

The Right-Hand Side Complexity of Constraint Satisfaction: A Short Course¹

Manuel Bodirsky², Hubie Chen³

August 13, 2007

¹This is the course script for the ESSLLI'2007 course *Complexity of Constraint Satisfaction*

²Abteilung Algorithmen und Komplexität, Humboldt-Universität zu Berlin,
bodirsky@informatik.hu-berlin.de

³Departament de Tecnologies de la Informació i les Comunicacions, Universitat Pompeu Fabra,
Barcelona, hubie.chen@upf.edu

Contents

1	Introduction	5
1.1	The Constraint Satisfaction Problem	5
1.2	Conjunctive Queries	7
1.3	Restricting The Right-Hand Side	8
1.4	Complexity	9
2	Logic and Algebra	13
2.1	A Quick Review of First-order Logic	13
2.2	Definability Notions and Preservation	14
2.3	pp-definability and the CSP	16
3	Clones	21
3.1	Basics	21
3.2	To Idempotence	23
3.3	The Boolean Case	24
4	Consistency Techniques	29
4.1	Hyperarc-Consistency	29
4.2	Local Consistency	33
4.3	Near-Unanimity Functions	36
5	Graph Algorithms	41
5.1	Temporal Reasoning	41
5.2	Phylogenetic Reconstruction	43
5.3	Branching Time Constraints	46

Chapter 1

Introduction

1.1 The Constraint Satisfaction Problem

An instance of the constraint satisfaction problem (CSP), intuitively, consists of a set of variables and a set of constraints on the variables. The question is to decide whether or not there is an assignment to the variables satisfying all of the constraints.

As an example, we could take the constraints

$$(u \neq v), (v \neq w), (u = w)$$

where both the disequality \neq and equality $=$ are interpreted over the two-element boolean domain $\{0, 1\}$. These constraints are satisfiable via the assignment

$$u = w = 0, v = 1$$

or alternatively, via the assignment

$$u = w = 1, v = 0$$

On the other hand, the constraints

$$(u \neq v), (v \neq w), (u \neq w)$$

cannot all be satisfied simultaneously, at least under our assumption that the relations \neq and $=$ are over the two-element domain $\{0, 1\}$.

We will see that there are a number of different formulations of the CSP, which indeed attests to its generality and usefulness! Our first and official definition of the CSP will be the following “relational homomorphism problem”.

We require some terminology. A *(relational) signature* σ is a set of relation symbols; for our purposes in these notes, a signature will always be a *finite* set of relation symbols. Each relation symbol $R \in \sigma$ has an associated *arity* $\text{ar}(R) \geq 1$.

A *relational structure* \mathbf{B} over σ consists of a *universe* B , which is simply a set, along with a relation $R^{\mathbf{B}} \subseteq B^{\text{ar}(R)}$ for each relation symbol $R \in \sigma$. The relation $R^{\mathbf{B}}$, which is a relation of arity $\text{ar}(R)$ over B , provides an interpretation for the symbol R .

When \mathbf{A} and \mathbf{B} are two relational structures over the same relational signature σ , we say that a mapping $h : A \rightarrow B$ from the universe A of \mathbf{A} to the universe B of \mathbf{B} is a *homomorphism* from \mathbf{A} to \mathbf{B} if for every relation symbol $R \in \sigma$ and every tuple $(a_1, \dots, a_k) \in R^{\mathbf{A}}$, it holds that $(h(a_1), \dots, h(a_k)) \in R^{\mathbf{B}}$. That is, for each relation symbol R , mapping a tuple in $R^{\mathbf{A}}$ under h results in a tuple in $R^{\mathbf{B}}$.

Definition 1.1.1 *The constraint satisfaction problem (CSP) is the following. Given two relational structures \mathbf{A} and \mathbf{B} over the same relational signature σ , does there exist a homomorphism from \mathbf{A} to \mathbf{B} ?*

Note that we will sometimes denote an instance of the CSP using the pair notation (\mathbf{A}, \mathbf{B}) . In working with relational structures, we will make use of the following conventions.

- We will say that a relational structure is *finite* if its universe is of finite size, and we will say that a relational structure is *infinite* if its universe is of infinite size.
- Typically, we will use boldface letters $\mathbf{A}, \mathbf{B}, \dots$ to denote relational structures, and the corresponding capital letters A, B, \dots to denote their universes.

As a first example of a CSP instance, let us take consider the first set of constraints given above:

$$(u \neq v), (v \neq w), (u = w).$$

We may view this as an instance of the CSP in the following way. Let τ be the relational signature $\{D, S\}$ where $\text{ar}(D) = \text{ar}(S) = 2$. Let \mathbf{T} be the relational structure with universe $T = \{0, 1\}$, $D^{\mathbf{T}} = \{(0, 1), (1, 0)\}$, and $S^{\mathbf{T}} = \{(0, 0), (1, 1)\}$. That is, $D^{\mathbf{T}}$ is the disequality relation on T , and $S^{\mathbf{T}}$ is the equality relation on T .

Let \mathbf{A} be the relational structure with universe $A = \{u, v, w\}$, $D^{\mathbf{A}} = \{(u, v), (v, w)\}$ and $S^{\mathbf{A}} = \{(u, w)\}$.

It may be verified that the problem of satisfying the given constraints is precisely the homomorphism problem on (\mathbf{A}, \mathbf{T}) . Indeed, the homomorphisms from \mathbf{A} to \mathbf{T} are precisely the assignments which satisfy the constraints. Intuitively, the universe A of \mathbf{A} consists of variables; the tuples (u, v) and (v, w) in $D^{\mathbf{A}}$ state that “ (u, v) and (v, w) must take on values in $D^{\mathbf{T}}$ ”, while the tuple (u, w) in $S^{\mathbf{A}}$ states that “ (u, w) must take on a value in $S^{\mathbf{T}}$ ”.

As another example, consider $\tau = \{E\}$ where E is of arity 2. Let \mathbf{K}_3 be the relational structure with universe $K_3 = \{1, 2, 3\}$ and where $E^{\mathbf{K}_3}$ is the disequality relation on $\{1, 2, 3\}$, that is, $E^{\mathbf{K}_3}$ is the set

$$\{(1, 2), (1, 3), (2, 1), (2, 3), (3, 1), (3, 2)\}.$$

Similarly, let \mathbf{K}_4 be the relational structure with universe $K_4 = \{1, 2, 3, 4\}$ and where $E^{\mathbf{K}_4}$ is the disequality relation on $\{1, 2, 3, 4\}$, that is, $E^{\mathbf{K}_4}$ is the set

$$\{(1, 2), (1, 3), (1, 4), (2, 1), (2, 3), (2, 4), (3, 1), (3, 2), (3, 4), (4, 1), (4, 2), (4, 3)\}.$$

The mapping $h : K_3 \rightarrow K_4$ where $h(1) = 1$, $h(2) = 2$ and $h(3) = 3$ is a homomorphism from \mathbf{K}_3 to \mathbf{K}_4 . Indeed, one can verify that any injective mapping $h : K_3 \rightarrow K_4$ is a homomorphism from \mathbf{K}_3 to \mathbf{K}_4 , and conversely that any homomorphism from \mathbf{K}_3 to \mathbf{K}_4 is an injective mapping.

On the other hand, there exists no homomorphism from \mathbf{K}_4 to \mathbf{K}_3 . (Why?)

In the following exercise, we will see that the notion of homomorphism naturally yields a preorder on the set of all relational structures over a fixed signature.

Exercise 1.1.2 *Fix σ to be a relational signature. Consider the relation \leq_σ defined on relational structures over σ where, for two such structures \mathbf{A} and \mathbf{B} ,*

$$\mathbf{A} \leq_\sigma \mathbf{B}$$

if and only if there exists a homomorphism from \mathbf{A} to \mathbf{B} . Prove that the relation \leq_σ is reflexive and transitive, but not symmetric.

We now look at the computational problem *3-colorability* and show how to formulate instances of it as instances of the CSP. An instance of 3-colorability is an undirected graph $G = (V_G, E_G)$. Recall that an undirected graph consists of a set of vertices V_G and a set of edges E_G ; every edge is a two-element set of vertices. As an example, consider the graph G with vertex set $V_G = \{a, b, c, d\}$ and $E_G = \{\{a, b\}, \{a, c\}, \{b, c\}, \{b, d\}, \{c, d\}\}$.

The problem 3-colorability is to decide, given an undirected graph G , if G is 3-colorable: if there exists a mapping defined on the vertices V_G of G to a 3-element set such that the two vertices of every edge $\{u, v\}$ in G are mapped to different values (or “colors”). Here, we take $\{1, 2, 3\}$ as our 3-element set of colors, and so the question is whether there exists a mapping $f : V_G \rightarrow \{1, 2, 3\}$ such that for every edge $\{u, v\} \in E_G$, it holds that $f(u) \neq f(v)$.

We claim that any instance $G = (V_G, E_G)$ of 3-colorability can be viewed as the CSP instance $(\mathbf{A}_G, \mathbf{K}_3)$ over signature $\sigma = \{E\}$ where \mathbf{A}_G is the relational structure with universe V_G and $E^{\mathbf{A}_G}$ contains an ordered pair (u, v) for every edge $\{u, v\} \in E_G$. (Note that, for our purposes here, the order in which the vertices of an edge are given in $E^{\mathbf{A}_G}$ is not important.) By definition, the CSP instance $(\mathbf{A}_G, \mathbf{K}_3)$ is a “yes” instance if and only if there exists a mapping $h : V \rightarrow \{1, 2, 3\}$ such that the following condition holds:

$$\text{for every pair } (u, v) \in E^{\mathbf{A}_G}, \text{ it holds that } (h(u), h(v)) \in E^{\mathbf{K}_3}.$$

As $E^{\mathbf{K}_3}$ is defined to be the disequality relation on $\{1, 2, 3\}$, and h is a mapping into $\{1, 2, 3\}$, this condition is equivalent to:

$$\text{for every pair } (u, v) \in E^{\mathbf{A}_G}, \text{ it holds that } h(u) \neq h(v).$$

It is readily seen that a mapping $h : V \rightarrow \{1, 2, 3\}$ satisfies this condition if and only if it is a 3-coloring of the graph G . Indeed, this last condition is almost exactly the definition of a 3-coloring, except in $E^{\mathbf{A}_G}$ the edges have been given (arbitrary) orientations.

1.2 Conjunctive Queries

We will now see some alternative ways of formulating the CSP; precisely, we will see some alternative ways of describing the condition of having a homomorphism from a structure \mathbf{A} to another structure \mathbf{B} .

To every relational structure \mathbf{A} (over σ) we can associate a logical formula called the *canonical conjunctive query* of \mathbf{A} , denoted by $Q_{\mathbf{A}}$. The formula $Q_{\mathbf{A}}$ consists of a quantifier prefix in which all elements of A are existentially quantified, followed by the conjunction of formulas $R(a_1, \dots, a_k)$ over all relation symbols R (of σ) and tuples $(a_1, \dots, a_k) \in R^{\mathbf{A}}$. For example, the canonical conjunctive query $Q_{\mathbf{A}}$ of \mathbf{A} (given above) is the formula

$$\exists u \exists v \exists w (D(u, v) \wedge D(v, w) \wedge S(u, w)).$$

Theorem 1.2.1 (*Chandra/Merlin*) *Let \mathbf{A} and \mathbf{B} be relational structures over the same relational signature σ . The following are equivalent:*

1. *There exists a homomorphism from \mathbf{A} to \mathbf{B} .*
2. *For all structures \mathbf{C} such that $\mathbf{C} \models Q_{\mathbf{B}}$, it holds that $\mathbf{C} \models Q_{\mathbf{A}}$.*
3. $\mathbf{B} \models Q_{\mathbf{A}}$

We give the idea of the proof, and leave a full verification as an exercise to the reader.

Proof. (idea) First, we observe that (1) and (3) are equivalent. If (1) holds and there is a homomorphism h from \mathbf{A} to \mathbf{B} , then the mapping h can be used to assign values to the existentially quantified variables of $Q_{\mathbf{A}}$ so that the quantifier-free part of $Q_{\mathbf{A}}$ is true. Conversely, if (3) holds, an assignment to the existentially quantified variables of $Q_{\mathbf{A}}$ satisfying the quantifier-free part can be verified to be a homomorphism.

We now show that (1) implies (2), and that (2) implies (3).

(1) \Rightarrow (2): Let \mathbf{C} be a structure such that $\mathbf{C} \models Q_{\mathbf{B}}$. By the equivalence of (1) and (3), there is a homomorphism from \mathbf{B} to \mathbf{C} , and we want to show that there is a homomorphism from \mathbf{A} to \mathbf{C} . By assumption, there is a homomorphism from \mathbf{A} to \mathbf{B} , and we obtain the desired homomorphism (from \mathbf{A} to \mathbf{C}) by transitivity of the relation \leq_{σ} ; see Exercise 1.1.2.

(2) \Rightarrow (3): This is straightforward from the observation that $\mathbf{B} \models Q_{\mathbf{B}}$. Notice that by the equivalence of (1) and (3), this observation amounts to showing that there is a homomorphism from \mathbf{B} to itself. This may be viewed as the reflexivity of the relation \leq_{σ} ; see Exercise 1.1.2.

□

1.3 Restricting The Right-Hand Side

The CSP, in its general formulation, is computationally intractable (in a sense to be made precise soon). A classic approach to coping with a computational intractable problem is to consider restricted versions of the problem. In this course, we will consider restricted versions of the CSP where the “right-hand side structure” \mathbf{B} is fixed.

Definition 1.3.1 *Let \mathbf{B} be a relational structure over signature σ . The computational problem $\text{CSP}(\mathbf{B})$ is the following. Given a finite relational structure \mathbf{A} over σ , does there exist a homomorphism from \mathbf{A} to \mathbf{B} ?*

We should emphasize that *each* relational structure \mathbf{B} gives rise to a computational problem $\text{CSP}(\mathbf{B})$. An input to $\text{CSP}(\mathbf{B})$ is a *single* relational structure \mathbf{A} , and *not* a pair of relational structures (as with the CSP).

This is clearly one of the simplest restrictions on the CSP that one can think of. Interestingly, it allows one to place into a unified framework many computational problems that have been studied independently. It also gives rise to a rich and multi-faceted theory that is fascinating in its own right.

Consider the following examples of problems $\text{CSP}(\mathbf{B})$.

- The problem $\text{CSP}(\mathbf{T})$ may be identified with the problem of deciding, given a set of equalities and disequalities over variables (as in the beginning of this chapter), whether or not they are satisfiable over $\{0, 1\}$.
- The problem $\text{CSP}(\mathbf{K}_3)$, as discussed, may be identified with the problem 3-colorability.
- Consider the relational structure \mathbf{PA} over signature $\{\neq, \leq\}$ with universe \mathbb{Q} and the usual interpretations to \neq and \leq . (Here, as usual, \mathbb{Q} denotes the set of rational numbers.) The problem $\text{CSP}(\mathbf{PA})$ can then be identified with the problem of deciding, given a set of disequalities $u \neq v$ and inequalities $u \leq v$ over variables, whether or not there is

an assignment to the variables over \mathbb{Q} such that all of the disequalities and inequalities are satisfied. For instance, one might consider

$$a \leq b, a \leq c, b \leq d, c \leq d, b \neq c.$$

(Exercises: is this a “yes” instance? How would one view these disequalities and inequalities as an instance \mathbf{A} of $\text{CSP}(\mathbf{PA})$?)

We always assume that an input of a problem $\text{CSP}(\mathbf{B})$ is a *finite* relational structure, that is, a relational structure with finite universe. Notice, however, that there is no problem if the fixed structure \mathbf{B} has infinite universe, as in the last example of $\text{CSP}(\mathbf{PA})$! Again, the fixed structure is not given as part of the input; an input to $\text{CSP}(\mathbf{PA})$, for instance, is simply a finite relational structure over the signature $\{\neq, \leq\}$.

Exercises for Section 1.3

Exercise 1.3.2 *The 2-SAT problem is to decide, given a set of 2-clauses over propositional (0-1 valued) variables, whether or not there is a satisfying assignment, that is, an assignment to the variables under which all of the clauses are true. A clause is a disjunction of literals, where a literal is either a variable or its negation. By a 2-clause, we mean a clause with exactly two literals.*

An example instance of 2-SAT is the set of clauses:

$$\{u \vee v, \neg u \vee \neg v, \neg u \vee v, \neg v \vee w\}$$

Give a relational structure \mathbf{B}_2 such that 2-SAT can be identified with the problem $\text{CSP}(\mathbf{B}_2)$. In particular, you should give a structure \mathbf{B} such that an arbitrary instance of 2-SAT can be readily translated to an instance \mathbf{A} of $\text{CSP}(\mathbf{B}_2)$ where the homomorphisms from \mathbf{A} to \mathbf{B}_2 are exactly the satisfying assignments of the 2-SAT instance.

Do the same for 3-SAT: give a relational structure \mathbf{B}_3 such that 3-SAT can be identified with the problem $\text{CSP}(\mathbf{B}_3)$. The 3-SAT problem is defined just as the 2-SAT problem, except an instance of 3-SAT is a set of 3-clauses. (A 3-clause is a clause with exactly three literals.)

Exercise 1.3.3 *A 3-uniform hypergraph consists of a (finite) set of vertices V and a set of edges E , where each edge $e \in E$ is a size 3 subset of V . Let us say that a mapping from f from V to a three-element set is a rainbow-constant coloring if for each edge $e = \{u, v, w\}$ in E , either:*

- *the vertices are mapped to different values (the edge is a “rainbow”), that is, $f(u) \neq f(v)$, $f(u) \neq f(w)$, and $f(v) \neq f(w)$ all hold; or,*
- *the vertices are mapped to the same value (the edge is “constant”), that is, $f(u) = f(v) = f(w)$.*

Describe how the the following problem may be viewed as a problem of the form $\text{CSP}(\mathbf{B})$. An instance of the problem is a 3-uniform hypergraph along with a mapping $p : S \rightarrow \{\text{red, green, blue}\}$ defined on a subset $S \subseteq V$ of the vertices, and the question is to decide if there is a rainbow-constant coloring $f : V \rightarrow \{\text{red, green, blue}\}$ of the hypergraph that extends the given mapping p . In other words, one is given a “pre-coloring” p of a 3-uniform hypergraph, and the question is to decide if this pre-coloring can be extended to a full rainbow-constant coloring.

1.4 Complexity

In this course, we will be primarily interested in distinguishing between problems that are in P, and problems that are NP-complete. Recall that P is the set of all decision problems for which

there exists an algorithm solving the problem in polynomial time (for a reasonable formulation of the notion of “algorithm”), and, intuitively, NP is the set of all decision problems for which there exists a polynomial-time algorithm that can *verify* a solution to the problem.

A useful tool for studying the complexity of problems is the notion of *reduction*. Recall that a decision problem Φ reduces to a decision problem Φ' (denoted $\Phi \leq_p \Phi'$) if there exists a polynomial-time computable function f such that:

- if ϕ is a “yes”-instance of Φ , then $f(\phi)$ is a “yes”-instance of Φ' , and
- if ϕ is a “no”-instance of Φ , then $f(\phi)$ is a “no”-instance of Φ' .

