Fundamenta Informaticae 28 (1996) 165–182
IOS Press

Satisfiability of Inequalities in a Poset

Vaughan Pratt*
Computer Science Department
Stanford University
Stanford, CA 94305
pratt@cs.stanford.edu

Jerzy Tiuryn†
Institute of Informatics
Warsaw University
Banacha 2, 02-097 Warsaw, POLAND
tiuryn@mimuw.edu.pl

Abstract. We consider tractable and intractable cases of the satisfiability problem for conjunctions of inequalities between variables and constants in a fixed finite poset. We show that crowns are intractable. We study members and closure properties of the class of tractable posets. We define a feasible poset to be one whose potential obstacles to satisfiability are representable by a certain formula of the first-order extended by the least fixed operator. For bipartite posets we give a complete classification of hard posets and feasible ones.

1. Introduction

We investigate the computational complexity of the P-satisfiability problem. This is the problem of deciding whether a given finite set of inequalities is simultaneously satisfiable in a given poset P, which we assume throughout to be finite. The permitted operands of the inequalities are constants and variables. The permitted constants are the elements of P. An assignment of elements of P to variables *satisfies* the set of inequalities when it makes all of them true. The set of inequalities is called *P-satisfiable* when there exists such a satisfying assignment.

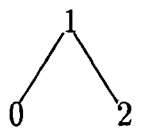

Figure 1.1. Poset P_1.

*The first author is supported by ONR under grant number N00014-92-J-1974.
†The second author is partly supported by NSF Grants CCR-9417382, CCR-9304144, and by Polish KBN Grant 2 P301 031 06.

For example take $P = P_1$ as per Figure 1.1, and consider the inequalities $x \leq 0$, $x \leq y$, $y \leq z$. Assigning either 1 or 2 to x would falsify $x \leq 0$, whence $x = 0$ in any satisfying assignment. Since $0 \leq 2$ does not hold in P, y and z must each be either 0 or 1. Of these four possibilities, $y \leq z$ rules out $y = 1, z = 0$, and the remaining three assignments are all satisfying assignments. Hence this set of inequalities is P-satisfiable, in three ways.

We write P-SAT for the set of those P-satisfiable sets of inequalities that have a "succinct certificate," namely a satisfying assignment. Such an assignment when straightforwardly presented can be checked in time linear in the size of the presentation in symbols, simply by evaluating each inequality to verify its truth for that assignment. It follows that for all posets P, P-SAT is in the class NP of problems solvable on a Turing machine in time nondeterministic in the size of the set of inequalities. When P-SAT is in the class PTIME of problems solvable in *deterministic* polynomial time, we say that P is *tractable*. We introduce a notion of feasible poset: P is *feasible* when there exists a predicate on sets of inequalities asserting in a certain language that the given set is P-satisfiable; this predicate can be evaluated for each given set in time polynomial in the length of presentation of the set.

In section 2 we shall show that there exist posets P for which P-SAT is an NP-complete problem, namely crowns. Section 3 introduces and treats aspects of tractable and feasible posets. Section 4 gives several transformations of posets and shows constructively that they preserve feasibility, by showing how to transform the predicate associated to that poset. In section 5 we give a complete classification of bipartite posets with respect to tractability of the satisfiability problem.

1.1. Background

Our interest in the poset satisfiability problem is motivated by the area of type reconstruction problems[1] for the case of simply typed lambda calculus with subtyping. The reader is referred to J. Mitchell's paper [Mitchell84] for introduction to that area as well as the basic reduction of the original problem of type reconstruction to the problem of poset satisfiability. O'Keefe and Wand [OKeefeWand89] treat a similar reduction.

The NP-hardness result of section 2 of this paper has been used by Mitchell and Lincoln [MitchellLincoln92] to show that the type reconstruction problem for simply typed λ-calculus with subtyping is NP-hard for certain posets of atomic subtypes.

The second author has shown [Tiuryn92] that solving inequalities in simple types (more general than the atomic types treatable with the results of the present paper) is PSPACE-hard, and that when the poset of atomic subtypings is a disjoint union of lattices then the type reconstruction problem is in PTIME.

With M. Wand [TiurynWand93], the second author has investigated the problem of type reconstruction for simple types with subtyping and recursive types. This naturally leads to a generalization of the problem of solving inequalities: instead of solving finite systems of inequalities in a finite poset one can consider solving an infinite system presented as a regular expression (or finite-state automaton). This problem is PSPACE hard for nontrivial posets and in PTIME otherwise.

M. Benke generalizes [Tiuryn92] in two ways. He shows [Benke93] that the PTIME property holds for a larger class of posets, namely those satisfying the Helly property, which include trees.[2] And in [Benke95] he generalizes the PSPACE-hardness construction of [Tiuryn92] via a suitable generalization of the conditions under which the NP-hardness result for the "flat" system treated in this paper can be transferred to the general subtype inequality system (over the same poset) yielding a PSPACE lower bound on complexity.

[1] The problem of type reconstruction for a type system $\mathcal{T}$ is: given a term M of pure (untyped) lambda calculus, decide whether M can be decorated with types so that is becomes correctly typable in $\mathcal{T}$. This problem has practical motivations coming from typed functional programming languages, such as ML.

[2] These posets naturally arise in connection with class inheritance.

More recently, Hoang and Mitchell have shown [HoangMitchell95] that the general algebraic problem of solving subtype inequalities (see [Tiuryn92]) is PTIME equivalent to the type reconstruction problem for simple types with subtyping.

A natural generalization of the satisfiability problem for posets is satisfiability in an arbitrary relational structure, which has been studied by Feder and Vardi [FederVardi93].

1.2. Satisfiability and Retractability

The P-satisfiability problem is of a logical nature. However it has a straightforward translation into an equivalent algebraic problem, that of P-retractability.

