re-use of old elements.)

(i) If $R \in \tau_s \cup \{=\}$, then $R(\boldsymbol{x})$ is in AGF_1.

(ii) AGF_1 is closed under Boolean operations.

(iii) If $G \in \tau_a$, $\varphi(\boldsymbol{y}) \in \mathrm{AGF}_1$, G is of arity $(|\boldsymbol{x}|; |\boldsymbol{y}|)$ and $\boldsymbol{y} \cap \boldsymbol{x} = \emptyset$, then $\exists \boldsymbol{y}(G(\boldsymbol{x}; \boldsymbol{y}) \wedge \varphi(\boldsymbol{y}))$ is in AGF_1.

Syntax of AGFI**.** (Split action/state vocabulary with directed actions and their inverse.)

(i) If $R \in \tau_s \cup \{=\}$, then $R(\boldsymbol{x})$ is in AGFI.

(ii) AGFI is closed under Boolean operations.

(iii) If $G \in \tau_a$, $\varphi(\boldsymbol{x}, \boldsymbol{y}) \in \mathrm{AGFI}$, G is of arity $(|\boldsymbol{x}|; |\boldsymbol{y}|)$ and $\boldsymbol{y} \cap \boldsymbol{x} = \emptyset$, then $\exists \boldsymbol{y}(G(\boldsymbol{x}; \boldsymbol{y}) \wedge \varphi(\boldsymbol{x}, \boldsymbol{y}))$ and $\exists \boldsymbol{x}(G(\boldsymbol{x}; \boldsymbol{y}) \wedge \varphi(\boldsymbol{x}, \boldsymbol{y}))$ are in AGFI.

Syntax of AGFI_1**.** (Split action/state vocabulary with directed actions and their inverse; disallow re-use of old elements.)

(i) If $R \in \tau_s \cup \{=\}$, then $R(\boldsymbol{x})$ is in AGFI_1.

(ii) AGFI_1 is closed under Boolean operations.

(iii) If $G \in \tau_a$, $\varphi(\boldsymbol{x}), \varphi'(\boldsymbol{y}) \in \mathrm{AGFI}_1$, G is of arity $(|\boldsymbol{x}|; |\boldsymbol{y}|)$ and $\boldsymbol{y} \cap \boldsymbol{x} = \emptyset$, then $\exists \boldsymbol{y}(G(\boldsymbol{x}; \boldsymbol{y}) \wedge \varphi'(\boldsymbol{y}))$ and $\exists \boldsymbol{x}(G(\boldsymbol{x}; \boldsymbol{y}) \wedge \varphi(\boldsymbol{x}))$ are in AGFI_1.

Finding the correct notion of bisimulation for a given guarded quantifier pattern is a straightforward adaptation of the standard guarded bisimulation, at least for logics with undirected actions. These variants implicitly contain global quantification in the sense of quantifying over all free variables of the subformula. This lets bisimulation be a notion between structures as such, without selected starting tuples. Concerning the format of formulae, this is a very prominent dividing line, since only logics that allow global quantification contain formulae without free variables, or sentences.

A Note on States. What is missing in the $\mathrm{AGF}_{(1)}$-logics, is a uniform notion of what a *state* is. In modal style logics, every node of the graph or transition system is a state. Formulae typically define sets of states. In the guarded fragment, this role is assumed by the guarded tuples.

For the $\mathrm{AGF}_{(1)}$-logics, there are two types of distinguished tuples, those that are the source of an action, and those that are the destination of an action. For directional actions, we called source tuples *active*, and destination tuples *guarded*, i.e. the two notions that coincide for the guarded fragment are split according to their, more or less natural, differing meaning. The sets of active and guarded tuples will in general not coincide, so taking either one or the other as set of states of a structure is non-uniform in an intuitive sense. What makes this problematic, is how an (action) guarded logic with directional

$$\begin{tikzcd}
\mathrm{GF} & \\
\mathrm{SGF} \arrow[u] & \mathrm{GF}_1 \arrow[ul] \\
\mathrm{AGFI} \arrow[u] & \mathrm{SGF}_1 \arrow[u] \arrow[ul] \\
\mathrm{AGF} \arrow[u] & \mathrm{AGFI}_1 \arrow[u] \arrow[ul] \\
 & \mathrm{AGF}_1 \arrow[u] \arrow[ul] \\
 & \mathrm{ML} \arrow[u, dashed]
\end{tikzcd}$$

Figure 6.1: Inclusion map of the major (action) guarded logics.

actions uses actions. A formula $\varphi(\boldsymbol{x})$ defines a set of active tuples, however it depends on local atomic properties of active *and* guarded tuples.

Calling every active or guarded tuple a state has the disadvantage that the question of whether a tuple is a state or not does not solely rely on (atomic) properties of said tuple, as is the case for GF where states and actions are not distinguished, or is trivially given, as for ML.

In logics with directional guards, but without their inverses, the set of active tuples can be easily defined, whereas no first-order formula that adheres to the prescribed quantifier pattern can define the set of guarded tuples. As soon as inverse actions are allowed, every active tuple is guarded, and vice versa, so this division disappears.

Definition 6.1.1. Let $\mathfrak{A}$ be a $\tau_a \dot\cup \tau_s$-structure.

A set $X \subseteq A$ is AGF$_1$-*active* if there are $\boldsymbol{a}$, $\boldsymbol{b}$ in $\mathfrak{A}$, and an $R \in \tau_a$, such that $(\boldsymbol{a}; \boldsymbol{b}) \in R^{\mathfrak{A}}$ and $X \subseteq \boldsymbol{a}$.

A set $X \subseteq A$ is AGF$_1$-*guarded* if there are $\boldsymbol{a}$, $\boldsymbol{b}$ in $\mathfrak{A}$, and an $R \in \tau_a$, such that $(\boldsymbol{a}; \boldsymbol{b}) \in R^{\mathfrak{A}}$ and $X \subseteq \boldsymbol{b}$.

In both cases the notation should imply that the arity of R is $(|\boldsymbol{a}|; |\boldsymbol{b}|)$.

Bisimulations. We give the precise definitions for the bisimulations corresponding to the AGF$_1$, AGFI and SGF$_1$ quantifier patterns. The remaining variants for AGF, AGFI$_1$, SGF and GF$_1$ can be interpolated from these, and the GF-bisimulation given in Section 3.4. Remember that the action vocabulary is directed for AGF and AGFI, and contains normal relations for SGF$_1$.

Definition 6.1.2. Let $\mathfrak{A}$ and $\mathfrak{B}$ be $\tau_a \dot{\cup} \tau_s$-structures.

An AGF$_1$-bisimulation between $\mathfrak{A}$ and $\mathfrak{B}$ is a non-empty set I of finite partial τ_s-isomorphisms f between $\mathfrak{A}$ and $\mathfrak{B}$ that is closed under subfunctions and satisfies the following conditions.

Forth. For every $f : X \to Y \in I$, active tuple $\boldsymbol{a} \subseteq X$, action predicate $R \in \tau_a$ and tuple $\boldsymbol{a}' \subseteq A$ where $(\boldsymbol{a}; \boldsymbol{a}') \in R^{\mathfrak{A}}$, and for $\boldsymbol{b} = f(\boldsymbol{a}) \subseteq Y$ there is a tuple $\boldsymbol{b}' \subseteq B$ such that $(\boldsymbol{b}; \boldsymbol{b}') \in R^{\mathfrak{B}}$ and $f' : \boldsymbol{a}' \mapsto \boldsymbol{b}' \in I$.

Back. Similarly for every $f : X \to Y \in I$, active tuple $\boldsymbol{b} \subseteq Y$, action predicate $R \in \tau_a$ and tuple $\boldsymbol{b}' \subseteq B$ where $(\boldsymbol{b}; \boldsymbol{b}') \in R^{\mathfrak{B}}$, and for $\boldsymbol{a} = f^{-1}(\boldsymbol{b}) \subseteq X$ there is a tuple $\boldsymbol{a}' \subseteq A$ such that $(\boldsymbol{a}; \boldsymbol{a}') \in R^{\mathfrak{B}}$ and $f' : \boldsymbol{a}' \mapsto \boldsymbol{b}' \in I$.

In this and the next chapter, AGF$_1$ will be the most prominently used action guarded logic. We reserve the notation $\sim_1$ for AGF$_1$-bisimulation. Two structures $\mathfrak{A}$ and $\mathfrak{B}$ with tuples $\boldsymbol{a} \in \mathfrak{A}$, $\boldsymbol{b} \in \mathfrak{B}$ are AGF$_1$-bisimilar if there is an AGF$_1$-bisimulation $I : \mathfrak{A} \sim_1 \mathfrak{B}$ such that $f : \boldsymbol{a} \to \boldsymbol{b}$ is in I. The bisimulation for AGFI differs from AGF$_1$ in that the old elements $\boldsymbol{a}$ resp. $\boldsymbol{b}$ have to be covered by the f' too. The inverse actions are treated by additional requirements that simply turn around the guard relations.

Definition 6.1.3. Let $\mathfrak{A}$ and $\mathfrak{B}$ be $\tau_a \dot{\cup} \tau_s$-structures.

An AGFI-bisimulation between $\mathfrak{A}$ and $\mathfrak{B}$ is a non-empty set I of finite partial τ_s-isomorphisms f between $\mathfrak{A}$ and $\mathfrak{B}$ that is closed under subfunctions and satisfies the following conditions.

Forth. For every $f : X \to Y \in I$, active tuple $\boldsymbol{a} \subseteq X$, action predicate $R \in \tau_a$ and tuple $\boldsymbol{a}' \subseteq A$ where $(\boldsymbol{a}; \boldsymbol{a}') \in R^{\mathfrak{A}}$, and for $\boldsymbol{b} = f(\boldsymbol{a}) \subseteq Y$ there is a tuple $\boldsymbol{b}' \subseteq B$ such that $(\boldsymbol{b}; \boldsymbol{b}') \in R^{\mathfrak{B}}$ and $f' : \boldsymbol{a}\boldsymbol{a}' \mapsto \boldsymbol{b}\boldsymbol{b}' \in I$.

Forth'. For every $f : X \to Y \in I$, guarded tuple $\boldsymbol{a} \subseteq X$, action predicate $R \in \tau_a$ and tuple $\boldsymbol{a}' \subseteq A$ where $(\boldsymbol{a}'; \boldsymbol{a}) \in R^{\mathfrak{A}}$, and for $\boldsymbol{b} = f(\boldsymbol{a}) \subseteq Y$ there is a tuple $\boldsymbol{b}' \subseteq B$ such that $(\boldsymbol{b}'; \boldsymbol{b}) \in R^{\mathfrak{B}}$ and $f' : \boldsymbol{a}\boldsymbol{a}' \mapsto \boldsymbol{b}\boldsymbol{b}' \in I$.

Back. Similarly for every $f : X \to Y \in I$, active tuple $\boldsymbol{b} \subseteq Y$, action predicate $R \in \tau_a$ and tuple $\boldsymbol{b}' \subseteq B$ where $(\boldsymbol{b}; \boldsymbol{b}') \in R^{\mathfrak{B}}$, and for $\boldsymbol{a} = f^{-1}(\boldsymbol{b}) \subseteq X$ there is a tuple $\boldsymbol{a}' \subseteq A$ such that $(\boldsymbol{a}; \boldsymbol{a}') \in R^{\mathfrak{B}}$ and $f' : \boldsymbol{a}\boldsymbol{a}' \mapsto \boldsymbol{b}\boldsymbol{b}' \in I$.

Back'. Finally for every $f : X \to Y \in I$, active tuple $\boldsymbol{b} \subseteq Y$, action predicate $R \in \tau_a$ and tuple $\boldsymbol{b}' \subseteq B$ where $(\boldsymbol{b}'; \boldsymbol{b}) \in R^{\mathfrak{B}}$, and for $\boldsymbol{a} = f^{-1}(\boldsymbol{b}) \subseteq X$ there is a tuple $\boldsymbol{a}' \subseteq A$ such that $(\boldsymbol{a}'; \boldsymbol{a}) \in R^{\mathfrak{B}}$ and $f' : \boldsymbol{a}\boldsymbol{a}' \mapsto \boldsymbol{b}\boldsymbol{b}' \in I$.

For SGF$_1$ we can forget about directions and inverses of actions. The combination with not allowing the re-use of old elements nevertheless makes the formalisation of SGF$_1$-bisimulation significantly more complex than the standard guarded bisimulation for GF. The distinction between action and state relations does not make a great difference.

Definition 6.1.4. Let $\mathfrak{A}$ and $\mathfrak{B}$ be $\tau_a \dot\cup \tau_s$-structures.

A SGF$_1$-bisimulation between $\mathfrak{A}$ and $\mathfrak{B}$ is a non-empty set I of finite partial τ_s-isomorphisms f between $\mathfrak{A}$ and $\mathfrak{B}$ that is closed under subfunctions and satisfies the following conditions.

Forth. For every $f : X \to Y \in I$ and every τ_a-guarded set $X' \dot\cup X'' \subseteq \mathfrak{A}$ guarded by an atom α, $X'' \subseteq X$, there are $Y', Y'' \subseteq \mathfrak{B}$, $Y'' \subseteq Y$, such that an $f' : X' \to Y'$ is in I, and for some enumerations $\boldsymbol{a}'' = X''$ and $\boldsymbol{a}' = X'$ there is a bijection g'' such that both $\mathfrak{A} \models \alpha(\boldsymbol{a}', \boldsymbol{a}'')$ and $\mathfrak{B} \models \alpha(f'(\boldsymbol{a}'), g''(\boldsymbol{a}''))$.

Back. Similarly for every $f : X \to Y \in I$ and every τ_a-guarded set $Y' \dot\cup Y'' \subseteq \mathfrak{B}$ guarded by an atom α, $Y'' \subseteq Y$, there are $X', X'' \subseteq \mathfrak{A}$, $X'' \subseteq X$, such that an $f' : X' \to Y'$ is in I, and for some enumerations $\boldsymbol{b}'' = Y''$ and $\boldsymbol{b}' = Y'$ there is a bijection g'' such that both $\mathfrak{B} \models \alpha(\boldsymbol{b}', \boldsymbol{b}'')$ and $\mathfrak{A} \models \alpha(f'(\boldsymbol{b}'), g''(\boldsymbol{b}''))$.

Now that we have the various bisimulations available it can easily be shown that the definitions above do not yield semantically equivalent logics.

Theorem 6.1.5. *All inclusions in Figure 6.1 are strict.*

Proof. To separate a logic L from its L$_1$ variant, consider a vocabulary consisting of one binary action predicate E and one binary state predicate R. For the logics with directional actions, E has arity $(1, 1)$. Let $\mathfrak{A}$ be the structure consisting of four nodes a_1, a_2 and b_1, b_2 such that $E(a_1, a_2)$, $E(b_1, b_2)$ and $R(a_1, a_2)$ hold. In AGF, a_1 and b_1 can be distinguished by the formula $(\exists y.Exy)Rxy$. In contrast, let I be the set of partial automorphisms on $\mathfrak{A}$ that contains id_A, the identity function on A, all its subfunctions, and the two functions $a_1 \mapsto b_1$ and $a_2 \mapsto b_2$. Then it can be verified that I is a GF_1-bisimulation between $\mathfrak{A}, a_1$ and $\mathfrak{A}, b_1$. In particular, GF_1 can not distinguish a_1 from b_1.

The other cases are shown similarly. For GF and SGF, consider the formula $(\forall xy.Exy)Rxy \wedge (\forall xy.Rxy)Exy \in \mathrm{GF}_1$ that forces E and R to coincide. Clearly, no SGF-bisimulation can check both inclusions — if e.g. E is a state predicate, then there is no requirement for E-guarded sets. If however both E and R are action predicates, then the partial homomorphisms in the bisimulation speak about an empty state vocabulary.

For SGF and AGFI, the difference is eminent from their syntactic definition. In SGF and SGF_1 it is possible to form closed formulae, or sentences, whereas every AGFI and AGFI_1-formula has free variables, and the quantifier pattern always moves from a tuple of old variables to a freshly quantified tuple. The result is that $\mathrm{AGFI}_{(1)}$-formulae can only make statements about the connected component of their respective arguments. However, even when closing AGFI under full quantification, i.e. allowing statements of the form $\exists \boldsymbol{xy}.E(\boldsymbol{x};\boldsymbol{y})\varphi(\boldsymbol{x},\boldsymbol{y})$, one discovers that SGF_1 has strictly greater expressive power. This time an example requires higher arity. Let E be an action predicate of arity $(2;1)$. Then $\exists yz.E(xy;z)\mathit{true}$ can be expressed in SGF, which ignores the splitting of the arity, but arguably not in AGFI_1.

For AGFI and AGF one can argue as for ML^- and ML. Namely, AGFI can express that a given node has a predecessor via some edge relation, whereas AGF is limited to following edges in their given direction. E.g. $(\exists y.Eyx)\mathit{true}$, an AGFI_1 formula, can distinguish between two AGF-bisimilar nodes, one of which has an E-predecessor. $\Box$

Bisimulation Invariance. The definitions of invariant relations and saturated structures first given for the modal case in Section 2.6 can be phrased likewise for any of the guarded fragments depicted in Figure 6.1. We make explicit the version with AGF_1 as placeholder.

Definition 6.1.6. A global relation R is *invariant* for AGF_1-bisimulation if, whenever $\mathfrak{A}, \boldsymbol{a} \sim_1 \mathfrak{B}, \boldsymbol{b}$ and $\boldsymbol{a} \in R^{\mathfrak{A}}$, then also $\boldsymbol{b} \in R^{\mathfrak{B}}$.

Definition 6.1.7. A τ-structure $\mathfrak{A}$ is AGF_1-saturated if for all active $\boldsymbol{a} \in A$, $G \in \tau_a$, $\boldsymbol{a}' \subseteq \boldsymbol{a}$ and every AGF_1-type p:
If $(\exists \boldsymbol{y}.G(\boldsymbol{a}',\boldsymbol{y}))p_0(\boldsymbol{y})$ is consistent with the AGF_1-theory of $\boldsymbol{a}'$ in $\mathfrak{A}$ for every finite $p_0 \subseteq p$, then there is a tuple $\boldsymbol{b} \in \mathfrak{A}$ such that $\mathfrak{A} \models G(\boldsymbol{a}',\boldsymbol{b}))p(\boldsymbol{b})$.

As mentioned the proofs to Lemma 2.6.3 and Theorem 2.6.5 can be nearly copied in verbatim to show the following statements about AGF_1, or indeed any other of the action guarded fragments.

Proposition 6.1.8. *If $\mathfrak{A}$ is ω-saturated, then $\mathfrak{A}$ is AGF_1-saturated.*

This follows from AGF_1 being a subset of FO. In preparation of the characterisation theorem below we again establish the following correspondence between logical equivalence and bisimilarity for saturated structures.

Lemma 6.1.9. *Let $\mathfrak{A}$ and $\mathfrak{B}$ be AGF_1-saturated structures. The relation of AGF_1-equivalence is an AGF_1-bisimulation for $\mathfrak{A}$ and $\mathfrak{B}$.*

Proof. Consider two tuples $\boldsymbol{a} \in \mathfrak{A}$, $\boldsymbol{b} \in \mathfrak{B}$ that have the same atomic type. If both $\boldsymbol{a}$ and $\boldsymbol{b}$ are not active, there is nothing to show; if one is active and the other not, then they are not AGF_1-equivalent. Suppose that $\boldsymbol{a}$ and $\boldsymbol{b}$ are AGF_1-equivalent. For the forth condition, consider an action guard α and tuple $\boldsymbol{a}' \in \mathfrak{A}$ that satisfy $\mathfrak{A} \models \alpha(\boldsymbol{a}, \boldsymbol{a}')$. Let p be the AGF_1-type of $\boldsymbol{a}'$. Since the AGF_1-types of $\boldsymbol{a}$ and $\boldsymbol{b}$ coincide, and $(\exists \boldsymbol{y}.\alpha(\boldsymbol{x}, \boldsymbol{y})p(\boldsymbol{y})$ is necessarily consistent with the theory of $\boldsymbol{a}$, by AGF_1-saturation of $\mathfrak{B}$ there is a tuple $\boldsymbol{b}' \in \mathfrak{B}$ that satisfies $\mathfrak{B} \models \alpha(\boldsymbol{b}, \boldsymbol{b}') \wedge p(\boldsymbol{b}')$. The back condition is shown similarly. □

Theorem 6.1.10. *A first-order definable global relation is invariant for AGF_1-bisimulation iff it is AGF_1 definable.*

Proof. Suppose that $\varphi(\boldsymbol{x}) \in \mathrm{FO}$ is AGF_1-bisimulation invariant. Let $\Psi = \{\psi \in \mathrm{AGF}_1 \ : \ \varphi(\boldsymbol{x}) \models \psi(\boldsymbol{x})\}$.

Let $\mathfrak{A} \models \Psi(\boldsymbol{a})$ and suppose that $\mathfrak{A}$ was chosen ω-saturated. Let p be the AGF_1-type of $\boldsymbol{a}$ in $\mathfrak{A}$ and note that $p \cup \{\varphi\}$ is consistent. For surely p is consistent, and if the union is not, then $\varphi \models \neg\pi_1 \vee \cdots \vee \neg\pi_n$ for some $\pi_i \in p$. Hence $\neg\pi_1 \vee \cdots \vee \neg\pi_n \in \Psi$, which means that $(\neg\pi_1 \vee \cdots \vee \neg\pi_n)(\boldsymbol{a})$ holds, a contradiction to $\pi_i \in p$, the AGF_1-type of $\boldsymbol{a}$.

Since $p \cup \{\varphi\}$ is consistent, there is an ω-saturated $\mathfrak{A}'$, and a $\boldsymbol{a}' \in \mathfrak{A}'$ such that $\mathfrak{A}' \models \bigwedge(p \cup \{\varphi\})(\boldsymbol{a})$. Now $\boldsymbol{a}$ and $\boldsymbol{a}'$ both have the same AGF_1-type, and both are tuples of AGF_1-saturated structures, so $\boldsymbol{a}$ and $\boldsymbol{a}'$ are AGF_1-bisimilar. Since φ is AGF_1-bisimulation invariant, φ holds at $\boldsymbol{a}$, too.

In other words, because φ is invariant for AGF_1-bisimulation, we have $\Psi \models \varphi$. By compactness for FO there is a finite $\Psi_0 \subset \Psi$ such that $\Psi_0 \models \varphi(\boldsymbol{x})$, and $\varphi(\boldsymbol{x}) \equiv \bigwedge \Psi_0$, i.e. $\bigwedge \Psi_0 \equiv \varphi$ is the desired AGF_1-formula. □

6.2 Action Guarded Safety

With respect to safety for bisimulation, the action guarded fragments show the two different behaviours encountered earlier. For GF, safety and invariance more or less coincide, whereas for ML they have a completely different format. As a byproduct of the two terms falling together for GF, the characterisation theorems for the safe, resp. invariant, fragments of first-order logic coincide too. Our goal is the generalisation of Theorem 2.6.8 that will be discussed in the subsequent section.

The first-order operation most critical wrt. safe actions often is the concatenation of actions. We will see that concatenation preserves safety only in the case of AGF_1. For the logics with undirected actions, only guarded relations are safe, and concatenation therefore not considered.

Theorem 6.2.1. *A global relation R is safe for* SGF*-bisimulation iff it is guarded and invariant for* SGF*-bisimulation.*

A global relation R is safe for SGF_1*-bisimulation iff it is guarded and invariant for* SGF_1*-bisimulation.*

The proof is an immediate corollary of Theorem 3.5.5. The point made there was that for logics beyond SGF_1, any relation that contains non-guarded tuples is immediately not safe for the corresponding notion of bisimulation.

Corollary 6.2.2. *A first-order definable global relation is safe for* SGF*-bisimulation iff it is equivalent to $\varphi(\boldsymbol{x}) \wedge \mathbb{G}(\boldsymbol{x})$ where $\varphi(\boldsymbol{x}) \in$* SGF *and $\mathbb{G}(\boldsymbol{x})$ says that $\boldsymbol{x}$ is τ_a-guarded.*

A first-order definable global relation is safe for SGF_1*-bisimulation iff it is equivalent to $\varphi(\boldsymbol{x}) \wedge \mathbb{G}(\boldsymbol{x})$ where $\varphi(\boldsymbol{x}) \in \mathrm{SGF}_1$ and $\mathbb{G}(\boldsymbol{x})$ says that $\boldsymbol{x}$ is τ_a-guarded.*

The AGF variants can hope to emulate the greater flexibility of ML concerning the possible inductive construction of safe relations. Consider two directional action guards $\alpha(\boldsymbol{x};\boldsymbol{y})$ and $\alpha'(\boldsymbol{y};\boldsymbol{z})$. We can form their concatenation $\alpha;\alpha'$, an action guard from $\boldsymbol{x}$ to $\boldsymbol{y}'$, with the obvious semantics.

$$(\alpha;\alpha')^{\mathfrak{A}} = \{(\boldsymbol{a};\boldsymbol{b}) \;:\; \mathfrak{A} \models \exists \boldsymbol{y}(\alpha(\boldsymbol{a},\boldsymbol{y}) \wedge \alpha'(\boldsymbol{y},\boldsymbol{b}))\}$$

In the next section the requirement will be relaxed to allow any combination $\alpha(\boldsymbol{x};\boldsymbol{y})$, $\alpha'(\boldsymbol{y}',\boldsymbol{z})$ where $\boldsymbol{y}' \subseteq \boldsymbol{y}$. However even the given form of concatenation is not safe for AGF. Take for example

$$\varphi(x) = (\forall y.Exz;Ezy)Qxy$$

with action predicate E and state predicate Q. Consider the structures $\mathfrak{A} = (\{a,b,c\}, E = \{(a,b),(b,c)\}, Q = \{a,c\})$ and $\mathfrak{A}' = (\{a,b,c\}, E = \{(a,b),(b,c)\}, Q = \emptyset)$. We can easily argue that $\mathfrak{A}, a$ and $\mathfrak{A}', a$ are AGF-bisimilar. Since the tuple (ac) is not $\tau_a = \{E\}$-guarded, the atomic type of (ac) in the two structures need not coincide. However $\mathfrak{A} \models \varphi(a)$, and $\mathfrak{A}' \not\models \varphi(a)$, so the global relation defined by $E;E$ is not safe for AGF. The problem arises from the resulting action not being guarded, in combination with AGF allowing the source variables to be used in the rest of the formula. As we will see, concatenation *is* safe for AGF_1.