We remark that this form of reduction is known as *many-one polynomial-time reduction*; many other notions of reductions have been studied, but (unless otherwise specified) this is the one that will be used in this course. We will write $\Phi \equiv_p \Phi'$ to indicate that $\Phi \leq_p \Phi'$ and $\Phi' \leq_p \Phi$.

Recall the following definitions. A problem Ψ is *NP-hard* if for all problems Φ in NP, it holds that $\Phi \leq_p \Psi$. A problem Ψ is *NP-complete* if it is in NP and it is NP-hard. Also recall the following fact, which we will use throughout.

Proposition 1.4.1 *If Φ is NP-hard and $\Phi \leq_p \Phi'$, then Φ' is NP-hard.*

We point out that all problems of the form $\text{CSP}(\mathbf{B})$ are in NP, when looking at finite relational structures \mathbf{B} .

Proposition 1.4.2 *For every finite relational structure \mathbf{B} , the problem $\text{CSP}(\mathbf{B})$ is in NP.*

The following give some well-known examples of problems $\text{CSP}(\mathbf{B})$ that are NP-hard.

Proposition 1.4.3 *The 3-SAT problem is NP-complete.*

Proposition 1.4.4 *The 3-colorability problem, or equivalently, the problem $\text{CSP}(\mathbf{K}_3)$, is NP-complete.*

The following conjecture concerning the complexity of problems $\text{CSP}(\mathbf{B})$ has been proposed.

Conjecture (Feder/Vardi): For every relational structure \mathbf{B} with a finite universe, the problem $\text{CSP}(\mathbf{B})$ is either in P or is NP-complete.

This conjecture has been established for some restricted classes of relational structures, for instance, for structures \mathbf{B} with universe size two (Schaefer, 1978). Such dichotomous behavior cannot be taken for granted, in light of the following result from complexity theory.

Theorem 1.4.5 (Ladner) *If P does not equal NP, then there exist languages $L \in \text{NP} \setminus P$ that are not NP-complete.*

That is, Ladner’s theorem states that, under the assumption P not equal NP, there are languages in NP of “intermediate” complexity: they are neither tractable (in P) nor maximally intractable (NP-complete).

Exercises for Section 1.4

Exercise 1.4.6 Show that $\text{CSP}(\mathbf{T})$ is in P .

Exercise 1.4.7 Define the relational structure \mathbf{B}^∞ over signature $\{=, \neq\}$ in the following way. The universe of \mathbf{B}^∞ is $B = \{1, 2, 3, \dots\}$, that is, the set of natural numbers. The relation $=^{\mathbf{B}^\infty}$ is the equality relation on B , and the relation $\neq^{\mathbf{B}^\infty}$ is the disequality relation on B . Show that $\text{CSP}(\mathbf{B}^\infty)$ is in P .

Note: this result can be compared with the previous exercise as well as the fact that the disequality relation over a 3-element set is NP-hard (Proposition 1.4.4).

Exercise 1.4.8 Is the “rainbow-constant coloring” problem described in Exercise 1.3.3 in P , or NP-complete?

Exercise 1.4.9 In this course, we focus on studying the complexity of the problems $\text{CSP}(\mathbf{B})$. One can similarly study the “dual restriction” where the “left-hand side structure” \mathbf{A} is restricted. That is, for a relational structure \mathbf{A} over signature σ , define $\mathbf{A}\text{-CSP}$ to be the problem of deciding, given a finite relational structure \mathbf{B} over σ , if there exists a homomorphism from \mathbf{A} to \mathbf{B} .

This family of problems is less interesting from a complexity-theoretic standpoint: prove that for every finite relational structure \mathbf{A} , the problem $\mathbf{A}\text{-CSP}$ is in P .

We can mention that, although restricting the left-hand side to a single relational structure is not so interesting, researchers have studied restricting the left-hand side relative to an infinite set of relational structures. This set restriction has given rise to a rich theory, in which graph parameters such as treewidth (and extensions thereof) play a crucial role.

Chapter 2

Logic and Algebra

2.1 A Quick Review of First-order Logic

We now briefly review the portion of first-order logic that we will need. Let σ be a relational signature.

Definition 2.1.1 *An atomic formula of σ is an equality expression ($v = v'$) for variables v, v' , or an expression having the form $R(v_1, \dots, v_k)$ where $R \in \sigma$ and v_1, \dots, v_k are variables.*

Definition 2.1.2 *The first-order formulas of σ are the smallest set of expressions containing the atomic formulas (of σ) and closed under the following operations:*

- if ϕ and ψ are first-order formulas, then the negation $\neg\phi$ of ϕ , the conjunction $(\phi \wedge \psi)$ of ϕ and ψ , and the disjunction $(\phi \vee \psi)$ of ϕ and ψ are first-order formulas.
- if ϕ is a first-order formula and v is a variable, then the existentially quantified formula $(\exists v\phi)$ and universally quantified formula $(\forall v\phi)$ are first-order formulas.

Here, we do not formally define that semantics of first-order logic. We assume that the reader has previous exposure to this definition, and refer the reader to a textbook on first-order logic for details.

Let \mathbf{B} be a relational structure over signature σ . We say that a relation $\mathcal{R} \subseteq B^k$ of arity k is *first-order definable* over \mathbf{B} if there exists a first-order formula $\phi(v_1, \dots, v_k)$ of σ with free variables v_1, \dots, v_k such that

$$\mathcal{R} = \{(b_1, \dots, b_k) \in B^k \mid \mathbf{B} \models \phi(b_1, \dots, b_k)\}$$

Example 2.1.3 *Let σ be the signature $\{<\}$, where the symbol $<$ has arity 2. Let \mathbf{B} be the structure over signature σ with universe \mathbb{Q} and where $<$ has its usual interpretation, that is, $<^{\mathbf{B}}$ is the relation of all pairs $(a, b) \in \mathbb{Q}^2$ where a is strictly less than b .*

Consider the relation $W \subseteq \mathbb{Q}^3$ containing all triples (a, b, c) where a and c are distinct, and b falls strictly inbetween a and b . Formally, $W = \{(a, b, c) \mid (a < b < c) \text{ or } (c < b < a)\}$.

Let $\phi(v_1, v_2, v_3)$ be the formula

$$((v_1 < v_2) \wedge (v_2 < v_3)) \vee ((v_3 < v_2) \wedge (v_2 < v_1))$$

The formula ϕ gives a first-order definition of W over \mathbf{B} .

Example 2.1.4 Let \mathbf{B} be a relational structure over the signature $\{E\}$, where the symbol E has arity 2. We view \mathbf{B} as a directed graph: its universe is the set of vertices, and $E^{\mathbf{B}}$ is the set of directed edges.

The following formula $\phi(v_1, v_2)$ is a first-order definition of all pairs of vertices that share a common out-neighbor. (By an out-neighbor of a vertex a , we mean a vertex b such that (a, b) is an edge.)

$$\phi(v_1, v_2) = \exists u(E(v_1, u) \wedge E(v_2, u))$$

2.2 Definability Notions and Preservation

We will now look at a way to study logical definability using algebra, by studying the operations that preserve relations—for notions of preservation that we will make precise. In particular, we will see results showing that the operations preserving a structure also preserve relations defined by the structure.

Our first result of this form concerns the following type of preservation. Let us say that a unary operation $f : B \rightarrow B$ is an *automorphism* of a relation $\mathcal{R} \subseteq B^k$ if it is a bijection, and for any tuple $(b_1, \dots, b_k) \in B^k$, it holds that

$$(b_1, \dots, b_k) \in \mathcal{R} \Leftrightarrow (f(b_1), \dots, f(b_k)) \in \mathcal{R}$$

We say that a unary operation $f : B \rightarrow B$ is an *automorphism* of a relational structure \mathbf{B} if it is an automorphism of every relation of \mathbf{B} .

Proposition 2.2.1 *If a relation \mathcal{R} is a first-order definable over a relational structure \mathbf{B} , then every automorphism of \mathbf{B} is an automorphism of \mathcal{R} .*

Proof. Exercise. \square

Example 2.2.2 Consider the relational structure \mathbf{B} over $\sigma = \{\neq\}$ with universe $B = \mathbb{Q}$ and $\neq^{\mathbf{B}}$ the disequality relation on \mathbb{Q} , that is, $\neq^{\mathbf{B}}$ contains exactly those tuples $(a, b) \in \mathbb{Q} \times \mathbb{Q}$ such that a does not equal b . It is readily verified that every bijection $f : \mathbb{Q} \rightarrow \mathbb{Q}$ is an automorphism of \mathbf{B} .

From Proposition 2.2.1, we can show that the relation \leq , interpreted naturally over \mathbb{Q} , is not first-order definable over \mathbf{B} . Let $\pi : \mathbb{Q} \rightarrow \mathbb{Q}$ be any bijection that swaps 3 and 4, that is, such that $\pi(3) = 4$ and $\pi(4) = 3$. As π is a bijection, it is an automorphism of \mathbf{B} (as we observed). However, the relation \leq does not have π as an automorphism: it holds that $(3, 4)$ is in \leq , but it does not hold that $(\pi(3), \pi(4)) = (4, 3)$ is in \leq ! From the proposition, we can thus infer that \leq is not first-order definable over \mathbf{B} .

We now consider a notion of definability that is more restrictive than first-order definability. We say that a relation \mathcal{R} is *existential positive definable* over a relational structure \mathbf{B} if it is first-order definable over \mathbf{B} via a formula that uses only the operators \wedge and \vee , and the quantifier \exists .

Clearly, Proposition 2.2.1 holds with “existential positive definable” in place of “first-order definable”, since existential positive definability is a special case of first-order definability. However, we can show that for this new notion of definability, defined relations are preserved

by a type of operation more general than automorphism. We define an *endomorphism* of a relation $\mathcal{R} \subseteq B^k$ to be a unary operation $f : B \rightarrow B$ such that

$$(b_1, \dots, b_k) \in \mathcal{R} \Rightarrow (f(b_1), \dots, f(b_k)) \in \mathcal{R}$$

Note that the definition of endomorphism differs from that of automorphism in two ways: first, an endomorphism need not be a bijection, and second, only one direction of the given implication need hold. As before, we say that an operation f is an *endomorphism* of a structure \mathbf{B} if it is an endomorphism of all relations of \mathbf{B} .

We have the following proposition relating existential positive definability to endomorphisms.

Proposition 2.2.3 *If a relation \mathcal{R} is existential positive definable over a relational structure \mathbf{B} , then every endomorphism of \mathbf{B} is an endomorphism of \mathcal{R} .*

Proof. (idea) In this proof, we will speak of endomorphisms of a first-order formula; in doing so, we refer to endomorphisms of the relation defined by the first-order formula.

The proof is by induction on the structure of the existential positive formula $\phi(v_1, \dots, v_k)$ that defines \mathcal{R} . The base case, in which ϕ is simply an atomic formula, is straightforward. We consider the inductive cases.

Case: ϕ is of the form $\psi \wedge \psi'$. Even if ψ does not contain all of v_1, \dots, v_k as free variables, we may view it as if it does—without changing its endomorphisms—and likewise for ψ' . So, we assume that f is an endomorphism of both $\psi(v_1, \dots, v_k)$ and $\psi'(v_1, \dots, v_k)$, and want to show that f is an endomorphism of $\psi(v_1, \dots, v_k) \wedge \psi'(v_1, \dots, v_k)$. Let $(b_1, \dots, b_k) \in B^k$ be a tuple such that $\phi(b_1, \dots, b_k)$ holds. Then, both $\psi(b_1, \dots, b_k)$ and $\psi'(b_1, \dots, b_k)$ hold. Since f is an endomorphism of both ψ and ψ' , we have that both $\psi(f(b_1), \dots, f(b_k))$ and $\psi'(f(b_1), \dots, f(b_k))$ hold, and it follows that $\phi(f(b_1), \dots, f(b_k))$ holds, as desired.

Case: ϕ is of the form $\psi \vee \psi'$. As in the previous case, we view ψ and ψ' as if they contain all of v_1, \dots, v_k as free variables. Suppose that $(b_1, \dots, b_k) \in B^k$ is a tuple such that $\phi(b_1, \dots, b_k)$ holds. Then either $\psi(b_1, \dots, b_k)$ or $\psi'(b_1, \dots, b_k)$ holds. In the first case, we have that $\psi(f(b_1), \dots, f(b_k))$ holds; in the second case, we have that $\psi'(f(b_1), \dots, f(b_k))$ holds. It follows that $\psi(f(b_1), \dots, f(b_k)) \vee \psi'(f(b_1), \dots, f(b_k)) = \phi(f(b_1), \dots, f(b_k))$ holds.

Case: ϕ is of the form $\exists v \psi(v_1, \dots, v_k, v)$. Suppose that $(b_1, \dots, b_k) \in B^k$ is a tuple such that $\phi(b_1, \dots, b_k)$ holds. Then, there exists a value $b \in B$ such that $\psi(b_1, \dots, b_k, b)$ holds. It follows that $\psi(f(b_1), \dots, f(b_k), f(b))$ holds, which in turn implies that $\phi(f(b_1), \dots, f(b_k))$ holds. \square

Exercise 2.2.4 *Let \mathbf{B} be a finite relational structure. Prove that a unary function $f : B \rightarrow B$ is an automorphism of \mathbf{B} if and only if it is a bijection and an endomorphism of \mathbf{B} .*

We now consider a third and yet more restrictive notion of definability. Say that a relation \mathcal{R} is *primitive positive definable* (for short, *pp-definable*) over a relational structure \mathbf{B} if it is first-order definable over \mathbf{B} via a formula that uses only the operator \wedge and the quantifier \exists .

The following type of operation can be used to study pp-definability. An operation $f : B^n \rightarrow B$ is a *polymorphism* of a relation $\mathcal{R} \subseteq B^k$ if for any choice of n tuples $(b_{11}, \dots, b_{1k}), \dots, (b_{n1}, \dots, b_{nk}) \in \mathcal{R}$, it holds that $(f(b_{11}, \dots, b_{n1}), \dots, f(b_{1k}, \dots, b_{nk}))$ is in \mathcal{R} . That is, applying f coordinate-wise to any n tuples in \mathcal{R} yields another tuple in \mathcal{R} . When f is a polymorphism of a relation \mathcal{R} , we also say that \mathcal{R} is *preserved* by f . We say that an operation f is a

polymorphism of a structure \mathbf{B} if f is a polymorphism of all relations of \mathbf{B} ; in this case, we also say that \mathbf{B} is *preserved* by f .

One may observe that an endomorphism is an arity 1 polymorphism.

Proposition 2.2.5 *If a relation \mathcal{R} is a pp-definable over a relational structure \mathbf{B} , then \mathcal{R} is preserved by every polymorphism of \mathbf{B} . (That is, every polymorphism of \mathbf{B} is a polymorphism of \mathcal{R} .)*

Proof. Exercise. \square

Let π_i^n be the n -ary *projection* operation that always returns its i th coordinate, that is,

$$\pi_i^n(x_1, \dots, x_n) = x_i$$

for all x_1, \dots, x_n . It is straightforward to verify that such projections are polymorphisms of any relation (exercise!).

From the standpoint of the definition of polymorphism, then, projections are trivial in that they are polymorphisms of any relation. We also have relations that are trivial in that they are preserved by any operations: the family of relations $\mathcal{R}_k = \{(b, \dots, b) \in B^k \mid b \in B\}$. That is, \mathcal{R}_k is the k -arity relation containing those tuples whose entries are all identical. Note that \mathcal{R}_2 is simply the equality relation on B .

Further Exercises for Section 2.2

Exercise 2.2.6 *What are the endomorphisms of \mathbf{K}_3 ?*

Exercise 2.2.7 *For two relational structures \mathbf{A} and \mathbf{B} over the same signature σ , define the product $\mathbf{A} \times \mathbf{B}$ to be the relational structure over σ having universe $A \times B$ and where*

$$R^{\mathbf{A} \times \mathbf{B}} = \{((a_1, b_1), \dots, (a_k, b_k)) \mid (a_1, \dots, a_k) \in R^{\mathbf{A}}, (b_1, \dots, b_k) \in R^{\mathbf{B}}\}$$

for all $R \in \sigma$. Show that an operation $f : B^k \rightarrow B$ is a polymorphism of a relational structure \mathbf{B} if and only if it is a homomorphism from \mathbf{B}^k to \mathbf{B} .

Exercise 2.2.8 *Let us say that a relation \mathcal{R} is few-definable over a structure \mathbf{B} if it can be defined using conjunction, universal quantification, and existential quantification. Prove the following: if a relation \mathcal{R} is few-definable over a relational structure \mathbf{B} , then \mathcal{R} is preserved by every surjective polymorphism of \mathbf{B} .*

Here, we mean that a polymorphism $f : B^k \rightarrow B$ is surjective if for all $b \in B$, there exists $(b_1, \dots, b_k) \in B^k$ such that $f(b_1, \dots, b_k) = b$.

2.3 pp-definability and the CSP

The notion of pp-definability has direct relevance to studying the class of problems $\text{CSP}(\mathbf{B})$. In this section, we first show how this notion can be applied to the study of CSP complexity, and then give an exact algebraic characterization of pp-definability.

For a relational structure \mathbf{B} , we use $\langle \mathbf{B} \rangle$ to denote the set of all relations pp-definable from \mathbf{B} .

Proposition 2.3.1 *Let \mathbf{B} , \mathbf{B}' be relational structures. If every relation of \mathbf{B} has a pp-definition over \mathbf{B}' , then $\text{CSP}(\mathbf{B}) \leq_p \text{CSP}(\mathbf{B}')$.*

Proof. (idea) Let \mathbf{A} be an instance of $\text{CSP}(\mathbf{B})$. Following Theorem 1.2.1, we view $\text{CSP}(\mathbf{B})$ as the problem of deciding whether or not $\mathbf{B} \models Q_{\mathbf{A}}$. We transform $Q_{\mathbf{A}}$ into an instance $Q_{\mathbf{A}'}$ of $\text{CSP}(\mathbf{B}')$ as follows. For each atomic formula $R(v_1, \dots, v_k)$ that appears in $Q_{\mathbf{A}}$, replace it with a pp-formula over \mathbf{B}' that defines $R^{\mathbf{B}}$. After this has been done, the formula can be transformed into a formula $Q_{\mathbf{A}'}$ that is in prenex form (that is, having a sequence of quantifiers followed by a quantifier-free part) by a standard procedure. After this, equalities can be eliminated one by one: for each equality $u = v$, pick one of the variables arbitrarily (say, u), remove v from the quantifier prefix, and then replace all remaining instances of v with u . \square

As an simple example, consider the instance

$$Q_{\mathbf{A}} = \exists u \exists v \exists w (D(u, v) \wedge D(v, w) \wedge S(u, w))$$

of $\text{CSP}(\mathbf{T})$.