A function f is *idempotent* when $f \circ f = f$, equivalently when its image or range coincides with the set of its fixpoints. A *retraction* of a poset Q is an idempotent monotone function $f : Q \to Q$; we say that f *retracts* Q onto its image $f(Q)$, and call $f(Q)$ a *retract* of Q. For a given poset P, the *P-retractability problem* is that of deciding whether P is a retract of a given extension Q of P.

The example above of P-satisfiability has an evident reformulation as a P-retractability problem. We extend P_1 to Q by adjoining to P the variables x, y, z treated as new points, ordered as in the inequalities, as shown in Figure 1.2.

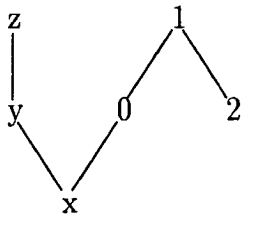

Figure 1.2. Poset Q.

Theorem 1.1. *P-SAT is polynomial-time equivalent to the P-retractibility problem.*

Proof:
To reduce the P-retractibility problem to P-SAT, translate the given extension Q of P to a set of inequalities by taking the set of variables to be $Q - P$ and taking the set of inequalities to be the graph of Q, i.e. all $q \leq q'$ holding in Q. Then Q retracts to P if and only if the set of inequalities is simultaneously satisfiable in P.

To reduce P-SAT to P-retractibility, translate the given set of inequalities to an extension Q of P whose non-P elements are the variables appearing in the inequalities, ordered according to the reflexive transitive closure of the given inequalities. Q is a preordered set: reflexive and transitive but not necessarily antisymmetric. Identify all equivalent elements, those pairs x, y such that $x \leq y \leq x$. (This extension might not be *conservative*, in the sense that for some $p \neq q \in P$, $p \leq q$ might hold in Q but not in P, in which case Q cannot retract to P.) The given inequalities are then satisfiable in P if and only if Q retracts to P. □

Viewing a poset P, defined as an irreflexive transitive relation, as an acyclic *directed* graph, we associate to P the undirected graph formed by "erasing arrowheads." (A common source of confusion here is that the Hasse diagram for $x \leq y \leq z$ conventionally omits the third edge $x \leq z$, obscuring the fact that the associated undirected graph forms a triangle.) More formally, we define the *graph of P* (understood henceforth to be undirected) to be the symmetric closure of P, whose cycles of length two constitute the undirected edges.

We define the *distance* between two points in a poset as that in its graph, namely the length in edges of the shortest path, or infinity if the points are not in the same connected component. This notion of distance can be seen to make a connected poset a metric space. A monotone function (including retractions) between connected posets is therefore a *contraction* or length-nonincreasing function (but not conversely since the contractions of a poset

include the antimonotone functions $x \leq y \supset f(x) \geq f(y)$). The *diameter* of a poset is the greatest distance between any two of its points, whence a chain of $n \geq 2$ elements has unit diameter.

A *zigzag* of length n is a poset whose graph is a simple path of length n, equivalently $n+1$ elements in alternating order, either as $x_0 \leq x_1 \geq x_2 \leq \ldots x_n$ or $x_0 \geq x_1 \leq x_2 \geq \ldots x_n$, these being the two minimal posets of diameter n connecting x_0 to x_n. We refer to the former zigzag as an *upper zigzag* from x_0 to x_n and to the latter zigzag as a *lower zigzag* from x_0 to x_n. Call a zigzag *proper* if all its elements are pairwise different. We shall make frequent use of zigzags as bits of string tying elements together; these create nonlinear constraints on retractions when the strings go taut, the basis for our NP-complete problems.

2. Intractable Cases

An *n-crown* C_n is a poset with $2n$ elements $0, 1, 2, \ldots, 2n-1$ partially ordered such that the only comparisons are $2i \leq 2i \pm 1$ (using addition modulo $2n$ so that $2n - 1 \leq 0$ is in the order). The 4-crown C_4 can be depicted as in Figure 2.1(a), but is less cluttered if we permit duplication of elements as in Figure 2.1(b).

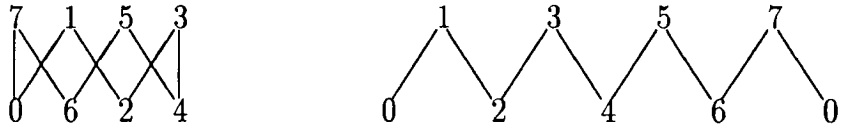

Figure 2.1. The 4-crown C_4 (a) usual view (b) unrolled.

Theorem 2.1. *C_n-SAT is NP-complete for $n \geq 2$.*

Proof:
We reduce 3SAT[3] to C_n-SAT. We give separate though similar proofs for $n = 2$ and $n \geq 3$. The common geometric intuition underlying the respective constructions is clearer for the latter and so we give it first.

Assume $n \geq 3$. For each CNF formula φ with v variables (hence at most $2v$ literals) and k clauses we construct a formula ψ with $(9n-3)v + nk$ variables and $(17n-2)v + (n+2)k$ inequalities which is satisfiable in C_n if and only if φ is satisfiable in $\{0, 1\}$.

The basic idea will be to simulate each literal of φ with a copy of C_n intended to retract bijectively to C_n itself. We then give constructions that (i) permit only two such retractions of each copy, which we associate with the two possible truth values of the corresponding literal; (ii) interpret $\neg$ standardly by forcing each literal x to retract oppositely to its negation $\neg x$; and (iii) enforce the constraints implied by the k clauses of φ.

Associate to each of the $2v$ literals x (which if negative will be of the form $\bar{y}$) a copy of C_n. Thinking of C_n as a circular crown, rotate the copy $1/2n$ of a full circle; Figure 2.2 illustrates the case $n = 3$. Each element of the copy that is midway between p and $p+2$ (mod $2n$) of the original C_n is named $x_{p,p+2}$. At this stage we have $2n$ variables per literal so $4nv$ variables altogether.

