

Constraint Satisfaction with Infinite Domains

DISSERTATION

zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften (doctor rerum naturalium)
im Fach Informatik

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät II
der Humboldt-Universität zu Berlin

von Dipl.-Inf.

Manuel Bodirsky

geboren am 30. Dezember 1976 in Freiburg im Breisgau

Präsident der Humboldt-Universität zu Berlin
Prof. Dr. Jürgen Mlynek

Dekan der Mathematisch-Naturwissenschaftlichen Fakultät II
Prof. Dr. Uwe Kuchler

Gutachter:

1. Prof. Dr. Hans Jürgen Prömel
2. Prof. Dr. Martin Grohe
3. Prof. Jaroslav Nešetřil

Tag der Einreichung: 6.7.2004, Vorsitzender Prof. Dr. Johannes Köbler
Tag der mündlichen Prüfung: 4.11.2004

Zusammenfassung. Constraint Satisfaction Probleme tauchen in vielen Gebieten der Informatik auf, insbesondere im Gebiet der künstlichen Intelligenz, zum Beispiel beim räumlichen und zeitlichen Schließen, maschinellen Sehen, Scheduling. Ein Überblick findet sich in [Kumar, 1992, Dechter, 2003]. Andere Gebiete sind Graphentheorie, Aussagenlogik, Typsysteme für Programmiersprachen, Datenbanktheorie, automatisches Beweisen, Computerlinguistik und Bioinformatik.

Viele Constraint Satisfaction Probleme können auf natürliche Weise als Homomorphieprobleme formuliert werden. Hier betrachten wir für eine festgehaltene relationale Struktur Γ das folgende Berechnungsproblem: Gegeben sei eine Struktur S mit der gleichen relationalen Signatur wie Γ , gefragt ist ob es einen Homomorphismus von S nach Γ gibt. Dieses Problem ist das *Constraint Satisfaction Problem* $CSP(\Gamma)$ für Γ , und wurde für *endliches* Γ – die sogenannte *Schablone* des Problems – intensiv untersucht. Allerdings gibt es viele Constraint Satisfaction Probleme, die nicht mit endlicher Schablone formuliert werden können.

Wenn wir beliebige unendliche Schablonen zulassen, wird Constraint Satisfaction zu einem sehr ausdrucksstarken Formalismus, und wir können dann beispielsweise unentscheidbare Probleme als Constraint Satisfaction Probleme formulieren. In dieser Arbeit machen wir daher zusätzliche Annahmen für die Schablone. Eine dieser Annahmen ist ω -Kategorizität. Eine abzählbare Struktur Γ ist ω -kategorisch, wenn alle abzählbaren Modelle der erststufigen Theorie von Γ isomorph zu Γ sind. Dies ist ein zentrales Konzept aus der Modelltheorie und eng verwandt mit Quantor-elimination und Homogenität. Wir führen Argumente an, warum ω -Kategorizität ein sinnvoller Begriff ist, wenn man Constraint Satisfaction Probleme mit einer Schablone über einem unendlichen Wertebereich systematisch untersuchen will.

Die Berechnungskomplexität von Constraint Satisfaction Problemen hängt im wesentlichen davon ab, welche Relationen der Schablone *primitiv positiv definierbar* sind. Für ω -kategorische Schablonen können wir zeigen, daß eine Relation in Γ primitiv positiv definierbar ist dann und genau dann, wenn sie von den *Polymorphismen* in Γ erhalten wird. Dieser Satz ist für endliche Strukturen wohlbekannt [Bodnarčuk et al., 1969], und war der Ausgangspunkt des algebraischen Ansatzes zur Untersuchung der Berechnungskomplexität von Constraint Satisfaction mit endlichen Schablonen – siehe zum Beispiel [Jeavons et al., 1997]. Wir zeigen an einem Beispiel, daß für nicht ω -kategorische Strukturen dieser Satz im allgemeinen nicht gilt.

Eine Konsequenz dieses Satzes ist, daß sowohl für endliche als auch für

unendliche ω -kategorische Schablonen die Existenz eines effizienten Algorithmus für das entsprechende Constraint Satisfaction Problem von der Existenz gewisser Polymorphismen der Schablone abhängt. Ein Beispiel sind die ω -kategorischen Strukturen mit einem k -stelligen *fast-einstimmigen* Polymorphismus. In diesem Fall kann das entsprechende Constraint Satisfaction Problem in polynomieller Zeit mit Hilfe eines *Datalog Programmes* gelöst werden. Datalog ist ein Konzept aus der Datenbanktheorie, und wurde im Zusammenhang mit Constraint Satisfaction zum erstenmal in [Feder and Vardi, 1999] betrachtet. Dort werden für Constraint Satisfaction Probleme mit endlicher Schablone auch sogenannte *kanonische Datalog Programme* eingeführt, und es wird gezeigt, daß sich jedes Constraint Satisfaction Problem mit endlicher Schablone, das mit einem Datalog programm mit k Variablen gelöst werden kann, auch vom sogenannten *kanonischen Datalog Programm mit k Variablen* gelöst werden kann. Wir verallgemeinern dies auf ω -kategorische Schablonen gilt.

Die zweite Anforderung, die wir in dieser Arbeit bisweilen an die Schablone stellen, ist, daß Γ durch verbotene induzierte Substrukturen beschrieben werden kann. In diesem Fall ist $\text{CSP}(\Gamma)$ in der Klasse *monoton SNP* enthalten, einem Fragment existentieller zweitstufiger Logik, das im Zusammenhang mit Constraint Satisfaction in [Feder and Vardi, 1999] betrachtet wurde. Diese Annahmen für die Schablone sind allgemein genug, um viele zusätzliche Constraint Satisfaction Probleme zu erfassen, die nicht mit endlichen Schablonen formuliert werden können. Beispielsweise kann jedes Problem in *monoton monadisch SNP* – einer anderen Klasse die von Feder und Vardi eingeführt wurde – als Constraint Satisfaction Problem mit einer solchen Schablone formuliert werden.

In den letzten zwei Kapiteln dieser Arbeit beschäftigen wir uns mit konkreten Constraint Satisfaction Problemen mit ω -kategorischer *baumartiger* Schablone. Manche dieser Berechnungsprobleme haben Anwendungen in Computerlinguistik [Koller et al., 2000, Niehren and Thater, 2003] und Bioinformatik [Steel, 1992]. Wir geben neuartige Graphalgorithmen an, die diese Probleme in Polynomialzeit lösen, und direkt Lösungen für erfüllbare Constraint Satisfaction Probleme konstruieren. Zentral ist hier der Begriff einer *freien Menge von Knoten* im Constraint Graph, mit dessen Hilfe wir durch wiederholte Zerlegungen des Constraintgraphen in Zusammenhangskomponenten Lösungen rekursiv konstruieren können. Insbesondere lösen wir damit ein Problem, das in [Cornell, 1994] gestellt wurde. Wir erreichen beim Algorithmus für Cornell's Problem subquadratische Laufzeit, wenn wir bekannte dynamische (dekrementelle) Algorithmen für starken Zusammenhang in ge-

richteten Graphen verwenden.

Dominanzconstraints wurden in der Computerlinguistik eingeführt [Marcus et al., 1983, Backofen et al., 1995] und finden zahlreiche Anwendungen in zum Beispiel unterspezifizierter Semantik [Egg et al., 2001, Copestake et al., 1999, Bos, 1996], unterspezifizierter Diskursanalyse [Gardent and Webber, 1998], und Syntaxanalyse mit Baumadjunktionsgrammatiken [Rogers and Vijay-Shanker, 1994]. Es handelt sich um eine Formalismus, in dem Bäume mit Hilfe der Eltern-Kind und der Vorfahre-Nachfahre Relation beschrieben werden können. Erfüllbarkeit von Dominanzconstraints ist NP-vollständig [Koller et al., 1998]. Allerdings genügt es für viele Anwendungen *normale* Dominanzconstraints zu betrachten, und diese haben einen polynomiellen Erfüllbarkeitstest [Althaus et al., 2003]. Mit einem ähnlichen algorithmischen Ansatz wie bei unserem Algorithmus für Cornell's Problem können wir einen neuen Algorithmus für normale Dominanzconstraints angeben, der direkt eine Lösung (oder, falls gewünscht, alle Lösungen) eines normalen Dominanzconstraints generiert. Der Algorithmus ist dabei effizienter als die bisher bekannten Verfahren. Wieder können wir subquadratische Laufzeit erreichen – hier verwenden wir effiziente dekrementelle Algorithmen für zweifachen Graphzusammenhang.

Schließlich suchen wir nach schwächeren Annahmen als Normalität, die immer noch Polynomialzeitalgorithmen zulassen, das heißt, nach größeren handhabbaren Fragmenten von Dominanzconstraints. In diesem Kontext definieren wir die Klasse der *surjektiven Homomorphieprobleme*. Wie im Falle von Homomorphieproblemen sind Probleme der Klasse durch eine (in diesem Falle immer endliche) Schablone T gegeben, und wir fragen ob es für eine gegebene endliche Struktur S mit der gleichen Signatur wie T einen Homomorphismus von S nach T gibt. Wir zeigen, daß bestimmte Fragmente von Dominanzconstraints unter Polynomialzeitreduktionen äquivalent zu surjektiven Homomorphieproblemen sind.

Abstract. Constraint satisfaction problems occur in many areas of computer science, most prominently in artificial intelligence including temporal or spacial reasoning, belief maintenance, machine vision, and scheduling (for an overview see [Kumar, 1992, Dechter, 2003]). Other areas are graph theory, boolean satisfiability, type systems for programming languages, database theory, automatic theorem proving, and, as for some of the problems discussed in this thesis, computational linguistics and computational biology.

Many constraint satisfaction problems have a natural formulation as a homomorphism problem. For a fixed relational structure Γ we consider the following computational problem: Given a structure S with the same relational signature as Γ , is there a homomorphism from S to Γ ? This problem is known as the *constraint satisfaction problem* $\text{CSP}(\Gamma)$ for the *template* Γ and is intensively studied for relational structures Γ with a finite domain. However, many constraint satisfaction problems can not be formulated with a finite template.

If we allow arbitrary infinite templates, constraint satisfaction is very expressive and e.g. contains undecidable problems. In this thesis, we impose two restrictions on the template. The first restriction is *ω -categoricity*, a natural and well-studied concept in model-theory. The computational complexity of $\text{CSP}(\Gamma)$ is determined by the relations of Γ that have a *primitive positive definition* in Γ . For ω -categorical templates we can show that a relation is primitive positive definable in Γ if and only if it is preserved by the *polymorphisms* of Γ . This theorem is well-known for finite templates [Bodnarčuk et al., 1969, Geiger, 1968], and was the starting point of the algebraic approach to study the complexity of constraint satisfaction with finite templates, described e.g. in [Jeavons et al., 1997]. It shows that also for ω -categorical templates, the complexity of a constraint satisfaction problem is determined by the *clone of polymorphisms* of the template. One example where the existence of a certain polymorphism of the (finite or infinite) template implies tractability of the corresponding constraint satisfaction problem is the case where the polymorphism is a *k-ary near-unanimity* operation. In this case the problem can be solved by a *Datalog-program of width k*. For finite templates, [Feder and Vardi, 1999] proved that every constraint satisfaction problem that can be solved by a Datalog program of width *k* can also be solved by the *canonical Datalog program* of width *k*. This is another result we can generalize to ω -categorical templates.

The second restriction is that the template Γ can be described by a finite set of forbidden induced substructures. In this case the constraint satisfaction problem for Γ is in *monotone SNP*, which is a fragment of existential second

order logic introduced in the context of constraint satisfaction in [Feder and Vardi, 1999]. Finitely constraint ω -categorical templates are general enough to capture many additional constraint satisfaction problems that can not be formulated with finite templates. In fact, every problem in the class *monotone monadic SNP* – another class introduced by Feder and Vardi – can be formulated as a constraint satisfaction problem with such a template.

We finally focus on several constraint satisfaction problems that have an ω -categorical *tree-like* template. Some of these problems have applications in computational linguistics [Koller et al., 2000] and computational biology [Steel, 1992]. We present graph algorithms that solve these problems in polynomial time. In particular we solve a problem posed in [Cornell, 1994], and present a new and more efficient algorithm for *normal dominance constraints* [Althaus et al., 2001]. Subquadratic running time can be achieved using decremental graph connectivity algorithms.

Acknowledgements

I want to thank my supervisor, Prof. Dr. Hans Jürgen Prömel, for providing me with the unique research environment at Humboldt-University, and the research group at the department for algorithms and complexity, who made this thesis possible. I am also indebted to the European Graduate Program “Combinatorics, Geometry, and Computation” for the great support, and grateful to all its members and staff, for discussions, cooperation, and the good atmosphere. I am also grateful to Prof. Jaroslav Nešetřil and the members of the research groups ITI and DIMATIA in Prague for their hospitality during my six month stay at Charles University.

I thank all the people who gave me feedback on earlier versions of this text: Jan Schwinghammer, Dr. Joachim Niehren, Katharina Bodirsky, Dr. Mihyun Kang, Stefan Kirchner, Dr. Sven Thiel, and Dr. Timo von Oerzen; very helpful were the valuable remarks of Prof. Dr. Martin Grohe. I also want to thank many other colleagues for discussions and suggestions, or for answering my emails concerning their work. Special thanks also to Prof. Dr. Anusch Taraz, who was always there when I had any queries at the department. Finally I thank Germany and my family for education and support.

Contents

1	Introduction	1
1.1	Constraint Satisfaction	2
1.2	Finite Templates	4
1.3	Countable Templates	6
1.4	Related Literature	14
1.5	Other Views on Constraint Satisfaction	15
1.6	Outline of the Thesis	17
2	Countably Categorical Structures	19
2.1	Fundamental Concepts from Model Theory	21
2.2	The Theorem Ryll-Nardzewski	25
2.3	Model-completeness	27
2.4	Fraïssé’s Theorem	28
2.5	Strong and Free Amalgamation	31
2.6	Homogeneous Digraphs	33
2.7	Tree-like Structures	38
2.7.1	Boron Trees	38
2.7.2	Semilinear Orders	39
2.7.3	Dominance and Immediate Dominance	41
2.8	Homomorphisms and Cores	45

CONTENTS

3	Constraint Satisfaction	51
3.1	Introduction	51
3.2	The Complexity of some CSPs	57
3.2.1	The CSPs for the Homogeneous Digraphs	58
3.2.2	Tree Descriptions	62
3.2.3	The Fragments of Allen’s Interval Algebra	63
3.3	Monotone SNP	64
3.4	Datalog	71
4	The Clone of Polymorphisms	81
4.1	Tools from Universal Algebra	82
4.2	Clones on Finite Domains	85
4.3	Clones on Infinite Domains	87
4.4	The Basic Galois-Connection Inv-Aut	88
4.5	Primitive Positive Definability	90
4.6	Near-unanimity Operations	91
4.7	Adding Constants to the Signature	95
5	Graph Algorithms for Tree Constraints	99
5.1	Constraints in Computational Linguistics	99
5.2	Tree Descriptions	100
5.3	An Algorithm for a Restricted Signature	102
5.4	Phylogenetic Analysis	105
5.5	Reduction to Four Base Literals	107
5.6	Constraint Graphs and Freeness	108
5.7	The Algorithm	110
5.8	Subquadratic Running Time	112

6	Graph Algorithms for Tree Constraints	115
6.1	Dominance Graphs and Solved Forms	117
6.2	An Algorithm for Dominance Graphs	118
6.2.1	Freeness	120
6.2.2	The Main Step	121
6.2.3	Testing Freeness Conditions	124
6.3	Normal Dominance Constraints	125
6.3.1	Preliminaries	126
6.3.2	Reduction to Dominance Graphs	128
6.3.3	A Duality Theorem	129
6.3.4	Implementation and Evaluation	131
6.4	Larger Tractable Fragments	132
6.5	Surjective Homomorphism Problems	135
7	Conclusion and Outlook	141
7.1	Summary of Closure Conditions	142
7.2	Discussion	143
7.3	Outlook	144
7.4	List of Open Problems	146
7.4.1	Model Theory and Combinatorics	146
7.4.2	Constraint Satisfaction and Datalog	148
7.4.3	Computational Questions	148

Chapter 1

Introduction

One of the main concerns in theoretical computer science is to understand which computational problems are *tractable*, and which problems are *hard to solve*. Tract-able means that instances of the problem can be solved within a reasonable amount of computational resources and time, i.e., we would like to find a feasible algorithm. In this thesis, we consider problems as *tractable* if there exists a polynomial time algorithm, and we consider problems as *hard*, if they are NP-hard.

The class of tractable problems P and the class of NP-hard problems are appealing from a theoretical point of view, for many reasons. They are surprisingly robust concepts and for example turned out to be invariant under any reasonable machine model that is used to formalize computation. There are even characterizations of these classes that do not rely on the notions of a machine and computation – e.g. in descriptive complexity theory.

Unfortunately we do not understand the class P very well. In particular we have the tantalizing open problem whether there is a polynomial time algorithm that solves an NP-hard problem. An assumption of this thesis will be that this is not the case. Also, tractability and NP-hardness are not the only options for a computational problem – in fact, under the above assumption, there are infinitely many complexity classes that lie between P and the class of all NP-hard problems [Ladner, 1975]. However, not many natural candidates are known for such intermediate classes. Thus some researchers posed the question: what are natural and large classes of computational problems that exhibit a *dichotomy*, i.e., only contain tractable and NP-hard problems?

Constraint satisfaction problems are computational problems that occur in many areas of computer science, most prominently in artificial intelligence, including temporal or spacial reasoning, belief maintenance, machine vision, and scheduling (for an overview see [Kumar, 1992, Dechter, 2003]). Other areas are graph theory [Hell and Nešetřil, 1990], boolean satisfiability [Schaeffer, 1978], type systems for programming languages [Lincoln and Mitchell, 1992], database theory [Aho et al., 1981, Kolaitis and Vardi, 1998], and, as for some of the problems discussed in this thesis, computational linguistics and computational biology. Many of these problems have sophisticated algorithmic solutions. On the other hand, hardness results for constraint satisfaction problems tend to have elegant proofs.

Several frameworks to formalize the notion of constraint satisfaction have been proposed, most prominently the class CSP of constraint satisfaction problems that are defined as *homomorphism problems*. Such problems are defined by a relational structure, the so-called *template* of the constraint satisfaction problem. If the template has a finite domain, researchers conjecture a dichotomy, and started a classification project to delineate the border between tractability and hardness. Not much attention, however, was paid to the class of homomorphism problems where the template has an infinite domain.

1.1 Constraint Satisfaction

There is no established formal definition that captures everything what is called constraint satisfaction in the literature. Most constraint satisfaction problems are *computational problems*, where, informally, the instances of the problem consist of a set of *variables* and so-called *constraints*, and the task is to find a *solution*, i.e., an assignment that maps the variables to *values*, chosen from some *domain*, that *satisfies* all the constraints.

In this thesis we look at constraint satisfaction problems that are *homomorphism problems*. A homomorphism problem is given by a relational structure Γ , the *template*. The computational problem $\text{CSP}(\Gamma)$ is then to determine for a *finite* structure S of the same signature as Γ whether S homomorphically maps to Γ . This means, the elements of S must be mapped to the domain of Γ such that if there is a tuple in a certain relation in S , the corresponding relation holds on the images of the elements in Γ (a formal definition is given in Chapter 3).

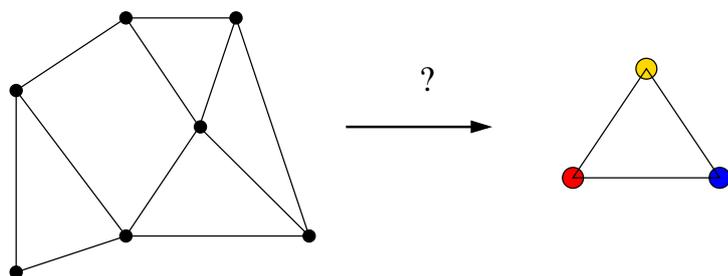


Figure 1.1: Three-colorability as a constraint satisfaction problem.

As an example, let the template be the graph K_2 , i.e., two vertices joined by an undirected edge (but without self-loops). In the corresponding constraint satisfaction problem $\text{CSP}(K_2)$ we have to check for a given graph G whether G homomorphically maps to K_2 . This computational task is often formulated as follows: Given a graph, find a two-coloring of the vertices of the graph such that adjacent vertices get different colors. This problem is clearly in P . However, if we replace K_2 by K_3 , i.e., three pairwise joined vertices, we get the well-known problem of graph 3-colorability, which is NP-hard (see Figure 1.1).

3-COLORABILITY

INSTANCE: A graph $G = (V; E)$.

QUESTION: Can we color the vertices V with three colors such that no two vertices adjacent in G get the same color?

One fundamental observation in this field is the trivial reformulation of such constraint satisfaction problems in more logical terms. Now we understand the instance of a constraint satisfaction problem as a first-order sentence of a very restricted form, namely an existentially quantified conjunction of positive literals, and we ask whether the template is a *model* of this sentence. For that, vertices in the instance correspond to existential variables in the sentence, and relations in the instance correspond to positive literals. These two formulations describe indeed one and the same thing: a solution to the above model checking problem is an assignment of values from the template to variables of the instance, and this assignment has to be a homomorphism.

Consider for example the graph k -coloring problem. This problem can be viewed as the constraint satisfaction problem for the complete graph on

k vertices, K_k . In the above interpretation, the vertices of an instance are the variables, and the vertices of K_k are the values of the domain of the constraint satisfaction problem. The edge relation between vertices of the instance represents the inequality relation on variables.

From now on, if we use the term *constraint satisfaction problem*, we mean a constraint satisfaction problem that is a homomorphism problem in the sense introduced above.

1.2 Finite Templates

Constraint satisfaction with finite templates was studied intensively. Clearly, every such problem is contained in NP. Schaeffer proved that for templates with two elements only – we identify these elements with *true* and *false* – the corresponding constraint satisfaction problems are either tractable or NP-complete [Schaeffer, 1978]. He explicitly described the templates T where $\text{CSP}(T)$ is tractable. On the other hand, all other constraint satisfaction problems over a two-element template can simulate the problem 1-in-3-SAT [Garey and Johnson, 1978] and are therefore NP-hard.

1-IN-3-SAT

INSTANCE: Set V of variables, and a ternary relation C over V , i.e., a set C of clauses over V such that each clause $c \in C$ has $|c| = 3$.

QUESTION: Is there a truth assignment for V such that each clause in C has at least one true literal and at least one false literal?

This problem remains NP-complete even if no $c \in C$ contains a negated literal, and this version of the problem can be cast as the constraint satisfaction problem with the finite template $(\{0, 1\}; C)$ where C is the ternary relation containing the tuples $(0, 0, 1)$, $(0, 1, 0)$, and $(1, 0, 0)$ only.¹ For simplicity, if we will later refer to the problem 1-in-3-SAT, we will mean the restricted version with positive literals only.

Hell and Nešetřil restricted the signature, rather than the cardinality of the template [Hell and Nešetřil, 1990]. They proved that if the template is a finite graph then the constraint satisfaction problem is tractable if and only if

¹It is also possible to formulate the original problem 1-in-3-SAT, or the well known problem 3-SAT, as constraint satisfaction problems, using several ternary relations (for each pattern of positive and negative occurrences of literals in the clause one relation).

the template is bipartite, under the assumption that $P \neq NP$. It is clear that a graph G homomorphically maps to a bipartite graph if and only if G itself is bipartite. But the difficult part was to prove that if the template contains an odd cycle, the constraint satisfaction problem is already NP-hard. The result does not extend to infinite templates: e.g. the complete graph on the natural numbers contains cycles but has a trivial constraint satisfaction problem. It is also difficult to extend this result to all digraphs: [Feder and Vardi, 1999] shows that the dichotomy question where the template is a digraph is already equivalent to the dichotomy question for arbitrary constraint satisfaction problems with a finite template.

Feder and Vardi identified two classes of tractable constraint satisfaction problems with finite templates over an *arbitrary finite signature* [Feder and Vardi, 1999]. The first class consists of the constraint satisfaction problems of *bounded width* – these problems can be solved by a Datalog program. They gave an equivalent characterization of bounded width using pebble games from finite model theory. The problems from the other class of tractable problems discussed in [Feder and Vardi, 1999] essentially reduce to group-theoretic problems – we will come back to such problems below.

The most systematic approach to constraint satisfaction is a connection to universal algebras developed in [Jeavons et al., 1997, Jeavons et al., 1998, Bulatov et al., 2000, Dalmau, 2000b, Bulatov et al., 2001, Bulatov, 2002b, Bulatov, 2002a, Bulatov, 2003]. The fundamental observation is that the complexity of a constraint satisfaction problem is already determined by the *clone of polymorphisms* of the template. *Clones* are studied in universal algebra: they are sets of functions on a common domain, containing all projections, and closed under compositions. A polymorphism of a relational structure Γ is a homomorphism from Γ^n to Γ , where Γ^n is a Cartesian power of Γ , see Section 4.1. The set of all polymorphisms $\text{Pol}(\Gamma)$ of Γ forms a clone.

Tractability of a constraint satisfaction problem is directly related to the presence of certain polymorphisms in Γ , and intractability by the absence of polymorphisms in Γ . We can describe classes of polymorphisms of a relational structure (also called *operations*) by functional identities. *Idempotent* operations are for instance defined by the identity $f(x, \dots, x) = x$. The case where the above mentioned group-theoretic algorithms apply is characterized by a *Malt'sev* operation, i.e., a polymorphism f satisfying the identities $f(x, x, y) = f(y, x, x) = y$. The constraint satisfaction problems for finite templates with a *Malt'sev* operation are all tractable [Bulatov, 2002a].

Bounded strict width problems, which also have a definition via Data-

log programs, are characterized by the presence of a *near-unanimity* operation, i.e., there is a k -ary polymorphism f of the template that satisfies $f(x, y, \dots, y) = f(y, x, y, \dots, y) = f(y, \dots, y, x) = y$ for some $k \geq 2$ [Feder and Vardi, 1999, Jeavons et al., 1998]. When classifying constraint satisfaction problems with finite templates, we study their polymorphism clones and can use nontrivial results from universal algebra for algebras over finite sets (see for example [Szendrei, 1986, Rosenberg, 1986]).

In this approach we can also elegantly describe all the tractable cases of Schaeffers dichotomy result: Assuming that $P \neq NP$, such templates have to have either a constant operation, a majority operation, an idempotent binary operation, or a Malt'sev operation. If there is no such operation, all polymorphisms f are *essentially unary*, i.e., $f(x_1, \dots, x_k) = g(x_i)$ for some $1 \leq i \leq k$ and some unary operation g , and the constraint satisfaction problem is NP-hard since it can simulate the problem 1-in-3-SAT. The algebraic approach also led to a classification of the complexity of the constraint satisfaction problem with templates over a 3-element set [Bulatov, 2002b]. The case where the template T contains a unary relation for each subset of the domain of T also exhibits a dichotomy [Bulatov, 2003].

1.3 Countable Templates

Many natural computational problems can be formulated as constraint satisfaction problems with a countable template, but not with a finite template. Consider for instance the set of rational numbers, linearly ordered by their size. The constraint satisfaction problem $CSP((\mathbb{Q}; <))$ can be understood as the problem *Digraph-acyclicity*: A digraph can be homomorphically mapped to the linear order if and only if it is acyclic. Later in this section we will see several other well-known and not so well-known computational problems that can be expressed as constraint satisfaction problems with infinite templates.

DIGRAPH-ACYCLICITY

INSTANCE: A digraph $D = (V; E)$.

QUESTION: Is there a directed cycle in D ?

If we do not impose any restriction on the template, constraint satisfaction with infinite templates is very expressive. The constraint satisfaction problems with arbitrary templates are precisely those problems that are closed under disjoint unions, and whose complement is closed under homomorphisms – see Section 3.1. There are also infinite templates with an *undecidable* constraint satisfaction problem – we show this with a counting argument in Section 3.1. To explore which techniques for constraint satisfaction with finite templates can be applied for infinite templates as well, we formulate some natural conditions on the structure of the template. These restrictions should be general enough to still contain interesting constraint satisfaction problems with infinite templates.

The first condition on the templates is ω -*categoricity*, a fundamental concept in model theory. Roughly speaking, we require that the relational structure is countable and up to isomorphism fully described by its first-order theory. Something similar holds trivially for finite structures: if we have equality in our language, we can describe a structure up to isomorphism with a first-order sentence. The concept of ω -categoricity concerns infinite structures: a countable structure Γ is called ω -categorical, if all countable models of the first-order theory of Γ are isomorphic to Γ . The dense linear order of the rational numbers is an example of such an ω -categorical structure. Another example is countable homogeneous K_n -free graph. We will see several other examples in this section. There are various alternative characterizations of ω -categoricity – see Chapter 2.

The second restriction concerns finite representations of our templates. Again, the formal definitions can be found in Chapters 2 and 3. The idea is that we describe countable relational structures by a finite set of finite forbidden induced substructures. For the examples above there is such a description: the dense linear order is characterized by the fact that it is a tournament not containing an oriented triangle, in the later case the forbidden substructure is K_n . We do not always need this assumption – some of the theorems we prove also hold without this condition.

In the remainder of this section we want to give an impression what kind of computational problems can still be formulated as constraint satisfaction problems under these restrictions on the template. We do this with a loose collection of examples. Given the problem, it is sometimes not immediately clear how the template should look like. This might be the case for the following computational problem, which I could not find in the literature.

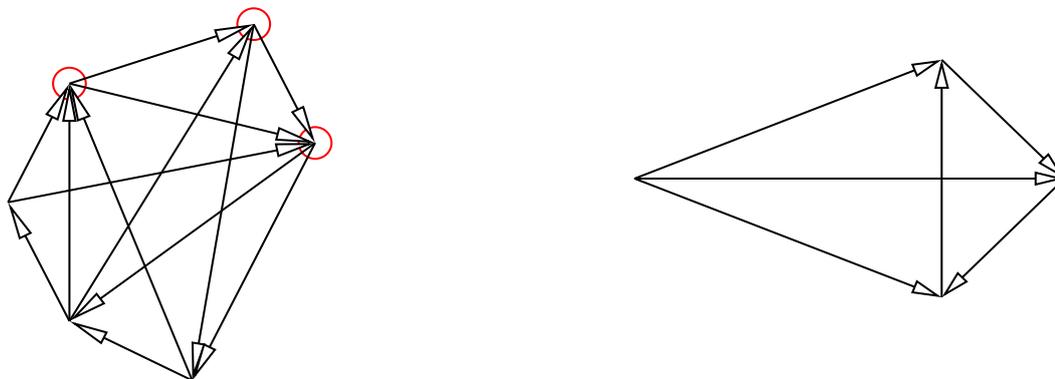


Figure 1.2: If we switch all arcs between the encircled and the other vertices in the left digraph, the resulting digraph is acyclic. The right digraph is not switching-equivalent to an acyclic graph.

SWITCHING-DIGRAPH-ACYCLICITY

INSTANCE: A digraph $D = (V; E)$.

QUESTION: Can we partition the vertices V into two parts, such that the graph that arises from D by switching all arcs between the two parts is acyclic?

We call two digraphs D_1 and D_2 *switching-equivalent* if there exists a subset S of vertices of D_1 such that the graph that arises from D_1 by switching all arcs between S and its complement in D_1 is isomorphic to D_2 . In the above problem we therefore ask whether a digraph is switching-equivalent to an acyclic graph. See Figure 1.2 for an example of a yes- and a no-instance of the problem.

To formulate this as a constraint satisfaction problem, partition the set of rational numbers \mathbb{Q} into two dense subsets Q_1 and Q_2 . Let a, b be distinct rational numbers. If both a and b are in Q_1 , or both are in Q_2 , then there is an arc between numbers a and b iff $a < b$. If a and b are in different parts then we put an arc between a and b iff $a > b$. We claim that the resulting countable tournament² is unique up to isomorphism, and we call it $S(2)$ (details can be found in Section 2.6 and 3.2). Moreover, the constraint satisfaction problem $\text{CSP}(S(2))$ is precisely the problem defined above. The tournament $S(2)$ has various other elegant representations: it is for instance isomorphic to a countable dense subset of the points on the unit circle in

²A tournament is a digraph where there is exactly one arc between every pair of vertices.

the plane without antipodal points, where two points ab are connected if the oriented line from a to b has the origin on the left side. Having that, we see that the above problem can exchangeably be stated in the following form (see Section 2.6):

CYCLIC-EMBEDDING

INSTANCE: A digraph $D = (V; E)$.

QUESTION: Can we map the vertices from V to the plane, such that every arc in E is embedded in the plane in such a way that it has the origin on its left side?

Another example that will be studied in this thesis is a special case of a problem that we called *pure dominance constraints* in [Bodirsky and Kutz, 2002]:

CONSISTENT-GENEALOGY

INSTANCE: A digraph D with two types of arcs, called *ancestors* and *non-ancestors* arcs.

QUESTION: Can we find a forest with oriented edges on the vertex set of D , such that for every ancestor arc in D there is a directed path in the forest, and for every non-ancestor arc there is no directed path in the forest?

The problem is a special case of a problem posed in computational linguistics [Cornell, 1994]. One could illustrate it with the following setting: Let us assume that, unlike the biological genealogy of sexual reproduction, where every creature has *two* parents, every person with a PhD has *one* academic parent – the advisor. An advisor can have many PhD students, and they may again have PhD students, and so on – the advisor is an *ancestor* for all of them. We are interested in the genealogy of the persons having a PhD – which forms under the above assumption a forest, where each tree in the forest has a unique root. (There are in fact public databases in the internet containing such information for e.g. mathematicians.)

To build the genealogy tree we are given information of the type “A is an ancestor of B” and information of the type “C is *not* an ancestor of D”. The task is to determine whether such information is *consistent*, i.e., whether there is a genealogy forest satisfying all the constraints. Consider Figure 1.3. Observe that certain ancestorship and non-ancestorship information might *imply* other information. For instance, any tree with ancestorship

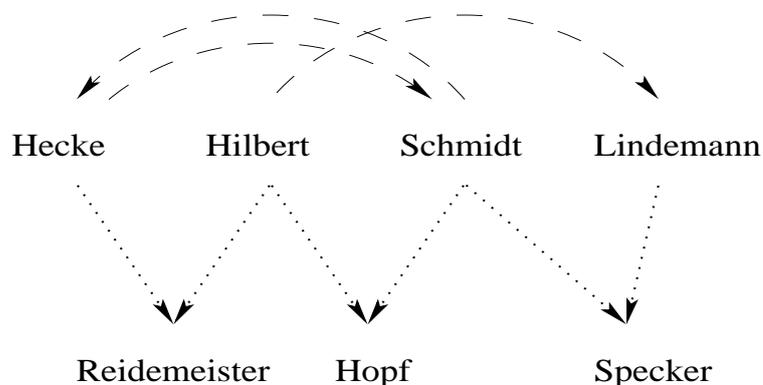


Figure 1.3: Partial information about some of the academic descendents of Felix Klein. Dotted arcs indicate ancestorship. Dashed arcs indicate non-ancestorship. The depicted information alone implies that Lindemann was an ancestor of Hilbert.

and non-ancestorship as specified by the arcs of the picture also satisfies that Lindemann is an academic ancestor of Hilbert (in fact, he was his academic father).

The same question could be asked for genealogies for the last name of humans – again under the assumption that every human gets the last name of one of the parents. Of course, these are toy problems. For the reconstruction of a genealogy tree from given data, we usually do not have the type of information that we are given in these problems. However, there are related problems that were studied in computational biology – see Section 5.4. In Chapter 6 we present an efficient algorithm that solves this problem as a special case. With the same algorithmic ideas we can then also efficiently solve tractable problems that came from phylogenetic analysis [Steel, 1992, Henzinger et al., 1996], optimization of relational expressions [Aho et al., 1981], and computational linguistics [Cornell, 1994].

The problem Consistent-genealogy can be formulated as a constraint satisfaction problem. To define the template we use the following *dense proper semilinear order* [Cameron, 1996, Adeleke and Neumann, 1985]. The domain of the structure is the set of all non-empty finite sequences $a = (q_0, q_1, \dots, q_{n-1})$ of rational numbers. Let $a < b$ if either

- b is a proper initial subsequence of a , or
- $b = (q_0, \dots, q_{n-1}, q_n)$ and $a = (q_0, \dots, q_{n-1}, q'_n, q_{n+1}, \dots, q_m)$, $q_n < q'_n$.

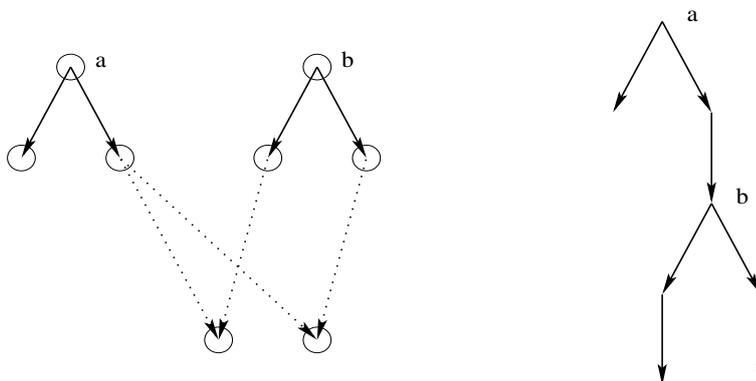


Figure 1.4: A dominance graph and a solution.

The relation $<$ corresponds to ancestorship edges, this is, we write $a < b$ if b is an ancestor of a . The set of all ordered pairs of distinct points not in $<$, denoted by $\not<$, corresponds to the non-ancestorship edges. Such and related structures will be discussed in Section 2.7.

A problem that might look similar, but has a different nature, is the following problem for *dominance-graphs* introduced in [Althaus et al., 2001], motivated by applications in computational linguistics.

DOMINANCE-GRAPH-SOLVABILITY

INSTANCE: A digraph with two types of arcs, called *dominance* and *immediate dominance* edges, respectively. The immediate dominance edges form a set of disjoint rooted trees of height one.

QUESTION: Is there a set of disjoint rooted trees on the vertices V containing the immediate dominance edges, where the edges are directed away from the roots, such that for every dominance edge there is a directed path in a tree?

Consider for example Figure 1.4. The rooted tree on the right is a solution for the dominance graph on the left.

We do not formulate this problem as a homomorphism problem, but as a *substructure problem*. Such problems are again given by a template, and we ask whether an instance is a substructure (in this thesis substructures are *weak substructures*, i.e., not necessarily induced substructures) of the template. Substructure problems and Homomorphism problems are closely

related, although the computational complexity for the same template might be different - this will be discussed in Chapter 3.

The problem Dominance-graph-solvability can be formulated as a substructure problem, using the following ω -categorical template Δ , which contains two relations denoted by \triangleleft and \triangleleft^+ . Again the domain of the structure Δ is the set of all non-empty finite sequences $a = (q_0, q_2, \dots, q_n)$ of rational numbers. We say that a *dominates* b , and write $a \triangleleft^+ b$, if either

- a is a proper initial subsequence of b , or
- $a = (p_0, \dots, p_{2n-1}, p_{2n})$ for $n \geq 0$, and $b = (p_0, \dots, p_{2n-1}, p'_{2n}, p_{2n+1}, \dots, p_m)$, where $p_{2n} < p'_{2n}$.

Note that Δ is infinitely branching at sequences of odd length, and that there are no maximal lower elements below sequences of even length. Also, the structure does not contain any maximal or minimal elements. We write ' $x \triangleleft y$ ', and say that x *immediately dominates* y , iff for every element z dominating y in Δ we have either $z \triangleleft^+ x$ or $z = x$. Then the (weak) substructures of $(\Delta; \triangleleft^+, \triangleleft)$ are the yes-instances of Dominance-graph-configurability.

In [Althaus et al., 2001], an efficient algorithm for a restricted version of the problem was presented. It is based on a duality theorem: An instance S of the restricted version of the problem is a substructure of Δ if and only if S does not contain certain bad cycles as a substructure (for details see Section 6.3.3). In Chapter ?? we present a different and more efficient algorithm.

Quartet-compatibility is a problem relevant in phylogenetic analysis in computational biology. It is NP-hard [Steel, 1992]. We refer to Section 5.4 for an introduction to these applications.

QUARTET-COMPATIBILITY

INSTANCE: A collection C of quartets $xy|uv$ over a set X .

QUESTION: Is there some tree with leaf set X such that for each quadruple $xy|uv$ in C the paths from x to y and from u to v do not have common vertices?

This problem is the constraint satisfaction problem of an ω -categorical structure arising from *Boron trees* that was studied e.g. in [Cameron, 1996] in the context of permutation groups of countable sets – for details we again refer to Section 2.7.

Finally we mention a problem that is an instance of an important subclass of constraint satisfaction with ω -categorical templates. We can find it in the list of NP-hard problems in [Garey and Johnson, 1978].

CYCLIC-ORDERING

INSTANCE: Finite set A , collection C of ordered triples (a, b, c) of distinct elements from A .

QUESTION: Is there an injective function $f : A \rightarrow \{1, 2, \dots, |A|\}$ such that for each $(a, b, c) \in C$, we have either $f(a) < f(b) < f(c)$ or $f(c) < f(b) < f(a)$?

This problem can be formulated as a constraint satisfaction problem on the rational numbers, with a single ternary relation, namely $\{(a, b, c) \in \mathbb{Q} \mid f(a) < f(b) < f(c) \text{ or } f(c) < f(b) < f(a)\}$. As a matter of fact, this is a structure which has – as model theorists say – *an interpretation in the dense linear order* $(\mathbb{Q}; <)$. Structures that have an interpretation in an ω -categorical structure are again ω -categorical. A whole class of problems that can be described with templates having an interpretation in \mathbb{Q} are the fragments of Allen’s interval algebra, containing also many tractable fragments. The domain there is \mathbb{Q}^2 , where the elements of this domain are viewed as closed intervals over the rational numbers. The signature contains symbols for relations between intervals: such intervals can e.g. overlap or include each other. Depending on which relations between intervals we have in the signature of our template, we have different computational problems, and these were called the *fragments of Allen’s interval algebra*. The fragments exhibit a dichotomy: they are either NP-hard or tractable [Bürckert and Nebel, 1995, Jeavons et al., 2003].

Uncountable Domains? We only consider countable domains. The reason is that if we had some template with an uncountable domain, we can find a template with a countable domain that has precisely the same constraint satisfaction problem. This follows from the Löwenheim-Skolem theorem – see Section 2.1. 1

1.4 Related Literature

This section surveys related literature, and points to introductory books and articles (here I make a subjective selection that reflects my personal perspective). We also discuss the choices for terminology and notations in this thesis.

Constraint satisfaction has its origins in artificial intelligence [Freuder, 1978, Freuder, 1982, Mackworth and Freuder, 1993, Montanari, 1974]. A recent book is [Dechter, 2003]. A landmark paper for the theory of constraint satisfaction is [Feder and Vardi, 1999] (first appeared as [Feder and Vardi, 1993]). They prove a great number of results concerning the complexity of constraint satisfaction problems, relations to finite model theory, and group theory. To learn about the connection of constraint satisfaction with finite templates to universal algebra we recommend to read [Jeavons et al., 1997]. With this approach one can use classification results for clones – for instance the beautiful classification of *minimal clones* [Rosenberg, 1986].

For constraint satisfaction with infinite templates we are interested in the model theory of countable structures; our favorite introduction is [Hodges, 1997]. Since we focus on ω -categorical structures, that also have a characterization via their automorphism group (a structure is ω -categorical if and only if its automorphism group is *oligomorphic*), topics from infinite permutation groups become relevant, e.g., from the inspiring book [Cameron, 1996] that influenced many parts of this thesis. The polymorphism clones of ω -categorical structures – I would like to suggest to call them *oligomorphic clones* – seem to be an untouched subject and I do not know of any reference.

In the later sections of this work we study constraint satisfaction problems for certain tree-like templates. For readers that are not into logic we note that both Chapter 6 and ?? are algorithmic, and essentially self-contained. The computational problems have a non model-theoretic formulation and independent motivations from various fields of applications. We will later give separate surveys for the relevant literature in the applications for computational linguistics and phylogenetic analysis. Some familiarity with fundamental graph theoretical concepts might be useful in these two chapters. An excellent introduction is [Diestel, 1997]. We also do not need much prerequisites in complexity theory: everything can be found in the classical book [Garey and Johnson, 1978].

Terminology and notation. Aside from conflicts between the nomenclature from model theory and from infinite permutation groups (this is discussed e.g. in [Adeleke and Neumann, 1985]), and the notion of *substructure* (either *induced* or not; see Section 2.1), there is always standard terminology, which we thus use.

Concerning notation, there are many possible ways to name the objects under consideration, due to the different areas that are touched by the subject. The most frequent mathematical symbol used here is Γ ; it always denotes a countable relational structure, usually ω -categorical, sometimes with additional properties. We started using this symbol in [Bodirsky and Nešetřil, 2003] because it was used for countable homogeneous structures in the monograph [Cherlin, 1998]. For constraint satisfaction problems, Γ denotes the template – only if the template is finite we use T for the template, as, e.g., in [Feder and Vardi, 1999]. Some special tree-like ω -categorical structures (or their domains) will be denoted by Λ, Δ , following [Adeleke and Neumann, 1985]. The relations on these structures will be denoted in the same way as for axiomatizations of finite trees in [Backofen et al., 1995]. Classes of finite structures are denoted by caligraphic letters $\mathcal{A}, \mathcal{B}, \mathcal{C}$.