Let \mathbf{B}' be the structure over signature $\{E\}$, where E has arity 2, with universe $B' = \{0, 1\}$ and such that $R^{\mathbf{B}'} = \{0, 1\}^2 \setminus \{(0, 1)\}$. Each relation of \mathbf{T} has a pp-definition over \mathbf{B}' : the relation $D^{\mathbf{T}} = \{(0, 1), (1, 0)\}$ can be defined as

$$\phi(v_1, v_2) \equiv E(v_1, v_2) \wedge E(v_2, v_1)$$

over \mathbf{B}' , and the relation $S^{\mathbf{T}} = \{(0, 0), (1, 1)\}$ can be defined as

$$\phi(v_1, v_2) \equiv (v_1 = v_2)$$

over \mathbf{B}' .

We may convert $Q_{\mathbf{A}}$ as follows. Replacing the relations appearing therein with defining formulas over \mathbf{B}' , we obtain

$$\exists u \exists v \exists w ((E(u, v) \wedge E(v, u)) \wedge (E(v, w) \wedge E(w, v)) \wedge (u = w))$$

Removing the equality, we replace all instances of w with u to obtain

$$\exists u \exists v (E(u, v) \wedge E(v, u) \wedge E(v, u) \wedge E(u, v))$$

Corollary 2.3.2 *Let \mathbf{B} , \mathbf{B}' be relational structures. If \mathbf{B} and \mathbf{B}' both pp-define the same relations (that is, if $\langle \mathbf{B} \rangle = \langle \mathbf{B}' \rangle$), then $\text{CSP}(\mathbf{B}) \equiv_p \text{CSP}(\mathbf{B}')$.*

Proof. Note that, trivially, every relation of \mathbf{B} has a pp-definition over \mathbf{B} (and, likewise for \mathbf{B}'). The corollary is then immediate from Proposition 2.3.1. \square

The following theorem shows that preservation under the polymorphisms of a structure Γ is an exact characterization of pp-definability, in the finite. This gives an algebraic, operation-based characterization of the logical notion of pp-definability.

Theorem 2.3.3 *Let \mathbf{B} be a relational structure with finite universe. A relation \mathcal{R} is pp-definable over \mathbf{B} if and only if \mathcal{R} is preserved by every polymorphism of \mathbf{B} .*

Clearly, the “only if” direction of this theorem is immediate from Proposition 2.2.5.

We remark that Propositions 2.2.1 and 2.2.3 can similarly be extended to exact characterizations, in the finite (proved by Krasner).

Exercise 2.3.4 Let us say that a relation $\mathcal{R} \subseteq B^k$ is equality-free if for all coordinates i, j (with $1 \leq i, j \leq k$) there exists a tuple $(b_1, \dots, b_k) \in \mathcal{R}$ such that $b_i \neq b_j$. That is, a relation is equality-free if no two of its coordinates are always identical.

Show that it suffices to prove Theorem 2.3.3 for such equality-free relations. That is, assuming that the theorem holds for equality-free relations, show that it holds in general.

Proof. We need to prove the “if” direction. Let \mathcal{R} be a relation preserved by all polymorphisms of \mathbf{B} . Let n denote the arity of \mathcal{R} , let m denote the number of tuples in \mathcal{R} , and let $(t_{11}, \dots, t_{1n}), \dots, (t_{m1}, \dots, t_{mn})$ denote the tuples of \mathcal{R} (in any order). Following Exercise 2.3.4, we may assume that there are no distinct coordinates i, j such that $(t_{1i}, \dots, t_{mi}) = (t_{1j}, \dots, t_{mj})$.

We create a conjunction of atomic formulas \mathcal{C} over the variable set B^m . Our conjunction \mathcal{C} contains, for each relation symbol S and sequence of m tuples $(s_{11}, \dots, s_{1k}), \dots, (s_{m1}, \dots, s_{mk}) \in S^{\mathbf{B}}$, an atomic formula

$$S((s_{11}, \dots, s_{m1}), \dots, (s_{1k}, \dots, s_{mk})).$$

Here, k denotes the arity of S . It is straightforward to verify that an assignment $f : B^m \rightarrow B$ satisfies the conjunction \mathcal{C} (interpreted under \mathbf{B}) if and only if it is an m -ary polymorphism of \mathbf{B} .

Now consider the relation \mathcal{R}' defined by

$$\mathcal{R}'((t_{11}, \dots, t_{m1}), \dots, (t_{1n}, \dots, t_{mn})) = \exists v_1 \dots \exists v_p \mathcal{C}$$

where v_1, \dots, v_p are the tuples in $D^m \setminus \{(t_{11}, \dots, t_{m1}), \dots, (t_{1n}, \dots, t_{mn})\}$, in any order, and the right-hand side of the equality is interpreted under \mathbf{B} . Since it is exactly the m -ary polymorphisms of Γ that satisfy \mathcal{C} , we have

$$\mathcal{R}' = \{(f(t_{11}, \dots, t_{m1}), \dots, f(t_{1n}, \dots, t_{mn})) : f \in \text{Pol}(\Gamma), f \text{ has arity } m\}.$$

We claim that $\mathcal{R}' = \mathcal{R}$, which yields the proof.

$\mathcal{R}' \subseteq \mathcal{R}$: The tuples $(t_{11}, \dots, t_{1n}), \dots, (t_{m1}, \dots, t_{mn})$ are in \mathcal{R} ; since every $f \in \text{Pol}(\Gamma)$ is a polymorphism of \mathcal{R} , it follows that $(f(t_{11}, \dots, t_{m1}), \dots, f(t_{1n}, \dots, t_{mn})) \in \mathcal{R}$.

$\mathcal{R} \subseteq \mathcal{R}'$: Let $\pi_i : D^m \rightarrow D$ denote the function that projects onto the i th coordinate, that is, the function such that $\pi_i(d_1, \dots, d_m) = d_i$ for all $(d_1, \dots, d_m) \in D^m$. Each function π_i is a polymorphism of all relations, and is hence a polymorphism of Γ ; it follows that for each $i \in \{1, \dots, m\}$, the tuple $(t_{i1}, \dots, t_{in}) = (\pi_i(t_{11}, \dots, t_{m1}), \dots, \pi_i(t_{1n}, \dots, t_{mn}))$ is in \mathcal{R}' . \square

The following notation will be useful. For a relational structure or set of relations Γ , let $\text{Pol}(\Gamma)$ denote the set containing those operations f such that f preserves all relations of Γ . Similarly, for a set of operations F , let $\text{Inv}(F)$ denote the set containing those relations \mathcal{R} such that \mathcal{R} is preserved by all relations on F . This notation allows for a succinct reformulation of Theorem 2.3.3.

Theorem 2.3.5 (reformulation of Theorem 2.3.3) For any finite relational structure \mathbf{B} , it holds that $\langle \mathbf{B} \rangle = \text{Inv}(\text{Pol}(\mathbf{B}))$.

Exercise 2.3.6 Show that the operators $\text{Pol}(\cdot)$ and $\text{Inv}(\cdot)$ are inclusion-reversing. That is, show that for any two sets of relations Γ and Γ' where $\Gamma \subseteq \Gamma'$, it holds that $\text{Pol}(\Gamma') \subseteq \text{Pol}(\Gamma)$. And, show that for any two sets of operations F and F' where $F \subseteq F'$, it holds that $\text{Inv}(F') \subseteq \text{Inv}(F)$.

Theorem 2.3.5 indicates that the relations pp-definable from a structure \mathbf{B} can be determined from the polymorphisms of \mathbf{B} : if you tell me the polymorphisms of a structure \mathbf{B} , I can tell you the pp-definable relations of \mathbf{B} , by applying $\text{Inv}(\cdot)$ to the polymorphisms you give me! In Proposition 2.3.1, we have seen that a relationship between the complexity of two problems $\text{CSP}(\mathbf{B})$ and $\text{CSP}(\mathbf{B}')$ can be derived based on the relative pp-definability of \mathbf{B} and \mathbf{B}' . This suggests that a relationship between these two problems might also be derived based on the polymorphisms of \mathbf{B} and \mathbf{B}' . This is carried out in the following theorem.

Theorem 2.3.7 *Let \mathbf{B}, \mathbf{B}' be relational structures. If $\text{Pol}(\mathbf{B}') \subseteq \text{Pol}(\mathbf{B})$, then $\text{CSP}(\mathbf{B}) \leq_p \text{CSP}(\mathbf{B}')$.*

Proof. If $\text{Pol}(\mathbf{B}') \subseteq \text{Pol}(\mathbf{B})$, then $\text{Inv}(\text{Pol}(\mathbf{B})) \subseteq \text{Inv}(\text{Pol}(\mathbf{B}'))$ (see Exercise 2.3.6). By Theorem 2.3.5, this implies $\langle \mathbf{B} \rangle \subseteq \langle \mathbf{B}' \rangle$. Applying Proposition 2.3.1, we obtain $\text{CSP}(\mathbf{B}) \leq_p \text{CSP}(\mathbf{B}')$. \square

We also have the following analog of Corollary 2.3.2.

Corollary 2.3.8 *Let \mathbf{B}, \mathbf{B}' be relational structures. If $\text{Pol}(\mathbf{B}') = \text{Pol}(\mathbf{B})$, then $\text{CSP}(\mathbf{B}) \equiv_p \text{CSP}(\mathbf{B}')$.*

Proof. Immediate from Theorem 2.3.7. \square

As a simple application of these ideas, we present the following proposition.

Proposition 2.3.9 *Let \mathbf{B} be a relational structure having universe of size two or greater, such that $\text{Pol}(\mathbf{B})$ contains only projections. Then $\text{CSP}(\mathbf{B})$ is NP-hard.*

This result should not come as a surprise. Tracing through the ideas that led to Theorem 2.3.7, we see that the fewer polymorphisms a relational structure has, the more relations it can express, and the more relations it can express, the higher its complexity. A relational structure having only projections as polymorphisms has “the fewest polymorphisms” possible!

Proof. By renaming elements of B if necessary, we may assume that $\{0, 1\} \subseteq B$. Let \mathbf{B}_0 be a structure with universe $\{0, 1\}$ such that $\text{CSP}(\mathbf{B}_0)$ is NP-hard. We may enlarge the universe of \mathbf{B}_0 to B without changing the complexity of $\text{CSP}(\mathbf{B}_0)$. We then have $\text{Pol}(\mathbf{B}) \subseteq \text{Pol}(\mathbf{B}_0)$. By Theorem 2.3.7, we have $\text{CSP}(\mathbf{B}_0) \leq_p \text{CSP}(\mathbf{B})$, implying that $\text{CSP}(\mathbf{B})$ is NP-hard. \square

Chapter 3

Clones

3.1 Basics

The results in the previous section demonstrate a relationship between the polymorphisms of a relational structure \mathbf{B} and the complexity of the CSP over the structure \mathbf{B} . As the polymorphisms of a structure forms a set of operations, this association allows us to use *algebraic, operation-based* methods to gain insight into and to study the family of problems $\text{CSP}(\mathbf{B})$.

A natural consideration at this point is: what, if any, structure do sets of polymorphisms—sets of the form $\text{Pol}(\mathbf{B})$ —have? We have already mentioned that such sets contain all projections. In fact, each such set is also closed under a notion of composition, and is an instance of an algebraic object called a *clone*.

Definition 3.1.1 *A clone (on the set B) is a set of operations that:*

- *contains all projections, that is, the operations π_i^n where $1 \leq i \leq n$ defined by*

$$\pi_i^n(b_1, \dots, b_n) = b_i$$

for all $b_1, \dots, b_n \in B$; and,

- *is closed under composition, where the composition of an n -ary operation $f : B^n \rightarrow B$ and a length n sequence of m -ary operations $g_1, \dots, g_n : B^m \rightarrow B$ is the m -ary operation $h : B^m \rightarrow B$ defined by*

$$h(b_1, \dots, b_m) = f(g_1(b_1, \dots, b_m), \dots, g_n(b_1, \dots, b_m)).$$

Proposition 3.1.2 *Let \mathbf{B} be a relational structure. The set of operations $\text{Pol}(\mathbf{B})$ is a clone.*

In the proof, we will use the following notation. When $e : B^m \rightarrow B$ is an m -ary function and $\overline{b_1} = (b_{11}, \dots, b_{1k}), \dots, \overline{b_m} = (b_{m1}, \dots, b_{mk})$ are m tuples of the same arity k , we use $e(\overline{b_1}, \dots, \overline{b_m})$ to denote the arity k tuple obtained by the coordinate-wise application of e , that is, the tuple $(e(b_{11}, \dots, b_{m1}), \dots, e(b_{1k}, \dots, b_{mk}))$.

Proof. It is straightforward to verify that all projections are polymorphisms of a relational structure \mathbf{B} .

Suppose that g_1, \dots, g_n are m -ary polymorphisms of \mathbf{B} , f is a n -ary polymorphism of \mathbf{B} , and h is the composition of f with g_1, \dots, g_n , that is,

$$h(b_1, \dots, b_m) = f(g_1(b_1, \dots, b_m), \dots, g_n(b_1, \dots, b_m)).$$

We want to show that h is a polymorphism of \mathbf{B} . Let $\mathcal{R} \subseteq B^k$ be a relation of \mathbf{B} and let $\overline{b_1} = (b_{11}, \dots, b_{1k}), \dots, \overline{b_m} = (b_{m1}, \dots, b_{mk})$ be m tuples in \mathcal{R} . Since each g_i is a polymorphism of \mathcal{R} , it holds that the tuple $g_i(\overline{b_1}, \dots, \overline{b_m})$ is contained in \mathcal{R} for all i . Then, since f is a polymorphism of \mathcal{R} , it holds that $f(g_1(\overline{b_1}, \dots, \overline{b_m}), \dots, g_n(\overline{b_1}, \dots, \overline{b_m}))$ is a tuple in \mathcal{R} . This last tuple is equal to $h(\overline{b_1}, \dots, \overline{b_m})$, yielding the proof. \square

Exercises for Section 3.1

The following exercises will acquaint us with the notion of a clone, and also introduce some concepts that are important for understanding the main theorem of the next section.

Exercise 3.1.3 Let B be a set. Show that the set of all projections on B is a clone.

We say that an operation $f : B^n \rightarrow B$ is *idempotent* if for all $b \in B$, it holds that $f(b, \dots, b) = b$.

Exercise 3.1.4 Let B be a set. Show that the set of all idempotent operations on B is a clone.

We say that a relational structure \mathbf{C} *contains all singletons* if for every element c in its universe C , there exists a unary relation symbol R_c in its signature with $R_c^{\mathbf{C}} = \{(c)\}$.

Exercise 3.1.5 Let \mathbf{C} be a relational structure that contains all singletons. Prove that all polymorphisms of \mathbf{C} are idempotent.

We will say that an operation $f : B^k \rightarrow B$ *generates* another operation $g : B^n \rightarrow B$ if any clone containing f also contains g .

Exercise 3.1.6 Let $f : B^3 \rightarrow B$ be a ternary operation. Show that f generates the binary operation $g : B^2 \rightarrow B$ defined by $g(x, y) = f(x, y, y)$.

Exercise 3.1.7 Let $f : B^3 \rightarrow B$ be a ternary operation. Show that f generates the ternary operation $g : B^3 \rightarrow B$ defined by $g(x, y, z) = f(y, z, x)$.

In general, an operation f generates all operations that can be obtained from it by associating or reordering arguments.

Exercise 3.1.8 Let $f : B^2 \rightarrow B$ be a binary operation. Show that f generates the binary operation $g : B^2 \rightarrow B$ defined by $g(x, y) = f(f(x, y), f(y, x))$.

Exercise 3.1.9 Let F be a set of operations. Say that an operation $g : B^k \rightarrow B$ is *interpolated by F* if for every finite subset $S \subseteq B^k$, there exists an operation $f \in F$ such that $g(t) = f(t)$ for every tuple $t \in S$. A set F of operations is said to be *locally closed* if for every operation g that is interpolated by F , it holds that g is contained in F .

Let \mathbf{B} be an infinite relational structure. Prove that the set of operations $\text{Pol}(\mathbf{B})$ is locally closed. (Note that we only consider functions and relations of finite arity.)

We remark that when \mathbf{B} is a finite relational structure, it is trivial to verify that $\text{Pol}(\mathbf{B})$ is locally closed.

3.2 To Idempotence

In the previous chapter, we showed that it is possible to use the set of operations $\text{Pol}(\mathbf{B})$ —which we now know is a clone—to study the complexity of $\text{CSP}(\mathbf{B})$. In this section, we show that, in fact, every problem $\text{CSP}(\mathbf{B})$ is equivalent (in complexity) to a problem $\text{CSP}(\mathbf{B}_1)$ where \mathbf{B}_1 has only *idempotent* polymorphisms. This implies that, from the standpoint of classifying the complexity of all problems $\text{CSP}(\mathbf{B})$, one need not study the space of all clones, but can restrict to idempotent clones (clones containing only idempotent polymorphisms).

Theorem 3.2.1 (*Bulatov, Jeavons, Krokhin*) *For every finite relational structure \mathbf{B} , there exists a finite relational structure \mathbf{B}_1 such that:*

- \mathbf{B}_1 contains all singletons, and hence all operations in $\text{Pol}(\mathbf{B}_1)$ are idempotent; and,
- $\text{CSP}(\mathbf{B}) \equiv_p \text{CSP}(\mathbf{B}_1)$.

We prove this theorem in a sequence of lemmas.

First, we show that when f is an endomorphism of a relational structure \mathbf{B} , the relational structure $f(\mathbf{B})$ has the same complexity as \mathbf{B} .

Lemma 3.2.2 *Let \mathbf{B} be a relational structure, and let f be an endomorphism of \mathbf{B} . Then $\text{CSP}(\mathbf{B}) = \text{CSP}(f(\mathbf{B}))$, and hence $\text{CSP}(\mathbf{B}) \equiv_p \text{CSP}(f(\mathbf{B}))$.*

Here, by $f(\mathbf{B})$, we denote the relational structure with universe $f(B)$ and where for every relation symbol R , $R^{f(\mathbf{B})}$ is defined as the relation $f(R^{\mathbf{B}}) = \{(f(b_1), \dots, f(b_k)) \mid (b_1, \dots, b_k) \in R^{\mathbf{B}}\}$. By the notation $\text{CSP}(\mathbf{B}) = \text{CSP}(f(\mathbf{B}))$, we mean that an input relational structure \mathbf{A} is a “yes” instance of $\text{CSP}(\mathbf{B})$ if and only if it is a “yes” instance of $\text{CSP}(f(\mathbf{B}))$.

Proof. We want to show that for any relational structure \mathbf{A} , there is a homomorphism from \mathbf{A} to \mathbf{B} if and only if there is a homomorphism from \mathbf{A} to $f(\mathbf{B})$. It suffices to show that the structures \mathbf{B} and $f(\mathbf{B})$ have homomorphisms to and from each other; the claim then follows from transitivity of the homomorphism preorder (see Exercise 1.1.2). By definition of $f(\mathbf{B})$, the mapping f is a homomorphism from \mathbf{B} to $f(\mathbf{B})$. On the other hand, for each relation symbol R it holds that $R^{f(\mathbf{B})}$ is a subset of $R^{\mathbf{B}}$ because f is an endomorphism, and hence the identity mapping on $f(B)$ is a homomorphism from $R^{f(\mathbf{B})}$ to $R^{\mathbf{B}}$. \square

The first step in proving the theorem will be to take an endomorphism f of \mathbf{B} and apply the previous lemma to obtain a new relational structure $f(\mathbf{B})$ of the same complexity. In particular, we will pick an endomorphism f having minimum image, which—as the following lemma shows—will yield the desirable property that “endomorphisms are automorphisms”.