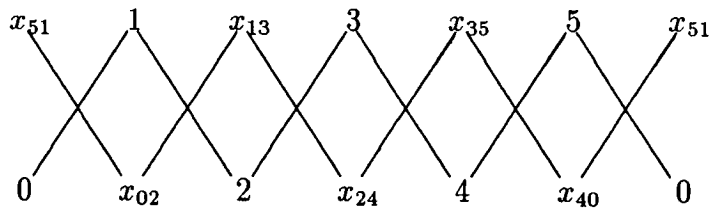

Figure 2.2. C_3 and copy.

[3]3SAT is the set of satisfiable conjunctive normal form formulas having 3 literals per clause.

We now impose constraints (i)-(iii).

(i) At the $2n$ places where the edges of C_n intersect the edges of the copy, place $2n$ additional elements, as shown in Figure 2.3. The element on the edge connecting p to $p+1$ (mod $2n$) is named $x_{p,p+1}$. This doubles the number of variables, to $8nv$. We can now count the inequalities; thus far we have $16nv$.

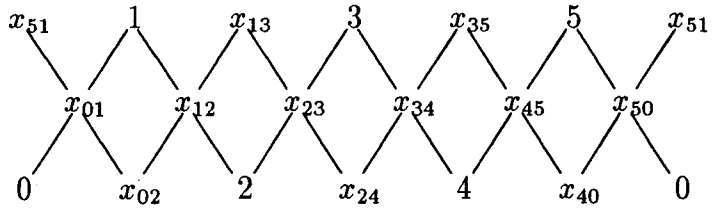

Figure 2.3. Connecting C_n to its copy.

Now consider the possible retracts of the poset of Figure 2.3 to C_n. If x_{50} is sent to 0 then so is x_{40}, whence x_{45} is sent to 5, whence so is x_{35}, and so on. Conversely if x_{50} is sent to 5 then so is x_{51}, whence x_{01} is sent to 0, whence so is x_{02}, and so on. It follows that this poset has just two retracts to P, one sending every x_{pq} to p, the other sending every x_{pq} to q. These will correspond, as poset valuations of x_{pq} in P, to truth valuations of x in $\{0,1\}$, respectively *false* and *true*.

(ii) For each of the v variables x of φ, we tie the literal pair x and $\bar{x}$ together in such a way that when x_{pq} retracts to p, $\bar{x}_{pq}$ can retract only to q. That is, the x and $\bar{x}$ crowns can only rotate in opposite directions when retracted, corresponding to always having opposite truth values. This is accomplished by a mechanism that can be visualized as a piece of string connecting a point of the x crown to a diametrically opposite point of the $\bar{x}$ crown (in the unretracted position). We use the fact that if these two crowns retract in the same direction (which we want to avoid), these diametrically opposite points remain diametrically opposite. The trick is to prevent this possibility by making the string one "notch" shorter than a half-perimeter of the crown.

The "string" is realized as a zigzag of length $n-2$ from $x_{2n-1,1}$ to $\bar{x}_{n-1,n+1}$, as shown in Figure 2.4, for $n=5$ (where $x_{2n-1,1} = x_{91}$ and $x_{n-1,n+1} = x_{46}$) and again for $n=6$ ($x_{2n-1,1} = x_{11,1}$ and $x_{n-1,n+1} = x_{57}$) to illustrate the treatment of odd and even n respectively. The $n-3$ variables $a, b, c, \ldots$ per variable of φ are new variables not used elsewhere, bringing the variable count to $(9n-3)v$. The $n-2$ inequalities bring the inequality count to $(17n-2)v$.

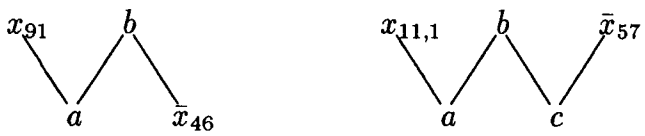

Figure 2.4. Inequalities making $\neg x$ the complement of x.

Any retraction to C_n must conform the string to one side or the other of C_n. Now the distance around C_n from $2n-1$ to $n-1$ is n (whichever way we go around), while the distance from $2n-1$ to $n+1$ is $n-2$ when measured around the $n+2, n+3, \ldots$ side (and $n+2$ around the other side). Hence if a retract sends $x_{2n-1,1}$ to $2n-1$ it must send $\bar{x}_{n-1,n+1}$ to $n+1$ in order that the "string" be able to retract to C_n. Likewise if $x_{2n-1,1}$ retracts to 1 then $\bar{x}_{n-1,n+1}$ must retract to $n-1$. This achieves this desired contrary motion of x and $\bar{x}$.

(iii) We implement each clause $x \vee y \vee z$ of φ by a more elaborate version of the above diameter trick. We first run a string of length 4 from an arbitrary point on x to a suitable point on $\bar{y}$ in such a way that the string becomes taut (in the sense of having a unique retraction) just when both x and y are false. We then run a second string, of length $n-2$, from the midpoint of the first string to a point on z, such that when the first (short) string

becomes taut the second (long) string forces z to the true position. When there is slack in the first string, this gives the second string just enough extra slack that it does not constrain z.

The length 4 string runs from $x_{2n-1,0}$ to $\bar{y}_{01}$, while the length $n-2$ string runs from the midpoint w of the first string to $z_{n-2,n}$. This is illustrated for $n=5$ in Figure 2.5.

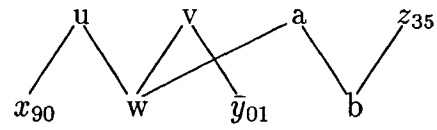

Figure 2.5. Poset for clause $x \vee y \vee z$, case $n=5$.