To circumvent the problem with concatenation for AGF, we could impose the restriction that in all structures the relations in τ_s are τ_a-guarded. However this restriction has undesirable side effects. Consider the formula $\varphi(x) = (\forall xy.(E;E)(x,y))R(xy)$ and let E be the only action predicate. If R is assumed E-guarded, $R(xy)$ implies $E(xy) \vee E(yx)$. The guard $(E;E)$ translates into something like $\exists z.(E(xz) \wedge E(zy))$. In combination, this enforces a kind of undirected transitivity of E.

The other program operations on actions as found in PDL are less rebellious. Union of actions is safe for any (action-)guarded fragment. It can be pulled in front of the quantification as disjunction. Similarly intersection is not safe for all logics with separated action and state vocabulary, since there is no way to check or enforce any kind of dependency between two distinct actions. The test operation $\varphi?$ can be replaced by a conjunction, instead of $\langle\varphi?\rangle\psi$ write $\varphi \wedge \psi$. In the next section we perform such a decomposition of programs in more detail.

6.2.1 Safety for AGF_1

This section is a generalisation of Theorem 2.6.8 that characterises the modal bisimulation safe fragment of first-order logic in terms of PDL programs. The definitions and proofs given here are mostly obtained by syntactic translation of the proof given in [40] to the more general AGF_1 setting. To this end we first require a guarded equivalent of PDL in order to define action relations of higher arities at all. We have found an appropriate guarded generalisation of PDL in what we call[1] GPDL_1.

[1]GPDL_1 is more than a propositional logic; we keep the "P" in the name to make the relationship to PDL more visible.

Syntax of GPDL$_1$.

1. If $R \in \tau_s \cup \{=\}$, then $R(\boldsymbol{x})$ is a GPDL$_1$ formula.
2. The GPDL$_1$ formulae are closed under Boolean operations.
3. If $\pi(\boldsymbol{x};\boldsymbol{y})$ is a GPDL$_1$program, $\varphi(\boldsymbol{y}')$ is a GPDL$_1$ formula and $\boldsymbol{y}' \subseteq \boldsymbol{y}$, then $\langle\pi(\boldsymbol{x};\boldsymbol{y})\rangle\varphi(\boldsymbol{y}')$ is a GPDL$_1$ formula with free variables $\boldsymbol{x}$.

1. If $G \in \tau_a$ and $\boldsymbol{x} \cap \boldsymbol{y} = \emptyset$, then $G(\boldsymbol{x};\boldsymbol{y})$ is a GPDL$_1$ program.
2. If $\pi(\boldsymbol{x};\boldsymbol{y})$ and $\pi'(\boldsymbol{x};\boldsymbol{y})$ are GPDL$_1$ programs, then so is $\pi(\boldsymbol{x};\boldsymbol{y}) \cup \pi'(\boldsymbol{x};\boldsymbol{y})$.
3. If $\pi(\boldsymbol{x};\boldsymbol{y})$ and $\pi'(\boldsymbol{x}';\boldsymbol{y}')$ are GPDL$_1$ programs, $\boldsymbol{y} \supseteq \boldsymbol{x}'$ and $\boldsymbol{x} \cap \boldsymbol{y}' = \emptyset$, then so is $\pi(\boldsymbol{x};\boldsymbol{y});\pi'(\boldsymbol{x}';\boldsymbol{y}')$.
4. If $\varphi(\boldsymbol{x})$ is a GPDL$_1$ formula, then $\varphi(\boldsymbol{x})?$ is a GPDL$_1$ program from $\boldsymbol{x}$ to $\boldsymbol{x}$.
5. If $\pi(\boldsymbol{x};\boldsymbol{y})$ is a GPDL$_1$ program and $|\boldsymbol{x}| = |\boldsymbol{y}|$, then so is $\pi(\boldsymbol{x};\boldsymbol{y})^*$.

This definition implicitly allows projection of programs, e.g. $G(x;yz); z = z$ is a program from x to z. Since we are inside a first-order setting, we will be mainly concerned with GPDL$_1^-$, the fragment of GPDL$_1$ that does not use the $*$-operator. This hand-tailored variant of PDL has a close relationship to AGF$_1$. Every AGF$_1$-formula can be syntactically transformed into a GPDL$_1^-$-formula by rewriting existential quantifiers ($\exists \boldsymbol{y} G(\boldsymbol{x};\boldsymbol{y})$) as $\langle G(\boldsymbol{x};\boldsymbol{y})\rangle$. Vice versa, for every GPDL$_1^-$ formula φ there is an equivalent AGF$_1$ formula $\varphi^\triangleright$ obtained by inductively decomposing the programs in φ until only atomic programs remain.

$$\begin{aligned}
(\langle G(\boldsymbol{x},\boldsymbol{y})\rangle\chi(\boldsymbol{y}))^\triangleright &= (\exists \boldsymbol{y}.G(\boldsymbol{x},\boldsymbol{y}))(\chi(\boldsymbol{y}))^\triangleright \\
(\langle \pi(\boldsymbol{x},\boldsymbol{y}) \cup \pi'(\boldsymbol{x},\boldsymbol{y})\rangle\chi(\boldsymbol{y}))^\triangleright &= (\langle \pi(\boldsymbol{x},\boldsymbol{y})\rangle\chi(\boldsymbol{y}))^\triangleright \wedge (\langle \pi'(\boldsymbol{x},\boldsymbol{y})\rangle\chi(\boldsymbol{y}))^\triangleright \\
(\langle \pi(\boldsymbol{x},\boldsymbol{y});\pi'(\boldsymbol{y},\boldsymbol{z})\rangle\chi(\boldsymbol{z}))^\triangleright &= (\langle \pi(\boldsymbol{x},\boldsymbol{y})\rangle\langle \pi'(\boldsymbol{y},\boldsymbol{z})\rangle\chi(\boldsymbol{z}))^\triangleright \\
(\langle \varphi(\boldsymbol{x})?\rangle\chi(\boldsymbol{x}))^\triangleright &= (\varphi(\boldsymbol{x}))^\triangleright \wedge (\chi(\boldsymbol{x}))^\triangleright
\end{aligned}$$

The $^\triangleright$-operator commutes with all other syntactic elements of GPDL$_1^-$.

The decomposition of the concatenation operation depends on AGF$_1$ disallowing the use of source tuples in the rest of the formula. As already mentioned earlier, this is the exact position where the method fails for AGF, SGF and GF.

Definition 6.2.3. A global relation R of arity $(i;j)$ is *safe* for AGF_1-bisimulation iff for all structures $\mathfrak{A}$ and $\mathfrak{B}$, any AGF_1-bisimulation $I : \mathfrak{A} \sim_1 \mathfrak{B}$ is an AGF_1-bisimulation $I : (\mathfrak{A}, R^{\mathfrak{A}}) \sim_1 (\mathfrak{B}, R^{\mathfrak{B}})$, too.

By choice of construction, GPDL_1^- was matched to AGF_1, and we obtain the following unsurprising statement about GPDL_1^-.

Proposition 6.2.4. *Every* GPDL_1^- *program is safe for* AGF_1 *bisimulation.*

Proof. Atomic actions are safe by definition. Union and concatenation retain safety by chasing one, resp. two edges. Projection is safe, since safe relations are closed under projections. For the test operation, let $\varphi(\boldsymbol{x})$ be an AGF_1 formula and consider the program $\varphi?$. If $\boldsymbol{x} \sim_1 \boldsymbol{y}$ then $\varphi(\boldsymbol{x}) \iff \varphi(\boldsymbol{y})$, so, if possible, following the φ edges from $\boldsymbol{x}$ resp. $\boldsymbol{y}$ leads to the bisimilar tuples, namely $\boldsymbol{x}$ and $\boldsymbol{y}$ again. □

Definition 6.2.5. A formula $\varphi(X, \boldsymbol{x})$ is *completely additive* in X if it distributes over arbitrary unions, that is for all structures $\mathfrak{A}$ and all families $(X_i)_{i\in I}$ of sets of tuples of A:

$$\varphi[\bigcup_{i\in I} X_i](\boldsymbol{x})^{\mathfrak{A}} = \bigcup_{i\in I} \varphi[X_i](\boldsymbol{x})^{\mathfrak{A}}$$

The second-order variable X is an additional state predicate.

Lemma 6.2.6. $\varphi(X, \boldsymbol{x})$ *is completely additive in* X *iff for all* $\mathfrak{A}$ *and sets of tuples* P *in* A*,*

1. $P' \subseteq P$ *implies* $\varphi[P', \boldsymbol{a}] \Rightarrow \varphi[P, \boldsymbol{a}]$*,*
2. *for all* $\boldsymbol{a}$ *there is a* $\boldsymbol{p} \in P$ *such that* $\varphi[P, \boldsymbol{a}] \Rightarrow \varphi[\{\boldsymbol{p}\}, \boldsymbol{a}]$*.*

Corollary 6.2.7. *If* $\pi(\boldsymbol{x};\boldsymbol{y})$ *is an* X*-free* GPDL_1^- *program, then* $\langle\pi(\boldsymbol{x};\boldsymbol{y})\rangle X\boldsymbol{y}$ *is completely additive in* X*.*

Theorem 6.2.8. *An* AGF_1 *formula* $\varphi(X, \boldsymbol{x})$ *is completely additive iff it is equivalent to a formula of the form* $\langle\pi(\boldsymbol{x};\boldsymbol{y})\rangle X\boldsymbol{y}$*, where* π *is a* X*-free* GPDL_1^- *program.*

Proof. Let $\varphi(\boldsymbol{x})$ in vocabulary $\hat\tau$, $\tau = \tau_a \cup \tau_s$, $\hat\tau$ be $\tau \cup \{X\}$. Consider the following equivalence $(*)$:

$$\varphi(\boldsymbol{x}) \equiv \bigvee \big\{ \langle\pi(\boldsymbol{x};\boldsymbol{y})\rangle X\boldsymbol{y} \quad : \quad \pi \text{ is an } X\text{-free } \mathrm{GPDL}_1^- \text{ program}, \\ \langle\pi(\boldsymbol{x};\boldsymbol{y})\rangle X\boldsymbol{y} \models \varphi(X, \boldsymbol{x}) \big\}$$

If $(*)$ holds, we can conclude by compactness that $\varphi(\boldsymbol{x})$ is equivalent to a finite disjunction of formulae of the form $\langle\pi(\boldsymbol{x};\boldsymbol{y})\rangle X\boldsymbol{y}$ and repeatedly apply

$$\langle\pi(\boldsymbol{x};\boldsymbol{y})\rangle X\boldsymbol{y} \vee \langle\pi'(\boldsymbol{x};\boldsymbol{y})\rangle X\boldsymbol{y} \equiv \langle\pi(\boldsymbol{x};\boldsymbol{y}) \cup \pi(\boldsymbol{x};\boldsymbol{y})\rangle X\boldsymbol{y}$$

to obtain the desired result.

Towards $(*)$, suppose that $\mathfrak{A}, \boldsymbol{a} \models \varphi(\boldsymbol{x})$. Consider an ω-saturated elementary extension $\mathfrak{A}^\omega$ of $\mathfrak{A}$. Next, let $\mathfrak{A}^*$ be the 2-unravelling of $\mathfrak{A}^\omega$ starting at $\boldsymbol{a}$ with underlying tree $\mathfrak{T}$. Denote the elements induced by the root $\boldsymbol{a}^*$ (obtained as copy of $\boldsymbol{a}$). Then $\mathfrak{A}^\omega, \boldsymbol{a}$ and $\mathfrak{A}^*, \boldsymbol{a}^*$ are AGF_1 $\hat{\tau}$-bisimilar and hence $\mathfrak{A}^*, \boldsymbol{a}^* \models \varphi(\boldsymbol{x})$.

Since $\varphi(\boldsymbol{x})$ is completely additive in X, we can restrict the interpretation of X to a single tuple $\boldsymbol{a}^*_X$, such that $\varphi(\boldsymbol{x})$ still holds at $\boldsymbol{a}^*$. Call the obtained structure $\mathfrak{A}^-$.

Let $\boldsymbol{a}^*_0\boldsymbol{a}^*_1 \cdots \boldsymbol{a}^*_n$ be the tuples occurring on the unique path from $\boldsymbol{a}^* = \boldsymbol{a}^*_0$ to $\boldsymbol{a}^*_X \subseteq \boldsymbol{a}^*_n$ and let α_i be the guard atoms such that each $\alpha_i(\boldsymbol{x}'_i;\boldsymbol{x}_{i+1})$ connects (a subset of) $\boldsymbol{a}^*_i$ to $\boldsymbol{a}^*_{i+1}$, as given in the edge labels of $\mathfrak{T}$. For every $i \leq n$ define $\Phi_i(\boldsymbol{x}_i)$ as the AGF_1 τ-type of $\boldsymbol{a}^*_i$ in $\mathfrak{A}^-$. Assuming $\boldsymbol{x}'_i \subseteq \boldsymbol{x}_i$ for all $i \leq n$ as appropriate, define

$$\Gamma(\boldsymbol{x}_0, \dots, \boldsymbol{x}_n) = \bigcup_{i \leq n} \Phi_i(\boldsymbol{x}_i) \cup \bigcup_{i<n} \alpha_i(\boldsymbol{x}'_i, \boldsymbol{x}_{i+1}) \cup X(\boldsymbol{x}'_n).$$

Claim. $\Gamma(\boldsymbol{x}_0, \dots, \boldsymbol{x}_n) \models \varphi(\boldsymbol{a}^*_0)$.

Suppose $\mathfrak{B} \models \Gamma(\boldsymbol{b}_0, \dots, \boldsymbol{b}_n)$. First, take an ω-saturated elementary extension $\mathfrak{B}^\omega$ of $\mathfrak{B}$, and further, by 2-unravelling with respect to $\boldsymbol{b}_0$, obtain $\mathfrak{B}^*$ with root tuple $\boldsymbol{b}^*$. Since $\mathfrak{B}, \boldsymbol{b}_0 \sim_1 \mathfrak{B}^*, \boldsymbol{b}^*$, there are again tuples $\boldsymbol{b}^*_0, \dots, \boldsymbol{b}^*_n$ on a path in $\mathfrak{B}^*$ such that all pairs $\boldsymbol{b}^*_i$ and $\boldsymbol{b}_i$ are bisimilar. Hence $\mathfrak{B}^* \models \Gamma(\boldsymbol{b}^*_0, \dots, \boldsymbol{b}^*_n)$. Let $\mathfrak{B}^-$ be the structure we obtain from $\mathfrak{B}^*$ by restricting the interpretation of X to $\boldsymbol{b}^*_X$, the appropriate subtuple of $\boldsymbol{b}^*_n$. Note that $\mathfrak{B}^- \models \Gamma(\boldsymbol{b}^*_0, \dots, \boldsymbol{b}^*_n)$, since the Φ_i do not contain X.

We claim that $\mathfrak{A}^-, \boldsymbol{a}^*$ and $\mathfrak{B}^-, \boldsymbol{b}^*$ are bisimilar. Let I be the maximal AGF_1 τ-bisimulation between $\mathfrak{A}^*$ and $\mathfrak{B}^*$. As $\mathfrak{A}^*$ and $\mathfrak{B}^*$ were bisimilar to ω-saturated structures, and are hence AGF_1-saturated, we can conclude that the functions sending $\boldsymbol{a}^*_i$ to $\boldsymbol{b}^*_i$, resp., belong to I ($\boldsymbol{a}^*_i$ and $\boldsymbol{b}^*_i$ have the same AGF_1-type Φ_i). The desired $\hat{\tau}$-bisimulation I' is a subset of I, where each $\boldsymbol{a}^*_i$ is *only* connected to $\boldsymbol{b}^*_i$. and vice versa.

We verify that I' is a AGF_1 $\hat{\tau}$-bisimulation for $\mathfrak{A}^-, \boldsymbol{a}^*$ and $\mathfrak{B}^-, \boldsymbol{b}^*$.

- Since I' is a subset of I and I was maximal, we immediately get that all $f \in I'$ are partial τ-isomorphisms.

Adding X to the language destroys nothing, as X only holds at $\boldsymbol{a}^*_X$ and $\boldsymbol{b}^*_X$, respectively, and I' only connects $\boldsymbol{a}^*_n$ to $\boldsymbol{b}^*_n$.

- Suppose I' contains $f : \boldsymbol{c} \to \boldsymbol{d}$, and there is a $\boldsymbol{c}'$ and a $G \in \tau_a$ such that $\mathfrak{A}^- \models R(\boldsymbol{c}; \boldsymbol{c}')$ and $\boldsymbol{c}'$ is *not* one of the $\boldsymbol{a}^*_i$. Since also $f \in I$, there is a $\boldsymbol{d}'$ such that $\mathfrak{B}^- \models R(\boldsymbol{d}; \boldsymbol{d}')$ and $f' : \boldsymbol{c}' \to \boldsymbol{d}'$ is in I. If $\boldsymbol{d}'$ is not one of the $\boldsymbol{b}^*_i$, then $f' \in I'$. If however $\boldsymbol{d}'$ *is* one of the $\boldsymbol{b}^*_i$, then, as $\mathfrak{B}^-$ is still 2-unravelled with respect to τ, we can choose a $\boldsymbol{d}''$, $\mathfrak{B}^- \models R(\boldsymbol{d}; \boldsymbol{d}'')$, such that $\boldsymbol{d}''$ is τ-bisimilar to $\boldsymbol{d}'$, and hence to $\boldsymbol{c}'$, and $f'' : \boldsymbol{c}' \to \boldsymbol{d}''$ is in I'.

  Now suppose I' contains $f : \boldsymbol{c} \to \boldsymbol{d}$, and $\mathfrak{A}^- \models R(\boldsymbol{c}; \boldsymbol{a}^*_i)$ for some $R \in \tau_a$ and some $i \leq n$. Then, as $\mathfrak{A}^-$ is unravelled, $\boldsymbol{c} = \boldsymbol{a}^*_{i-1}$, and, by definition of I', $\boldsymbol{d} = \boldsymbol{b}^*_{i-1}$. Then $\boldsymbol{b}^*_i$ is the desired successor of $\boldsymbol{d}$ in $\mathfrak{B}^-$.

- The zag-clause is shown similarly.

As $\mathfrak{A}^- \models \varphi(\boldsymbol{a}^*_0)$ and $\boldsymbol{b}^*_0$ is AGF$_1$ $\hat{\tau}$-bisimilar to $\boldsymbol{a}^*_0$, $\mathfrak{B}^- \models \varphi(\boldsymbol{b}^*_0)$. By monotonicity of $\varphi(\boldsymbol{x})$ we can enlarge the interpretation of X as much as we like, while still maintaining the truth of φ at $\boldsymbol{b}^*_0$, so in particular $\mathfrak{B}^* \models \varphi(\boldsymbol{b}^*_0)$. As $\varphi(\boldsymbol{x})$ is invariant for bisimulation, we get $\mathfrak{B}^\omega \models \varphi(\boldsymbol{b}_0)$. We conclude that $\mathfrak{B} \models \varphi(\boldsymbol{b}_0)$ by the fact that $\varphi(\boldsymbol{x})$ is first-order.

Thus we have proved $\Gamma(\boldsymbol{x}_0, \dots, \boldsymbol{x}_n) \models \varphi(\boldsymbol{x}_0)$. By compactness, there is a finite subset Γ_0 of Γ that also implies φ. So there are X-free AGF$_1$ formulae $\varphi_0, \dots, \varphi_n$ such that

$$\langle \pi(\boldsymbol{x}_0; \boldsymbol{x}_n) \rangle X(\boldsymbol{x}'_n) \models \varphi(X, \boldsymbol{x}_0)$$

for

$$\pi = \varphi_0(\boldsymbol{x}_0)?; \alpha_0(\boldsymbol{x}'_0; \boldsymbol{x}_1); \cdots; \alpha_{n-1}(\boldsymbol{x}'_{n-1}; \boldsymbol{x}_n); \varphi_n(\boldsymbol{x}_n)?; \boldsymbol{x}'_n = \boldsymbol{x}'_n?$$

Note that π is an X-free GPDL$^-_1$ program and that $\mathfrak{A} \models \langle \pi(\boldsymbol{a}; \boldsymbol{x}_n) \rangle X(\boldsymbol{x}'_n)$.

We have therefore shown that whenever $\mathfrak{A} \models \varphi(\boldsymbol{a})$, there is an X-free GPDL$^-_1$ program π such that $\mathfrak{A} \models \langle \pi(\boldsymbol{a}; \boldsymbol{x}_n) \rangle X(\boldsymbol{x}'_n)$ and $\langle \pi(\boldsymbol{x}; \boldsymbol{x}_n) \rangle X(\boldsymbol{x}'_n) \models \varphi(\boldsymbol{x})$. This is precisely what is needed to prove $(*)$. □

Theorem 6.2.9. *A first-order definable global relation $\varphi(\boldsymbol{x}; \boldsymbol{y})$ that is safe for* AGF$_1$*-bisimulation is equivalent to a* GPDL$^-_1$ *program.*

Proof. Suppose $\varphi(\boldsymbol{x}; \boldsymbol{y})$ is a first-order τ-formula that is safe for AGF$_1$-bisimulation. Then $\psi(\boldsymbol{x}) = \exists \boldsymbol{y}(\varphi(\boldsymbol{x}; \boldsymbol{y}) \wedge X(\boldsymbol{y}))$ is invariant for AGF$_1$-bisimulation, for any $X \notin \tau$ of arity $|\boldsymbol{y}|$. By Theorem 6.1.10, ψ is equivalent to the translation of an AGF_1 formula $\chi(\boldsymbol{x})$. Since φ does not contain X, ψ, and hence χ, are obviously monotone and downward additive in X. By Lemma 6.2.6, $\chi(\boldsymbol{x})$ is completely additive in X. By Theorem 6.2.8, $\chi(\boldsymbol{x})$ is equivalent to $\langle\pi(\boldsymbol{x};\boldsymbol{y})\rangle X\boldsymbol{y}$ for some X-free GPDL_1^- program π.

Claim: $\pi(\boldsymbol{x};\boldsymbol{y}) \equiv \varphi(\boldsymbol{x};\boldsymbol{y})$.

Let $\mathfrak{A}$ be a $\tau \cup \{X\}$-structure and $\boldsymbol{a},\boldsymbol{b} \in \mathfrak{A}$ and let $\mathfrak{B}$ be obtained from $\mathfrak{A}$ by restricting X to $\{\boldsymbol{b}\}$. Then, since π is X-free, if $\mathfrak{A} \models \pi(\boldsymbol{a};\boldsymbol{b})$ then $\mathfrak{B} \models \pi(\boldsymbol{a};\boldsymbol{b})$, and consequently $\mathfrak{B} \models \langle\pi(\boldsymbol{a};\boldsymbol{y})\rangle X\boldsymbol{y}$. Then also $\mathfrak{B} \models \chi(\boldsymbol{a})$, so $\mathfrak{B} \models \psi(\boldsymbol{a})$. As X only holds at $\boldsymbol{b}$ in $\mathfrak{B}$, it follows that $\mathfrak{B} \models \varphi(\boldsymbol{a};\boldsymbol{b})$, and, since φ does not contain X, $\mathfrak{A} \models \varphi(\boldsymbol{a};\boldsymbol{b})$.

Similarly φ implies π. $\square$

Piecing together the puzzle we obtain the following result.

Theorem 6.2.10. *A first-order definable global relation $\varphi(\boldsymbol{x};\boldsymbol{y})$ is safe for AGF_1-bisimulation iff it is equivalent to a GPDL_1^- program.*

Following the lead from Section 3.6 we now give a characterisation in form of a suitable variant of guarded relational algebra. Unfortunately, as is the case with the corresponding bisimulations, the syntax for a relational algebra matching AGF_1 is more complex than for plain guarded GRA.

Syntax of AGRA_1.

1. U is a term of width 1.
2. Every $R \in \tau_s$ is a term of width $k = \mathrm{width}(R)$.
3. Every $G \in \tau_a$ is a term of width $(k;\ell) = \mathrm{width}(G)$.
4. If M, N are terms of width k then so are $M \setminus N$, $M \cap N$ and $M \cup N$.
5. If M, N are terms of width $(k;\ell)$ then so is $M \cup N$.
6. If M is a term of width k and $i, j \leq k$ then so is $\sigma_{i=j}(M)$.
7. If M is a term of width k, $1 \leq n_1, \dots, n_j \leq k$ and $\{1, \dots, k\} = \{n_1, \dots, n_j\}$, then $\pi_{n_1,\dots,n_j}(M)$ is a term of width j.
8. If M is a term of width $(k;\ell)$ and $1 \leq n_1, \dots, n_j \leq k$, $k < m_1, \dots, m_i \leq k+\ell$ and $\{1, \dots, k\} = \{n_1, \dots, n_j\}$, then $\pi_{n_1,\dots,n_j;m_1,\dots,m_i}(M)$ is a term of width $(j;i)$, or of width j if $i = 0$.
9. If M is a term of width k and N is a term of width ℓ then $M \times N$ is a term of width $k + \ell$.

10. If M is a term of width $(k;\ell)$ and N is a term of width $(\ell;m)$ then $M \circ N$ is a term of width $(k;m)$.

11. If M is a term of width k then $M?$ is a term of width $(k;k)$.

Given a $\tau_a \dot{\cup} \tau_s$-structure $\mathfrak{A}$, the semantics of AGRA_1 is defined along the lines of GRA. The cases 1.–9. are handled exactly analogous, with the intuitive generalisations from singular arities to arities of the form $(i;j)$.

Semantics of AGRA_1**.**

10. $(M \circ N)^{\mathfrak{A}} = \{(\boldsymbol{a};\boldsymbol{b}) \ : \ \text{exists } \boldsymbol{c} \in \mathfrak{A}, (\boldsymbol{a};\boldsymbol{c}) \in M^{\mathfrak{A}} \text{ and } (\boldsymbol{c},\boldsymbol{b}) \in N^{\mathfrak{A}}\}$.

11. $(M?)^{\mathfrak{A}} = \{(\boldsymbol{a};\boldsymbol{a}) \ : \ \boldsymbol{a} \in M^{\mathfrak{A}}\}$.