Lemma 3.2.3 *Let \mathbf{B} be a finite relational structure, and let f be an endomorphism of \mathbf{B} having minimum image. Then, all endomorphisms of $f(\mathbf{B})$ are automorphisms.*

Proof. Let g be an endomorphism of $f(\mathbf{B})$. Then $g(f(B)) \subseteq f(B)$, and for every relation symbol R , we have $g(R^{f(\mathbf{B})}) \subseteq R^{f(\mathbf{B})}$. Since by definition $R^{f(\mathbf{B})} = f(R^{\mathbf{B}})$, we obtain

$$g(f(R^{\mathbf{B}})) \subseteq f(R^{\mathbf{B}}) \subseteq R^{\mathbf{B}}.$$

Thus, $g \circ f$ is an endomorphism of \mathbf{B} . Since f has minimum image among endomorphisms and $g(f(B)) \subseteq f(B)$, we have $g(f(B)) = f(B)$, and thus g is a bijection on $f(B)$. On finite structures, an endomorphism that is a bijection is an automorphism (see Exercise 2.2.4). \square

We mention that a finite structure where all endomorphisms are automorphisms is called a *core*. Our last lemma shows that for such structures, we may adjoin singletons to the structure without changing its complexity.

Lemma 3.2.4 *Let \mathbf{B}' be a finite relational structure whose endomorphisms are all automorphisms. Then, $\text{CSP}(\mathbf{B}') \equiv_p \text{CSP}(\mathbf{B}'_c)$, where \mathbf{B}'_c is the expansion of \mathbf{B}' with singletons.*

By the expansion of \mathbf{B}' with singletons, we mean the structure \mathbf{B}'_c that has as signature all relation symbols in the signature σ of \mathbf{B}' , as well as an additional relation symbol R_b for each b in the universe B' of \mathbf{B}' . For all relation symbols $R \in \sigma$, the structure \mathbf{B}'_c is defined to be equal to \mathbf{B}' : $R^{\mathbf{B}'_c} = R^{\mathbf{B}'}$. In addition, for each $b \in B'$, $R_b^{\mathbf{B}'_c}$ is defined to be $\{(b)\}$.

In the proof, we will use another construction, the union of two relational structures \mathbf{A} and \mathbf{B} over the same signature. This is defined to be the structure $\mathbf{A} \cup \mathbf{B}$ with universe $A \cup B$ and $R^{\mathbf{A} \cup \mathbf{B}} = R^{\mathbf{A}} \cup R^{\mathbf{B}}$ for all relation symbols R .

Proof. The direction $\text{CSP}(\mathbf{B}') \leq_p \text{CSP}(\mathbf{B}'_c)$ is straightforward: given an instance \mathbf{A} of $\text{CSP}(\mathbf{B}')$, simply expand it so that $R_b^{\mathbf{A}} = \emptyset$ for all $b \in B'$. Clearly, the original instance \mathbf{A} has a homomorphism to \mathbf{B}' if and only if the expansion has a homomorphism to \mathbf{B}'_c .

We now show $\text{CSP}(\mathbf{B}'_c) \leq_p \text{CSP}(\mathbf{B}')$. Given an instance \mathbf{A} of $\text{CSP}(\mathbf{B}'_c)$, we create an instance of $\text{CSP}(\mathbf{B}')$ as follows. First, rename the elements of A if necessary so that $A \cap B' = \emptyset$. Then, let \mathbf{A}_2 be the structure over the signature of \mathbf{B}' obtained from \mathbf{A} by:

- forgetting the relation symbols $\{R_b\}$, and then
- replacing all instances of a with b for every $b \in B'$ and every $a \in R_b^{\mathbf{A}}$.

(In order that the second step is well-defined, we assume that the sets $R_b^{\mathbf{A}}$ for different $b \in B'$ are disjoint. If this is not the case, \mathbf{A} is clearly a “no”-instance of $\text{CSP}(\mathbf{B}'_c)$.) The instance of $\text{CSP}(\mathbf{B}')$ we create is $\mathbf{A}_2 \cup \mathbf{B}'$.

Suppose that \mathbf{A} is a “yes”-instance of $\text{CSP}(\mathbf{B}'_c)$ via the homomorphism h . Then, the restriction of h to A_2 extended to act as the identity on B' gives a homomorphism from $\mathbf{A}_2 \cup \mathbf{B}'$ to \mathbf{B}' .

Suppose that $\mathbf{A}_2 \cup \mathbf{B}'$ is a “yes”-instance of $\text{CSP}(\mathbf{B}')$ via the homomorphism g . By definition of \mathbf{A}_2 , to show that \mathbf{A} is a “yes”-instance of $\text{CSP}(\mathbf{B}'_c)$, it suffices to show that there is a homomorphism from $\mathbf{A}_2 \cup \mathbf{B}'$ to \mathbf{B}' that acts as the identity on B' . Let g' denote the restriction of g to B' . The mapping g' is an endomorphism of \mathbf{B}' and hence an automorphism of \mathbf{B}' . If g' itself is the identity mapping, we are done: g is the desired homomorphism. Otherwise, we may observe that there exists a constant $K \geq 1$ such that $g'^K \circ g'$ acts as the identity on B' . Then, the mapping $g'^K \circ g$ is the identity on B' , and yields the desired homomorphism. \square

Proof. (Theorem 3.2.1) Let f be an endomorphism of \mathbf{B} having minimum image, and let $\mathbf{B}' = f(\mathbf{B})$. By Lemma 3.2.2, we have $\text{CSP}(\mathbf{B}) \equiv_p \text{CSP}(\mathbf{B}')$. By Lemma 3.2.3, all endomorphisms of \mathbf{B}' are automorphisms of \mathbf{B}' . Hence, we may apply Lemma 3.2.4 to \mathbf{B}' , from which we obtain that the expansion \mathbf{B}'_c of \mathbf{B}' with singletons has the property $\text{CSP}(\mathbf{B}') \equiv_p \text{CSP}(\mathbf{B}'_c)$. Defining \mathbf{B}_1 to be \mathbf{B}'_c , we obtain the result. \square

3.3 The Boolean Case

We saw (Proposition 2.3.9) that a relational structure whose clone of polymorphisms contains only projections, gives rise to a NP-hard constraint satisfaction problem. Given a relational

structure, is there any way to effectively check if all polymorphisms are projections? We will see in this section that the answer is “yes” in the boolean case—specifically, that any clone that contains non-projections, must contain non-projections of a very particular form.

We begin with the idempotent case. We show that any idempotent clone containing non-projections must contain one of four operations. The operations are boolean OR (\vee), the binary idempotent operation with

$$0 \vee 1 = 1 \vee 0 = 1$$

boolean AND (\wedge), the binary idempotent operation with

$$0 \wedge 1 = 1 \wedge 0 = 0$$

majority (major), the ternary idempotent operation with

$$\text{major}(0, 0, 1) = \text{major}(0, 1, 0) = \text{major}(1, 0, 0) = 0$$

$$\text{major}(1, 1, 0) = \text{major}(1, 0, 1) = \text{major}(0, 1, 1) = 1$$

and minority (minor), the ternary idempotent operation with

$$\text{minor}(0, 0, 1) = \text{minor}(0, 1, 0) = \text{minor}(1, 0, 0) = 1$$

$$\text{minor}(1, 1, 0) = \text{minor}(1, 0, 1) = \text{minor}(0, 1, 1) = 0$$

Theorem 3.3.1 *An idempotent clone over the two-element set $\{0, 1\}$ either consists only of projections, or contains one of the operations $\{\vee, \wedge, \text{major}, \text{minor}\}$.*

Recall that a clone is *idempotent* if all of its operations are idempotent.

Proof. Let $g : \{0, 1\}^m \rightarrow \{0, 1\}$ be an idempotent operation that is not a projection in an idempotent clone C . We assume that g is of minimal arity, that is, all operations of arity strictly lower than m in C are projections. We will show that g generates one of the four operations given in the statement of the theorem. We break into cases based on the arity m of g . Note that the only arity 1 idempotent operation is the identity mapping, which is a projection, and so the arity m of g is strictly greater than 1.

Case $m = 2$: We cannot have $g(0, 1) \neq g(1, 0)$, otherwise the operation g is a projection. Thus $g(0, 1) = g(1, 0)$. If $g(0, 1) = g(1, 0) = 0$, we have that g is the operation \wedge , and if $g(0, 1) = g(1, 0) = 1$, we have that g is the operation \vee .

Case $m = 3$: By the minimality of the arity m of g , if any two arguments of g are identified, we obtain a binary operation that is a projection onto either its first or second coordinate. Considering the three possible ways of identifying two arguments of g , we obtain the following eight possibilities:

	(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)
$g(x, x, y) =$	x	x	x	x	y	y	y	y
$g(x, y, x) =$	x	x	y	y	x	x	y	y
$g(y, x, x) =$	x	y	x	y	x	y	x	y

In case (1), we have $g = \text{major}$, and in case (8), we have $g = \text{minor}$. The cases (2), (3), and (5) cannot occur, since then f is a projection. The remaining cases, (4), (6), and (7), are symmetric; in each of them, g generates the operation major . For example, in case (6), we have $\text{major}(x, y, z) = g(x, g(x, y, z), z)$.

Case $m \geq 4$: We show that this case cannot occur. First suppose that whenever two coordinates of g are identified, the result is a projection onto the identified coordinates. Then, we have $g(x_1, x_1, x_3, \dots, x_m) = x_1$ and $g(x_1, x_2, x_3, \dots, x_m) = x_3$. But, this implies that $g(0, 0, 1, \dots, 1)$ is equal to both 0 and 1, a contradiction. So, there exist two coordinates such that, when identified, the result is a projection onto a different coordinate. Assume for the sake of notation that $g(x_1, x_1, x_3, x_4, \dots, x_m) = x_4$. We claim that $g(x_1, x_2, x_1, x_4, \dots, x_m) = x_4$. Let j be such that $g(x_1, x_2, x_1, x_4, \dots, x_m) = x_j$. This implies $g(x_1, x_1, x_1, x_4, \dots, x_m) = x_j$. Observe that

$$g(x_1, x_1, x_3, x_4, \dots, x_m) = x_4 \text{ implies } g(x_1, x_1, x_1, x_4, \dots, x_m) = x_4$$

and hence $x_j = x_4$. We can similarly show that $g(x_1, x_2, x_2, x_4, \dots, x_m) = x_4$. For any assignment to the variables x_1, \dots, x_m , one of the equalities $x_1 = x_2$, $x_1 = x_3$, $x_2 = x_3$ must hold, since our domain $\{0, 1\}$ has two elements, and thus g is a projection. \square

Theorem 3.3.1 follows from a classification result due to Post (1941) that gives a complete description of all clones over a two-element domain.

As an application of this theorem, we may give a sufficient condition for the NP-completeness of an idempotent clone (in the two-element case).

Corollary 3.3.2 *Let \mathbf{B} be a relational structure with two-element universe $\{0, 1\}$ such that $\text{Pol}(\mathbf{B})$ is an idempotent clone. If \mathbf{B} does not have one of the operations $\{\vee, \wedge, \text{major}, \text{minor}\}$ as a polymorphism, then $\text{CSP}(\mathbf{B})$ is NP-complete.*

Proof. We have that $\text{Pol}(\mathbf{B})$ is an idempotent clone over $\{0, 1\}$. If $\text{Pol}(\mathbf{B})$ does not contain one of the four named operations, then by Theorem 3.3.1, $\text{Pol}(\mathbf{B})$ consists only of projections. It follows from Proposition 2.3.9 that $\text{CSP}(\mathbf{B})$ is NP-hard. \square

We now give a generalization of Corollary 3.3.2 that applies to all two-element structures, and not just those having an idempotent polymorphism clone. We use c_0 to denote the unary operation such that $c_0(0) = c_0(1) = 0$, and similarly we use c_1 to denote the unary operation such that $c_1(0) = c_1(1) = 1$.

Theorem 3.3.3 *Let \mathbf{B} be a relational structure with two-element universe $\{0, 1\}$. If \mathbf{B} does not have one of the operations $\{c_0, c_1, \vee, \wedge, \text{major}, \text{minor}\}$ as a polymorphism, then $\text{CSP}(\mathbf{B})$ is NP-complete.*

Proof. Since c_0 and c_1 are not polymorphisms of \mathbf{B} , all endomorphisms of \mathbf{B} must be bijective. (Recall that an endomorphism is an arity 1 polymorphism.) It can be verified that, in the case of a finite universe, a bijective endomorphism is an automorphism. Hence, all endomorphisms of \mathbf{B} are automorphisms.

We may thus apply Lemma 3.2.4; we obtain that $\text{CSP}(\mathbf{B}) \equiv_p \text{CSP}(\mathbf{B}_c)$, where \mathbf{B}_c is the expansion of \mathbf{B} with singletons. The structure \mathbf{B}_c has more relations than \mathbf{B} , so if \mathbf{B}_c has any of $\{\vee, \wedge, \text{major}, \text{minor}\}$ as a polymorphism, so does \mathbf{B} . Since \mathbf{B} does not have one of these four operations as a polymorphism, neither does \mathbf{B}_c . As all polymorphisms of \mathbf{B}_c are idempotent,

we obtain from Corollary 3.3.2 that $\text{CSP}(\mathbf{B}_c)$ is NP-hard. From $\text{CSP}(\mathbf{B}) \equiv_p \text{CSP}(\mathbf{B}_c)$, it follows that $\text{CSP}(\mathbf{B})$ is NP-hard. \square

It in fact turns out that—in the boolean case—the sufficient condition for NP-completeness given by Theorem 3.3.3 is an exact criterion for NP-completeness. Presence of any of the named polymorphisms implies that $\text{CSP}(\mathbf{B})$ is polynomial-time decidable. In the case of one of the polymorphisms c_0 or c_1 , polynomial-time tractability is straightforward to verify (exercise!), and the polymorphisms \vee , \wedge , and **major** will be treated in the coming sections. We will not explicitly treat the polymorphism **minor** in this course, but can mention that a relation has this polymorphism if and only if its tuples are the solutions to a system of linear equations over $\{0, 1\}$ with respect to \oplus . Hence, algebraic techniques such as Gaussian elimination can be used to solve such constraints.

Exercises for Section 3.3

Exercise 3.3.4 Let \mathbf{B} be a relational structure with universe $\{0, 1\}$ and relation $\{(1, 0, 0), (0, 1, 0), (0, 0, 1)\}$. Using Theorem 3.3.3, show that $\text{CSP}(\mathbf{B})$ is NP-complete.

Exercise 3.3.5 Let \mathbf{B} be a relational structure with universe $\{0, 1\}$ and relation $\{0, 1\}^3 \setminus \{(0, 0, 0), (1, 1, 1)\}$. Using Theorem 3.3.3, show that $\text{CSP}(\mathbf{B})$ is NP-complete.

Exercise 3.3.6 Let \mathbf{B} be a relational structure with universe $\{0, 1\}$ and relation $\{(a, b, c) \in \{0, 1\}^3 \mid (a = b) \text{ or } (b = c)\}$. Let \mathbf{B}_c be the expansion of \mathbf{B} with singletons. Using Theorem 3.3.3, show that $\text{CSP}(\mathbf{B}_c)$ is NP-complete.

For the next few exercises, we will need the following notion. Say that an operation $f : B^k \rightarrow B$ is *essentially unary* if there exist a coordinate $i \in \{1, \dots, k\}$ and a unary operation $g : B \rightarrow B$ such that $f(b_1, \dots, b_k) = g(b_i)$ for all tuples $(b_1, \dots, b_k) \in B^k$.

Exercise 3.3.7 Fix a set B . Show that the set of all essentially unary operations over B is a clone.

Exercise 3.3.8 Let \mathbf{B} be a finite relational structure such that all polymorphisms of \mathbf{B} are essentially unary operations. Observe that $\text{CSP}(\mathbf{B})$ is in P if \mathbf{B} has a constant polymorphism, that is, a polymorphism of the form $c_b : B \rightarrow B$ with $c_b(d) = b$ for all $d \in B$. Prove that $\text{CSP}(\mathbf{B})$ is NP-complete otherwise.

Exercise 3.3.9 (*) Let C be a clone—not necessarily idempotent—over the set $\{0, 1\}$. Prove that either C contains only essentially unary operations, or it contains one of the operations $\{\vee, \wedge, \text{major}, \text{minor}\}$. (Hint: one way to do this is as follows. Let f be an operation that is not essentially unary. Consider the unary operation $f(x, \dots, x)$; there are four possibilities. For each of these four cases, show that f generates an idempotent non-projection, and then invoke Theorem 3.3.1.)

In the next two exercises, we will establish a version of *Rosenberg's Theorem*. This version shows that any non-trivial clone (clone not consisting of all projections) must contain one of five types of operations. It may be conceived of as a generalization of Theorem 3.3.1.

Exercise 3.3.10 Define an operation $g : D^m \rightarrow D$ to be a *semiprojection* if there exists a coordinate $i \in \{1, \dots, m\}$ such that for all $d_1, \dots, d_m \in D$,

$$g(d_1, \dots, d_m) = d_i$$

whenever the d_j are not pairwise distinct, that is, whenever $|\{d_1, \dots, d_m\}| < m$.

Prove the Świerczkowski Lemma: Given an operation $g : D^m \rightarrow D$ of arity $m \geq 4$, if every operation arising from identifying two variables of g is a projection, then these projections coincide (and hence that g is a semiprojection).

Exercise 3.3.11 Prove the following version of Rosenberg’s theorem: every non-trivial clone (clone containing non-projections) contains an operation of one of the following five types:

1. a unary operation,
2. a binary idempotent operation,
3. a ternary majority operation, that is, a ternary operation m satisfying the identities $m(x, x, y) = m(x, y, x) = m(y, x, x) = x$,
4. a ternary minority operation, that is, a ternary operation m satisfying the identities $m(x, x, y) = m(x, y, x) = m(y, x, x) = y$,
5. an m -ary semiprojection with $m \geq 3$.

As a (big) hint, we provide the following outline of a proof. Let C be a non-trivial clone and let $g : B^m \rightarrow B$ be a non-projection in C of minimal arity. If $m = 1$, then we are done; trivially, our clone C contains an operation of type (1). So assume $m \geq 2$. Notice that, by minimality of m , identifying two (or more) arguments of g results in a projection; in particular, identifying all arguments g yields $g(x, \dots, x)$, which must be the unary identity operation, from which we may infer that g is idempotent. If $m = 2$, then our clone contains an operation of type (2) and we are done. If $m = 3$, work from this case of the proof of Theorem 3.3.1 to show that the clone must contain an operation of type (3), (4), or (5). If $m \geq 4$, apply the previous exercise to obtain an operation of type (5).