Now x false forces $u=n-1$, while y false forces $v=1$. Hence both false forces $w=0$, which in turn forces $z_{n-2,n}=n-2$, i.e. z true. If however x is true, i.e. $x_{2n-1,0}=0$, then $u=1$, $w=2$ becomes possible regardless of the truth of y. Then $z_{n,n-2}$ can be either n (z false, making the long string taut) or $n-2$ (z true, and the first zigzag of the long string can "idle" by setting $b=w$). When x is false but y is true, i.e. $\bar{y}_{01}=0$, then $v=n-1$, $w=n-2$, $u=n-1$ becomes possible. In this case the long string can be run around the $n-2, n-3, \ldots$ side, allowing $z_{n-2,n}$ to be either $n-2$ (making the long string taut) or n (slack).

Hence no retract makes all of x, y, z false, and this is the only constraint on the truth values of variables imposed by this construct. The construct adds n variables and $n+2$ inequalities per clause, making the final totals $(9n-3)v+nk$ variables and $(17n-2)v+(n+2)k$ inequalities.

(Side remark: For $n \geq 5$ the length 4 string could have been attached to crown tips; attaching it to crown intersections in effect further shortens the string to length 2 when x and y are both false, preventing it from going around the other side of the crown when n is 3 or 4. This yields a uniform construction for all $n \geq 3$.)

This completes the proof for the case $n \geq 3$. We now treat the case $n=2$. We proceed as for $n \geq 3$, but omit the explicit representation of $\bar{x}$.

The formula φ having v variables and k clauses translates to a formula ψ having $8v+3k$ variables and $16v+6k$ inequalities. We retain construction (i), which gives $8v$ variables and $16v$ inequalities. Without the $\bar{x}$ construction, (ii) is no longer relevant. In place of construction (iii), when all three literals of $x \vee y \vee z$ are positive we use the construction of Figure 2.6 for each of the k clauses, giving the remaining $3k$ variables (3 new variables per clause) and $6k$ inequalities (6 edges per clause).

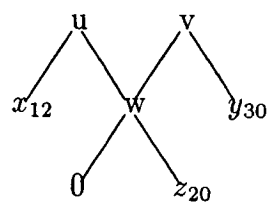

Figure 2.6. Poset for clause $x \vee y \vee z$ when $n=2$.

This works as follows. When x and y are both false, $u=1$ and $v=3$, forcing $w=0$ (since $0 \leq w$) and hence $z_{20}=0$, i.e. z is forced to true. But if x is true, then $u=v=w=3$ becomes possible, which removes all constraints on y and z. Similarly y true permits $u=v=w=1$, removing all constraints on x and z.

Clauses with negative literals are accommodated by using x_{01}, y_{23}, and z_{02}, in both the figure and the proof, when the corresponding variables are negated. For example when the clause is $\neg x \vee y \vee \neg z$, we replace x_{12} by x_{01} and z_{20} by z_{02} throughout. The proof then goes through when we interchange true and false for x and z (equivalently, if we replace x and z by $\neg x$ and $\neg z$).

This completes the proof. □

3. Tractable Cases

This section introduces the notion of obstacle for a poset as a formula characterizing its non-retractible extensions. Subsection 3.1. motivates the notion of obstacle, 3.2. defines it and establishes its reliability as an indicator of non-retractibility.

3.1. Some Motivating Examples

By way of motivation, we start with the simplest case, that of a lattice. Let us first notice the following easy observation which implies that every lattice is a tractable poset.

Proposition 3.1. If Q extends a lattice P, then Q retracts to P.

Proof:
Retract every point $q \in Q$ to meet of all points in P which are above q. □

Next, consider the poset P_1 of Figure 1.1. Again it is easy to check for P–retractability.

Proposition 3.2. Let Q extend P_1. Q retracts to P_1 iff $\{0,2\}$ has no lower bound in Q.

Proof:
The necessity is obvious. For sufficiency, retract every point $q \in Q$ to meet of all points in P_1 which are above q. This leaves unretracted points whose set of upper bounds in P_1 is P_1. By the assumption there are no such points in Q. □

For the next example take the following four element zigzag P_2 presented in Fig.3.2.

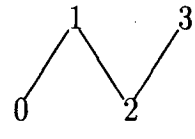

Figure 3.2. Poset P_2.

Again it follows from the next result that P_2 is tractable.

Proposition 3.3. Let Q extend P_2. Q retracts on P_2 iff there are no elements $x, y \in Q$ which satisfy the following constraints $0 \geq x \leq y \geq 3$.

Proof:
Such x and y clearly prevent any retraction, so it suffices to show that their absence permits a retraction. Their absence immediately implies the retractibility to 3 of the elements above 3, and to 0 of those below 0, which we therefore perform. No element strictly below 3 can be above either 0 or 1, and therefore all such may retract to 2. No element not yet retracted to P_2 can be below either 0 or 2, or above 3, whence all such may retract to 1, yielding the desired retraction to P_2. □

The following example illustrates a different form of condition. Let $P_3 = \{0,1\}$ be the discrete two-element poset, for which the distance from 0 to 1 is infinite.

Proposition 3.4. Let Q extend P_3. Q retracts to P_3 iff there is no path in Q which connects 0 and 1.

Proof:
If 0 is not connected to 1 then it is possible to retract to 0 the elements connected to 0, and everything else to 1, Conversely, when 0 is connected to 1 in Q their distance is finite, which retraction cannot increase, but this contradicts the infinite distance from 0 to 1 in P_3. □

In section 4 we will present machinery for producing some tractable posets, P_1 through P_3 being special cases.

As the last example, take the poset P_4 of Fig.3.3

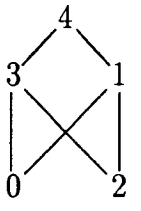

Figure 3.3. Poset P_4.