The choice for other symbols was more canonical: Small letters a, b, c denote elements of some structure, x, y, z first-order variables, big letters A, B, C finite subsets of elements of a structure, letters S, T finite relational structures, and R denotes relations in a relational structure etc. In Section 2.1 most notation is formally introduced. The notation from universal algebra is as in [Kaluzhnin and Pöschel, 1979], the notation from model theory mostly as in the mentioned book [Hodges, 1997].

1.5 Other Views on Constraint Satisfaction

As already mentioned, the term constraint satisfaction is used in many different ways in the literature. Current research on constraint satisfaction can be grouped into several areas, briefly described in the following paragraphs. In this thesis, we will not deal with the questions in these paragraphs.

Uniform homomorphism problems. For a fixed relational structure T , we consider the computational problem whether a given finite structure S homomorphically maps to T . One generalization of constraint satisfaction is that both S and T are given in the input (here T is assumed to be a finite

structure as well). Now we ask for the complexity of this problem, if we restrict the potential choices for S and T . Suppose \mathcal{C} , \mathcal{D} are classes of finite structures with finite for the case relational signature τ . Then $\text{CSP}(\mathcal{C}, \mathcal{D})$ denotes the computational problem to determine for given $S \in \mathcal{C}, T \in \mathcal{D}$ whether there is a homomorphism from S to T .

It was noted e.g in [Freuder, 1990] that $\text{CSP}(\mathcal{C}, \mathcal{D})$ is tractable if the class \mathcal{C} has *bounded tree-width*. This was generalized in [Dalmau et al., 2002] and finally led to a full classification of the tractable problems of the form $\text{CSP}(\mathcal{C}, \mathcal{D})$ where \mathcal{D} is the set of *all* τ -structures [Grohe, 2003]. Such a problem is tractable if and only if the class \mathcal{C} has *bounded tree-width*, under some complexity assumptions from parameterized complexity theory.

Function symbols. In this thesis we only look at *relational* structures. The constraint satisfaction problem can be posed in the very same form for first-order structures that might as well contain *function symbols*. In fact, corresponding computational problems have been studied in the literature. If Γ is the free term algebra of function symbols from Σ , and the only relation symbol is first-order equality, this is nothing but the well-known first-order unification problem (which can be solved in linear time, [Paterson and Wegman, 1978]).

As a first step towards a systematic picture in this setting, [Feder et al., 2002] looked at constraint satisfaction problems with unary functions over a finite domain – for a single function symbol and for two function symbols with special properties a dichotomy is proven. If the template contains two function symbols without any restriction, the dichotomy question is equivalent to the dichotomy question for CSP with finite relational templates.

The literature on *combining constraint solving* (see the survey article [Baader and Schulz, 2001]) has an even broader view on constraint satisfaction as compared to here. They also stress the connection to model theory and universal algebra, but are mainly concerned with decidability questions of more expressive constraint languages.

Maximum constraint satisfaction. Another typical computational goal for a given constraint satisfaction problem is to find an ‘optimal’ assignment, i.e., an assignment of values to the variables that *maximizes* the number of satisfied constraints. A number of problems including MaxSat, MaxCut, and MaxDicut can be represented in this framework. It is well-studied in the Boolean case, that is, if the template is defined on a two-element domain,

see e.g. [Creignou et al., 2001]. For general domains, the complexity question [Cohen et al., 2004] and approximability [Datar et al., 2003] recently attracted attention. The corresponding maximization class for SNP, called MaxSNP [Creignou et al., 2001, Papadimitriou and Yannakakis, 1991, Mayr et al., 1998], is even larger and of particular importance to the theory of approximation algorithms; every problem in MaxSNP can be approximated within a constant ratio.

Quantified constraints. We already mentioned that an instance of a constraint satisfaction problem can be considered as a primitive positive sentence, and the constraint satisfaction problem is whether the template is a model for this sentence. Allowing negated atomic formulas in the input is a special case, since we can expand the signature appropriately. But we can also increase the expressive power of constraint satisfaction by allowing not only existential, but also universal or other quantifiers in the input. This was studied in [Feder and Kolaitis, 2004, Dalmau, 2000a, Boerner et al., 2003].

Counting constraint satisfaction problems. How difficult is it to compute the *number* of solutions of a constraint satisfaction problem? This is called the counting constraint satisfaction problem and was studied in [Dyer and Greenhill, 2000, Bulatov and Dalmau, 2003, Bulatov and Grohe, 2004]. Is there a dichotomy into P and $\#P$ -hard? It turned out that many techniques for constraint satisfaction are useful for counting constraint satisfaction as well.

If the template is ω -categorical, we can generalize this question to ω -categorical structures, and ask for the number of nonisomorphic solutions to a given instance (for ω -categorical structures, this is always a finite number). Counting the number of types realized in an ω -categorical structure is considered e.g. in [Cameron, 1996]).

1.6 Outline of the Thesis

In Chapter 2 we introduce some fundamental concepts from model theory that are necessary to describe the infinite templates we are dealing with here. We focus on concepts and theorems in model theory that are relevant for constraint satisfaction with ω -categorical templates.

In Chapter 3 we introduce the framework of constraint satisfaction problems studied in this thesis. We give several examples of computational problems that have been studied in the literature. We also discuss the relationship to fragments of existential second order logic, and to the theory of Datalog programs.

In Chapter 4 the algebraic method, which is known from constraint satisfaction with finite templates, is generalized to ω -categorical templates.

Chapter 6 contains the description of efficient algorithms that solve several problems in the literature, containing open problems from computational linguistics [Cornell, 1994, Bodirsky and Kutz, 2002] and well-known tractable problems from phylogenetic analysis in computational biology [Aho et al., 1981, Steel, 1992, Henzinger et al., 1996].

Chapter ?? applies similar algorithmic ideas to solve *normal dominance constraints* used in computational linguistics [Althaus et al., 2003, Bodirsky et al., 2004, Niehren and Thater, 2003]. Normal dominance constraints are a restricted form of dominance constraints, where the satisfiability problem is NP-hard [Koller et al., 1998, Egg et al., 2001]. In fact, our algorithm applies not only to normal dominance constraints; in Section 6.4 we determine the border between the tractable and the NP-hard fragments of dominance constraints.

Chapter 2

Countably Categorical Structures

A countable structure whose first-order theory has precisely one countable model up to isomorphism is said to be ω -categorical (or, exchangeably used in the literature, \aleph_0 -categorical). These structures play an important rôle for constraint satisfaction, since many techniques to study the computational complexity of constraint satisfaction for finite templates carry over to ω -categorical templates. On the other hand, various constraint satisfaction problems from different areas of computer science can be formulated with ω -categorical templates, and not with finite templates.

Many examples of ω -categorical structures are easily defined via *homogeneous* structures, a concept which links model theory with combinatorics, via Fraïssé's theorem. In fact, every ω -categorical structure can be made homogeneous by expanding the structure with first-order definable relations. In this chapter we will mention results that have been made towards a classification of countable homogeneous relational structures. These structures are studied by model theorists, and they have many remarkable properties. For signatures with finitely many k -ary relation symbols for each $k \geq 1$ they allow quantifier elimination and are ω -categorical. This will give us many examples of ω -categorical structures. These examples will later be useful to formulate several interesting computational problems as constraint satisfaction problems.

Usually, ω -categoricity and various other notions introduced in this chapter are in model theory more generally defined for first-order *theories*, and not, as it is done here, only for relational structures. But since we consider

ω -categorical relational structures in this thesis, and since they are up to isomorphism in a one-to-one correspondence with their first-order theories, it suffices for us to define these concepts for the structures themselves. Another model-theoretic aspect, where we have a slightly shifted focus compared to classical model theory, is that structures in model-theory are usually considered up to first-order definability. Because of our applications in constraint satisfaction, we need to take a closer look, since the complexity of a constraint satisfaction problem is very sensitive to the choice of the signature of the template. But we can still consider structures up to so-called *primitive positive definability*. Primitive positive definable relations are also important in e.g. the model theory of modules (see [Hodges, 1993]). The relevance of primitive-positive definability in constraint satisfaction comes from the simple fact that finite primitive positive expansions of a template do not change the complexity of the corresponding constraint satisfaction problem.

Two different ω -categorical structures might have the same constraint satisfaction problem. An important concept in this context is the concept of a *core of a relational structure*. Cores were originally introduced for finite templates. A finite relational structure is a *core*, if every endomorphism of the structure is an automorphism. In Section 2.8 we have a look at a possible generalization of the notion of a core to infinite structures. We say that a relational structure is a core, if every endomorphism is an embedding, i.e., is injective and also preserves the complements of the relations in the structure. For finite structures, this is clearly equivalent to the previous definition. We discuss properties of ω -categorical cores that are relevant to constraint satisfaction.

Outline of the chapter. We first recall some fundamental concepts from model theory, and give several equivalent characterizations of ω -categoricity. The following sections describe a sequence of stronger and stronger properties that an ω -categorical structure (or its finite induced substructures) might satisfy: *model-completeness*, *amalgamation*, *strong amalgamation*, and finally *free amalgamation*. Free amalgamation is crucial to state the classification of the homogeneous digraphs [Cherlin, 1998], which will be presented next.

Other examples of ω -categorical structures are various *tree-like* structures, and they will be of particular interest in constraint satisfaction later. We close with a discussion of core-like properties for infinite structures.

2.1 Fundamental Concepts from Model Theory

We introduce the fundamental concepts used throughout the thesis. They are standard, see e.g. [Hodges, 1997]. A *relational signature* τ is a (here always at most countable) set of *relation symbols* R_i , each associated with an *arity* k_i .

Structures and maps. A (*relational*) *structure* Γ over relational signature τ (also called τ -*structure*) is a set D_Γ (the *domain*) together with a relation $R_i \subseteq D_\Gamma^{k_i}$ for each relation symbol of arity k_i . If necessary, we write R^Γ to indicate that we are talking about the relation R belonging to the structure Γ . For simplicity, we denote both a relation symbol and its corresponding relation with the same symbol. For a τ -structure Γ and $R \in \tau$ it will also be convenient to say that $R(u_1, \dots, u_k)$ *holds in* Γ iff $(u_1, \dots, u_k) \in R$. We sometimes use the shortened notation \bar{x} for a vector x_1, \dots, x_n of any length, and sometimes also call relations *predicates*. Sometimes we do not distinguish between the symbol for a relational structure and its domain. The cardinality of the domain of a relational structure Γ is denoted by $|\Gamma|$.

Let Γ and Γ' be τ -structures. A *homomorphism* from Γ to Γ' is a function f from D_Γ to $D_{\Gamma'}$ such that for each n -ary relation symbol in τ and each n -tuple \bar{a} , if $\bar{a} \in R^\Gamma$, then $(f(a_1), \dots, f(a_n)) \in R^{\Gamma'}$. In this case we say that the map f *preserves* the relation R . A *strong homomorphism* f satisfies the stronger condition that for each n -ary relation symbol in τ and each n -tuple \bar{a} , $\bar{a} \in R^\Gamma$ if and only if $(f(a_1), \dots, f(a_n)) \in R^{\Gamma'}$. An *embedding* of a Γ in Γ' is an injective strong homomorphism, and an *isomorphism* is a surjective embedding. Isomorphisms from Γ to Γ are called *automorphisms*. The set of all automorphisms of a structure Γ is a group with respect to composition, and denoted by $\text{Aut}(\Gamma)$. Homomorphisms from Γ to Γ are called *endomorphisms*. The set of all endomorphisms of a structure Γ is monoid with respect to composition, and denoted by $\text{End}(\Gamma)$. It is sometimes convenient to let these mappings act on the right; it will always be possible to distinguish between these, because we then do not use brackets, i.e., we write $aef = f(e(a))$ for the application of the two mappings e and f to an element a .

A definition where we deviate from the notation in [Hodges, 1997] is that of a *substructure*; here we rather generalize the notion of a *subgraph* [Diestel, 1997] and say that a τ -structure Γ' is a *substructure* of a τ -structure Γ , iff

$D_{\Gamma'} \subseteq D_{\Gamma}$ and every relation from Γ' is a subset of the corresponding relation in Γ . Such substructures are also called *weak substructures* in the literature. The set of all finite substructures of a relational structure Γ is denoted by $\text{wSub}(\Gamma)$. We say that a structure Γ' is an *induced substructure* of Γ iff the inclusion relation is an embedding. Substructures in Hodges' book are induced substructures in our sense. The set of all finite induced substructures of a relational structure Γ is called the *age* of Γ , denoted by $\text{Age}(\Gamma)$.

The *disjoint sum* of a set of τ -structures $\Gamma_1, \Gamma_2, \dots$ is the τ -structure Γ defined on the union of the domains of these structures, where the relations in Γ are defined to be the unions of the corresponding relations of the summands Γ_i .

First-order logic. First-order formulae φ over the signature τ (or, short, τ -formulae) are inductively defined using the logical symbols of universal and existential quantification, disjunction, conjunction, negation, equality, bracketing, variable symbols and the symbols from τ . The semantics of a first-order formula over some τ -structure is defined in the usual Tarskian style. A τ -formula without free variables is called a τ -sentence. We write $\Gamma \models \varphi$ iff the τ -structure Γ is a model for the τ -sentence φ ; this notation is lifted to sets of sentences in the usual way. For a beautiful introduction to logic and model theory see [Hodges, 1997].

We can use first-order formulae over the signature τ to define relations over a given τ -structure: for a formula φ with k free variables the corresponding relation R is the set of all k -tuples satisfying the formula φ in Γ . The relational structure that contains all first-order definable relations from Γ is denoted by $\langle \Gamma \rangle_{fo}$ – thus it has a countable signature, with a relation symbol for each first-order definable relation.

If we add relations to a given structure Γ we call the resulting structure Γ' an *expansion* of Γ , and Γ is called a *reduct* of Γ' . This should not be confused with the notions of *extension* and *restriction*. Recall from [Hodges, 1997]: If Γ and Γ' are structures of the same signature, with $D_{\Gamma} \subseteq D_{\Gamma'}$, and the inclusion map is an embedding, then we say that Γ' is an *extension* of Γ , and that Γ a *restriction* of Γ' .

We now look at various syntactic restrictions of first-order logic. The first is that we only allow existential quantifiers, and only atomic negation. The corresponding formulae and sentences we call *existential*, and if they are negation-free *existential positive*. The expanded relational structure that contains all existentially (positive) definable relations in Γ is denoted by $\langle \Gamma \rangle_{\exists}$

(or $\langle \Gamma \rangle_{\exists p}$, respectively).

Of the utmost importance for constraint satisfaction is the syntactic restriction called *primitive positivity*: A first-order formula φ over the signature τ is said to be *primitive positive* (we say φ is a *p.p.-formula*, for short) iff it is of the form

$$\exists \bar{x}(\varphi_1(\bar{x}) \wedge \cdots \wedge \varphi_k(\bar{x})),$$

where $\varphi_1, \dots, \varphi_k$ are atomic formulae. Let Γ be a relational structure of signature τ . Then a p.p.-formula φ over τ with k free variables defines a k -ary relation $R \subseteq D_\Gamma^k$. We call these relations *p.p.-definable*, and denote the expanded relational structure that contains all such relations for a given Γ by $\langle \Gamma \rangle_{pp}$. In universal algebra the relational structures that are closed under primitive positive definability are called *relational clones* [Kaluzhnin and Pöschel, 1979]. It is easy to see that there is a p.p.-formula defining a relation R if and only if there exists a finite relational τ -structure S containing k designated vertices x_1, \dots, x_k such that

$$R = \{ (f(x_1), \dots, f(x_k)) \mid f: S \rightarrow \Gamma \text{ homomorphism} \}.$$

Interpretations. Definability is often too weak to capture close relationships between structures – therefore we introduce *interpretations*. The definition will be a special case of the model theoretical definition, which applies not only to structures but more generally to theories; see e.g. [Hodges, 1997]. Since we are mainly interested in ω -categorical relational structures, the definition given here suffices for our purposes.

Definition 2.1. *A τ' -structure Γ' is interpretable in a τ -structure Γ iff there exists a natural number n , called the dimension of the interpretation, and*

- a τ -formula $\delta(x_1, \dots, x_n)$ – called domain formula,
- for each m -ary relation symbol R in τ' a τ -formula $\phi_R(\bar{x}_1, \dots, \bar{x}_m)$ where the \bar{x}_i denote disjoint n -tuples of distinct variables – called the defining formulae, and
- a surjective map $f: \delta(\Gamma^n) \rightarrow D_{\Gamma'}$ – called coordinate map,

such that for all relations R in Γ' and all tuples $\bar{a}_i \in \delta(\Gamma^n)$

$$\Gamma' \models R(f(\bar{a}_1), \dots, f(\bar{a}_m)) \Leftrightarrow \Gamma \models \phi_R(\bar{a}_1, \dots, \bar{a}_m).$$

We say that B is *interpretable in Γ with finitely many parameters* iff there is a finite tuple \bar{a} of elements of Γ such that Γ' is interpretable in the expansion of Γ by the singleton relations $\{a_i\}$ for a_i in \bar{a} .

Fundamental theorems. We close this section with three theorems that express some of the main features of first-order logic. A (*first-order*) *theory* is a set of first-order sentences. The *first-order theory* $Th(\Gamma)$ of a τ -structure Γ is the set of all first-order τ -sentences that are true in Γ . All the theorems are classical, and we again recommend [Hodges, 1997] as an introduction.

Theorem 2.2 (Compactness). *Let T be a first-order theory. If every finite subset of T has a model then T has a model.*

A n -type of a theory T is a set t of formulae in n free variables such that for some model Γ of the theory and some n -tuple \bar{a} of Γ , $\Gamma \models \varphi(\bar{a})$ for all $\varphi \in t$. We say then that \bar{a} *realizes* t . We say that Γ *omits* t iff no tuple in A realizes t . (If t contains the set of *all* τ -formulae φ such that $\Gamma \models \varphi(\bar{a})$ we say that t is a *complete n -type*.) An n -type t of T is *principal* iff t contains a formula $\varphi(\bar{x})$ such that $T \cup \{\exists \bar{x}\varphi\}$ has a model, and for every formula $\psi(\bar{x}) \in t$, $T \models \forall \bar{x}(\varphi \rightarrow \psi)$.

Theorem 2.3 (Countable omitting types theorem). *Let T be a theory over a countable signature τ , and let t be a type that is not principal. Then T has a model that omits t .*

We need the following theorem to justify that it suffices to consider countable structures for constraint satisfaction. An embedding is called *elementary*, iff it preserves all first-order formulae.

Theorem 2.4 (Downward Löwenheim-Skolem theorem). *Let Γ be a structure. Then there is a countable structure Γ' with an elementary embedding in Γ .*

In particular we make use of the consequence that there is a homomorphism from a finite relational structure S to Γ if and only if there is a homomorphism from S to the countable structure Γ' from Theorem 2.4. Conceptually related to Theorem 2.4 is the following theorem.

Theorem 2.5 (Upward Löwenheim-Skolem theorem). *Let Γ be an infinite structure. Then Γ has an elementary embedding in a structure Γ' of arbitrary, but sufficiently large cardinality.*

2.2 The Theorem Ryll-Nardzewski

Finite structures are up to isomorphism determined by their first-order theory. We can not expect this for infinite structures: by the upward Löwenheim-Skolem theorem, every consistent theory with an infinite model has models of larger cardinalities. However, it might still be the case that all models of a certain cardinality are isomorphic. If this is the case for the countable models, we call the theory ω -categorical. A countable structure is called ω -categorical (or *countably categorical*), if its first-order theory is ω -categorical. This is for instance the case for the dense linear order of the rational numbers $(\mathbb{Q}, <)$. Despite the powerful theorems of this section, the class of ω -categorical structures remains somewhat mysterious, and all classification results require some additional properties (stability in e.g. [Lachlan, 1996], or homogeneity in [Cherlin, 1998]).

We first present a theorem independently discovered by Engeler, Ryll-Nardzewski, and Svenonius - for a proof again see e.g. [Hodges, 1997]. A *permutation group* G on a set D is a subgroup of the group S_D of all permutations of D . The *pointwise stabilizer* of points v_1, \dots, v_n of a permutation group G is the subgroup of permutations from G that fix each point v_1, \dots, v_n . For two n -tuples $\bar{a}, \bar{b} \in D^n$ set $\bar{a} \sim \bar{b}$ iff there exists a permutation $g \in G$ with $a_i = b_i g$ for $1 \leq i \leq n$. Note that we let permutations act on the right. Sometimes it is also convenient to let them act on the left, but then we use the standard notation for the application of a function to an element, i.e., we use brackets and write $g(a)$ for the image of a under the permutation g . The equivalence classes of the equivalence relation \sim are called the *orbits* of G on D^n . A permutation group G over an infinite set D is called *oligomorphic* iff for every $n \geq 1$ there is only a finite number of orbits of n -tuples over D .

Theorem 2.6 (Engeler, Ryll-Nardzewski, Svenonius). *Let Γ be a countable structure. Then the following are equivalent:*

1. Γ is ω -categorical.
2. Over Γ , there are only finitely many pairwise inequivalent formulae with n free variables, for all $n \geq 1$.
3. $\text{Aut}(\Gamma)$ is oligomorphic.
4. All n -types of $\text{Th}(\Gamma)$ are principal and realized in Γ .

The proof can be found in [Hodges, 1997]. It also follows that the complete n -types of $\text{Th}(\Gamma)$ are the orbits of n -tuples in $\text{Aut}(\Gamma)$. In Chapter 4

we give an interpretation of this theorem as a Galois connection between the automorphisms of Γ and the first-order definable relations in Γ . In contrast to homogeneity introduced next, the notion of ω -categoricity is fairly independent of the chosen signature. In particular, if we take reducts of an ω -categorical structure, i.e., drop certain relations in the signature, the structure stays ω -categorical. The same is true if we insert finitely many singleton-relations in an ω -categorical structure. The following is well-known.

Proposition 2.7. *Let Γ be a relational structure interpretable in the relational structure Γ' with finitely many parameters. If Γ' is ω -categorical, then Γ is also ω -categorical.*

Proof. Let n be the dimension of the interpretation of Γ in Γ' with the parameters a_1, \dots, a_m . If Γ' is ω -categorical, then the expansion Γ_1 of Γ' by the singleton relations $R_i = \{a_i\}$ for $1 \leq i \leq m$ is ω -categorical as well. To prove this we use Theorem 2.6, which implies that the orbit of (a_1, \dots, a_m) in $\text{Aut}(\Gamma')$ has a first-order definition ϕ . Suppose Γ_2 is a countable model of the first-order theory of Γ_1 . We have to find an isomorphism between Γ_1 and Γ_2 that maps the (unique) tuple (a_1, \dots, a_m) satisfying ϕ in Γ_1 , to the (unique) tuple (b_1, \dots, b_m) satisfying ϕ in Γ_2 . Consider the reducts of Γ_2 and Γ_1 to the signature of Γ' ; by ω -categoricity of Γ' there is an isomorphism i between these reducts. This isomorphism i in particular preserves the relation ϕ , and hence $(i(b_1), \dots, i(b_m))$ satisfies ϕ in Γ' and in Γ_1 . As ϕ defines an orbit in $\text{Aut}(\Gamma')$, we find an automorphism α of Γ' , that maps $(i(a_1), \dots, i(a_m))$ to (b_1, \dots, b_m) . The mapping $i\alpha$ clearly preserves all the relations in the signature of Γ' , but by construction maps (b_1, \dots, b_m) to (a_1, \dots, a_m) , and therefore also preserves the relations R_1, \dots, R_m . We conclude that any two countable models of the first-order theory of Γ_1 are isomorphic.

Finally, suppose Γ is not ω -categorical. By Theorem 2.6, there are infinitely many inequivalent formulae with m free variables in Γ . We immediately have infinitely many inequivalent formulae in Γ_1 , replacing the relation symbols of the formula by their interpretations. Thus Γ_1 would not be ω -categorical, a contradiction to what we proved in the last paragraph. \square

Example 2.8. Consider for example the structure defined on the set of closed intervals over the rational numbers, containing a relation that denotes interval-inclusion, and a unary relation that holds for all intervals that contain zero. This structure has an interpretation with a single parameter over the dense order of the rational numbers, and is therefore ω -categorical.

Example 2.9. Another important example of ω -categorical structures are countable abelian groups of finite exponent. Let A be a countable abelian

group. We assume that the group operation of A is given by a ternary relation R defined by $R(x, y, z) :\Leftrightarrow xy = z$. Then $(A; R)$ is ω -categorical if and only if it is of finite exponent, i.e., there is an $n > 1$ such that $a^n = 0$ for all $a \in A$. See A.2 in [Hodges, 1993].

2.3 Model-completeness

In this section we look at ω -categorical structures that are *model-complete*. The notion of model-completeness is an essential tool in algebraic model theory, developed by Abraham Robinson. It is again defined for arbitrary first-order theories. Here we concentrate on the special case where we deal with ω -categorical model-complete structures and their theories. In the next sections we will then look at ω -categorical structures that have even stronger properties, namely at *homogeneous* ω -categorical structures. Homogeneous structures are model-complete. All these properties have an effect on the nature of the corresponding constraint satisfaction problems.

A homomorphism that preserves all first-order formulae must be an embedding, and is called *elementary*. An ω -categorical structure Γ is called *model-complete* iff every embedding of Γ in Γ is elementary. A first-order theory is called a $\forall\exists$ -theory iff every sentence of the theory is equivalent to a sentence that starts with universal quantifiers, followed by existential quantifiers, and ends with a quantifier-free formula.

Theorem 2.10. *Let Γ be an ω -categorical relational structure. Then the following are equivalent:*

1. Γ is model-complete, i.e., every embedding of Γ in Γ is elementary.
2. The first-order theory of Γ is an $\forall\exists$ -theory.
3. Every formula is over Γ equivalent to an universal formula.
4. Every formula is over Γ equivalent to an existential formula.

Proof. The equivalence of 1 and 2 can be found in [Cameron, 1990]; the equivalence of 1 and 3 follows from Theorem 7.3.1 in [Hodges, 1997], and the equivalence of 1 and 4 follows from Corollary 5.4.5 in [Hodges, 1997], which states more generally that a first-order formula ϕ is preserved by embeddings between models of a theory T if and only if ϕ is equivalent modulo T to an existential formula. Let T be the theory of Γ . Then by ω -categoricity the

embeddings between the models of T correspond to embeddings of Γ in Γ , and this implies the equivalence of 1 and 4. \square

For illustration, we show an example of an ω -categorical structure that is not model-complete (from [Chang and Keisler, 1977]). Consider a countable structure Γ that contains an equivalence relation E with infinitely many infinite equivalence classes C_1, C_2, \dots , and a single class containing only one element a . The function that maps a to C_1 and the elements of C_i bijectively to C_{i+1} is an embedding of Γ in Γ . But it is not elementary, since the first-order formula $\neg\exists y : E(x, y) \wedge x \neq y$ only holds for $x = a$.

Another example is the dense linear order defined on the interval $[-1, 1]$ of rational points. It is ω -categorical: if we add one singleton-relation containing the element -1 and one containing the element 1 , we obtain a homogeneous structure over a finite signature. But it is not model-complete, since the mapping $x \mapsto (x + 1)/2$ preserves the order, but does not preserve the first-order formula $\neg\exists y : y < x$, which holds for $x = -1$ but not for $x = 0$.

2.4 Fraïssé's Theorem

A relational structure Γ is called *homogeneous* (in the literature sometimes *ultra-homogeneous*) if every isomorphism between two finite induced substructures can be extended to an automorphism of Γ . Prominent examples of countable homogeneous structures are the dense linear order $(\mathbb{Q}, <)$, and the *Rado graph* \mathbf{R} (also called the *Random graph*). The Rado graph can be defined as the unique (up to isomorphism) model of the almost-sure theory of finite random graphs. Homogeneous structures have been classified for graphs [Lachlan and Woodrow, 1980], for tournaments, for posets [Schmerl, 1979], and finally digraphs [Cherlin, 1998] (there are uncountably many homogeneous digraphs; see Section 2.6). For homogeneous structures with arbitrary relational signatures a classification is not yet known.

The *age* of a relational structure Γ with the signature τ is the set of all finite τ -structures that embed in Γ , denoted by $\text{Age}(\Gamma)$. An important property of countable homogeneous structures is their characterization by *amalgamation classes*. A class of finite relational structures \mathcal{C} is an amalgamation class if \mathcal{C} is nonempty, closed under isomorphisms and taking induced substructures, and has the *amalgamation property*. The amalgamation property says that for all $A, B_1, B_2 \in \mathcal{C}$ and embeddings $e_1 : A \rightarrow B_1$ and $e_2 : A \rightarrow B_2$ there exists $C \in \mathcal{C}$ and embeddings $f_1 : B_1 \rightarrow C$ and $f_2 : B_2 \rightarrow C$ such that

$f_1e_1 = f_2e_2$. We call C an *amalgam of B_1 and B_2 over A* .

Theorem 2.11 (of [Fraïssé, 1986]). *A countable class \mathcal{C} of finite relational structures with countable signature is the age of a countable homogeneous structure if and only if \mathcal{C} is an amalgamation class. If this is the case, the countable structure is unique up to isomorphism, and called the Fraïssé-limit of \mathcal{C} , denoted by $Fl(\mathcal{C})$.*

Since amalgamation classes are closed under taking induced substructures, they can be defined by a set of *forbidden finite induced substructures*. For a set of finite structures \mathcal{N} over τ we denote by $\text{Forb}(\mathcal{N})$ the set of finite τ -structures S such that no structure in \mathcal{N} is embeddable in S .

The class of all finite graphs clearly is an amalgamation class. The Fraïssé-limit is often called the *Rado graph* \mathbf{R} . Another example is the set of all finite total orders, i.e., transitive acyclic tournaments. The Fraïssé-limit is isomorphic to the dense linear order of the rational numbers $(\mathbb{Q}, <)$. We could describe the age of this digraph by $\text{Forb}(\{C_3, C_2, C_1, I_2\})$, where C_k is the directed circle on k vertices (and C_1 is by definition a single-vertex with a self-loop), and I_2 the empty graph on two vertices.

If the signature is finite, as in the two examples above, the Fraïssé-limit is ω -categorical. In this case we find other pleasant properties. A relational structure Γ has *quantifier elimination*, if every formula is in Γ equivalent to a quantifier-free formula. The following theorem is stated for finite signatures in theorem [Hodges, 1997], but the generalization below can be proved in the same way.

Theorem 2.12. *Let Γ be a countable τ -structure, where τ contains finitely many k -ary relation symbols for each $k \geq 1$. Then Γ is homogeneous if and only if it is ω -categorical and has quantifier elimination.*

For ω -categorical structures without any restriction to the signature, quantifier elimination is equivalent to homogeneity.

Theorem 2.13 (see [Cameron, 1990]). *If Γ is a countable ω -categorical relational structure, then Γ is homogeneous if and only if it has quantifier elimination.*

Every ω -categorical structure Γ can be made homogeneous, by expanding Γ by all existentially definable relations – see e.g. Satz 9.2.2 in [Rothmaler, 1995].

Proposition 2.14. *If Γ is a countable ω -categorical relational structure, then $\langle \Gamma \rangle_{\exists}$ is homogeneous.*

Proof. By Fraïssé’s theorem we only have to check that $\text{Age}(\langle \Gamma \rangle_{\exists})$ has the amalgamation property. Let \bar{a} be a tuple of elements from Γ , let B_1, B_2 be induced substructures of Γ and $e_1 : \bar{a} \rightarrow B_1$ and $e_2 : \bar{a} \rightarrow B_2$ be embeddings. Since there are relation symbols for every existential formula in the signature, there is a relation R_1 that holds on the tuple $e_1(\bar{a})$ and corresponds to the structure B_1 where the points from $B_1 - e_1(\bar{a})$ are existentially quantified. We also have a relation R_2 corresponding to B_2 where the points from $B_2 - e_2(\bar{a})$ are existentially quantified. Since e_1 and e_2 are *isomorphic* mappings, these relations also hold on \bar{a} , and using ω -categoricity we can find an extension C of \bar{a} with embeddings $f_1 : B_1 \rightarrow C$, $f_2 : B_2 \rightarrow C$ such that $f_1 e_1 = f_2 e_2$. \square

The expansion $\langle \Gamma \rangle_{\exists}$ of an ω -categorical structure Γ by all existential formulae can be characterized in terms of a closure condition. This is a direct consequence of the Łos-Tarski theorem, see [Hodges, 1997]. (Similar closure conditions can be formulated for the expansion of Γ by all first-order, by all existential, by all existential positive, or by all primitive positive formulae. A summary on such results can be found in Section 7.1.)

Proposition 2.15. *Let Γ be an ω -categorical structure. Then a relation R is existentially definable in Γ if and only if R is preserved by embeddings of Γ in Γ .*

Proof. Since R is in particular preserved by all automorphisms, the relation R has by ω -categoricity of Γ a definition by a first-order formula ϕ . This follows from Theorem 2.6 of Ryll-Nardzewski and will be discussed in detail in Section 4.4. By the Łos-Tarski theorem, ϕ is equivalent to an existential formula modulo $\text{Th}(\Gamma)$ if and only if R is preserved by embeddings between models of $\text{Th}(\Gamma)$. In fact, the proof of the theorem of Łos-Tarski, or a standard model-theoretic argument based on the theorem of Löwenheim-Skolem (Theorem 2.4) shows that it suffices to consider embeddings between the countable models of $\text{Th}(\Gamma)$ only. Since any two countable models Γ_1 and Γ_2 of $\text{Th}(\Gamma)$ are by ω -categoricity isomorphic, every embedding of Γ_1 in Γ_2 can be seen as an injective strong endomorphism. \square

Sometimes an ω -categorical structure Γ has an expansion that is homogeneous and has a finite signature. In this case we call Γ *finitely homogeneous*. The templates that we study of the constraint satisfaction problems that we study later in this thesis will mostly be finitely homogeneous.

2.5 Strong and Free Amalgamation

Sometimes the age of a countable structure satisfies a stronger form of the amalgamation property: the *strong amalgamation property*. An even stronger form is the *free amalgamation property*. These properties of the age of a relational structure will have an effect on the nature of the corresponding constraint satisfaction problems.

Strong Amalgamation. First we recall: a class of finite structures \mathcal{C} has the *amalgamation property* if for all $A, B_1, B_2 \in \mathcal{C}$ and embeddings $e_1 : A \rightarrow B_1$ and $e_2 : A \rightarrow B_2$ there exists $C \in \mathcal{C}$ and embeddings $f_1 : B_1 \rightarrow C$ and $f_2 : B_2 \rightarrow C$ such that $f_1 e_1 = f_2 e_2$. For *strong amalgamation* we strengthen this and require that if there are $b_1 \in B_1$ and $b_2 \in B_2$ with $f_1(b_1) = f_2(b_2)$ then there is an $a \in A$ such that $e_1(a) = b_1$ and $e_2(a) = b_2$. In other words, we can amalgamate any two structures B_1, B_2 over A without identifications outside A .

We give an alternative characterization of strong amalgamation for the age of a homogeneous structure. For that we need the notion of *algebraic closure*. Let Γ be an ω -categorical structure with relational signature τ , and A a finite subset of elements from Γ . The *algebraic closure* $\text{acl}(A)$ of A is the set of all those elements of Γ that lie in finite orbits of the pointwise stabilizer of A . By Theorem 2.6 this equals the set of elements b for which we have a first-order τ -formula $\phi(x, \bar{y})$ and a tuple \bar{a} from A such that $\Gamma \models \phi(b, \bar{a}) \wedge \exists_{\leq n} x. \phi(x, \bar{a})$ for some finite n . We say that A is *algebraically closed* if $\text{acl}(A) = A$.

Theorem 2.16 (see [Cameron, 1996]). *Let Γ be a homogeneous ω -categorical structure. Then the following are equivalent:*

- *The age of Γ has the strong amalgamation property.*
- *Every finite set $A \subseteq \Gamma$ is algebraically closed.*

We now mention an observation that describes the effect of the strong amalgamation property on the nature of the constraint satisfaction problem. Let $\text{CSP}(\Gamma)$ be the set of all finite structures that homomorphically map to Γ ; recall that $\text{wSub}(\Gamma)$ denotes the set of all finite structures isomorphic to a substructure of Γ .

Proposition 2.17. *If the age of a relational structure Γ has the strong amalgamation property, then $\text{CSP}(\Gamma) = \text{wSub}(\Gamma)$.*

Proof. Let S be a structure with a homomorphism f to Γ . We have to show that we can also find an injective homomorphism to Γ . Assume that S contains two points u, v from S such that $f(u) = f(v) = p$. By Theorem 2.16 the set $f(S) - \{p\}$ is algebraically closed. Hence we can find a point p' distinct from p that has the same type as p over the parameters $f(S) - \{p\}$. We modify the homomorphism f and map v to p' . If we iterate this, we finally obtain a function f' that is injective and still preserves all relations that hold in S . \square

Free amalgamation. We sometimes need an even stronger version of amalgamation, called *free amalgamation*. The idea here is that we might be able to amalgamate two structures without adding any new relations at all. Let Γ be homogeneous and A, B_1, B_2, e_1, e_2 as above. By homogeneity we may suppose that the embeddings e_1 and e_2 are inclusions. Then the *free amalgam* C of B_1 and B_2 is the structure defined on the vertices of B_1 and B_2 , where the relations on C are the unions of the relations in B_1 and B_2 . Clearly, free amalgamation implies strong amalgamation.

If the signature consists of a single binary relation and we are talking about directed graphs, we describe as in [Cherlin, 1998] how to specify a free amalgamation class with a set of tournaments \mathcal{T} . A *tournament* is a directed graph where between any two distinct vertices there is precisely one directed arc. First define:

$$\mathcal{A}(\mathcal{T}) := \{D \mid \text{if } T \subseteq D \text{ is a tournament then } T \in \mathcal{T}\} \quad (2.1)$$

The set of graphs $\mathcal{A}(\mathcal{T})$ is a free amalgamation class, and for every free amalgamation class \mathcal{A} there exists a set \mathcal{T} of tournaments such that $\mathcal{A} = \mathcal{A}(\mathcal{T})$.

In later sections the following observation will be important. If the age of a structure Γ with a single binary relation has the free amalgamation property, it equals the set of finite graphs that homomorphically map to Γ .

Proposition 2.18. *If the age of a structure Γ with a single binary relation has the free amalgamation property, then $\text{Age}(\Gamma) = \text{wSub}(\Gamma) = \text{CSP}(\Gamma)$.*

Proof. Free amalgamation implies strong amalgamation, and hence $\text{CSP}(\Gamma) = \text{wSub}(\Gamma)$ by Proposition 2.17. So we only have to show that $\text{wSub}(\Gamma) \subseteq \text{Age}(\Gamma)$. Let S be an induced subgraph of Γ , and let S' be the substructure obtained by deleting an edge uv in S . We can then freely amalgamate $S - \{u\}$

G^c :	The complement of the graph or digraph G .
K_n :	The complete graph on n vertices ($n \geq 1$, and n might be ∞).
I_n :	An <i>independent set</i> of size n , i.e., the complement K_n^c of K_n .
C_n :	The oriented cycle on n vertices.
S_n :	The star graph on $n + 1$ vertices.
$G_1[G_2]$:	The <i>composition</i> or <i>wreath product</i> of G_1 and G_2 : each vertex of G_1 is replaced by a copy of G_2 , and arcs between distinct copies of G_2 are controlled by the edges of G_1 .
$[G_1, G_2]$:	The disjoint sum of G_1, G_2 that is extended by arcs $a \rightarrow b$ whenever $a \in G_1, b \in G_2$. If G_1 and G_2 and $[G_1, G_2]$ are understood as undirected graphs we correspondingly insert undirected edges.

Figure 2.1: Convenient notation for the description of digraphs.

and $S - \{v\}$, and thus $S' \in \text{Age}(\Gamma)$. By induction it follows that every finite subgraph of Γ is an induced subgraph of Γ . \square

For general signature, Proposition 2.18 is not true. Consider the Rado-graph $\mathbf{R} = (V; E)$, and introduce a second binary relation E' such that for all vertices u, v from V the relation $E'(u, v)$ holds if and only if $E(u, v)$ holds. Like the Rado-graph, this structure has the free amalgamation property. But $\text{CSP}((V; E, E'))$ contains graphs that have edges in E but no edges in E' , and are thus not induced substructures of $(V; E, E')$.

2.6 Homogeneous Digraphs

In this section we consider countable homogeneous digraphs. Many of them serve as examples in later sections. Although there are uncountably many, there is a classification by Cherlin [Cherlin, 1998] that shows that all but a countable number of well-understood homogeneous digraphs have the free amalgamation property (Section 2.5). First we fix some useful notation for graphs and digraphs. For adjacency between nodes a and b we use the infix-notation $a \rightarrow b$.

The countable homogeneous undirected graphs. Lachlan and Woodrow [Lachlan and Woodrow, 1980] showed that every countable homogeneous graph is either the Rado-graph, the Fraïssé-limit of all K_n -free graphs, the

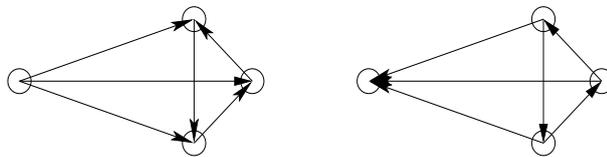


Figure 2.2: The forbidden induced subgraphs $[I_1, C_3]$ and $[C_3, I_1]$ of $S(2)$.

complete n -partite graphs, or a complement of these. In the notation above they showed that for every infinite countable homogeneous graph G either G or G^c has one of the following forms:

1. $I_m[K_n]$ where m or n is ∞ ;
2. $\text{Fl}(\text{Forb}(K_{n+1}))$, the universal graph for the class of all graphs omitting K_{n+1} ;
3. $\text{Fl}(\mathcal{G})$, the Fraïssé-limit of the set of all finite graphs \mathcal{G} (the *Rado-graph*).

Unlike the Rado-graph, which is also called the Random-graph, since it is the countable model of the ω -categorical almost-sure theory of finite graphs, the graph $\text{Fl}(\text{Forb}(K_{n+1}))$ is *not* a model of the almost-sure theory of the K_{n+1} -free graphs. However, the almost-sure theory of K_{n+1} -free graphs surprisingly *is* ω -categorical, and the countable model is the universal graph for the class of all n -colorable graphs [Kolaitis et al., 1987], called $\mathbf{D}(n)$ there, which is not homogeneous.

The countable homogeneous tournaments. We start with the homogeneous tournaments, which have been classified by Lachlan [Lachlan, 1984]. There are only three countable homogeneous tournaments: The dense linear order on the rational numbers \mathbb{Q} , the dense local order $S(2)$, and the tournament T^∞ that is universal for the set of all finite tournaments. We have a closer look at $S(2)$. It is defined as the up to isomorphism unique structure described by the following proposition. The result is well-known, but we do not know of any reference containing the presented proof.

Proposition 2.19. *Let \mathcal{N} be the set containing the two digraphs shown in Figure 2.2. Then the set of digraphs in $\text{Forb}(\mathcal{N})$ is an amalgamation class, and its Fraïssé-limit is called $S(2)$ and is isomorphic to the following structures:*

- (i) A partition of the rational numbers \mathbb{Q} into two dense subsets Q_1 and Q_2 (i.e., for every rational number we will find sequences in Q_1 and in Q_2 that converge against this number), where two points a, b are linked by an arc iff $a < b$ and they belong to the same class Q_i , or $b < a$ and they belong to different classes.
- (ii) A countable dense set of points on the unit circle without antipodal points, where two vertices a, b are linked by an arc iff the directed line from a to b passes the origin on the right side.

Proof. Any digraph described in (i) clearly does not contain subgraph from \mathcal{N} . On the other hand, consider a digraph C from $\text{Forb}(\mathcal{N})$. If we fix a certain point a in C we see that the sets $B_1 = \{b \mid a \rightarrow b\}$ and $B_2 = \{b \mid b \rightarrow a\} \cup \{a\}$ have to be linearly ordered by the relation ' \rightarrow '. It is easy to check that if we reverse the edges between both of these linearly ordered subsets, the resulting tournament will be a linear order. We take an arbitrary embedding of this linear order into the rational numbers, and found an embedding of C into a structure described in (i).

We now specify an isomorphism from such a structure to a structure described in (ii). Every rational number can be uniquely written as u/v for an integer u and a relatively prime natural number v . Zero is represented as $0/1$. We consider these numbers as vectors (u, v) for points from Q_1 and $(u, -v)$ for points from Q_2 , and scale these vectors to unit length. The resulting structure on these vectors is a structure described in 2, where there is an arc between vectors (u_1, v_1) and (u_2, v_2) iff $u_1/v_1 < u_2/v_2$ and $v_1v_2 > 0$, or $u_1/v_1 > u_2/v_2$ and $v_1v_2 < 0$. But $u_1v_2 - u_2v_1 > 0$ expresses the fact that the directed line between point (u_1, v_1) and (u_2, v_2) has the origin on the left side.