Exercise 3.3.12 Using the version of Rosenberg’s theorem given in the previous exercise, give an algorithm that, on an input finite relational structure \mathbf{A} , decides whether or not $\text{Pol}(\mathbf{A})$ contains any non-projections.

The next two exercises concern the *quantified* constraint satisfaction problem relative to a relational structure \mathbf{B} , denoted $\text{QCSP}(\mathbf{B})$. An instance of $\text{QCSP}(\mathbf{B})$ is a logical sentence over the signature of \mathbf{B} built from the connective \wedge and the quantifiers \exists and \forall , and the question is to decide whether or not the sentence is true over \mathbf{B} .

Exercise 3.3.13 Let \mathbf{B} be a finite relational structure with universe size two or greater. Prove that if $\text{Pol}(\mathbf{B})$ contains only projections, then $\text{QCSP}(\mathbf{B})$ is PSPACE-complete. (Use the fact that Quantified 3-SAT is PSPACE-complete.)

Exercise 3.3.14 Let \mathbf{B} be a finite relational structure with universe size two or greater. Prove that if $\text{Pol}(\mathbf{B})$ contains only essentially unary operations, then $\text{QCSP}(\mathbf{B})$ is PSPACE-complete. (Hint: you might start by trying to prove this in the case of universe size two.)

This exercise clearly generalizes the previous one; one may also compare it with Exercise 3.3.8.

For a finite relational structure \mathbf{B} , we say that $f(\mathbf{B})$ is a *core* of \mathbf{B} if f is an endomorphism of \mathbf{B} with minimal image.

Exercise 3.3.15 Show that if f_1 and f_2 are two endomorphisms of a finite structure \mathbf{B} having minimal image, then $f_1(\mathbf{B})$ and $f_2(\mathbf{B})$ are isomorphic.

In light of this exercise, we speak of *the core* of \mathbf{B} .

Chapter 4

Consistency Techniques

Establishing local consistency is a very general algorithmic tool in constraint satisfaction. Local consistency is intensively studied, both in theory, where the main research questions are still open, and in the literature that is directed towards applications, where efficient constraint propagation algorithms are of central interest.

The general idea of local consistency algorithms is to compute from a given instance a new instance that is ‘*locally consistent*’, in a process that is also known as *constraint propagation*. In this process we derive new constraints that are implied by given constraints or by constraints that have already been derived before. Frequently, it is necessary to formulate the inferred constraints in a different constraint language than the language of the CSP we started with. Formally, to solve $\text{CSP}(\mathbf{B})$ we might formulate algorithms for $\text{CSP}(\mathbf{B}')$ where \mathbf{B}' is an expansion of \mathbf{B} by additional relations.

If we are able to derive ‘false’ in that way, we know that the instance of the CSP has no solution. In our course on the right-hand side complexity of constraint satisfaction, we will be interested in the question for which relational structures \mathbf{B} the problem $\text{CSP}(\mathbf{B})$ can be *solved* by such a procedure, i.e., ‘false’ is derived for an instance \mathbf{A} if and only if \mathbf{A} does not homomorphically map to \mathbf{B} .

4.1 Hyperarc-Consistency

One of the most important local consistency algorithms is the so-called *hyperarc-consistency procedure*¹. It has many good features. For instance, there are elegant characterizations of those structures \mathbf{B} where $\text{CSP}(\mathbf{B})$ can be solved by the hyperarc-consistency procedure. From a more practical perspective, it is a good property that the procedure can be implemented such that the running time and the memory space requirements are linear in the size of the input.

To define the hyperarc-consistency procedure for $\text{CSP}(\mathbf{B})$, we assume that \mathbf{B} contains a relation for each unary primitive positive definable relation in \mathbf{B} . This is not a strong assumption, since we might always study the expansion \mathbf{B}' of \mathbf{B} by all such unary relations; if the hyperarc-consistency solves $\text{CSP}(\mathbf{B}')$ then it clearly also solves $\text{CSP}(\mathbf{B})$. Also recall Proposition 2.2.5, which states that expanding a core structure \mathbf{B} by a primitive positive definable relation does not change the computational complexity of $\text{CSP}(\mathbf{B})$.

¹sometimes also called the *generalized arc-consistency procedure*.

To conveniently formulate the algorithm, we write $R_{\phi(x)}$ for the relation symbol of the relation that is defined by a primitive positive formula $\phi(x)$ in \mathbf{B} . We write $Q_{\mathbf{A}}^1(a_1, \dots, a_l)$ for the conjunction of $S(a_i)$ where S is a unary relation symbol such that $a_i \in S^{\mathbf{A}}$. The pseudo-code of the hyperarc-consistency procedure can be found in Figure 4.1.

```

hyperarc-consistency( $\mathbf{A}$ )
Input: a finite relational structure  $\mathbf{A}$ .

Do
  For every relation symbol  $R$ , every tuple  $(a_1, \dots, a_l) \in R^{\mathbf{A}}$ , and every  $i \in \{1, \dots, l\}$ 
    Let  $\phi$  be the formula  $\exists a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_l. R(a_1, \dots, a_l) \wedge Q_{\mathbf{A}}^1(a_1, \dots, a_l)$ 
    If  $\phi$  defines the empty unary relation over  $\mathbf{B}$  then reject
    Else, add  $a_i$  to  $R_{\phi}^{\mathbf{A}}$ 
Loop until no relation in  $\mathbf{A}$  is changed

```

Figure 4.1: The hyperarc-consistency procedure for $\text{CSP}(\mathbf{B})$, where \mathbf{B} contains all unary primitive positive definable relations in \mathbf{B} .

The space requirements of the given hyperarc-consistency procedure are clearly linear. The hyperarc-consistency procedure can be implemented such that its running time is *linear* in the size of the input (Exercise 4.1.11).

We say that the hyperarc-consistency procedure *accepts* an instance \mathbf{A} of $\text{CSP}(\mathbf{B})$ if it does not reject. We say that the hyperarc-consistency procedure *solves* $\text{CSP}(\mathbf{B})$ if for all finite structures \mathbf{A} it accepts \mathbf{A} if and only if there is a homomorphism from \mathbf{A} to \mathbf{B} . As we will see in this section, there is an elegant characterization of those structures \mathbf{B} where $\text{CSP}(\mathbf{B})$ can be solved by the hyperarc-consistency procedure. For this characterization, we have to introduce the notion of the (*power-*) *set structure* $P(\mathbf{B})$ of \mathbf{B} . Other interesting characterizations are known, e.g., in terms of *tree-duality* [16], but this is out of the scope of this short course.

For a set $S \subseteq B^k$, we write $pr_i S$ for the set $\{s_i \mid (s_1, \dots, s_k) \in S\}$, that is, the set of all values as the i th coordinate in some tuple in S .

Definition 4.1.1 *Let \mathbf{B} be a relational structure over τ . Then the (power-) set structure $P(\mathbf{B})$ is the τ -structure whose vertices are the non-empty subsets of B , and where $R^{P(\mathbf{B})} = \{(pr_1 S, \dots, pr_k S) \mid S \neq \emptyset, S \subseteq R^{\mathbf{B}}\}$.*

At this point, we recommend the reader to play with the concept of set structures and to solve for example Exercises 4.1.8 and 4.1.9, and 4.1.10.

Proposition 4.1.2 *The hyperarc-consistency procedure accepts \mathbf{A} if and only if \mathbf{A} homomorphically maps to $P(\mathbf{B})$.*

Proof. First note that at all times during the execution of the procedure on \mathbf{A} the set of homomorphisms from \mathbf{A} to $P(\mathbf{B})$ is not changed by the algorithm. To see this, let h be a homomorphism from \mathbf{A} to $P(\mathbf{B})$. Suppose that at some point the algorithm adds a_i to $R_{\phi}^{\mathbf{A}}$. We have to show that $h(a_i) \in R_{\phi}^{P(\mathbf{B})}$. Since h is a homomorphism, $(h(a_1), \dots, h(a_l)) \in R^{\mathbf{B}}$.

We inductively assume that $h(a_j)$ satisfies the formula $Q_{\mathbf{A}}^1(a_j)$ in $P(\mathbf{B})$ for all $j \in \{1, \dots, l\}$. Therefore each element in $h(a_i)$ satisfies ϕ in \mathbf{B} , and $h(a_i)$ is a subset of $R_{\phi}^{\mathbf{B}}$. Hence by the definition of the set structure $h(a_i) \in R_{\phi}^{P(\mathbf{B})}$. Thus, if the algorithm rejects an instance, there can be no homomorphism from \mathbf{A} to $P(\mathbf{B})$.

For the other direction, suppose that the hyperarc-consistency procedure accepts \mathbf{A} . Observe that if $a \in R_{\phi_1(x)}^{\mathbf{A}}$ and $a \in R_{\phi_2(x)}^{\mathbf{A}}$, then at the final stage of the algorithm $a \in R_{\phi_1(x) \wedge \phi_2(x)}^{\mathbf{A}}$. Hence, there is a unique smallest set $S_a \subseteq B$ such that S_a has a primitive positive definition ψ in \mathbf{B} and $a \in R_{\psi}^{\mathbf{A}}$. We claim that the function g that maps $a \in A$ to the vertex S_a of $P(\mathbf{B})$ is a homomorphism from \mathbf{A} to $P(\mathbf{B})$. Let $(a_1, \dots, a_k) \in R^{\mathbf{A}}$. We know that at the final stage of the algorithm for all choices of $i \in \{1, \dots, l\}$ the element a_i is already in $R_{\phi}^{\mathbf{A}}$, where ϕ is the formula specified in the pseudo-code. Hence, each element of $g(a_i)$ appears in some tuple of $R_{\phi}^{\mathbf{B}}$ which is a subset of $R^{\mathbf{B}}$. By the definition of $P(\mathbf{B})$, $(g(a_1), \dots, g(a_k)) \in R^{P(\mathbf{B})}$. We have thus shown that \mathbf{A} homomorphically maps to $P(\mathbf{B})$. \square

Corollary 4.1.3 *If the hyperarc-consistency procedure rejects \mathbf{A} then there is no homomorphism from \mathbf{A} to \mathbf{B} .*

Proof. First observe that the function from B to $2^B \setminus \emptyset$ that maps $a \in B$ to $\{a\}$ is a homomorphism from \mathbf{B} to $P(\mathbf{B})$. If the hyperarc-consistency procedure rejects \mathbf{A} , Proposition 4.1.2 shows that there is no homomorphism from \mathbf{A} to $P(\mathbf{B})$. If there was a homomorphism from \mathbf{A} to \mathbf{B} , we could compose it with the homomorphism from \mathbf{B} to $P(\mathbf{B})$ and obtain a contradiction. \square

Theorem 4.1.4 (of [15, 12]) *Let \mathbf{B} be a finite structure. Then the following are equivalent.*

1. *The hyperarc-consistency procedure solves $\text{CSP}(\mathbf{B})$.*
2. *$P(\mathbf{B})$ homomorphically maps to \mathbf{B} ;*
3. *There is an m -ary polymorphism f of \mathbf{B} satisfying $f(x_1, \dots, x_m) = f(y_1, \dots, y_m)$ whenever $\{x_1, \dots, x_m\} = \{y_1, \dots, y_m\}$, where m is the maximum number of tuples in a relation of \mathbf{B} .*

Proof. To show the equivalence of 1 and 2, we first prove that 1 implies 2. Suppose the hyperarc-consistency procedure solves $\text{CSP}(\mathbf{B})$. We apply the procedure to $P(\mathbf{B})$. Then Proposition 4.1.2 implies that the procedure does not reject $P(\mathbf{B})$. Since the hyperarc-consistency procedure solves $\text{CSP}(\mathbf{B})$, we have that $P(\mathbf{B})$ homomorphically maps to \mathbf{B} .

To prove that 2 implies 1, suppose that there is a homomorphism f from $P(\mathbf{B})$ to \mathbf{B} , and let \mathbf{A} be an instance of $\text{CSP}(\mathbf{B})$. If the hyperarc-consistency procedure rejects \mathbf{A} then by Corollary 4.1.3 there is no homomorphism from \mathbf{A} to \mathbf{B} . Otherwise, Proposition 4.1.2 implies that \mathbf{A} homomorphically maps to $P(\mathbf{B})$. But then we can compose this homomorphism with the homomorphism from $P(\mathbf{B})$ to \mathbf{B} and found a homomorphism from \mathbf{A} to \mathbf{B} .

We now show the equivalence between 2 and 3. Suppose first that there is a homomorphism g from $P(\mathbf{B})$ to \mathbf{B} . Define f by $f(b_1, \dots, b_m) = g(\{b_1, \dots, b_m\})$ for all $b_1, \dots, b_m \in B$. We claim that f is a polymorphism of \mathbf{B} . Because if $(b_{11}, \dots, b_{1k}), \dots, (b_{m1}, \dots, b_{mk})$ are tuples in $R^{\mathbf{B}}$, then $(\{b_{11}, \dots, b_{m1}\}, \dots, \{b_{1k}, \dots, b_{mk}\})$ is in $R^{P(\mathbf{B})}$, and hence $(f(b_{11}, \dots, b_{m1}), \dots, f(b_{1k}, \dots, b_{mk})) \in R^{\mathbf{B}}$.

Conversely, suppose that f is an m -ary polymorphism as described in the statement. For any elements $b_1, \dots, b_m \in B$, let $g : 2^B \setminus \emptyset \rightarrow B$ be such that $g(\{b_1, \dots, b_m\}) = f(b_1, \dots, b_m)$; in this way, the mapping g is well-defined by the properties of f .

We prove that g preserves each relation $R^{P(\mathbf{B})}$ in $P(\mathbf{B})$. Let $(U_1, \dots, U_k) \in R^{P(\mathbf{B})}$. Let t_1, \dots, t_m be any enumeration of the tuples in $R^{\mathbf{B}} \cap (U_1 \times \dots \times U_m)$, possibly with repetitions (by choice of m , the number of tuples in $R^{P(\mathbf{B})}$ is less than or equal to m). By the definition of $P(\mathbf{B})$, each element $u_i \in U_i$ appears in the i -th entry of some of those tuples t_1, \dots, t_m , and hence $(f(t_{11}, \dots, f_{m1}), \dots, f(t_{1k}, \dots, f(t_{mk}))) = (g(U_1), \dots, g(U_k))$. Since f is a polymorphism, this tuple must be in $R^{\mathbf{B}}$. \square

Note that the second condition in Theorem 4.1.4 can be used to decide algorithmically whether a given finite structure \mathbf{B} has width one, because we can effectively decide whether $P(\mathbf{B})$ homomorphically maps to \mathbf{B} . However, the naive algorithm for this approach has a very bad worst-case running time, because $P(\mathbf{B})$ has exponential size in \mathbf{B} , and moreover we only know a *non-deterministic* algorithm to test whether $P(\mathbf{B})$ homomorphically maps to \mathbf{B} which runs in exponential time in the size of \mathbf{B} . The following remains open.

Question 4.1.5 *What is the computational complexity to decide for a given structure \mathbf{B} whether \mathbf{B} has width one?*

Theorem 4.1.4 is a surprisingly powerful condition that implies that $\text{CSP}(\mathbf{B})$ is tractable. For example, if \mathbf{B} is a relational structure over the boolean domain $\{0, 1\}$ and has \wedge (or, similarly, \vee) as a polymorphism, then the hyperarc-consistency procedure solves $\text{CSP}(\mathbf{B})$. To show this, we use the third condition in Theorem 4.1.4, and consider the m -ary operation f defined by $f(x_1, \dots, x_m) = \wedge(x_1, \wedge(x_2, \dots \wedge (x_{m-1}, x_m) \dots))$. By Proposition 3.1.2 the operation f is also a polymorphism of \mathbf{B} . Moreover, $f(x_1, \dots, x_m) = f(y_1, \dots, y_m)$ whenever $\{x_1, \dots, x_m\} = \{y_1, \dots, y_m\}$, because $f(x_1, \dots, x_m) = 0$ precisely if one of the arguments is 0. This condition only depends on the set $\{x_1, \dots, x_m\}$, and not on the order and multiplicity of the arguments.

There are also several natural families of directed graphs \mathbf{H} where hyperarc-consistency solves $\text{CSP}(\mathbf{H})$. This can often be conveniently deduced from Theorem 4.1.4 by verifying that $P(\mathbf{H})$ homomorphically maps to \mathbf{H} . We now view a directed graph \mathbf{H} as a relational structure over a signature that contains a single binary relation symbol E . In the following, we frequently use directed graphs to illustrate the concepts of the course. Here are some fundamental notions from graph theory.

Definition 4.1.6 *Let \mathbf{H} be a directed graph with the vertex set $\{1, \dots, n\}$. We say that \mathbf{H} is an oriented path if for all $i < n$ either $(i, i+1)$ or $(i+1, i)$ is in $E^{\mathbf{H}}$ (and no other edges are in $E^{\mathbf{H}}$). A special case of oriented paths are directed paths, where $(i, i+1)$ is in $E^{\mathbf{H}}$ for all $i < n$. We say that \mathbf{H} is an oriented cycle if either $(i, i+1)$ or $(i+1, i)$ is in $E^{\mathbf{H}}$ for all $i < n$, and either $(n, 1)$ or $(1, n)$ is in $E^{\mathbf{H}}$ (and not other edges are in $E^{\mathbf{H}}$). A special case of oriented cycles are directed cycles where $(n, 1) \in E^{\mathbf{H}}$ and $(i, i+1) \in E^{\mathbf{H}}$ for all $i < n$.*

Theorem 4.1.7 *The hyperarc-consistency procedure solves $\text{CSP}(\mathbf{H})$ if \mathbf{H} is an oriented path.*

Proof. We claim that the mapping $f : (2^B \setminus \emptyset) \rightarrow B$ defined by $f(S) = \min(S)$ is a homomorphism from $P(\mathbf{H})$ to \mathbf{H} .

Let (U, W) be in $E^{P(\mathbf{H})}$, and suppose for contradiction that $(\min(U), \min(W))$ is not an arc in \mathbf{H} . By the property of $P(\mathbf{H})$ there must be an element w of W such that $(\min(U), w) \in E^{\mathbf{H}}$,

and an element u of U such that $(u, \min(W)) \in E^{\mathbf{H}}$. Note that due to the way in which the vertices are numbered, if (k, l) is in $E^{\mathbf{H}}$ then $k - 1 \leq l \leq k + 1$. Hence,

$$\min(U) \leq w + 1 \leq \min(W) \leq u + 1 \leq \min(U),$$

and therefore both $(\min(U), w)$ and $(w, \min(U)) = (u, \min(W))$ are in $E^{\mathbf{H}}$, which is impossible in an oriented path. The corollary now follows from Theorem 4.1.4. \square

Exercises for Section 4.1.

Exercise 4.1.8 Determine $P(\mathbf{K}_3)$.

Exercise 4.1.9 Determine $P(\mathbf{H})$ where \mathbf{H} is the directed path with four vertices.