Proposition 6.2.11. *A global relation of arity k is definable by an* AGRA_1 *term iff it is definable by an* AGF_1 *formula.*

A global relation of arity $(k;\ell)$ is definable by an AGRA_1 *term iff it is definable by a* GPDL_1^- *program.*

Proof. Simultaneous inductive translation from AGRA_1-terms to GPDL_1^--programs and formulae.

1. $\psi_U(x) = (x = x)$.

2. $\psi_R(\boldsymbol{x}) = R(\boldsymbol{x})$.

3. $\psi_G(\boldsymbol{x};\boldsymbol{y}) = G(\boldsymbol{x},\boldsymbol{y})$.

4. $\psi_{M \cdot N}(\boldsymbol{x}) = \psi_M(\boldsymbol{x}) \cdot \psi_N(\boldsymbol{x}), \ \cdot \in \{\setminus, \cap, \cup\}$.

5. $\psi_{M \cup N}(\boldsymbol{x};\boldsymbol{y}) = \psi_M(\boldsymbol{x};\boldsymbol{y}) \cup \psi_N(\boldsymbol{x};\boldsymbol{y})$.

6. $\psi_{\sigma_{i=j}(M)}(\boldsymbol{x}) = \psi_M(\boldsymbol{x}) \wedge x_i = x_j$.

7. $\psi_{\pi_{n_1,\ldots,n_j}(M)}(\boldsymbol{x}')$ is obtained from $\psi_M(\boldsymbol{x})$ by (a) syntactically letting $x'_i = x_{n_i}$ if i is the first occurrence of n_i in $\boldsymbol{n}$, and leaving x'_i at all other positions, and (b) taking the conjunction with $\bigwedge\{x'_i = x'_{i'} \ : \ n_i = n_{i'}\}$.

8. $\psi_{\pi_{n_1,\ldots,n_j;m_1,\ldots,m_i}(M)}(\boldsymbol{x}';\boldsymbol{y}')$ is obtained from $\psi_M(\boldsymbol{x};\boldsymbol{y})$ by existentially quantifying over all y_k where $k \notin m_1,\ldots,m_i$, and then proceeding as in the previous case 7.

9. $\psi_{M \times N}(\boldsymbol{x},\boldsymbol{y}) = \psi_M(\boldsymbol{x}) \wedge \psi_N(\boldsymbol{y})$.

10. $\psi_{M \circ N}(\boldsymbol{x};\boldsymbol{y}) = \psi_M(\boldsymbol{x};\boldsymbol{z}); \psi_N(\boldsymbol{z};\boldsymbol{y})$.

11. $\psi_{M?}(\boldsymbol{x};\boldsymbol{x}) = \psi_M(\boldsymbol{x})$.

And vice versa from GPDL_1^--programs and $\mathrm{GPDL}_1^-$and AGF_1-formulae to AGRA_1-terms.

1. If $\psi(x_1,\dots,x_k) = (x_i = x_j)$ then $N_\psi = \sigma_{i=j}(U^k)$.
2. If $\psi(x_1,\dots,x_k) = R(x_{i_1},\dots,x_{i_r})$ then $N_\psi = \pi_{j_1,\dots,j_k}(R \times U)$ where $j_a = b$ for $i_b = a$ and $j_a = r+1$ otherwise.
3. If $\psi(x_1,\dots,x_k) = \alpha(x_1,\dots,x_k) \vee \beta(x_1,\dots,x_k)$ then $N_\psi = N_\alpha \vee N_\beta$.
4. If $\psi(x_1,\dots,x_k) = \neg\alpha(x_1,\dots,x_k)$ then $N_\psi = U^k \setminus N_\alpha$.
5. If $\psi(x_1,\dots,x_k) = \exists y_1,\dots,y_\ell(G(\boldsymbol{x},\boldsymbol{y}) \wedge \alpha(\boldsymbol{y}))$, then $N_\psi = \pi_{1,\dots,k;}(N_G \circ N_\alpha?)$.

1. If $\psi(\boldsymbol{x};\boldsymbol{y}) = G(\boldsymbol{x};\boldsymbol{y})$, then $N_\psi = G$.
2. If $\psi(\boldsymbol{x};\boldsymbol{y}) = \rho(\boldsymbol{x};\boldsymbol{y}) \vee \rho'(\boldsymbol{x};\boldsymbol{y})$, then $N_\psi = N_\alpha \cup N_\beta$.
3. If $\psi(\boldsymbol{x};\boldsymbol{z}) = \rho(\boldsymbol{x};\boldsymbol{y});\rho'(\boldsymbol{y}';\boldsymbol{z})$, $\boldsymbol{y}' \subseteq \boldsymbol{y}$, then $N_\psi = \pi_{n_1,\dots,n_k}(N_\rho) \circ N_{\rho'}$, where $n_1,\dots,n_k$ are such that $\boldsymbol{y}' = (y_{n_1},\dots,y_{n_k}) \subseteq \boldsymbol{y}$.
4. If $\psi(\boldsymbol{x};\boldsymbol{y}) = \rho(\boldsymbol{x};\boldsymbol{x})?$, i.e. syntactically $\boldsymbol{x} = \boldsymbol{y}$, then $N_\psi = N_\rho?$.

The proof that these translations are correct is again a rather technical, straightforward definition chasing. □

6.3 Functional Guards and Counting

We now take a look at the border of decidability for the counting extensions of action guarded fragments. The counting extension LC of a logic L adds counting quantifiers of the form $\exists^{\leq n}.\boldsymbol{x}$ and $\exists^{\geq n}.\boldsymbol{x}$. A weaker form of counting consists of functionality constraints for binary predicates. This only allows counting quantifiers to enforce the functionality of binary relations, $\forall x \exists^{\leq 1} y.Fxy$. Following is an overview on the results most relevant to this work.

Theorem 6.3.1 (Grädel). GFC^3 *is undecidable.*

Theorem 6.3.2 (Grädel, Rosen, Otto). C^2 *is decidable.*

Theorem 6.3.3 (Goncalves, Grädel). AGFCI *is decidable.*

On closer inspection, the undecidability proof in [24] for GFC uses GFC_1 formulae, plus one functionality statement, effectively proving a stronger claim than the one stated, namely that ${\mathrm{GFC}_1}^3$ is undecidable. This leaves SGFC and SGFC_1, with at least three variables, as open cases. We close this gap by showing the following theorem.

Theorem 6.3.4. ${\mathrm{SGFC}_1}^3$ *is undecidable.*

The method used for the undecidability results is again a reduction from domino systems to guarded formulae, cf. Section 3.7. The interesting part is to find a formula that axiomatises something that contains a homomorphic image of the $\mathbb{N} \times \mathbb{N}$-grid. This property is obviously far away from the generalised tree model property that we have become used to in the context of guarded logics. We require the following background on decidability and reductions, for more details see [9].

Let L and L′ be formula classes. A *conservative reduction* from L to L′ is a recursive function $r : \mathrm{L} \to \mathrm{L}'$ such that for all $\varphi \in \mathrm{L}$, φ is (finitely) satisfiable iff $r(\varphi)$ is (finitely) satisfiable, respectively. A logic L is a *conservative reduction class* if there is a conservative reduction from FO to L. We utilise the fact that a conservative reduction class inherits the undecidability of the (finite) satisfiability problem for FO.

Lemma 6.3.5 (Grädel). *Let* $\mathrm{L} \subseteq \mathrm{FO}$ *be a recursive class of sentences that is closed under conjunction with* GF^2*-sentences. To establish that* L *is a conservative reduction class, it suffices to exhibit a sentence* $\varphi_{\mathrm{grid}} \in \mathrm{L}$*, containing the binary predicates* F, G *(and possibly further auxiliary relations) such that*

(i) For all $r \in \mathbb{N}$ *there exists a* $k \in \mathbb{N}$ *such that some expansion* $\mathfrak{A}$ *of the* $k \cdot r$*-grid is a model of* φ_{grid}*.*

(ii) If $\mathfrak{A} \models \varphi_{\mathrm{grid}}$*, then there exists a homomorphism from the infinite grid to* $\mathfrak{A}$*.*

To encode a domino system $\mathcal{D}$, the formula $\varphi_{\mathrm{grid}} \in \mathrm{L}$ is combined with a formula $\varphi_{\mathcal{D}} \in \mathrm{GF}^2$ — therefore the requirement that L be closed under conjunction with GF^2 sentences. In our setting we use two ternary relations NW and SE, or “North-West” and “South-East”, instead of the horizontal and vertical successor relations, F and G. Additionally, for each domino type

d we have a unary predicate P_d that gives the set of positions tiled by d. Given a domino system $\mathcal{D} = (D, H, V)$, consisting of a finite set of dominoes D, and the horizontal and vertical compatibility relations $H, V \subseteq D \times D$, the following $\mathrm{SGF_1}^3$–-sentence $\psi_{\mathcal{D}}$ encodes the behaviour of $\mathcal{D}$, whence the above lemma can be equivalently stated with $\mathrm{SGF_1}^3-$ in place of GF^2.

$$\forall x \bigvee_{d\in D} P_d(x) \wedge \forall x \bigwedge_{d\neq d'} \neg\big(P_d(x) \wedge P_{d'}(x)\big) \wedge$$
$$(\forall xyz.NW(xyz))\Big[\bigvee_{(d,d')\in H} \big(P_d(x) \wedge P_{d'}(y)\big) \wedge \bigvee_{(d,d')\in V} \big(P_d(z) \wedge P_{d'}(x)\big)\Big]$$

For the formula φ_{grid} we only need counting quantifiers of the form $\exists^{=1}$, which is still very near to only using functionality statements. For a more intuitive reading we write $\begin{pmatrix} x & y \\ z & \end{pmatrix}$ for $NW(xyz)$ and $\begin{pmatrix} & x \\ y & z \end{pmatrix}$ for $SE(xyz)$ to give a semi-graphical representation of these atomic statements.

For each NW-triangle we require exactly one SE-triangle that completes the square, and one SE-triangle that continues the grid to the North.

$$\varphi_{NW} = \forall xyz.\begin{pmatrix} x & y \\ z & \end{pmatrix} \Big[\exists^{=1}u.\begin{pmatrix} & y \\ z & u \end{pmatrix} \wedge \exists^{=1}uv.\begin{pmatrix} & y \\ v & u \end{pmatrix} \wedge \exists^{=1}uv.\begin{pmatrix} & v \\ z & u \end{pmatrix}$$
$$\wedge\ \exists^{=1}uv.\begin{pmatrix} x & u \\ v & \end{pmatrix} \wedge \exists^{=1}u.\begin{pmatrix} & u \\ x & y \end{pmatrix} \wedge \exists^{=1}uv.\begin{pmatrix} & u \\ x & v \end{pmatrix}\Big]$$

And, symmetrically, the same with switched roles for NW and SE.

$$\varphi_{SE} = \forall xyz.\begin{pmatrix} & x \\ y & z \end{pmatrix} \Big[\exists^{=1}u.\begin{pmatrix} u & x \\ y & \end{pmatrix} \wedge \exists^{=1}uv.\begin{pmatrix} u & x \\ v & \end{pmatrix} \wedge \exists^{=1}uv.\begin{pmatrix} u & v \\ y & \end{pmatrix}$$
$$\wedge\ \exists^{=1}uv.\begin{pmatrix} & u \\ v & z \end{pmatrix} \wedge \exists^{=1}u.\begin{pmatrix} x & u \\ z & \end{pmatrix} \wedge \exists^{=1}uv.\begin{pmatrix} v & u \\ z & \end{pmatrix}\Big]$$

Let $\varphi_{\text{grid}} = \exists xyz.NW(xyz) \wedge \varphi_{NW} \wedge \varphi_{SE}$. The first conjunction merely has the role to exclude the empty structure as model for φ_{grid}. Now φ_{grid} axiomatises structures that contain at least one grid of NW and SE triangles as shown in Figure 6.2.

Proposition 6.3.6. *φ_{grid} axiomatises structures with disjoint infinite grids.*

The proof is straightforward. To recapitulate, both φ_{grid} and $\psi_{\mathcal{D}}$ can be written in $\mathrm{SGFC_1}^3-$ with NW and SE as guard atoms, or action predicates, and state predicates P_d. Therefore $\mathrm{SGFC_1}^3-$ is undecidable.

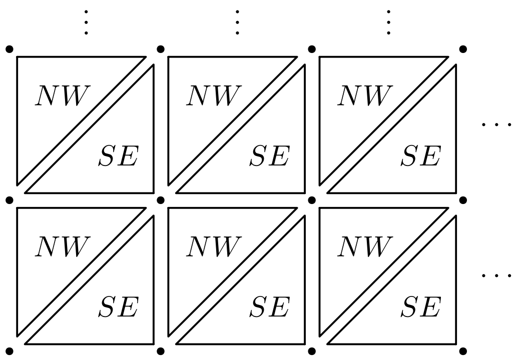

Figure 6.2: The grid defined by NW and SE.

6.4 Transitive Guards

We now take a look at guarded fragments with transitive relations. The base case is to consider transitivity for binary relations; they can be treated like edge relations in graphs. By L+TRS we denote the extension of L by transitive binary predicates, i.e. binary predicates that are required to be interpreted by transitive relations. The restricted variant L+TG, for *transitive guards*, allows transitive relations only at guard positions.

Theorem 6.4.1 (Grädel). $\mathrm{GF}_1{}^3$+TRS *is undecidable.*

Theorem 6.4.2 (Ganzinger et al.). GF^2+TRS without equality *is undecidable.*

The proofs shows that transitivity statements can be used to enforce functionality of a binary predicate. This is strong enough to produce undecidability as in the proof of Theorem 6.3.1. On the other hand we have the following recent theorem from [56].

Theorem 6.4.3 (Szwast, Tendera). GF+TG *is decidable.*

We now give an alternative, more straightforward proof of Theorem 6.4.3 that makes the tree-model property of GF+TG explicit. One part of the given construction takes arbitrary sized sets that are guarded by a transitive relation and turns them into "equivalent" finite sets. This method necessarily has one caveat.

Example. Consider the structure $\mathfrak{A} = (\mathbb{N}, T)$ with one transitive binary relation T, defined as the usual $<$-order on $\mathbb{N}$. This structure satisfies the

GF+TG formula

$$\psi = (\forall xy.T(xy))(x \neq y \wedge \neg T(yx) \wedge (\exists z.Tyz)).$$

Any *finite* candidate $\mathfrak{A}'$ for a structure that is GF+TG-equivalent to $\mathfrak{A}$ will have to consist of circles of T-edges in order to satisfy the final clause of ψ. However the transitive closure of a circle includes the reflexive closure, so the first clause of ψ will not hold in $\mathfrak{A}'$.

To circumvent this issue we assume transitivity to include reflexivity; in this sense a relation T is transitive iff it satisfies the following axiom.

$$\forall xyz((Txy \wedge Tyz) \rightarrow Txz) \wedge \forall x Txx$$

Define $\tau_R = \{R_1, \dots, R_{n_r}, =\}$ to be a set of relation symbols, and $\tau_{\mathrm{trs}} = \{T_1, \dots, T_{n_t}\}$ a set of transitive binary action predicates. In the following we consider vocabularies of the form $\tau = \tau_R \dot{\cup} \tau_{\mathrm{trs}}$.

Definition 6.4.4. Let $\mathfrak{B}$ be a τ-structure. A set $X \subseteq B$ is *transitive-guarded* if there is a $T \in \tau_{\mathrm{trs}}$ such that the Gaifman graph of the $\{T\}$-reduct of the substructure induced by X is connected.

Note that transitive-guarded sets can be arbitrarily large. To state that a singleton or a pair of elements b or (b_1, b_2) is a guarded tuple in the usual sense with guard atom from τ_{trs} we will write τ_{trs}-guarded. The equality "=" is explicitly put into τ_R; the intention is that τ_R-guarded now refers to guardedness in the usual sense, with all singletons guarded by $x = x$, whereas τ_{trs}-guarded strictly requires a τ_{trs}-guard.

Without explicitly using the notion of a GF+TG-bisimulation we use a special construction to build tree-like models for satisfiable GF+TG formulae. The method is a blending of the unravelling technique for modal and guarded logics, with some ideas from the canonisation for width-two guarded bisimulation. The main observation is that we can perform a kind of guarded unravelling of a GF+TG model with an upper bound on the size of transitive-guarded sets. However this method is always relative to a given formula, i.e. it does not give rise to, or employ, a general notion of GF+TG-bisimulation.

Definition 6.4.5. Consider a sentence $\psi \in$ GF+TG that is satisfiable, and let $\mathfrak{B}$ be a model of ψ. For $\boldsymbol{b} \in \mathfrak{B}$ and $\rho \subseteq \tau$ the *type* of $\boldsymbol{b}$ is defined as

$$\mathrm{tp}_{\psi,\mathfrak{B},\rho}(\boldsymbol{b}) = \{\psi'(\boldsymbol{x}) \in \mathrm{cl}(\psi) \ : \ \mathfrak{B} \models \psi'(\boldsymbol{b}'), \boldsymbol{b}' \subseteq \boldsymbol{b}\} \cup \mathrm{atp}_{\mathfrak{B},\rho}(\boldsymbol{b}).$$

For brevity, we make it a habit to omit parameters that are clear from the context. Since $\mathrm{cl}(\psi)$ is finite, there is a finite upper bound on the number of distinct types that is independent of $\mathfrak{B}$ — and independent of $\rho \subseteq \tau$, because τ is finite and fixed.

Definition 6.4.6. Let $C \subseteq B$ be a maximal transitive-guarded set and let $T \in \tau_{\mathrm{trs}}$ be a transitive guard relation that verifies that C is indeed transitive guarded in $\mathfrak{B}$. For every $b \in C$ the *augmented type* $\mathrm{aut}(b)$ is defined as

$$\begin{aligned}\mathrm{aut}_{\psi,\mathfrak{B},C,T}(b) \quad = \quad & \mathrm{tp}_{\psi,\mathfrak{B},\tau_R\cup\{T\}}(b) \\ & \cup\{\mathrm{tp}_{\psi,\mathfrak{B},\tau_R\cup\{T\}}(a,b) \ : \ \{ab\} \subseteq C, \mathfrak{B} \models T(a,b)\} \\ & \cup\{\mathrm{tp}_{\psi,\mathfrak{B},\tau_R\cup\{T\}}(b,a) \ : \ \{ab\} \subseteq C, \mathfrak{B} \models T(b,a)\}.\end{aligned}$$

Note that the second and third sets in the union are not necessarily disjoint. Although the intention is in a fashion to disentangle transitive guarded sets, there is no problem with keeping back and forth edges, i.e. where $\mathfrak{B} \models T(a,b) \wedge T(b,a)$. The aim is merely to prevent a pair of elements being guarded by two *different* predicates, if not both of them belong to τ_R.

The size of $\mathrm{aut}(C) = \{\mathrm{aut}(b) \ : \ b \in C\}$ depends on all parameters ψ, $\mathfrak{B}$, C and T. However, for a finite number of types as seen above, there is an upper bound $N_{\mathrm{aut}(\psi)}$ for the number of distinct augmented types, that depends only on ψ.

Two elements a and a' with $\mathrm{aut}(a) = u$ and $\mathrm{aut}(a') = u'$ are *compatible*, if $\mathrm{tp}_{\psi,\mathfrak{B},\tau_R\cup\{T\}}(a,a') \in u$ and $\mathrm{tp}_{\psi,\mathfrak{B},\tau_R\cup\{T\}}(a',a) \in u'$, or vice versa.

We now build a transitive guarded $\tau_R \cup \{T\}$-structure $\mathfrak{B}[C,T]$, that depends on ψ, T, C, and an optional parameter $c \in C$. The structure $\mathfrak{B}[C,T]$ has universe B_C, and is built to adhere to the following properties.

(i) The size of $\mathfrak{B}[C,T]$ is bounded by $\mathcal{O}(|\mathrm{aut}(C)|)$.

(ii) Each element $a \in B_C$ is associated with an element $b = \pi(a)$ in $\mathfrak{B}$ (and c is in the image of π).

(iii) $\mathfrak{B}[C,T]$ behaves just like $\mathfrak{B}|_C$ with respect to T-guarded moves in the semantic game for ψ.

The first two properties will directly follow from the construction algorithm. The third property will be established in the proof of the following theorem that uses the construction to build models of bounded tree width for GF+TG formulae. We also assume that each $\mathfrak{B}[C,T]$ has normalised universe $\{1,\ldots,|\mathfrak{B}[C,T]|\}$.

First, for each $u \in \mathrm{aut}(C)$ choose an element $b_u \in C$ that realises u. If a parameter $c \in C$ is given, take that as representative for the type $\mathrm{aut}(c)$. Start the construction of $\mathfrak{B}[C,T]$ with a fresh set of elements $A_C = \{a_u : u \in \mathrm{aut}(C)\}$ and define $\pi(a_u) = b_u$, $u \in \mathrm{aut}(C)$. Until the last step of the construction, $\mathfrak{B}[C,T]$ will be a graph consisting of the universe A_C and some T-edges, starting with $T^{\mathfrak{B}[C,T]} = \emptyset$.

Inductively continue as follows. For each $a \in A_C$, and each binary type $t \in \mathrm{aut}(\pi(a))$ where a does *not* have a T-successor, or in fact a T-predecessor, respectively, depending on t; we make explicit the case of a successor, the predecessor case is handled similarly, a' in $\mathfrak{B}[C,T]$ such that $\mathrm{tp}(\pi(a), \pi(a')) = t$, check whether there is an element $a' \in A_C$ that is distinct from a and its T-successors, such that $\mathrm{tp}(\pi(a), \pi(a')) = t$. If such an a' exists, add a T-edge from a and a' in $\mathfrak{B}[C,T]$. If no such a' exists, introduce a fresh element a', make a' a T-successor of a and define $\pi(a') = b'$ for some element b' where $\mathrm{tp}(\pi(a), b_t) = t$.

This construction requires at most $3 \cdot |\,\mathrm{aut}(C)|$ steps. More precisely, for each $u \in \mathrm{aut}(C)$ there are at most 3 elements a_1, a_2, a_3 in $\mathfrak{B}[C,T]$ with $\mathrm{aut}(\pi(a_i)) = u$. For suppose that there is an element $a \in \mathfrak{B}[C,T]$ that requires, but does not yet have, a successor of type u. If a is distinct to all of the a_i, the construction can and will make one of them a successor of a. If a is one of the a_i, w.l.o.g. $a = a_1$, then at most one of a_2 and a_3 is a predecessor of a, and the other can and will be made a successor of a to satisfy its requirement. Therefore the construction need never add a fourth element of type u.

At this point, $T^{\mathfrak{B}[C,T]}$ is, in general, not transitive. We need to show that taking the transitive closure will only connect compatible elements. Consider a pair (a_1, a_n) that are the end-points of a T-path $(a_1, \ldots, a_n)$, where $n > 2$, i.e. two elements that will be connected by the transitive closure of $T^{\mathfrak{B}[C,T]}$. Let $b_1 = \pi(a_1)$ and successively choose the b_i for $i \geq 2$ as T-successor of b_{i-1} with $\mathrm{aut}(b_i) = \mathrm{aut}(\pi(a_i))$. Then, by transitivity of T in $\mathfrak{B}$, there is a T-edge from b_1 to b_n, so $\mathrm{aut}(b_1) = \mathrm{aut}(\pi(a_i))$ and $\mathrm{aut}(b_n) = \mathrm{aut}(\pi(a_n))$ are compatible.

To complete the construction of $\mathfrak{B}[C,T]$, replace $T^{\mathfrak{B}[C,T]}$ by its transitive closure. Then, for each pair (a, a') that is T-guarded, and hence compatible, update $\mathfrak{B}[C,T]$ to reflect the atomic τ_R-type of (a, a') as given by $\mathrm{aut}(\pi(a))$ — and, equally, by $\mathrm{aut}(\pi(a'))$.

Note that although the construction of the $\mathfrak{B}[C,T]$ has an air of canonisation as considered earlier, it is not the real thing. Here the types of the individual elements are only with respect to the (finite!) set of subformulae of ψ and atomic formulae, not any — however defined — notion of bisimulation, that could, on infinite structures, produce an infinite number of distinct types.

We are now in a position to extend the definitions of $\mathfrak{K}(\mathfrak{B})$ and $\mathfrak{T}(\mathfrak{B})$ as given in Section 4.4 to structures with transitive guards. This time the size of the structures at each node is bounded by $m = \max(\mathrm{width}(\tau), 3 \cdot N_{\mathrm{aut}(\psi)})$ and S contains all τ_R-guarded τ_R-structures and, for every $T \in \tau_{\mathrm{trs}}$, all transitive-guarded $\tau_R \cup \{T\}$-structures of size at most m. Let $G(\mathfrak{B})$ contain all pairs (g, R) where $g \subseteq B$ is a τ_R-guarded list guarded by R, and all pairs (t, T) where $t \subseteq B$ is a maximal transitive-guarded list via $T \in \tau_{\mathrm{trs}}$. Again we assume the $\mathfrak{A} \in S$ to have the normalised universe $\{1, \dots, |\mathfrak{A}|\}$, and let F be the set of all partial 1-1 maps ρ from $\{1, \dots, m\}$ to $\{1, \dots, m\}$ and $\tilde{\tau} = \{P_{\mathfrak{A}} : \mathfrak{A} \in S\} \cup \{E_\rho : \rho \in F\}$. Towards $\mathfrak{K}(\mathfrak{B})$, define the E_ρ and $P_{\mathfrak{A}}$ according to the following rules.

For $Q \in \tau_R$, put $(\boldsymbol{b}, Q) \in G(\mathfrak{B})$ into the $P_{\mathfrak{A}}$ where $\mathfrak{B}|_{\boldsymbol{b}}$ is τ_R-isomorphic to $\mathfrak{A}$ via $b_i \mapsto i$.

For $T \in \tau_{\mathrm{trs}}$, put $(\boldsymbol{b}, T) \in G(\mathfrak{B})$ into $P_{\mathfrak{B}[\boldsymbol{b},T]}$.

For $Q, Q' \in \tau_R$, there is an E_ρ-edge from $(\boldsymbol{b}, Q)$ to $(\boldsymbol{c}, Q')$ iff $c_i = b_{\rho(i)}$ for all $i \in \mathrm{dom}(\rho)$.