We finally note that this structure is homogeneous and unique up to isomorphism. By Theorem 2.11 it suffices to show that its age has the amalgamation property. Let A and B be two finite induced substructures of the structure described in (ii). We can easily check that the following structure C is an amalgam of A and B : C is defined on the union of the domains of A and B , and two points a and b in C are linked if the directed line from a to b passes the origin on the right side. \square

Another proof that $S(2)$ is homogeneous can be found in [Cameron, 1981]. Recall from the introduction that two digraphs D_1 and D_2 are *switching-equivalent* iff there exists a subset S of vertices of D_1 such that the graph that arises from D_1 by switching all arcs between S and its complement in

D_1 is isomorphic to D_2 . The age of the homogeneous structure $S(2)$ can also be characterized as the class of finite graphs that is switching equivalent to linear orders.

The countable homogeneous digraphs. They have been classified by Cherlin [Cherlin, 1998], and there are uncountably many. But the classification shows that the age of all but a countable number of well-understood homogeneous digraphs has the free amalgamation property. Free amalgamation classes of digraphs can be specified as $\mathcal{A}(\mathcal{T})$ where \mathcal{T} is a set of tournament types, see Section 2.5. Henson showed that the partial order of isomorphism types of finite tournaments ordered by embeddability contains an infinite antichain [Henson, 1972]. For each tournament T in this antichain $\mathcal{A}(\mathcal{T})$ has the free amalgamation property, and all of these classes $\mathcal{A}(\mathcal{T})$ are distinct. Thus there is an uncountable number of classes having the free amalgamation property and by Fraïssé’s theorem an uncountable number of different homogeneous digraphs. But we will see in Section 3.2 that the constraint satisfaction problem of such structures is easy to analyze.

As in [Cherlin, 1998], we group the homogeneous digraphs without free amalgamation into three classes. The first class contains the countable homogeneous tournaments that we already know from the last paragraph in this section, and the empty graph on a countable number of vertices.

The second class contains *imprimitive* structures, i.e., structures that have a first-order definable nontrivial equivalence relation. In all cases except the first the definable equivalence relation corresponds to the pairs of vertices that are not connected. There are four types, classified in [Cherlin, 1987].

1. Wreathed (Composite): $I_n[\mathbb{Q}]$, $I_n[S(2)]$, $I_n[T^\infty]$ and $\mathbb{Q}[I_n]$, $S(2)[I_n]$, $T^\infty[I_n]$, $C_3[I_\infty]$, where n might be infinite.
2. Twisted: $\hat{\mathbb{Q}}$, \hat{T}^∞ . The graph $\hat{\mathbb{Q}}$ is a variant of $S(2)$ where every point has an unconnected antipodal. The graph \hat{T}^∞ is universal for the class of all graphs where the pairs of disconnected vertices form an equivalence relation with classes of size two, and the union of two such classes is a copy of C_4 .
3. Generified n -partite (denoted by $n * I_\infty$ in [Cherlin, 1987]): Starting from $K_n[I_\infty]$ we orient the undirected edges randomly.
4. Semigeneric (denoted by $\infty * I_\infty$ in [Cherlin, 1987]): Starting from the undirected homogeneous graph $K_\infty[I_\infty]$ we orient the edges arbitrarily

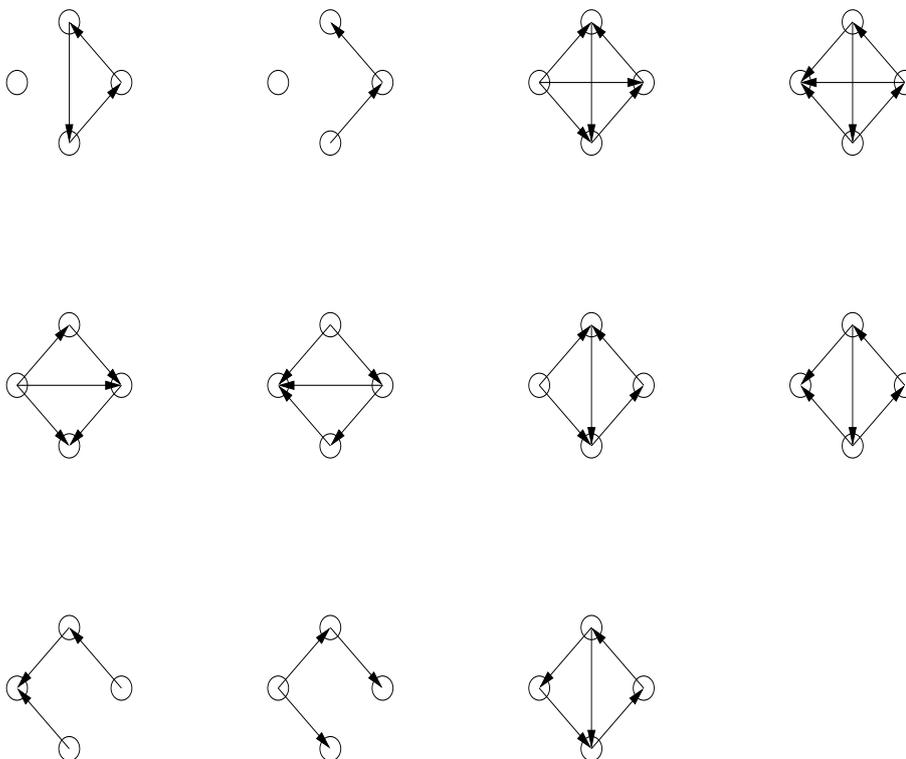


Figure 2.3: The forbidden induced subgraphs of $P(3)$.

such that for two distinct parts U and V and distinct vertices $u_1, u_2 \in U$ and $v_1, v_2 \in V$ the number of edges from u_1 or u_2 to v_1 or v_2 is even.

Finally we have the *exceptional class* containing the digraphs $S(3)$, \mathbb{P} , and $P(3)$. The graph $S(3)$ denotes the set of points lying at a rational angle ϕ on the unit circle, and two points a, b are joined by an arc $a \rightarrow b$ iff the angle from a to b is in the range $(0, 2\pi/3)$. Equivalently we can produce the graph similarly to $S(2)$, starting from a partition of \mathbb{Q} into three dense sets Q_1, Q_2, Q_3 . We identify the two possible orientations of an edge with $+1$ and -1 , and 0 represents the absence of an edge. We then cyclically shift the edges between Q_i and Q_j by $j - i$ (the indices are integers modulo 3). For the proof that $S(3)$ is homogeneous we refer to [Cherlin, 1987].

The countable homogeneous partial order $\mathbb{P} = \text{Fl}(\mathcal{P})$ is the Fraïssé-limit of the class of all finite partial orders \mathcal{P} . We call a subset P of \mathbb{P} *dense*, if for any pair a, b in \mathbb{P} with $a \rightarrow b$ we have an element $c \in P$ such that $a \rightarrow c \rightarrow b$. $P(3)$ is the analogue of $S(3)$, based on \mathbb{P} instead of \mathbb{Q} . Let P_0, P_1, P_2 be

three dense subsets of \mathbb{P} . The structure $P(3)$ is defined on $P_0 \uplus P_1 \uplus P_2$. The relation \rightarrow on $P(3)$ restricted to P_i is defined as in \mathbb{P} . Again we identify the complete 2-types between elements of distinct parts with $\{-1, 0, +1\}$, and cyclically shift these types between P_i and P_j by $j - i$. A proof that $P(3)$ is indeed homogeneous can be found in [Cherlin, 1998] in Section 5.2. Cherlin also specified the age of $P(3)$ by the forbidden induced substructures shown in Figure 2.3 (their description can be found in the proof of Proposition 24 on page 126f in [Cherlin, 1998]).

2.7 Tree-like Structures

In this section we have a look at various ω -categorical structures that have been investigated in model theory and for infinite permutation groups. In later sections they will show up again as the templates for various constraint satisfaction problems.

2.7.1 Boron Trees

A *boron tree* [Cameron, 1996] is a finite tree in which all vertices have degree one (*hydrogen- or H-atoms*) or degree three (*boron- or B-atoms*). On the H-atoms of a boron tree we can define a quaternary relation $uv|xy$ that holds when the paths joining u to v and c to d are disjoint (called *the set of quartets* in [Steel, 1992] in the application for phylogenetics; this corresponds to a *D-relation* in [Adeleke and Neumann, 1985] in the application for infinite permutation groups). This relation holds for exactly one of the three partitions of a 4-set into two 2-sets, and it already determines the boron tree up to isomorphism: this is an exercise in [Cameron, 1996]; also see [Adeleke and Neumann, 1985].

The class of all structures \mathcal{D} with a quaternary relation that stem from a boron tree as defined above is an amalgamation class. Let D be the Fraïssé-limit of \mathcal{D} . This structure is mentioned in [Cameron, 1996] since its automorphism group has many remarkable properties. A permutation group on Ω is called *k-transitive*, if there is only one orbit of k -tuples of Ω . It is called *k-homogeneous*, if there is only one orbit of k -subsets of Ω . Note that this terminology from permutation group theory conflicts with the notion of homogeneity in model theory. Therefore homogeneous structures from model theory are sometimes called *ultra-homogeneous*, and permutation

groups that are k -homogeneous for every k are called *highly homogeneous*. The structure B is ultra-homogeneous, 5- but not 6-homogeneous, and 3- but not 4-transitive.

If we fix a point a in D and consider the ternary relation ‘:’ defined by $x : yz \Leftrightarrow ax|yz$ we again obtain a homogeneous structure (this is a C -set in [Adeleke and Neumann, 1985]). The age of this structure now contains the finite structures T that come from finite rooted trees, and the relation $x : yz$ says that the common ancestor of y and z is below the common ancestor of x, y and z in the tree T (the relation ‘:’ on a rooted tree is called the *set of rooted triples* in [Bryant, 1997, Ng et al., 2000]).

2.7.2 Semilinear Orders

Concerning semilinear orders we use the definitions and notation as in [Adeleke and Neumann, 1985, Cameron, 1996].

Definition 2.20. *A partial order (Λ, \leq) is called semilinear iff every two elements have an upper bound, and for all x , the set $\{y \mid y \geq x\}$ is linearly ordered. A semilinear order is proper, if it contains two incomparable elements, i.e., a pair b, c such that neither $b \leq c$ nor $c \leq b$.*

We say that a semilinear order *has least upper bounds* if for any two incomparable elements there is a unique least upper bound. A partial order is *dense*, if for any two elements $a \leq b$ there is an element $c \in D$ such that $a \leq c \leq b$. These and related structures and the corresponding automorphism groups were studied and classified in [Droste, 1985] – he called them *trees*. This should not be confused with infinite trees as we know them from graph theory or model theory. We referred to these objects as *tree-like*, since they are only branching downwards. But note that formally these are not trees since there might be elements without a unique parent.

We give an explicit construction of a dense proper semilinear order [Cameron, 1996, Adeleke and Neumann, 1985]. The domain of the structure is the set of all non-empty finite sequences $a = (q_0, q_1, \dots, q_{n-1})$ of rational numbers. Let $a \leq b$ if either

- b is an initial subsequence of a , or
- $b = (q_0, \dots, q_{n-1}, q_n)$ and $a = (q_0, \dots, q_{n-1}, q'_n, q_{n+1}, \dots, q_m)$, where $q_n < q'_n$.

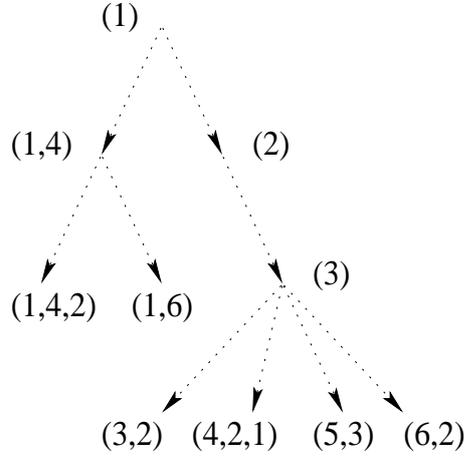


Figure 2.4: A small substructure of Λ .

We call this dense semilinear order Λ throughout the thesis. Also see Figure 2.4.

For the constraint satisfaction problems of the structures discussed here, the choice of the signature is essential for their computational complexity. We now use notation that has been used for finite trees in computational linguistics and computer science (e.g. in [Backofen et al., 1995]). We say that a dominates b and write $a \triangleleft^* b$ for $a \geq b$ (In computer science, trees grow from top to bottom, and correspondingly the notation is reversed). We say that a is disjoint to b and write $a \perp b$ if neither $a \triangleleft^* b$ nor $b \triangleleft^* a$, and $a \perp\!\!\!\perp b$ if $a \perp b$ or $a = b$. Note that \perp and $\perp\!\!\!\perp$ are not primitive positive definable in (Λ, \leq) . We therefore also consider the structure $(\Lambda; \triangleleft^*, \perp, \neq)$. As we will see in Section 5.5, with this signature all binary first-order formulae are equivalent to a primitive positive formula. An alternative choice of a structure with the same set of primitive positive definable relations would have been $(\Lambda; \triangleleft^* \cup \perp, \triangleleft^* \cup \triangleright, \neq)$.

If a is lexicographically smaller than b , but not $a \triangleleft^+ b$, we write $a \prec b$. The relation $a \preceq b$ holds iff either $a \prec b$ or $a = b$. These relations are not first-order definable with the other relations mentioned so far. We will see in this Section below that the 2-types in this expanded structure $(\Lambda; \triangleleft^+, \prec)$ are \triangleleft^+, \prec , their inverse relations, and equality. Correspondingly we have 2^5 inequivalent first-order definable binary relations in Λ . We will show in Section 5.5 that all of them have primitive positive definitions in the structure $(\Lambda; \triangleleft^*, \preceq, \neq, \perp)$.

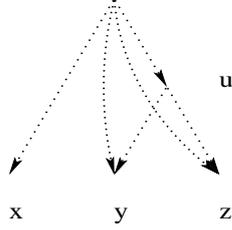


Figure 2.5: In this induced substructure of $(\Lambda; \triangleleft^+)$ the relation $x : yz$ holds.

We can show the ω -categoricity of Λ by adding the relation ‘:’ to the signature (defined below as in [Droste et al., 1991]. Also see Figure 2.5), and show that the age of the resulting structure has the amalgamation property [Droste et al., 1991]. Since the signature is finite, ω -categoricity follows from Theorem 2.12.

$$x : yz \Leftrightarrow_{def} \exists u. u \perp x \wedge u \triangleleft^+ y \wedge u \triangleleft^+ z \quad (2.2)$$

The name-clash with the definition of the C-relation above is no coincidence. If we consider the class of finite structures containing pairwise disjoint points in Λ , carrying the relation $x : yz$, it has the amalgamation property, and the Fraïssé-limit is isomorphic to the structure with the ternary relation ‘:’ derived from the structure B above. By Theorem 2.13 this structure also has quantifier elimination. Thus, if we have a binary first-order definable relation in $(\Lambda; \triangleleft^+, \prec)$, it is equivalent to a boolean combination of the relations \triangleleft^* , \prec , ‘:’. Every projection of the ternary relation ‘:’ to two variables is trivial, and therefore the binary formula is also equivalent to a boolean combination of \triangleleft^+ , \prec , and equality.

2.7.3 Dominance and Immediate Dominance

The semilinear order Λ is ‘everywhere dense’. We next introduce a semilinear order containing certain points that are above countably many maximal lower elements. The order will be ‘dense except below these points’. Again we could describe the structure via its first-order theory, since it will be ω -categorical. But there is an explicit construction, which is similar to the last construction. Again the domain of the structure is the set of all non-empty finite sequences $a = (q_1, q_2, \dots, q_n)$ of rational numbers. Let $a \triangleleft^+ b$ if either

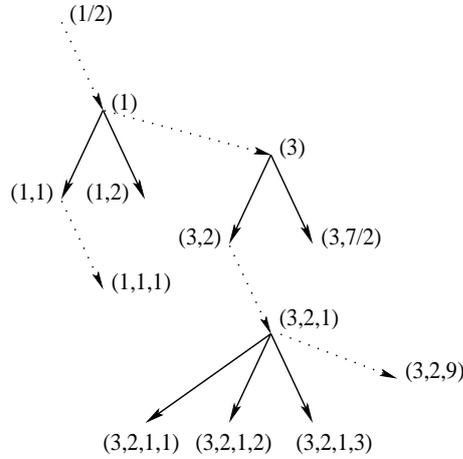


Figure 2.6: A finite (weak) substructure of Δ .

- a is a proper initial subsequence of b , or
- $a = (p_0, \dots, p_{2n-1}, p_{2n})$ for $n \geq 0$, and $b = (p_0, \dots, p_{2n-1}, p'_{2n}, p_{2n+1}, \dots, p_m)$, where $p_{2n} < p'_{2n}$.

We call this dense semilinear order Δ throughout the thesis. Again, for later chapters the choice of the signature for Δ is crucial. Besides the relation ' \triangleleft^+ ' for dominance we use ' \triangleleft ' to denote *immediate dominance*, which is the relation $\{(a, b) \mid \forall c. c \triangleleft^+ b \rightarrow (c \triangleleft a \vee c=a)\}$. This means, $a \triangleleft b$ if and only if a is a prefix of b of odd length $|a|$ and $|a| = |b| - 1$. In pictures we draw dominance edges as dotted arcs and immediate dominance as solid arcs. See Figure 2.6 for a finite substructure of Δ .

Since to our knowledge this structure has not yet been described in the literature, and since we need this structure to describe templates for constraint satisfaction problems, we give a proof of its ω -categoricity. We can show ω -categoricity of several other structures needed in Chapter ?? in the same way. Again we expand Δ by the ternary relation ' \cdot ', defined as above in (2.2), and additionally by the unary relation 'even' that holds for sequences x in Δ of even length, and has the following first-order definition: $\text{even}(x) \Leftrightarrow \exists z : z \triangleleft x$. Then we show that the age of the resulting structure $(\Delta; \triangleleft, \triangleleft^+, \cdot, \text{even})$ has the amalgamation property. Theorem 2.12 then implies ω -categoricity of Δ . The structure Δ is also interesting because the amalgamation is not a *strong* amalgamation.

To show that the age of $(\Delta; \triangleleft, \triangleleft^+, \cdot, \text{even})$ has the amalgamation property we use the following Lemma, mentioned in [Cherlin, 1998] on page 9. It states that to verify the amalgamation property it always suffices to check the so-called *two-point amalgamation property*, which says that for all A, B_1, B_2 in a class of finite relational structures \mathcal{C} , and all embeddings $e_1 : A \rightarrow B_1$ and $e_2 : A \rightarrow B_2$, where $|B_1 - e_1(A)| = |B_2 - e_2(A)| = 1$, there exists $C \in \mathcal{C}$ and embeddings $f_1 : B_1 \rightarrow C$ and $f_2 : B_2 \rightarrow C$ such that $f_1 e_1 = f_2 e_2$.

Lemma 2.21 (Two-point Amalgamation). *Let \mathcal{C} be a class of finite relational structures that is closed under isomorphisms and induced substructures, and has the two-point amalgamation property. Then \mathcal{C} is an amalgamation class.*

Proof. We have to show that \mathcal{C} has the amalgamation property. Let A, B_1, B_2 be structures in \mathcal{C} and let $e_1 : A \rightarrow B_1$ and $e_2 : A \rightarrow B_2$ be embeddings. We have to show the existence of a structure $C \in \mathcal{C}$ and embeddings $f_1 : B_1 \rightarrow C$ and $f_2 : B_2 \rightarrow C$ such that $f_1 e_1 = f_2 e_2$. Let b_1^1, \dots, b_1^k and b_2^1, \dots, b_2^k be the elements from $B_1 - e_1(A)$ and $B_2 - e_2(A)$, respectively. We prove by induction, that for all $0 \leq i \leq k_1, 0 \leq j \leq k_2$ we find structures $C_{i,j}$ such that $C_{i,j}$ is the amalgam of the structures induced by $e_1(A) \cup \{b_1^1, \dots, b_1^i\}$ and $e_2(A) \cup \{b_2^1, \dots, b_2^j\}$. Note that $C_{0,0}$ is a copy of A , and that $C_{k_1,0} = B_1$ and $C_{0,k_2} = B_2$. In particular we know that $C_{i,0}$ and $C_{0,j}$ exist for $0 \leq i \leq k_1, 0 \leq j \leq k_2$. Initially, we form a two-point amalgamation of $C_{1,0}$ and $C_{0,1}$, and obtain $C_{1,1}$. In general, we can amalgamate $C_{i-1,j}$ with $C_{i,j-1}$ and obtain $C_{i,j}$. Finally, by induction, we also find C_{k_1,k_2} , which is the amalgam of B_1 and B_2 that we were looking for. \square

Proposition 2.22. *The age of the structure $(\Delta; \triangleleft, \triangleleft^+, \cdot, \text{even})$ has the amalgamation property.*

Proof. By Lemma 2.21 it suffices to check the two-point amalgamation property. Let A, B_1, B_2 be finite substructures of Δ , and $e_1 : A \rightarrow B_1$ and $e_2 : A \rightarrow B_2$ be embeddings, where $B_1 - e_1(A) = \{v\}$ and $B_2 - e_2(A) = \{u\}$. In the proof, we find a vertex v' in Δ such that $B_1 \cup \{v'\}$ is the amalgam C of B_1 and B_2 , where $f_1 : B_1 \rightarrow C$ is the identity and $f_2 : B_2 \rightarrow C$ maps v to v' , and all other points x in B_2 to $e_2^{-1} e_1(x)$. We can also assume without loss of generality that e_1 is the inclusion map, i.e., A is an induced subgraph of B_1 .

In the first part of the proof we consider the case where $u, v \in \Delta$ are vertices that are not connected via an edge from \triangleleft to any of the vertices in B_1 or B_2 , respectively. Let L be the set of topmost vertices below v in the forest defined by the transitive reduction of the relation \triangleleft^+ in B_2 , and let L'

predicate 'even', and so does $b' := be_2^{-1}$. Hence b' is a vertex of Δ that is a sequence of even length, and there exists a unique vertex v' such that $v' \triangleleft b'$. Note that this vertex v' might be equal to u . The mapping f_2 described above clearly is an embedding.

Next, assume that there is a vertex $t \in B_2$ such that $t \triangleleft v$ holds in Δ . In this case t and also $t' = te_2^{-1}$ do not satisfy 'even'. We pick a direct successor of t' in Δ that is disjoint to all other vertices in B_1 below t' , which again clearly exists by the definition of Δ . We can again easily check that the corresponding mapping $f_2 : B_2 \rightarrow B_1 \cup \{v'\}$ is an embedding. \square

2.8 Homomorphisms and Cores

One of the fundamental concepts for constraint satisfaction with finite templates is the notion of a *core* of a relational structure. Finite structures have a core that is unique up to isomorphism, and the core has the same constraint satisfaction problem. This notion is important for the algebraic approach to model theory, because for cores all endomorphisms are automorphisms. Unfortunately we do not have such a strong concept for infinite relational structures. In this section we first recall the definition and simple facts for cores of finite structures. Then we discuss related issues for infinite structures.

The core of a finite structure. A substructure A of a relational structure B is called a *retract* of B iff there exists a homomorphism from B to A that fixes the elements in A . The retract is called *proper*, if $|A| < |B|$. A finite relational structure A is called a *core* iff every endomorphism of A is an automorphism of A . A core A is called a *(proper) core of B* if A is a (proper) retract of B . The following is well-known.

Proposition 2.23. *Every finite relational structure has a core, which is unique up to isomorphism.*

Proof. Any finite structure B has a core, since we can select the retract of B with the smallest cardinality. Let A_1 and A_2 be cores of B , and $f_1 : B \rightarrow A_1$ and $f_2 : B \rightarrow A_2$ be endomorphisms. Then $f_1 f_2$ restricted to A_1 is an endomorphism of A_1 , and since A_1 is a core $f_1(f_2(A_1))$ is isomorphic to A_1 . For finite structure this implies that f_1 restricted to A_2 is an surjective

homomorphism to A_1 . By the symmetric argument $f_2|_{A_1}$ is also a surjective homomorphism, and thus the finite structures A_1 and A_2 are isomorphic. \square

Therefore, we speak of *the core* of a finite relational structure A , and denote it by $\text{core}(A)$. The notion is important for constraint satisfaction, since the image of A under an endomorphism clearly has the same constraint satisfaction problem as A – this also holds for infinite structures. For finite structures, the converse is trivially true:

If $\text{CSP}(A) \subseteq \text{CSP}(B)$ then there is a homomorphism from A to B . (2.3)

Homomorphisms between templates. Statement (2.3) is in general not true for infinite structures. Consider for example the structure $\Gamma = (\mathbb{Q}^+; R)$, where R is a binary relation defined by $R(x, y) :\Leftrightarrow 2x \leq z$. Let $\Gamma' = (\mathbb{N}; R)$ be the substructure of Γ restricted to the natural numbers. There is no homomorphism from Γ to Γ' : Assume for contradiction that f is such a homomorphism. Suppose f maps the point 1 to the integer $p > 0$. Since f preserves R and we know that $2f(1/2) \leq f(1)$ and thus $f(1/2) \leq p/2$. Iterating this argument we finally find a point $1/2^{n_0}$ such that $f(1/2^{n_0}) \leq p/2^{n_0} < 1$, and thus f can not be a homomorphism from \mathbb{Q} to \mathbb{N} .

On the other hand, every finite induced substructure S of Γ homomorphically maps to Γ' . To see this, multiply all numbers u/v in S , where u and v are natural numbers and relatively prime, by the smallest common multiple of their denominators v . The resulting numbers are positive integers, and the multiplication preserves the relation R and hence defines a homomorphism from Γ to Γ' .

For ω -categorical structures, however, we can show that 2.3 holds. The proof contains an application of König's tree lemma that we will encounter several times in later proofs.

Proposition 2.24. *Let Γ be an ω -categorical structure. Then for every countable relational structure Γ' , $\text{CSP}(\Gamma') \subseteq \text{CSP}(\Gamma)$ if and only if there is a homomorphism from Γ' to Γ .*

Proof. Clearly, if there is a homomorphism from Γ' to Γ , then every structure that homomorphically maps to Γ' also maps to Γ . Conversely, suppose $\text{CSP}(\Gamma') \subseteq \text{CSP}(\Gamma)$. Let a_1, a_2, \dots be an enumeration of the elements in Γ' . Consider the following infinite but finitely branching tree. The nodes on level n in the tree are the equivalence classes of homomorphisms from

a_1, \dots, a_n to Γ_2 , where two homomorphisms are equivalent if $f_1 = gf_2$ for some $g \in \text{Aut}(\Gamma)$. Adjacency between nodes on consecutive levels is defined by restriction. By our assumption, for each finite substructure of Γ' there is a homomorphism to Γ , and thus the tree contains a node on each level. By the theorem of Ryll-Nardzewski, there are only finitely many nodes at each level. By König's lemma the tree contains an infinite path. This path defines a homomorphism from Γ' to Γ . \square

Note that we only used the ω -categoricity of Γ , and only used $\text{Age}(\Gamma') \subseteq \text{CSP}(\Gamma)$ in the proof. The same idea also works to prove that if $\text{Age}(\Gamma') \subseteq \text{wSub}(\Gamma)$ then Γ' is a substructure of Γ . Likewise, if $\text{Age}(\Gamma') \subseteq \text{Age}(\Gamma)$ then there is an embedding of Γ' in Γ (the last statement is an exercise in [Cameron, 1990]). The set of all structures of a given signature are partially ordered by the homomorphism relation. General facts about this *homomorphism order* for finite structures can be found e.g. in [Nešetřil and Tardif, 2000].

Core-like properties of infinite structures. Core-like properties for arbitrary infinite relational structures Γ were studied by Bauslaugh [Bauslaugh, 1996, Bauslaugh, 1995]:

- $I(\Gamma)$ holds if any endomorphism of Γ is an injection.
- $S(\Gamma)$ holds if any endomorphism of Γ is a surjection.
- $N(\Gamma)$ holds if any endomorphism of Γ preserves the complements of the relations of Γ – we also say that they *preserve non-relations*.
- $R(\Gamma)$ holds if Γ has no proper retractions.
- $i(\Gamma)$ holds if there is an endomorphism e such that $e(\Gamma)$ satisfies I .
- $s(\Gamma)$ holds if there is an endomorphism e such that $e(\Gamma)$ satisfies H .
- $n(\Gamma)$ holds if there is an endomorphism e such that $e(\Gamma)$ satisfies N .
- $r(\Gamma)$ holds if there is a retraction e of Γ that satisfies R .

We can canonically define combinations of the uppercase properties. For instance the property ISN of a finite structure A states that A is a core. For finite structures, ISN is equivalent to e.g. IN , IS , I , S and to R . For infinite structures, these mentioned properties are all inequivalent, with the single

exception that ISN is equivalent to SN [Bauslaugh, 1996]. For the combinations of the lowercase properties we additionally require that the image of the *same* endomorphism e of Γ has the required properties. Also here most of the properties and combinations are inequivalent, and the dependencies between these concepts were exhaustively examined in [Bauslaugh, 1996]. Finite structures A satisfy $isnr$. Infinite structures in general satisfy none of the properties i, s, n , or r .

Cores of ω -categorical structures. We now focus on one of the definitions of a core of an infinite structure that turned out to be useful if we are interested in constraint satisfaction and in ω -categorical structures. If we require that every endomorphism is an automorphism, this seems too restrictive for the application to constraint satisfaction. In this case, even the dense linear order would not be a core. Instead we say that Γ is a *core* if the image $f(\Gamma)$ of every endomorphism f is isomorphic to Γ . Note that for finite structures this definition and the classical definition are equivalent. If a core Γ' is the image of Γ under an endomorphism of Γ , we say that Γ' is a *core of* Γ .

Definition 2.25. *A structure Γ is called a core, if every endomorphism of Γ preserves non-relations and inequality; in other words, every endomorphism is an embedding. We say that a structure Γ' is a core of Γ , or that Γ has a core Γ' , if Γ' is a core and is induced by the image of an endomorphism of Γ .*

This definition of a core corresponds to the properties IN and in in the terminology of [Bauslaugh, 1996]. Although the definition is weaker than ISN , it suffices to prove some results on cores known for finite cores used in constraint satisfaction. We will see such applications in Section 4.7.

Examples. As examples, we now discuss the dense linear order of the rational numbers, the ω -categorical semilinear order Λ from Section 2.7, the Rado-graph, and the homogeneous triangle-free graph. The dense linear order clearly is a core: for a linear order, every edge-preserving map also preserves non-edges - and has to be injective. Next we consider the Rado-graph \mathbf{R} : It is up to isomorphism described by the following extension property: (EP) *For every two finite subsets P and Q of the graph \mathbf{R} there exists a vertex not contained in P or Q and adjacent to each vertex in P and to no vertex in Q .* By property (EP) of the Rado graph we can find in an infinite process an infinite complete subgraph K_ω , which is the up to isomorphism unique core of the Rado-graph.

Another example of a core is the universal graph for all triangle-free graphs Fraïssé-limit $\mathcal{A} = \text{Fl}(\text{Forb}(K_3))$. It is a core, since there are existential positive definitions of non-adjacency and inequality. If we denote the edge-relation of the graph by E , then the definition for non-adjacency of the nodes x and y is $\exists z. E(x, z) \wedge E(z, y)$ and the definition for inequality of the nodes x and y is $\exists u, v. E(x, u) \wedge E(u, v) \wedge E(v, y)$.

Finally we look at the semilinear order Λ from Section 2.7. It is important which signature we use: $(\Lambda; \triangleleft^+)$ is not a core, since we find an endomorphism to a countable dense linear order contained in Λ . It is easy to show that the structure $(\Lambda; \triangleleft^+, \perp)$ is a core.

Chapter 3

Constraint Satisfaction

In this chapter we formally introduce the class of constraint satisfaction problems studied in this thesis, and make some fundamental observations. In Section 3.2 we discuss concrete examples of constraint satisfaction problems with ω -categorical templates. In Section 3.3 we prove that every problem in *monotone monadic SNP without inequalities* – a notion introduced in the context of constraint satisfaction in [Feder and Vardi, 1999] – is a constraint satisfaction problem with an ω -categorical template, described by a finite set of finite forbidden substructures. On the other hand, all such constraint satisfaction problems are contained in *monotone SNP*. Finally, in Section 3.4, we describe the power of Datalog programs for constraint satisfaction.

3.1 Introduction

Let Γ be an arbitrary structure with relational signature τ . The constraint satisfaction problem for Γ is the following computational problem:

CSP(Γ)

INSTANCE: A finite structure S of the same relational signature τ as Γ .

QUESTION: Is there a homomorphism f from S to Γ ?

We denote by $\text{CSP}(\Gamma)$ the set of all finite τ -structures that homomorphically map to Γ , and call Γ the *template* of the constraint satisfaction problem. Let S be a structure that has a homomorphism f to Γ ; in this case we say that f is a *solution of S* for the constraint satisfaction problem for Γ .

Sometimes the elements of S are called nodes, for instance if the signature is binary and we are talking about digraphs, or more general if we want to stress that the instance can be viewed as a *constraint network*, a term often used in the literature on constraint satisfaction in artificial intelligence.

The class of all constraint satisfaction problems with a template T over a finite domain we denote by CSP. This class is clearly contained in NP, since for a given instance S we can guess a mapping $f : S \rightarrow T$ and verify in polynomial time that it is a homomorphism. For many infinite templates Γ the problem CSP(Γ) is also contained in NP: for instance if the template is specified by finitely many finite forbidden induced substructures.

Definition 3.1. *A relational τ -structure Γ is called finitely constrained, if there is a first-order expansion Γ' of Γ , such that the structure Γ' with the expanded signature τ' satisfies $\text{Age}(\Gamma') = \text{Forb}(\mathcal{N})$ for some finite set \mathcal{N} of finite τ' -structures.*

The term 'finitely constrained' is used here in the same sense as in [Latka, 1994]. Note that we did not require that there is a set \mathcal{N} of finite τ -structures such that $\text{Age}(\Gamma) = \text{Forb}(\mathcal{N})$. We only require this for a structure Γ' with an expanded signature τ' .

Proposition 3.2. *Let Γ be a finitely constrained relational structure. Then CSP(Γ) is in NP.*

Proof. Let Γ' be an expansion of Γ with signature τ' , and \mathcal{N} a finite set of τ' -structures such that $\text{Age}(\Gamma') = \text{Forb}(\mathcal{N})$ – such an \mathcal{N} exists because Γ is finitely constrained. Suppose we are given an instance S of CSP(Γ) of size n . We first guess a structure T' of size at most n in the expanded signature; let T be the reduct of T' that has the original signature. Then we can verify in polynomial time whether there is a *strong* homomorphism from S to T , and whether T' does not contain any substructure from \mathcal{N} . \square

Let CSP* be the class of constraint satisfaction problems with finitely constrained ω -categorical templates. Without the assumption that the template is finitely constrained, we find ω -categorical templates Γ such that CSP(Γ) is undecidable; we already find countable homogeneous digraphs with an undecidable constraint satisfaction problem: Recall from Section 2.6 that there are uncountably many homogeneous digraphs Γ with an age that has the free amalgamation property, and all of them have different constraint satisfaction problems – we discuss this at the end of Subsection 3.2.1. Since there is a countable number of algorithms, undecidable constraint satisfaction problems exist.

Constraint satisfaction with homogeneous digraphs. If we assume that the class of finite induced substructures of a countable homogeneous digraph Γ is described by finitely many forbidden induced subgraphs, we can determine the complexity of $\text{CSP}(\Gamma)$. In the case where we have free amalgamation we only have to check whether the input does not contain any of the forbidden substructures – this can be done in polynomial time. In the other cases we can use the classification result in [Cherlin, 1998]. This is discussed in Section 3.2.

We discuss this classification in Section 3.2 to provide more examples of ω -categorical structures. Another reason why we present this catalog is that – in contrast to the class of ω -categorical structures – there is some hope that the class of homogeneous ω -categorical structures can be classified in the spirit of [Cherlin, 1998]. Some systematic results in this direction were made for so-called *stable* homogeneous structures (see e.g. [Cherlin and Lachlan, 1986, Lachlan, 1996]). Also recall that every ω -categorical template can be made homogeneous by expanding the signature (Proposition 2.14 on page 30; however, this might affect the computational complexity of the constraint satisfaction problem).

Sometimes the age of a homogeneous structure satisfies a stronger version of the amalgamation property - the so-called *strong* amalgamation property, see Section 2.5. This property affects the nature of its constraint satisfaction problem. If the template of a constraint satisfaction has strong amalgamation, every structure homomorphically mapping to Γ is also a substructure of Γ .

Substructure problems. For a fixed infinite relational structure Γ there is another computational question, which turns out to stand in an interesting relationship to the constraint satisfaction problem. Again the instances are finite structures of the same relational signature as Γ , but we ask whether the structure is a substructure of Γ (we mean a *weak* substructure, i.e, a not necessarily an induced substructure). We call this computational problem the *substructure problem* of Γ , and write $\text{wSub}(\Gamma)$ for the set of all finite substructures of Γ . Analogously to Proposition 3.2, if Γ is a finitely constrained structure, then $\text{wSub}(\Gamma)$ is in NP.

Note that this problem is trivial for finite Γ . However, in the *uniform* setting, i.e., if an instance of the problem contains two finite structures A and B , and we ask whether A is a substructure of B , it was studied before, and is also called the *substructure isomorphism problem*; see e.g. [Gupta and

Nishimura, 1996]. Here, however, we will only study the problem in its non-uniform version and for infinite Γ .

If Γ has the strong amalgamation property, then the constraint satisfaction problem of Γ and the substructure problem of Γ are the same, i.e., $\text{CSP}(\Gamma) = \text{wSub}(\Gamma)$. In Chapter ?? we see an example of an infinite structure Γ not having strong amalgamation, where also the computational complexity of the substructure for Γ and the constraint satisfaction problem for Γ is different (unless $\text{P}=\text{NP}$).

Both constraint satisfaction and substructure problems with finitely constrained ω -categorical template generalize the well-known class CSP of constraint satisfaction problems with *finite* templates. For every finite structure T we can find a finitely constrained ω -categorical structure Γ such that $\text{wSub}(\Gamma) = \text{CSP}(\Gamma) = \text{CSP}(T)$. We replace every vertex u in T by infinitely many vertices that have the same neighborhood¹. Generalizing the terminology of Section 2.6 for graphs, we write $\Gamma = T[I_\infty]$. We also add a unary predicate for each vertex $u \in T$ that holds on precisely these copies of u . If we do this for all vertices u in T , it is easy to check that the age of the resulting structure Γ' has the strong amalgamation property. It is also finitely constrained, since $\text{Age}(\Gamma') = \text{Forb}(\mathcal{N})$ where \mathcal{N} is the set of all relational structures of the same signature as Γ' of size $|T| + 1$ that are not in $\text{Age}(\Gamma)$. Thus, substructure problems and constraint satisfaction with finitely constrained ω -categorical templates generalize constraint satisfaction with finite templates. This also follows from Proposition 3.9 on page 68.

Constructing solutions. Besides the question whether an instance of a constraint satisfaction problem *has* a solution, we could ask to *construct* this solution. For finite Γ it was observed in [Bulatov et al., 2000, Kolaitis and Vardi, 1998] that if Γ contains all singleton relations, then these two problems are computationally equivalent. In fact, the assumption that all the singletons are in the signature is not necessary.

We now describe an algorithm that constructs a solution of a given instance S of $\text{CSP}(T)$ under the assumption that the decision problem $\text{CSP}(T)$ is tractable. For a relation symbol R in the signature of Γ we add a tuple to the relation R in the instance, and check whether the resulting structure S' is in $\text{CSP}(\Gamma)$. If yes, we proceed with S' and do this for all relation symbols from Γ and all tuples in S . Finally we end with some structure S'' with the

¹We say that two vertices a, b of a relational structure Γ *have the same neighborhood* in Γ if there is a nontrivial automorphism of Γ that fixes all points but a and b .

property that every solution of S'' is a *strong* homomorphism from S'' to Γ . Such a solution can be found in polynomial time: Clearly we can map nodes in S'' that have the same neighborhood to the same vertex, and thus we identify vertices in S'' that have the same neighborhood. Finally we only have to find the resulting relational structure as a substructure of T , which can be done in constant time, since T is of constant size. This gives us a solution of S for $\text{CSP}(T)$.

For infinite templates it is in general not clear whether and how a solution of a constraint satisfaction problem can be represented. If the structure is ω -categorical, first-order logic provides an elegant way to represent solutions of a constraint satisfaction problem *up to isomorphism of solutions*. Two solutions $f, f' : S \rightarrow \Gamma$ of an instance S of $\text{CSP}(\Gamma)$ are called *equivalent* iff there exists an automorphism α of Γ such that $f = f'\alpha$. Let a_1, \dots, a_n be the nodes of S . For ω -categorical Γ , by Theorem 2.6, there are only finitely many inequivalent solutions for each instance S of $\text{CSP}(\Gamma)$. Moreover, there exists a first-order formula ϕ in n free variables, that holds on $x_1, \dots, x_n \in \Gamma$ if and only if $f : a_i \mapsto x_i, 1 \leq i \leq n$, is a solution of S for $\text{CSP}(\Gamma)$.

Primitive positive definitions. Sometimes it is convenient to use logic language to talk about constraint satisfaction - we already mentioned this briefly in the introduction. In this case we call the instance a *constraint*, which is a conjunction of several atomic constraints, which are positive literals with relation symbols taken from the signature of the template. The elements from the domain of an instance of a constraint satisfaction problem are then called *variables*, and the instance can be viewed as a primitive positive sentence that is interpreted over the template.

For both finite and infinite Γ , the following simple lemma explains the relevance of p.p.-definable relations in constraint satisfaction. Suppose we extend a relational structure Γ by a p.p.-definable relation R . This does not change the computational complexity of the corresponding constraint satisfaction problem, since we can replace every occurrence of R in an instance of $\text{CSP}(\Gamma)$ by the τ -structure that defines R .

Lemma 3.3. *Let Γ be a τ -structure and let Γ' be the extension of this structure by a relation R that is p.p.-definable over Γ . Then $\text{CSP}(\Gamma)$ is polynomial-time equivalent to $\text{CSP}(\Gamma')$.*

In Chapter 4 we characterize p.p.-definability algebraically.

Infinite templates that are not ω -categorical. Constraint satisfaction with arbitrary templates allows to describe precisely those problems that are closed under disjoint unions, and whose complement is closed under homomorphisms. Clearly, these two properties are necessary, since the disjoint union of two satisfiable instances is again satisfiable, and since a homomorphic image of an unsatisfiable instance is again unsatisfiable. On the other hand, every problem satisfying the two mentioned properties can be expressed as a constraint satisfaction problem with an infinite template, since we can use the infinite disjoint sum of all satisfiable instances as a template. We call this template Γ , and call the summands of which Γ is formed the *components* of Γ . Clearly, for every yes-instance of the problem we can find a homomorphism to Γ . Now suppose for contradiction that a no-instance homomorphically maps to Γ . Let S be a minimal such instance. The homomorphic image I of S is by assumption again a no-instance, and therefore it can not be isomorphic to one of the components of Γ . Hence, I is divided into several substructures I_l that lie in different components of Γ . Each such substructure I_l has less vertices than S and hence is a yes-instance. By assumption, the disjoint union I of the substructures I_l is also a yes-instance, a contradiction.

There are concrete constraint satisfaction problems that can not be formulated with an ω -categorical template, e.g., $\text{CSP}((\mathbb{N}; +, *, 0, 1))$, where '+' stands for the ternary relation defined by $x + y = z$, '*' for $x * y = z$, and '0' and '1' are unary singleton relations containing only 0 and 1, respectively. This problem is equivalent to Hilbert's 10th problem on Diophantine equations (which is undecidable [Matijasevich, 1993]).

To deal with such problems in a unified framework we have to deal with the problem how such infinite templates are represented. Finite axiomatizations of the first-order theory of a structure do not seem appropriate to represent templates of constraint satisfaction problems: For example, only few ω -categorical structures have finite axiomatizations – such structures were studied in [Lippel, 2001]. In this thesis we instead describe infinite templates by finite sets \mathcal{N} of forbidden induced finite substructures; in the case that $\text{Forb}(\mathcal{N})$ is an amalgamation class, this describes the template uniquely, up to isomorphism. In most cases, the corresponding relational structures are not finitely axiomatizable (but ω -categorical, see Section 2.4).

Infinite signatures. Following [Bulatov et al., 2000], we call the constraint satisfaction problem for a relational structure Γ with a countably *infinite* signature *tractable*, if it is tractable for the reduct of Γ to any finite signa-

ture $\tau' \subset \tau$. With this definition and Lemma 3.3 it follows that $\text{CSP}(\Gamma)$ is tractable if and only if $\text{CSP}(\langle \Gamma \rangle_{pp})$ is tractable.

Note that there is a subtle difference to the following stronger version of tractability, called *global tractability* in [Bulatov et al., 2000], where we require that there is a polynomial time algorithm for the constraint satisfaction problem for the entire infinite signature (given an appropriate representation). It is open whether there is a constraint satisfaction problem with finite template that is tractable but not globally tractable [Bulatov et al., 2000]. We do not know of such an example even for ω -categorical templates. In particular we can not exclude that there is a structure Γ such that $\text{CSP}(\Gamma)$ is tractable, but $\text{CSP}(\langle \Gamma \rangle_{pp})$ is not globally tractable.

3.2 The Complexity of some CSPs

In this section we look at the constraint satisfaction problems and their computational complexity for various ω -categorical structures that we encountered in earlier sections. Some of them were independently studied in the literature.