Exercise 4.1.10 Determine $P(\vec{\mathbf{T}}_3)$ for the transitive tournament $\vec{\mathbf{T}}_3$ on n vertices, that is, the directed graph with vertices $\{1, \dots, n\}$ where $(u, v) \in E^{\vec{\mathbf{T}}_n}$ if and only if $u < v$.

Exercise 4.1.11 Design an algorithm that establishes hyperarc-consistency in linear time in the size of the input instance. (Hint: maintain a ‘worklist’.)

Exercise 4.1.12 Let \mathbf{H} be a finite digraph. Show that there is a $U \subseteq H$ such that $(U, U) \in E^{P(\mathbf{H})}$ if and only if there is a directed cycle that homomorphically maps to \mathbf{H} .

Exercise 4.1.13 Show that the previous statement is false for infinite digraphs \mathbf{H} .

Exercise 4.1.14 A digraph is called balanced if it homomorphically maps to a directed path. Prove that if a directed graph \mathbf{H} is balanced, then $P(\mathbf{H})$ is balanced.

Exercise 4.1.15 There is only one unbalanced oriented cycle \mathbf{H} on four vertices that is a core (see Section 3.2) and not a directed cycle. Show that the hyperarc-consistency procedure does not solve $CSP(\mathbf{H})$.

Exercise 4.1.16 Suppose that \mathbf{B} is a relational structure on the domain $A = \{1, \dots, n\}$ with a binary polymorphism \min which maps its two arguments x, y to the minimum of x and y . Show that $CSP(\mathbf{A})$ can be solved by the hyperarc-consistency procedure. (Hint: Show that $P(\mathbf{B})$ homomorphically maps to \mathbf{B} .)

Exercise 4.1.17 Show that if \mathbf{A} is a finite relational structure with a binary polymorphism f that satisfies for all $x, y, z \in A$ the equations $f(x, f(y, z)) = f(f(x, y), z)$ (associativity), $f(x, y) = f(y, x)$ (commutativity), and $f(x, x) = x$ (idempotency), then $CSP(\mathbf{A})$ is in P . (Hint: see previous exercise.)

4.2 Local Consistency

In this section, we introduce a powerful generalization of the hyperarc-consistency procedure, called the k -consistency procedure. There are many constraint satisfaction problems that can be solved in polynomial time by the k -consistency procedure, but cannot be solved by the hyperarc-consistency procedure from Section 4.1.

Let \mathbf{B} be a structure over the relational signature τ . Let \mathbf{A} be an instance of $CSP(\mathbf{B})$, and let S be a subset of A . We say that $f : S \rightarrow B$ is a *partial solution* for \mathbf{A} if for all relation symbols R in τ and all tuples $(a_1, \dots, a_k) \in R^{\mathbf{A}}$ there is a tuple $(b_1, \dots, b_k) \in R^{\mathbf{B}}$ such that $f(a_i) = b_i$ for all $a_i \in S$.

<p>k-consistency(\mathbf{A}) Input: a finite relational structure \mathbf{A}. Output: a k-consistent instance of $\text{CSP}(\mathbf{B})$ equivalent to \mathbf{A}.</p> <p>Do For every subset $S = \{a_1, \dots, a_l\}$ of A with $l < k$ For every relation $R^{\mathbf{A}}$ and every tuple $(t_1, \dots, t_s) \in R^{\mathbf{A}}$ Let ϕ be the formula $Q_{\mathbf{A}}(S \cup \{t_1, \dots, t_s\}, (a_1, \dots, a_l))$ Add the tuple (a_1, \dots, a_l) to the relation $R_{\phi}^{\mathbf{A}}$</p> <p>Loop until no relation in \mathbf{A} is changed Return \mathbf{A}</p>

Figure 4.2: The k -consistency procedure for $\text{CSP}(\mathbf{B})$, where \mathbf{B} contains all at most $k - 1$ -ary primitive positive definable relations.

Definition 4.2.1 *An instance \mathbf{A} of $\text{CSP}(\mathbf{B})$ is called k -consistent² ($k \geq 1$) if every partial solution $f : S \rightarrow B$ for \mathbf{A} with $|S| < k$ can be extended to a partial solution with domain $S \cup \{a\}$ for any $a \in A$.*

We will show that any instance of $\text{CSP}(\mathbf{B})$ can be transformed into a k -consistent instance, under the assumption that \mathbf{B} contains all at most $k - 1$ -ary relations that have a primitive positive definition in \mathbf{B} . Similarly as in discussed in Section 4.1 this assumption is hardly restrictive.

Clearly, if \mathbf{B} is finite and k is fixed there is only a finite number of such relations. But also for many interesting infinite structures \mathbf{B} there is only a finite number of primitive positive definable relations of any fixed arity; several such structures \mathbf{B} will be discussed in Chapter 5. Similarly as in the definition of the *canonical conjunctive query* of \mathbf{A} in Section 1.2, we write $Q_{\mathbf{A}}(\{a_1, \dots, a_l\}, (a_{i_1}, \dots, a_{i_m}))$ for the conjunction of $R(t_1, \dots, t_k)$ over all relation symbols R of τ and tuples $(t_1, \dots, t_k) \in R^{\mathbf{A}} \cap \{a_1, \dots, a_l\}^k$, where all variables but a_{i_1}, \dots, a_{i_m} are existentially quantified (we always assume that $i_1, \dots, i_m \in \{1, \dots, l\}$). Clearly, this formula is primitive positive. The m -ary relation that is defined in \mathbf{B} by a formula $\phi(x_1, \dots, x_m)$ with free variables x_1, \dots, x_m will be denoted by $R_{\phi(x_1, \dots, x_m)}$.

We present a polynomial-time procedure, called the *k -consistency procedure*, that computes for a given instance \mathbf{A} of $\text{CSP}(\mathbf{B})$ a k -consistent instance \mathbf{A}' that is *equivalent* to \mathbf{A} in the sense that \mathbf{A}' and \mathbf{A} have the same set of solutions. The simple idea of the algorithm is to iteratively add a tuple (s_1, \dots, s_l) , for $l < k$, to the relation $R_{\phi}^{\mathbf{A}}$ if ϕ is a primitive positive formula that holds on (s_1, \dots, s_l) in \mathbf{A} . We can add this tuple to R_{ϕ} without changing the set of solutions, because for all solutions f the tuple $(f(s_1), \dots, f(s_l))$ must satisfy ϕ in \mathbf{B} .

The pseudo-code of the k -consistency procedure can be found in Figure 4.2.

Proposition 4.2.2 *Let \mathbf{B} be a finite structure that contains all at most $k - 1$ -ary relations with a primitive positive definition in \mathbf{B} . Then the k -consistency procedure as shown in Figure 4.2 computes for a given instance \mathbf{A} of $\text{CSP}(\mathbf{B})$ in polynomial time a k -consistent instance of $\text{CSP}(\mathbf{B})$ that is equivalent to \mathbf{A} .*

²In some papers, this form of consistency has been called *strong k -consistency*. We want to remark that many different, and sometimes closely related notions of k -consistency have been studied in the literature.

Proof. It is clear that the tuples we add to relations in the instance \mathbf{A} of $\text{CSP}(\mathbf{B})$ do not change the set of solutions for \mathbf{A} . If no tuples can be added to relations of \mathbf{A} in the loop of the algorithm, the resulting instance \mathbf{A} of $\text{CSP}(\mathbf{B})$ must be k -consistent: suppose otherwise that there is a partial solution $f : \{a_1, \dots, a_l\} \rightarrow B$, for $l < k$, such that f has no extension to a partial solution for \mathbf{A} with domain $S = \{a_1, \dots, a_l, a_{l+1}\}$. In other words, the tuple $(f(s_1), \dots, f(s_l))$ does not satisfy the primitive positive formula $\phi = Q_{\mathbf{A}}(S, (s_1, \dots, s_l))$. However, the procedure has added the tuple (s_1, \dots, s_l) to the relation $R_{\phi}^{\mathbf{A}}$, contradicting the assumption that f is a partial solution $\{s_1, \dots, s_l\} \rightarrow B$ for \mathbf{A} .

The running time of the k -consistency procedure is for any fixed k and fixed \mathbf{B} polynomial in the size of \mathbf{A} : since the number of at most $k - 1$ -ary relations with a primitive positive definition in \mathbf{B} is for fixed \mathbf{B} constant, there is only a polynomial number of tuples we might add to relations in \mathbf{A} . Since the inner loop is also polynomially bounded, the total running time is polynomial in the size of \mathbf{A} . We want to point out that this is not the most efficient algorithm for establishing k -consistency – see Exercise 4.2.8. \square

With the same argument we can show that Proposition 4.2.2 also holds if \mathbf{B} is an infinite structure, as long as \mathbf{B} has a finite number of primitive positive definable relations of arity at most $k - 1$.

We now define formally what it means that the k -consistency procedure *solves* $\text{CSP}(\mathbf{B})$. We shall assume that there is an unsatisfiable instance \mathbf{A} of $\text{CSP}(\mathbf{B})$, since otherwise $\text{CSP}(\mathbf{B})$ would be trivial. Then the primitive positive formula $\phi = Q_{\mathbf{A}}(A, ())$ defines the empty 0-ary relation over \mathbf{B} . We also write *false* for the relation symbol R_{ϕ} . Finally, we say that the k -consistency procedure *derives false* if in the instance \mathbf{A}' that is computed by the procedure the relation $\text{false}^{\mathbf{A}'}$ is non-empty.

Definition 4.2.3 *The k -consistency procedure solves $\text{CSP}(\mathbf{B})$ if the k -consistency procedure derives false for a given input instance \mathbf{A} if and only if there is no homomorphism from \mathbf{A} to \mathbf{B} .*

It is an important open problem to describe those structures \mathbf{B} where the k -consistency procedure solves $\text{CSP}(\mathbf{B})$. It is not known whether there exists an algorithm that takes \mathbf{B} as input and decides whether $\text{CSP}(\mathbf{B})$ can be solved with the k -consistency procedure. However, as we will see in the remainder of this section, a powerful sufficient condition is known.

Exercises for Section 4.2.

Exercise 4.2.4 *Verify that the 2-consistency procedure derives false on \mathbf{A} if and only if the hyperarc-consistency algorithm rejects \mathbf{A} .*

Exercise 4.2.5 *Determine the number of at most binary primitive positive definable relations in \mathbf{B}^{∞} (defined in Exercise 1.4.7).*

Exercise 4.2.6 *Show that the 3-consistency procedure solves $\text{CSP}(\mathbf{B}^{\infty})$.*

Exercise 4.2.7 *Determine the number of at most binary primitive positive definable relations in the structure \mathbf{PA} introduced in Section 1.3.*

Exercise 4.2.8 *Develop an algorithm that computes for a given instance \mathbf{A} of $\text{CSP}(\mathbf{B})$ an algorithm that computes in time less than Cn^k , where C is a constant and n is the size of the input \mathbf{A} , a k -consistent instance \mathbf{A}' that is equivalent to \mathbf{A} . (Hint: use a ‘worklist’.)*

4.3 Near-Unanimity Functions

We present a result by Feder and Vardi [15] that shows that if \mathbf{B} has a polymorphism that satisfies certain identities, then the k -consistency procedure solves $\text{CSP}(\mathbf{B})$. This result implies many earlier results in Artificial Intelligence and graph theory. We also touch extensions of this result for infinite right-hand sides \mathbf{B} .

Definition 4.3.1 *Let B be a set, and $k \geq 3$. A function f from B^k to B is called a (k -ary) near-unanimity function (short, an nuf) if f satisfies the following equations, for all $x, y \in B$:*

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

A ternary near-unanimity function is called a *majority function*. The operation `major` defined in Section 3.3 is an example of a majority operation over the two-element domain (actually, over a two-element domain there can be only one majority operation – exercise). For an example of a majority operation on a larger domain, let B be any set that is linearly ordered by \leq , and consider the ternary *median* operation, which is defined as follows. Let x, y, z be three elements from B . Suppose that $\{x, y, z\} = \{a, b, c\}$, where $a \leq b \leq c$. Then $\text{median}(x, y, z)$ is defined to be b . Many more examples of near-unanimity operations will follow soon.

We will show that if \mathbf{B} has a polymorphism that is a near-unanimity function, then $\text{CSP}(\mathbf{B})$ can be solved in polynomial time by the k -consistency procedure. In the proof presented below we actually show something stronger. To describe the result, we introduce the following concept.

Definition 4.3.2 *An instance \mathbf{A} of $\text{CSP}(\mathbf{B})$ is called globally consistent if it is k -consistent for all $k \geq 1$.*

Note that there are unsatisfiable instances \mathbf{A} that are globally consistent: this is the case if the 0-ary tuple is in $\text{false}^{\mathbf{A}}$.

Lemma 4.3.3 *A globally consistent instance \mathbf{A} of $\text{CSP}(\mathbf{B})$ has a solution if and only if $\text{false}^{\mathbf{A}}$ is empty.*

Proof. It is clear that if \mathbf{A} has a solution then $\text{false}^{\mathbf{A}}$ must be empty. For the opposite direction, we can iteratively construct a solution to \mathbf{A} as follows. We know that the mapping with the empty domain is a partial solution for \mathbf{A} , because $\text{false}^{\mathbf{A}}$ is empty. If we have already constructed a partial solution with a domain of size l , we can extend it to any other variable $a \in A$ such that the resulting map is again a partial solution, because \mathbf{A} is in particular $l + 1$ -consistent. If we have found a partial solution that is defined on all elements of A , we have found a solution for \mathbf{A} . \square

Theorem 4.3.4 (of [15]) *Let \mathbf{B} be a finite relational structure. If \mathbf{B} has a polymorphism that is a k -ary near-unanimity function, then the k -consistency procedure computes globally consistent instances.*

Proof. Suppose that \mathbf{B} has a k -ary near-unanimity polymorphism $f : B^k \rightarrow B$. Let \mathbf{A} be the instance computed by the k -consistency procedure. The proof shows by induction on i

that any partial solution of \mathbf{A} with a domain $S = \{a_1, \dots, a_i\}$ of size i can be extended to any other variable x in A such that the resulting mapping is a partial solution.

The base case is that $1 \leq i \leq k - 1$. Suppose for contradiction that there is no way to extend h to a partial solution with domain $S \cup \{x\}$. Let ϕ be the formula $Q_{\mathbf{A}}(S, (a_1, \dots, a_i))$. Then the k -consistency procedure must have added the tuple (a_1, \dots, a_i) to $R_{\phi}^{\mathbf{A}}$. But note that ϕ does not hold for $(h(a_1), \dots, h(a_i))$, because we can not extend h to x . This is a contradiction to the assumption that h is a partial solution.

For the inductive step, let h be any partial solution of \mathbf{A} with domain S , for $|S| = i \geq k$. Let x_1, \dots, x_k be distinct elements of S , and let h_j be the restriction of h to $S \setminus \{x_j\}$, for $1 \leq j \leq k$. By inductive assumption, h_j can be extended to x such that the resulting mapping h'_j is a partial solution of \mathbf{A} . We claim that the extension h' of h that maps x to $f(h'_1(x), \dots, h'_k(x))$ is partial solution for \mathbf{A} .

Let (u_1, \dots, u_s) be a tuple in a relation $R^{\mathbf{A}}$. Since each h'_j is a partial solution, for each $j \in \{1, \dots, k\}$ there exists a tuple $(v_1^j, \dots, v_s^j) \in R^{\mathbf{B}}$ such that $h'_j(u_t) = v_t^j$ when h'_j is defined on u_t . Since f is a polymorphism, the tuple $(f(v_1^1, \dots, v_s^1), \dots, f(v_s^1, \dots, v_s^k))$ is in $R^{\mathbf{B}}$ as well. We claim that if h' is defined on u_t then $h'(u_t) = f(v_t^1, \dots, v_t^k)$. We distinguish the following cases:

- $u_t = x_j$ for some $j \in \{1, \dots, k\}$. For $l \in \{1, \dots, k\} \setminus \{j\}$ we have that $h_l(u_t)$ is defined and equal to v_t^l . So, all but one of the values v_t^1, \dots, v_t^k are equal to $h(u_t)$ and thus $h'(u_t) = h(u_t) = f(v_t^1, \dots, v_t^k)$.
- $u_t \in S \setminus \{x_1, \dots, x_k\}$. For each $l \in \{1, \dots, k\}$, it holds that $h'(u_t) = h(u_t) = h_l(u_t) = v_t^l$ and thus $h'(u_t) = f(v_t^1, \dots, v_t^k)$.
- $u_t = x$. We have $h'(x) = f(h'_1(x), \dots, h'_k(x)) = f(v_t^1, \dots, v_t^k)$.

This concludes the proof that h' is a partial solution for \mathbf{A} . \square

As a corollary, we obtain that $\text{CSP}(\mathbf{B})$ can be solved in polynomial time if \mathbf{B} has the boolean domain $\{0, 1\}$ and the operation *major* as a polymorphism; this is a result that was already mentioned in Section 3.3.

Many classes of directed graphs have a majority polymorphism. If \mathbf{H} is a *transitive tournament* (defined in Exercise 4.1.10), then the median operation is a polymorphism of \mathbf{H} , because if $u_1 < v_1$, $u_2 < v_2$, and $u_3 < v_3$, then clearly $\text{median}(u_1, u_2, u_3) < \text{median}(v_1, v_2, v_3)$. Hence, Theorem 4.3.4 implies that $\text{CSP}(\mathbf{H})$ can be solved in polynomial time.

Another class of examples of graphs having a majority polymorphism are *unbalanced cycles*, i.e., oriented cycles that do not homomorphically map to a directed path [14]. We only prove a weaker result here.

Proposition 4.3.5 *Directed cycles have a majority polymorphism.*

Proof. Let \mathbf{H} be a directed cycle. Let f be the ternary operation on H that maps $u, v, w \in H$ to u if u, v, w are pairwise distinct, and otherwise acts as a majority operation. We claim that f is a polymorphism of \mathbf{H} . Let $(u, u'), (v, v'), (w, w') \in E^{\mathbf{H}}$. If u, v, w are all distinct, then u', v', w' are clearly all distinct as well, and hence $(f(u, v, w), f(u', v', w')) = (u, u') \in E^{\mathbf{H}}$. Otherwise, if two elements of u, v, w are equal, say $u = v$, then u' and v' must be equal as well, and hence $(f(u, v, w), f(u', v', w')) = (u, u') \in E^{\mathbf{H}}$. \square

It has been shown in [15] and [18] that for finite structures \mathbf{B} the k -consistency procedure computes globally consistent instances of $\text{CSP}(\mathbf{B})$ if and only if \mathbf{B} has a polymorphism that is a k -ary near-unanimity function. We shall soon see examples that this is no longer true for infinite structures \mathbf{B} : many infinite structures do not have a near-unanimity operation, but still the k -consistency procedure computes globally consistent instances. We therefore introduce a slightly relaxed notion of near-unanimity.