Proposition 3.5. Let Q extend P_4. Q retracts to P_4 iff there is no path in Q which connects 0 and 2 and whose all points are bounded above by 1 and 3.

Proof:
Necessity is easily shown by induction on the length of such a path. For sufficiency, retract each point $q \in Q$ to the meet in P_4 of the set of elements of P_4 above q in Q, when that meet exists. This leaves unretracted only those points whose set of upper bounds is either P or $\{1,3,4\}$. The path condition rules out existence of a point of the former kind. Any point q of the latter kind can lie only on a path of points of the latter kind reaching at most one of 0 or 2: retract q to that one, or to 0 by default if q is connected to neither 0 or 2. □

The common feature of the above examples was that we were able to single out some finite set of *obstacles* which prevented Q from being retracted onto P and proved that these were the only possible obstacles for the existence of a retraction on P. Each such obstacle was decidable in time polynomial in the size of the given extension Q, giving a polynomial-time algorithm for P-retractability. In this section we will generalize this method by introducing a language for expressing obstacles. This will be a fragment of the first-order language with the least fixed-point operator. We prove that the obstacles expressible in this fragment are always guaranteed to be sound (i.e. closed under retractions) and that they are decidable in polynomial time for any poset. We call a poset P *feasible* if there is a finite set of obstacles expressible in the above-mentioned language such that any extension Q of P retracts to P iff none of the obstacles holds in Q. In section 4 we study some order-theoretic constructions under which feasible posets are closed.

3.2. Feasible Posets

We assume that we have a countable set of individual variables $x, y, z, \ldots$ and a countable set of predicate variables $X, Y, Z, \ldots$ for each arity $n > 0$.

Let P be a fixed poset. Our language has one constant symbol c_p for each element $p \in P$. A *P-term* is either an individual variable or a constant symbol c_p.

The set of *P-formulas* is the least set of formulas which satisfies the following conditions.

- Every *atomic* formula, $t_1 \leq t_2$, $t_1 = t_2$, or $X(t_1, \ldots, t_n)$ is a P-formula, where $t_1, \ldots, t_n$ are P-terms, and X is n-ary predicate variable.
- If φ and ψ are P-formulas, then so are $(\varphi \vee \psi)$, $(\varphi \wedge \psi)$, and $(\exists x.\varphi)$.
- If φ is a P-formula, X is n-ary predicate variable, $\vec{x} = x_1, \ldots, x_n$ is a vector of n individual variables, and $\vec{t} = t_1, \ldots, t_n$ is a vector of n P-terms, then $(\mu X, \vec{x}.\ \varphi)(\vec{t})$ is also a P-formula.

We interpret P-formulas as follows. A *P-model* Q is any poset which contains P as a subposet. A *valuation* v in Q assigns to each individual variable x an element $v(x) \in Q$, and to each n-ary predicate variable X an n-ary predicate $v(X) \subseteq Q^n$. All valuations assign to the constant c_p the element p. We write v_x^d for the valuation differing from v only at the

individual variable x, where it satisfies $v_x^d(x) = d$ and otherwise satisfies $v_x^d(y) = v(y)$ and $v_x^d(X) = v(X)$.

A similar notation applies to predicate variables X and predicates S in Q as well, namely v_X^S. This notation is naturally extended to vectors of elements and individual variables, namely $v_{\vec{x}}^{\vec{d}}$.

The semantics of P-formulas is defined recursively with the help of the predicate $Q \models \varphi[v]$, where Q is a P-model, φ is a P-formula and v is a valuation in Q. These data suffice to determine a truth value for $Q \models \varphi[v]$, defined by induction on the height of φ. Q and v determine in the evident way the truth value of atomic P-formulas as well as that of P-formulas of the form $(\varphi \vee \psi)$, $(\varphi \wedge \psi)$, and $(\exists x.\varphi)$, leaving only one more form of P-formula to be explained.

The P-formula $(\mu X, \vec{x}.\ \varphi)(\vec{t})$ asserts $X(\vec{t})$ where the n-ary relation X is defined recursively by $X(\vec{x}) \equiv \varphi$ where φ may refer recursively to X and any or all of the variables in $\vec{x}$. As an example of its use consider the paths described in Proposition 3.5. We state formally that such a path exists by asserting $X(0,2)$ where the binary relation X is defined recursively by

$$x \leq y \vee y \leq x \rightarrow X(x,y)$$
$$\exists z[X(x,z) \wedge X(z,y) \wedge z \leq 1 \wedge z \leq 3] \rightarrow X(x,y)$$

By taking the recursive definition for φ we may combine it and the assertion in the one P-formula $(\mu X, x, y.(x \leq y \vee y \leq x \rightarrow X(x,y)) \wedge (\exists z[X(x,z) \wedge X(z,y) \wedge z \leq 1 \wedge z \leq 3] \rightarrow X(x,y)))(0,2)$. The latter form is less convenient for application, but more convenient for metatheory.

If m distinct variables other than those in $\vec{x}$ occur free in φ, and if the sum of the arities of the free predicate symbols other than X occurring free in φ is w, then $Q \models \varphi$ is a predicate on $Q^{m+n} \times 2^{Q^{w+n}}$, that is, a function $F_\varphi : Q^m \times 2^{Q^w}$. We reorganize this by moving the portion associated with X and $\vec{x}$ inwards to make it $F_\varphi : Q^m \times 2^{Q^w} \times 2^{Q^n} \times Q^n \rightarrow 2$ and then further "curry" it to yield $F'_\varphi : Q^m \times 2^{Q^w} \rightarrow (2^{Q^n} \rightarrow 2^{Q^n})$. Hence each possible valuation v of all variables and predicate symbols determines an element $q \in Q^m \times 2^{Q^w}$ and hence a function $F_{\varphi,v} : 2^{Q^n} \rightarrow 2^{Q^n}$, namely $F'_\varphi(q)$. We then define the n-ary relation $\mu X, \vec{x}.\ \varphi$ to be the least fixpoint of $F_{\varphi,v}$. The logical connectives of our language all being monotone, $F_{\varphi,v}$ must be monotone on the complete lattice 2^{Q^n}, whence by Tarski-Knaster this fixpoint exists. When $q = |Q|$ as in our application, the fixpoint is given explicitly by $F_{\varphi,v}^{(q^n)}(\emptyset)$ where $F^{(k)}$ denotes the k-fold composition of F with itself. This yields a simple algorithm for computing the fixpoint which can then be tuned for greater efficiency.