Whereas for finite templates all known hard constraint satisfaction problems are hard because there is a primitive positive definition of a relation encoding 1-in-3-SAT, for infinite templates the problem *Betweenness* is often used to prove hardness. The following is taken from [Garey and Johnson, 1978]:

BETWEENNESS

INSTANCE: A finite set V , and a collection C of ordered triples (x, y, z) of distinct elements from V .

QUESTION: Is there a one-to-one function $f : V \rightarrow \{1, \dots, |V|\}$ such that, for each $(a, b, c) \in C$, we have either $f(a) < f(b) < f(c)$ or $f(a) < f(c) < f(b)$.

The NP-hardness of Betweenness was first proven by Opatrný in 1978. Furthermore the problem is MAX SNP complete [Garey and Johnson, 1978]. In [Chor and Sudan, 1998] semidefinite programming is applied to a translation of the problem into a set of quadratic inequalities. If there exists a solution, this procedure finds a solution satisfying half of the triples.

Betweenness is a constraint satisfaction problem with an ω -categorical template: Since interpretations preserve ω -categoricity (Proposition 2.7) and \mathbb{Q} is ω -categorical, the structure B below is also ω -categorical. Clearly, $\text{CSP}(B)$ is the computational problem Betweenness.

$$B = (\mathbb{Q}, \{(x, y, z) \subseteq \mathbb{Q}^3 \mid x < y < z \text{ or } z < y < x\})$$

3.2.1 The CSPs for the Homogeneous Digraphs

In Section 2.6 we described the classification of the countable homogeneous digraphs. In this paragraph we discuss their constraint satisfaction problems.

The constraint satisfaction problem of the dense linear order on the rational number \mathbb{Q} corresponds to digraph acyclicity and is tractable, via breadth-first search. The wreathed structures $\mathbb{Q}[I_n]$ and $I_n[\mathbb{Q}]$ and the partial order \mathbb{P} have the same constraint satisfaction problem. The graph $K_n[I_\infty]$ is universal for the class of all n -colorable directed graphs. Thus the constraint satisfaction problem is NP-hard. Since T^∞ is universal for the set of all tournaments, $\text{CSP}(T^\infty)$ is the set of all digraphs and therefore trivial. The same holds for \hat{T}^∞ , $T^\infty[I_n]$, $I_n[T^\infty]$ and for $\infty * I_\infty$.

The CSPs for the exceptional class. A more interesting example is the constraint satisfaction problem for $S(2)$. We show that $\text{CSP}(S(2))$ is NP-hard. Recall the definitions of $S(2)$ in Section 2.6. If we fix a vertex u in $S(2)$, the structures induced by the point sets $\{x \mid u < x\}$ and $\{x \mid u > x\}$ in the tournament $S(2)$ are isomorphic to $(\mathbb{Q}, <)$. The idea for the hardness proof is that the Betweenness-relation B has on these sets the following primitive positive definition.

$$\exists v, w : v > x \wedge v > y \wedge v < z \wedge w < x \wedge w > y \wedge w > z . \quad (3.1)$$

A proof for that can be found in the following proposition. Also see Figure 3.1.

Proposition 3.4. *$\text{CSP}(S(2))$ is NP-complete.*

Proof. Proposition 3.2 shows that $\text{CSP}(S(2))$ is contained in NP, because $S(2)$ is finitely constrained – see Section 2.6. To prove hardness we reduce

the problem Betweenness to $\text{CSP}(S(2))$. Let $(V; C)$ be an instance of Betweenness. We define a polynomial size instance S of $\text{CSP}(S(2))$ which is satisfiable if and only if $(V; C)$ is a yes-instance of Betweenness. The vertices of S consist of the vertices V and some additional vertices. We first introduce a new vertex u , and add $u < x$ to S for all $x \in V$. Then introduce for each triple x, y, z from C two new vertices v, w and add the constraints $v > x, v > y, v < z, w < x, w > y, w > z$ to the instance S of $\text{CSP}(S(2))$. Also see Figure 3.1.

If there is a solution to the Betweenness instance $(V; C)$, there is also a homomorphism from S to $S(2)$: We map the vertex $u \in S$ to *some* vertex $f(u)$ in $S(2)$. Then the linearly ordered set $\{w \mid f(u) < w\}$ is isomorphic to the linear order of the rational numbers, and we map the vertices in V to this set in the same way as the solution of the Betweenness instance maps them to the rational line. Finally we can map the existentially quantified variables to $S(2)$, in either of the two ways displayed in Figure 3.1: this is, if $x < y < z$ in the solution to the Betweenness instance we let $x < v, v < y, v < z, z < w, y < w, w < x$. Otherwise, if $z < y < x$ we let $z < w, y < w, w < x, y < v, x < v, v < z$.

Conversely, if there is a homomorphism f from S to $S(2)$, we also have a solution to the Betweenness instance $(V; C)$. The homomorphism f maps all vertices in V except u to the linearly ordered set $\{w \mid f(u) < w\}$. We claim that in the corresponding linear order of the vertices V , for each triple in C either $x < y < z$ or $z < y < x$. Assume otherwise that $y < x$ and $y < z$. Since x and y have to be comparable, either $x > z$ or $x < z$. In the first case we find the oriented three-cycle x, v, z, x , in the second the cycle x, z, w, x . In both cases there are arcs from y towards the vertices of that cycle, and we found the subgraph $[I_1, C_3]$, which is forbidden in $P(3)$. The case where $y > x$ and $y > z$ is analogous with the forbidden constellation $[C_3, I_1]$. Thus we have a contradiction with the assumption that f is a homomorphism from S to $P(3)$. \square

Since there is a homomorphism from $S(2)$ to $\hat{\mathbb{Q}}$ and from $\hat{\mathbb{Q}}$ to $S(2)$, these problems have the same constraint satisfaction problem. For the hardness of $\text{CSP}(S(3))$ we can use the same gadget as above to simulate Betweenness on a definable subset of vertices. Similarly to Proposition 3.4 we can prove the NP-hardness of $\text{CSP}(P(3))$. Before we present a hardness proof, we reformulate the computational problem $\text{CSP}(P(3))$; in this form we call the problem *switching-trigraph-transitivity*.

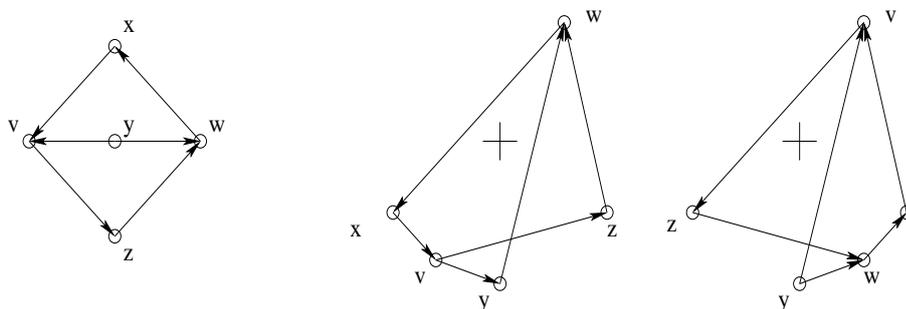


Figure 3.1: On the left we find the gadget to simulate the betweenness relation on x, y, z , where an arc from a to b means that there is the constraint $a < b$. On the right we find two possible ways to map these vertices homomorphically to $S(2)$, where the cross marks the origin in the plane.

SWITCHING-TRIGRAPH-TRANSITIVITY

INSTANCE: A digraph $D = (V; E)$.

QUESTION: Can we partition the vertices V into three parts P_1, P_2, P_3 , such that the graph that arises from D by the following three operations is transitive?

- i) deleting the edges from P_1 to P_2 , P_2 to P_3 , and P_3 to P_1 ;
- ii) reversing the arcs from P_2 to P_1 , P_3 to P_2 , and P_1 to P_3 ;
- iii) adding arcs between unconnected pairs from P_2 to P_1 , P_3 to P_2 , and P_1 to P_3 .

Proposition 3.5. *$CSP(P(3))$, alias Switching-trigraph-transitivity, is NP-complete.*

Proof. As in the proof of Proposition 3.4 we use Proposition 3.2 to observe $CSP(P(3))$ is contained in NP, because $P(3)$ is finitely constrained – see Section 2.6. Again we prove hardness by reducing Betweenness to $CSP(P(3))$, and we construct an instance S of $CSP(P(3))$ from an instance $(V; C)$ of Betweenness in the same way as in the proof of Proposition 3.4. Now, we additionally introduce new vertices a, b for all pairs of vertices $x, y \in V$, and impose the constraint $a < b, x > b, b > y, y > a, a > x$; see Figure 3.2.

First we show that given a solution to the Betweenness instance $(V; C)$, we can construct a homomorphism from S to $P(3)$. Note that $P(3)$ contains copies of $S(2)$ as subgraphs; fix such a copy of $S(2)$, which we will sloppily also denote by $S(2)$. We map the vertex $u \in S$ to *some* vertex $f(u)$ in $S(2)$. The subset $\{x \mid f(u) < x\}$ of $S(2)$ induces the linear order of the rational

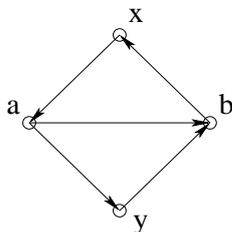


Figure 3.2: We use this gadget to ensure that the vertices x and y are mapped to comparable vertices in $P(3)$.

numbers. To construct a solution for S , we map the vertices in V to this set in the same way as the solution of the Betweenness instance maps them to the rational line. As in the proof of Proposition 3.4, we can map the remaining existentially quantified variables to $P(3)$ in either of the two ways displayed in Figure 3.1.

Conversely, if there is a homomorphism f from the constructed instance S to $P(3)$, we also have a solution to the Betweenness instance $(V; C)$. Every solution to S maps pairs of distinct vertices $x, y \in V$ to *comparable* vertices in $P(3)$: Consider the corresponding subgraph x, y, a, b in S . Since this structure is a forbidden induced subgraph of $P(3)$ (see Figure 2.3 on page 37), any homomorphism of this structure to $P(3)$ has to map x and y to comparable vertices in $P(3)$. Hence, the set $f(V)$ induces in $P(3)$ a tournament. Because of the vertex $f(u) < x$ for all x in $f(V)$, and because $[I_1, C_3]$ is a forbidden induced subgraph in $P(3)$, the set $f(V)$ has to be isomorphic to the linear order of the rational numbers. We claim that in the corresponding linear order of the vertices V , for each triple in C either $x < y < z$ or $z < y < x$. This can be shown exactly as in the proof of Proposition 3.4, since the vertices x, y, z have to be pairwise comparable in every solution of S . \square

Templates with free amalgamation. Finally we discuss the homogeneous digraphs whose age has the free amalgamation property. For a set of tournaments \mathcal{T} , the \mathcal{T} -generic structure is universal for the class $\mathcal{A}(\mathcal{T})$ introduced in Section 2.5. Every different substructure-closed set of tournaments \mathcal{T} yields a different \mathcal{T} -generic structure. There is an uncountable number of such sets of tournaments [Henson, 1972]. Each of these has a different constraint satisfaction problem, because $\text{CSP}(\Gamma) = \text{Age}(\Gamma)$ for digraphs whose age has the free amalgamation property – see Section 2.5. Since there is

a countable number of algorithms, undecidable constraint satisfaction problems exist, with templates that are countable homogeneous digraphs.

Proposition 3.6. *There are undecidable constraint satisfaction problems $CSP(\Gamma)$, even if Γ is a homogeneous digraph.*

Nevertheless, if the age is described by a finite set \mathcal{N} of finite forbidden induced subgraphs, then the constraint satisfaction problem for these templates is simple, since $CSP(\Gamma) = \text{Age}(\Gamma) = \text{Forb}(\mathcal{N})$. Hence it suffices to check whether the input contains a forbidden induced subgraph, which can be done in polynomial time. If not, the input graph is in the age of the template and is a yes-instance.

3.2.2 Tree Descriptions

The following constraint satisfaction problem is a special case of a problem studied in [Cornell, 1994]. The algorithm presented there is based on local consistency (or *constraint propagation*), and turned out to be incomplete. In Section 3.4 we explain this approach and give an example that shows that the algorithm in general fails to solve the problem. However, the problem is tractable. We present a polynomial time algorithm solving a more general problem in Chapter 6 on page 115.

PURE-DOMINANCE-CONSTRAINT

INSTANCE: A digraph $(V; D)$ where each edge is labeled by a subset of the symbols from $\{\triangleleft^+, \triangleright^+, \equiv, \perp\}$.

QUESTION: Is there a mapping f from V to some rooted tree T such that for each edge at least one of the labels is *satisfied*: \triangleleft^+ (\triangleright^+) is satisfied for uv if there is a directed path from u to v (from v to u) in T . The label \equiv is satisfied if $u = v$. The label \perp is satisfied if none of the other labels is satisfied.

Using the ω -categorical semilinear order $\Lambda = (D, <)$ from Section 2.7 we can formulate this problem (and related problems, e.g. the problem Centaurenealogy from the introduction) as a constraint satisfaction problem: the signature of the corresponding template are all binary first-order definable relations over Λ . This fact is easy to see: the relations play the rôle of the various labelings in the above computational problem.

It is interesting to note that the constraint satisfaction problem for other relations that are first-order definable over the semilinear order Λ can be NP-hard. The following problem called *Forbidden-triples* was studied in phylogenetic analysis [Bryant, 1997]; also see [Ng et al., 2000].

FORBIDDEN-TRIPLES

INSTANCE: A set of ordered triples (x, y, z) over a set L .

QUESTION: Is there a rooted binary tree T with leaf set L such that for every ordered triple (x, y, z) every rooted subtree containing y and z also contains x ?

To formulate this as a constraint satisfaction problem, we again use Λ (see Section 2.7), but this time the template contains the ternary relation $\neg(x:yz)$ in the signature, where $x:yz$ is defined by the first-order formula $\exists u. u > y \wedge u > z \wedge u \perp x$ over Λ .

Another problem from phylogenetic analysis is the problem Quartet-compatibility, which we already mentioned in the introduction. It is again an NP-hard problem [Steel, 1992], by reduction of Betweenness. To formulate this as a constraint satisfaction problem we use the 4-ary relation that arises from the countable homogeneous Boron tree introduced in Section 2.7.

3.2.3 The Fragments of Allen's Interval Algebra

Allen's Interval Algebra was first introduced and studied in artificial intelligence [Allen, 1983]. The maximal tractable fragments were investigated in a series of papers, in particular we want to mention [Bürckert and Nebel, 1995, Jeavons et al., 2003]. As we will see now, all fragments correspond to constraint satisfaction problems with ω -categorical templates.

Consider as a base set D the closed intervals on the rational numbers, and the binary relations on these intervals defined in Figure 3.3. For any set of relations that are unions of p, o, d, s, m, f , and \equiv , the corresponding countable relational structure is ω -categorical. In fact, the listed basic relations are nothing but the complete 2-types of \mathbb{Q}^2 , and every first-order definable binary relation over $(\mathbb{Q}, <)$ is a union of such complete 2-types (see Chapter 2). Allen's algebra then is the constraint satisfaction problem for \mathbb{Q}^2 where the signature consists of all first-order definable binary relations over \mathbb{Q} . The *fragments* of this algebra are the constraint satisfaction problems of

Relation	In symbols	Definition
The interval x <i>precedes</i> y	$x \text{ p } y$	$x^+ < y^-$
The interval x <i>overlaps</i> y	$x \text{ o } y$	$x^- < y^- < x^+$ and $x^+ < y^+$
The interval x is <i>during</i> y	$x \text{ d } y$	$y^- < x^-$ and $x^+ < y^+$
The interval x <i>starts</i> y	$x \text{ s } y$	$x^- = y^-$ and $x^+ > y^-$
The interval x <i>finishes</i> y	$x \text{ f } y$	$x^+ = y^+$ and $x^- > y^-$
The interval x <i>meets</i> y	$x \text{ m } y$	$x^+ = y^-$.
The interval x <i>equals</i> y	$x \equiv y$	$x^- = y^-$ and $x^+ = y^+$

Figure 3.3: The basic relations of Allen’s interval algebra. The variables $x = (x^-, x^+)$ and $y = (y^-, y^+)$ denote closed intervals.

the various (ω -categorical) reducts of this structure.

The complexity of these fragments have a dichotomy [Jeavons et al., 2003, Bürckert and Nebel, 1995]. We conjecture that this result can be generalized to arbitrary structures that have an interpretation in $(\mathbb{Q}, <)$: Every constraint satisfaction problem with such a template is either NP-hard or tractable. See Section 7.3.

3.3 Monotone SNP

The class SNP is an important subclass of NP [Kolaitis and Vardi, 1992], and was originally introduced in the context of the theory of optimization problems [Papadimitriou and Yannakakis, 1991, Mayr et al., 1998] – but turned out to be closely related to homomorphism problems [Feder and Vardi, 1999]. An SNP formula Φ is an existential second-order formula with a universal first-order part. The first order part might contain the existentially quantified relation symbols and additional relation symbols from a given signature τ (these relations will sometimes be called the *input* relations). We shall assume that the first-order part is written in conjunctive normal form, and each disjunction is written as a conjunction of positive and negative literals. As in [Hodges, 1997], we say that such formulae Φ are in *negation normal form*. As usual, an SNP sentence is an SNP formula without free variables. The class of SNP sentences corresponds to a class of computational problems, which we will also call SNP. For a fixed SNP sentence Φ of signature τ the computational task is to determine for a given structure S of signature τ whether S satisfies Φ .

Note that logic plays a different rôle here compared to Section 1.2. There we considered the instance as a (primitive positive) sentence, and asked whether the fixed *template* is a model of that formula. Now we ask whether the *instance* is a model of our fixed SNP formula.

Clearly, SNP is contained in NP. It also contains NP-hard problems, for instance graph 3-colorability. Given a graph $G = (V; E)$, 3-colorability of G can be expressed for example by the following SNP sentence:

$$\begin{aligned} \exists E' \forall x, y, u, v : & \neg(E(x, y) \wedge \neg E'(x, y)) \\ & \wedge \neg(E'(x, y) \wedge E'(x, u) \wedge E'(x, v) \wedge E'(y, u) \wedge E'(y, v) \wedge E'(u, v)) \\ & \wedge \neg(\neg E'(x, u) \wedge \neg E'(x, v) \wedge E'(u, v)) \end{aligned}$$

This sentence says that we can find a superset E' of E that does not contain a K_4 , and where for every edge uv all vertices are connected to u or to v . There might be many ways how to define the same problem with different SNP formulae. The following formula is again satisfied by exactly the three-colorable graphs, but this time the SNP sentence is *monadic*, i.e. all existentially quantified relations are unary.

$$\begin{aligned} \exists R, B, G \forall x, y : & \neg(R(x) \wedge B(x)) \wedge \neg(R(x) \wedge G(x)) \wedge \neg(B(x) \wedge G(x)) \\ & \wedge \neg(E(x, y) \wedge R(x) \wedge R(y)) \\ & \wedge \neg(E(x, y) \wedge B(x) \wedge B(y)) \\ & \wedge \neg(E(x, y) \wedge G(x) \wedge G(y)) \\ & \wedge \neg(\neg R(x) \wedge \neg B(x) \wedge \neg G(x)) \end{aligned}$$

The class of SNP is important for constraint satisfaction [Feder and Vardi, 1999]. Unlike CSP, the class SNP contains problems that are not *monotone*, i.e., an unsatisfiable instance could become satisfiable if we add tuples to the relations of the instance, or identify nodes from the instance. Equivalently, we say that a problem is monotone, iff the class of unsatisfiable structures \mathcal{C} is *closed under homomorphisms*, i.e., if $A \in \mathcal{C}$ and there is a homomorphism from A to B , then $B \in \mathcal{C}$. Homomorphism problems and constraint satisfaction problems are clearly monotone.

Therefore the class *monotone SNP without inequalities (MSNP)* was introduced [Feder and Vardi, 1999]. For this class we require that there is a

defining SNP sentence Φ , written in negation normal form, where all input relations from τ occur within a negative number of negations, and Φ does not contain inequalities – we call such formulae *negative*. This is for instance the case for both of the formulae shown above. A recent result of [Feder and Vardi, 2003] says that for classes of finite structures closed under homomorphisms, MSNP is as expressive as SNP where we can also use negations and inequality.

The class MSNP still contains a polynomially equivalent problem for every problem in NP [Feder and Vardi, 1999], and hence is unlikely to exhibit a dichotomy. However, Feder and Vardi showed that every problem in the class of *monotone monadic SNP without inequality (MMSNP)* is under randomized Turing reductions equivalent to a constraint satisfaction problem with a finite template. Conversely, it is easy to see that MMSNP contains all constraint satisfaction problems with finite templates. Thus, CSP has a dichotomy if and only if MMSNP has a dichotomy.

Monotone SNP and constraint satisfaction. Recall the class CSP^* that contains all constraint satisfaction problems with a finitely constrained ω -categorical template. In the remainder of this section we will show that CSP^* contains all problems in MMSNP, and that CSP^* is contained in MSNP. Thus we have the following set of inclusions.

$$\text{CSP} \subset \text{MMSNP} \subset \text{CSP}^* \subset \text{MSNP} \subset \text{SNP} \subset \text{SO}\exists = \text{NP}$$

All of the inclusions ' \subset ' are strict. For the first we will show an example below. For the second, consider the problem $\text{CSP}((\mathbb{Q}, <))$, which is in CSP^* , but not in MMSNP: see Proposition 3.7 below. For the third inclusion $\text{CSP}^* \subset \text{MSNP}$ we present a problem in MSNP which is not closed under taking disjoint sums, and therefore can not be a constraint satisfaction problem for any template. For the last two strict inclusions, such examples are again easy to find.

Proposition 3.7. *$\text{CSP}((\mathbb{Q}, <))$ is not in MMSNP.*

Proof. Let Φ be an MMSNP sentence. We claim that if Φ is true on all directed paths, then it is also true on a directed cycle of a certain length n , and therefore it can not express acyclicity. We assume that Φ is in negation normal form. Let k be the number of existential monadic predicates in Φ ,

and let l be the length of a longest (not necessarily directed) path in one of the clauses with respect to the binary input relation. Now consider a directed path of length $(2^k)^l + l$. Since this path satisfies Φ , there are subsets of vertices of the path for each monadic predicate, avoiding the forbidden configurations in the clauses of Φ . These subsets can be considered as a coloring of the path with 2^k colors. By the pigeon hole principle, a consecutive subsequence w of the path of length $(2^k)^l + l$ must occur twice, say starting at position p_0 and p_1 . Now, if we color a directed cycle with the word induced by the positions from p_0 to p_1 , we can be sure that the colored cycle also avoids all forbidden configurations in the clauses of Φ . Hence, Φ will also be true on that cycle, and does not express acyclicity. \square

Now we consider an example of an MSNP sentence, whose finite models are not closed under disjoint sums, and which can therefore not be stated as a constraint satisfaction problem with an arbitrary finite or infinite template. The sentence does not contain any input relation, and is true on all structures with less than $R(3, 3) = 6$ elements, but false on all larger structures. This is because for every graph on at least 6 vertices either the graph or its complement contains a triangle. (Using similar Ramsey arguments, it is possible to construct more involved examples.)

$$\begin{aligned} \exists E \forall x, y, z : & \neg(\neg E(x, y) \wedge \neg E(x, z) \wedge \neg E(y, z)) \\ & \wedge \neg(E(x, y) \wedge E(y, z) \wedge E(z, x)) \end{aligned}$$

To prove the inclusions $\text{MMSNP} \subseteq \text{CSP}^* \subseteq \text{MSNP}$, we will make use of an important theorem from [Cherlin et al., 1999] (stated there for graphs only, but the proof does not make use of this assumption on the signature, and the theorem holds indeed for arbitrary relational signatures).

Theorem 3.8 (of [Cherlin et al., 1999]). *Let \mathcal{N} be a finite set of finite structures that is closed under homomorphisms. Then there exists an ω -categorical structure Γ such that $\text{Age}(\Gamma) = \text{wForb}(\mathcal{N})$.*

In fact, the structure Γ from Theorem 3.8 is *universal* for the set of all at most countable \mathcal{N} -free structures, i.e., every at most countable structure that does not contain a subgraph from \mathcal{N} is an induced substructure of Γ . Moreover, the ω -categorical structures that are constructed in Theorem 3.8 are model-complete; see [Cherlin et al., 1999].

Before we prove the two inclusions $\text{MMSNP} \subseteq \text{CSP}^* \subseteq \text{MSNP}$, it will be instructive to look at an example called *No-mono-tri* – a problem in monotone monadic SNP, which can not be expressed as a constraint satisfaction problem with finite templates. This was observed in [Madelaine and Stewart, 1999].

NO-MONO-TRI

INSTANCE: Graph $G = (V; E)$.

QUESTION: Is there a 2-coloring of the vertices so that the vertices of every triangle in the graph are not monochromatically colored?

This problem is contained in the class of problems called *G-free colorability* problems. Let G be a graph. A k -coloring of a graph H is said to be *G-free* iff each color class of the coloring induces a subgraph that does not contain G as a subgraph. All such coloring problems are clearly contained in *monotone monadic SNP with inequality* ($\text{MMSNP}(\neq)$). *G-free colorability* is NP-hard if G has more than two vertices and $k \geq 2$ [Achlioptas, 1997]. Thus No-mono-tri is NP-hard.

A graph is a yes-instance of No-mono-tri if and only if it satisfies the following MMSNP sentence

$$\begin{aligned} \exists C \quad \forall x, y, z. \quad & \neg(E(x, y) \wedge E(y, z) \wedge E(z, x) \wedge C(x) \wedge C(y) \wedge C(z)) \\ & \wedge \neg(E(x, y) \wedge E(y, z) \wedge E(z, x) \wedge \neg C(x) \wedge \neg C(y) \wedge \neg C(z)) \end{aligned}$$

This problem can be formulated as $\text{CSP}(K_2[\not\triangle])$, where $\not\triangle$ is the countable homogeneous triangle-free graph, i.e., $\not\triangle = \text{Fl}(\text{Forb}(K_3))$, see Section 2.6. Since $K_2[\not\triangle]$ has an interpretation in $\not\triangle$ it is ω -categorical by Proposition 2.7.

Proposition 3.9. *Every problem in MMSNP is in CSP^* .*

Proof. For a given problem in MMSNP we have to find a finitely constrained ω -categorical structure Γ , such that the problem equals $\text{CSP}(\Gamma)$. Let Φ be an MMSNP sentence, and let P_1, \dots, P_k be the existential monadic predicates in Φ .

Again we assume that the first-order part of Φ is in negation normal form, i.e., in conjunctive normal form where each clause is written as a negated conjunction of literals. By definition of MMSNP, all such literals with input relations are positive. For each existential monadic relation P_i we introduce

a relation symbol P'_i , and replace negative literals $\neg P_i(x)$ in Φ by the corresponding new monadic predicates $P'_i(x)$. After this replacement, the set of clauses in Φ corresponds to a set of relational τ -structures \mathcal{N} where the vertices in a structure correspond to variables in a clause, and the signature τ contains the input relations, the existential monadic relations P_i , and the symbols P'_i for the negative occurrences of the existential relations. We also add to \mathcal{N} those τ -structures that arise from clauses by identification of variables.

Then the class of structures $\text{wForb}(\mathcal{N})$ is, as Feder and Vardi call it, *closed under inverse homomorphisms*. A class \mathcal{C} is closed under inverse homomorphisms if $B \in \mathcal{C}$ implies that $A \in \mathcal{C}$, whenever there is a homomorphism from A to B . Cherlin, Shelah and Shi directly call \mathcal{N} *closed under homomorphisms*, which means in this case that for any $S \in \mathcal{N}$ and any homomorphism $h : S \rightarrow S'$, that S' contains a substructure in \mathcal{N} . Theorem 3.8 states that in this case there exists an ω -categorical structure Γ , which is universal for $\text{wForb}(\mathcal{N})$.

We use Γ to define the template for the constraint satisfaction problem. To do this, restrict the domain of Γ to those points that have the property that either P_i or P'_i holds (but not both P_i and P'_i) for all existential monadic predicates P_i . The resulting structure is by universality nonempty. Then we take the reduct Γ' of the structure that only contains the input relations. This structure is universal for the set of all structures in $\text{wForb}(\mathcal{N})$ restricted to the input signature. By Proposition 2.7 on page 26 the restricted and reduced structure Γ' is still ω -categorical. Moreover, Γ' is finitely constrained: If we add back all the relations P_i and P'_i , for all $1 \leq i \leq k$, the set of forbidden induced finite substructures is \mathcal{N} together with substructures that exclude vertices where P_i and P'_i hold simultaneously, and substructures that exclude vertices where neither P_i nor P'_i holds.

We claim that $S \in \text{CSP}(\Gamma')$ if and only if $S \models \Phi$. Let $S \in \text{CSP}(\Gamma')$. Since $\Gamma' \subseteq \Gamma$, and since every structure embedding in Γ does not contain a structure from \mathcal{N} , S satisfies Φ . Conversely, let A be a structure satisfying Φ , i.e., there exist relations P_i satisfying the first-order part of Φ . We expand the signature of A by the relation symbols P'_i and impose the relation P'_i on all vertices where the relation P_i does not hold. The expanded structure is in $\text{wForb}(\mathcal{N})$, and by universality of Γ an induced substructure of Γ . Thus the structure A in the original signature is an induced substructure of Γ' . This completes the proof of Proposition 3.9. \square

Note that $\text{CSP}(\Gamma) = \text{wSub}(\Gamma)$ for all the constructed templates Γ in

the above proof. Another interesting illustrating example also taken from [Madelaine and Stewart, 1999] is the following problem, which is again NP-hard.

TRI-FREE-TRI

INSTANCE: Graph $G = (V; E)$.

QUESTION: Is G tripartite *and* does not contain a triangle?

To reduce an instance G of Three-colorability to this problem we replace all edges xy in G by a gadget attached to x and y ; all the other vertices of the gadget are new vertices. The gadget for x and y looks as follows: Let H be a edge-minimal triangle-free graph which is not 3-colorable (clearly, such a graph exists; we even find graphs with arbitrarily high chromatic number and girth [Erdős, 1959]). Consider one vertex v in H , and link some non-empty subset of the neighbours of v in H to x and the other neighbours of v to y . Then delete v from H . It is clear, that if we assign different colors to x and y , then it is possible to properly 3-color the vertices from H , since the original graph H was a *minimal* triangle-free graph which is not 3-colorable. On the other hand, if x and y are assigned the same colors, we can not find a proper 3-coloring of the remaining vertices of the gadget.

By Proposition 3.9 this problem is in CSP*. The template looks as follows: Let \mathcal{C} be the class of all three-colored triangle-free graphs. This class is an amalgamation class, and thus there is a homogeneous Fraïssé-limit. Considering only the binary edge-relation, the reduced structure Γ is still ω -categorical. The problem CSP(Γ) is a reformulation of the above problem, and in CSP*.

Proposition 3.10. *The class CSP* is contained in monotone SNP.*

Proof. Let Γ be a finitely constrained ω -categorical structure. For each relation symbol R in the signature we introduce an existentially quantified relation R' of the same arity, and let these new relations be supersets of the input relations by the sentence $\forall \bar{x} \neg (R(\bar{x}) \wedge \neg R'(\bar{x}))$. Then use an universally quantified sentence to forbid the induced substructures from the finite axiomatization for the new existentially quantified relations R' ; here we need inequalities and negation.

The resulting sentence is in SNP, but contains input relations that do not occur negatively, and also contains inequalities. But since CSP(Γ) is closed under inverse homomorphisms, using the result from [Feder and Vardi,

2003], we can remove negation and inequalities, i.e., we can find an equivalent monotone SNP sentence without inequalities. \square

3.4 Datalog

Datalog is the language of logic programs without function symbols, see e.g. [Abiteboul et al., 1995, Kolaitis and Vardi, 1998, Immerman, 1998]. Very often the tractability of constraint satisfaction problems can be shown by Datalog programs. Datalog generalizes local consistency methods used in artificial intelligence, and also generalizes the concept of tree-duality from the theory of H -coloring problems.

This section discusses the power of Datalog for constraint satisfaction with infinite templates. Again, if we assume that the template is ω -categorical, some results for Datalog on constraint satisfaction problems with finite templates remain valid. We first define Datalog programs and bounded width problems, and then generalize the notion of *canonical Datalog programs* to ω -categorical templates. We also adapt a characterization of bounded width with pebble games from [Feder and Vardi, 1999, Kolaitis and Vardi, 1995] to the ω -categorical case. Using this we show that the problem Consistent-genealogy formulated in the introduction in Section 1.3 does not have width $(2, 3)$. This is interesting, since we do not know a bounded width constraint satisfaction problem with a finite template that does not have width $(2, 3)$. However, there is a (subquadratic) graph algorithm for Consistent-genealogy, see Chapter 6.

Datalog programs. Let τ be a relational signature; the relation symbols in τ will also be called the *input relation symbols*. A Datalog program consists of a set of propositional Horn clauses C_1, \dots, C_k containing atomic formulae with relation symbols from the signature τ , together with atomic formulae with some new relation symbols. Each clause is a set of literals where at most one of these literals is positive. The positive literals should not contain an input relation. The semantics of a Datalog program can be specified using *fixed point operators*, as e.g. in [Chandra and Harel, 1982, Kolaitis and Vardi, 1998]. Instead of giving these definitions, we first illustrate the concepts by

an example:

$$\begin{aligned} \text{oddpath}(x, y) &\leftarrow \text{edge}(x, y) \\ \text{oddpath}(x, y) &\leftarrow \text{oddpath}(x, s), \text{edge}(s, t), \text{edge}(t, y) \\ \text{false} &\leftarrow \text{oddpath}(x, x) \end{aligned}$$

In this example, the binary relation `edge` is the only input relation, `oddpath` is a binary relation computed by the program, and `false` is a 0-ary relation computed by the program. The Datalog program derives `false` if and only if the input graph is not 2-colorable. The first rule says that a single edge forms an odd path, the second rule says that adding two edges to an odd path forms an odd path, and the third rule says that the input graph is not bipartite, since it contains an odd cycle.

A Datalog program *solves* a problem, if the distinguished 0-ary predicate `false` is derived on an instance of the problem if and only if the instance has no solution. We say that a Datalog program has *width* (k, l) , $k \leq l$, if it has at most k variables in rule heads and at most l variables per rule. A problem is *of width* (k, l) , if it can be solved by a Datalog program of width (k, l) . The problem 2-colorability has for instance width $(2, 4)$, as demonstrated above. A problem has *width* k if it is of width (k, l) for some $l \geq k$, and it is of *bounded width*, if it has width k for some $k \geq 0$. It is easy to see that all bounded width problems are tractable, since the rules can derive only a polynomial number of facts.

Least fixed point logic. [Chandra and Harel, 1982] showed that Datalog has the same expressive power as *existential least fixed-point logic* ($\exists LFP$). To define an existential least fixed-point formula, we consider systems ϕ_1, \dots, ϕ_s of existential positive first-order logic formulae $\phi_i(x_1, \dots, x_{n_i}, S_1, \dots, S_s)$, where the S_j are n_j -ary relations symbols that are not in the input signature τ , for $1 \leq i, j \leq s$. We now closely follow the presentation in [Kolaitis and Vardi, 1998]. Let A be a τ -structure. Then such a system of formulae gives rise to an operator Φ from tuples (R_1, \dots, R_s) of relations R_i of arity n_i on the domain of A to tuples of relations of the same arity. If we apply the operator Φ to R_1, \dots, R_s , then the i th relation in the result is determined by

$$\Phi_i := \{(a_1, \dots, a_{n_i}) \mid A \models \phi_i(x_1/a_1, \dots, x_{n_i}/a_{n_i}, S_1/R_1, \dots, S_s/R_s)\}.$$

The *stages* $\Phi^m = (\Phi_1^m, \dots, \Phi_s^m)$ of Φ on A are defined by the following induction on m simultaneously for all $i \leq s$: $\Phi_i^1 = \Phi_i(\emptyset, \dots, \emptyset)$, $\Phi_i^{m+1} = \Phi_i(\Phi_1^m, \dots, \Phi_s^m)$, $i \leq s$, $1 \leq m$. Since all the formulae ϕ_i are positive in

the relation symbols S_1, \dots, S_s , the associated operator Φ is monotone in each of its arguments and has a least fixed point, and we denote its components by Φ_i^∞ . It is well-known that these least fixed points can be computed *bottom-up*, i.e., $\Phi_i^\infty = \bigcup_{m=1}^\infty \Phi_i^m$, $1 \leq i \leq s$ [Abiteboul et al., 1995]. On a finite structure A the operator Φ converges after finitely many iterations, i.e., there is an integer $m_0 \geq 0$ such that $\Phi^m = \Phi^{m_0}$ for every $m \geq m_0$. In general, we do not require that A is finite, but define the semantics of least fixed-point logic for arbitrary structures. We say that a formula is in $\exists\text{LFP}$ iff it is a component of the least fixed-point for some system of formulae. A formula is in $\exists\text{LFP}^l$, if the the system contains only formulae with at most l variables.

It was noted in [Chandra and Harel, 1982] that for every Datalog program of width (k, l) there is an $\exists\text{LFP}^l$ -sentence that is true on a finite structure A if and only if the Datalog program Π derives **false** on A . Conversely, for every $\exists\text{LFP}^l$ -sentence we find a corresponding Datalog program of width l . In fact, every Datalog program Π of width (k, l) can be *simulated* by a system of $\exists\text{LFP}^l$ -formulae in the sense that every non-input relation in Π corresponds to a disjunction of the primitive positive formulae that define the bodies of the rules having this relation as a head. The resulting system of existential positive l -variable formulae simulates Π *step-by-step*: Each stage of the system corresponds to a stage in the bottom-up evaluation of Π .

The expressive power of Datalog. Unfortunately we do not know of a characterization of width k or bounded width of constraint satisfaction problems in terms of polymorphisms, except for $k = 1$ for finite templates, where bounded width corresponds to the existence of a polymorphism that is a *set-operation* [Dalmau and Pearson, 1999, Feder and Vardi, 1999], i.e., a function f such that $f(x_1, \dots, x_n) = F(\{x_1, \dots, x_n\})$ for a function F mapping subsets of the domain D to D . It is not known whether bounded width, width k , or width (k, l) are decidable, for a given finite template T . Unboundedness of the constraint satisfaction problem for some concrete templates can be shown via the embedding of Datalog expressivity into the logic $\exists\mathcal{L}_{\infty\omega}^\omega$ [Kolaitis and Vardi, 2000, Kolaitis and Vardi, 1998], which in turn can be characterized by a certain two-person existential pebble game.

For every $l \geq 1$, let $\exists\mathcal{L}_{\infty\omega}^l$ be the l -variable existential positive fragment of $\mathcal{L}_{\infty\omega}^\omega$, that is, the collection of all formulae that have at most l distinct variables and are obtained from atomic formulae using infinitary disjunction, infinitary conjunction, and existential quantification only.

Theorem 3.11 (of [Kolaitis and Vardi, 1998]). *For every Datalog program*

Π of width (k, l) there is a sentence in $\exists\mathcal{L}_{\infty\omega}^l$ that is true in a structure S if and only if Π derives **false** on S .

In fact, by inspection of the proof of Theorem 4.3 in [Kolaitis and Vardi, 1998], we can assume that for each $\exists\text{LFP}^l$ -formula that corresponds to a relation computed by the Datalog program we find an $\exists\mathcal{L}_{\infty\omega}^l$ -formula that is an infinite disjunction of primitive positive formulae (this was already mentioned in [Chaudhuri and Vardi, 1992]). Also note that Theorem 4.3 in [Kolaitis and Vardi, 1998] (this is Theorem 3.11 above) was explicitly stated also for infinite structures S .

Canonical Datalog programs. Feder and Vardi [Feder and Vardi, 1999] observed that any problem of width (k, l) in CSP can also be solved by a *canonical Datalog program* of width (k, l) . The idea is to introduce new relation symbols for all at most k -ary relations on T , and to use them to infer all constraints on k variables that are entailed by a substructure of size l of the instance. The newly introduced 0-ary relation symbol corresponding to the empty relation will serve as **false**. This procedure can easily be modeled by a Datalog program of width (k, l) , see [Feder and Vardi, 1999], page 23. The same computational approach was studied independently in different terminology in [Dechter and van Beek, 1997], and the connection is explained in [Kolaitis and Vardi, 2000].

We can generalize canonical Datalog programs to ω -categorical templates Γ . To define the canonical Datalog program of width (k, l) (also called the *canonical (k, l) -program*), we introduce relation symbols for all at most k -ary first-order definable relations. By ω -categoricity (Theorem 2.6), there are only finitely many such relations. Then we insert a rule over this expanded signature with at most l variables and at most k variables in the rule head into our canonical (k, l) -program iff the corresponding implication is valid in Γ . Again the empty 0-ary relation serves as **false**.

Theorem 3.12. *Every constraint satisfaction problem with an ω -categorical template Γ of width (k, l) is solved by the canonical Datalog program of width (k, l) .*

Proof. First, we show that if the canonical Datalog program derives **false** on a given instance S , then this instance is unsatisfiable. Assume for contradiction that there is a homomorphism $f : S \rightarrow \Gamma$ although the canonical (k, l) -program for Γ derives **false** on the instance S . The derivation tree of **false** corresponds via f to a set of valid implications in the template. Finally, the

implication of **false** corresponds to an implication of the 0-ary empty relation in the template, a contradiction.

Let Π be a Datalog program of width (k, l) that solves the constraint satisfaction problem $\text{CSP}(\Gamma)$. We have to show that if an instance S of the constraint satisfaction problem is unsatisfiable, then also the canonical (k, l) -program for Γ derives **false** on S . Let τ be the set of non-input relation symbols that are computed by Π . Below we will find a mapping from τ to the signature of the canonical program such that for every rule r in Π , if we replace all symbols in τ by their image under this mapping, we obtain a rule from the canonical program. Moreover, this mapping will map the symbol **false** $\in \tau$ of Π to the symbol for the empty 0-ary relation in the canonical Datalog program. This clearly proves the claim.

We now define the mentioned mapping. Let ϕ be the $\exists\text{LFP}^l$ -formula that corresponds to a relation computed by Π on the template Γ (see the remarks after Theorem 3.11). Because of ω -categoricity, there are only finitely many inequivalent first-order formulae with k variables, for every k . Hence, every infinite disjunction or conjunction is equivalent to a finite conjunction or disjunction, and ϕ is over Γ equivalent to a first-order formula with at most l variables. For every relation computed by the Datalog program corresponds to an at most l -ary first-order definable relation over Γ , for which we have a relation symbol in the canonical Datalog program. We claim that this correspondence is the mapping we are looking for.

First we have to show that every rule in Π corresponds to a rule in the canonical program. Let r be a rule $R \leftarrow R_1, \dots, R_k$ in the Datalog program. Let $\phi, \phi_1, \dots, \phi_k$ be the first-order formulae corresponding to the relation symbols from the atomic formulae R, R_1, \dots, R_k , respectively. Since the rule r is part of the definition of the relation R computed by Π , $\phi_1 \wedge \dots \wedge \phi_k$ implies ϕ , and is in particular a valid implication in Γ . Hence r corresponds via renaming of the relation symbols to a rule in the canonical Datalog program.

Finally, we show that the correspondence maps **false** in Π to the empty relation over Γ . Also the least fixed-point corresponding to the 0-ary relation **false** in Π can be defined in the form $\bigcup_{i=1}^{\infty} \phi_i$, where ϕ_i is a primitive positive formula (again, see the remarks after Theorem 3.11). Here, the formulae ϕ_i do not have free variables and are sentences. We will show that all the sentences ϕ_i are not true in Γ , and therefore **false** corresponds to the empty 0-ary relation over Γ . Suppose one of these sentences ϕ is true in Γ . Let w be the number of existentially quantified variables in ϕ . We can view ϕ as an instance S' of $\text{CSP}(\Gamma)$ on w nodes. Because ϕ holds in Γ , it is a

satisfiable instance of $\text{CSP}(\Gamma)$. But Π derives **false** on S' , which contradicts the assumption that Π solves $\text{CSP}(\Gamma)$. \square

Remark. The fact that every fixed-point formula is over an ω -categorical structure equivalent to a first-order formula is well-known and has been used to extend the so-called 0-1 law [Fagin, 1976] from first-order logic to fixed-point logic. A language over a relational signature τ has a 0-1 law if for every sentence ϕ in this language the fraction of models with universe $\{1, \dots, n\}$ that satisfy ϕ either tends to 0 or to 1 when n approaches infinity. We also say that ϕ *holds almost surely* over the class of all finite τ -structures, and the set of all such sentences we call the *almost sure theory* of that class. It is an interesting question for which ω -categorical structures Γ the age of Γ has a first-order 0-1 law. This is for instance the case for the class of K_n -free graphs [Kolaitis et al., 1987]. In this case the almost sure theory is again ω -categorical and therefore we even have a 0-1 law for fixed point formulae over the class of K_n -free graphs.

Pebble games. We write $A \preceq^l B$ to denote that every sentence in $\exists \mathcal{L}_{\infty\omega}^l$ that is true in the structure A is also true in the structure B . This relation can be characterized in terms of the following infinitary pebble game.