Quasi Near-unanimity functions. A function f from B^k to B is called a (k -ary) *quasi near unanimity function* (short, an *qnuf*) if f satisfies the following equations, for all $x, y \in B$:

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

A ternary quasi near-unanimity function is also called a *quasi majority function*. Consider the structure \mathbf{B}^∞ with domain \mathbb{N} and the signature $\{=, \neq\}$ that was defined in Exercise 1.4.7. This structure has a quasi majority polymorphism (but no majority polymorphism, see Exercise 4.3.14). The idea is that *any* mapping from \mathbb{N}^3 to \mathbb{N} that satisfies the identities of a quasi near-unanimity operation, and is ‘*injective otherwise*’ will do the job. To make this precise, let f be any bijection from \mathbb{N}^3 to \mathbb{N} . Such bijections exist. Let g be defined as follows: for $x, y \in \mathbb{N}$, let $g(x, x, y), g(x, y, x), g(y, x, x), g(x, x, x)$ all be equal to $\min_{z \in \mathbb{N}} \{f(x, x, z), f(x, z, x), f(z, x, x)\}$. If $\{x, y, z\} = 3$, let $g(x, y, z) = f(x, y, z)$.

Proposition 4.3.6 *The function g described above is a quasi majority polymorphism of the structure \mathbf{B}^∞ .*

Proof. From the construction it is clear that g satisfies the equations of quasi near-unanimity functions. Every operation preserves the relation $=^{\mathbf{B}^\infty}$. We show that also the relation $\neq^{\mathbf{B}^\infty}$ is preserved by g . Let $(u_1, v_1), (u_2, v_2), (u_3, v_3) \in \neq^{\mathbf{B}^\infty}$. If $\{u_1, u_2, u_3\} = 3$ or $\{v_1, v_2, v_3\} = 3$ then $g(u_1, u_2, u_3) \neq g(v_1, v_2, v_3)$, by injectivity of f . Otherwise, $\{u_1, u_2, u_3\} = 2$ and $\{v_1, v_2, v_3\} = 2$. Hence, if $f(u_1, u_2, u_3) = f(v_1, v_2, v_3)$, then $u_1 = v_1$, $u_2 = v_2$, or $u_3 = v_3$ must be true, in contradiction to our assumption. \square

More results on local and global consistency (e.g. for the structure \mathbf{PA} introduced in Section 1.3) and the connection to quasi near-unanimity polymorphisms can be found in [4].

Exercises for Section 4.3.

Exercise 4.3.7 *Let \mathbf{B} be the structure with signature $\{\leq, <\}$ and domain \mathbb{Q} , with the usual interpretation of \leq and $<$ over the rational numbers. What is the computational complexity of $\text{CSP}(\mathbf{B})$?*

Exercise 4.3.8 *Let \mathbf{K}_2 be the complete undirected graph with two vertices (defined similarly to \mathbf{K}_3 and \mathbf{K}_4 in Section 1.1). Show that \mathbf{K}_2 has a majority polymorphism.*

Exercise 4.3.9 *Show that every orientation of a path (we introduced orientations of paths at the end of Section 4.1) has a majority polymorphism.*

Exercise 4.3.10 *There is only one unbalanced cycle \mathbf{H} on four vertices that is a core and not the directed cycle (we have seen this graph already in Exercise 4.1.15). Show that $\text{CSP}(\mathbf{H})$ can be solved by the 3-consistency procedure.*

Exercise 4.3.11 An interval graph \mathbf{H} is an (undirected) graph such that there is an interval I_x of the real numbers for each $x \in H$, and $(x, y) \in E^{\mathbf{H}}$ iff I_x and I_y have a non-empty intersection. Note that with this definition interval graphs are necessarily reflexive, i.e., $(x, x) \in E^{\mathbf{H}}$. Let \mathbf{H}_1 be the expansion of an interval-graph by all singletons. Show that $\text{CSP}(\mathbf{H}_1)$ can be solved in polynomial time.

Exercise 4.3.12 Hell and Nešetřil proved that if an undirected graph \mathbf{H} does not homomorphically map to \mathbf{K}_2 (see Exercise 4.3.8), then $\text{CSP}(\mathbf{H})$ is NP-complete. Use this result to show that for every undirected graph \mathbf{H} there is a polynomial-time algorithm that solves $\text{CSP}(\mathbf{H})$ if \mathbf{H} has a quasi majority polymorphism, and that $\text{CSP}(\mathbf{H})$ is NP-complete otherwise.

Exercise 4.3.13 Show that 3-consistency implies global consistency for $\text{CSP}(\mathbf{B}^\infty)$.

Exercise 4.3.14 Show that \mathbf{B}^∞ has no near-unanimity operation.

Chapter 5

Graph Algorithms

In this chapter we present efficient graph algorithms for constraint satisfaction problems. The problems we study here can usually not be solved by the k -consistency procedure. They all have in common that they work on appropriately defined (directed or undirected) graphs on the variables of the constraint satisfaction problem. This graph then guides a recursive decomposition procedure that correctly decides the CSP. Moreover, it is usually easy to adapt this “divide-and-conquer” procedure such that it directly constructs a solution for the CSP. How these graphs are defined, and how the decomposition is performed highly depends on the CSP under consideration. However, we will see examples of this approach in various application areas, and it turns out that the algorithms and the correctness proofs have quite some similarities.

5.1 Temporal Reasoning

As a warm-up, we present a simple algorithm for a CSP called the *min-ordering problem*. In this problem, we are given a set V of variables, and a set of triples on these variables. We want to find a mapping $h : V \rightarrow \mathbb{Q}$ that assigns the variables to rational numbers such that for each triple (x, y, z) either $h(x) > h(y)$ or $h(x) > h(z)$. This problem can be formalized as a $\text{CSP}(\mathbf{M})$, where \mathbf{M} is a relational structure whose domain are the rational numbers, and which has a single ternary relation

$$T^{\mathbf{M}} = \{(x, y, z) \mid x > y \vee x > z\}.$$

The relation $T^{\mathbf{M}}(x, y, z)$ specifies that x is larger than one of y, z ; equivalently, that x is larger than the minimum of y and z . This is an example of a *temporal* constraint satisfaction problem, because we can imagine the elements of \mathbb{Q} as *time points*. It can be shown [6] that for all k the k -consistency procedure does not solve the min-ordering problem. However, here is a simple linear-time algorithm for $\text{CSP}(\mathbf{M})$.

Note that \mathbf{M} has the binary operation *min* as a polymorphism, where $\min(x, y)$ is by definition the minimum of x and y (Exercise 5.1.6). The operation *min* is an example of an associative, commutative, and idempotent operation (an *ACI operation*). We have seen in Exercise 4.1.17 in Chapter 4 that if \mathbf{B} is a finite structure with an ACI polymorphism, then the generalized arc-consistency procedure solves $\text{CSP}(\mathbf{B})$. This result does not carry over to infinite structures \mathbf{B} (we have already mentioned that \mathbf{M} has an ACI polymorphism, but cannot be solved by the k -consistency procedure).

The algorithm that we present for the min-ordering problem is not yet a graph algorithm, but it illustrates several of the ideas that are employed later for more complicated problems. In Exercise 5.1.8 we will see that the algorithm can be generalized to other temporal constraint satisfaction problems $\text{CSP}(\mathbf{B})$ if \mathbf{B} is closed under the operation *min* (or, similarly, *max*).

Let \mathbf{A} be an instance of $\text{CSP}(\mathbf{M})$ that has a solution h (i.e., h is a homomorphism from \mathbf{A} to \mathbf{M}). Clearly there must be a variable $x \in A$ such that $f(x) \leq f(y)$ for all $y \in A$.

Definition 5.1.1 *A set $S \subset A$ of the an instance \mathbf{A} of $\text{CSP}(\mathbf{B})$ is called free if \mathbf{A} has a solution h such that for all $y \in A$ and $x \in S$ it holds that $h(x) \leq h(y)$.*

If x is from a free set of variables S , then $T^{\mathbf{A}}$ cannot contain a triple (x, y, z) where x appears in the first entry, because then either $h(y)$ or $h(z)$ are strictly smaller than $h(x)$, a contradiction. We say that a variable $x \in A$ is *blocked* if $T^{\mathbf{A}}$ contains a tuple (x, y, z) where x appears in the first entry.

Lemma 5.1.2 *If an instance of the min-ordering problem only contains blocked variables, then it is inconsistent.*

Proof. Suppose the instance has a solution, then there must exist a non-empty free set. But as we have seen, blocked variables cannot be in a free set. Since all variables are blocked, we obtain a contradiction. \square

It turns out that if an instance of the min-ordering problem is satisfiable, then the set of *all* variables that are not blocked is free. This follows from the correctness of the following algorithm. In the algorithm (and also in the following sections), we write $\mathbf{A}[A']$, for $A' \subseteq A$, for the relational structure \mathbf{A}' with domain A' and the same signature as \mathbf{A} where $(t_1, \dots, t_s) \in R^{\mathbf{A}'}$ if $(t_1, \dots, t_s) \in R^{\mathbf{A}}$, for every relation symbol R in the signature.

```

Min-Ordering( $\mathbf{A}$ )
Input: A finite structure  $\mathbf{A}$  over  $\{T\}$  where  $T$  is a ternary relation symbol.

If  $A = \emptyset$  then accept
else
  Compute the set  $S \subseteq A$  of variables that are not blocked;
  If  $S = \emptyset$  then reject
  else Min-Ordering( $\mathbf{A}[A \setminus S]$ )
end if

```

Figure 5.1: The Min-Ordering algorithm.

Proposition 5.1.3 *The Min-Ordering procedure given in Figure 5.1 correctly decides the min-ordering problem in linear time in the input size.*

Proof. Let \mathbf{A} be an instance of the min-ordering problem. If at some point during the execution of the procedure on \mathbf{A} we recursively apply the procedure to a substructure \mathbf{A}' of \mathbf{A} and do not find a min-candidate, then Lemma 5.1.2 implies that \mathbf{A}' and therefore also \mathbf{A} does not have a solution.

Otherwise, it is clear that \mathbf{A} has a solution if the set of variables is empty. So, suppose the algorithm finds a non-empty set S . Inductively assume that $\mathbf{A}[A \setminus S]$ has a solution h . Let h' be the extension of h that maps all variables in S to a value smaller than the value of $h(y)$ for all other variables $y \in A \setminus S$ (such values always exists in \mathbb{Q}). We claim that h' is a homomorphism from \mathbf{A} to \mathbf{M} . If $(x, y, z) \in T^{\mathbf{A}}$, and $x, y, z \notin S$, then this is true by inductive assumption. Otherwise, since x is blocked, only y, z , or both y and z can be in S . In all three cases the triple $(h'(x), h'(y), h'(z))$ is in $T^{\mathbf{M}}$. Hence, the algorithm is correct. It is not hard to see that this procedure can be implemented such that it has a linear worst-case running time. \square

Exercises for Section 5.1.

Exercise 5.1.4 Let \mathbf{B} be the structure with signature $\{\leq, <\}$ and domain \mathbb{Q} , with the usual interpretation of \leq and $<$ over the rational numbers. Show that there is a linear time algorithm for $\text{CSP}(\mathbf{B})$.

Exercise 5.1.5 Show that if an instance of the min-ordering problem $\text{CSP}(\mathbf{M})$ has a solution, then it also has an injective solution (i.e., no two variables receive the same value). Hint: modify the presented algorithm for $\text{CSP}(\mathbf{M})$ such that it produces injective solutions.

Exercise 5.1.6 Show that $\text{CSP}(\mathbf{PA})$ (as introduced in Section 1.1) can be solved in polynomial time. How fast is the best algorithm you can find?

Exercise 5.1.7 Show that the structure \mathbf{M} introduced in this section has indeed the operation \min as a polymorphism.

Exercise 5.1.8 Show that $\text{CSP}(\mathbf{B})$ can be solved in polynomial time, where \mathbf{B} is a structure with a ternary relation $R^{\mathbf{B}}$ consisting of all $(x, y, z) \in \mathbb{Q}^3$ where $x = y \wedge y < z$ or $x > y \wedge y = z$ or $x = y = z$. Additionally, \mathbf{B} has the relation $<^{\mathbf{B}}$ with the usual interpretation.

5.2 Phylogenetic Reconstruction

The *rooted triple consistency problem* is one of the fundamental problems in phylogenetic reconstruction. It is motivated by the problem to reconstruct evolutionary trees from partial information about the evolutionary tree. When we acquire such partial information, one of the important tasks is it to test whether the collection of partial information is *consistent* in the sense that there is an evolutionary tree that ‘explains’ the data. Consistency in this sense should not be confused with the notions of consistency introduced in Chapter 4. In 1981, Aho, Sagiv, Szymanski, and Ullman [1] presented a polynomial time algorithm to this problem, motivated independently from computational biology by questions in database theory.

We fix some standard terminology concerning *rooted trees*. A *rooted tree* is a directed graph \mathbf{T} with a distinguished vertex r , the *root* of \mathbf{T} , such that for every vertex $v \in T$ there is a unique directed path $r = p_1, \dots, p_l = v$ with $(p_i, p_{i+1}) \in E^{\mathbf{T}}$ for all $i < l$ (we say that p_{i+1} is the *child* of p_i). For $u, v \in V$, we say that u *lies below* v if the directed path from r to u passes through v . We say that u *lies strictly below* v if u lies below v and $u \neq v$. The *youngest common ancestor (yca)* of two vertices $u, v \in T$ is the node w such that both u and v lie below w and w has maximal distance from r . A *leave* is a vertex v in \mathbf{T} without any outgoing edges, i.e., there is no edge $(v, w) \in E^{\mathbf{T}}$.

Rooted-Triple-Consistency

INSTANCE: A finite structure \mathbf{A} over $\{R\}$ where R is a ternary relation symbol.

QUESTION: Is there a rooted tree \mathbf{T} with leaves X and a mapping $h : A \rightarrow X$ such that for every triple $(x, y, z) \in R^{\mathbf{A}}$ the yca of $h(x)$ and $h(y)$ lies strictly below the yca of $h(x)$ and $h(z)$ in \mathbf{T} ?

Once you understand the setting, this is a fairly natural computational problem. It can be formulated as a constraint satisfaction problem $\text{CSP}(\mathbf{B})$ for a fixed, but infinite structure \mathbf{B} . The domain of \mathbf{B} is $\mathbb{N} \rightarrow \mathbb{Q}$, i.e., the set of all functions from the natural numbers to the rational numbers (hence, the domain of \mathbf{B} is uncountable; we can also find formulations of the problem with a countably infinite right hand side, but it will be convenient for us to use an uncountably infinite structure \mathbf{B} here). For two elements f, g of B , let $k_{f,g}$ be the largest natural number such that $f(i) = g(i)$ for all $i < k_{f,g}$. The ternary relation $R^{\mathbf{B}}$ is the relation consisting of all triples (f, g, h) such that either $k_{f,g} > k_{f,h}$ or $k_{f,g} = k_{f,h}$ and $f(k_{f,g}) < h(k_{f,h})$. The proof that $\text{CSP}(\mathbf{B})$ is indeed the rooted triple consistency problem is left as Exercise 5.2.3.

We would like to remark that for all l there is only a finite number of l -ary first-order definable relations in \mathbf{B} ; for related results consult see [10]. Hence, the k -consistency procedure can be applied for $\text{CSP}(\mathbf{B})$. But it is not known whether the k -consistency procedure solves $\text{CSP}(\mathbf{B})$, for some k (it is known that the 3-consistency procedure does not, though).

The ASSU algorithm for rooted triple consistency. The basic idea of the algorithm of Aho et al. is to consider for a given instance \mathbf{A} of the rooted triple consistency problem an (undirected) graph \mathbf{G} with vertex set $G = A$ and where

$$E^{\mathbf{G}} = \{\{x, y\} \mid x, y \in A \text{ and there is } z \in A \text{ so that } (x, y, z) \in R^{\mathbf{A}}\}.$$

We say that \mathbf{G} is *connected* if for any two vertices $a, b \in G$ there exists a path $a = p_1, \dots, p_n = b$ with $\{p_i, p_{i+1}\} \in E^{\mathbf{G}}$ for all $i < n$.

Lemma 5.2.1 *If the graph \mathbf{G} defined for an instance \mathbf{A} of the rooted triple consistency problem is connected, then \mathbf{A} is unsatisfiable.*

Proof. Suppose that there is a rooted tree \mathbf{T} with leaf set X and a mapping $h : A \rightarrow X$ such that for every triple $(x, y, z) \in R^{\mathbf{A}}$ the yca of x, y lies strictly below the yca of x, z in \mathbf{T} .

Let s be the yca of $h(A)$, i.e., s is the node with the maximal distance from the root of T such that for every $a \in A$ the node $h(a)$ lies below s . It can not be that all vertices in $h(A)$ lie below the same child of s in \mathbf{T} (otherwise the child would have been the yca of $h(A)$). Since the graph \mathbf{G} is connected, there is $\{x, y\} \in \mathbf{G}$ such that $h(x)$ and $h(y)$ lie below different children of s in \mathbf{T} . Hence, there is a $z \in A$ such that $(x, y, z) \in R^{\mathbf{A}}$. By assumption, the yca of $h(x)$ and $h(y)$, which is s , lies strictly below the yca of $h(x)$ and $h(z)$, a contradiction to the choice of s . \square

We say that \mathbf{H} is a *subgraph* of \mathbf{G} if $H \subseteq G$ and $E^{\mathbf{H}} \subseteq E^{\mathbf{G}}$. A *connected component* \mathbf{H} of \mathbf{G} is connected subgraph of \mathbf{G} where H is maximal. Figure 5.2 shows the pseudo-code of an algorithm for the rooted triple consistency problem that for a given instance \mathbf{A} also constructs a tree \mathbf{T} and a mapping h from A to the leaves of \mathbf{T} satisfying the conditions from the problem description, in case that such a tree exists.

```

ASSU(A)
Input: a finite structure A over  $\{R\}$  where  $R$  is a ternary relation symbol.

If  $R^{\mathbf{A}} = \emptyset$  then
    Create a new vertex  $r$ , and link all  $a \in A$  below  $r$ .
    return the tree rooted at  $r$ 
else
    Construct the graph G for A;
    Compute the connected components  $\mathbf{G}_1, \dots, \mathbf{G}_k$  of  $G$ ;
    If there is only one connected component then reject
    else
        Let  $\mathbf{T}_i$  be the tree ASSU( $\mathbf{A}[G_i]$ ), for all  $i \leq k$ ;
        Create a new vertex  $r$ , and link  $\mathbf{T}_i$  below  $r$  for all  $i \leq k$ ;
        return the tree rooted at  $r$ 
    end if
end if

```

Figure 5.2: The ASSU algorithm.