Now $\mu X, \vec{x}.\ \varphi$ takes two arguments, namely the valuation v and the n-tuple over Q at which the fixpoint is to be evaluated. Alternatively we may assume that the n-tuple consists of P-terms, since we may then apply v to map them to elements of Q. The latter convention leads naturally to the order of application $(\mu X, \vec{x}.\ \varphi)(\vec{t})[v]$. Omitting the valuation then leaves us with a P-formula $(\mu X, \vec{x}.\ \varphi)(\vec{t})[v]$ whose interpretation for any given valuation v is $(\mu X, \vec{x}.\ \varphi)(\vec{t})[v]$ as just defined.

A special situation of using the least fixed point operator is that of a *transitive closure* operator of a $2n$-ary relation. Let $n \geq 1$ and let φ be a P-formula, let $\vec{x}, \vec{y}$ be n-vectors of individual variables and let $\vec{t_1}, \vec{t_2}$ be n-vectors of P-terms. By $TC(\lambda\ \vec{x}, \vec{y}.\ \varphi)(\vec{t_1}, \vec{t_2})$ we will denote the P-formula[4]

$$(\mu X, \vec{x}, \vec{y}.\ \varphi \vee \exists \vec{z}.\ (X(\vec{x}, \vec{z}) \wedge X(\vec{z}, \vec{y})))(\vec{t_1}, \vec{t_2})$$

We introduce a subset of the set of all P-formulas. *Transitive closure P-formulas* form the least set of P-formulas which satisfies the following conditions.

[4]We assume that X does not occur free in φ.

- Every atomic formula $t_1 \leq t_2$, or $t_1 = t_2$ is a transitive closure P-formula.
- If φ and ψ are transitive closure P-formulas, then so are

$$(\varphi \vee \psi),\ (\varphi \wedge \psi),\ (\exists x.\varphi),\ \text{ and } TC(\lambda\, \vec{x}, \vec{y}.\ \varphi)(\vec{t_1}, \vec{t_2})$$

where $\vec{x}, \vec{y}$ are n-vectors of individual variables and $\vec{t_1}, \vec{t_2}$ are n-vectors of P-terms.

A *P-retraction* consists of a pair of P-models Q_1, Q_2 forming a chain of $P \subseteq Q_2 \subseteq Q_1$ of poset embeddings, together with a retraction of Q_1 onto Q_2.

Let $f : Q_1 \to Q_2$ be a function and let v be a valuation in Q_1. Define a valuation fv in Q_2, as follows.

$$fv(x) = f(v(x))$$

and

$$fv(X) = \{(f(d_1), \ldots, f(d_n)) \mid (d_1, \ldots, d_n) \in v(X)\}$$

Theorem 3.1. *Let $f : Q_1 \to Q_2$ be a P-retraction. For every P-sentence φ, we have the following equivalence: $Q_1 \models \varphi$ iff $Q_2 \models \varphi$.*

Proof:

Let $f : Q_1 \to Q_2$ be a P-retraction.

Let φ be a P-formula, v a valuation in Q_1, and u a valuation in Q_2. We prove by induction on φ the following two statements.

$$\text{if } Q_1 \models \varphi[v], \text{ then } Q_2 \models \varphi[fv] \tag{1}$$

$$\text{if } Q_2 \models \varphi[u], \text{ then } Q_1 \models \varphi[u] \tag{2}$$

The case of φ being an atomic formula, a disjunction, conjunction, or existential quantification is obvious and we omit the details.

Let us consider φ being of the form $(\mu X, \vec{x}.\ \psi)(\vec{t})$, and let

$$Q_1 \models (\mu X, \vec{x}.\ \psi)(\vec{t})[v] \tag{3}$$

Let $F_{\psi,v}$ be the functional in Q_1, associated with ψ and v (with respect to the choice of X and $\vec{x}$). And similarly, let $F_{\psi,fv}$ be the functional in Q_2, associated with ψ and fv. We prove by induction on $k \geq 0$ that for all $\vec{d} \in Q_1^n$,

$$\text{if } \vec{d} \in F^k_{\psi,v}(\emptyset), \text{ then } f(\vec{d}) \in F^k_{\psi,fv}(\emptyset) \tag{4}$$

For $k = 0$ (4) is obvious. Assume that (4) holds for k and let

$$\vec{d} \in F^{k+1}_{\psi,v}(\emptyset) = F_{\psi,v}(F^k_{\psi,v}(\emptyset))$$