The *existential l -pebble game* is played by the players Spoiler and Duplicator on finite structures A and B of the same relational signature. Each player has l pebbles, p_1, \dots, p_l for the Spoiler and q_1, \dots, q_l for the Duplicator. Spoiler places his pebbles on elements of A , Duplicator on elements of B . Initially, no pebbles are placed. In each round of the game, Spoiler picks up some pebble, say p_i . If p_i is already placed on A , then Spoiler removes p_i from A , and Duplicator responds by removing the pebble q_i from B . If p_i is not placed on an element of A , then Spoiler places p_i on some element of A , and Duplicator responds by placing q_i on some element of B .

Let i_1, \dots, i_m be the indices of the pebbles that are placed on A (and B) after the i -th round. Let a_{i_1}, \dots, a_{i_m} (b_{i_1}, \dots, b_{i_m}) be the elements of A (B) pebbled by the pebbles p_{i_1}, \dots, p_{i_m} (q_{i_1}, \dots, q_{i_m}) after the i -th round. If the mapping h with $h(a_{i_j}) = b_{i_j}$, $1 \leq j \leq m$, is not a homomorphism between A restricted to $\{a_{i_1}, \dots, a_{i_m}\}$ and B restricted to $\{b_{i_1}, \dots, b_{i_m}\}$, then Spoiler wins the game. Duplicator wins, if the game continues *forever*, i.e., if Spoiler can never win the game.

Theorem 3.13. *Let A and B be relational structures. Then $A \preceq^l B$ if and only if Duplicator wins the existential l -pebble game on A and B .*

A proof of this theorem can be found according to Theorem 4.8 and Remark 4.12 in [Kolaitis and Vardi, 1995]. Again, we would like to stress that there it is not required that A and B are finite.

For constraint satisfaction problems, we even have the stronger property that $\text{CSP}(\Gamma)$, where Γ is ω -categorical, can not be solved by a Datalog program *if and only if* for each k we find a structure such that Duplicator wins the game on that structure and the template. Using the canonical Datalog program, this was stated for finite templates in [Feder and Vardi, 1999]. The same proof works for ω -categorical structures, using the definition of canonical (k, l) -program presented in this section. Feder and Vardi could also characterize width (k, l) by the following modification of the pebble game from [Kolaitis and Vardi, 1995] that we described above. The idea is that in this modified so-called *existential (k, l) -pebble game*, in each round the players keep only k out of l placed pebbles, and then Spoiler places at most $l - k$ of the other pebbles one-by-one on new vertices, and Duplicator answers as before. Hence, each round now consists of at most $l - k$ placements of pebbles.

Theorem 3.14 (of [Feder and Vardi, 1999]). *Let Γ be ω -categorical. Then the canonical (k, l) -program for $\text{CSP}(\Gamma)$ accepts an instance S if and only if Spoiler has a winning strategy in the existential (k, l) -pebble game on S and Γ .*

A lower bound for Consistent-genealogy. Now we prove that the problem Consistent-genealogy does not have width $(2, 3)$. Recall that this problem can be seen as $\text{CSP}((\Lambda; <, \not\leq))$; see Section 1.3. To formulate the canonical $(2, 3)$ -program for this problem we introduce names for all binary first-order definable relations. It will turn out that we can restrict our attention to the binary relations on *distinct* elements. In Section 2.7 we showed that they are all boolean combinations of the two complete 2-types ‘ $<$ ’ and ‘ \perp ’. The relation $\not\leq$ in the signature of the template is for instance defined by $x \not\leq y \Leftrightarrow x > y \vee x \perp y$. Besides the symbols \mathcal{R} and \emptyset that we use for the full and the empty relation on two distinct elements, the only remaining relation is the *comparability* relation in the semilinear order Λ , denoted by ‘ \sim ’, with the following definition: $x \sim y \Leftrightarrow x < y \vee y < x$.

The canonical $(2, 3)$ -program is already quite large: It contains a rule for all implications of the form $\exists y. xR_1y \wedge yR_2z \rightarrow xR_3z$ that are valid in Λ , where R_1, R_2, R_3 are binary first-order definable relations in Λ . It is most convenient to describe it by a composition table. The composition operation

\circ	\mathcal{R}	$<$	$>$	\perp	$\not\leq$	$\not\geq$	\sim	\emptyset
\mathcal{R}	\emptyset							
$<$	\mathcal{R}	$<$	\mathcal{R}	\perp	\mathcal{R}	$\not\geq$	\mathcal{R}	\emptyset
$>$	\mathcal{R}	\sim	$>$	$\not\leq$	$\not\leq$	\mathcal{R}	\sim	\emptyset
\perp	\mathcal{R}	$\not\geq$	\perp	\mathcal{R}	\mathcal{R}	\mathcal{R}	$\not\leq$	\emptyset
$\not\leq$	\mathcal{R}	\mathcal{R}	$\not\leq$	\mathcal{R}	\mathcal{R}	\mathcal{R}	\mathcal{R}	\emptyset
$\not\geq$	\mathcal{R}	$\not\geq$	\mathcal{R}	\mathcal{R}	\mathcal{R}	\mathcal{R}	\mathcal{R}	\emptyset
\sim	\mathcal{R}	\sim	\mathcal{R}	$\not\leq$	\mathcal{R}	\mathcal{R}	\mathcal{R}	\emptyset
\emptyset	\emptyset							

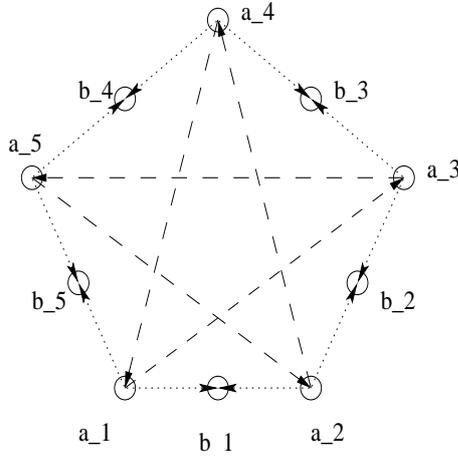


Figure 3.4: The instance C . A dashed arc from vertex a to vertex b indicates $a < b$, a dotted arc from a to b indicates $a > b$.

\circ maps two binary relations to another binary relation:

$$R_1 \circ R_2 := \{(x, z) \mid \exists y. xR_1y \wedge yR_2z\}$$

This composition operator essentially describes the canonical $(2, 3)$ -program, and the composition table for this operator is shown for $\text{CSP}((\Lambda; <, \not\leq))$ in Figure 3.4. We now show that $\text{CSP}((\Lambda; <, \not\leq))$ does not have width $(2, 3)$. To this end, we define an instance C without a homomorphism to $(\Lambda; <, \not\leq)$, but where the canonical $(2, 3)$ -program does not derive false.

Proposition 3.15. *The instance C of $\text{CSP}((\Lambda; \sim, \not\leq))$ shown in Figure 3.4 is not satisfiable.*

Proof. Assume there is a homomorphism f from C to $(\Lambda; <, \not\leq)$. Clearly, the

transitive reduction of the relation $<$ on a finite subset of Λ is a forest. First note that for $f(C)$, this forest is a single tree: Suppose for contradiction that the solution contains *two* trees, and let u, v be vertices in C that are mapped to the two roots of these trees. Then there is a (undirected) path of vertices in C that connects u with v in ' $<$ '. If we follow this path, all vertices have to be mapped below $f(u)$, since they are connected to the previous vertex either with ' $<$ ' or ' $>$ ', and because the vertex u was topmost in the solution. Eventually, we end at v , but $f(v)$ lies disjoint to $f(u)$, a contradiction. Next, note that no vertex in C can denote the root of a solution to C , since every vertex has an in-coming edge in $\not\prec$, which prevents the vertex to lie top-most in the solution. Hence, C has no solution. \square

Proposition 3.16. *The canonical (2,3)-program does not derive false on the instance C of $CSP((\Lambda; \sim, \not\prec))$.*

Proof. The canonical program first derives that $a_i \sim a_{i+1}$, where $i \geq$ is understood modulo 6. It also derives that a_i that $a_i \not\prec b_{i+1}$ and $a_i \not\prec b_{i+2}$. Then it reaches a fixed-point: for all further triples in C , none of the composition rules shown in Figure 3.4 applies. All the triples such that the induced substructure contains at least two constraints are shown in Figure 3.5: up to isomorphism, there are 13. It can be checked that in each of them no further constraint is entailed. \square

Proposition 3.16 and 3.15, together with Theorem 3.12, yield the following.

Corollary 3.17. *There is no Datalog program of width (2,3) that solves the problem Consistent-genealogy.*

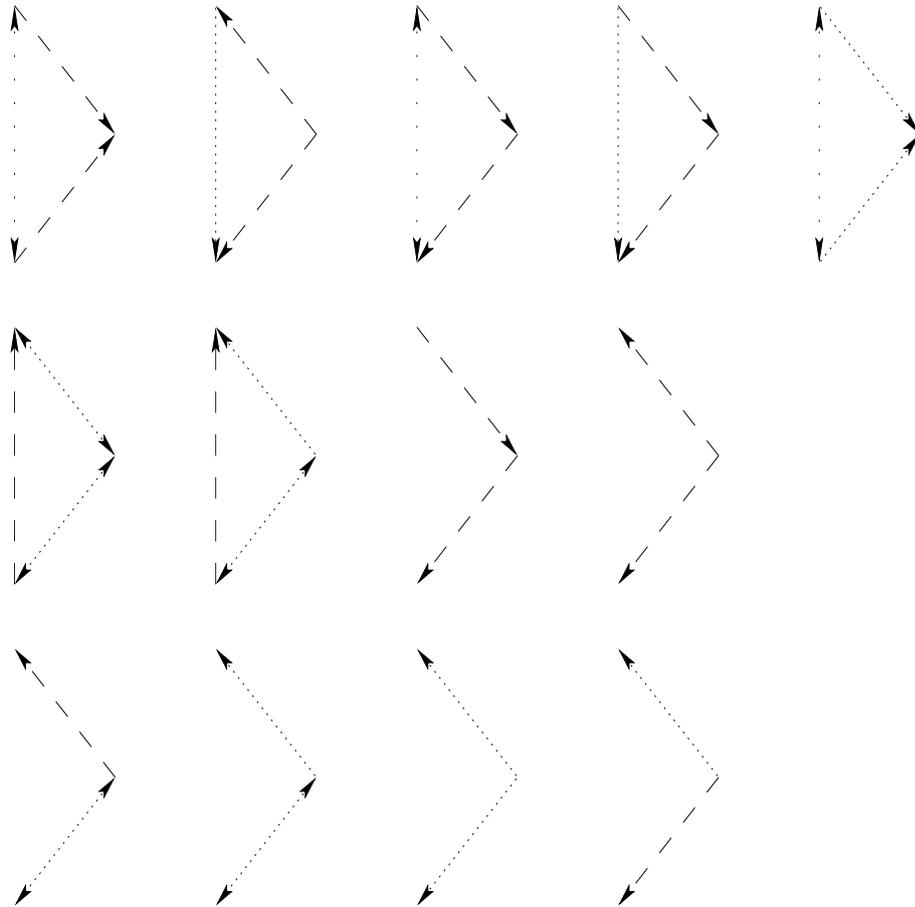


Figure 3.5: These are all the 13 constellations on three vertices containing at least two constraints in the fixed point computed by the canonical $(2, 3)$ -program on the instance C .

Chapter 4

The Clone of Polymorphisms

Adding finitely many relations to a template Γ that are *primitive positive* definable over Γ does not change the computational complexity of $\text{CSP}(\Gamma)$. For finite templates Γ , it is a central theorem for the approach in [Jeavons, 1998] that a relation is primitive positive definable over Γ if and only if it is invariant under the polymorphisms of Γ [Bodnarčuk et al., 1969, Geiger, 1968]. This was first used in the context of constraint satisfaction in [Jeavons et al., 1997, Jeavons et al., 1998], and initiated the algebraic approach to constraint satisfaction, which has successfully been carried further e.g. in [Jeavons et al., 1997, Jeavons et al., 1998, Bulatov et al., 2000, Dalmau, 2000b, Bulatov et al., 2001, Bulatov, 2002b, Bulatov, 2002a, Bulatov, 2003]. We generalize this result to ω -categorical structures Γ : A relation is p.p.-definable in Γ if and only if it is invariant under the polymorphisms of Γ .

We first introduce the necessary tools from universal algebra in Section 4.1. We then state some facts for finite clones 4.2, and show how some of them fail for arbitrary infinite clones 4.3. Section 4.4 will be the starting point to generalize various Galois-connections from finite to ω -categorical structures. In Section 4.5 we then present the characterization of primitive positive definability in ω -categorical structures. Section 4.6 contains a discussion of one case where the presence of a polymorphism in a template implies tractability of the constraint satisfaction problem: this is so if the template of a constraint satisfaction problem has a *near-unanimity* operation. Finally we present an application of the notion of a core, and show that we can expand ω -categorical cores by singleton-relations without increasing the computational complexity of the corresponding constraint satisfaction problem – for finite templates, this was proven in [Bulatov et al., 2003].

4.1 Tools from Universal Algebra

In this section, D will stand for a countable set and O for the set of *finitary operations* on D , i.e., functions from D^k to D for finite k . We say that $f \in O$ *preserves* a k -ary relation $R \subseteq D^k$ iff R is a subalgebra of $(D, f)^k$. An operation that preserves all relations of a relational structure Γ is called a *polymorphism* of Γ . The set of all k -ary polymorphisms of Γ is denoted by $Pol^{(k)}(\Gamma)$, and we write $Pol(\Gamma)$ for the set of all finitary polymorphisms $Pol(\Gamma) = \bigcup_{i=1}^{\infty} Pol^{(i)}(\Gamma)$.

The notion of a *product* of relational structures allows an equivalent definition of polymorphisms, relating polymorphisms to homomorphisms. The (*categorical- or cross-*) *product* $\Gamma_1 \times \Gamma_2$ of two relational τ -structures Γ_1 and Γ_2 is a τ -structure on the domain $D_{\Gamma_1} \times D_{\Gamma_2}$. For all relations $R \in \tau$ the relation $R((x_1, y_1), \dots, (x_k, y_k))$ holds in $\Gamma_1 \times \Gamma_2$ iff $R(x_1, \dots, x_k)$ holds in Γ_1 and $R(y_1, \dots, y_k)$ holds in Γ_2 . Comparing the corresponding definitions we see that a k -ary polymorphism f of a relational structure is a homomorphism from $\Gamma^k = \Gamma \times \dots \times \Gamma$ to Γ , i.e., for an m -ary relation R in τ , iff $R(x_1, \dots, x_m)$ holds in Γ^k then $R(f(x_1), \dots, f(x_m))$ holds in Γ .

An operation π is a *projection* (or a *trivial polymorphism*) iff $\pi(x_1, \dots, x_n) = x_i$ for all n -tuples and some fixed $i \in \{1, \dots, n\}$. The *composition* of a k -ary operation f and k operations g_1, \dots, g_k of arity n is an n -ary operation defined by

$$f(g_1, \dots, g_k)(x_1, \dots, x_n) = f(g_1(x_1, \dots, x_n), \dots, g_k(x_1, \dots, x_n)) \quad .$$

A *clone* C is a set of operations from O that is closed under composition and that contains all projections. We write D_C for the *domain* D of the clone C . For a set of operations F from O we write $\langle F \rangle$ for the smallest clone containing all operations in F . We say that the clone $\langle F \rangle$ is *generated* by F in this way; this defines an algebraic closure operator. Thus, the clones of operations on D form an algebraic lattice. Observe that $Pol(\Gamma)$ is a clone with the domain D_{Γ} [Kalužnin and Pöschel, 1979].

Local Closure. Moreover, $Pol(\Gamma)$ is also closed under interpolation: We say that an operation $f \in O$ is an *interpolation* of a subset F of O iff for every finite subset B of D there is some operation $g \in \langle F \rangle$ such that $f|_B = g|_B$ (f restricted to B equals g restricted to B , i.e., $f(a) = g(a)$ for every tuple a over the set B). The set of interpolations of F is called the *local closure* of F and denoted by $Loc(F)$. Note that the clone of polymorphisms of a

countable structure is locally generated by a countable number of polymorphisms: choose for each finite set B and all potential images of tuples from B a polymorphism if there exists such a polymorphism. The following is a well-known fact.

Proposition 4.1. *A set $F \subseteq O$ of operations is locally closed if and only if F is the set of polymorphisms of Γ for some relational structure Γ .*

Many results on clones in general can be found in [Szendrei, 1986]. Important properties of operations in a clone can be specified with identities that are satisfied by the operations. We list some fundamental properties below, where the free variables in the identities below are understood as universally quantified. A k -ary operation f is

- a *projection* iff there is an $i \in \{1, \dots, k\}$ such that $f(x_1, \dots, x_k) = x_i$;
- *conservative* iff $f(x_1, \dots, x_k) \in \{x_1, \dots, x_k\}$;
- *idempotent* iff $f(x, \dots, x) = x$;
- *essentially unary* iff there is an $i \in \{1, \dots, k\}$ and an unary operation f_0 such that $f(x_1, \dots, x_k) = f_0(x_i)$;
- a ternary *majority operation* iff $f(x, y, y) = f(y, x, y) = f(y, y, x) = y$;
- a ternary *minority operation* iff $f(x, y, y) = f(y, x, y) = f(y, y, x) = x$;
- a *semiprojection* iff $f(x_1, \dots, x_k) = x_i$ whenever $|\{x_1, \dots, x_k\}| < k$, and otherwise $f(x_1, \dots, x_k) = x_j$ with fixed i, j .

Let F be a (local) clone with domain D . Then $R \subseteq D^m$ is invariant under F iff every $f \in F$ preserves R . We denote by $Inv(F)$ the relational structure containing the set of all relations invariant under F . The set of relations in this relational structure is closed under *projections* and various other closure operators on sets of relations over D , and is called a *relational clone* [Kalužnin and Pöschel, 1979].

Minimal clones. The atoms of the lattice of all clones on a set are called *minimal clones*. This implies immediately that a clone C is minimal if and only if every nontrivial $f \in C$ generates every nontrivial $g \in C$. Nontrivial operations of minimal arity in a minimal clone are called *minimal operations*. Clearly all nonunary minimal operations are idempotent.

For infinite structures, such atoms need not necessarily exist: This is for instance the case for the clone generated by the successor function on the natural numbers. In this clone we find the operations $s_z(x) := x + 2^z$. If $z_1 < z_2$, then s_{z_1} generates s_{z_2} , but not the other way. This clone does not contain a minimal clone.

However, if a clone is minimal, it falls into one out of five classes. Although this was mainly used for clones over a finite domain, it also holds for clones on infinite domains [Rosenberg, 1986].

Theorem 4.2 (of [Rosenberg, 1986]). *Every minimal operation is of one of the following types:*

1. a unary operation,
2. a binary idempotent operation,
3. a ternary majority operation,
4. a ternary minority operation,
5. an m -ary semiprojection ($m \geq 3$).

Let $m > n \geq 1$. Then an m -ary semiprojection on an n -element set necessarily is a projection. Thus the theorem shows that a clone on a finite set only has a finite number of minimal operations, and only contains a finite number of minimal clones.

The unary case. We say that a k -ary operation f *depends* on an argument i iff there is no $k-1$ -ary operation f' such that $f(x_1, \dots, x_k) = f'(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_k)$. Hence, an essentially unary operation is an operation that depends on one argument only. We claim that a k -ary operation f depends on argument i if and only if there are x_1, \dots, x_k and x'_i such that $f(x_1, \dots, x_k) \neq f(x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_k)$: first suppose there were x_1, \dots, x_k and x'_i such that $f(x_1, \dots, x_k) \neq f(x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_k)$, and suppose there was such a function f' . Then we have $f(x_1, \dots, x_i, \dots, x_k) = f'(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_k) = f(x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_k)$, a contradiction. For the converse, suppose that we do not find points x_1, \dots, x_k and x'_i such that $f(x_1, \dots, x_k) \neq f(x_1, \dots, x_{i-1}, x'_i, x_{i+1}, \dots, x_k)$. In this case we can define a function f' by $f'(x_1, \dots, x_{i-1}, x_{i+1}, \dots, x_k) = f(x_1, \dots, x_i, \dots, x_k)$ for some value of x_i – this is well-defined since the result is the same for all

possible choices for x_i . This operation f' shows that f does not depend on the i -th argument.

The essentially unary clones have a characterization on the relational side, which also holds for infinite domains. Consider the following relation on Γ .

$$P_4 := \{(a, b, c, d) \mid a = b \text{ or } c = d; a, b, c, d \in D_\Gamma\}$$

Proposition 4.3 (Lemma 1.3.1 in [Kaluzhnin and Pöschel, 1979], page 56). *A clone F is essentially unary if and only if $P_4 \in \text{Inv}(F)$.*

Proof. Clearly every essentially unary operation preserves P_4 . Suppose a k -ary function f is not essentially unary, but depends on the i -th and j -th argument, $1 \leq i \neq j \leq k$. Hence there exist tuples $a_1, b_1, a_2, b_2 \in D_F^k$ where a_1, b_1 and a_2, b_2 only differ at the entries i and j , respectively, such that $f(a_1) \neq f(b_1)$ and $f(a_2) \neq f(b_2)$. Since $(a_1(l), b_1(l), a_2(l), b_2(l)) \in P_4$ for all $l \leq k$, but $(f(a_1), f(b_1), f(a_2), f(b_2)) \notin P_4$, we conclude that $P_4 \notin \text{Inv}(F)$. \square

4.2 Clones on Finite Domains

Most results on clones were formulated for clones with a finite domain. See [Szendrei, 1986]. We start with a fundamental result of [Bodnarčuk et al., 1969, Geiger, 1968] (other presentations can be found in [Kaluzhnin and Pöschel, 1979]), which says that for arbitrary finite relational structures T the p.p.-definable relations can be characterized as the invariants of the polymorphisms of T .

Theorem 4.4 (of [Bodnarčuk et al., 1969, Geiger, 1968]). *Let T be a finite relational structure. Then*

$$\langle T \rangle_{pp} = \text{Inv}(\text{Pol}(T)) .$$

The proof of Theorem 4.4 also shows that it is decidable whether for a given finite relational structure T a given relation R is p.p.-definable or not. If $w = |R|$ is the number of tuples in the relation R then $\text{Pol}(R)$ is generated by the polymorphisms of arity w . To test whether R is p.p.-definable we only have to check whether all functions $f : T^w \rightarrow T$ that are polymorphisms of T also preserve R . There is a finite number of such functions.

Projectivity. A relational structure Γ is called *projective* iff all idempotent polymorphisms of Γ are projections.

Proposition 4.5. *If a finite relational structure Γ is a core, then Γ is projective if and only if every polymorphism is essentially unary.*

Proof. Let f be an idempotent operation, and assume that f is essentially unary, i.e., $f(x_1, \dots, x_k) = f_0(x_i)$ for some unary function f_0 and some $i \leq k$. But $f(x_1, \dots, x_k) = f_0(x_i) = f(x_i, \dots, x_i) = x_i$, i.e., f is a projection.

Conversely, assume that Γ is a projective core, and let f be a polymorphism of Γ . Then $f^*(x) = f(x, \dots, x)$ is an endomorphism, and, since Γ is a finite core, an automorphism of Γ . Let $g(x)$ be the inverse of the automorphism f^* . The operation $g(f(x_1, \dots, x_k))$ is idempotent, and by projectivity of Γ a projection, say, a projection to the i -th argument. Then $f(x_1, \dots, x_k) = f^*(g(f(x_1, \dots, x_k))) = f^*(x_i)$, and therefore f is essentially unary. \square

Projectivity and hardness. Theorem 4.4 and Proposition 4.5 can be used to prove the hardness of a constraint satisfaction problem with a finite template T . First assume without loss of generality that T is a core – see Section 2.8. If a finite structure T is projective, then all polymorphisms are essentially unary, i.e., for every polymorphism f of arity k there exists a unary function f_0 and an index $i \leq k$ such that $f(x_1, \dots, x_k) = f_0(x_i)$. Since T is a finite core the endomorphism f_0 is an automorphism, and thus preserves the inequality relation. Therefore every polymorphism preserves the inequality relation. By Theorem 4.4, \neq has a primitive positive definition in T . By Lemma 3.3, and because n -colorability is NP-hard for any finite $n \geq 3$, we conclude that $\text{CSP}(T)$ is NP-hard for $|T| \geq 3$.

For finite structures T we have special tools to test whether a structure is projective. Observe that every clone C on a finite domain contains a minimal clone. To prove this take any nontrivial operation in C of minimal arity k . By Theorem 4.2 k cannot exceed the number of elements in the domain of C , and thus there is only a finite number of such operations. The set of all clones generated by such operations, partially ordered by inclusion, contains at least one minimal element, which is a minimal clone contained in C .

Since T is a core, all unary polymorphisms are automorphisms. To find out whether T is projective, we use the classification of minimal clones from Section 4.1, and it suffices to check whether the clone contains a binary

idempotent operation, a ternary majority or minority operation, or a semi-projection of arity $\leq |T|$.

Of course, it could be the case that T is not projective, but $\text{CSP}(T)$ is still NP-hard. But in any case the above technique helps to understand the polymorphism clone of T . For infinite structures this method fails: The random graph for instance is projective [Luczak and Nešetřil, 2004], but has a trivial constraint satisfaction problem.

4.3 Clones on Infinite Domains

Generalizations of Theorem 4.4 and the related Galois connections were also studied for infinite domains. For arbitrary relational structures Γ the structure $\text{Inv}(\text{Aut}(\Gamma))$ is homogeneous. The set of relations $\text{Inv}(\text{Pol}(\Gamma))$ was characterized with local closure operators on relational algebras in [Szabó, 1978] (see also [Pöschel, 1980], page 32).

In Section 4.5 we give a direct proof that for ω -categorical structures Γ a relation is in $\langle \Gamma \rangle_{pp}$ if and only if it is preserved by all polymorphisms. But first we note that the following is well-known for arbitrary cardinalities of the domain.

Proposition 4.6 (see e.g. [Kaluzhnin and Pöschel, 1979]). *Let Γ be a relational structure. Then*

$$\langle \Gamma \rangle_{pp} \subseteq \text{Inv}(\text{Pol}(\Gamma)) .$$

Proof. Let R be a relation in $\langle \Gamma \rangle_{pp}$. We prove that $R \in \text{Inv}(\text{Pol}(\Gamma))$ by induction on the length of a defining p.p.-formula φ . The claim is true for $\varphi = R(x_1, \dots, x_n)$. For $\varphi = \exists x.\varphi'$ we observe that every polymorphism that preserves φ' also preserves φ . The same holds for $\varphi = \varphi_1 \wedge \varphi_2$. \square

For a structure with a countable domain the inclusion of Proposition 4.6 might be strict. Consider for instance the following relational structure $\Gamma = (\mathbb{N}; P_4, \{0\}, \text{suc})$ on the natural numbers \mathbb{N} . Here P_4 is the relation introduced in Section 4.1, and ‘suc’ is the binary successor relation $\{(a, a + 1) \mid a \in \mathbb{N}\}$ on the natural numbers. We show that $\text{Inv}(\text{Pol}(\Gamma))$ contains relations that are not p.p.-definable. Every function preserving P_4 is essentially unary. If a unary function f preserves the unary predicate containing 0 only, then $f(0) = 0$. Furthermore, if f preserves the successor relation we have $f(a +$

$1) = f(a) + 1$ for all a , and inductively it follows that $f(a) = a$. Therefore $Pol(\Gamma)$ is the set of all projections. Every projection preserves all relations. There are uncountably many relations but only countably many primitive positive formulas. Therefore $\langle \Gamma \rangle_{pp}$ is a strict subset of $Inv(Pol(\Gamma))$.

For ω -categorical structures Γ the situation looks better: It is known that the first-order definable relations are precisely the relations that are preserved by the automorphisms of Γ , i.e.

$$\langle \Gamma \rangle_{fo} = Inv(Aut(\Gamma)) \tag{4.1}$$

See e.g. [Hodges, 1997, Cameron, 1990]. We prove a corresponding theorem for primitive positive definability in Section 4.5; but first we take a closer look at (4.1) in Section 4.4.

4.4 The Basic Galois-Connection Inv-Aut

In this section we discuss what is well-known in model theory (see again [Hodges, 1997, Cameron, 1990]), but transfer it to the language of [Kalužnin and Pöschel, 1979] in universal algebra. In particular, we will reformulate the theorem of Ryll-Nardzewski as a Galois connection. This aspect of the classical result in model theory is our starting point to generalize other Galois connections from finite structures to ω -categorical structures. As we have just seen in Section 4.3, these theorems do not hold for arbitrary structures on infinite domains.

The first observation is that the assumption of ω -categoricity for (4.1) in Section 4.3 is even best-possible, since (4.1) holds if and only if the structure Γ is ω -categorical.

Theorem 4.7. *A structure Γ is ω -categorical if and only if*

$$Inv(Aut(\Gamma)) = \langle \Gamma \rangle_{fo} .$$

Proof. Let Γ be an ω -categorical structure. Since clearly $\langle \Gamma \rangle_{fo} \subseteq Inv(Aut(\Gamma))$ we only have to prove that a relation R preserved by all automorphisms is first-order definable. The relation R is a union of orbits of Γ^k for some k . On ω -categorical structures the orbits of k -tuples are the complete k -types. The Theorem 2.6 of Ryll-Nardzewski implies that all types are principal the relation R is first-order definable.

Now suppose Γ is not ω -categorical. Then for some k , there is an infinite number of orbits of k -tuples. Since the union of every subset of such orbits is preserved by all automorphisms, there is an uncountable number of relations in $Inv(Aut(\Gamma))$. Since there is only a countable number of formulas, there exists an invariant relation which is not first-order definable over Γ . \square

Note that this establishes a *Galois connection*: Generally, suppose P and Q are partially ordered sets and $f : P \rightarrow Q$ and $g : Q \rightarrow P$ are order preserving maps. Assume $fgf \geq f$ and $gfg \geq g$. Then $gfgf = gf$. It follows that f and g set up a bijective correspondence between $gf[P]$ and $fg[Q]$. In our case, P is the set of ω -categorical structures closed under first-order definability, and $g=Inv$ and $f=Aut$. Theorem 4.7 characterizes the closure operator gf .

Next we present a well-known characterization of the closure operator fg . There is a natural topology defined on the symmetric group, that of *pointwise convergence*: A sequence (g_n) of permutations *tends to the limit* g iff for any $a \in D$ there is a $n_0 \geq 0$ such that for all $n \geq n_0 : g_n(a) = g(a)$. A permutation group is closed iff for all sequences (g_n) in the group that tend to some limit g , the permutation g is also in the group.

Theorem 4.8 (see [Cameron, 1996]). *A permutation group G is closed if and only if $G = Aut(Inv(G))$.*

Note that the structure $Inv(G)$ for a closed permutation group G is related, but slightly different to the *canonical structure* of G defined in [Cameron, 1996, Hodges, 1997]. The canonical structure associated to a closed permutation group contains for each orbit $O \subseteq D^k$ a k -ary relation symbol R which is interpreted by O . Thus each relation in $Inv(G)$ has a definition in the canonical structure with a disjunction, and if G is oligomorphic has a definition with a *finite* disjunction.

Finally we make the remark that the automorphism group of an ω -categorical structure always has cardinality 2^ω . To see this note that by ω -categoricity the pointwise stabilizer of a finite set of points is always non-trivial. Thus we find uncountably many permutations as the limit points in the closed group.

4.5 Primitive Positive Definability

We characterize the primitive positive first-order definable relations over an ω -categorical structure Γ by the polymorphisms of Γ of finite arity.

Theorem 4.9. *Let Γ be an ω -categorical structure with relational signature τ . Then a relation R on Γ is invariant under the polymorphisms of Γ if and only if R is p.p.-definable, i.e.,*

$$\langle \Gamma \rangle_{pp} = \text{Inv}(\text{Pol}(\Gamma)).$$

Proof. We already stated in Proposition 4.6 that the p.p.-definable relations over Γ are invariant under the polymorphisms of Γ . For the converse, let R be a k -ary relation from $\text{Inv}(\text{Pol}(\Gamma))$. Note that R is first-order definable in Γ : By ω -categoricity and Ryll-Nardzewski, and since Γ and $\text{Inv}(\text{Pol}(\Gamma))$ have the same automorphism group, the relation R is a union of *finitely* many orbits of the automorphism group of Γ , and it can be defined by a disjunction φ of τ -formulas that define these orbits. Let M_1, \dots, M_w be the satisfiable monomials in this disjunction, and let x_1, \dots, x_k be the variables of the monomials.

We have to construct a finite τ -structure Q with designated vertices v_1, \dots, v_k such that

$$R = \{(f(v_1), \dots, f(v_k)) \mid f: Q \rightarrow \Gamma \text{ homomorphism}\}.$$

The idea is to first consider an *infinite* τ -structure, namely the categorical product Γ^w , and then to apply König's Lemma to prove the existence of a suitable finite substructure.

For each monomial $M_j \in M_1, \dots, M_w$ of φ we find a substructure a_1^j, \dots, a_k^j of Γ , such that a_1^j, \dots, a_k^j satisfies M_j in Γ . Let b_1, b_2, \dots be an enumeration of the w -tuples in D_Γ^w , starting with $b_i = (a_1^i, \dots, a_w^i)$ for $1 \leq i \leq k$. Let us call a partial mapping from Γ^w to Γ a *bad* mapping if it maps b_1, \dots, b_k to a tuple not satisfying φ . Since R is invariant under all polymorphisms, no homomorphism from Γ^w to Γ is bad.

We now claim that there is a finite substructure Q of Γ^w such that no homomorphism from Q to Γ is bad. Assume for contradiction that all finite substructures of Γ^w containing b_1, \dots, b_k have a homomorphism to Γ mapping b_1, \dots, b_k to a tuple not satisfying φ . We now construct a bad homomorphism from Γ^w to Γ , i.e. the images of b_1, \dots, b_k do not satisfy φ . This contradicts the fact that R is invariant under all polymorphisms.

To this end, consider the following infinite but finitely branching tree. The nodes on level n in the tree are the equivalence classes of the bad homomorphisms from $\Gamma^w|_{b_1, \dots, b_n}$ to Γ , where two homomorphisms f_1 and f_2 are equivalent if $f_1 = gf_2$ for some $g \in \text{Aut}(\Gamma)$. Adjacency between nodes on consecutive levels is defined by restriction. By our assumption, for each finite substructure of Γ^w there is a bad homomorphism, and thus the tree contains a node on each level. By the theorem of Ryll-Nardzewski, there are only finitely many nodes at each level. By König's Lemma the tree contains an infinite path. This path defines a bad homomorphism from Γ^w to Γ .

We proved by contradiction that there must be a finite substructure Q containing the vertices b_1, \dots, b_k of Γ^w such that all homomorphisms from Q to Γ map b_1, \dots, b_k to a tuple satisfying φ . Conversely, every mapping $f : Q \rightarrow \Gamma$ such that the tuple $(f(b_1), \dots, f(b_k))$ satisfies in Γ the monomial M_j can be extended to a homomorphism $f : \Gamma^w \rightarrow \Gamma$. To see this note that both a_1^j, \dots, a_k^j and $(f(b_1), \dots, f(b_k))$ satisfy M_j and thus both lie in the same orbit of $\text{Aut}(\Gamma)$. Thus we can choose f to be the j th projection combined with the automorphism sending (a_1^j, \dots, a_k^j) to $(f(b_1), \dots, f(b_k))$. This completes the proof. \square

The proof says something about the arity of the polymorphisms we have to consider if we are interested in a particular relation. Let R be a first-order definable k -ary relation over an ω -categorical structure Γ . Then R is the union of a finite number w of orbits of k -tuples in Γ , or equivalently, R can be defined by a disjunction of a finite number w of complete k -types. The proof shows that $\text{Pol}(R)$ is locally generated by all w -ary polymorphisms.

4.6 Near-unanimity Operations

In this section we study the case where the clone of polymorphisms of a relational structure contains a so-called *near-unanimity operation*. The existence of such an operation has several different characterizations, in terms of local consistency, and Datalog programs. The connections were exhibited for finite structures in [Feder and Vardi, 1999, Kolaitis and Vardi, 2000, Jeavons et al., 1998]. We will study these connections and their implications for infinite templates of constraint satisfaction problems.

A prominent concept in the early work on constraint satisfaction is the concept of *local consistency* [Freuder, 1978, Freuder, 1982]. The relationship between local and global consistency were studied in [Dechter, 1992, Cooper,

1989, Dechter and van Beek, 1997].

Definition 4.10. *Let Γ be a relational structure. An instance S of $\text{CSP}(\Gamma)$ is called k -consistent (with respect to Γ) iff for any subset V of nodes of S containing $k-1$ variables, and any variable $v \in S-V$, any solution to $S|_V$ can be extended to a solution to $S|_{V \cup \{v\}}$. If S is i -consistent for $i = 2, 3, \dots, k$, then it is said to be strongly k -consistent. If S is k -consistent for all k , then it is said to be globally consistent.*

If an instance of a constraint satisfaction problem is globally consistent, we can find a solution with back-track free search [Freuder, 1982]. To apply the notion of k -consistency computationally, there is the notion of *establishing strong k -consistency*. As already pointed out in [Kolaitis and Vardi, 2000], this notion has been defined rather informally in the literature. The idea is to *propagate* constraints in an appropriately extended language. We introduce the notion similarly as in [Kolaitis and Vardi, 2000], and then relate it to the notion of the canonical Datalog program.

Definition 4.11. *Let Γ and S be structures with relational signature τ . Establishing strong k -consistency for S (with respect to Γ) means that we can find expansions S' and Γ' of S and Γ that have the same signature τ' , such that*

- S' is strongly k -consistent with respect to Γ' .
- A mapping from the domain of S to the domain of Γ is a homomorphism from S to Γ if and only if it is a homomorphism from S' to Γ' .

For finite structures, we can use the canonical Datalog program of width $(k-1, k)$ to establish k -consistency. The expansion from Definition 4.11 is the set of all $k-1$ -ary relations on the domain of the template, and the fixed-point of the Datalog program is the structure S' that established k -consistency. For arbitrary infinite structures it might be the case that establishing i -consistency is computationally not feasible, since it would require to introduce an infinite number of distinct k -ary relations.

Proposition 4.12. *There exists a relational structure with a near-unanimity function where we can not establish 1-consistency with a finite signature.*

Proof. Consider the relational structure $P := (\mathbb{Q}; y \geq x - 1, y \geq x^2, y \leq 0)$. This structure is closed under the ternary median operation, since the median

preserves the *monotone* operations of translation and taking squares. We claim that we can in general not establish 1-consistency for an instance of CSP(P). Suppose Σ' is the expanded but still finite signature that we want to use to establish 1-consistency. Consider the range of values of a variable where we imposed a set of unary constraints from the signature Σ' . Since there are still only finitely many ways how this range can look like, there is a range that consists of a finite union of disjoint intervals I_1, \dots, I_n where n is maximal.

Using large degree polynomials we can find a primitive positive definition of a unary predicate that consists of a disjoint union of an arbitrarily large but finite number of intervals. For example, $x \in [-1, 1]$ can be defined by

$$\exists y_1, y_2. y_1 \geq x^2 \wedge y_2 \geq y_1 - 1 \wedge y_2 \leq 0.$$

The predicate $x \in [-b, a] \cup [a, b]$, where $0 < a < b$ are certain constants, can be defined by

$$\exists y_1, y_2, y_3, y_4, y_5. y_1 \geq x^2 \wedge y_2 \geq y_1 - 1 \wedge y_3 \geq y_2 - 1 \wedge y_4 \geq y_3^2 \wedge y_5 \geq y_4 - 1 \wedge y_5 \leq 0.$$

In the last example, the relation between x and y_5 is given by $y_5 \geq (x^2 - 2)^2 - 1$. The values $-b, -a, a, b$ are the roots of this polynomial. Imposing the constraint $y_5 \leq 0$ yields the two intervals of solutions for x . Using larger degree polynomials it should be clear now that we can obtain primitive positive definitions of predicates that consist of arbitrarily many intervals.

Let S be an instance to the constraint satisfaction problem and u a variable in S , such that S defines on u a predicate P that consists of a disjoint union of more than n intervals. No superstructure of S in the signature Σ' can be 1-consistent, since every predicate from Σ' that we can impose on u must be larger than P . Thus we always find assignments of u satisfying all these predicates and that can not be extended to a solution of S . \square

Near-unanimity operations. Recall that a polymorphism of Γ is called a near-unanimity operation iff it satisfies for all x, y

$$f(x, y, \dots, y) = f(y, x, y, \dots, y) = f(y, \dots, y, x) = y.$$

An example of an infinite structure with a ternary near-unanimity operation is the dense linear order on the rational numbers $(\mathbb{Q}; <)$, which admits the median operation, i.e., the operation which returns the medium size element of its three arguments.

Theorem 4.13. *Let Γ be a structure with relational signature τ . Then for $k \geq 2$ the following are equivalent:*

1. *If we can establish strong k -consistency of a (finite or infinite) τ -structure S , then this ensures global consistency.*
2. *Γ has a k -ary polymorphism that is a near-unanimity operation.*

The equivalence of one and two is proven in [Jeavons et al., 1998], and it is stated there that the proof works for arbitrary infinite structures Γ . If Γ is finite, it suffices to consider finite instances S in the first statement in Theorem 4.13. But if Γ is infinite, it is in fact necessary to consider countable τ -structures S . To illustrate this, we give an example of a relational structure Γ where every finite k -consistent structure has a homomorphism to Γ , but where we do not have a k -ary near-unanimity operation in $\text{Pol}(\Gamma)$.

The example is again the countable homogeneous graph $\not\triangleleft$ that is universal for the class of all finite triangle-free graphs, $\not\triangleleft := \text{Fl}(\text{Forb}(K_3))$; see Section 2.4. This graph is projective; see [Łuczak and Nešetřil, 2004]. In particular, there is no majority operation in $\text{Pol}(\not\triangleleft)$, which directly follows from Lemma 2.3 in [Larose and Tardif, 2001]: This lemma says that a (finite or infinite) graph without isolated vertices and admitting a majority operation is bipartite. Clearly, the graph $\not\triangleleft$ is not bipartite and does not contain isolated vertices.

Consistency and Datalog. For finite structures Γ , we say that a constraint satisfaction problem for Γ has *strict width* (k, l) , if the canonical Datalog program computes globally consistent expansions S' of instances S of $\text{CSP}(\Gamma)$. Since we generalized the notion of canonical Datalog programs to ω -categorical templates, we can define strict width also for ω -categorical templates, and obtain the following generalization of a result by [Feder and Vardi, 1999].

Theorem 4.14. *Let Γ be ω -categorical with a $k+1$ -ary near-unanimity operation, for $k \geq 2$. Then $\text{CSP}(\Gamma)$ has strict width k .*

Proof. Theorem 4.13 shows that if we can establish strong k -consistency of an instance S of $\text{CSP}(\Gamma)$, this ensures global consistency. For ω -categorical structures, the canonical $(k, k+1)$ -program computes a strong k -consistent expansion of S' , which will then be globally consistent. \square

In fact we could also use the same proof as in [Feder and Vardi, 1999], which was stated there only for finite Γ , but also works if we have the notion of canonical Datalog programs for ω -categorical structures, and thereby avoid the reference to Theorem 4.13.

4.7 Adding Constants to the Signature

Why can it be useful to look at the constraint satisfaction problem of a core of Γ , instead of the constraint satisfaction problem of Γ itself? We provide some arguments in this sections.

The endomorphism monoid. Homomorphisms from Γ to Γ are called *endomorphisms*. The set of all endomorphisms of a given structure is a monoid with respect to concatenation, i.e., it is a semigroup with an identity element (which is the identity mapping). We study the relations that are preserved by all endomorphisms.

Proposition 4.15. *Let Γ be an ω -categorical structure. Then a relation R has a positive definition if and only if R is preserved by surjective endomorphisms of Γ .*

Proof. Since R is in particular preserved by all automorphisms, the relation R has by ω -categoricity of Γ and Theorem 2.6 a definition by a first-order formula ϕ . The well-known preservation theorem of Lyndon asserts (see [Hodges, 1997]) that ϕ is equivalent to a positive formula modulo $\text{Th}(\Gamma)$ if and only if ϕ is preserved by surjective homomorphisms between models of $\text{Th}(\Gamma)$. Again, it is also known that we can restrict to preservation between *countable* models in Lyndon's theorem. Since any two countable models of $\text{Th}(\Gamma)$ are by ω -categoricity isomorphic, this is the case if and only if R is preserved by all surjective endomorphisms of Γ . \square

It is also well-known that one can combine the proof technique for Lyndon's preservation theorem and for the preservation theorem of Łoś-Tarski to show that a formula is preserved by all homomorphisms between models of a theory T if and only if the formula is over T equivalent to an existential positive formula. This is called the *homomorphism preservation theorem*. Thus we can deduce the following proposition in the same way as above.

Proposition 4.16. *Let Γ be an ω -categorical structure. Then a relation R has an existential positive definition in Γ if and only if R is preserved by all endomorphisms of Γ .*

For finite Γ , Proposition 4.16 goes back to [Krasner, 1945, Krasner, 1968]. Finite relational structures that contain all existential positive definable relations (all first-order definable relations) were called *weak Krasner-clones*

(*Krasner-clones*) in [Kaluzhnin and Pöschel, 1979] and are central notions there. Because of Proposition 4.16 we suggest to extend this terminology to ω -categorical structures.