Proposition 5.2.2 *The algorithm shown in Figure 5.2 correctly decides the rooted triple consistency problem in time that is quadratic in the size of the input instance.*

Proof. Clearly, if $R^{\mathbf{A}}$ is empty, any rooted tree with at least one vertex is a solution. If the constructed graph \mathbf{G} is connected, Lemma 5.2.1 shows that \mathbf{A} is unsatisfiable. Otherwise, let $\mathbf{G}_1, \dots, \mathbf{G}_k$ be the distinct connected components of \mathbf{G} (they can be computed using a standard linear time algorithm for computing the connected components of a graph). For each \mathbf{G}_i , recurse on the problem $\mathbf{A}[G_i]$, i.e., the structure with domain G_i and with a ternary relation that contains all triples $(x, y, z) \in R^{\mathbf{A}}$ for $x, y, z \in G_i$. If any of these recursive calls reports unsatisfiability, then the instance is clearly unsatisfiable as well. Otherwise, by inductive assumption there is a tree \mathbf{T}_i and a mapping $h_i : G_i \rightarrow T_i$ showing that $\mathbf{A}[G_i]$ is consistent. Let \mathbf{T} be the tree obtained by creating a new vertex r and adding $\mathbf{T}_1, \dots, \mathbf{T}_k$ by making the roots of these trees the children of r . Let h be the mapping that maps x to $h_i(x)$ if $x \in G_i$. We claim that \mathbf{T} and h are a solution for \mathbf{A} . If $(x, y, z) \in R^{\mathbf{A}}$ and x, y, z all are from one component G_i , there is nothing to show, since \mathbf{T}_i and h_i guarantee that the yca of $h(x)$ and $h(y)$ lies strictly below the yca of $h(x)$ and $h(z)$. It can not be that x and y are in distinct components, since they are connected by an edge in \mathbf{G} . Hence, all other tuples must be of the form that $x, y \in G_i$ and $z \in G_j$ for $i \neq j$. But in this case the yca of $h(x)$ and $h(y)$ lies below the root of \mathbf{T}_i and hence strictly below r , which is the yca of $h(x)$ and $h(z)$. \square

Exercises for Section 5.2.

Exercise 5.2.3 *Prove the claim of the section that the rooted triple consistency problem can be formulated as CSP(\mathbf{B}) for the infinite structure \mathbf{B} that was described above.*

Exercise 5.2.4 *Show that the running time of the ASSU algorithm is in $O(nm)$, where $n = |A|$ and m is the size of the input structure \mathbf{A} .*

5.3 Branching Time Constraints

Our next example is a computational problem that has been discovered independently in computational linguistics and in temporal reasoning. In temporal reasoning the problem is known as constraint satisfaction problem for constraints about *branching time*. In computational linguistics, the problem is a special case of a more powerful tree description language that has been introduced by Cornell [11]. There are other constraint languages in computational linguistics that talk about trees [13, 20, 21]; many of them can be solved in polynomial time with the algorithmic techniques described in this Chapter [5].

The first polynomial-time algorithm for this problem is due to Hirsch [17], and has a worst-case running time in $O(n^5)$. This was later improved by Jonsson and Broxvall [9], who presented an algorithm running in $O(n^{3.376})$ (this algorithm uses an $O(n^{2.376})$ algorithm for fast integer matrix multiplication). The simple algorithm we present here does not use fast matrix multiplication and runs in $O(nm)$, and is due to [7]. Our algorithm combines the ideas from the algorithms for the min-ordering problem in Section 5.1 and the rooted triple consistency problem in Section 5.2. For terminology concerning rooted trees, see Section 5.2.

Branching-Time-Consistency

INSTANCE: A finite structure \mathbf{A} over $\{\leq, \parallel, \neq\}$ where \leq, \parallel, \neq are binary relation symbols.

QUESTION: Is there a rooted tree \mathbf{T} and a mapping $\alpha : V \rightarrow T$ such that in \mathbf{T} the following is satisfied:

- a) If $x \leq y$, then $\alpha(x)$ lies above $\alpha(y)$;
- b) If $x \parallel y$, then neither $\alpha(x)$ lies strictly above $\alpha(y)$ nor $\alpha(y)$ lies strictly above $\alpha(x)$;
- c) If $x \neq y$, then $\alpha(x) \neq \alpha(y)$.

This problem can be formulated as $\text{CSP}(\mathbf{B})$ where \mathbf{B} is an infinite structure over the signature that consists of the binary relation symbols \leq, \parallel , and \neq . The domain B of \mathbf{B} 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 a proper initial subsequence of a , i.e., $b = (q_0, \dots, q_{n-1})$ and $a = (q_0, \dots, q_{n-1}, q_n, \dots, q_m)$, or
- $b = (q_0, \dots, q_{n-1}, q_n)$ and $a = (q_0, \dots, q_{n-1}, q'_n, q_{n+1}, \dots, q_m)$, and $q_n \leq q'_n$.

We can now define the relations of \mathbf{B} :

- Let $(a, b) \in \leq^{\mathbf{B}}$ if $a < b$ or $a = b$.
- Let $(a, b) \in \neq^{\mathbf{B}}$ if $a \neq b$.
- Let $(a, b) \in \parallel^{\mathbf{B}}$ if neither $x < y$ nor $y < x$.

We usually use infix-notation for these three relations, i.e., we write $a \parallel^{\mathbf{B}} b$ for $(x, y) \in \parallel^{\mathbf{B}}$. The definition of $\parallel^{\mathbf{B}}$ is symmetric, and therefore we do not distinguish between the constraint $x \parallel y$ and $y \parallel x$.

It can be shown (Exercise 5.3.7) that $\text{CSP}(\mathbf{B})$ is indeed the branching time consistency problem. Note that $\leq^{\mathbf{B}}$ is a partial order with the property that for all $x \in B$, the set $\{y \mid y \leq x\}$ is linearly ordered. This is the mentioned connection to temporal reasoning and *branching time*: we assume that for every time point the past is linearly ordered, but the

future is only partially ordered. Moreover, it can be shown that \mathbf{B} has for every l a finite number of l -ary first-order definable relations.

For the description of the algorithm, though, we decide to return to the original description of the branching time consistency problem, and do not make further reference to \mathbf{B} . A *solution* to an instance \mathbf{A} of the branching time consistency problem is thus a pair (\mathbf{T}, h) where \mathbf{T} is a rooted tree and $h : A \rightarrow \mathbf{T}$ is a mapping such that the conditions specified for \mathbf{T} and h in the problem description are satisfied.

Definition 5.3.1 *A subset of variables S of an instance \mathbf{A} of the branching-time-consistency problem is called free if there exists a solution (\mathbf{T}, h) such that for all $a \in A$ the vertex $h(a)$ is the root of \mathbf{T} if and only if $a \in S$.*

Let \mathbf{A} be an instance of the branching-time-consistency problem. Let \mathbf{U} be the undirected graph with vertex set $U = A$ where $\{u, v\} \in E^{\mathbf{U}}$ if $u \leq^{\mathbf{A}} v$ or $v \leq^{\mathbf{A}} u$.

Lemma 5.3.2 *Let \mathbf{A} be a satisfiable instance of the branching-time-consistency problem, and let \mathbf{U} be the graph as described above. If \mathbf{U} is connected, then there exists a free set of variables.*

Proof. Let (\mathbf{T}, h) be a solution of \mathbf{A} . Assume for contradiction that there are two distinct elements $x, x' \in A$ with the property that there is no $y \in A$ such that both x and x' lie below y in \mathbf{T} . Since \mathbf{U} is connected, there must be a sequence $x = x_0, \dots, x_k = x'$ of vertices in A such that $x_i \leq^{\mathbf{A}} x_{i+1}$ or $x_{i+1} \leq^{\mathbf{A}} x_i$ for all $0 \leq i < k$. It is easy to see that the sequence $h(x_0), \dots, h(x_k)$ must contain a vertex $h(x_j)$ such that both $h(x)$ and $h(x')$ lie below $h(x_j)$ in \mathbf{T} . This contradicts the assumption for x and x' . \square

Definition 5.3.3 *A set of variables $S \subseteq V$ of an instance \mathbf{A} of the branching-time-consistency problem is blocked if one of the following is true.*

- *There is $y \in A \setminus S$ and $x \in S$ such that $y \leq^{\mathbf{A}} x$.*
- *There is $y \in A \setminus S$ and $x \in S$ such that $y \parallel^{\mathbf{A}} x$.*
- *There are $x, y \in S$ such that $x \neq^{\mathbf{A}} y$.*

Clearly, a set of variables that is blocked can not be free. Note that it can be checked efficiently whether a given set of variables is blocked.

Corollary 5.3.4 *An instance whose graph \mathbf{U} is connected and where all sets of variables are blocked is unsatisfiable.*

Proof. Suppose for contradiction there was a solution for the instance \mathbf{A} . By Lemma 5.3.2, there exists a free set of variables. But since all sets are blocked, and free sets can not be blocked, we obtain a contradiction. \square

To find unblocked sets of variables, let \mathbf{G} be the directed graph with vertex set A where $(u, v) \in E^{\mathbf{G}}$ if $x \leq^{\mathbf{A}} y$ or $x \parallel^{\mathbf{A}} y$ in I . A directed graph \mathbf{G} is *strongly connected* if any two vertices $x, y \in G$ are connected via a path $x = p_1, \dots, p_n = y$ such that $(p_i, p_{i+1}) \in E^{\mathbf{G}}$ for all $1 \leq i \leq n - 1$. A *strongly connected component (SCC)* of \mathbf{G} is a maximal strongly connected subgraph of \mathbf{G} .

Proposition 5.3.5 *All free sets S of variables of \mathbf{A} must be strongly connected components of \mathbf{G} , and they must be terminal, i.e., there is no $(u, v) \in E^{\mathbf{G}}$ where $u \in S$ and $v \in A \setminus S$.*

Proof. Assume otherwise. Then S would be blocked, a contradiction by Corollary 5.3.4. \square

We can now state the algorithm, see Figure 5.3.

```

PDC( $\mathbf{A}$ )
Input: a structure  $\mathbf{A}$  over  $\{\leq, \parallel, \neq\}$  where  $\leq, \parallel, \neq$  are binary relation symbols.

Compute the connected components  $\mathbf{U}_1, \dots, \mathbf{U}_k$  of the graph  $\mathbf{U}$  for  $\mathbf{A}$ ;
For  $i = 1$  to  $k$ 
  Compute the SCCs of the graph  $\mathbf{G}$  of  $\mathbf{A}[U_i]$ ;
  if there is an unblocked terminal SCC  $\mathbf{G}'$  of  $\mathbf{G}$  then call PDC( $\mathbf{A}[U_i \setminus G']$ )
  else reject
end for

```

Figure 5.3: The branching-time-consistency algorithm.

Proposition 5.3.6 *The PDC algorithm given in Figure 5.3 rejects an instance of the branching time consistency problem if and only if the given instance \mathbf{A} is unsatisfiable, in time that is quadratic in the size of the instance.*

Proof. We show the correctness of the algorithm by induction on the recursion of the algorithm. If the all terminal strongly connected components are blocked, the algorithm correctly rejects, because Proposition 5.3.5 implies that \mathbf{A} is unsatisfiable.

Otherwise, let $\mathbf{U}_1, \dots, \mathbf{U}_k$ be the connected components of the graph \mathbf{U} for \mathbf{A} . We first show that $\mathbf{A}[U_i]$ has a solution, for all $i \in \{1, \dots, k\}$. Let \mathbf{G} be the directed graph of $\mathbf{A}[U_i]$ that was described above. Since the algorithm does not reject, there is an unblocked terminal SCC \mathbf{G}' of \mathbf{G} . If PDC($\mathbf{A}[U_i \setminus G']$) rejects, then by inductive assumption there is an unsatisfiable substructure of \mathbf{A} , and hence \mathbf{A} is unsatisfiable as well. Otherwise, by inductive assumption there exists a solution (\mathbf{T}', h') for $\mathbf{A}[U_i \setminus G']$. Create a new vertex r , link the root of \mathbf{T}' below, and let \mathbf{T}_i be the resulting tree with root r . Let h_i be the extension of h' that maps all vertices in G' to r .

We claim that (\mathbf{T}_i, h_i) is a solution for $\mathbf{A}[U_i]$. If $x \leq^{\mathbf{A}} y$ and x, y are both in \mathbf{G}' , then $h_i(x) = h_i(y)$, and if x, y are both in $U_i \setminus G'$ then $h_i(y)$ lies below $h_i(x)$ by inductive assumption. Since $y \in \mathbf{G}'$ implies that $x \in \mathbf{G}'$, the only remaining case is that $x \in \mathbf{G}'$ and $y \in U_i \setminus G'$. In this case $h_i(y)$ lies below $h_i(x)$ by construction of h_i . If $x \neq^{\mathbf{A}} y$, then x and y can not both be in \mathbf{G}' (otherwise \mathbf{G} would be blocked). If they are both in $U_i \setminus G'$ then $h_i(x) \neq h_i(y)$ by inductive assumption. In all other cases, $h_i(x) \neq h_i(y)$ by construction of h_i . Finally, if $x \parallel^{\mathbf{A}} y$, then by definition of \mathbf{G} and strongly connectedness, either $x, y \in G'$ or $x, y \in U_i \setminus G'$. In the first case, we are again done by inductive assumption, in the second case neither $h_i(x)$ lies above $h_i(y)$ nor vice versa by construction of h_i . This concludes the proof that (\mathbf{T}_i, h_i) is a solution for $\mathbf{A}[U_i]$.

To show that \mathbf{A} is satisfiable, create a new vertex s , and link the roots of all trees $\mathbf{T}_1, \dots, \mathbf{T}_k$ below s . Let \mathbf{T} be the resulting tree, and let h be the common extension of

the mappings h_1, \dots, h_k with domain A . Note that all h_i have disjoint domains, and hence this is a well-defined mapping. We claim that (\mathbf{T}, h) is a solution for \mathbf{A} . By the definition of the graph \mathbf{U} and connectedness, for all x, y such that $x \leq^{\mathbf{A}} y$ both x and y must be in the same component \mathbf{U}_i , and hence $h_i(y)$ lies below $h_i(x)$ in \mathbf{T}_i and therefore also in \mathbf{T} . If $x \neq^{\mathbf{A}} y$ or $x \parallel^{\mathbf{A}} y$, then either x and y are in the same component \mathbf{U}_i , and \neq and \parallel are preserved by inductive assumption, or \neq and \parallel is preserved by construction of h .

The running time is bounded by Cnm where n is the number of variables, m is the size of the input instance \mathbf{A} , and C is a constant. \square

There is a series of problems in computational problems that can be solved by extensions or variations of the algorithms presented in this Section. Cornell's original logic is an extension of branching time constraints where we have additional constraints that can talk about left and right for the vertices of a (plane) tree. Cornell's logic can be solved by an extension of the PDC algorithm presented here [8].

Yet another important field of efficient graph algorithms are dominance constraints used in semantic underspecification formalisms in computational semantics [2, 19, 3, 5]; again, the notion of free variables is crucial.

Exercises for Section 5.3.

Exercise 5.3.7 *Verify the claim made in this section that the branching time consistency problem can be described as CSP(\mathbf{B}) for the relational structure \mathbf{B} described above.*

Exercise 5.3.8 *Show that there is a polynomial-time reduction from the rooted triple consistency problem to the branching-time-consistency problem.*

Exercise 5.3.9 *Let C, D, E, F be binary relation symbols. Let \mathbf{L} and \mathbf{L}' be the following structures. Both have the same domain as the structure \mathbf{B} described in this section. The signature of \mathbf{L} is $\{C, D\}$, and $(a, b) \in A^{\mathbf{L}}$ if $a < b$ as described in this section, and $(a, b) \in B^{\mathbf{L}}$ if neither $a < b$, $a > b$, nor $a = b$. The signature of \mathbf{L}' is $\{E, F\}$, and $(x, y) \in E^{\mathbf{L}'}$ iff not $x \parallel^{\mathbf{B}} y$, and $(x, y) \in F^{\mathbf{L}'}$ iff not $x \leq^{\mathbf{B}} y$.*

Show that every relation in \mathbf{L} has a primitive positive definition in \mathbf{L}' and vice versa.

Bibliography

- [1] A. Aho, Y. Sagiv, T. Szymanski, and J. Ullman. Inferring a tree from lowest common ancestors with an application to the optimization of relational expressions. *SIAM Journal on Computing*, 10(3):405–421, 1981.
- [2] E. Althaus, D. Duchier, A. Koller, K. Mehlhorn, J. Niehren, and S. Thiel. 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, 2001.
- [3] E. Althaus, D. Duchier, A. Koller, K. Mehlhorn, J. Niehren, and S. Thiel. An efficient graph algorithm for dominance constraints. *Journal of Algorithms*, pages 194–219, 2003.
- [4] M. Bodirsky and H. Chen. Qualitative temporal and spatial reasoning revisited. In *16th EACSL Annual Conference on Computer Science and Logic (CSL07)*, 2007.
- [5] M. Bodirsky, D. Duchier, J. Niehren, and S. Miele. A new algorithm for normal dominance constraints. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA'04)*, pages 59–67, New Orleans, January 2004.
- [6] M. Bodirsky and J. Kára. A fast algorithm and lower bound for temporal reasoning. Preprint, 2007.
- [7] M. Bodirsky and M. Kutz. Pure dominance constraints. In *Proceedings of STACS'02*, pages 287–298, 2002.
- [8] M. Bodirsky and M. Kutz. Determining the consistency of partial tree descriptions. *Artificial Intelligence*, 171:185–196, 2007.
- [9] M. Broxvall and P. Jonsson. Point algebras for temporal reasoning: Algorithms and complexity. *Artif. Intell.*, 149(2):179–220, 2003.
- [10] P. J. Cameron. *Oligomorphic Permutation Groups*. Cambridge Univ. Press, 1990.
- [11] T. Cornell. On determining the consistency of partial descriptions of trees. In *Proceedings of the ACL*, pages 163–170, 1994.
- [12] V. Dalmau and J. Pearson. Closure functions and width 1 problems. *CP'99*, pages 159–173, 1999.
- [13] M. Egg, A. Koller, and J. Niehren. The Constraint Language for Lambda Structures. *Journal of Logic, Language, and Information*, 10:457–485, 2001.

- [14] T. Feder. Classification of homomorphisms to oriented cycles and of k -partite satisfiability. *SIAM J. Discrete Math.*, 14(4):471–480, 2001.
- [15] T. Feder and M. Vardi. The computational structure of monotone monadic SNP and constraint satisfaction: A study through Datalog and group theory. *SIAM Journal on Computing*, 28:57–104, 1999.
- [16] P. Hell and J. Nešetřil. *Graphs and Homomorphisms*. Oxford University Press, 2004.
- [17] R. Hirsch. Expressive power and complexity in algebraic logic. *Journal of Logic and Computation*, 7(3):309 – 351, 1997.
- [18] P. Jeavons, D. Cohen, and M. Cooper. Constraints, consistency and closure. *AI*, 101(1-2):251–265, 1998.
- [19] A. Koller, K. Mehlhorn, and J. Niehren. A polynomial-time fragment of dominance constraints. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL'00)*, Hong Kong, Oct. 2000.
- [20] A. Koller, J. Niehren, and R. Treinen. Dominance constraints: Algorithms and complexity. In *Logical Aspects of Computational Linguistics (LACL'01)*, volume 2014 of *LNAI*, pages 106–125, 2001.
- [21] J. Niehren and S. Thater. Bridging the gap between underspecification formalisms: Minimal recursion semantics as dominance constraints. In *41st Meeting of the Association for Computational Linguistics*, pages 367–374, July 2003.