Hence,

$$Q_1 \models \psi[v^{F^k_{\psi,v}(\emptyset),\vec{d}}_{X,\vec{x}}]$$

By induction assumption (1) for ψ we get

$$Q_2 \models \psi[f(v^{F^k_{\psi,v}(\emptyset),\vec{d}}_{X,\vec{x}})] \tag{5}$$

It follows that

$$f(v^{F^k_{\psi,v}(\emptyset),\vec{d}}_{X,\vec{x}}) = (fv)^{S,f(\vec{d})}_{X,\vec{x}} \tag{6}$$

where

$$S = \{f(\vec{e}) \mid \vec{e} \in F^k_{\psi,v}(\emptyset)\}$$

hence by induction assumption (4),

$$S \subseteq F^k_{\psi,fv}(\emptyset)$$

Since X occurs only positively in ψ, it follows from the above inclusion, (5) and (6) that

$$Q_2 \models \psi[(fv)^{F^k_{\psi,fv}(\emptyset),f(\vec{d})}_{X,\vec{x}})]$$

Hence, by the definition of $F_{\psi,fv}$, we conclude that

$$f(\vec{d}) \in F_{\psi,fv}(F^k_{\psi,fv}(\emptyset)) = F^{k+1}_{\psi,fv}(\emptyset)$$

This proves (4). From (4) and (3) we immediately get

$$Q_2 \models (\mu X, \vec{x}.\ \psi)(\vec{t})[fv]$$

This completes proof (1). Proof of (2) is just the same. □

Call a poset P *feasible* if there is a P-sentence φ such that for every P-model Q, Q retracts to P iff $Q \not\models \varphi$. We call such a φ a *complete obstacle* for P. A *P-obstacle* is any P-sentence φ such that $P \not\models \varphi$. It follows from Theorem 3.1., that if φ is a P-obstacle and $Q \models \varphi$, then Q does not retract to P. Call a poset P *TC-feasible* if it has a complete obstacle which is a transitive closure P-formula.

It follows from Proposition 3.1. that every lattice is TC-feasible. By Propositions 3.2., 3.3., 3.4., and 3.5. we conclude that posets P_1, P_2, P_3, and P_4 are TC-feasible. For example, a complete obstacle for P_4 is

$$TC(\lambda x, y.\ (x \leq y \vee y \leq x) \wedge x \leq 1 \wedge x \leq 3 \wedge y \leq 1 \wedge y \leq 3)(0,2)$$

Proposition 3.6. Every feasible poset is tractable. Moreover the retraction problem for every TC-feasible poset is in NLOGSPACE.

Proof:
It is well known that validity of a sentence of the first-order language with least fixed-point operator in a finite model can be checked in time polynomial in the size of the model. If the sentence is a transitive closure P-formula then validity can be checked in NLOGSPACE. □

For any P-model Q, let σ_Q be $\exists x_1 \ldots x_n.\ \sigma'_Q$, where $\{x_1, \ldots, x_n\} = Q - P$ and σ'_Q is the conjunction of all formulas of the form $t \leq t'$, where $t, t' \in Q$ and $t \leq t'$ holds in Q. The next result is a corollary of Theorem 3.1.

Proposition 3.7. For every P-model Q, Q retracts to P iff $P \models \sigma_Q$, i.e. iff σ_Q is not a P-obstacle.

Proof:
If Q retracts to P, then since σ_Q is a P-sentence which holds in Q, by Theorem 3.1. it must hold in P. Conversely, if $P \models \sigma_Q$, then the solution to σ'_Q defines a retraction of Q to P. □

It follows that if we were allowed to write in our language infinite disjunctions (which make sense and yield formulas which still are preserved under retractions, but are hardly polynomial-time computable), then the infinite disjunction

$$\bigvee_{Q \text{ does not retract to } P} \sigma_Q$$

would always be a complete obstacle for P. Feasible posets are those where this disjunction can be replaced by one P-sentence.

4. Constructions Preserving Feasibility

In this section we will study several constructs on posets which preserve feasibility: finite products, disjoint union, and retractions. We also show that feasible posets are closed under taking dual posets and under isomorphism, and that they properly contain posets satisfying Helly property.

4.1. Duality and Isomorphism

Proposition 4.1. The class of feasible posets is closed under isomorphisms. Also the class of TC-feasible posets is closed under isomorphisms.

Proof:
Let $f : P_1 \to P_2$ be an isomorphism of posets. For every P_1-formula φ let φ_f be a P_2-formula which is obtained from φ by replacing each constant c_p for $p \in P_1$ by a constant $c_{f(p)}$. Clearly if φ is a transitive closure P_1-formula, then φ_f is a transitive closure P_2-formula.

For a P_1 model Q let Q_f denote a P_2-model which is obtained from Q by replacing each element $p \in P_1$ by $f(p)$. Let $g_f : Q \to Q_f$ be the isomorphism resulting from the above construction. The proof now follows from the following two obvious observations.

For every poset Q which extends P_1,

$$Q \text{ retracts to } P_1 \text{ iff } Q_f \text{ retracts to } P_2$$

For every poset Q which extends P_1, for every valuation v in Q, and for every P_1-formula φ,

$$Q \models \varphi[v] \text{ iff } Q_f \models \varphi_f[g_f\, v]$$

It follows from the above two observations that φ is a complete obstacle for P_1 iff φ_f is a complete obstacle for P_2. □

Proposition 4.2. A poset dual to a feasible poset is feasible. A poset dual to a TC-feasible poset is TC-feasible.

Proof:
Let $P^\perp$ be the dual of a poset P. A complete obstacle $\varphi^\perp$ for $P^\perp$ is obtained from a complete obstacle φ for P by replacing every occurence of $\leq$ in φ by $\leq^{-1}$ (i.e. by $\geq$). The details are left for the reader. □

4.2. Finite Products

In this section we show that feasible (TC-feasible) posets are closed under finite products. Let P_1 and P_2 be two finite posets.

Proposition 4.3. Let Q be a poset which extends $P_1 \times P_2$. Let $a \in P_1$ and $b \in P_2$ be arbitrary elements. Then Q retracts on $P_1 \times P_2$ iff Q retracts on both: $P_1 \times \{b\}$ and $\{a\} \times P_2$.