For $T, T' \in \tau_{\mathrm{trs}}$, there is an E_ρ-edge from $(\boldsymbol{b}, T)$ to $(\boldsymbol{c}, T')$ iff there is $a \in \boldsymbol{b} \cap \boldsymbol{c}$ with $\mathrm{tp}_{\psi,\mathfrak{B},\tau_R}(a) = \mathrm{tp}_{\psi,\mathfrak{B},\tau_R}(\pi(c_i)) = \mathrm{tp}_{\psi,\mathfrak{B},\tau_R}(\pi(b_{\rho(i)}))$ for $\mathrm{dom}(\rho) = \{i\}$.

For $Q \in \tau_R$ and $T \in \tau_{\mathrm{trs}}$, there is an E_ρ-edge from $(\boldsymbol{b}, Q)$ to $(\boldsymbol{c}, T)$ iff $(b_{\rho(i)})_{i \in \mathrm{dom}(\rho)}$ is T-guarded in $\mathfrak{B}$ and has the same $(\psi, \mathfrak{B}, \tau_R)$-type as the tuple $(\pi(c_i))_{i \in \mathrm{dom}(\rho)}$, and vice versa from $(\boldsymbol{b}, T)$ to $(\boldsymbol{c}, Q)$ iff $(c_i)_{i \in \mathrm{dom}(\rho)}$ is T-guarded in $\mathfrak{B}$ and has the same $(\psi, \mathfrak{B}, \tau_R)$-type as $(\pi(b_{\rho(i)}))_{i \in \mathrm{dom}(\rho)}$.

Definition 6.4.7. The transition system $\mathfrak{K}(\mathfrak{B})$ is the $\tilde{\tau}$-structure

$$\mathfrak{K}(\mathfrak{B}) \quad = \quad \big(G(\mathfrak{B}), (E_\rho)_{\rho \in F}, (P_{\mathfrak{A}})_{\mathfrak{A} \in S}\big).$$

In this setup, if $v \in G(\mathfrak{B})$ we denote by $\mathfrak{A}_v$ the unique $\mathfrak{A} \in S$ for which $P_{\mathfrak{A}}$ holds at v. The unravelling $\mathfrak{T}(\mathfrak{B})$ is obtained from $\mathfrak{K}(\mathfrak{B})$ as in Definition 4.4.3. These trees satisfy the consistency conditions (a) and (b) according to Definition 4.5.1, which suffices to build $\mathfrak{D}(\mathfrak{T}(\mathfrak{B}))$ as per Definition 4.5.3.

Theorem 6.4.8. GF+TG *has the generalised tree model property.*

Proof. Let ψ be a satisfiable GF+TG sentence in GF+TG$_0$ normal form and let $\mathfrak{B}$ be a τ-structure. We show that $\mathfrak{B} \models \psi$ iff $\mathfrak{B}^* = \mathfrak{D}(\mathfrak{T}(\mathfrak{B})) \models \psi$.

Claim. For every $\varphi(\boldsymbol{x}) \in \mathrm{cl}(\psi)$ and $v = v_0\rho_1 v_1 \cdots v_n \in \mathfrak{T}(\mathfrak{B})$, $v_n = (\boldsymbol{b}, Q)$,

1. if $Q \in \tau_R$, then $\mathfrak{B} \models \varphi(\boldsymbol{b}')$ iff $\mathfrak{B}^* \models \varphi(\boldsymbol{b}^*)$
   for every $\boldsymbol{b}' = (b_{i_1}, \dots, b_{i_j}) \subseteq \boldsymbol{b}$ and $\boldsymbol{b}^* = ([v, i_1], \dots, [v, i_j])$,

2. if $Q = T \in \tau_{\mathrm{trs}}$, then $\mathfrak{B} \models \varphi(\boldsymbol{b}')$ iff $\mathfrak{B}^* \models \varphi(\boldsymbol{b}^*)$
   for every τ_{trs}-guarded $\boldsymbol{b}' \subseteq \boldsymbol{b}$ and τ_{trs}-guarded $\boldsymbol{a}' \subseteq \mathfrak{A}_{v_n} \cong \mathfrak{B}[\boldsymbol{b}, T]$ with $\mathrm{tp}_{\psi,\mathfrak{B},\tau_R \cup \{T\}}(\pi(\boldsymbol{a}')) = \mathrm{tp}_{\psi,\mathfrak{B},\tau_R\cup\{T\}}(\boldsymbol{b}')$ and $b_i^* = [v, a_i']$, $1 \le i \le |\boldsymbol{a}'|$, and φ not a $\tau_{\mathrm{trs}} \setminus \{T\}$-literal.

If φ is a literal, the claim follows from the construction. In case 1 we can track the atomic τ_R-type of $\boldsymbol{b}$ in $\mathfrak{B}$ to $\mathfrak{A}_{v_n}$, i.e. the function $(1, \dots, j) \mapsto (b_1, \dots, b_j)$ is a τ_R-isomorphism. Going further from $\mathfrak{K}(\mathfrak{B})$ to $\mathfrak{T}(\mathfrak{B})$ does not change the atomic τ_R-type, just as the final step from $\mathfrak{T}(\mathfrak{B})$ to $\mathfrak{B}^*$, which maps each ℓ in $\mathfrak{A}_{v_n}$ to $[v, \ell]$ in $\mathfrak{B}^*$. In case 2 it is explicitly required that $\mathrm{tp}(\pi(\boldsymbol{a}')) = \mathrm{tp}(\boldsymbol{b}')$, so in particular the atomic $\tau_R \cup \{T\}$-types of $\boldsymbol{a}'$ in $\mathfrak{A}_{v_n}$ and $\boldsymbol{b}'$ in $\mathfrak{B}$ coincide. The further steps are as in case 1, this time with $\boldsymbol{b}^*$ defined in terms of $\boldsymbol{a}'$ at v.

If φ is of the form $\vartheta \vee \zeta$, $\vartheta \wedge \zeta$ or $\neg\vartheta$, the claim follows immediately by induction hypothesis for ϑ and ζ.

If φ is of the form $(\exists^{\neq} \boldsymbol{y}.\rho(\boldsymbol{x}, \boldsymbol{y}) \wedge \alpha(\boldsymbol{y}))\vartheta(\boldsymbol{y})$, and α is a τ_R-atom, we are actually in the same situation as in Lemma 4.5.4, if slightly in disguise. The same arguments basically go through here too, e.g. suppose that there is a guarded list $\boldsymbol{a}' \subseteq \mathfrak{B}$ for which $\mathfrak{B} \models \rho(\boldsymbol{b}', \boldsymbol{a}') \wedge \alpha(\boldsymbol{a}') \wedge \vartheta(\boldsymbol{a}')$. Then, by the properties of an unravelling, there is a node $v_{n+1} = (\boldsymbol{a}, R)$ in $\mathfrak{B}^*$, where R is the guard relation from α, such that $\boldsymbol{a} \supseteq \boldsymbol{a}'$, and we can choose $\boldsymbol{a}^*$ as a tuple corresponding to $\boldsymbol{a}'$ such that $\mathfrak{B}^* \models \rho(\boldsymbol{b}^*, \boldsymbol{a}^*)$, and by induction $\mathfrak{B}^* \models \alpha(\boldsymbol{a}^*) \wedge \vartheta(\boldsymbol{a}^*)$.

Similarly if φ is of the form $(\exists^{\neq} \boldsymbol{x}.\alpha(\boldsymbol{x}))\vartheta(\boldsymbol{x})$, and α is a τ_R-atom, we get the claim by induction hypothesis for $\alpha(\boldsymbol{x})$ and $\vartheta(\boldsymbol{x})$, case 1, considering all guarded lists simultaneously. And if α is a τ_{trs}-atom, then ϑ is a positive Boolean combination of τ_R-literals and quantified formulae, hence induction for $\vartheta(\boldsymbol{x})$ and $\alpha(\boldsymbol{x})$, case 2, yields the claim for φ.

If φ is of the form $(\exists y.T(x, y))\vartheta(x, y)$, $T \in \tau_{\mathrm{trs}}$, consider any maximal transitive-guarded sets $C \subseteq \mathfrak{B}$ and $C^* \subseteq \mathfrak{B}^*$, both transitive-guarded by T. Suppose there are elements $a \in C$ and $a^* \in C^*$, $a^* = [v, i]$ for the v where C^* lives and some i, and $\mathrm{tp}_{\psi,\mathfrak{B},\tau_R\cup\{T\}}(a) = \mathrm{tp}_{\psi,\mathfrak{B},\tau_R\cup\{T\}}(\pi(i))$, for the π used in constructing $\mathfrak{B}[C, T]$. Then, according to the construction of $\mathfrak{B}[C, T]$, for every T-successor $b \in C$ of a, there is a T-successor $b^* \in C^*$ of a^* such that for $b^* = [v, j]$ we have $\mathrm{tp}_{\psi,\mathfrak{B},\tau_R \cup \{T\}}(a, b) = \mathrm{tp}_{\psi,\mathfrak{B},\tau_R \cup \{T\}}(\pi(i), \pi(j))$, and vice versa. For every such pair b, b^*, induction via case 2 of the claim yields $\mathfrak{B} \models \vartheta(a, b)$ iff $\mathfrak{B}^* \models \vartheta(a^*, b^*)$. Putting everything together, there is a T-edge starting from a that satisfies ϑ iff there is a T-edge from a^* that satisfies ϑ. With the preconditions of case 2 this shows the claim.

The case of quantifying over T-predecessors is handled likewise. □

Theorem 6.4.9 (Goncalves, Grädel). *Let* L *be a logic,* $\mathcal{C}$ *a class of structures such that*

(i) L *has the generalised tree model property on* $\mathcal{C}$*,*

(ii) L *can be equivalently translated into* GSO *on* $\mathcal{C}$*.*

Then $\mathrm{Sat}_{\mathcal{C}}(\mathrm{L})$*, the satisfiability problem for* L *on* $\mathcal{C}$*, is decidable.*

By Theorem 6.4.8, GF+TG has the generalised tree model property on the class $\mathcal{T}$ of all τ-structures where the τ_{trs}-predicates are interpreted by transitive relations. Clearly, on $\mathcal{T}$ GF+TG is syntactically a subset of GSO.

Corollary 6.4.10. GF+TG *is decidable.*

Chapter 7

Canonical Structures

For many applications it is necessary to concern oneself with the question of how amiable towards canonisation a given logic, or rather its corresponding bisimulation, is. The term "canonisation" may seem, and indeed is, a little shaky with respect to a precise mathematical definition. Given a notion of bisimulation equivalence, the idea is to find a canonical, in the spirit of canonisation usually small or smallest possible, representative for every bisimulation equivalence class of structures.

An application of canonisation is in the realm of relational databases that are given as finite relational structures. The small size, and possible other structural properties of the canonical representatives, can lead to more efficient query evaluation; find the canonical companion to the given database and perform all subsequent queries on this — small — structure.

In descriptive complexity theory, canonisation can help to show capturing results, since any computation can be made bisimulation invariant by prepending an appropriate canonisation procedure [51]. Here the crucial feature of the representative structure is not its size, but the fact that it can be deterministically computed.

Note. Throughout this chapter all structures are assumed to be finite. Furthermore, the following complexity assessments are based on an arbitrary, but fixed, vocabulary, i.e. the size of the vocabulary, and, wherever appropriate, the maximal occuring arity of a relation symbol, are taken as constant.

7.1 Modal Canonisation

For modal bisimulation, factoring out modal bisimulation equivalence yields minimal canonical structures — in the canonical version every bisimulation type occurs only once. We use the Paige-Tarjan-algorithm as core of an efficient implementation of modal canonisation [53].

Theorem 7.1.1 (Paige,Tarjan). *Let $\mathfrak{G} = (V, E, (P_i)_{1\leq i\leq k})$ be a transition system where the $(P_i)_{1\leq i\leq k}$ partition V. The bisimulation classes of $\mathfrak{G}$ are computable in time $\mathcal{O}(|E|\log|V|)$.*

A graph in our usual format $\mathfrak{K} = (V, (E_a)_{a\in A}, (P_b)_{b\in B})$ can be made eligible as input to Theorem 7.1.1 by adding intermediate nodes to carry the edge labels, and initially partitioning the nodes according to their atomic type.

For every edge (u, v) with label a this produces a new node (u, a, v) that carries the (new) predicate P_a, and two unlabelled edges $(u,(u, a, v))$ and $((u, a, v), v)$. The size of the vocabulary is seen as constant, hence the complexity of this preprocessing is in the order of the size of the output, $|V|+|E|$ nodes and $2|E|$ edges. The combined complexity of preprocessing and the Paige–Tarjan procedure therefore is $\mathcal{O}(|E|+|V|+2|E|\log(|E|+|V|))$, which reduces to $\mathcal{O}(|V| + |E|\log|V|)$ for non-trivial inputs. Under the (mild) assumption that the input graphs are "not too sparse" in the sense that $|E| \geq c\cdot|V|$, for some fixed $c > 0$, we end up with the original $\mathcal{O}(|E|\log|V|)$.

Performing the Paige–Tarjan procedure yields a partitioning of the preprocessed input structure into the occuring bisimulation classes. The nodes that represent edges, and the nodes that are copies of original nodes, can never coexist inside of a bisimulation class, as the latter never carry labels P_a for $a \in A$.

The universe of the canonical companion $\mathfrak{K}^- = \mathfrak{K}/_\sim$ consists of the bisimulation classes $[u]$ for every class where u is a node of the original input, i.e. $V^- = \{[v] \ : \ v \in V\}$. Then let $[v] \in P_b^- \iff v \in P_b$. This construction is well defined. If $[v] = [w]$, then $\mathfrak{K}, v \sim \mathfrak{K}, w$ and $v \in P_b \iff w \in P_b$. Towards the edges, to decide whether $([v], [w]) \in E_a^-$ choose an arbitrary $v' \in [v]$ and check all E_a successors w' in $\mathfrak{K}$ if they belong to $[w]$. This step requires at most one iteration over all nodes, and one over all edges, respectively, and thus does not increase the total complexity.

It can be easily verified that $Z = \{(v, [v]) \ : \ v \in V\}$ is a bisimulation between $\mathfrak{K}$ and $\mathfrak{K}^-$. For every $v \in V$, compatibility for the unary predicates P_b with $[v] \in V^-$ is ensured by the definition. For the back condition, consider a pair $(v, [v]) \in Z$ and some $[w]$ such that $([v], [w]) \in E_a^-$. It is not necessarily the case that $(v, w) \in E_a$, but since $([v], [w]) \in E_a^-$, there are $v' \in [v]$ and $w' \in [w]$ such that $(v', w') \in E_a$. Now $v' \sim v$, so there has to be an E_a-successor u of v such that $w' \sim u$, or $u \in [w] = [w']$ is the desired E_a-successor such that $(u, [w]) \in Z$. The forth condition is checked similarly.

Algorithm 1 Modal Canonisation

```
Input: Graph $\mathfrak{K}$
  let $B' = A \cup B$, $A' = \emptyset$
  initialise $\mathfrak{K}'$ with universe $V$ and no edges
  for all $a \in A$ and $(u, v) \in E_a^{\mathfrak{K}}$ do
    add new node $(u, a, v)$ to $V'$ and let $(u, a, v) \in P_a^{\mathfrak{K}'}$
    add new edges $(u, (u, a, v))$ and $((u, a, v), v)$ to $E'$
  end for
  invoke Theorem 7.1.1 for $\mathfrak{K}'$
  initialise $\mathfrak{K}^-$ with universe $V^- = \{[v] : v \in V\} \subset \mathfrak{K}'^-$
  for all $a \in A$ and $[v] \in V^-$ do
    for all $a$-successors $w$ of $v$ in $\mathfrak{K}$ do
      add new edge $([v], [w])$ to $E_a^{\mathfrak{K}^-}$
    end for
  end for
Output: Canonical $\mathfrak{K}^-$
```

Beating around the bush of an exact formal definition, we nail down canonisation as follows. Given a notion of bisimulation equivalence, canonisation is an efficient, in this context: polynomial time, procedure that maps its input to a smallest possible bisimilar output structure that depends only on the bisimulation class of the input.

The construction given above effectively shows that modal canonisation can be performed efficiently. Even more, it actually retains all occurring bisimulation types, not just the ones pertaining to some connected component. Therefore it respects total bisimulations between structures, rather than the weaker form that compares the subgraphs reachable from some pair of distinguished nodes.

Proposition 7.1.2. *Modal bisimulation allows canonisation.*

The canonisation procedure for modal bisimulation is a good example of how well behaved modal bisimulation is with respect to manipulations of structures. The problems of building a finite model from a tableau in Chapter 5 showed that this property unfortunately does not always carry over into the guarded world.

On the other hand, the transformations between structures and graphs given in Chapter 4 suggest something like forming $\mathfrak{D}(\mathfrak{K}(\mathfrak{B})^-)$ to find a canonical structure wrt. guarded bisimulation, for relational structures $\mathfrak{B}$. Unfortunately this simple approach does not work, and has so far appeared to be rather stubborn towards regaining a canonisation procedure — at least for the case of full guarded bisimulation.

As first counter example, let $\mathfrak{B}$ be the graph that consists of two nodes $1, 2$, connected by one undirected edge. Then $\mathfrak{K}(\mathfrak{B})$ has four nodes, corresponding to the guarded lists (1), (2), $(1, 2)$ and $(2, 1)$. The two singleton lists have the same bisimulation type in $\mathfrak{K}(\mathfrak{B})$, so they will be identified in $\mathfrak{K}(\mathfrak{B})^-$; the same is true for $(1, 2)$ and $(2, 1)$. Going further, the universe of $\mathfrak{D}(\mathfrak{K}(\mathfrak{B})^-)$ only has a single element, which can not be guarded bisimilar to $\mathfrak{B}$.

With a little additional work, a solution can be found for guarded bisimulation on graphs, and for AGF_1-bisimulation on arbitrary structures.

7.2 Width-2 Guarded Canonisation

To obtain a deterministic algorithm, this is the point where we require an order on the edge types that were introduced for the atomic expansions in Section 4.1, and an order on GF-types of width 1. Following the lead from Section 2.2, we assume that τ is ordered, whereby the edge types — sets of literals — are ordered according to their val-values. For GF-types of width 1 that can be realised on finite graphs, we use the following weaker version of a theorem shown in [16, 19].

Theorem 7.2.1 (Dawar). *For every ordered finite vocabulary of graphs τ there is an* FO + LFP*-formula that defines an order on the canonical version of the atomic expansion of all τ-graphs $\mathfrak{K}$, i.e. an order on $(\mathfrak{K}^+)^-$.*

Every GF-type of width 1 in $\mathfrak{K}$ has a corresponding[1] bisimulation type in $\mathfrak{K}^+$, each of which is realised exactly once in $(\mathfrak{K}^+)^-$. In other words, Theorem 7.2.1 implies that an order on the width-1 guarded bisimulation types

[1]Since finite structures are ω-saturated, two nodes are guarded bisimilar iff they are GF-equivalent

occurring in $\mathfrak{K}$ can be found in polynomial time. That is, as long as computation of $\mathfrak{K}^+$ and $(\mathfrak{K}^+)^-$ can be achieved in polynomial time, the latter which was shown in the previous section.

Creating $\mathfrak{K}^+$ from $\mathfrak{K}$ requires an iteration over all pairs of nodes (v, w) with accumulation of the respective edge type. The val-value of an edge type can be found by replacing each literal by its respective val-value, and then sorting the resulting vector of integers.

A canonisation algorithm whose result is merely required to be *one of the possible* smallest structures that is guarded bisimilar to the input, not necessarily the same for distinct, but guarded bisimilar, inputs, the fiddling around with orders is unnecessary. The previously mentioned case of canonisation for more efficient database queries is an obvious candidate where this weaker version will normally suffice. In the proof below we present the full deterministic construction.

Theorem 7.2.2. *Guarded bisimulation for width* 2 *allows canonisation.*

Proof. Let $\mathfrak{K} = (V, (E_a)_{a\in A}, (P_b)_{b\in B})$ be a finite graph and, for some suitable index set I, let $T_1 = \{p_i(x) \ : \ i \in I\}$ be the set of GF-types of width 1 that are realised in $\mathfrak{K}$. Let T_2 be the set of edge types of $\mathfrak{K}$, and let $\mathfrak{K}^+$ be the atomic expansion of $\mathfrak{K}$. Suppose that A and B are ordered, whence the discussion above shows that an order on T_1 and T_2 can be found in polynomial time. Further, assume that I is ordered such that $p_i < p_j$ iff $i < j$ for all $i, j \in I$.

For each $i \in I$ let $k_i \in \mathfrak{K}$ be an element that realises p_i. For every pair $i, j \in I$ let $\mathrm{types}(i, j) \subseteq T_2$ be the set of all edge types $t \in T_2$ where there are $a, b \in \mathfrak{K}$ such that a realises p_i, b realises p_j, and t is the edge type of (a, b). Using $\mathfrak{K}^+$, the sets $\mathrm{types}(i, j)$ can be found by iterating over all pairs of nodes and collecting the edge types for each occuring pair of GF-types.

Let $n_i = \max\{|\,\mathrm{types}(i, j)|+\varepsilon(i, j) \ : \ j \in I\}$, where $\varepsilon(i, j) \in \{0, 1\}$ is 1 iff $i = j$. We define the canonical graph $\mathfrak{C}$ for $\mathfrak{K}$ via its edge expansion $\mathfrak{C}^+$. For every type $p_i \in T_1$ the universe of $\mathfrak{C}^+$ contains the elements $\{p_i^0, p_i^1, \ldots, p_i^{n_i-1}\}$. The unary predicates in $\mathfrak{C}^+$ are chosen such that the atomic type of every p_i^ℓ matches that of p_i.

The edges are added inductively, starting with the smallest $p_i \in T_1$. Suppose that all p_j^h for $j < i$ already have their neighbours in $\mathfrak{C}^+$, i.e. all edges that will be adjacent to any of the p_j^h have been added earlier. For all $j \in I$ proceed as follows. Let $t_1 < \cdots < t_n$ be an enumeration of $\mathrm{types}(i, j)$ that respects the order, and subsequently consider every t_k. If p_i^ℓ is not yet

Algorithm 2 GF$_2$ Canonisation

```
Input: Graph $\mathfrak{K}$ over ordered vocabulary $\tau$
  let $T_1$ be the set of occuring GF-types of width 1
  let $T_2$ be the set of occuring edge types
  create $\mathfrak{K}^+$ as per Section 4.1
  invoke Proposition 7.1.2 on $\mathfrak{K}^+$ to create $(\mathfrak{K}^+)^-$
  invoke Theorem 7.2.1 on $(\mathfrak{K}^+)^-$ to order $T_1$
  for all $i, j \in I$ do
    compute types$(i, j)$ /* see text */
  end for
  let $\mathfrak{C}^+$ be the empty structure
  for all $i \in I$ do
    let $n_i = \max\{|\text{types}(i, j)| + \varepsilon(i, j) \;:\; j \in I\}$
    add $\{p_1^0, \ldots, p_i^{n_i - 1}\}$ to $C^+$, atomic type same as $i$-th type of $T_1$
  end for
  for all $i \in I$ do /* from smallest to largest */
    for all $j \in I$ do /* from smallest to largest */
      for all $0 \leq \ell \leq n_i$ do /* from smallest to largest */
        for all $k \in |\text{types}(i, j)|$ do /* from smallest to largest */
          if there is no $m$ with a $t_k$-edge from $p_i^\ell$ to $p_j^m$ then
            let $m = (\ell + k + 1 \mod n_j)$
            while $p_i^\ell$ and $p_j^m$ are connected do
              let $m = (m + 1 \mod n_j)$
            end while
            add $t_k$-edge from $p_i^\ell$ to $p_j^m$ in $\mathfrak{C}^+$
          end if
        end for
      end for
    end for
  end for
  undo encoding of edge types to create $\mathfrak{K}^- = \mathfrak{C}$ from $\mathfrak{C}^+$
Output: Canonical $\mathfrak{K}^-$
```

incident to a node of type p_j via a t_k-edge, let m be the first integer modulo n_j, starting from $\ell + k + 1$, where p_i^ℓ and p_j^m are not connected in $\mathfrak{C}^+$, and add a t_k edge from p_i^ℓ to p_j^m.

The total number of repetitions of the inside of this nested loop is polynomial in the number of occuring width-1 GF-types, the number of edge types, and the sum of the n_i. Further, it is shown below that the resulting canonical structure is of minimal possible size. This implies that the sum of the n_i values is at most the size of the given input structure. Summing up, the computation of the sets types(i, j), and therefore $\mathfrak{C}^+$, is polynomial time.

The n_j are large enough to prevent the construction from introducing multiple edges into $\mathfrak{C}^+$. Therefore we can undo the edge-type encoding of $\mathfrak{C}^+$ to obtain the desired $\mathfrak{C}$. For a reasonable representation of $\mathfrak{C}^+$, this can be done in linear time in the size of this representation.

Towards correctness, regard the set $Z = \{(a, p_i^\ell) \in K \times C \ : \ \mathfrak{K} \models p_i(a)\}$ which clearly is a total bisimulation $Z : \mathfrak{K}^+ \sim \mathfrak{C}^+$, so, by Corollary 4.1.3, $\mathfrak{K}$ and $\mathfrak{C}$ are guarded bisimilar.

Further, $\mathfrak{C}$ is minimal in the sense that for every $p_i \in T_1$, and every structure $\mathfrak{B} \sim_g \mathfrak{K}$, the number of elements that realise p_i in $\mathfrak{B}$ has to be at least n_i. For suppose that this is not the case. Then there is a j such that there are less than $|\,\text{types}(i, j)| + \varepsilon(i, j)$ elements that realise p_i in $\mathfrak{B}$. For every $b_j \in \mathfrak{B}$ that realises p_j, there are at most $|\,\text{types}(i, j)| - 1$ elements of type p_i to connect to, however $|\,\text{types}(i, j)|$ different atomic types of width 2 to realise, a contradiction. □

7.3 AGF$_1$ Canonisation

The other case where we can give a concise algorithm allows arbitrary width, but is restricted to the AGF$_1$ quantifier pattern. Then a re-use of modal canonisation through an appropriate encoding of structures, similar to the methods presented in Chapter 4, can be used.