Remark. As we just stated, Proposition 4.16 also holds for finite Γ . On the other hand, it is open whether the models of a first-order sentence ϕ are closed under homomorphisms if and only if the sentence is equivalent to an existential positive sentence. The corresponding finite version of the theorem of Los-Tarski is wrong [Tait, 1959]. For related results see [Gurevich, 1984, Ebbinghaus and Flum, 1999, Rosen, 2002].

Lemma 4.17. *Let Γ be an ω -categorical core. Then every existential formula is equivalent to an existential positive formula.*

Proof. Proposition 2.15 states that a relation has an existential definition in Γ if and only if it is preserved by embeddings between models of the first-order theory of Γ . Since Γ is a core, all endomorphisms are embeddings. The homomorphism preservation theorem, Theorem 4.16, implies that if a relation is preserved by all endomorphisms of an ω -categorical structure, it has an existential positive definition. \square

A structure Γ admits quantifier elimination, if every first-order formula has in Γ a quantifier-free definition. For an example, consider modules, and add all p.p.-definable relations to the signature. The theorem of Baur and Monk says that the resulting structure admits quantifier elimination (see e.g. [Hodges, 1997]). Cores behave similarly in this aspect. But we first need the following crucial lemma.

Theorem 4.18. *Let Γ be an ω -categorical core. If we expand Γ by all primitive positive definable relations, the resulting structure is homogeneous and admits quantifier elimination.*

Proof. Let ϕ be a first-order formula. It holds in general, that if we add all primitive formulae to a structure, its age has the amalgamation property, see Proposition 2.14. Thus the structure with all primitive relations is homogeneous, and admits quantifier elimination. In particular, ϕ can be written as a boolean combination of existential formulae. Since Γ is a core, Lemma 4.17 shows that ϕ can be written as a boolean combination of existential positive formulae. Since existential quantification and disjunction commute, this is the case if and only if every first-order formula can be written as a boolean

combination of p.p.-formulae. Thus also the expansion of Γ by all primitive positive definable relations has quantifier elimination. \square

In particular, the expanded core Γ' is model-complete, i.e., every embedding of Γ' into itself preserves all first-order formulae. Since this is equivalent to the property that every first-order formula is over Γ' equivalent to an existential formula (see e.g. Theorem [Hodges, 1997]), it follows from quantifier elimination of Γ' .

As we will see soon the following will be of importance for the theory of constraint satisfaction:

Corollary 4.19. *Let Γ be an ω -categorical core, and let R be an orbit of k -tuples in ω . Then R has a primitive positive definition in Γ .*

Proof. Let Γ' be the expansion of Γ by all primitive positive definable relations. By Theorem 4.18, Γ' admits quantifier elimination. Every quantifier-free definition φ of R in Γ' can be written in disjunctive normal. Let M_1 and M_2 be two monomials in the disjunctive normal form of φ . Then M_1 and M_2 have to denote the same relation, since R is the orbit of an k -tuple also in $\text{Aut}(\Gamma')$. Thus we can assume without loss of generality that φ is a conjunction of atomic formulae. Finally we replace all relation symbols in φ that are contained in the signature of Γ' , but not in the signature of Γ , by their primitive positive definition. The resulting formula is equivalent to a p.p.-definition of R in Γ . \square

It is one of the most often cited results in [Bulatov et al., 2003], that if Γ is a finite core, adding a singleton-relation does not increase the complexity of the constraint satisfaction problem. In this section we will show that the same holds for constraint satisfaction problems where the template is an ω -categorical core.

Theorem 4.20. *Let Γ be an ω -categorical core, and Γ' be the expansion of Γ by a unary singleton relations $P = \{c\}$. If $\text{CSP}(\Gamma)$ is tractable, then so is $\text{CSP}(\Gamma')$.*

Proof. Let S' be an instance of $\text{CSP}(\Gamma')$. The orbit O_c of c in the automorphism group of Γ is by Corollary 4.19 primitive positive definable in Γ . By Lemma 3.3 we can assume without loss of generality that Γ contains the relation O_c . Replace all occurrences of the relation P in S' by the relation O_c . Solve the resulting instance S of $\text{CSP}(\Gamma)$; by assumption this is possible

in polynomial time. If S is not satisfiable, then in particular S' could not have been satisfiable. On the other hand, if there is a homomorphism h from S to Γ , we also find a homomorphism from S' to Γ' : Since O_c is the orbit of vertex c , there is an automorphism a of Γ such that ha is a solution of the instance S' of $\text{CSP}(\Gamma')$. \square

Chapter 5

Graph Algorithms for Tree Constraints

In this chapter we describe efficient algorithms solving constraint satisfaction problems for ω -categorical tree-like structures. With these algorithms we can also solve several constraint satisfaction problems that we already encountered earlier, which are motivated by applications in phylogenetic analysis and computational linguistics. Compared to the algorithms for the tractable constraint satisfaction problems with finite templates, the algorithms of this chapter are of a new type.

5.1 Constraints in Computational Linguistics

Tree description languages became an important tool in computational linguistics over the last twenty years. Grammar formalisms have been proposed which derive logical descriptions of trees representing the syntax of a string [Marcus et al., 1983, Rogers and Shanker, 1992, Duchier and Thater, 1999]. Acceptance of a string is then equivalent to the satisfiability of the corresponding logical formula.

In computational semantics, the paradigm of *underspecification* aims at manipulating the partial description of tree-structured semantic representations of a sentence rather than at manipulating the representations themselves [Pinkal, 1996, Egg et al., 2001]. So the key issue in both, constraint based grammar and semantic formalisms, is to collect partial descriptions of trees and to *solve* them, i.e., to find a tree structure that satisfies all

constraints.

Cornell [Cornell, 1994] introduced a simple but powerful tree description language. It contains literals for *dominance*, *precedence* and equality between nodes, and disjunctive combinations of these. He also gave a saturation algorithm based on local propagations which turned out to be incomplete: there is a counterexample even for the restricted version of the problem Genealogy-consistency, which we presented in Section 3.4.

If nodes in a tree are interpreted as intervals over the real line, we can translate tree descriptions into a fragment of Allen’s interval algebra [Allen, 1983] where all the intervals are *laminar*, i.e., they are overlap-free. Dominance between nodes then corresponds to containment of intervals, and disjointness of nodes corresponds to disjointness of intervals. Allen’s full interval constraint logic has many applications in temporal reasoning but is NP-complete in its unrestricted form. There are fragments of this logic [Bürckert and Nebel, 1995, Jeavons et al., 2003] that are tractable. The non-overlap constraint together with disjointness and containment constraint is not one of them. Thus we can not use the known algorithms for interval constraints to solve Cornell’s problem.

In this chapter we present an efficient algorithm that tests satisfiability of a tree description of Cornell’s tree description language and directly constructs a solution to the problem instance. A predecessor of this algorithm, which applies to a slightly restricted language, appeared in [Bodirsky and Kutz, 2002]. It runs in time $O(nm)$, where n is the number of nodes and m the number of constraints in the input. The performance is achieved by a recursive strategy that works directly on the constraint graph, and avoids the saturation techniques described in Section 3.4. Subquadratic running time can be achieved using decremental graph algorithms for strong connectivity - this is explained in Section 5.8.

5.2 Tree Descriptions

We first fix some conventions for finite digraphs and trees. An *undirected* path in a directed graph may use arcs in any direction, ignoring their orientation. A *strongly (weakly) connected component* of $D = (V; E)$ is a maximal induced subgraph U of D with a directed (an undirected) path from a to b for any two vertices $a, b \in V$. $D|_S$ is the subgraph of D induced by the subset S of its vertices. A digraph $D = (V; E)$ is *transitive* if for all $(u, v), (v, w) \in E$

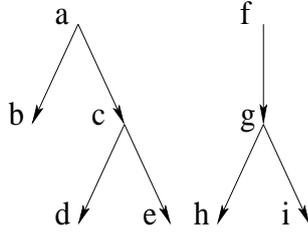


Figure 5.1: A forest, consisting of rooted and ordered trees.

we have $(u, w) \in E$. The *transitive closure* of a digraph $D = (V; E)$ is the unique transitive digraph $D' = (V; E')$ where E' is minimal and contains E .

The trees considered here are always rooted and thus we consider the edges as directed, pointing away from the root. The *height* of a tree is the length of the longest directed path in the tree. If we say that the tree is *ordered* we mean that the children of the vertices are linearly ordered, and we use the terms *left* and *right* to compare them. We call a set of such rooted trees a *forest*. If the rooted trees in a forest are ordered, and also the roots of the trees are linearly ordered, we call the forest *ordered*. In pictures we indicate this ordering by distinguishing between left and right. Figure 5.1 shows an example of such a forest containing two trees, rooted at the vertices a and f . In this picture, $a \prec f$ and $b \prec e$, and $a \triangleleft^+ e$, for example.

Note that in graph theory the notion of forest and tree are mostly used differently: a forest is an acyclic undirected graph, and a tree is a connected component of a forest. But this should cause no confusion here. The notation that we introduce now is as in [Backofen et al., 1995].

Usually the vertices of a tree are denoted by u, v, w . If u is the father of v in a tree, we write $u \triangleleft v$. We write $u \triangleleft^* v$ and say that u dominates v if u is an ancestor of v in the tree. We write $u \triangleleft^+ v$ for $u \triangleleft^* v$ and $u \neq v$. If for two vertices u and v neither $u \triangleleft^* v$ nor $v \triangleleft^* u$ we say that u and v are *disjoint*, and write $u \perp v$. In this case we distinguish two cases: either u *precedes* or *succeeds* v . A vertex u *precedes* (*succeeds*) a vertex v , and we write $u \prec v$ ($u \succ v$), if there is a common ancestor of u and v in the tree that has two children w_1 and w_2 , where $w_1 \triangleleft^* u$ and $w_2 \triangleleft^* v$, and u is to the left of v . We write $u \preceq v$ if $u \prec v$ or $u = v$.

We can use the notation for trees introduced above for forests. Note that these notations coincide with the notations on ω -categorical tree-like structures of Section 2.7, restricted to finite induced substructures. For every

pair u, v of vertices in an ordered tree or ordered forest, exactly one of the following cases holds:

$$u \triangleleft^+ v, \quad u \triangleleft^+ v, \quad u \prec v, \quad u \succ v, \quad u = v.$$

The following problem allows to partially describe the structure of a tree using arbitrary disjunctions of these five cases. It was introduced by Cornell in computational linguistics [Cornell, 1994]. To clearly distinguish between equality in this language and the common usage of the symbol ‘=’, we denote equality in this language by ‘ \equiv ’.

TREE-DESCRIPTION-CONSISTENCY

INSTANCE: A set of variables V ranged over by x, y, z , and a set C of binary constraints of the form $x\mathcal{R}y$, where $\mathcal{R} \subseteq \{^+\triangleright, \triangleleft^+, \prec, \succ, \equiv\}$.

QUESTION: Is there an ordered forest F together with a mapping $\alpha : V \rightarrow F$ from the variables to the vertices of the forest *satisfying* all constraints in C ? A constraint $x\mathcal{R}y$ is *satisfied* by (F, α) if $\alpha(x)R\alpha(y)$ holds in the forest for *some* relation $R \in \mathcal{R}$.

Matching the definitions we see that this problem corresponds to

$$\text{CSP}((\Lambda; \{\cup \mathcal{R} \mid \mathcal{R} \subseteq \{\triangleleft^+, \triangleleft^+, \prec, \succ, \equiv\}\})),$$

where the semilinear order Λ is introduced in Section 2.7 - and the signature of the template contains all boolean combinations of relations in Λ . A *solution* of this constraint satisfaction problem, as it was defined in Chapter 3, corresponds to the ordered forest of a yes-instance of the problem tree-description-consistency, and instances having such a solution are called *satisfiable*.

Figure 5.2 shows an unsatisfiable instance. Solid arcs stand for the constraint $\{\triangleleft^+\}$ and dashed arcs for the constraint $\{\prec\}$. In this chapter we develop an efficient algorithm for this problem.

5.3 An Algorithm for a Restricted Signature

We will first illustrate the algorithmic idea of our approach for a small fragment of pure dominance constraints. As we will see in Section 5.4, this fragment is already powerful enough to express the *Common-supertree* problem.

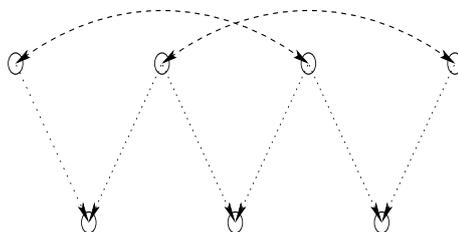


Figure 5.2: A constraint graph with no solution.

Thus we can solve this computational problem from phylogenetic analysis by the algorithm presented in this section – even without loss in asymptotic running time, compared to the best known algorithm described in [Henzinger et al., 1996]. In this fragment, we only allow the following two types of constraints:

$$x \{\triangleleft^+\} y \quad \text{and} \quad x \{\prec, \succ\} y$$

The first constraint is called (*strict*) *dominance*, and the second *disjointness*, and we use the shorthands $x \triangleleft^+ y$ and $x \perp y$ for these two cases. Thus, the constraint can be viewed as a graph $(V; \triangleleft^+, \perp)$ with two types of edges, namely directed edges \triangleleft^+ and undirected edges \perp . Because our algorithm is given in terms of graph theory, we will also call elements $x \in V$ *nodes*. Observe that the binary relations \triangleleft^+ and \perp are now defined on the vertices of a tree F and on the nodes of a constraint C . The reference will always be clear.

In pictures we draw this *constraint graph* using two types of arcs. For a dominance edge $x \triangleleft^+ y$ we draw a directed arc from x to y , for a disjointness edge $x \perp y$ we draw a dashed line without direction. Figure 5.2 for instance shows such a constraint.

Before we start explaining the algorithm, we would like to point out that also this problem is a constraint satisfaction problem in the sense of Chapter 3. The ω -categorical template is the reduct of the structure Λ with the two relations \triangleleft^+ and \perp only. The algorithm in this section solves the problem $\text{CSP}((\Lambda; \triangleleft^+, \perp))$. The age of the template of this constraint satisfaction problem is $\text{Forb}(\mathcal{N})$, where \mathcal{N} is the set of structures shown in Figure 5.3 (however, $\text{Forb}(\mathcal{N})$ does not have the amalgamation property; see Section 2.7).

Freeness. The key idea of the algorithm is the notion of *freeness* of nodes.

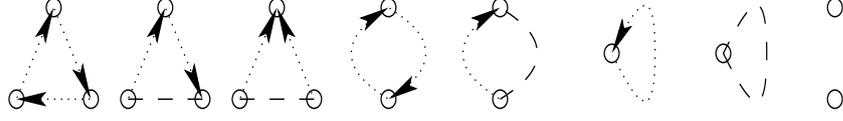


Figure 5.3: The set \mathcal{N} of forbidden induced subgraphs of $(\Lambda; \triangleleft^+, \perp)$.

Definition 5.1. A node x in a constraint C is free, if C has a solution (F, α) where F is a tree and $\alpha(x)$ is the root in F .

Hence, a constraint with a free node is by definition satisfiable. We use freeness algorithmically: Informally, the algorithm takes out a free node, decomposes the remaining constraint graph into weakly connected components, and recursively solves these parts. If there is no free node of a connected constraint graph, the constraint is not satisfiable. To prove this we need the following key lemma:

Lemma 5.2. Let C be a constraint with a weakly connected constraint graph $G = (V; \triangleleft^+, \perp)$, and let y and y' be variables in C . In every solution (F, α) of C there exists a variable $x \in V$ such that $\alpha(x) \triangleleft^* \alpha(y)$ and $\alpha(x) \triangleleft^* \alpha(y')$ in F .

Proof. Since the vertices y and y' are weakly connected there exists a chain of nodes (y_1, y_2, \dots, y_r) that starts at $y = y_1$, ends at $y' = y_r$, and is linked by edges $(y_i, y_{i+1}) \in \triangleleft^+ \cup (\triangleleft^+)^{-1}$. We prove by induction on r that for every solution (F, α) of C there exists an index $j \in \{0, \dots, r\}$ with $\alpha(y_j) \triangleleft^+ \alpha(y_0)$ and $\alpha(y_j) \triangleleft^+ \alpha(y_r)$ in F . If $r = 0$ or $r = 1$ then we can choose x to be either $\alpha(y_0)$ or $\alpha(y_r)$. Otherwise, we can apply the induction hypothesis to the chain (y_1, \dots, y_{r-1}) . Thus, there exists a j , $0 \leq j \leq r - 1$, such that $\alpha(y_j)$ is a common ancestor of $\alpha(y_1)$ and $\alpha(y_{r-1})$ in F . If $\alpha(y_{r-1}) \triangleleft^+ \alpha(y_r)$ then $\alpha(y_j)$ is also a common ancestor of $\alpha(y_1)$ and $\alpha(y_r)$, so we can choose $x = y_j$. Otherwise, $\alpha(y_r) \triangleleft^+ \alpha(y_{r-1})$ in F . Thus, both $\alpha(y_0)$ and $\alpha(y_r)$ are ancestors of y_{r-1} in F . Since F is a forest, it follows that either $\alpha(y_r) \triangleleft^+ \alpha(y_j)$ or $\alpha(y_j) \triangleleft^+ \alpha(y_r)$ in F . In the first case, we choose $x = y_r$, and in the second one $x = y_j$. \square

Proposition 5.3. A constraint C whose constraint graph has a weakly connected component without a free node does not have a solution.

Proof. Assume there is a solution (F, α) and consider the nodes of C whose interpretations are maximal in F with respect to \triangleleft^+ . If there is only one, say

x , then the subtree rooted at $\alpha(x)$ is also a solution of C , thus contradicting the assumption that x was not free. If there are at least two upmost nodes x, x' , then $\alpha(x)$ and $\alpha(x')$ are disjoint in F . Since G is weakly connected, we can apply Lemma 5.2. This contradicts that the interpretations of x and x' lie highest in F . \square

Free nodes have a simple equivalent graph characterization.

Proposition 5.4. *Let $G = (V; \triangleleft^+)$ be the constraint graph of a satisfiable constraint C . Then a node x of C is free if and only if*

- (P1) x has in-degree zero in G , and
- (P2) there is no constraint $x \perp y$ in C .

Proof. If there is another y s.t. $y \triangleleft^+ x$ the vertex x can not be topmost in any solution of C , and thus can not be free. If the vertex x is involved in a disjointness constraint it can also not be free, since the root of a tree is not disjoint to the other nodes in the tree.

Conversely, assume that the node x of a satisfiable constraint satisfies (P1) and (P2). The weakly connected components of $G|_{V-\{x\}}$ are also satisfiable. Then we have the following solution for C : introduce a tree node denoted by x , and let the solutions of the weakly connected components be the children of this node. The disjointness constraints between the different weakly connected components are thus satisfied by construction. It is clear from (P1) and (P2) all the constraints adjacent to x are also satisfied. \square

Figure 5.4 on the following page shows the entire algorithm for the restricted constraint language. If it detects a weakly connected component without a free node, we know by Proposition 5.3 that the constraint does not have a solution. Otherwise Proposition 5.4 guarantees that we can proceed by making the free node the root of our solution. The algorithm runs in time $O(nm)$, where n is the number of nodes and m the number of constraints. We will later see in Section 5.8 how to further improve the running time.

5.4 Phylogenetic Analysis

We quote from [Gusfield, 1997]: *The dominant view of the evolution of life is that all existing organisms are derived from some common ancestor and*

Solve($C = (V; \triangleleft^+, \perp)$):

```
    Compute the weakly connected components  $C_1, \dots, C_k$ 
    of the constraint graph  $(V; \triangleleft^+)$  of  $C$ 
for  $i = 1$  to  $k$  do
    if no node satisfying (P1) and (P2) exists
    then return “problem has no solution”
    else pick a node  $x \in C_i$  satisfying (P1) and (P2)
    create a new vertex  $r_i$ , and let  $\alpha[x] := r_i$ 
    add the roots of Solve( $C|_{C_i - \{x\}}$ ) as new children to  $r_i$ 
od
return the set of trees rooted at  $r_1, \dots, r_k$ 
```

Figure 5.4: The function Solve for a constraint with the literals \triangleleft^+ and \perp .

that a new species arises by a splitting of one population into two (or more) populations that do not cross-breed, rather than by a mixing of two populations into one. Therefore, the high level history of life is ideally organized and displayed as a rooted, directed tree. The extant species (and some of the extinct species) are represented at the leaves of the tree, each internal node represents a point when the history of two sets of species diverged (or represents a common ancestor of those species), . . . , and so the path from the root of the tree to each leaf represents the evolutionary history of the organisms represented there.

An evolutionary tree for a set of species is a rooted tree, where the leaves are bijectively labeled by the species from the set. Constructing evolutionary trees from biological data is a difficult problem for a variety of reasons, see [Gusfield, 1997]. Many approaches assume that the evolutionary tree is built from a set of taxa based on the comparison of a single protein or a single position in aligned protein sequences, but very often the resulting tree will be different depending on which particular protein or position is used. Now several trees, each from a different protein or position, must be built and be shown to be ‘generally consistent’ before the implied evolutionary history is considered reliable.

This motivates to consider the following problem.

COMMON-SUPERTREE

INSTANCE: A set S of evolutionary trees with common leaf set L .

QUESTION: Is there an evolutionary tree T on the leaf set L such that every tree in S is a *refinement* of T , i.e., can be obtained by a series of contractions of edges from T ?

We reduce this problem to a tree-description problem in the restricted language of 5.3. The variables of the tree-description are the vertices of the trees in S . Tree-edges translate to dominance constraints. Siblings x, y in a tree from S will be related via $x \perp y$ in the tree description. Clearly the resulting tree-description is of linear size in the representation size of S and has a solution if and only if there is a common supertree for the trees from S . Note that it is not possible to reduce the consistency problem for tree-descriptions to the problem Common-supertree in a similar straightforward way.

Often the biological data comes without an information about the root of the trees. In this case we analogously define the notion of refinements between (unrooted) trees. The corresponding Common-supertree problem already becomes hard if we consider trees with four leaf vertices only [Steel, 1992]. In this case, we can again formulate the problem as a constraint satisfaction problem: We called this problem *Quartet-consistency* in Section 1.3 in the introduction, and the corresponding template is described in Section 2.7.

5.5 Reduction to Four Base Literals

In this section we show that every binary first-order definable relation in $(\Lambda; \triangleleft^+, \prec)$ has a primitive positive definition in a restricted signature, where we only use the following set of four basic relations. Thus we can reduce all constraints from Cornell's tree description language to this simpler language.

$$\begin{aligned} x \{ \triangleleft^+, \equiv \} y, & \quad x \{ \prec, \equiv \} y, \\ x \{ \triangleleft^+, {}^+\triangleright, \prec, \succ \} y, & \quad x \{ \prec, \succ, \equiv \} y. \end{aligned} \tag{5.1}$$

The constraints $\{ {}^+\triangleright, \equiv \}$ and $\{ \succ, \equiv \}$ are simply the first and second constraint of (5.1) flipped, and the singletons $\{ \triangleleft^+ \}$, $\{ \prec \}$, and $\{ \equiv \}$ can be written as intersections of these. The two extremal sets $\{ {}^+\triangleright, \triangleleft^+, \prec, \succ, \equiv \}$ and \emptyset are not needed since the former imposes no restrictions on the tree and the latter is,

by definition, unsatisfiable. If we show how to express the constraints

$$\begin{aligned} x \{\triangleleft^+, \triangleright^+, \equiv\} y, & \quad x \{\triangleleft^+, \prec, \succ, \equiv\} y, \\ x \{\triangleleft^+, \prec, \equiv\} y, & \quad x \{\triangleleft^+, \succ, \equiv\} y \end{aligned} \quad (5.2)$$

with those in (5.1) we are finished, since the other constraints are again representable as intersections of constraints from (5.1) and (5.2). It is easy to check that the constraints in (5.2) have the following primitive positive definitions in the language with the relations from (5.1); we show two examples.

$$\begin{aligned} x \{\triangleleft^+, \triangleright^+, \equiv\} y & \Leftrightarrow \exists z. x \{\triangleleft^+, \equiv\} z \wedge y \{\triangleleft^+, \equiv\} z \\ x \{\triangleleft^+, \prec, \equiv\} y & \Leftrightarrow \exists z. x \{\triangleleft^+, \equiv\} z \wedge z \{\prec, \equiv\} y \end{aligned}$$

Thus we can express any 2-type in $(\Lambda; \triangleleft^+, \prec)$ with the four basic binary relations from (5.1). Our algorithm works on pure dominance constraints containing these four kinds of constraints only. We therefore introduce special names and notation:

- $x \{\triangleleft^+, \equiv\} y$: the *dominance* constraint $x \triangleleft^* y$,
- $x \{\prec, \equiv\} y$: the *precedence* constraint $x \preceq y$,
- $x \{\prec, \succ, \equiv\} y$: the *disjoint-or-equal* constraint $x \perp y$,
- $x \{\triangleleft^+, \triangleright^+, \prec, \succ\} y$: the *inequality* constraint $x \not\equiv y$.

Note that from now on, the notions of *dominance* and *precedence* relation are meant in the non-strict sense, i.e., they are reflexive relations.

5.6 Constraint Graphs and Freeness

We will give a graph theoretical characterisation of unsatisfiable pure dominance constraints. To this end, for a given constraint C we consider two different graphs:

- The *dominance graph* (V, D) , where

$$D = \{xy \mid x \triangleleft^* y \text{ or } x \perp y \text{ or } x \preceq y \text{ or } y \preceq x\}.$$

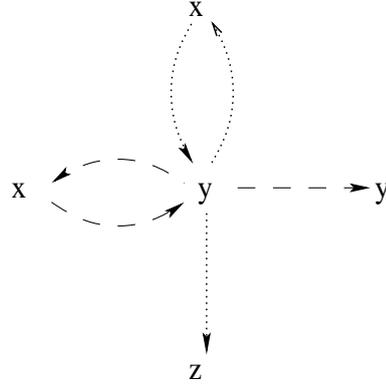


Figure 5.5: A tree description. The precedence constraints are dashed arcs, the dominance constraints dotted arcs. The strongly connected components of the dominance graph are $\{v_1, v_2, v_4, u\}$, and $\{v_3\}$. The strongly connected components of the precedence graph are $\{v_1, v_3, v_4, u\}$, and $\{v_2\}$.

- The *precedence graph* (V, P) , where

$$P = \{xy \mid x \preceq y \text{ or } x \triangleleft^* y \text{ or } y \triangleleft^* x\}.$$

Note that the precedence constraints are bidirectional in the dominance graph, and the dominance edges are bidirectional in the precedence graph. Moreover, the dominance graph contains edges corresponding to the disjoint-or-equal literals. See Figure 5.5 for an illustration.

Cycles in the dominance graph only admit solutions that map the entire cycle on the same node in a tree. The fact that the constraints can express equality on nodes must be reflected in our definition of freeness, which is now defined on *sets* of nodes.

Definition 5.5. *A set S of nodes of an instance C is called free, if there is a solution (F, α) where F is a tree and for all nodes $x \in S$ the vertex $\alpha(x)$ is the root of F .*

Again, if C is satisfiable, the following two straightforward necessary properties of free sets of nodes turn out to be sufficient for freeness.

Proposition 5.6. *Let C be a satisfiable constraint. Then S is a free set of nodes, if and only if all of the following holds:*

- (C1) *There is no edge $xy \in D$ in the dominance graph such that $x \notin S$ and $y \in S$.*

(C2) *There is no pair $x, y \in S$ such that $x \neq y$.*

Proof. It is easy to verify the only-if direction. For the other direction, let S be a set of nodes that satisfies conditions (C1) and (C2). Fix an arbitrary linear extension of the acyclic structure formed by the strongly connected subgraphs of the precedence graph without S .¹ We inductively assume that every such subgraph has a solution. Take these solutions and arrange their roots in the order of the linear extension as the children of a root vertex. This satisfies all constraints in the subgraphs and in S , and all dominance, precedence, and inequality constraints between the subgraphs, and between the subgraphs and S . Thus we have constructed a solution of C , where S denotes the root of the solution. \square

We now show that a satisfiable constraint with a strongly connected precedence graph always has such a free set.

Proposition 5.7. *Let C be a constraint whose constraint graph has a strongly connected precedence subgraph without a free set. Then C has no solution.*

Proof. Suppose (F, α) is a solution of C . We consider the nodes S in C that are minimal with respect to the linear order $\triangleleft^+ \cup \prec$ in F , i.e., we look at the set of nodes S that denote the topmost vertex u in the leftmost tree in F . The vertex u can not dominate all vertices in $\alpha(V)$, otherwise S is a free set. So let $y \in V$ be a variable such that $\alpha(y)$ is not dominated by u . Since the precedence constraint of C is strongly connected, there is a path x_1, x_2, \dots, x_k, y from some $x_1 \in S$ to $y \notin S$ in the precedence graph. The path $\alpha(x_1), \alpha(x_2), \dots, \alpha(x_k), \alpha(y)$ must eventually leave the subtree rooted at u , contradicting the minimality of S with respect to the relation $\triangleleft^+ \cup \prec$ in the forest F . \square

5.7 The Algorithm

In this section we present an algorithm that checks whether a tree description has a solution. If there exists a solution, it is explicitly constructed. Figure 5.6 shows the algorithm. The algorithm consists of two procedures,

¹By this we mean that there is an arc between two connected components A and B , iff there is an arc from a node in A to a node in B . If there was a cycle in this graph defined on the strongly connected components, all nodes on that cycle must belong to the same strongly connected component in the underlying graph, a contradiction.

Solve(C):

Compute the scc's of the precedence graph of C ,
 and let C_1, \dots, C_k be a linear extension of their acyclic structure
return the ordered forest consisting of the ordered trees
 Solve_con($C|_{C_i}$), for $1 \leq i \leq k$, in this order

Solve_con(C):

precond: The precedence graph of C is strongly connected.
if no set of nodes satisfying (C1) and (C2) exists
then return “problem has no solution”
else choose a set of nodes S satisfying (C1) and (C2)
create a new vertex r , and set $\alpha[S] = r$
add the ordered roots of the forest Solve($C|_{V-S}$) as children below r
return the ordered tree rooted at r

Figure 5.6: The function Solve for the full tree description language.

called Solve and Solve_con. Initially, for a given instance C , we call Solve(C). There the graph is decomposed into the strongly connected components with respect to the precedence graph. If the precedence graph of C is strongly connected, we can compute Solve_con(C). The algorithm contains a statement *choose*, that influences *which* free set of nodes is chosen, and which solution is constructed.

Finally we are left with the problem to efficiently find free sets of nodes, if they exist. Because of (C1) the set has to be a union of strongly connected components. The strongly connected components of a graph form an acyclic structure. It suffices to check whether there is an *initial* strongly connected component, i.e., a component S with no edge $uv \in D$ where $u \notin S$ and $v \in S$, which does not contain an inequality constraint - and this can be done in linear time in the input, using e.g. depth first search.

Proposition 5.8. *If we run algorithm Solve_con of Figure 5.6 on a tree description C it returns in time $O(n(n+m))$ either a forest that is together with the assignment α a solution of C - if it exists - or it returns “problem has no solution”.*

Proof. The procedure Solve_con requires that the precedence graph of the input is strongly connected. Thus, if it returns “problem has no solution”,

the constraint was unsatisfiable by Proposition 5.7 on page 110. Otherwise, Proposition 5.6 guarantees that we can construct a solution by taking out a free component. On each level of the recursive procedures we have to compute the strongly connected components either of the precedence or of the dominance graph, which can be done using depth first search in time $O(n + m)$. Since we take out at least one vertex in each call of `Solve_con`, the overall running time is in $O(n(n + m))$. \square

5.8 Subquadratic Running Time

During the performance of the algorithm, the connectivity structure of the constraint graph is computed several times. However, the input graphs for our connectivity computations are very similar: they are obtained from each other by vertex-deletions only. We want to find a faster implementation that avoids redundant computation. *Decremental*, or more generally, *dynamic connectivity* has been studied intensively in the last years. In fact, it is possible to maintain the weakly connected components of a graph on n vertices under a series of m edge-deletions in amortized and deterministic $O(m \log^2 n)$ time [Holm et al., 2001]. For *decremental strong connectivity* amortized subquadratic algorithms are known [Henzinger and King, 1995, Demetrescu and Italiano, 2000, Thorup, 1999].

We demonstrate how to use these dynamic algorithms for the restricted language introduced in Section 5.3 on page 102, and show how to maintain data structures for connectivity queries and detect the free nodes in sublinear time. We can thereby improve the overall running time to $O(m \log^2 n)$, using the decremental deterministic graph connectivity algorithm from [Holm et al., 2001]. Similarly it is possible to use decremental strong connectivity results for the general problem [Thorup, 1999].

Employing a decremental weak connectivity algorithm. In a *decremental connectivity problem*, we are considering a graph G over a fixed vertex set V , $|V| = n$. The graph G may be updated by deletions of edges. The updates might be interspersed with *connectivity queries*, asking whether two given vertices are connected in G . The connectivity problem reduces to the spanning tree problem [Sleator and Tarjan, 1983]: if we can maintain any spanning forest F for G at cost $O(t(n) \log n)$ per update, then we can answer connectivity queries in time $O(\log n / \log t(n))$. There is a deterministic algorithm for maintaining a spanning forest in a graph in amortized time

$O(\log^2 n)$ per update [Holm et al., 2001]. Connectivity queries are then answered in time $O(\log n / \log \log n)$.

If we want to improve the asymptotic running times of our original algorithm in Section 5.3 on page 102, we do not only have to maintain the connectivity structure, but also find a free node at each step of the recursion in sublinear time. This can be done by a heap, the *candidate heap*, which stores nodes that are root candidates. The priority of elements in the heap is the number of adjacent disjointness edges plus the in-degree in the dominance graph. If a node has priority 0 and hence is topmost in the heap, the node is free.

It is convenient to formulate the improved algorithm in an iterative fashion. In each iteration we have to take out a free node of *some* connected component, which is then linked under the corresponding leaf of the incrementally built solution tree. It is easy to see that this corresponding leaf can be found using the data structure of the decremental connectivity algorithm in [Holm et al., 2001].

With the free node, all adjacent constraints are deleted. This might cause updates both in the connectivity structure, and the priorities in the candidate heap. If a connected component got separated into two parts by an edge deletion, the decremental connectivity algorithm provides for a representative in the smaller of the two components. We check for every disjointness edge incident to this component, whether the other end of the edge lies in the rest of the graph. In this case the disjointness edge becomes redundant and we remove it, and the priorities of the incident nodes in the candidate heap is updated. Every time an edge is considered for deletion, the size of the involved component has at most half the size of the component where the edge was considered for the last time. Thus every disjointness edge is considered at most a logarithmic number of times. The overall amortized running time is thus dominated by the update operations of the decremental connectivity algorithm, and in $O(m \log^2 n)$, where m is the number of edges in the constraint graph.

Chapter 6

Graph Algorithms for Tree Constraints

In this section we present a polynomial time algorithm for the problem *Dominance-graph-solvability*, stated below. We later solve several substructure and constraint satisfaction problems for tree-like templates from Section 2.7 with the same algorithmic ideas. The problems are motivated by applications in computational linguistics [Althaus et al., 2003, Koller et al., 2000, Bodirsky et al., 2004, Niehren and Thater, 2003].

DOMINANCE-GRAPH-SOLVABILITY

INSTANCE: A *dominance graph* $G = (V; \triangleleft^+, \triangleleft)$ with two sets of directed edges, called *dominance* and *immediate dominance* edges, respectively. The immediate dominance edges in \triangleleft form a forest of height one.

QUESTION: Is there a forest on the vertices V containing \triangleleft , where the edges are directed away from the root for each tree in the forest, such that for every dominance edge in \triangleleft^+ there is a directed path in the forest?

If there exists such a forest for a dominance graph G we call it a *solution* of G , and say that G is *solvable*. In pictures we draw immediate dominance edges as solid arcs, and dominance edges as dotted arcs. See Figure 6.1 for an example of an unsolvable dominance graph.

The problem Dominance-graph-solvability is precisely the substructure problem $\text{wSub}(\Delta)$ of the ω -categorical semilinear order Δ defined in Section 2.7. Recall the definition of Δ : the domain is the set of all non-empty finite sequences $a = (q_1, q_2, \dots, q_n)$ of rational numbers. We say that a *dom-*

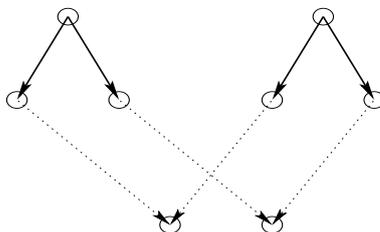


Figure 6.1: An unsolvable dominance graph.

dominates b in Δ , and write $a \triangleleft^+ b$, if either

- a is a proper initial subsequence of b , or
- $a = (p_0, \dots, p_{2n-1}, p_{2n})$ for $n \geq 0$, and $b = (p_0, \dots, p_{2n-1}, p'_{2n}, p_{2n+1}, \dots, p_m)$, where $p_{2n} < p'_{2n}$.

The *immediate dominance* relation $a \triangleleft b$ holds iff for every element c dominating b in Δ we have either $c \triangleleft^+ a$ or $c = a$. See also Figure 2.6 on page 42.

Proposition 6.1. *Let $G = (V; \triangleleft^+, \triangleleft)$ be a dominance graph. Then G is solvable if and only if G is a substructure of Δ .*

Proof. Let G be a solvable dominance graph, i.e., there exists a forest on V containing \triangleleft , where the edges are directed away from the root for each tree in the forest, such that for every edge in \triangleleft^+ there is a directed path in the forest. If we replace \triangleleft^+ by the transitive closure of this forest we obtain a dominance graph G' , and we can recursively embed G' in Δ .

To see this, let us furthermore assume that G is a forest and we want to find a solution that maps the root x of G to a given sequence a of odd length. If there are immediate dominance edges $x \triangleleft y_i$, then we assign the sequence (a, i) to y_i . Since \triangleleft forms a tree of height one, the maximal elements z_j^i below the nodes y_i with respect to \triangleleft^+ can not be linked to y_i by \triangleleft . We can thus map z_j^i to (a, i, j) . Then there might be other maximal elements z_i below x , that are not linked to x in \triangleleft . These nodes are assigned the sequence $(a, 0, i)$. Then we recursively construct solutions for all trees rooted at some vertex z_i or z_j^i . Initially, and if G is not a tree, we apply this procedure for all roots with an arbitrary set of odd sequences at disjoint positions in Δ .

Conversely, the transitive reduction of a substructure of Δ clearly is a forest, and the edges from \triangleleft form a forest of height one. Thus every substructure of Δ is a solvable dominance graph. \square

A polynomial time algorithm for a slightly restricted version of the problem was presented in [Althaus et al., 2001], and is based on a *duality theorem* (see Section 6.3.3). Dualities as a general concept for homomorphism problems were studied in [Nešetřil and Tardif, 2000]. However, for the problem introduced above we do not know of a duality theorem that leads to an efficient algorithm.

Results and outline of the chapter. We first present a novel algorithm for the problem Dominance-graph-solvability. The algorithm does not depend on a duality argument, but rather uses the notion of a free node that we already used in Chapter 6. It will be easy to adapt it to also deal with disjointness constraints, see Section 6.3.3. The new algorithm has better running times than the running time of the algorithm in [Althaus et al., 2003]. However, there is an improved version of [Althaus et al., 2003] that has a linear running time for the restricted version of the solvability problem where we can apply the duality argument [Thiel, 2004b]. If we are interested in constructing a solution for a given input, our algorithm is still faster. Moreover, subquadratic running time can be achieved by decremental graph biconnectivity algorithms, see Section 5.8. An evaluation of an implementation of the new algorithm shows that our algorithm also behaves well in practice.

We then introduce *dominance constraints*, which is a tree description language used in computational linguistics. Satisfiability of dominance constraints is NP-hard in general, but there is a tractable fragment called *normal dominance constraints* that suffices for many applications. Satisfiability of normal dominance constraints can be reduced to solvability of dominance graphs, and thus we can apply our algorithm. Finally we investigate larger fragments of dominance constraints and explore the border between tractable and intractable tree description languages.

6.1 Dominance Graphs and Solved Forms

The solutions of a dominance graph might contain a quadratic number of edges. For efficiency reasons, we will not explicitly construct solutions. Instead we introduce the notion of *solved forms* of a constraint, which are sparse representations of solutions that still allow to easily read off the solutions.

The general ideas behind solved forms apply to constraint satisfaction and substructure problems in general. Often we would like to work with

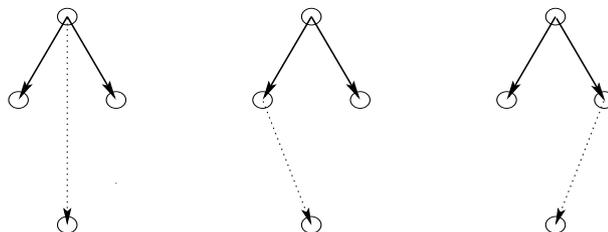


Figure 6.2: Three solved forms of which the left one is least specific.

concise representations of the solutions of a constraint satisfaction problem. A solved form of a constraint C is a constraint that contains at least the information of the constraint C . Moreover it should be easy to read off the solutions from the set of solved forms of a constraint. For efficiency, we are interested in solved forms that contain as few constraints as possible to represent a solution.

For dominance graphs we give a formal definition of solved forms that accomplishes these goals. The *reachability relation* R_G of a dominance graph $G = (V; \triangleleft^+, \triangleleft)$ is the reflexive transitive closure of $\triangleleft^+ \cup \triangleleft$. A dominance graph G is *less specific* than G' if:

- G and G' differ only in their sets of dominance edges, and
- the reachability relation is extended: $R_G \subseteq R_{G'}$

Definition 6.2. We call a dominance graph $(V; \triangleleft^+, \triangleleft)$ a solved form if it is a forest, i.e., a collection of rooted trees. A solved form of a dominance graph G is a solved form that is more specific than G . A minimal solved form (of G) is a solved form (of G) that is minimal with respect to specificity.

Fig. 6.2 shows a minimal solved form and two others that are more specific. Fig. 6.3 presents a dominance graph and its two minimal solved forms. In general, the solutions of a normal dominance constraint are partitioned by the minimal solved forms.

6.2 An Algorithm for Dominance Graphs

In this section we present a graph algorithm that computes the minimal solved forms of a dominance graph.

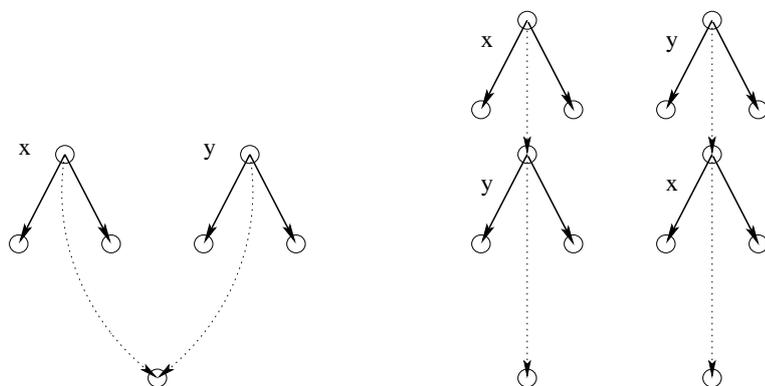


Figure 6.3: A dominance graph with two distinct minimal solved forms.

A dominance graph $G = (V; \triangleleft^+, \triangleleft)$ is *weakly connected* if for any two nodes x and y there is an undirected path from x to y in $\triangleleft^+ \uplus \triangleleft$. A weakly connected component (wcc) of G is a maximal weakly connected subgraph of G . Given a set of nodes $V' \subseteq V$, we write $G|_{V'}$ for the restriction of G to nodes in V' and edges in $V' \times V'$. The wccs of $G = (V; \triangleleft^+, \triangleleft)$ form a proper partition of V , \triangleleft^+ and \triangleleft .

Proposition 6.3. *A dominance graph is solvable if and only if all its weakly connected components are solvable.*

Proof. Let G be a dominance graph. If all wccs are solvable then we can choose some solved form for each component. The union of these solved forms is a solved form of G . Conversely, if G' is a solved form of G , and W are the nodes of a wcc of G , then $G'|_W$ is a solved form of $G|_W$. \square

We will now prove that a solved form of a weakly connected dominance graph is a rooted tree. This is equivalent to the following lemma, which states the key inductive property underlying the proofs of this chapter.

Lemma 6.4. *Let $G = (V, E)$ be a dominance graph with solved form G' . If y, y' are weakly connected nodes in G then there exists a node x that is a common ancestor of y and y' in G' .*

Proof. Analogously to Lemma 5.2 in Section 5.3 (by induction). \square

6.2.1 Freeness

This section prepares a satisfiability criterion for dominance graphs. As for the algorithms in Chapter 6, the idea of the efficient algorithm is based on the notion of *free nodes* in the constraint graph.

Definition 6.5. *A node x of a dominance graph G is called free in G if there exists a solution of G where x has in-degree zero.*

Proposition 6.6. *A weakly connected dominance graph without free nodes is unsolvable.*

Proof. If G is weakly connected and solvable then it has a solved form which is a tree (Lemma 6.4). The root of this tree is free for G . \square

The absence of solved forms can thus be proved by showing the absence of free nodes. Lemma 6.7 states two properties of free nodes, which can be used for this purpose.

Lemma 6.7. *A dominance graph $G = (V; \triangleleft^+, \triangleleft)$ with free node x satisfies:*

- (F1) *The node x has in-degree zero in G .*
- (F2) *No distinct nodes y, y' that are linked to x by tree edges in G are weakly connected in $G|_{V \setminus \{x\}}$.*

Proof. We assume that x is a free node of G and show that both conditions hold. (F1) A free node cannot have any incoming edge in G since it would have one in all its solved forms. (F2) Let y and y' be distinct nodes that are linked to x in G via tree edges. If y and y' are weakly connected in $G|_{V \setminus \{x\}}$ then some node z of $G|_{V \setminus \{x\}}$ must be a common ancestor of y and y' in all solved forms of G (Lemma 6.4), yet distinct from their mother x : therefore z is an ancestor of x in all solved forms of G , in contradiction with the assumption that x is free. \square

Proposition 6.6 and Lemma 6.7 imply the unsolvability of the dominance graph in Figure 6.1, where the nodes x_0 and y_0 violate (F2) while all other nodes violate (F1).

Sat(G):
forall weakly connected components $G' = (V'; \triangleleft^+, \triangleleft)$ of G :
 choose a node $x \in V'$ satisfying (F1) and (F2) in $G'|_{V'}$ **else fail**
 Let $y_1, \dots, y_n \in V'$ be all nodes s.t. $x \triangleleft y_i$.
 call Sat($G|_{V' - \{x, y_1, \dots, y_n\}}$)

Figure 6.4: Checking solvability of dominance graphs.

6.2.2 The Main Step

The idea to construct a solved form of a given weakly connected dominance graph $G = (V; \triangleleft^+, \triangleleft)$ is to remove a node satisfying (F1) and (F2) together with its neighborhood in \triangleleft , and to fail if there is no such node. We then decompose the remaining digraph of G into weakly connected components, and recursively solve these subgraphs. If successful it is easy to construct a solved form of the whole graph.

Lemma 6.8. *Let G be a non-empty dominance graph. Then the following properties are equivalent:*

1. *The procedure Sat(G) in Fig. 6.4 fails for some nondeterministic choice.*
2. *G is not solvable.*
3. *The procedure Sat(G) fails for all nondeterministic choices.*

Proof. (1 \Rightarrow 2) If Sat(G) fails for some nondeterministic choices, then G contains a weakly connected subgraph G' whose nodes all violate (F1) or (F2). By Lemma 6.7, G' has no free node and by Prop. 6.6 it is unsolvable. Since any graph which has an unsolvable subgraph is unsolvable, G must be unsolvable, too.

(2 \Rightarrow 3) Suppose Sat(G) is successful for some nondeterministic choices. We will prove by induction on the size of $G = (V; \triangleleft^+, \triangleleft)$ that G has a solved form. If G is not weakly connected, then G' is solvable by induction hypothesis for all wcc's G' of G , and hence G is solvable by Prop. 6.3. Otherwise G is weakly connected. Since the algorithm did not fail, there exists some node x in G that satisfies (F1) and (F2). By induction hypothesis, the wcc's $G|_{V - \{x, y_1, \dots, y_n\}}$ have solved forms G'_1, \dots, G'_k . For each G'_i , if G has a dominance edge from y_j to some node in G'_i , we attach G'_i with a dominance edge under y_j ; otherwise, we attach it with a dominance edge under x . Using

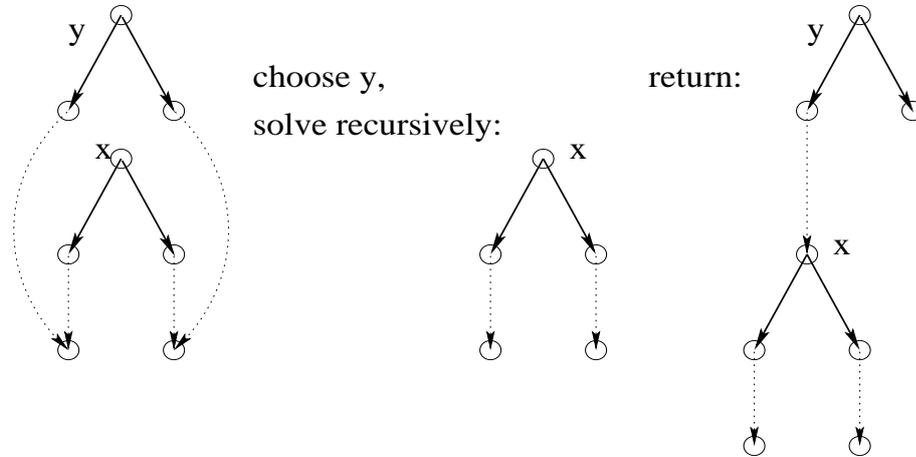


Figure 6.5: The algorithm in action.

(F1) and (F2) and the definition of dominance graphs we verify that (0) the resulting graph G' is a dominance graph, (1) G' is a tree that (2) contains all tree edges of G , and (3) refines $\triangleleft^+ \subseteq R_{G'}$. Thus G' is a solved form that is more specific than G .

(3 \Rightarrow 1) Clearly, if $\text{Sat}(G)$ fails for all nondeterministic choices then it also fails for some nondeterministic choice. \square

The algorithm contains a nondeterministic choice statement. We will now show that if we consider all different possibilities at this point all minimal solved forms of a given dominance graph are enumerated. The enumeration algorithm is given in Figure 6.6.

An example for a run of the algorithm is given in Figure 6.5. Procedure `Solve` applied to the graph on the left first computes the weakly connected components – there is only one. Then the procedure `Solve_con` is applied to this component. The only node satisfying (F1) and (F2) in the graph on the left is x_3 . Note in particular that x_0 violates (F2). The algorithm thus removes the tree fragment of x_3 , i.e., nodes x_3, x_4, x_5 . The resulting graph, drawn in the middle of Figure 6.5, has only one connected component, and we again apply `Solve`. After several steps, a single solved form is returned (it equals the graph in the middle again). For the final result, the algorithm adds the previously removed tree fragment of x_0 on top of this graph. A dominance edge from x_4 to x_0 is inserted since x_4 dominates x_6 in G , and since x_6 belongs to the same component as x_0 in the middle.

Theorem 6.9. *The algorithm Solve of Figure 6.6 applied to a dominance graph G produces all and only the minimal solved forms of G .*

Proof. The algorithm reflects the construction of a solved form in the proof of Lemma 6.8. By its recursive structure, the dominance graphs returned by Solve_con(G) are rooted trees, and therefore Solve(G) is a forest. It is also easy to see that it is more specific than G and thus the algorithm only produces solved forms of G . Different choices of free nodes lead to different solved forms, and thus each solved form is produced at most once.

We only have to prove that the algorithm produces *all minimal solved forms of G* . Since at each recursive step in Solve_con we process a weakly-connected graph, by Lemma 6.4 the algorithm must return a solved form that is a tree; therefore some node x must be chosen to be top-most. If we want to make this node the root of our solved form, we *must* insert the dominance edges added in the last or next to last line of the algorithm. Therefore, we only enumerate *minimal* solved forms of G . The node x is by definition free, and thus the algorithm will eventually choose it. Thus we enumerate *all* minimal solved forms of G . \square

Corollary 6.10. *For a node of a solvable dominance graph, (F1) and (F2) are necessary and sufficient conditions for freeness.*

Proof. Free nodes satisfy (F1) and (F2) by Lemma 6.7. Conversely, let x be a node of a dominance graph G satisfying (F1) and (F2). Then algorithm Solve(G) calls Solve_con(G) for the weakly connected component G' of x , and in Solve_con(G) the node x can be chosen to construct a solved form of G' . It will never fail since G is solvable (Lemma 6.8) and finally produces a solved form of G' (Theorem 6.9), which is a tree rooted at x . \square

In Section 6.2.3 we describe how to compute all nodes satisfying (F1) and (F2) in time $O(n + m)$ where n is the number of nodes and m the number of edges in a dominance graph G . This will prove:

Theorem 6.11. *The overall running time of the enumeration algorithm in Figure 6.6 is in $O(n \cdot (n + m))$ per solved form for solvable dominance graphs; otherwise it returns “problem has no solution” in time $O(n \cdot (n + m))$.*

Solve(G):

- Let G_1, \dots, G_k be the wcc's of $G = (V; \triangleleft^+, \triangleleft)$
- Let $(V_i; \triangleleft_i^*, \triangleleft_i)$ be the result of Solve_con(G_i)
- return** $(V; \cup_{i=1}^k \triangleleft_i^*, \triangleleft)$

Solve_con(G):

- precond:** $G = (V; \triangleleft^+, \triangleleft)$ is weakly connected
- choose** a node x satisfying (F1) and (F2) in G
- else return** “problem has no solution”
- Let y_1, \dots, y_n be all nodes s.t. $x \triangleleft y_i$
- Let G_1, \dots, G_k be the weakly connected components of $G|_{V-\{x, y_1, \dots, y_n\}}$
- Let $(W_j; \triangleleft_j^*, \triangleleft_j)$ be the result of Solve_con(G_j), and $x_j \in W_j$ its root
- return** $(V; \cup_{j=1}^k \triangleleft_j^* \cup \triangleleft_{k+1}^* \cup \triangleleft_{k+2}^*, \triangleleft)$, where
 - $\triangleleft_{k+1}^* = \{(y_i, x_j) \mid \exists x' : (y_i, x') \in \triangleleft^+ \wedge x' \in W_j\}$,
 - $\triangleleft_{k+2}^* = \{(x, x_j) \mid \neg \exists x' : (y_i, x') \in \triangleleft^+ \wedge x' \in W_j\}$

Figure 6.6: Algorithm enumerating the solved forms of a dominance graph.

6.2.3 Testing Freeness Conditions

We now want to efficiently compute the set of nodes that satisfy the freeness conditions (F1) and (F2). We have to check for each node u with in-degree zero whether it has a pair of children (linked to u by outgoing tree edges) that are weakly connected in the dominance graph of the constraint without u . Thus the naive way would be to compute for all these nodes u the weakly-connected components of the constraint graph without node u . This takes quadratic time per node, and thus cubic time to compute *all* free nodes.

We now show how to compute the set of free nodes in linear time. A (*vertex-*) *biconnected component* of a graph is a maximal subgraph that remains connected when one of its nodes is deleted. Biconnected components form a proper partition of the graph edges.

Proposition 6.12. *A node x in a dominance graph G satisfies (F2) if and only if all different tree edges (x, y) and (x, y') of G lie in different biconnected components of G .*

Proof. If x does not have any children there is nothing to show. So first assume that y and y' are in the same weakly connected component of $G|_{V_G \setminus \{x\}}$. Then there is a path between y and y' not using x and thus the edges (x, y) and (x, y') must be in the same biconnected component. Conversely, assume

that (x, y) and (x, y') are in the same biconnected component. Then there is a path from y to y' not using x and therefore there still must be a path from y to y' after removing x . \square

A linear time algorithm that given a graph computes a mapping from each edge to a unique element of its biconnected component can be found in textbooks that cover graph algorithms. To compute the set of free nodes, we use a control bit for each biconnected component. When processing a candidate x , we can use the bit to check for each edge $x \triangleleft y$ whether we already encountered another edge $x \triangleleft y'$ in the same biconnected component. Since we spend only constant time on every tree edge, we have a linear time procedure that finds the free roots.

In each step, the algorithm computes the weakly connected components and biconnected components from scratch again. It is possible to further improve the running time and to avoid these redundant computations using dynamic graph connectivity algorithms, that allow to answer weak connectivity and biconnectivity queries in amortized sublinear time [Holm et al., 2001].

6.3 Normal Dominance Constraints

Dominance constraints were introduced in computational linguistics [Marcus et al., 1983, Backofen et al., 1995], and they are logical descriptions of trees that can talk about the parent and the ancestor relations between the nodes of a tree. They have numerous applications e.g. in underspecified semantics [Egg et al., 2001, Copestake et al., 1999, Bos, 1996], underspecified discourse [Gardent and Webber, 1998], and parsing with tree adjoining grammar [Rogers and Vijay-Shanker, 1994]. Satisfiability of dominance constraints was proved to be NP-complete in [Koller et al., 1998]. This shed doubts on the feasibility of dominance based applications. The doubts were removed by [Althaus et al., 2003], who distinguished the language of *normal dominance constraints* which suffices for many applications and has a polynomial time satisfiability problem.

One relevant problem for normal dominance constraints is to enumerate solved forms, i.e., all trees satisfying a constraint. [Althaus et al., 2003] presented an enumeration algorithm whose running time is $O(n^4)$ per solved form. This algorithm relies on an efficient satisfiability test. Thiel [Thiel, 2004b] improved this result to $O(n^3)$ by faster satisfiability testing. We show

here that this problem can be solved by the algorithm described in the preceding sections. We can enumerate all solved forms of a normal dominance constraint in $O(n^2)$ per solved form, and thereby improve on the best previously known algorithm in efficiency. If we use decremental graph biconnectivity algorithms we can achieve subquadratic running time, as mentioned in Section 6.2.3. Our algorithm even applies to the extended language of *weakly normal dominance constraints* introduced in [Bodirsky et al., 2004]. This improves the applicability of dominance constraints in the area of natural language semantics [Niehren and Thater, 2003].

6.3.1 Preliminaries

We start with a brief exposition of dominance constraints [Marcus et al., 1983, Backofen et al., 1995], recall the notion of normality [Althaus et al., 2003], and then introduce weak normality.

Let f, g range over the elements of some finite signature Σ of function symbols with fixed arities and a, b over constants, i.e., function symbols of arity 0. A *constructor tree* over this signature is a ground term τ constructed from the function symbols, as for instance $f(g(h, i))$. We identify ground terms with trees that are rooted, edge-ordered, and node-labeled. See Figure 6.7 for a graphical representation of the tree $f(g(h, i))$. Clearly, if Σ does not contain a constant symbol, there is no such constructor tree. If Σ only contains function symbols of arity zero and one, all these trees do not branch. Thus we assume that Σ contains at least one constant and at least one function symbol of arity larger than one.

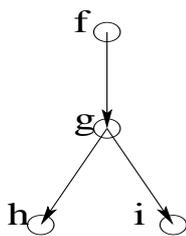


Figure 6.7: $f(g(h, i))$

Dominance constraints are logic formulae that describe constructor trees. They can talk about the mother-child relation \triangleleft between the nodes of a tree, *dominance* \triangleleft^* which is the reflexive ancestor relation, and inequality \neq . Let x, y, z range over an infinite set of *node variables*. A *dominance constraint* ϕ

is then a conjunction of literals:

$$\phi ::= x:f(y_1, \dots, y_n) \mid x \triangleleft^* y \mid x \neq y \mid \phi' \wedge \phi''$$

A solution of a dominance constraint consists of a constructor tree τ and a variable assignment α to the nodes of τ . A *labeling literal* $x:f(y_1, \dots, y_n)$ is satisfied if $\alpha(x)$ is labeled by f in τ and has the children $\alpha(y_1), \dots, \alpha(y_n)$ in that order. A *dominance literal* $x \triangleleft^* y$ requires that $\alpha(x)$ dominates $\alpha(y)$ in τ . An *inequality literal* $x \neq y$ requires $\alpha(x)$ and $\alpha(y)$ to be distinct.

The constraint of Figure 6.8 requires that the node values of x_1 and x_2 are sisters, and ancestors of the node value of y . This is clearly impossible in a tree since the subtrees rooted in two sister nodes, the values of x_1 and x_2 , are necessarily disjoint and therefore cannot share a common node y .

Definition 6.13. A dominance constraint ϕ is *normal* if it satisfies:

1. (a) each variable of ϕ occurs at most once in the labeling literals of ϕ .
A variable y_i is a *hole* of ϕ whenever it occurs at the right of some labeling literal $x:f(y_1, \dots, y_n)$ of ϕ ; else it is a *head* of ϕ .
- (b) each variable of ϕ occurs at least once in the labeling literals of ϕ .
2. if x and y are distinct heads in ϕ , $x \neq y$ occurs in ϕ .
3. (a) if $x \triangleleft^* y$ occurs in ϕ , y is a head in ϕ .
- (b) if $x \triangleleft^* y$ occurs in ϕ , x is a hole in ϕ .

A dominance constraint is *weakly normal* if it satisfies all above properties except for 1.(b) and 3.(b).

Dropping 3.(b) allows head-to-head dominances which improves the applicability in natural language processing discussed in [Bodirsky et al., 2004,

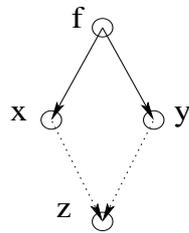


Figure 6.8: Unsatisfiable constraint: $u:f(x, y) \wedge x \triangleleft^* z \wedge y \triangleleft^* z$

Niehren and Thater, 2003]. Dropping condition 1.(b) adds convenience. The unsatisfiable constraint in Figure 6.8, for instance, has holes x_1 and x_2 and heads x and y . Extended by the entailed inequality $x \neq y$, this constraint becomes weakly normal, but not normal as head y violates condition 1.(b).

Compactification. Condition 1.(a) could be relaxed in order to support nested labelings, such as $x_1:f(y_1, x_2) \wedge x_2:g(y_2, y_3)$ modeling the *nested* term $f(y_1, g(y_2, y_3))$. We do not do so here, as nested labeling can always be eliminated by *compactification*: The idea is to replace $x_1:f(y_1, x_2) \wedge x_2:g(y_2, y_3)$ by $x_1:h(y_1, y_2, y_3)$ where h is a new function symbol. In general, compactification preserves (un)solvability.

We call ϕ *well-formed* if it does not contain any subconstraint of the form $x:a \wedge x \triangleleft^* y$. In the remainder of this chapter we assume well-formedness. This does not restrict generality as every constraint can be made well-formed in a linear time: suppose that ϕ contains a subconstraint $x:a \wedge x \triangleleft^* y$. If x is the same variable as y then we can remove $x \triangleleft^* y$. Otherwise, x and y are distinct heads of ϕ so that $x \neq y$ belongs to ϕ ; this is unsatisfiable since x must denote a leaf (which does not properly dominate any other node).

6.3.2 Reduction to Dominance Graphs

To every weakly normal dominance constraint we assign a unique dominance graph $G = (V; \triangleleft, \triangleleft^+)$. The nodes of the graph of ϕ are the variables of the constraint ϕ . Labeling literals $x:f(x_1, \dots, x_n)$ of ϕ contribute *immediate dominance edges* $x \triangleleft x_i$, and dominance literals contribute *dominance edges* $x \triangleleft^+ y$. Weak normality (Definition 6.13) ensures that the immediate dominance edges form indeed a forest of height one.

Proposition 6.14. *A well-formed weakly normal dominance constraint is satisfiable if and only if its dominance graph is solvable.*

Proof. From a solution of a dominance constraint, one can easily read off a solved form for the corresponding dominance graph. It is sufficient to ignore all nodes of solutions that are not values of variables (together with all adjacent edges), and to replace vertices that correspond to both a head and a tail variable by two vertices $v_1 \triangleleft^+ v_2$, where the tail variable denotes v_1 and the head variable denotes v_2 . Vice versa, one can construct a solution for a solved form of a weakly normal dominance constraint inductively top down, as long as the constraint is well-formed, which is always possible since

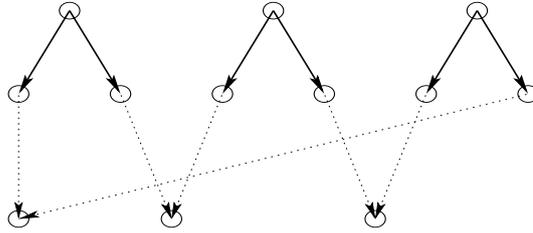


Figure 6.9: A harmful cycle.

Σ contains a function symbol of arity larger than one and a constant symbol; we need this to construct the tree such that it satisfies multiple outgoing dominance edges. \square

6.3.3 A Duality Theorem

If we look at the dominance graphs of *normal dominance constraints*, rather than the more powerful weakly normal dominance constraints, there is a beautiful characterization of unsatisfiability, based on a duality theorem. This duality theorem underlies the first polynomial time algorithm for normal dominance constraints [Althaus et al., 2003]. We present this approach here to show that it fails for weakly normal dominance constraints.

Let τ be a relational signature. In the same spirit as in [Nešetřil and Tardif, 2000] we say that a *homomorphism-duality* is a pair (\mathcal{C}, A) , where A is a τ -structure and \mathcal{C} a class of τ -structures, such that every τ -structure B homomorphically maps to A if and only if there is no element $C \in \mathcal{C}$ such that C homomorphically maps to B . Analogously, we can define a *substructure-duality*. Consider for example the countable homogeneous complete bipartite graph $K_2[I_\infty]$, and the set \mathcal{C} of all cycles of odd length. Then $(\mathcal{C}, K_2[I_\infty])$ is a substructure-duality.

Now we present a class of structures \mathcal{C} with the signature $\{\triangleleft, \triangleleft^+\}$ such that (\mathcal{C}, Δ) is a substructure-duality. The members of \mathcal{C} were called *hypernormal cycles* in [Althaus et al., 2003], and *harmful cycles* in [Thiel, 2004a]. By a *cycle* of a graph $G = (V, E)$ we mean a connected two-regular subgraph.

Definition 6.15. *An undirected cycle of a dominance graph G is called harmful iff it does not contain nodes x, y, z such that $y \triangleleft^+ x$ and $y \triangleleft^+ z$.*

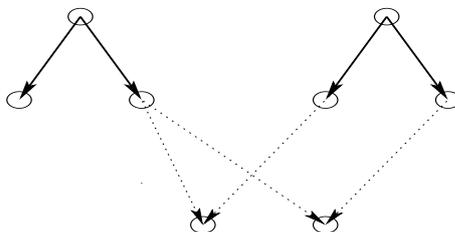


Figure 6.10: A harmless example: a graph without a harmful cycle.

See Figure 6.9 for an example of a harmful cycle, and Figure 6.10 for a solvable example with a graph that contains a cycle, but no harmful cycle.

Proposition 6.16. *The dominance graph of a normal dominance constraint is contained in Δ if and only if it does not contain a harmful cycle.*

The algorithm from [Althaus et al., 2003] is based on a procedure to test whether a given graph contains a harmful cycle. A linear time procedure for this test can be found in [Thiel, 2004b, Thiel, 2004a].

To demonstrate the difference of the algorithms we consider a natural extension of our constraint language by means of *disjointness literals*. These are part of various tree description languages [Egg et al., 2001, Backofen et al., 1995, Cornell, 1994], and they were part of the input of the algorithms in Chapter 6. To recall, a *disjointness constraint* $x \perp y$ is satisfied by a solution (F, α) if neither $\alpha(x)$ dominates $\alpha(y)$ nor $\alpha(y)$ dominates $\alpha(x)$ in the forest F . We draw disjointness constraints in the constraint graph with dashed two-headed arcs, as in Figure 5.2 on page 103. This graph is unsatisfiable, since it does not contain a free node.

Our algorithm can easily be adapted to deal with this extended constraint language. We now have the following additional property of a free node x :

$$(F3) \quad x \text{ has no incident disjointness edge: } \forall y \in V : (x \perp y) \notin \Phi$$

The proofs can readily be adapted to show that, for satisfiable constraints, (F1), (F2) and (F3) are again necessary and sufficient conditions for freeness.

In contrast, we do not know of any way to adapt the approach of [Althaus et al., 2003] to deal also with disjointness constraint. To test unsatisfiability via the presence of certain kinds of ‘harmful cycles’ in the constraint graph does not naturally extend to graphs also containing disjointness edges. Observe that after deleting any edge in Figure 5.2 the constraint is satisfiable.

6.3 NORMAL DOMINANCE CONSTRAINTS

Problem	[Althaus et al., 2003]	[Thiel, 2004a]	New	n	m	N
Chain 2	0.128	0.055	0.0435	8	8	2
Chain 3	0.2136	0.0818	0.0422	12	13	5
Chain 4	0.292857	0.104286	0.040714	16	18	14
Chain 5	0.369524	0.131429	0.04	20	23	42
Chain 6	0.435379	0.149545	0.038409	24	28	132
Chain 7	0.498834	0.164336	0.036596	28	33	429
Chain 8	0.561818	0.18042	0.036363	32	38	1430
Chain 9	0.641094	0.214109	0.034348	36	43	4862
Chain 10	0.713027	0.234996	0.030304	40	48	16796
Chain 11	0.77379	0.249430	0.027268	44	53	58786
Chain 12	0.837788	0.263831	0.023364	48	58	208012
Chain 13	0.896218	0.277453	0.020016	52	63	742900
Chain 14	0.944325	0.292304	0.017992	56	68	2674440
Chain 15		0.307242	0.015540	60	73	9694845
H&S	0.342262	0.117262	0.03869	19	20	168

Table 6.1: Average time in milliseconds per solved form, for constraint graphs with n nodes, m edges, and N solved forms

It seems that harmful cycles and disjointness constraints do not go together well. It already seems impossible to adapt the duality for the dominance graphs of weakly normal dominance constraints. This observation and an illustrating example can be found in [Thiel, 2004a].

6.3.4 Implementation and Evaluation

We implemented the algorithm (without using the mentioned decremental graph connectivity algorithms) and compared it to the best known previous implementations of [Althaus et al., 2003] and of [Thiel, 2004a]. All implementations are done with C++/LEDA and were run on a Pentium-IV 2GHz. The software and the instances for experimentation are available at <http://ps.uni-sb.de/~smiele/dom-solving>.

Table 6.1 reports on comparative benchmarks for a collection of prototypical examples in the application to scope underspecification. Chain k is a dominance constraint that models a natural language sentence with k quantifiers. Furthermore, we consider a famous example of Hobbs and Shieber (H&S).

6.4 Larger Tractable Fragments

In this section we explore how the definition of *normality* of dominance constraints can be relaxed such that the satisfiability problem remains tractable.

We did not give a formulation of normal dominance constraints as a homomorphism problem. The reason is Condition 2 of Definition 6.13, where we require that distinct variables denote distinct nodes in the template in normal dominance constraints. This is why we can only formulate the problem as a substructure problem in the form $\text{wSub}(\Gamma)$ for an appropriate relational structure Γ . It will turn out that we can use the structure Δ that we defined in Section 2.7.

In this section we drop this distinctness requirement in the definition of normality. The resulting consistency problem can then be formulated as a constraint satisfaction problem $\text{CSP}(\Gamma)$ for an appropriately chosen relational structure Γ . Now several variables can denote the same vertex in Γ . To apply the notion of freeness, we have to again consider *sets* of free nodes, as we did in Chapter 6. We show that the problem is computationally equivalent to a certain *surjective homomorphism problem*. We show that under certain assumptions on the signature Σ of the labeling constraints this problem becomes tractable, and under other assumptions intractable.

If we drop Conditions 1.(b), 2, and 3.(b) in the definition of normality (Definition 6.13 on page 127), the only remaining conditions on dominance constraints are:

- each variable of ϕ occurs at most once in the labeling literals of ϕ . A variable y_i is a *hole* of ϕ whenever it occurs at the right of some labeling literal $x:f(y_1, \dots, y_n)$ of ϕ ; else it is a *head* of ϕ ;
- if $x \triangleleft^* y$ occurs in ϕ , y is a head in ϕ .

In this case the satisfiability problem for dominance constraints translates to the following computational problem.

Σ -DOMINANCE-CONSTRAINT-SATISFIABILITY

INSTANCE: A finite set V variables, and a set of *labeling constraints* of the form $x_0 : f(x_1, \dots, x_{\text{ar}(f)})$, where $x_0, \dots, x_{\text{ar}(f)} \in V$ and $f \in \Sigma$. We call $x_1, \dots, x_{\text{ar}(f)}$ the *holes* of the labeling constraint. We are also given a set of inequality constraints $x_1 \neq x_2$, for $x_1, x_2 \in V$, and a set of dominance

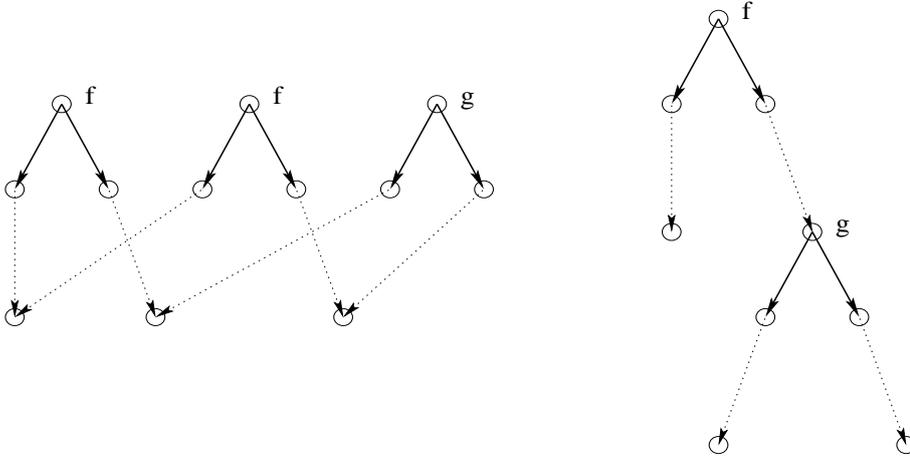


Figure 6.11: An $\{f, g\}$ -dominance graph (left side) and its solution (right side). Both f -labeled vertices on the left are mapped to the root on the right.

constraints $x_1 \triangleleft^* x_2$. For the dominance constraints we require that x_2 is not a hole of some labeling constraint.

QUESTION: Can we find a mapping α from V to a rooted forest F in which some vertices are labeled by at most one label from Σ , such that

- (i) for every constraint $x_1 \triangleleft^* x_2$ there is a directed path in F from $\alpha(x_1)$ to $\alpha(x_2)$,
- (ii) for every inequality constraint $x_1 \neq x_2$ we have $\alpha(x_1) \neq \alpha(x_2)$, and
- (iii) for every labeling constraint $x_0 : f(x_1, \dots, x_{\text{ar}(f)})$ the vertices $\alpha(x_1), \dots, \alpha(x_{\text{ar}(f)})$ are the distinct children of the f -labeled vertex $\alpha(x_0)$, in this order?

For every Σ this problem is in CSP*, and can be described as CSP(Γ) for an appropriate ω -categorical template Γ – see the techniques to construct tree-like structures in Section 2.7. Instances C of this constraint satisfaction problem we call Σ -dominance-constraints. As before, the pair (F, α) described in the problem definition is called a *solution*, and a Σ -dominance-graph having a solution is called *satisfiable*.

Consider Figure 6.11. We write the labels of the nodes close to the node of the shown $\{f, g\}$ -dominance-graph. Note that there is no free node in the sense of Section 6.2.1, and that it contains a harmful cycle in the sense of Section 6.3.3. But still it is satisfiable since we can map both f -labeled vertices to the root of the tree shown on the right hand side. Figure 6.12 shows an unsatisfiable constraint.

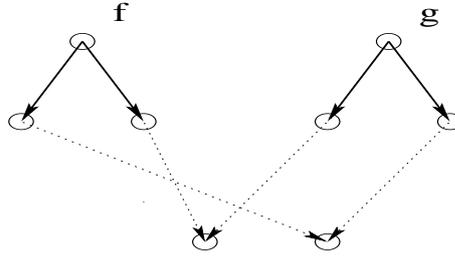


Figure 6.12: An unsatisfiable $\{f, g\}$ -dominance graph.

Free sets of nodes. We again use the concept of freeness.

Definition 6.17. A set of nodes S of a Σ -dominance-constraint is called free, if there exists a solution (F, α) where F is a tree and α maps all nodes in S to the root of F .

As in Section 6.2.1 we see that a weakly connected Σ -dominance-graph without a free set of nodes is unsatisfiable. To characterize freeness it will be convenient to use the notion of the dominance graph G of a Σ -dominance constraint C . This is defined exactly as in Section 6.3.1.

Proposition 6.18. Let C be a satisfiable Σ -dominance-constraint. Then S is a free set of nodes if and only if S satisfies the following conditions:

- (F1) There are no nodes $x \in S, y \notin S$ such that $y \triangleleft^* x$ is in C .
- (F2) There are no nodes $x, y \in S$ such that the constraint $x \neq y$ is in C .
- (F3) All nodes in S that are heads in labeling constraints are labeled with the same label $f \in \Sigma$.
- (F4) If $x_0 : f(x_1, \dots, x_{ar(f)})$ is a labeling constraint in C then the edges x_0x_i and x_0x_j are in different biconnected components in the dominance graph G of C , for $1 \leq i \neq j \leq ar(f)$.

With the same algorithmic approach as in Section 6.2 we thus reduced dominance graph satisfiability to the problem of finding a free set of nodes. However, for Σ -dominance-constraints this problem might be hard. We now discuss for which Σ this problem is hard and for which Σ it is tractable.

A tractable case. First we give a polynomial time algorithm for a special case: We assume that the input contains no inequalities, and that the signature only contains symbols of arity at most two. For each $f \in \Sigma$ we define the following auxiliary digraph. The vertices of the graph are the holes of the labeling constraints $x_0 : f(x_1, \dots, x_{\text{ar}(f)})$ in C where x_0 has indegree zero in the dominance graph of C . Two vertices uv are joined by an arc if they belong to the same labeling constraint, and u is to the left of v . Two vertices are joined by arcs uv and vu if there is an undirected path from u to v in the dominance graph avoiding f -labeled vertices. A set of free f -labeled nodes corresponds to a directed nontrivial cut (i.e., a partition of the vertices into two nontrivial sets such that all edges between the two parts are oriented in the same way) in the auxiliary graph for f , and vice versa. Using depth-first search it is easy to compute whether such a cut exists. These ideas can be used to show the following:

Proposition 6.19. *Let n be the number of nodes and m be the number of constraints in a Σ -dominance-constraint without inequalities. If Σ contains function symbols of arity at most two, then there is an algorithm that tests satisfiability in time $O(n(n + m))$.*

6.5 Surjective Homomorphism Problems

The problem of finding a directed nontrivial cut in the auxiliary digraph is an example of a class of computational problems, which we call *surjective homomorphism problems*. Let T be a fixed finite structure with relational signature τ .

SCSP(T)

INSTANCE: A finite structure S of the same relational signature τ as T .

QUESTION: Is there a *surjective* homomorphism from S to T ?

As for constraint satisfaction, we call T the *template* and S an *instance* of the problem SCSP(T). The above mentioned problem is thus the following surjective homomorphism problem where the template is a single directed edge.

If Σ contains function symbols of maximal arity k , the Σ -dominance-constraint problem becomes equivalent to the surjective homomorphism problem with the following template $Q = (\{0, \dots, k-1\}; R)$ where R is defined

as follows

$$R := \{(0, 1, \dots, k-1), (0, 0, \dots, 0), \dots, (k-1, \dots, k-1)\}.$$

Conversely, it is easy to reduce every instance of the surjective homomorphism problem for Q to satisfiability of a Σ -dominance-constraint without inequalities. We believe that the problem $\text{SCSP}(Q)$ is NP-hard. Note that Q is projective:

Proposition 6.20. *The only polymorphisms of Q are constants or projections.*

Proof. It is easy to see that the only unary polymorphisms of Q are the constants and the identity. We will now prove that there cannot be any essential polymorphism of Q (i.e. a function depending on at least two arguments). Suppose otherwise there is such an essential function f and wlog. f depends on at least the first two arguments. Then define $g(x, y) = f(x, y, c, \dots, c)$ for some constant c . Consider $g^*(x) = g(x, x)$, which is either a constant d or the identity.

In the first case, we directly see that all the functions $h_p(x) = g(x, p)$ and $h'_q(x) = g(q, x)$ are also the constant d , except for h_d and h'_d . But then all other function values of g must be d as well, a contradiction to the essentiality of g .

In the latter case, where $g^*(x) = x$ for all x , consider the function $h_d(y) = g(d, y)$ (the choice of d is arbitrary). Again h_d is either constant or the identity. If it is constant d then all functions $h_e(x) = g(x, e)$ except for $e = d$ have to be the identity. But then we see that g only depends on the first variable, a contradiction to the essentiality of g . If $h_d(y) = g(d, y)$ is also the identity, then all functions $h_e(x) = g(x, e)$ have to be constants, and we see that g only depends on its second variable, a contradiction to the essentiality of g . \square

Combining Theorem 4.4 and Lemma 3.3, we see that the constraint satisfaction problem for the template

$$Q' = (\{0, \dots, k-1\}; R, P_1, \dots, P_k)$$

where $P_i = \{i\}$ is the unary singleton relation containing element i only, is NP-hard, by reduction of 3-colorability. For surjective homomorphism problems, however, there is no such general theory, and the complexity remains open.

The constraint satisfaction problem $\text{CSP}(Q')$ for Q' can simulate the surjective homomorphism problem $\text{SCSP}(Q)$ for Q . If the skeleton graph¹ of an instance S to $\text{SCSP}(Q)$ consists of k or more connected components, the problem is trivial and obviously there exists a desired labeling. Otherwise, for each edge in S , impose singleton relations on the vertices of the edge consistently with their ordering and solve the corresponding constraint satisfaction problem for Q' . A valid solution of this algorithm is also a solution for the surjective problem. On the other hand, if there was no solution for none of the edges in S , clearly there is no surjective solution.

Describing unordered trees. As another example for the connection between constraint satisfaction problems for tree-like templates and surjective homomorphism problems, consider the following variant of the Σ -dominance-constraint problem. We relax the third condition and do no longer require that the vertices $x_1, \dots, x_{\text{ar}(f)}$ of a labeling constraint $x_0 : f(x_1, \dots, x_{\text{ar}(f)})$ are mapped to the children of $\alpha(x_0)$ *in this order*. Thus we only describe *unordered* trees. Consider again Figure 6.12. This constraint now becomes satisfiable.

For unordered trees we can in fact delineate the border between tractability and intractability. The problem is NP-hard if Σ contains a symbol of arity larger than three, and tractable in the case where Σ only contains symbols of arity at most three.

As in the tractable case for binary signatures, we solve a surjective homomorphism problem for each relation symbol $f \in \Sigma$. If f is k -ary, the surjective homomorphism problem we have to solve is defined as follows. We use the template $T = (\{0, \dots, k-1\}; R)$ where R consists of all tuples with only distinct elements and all k -tuples with only identical elements. Note that for $k = 3$ this is equivalent to saying that $(v_0, v_1, v_2) \in R$ iff $v_0 + v_1 + v_2 \equiv 0 \pmod{3}$. If $\text{SCSP}(T)$ is tractable for all relation symbols $f \in \Sigma$, then we can solve the unordered Σ -dominance-constraint problem analogously to Section 6.4.

Proposition 6.21. *Let $T_k = (\{0, 1, \dots, k\}; R)$ be the relational structure described above. Then $\text{SCSP}(T_k)$ is NP-hard for $k \geq 4$, and tractable for $k \leq 3$.*

¹Sometimes also called the *shadow graph* or the *Gaifman graph* of S , i.e., the graph on the nodes of S where we put an edge between between nodes x and y if there exists an entry of a relation that contains both x and y .

Proof. For $k = 1$ or $k = 2$ the problem is trivial. For larger k , we can view an instance of $\text{SCSP}(T_k)$ as a k -uniform hypergraph. If the skeleton of the hypergraph has k components, the problem has a trivial solution, since we can assign the same color to all vertices of one component, and different colors to the k different components, and thereby found a homomorphism that uses all the k colors. If the skeleton of the hypergraph has less than k components, every surjective homomorphism has to color the vertices of some hyperedge with k colors.

For $k = 3$ we can think of the nodes in the hypergraph are variables denoting values in Z_3 . A hyperedge $\{v_0, v_1, v_2\}$ in the instance is considered as an equation $v_0 + v_1 + v_2 \equiv 0 \pmod{3}$. Now we select some hyperedge $\{v_0, v_1, v_2\}$, and set the value of v_0 to 0, v_1 to 1, and v_2 to 2, and solve the resulting equation system, e.g., with Gaussian elimination. If there is a solution, we found a three-coloring that uses all colors. If there is no solution, we try the same with a different hyperedge. Suppose there is a surjective homomorphism. As already mentioned, this homomorphism colors the vertices of some hyperedge $\{v_0, v_1, v_2\}$ with k colors. By symmetry of the colors, we can assume without loss of generality that the homomorphism maps v_0 to 0, v_1 to 1, and v_2 to 2. Since the algorithm will eventually choose this edge, it finds a surjective homomorphism.

Then we prove that the problem is hard for $k = 4$ (for larger k we just use the first k entries of the tuples in R and can use the same proof). We reduce 3-colorability to the corresponding surjective homomorphism problem. Let S be an instance of $\text{CSP}(K_3)$. Without loss of generality we can assume that the skeleton graph of S is connected. We define an instance S' of $\text{SCSP}(T_4)$, defined on the vertices of S and a polynomial number of additional vertices. In the trivial case that S does not contain any edge, we let S' be any satisfiable instance to the surjective homomorphism problem. Otherwise arbitrarily order the edges e_1, \dots, e_m of S . Let (xy, uv) be a pair of edges, chosen from among the pairs (e_i, e_{i+1}) and (e_m, e_1) , and insert the following gadget G , where a_0, \dots, a_9 are new vertices for each pair of edges in S' . The gadget is also illustrated in Figure 6.13.

$$\begin{aligned} & \{\{a_0, a_1, x, x_3\}, \{a_1, a_2, a_3, a_4\}, \{a_4, u, a_6, v\}, \\ & \{x, a_3, y, a_5\}, \{y, a_5, a_7, a_8\}, \{a_5, a_6, a_8, a_9\}\} \end{aligned}$$

We can see that if $\alpha(x) = \alpha(y)$ in a solution α of G , then all nodes of the gadget have to be mapped to the same vertex, and $\alpha(u) = \alpha(v)$. Moreover we can exhaustively check that if we assign two different values to u and v , then we can still consistently assign any two distinct values to x and y and still extend this mapping to a solution of G .

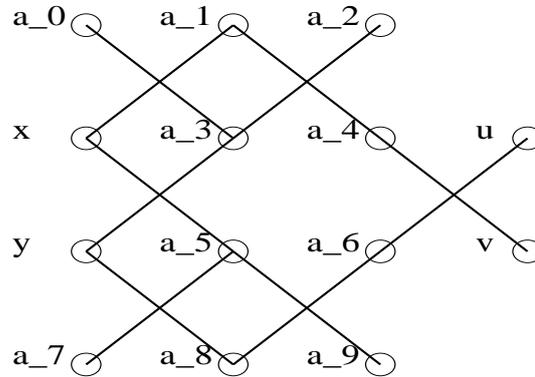


Figure 6.13: The gadget for the simulation of 3-colorability with $\text{SCSP}(T_4)$.

We claim that S' is a satisfiable instance of $\text{SCSP}(T_4)$ if and only if S is 3-colorable. If S is 3-colorable, we can consistently satisfy all hyperedges in G according to the above remark and find a surjective homomorphism from S' to T_4 . Now let S be not 3-colorable, and let α be an arbitrary mapping from S to three vertices. By construction the mapping α corresponds to a partial mapping from S' to T_4 . We show that this mapping can not be extended to a surjective homomorphism from S' to T_4 . Since α was chosen arbitrary this suffices for the claim. Since S is not 3-colorable there is an edge xy such that $\alpha(x) = \alpha(y)$. Because the skeleton of S is connected, and since all of the edges in S' are strongly connected by the gadget G , all nodes in S' have to be mapped to the same vertex. Thus no surjective solution exists. \square

Chapter 7

Conclusion and Outlook

Constraint satisfaction problems with ω -categorical templates cover several classes of constraint satisfaction problems that were investigated in the literature, for example several tree description languages, the fragments of Allen's interval algebra, and all problems in monotone monadic SNP. If the template of the constraint satisfaction problem is finitely constrained, we can place the corresponding class of constraint satisfaction problems between monotone monadic SNP and monotone SNP, two classes introduced in [Feder and Vardi, 1999] to study the complexity of constraint satisfaction problems with finite templates. In the first part of the thesis we demonstrated that several techniques for constraint satisfaction problems with finite templates also apply to ω -categorical templates. In the second part we discussed some concrete constraint satisfaction problems, motivated in computational linguistics and bio-informatics, and developed a new type of graph algorithm, which was not yet used for constraint satisfaction with finite templates.

The algebraic approach of constraint satisfaction. To study the complexity of a constraint satisfaction problem $\text{CSP}(\Gamma)$ it is useful to know which relations have a primitive positive definition over Γ . In Chapter 4 we showed that p.p.-definability in an ω -categorical structure Γ is characterized by the clone of polymorphisms of Γ . Between primitive positive definability and first-order definability we find several intermediate levels of definability, for instance *positive*, *existential*, and *positive existential definability*. On ω -categorical structures, they can again be characterized by closure conditions under certain sets of polymorphisms - for the three classes mentioned above it suffices to look at certain *unary* polymorphisms, i.e., endomorphisms.

The remainder of this chapter is organized as follows: we first give in Section 7.1 a summary on various preservation results for ω -categorical structures mentioned in this thesis, since they played a crucial rôle for a systematic approach to the constraint satisfaction problem on countable templates. Some of them are direct consequences from general results in model theory (Los-Tarski, Lyndon), others are new. A general picture was given in the literature only for the special case of finite structures: [Kalužnin and Pöschel, 1979] distinguish between *relational clones*, *weak Krasner clones* (also called *Krasner algebras of the first kind*), and *Krasner clones* (also called *Krasner algebras of the second kind*), and characterize all of them by closure conditions. For ω -categorical structures the Galois-connections that relate definability with closure conditions remain valid, but now we find several intermediate levels of definability.