Theorem 7.3.1. AGF$_1$*-bisimulation allows canonisation.*

Proof. Let $\mathfrak{B}$ be a $\tau_s \dot\cup \tau_a$-structure. Similar to the construction of $\mathfrak{K}(\mathfrak{B})$ in Section 4.4, we associate a graph $\mathfrak{G}(\mathfrak{B})$ with $\mathfrak{B}$. Suppose that the maximal arities in τ_a are m_{in} and m_{out}, i.e. for all $R \in \tau_a$, the arity (i, j) of R satisfies $i \leq m_{\text{in}}$ and $j \leq m_{\text{out}}$. Let m_{state} be the maximal arity of the state predicates in τ_s. Let S be the set of all $\tau_s \dot\cup \tau_a$-structures with universes

$\{1, \ldots, k\}$, $1 \leq k \leq m = \max\{m_{\text{in}}, m_{\text{out}}, m_{\text{state}}\}$. Let H contain all atomic statements $\alpha(\boldsymbol{x}; \boldsymbol{y})$ built with actions from τ_a.

The vocabulary $\hat{\tau}$ has predicates

$P_{\mathfrak{A}}$	(monadic)	for $\mathfrak{A} \in S$,
E_α	(binary)	for $\alpha \in H$.

Let $M = M(\mathfrak{B})$ contain, as tuple, an arbitrary linearisation of every maximal active set, and every maximal guarded set, of $\mathfrak{B}$. Technically this requires the input structure to be ordered, as is implicitly the case for most common computation models. The algorithm is however order invariant. The isomorphism class of the canonical structure obtained at the end of this proof does not depend on the precise choice of how the universe, and the considered subsets, are ordered. The transition system $\mathfrak{G}(\mathfrak{B})$ is the following $\hat{\tau}$-structure.

$$\begin{aligned}
\mathfrak{G}(\mathfrak{B}) &= \big(M(\mathfrak{B}), (E_\alpha)_{\alpha \in H}, (P_{\mathfrak{A}})_{\mathfrak{A} \in S}\big) \\
P_{\mathfrak{A}} &= \big\{(b_1, \ldots, b_k) \in M \;:\; \mathfrak{B}|_{\{b_1, \ldots, b_k\}} \simeq \mathfrak{A} \text{ via } b_i \mapsto i\big\} \\
E_\alpha &= \big\{((c_1, \ldots, c_l), (d_1, \ldots, d_k)) \;:\; \mathfrak{B} \models \alpha(c_1, \ldots, c_l; d_1, \ldots, d_k)\big\}
\end{aligned}$$

Here, as an exception, $\alpha(\boldsymbol{x}; \boldsymbol{y})$ does *not* mean that all variables $\boldsymbol{x}$ and $\boldsymbol{y}$ actually occur in the atomic statement α. This construction can be done in polynomial time. Finding $M(\mathfrak{B})$ is a matter of iterating over all tuples in action relations and updating a sorted list of maximal active or guarded sets. Finding the correct $P_{\mathfrak{A}}$-label for each resulting set is a sequence of lookups for all possible atomic statements. Since m is assumed constant, this is still polynomial. For the same reason, finding all E_α-edges can be done efficiently by checking every possible combination of subsets of pairs of tuples in $M(\mathfrak{B})$ for edges in $\mathfrak{B}$.

We claim that the reduced graph $\mathfrak{G}(\mathfrak{B})^-$ can be converted back into a $\tau_s \dot{\cup} \tau_a$-structure $\mathfrak{B}^-$ that is AGF$_1$-bisimilar to $\mathfrak{B}$. Towards this proposition, note that $\mathfrak{G}(\mathfrak{B})$ does not induce an equivalence relation on the elements in the $P_{\mathfrak{A}}$-labels — as was the case in the $\mathfrak{K}(\mathfrak{B})$ construction. Then, for every $v \in \mathfrak{G}(\mathfrak{B})^-$ let $\mathfrak{A}_v$ be the unique $\mathfrak{A} \in S$ that holds at v. The canonical-to-be $\mathfrak{B}^-$ is defined as disjoint union of all $\mathfrak{A}_v$, $v \in \mathfrak{G}(\mathfrak{B})^-$, plus the atomic information recorded in the E_α. That is,

$$B^- \;=\; \dot{\bigcup}_{v \in \mathfrak{G}(\mathfrak{B})^-} A_v = \big\{(v, i) \colon v \in \mathfrak{G}(\mathfrak{B})^-, i \in \mathfrak{A}_v\big\},$$

and for every $v \in \mathfrak{G}(\mathfrak{B})^-$ the function that maps all $i \in A_v$ to (v, i) is an isomorphic embedding of $\mathfrak{A}_v$ into $\mathfrak{B}^-$, and finally $E_\alpha(v, w)$ holds in $\mathfrak{G}(\mathfrak{B})^-$

iff $\alpha((v,1),\dots,(v,l);(w,1),\dots,(w,k))$ holds in $\mathfrak{B}^-$, for $l = |\mathfrak{A}_v|$, $k = |\mathfrak{A}_w|$. This is a straightforward transformation that iterates once over the nodes and edges, in that order, to successively build $\mathfrak{B}^-$.

Algorithm 3 AGF$_1$ Canonisation

```
Input: Structure $\mathfrak{B}$ over ordered vocabulary $\tau_a \dot\cup \tau_s$
  find maximal arity $m = \max\{m_{\text{in}}, m_{\text{out}}, m_{\text{state}}\}$
  create set $H$ of all $\tau_a$-action statements
  create set $S$ of all $\tau$-structures of size $\leq m$
  initialise $M(\mathfrak{B})$ as empty list
  for all $R \in \tau_s$, $\boldsymbol{a} \in R$ do
    let $M^{\subsetneq} = \{\boldsymbol{c} \in M \ : \ \boldsymbol{c} \subsetneq \boldsymbol{a}\}$
    let $M^{\supseteq} = \{\boldsymbol{c} \in M \ : \ \boldsymbol{c} \supseteq \boldsymbol{a}\}$
    if $M^{\supseteq}$ is empty then
      insert $\boldsymbol{a}$ into $M(\mathfrak{B})$
      erase all $\boldsymbol{c} \in M^{\subsetneq}$ from $M(\mathfrak{B})$
    end if
  end for
  for all $R \in \tau_a$, $(\boldsymbol{a};\boldsymbol{b}) \in R$ do
    /* conditionally update $M$ with $\boldsymbol{a}$ and $\boldsymbol{b}$ as above */
  end for
  initialise $\mathfrak{G}(\mathfrak{B})$ with universe $M(\mathfrak{B})$
  for all $\boldsymbol{c} \in M(\mathfrak{B})$ do
    add correct $P_{\mathfrak{A}}$-label to $\mathfrak{G}(\mathfrak{B})$
  end for
  for all $\boldsymbol{a}, \boldsymbol{b} \in M(\mathfrak{B})$ do
    add all appropriate $E_\alpha$-edges to $\mathfrak{G}(\mathfrak{B})$
  end for
  invoke Proposition 7.1.2 for $\mathfrak{G}(\mathfrak{B})$
  let $\mathfrak{B}^- = \dot\bigcup\{\boldsymbol{b} \in \mathfrak{G}(\mathfrak{B})^-\}$
  for all $\alpha \in H$, $(v,w) \in E_\alpha$ do
    add action statement corresponding to $(v,w)$ to $\mathfrak{B}^-$
  end for
Output: Canonical $\mathfrak{B}^-$
```

For $\boldsymbol{b} \in M$, let $[\boldsymbol{b}]_\sim$ be the bisimulation equivalence class of $\boldsymbol{b}$, seen as node of $\mathfrak{G}(\mathfrak{B})$, which is the same as to say that $v = [\boldsymbol{b}]_\sim$ is *the* node of $\mathfrak{G}(\mathfrak{B})^-$ that is bisimilar to $\boldsymbol{b}$. Let $f_{\boldsymbol{b}}$ be the function that maps every $b_i \in \mathfrak{B}$ to $(v,i) \in \mathfrak{B}^-$. Clearly every $f_{\boldsymbol{b}}$ is a partial τ_s-isomorphism. Define I as the smallest set of functions that contains $\{f_{\boldsymbol{b}} \ : \ \boldsymbol{b} \in M\}$ and is closed under subfunctions.

If $f : \mathfrak{B} \to \mathfrak{B}^- \in I$, given as $\boldsymbol{b} = f(\boldsymbol{a})$, then I is an AGF$_1$-bisimulation from $\mathfrak{B}, \boldsymbol{a}$ to $\mathfrak{B}^-, \boldsymbol{b}$. Since I covers $\mathfrak{B}$ and $\mathfrak{B}^-$ completely, it suffices to show that I satisfies the back and forth conditions to prove that $\mathfrak{B}$ and $\mathfrak{B}^-$ are AGF$_1$-bisimilar.

For forth, let $f : X \to Y \in I$, $\boldsymbol{a} \subseteq X$ active and $\mathfrak{B} \models \alpha(\boldsymbol{a}; \boldsymbol{a}')$ for some $\alpha = R(i_1, \ldots, i_l; j_1, \ldots, j_k)$, $R \in \tau_a$, and $\boldsymbol{c}, \boldsymbol{c}' \in M$, $X \subseteq \boldsymbol{c}$, such that $\boldsymbol{a} = (a_1, \ldots, a_l) = (c_{i_1}, \ldots, c_{i_l})$, $\boldsymbol{a}' = (a'_1, \ldots, a'_k) = (c'_{j_1}, \ldots, c'_{j_k})$. Let $f' = f_{\boldsymbol{c}'}|_{\boldsymbol{a}'}$, by definition an element of I, and let $\boldsymbol{b}' = f'(\boldsymbol{a}')$, $\boldsymbol{b} = f(\boldsymbol{a})$. By construction, in this situation there is an E_α-edge from $\boldsymbol{c}$ to $\boldsymbol{c}'$ in $\mathfrak{G}(\mathfrak{B})$, and consequently also an E_α-edge from $[\boldsymbol{c}]_\sim$ to $[\boldsymbol{c}']_\sim$ in $\mathfrak{G}(\mathfrak{B})^-$. Now $\boldsymbol{b}$ is nothing else than $(([\boldsymbol{c}]_\sim, i_1), \ldots, ([\boldsymbol{c}]_\sim, i_l))$, and $\boldsymbol{b}'$ is $(([\boldsymbol{c}]_\sim, j_1), \ldots, ([\boldsymbol{c}]_\sim, j_k))$, so $\mathfrak{B}^- \models R(\boldsymbol{b}; \boldsymbol{b}')$ as desired.

In the back direction, let f and α be as above, and let $\mathfrak{B}^- \models \alpha(\boldsymbol{b}, \boldsymbol{b}')$ for some $\boldsymbol{b} \subseteq Y$, $\boldsymbol{b}' \in \mathfrak{B}^-$. Again we can find some $\boldsymbol{c}, \boldsymbol{c}' \in M$ such that $\boldsymbol{b} = (([\boldsymbol{c}]_\sim, i_1), \ldots, ([\boldsymbol{c}]_\sim, i_l))$ and $\boldsymbol{b}' = (([\boldsymbol{c}]_\sim, j_1), \ldots, ([\boldsymbol{c}]_\sim, j_k))$. By definition of $\mathfrak{B}^-$, $\mathfrak{B}^- \models \alpha(\boldsymbol{b}, \boldsymbol{b}')$ implies that there is an E_α-edge from $[\boldsymbol{c}]_\sim$ to $[\boldsymbol{c}']_\sim$ in $\mathfrak{G}(\mathfrak{B})^-$, therefore we can find a $\boldsymbol{c}'' \in M$ that is bisimilar to $\boldsymbol{c}'$, and an E_α-successor of $\boldsymbol{c}$ in $\mathfrak{G}(\mathfrak{B})$; we have $\mathfrak{G}(\mathfrak{B}), \boldsymbol{c}'' \sim \mathfrak{G}(\mathfrak{B}), \boldsymbol{c}' \sim \mathfrak{G}(\mathfrak{B})^-, [\boldsymbol{c}']_\sim$. Then with $\boldsymbol{a} = f^{-1}(\boldsymbol{b})$ and $\boldsymbol{a}'' = (c''_{i_1}, \ldots, c''_{i_k})$ we know that $\mathfrak{B} \models \alpha(\boldsymbol{a}, \boldsymbol{a}'')$. With our choice of $\boldsymbol{c}''$, the desired f' can be obtained as $f_{\boldsymbol{c}''}|_{\boldsymbol{a}''}$. □

7.4 Guarded Canonisation

In this section we try to give some intuition into why no canonisation procedure for guarded bisimulation has been found so far, and indeed might not even exist at all. Of course it is possible to enumerate all structures, from small to large, and check each one whether it is guarded bisimilar to the given input structure. While this does succeed in deterministically finding a minimal bisimilar companion structure, the complexity of the enumeration of structures and bisimulations obviously prohibits any practical use.

Conjecture 7.4.1. *Guarded bisimulation does not allow canonisation.*

The crux seems to lie in *finite* graph representations of structures, in particular their consistency conditions. For our means, a graph representation is consistent if it can be transformed back into a relational structure in an appropriate fashion. For guarded bisimulation we considered the transition systems $\mathfrak{K}(\mathfrak{B})$, and their tree versions $\mathfrak{T}(\mathfrak{B})$. We showed that, independent of these transformations, a first-order definable class of trees of the right format can be transformed back into structures in a manner that bisimulation on trees corresponds to guarded bisimulation on the transformed-back structures. The other crucial property of the class of consistent trees is that it is closed under bisimulation. The heart of the problems is the combination of these two requirements.

Suppose that τ contains a ternary predicate R, and consider the class of graphs of vocabulary $\tilde{\tau}$, i.e. the graphs with the same vocabulary as the $\mathfrak{K}(\mathfrak{B})$-representations, cf. Section 4.4, for τ-structures $\mathfrak{B}$. Some of these graphs exhibit exactly the same situation as the dormant clashes in Chapter 5. For an example, consider a graph $\mathfrak{K}$ that contains two nodes v, w, each labelled with the structure $\mathfrak{A} = (\{1,2,3\}, R = \{(1,2,3)\})$. Suppose that there are three distinct paths p_1, p_2, p_3 of length at least 2 that lead from v to w, and that the ρ-labels along the paths amount to $\rho_1 = (1 \mapsto 2)$, $\rho_2 = (2 \mapsto 3)$ and $\rho_3 = (3 \mapsto 1)$. A simple local consistency condition is unable to detect that the atomic information at v and w is contradictory. To make all relevant information locally available one could enforce that, in the terms of Section 4.5, the set of nodes where a $\approx$-class is represented forms a clique, and for every pair of nodes there is a ρ-edge that contains a mapping for the element in question. This requirement however is completely incompatible with closure under bisimulation equivalence.

For guarded logics on graphs, i.e. guarded bisimulation of width 2, edge expansion creates this correspondence between width-2 guarded bisimulation, and modal bisimulation. The consistency condition for any finite or infinite graph or tree in the vocabulary of an atomic expansion is that it may not have multiple edges, again expressible in first-order logic. The simple consistency check made the canonisation algorithm for this case possible, although the modal canonisation procedure failed.

The other case was AGF_1-bisimulation, where the encoding of structures as graphs differs from the guarded case in that a guard statement $R(\boldsymbol{a}; \boldsymbol{b})$ leads to two distinct nodes, one each for $\boldsymbol{a}$ and $\boldsymbol{b}$, respectively. Although the arity is not restricted, this more closely matches the modal case where quantification is a move that loses the old position. The effect is that without the need to share elements between nodes, we do not require any consistency condition at all. This made it possible to use modal canonisation as core of AGF_1-canonisation.

In other words, graph encodings are intuitively easy to handle when at least one of (a) graphs are trees, (b) graphs are for structures of width-2, or (c) background is a bisimulation for a quantifier pattern that does not retain source elements, applies. Compare also the tableau algorithm for modal logics. Building the actual tableau, and retrieval of an infinite, tree-like model [35] are significantly less involved with respect to consistency conditions than the construction that gave us finite models. What this boils down to is that *guarding the quantifiers is not the whole truth about modal logics*.

A solution, and in particular an encoding that allows a simple consistency condition for finite graph representations with respect to full guarded bisimulation, has so far been quite elusive. Unfortunately this also hinders progress in direction of finite model theory versions of the bisimulation invariant fragment characterisation theorems, as will be seen in Chapter 8.

Chapter 8

Higher-Order Logics

In this chapter we will give an example of how the transformations between structures and graphs can be used to obtain a non-trivial result about guarded logic, employing the equivalent modal version. To this end, the back and forth transformations for structures from Chapter 4 are supplemented with corresponding translations for guarded second-order logic in one, and the modal μ-calculus in the other direction. A characterisation theorem for the guarded bisimulation invariant fragment of guarded second-order logic is obtained. This closes a gap, namely that the corresponding results from modal logic have so far only been known for the first-order level of guarded logics.

Beyond the general case, the study of the corresponding finite model theory variants has up to this point proven to be more difficult, and indeed beyond the first-order level little is yet known. This chapter concludes with an automata-theoretic approach that enables a small step in this direction for a weaker variant of monadic second-order logic.

8.1 Back and Forth

Recall the characterisation of the modal μ-calculus from Theorem 2.6.6. We want to apply this characterisation in restriction to trees, and therefore refer to the following variant, which is proved en route to Theorem 2.6.6 in [44].

Theorem 8.1.1 (Janin, Walukiewicz). *A class of trees is definable in the modal μ-calculus if, and only if, it is definable in monadic second-order logic and closed under bisimulation within the class of all trees.*

Towards a reduction of Theorem 8.1.11, in the case of sentences, to Theorem 8.1.1, we define a "forth" translation that maps every sentence $\psi \in \text{GSO}[\tau]$ to a formula $\psi^{\rightarrow}(x) \in \text{MSO}[\tilde{\tau}]$ with one free variable, and a "back" translation that maps every formula $\varphi \in \text{L}_\mu[\tilde{\tau}]$ to a sentence $\varphi^{\leftarrow} \in \mu\text{GF}[\tau]$. These translation will be such that

(1) If $\mathfrak{T}$ is a consistent tree with root λ, then $\mathfrak{T} \models \psi^{\rightarrow}(\lambda)$ iff $\mathfrak{D}(\mathfrak{T}) \models \psi$.

(2) If $\mathfrak{B}$ is a τ-structure and λ the root of $\mathfrak{T}(\mathfrak{B})$, then $\mathfrak{T}(\mathfrak{B}), \lambda \models \varphi$ iff $\mathfrak{B} \models \varphi^{\leftarrow}$.

It follows from (1) with Proposition 4.5.5 that GSO sentences that are invariant under guarded bisimulation are mapped to MSO formulae that are bisimulation invariant on consistent trees; cf. Corollary 8.1.7 below.

Before giving the formal definitions, we informally discuss the main problems arising from the differences between the guarded and the modal viewpoint.

We wish to translate GSO sentences to MSO formulae and L_μ formulae back to μGF sentences. We want a modal formula to hold at some node v if, and only if, the corresponding guarded formula holds of the guarded list represented by v. For second-order variables we need to map sets of guarded tuples to sets of nodes and vice versa.

For each node v of a consistent tree $\mathfrak{T}$, the associated structure $\mathfrak{A}_v$ (if not empty) represents many different guarded tuples, which may or may not occur in a given guarded set, independently of each other.

In the translation that takes us from relational structures to trees, a second-order variable, which ranges over some guarded relation in the relational structure, has to be coded by several monadic second-order variables over the tree, each one containing the information about membership of one particular guarded tuple of components that live at the corresponding node. We translate an r-ary second-order variable Z in a GSO sentence into a sequence $Z^{\rightarrow}$ of monadic variables $Z_{i_1,\dots,i_r}$, one for each (local) choice of elements from the guarded list. The idea is that $v \in Z_{i_1,\dots,i_r}$ for a node v representing a guarded list $([v,1],\dots,[v,k])$ stands for $([v,i_1],\dots,[v,i_r]) \in Z$.

In the other direction we have to deal with monadic second-order variables, which range over arbitrary sets of nodes of the tree, and in particular over sets of nodes of different sizes, corresponding to guarded lists of different lengths. We choose a translation that effectively splits up any monadic second-order variable X into disjoint parts, each corresponding to nodes of a fixed size, before translating these parts into guarded relations of appropriate arity. Consequently, a monadic second-order variable X is translated into a sequence $X^{\leftarrow}$ of second-order variables $X_i, 0 \leq i \leq m$, such that each X_i ranges over guarded lists of length i. The idea is that $\boldsymbol{d} \in X_i$ stands for $v \in X$ where the guarded list $\boldsymbol{d}$ has length i and is represented at node v.

8.1.1 From GSO to MSO

Without loss of generality we restrict attention to GSO sentences in GSO_0, cf. Corollary 4.2.3. Let m be the width of τ, i.e. the maximal arity of relations in τ. If Z is an r-ary second-order variable, $Z^{\rightarrow} = (Z_{i_1,\ldots,i_r})_{i_1,\ldots,i_r \leq m}$ is the corresponding sequence of monadic predicates according to the above discussion.

Definition 8.1.2. Let $\mathfrak{T}$ be a consistent $\tilde{\tau}$-tree, and let $\mathfrak{D}(\mathfrak{T})$ be the associated τ-structure. A tuple $J^{\rightarrow}$ of monadic predicates $J_{i_1,\ldots,i_r}$ on $\mathfrak{T}$ *encodes* an r-ary guarded relation J on $\mathfrak{D}(\mathfrak{T})$ iff $J_{i_1,\ldots,i_r} = \{v : ([v, i_1], \ldots, [v, i_r]) \in J\}$ for all $i_1, \ldots, i_r \leq m$.

Not all sequences $\boldsymbol{J}$ of monadic predicates over $\mathfrak{T}$ do indeed encode a guarded relation over $\mathfrak{D}(\mathfrak{T})$. To do so, they have to satisfy the following correctness conditions.

(a) $J_{i_1,\ldots,i_\ell}$ only contains nodes v where all i_j are in $\mathfrak{A}_v$.

(b) $\boldsymbol{J}$ is consistent on tuples living at different nodes, i.e. if in $\mathfrak{D}(\mathfrak{T})$ a tuple $(d_1, \ldots, d_r)$ is represented by $(i_1, \ldots, i_r)$ at node u and by $(j_1, \ldots, j_r)$ at node v, then $u \in J_{i_1 \cdots i_r}$ iff $v \in J_{j_1 \cdots j_r}$.

Lemma 8.1.3. *For each $r \leq m$ there exists a first-order formula* $\mathrm{correct}(Z^{\rightarrow})$ *that expresses the correctness conditions (a) and (b) above.*

These conditions are necessary and sufficient in the sense that a tuple $\boldsymbol{J}$ over a consistent tree $\mathfrak{T}$ encodes a guarded relation on $\mathfrak{D}(\mathfrak{T})$ if, and only if, $\mathfrak{T} \models \mathrm{correct}(\boldsymbol{J})$.

Proof. Note that it suffices to express condition (b) for adjacent nodes to enforce it globally. Thus the consistency requirement for $Z^{\rightarrow}$ can be expressed by a first-order formula $\mathrm{correct}(Z^{\rightarrow})$ that states condition (a) in an obvious way and contains, for each $\rho \in F$ and each tuple $(i_1, \ldots, i_r)$ over $\mathrm{dom}(\rho)$ a clause

$$\forall x \forall y \big[E_\rho xy \rightarrow \big(Z_{i_1,\ldots,i_r}(y) \iff Z_{\rho(i_1),\ldots,\rho(i_r)}(x)\big)\big].$$

The proof of the adequacy claim is straightforward. □

First-order quantifiers also require special treatment. Let us first consider the case where $\mathfrak{T}$ is a tree representation $\mathfrak{T}(\mathfrak{B})$. As noted earlier, such trees satisfy the strong homogeneity condition that for all nodes u, u' and all successors v of u there is a successor v' of u' such that the subtrees with roots v and v' are isomorphic. To put it differently, an indistinguishable copy of any guarded list anywhere in $\mathfrak{B}$ is available locally, at some child of the current node in $\mathfrak{T}(\mathfrak{B})$. Therefore guarded first-order quantifications over $\mathfrak{B}$ can be simulated over $\mathfrak{T}(\mathfrak{B})$ by moving to an immediate successor of the current node (i.e. by a modal quantifier $\Diamond$ or $\Box$). However, in the relationship between $\mathfrak{T}$ and $\mathfrak{D}(\mathfrak{T})$, if $\mathfrak{T}$ is an arbitrary consistent $\tilde{\tau}$-tree, this is no longer the case. To verify a formula of the form $(\exists^{\neq}\boldsymbol{y} \,.\, \rho(\boldsymbol{x}, \boldsymbol{y}) \wedge \alpha(\boldsymbol{y}))\varphi(\boldsymbol{y})$ we want to move from the current tuple $\boldsymbol{x}$ to a new tuple $\boldsymbol{y}$, guarded by α, such that $\varphi(\boldsymbol{y})$ is true and the overlap conditions for $\boldsymbol{x}$ and $\boldsymbol{y}$ as stated by $\rho(\boldsymbol{x}, \boldsymbol{y})$ are satisfied. In arbitrary consistent trees, such a witness need not exist locally, but may only occur as a remote node, which is linked to the current node by a path along which the common components according to ρ are kept.

Example. Let $\tau = \{R\}$, R ternary, and consider the GF_0 formula $\varphi(y_1) = (\exists^{\neq} x_2 x_3 . R(y_1 x_2 x_3))$. We give a partial description of a series of consistent $\tilde{\tau}$-trees $\mathfrak{T}_n$ as follows (explicitly exhibiting only the parts relevant to this example). $\mathfrak{T}_n$ has nodes $\lambda, v_0, v_1, \ldots, v_n$ (plus some extra nodes required by consistency condition (c) in Definition 4.5.1); $\mathfrak{A}_\lambda = \emptyset$, $\mathfrak{A}_{v_i} = (\{1\}, R^{\mathfrak{A}_{v_i}} = \{(1,1,1)\})$ for $i < n$ and $\mathfrak{A}_{v_n} = (\{1,2,3\}, R^{\mathfrak{A}_{v_n}} = \{(1,1,1),(1,2,3)\})$. There is an $E_\emptyset$-edge from λ to v_0, and for each $i = 1, \ldots, n$ there is an $E_\emptyset$ and an E_ρ-edge from v_{i-1} to v_i, where ρ is the function that maps 1 to 1.

Observe that $\mathfrak{D}(\mathfrak{T}_n) = \mathfrak{A}_{v_n}$ and consequently $\mathfrak{D}(\mathfrak{T}_n) \models \varphi([v_0, 1])$ for all n. Whatever the exact definition of $\varphi^{\rightarrow}(z)$, we expect that $\mathfrak{T}_n \models \varphi^{\rightarrow}(v_0)$ due to the fact that if $y_1 = [v_0, 1] = [v_n, 1]$ then $\exists^{\neq} x_2 x_3 . R(y_1 x_2 x_3)$ holds in $\mathfrak{A}_{v_n}$. However, in each $\mathfrak{T}_n$ the distance from v_0 to a node that can verify the existence of the additional elements as required by φ is n. This shows that it will not be possible to simply translate our given guarded first-order quantification into their modal counterparts $\Diamond$ and $\Box$, since those are local.

To capture this situation in MSO we use a sequence $\boldsymbol{W}$ of monadic predicates W^k_ρ, for $k \leq m$ and $\rho \in F$, that — relative to a given node u — partition the set of nodes according to their size and their overlap with u. The proof of the following is then straightforward.

Lemma 8.1.4. *There is a first-order formula* F-part$(z, \boldsymbol{W})$ *expressing the* *following correctness conditions on partitions with respect to node z. For every consistent tree $\mathfrak{T}$, $u \in \mathfrak{T}$, and every sequence $\boldsymbol{W} = (W^k_\rho)_{k \leq m, \rho \in F}$ of monadic predicates on $\mathfrak{T}$, we have $\mathfrak{T} \models \text{F-part}(u, \boldsymbol{W})$ if, and only if, for all k, ρ*

$$W^k_\rho = \{w : |\mathfrak{A}_w| = k \textit{ and } \rho(i) = j \Leftrightarrow [w,i] = [u,j]\}.$$

Proof. As in the previous lemma, it suffices to impose corresponding conditions locally and along all edges. The formula $\text{F-part}(z, \boldsymbol{W})$ states that

(a) The sets $\boldsymbol{W}$ form a partition of the universe.
(b) The node z itself belongs to $W^{|\mathfrak{A}_z|}_{\text{id}}$.
(c) If $y \in W^k_\rho$, then $|\mathfrak{A}_y| = k$.
(d) For all ρ, k, ℓ, and σ: if $(x,y) \in E_\rho$, $|\mathfrak{A}_x| = \ell$, $|\mathfrak{A}_y| = k$, then $x \in W^\ell_\sigma \Leftrightarrow y \in W^k_{\sigma \circ \rho}$ and $y \in W^k_\sigma \Leftrightarrow x \in W^\ell_{\sigma \circ \rho^{-1}}$.

One shows by induction on the distance from u that $\text{F-part}(u, \boldsymbol{W})$ expresses the right property. □

The Translation. Recall that formulae in GSO_0 either belong to GSO_X and have all their free first-order variables in X, or they belong to GSO_Y and have all their free first-order variables in Y, where X and Y are two disjoint sets of variables. Further, formulae in GSO_X that start with a quantifier are of the form $\psi(\boldsymbol{x}) = (\exists^{\neq} \boldsymbol{y} \,.\, \rho(\boldsymbol{x}, \boldsymbol{y}) \wedge \alpha(\boldsymbol{y}))\varphi(\boldsymbol{y})$ with $\alpha(\boldsymbol{y}), \varphi(\boldsymbol{y}) \in \text{GSO}_Y$.

We inductively translate every formula $\psi(x_1, \dots, x_k, Z_1, \dots, Z_r) \in \text{GSO}_X[\tau]$ into an $\text{MSO}[\tilde{\tau}]$ formula $\psi^{\rightarrow}(x, Z_1^{\rightarrow}, \dots, Z_r^{\rightarrow})$, with a single free first-order variable x and sequences of monadic second-order variables $Z_i^{\rightarrow}$ that correspond to the second-order variables Z_i. Similarly, formulae in GSO_Y are translated into formulae $\psi^{\rightarrow}$ with free first-order variable y. We just present the translation for formulae in GSO_X:

(1) If $\psi = R(x_{i_1}, \dots, x_{i_r})$ for $R \in \tau$,
set $\psi^{\rightarrow}(x) = \bigvee\{P_{\mathfrak{A}}(x) \;:\; \mathfrak{A} \in S, \mathfrak{A} \models R(i_1, \dots, i_r)\}$.
(2) If $\psi = Z(x_{i_1}, \dots, x_{i_r})$, for some r-ary relation variable Z,
set $\psi^{\rightarrow}(x) = Z_{i_1, \dots, i_r}(x)$.
(3) The translation commutes with $\neg$, $\wedge$, $\vee$.
(4) If $\psi = (\exists^{\neq} \boldsymbol{y} \,.\, \rho(\boldsymbol{x}, \boldsymbol{y}) \wedge \alpha(\boldsymbol{y}))\varphi(\boldsymbol{y})$,
let $\psi^{\rightarrow}(x) = (\exists \boldsymbol{W} \,.\, \text{F-part}(x, \boldsymbol{W})) \bigvee_{\sigma \supseteq \rho} \big((\exists y \,.\, W^{|\boldsymbol{y}|}_\sigma(y))(\alpha \wedge \varphi)^{\rightarrow}(y)\big)$.
(5) For $\psi = \exists Z \varphi$ with r-ary relation variable Z,
let $\psi^{\rightarrow}(x) = (\exists Z^{\rightarrow} \,.\, \text{correct}(Z^{\rightarrow}))\varphi^{\rightarrow}(x)$.

Theorem 8.1.5. *Let $\mathfrak{T}$ be a consistent $\tilde{\tau}$-tree with root λ, $\mathfrak{D}(\mathfrak{T})$ the associated τ-structure and ψ a sentence in* GSO$[\tau]$. *Then $\mathfrak{D}(\mathfrak{T}) \models \psi$ iff $\mathfrak{T} \models \psi^{\rightarrow}(\lambda)$.*

Proof. This theorem is a consequence of the following more general statement. Consider any formula $\psi(x_1, \dots, x_k, Z_1, \dots, Z_s)$ in GSO$_X[\tau]$ with free first- and second-order variables as displayed. Its translation into MSO$[\tilde{\tau}]$ is $\psi^{\rightarrow}(x, Z_1^{\rightarrow}, \dots, Z_s^{\rightarrow})$. Let $(d_1, \dots, d_k)$ be a guarded list in $\mathfrak{D}(\mathfrak{T})$, and let v be a node of $\mathfrak{T}$ such that $|\mathfrak{A}_v| = k$ and $d_i = [v, i]$. Let $J_1, \dots, J_s$ be sets of guarded tuples in $\mathfrak{D}(\mathfrak{T})$, and let $J_1^{\rightarrow}, \dots, J_s^{\rightarrow}$ be their representations according to Definition 8.1.2.

Claim 8.1.6. $\mathfrak{D}(\mathfrak{T}) \models \psi(d_1, \dots, d_k, J_1, \dots, J_r) \Leftrightarrow \mathfrak{T} \models \psi^{\rightarrow}(v, J_1^{\rightarrow}, \dots, J_r^{\rightarrow})$.

Note that for sentences (i.e. $k = r = 0$) and for $v = \lambda$, the claim implies the theorem. The claim itself is established inductively. The cases corresponding to (1) — (3) are immediate.

(4) Let $\psi = (\exists^{\neq} \boldsymbol{y} \,.\, \rho(\boldsymbol{x}, \boldsymbol{y}) \wedge \alpha(\boldsymbol{y}))\varphi(\boldsymbol{y})$. As second-order variables play no role for this case, we suppress them.

$$\psi^{\rightarrow}(x) = (\exists \boldsymbol{W} \,.\, \text{F-part}(x, \boldsymbol{W})) \bigvee_{\sigma \supseteq \rho} (\exists y \,.\, W_{\sigma}^{|\boldsymbol{y}|}(y))(\alpha \wedge \varphi)^{\rightarrow}(y)$$

Suppose that $\mathfrak{D}(\mathfrak{T}) \models \psi(\boldsymbol{d})$. Then there exists a guarded list $\boldsymbol{e} = (e_1, \dots, e_\ell)$ of length $\ell = |\boldsymbol{y}|$ such that $\mathfrak{D}(\mathfrak{T}) \models \alpha(\boldsymbol{e}) \wedge \varphi(\boldsymbol{e})$ and $e_i = d_j$ for $\rho(i) = j$. Thus $\sigma = \{(i, j) \colon e_i = d_j\}$ extends ρ. Since $\boldsymbol{e}$ is guarded, there exists a node w of size $|w| = \ell$ such that all e_i live together at w. Actually, due to condition (c) for consistent trees, we can assume that $\boldsymbol{e} = ([w, 1], \dots, [w, \ell])$. We know that there exists a $\boldsymbol{W}$ satisfying F-part$(v, \boldsymbol{W})$. Since $\sigma = \{(i, j) \colon [w, i] = [v, j]\}$, it follows that $\mathfrak{T} \models W_{\sigma}^{\ell}(w)$ and, by induction hypothesis, $\mathfrak{T} \models (\alpha \wedge \varphi)^{\rightarrow}(w)$. Therefore $\mathfrak{T} \models \psi^{\rightarrow}(v)$.

Conversely, suppose that $\mathfrak{T} \models \psi^{\rightarrow}(v)$. For the (unique) tuple $\boldsymbol{W}$ satisfying F-part$(v, \boldsymbol{W})$, there exist $\sigma \supseteq \rho$ and a node $w \in W_{\sigma}^{\ell}$ such that $\mathfrak{T} \models (\alpha \wedge \varphi)^{\rightarrow}(w)$. For $e_i = [w, i]$ we find that $e_i = d_j$ for all $(i, j) \in \rho$ and, by induction hypothesis, $\mathfrak{D}(\mathfrak{T}) \models (\alpha \wedge \varphi)(\boldsymbol{e})$. Therefore $\mathfrak{D}(\mathfrak{T}) \models \psi(\boldsymbol{d})$.

For (5), the claim is immediate from the induction hypothesis and from Lemma 8.1.3. □

Corollary 8.1.7. *If $\psi \in$ GSO$[\tau]$ is a sentence that is invariant under guarded bisimulation, then $\psi^{\rightarrow}(x)$ is bisimulation invariant on consistent $\tilde{\tau}$-trees.*

Proof. Let $\mathfrak{T}, \mathfrak{T}'$ be two bisimilar, consistent $\tilde{\tau}$-trees. Then $\mathfrak{D}(\mathfrak{T}) \sim_g \mathfrak{D}(\mathfrak{T})$ by Proposition 4.5.5. It follows that

$$\mathfrak{T} \models \psi^{\rightarrow}(\lambda) \;\Leftrightarrow\; \mathfrak{D}(\mathfrak{T}) \models \psi \;\Leftrightarrow\; \mathfrak{D}(\mathfrak{T}') \models \psi \;\Leftrightarrow\; \mathfrak{T}' \models \psi^{\rightarrow}(\lambda')$$

□

8.1.2 From L_μ to μGF

The translation back from the modal into the guarded world also requires some preliminary discussion. Every formula φ of the μ-calculus, evaluated on a tree $\mathfrak{T} = \mathfrak{T}(\mathfrak{B})$, defines the set $\varphi^{\mathfrak{T}}$ of all nodes v such that $\mathfrak{T}, v \models \varphi$. Recall from the discussion leading up to Definition 4.4.3 how each node v of $\mathfrak{T}$ represents a guarded list $\pi(v)$ in $\mathfrak{B}$. So the idea is to translate φ into a guarded formula $\psi(\boldsymbol{x})$ defining in $\mathfrak{B}$ the set $\psi^{\mathfrak{B}} = \{\boldsymbol{b} \in \mathfrak{B} : \mathfrak{B} \models \psi(\boldsymbol{b})\}$, which should be equal to $\{\pi(v) \in \mathfrak{B} : \mathfrak{T}, v \models \varphi\}$. The main problem is that the guarded lists represented in $\varphi^{\mathfrak{T}}$ are not in general of the same lengths, whence the full set cannot be described by a single guarded formula $\psi^{\mathfrak{B}}$ at all. We will actually translate φ into a tuple $(\varphi_0^{\leftarrow}, \ldots, \varphi_m^{\leftarrow})$ of formulae over $\mathfrak{B}$ where $\varphi_k = \varphi(x_1, \ldots, x_k)$, with the intention that φ_k defines the set of those guarded lists $\boldsymbol{b} = \pi(v)$ of length k, for which $v \in \varphi^{\mathfrak{T}}$. Note that the 0-component, $\varphi_0^{\leftarrow}$, will be a sentence whose truth value tells us whether φ is satisfied in λ, the only node of size 0. As pointed out in the introduction to this chapter, the same problem occurs in connection with the translation of monadic second-order variables, whose interpretations, like $\varphi^{\mathfrak{T}}$, can consist of arbitrary collections of nodes that correspond to guarded lists of different lengths. The same representation mechanism works in both cases.

With any monadic second-order variable X we associate a tuple $X^{\leftarrow} = (X^{(0)}, \ldots, X^{(m)})$, where each $X^{(k)}$ is a k-ary second-order variable. $X^{(0)}$ in particular should be regarded as a Boolean variable with values *true* or *false*. Formally one can of course replace this null-ary second-order variable by using a unary one (which is automatically guarded) whose interpretations are restricted to the empty set (for *false*) or the full universe (for *true*).

Definition 8.1.8. Let N be a set of nodes in a tree $\mathfrak{T}(\mathfrak{B})$. The *representation* of N in $\mathfrak{B}$ is $N^{\leftarrow} = (N^{(0)}, \ldots, N^{(m)})$ where $N^{(k)} = \{\pi(v) : v \in N \text{ and } |\mathfrak{A}_v| = k\} \subseteq B^k$.

Note again that, since the root λ is the only node of size 0, $N^{(0)} = \mathit{true}$ iff $\lambda \in N$.

It is known (see e.g. [50]) that we can assume w.l.o.g. that L_μ formulae are written without ν-operators, that in any formula $\mu X.\xi$ the fixed point variable X occurs in ξ only inside the scope of modal operators, and even that in each $\varphi \in \mathrm{L}_\mu$, the fixed point formulae $\mu X.\xi(X)$ themselves only occur inside the scope of modal operators.

The Translation. For every formula $\varphi \in \mathrm{L}_\mu[\tilde{\tau}]$ we now define formulae $\varphi_k^{\leftarrow}(x_1, \ldots, x_k) \in \mu\mathrm{GF}[\tau]$, one for each $k \leq m$, in which each monadic second-order variable X of φ is represented by a tuple of second-order variables $X^{\leftarrow}$.

(1) If $\varphi = P_{\mathfrak{A}}$, then set $\varphi_k^{\leftarrow} = \mathit{false}$ if $|\mathfrak{A}| \neq k$, and let $\varphi_k^{\leftarrow}(x_1, \ldots, x_k)$ otherwise be the atomic type of $(1, \ldots, k)$ in $\mathfrak{A}$, i.e., the conjunction over all atomic and negated atomic τ-formulae α such that $\mathrm{free}(\alpha) \subseteq \{x_1, \ldots, x_k\}$ and $\mathfrak{A} \models \alpha(1, \ldots, k)$. Note that for $k = 0$, this gives $\varphi_k^{\leftarrow} = \mathit{true}$ if $\mathfrak{A} = \emptyset$ and $\varphi_k^{\leftarrow} = \mathit{false}$ otherwise.

(2) If $\varphi = X$, then $\varphi_0^{\leftarrow} = \mathit{false}$, and for $1 \leq k \leq m$
$$\varphi_k^{\leftarrow}(x_1, \ldots, x_k) = X^{(k)}(x_1, \ldots, x_k).$$

(3) The translation commutes with $\neg$, $\wedge$ and $\vee$ (for each k).

(4) If $\varphi = \langle\rho\rangle\vartheta$, then
$$\varphi_k^{\leftarrow}(x_1, \ldots, x_k) = \bigvee_{\alpha} (\exists^{\neq} y_1, \ldots, y_\ell \,.\, \rho(\boldsymbol{x}, \boldsymbol{y}) \wedge \alpha(\boldsymbol{y}))\vartheta_\ell^{\leftarrow}(\boldsymbol{y}),$$
where α ranges over all guards in variables $y_1, \ldots, y_\ell$ for ℓ such that $\mathrm{dom}(\rho) \subseteq \{1, \ldots, \ell\}$.

(5) If $\varphi = \mu X.\vartheta$, then $\varphi_0^{\leftarrow}, \varphi_1^{\leftarrow}, \ldots, \varphi_m^{\leftarrow}$ are the components $[\mathbf{lfp}\; X^{(i)} \,.\, S]$ of the simultaneous least fixed point defined by the system
$$S = \begin{cases} X^{(0)} & = \vartheta_0^{\leftarrow}(X^{\leftarrow}) \\ X^{(1)} y_1 & = \vartheta_1^{\leftarrow}(y_1, X^{\leftarrow}) \\ & \vdots \\ X^{(m)} \boldsymbol{y}_m & = \vartheta_m^{\leftarrow}(\boldsymbol{y}_m, X^{\leftarrow}) \end{cases}$$
of fixed point equations, where $\boldsymbol{y}_k = (y_1, \ldots, y_k)$.

In connection with $X^{(0)}$ in (5) note that in the light of the assumption that all μ-operators and fixed-point variables occur only in the scope of modal operators, we are not ultimately interested in whether λ enters a fixed point. In this sense the Boolean component is redundant.

Theorem 8.1.9. *Let $\mathfrak{B}$ be a τ-structure, let λ be the root of $\mathfrak{T}(\mathfrak{B})$ and let ψ be a formula in $\mathrm{L}_\mu[\tilde{\tau}]$. Then $\mathfrak{B} \models \psi_0^{\leftarrow}$ iff $\mathfrak{T}(\mathfrak{B}), \lambda \models \psi$.*

Proof. Again we inductively prove a more general statement, involving free second-order variables and the semantics of ψ on the whole of $\mathfrak{T}(\mathfrak{B})$. We consider the case of just one free monadic second-order variable Y. This case is entirely indicative of the general case, but notationally somewhat lighter. Let $\varphi(Y)$ be a formula in $\mathrm{L}_\mu[\tilde{\tau}]$, N a set of nodes in $\mathfrak{T} = \mathfrak{T}(\mathfrak{B})$ with representation $N^{\leftarrow}$.

Claim 8.1.10. $(\varphi_0(N^{\leftarrow})^{\mathfrak{B}}, \dots, \varphi_m(N^{\leftarrow})^{\mathfrak{B}})$ *is the representation of* $\varphi(N)^{\mathfrak{T}}$ *in* $\mathfrak{B}$ *in the sense of Definition 8.1.8.*

The claim is proved inductively. The cases corresponding to (1) — (3) are trivial.

Consider (4) and let $\varphi = \langle\rho\rangle\vartheta$. Suppose first that $\mathfrak{T}, v \models \varphi$, where $|v| = k$, $\pi(v) = (b_1, \dots, b_k)$. We need to show that $\mathfrak{B} \models \varphi_k^{\leftarrow}(b_1, \dots, b_k)$. As $\mathfrak{T}, v \models \varphi$, there is a node w such that $(v, w) \in E_\rho$ and $\mathfrak{T}, w \models \vartheta$. Let $\pi(w) = (c_1, \dots, c_\ell)$ and note that $(v, w) \in E_\rho$ ensures that $\mathrm{dom}(\rho) \subseteq \{1, \dots, \ell\}$. Then $b_i = c_j$ for all $(i, j) \in \rho$ and, by induction hypothesis, $\mathfrak{B} \models \vartheta_\ell^{\leftarrow}(c_1, \dots, c_\ell)$. Hence $\mathfrak{B} \models \varphi_k^{\leftarrow}(b_1, \dots, b_k)$.

Conversely, suppose that $\mathfrak{B} \models \varphi_k^{\leftarrow}(b_1, \dots, b_k)$. This means that there exists a guarded list $(c_1, \dots, c_\ell)$ with $b_i = c_j$ for all $(i, j) \in \rho$ and such that $\mathfrak{B} \models \vartheta_\ell^{\leftarrow}(c_1, \dots, c_\ell)$. By the construction of $\mathfrak{T}(\mathfrak{B})$, there exists a node w such that $(v, w) \in E_\rho$ and $\pi(w) = (c_1, \dots, c_\ell)$. The induction hypothesis implies that $\mathfrak{T}, w \models \vartheta$. It follows that $\mathfrak{T}, v \models \varphi$.

For (5) finally let $\varphi = \mu X.\vartheta(X)$. Consider the stages X^α of the fixed point induction on φ,

$$\begin{array}{ll} X^0 & = \emptyset \\ X^{\alpha+1} & = \vartheta(X^\alpha) \ = \ \{v \in \mathfrak{T} \ : \ \mathfrak{T}, v \models \vartheta(X^\alpha)\} \\ X^\delta & = \bigcup_{\alpha<\delta} X^\alpha \text{ for limit ordinals } \delta. \end{array}$$

Let similarly, for the stages of the simultaneous fixed point of the system S in $\varphi^{\leftarrow}$,

$$\begin{array}{ll} (X^{(k)})^0 & = \emptyset \\ (X^{(k)})^{\alpha+1} & = \vartheta_k^{\leftarrow}((X^{\leftarrow})^\alpha) \\ (X^{(k)})^\delta & = \bigcup_{\alpha<\delta} (X^{(k)})^\alpha. \end{array}$$

By induction hypothesis, if $M^{\leftarrow}$ is the representation of M in $\mathfrak{B}$, then $(\vartheta_0^{\leftarrow}(M^{\leftarrow}), \dots, \vartheta_m^{\leftarrow}(M^{\leftarrow}))$ is the representation of $\vartheta(M)^{\mathfrak{T}}$ in $\mathfrak{B}$. By induction on α, this implies that the stage $(X^{\leftarrow})^\alpha$ of $[\mathbf{lfp}\ X^{\leftarrow}.\, S]$ represents in $\mathfrak{B}$ the stage X^α of $\mu X.\vartheta$ in $\mathfrak{T}$. Hence the same is true for the least fixed points. □

We are now in a position to prove our main theorem for sentences.

Theorem 8.1.11. *Every sentence in* GSO *that is invariant under guarded bisimulation is equivalent to a sentence in* μGF.

Proof. Let $\psi \in \mathrm{GSO}[\tau]$ be invariant under guarded bisimulation and let $\psi^{\rightarrow}(x)$ be its translation into $\mathrm{MSO}[\tilde{\tau}]$. By Corollary 8.1.7 $\psi^{\rightarrow}(x)$ is bisimulation-invariant on consistent trees. Recall that the consistency condition for trees can be formulated by a monadic second-order (in fact even first-order) sentence γ, which is bisimulation invariant with respect to all trees. As a consequence, the formula $(\gamma \rightarrow \psi^{\rightarrow})(x)$ is bisimulation invariant on arbitrary trees. By the Janin-Walukiewicz Theorem, Theorem 8.1.1 above, there exists an equivalent formula φ in the μ-calculus. Let $\varphi_0^{\leftarrow}$ be the 0-component of its translation into $\mu\mathrm{GF}[\tau]$. Putting everything together, we have the following chain of equivalences.

$$\mathfrak{B} \models \psi \Leftrightarrow \mathfrak{D}(\mathfrak{T}(\mathfrak{B})) \models \psi \Leftrightarrow \mathfrak{T}(\mathfrak{B}) \models \psi^{\rightarrow}(\lambda) \Leftrightarrow \mathfrak{T}(\mathfrak{B}), \lambda \models \varphi \Leftrightarrow \mathfrak{B} \models \varphi_0^{\leftarrow}$$

The first equivalence uses Lemma 4.5.4 and the guarded bisimulation invariance of ψ; the second one is an application of Theorem 8.1.5; the third equivalence follows from the Janin Walukiewicz Theorem; the fourth is an application of Theorem 8.1.9. □

Remark. Again, there are straightforward extensions of all the results in this section at least to the case of structures with distinguished guarded lists of parameters, and to formulae whose free variables get interpreted by guarded lists. Compare in particular the claims 8.1.6 and 8.1.10.

8.2 Back and Forth with Parameters

We look at extensions of our main results to the analysis of formulae with free variables, and to structures with parameters. The most natural case, in the spirit of the guarded scenario, is that of guarded parameter tuples. Most of our techniques and results, however, can meaningfully be generalised even to the case of formulae in arbitrary tuples of variables, and correspondingly to structures with arbitrary parameter tuples rather than guarded ones.

As pointed out above, a special case of parameter tuples, namely that of guarded lists, was actually implicit in the basic case as treated in Sections 4.4 and 8.1. One can put a distinguished guarded list from $\mathfrak{B}$ at the root of the tree representation $\mathfrak{T}(\mathfrak{B})$, rather than using the empty guarded list as a default as we did above. For guarded lists of parameters – and for formulae whose free variables get interpreted as guarded lists – the above treatment in fact goes through almost verbatim. The claims 8.1.6 and 8.1.10 that were established in the proofs of Theorems 8.1.9 and 8.1.9, in particular, already deal with formulae whose free variables refer to guarded lists.

This approach immediately yields the further generalisation to the important case of variable-guarded formulae. We first describe the simple argument required for this extension, as a shortcut to the variable-guarded case for our main theorem. The most general case of formulae, whose free variables refer to arbitrary parameter tuples, is then discussed in Sections 8.2.1, 8.2.2 and 8.2.3 below, thereby also indicating an alternative approach to the case of variable-guarded formulae.