In Section 7.2 we discuss our approach to consider ω -categorical templates for constraint satisfaction problems with templates over an infinite domain. We list some of the results that hold for ω -categorical structures, but do not hold in general.

In Section 7.3 several lines of future research are mentioned. Finally, in Section 7.4, we collect interesting open questions related to the topics discussed here, most of which were already mentioned in the previous chapters. Some of them are well-known open problems, others are new.

7.1 Summary of Closure Conditions

If we look at the various preservation theorems for ω -categorical structures, a general picture suggests itself. Let \mathcal{C} be a subset of the logical connectives $\{\forall, \exists, \neg, \neq\}$ that can be used together with \exists and \wedge in a definition of a logical formula (where we always assume that the formula is written in negation normal form). Then the corresponding closure condition can be found by looking at all polymorphisms subject to the following conditions:

1. If \forall is in \mathcal{C} , then we only consider *surjective* polymorphisms.
2. If \exists is in \mathcal{C} , then we only consider *unary* polymorphisms.
3. If \neq is in \mathcal{C} , then we only consider *injective* polymorphisms.
4. If \neg is in \mathcal{C} , then we only consider *strong* polymorphisms.

This gives rise to 16 Galois-connections. For finite templates, several of them collapse. Such collapses can be proven on both sides: For example surjective endomorphisms are isomorphisms on finite structures. Equivalently we observe that every first-order formula including equality can be defined in negation normal form using the connectives $\forall, \exists, \wedge, \vee$ only. For ω -categorical structures all the Galois-connections are different.

Note that what looks canonical is not yet proven in a systematic way, but consists of several single results and proofs for each Galois-connection. The proofs are different, and some of them part of classical model theory, for instance the theorems of Łos-Tarski, Lyndon, and Ryll-Nardzewski. Some others were only found recently, for instance the correspondence of definability with the connectives $\{\exists, \forall, \wedge\}$ to all surjective polymorphisms for finite structures in [Boerner et al., 2003], and also the correspondence of definability with the connectives $\{\exists, \wedge\}$ to all polymorphisms for ω -categorical structures in [Bodirsky and Nešetřil, 2003], described in Section 4.5.

Name	Syntactic connectives	Closure condition	Sect.	Related concept or theorem
Primitive Positive	\exists, \wedge	Polymorphisms	4.5	Projectivity
Existential Positive	\exists, \wedge, \vee	Endomorphisms	2.8	Cores
Existential	$\exists, \wedge, \vee, \neg$	Injective strong	2.4	Model-compl,
		Endomorphisms		Łos-Tarski
Positive	$\exists, \forall, \wedge, \vee$	Surjective	2.8	Lyndon
		Endomorphisms		
First-order	$\exists, \forall, \wedge, \vee, \neg$	Automorphisms	4.4	Ryll-Nardzewski

7.2 Discussion

We believe that ω -categoricity is an important concept if we want to systematically study constraint satisfaction problems for templates with an infinite domain. One can view ω -categoricity as the absence of *non-standard* models of the first-order theory of a template. Every structure can be made homogeneous by expanding the structure with a k -ary relation for every orbit of k -tuples, for each k . To make an ω -categorical structure Γ homogeneous, it suffices to expand with finitely many first-order definable k -ary relations, for each k . We can thus describe Γ by (the amalgamation class of) its *finite induced substructures* in this expanded signature.

For infinite templates it is in general not clear how to represent *solutions* of a given instance S . If the template is ω -categorical, we have a convenient way to specify a solution. The reason is that a solution is uniquely characterized by its first-order formulas, up to automorphisms of the template.

We now give a summary of the points in this thesis where the assumption of ω -categoricity of a structure Γ was essential. All of the following facts fail for arbitrary relational structures.

1. $\text{CSP}(\Gamma_1) \subseteq \text{CSP}(\Gamma_2)$ if and only if there is a homomorphism from Γ_1 to Γ_2 . (Proposition 2.24 on page 46)
2. Every constraint satisfaction problem with an ω -categorical template Γ of width (k, l) is solved by the canonical Datalog program of width (k, l) . (Theorem 3.12 on page 74)
3. $\langle \Gamma \rangle_{pp} = \text{Inv}(\text{Pol}(\Gamma))$. (Theorem 4.9 on page 90)
4. $\langle \Gamma \rangle_{fo} = \text{Inv}(\text{Aut}(\Gamma))$. (Theorem 4.7 on page 88)
5. The existence of a near-unanimity function implies bounded strict width. (Theorem 4.14 on page 94)
6. Adding singleton relations to an ω -categorical core does not change the computational complexity of the corresponding constraint satisfaction problem. (Theorem 4.20 on page 97)

7.3 Outlook

The polymorphisms of the semilinear order Λ . The constraint satisfaction problem for the ω -categorical structure $(\Lambda; \triangleleft^+, \perp)$ is tractable. There are many first-order definable relations in Λ that are not primitive positive definable. This implies that there are nontrivial polymorphisms. We would like to characterize the applicability of the algorithmic techniques in Chapters 6 and ?? in terms of these polymorphisms.

Primitive positive interpretations. In Section 3.2 in Garey and Johnson's Book on the theory of NP-hardness three techniques of proving NP-hardness of a computational problem are mentioned: *restriction*, *local replacement*, and *component design*. The second type of reductions is a very

powerful concept to study a problem $\text{CSP}(\Gamma)$. By Lemma 3.3 on page 55 the concept of primitive positive definability captures the technique of *local replacement* for proving hardness.

It would be interesting to fully characterize polynomial time reductions between constraint satisfaction problems by a certain relationship between templates. Clearly, primitive positive definability is not enough – this can already be seen for finite templates. It is easy to see that also for infinite templates Γ and Γ' , if Γ has a primitive positive interpretation in Γ' then the problem $\text{CSP}(\Gamma)$ reduces in polynomial time to $\text{CSP}(\Gamma')$. An example that shows that the converse is not true is $S(2)$ and the ternary betweenness relation on \mathbb{Q} . The problem Betweenness reduces to $\text{CSP}(S(2))$, but there is no primitive positive interpretation (without additional parameters) that allows to define \mathbb{Q} . However, there is a primitive positive definition of the betweenness relation of a set that is definable with a single parameter – see also Section 3.2.

We give another example where even a finite number of parameters does not suffice to define the domain of the simulated template. Consider the Fraïssé-limit of the leaves of the Boron trees described in Section 2.7, where we fixed a point (a so-called *C-set* in [Adeleke and Neumann, 1985]). Let us call this structure C , where we use the signature containing the ternary relation ‘:’ only, defined in Section 2.7. The constraint satisfaction problem $\text{CSP}(C)$ can be solved by $\text{CSP}(\Lambda)$, using primitive positive definitions. However, there is no primitive positive interpretation with finitely many parameters of C in Λ . It seems that we need an adjustment in how we can define the universe of the template that is simulated by another template.

Allen’s interval algebra. We already mentioned that the fragments of Allen’s interval algebra exhibit a complexity dichotomy. We would like to generalize this dichotomy to all constraint satisfaction problems with a template that has an interpretation over the dense linear order. We believe that hardness of such a constraint satisfaction problem can always be proven by a simulation of the problem Betweenness, and we conjecture a dichotomy: In fact we think that all other cases can be solved by a Datalog program.

7.4 List of Open Problems

There are plenty of open problems and questions, and we are hopeful that for some of them solutions and answers are not out of reach. They are grouped into questions from model theory, finite model theory, and computational problems.

7.4.1 Model Theory and Combinatorics

We already wrote that algebraic techniques and cores might be applied to simplify the technical and intricate proofs in the classification of the tractable fragments of Allen's interval algebra [Jeavons et al., 2003]. The result might them be generalized for the following more general problem.

Question 1. *Which structures interpretable in the dense linear order of the rational numbers $(\mathbb{Q}; <)$ have tractable constraint satisfaction problems?*

We can often turn a mathematical classification or characterization question, such as Question 1, into a concrete and formal question, if we pose it as a decision problem. Usually the solution to such a decision problem requires the mathematical knowledge that we are interested in. For instance we could ask instead of Question 1: Is it decidable whether a structure interpretable in \mathbb{Q} has a tractable constraint satisfaction problem (under the assumption that $P \neq NP$)?

Correspondingly we have the same problem for the semilinear order Λ :

Question 2. *Which structures interpretable in the ω -categorical semilinear order Λ (defined in Section 2.7) have tractable constraint satisfaction problems?*

For Λ we presented in Chapter 6 efficient algorithms for the constraint satisfaction problem for Λ with various choices of the relational signature. We do not yet know a criterion for tractability that is formulated in terms of polymorphisms.

Question 3. *What are the polymorphisms of Λ that are responsible for the existence of the polynomial time algorithms?*

To answer the question whether the model checking problem of a given monotone SNP formula can be described as a constraint satisfaction problem

with a countable homogeneous structure, the following problem posed by Cherlin [Cherlin, 1998] is of importance. Some partial results for graphs, and forbidden subgraphs instead of forbidden induced substructures, can be found in [Cherlin et al., 1999]. Let τ be a relational signature and \mathcal{N} a finite set of finite τ -structures.

Question 4 (Cherlin). *When does there exist an ω -categorical structure that is universal for $\text{Forb}(\mathcal{N})$? When is $\text{Forb}(\mathcal{N})$ an amalgamation class?*

We do not know an example of a monotone SNP sentence Φ that closed under disjoint sums, that can not be stated as a constraint satisfaction problem with a finitely constrained ω -categorical template. We combine this observation with Theorem 3.10, and ask the following question.

Question 5. *Let Q be a problem that is closed under disjoint sums. Is it true that Q is in MSNP if and only if it is contained in CSP^* ?*

The following question has a positive answer for finite templates, and is open for countable homogeneous templates of finitely axiomatizable age. It is already open for special ω -categorical structures, for instance for Λ .

Question 6. *Let Γ be an ω -categorical structure. Given a first-order formula φ over τ , is it decidable whether φ is on Γ equivalent to a primitive positive formula?*

The following question is already open for finite Γ ; see the paragraph on infinite signatures in Section 3.1.

Question 7. *Is there a problem in CSP^* which is tractable, but not globally tractable?*

Random \aleph_0 -categorical Structures. For which sets \mathcal{N} of forbidden subgraphs has the class $\text{Forb}(\mathcal{N})$ a first-order 0 – 1-law? For the classes defined by forbidden complete graphs K_n this is for instance the case [Kolaitis et al., 1987], and the almost sure theory of the class of K_n -free graphs is even ω -categorical. We would like to know:

Question 8. *For which sets \mathcal{N} of finite graphs (or, more generally, finite structures over a finite signature) the almost-sure theory of $\text{Forb}(\mathcal{N})$ is ω -categorical?*

7.4.2 Constraint Satisfaction and Datalog

Other questions concern the power of Datalog for constraint satisfaction. We already mentioned the following question in Section 3.4.

Question 9. *Let \mathcal{N} be a finite set of finite structures, such that $\text{Forb}(\mathcal{N})$ is an amalgamation class, and let Γ be its Fraïssé-limit. Is it decidable whether $\text{CSP}(\Gamma)$ has width k , or bounded width? Is this decidable for a given reduct Γ' of Γ ?*

The above problem is already open for finite Γ . It would also be interesting to learn more about the difference of bounded and of strict bounded width. All known constraint satisfaction problems of bounded width either have strict bounded width or width 2, and the following problem was in [Feder and Vardi, 1993] asked in a similar form for finite structures:

Question 10. *Is there a constraint satisfaction problem which has bounded width k for $k \geq 3$, but does not have bounded strict width or width 2?*

Also the following problem was asked for finite structures Γ in [Feder and Vardi, 1999].

Question 11. *Given a finite set of finite forbidden substructures of a homogeneous relational structure Γ , is it decidable whether $\text{CSP}(\Gamma)$ has bounded strict width?*

Finally we return to the computational problem of Consistent-genealogy. We showed in Section 3.4 that this problem does not have width $(2, 3)$. Chapter 6, however, contains a polynomial time algorithm for that problem. We ask:

Question 12. *Does the problem of Consistent genealogy, i.e., $\text{CSP}((\Lambda; <, \not\leq))$, have bounded width?*

7.4.3 Computational Questions

One class of tractable constraint satisfaction problems we mentioned only briefly: the class of constraint satisfaction problems with a template having a Maltsev operation. For finite templates, such problems can be solved by group theoretic algorithms [Bulatov, 2002a].

Question 13. *Let Γ be an ω -categorical structure with a finite relational signature, where $\text{Pol}(\Gamma)$ contains a Malt'sev operation. Is $\text{CSP}(\Gamma)$ tractable?*

We mentioned the G -free coloring problem in Section 3.3. This is generalized to the \mathcal{G} -free coloring problem in [Cowen and Hechler, 2003]. They ask: for which sets of graphs \mathcal{G} is the \mathcal{G} -free coloring problem NP-hard? This is a special case of the following question:

Question 14. *For which $\text{MMSNP}(\neq)$ -sentences on graphs the model-checking problem is NP-hard?*

We know that the dichotomy holds for $\text{CSP}(H)$, if H is a finite graphs. It could even be the case that the dichotomy holds for $\text{CSP}(H)$, where H is an arbitrary infinite graph.

Question 15. *Let H be an infinite graph. Is it true that $\text{CSP}(H)$ is either tractable or NP-hard?*

Surjective homomorphism problems. We introduced the class of surjective homomorphism problems because several constraint satisfaction problems with tree-like ω -categorical template turned out to be computationally equivalent to such a problem. The class of surjective constraint satisfaction problems itself appears to be an interesting class of computational problems. We ask:

Question 16. *Is every surjective homomorphism problem computationally equivalent to a problem in CSP^* ?*

Correspondingly to the *primary classification question* of [Feder and Vardi, 1999] for constraint satisfaction we can ask the following:

Question 17. *Which finite structures have tractable surjective homomorphism problems?*

The answer to this question seems to be harder as compared to CSP, since hardness proofs tend to be tedious here. We even find single instances of unknown computational complexity in this class.

3-PARTITION

INSTANCE: A ternary relation R on a vertex set V .

QUESTION: Is there a partition of V into three non-empty sets V_1, V_2, V_3 such that for every tuple $(x, y, z) \in R$ either x, y, z lie in the same part, or $x_1 \in V_1, x_2 \in V_2, x_3 \in V_3$?

This problem is hard if we can specify in the instance that certain vertices have to be in different parts; see Section 6.5.

Question 18. *Is the problem 3-partition NP-hard?*

We believe that this is indeed the case, as well as for the following problem.

NO-RAINBOW-COLORING

INSTANCE: A hypergraph H .

QUESTION: Is there a nontrivial coloring of the vertex set such that no edge is *rainbow-colored*, at least two vertices of each edge edge is assigned the same color?

This is open already for 3-uniform hypergraphs. If we could settle this question, it would solve one of the remaining open problems in a classification of computational problems in [Kral et al., 2002].

Question 19. *Is the problem No-rainbow-coloring NP-hard?*

Bibliography

- [Abiteboul et al., 1995] Abiteboul, S., Hull, R., and Vianu, V. (1995). *Foundations of Databases*. Addison Wesley.
- [Achlioptas, 1997] Achlioptas, D. (1997). The complexity of G -free colourability. *Discrete Mathematics*, 165:21–30.
- [Adeleke and Neumann, 1985] Adeleke, S. and Neumann, P. M. (1985). Structure of partially ordered sets with transitive automorphism groups. *AMS Memoir*, 57(334).
- [Aho et al., 1981] Aho, A., Sagiv, Y., Szymanski, T., and Ullman, J. (1981). Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. *SIAM Journal on Computing*, 10(3):405–421.
- [Allen, 1983] Allen, J. F. (1983). Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.
- [Althaus et al., 2001] Althaus, E., Duchier, D., Koller, A., Mehlhorn, K., Niehren, J., and Thiel, S. (2001). An efficient algorithm for the configuration problem of dominance graphs. In *Proceedings of the 12th ACM-SIAM Symposium on Discrete Algorithms (SODA'01)*, pages 815–824, Washington, DC.
- [Althaus et al., 2003] Althaus, E., Duchier, D., Koller, A., Mehlhorn, K., Niehren, J., and Thiel, S. (2003). An efficient graph algorithm for dominance constraints. *Journal of Algorithms*, pages 194–219.
- [Baader and Schulz, 2001] Baader, F. and Schulz, K. (2001). Combining constraint solving. *H. Comon, C. March, and R. Treinen, editors, Constraints in Computational Logics*.

BIBLIOGRAPHY

- [Backofen et al., 1995] Backofen, R., Rogers, J., and Vijay-Shanker, K. (1995). A first-order axiomatization of the theory of finite trees. *Journal of Logic, Language, and Information*, 4:5–39.
- [Bauslaugh, 1995] Bauslaugh, B. (1995). Core-like properties of infinite graphs and structures. *Disc. Math.*, 138(1):101–111.
- [Bauslaugh, 1996] Bauslaugh, B. (1996). Cores and compactness of infinite directed graphs. *Journal of Combinatorial Theory, Series B*, 68(2):255–276.
- [Bodirsky et al., 2004] Bodirsky, M., Duchier, D., Niehren, J., and Miele, S. (2004). A new algorithm for normal dominance constraints. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA'04)*, pages 59–67, New Orleans.
- [Bodirsky and Kutz, 2002] Bodirsky, M. and Kutz, M. (2002). Pure dominance constraints. In *Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science (STACS'02)*, LNCS 2285, pages 287–298, Antibes - Juan le Pins.
- [Bodirsky and Nešetřil, 2003] Bodirsky, M. and Nešetřil, J. (2003). Constraint satisfaction with countable homogeneous templates. In *Proceedings of Computer Science Logic (CSL'03)*, pages 44–57, Vienna.
- [Bodnarčuk et al., 1969] Bodnarčuk, V. G., Kalužnin, L. A., Kotov, V. N., and Romov, B. A. (1969). Galois theory for post algebras, part I and II. *Cybernetics*, 5:243–539.
- [Boerner et al., 2003] Boerner, F., Bulatov, A., Krokhn, A., and Jeavons, P. (2003). Quantified constraints: Algorithms and complexity. In *Proceedings of 17th International Workshop Computer Science Logic (CSL'03)*, LNCS 2803, pages 58–70.
- [Bos, 1996] Bos, J. (1996). Predicate logic unplugged. In *Proceedings of the 10th Amsterdam Colloquium*, pages 133–143.
- [Bryant, 1997] Bryant, D. (1997). Building trees, hunting for trees, and comparing trees. PhD-thesis at the University of Canterbury.
- [Bulatov, 2002a] Bulatov, A. (2002a). Malt'sev constraints are tractable. Technical report PRG-RR-02-05, Oxford University.

- [Bulatov, 2003] Bulatov, A. (2003). Tractable conservative constraint satisfaction problems. In *Proceedings of 18th Symposium on Logig in Computer Science (LICS'03)*, pages 321–330.
- [Bulatov and Dalmau, 2003] Bulatov, A. and Dalmau, V. (2003). Towards a dichotomy theorem for the counting constraint satisfaction problem. In *Annual Symposium on Foundations of Computer Science (FOCS'03)*, pages 562–574.
- [Bulatov and Grohe, 2004] Bulatov, A. and Grohe, M. (2004). The complexity of partition functions. Preprint.
- [Bulatov et al., 2000] Bulatov, A., Krokhin, A., and Jeavons, P. (2000). Constraint satisfaction problems and finite algebras. In *Proceedings of International Colloquium on Automata, Languages and Programming (ICALP'00)*, LNCS 1853, pages 272–282.
- [Bulatov et al., 2001] Bulatov, A., Krokhin, A., and Jeavons, P. (2001). The complexity of maximal constraint languages. In *Proceedings of the 33rd Annual ACM Symposium on Theory of Computing (STOC'01)*, pages 667–674.
- [Bulatov et al., 2003] Bulatov, A., Krokhin, A., and Jeavons, P. G. (2003). Classifying the complexity of constraints using finite algebras. *Submitted*.
- [Bulatov, 2002b] Bulatov, A. A. (2002b). A dichotomy theorem for constraints on a three-element set. In *Annual Symposium on Foundations of Computer Science (FOCS'02)*, pages 649–658.
- [Bürckert and Nebel, 1995] Bürckert, H.-J. and Nebel, B. (1995). Reasoning about temporal relations: A maximal tractable subclass of Allen's interval algebra. *Journal of the ACM*, 42(1):43–66.
- [Cameron, 1981] Cameron, P. J. (1981). Orbits of permutation groups on unordered sets, II. *Journal of the London Mathematical Society*, 2:249–264.
- [Cameron, 1990] Cameron, P. J. (1990). *Oligomorphic Permutation Groups*. Cambridge University Press.
- [Cameron, 1996] Cameron, P. J. (1996). The random graph. *R. L. Graham and J. Nešetřil, Editors, The Mathematics of Paul Erdős*.

BIBLIOGRAPHY

- [Chandra and Harel, 1982] Chandra, A. and Harel, D. (1982). Structure and complexity of relational queries. *Journal of Computer and System Sciences*, 25:99–128.
- [Chang and Keisler, 1977] Chang, C. and Keisler, H. (1977). *Model Theory*. volume 73 of *Studies in Logic and the Foundations of Mathematics*. Elsevier, Amsterdam.
- [Chaudhuri and Vardi, 1992] Chaudhuri, S. and Vardi, M. Y. (1992). On the equivalence of recursive and nonrecursive datalog programs. In *Proceedings of the Symposium on Principles of Database Systems (PODS'92)*, pages 107–116.
- [Cherlin, 1987] Cherlin, G. (1987). Homogeneous digraphs I. The imprimitive case. *Logic Colloquium 1985*.
- [Cherlin, 1998] Cherlin, G. (1998). The classification of countable homogeneous directed graphs and countable homogeneous n -tournaments. *AMS Memoir*, 131(621).
- [Cherlin and Lachlan, 1986] Cherlin, G. and Lachlan, A. H. (1986). Stable finitely homogeneous structures. *Transactions of the AMS*, 296:815–850.
- [Cherlin et al., 1999] Cherlin, G., Shelah, S., and Shi, N. (1999). Universal graphs with forbidden subgraphs and algebraic closure. *Advances in Applied Mathematics*, 22:454–491.
- [Chor and Sudan, 1998] Chor, B. and Sudan, M. (1998). A geometric approach to betweenness. *SIAM Journal on Discrete Mathematics*, 11(4):511–523.
- [Cohen et al., 2004] Cohen, D., Cooper, M., Jeavons, P., and Krokhin, A. A. (2004). Identifying efficiently solvable cases of max csp. In *Proceedings of the 21st Annual Symposium on Theoretical Aspects of Computer Science (STACS'04)*, LNCS 2285, pages 152–163.
- [Cooper, 1989] Cooper, M. (1989). An optimal k -consistency algorithm. *Artificial Intelligence*, 41(1):89–95.
- [Copestake et al., 1999] Copestake, A., Flickinger, D., Sag, I., and Pollard, C. (1999). Minimal recursion semantics: An introduction. CSLI Draft.
- [Cornell, 1994] Cornell, T. (1994). On determining the consistency of partial descriptions of trees. In *32nd Annual Meeting of the Association for Computational Linguistics*, pages 163–170.

- [Cowen and Hechler, 2003] Cowen, R. and Hechler, S. H. (2003). G-free colorability and the boolean prime ideal theorem. Preprint.
- [Creignou et al., 2001] Creignou, N., Khanna, S., and Sudan, M. (2001). Complexity classifications of boolean constraint satisfaction problems. *Monographs on Discrete Mathematics and Applications*, 7.
- [Dalmau, 2000a] Dalmau, V. (2000a). Computational complexity of problems over generalized formulas. PhD-thesis at the Departament de Llenguatges i Sistemes Informàtics at the Universitat Politècnica de Catalunya.
- [Dalmau, 2000b] Dalmau, V. (2000b). A new tractable class of constraint satisfaction problems. Presented at the 6th International Symposium on Mathematics and Artificial Intelligence.
- [Dalmau et al., 2002] Dalmau, V., Kolaitis, P. G., and Vardi, M. Y. (2002). Constraint satisfaction, bounded treewidth, and finite-variable logics. In *Constraint Programming (CP'02)*, pages 310–326.
- [Dalmau and Pearson, 1999] Dalmau, V. and Pearson, J. (1999). Closure functions and width 1 problems. *Principles and Practice of Constraint Programming (CP'99)*, pages 159–173.
- [Datar et al., 2003] Datar, M., Feder, T., Gionis, A., Motwani, R., and Panigrahy, R. (2003). A combinatorial algorithm for MAX CSP. *Information Processing Letters*, 85(6):307–315.
- [Dechter, 1992] Dechter, R. (1992). From local to global consistency. *Artificial Intelligence*, 55(1):87–107.
- [Dechter, 2003] Dechter, R. (2003). *Constraint Processing*. Morgan Kaufmann.
- [Dechter and van Beek, 1997] Dechter, R. and van Beek, P. (1997). Local and global relational consistency. *Journal of Theoretical Computer Science*, 173(1):283–308.
- [Demetrescu and Italiano, 2000] Demetrescu, C. and Italiano, G. F. (2000). Fully dynamic transitive closure: breaking through the $O(n^2)$ barrier. In *Proceedings of the 41st IEEE Symposium on the Foundations of Computing (STOC'00)*, pages 381–389.
- [Diestel, 1997] Diestel, R. (1997). *Graph Theory*. Springer-Verlag, New York.

BIBLIOGRAPHY

- [Droste, 1985] Droste, M. (1985). Structure of partially ordered sets with transitive automorphism groups. *AMS Memoir*, 57(334).
- [Droste et al., 1991] Droste, M., Holland, W., and Macpherson, D. (1991). Automorphism groups of homogeneous semilinear orders: normal subgroups and commutators. *Canadian Journal of Mathematics*, 43:721–737.
- [Duchier and Thater, 1999] Duchier, D. and Thater, S. (1999). Parsing with tree descriptions: a constraint-based approach. In *Sixth International Workshop on Natural Language Understanding and Logic Programming (NLULP'99)*, pages 17–32.
- [Dyer and Greenhill, 2000] Dyer, M. E. and Greenhill, C. S. (2000). The complexity of counting graph homomorphisms. *Random Structures and Algorithms*, 17:260–289.
- [Ebbinghaus and Flum, 1999] Ebbinghaus, H.-D. and Flum, J. (1999). *Finite Model Theory*. Springer. 2nd edition.
- [Egg et al., 2001] Egg, M., Koller, A., and Niehren, J. (2001). The Constraint Language for Lambda Structures. *Journal of Logic, Language, and Information*, 10:457–485.
- [Erdős, 1959] Erdős, P. (1959). Graph theory and probability. *Canad J. Math.*, 11:34–38.
- [Fagin, 1976] Fagin, R. (1976). Probabilities on finite models. *Journal of Symbolic Logic*, 41:50–58.
- [Feder and Kolaitis, 2004] Feder, T. and Kolaitis, P. (2004). Closures and dichotomies for quantified constraints. Preprint.
- [Feder et al., 2002] Feder, T., Madelaine, F., and Stewart, I. (2002). Dichotomies for classes of homomorphism problems involving unary functions. To appear in *Theoretical Computer Science*.
- [Feder and Vardi, 1993] Feder, T. and Vardi, M. (1993). Monotone monadic snp and constraint satisfaction. In *Proceedings of the Symposium on Theory of Computing (STOC'93)*, pages 612–622.
- [Feder and Vardi, 1999] Feder, T. and Vardi, M. (1999). The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM Journal on Computing*, 28:57–104.

- [Feder and Vardi, 2003] Feder, T. and Vardi, M. (2003). Homomorphism closed vs. existential positive. In *Symposium on Logic in Computer Science (LICS'03)*, pages 311–320.
- [Fraïssé, 1986] Fraïssé, R. (1986). *Theory of Relations*. North-Holland.
- [Freuder, 1978] Freuder, E. C. (1978). Synthesizing constraint expressions. *Communications of the ACM*, 21(11):958–966.
- [Freuder, 1982] Freuder, E. C. (1982). A sufficient condition for backtrack-free search. *Journal of the Association for Computing Machinery*, 29(1):24–32.
- [Freuder, 1990] Freuder, E. C. (1990). Complexity of k -tree structured constraint satisfaction problems. In *Proceedings of the 8th National Conference on Artificial Intelligence*, pages 4–9.
- [Gardent and Webber, 1998] Gardent, C. and Webber, B. (1998). Describing discourse semantics. In *Proceedings of the 4th TAG+ Workshop*, Philadelphia. University of Pennsylvania.
- [Garey and Johnson, 1978] Garey, M. and Johnson, D. (1978). *A Guide to NP-completeness*. CSLI Press, Stanford.
- [Geiger, 1968] Geiger, D. (1968). Closed systems of functions and predicates. *Pacific Journal of Mathematics*, 27:95–100.
- [Grohe, 2003] Grohe, M. (2003). The complexity of homomorphism and constraint satisfaction problems seen from the other side. In *Proceedings of the 44th IEEE Symposium on Foundations of Computer Science (FOCS'03)*.
- [Gupta and Nishimura, 1996] Gupta, A. and Nishimura, N. (1996). Characterizing the complexity of subgraph isomorphism for graphs of bounded path-width. In *Proceedings of the 13th Annual Symposium on Theoretical Aspects of Computer Science (STACS'96)*, pages 453–464.
- [Gurevich, 1984] Gurevich, Y. (1984). Toward logic tailored for computational complexity. *Computation and Proof Theory*, pages 175–216.
- [Gusfield, 1997] Gusfield, D. (1997). *Algorithms on strings, trees, and sequences. Computer Science and Computational Biology*. Cambridge University Press, New York.
- [Hell and Nešetřil, 1990] Hell, P. and Nešetřil, J. (1990). On the complexity of H-coloring. *Journal of Combinatorial Theory, Series B*, 48:92–110.

- [Henson, 1972] Henson, C. W. (1972). Countable homogeneous relational systems and categorical theories. *Journal of Symbolic Logic*, 37:494–500.
- [Henzinger and King, 1995] Henzinger, M. and King, V. (1995). Fully dynamic biconnectivity and transitive closure. In *Proceedings 36th Symposium on Foundations of Computer Science (FOCS'95)*, pages 664–672.
- [Henzinger et al., 1996] Henzinger, M., King, V., and Warnow, T. (1996). Combining constraint solving. In *Proceedings of the 7th Symposium on Discrete Algorithms (SODA'96)*, pages 333–340.
- [Hodges, 1993] Hodges, W. (1993). *Model theory*. Cambridge University Press.
- [Hodges, 1997] Hodges, W. (1997). *A shorter model theory*. Cambridge University Press.
- [Holm et al., 2001] Holm, J., de Lichtenberg, K., and Thorup, M. (2001). Poly-logarithmic deterministic fully-dynamic algorithms for connectivity, minimum spanning tree, 2-edge, and biconnectivity. *Journal of the ACM*, 48(4):723–760.
- [Immerman, 1998] Immerman, N. (1998). *Descriptive Complexity*. Graduate Texts in Computer Science, Springer.
- [Jeavons et al., 1998] Jeavons, P., Cohen, D., and Cooper, M. (1998). Constraints, consistency and closure. *Artificial Intelligence*, 101(1-2):251–265.
- [Jeavons et al., 1997] Jeavons, P., Cohen, D., and Gyssens, M. (1997). Closure properties of constraints. *Journal of the ACM*, 44(4):527–548.
- [Jeavons et al., 2003] Jeavons, P., Jonsson, P., and Krokhin, A. A. (2003). Reasoning about temporal relations: The tractable subalgebras of Allen's interval algebra. *Journal of the ACM*, 50(5):591–640.
- [Jeavons, 1998] Jeavons, P. G. (1998). On the algebraic structure of combinatorial problems. *Theoretical Computer Science*, 200:185–204.
- [Kalužnin and Pöschel, 1979] Kalužnin, L. A. and Pöschel, R. (1979). *Funktionen- und Relationenalgebren*. Deutscher Verlag der Wissenschaften.
- [Kolaitis et al., 1987] Kolaitis, P. G., Prömel, H. J., and Rothschild, B. L. (1987). K_{l+1} -free graphs: asymptotic structure and a 0–1 law. *Transactions of the AMS*, 303:637–671.

- [Kolaitis and Vardi, 1992] Kolaitis, P. G. and Vardi, M. Y. (1992). 0-1 laws and decision problems for fragments of second order logic. *Journal of Information and Computation*, 98:258–294.
- [Kolaitis and Vardi, 1995] Kolaitis, P. G. and Vardi, M. Y. (1995). On the expressive power of Datalog: Tools and a case study. *Journal of Computer and System Sciences*, 51(1):110–134.
- [Kolaitis and Vardi, 1998] Kolaitis, P. G. and Vardi, M. Y. (1998). Conjunctive-query containment and constraint satisfaction. In *Proceedings of the 17th ACM SIGACT SIGMOD SIGART Symposium on Principles of Database Systems (PODS'98)*, pages 205–213.
- [Kolaitis and Vardi, 2000] Kolaitis, P. G. and Vardi, M. Y. (2000). A game-theoretic approach to constraint satisfaction. In *Proceedings of the 17th National Conference on Artificial Intelligence*, pages 175–181.
- [Koller et al., 2000] Koller, A., Mehlhorn, K., and Niehren, J. (2000). A polynomial-time fragment of dominance constraints. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL'00)*, pages 368–375, Hong Kong.
- [Koller et al., 1998] Koller, A., Niehren, J., and Treinen, R. (1998). Dominance constraints: Algorithms and complexity. In *Third International Conference on Logical Aspects of Computational Linguistics (LACL'98)*, pages 106–125, Grenoble, France.
- [Kral et al., 2002] Kral, D., Kratochvíl, J., Proskurowski, A., and Voss, H.-J. (2002). Mixed hypertrees. *ITI Series*, 048.
- [Krasner, 1945] Krasner, M. (1945). Généralisation et analogues de la théorie de Galois. *Congrès de la Victoire de l'Ass. France avancement des sciences*, pages 54–58.
- [Krasner, 1968] Krasner, M. (1968). Endothéorie de Galois abstraite. *Séminaire P. Dubreil (Algèbre et Théorie des Nombres)*, 1(6).
- [Kumar, 1992] Kumar, V. (1992). Algorithms for constraints satisfaction problems: A survey. *The AI Magazine, by the AAAI*, 13(1):32–44.
- [Lachlan, 1984] Lachlan, A. H. (1984). Countable homogeneous tournaments. *Transactions of the AMS*, 284:431–461.

BIBLIOGRAPHY

- [Lachlan, 1996] Lachlan, A. H. (1996). Stable finitely homogeneous structures: A survey. In *Algebraic Model Theory, NATO ASI Series*, volume 496, pages 145–159.
- [Lachlan and Woodrow, 1980] Lachlan, A. H. and Woodrow, R. (1980). Countable ultrahomogeneous undirected graphs. *Transactions of the AMS*, 262(1):51–94.
- [Ladner, 1975] Ladner, R. E. (1975). On the structure of polynomial time reducibility. *Journal of the ACM*, 22(1):155–171.
- [Larose and Tardif, 2001] Larose, B. and Tardif, C. (2001). Strongly rigid graphs and projectivity. *Multiple-Valued Logic 7*, pages 339–361.
- [Latka, 1994] Latka, B. (1994). Finitely constrained classes of homogeneous directed graphs. *J. of Symb. Logic*, 59(1):124–139.
- [Lincoln and Mitchell, 1992] Lincoln, P. and Mitchell, J. C. (1992). Algorithmic aspects of type inference with subtypes. In *Proceedings of the 19th Symposium on Principles of Programming Languages*, pages 293–304.
- [Lippel, 2001] Lippel, D. (2001). Finitely axiomatizable omega-categorical theories. PhD-dissertation, University of California, Berkeley.
- [Łuczak and Nešetřil, 2004] Łuczak, T. and Nešetřil, J. (2004). A note on projective graphs. *J. Graph Theory*, 47:81–86.
- [Mackworth and Freuder, 1993] Mackworth, A. K. and Freuder, E. C. (1993). The complexity of constraint satisfaction revisited. *Artificial Intelligence*, 59(1-2):57–62.
- [Madelaine and Stewart, 1999] Madelaine, F. and Stewart, I. A. (1999). Some problems not definable using structure homomorphisms. *MCS technical report. University of Leicester*, 99(18).
- [Marcus et al., 1983] Marcus, M. P., Hindle, D., and Fleck, M. M. (1983). D-theory: Talking about talking about trees. In *Proceedings of the 21st Annual Meeting of the Association for Computational Linguistics (ACL'83)*, pages 129–136.
- [Matijasevich, 1993] Matijasevich, Y. V. (1993). *Hilbert's Tenth Problem*. MIT Press.

- [Mayr et al., 1998] Mayr, E. W., Prömel, H. J., and A. Steger, H. (1998). *Lectures on Proof Verification and Approximation Algorithms*. Lecture Notes in Computer Science Vol. 1367, Springer Verlag.
- [Montanari, 1974] Montanari, U. (1974). Networks of constraints: Fundamental properties and applications to picture processing. *Information Sciences*, 7:95–132.
- [Nešetřil and Tardif, 2000] Nešetřil, J. and Tardif, C. (2000). Duality theorems for finite structures (characterising gaps and good characterisations). *Journal of Combinatorial Theory Series B*, 80:80–97.
- [Ng et al., 2000] Ng, M. P., Steel, M., and Wormald, N. C. (2000). The difficulty of constructing a leaf-labelled tree including or avoiding given subtrees. *Discrete Applied Mathematics*, 98:227–235.
- [Niehren and Thater, 2003] Niehren, J. and Thater, S. (2003). Bridging the gap between underspecification formalisms: Minimal recursion semantics as dominance constraints. In *41st Meeting of the Association for Computational Linguistics*, pages 367–374.
- [Papadimitriou and Yannakakis, 1991] Papadimitriou, C. H. and Yannakakis, M. (1991). Optimization, approximation, and complexity classes. *Journal of Computer and System Sciences*, 43:425–440.
- [Paterson and Wegman, 1978] Paterson, M. S. and Wegman, M. N. (1978). Linear unification. *Journal of Computer and System Sciences*, 16:158–167.
- [Pinkal, 1996] Pinkal, M. (1996). Radical Underspecification. In *Proceedings of the 10th Amsterdam Colloquium*, pages 587–606.
- [Pöschel, 1980] Pöschel, R. (1980). A general galois theory for operations and relations and concrete characterization of related algebraic structures. *Technical Report of Akademie der Wissenschaften der DDR (Berlin)*.
- [Rogers and Shanker, 1992] Rogers, J. and Shanker, V. (1992). Reasoning with descriptions of trees. In *Proceedings of the 30th Meeting of the Association for Computational Linguistics*, pages 72–80.
- [Rogers and Vijay-Shanker, 1994] Rogers, J. and Vijay-Shanker, K. (1994). Obtaining trees from their descriptions: An application to tree-adjointing grammars. *Computational Intelligence*, 10:401–421.
- [Rosen, 2002] Rosen, E. (2002). Some aspects of model theory and finite structures. *Bulletin of Symbolic Logic*, 8(3):380 – 403.

BIBLIOGRAPHY

- [Rosenberg, 1986] Rosenberg, I. G. (1986). Minimal clones I: the five types. *Lectures in Universal Algebra (Proc. Conf. Szeged, 1983), Colloq. Math. Soc. J. Bolyai*, 43:405–427.
- [Rothmaler, 1995] Rothmaler, P. (1995). *Einführung in die Modelltheorie*. Spectrum Akademischer Verlag.
- [Schaeffer, 1978] Schaeffer, T. J. (1978). The complexity of satisfiability problems. In *Proceedings of the 10th Symposium on Theory of Computing (STOC'78)*, pages 216–226.
- [Schmerl, 1979] Schmerl, J. H. (1979). Countable homogeneous partially ordered sets. *Algebra Universalis*, 9:317–321.
- [Sleator and Tarjan, 1983] Sleator, D. and Tarjan, R. (1983). A data structure for dynamic trees. *Journal of Computer and System Sciences*, 26:362–390.
- [Steel, 1992] Steel, M. (1992). The complexity of reconstructing trees from qualitative characters and subtrees. *Journal of Classification*, 9:91–116.
- [Szabó, 1978] Szabó, L. (1978). Concrete representation of relational structures of universal algebras. *Acta Sci. Math. (Szeged)*, 40:175–184.
- [Szendrei, 1986] Szendrei, A. (1986). *Clones in universal Algebra*. Seminaire de mathematiques superieures. Les Presses de L'Universite de Montreal.
- [Tait, 1959] Tait, W. (1959). A counterexample to a conjecture of Scott and Suppes. *Journal of Symbolic Logic*, 24(1):15–16.
- [Thiel, 2004a] Thiel, S. (2004a). Efficient algorithms for constraint propagation and for processing tree descriptions. Dissertation, Max-Planck Institut, Saarbrücken.
- [Thiel, 2004b] Thiel, S. (2004b). A linear time algorithm for the configuration problem of dominance graphs. Submitted. Max-Planck Institut, Saarbrücken.
- [Thorup, 1999] Thorup, M. (1999). Decremental dynamic connectivity. *Journal of Algorithms*, 33(2):229–243.

Erklärung

Hiermit erkläre ich, daß

- ich die vorliegende Dissertationsschrift selbständig und ohne unerlaubte Hilfe verfaßt habe;
- ich mich nicht bereits anderwärts um einen Doktorgrad beworben habe oder einen solchen besitze;
- mir die Promotionsordnung der Mathematisch-Naturwissenschaftlichen Fakultät II der Humboldt-Universität zu Berlin bekannt ist.

Manuel Bodirsky

Manuel Bodirsky

Curriculum Vitae

Geboren am 30. Dezember 1976 in Freiburg im Breisgau, Deutschland.

Dienstadresse: Humboldt-Universität zu Berlin, Institut für Informatik,
Raum 3.410, Rudower Chausse 25, Berlin-Adlershof

Postanschrift: Unter den Linden 6, 10099 Berlin, Deutschland

Email: bodirsky@informatik.hu-berlin.de

Tel./Fax: +49 30 2093 3827/3191

FORSCHUNGSINTERESSEN

Constraint Satisfaction, Logik in der Informatik, endliche und unendliche Modelltheorie, deskriptive Komplexität, kombinatorische Spiele.

Zufällige Strukturen und deren effiziente Erzeugung, Komplexität von Zählproblemen, randomisierte Algorithmen. Planare Strukturen.

AUSBILDUNG

2001 - 2004: Promotionsstudent der Informatik an der Humboldt-Universität zu Berlin im europäischen Graduiertenkolleg *Combinatorics, Geometry and Computation* (Thema der Arbeit ist “Constraint Satisfaction with Infinite Domains”; Betreuer ist Prof. Hans Jürgen Prömel).

1997 - 2001: Diplom in Informatik, Universität des Saarlandes (Note 1,0; Thema der Diplomarbeit: “Beta Reduction Constraints”, betreut von Prof. Gerd Smolka und Dr. Joachim Niehren).

1996 - 1997: Zivildienst am Institut für medizinische Informatik, Freiburg.

1989 - 1996: Abitur am Martin Schongauer Gymnasium, Breisach. (Durchschnittsnote 1,2; Leistungskurse Mathematik und Physik).

FORSCHUNGSTÄTIGKEITEN

- 2001 - : Arbeitsgruppe Algorithmen und Komplexität, Institut für Informatik, Humboldt-Universität zu Berlin.
- 2002 - 2003: Sechsmonatiger Gastaufenthalt an der Karlsuniversität Prag, Tschechien.
- 1998 - 2000: Wissenschaftliche Hilftkraft im *Chorus Projekt*, Sonderforschungsbereich 378 *Resourcenadaptive Kognitive Prozesse*, Universität des Saarlandes.
- 1997 - 1998: Wissenschaftliche Hilfskraft am DFKI Saarbrücken, Projekt *Flag*, Prof. Hans Uszkoreit.

LEHRTÄTIGKEITEN

- SS 2003: Seminar *The Strange Logic of Random Graphs*, auf Englisch, zusammen mit Mihyun Kang.
- WS 2000: Tutor der Vorlesung *Algorithmen und Datenstrukturen*, Prof. Kurt Mehlhorn.
- SS 2000: Tutor der Vorlesung *Logik, Semantik und Verifikation*, Prof. Gerd Smolka.
- WS 1999: Tutor der Vorlesung *Formale Sprachen*, Prof. Günter Hotz.
- SS 1999: Tutor der Vorlesung *Algorithmen und Datenstrukturen*, Prof. Raimund Seidel.

FÖRDERUNG UND STIPENDIEN

- 2001 - 2004: Teilnehmer des Graduiertenkollgs *Combinatorics, Geometry, and Computation*, unterstützt durch die Deutsche Forschungsgemeinschaft (DFG).
- 1997 - 2001: Stipendiat der Studienstiftung des Deutschen Volkes.

PREISE UND AUSZEICHNUNGEN

- 2001: *Günter Hotz Medaille* für bestes Diplom in Informatik an der Universität des Saarlandes.
- 1996: Bundessieger im *Bundeswettbewerb Informatik*.

MITGLIEDSCHAFTEN

- Deutsche Mathematiker Vereinigung (DMV)
DMV Subject Group Discrete Mathematics
Gesellschaft für Informatik e.V. (GI)
Bundeswettbewerb Informatik, Alumni und Freunde e.V.
Freunde der Saarbrücker Informatik e.V.