Recall that a formula $\varphi(\boldsymbol{x})$ is variable-guarded if it is logically equivalent to $\varphi(\boldsymbol{x}) \wedge \mathbb{G}(\boldsymbol{x})$. Also recall from Section 3.2 that the GF formula $\mathbb{G}(\boldsymbol{x})$ is a disjunction over formulae $\xi_{\eta\alpha}(\boldsymbol{x})$ specifying the equality type $\eta(\boldsymbol{x})$ of $\boldsymbol{x}$ and an extension of $\boldsymbol{x}$ to an α-atom. Putting $\xi_{\eta\alpha}(\boldsymbol{x})$ into GF_0 normal form we may assume

$$\xi_{\eta\alpha}(\boldsymbol{x}) = \eta(\boldsymbol{x}) \wedge \big(\exists^{\neq}\boldsymbol{y}.\rho(\boldsymbol{x},\boldsymbol{y}) \wedge \alpha(\boldsymbol{y})\big)\alpha(\boldsymbol{y}).$$

For a variable-guarded GSO formula $\varphi(\boldsymbol{x})$, we therefore find that it is equivalent to a disjunction of formulae of the form $\eta(\boldsymbol{x})\wedge\big(\exists^{\neq}\boldsymbol{y}.\rho(\boldsymbol{x},\boldsymbol{y})\wedge\alpha(\boldsymbol{y})\big)\tilde{\varphi}(\boldsymbol{y})$, where the $\tilde{\varphi}(\boldsymbol{y})$ are obtained through suitable variable substitutions in $\varphi(\boldsymbol{x})$ (not all y necessarily free in $\tilde{\varphi}(\boldsymbol{y})$). The new variables $\boldsymbol{y}$ in $\tilde{\varphi}(\boldsymbol{y})$ stand for guarded lists. Assuming this normal form for variable-guarded $\varphi(\boldsymbol{x}) \in \mathrm{GSO}$, and using the formalisation of guarded bisimulation with parameters as discussed in connection with Definition 3.4.1, we find that $\varphi(\boldsymbol{x})$ is invariant under guarded bisimulation if, and only if, each $\tilde{\varphi}(\boldsymbol{y})$ is invariant under guarded bisimulation. The case of formulae whose free variables stand for guarded lists, and the translation machinery treated in Sections 8.2.1 and 8.1 are therefore sufficient to yield the following strengthening of Theorem 8.1.11.

Theorem 8.2.1. *Every variable-guarded formula of* GSO *that is invariant under guarded bisimulation is equivalent to a formula in* μGF.

8.2.1 Structures and Trees

We outline the modifications required in the back and forth translations of Sections 4.4 and 8.1 in order to incorporate arbitrary tuples of parameters right from the beginning. In particular we extend the correspondence between relational structures and suitable trees to incorporate arbitrary parameter tuples.

To see the motivation behind the following stipulations, consider the variation in the guarded bisimulation game that is required to capture equivalence with respect to arbitrary tuples of parameters. This variation merely concerns the start position in the game. For the game on $\mathfrak{A}, \boldsymbol{a}$ and $\mathfrak{B}, \boldsymbol{b}$, we initially put pebbles on $\boldsymbol{a}$ and $\boldsymbol{b}$. Eve has lost if $\boldsymbol{a} \mapsto \boldsymbol{b}$ is not a partial isomorphism. Otherwise, i.e., if the initial pebbling describes a partial isomorphism, we proceed as before. Adam can choose either structure and remove any number of pebbles from that structure, then put pebbles so as to end up with a pebbling of a guarded list; Eve must respond by first removing corresponding pebbles in the opposite structure, then putting pebbles in such a way that the resulting correspondence is again a partial isomorphism. Note that all information about the initial parameters is lost after this first round, as far as it is not carried over via pebbles that are left fixed by Adam in his first move.

Recall the vocabulary $\tilde{\tau}$ used for the trees $\mathfrak{T}(\mathfrak{B})$ associated with τ-structures $\mathfrak{B}$: $\tilde{\tau}$ has monadic predicates $P_{\mathfrak{A}}$ for all $\mathfrak{A} \in S$, S the set of guarded τ-structures on universes $\{1, \dots, k\}$, k bounded by the width m of τ, and binary relations E_ρ for $\rho \in F$, F the set of partial bijections on $\{1, \dots, m\}$. In order to treat tuples of n distinct parameters $\boldsymbol{b} = (b_1, \dots, b_n)$, which need not form a guarded tuple, we expand $\tilde{\tau}$ to $\tilde{\tau}_n$ by adding further predicates $P_{\mathfrak{A}/n}$ and $E_{\rho/n}$ for $\mathfrak{A} \in S_n$ and $\rho \in F_n$ where S_n is the set of all τ-structures on universe $\{1, \dots, n\}$, F_n the set of partial 1-1 maps $\rho\colon \{1, \dots, m\} \to \{1, \dots, n\}$.

Let $\mathfrak{B}$ be a τ-structure, $\boldsymbol{b} = (b_1, \dots, b_n)$ a tuple of *distinct* parameters in $\mathfrak{B}$. With $\mathfrak{B}, \boldsymbol{b}$ we associate a tree $\mathfrak{T}(\mathfrak{B}, \boldsymbol{b})$ of type $\tilde{\tau}_n$, obtained as an unravelling of a transition system $\mathfrak{K}(\mathfrak{B}, \boldsymbol{b})$, which captures the guarded bisimulation game on $\mathfrak{B}$ with initial position $\boldsymbol{b}$.

$\mathfrak{K}(\mathfrak{B}, \boldsymbol{b})$ has nodes for all guarded lists g in $\mathfrak{B}$ and one extra distinguished node p for the parameter tuple $\boldsymbol{b}$. We use a separate node p to represent $\boldsymbol{b}$ even if $\boldsymbol{b}$ happens to be a guarded list itself.

The restriction of $\mathfrak{K}(\mathfrak{B}, \boldsymbol{b})$ to the set of nodes $g \neq p$ is isomorphic with the old $\mathfrak{K}(\mathfrak{B})$. In the neighbourhood of the distinguished node p we interpret the new predicates. Put $p \in P_{\mathfrak{A}/n}$ for that $\mathfrak{A} \in S_n$ that is isomorphic with $\mathfrak{B}|_{\{b_1, \dots, b_n\}}$ via $i \mapsto b_i$, and let $(p, g) \in E_{\rho/n}$ if g is a guarded list in $\mathfrak{B}$ such that ρ describes a partial isomorphism between $\mathfrak{B}|_g$ and $\mathfrak{B}|_{\{b_1, \dots, b_n\}}$. Note that we put no in-going edges to p. These choices capture the special role that the parameters $\boldsymbol{b}$ play as initial conditions in the bisimulation game. In the first round the game passes to guarded lists extending a chosen subtuple of $\boldsymbol{b}$, and the identity of that subtuple needs to be preserved by Eve; any components of the parameter tuple that are not kept in this move lose their special identity.

For $\mathfrak{T}(\mathfrak{B}, \boldsymbol{b})$ we take the unravelling of $\mathfrak{K}(\mathfrak{B}, \boldsymbol{b})$ from the distinguished node p. The root λ of $\mathfrak{T}(\mathfrak{B}, \boldsymbol{b})$ represents $\boldsymbol{b}$ and is uniquely distinguished by being coloured by a monadic predicate $P_{\mathfrak{A}/n}$ and by having outgoing edges of types $E_{\rho/n}$.

The corresponding class of consistent $\tilde{\tau}_n$-trees $\mathfrak{T}$ is defined analogously to the class of consistent $\tilde{\tau}$-trees, with the additional stipulations that

(a) There is a unique $\mathfrak{A} \in S_n$ such that the root of $\mathfrak{T}$ is in $P_{\mathfrak{A}/n}$; we denote it by $\mathfrak{A}_\lambda$.

(b) All edges from the root are $E_{\rho/n}$-edges for $\rho \in F_n$, and such edges only occur from the root; if $(\lambda, v) \in E_{\rho/n}$, then ρ is a partial isomorphism between $\mathfrak{A}_v$ and $\mathfrak{A}_\lambda$.

(c) If $\boldsymbol{c} = (c_1, \ldots, c_k)$ is a guarded list in $\mathfrak{A}_\lambda$, then there is a node w such that $(\lambda, w) \in E_{\rho/n}$ for $\rho = \{(i, c_i) \colon 1 \leq i \leq k\}$.

It is clear how to construct a τ-structure $\mathfrak{D}(\mathfrak{T})$ from a consistent $\tilde{\tau}_n$-tree $\mathfrak{T}$, in complete analogy with the procedure described in Section 4.4. The distinguished parameters $\boldsymbol{d} = (d_1, \ldots, d_n)$ for $\mathfrak{D}(\mathfrak{T})$ and their quantifier free type are extracted from $\mathfrak{A}_\lambda$.

With these modifications we find that all the results of Section 4.4 remain valid in the extended setting. In particular,

- $\mathfrak{B}, \boldsymbol{b} \sim_g \mathfrak{B}', \boldsymbol{b}'$ iff $\mathfrak{T}(\mathfrak{B}, \boldsymbol{b}) \sim \mathfrak{T}(\mathfrak{B}', \boldsymbol{b}')$; cf. Lemma 4.4.4.

- Consistency is first-order definable and bisimulation invariant within the class of $\tilde{\tau}_n$-trees; cf. Lemma 4.5.2.

- For all $\mathfrak{B}, \boldsymbol{b}$: $\mathfrak{D}(\mathfrak{T}(\mathfrak{B}, \boldsymbol{b})), \boldsymbol{d} \sim_g \mathfrak{B}, \boldsymbol{b}$; cf. Lemma 4.5.4.

- For all consistent $\tilde{\tau}_n$-trees $\mathfrak{T}$ and $\mathfrak{T}'$: if $\mathfrak{T} \sim \mathfrak{T}'$, then $\mathfrak{D}(\mathfrak{T}), \boldsymbol{d} \sim_g \mathfrak{D}(\mathfrak{T}'), \boldsymbol{d}'$; cf. Proposition 4.5.5.

We now address the issue of translating formulae between the guarded world of τ-structures with parameters and the modal world of their tree representations.

8.2.2 From GSO to MSO

In order to incorporate the not necessarily guarded tuple of parameters it is convenient to augment the GSO_0 normal form by a third set of new variables, $z_1, \dots, z_n$, which are only to be used as free variables for the parameters. A GSO-formula is in extended GSO_0 normal form if its free variables are among the z_i and if all its subformulae beginning with (guarded) first-order quantifications are in ordinary GSO_0 normal form. W.l.o.g. we consider only GSO formulae in this normal form which moreover require the free variables $\boldsymbol{z} = (z_1, \dots, z_n)$ to stand for pairwise distinct elements through conjuncts $z_i \neq z_j$ for $i \neq j$, so that their semantics fits the above format of τ-structures $\mathfrak{B}$ with tuples $\boldsymbol{b} = (b_1, \dots, b_n)$ of pairwise distinct parameters.

For a second-order variable Z of arity r, ranging over guarded relations over the τ-structures $\mathfrak{B}$, we again introduce a tuple $Z^{\rightarrow}$ of monadic second-order variables. These range over subsets of the $\mathfrak{T}(\mathfrak{B}, \boldsymbol{b})$ that do not contain the root. This latter condition is to be added to the correctness criterion, since the root of $\mathfrak{T}(\mathfrak{B}, \boldsymbol{b})$ does not in general represent a guarded tuple. With this modification Lemma 8.1.3 carries over directly.

The first-order formula F-part$(z, \boldsymbol{W})$ of Lemma 8.1.4 needs to be modified in order to take edges $E_{\rho/n}$ into account. It is convenient for the further translation to choose the $\boldsymbol{W}$ such that none of the W^k_ρ contain the root, and to stipulate this condition in F-part$(z, \boldsymbol{W})$. The compatibility conditions imposed by links through the root, then, have to be phrased in terms of pairs of nodes (both adjacent to the root) linked via $E_{\rho/n} \circ (E_{\rho'/n})^{-1}$ combinations.

With these straightforward modifications, Lemma 8.1.4 carries over unchanged. For the translation from our GSO-formulae $\varphi(\boldsymbol{z})$ to its MSO-counterpart $\varphi^{\rightarrow}(z)$, we may now use exactly the same translation as in Section 8.1, with the minor difference that the quantifier free subformulae in variables $\boldsymbol{z}$ are translated into disjunctions over $P_{\mathfrak{A}/n}$ rather than $P_{\mathfrak{A}}$. We then obtain the analogue of Theorem 8.1.5, to the effect that

- for all consistent $\tilde{\tau}_n$-trees $\mathfrak{T}$ with root λ and for all GSO-formulae $\varphi(\boldsymbol{z})$ in the normal form as considered above: $\mathfrak{D}(\mathfrak{T}), \boldsymbol{d} \models \varphi(\boldsymbol{z})$ iff $\mathfrak{T} \models \varphi^{\rightarrow}(\lambda)$.

As before, it follows that guarded bisimulation invariant φ translate into bisimulation invariant $\varphi^{\rightarrow}$, which by the Janin Walukiewicz Theorem are therefore equivalently expressible in the μ-calculus.

8.2.3 From L_μ to μGF

For the converse translation of formulae, which takes us from the trees $\mathfrak{T}(\mathfrak{B}, \boldsymbol{b})$ back to the $\mathfrak{B}, \boldsymbol{b}$, we now need to deal with one additional issue. This concerns the fact that an arbitrary monadic second-order or L_μ formula over the tree may involve monadic second-order variables or monadic fixed points that contain the root. For L_μ we can simply eliminate these. Indeed, any L_μ formula over trees is logically equivalent to a Boolean combination of atomic propositions and L_μ formulae of the form $\Diamond\xi$. We may appeal to the same restricted syntax as used for the proof of Theorem 8.1.9, cf. [50]. Note that the evaluation of an L_μ formula $\Diamond\xi$ at the root of a tree $\mathfrak{T}$ only requires the evaluation of ξ in all subtrees rooted at children of the root. For the $\mathfrak{T}(\mathfrak{B}, \boldsymbol{b})$ this means that the translations provided in Section 8.1, cf. in particular claim 8.1.10 in the proof of Theorem 8.1.9, can be used without changes for such ξ.

From the resulting tuple of formulae $\xi_k^{\leftarrow}(x_1, \ldots, x_k)$ in μGF we may then obtain the correct translation of $\langle\rho/n\rangle\xi$ — where $\langle\rho/n\rangle$ is the modal operator associated with $E_{\rho/n}$ — with $\big(\langle\rho/n\rangle\xi\big)^{\leftarrow}(\boldsymbol{z})$ defined as

$$\textstyle\bigvee_\alpha \big(\exists^{\neq}(x_1, \ldots, x_k).\rho(\boldsymbol{z}, x_1, \ldots, x_k) \wedge \alpha(x_1, \ldots, x_k)\big)\xi_k^{\leftarrow}(x_1, \ldots, x_k),$$

where α ranges over all guards in variables $x_1, \ldots, x_k$ for all k such that $\mathrm{dom}(\rho) \subseteq \{1, \ldots, k\}$.
We thus obtain the following extension of Theorem 8.1.9:

- For any τ-structure $\mathfrak{B}$ with n pairwise distinct parameters $\boldsymbol{b}$, and for any formula ψ in $\mathrm{L}_\mu[\tilde{\tau}_n]$: $\mathfrak{B} \models \psi^{\leftarrow}(\boldsymbol{z})$ iff $\mathfrak{T}(\mathfrak{B}, \boldsymbol{b}), \lambda \models \psi$.

Putting the results together, yields the following variant of Theorem 8.1.11 for arbitrary formulae.

Theorem 8.2.2. *Every formula of* GSO *that is invariant under guarded bisimulation is equivalent to a formula in* μGF.

Recall that Proposition 3.3.1 states that any formula of μGF is logically equivalent to one in which all fixed points are strictly guarded, i.e., in which μ or ν operators are only applied to variable-guarded formulae.

Corollary 8.2.3. μGF $\subseteq$ GSO *for formulae in arbitrary tuples of free variables.*

Together with Theorem 8.2.2 this finally gives the full content of our main theorem for arbitrary formulae. A formula of GSO is invariant under guarded bisimulation if, and only if, it is equivalent to a formula in μGF.

8.3 Bisimulation Invariant MSO*

In this section we take a small stab in the direction of a finite model theory version of the Janin-Walukiewicz characterisation theorem. The first characterisation result in the finite was obtained by Rosen [55].

Theorem 8.3.1 (Rosen). *A class of finite graphs is* ML*-definable iff it is definable by a* FO*-formula that is bisimulation invariant on the class of finite graphs.*

For guarded logics, so far only the case of width-2 has been solved [52]. This is a further indication that the restriction to finite structures has a greater impact on guarded bisimulation than on modal bisimulation, cf. Chapter 7.

Theorem 8.3.2 (Otto). *A class of finite graphs is* GF*-definable iff it is definable by a* FO*-formula that is guarded bisimulation invariant on the class of finite graphs.*

Starting point is the simple observation that MSO and GSO contain formulae that are invariant for (guarded) bisimulation on the class of finite structures, but not (guarded) bisimulation invariant in general.

Example 8.3.3. Consider any vocabulary of graphs that contains one or more edge relations. Let $\psi \in$ MSO express that there is an infinite binary tree. In particular ψ is an infinity axiom. Let $\varphi \in$ ML be the simple requirement that there is a successor node, a bisimulation invariant statement. Combine the two formulae as $\vartheta(x) = \psi \vee (\neg\psi \wedge \varphi^\circ(x))$. On finite structures, ϑ behaves like φ. On infinite structures, $\vartheta(x)$ is not bisimulation invariant. To see this, take an isolated point as interpretation for x. The truth value of $\vartheta(x)$ then depends on whether an additional connected component is a binary tree, or merely something else infinite. So ϑ is only bisimulation invariant in the sense of finite model theory.

We consider the logic MSO*, a variant of MSO presented in [15], where formulae are evaluated over the unravellings of transition systems. The syntax of MSO* formulae is the same as for MSO. We will write φ, resp. φ^*, for the (syntactically) same formula to denote whether it is interpreted as formula of MSO or MSO*. Correspondingly, as we are back in the realm of modal logics, the structures considered are again labelled graphs.

Semantics of MSO*. Given a graph $\mathfrak{G}$ with node $v \in \mathfrak{G}$, a formula $\varphi^*(x) \in$ MSO* holds in v, denoted $\mathfrak{G}, v \models \varphi^*$, iff $\mathfrak{T}(\mathfrak{G}, v) \models \varphi$.

8.3.1 Automata for MSO

We use the following automata-theoretic characterisation of MSO and L_μ over trees [44, 62]. The tree automata considered belong to the class of alternating ω-tree automata with parity acceptance conditions. They work on infinite trees and have the form $\mathcal{A} = (Q, \Sigma, q_0, \delta, \Omega)$, where

- Q is a finite set of states,
- Σ is a finite input alphabet,
- $q_0 \in Q$ is the initial state,
- δ is the transition function and
- $\Omega : Q \to \mathbb{N}$ is the parity function.

For each pair $(q, a) \in Q \times \Sigma$, $\delta(q, a)$ is a disjunction of formulae of one of the two forms,

$$\exists \boldsymbol{x}.\big(\operatorname{diff}(\boldsymbol{x}) \wedge \bigwedge_{j=1}^{k} q_{i_j}(x_j) \wedge \forall z(\operatorname{diff}(z, \boldsymbol{x}) \to \bigvee_{i=1}^{\ell} q_{j_i}(z))\big) \tag{8.1}$$

$$\exists \boldsymbol{x}.\big(\bigwedge_{j=1}^{k} q_{i_j}(x_j) \wedge \forall z \bigvee_{i=1}^{\ell} q_{j_i}(z)\big) \tag{8.2}$$

where $|\boldsymbol{x}| = k$, all $q_i \in Q$, and $\operatorname{diff}(\boldsymbol{y})$ states that all arguments are distinct.

If $\mathcal{A}$ only uses formulae of the form (8.2), we call it a μ-automaton; if it only uses (8.1) it is an MSO-automaton. In the following, any automaton is either a μ-automaton or an MSO-automaton. These automata operate on a kind of normal form of trees, rather than trees as per Definition 2.3.1.

Definition 8.3.4. A tree $\mathfrak{T} = (V, (P_b)_{b \in B}, (E_a)_{a \in A})$ is *normal*, if $|A| = 1$ and $|\{b \ : \ s \in P_b\}| = 1$ for all $s \in V$.

If $\mathfrak{T}$ is a normal tree we write E for the single edge relation of $\mathfrak{T}$. For every $s \in V$ we write b_s for the (unique) $b \in B$ with $\mathfrak{T} \models P_b(s)$ and, in this case, let $P_s = P_b$. Automata for trees with atomic properties B have input alphabet $\Sigma = B$. In this context, when a node $v \in V$ is used in place of a letter $b \in \Sigma$, this is a shortcut for b_v.

For every signature of transition systems $\tau = \{(P_b)_{b \in B}, (E_a)_{a \in A}\}$ we define the derived signature $\tau' = \{(P_b)_{b \in B'}, E\}$ where $B' = \mathcal{P}(B) \times A$. There is an obvious bijection between the class of τ-trees and the class of normal τ'-trees. In the one direction, a node v that is reachable via E_a from its parent, and where P_b holds at v iff $b \in B_v$ for some suitable $B_v \subseteq B$, will get the label $(\{P_b : b \in B_v\}, a)$. In the other direction, the first component determines which P_b hold at the node in question, and the second component gives the edge label. The second component is ignored at the root, which has no predecessor and hence no incoming edge.

Since the automata can only digest normal trees, we assume forthwith that any considered class of trees consists only of normal trees. With the above translation this is a mere technicality, since every class of trees can be brought into that form. The semantics of automata is defined via the usual notion of a run.

A *run* of $\mathcal{A}$ over $\mathfrak{T}$ is a further tree $\rho = (W, F)$ where $W \subseteq V^{\mathfrak{T}} \times Q^{\mathcal{A}}$ and $F \subseteq W \times W$ such that $(\lambda, q_0) \in W$ and the following hold.

- The canonical projection $\pi : W \to V$ is a graph homomorphism from ρ to the $\{E\}$-reduct of $\mathfrak{T}$. That is, $((s,q),(s',q')) \in E^\rho$ only if $(s,s') \in E$.
- For every $(s,q) \in W$, the Q-structure $\mathfrak{A}(s,q)$ with universe $\{s' : (s,s') \in E\}$ and monadic predicates $q' \in Q$ interpreted as $\{s' : ((s,q),(s',q')) \in F\}$ has to satisfy $\mathfrak{A}(s,q) \models \delta(q, b_s)$.

A run ρ of $\mathcal{A}$ over $\mathfrak{T}$ is *accepting* if for each infinite path $\alpha = (\lambda, q_0)(s_1, q_1)\cdots$ in ρ it is the case that $\min\{n : \exists^\omega i.\Omega(q_i) = n\}$ is even. A tree is accepted by $\mathcal{A}$ if there is an accepting run of $\mathcal{A}$ over $\mathfrak{T}$. Furthermore, ρ is *pruned* if it has the same "format" as $\mathfrak{T}$, i.e. if for every $v \in V$ there is precisely one $q \in Q$ with $(v,q) \in W$.

Given an automaton $\mathcal{A}$ we denote by $L(\mathcal{A})$ the set of trees accepted by $\mathcal{A}$, also called the tree language recognised by $\mathcal{A}$.

Our considerations for MSO* will build on manipulating trees and using a given accepting run as template for an accepting run on the new tree. Towards this end we make the following technical observations.

Lemma 8.3.5. *For every* MSO-*automaton $\mathcal{A}$, every tree $\mathfrak{T} \in L(A)$ and every accepting run ρ of $\mathcal{A}$ on $\mathfrak{T}$ there is a subtree ρ' of ρ that is pruned and an accepting run of $\mathcal{A}$ on $\mathfrak{T}$.*

Proof. We inductively define a sequence of runs ρ_i, $i < \omega$, where the first i levels, where the root is at level 1, of each $\rho_i \subseteq \rho$ satisfy the sparsity condition of a pruned run. Start with $\rho_1 = \rho$; ρ only has one root (λ, q_0).

For the inductive step suppose that ρ_i was already constructed. For every $(s, p) \in W$ on level i that has successors that violate the requirement, i.e. successors $(t, q) \in W$ and $(t, q') \in W$ for some $q \neq q'$, proceed as follows.

Choose δ' as one of the possibly several disjuncts of $\delta(p, s)$ that is satisfied in the accepting run ρ, i.e. $\mathfrak{A}(s, p) \models \delta'$. We can write δ', a formula of the form (8.1), as $\exists \boldsymbol{x}.(\gamma(\boldsymbol{x}) \wedge \forall z.\zeta(\boldsymbol{x}, z))$ and choose a concrete sequence of, necessarily distinct, elements $\boldsymbol{v} \in \mathfrak{A}(s, p)$ such that $\mathfrak{A}(s, p) \models \gamma(\boldsymbol{v}) \wedge \forall z.\zeta(\boldsymbol{v}, z)$. For each $v \in \boldsymbol{v}$ the γ-part of such an MSO-formula contains precisely one statement $q_v(v)$, $q_v \in Q$.

Taking this further, for every $v \in \mathfrak{A}(s, p)$, $v \notin \boldsymbol{v}$ we can choose one of the possibly several disjuncts $q_v(z)$ of ζ that is satisfied by v in ρ, i.e. $\mathfrak{A}(s, p) \models q_v(v)$ — which is the case iff $((p, s), (q_v, v)) \in F$.

All in all, for every $v \in \mathfrak{A}(s, p)$ there is one q_v such that the structure $\mathfrak{A}'$ obtained from $\mathfrak{A}(s, p)$ by interpreting each q_v as $\{v\}$, and all other q-predicates as $\emptyset$, is a model of δ', and hence $\mathfrak{A}' \models \delta(p, s)$. The actual pruning consists of deleting all successor nodes (v, q) of (s, p) where $q \neq q_v$. This obviously has the desired effect that every successor t of s in $\mathfrak{T}$ occurs in no more than one node of the form (t, q) in ρ.

Simultaneous application of the above construction to all nodes on level i yields the intermediate run ρ_i, and we finally obtain ρ' defined by $\rho' = \bigcap \rho_i$. Since $\rho' \subseteq \rho$, it satisfies the homomorphism property of a run. By the same argument the global parity acceptance condition for the paths of ρ' is satisfied. The local condition that $\mathfrak{A}(s, p) \models \delta(p, s)$ for every node $(s, p) \in \rho'$ was ensured by the construction. So ρ' is a pruned accepting run of $\mathcal{A}$ on $\mathfrak{T}$. □

Corollary 8.3.6. *For every* MSO*-automaton $\mathcal{A}$, every tree $\mathfrak{T} \in L(\mathcal{A})$ and every accepting run ρ of $\mathcal{A}$ on $\mathfrak{T}$ there is a finitely branching $\mathfrak{T}' \subseteq \mathfrak{T}$ such that the restriction of ρ to $\mathfrak{T}'$ is an accepting run of $\mathcal{A}$.*

Proof. Let $(s, p) \in \rho$ and let $\boldsymbol{v} \in \mathfrak{A}(s, p)$ be a sequence of tuples for the existentially quantified variables of $\delta(p, s)$ as in the proof of Lemma 8.3.5 above. Close inspection of the procedure there shows that all successors (v, q) of (s, p) in ρ where $v \notin \boldsymbol{v}$ can be deleted while retaining the properties of an accepting run. The obtained ρ' is finitely branching, and the desired $\mathfrak{T}'$ is the projection of $\mathfrak{T}$ to $\{v \in V \ : \ (v, q) \in \rho', q \in Q\}$. □

The following theorems sum up the automata-theoretic framework used in the proof of Theorem 8.1.1.

Theorem 8.3.7 (Walukiewicz). *A tree language L is* MSO*-definable iff $L = L(\mathcal{A})$ for an* MSO*-automaton $\mathcal{A}$.*

Theorem 8.3.8 (Janin, Walukiewicz). *A tree language is* L_μ*-definable iff $L = L(\mathcal{A})$ for some μ-automaton $\mathcal{A}$.*

We immediately obtain the following corollary to Theorem 8.1.1 for MSO^*.

Corollary 8.3.9. *Every formula in* MSO^* *that is invariant under bisimulation is equivalent to a formula in* L_μ*.*

Proof. If $\varphi^* \in \mathrm{MSO}^*$ is bisimulation-invariant in general, then $\varphi \in \mathrm{MSO}$ is bisimulation invariant on the class of all trees. Hence, on trees, φ is equivalent to an L_μ-formula ψ. Then, for all transition systems $\mathfrak{K}$, and all nodes $v \in \mathfrak{K}$, by definition φ^* holds at v in $\mathfrak{G}$ iff φ holds at the root of $\mathfrak{T}(\mathfrak{K}, v)$. Using the equivalence on trees, the latter is the case iff ψ holds at the root of $\mathfrak{T}(\mathfrak{K}, v)$, which, since L_μ is bisimulation invariant and $\mathfrak{T}(\mathfrak{K}, v) \sim \mathfrak{K}, v$, is iff ψ holds at v in $\mathfrak{K}$. □

8.3.2 Finite Model Theory and MSO^*

Theorem 8.3.10. *Every formula in* MSO^* *that is invariant under bisimulation on the class of finite graphs is equivalent to a formula in* L_μ*.*

The theorem as such is not particularly thrilling. What is more interesting is the following, stronger result that it is a corollary of.

Lemma 8.3.11. *Every formula in* MSO^* *that is invariant under bisimulation on the class of finite graphs is bisimulation-invariant on the class of all transition systems.*

Proof. Let $\varphi^* \in \mathrm{MSO}^*$ be bisimulation-invariant in the finite. Assume to the contrary that there are infinite structures $\mathfrak{K}_1$, $\mathfrak{K}_2$ and nodes $v_1 \in \mathfrak{K}_1$, $v_2 \in \mathfrak{K}_2$ such that $\mathfrak{K}_1, v_1 \sim \mathfrak{K}_2, v_2$ and $\mathfrak{K}_1 \models \varphi^*(v_1)$, $\mathfrak{K}_2 \not\models \varphi^*(v_2)$, i.o.w. $\mathfrak{T}_1 \models \varphi_1$, $\mathfrak{T}_2 \models \varphi_2$ for $\mathfrak{T}_1 = \mathfrak{T}(\mathfrak{K}_1, v_1)$, $\mathfrak{T}_2 = \mathfrak{T}(\mathfrak{K}_2, v_2)$, $\varphi_1 = \varphi$, $\varphi_2 = \neg\varphi_1$.

Let $\mathcal{A}_i$ be the automaton for φ_i and let ρ_i be an accepting run of $\mathcal{A}_i$ on $\mathfrak{T}_i$. By Lemma 8.3.5 and Corollary 8.3.6 the $\mathfrak{T}_i$ and ρ_i can be chosen finitely branching, and the ρ_i can be chosen pruned.

The nodes of ρ_i are pairs of the form (s, q), where $s \in V_i$ is a node of $\mathfrak{T}_i$ and $q \in Q_i$ is a state of $\mathcal{A}_i$. With ρ_i being pruned, the second component of these tuples is redundant; there is a function $q_i : V_i \to Q_i$ such that every node of ρ_i has the form $(s, q_i(s))$. Instead of talking about ρ_i, we can pretend that the nodes of $\mathfrak{T}_i$ are, in addition to their respectively associated P_s, labelled by their corresponding state $q_i(s)$. Further, for the construction we will require every node to be labelled by a set of states of $\mathcal{A}_1$, and a set of states of $\mathcal{A}_2$, the choice of which will be detailed below. The result is effectively a $\tau_i^+ = \{P_b\}_{b\in B} \times Q_i \times \mathcal{P}(Q_1) \times \mathcal{P}(Q_2)$-structure $\mathfrak{T}_i^+$. We can now talk about $\mathfrak{T}_i$, which is the actual input tree, ρ_i, which is the run of $\mathcal{A}_i$ on $\mathfrak{T}_i$ (or, equivalently, $\mathfrak{T}_i$ with the corresponding states as additional label) and $\mathfrak{T}_i^+$, which in addition to ρ_i has the pair of sets of states as further label. Conversely, every $\mathfrak{T}_i^+$ implicitly describes a normal tree of the same format as the input, and a run of $\mathcal{A}_i$ on that tree.

By assumption, the trees $\mathfrak{T}_1$ and $\mathfrak{T}_2$ are bisimilar. Let $Z : \mathfrak{T}_1 \sim \mathfrak{T}_2$ be the maximal two-way bisimulation between $\mathfrak{T}_1$ and $\mathfrak{T}_2$. That is, Z is *minimal* in the sense that it is a union of cross products of the nodes on bisimilar paths starting at the respective root nodes, and *maximal* in the sense that it is the union of all such products. That is, Z is the maximal bisimulation that connects only nodes at the same distance from their respective roots.

Towards the additional set labels, consider a node $s \in V_1$ and let $T_{2,s} \subset \mathfrak{T}_2$ be the image of s under Z, and let $T_{1,s} \subset \mathfrak{T}_1$ be the inverse image of the elements of $T_{2,s}$ under Z. Next define $Q_{1,s} = \{q \in Q_1 \ : \ q = q_1(v), v \in T_{1,s}\}$ and $Q_{2,s} = \{q \in Q_2 \ : \ q = q_2(v), v \in T_{2,s}\}$. Then the label of s in $\mathfrak{T}_1^+$ is $(P_s, q_1(s), Q_{1,s}, Q_{2,s})$.

Similarly ρ_2 is extended to a $\tau_2^+ = \{P_b\}_{b\in B} \times Q_2 \times \mathcal{P}(Q_1) \times \mathcal{P}(Q_2)$-structure $\mathfrak{T}_2^+$. We now take the finitely branching $\mathfrak{T}_i^+$ and transform them into finite $\mathfrak{T}_i^-$ such that the unravelling of $\mathfrak{T}_i^-$ describes an (infinite) accepting run of $\mathcal{A}_i$ on the unravelling of the (finite) structure described by $\mathfrak{T}_i^-$. We will write s for nodes of $\mathfrak{T}_i$, and s^+ for the same node in $\mathfrak{T}_i^+$.

The following procedure is based on certain properties of the grouping of nodes as implied by the sets $T_{i,s}$. Since Z is a two-way bisimulation, and $Z \supset T_{1,s} \times T_{2,s}$ for every $s \in V_1 \cup V_2$, every group $T_{1,s} \cup T_{2,s}$ contains nodes that are at the same distance from the root node λ_1 or λ_2, respectively. The set of all groups $\{T_{1,s} \cup T_{2,s} \ : \ s \in V_1\} = \{T_{1,s} \cup T_{2,s} \ : \ s \in V_2\}$ is a partition of $V_1 \cup V_2$. By construction, the τ^+-labels within a group only differ at the second position, however for all $v \in T_{i,s}$ we have $q_i(v) \in Q_{i,s}$, and vice versa for every $q \in Q_{i,s}$ there is a $v \in T_{i,s}$ such that $q_i(v) = q$. This will be used in the "bending back" below; if two nodes s^+ and t^+ in V_i have the same τ_i^+-label, then for every node $v \in T_{j,s}$ there is a node $v' \in T_{j,t}$ such that v^+

and v'^{+} share the same τ_i^+-label.

The language τ_i^+ is finite, and $\mathfrak{T}_i^+$ is finitely branching. By König's Lemma we can find a depth N_i^{fin} such that all labels in τ_i^+ that occur only finitely often in $\mathfrak{T}_i^+$ do not occur at a distance from the root greater than N_i^{fin}. Let $N^{\mathrm{fin}} = \max\{N_i^{\mathrm{fin}}\}$.

By the same argument there is a depth $N_i^{\geq 0}$ such that for all paths α in $\mathfrak{T}_i^+$, every τ_i^+-label that occurs infinitely often along α occurs at least once at some depth d, $N^{\mathrm{fin}} < d < N_i^{\geq 0}$. Again let $N^{\geq 0} = \max\{N_i^{\geq 0}\}$.

The runs ρ_i implicitly contained in $\mathfrak{T}_i^+$ are accepting, hence the minimal parity index Ω_α occurring infinitely often along each path α is even. The same argument as above this time yields a depth N_i^{min} such that, for all paths α in $\mathfrak{T}_i^+$, one node with minimal parity index ω_α occurs at some depth d, $N^{\mathrm{fin}} < d < N_i^{\mathrm{min}}$; let $N^{\mathrm{min}} = \max\{N_i^{\mathrm{min}}\}$.

For any tree $\mathfrak{T}$, let $\mathfrak{T}|_d$ denote the set of nodes at distance d from the root, and $\mathfrak{T}|_{[d,d+k]}$ denote all nodes at distance between d and $d+k$ (inclusive). We define a function $\Xi_i : \mathfrak{T}_i^+|_{N^{\mathrm{min}}+1} \to \mathfrak{T}_i^+|_{[N^{\mathrm{fin}}:N^{\geq 0}]}$ such that $\mathfrak{T}_i^-$ is obtained by cutting off $\mathfrak{T}_i^+$ at level $N^{\mathrm{min}}+1$, and replacing the successors of nodes at level N^{min} by their respective Ξ_i images.

Let $s^+ \in \mathfrak{T}_1^+|_{N^{\mathrm{min}}+1}$ with label $(P_s, q, Q_{1,s}, Q_{2,s})$. Since the depth of s^+ is greater than N^{fin}, the label of s^+ occurs infinitely often in $\mathfrak{T}$. Hence there is a node $t^+ \in \mathfrak{T}_i^+|_{[N^{\mathrm{fin}}:N^{\geq 0}]}$ that has the same label as s^+. Let $\Xi(s^+) = t^+$. Then extend Ξ by choosing, for every node $T_{i,s}$, a node in $T_{i,t}$ with the same τ_i^+-label.

Then $\mathfrak{T}_i^-$ is obtained by cutting off $\mathfrak{T}_i^+$ after level min and replacing the successors of the nodes on level min by their respective Ξ-image. Consequently $Z|_{\mathfrak{T}_1^- \times \mathfrak{T}_2^-} : \mathfrak{T}_1^- \sim_\tau \mathfrak{T}_2^-$ is a τ-bisimulation.

Claim. The unravellings of the $\mathfrak{T}_i^-$ are accepting runs of $\mathcal{A}_i$ on the unravellings of the τ-reducts of the respective $\mathfrak{T}_i^-$.

The unravelling of $\mathfrak{T}_i^-$ describes a run, that is, at each node the appropriate formula as given by the transition function holds true. Regarding the parity index, observe that every infinite path α in $\mathfrak{T}_i^-$ only sees finitely many nodes of depth less than N^{fin}, so we can forget about them. Now regard the infinitely many segments of α, each going from where α enters $\mathfrak{T}_i^+|_{[N^{\mathrm{fin}}:N^{\geq 0}]}$ to the loop-back point on level N^{min}. Each of these segments is part of an infinite path β in $\mathfrak{T}_i^+$. Remember that $\mathfrak{T}_i^+|_{[N^{\geq 0}:N^{\mathrm{min}}]}$ was chosen such that every of these infinite paths β in $\mathfrak{T}_i^+$ sees its minimal, since the original run was accepting: even, parity index in that part of the path. Hence, for each segment of α, the minimal parity index seen is even.

We started with the assumption that φ^* is bisimulation invariant on the class of finite structures. However, if φ^* can distinguish two bisimilar (infinite) trees $\mathfrak{T}_1$ and $\mathfrak{T}_2$, we can transform them into two bisimilar finite structures, the τ-reducts of $\mathfrak{T}_1^-$ and $\mathfrak{T}_2^-$, that are distinguished by φ^*, a contradiction. Hence φ^* is bisimulation invariant on the class of all structures. $\square$

8.4 GSO and GSO*

The goal is to lift all current bisimulation-invariance theorems from the modal to the guarded world. The following corollary can be shown in exactly the same fashion as the corresponding result for MSO*.

Corollary 8.4.1. *Every* GSO**-formula that is bisimulation invariant is equivalent to a* μGF*-formula.*

Proof. If $\psi^*(\boldsymbol{x}) \in$ GSO* is guarded bisimulation invariant, then $\psi(\boldsymbol{x})$ is guarded bisimulation invariant on the class of all unravelled structures and hence, on these structures, equivalent to a μGF-formula $\varphi(\boldsymbol{x})$. Then, for all structures $\mathfrak{A}$ and all tuples $\boldsymbol{a} \in A$, $\mathfrak{A} \models \psi^*(\boldsymbol{a})$ iff, by definition, $\mathfrak{T}(\mathfrak{A}, \boldsymbol{a}) \models \psi(\boldsymbol{a})$ iff, by Theorem 8.2.2, $\mathfrak{T}(\mathfrak{A}, \boldsymbol{a}) \models \varphi(\boldsymbol{a})$ iff, since φ is guarded bisimulation invariant and $\mathfrak{A}, \boldsymbol{a} \sim_g \mathfrak{T}(\mathfrak{A}, \boldsymbol{a})$, $\mathfrak{A} \models \varphi(\boldsymbol{a})$. $\square$

At this point we hit a, for now, final dead end. What we need is the "Holy Grail" for transforming finite structures into finite graphs with a well-behaved consistency condition. Until then, the following conjectures are not more than wishful thinking, lacking any strong indication in either direction. The interested reader may see them as incentive to continue with research centred around finite structures, rather than trees.

Conjecture 8.4.2. *Every* GSO* *formula that is guarded bisimulation invariant on the class of finite structures is bisimulation invariant on the class of all structures.*

Conjecture 8.4.3. *Every* MSO*-formula that is bisimulation invariant on the class of finite graphs is, on finite graphs, equivalent to an* L_μ*-formula.*

Conjecture 8.4.4. *Every* GSO*-formula that is bisimulation invariant on the class of finite structures is, on finite structures, equivalent to a* μGF*-formula.*

Bibliography

[1] H. Andréka, J. van Benthem and I. Németi, *Modal languages and bounded fragments of predicate logic*, in Journal of Philosophical Logic, vol. 27, 1998, pp. 217–274.

[2] A. Arnold, *The μ-Calculus Alternation-Depth Hierarchy is Strict on Binary Trees*, Theoretical Informatics and Applications, vol. 33, 1999, pp. 329–339.

[3] A. Arnold and D. Niwiński, *Rudiments of μ-calculus*, North Holland, 2001.

[4] J. van Benthem, *Modal Logic and Classical Logic*, Bibliopolis, Napoli, 1983.

[5] J. van Benthem, *Dynamic bits and pieces*, ILLC research report LP-79-01, University of Amsterdam, 1997.

[6] J. van Benthem, *Modal Logic in Two Gestalts*, ILLC research report ML-98-12, University of Amsterdam, 1998.

[7] D. Berwanger and E. Grädel, *Games and model checking for guarded logics*, in Proceedings of Logic for Programming and Artificial Reasoning (LPAR), 2001.

[8] P. Blackburn, M. de Rijke and Y. Venema, *Modal Logic*, Cambridge University Press, 2001.

[9] E. Börger, E. Grädel and Y. Gurevich, *The Classical Decision Problem*, Springer, 1997.

[10] J. Bradfield, *The modal mu-calculus alternation hierarchy is strict*, Theoretical Computer Science, vol. 195, 1998, pp. 133–153.

[11] J. Bradfield and C. Stirling, *Modal logics and mu-calculi*, in Handbook of Process Algebra, Elsevier, 2001, pp. 293–332.

[12] D. Calvanese, G. De Giacomo and M. Lenzerini, *On the Decidability of Query Containment under Constraints*, in Proceedings of the 17th Symposium on Principles of Database Systems (PODS), 1998, pp. 149–158.

[13] B. Courcelle, *On the expression of graph properties in some fragments of monadic second-order logic*, Descriptive Complexity and Finite Models, in DIMACS Series in Discrete Mathematics, vol. 31, AMS, 1997, pp. 33–62.

[14] B. Courcelle, *The monadic second-order logic of graphs XIV: Uniformly sparse graphs and edge set quantifications*, Submitted for publication.

[15] B. Courcelle, I. Walukiewicz, *Monadic Second-Order Logic, Graph Converings and Unfoldings of Transition Systems*, Annals of Pure and Applied Logic, vol. 92, 1998, pp. 35–62.

[16] A. Dawar, *Feasible Computation Through Model Theory*, Dissertation, 1993.

[17] F. Donnini, M. Lenzerini, D. Nardi and A. Schaerf, *Reasoning in description logics*, in Principles of Knowledge Representation, CSLI Publications, 1996, pp. 193–238.

[18] F. M. Donini, M. Lenzerini, D. Nardi and W. Nutt *The Complexity of Concept Languages*, in Information and Computation, vol. 134(1), 1997, pp. 1–58.

[19] H.-D. Ebbinghaus and J. Flum, *Finite Model Theory*, Springer, 2nd edition, 1999.

[20] H. Ganzinger, C. Meyer and M. Veanes, *The Two-Variable Guarded Fragment with Transitive Relations*, in Proceedings of the 14th IEEE Symposium on Logic in Computer Science (LICS), 1999, pp. 24–34.

[21] H. Ganzinger and H. de Nivelle, *A Superposition Decision Procedure for the Guarded Fragment with Equality*, in Proceedings of the 14th IEEE Symposium on Logic in Computer Science (LICS), 1999, pp. 295-305.

[22] E. Goncalves and E. Grädel, *Decidability issues for action guarded logics*, in Proceedings of the International Workshop on Description Logics (DL), 2000, pp. 123–132.

[23] G. Gottlob, E. Grädel and H. Veith, *Datalog LITE: A deductive query language with linear time model checking*, ACM Transactions on Computational Logic vol. 3(1), 2002.

[24] E. Grädel, *On the restraining power of guards*, in Journal of Symbolic Logic vol. 64, 1999, pp. 1719–1742.

[25] E. Grädel, *Decision procedures for guarded logics*, in Proceedings of the 16th International Conference on Automated Deduction (CADE), Trento, 1999, LNAI 1632, Springer, 1999.

[26] E. Grädel, *Guarded fixed point logics and the monadic theory of countable trees*, to appear in Theoretical Computer Science.

[27] E. Grädel, *Why are modal logics so robustly decidable?*, in Current Trends in Theoretical Computer Science Entering the 21st Century, World Scientific, 2001, pp. 393–498.

[28] E. Grädel, M. Otto and E. Rosen, *Two-Variable Logic with Counting is Decidable*, in Proceedings of the 12th IEEE Symposium on Logic in Computer Science (LICS), 1997.

[29] E. Grädel, M. Otto and E. Rosen, *Undecidability Results for Two-Variable Logics*, in Archive for Mathematical Logic, vol. 38, 1999, pp. 313–354.

[30] E. Grädel, C. Hirsch and M. Otto, *Back and Forth Between Guarded and Modal Logics*, in Proceedings of the 15th IEEE Symposium on Logic in Computer Science (LICS), 2000, pp. 217–228.

[31] E. Grädel and I. Walukiewicz, *Guarded fixed point logic*, in Proceedings of the 14th IEEE Symposium on Logic in Computer Science (LICS), 1999, pp. 45–54.

[32] J. Y. Halpern and Y. Moses, *A Guide to Completeness and Complexity for Model Logics of Knowledge and Belief*, in Artificial Intelligence, vo. 54(3), 1992, pp. 319–379.

[33] D. Harel, D. Kozen and J. Tiuryn, *Dynamic Logic*, MIT Press, 2000.

[34] B. Herwig, *Extending Partial Isomorphisms*, in Combinatorica, vol. 15, 1995, pp. 365–371.

[35] C. Hirsch, and S. Tobies, *A Tableau Algorithm for the Clique Guarded Fragment*, in Advances in Modal Logics, vol. 3, 2001.

[36] C. Hirsch, and S. Tobies. *A Tableau Algorithm for the Clique Guarded Fragment*, LTCS-Report 00-03, 2000.

[37] J. Hladik, *Implementierung eines Entscheidungsverfahrens für das Bewachte Fragment der Prädikatenlogik*, diploma thesis, RWTH-Aachen, 2001.

[38] W. Hodges, *Model theory*, Cambridge University Press, 1993.

[39] I. Hodkinson, *Loosely guarded fragment of first-order logic has the finite model property*, preprint.

[40] M. Hollenberg, *Logic and Bisimulation*, Dissertation, 1998.

[41] E. Hoogland and M. Marx, *Interpolation in Guarded Fragments*, preprint.

[42] I. Horrocks, P. F. Patel-Schneider and R. Sebastiani, *An Analysis of Empirical Testing for Modal Decision Procedures*, in Logic Journal of the IGPL, vol. 8(3), 2000, pp. 293–323.

[43] I. Horrocks, U. Sattler and S. Tobies. *Practical Reasoning for Expressive Description Logics*, in Proceedings of the 6th International Conference on Logic for Programming and Automated Reasoning (LPAR), LNAI 1705, Springer, 1999, pp. 161–180.

[44] D. Janin and I. Walukiewicz, *On the expressive completeness of the propositional mu-calculus with respect to monadic second order logic*, in Proceedings of 7th International Conference on Concurrency Theory (CONCUR), LNCS 1119, Springer, 1996, pp. 263–277.

[45] D. Kozen, *Results on the propositional μ-calculus*, Theoretical Computer Science, vol. 27, 1983, pp. 333–354.

[46] R. Ladner, *The computational complexity of provability in systems of propositional modal logic*, in SIAM Journal on Computing, vol. 6, 1977, pp. 467–480.

[47] G. Lenzi, *A hierarchy theorem for the mu-calculus*, in Proceedings of the International Colloquium on Automata, Languages and Programming (ICALP), LNCS 1099, Springer, 1996, pp. 87–109.

[48] C. Lutz, U. Sattler and S. Tobies, *A suggestion for an n-ary Description Logic*, in Proceedings of the International Workshop on Description Logics (DL), 1999, pp. 81–85.

[49] M. Marx, *Tolerance Logic*, in Journal of Logic, Language and Computation, to appear.

[50] D. Niwinski and I. Walukiewicz, *Games for the mu-calculus*, Theoretical Computer Science, vol. 163, 1997, pp. 99–116.

[51] M. Otto, *Bisimulation-Invariant PTime and Higher-Dimensional mu-Calculus*, Theoretical Computer Science, vol. 224, 1999, pp. 237–265.

[52] M. Otto, *Modal and guarded characterisation theorems over finite transition systems*, in Proceedings of the 17th IEEE Conference on Logic in Computer Science (LICS), 2002.

[53] R. Paige and R. Tarjan, *Three Partition Refinement Algorithms*, in SIAM Journal on Computing, vol. 16, 1987, pp. 973–989.

[54] D. Park, *Finiteness is μ-ineffable*, Theoretical Computer Science, vol. 3, 1976, pp. 173–181.

[55] E. Rosen, *Modal logic over finite structures*, Journal of Logic, Language and Information, vol. 6, 1997, pp. 427–439.

[56] W. Szwast and L. Tendera, *On the decision problem for the guarded fragment with transitivity* in Proceedings of the 16th IEEE Conference on Logic in Computer Science (LICS), 2001, pp. 147–156.

[57] D. Seese, *The structure of the models of decidable monadic theories of graphs*, Annals of Pure and Applied Logic, vol. 53, North-Holland, 1991, pp. 169–195.

[58] W. Thomas, *Automata on infinite objects*, in Handbook of Theoretical Computer Science, vol. B, Elsevier, 1990, pp. 133–191.

[59] S. Tobies, *Complexity Results and Practical Algorithms for Logics in Knowledge Representation*, Dissertation, 2001.

[60] M. Vardi, *Why is modal logic so robustly decidable?*, Descriptive Complexity and Finite Models, in DIMACS Series in Discrete Mathematics and Theoretical Computer Science, vol. 31, AMS, 1997, pp. 149–184.

[61] M. Vardi, *Reasoning about the Past with Two-Way Automata*, in Proceedengs of the International Colloquium on Automata, Languages and Programming (ICALP), 1998.

[62] I. Walukiewicz, *Monadic Second-Order Logic on Tree-Like Structures*, Accepted to the Journal on Theoretical Computer Science.

Index

Curriculum Vitae

Zur Person

Name:	Colin Helmut Charles Hirsch
Geboren:	1975-01-08 in Wien
Staatsangehörigkeit:	österreichisch
Eltern:	Dr. Helmut Hirsch, Physiker
	Jennifer Hirsch (geb. Warren), Biochemikerin

Schulausbildung

1981 – 1984	Besuch der Volksschule Tempelgasse, Wien, Österreich
1984 – 1985	Besuch der Volksschule in Kautzen, Österreich
1985 – 1993	Besuch der Integrierten Gesamtschule Hannover-Linden, Hannover
1993	Allgemeine Hochschulreife (Abitur)

Hochschulausbildung

1993 – 1998	Informatikstudium an der RWTH-Aachen
1998	Diplom in Informatik, Nebenfach Mathematik
	Thema der Diplomarbeit: “Reliability of Database Queries”
1998 – 2002	Wissenschaftlicher Angestellter am Lehrstuhl für Informatik VII (Prof. Dr. W. Thomas) der RWTH-Aachen
2002	Promotion in Informatik

Studienbegleitende Tätigkeiten

1995-1996	Leitung einer Übungsgruppe im Fach “Mathematische Logik” am Lehr- und Forschungsgebiet Mathematische Grundlagen der Informatik (Prof. Dr. E. Grädel) der RWTH-Aachen
1996-1998	Entwurf und Implementierung von Algorithmen zur Kanalvergabe in Mobilfunknetzen am Lehr- und Forschungsgebiet Stochastik (Prof. Dr. R. Mathar) der RWTH-Aachen

Aachen, 2002-12-09