Proof:
The "if" part follows from the observation that $P_1 \times P_2$ retracts on $P_1 \times \{b\}$ via retraction which sends (x, y) to (x, b). Similarly $P_1 \times P_2$ retracts on $\{a\} \times P_2$.

For the proof of "only if" take two retractions $f_1 : Q \to P_1 \times \{b\}$ and $f_2 : Q \to \{a\} \times P_2$ and define $f : Q \to P_1 \times P_2$ by $f(q) = (\pi_1 f_1(q), \pi_2 f_2(q))$, for $q \in Q$, where π_i denotes the projection on i-th component. It is easy to check that f is a retraction. □

Theorem 4.1. *If P_1 and P_2 are feasible, then so is $P_1 \times P_2$. Also TC-feasible posets are closed under finite products.*

Proof:
Take any $a \in P_1$ and $b \in P_2$. Let φ_1 be a complete obstacle for P_1. Since $P_1 \times \{b\}$ is isomorphic to P_1 it follows that $\hat{\varphi_1}$ is a complete obstacle for $P_1 \times \{b\}$, where $\hat{\varphi_1}$ is obtained from φ_1 by replacing each constatnt c_p by $c_{(p,b)}$, for $p \in P_1$ (see Proposition 4.1.). By a similar construction we obtain a complete obstacle $\hat{\varphi_2}$ for $\{a\} \times P_2$. By Proposition 4.3. it follows that $\hat{\varphi_1} \vee \hat{\varphi_2}$ is a complete obstacle for $P_1 \times P_2$. □

4.3. Helly Posets

In this section we show that a broad class of posets which satisfy *Helly property* is contained in TC-feasible posets. We start with some definitions.

A *disc* D in a poset P is specified by its *center* $p \in P$ and by a pair of non negative integers n_1, n_2. Given the above data, D is defined as the set of all points of P which are reachable from p via a lower zigzag of length at most n_1 and via an upper zigzag of legth at most n_2. Of course if the difference between n_1 and n_2 is larger than 1, then D can be equally specified by $n_1, n_1 + 1$, if $n_1 < n_2$, or by $n_2 + 1, n_2$, if $n_2 < n_1$.

A poset P satisfies *Helly propoerty* (see [Benke93, NevermannRival85]) if for every finite family $D_1, \ldots, D_k$ of discs, if $D_1 \cap \ldots D_k = \emptyset$, then for some $i, j \in \{1, \ldots k\}$, $D_i \cap D_j = \emptyset$.

Call an extension Q of P *isometric* if for every two points $p_1, p_2 \in P$ and for every n, if p_2 is accessible in Q from p_1 via a lower (resp. upper) zigzag of length n, then it is also accessible in P from p_1 via a lower (resp. upper) zigzag of length n.

The following result shows that Helly posets are feasible.

Theorem 4.2. (Benke93)
Let P be a poset satisfying Helly property. An extension Q of P retracts on P iff Q is isometric over P.

Corollary 4.1. Every poset satisfying the Helly property is TC-feasible.

Proof:
A complete obstacle for P is a disjunction of formulas which express the property that p_2 is accessible from p_1 via a lower (or an upper) zigzag of length at most $n - 1$, where n is the length of the shortest lower (or upper) zigzag in P from p_1 to p_2. □

Poset P_4 of Fig.3.3 is an example of a TC-feasible poset which doesn't satisfy Helly property. For $p \in \{0, 2\}$ let $D_p = \{x \in P_4 \mid p \leq x\}$ and for $q \in \{1, 3\}$ let $D_q = \{x \in P_4 \mid x \leq q\}$. The reader can easily check that D_0, D_1, D_2, D_3 are discs such that $D_0 \cap D_1 \cap D_2 \cap D_3 = \emptyset$, but every two (even every three) discs have non empty intersection.

We conclude this section with a result which will be used at the end of the paper. Let's call a poset P a *generalized zigzag* if for every two points $p_1, p_2 \in P$ there is at most one proper zigzag from p_1 to p_2. For example posets represented at Figures: 1.1, 2.4, 2.5, and 3.2 are all generalized zigzags, while poset P_4 of Fig.3.3 is not since there are two zigzags from 0 to 2: 0,3,2 and 0,1,2. Let us also observe that every generalized zigzag contains no chain of length greater than 1. Indeed, if $a < b < c$ then we have two different proper zigzags from a to b: a, b and a, c, b.

Proposition 4.4. If P is a generalized zigzag, then it satisfies Helly property.

Proof:
Take any family of discs $D_1, \ldots, D_n$ in P such that $D_1 \cap \ldots \cap D_n = \emptyset$, but $D_i \cap D_j \neq \emptyset$ for all $i, j \in \{1, \ldots, n\}$. Let n be the smallest with this property. Let $A = D_1 \cap \ldots \cap D_{n-1}$. It follows that $A \neq \emptyset$.

Take the family of all proper zigzags from the center d of D_n to an element in A. This family is non empty sine D_n has nonempty intersection with each D_i for $i < n$. Take a proper zigzag from this family of a minimal length. Let $a \in A$ be the right end of that zigzag. Since $A \cap D_n = \emptyset$ it follows that $d \notin A$. Let $b \neq a$ be the element adjacent to a on that zigzag. By the minimality condition it follows that $b \notin A$. Hence there is $k \in \{1, \ldots, n-1\}$ such that $b \notin D_k$.

We claim that $D_k \cap D_n = \emptyset$. Assume that $c \in D_k \cap D_n$. Take the unique zigzag connecting d with the center e of D_k. By the zigzag uniqueness condition it follows that a, b and c belong to that zigzag. Since $a \in D_k$, and $b \notin D_k$, again it follows by the zigzag uniqueness condition that c being an element of D_k must be positioned on that zigzag to the right of b. On the other hand since $a \notin D_n$ and since a is a right neighbor of b on that zigzag, it follows that no element to the right of b can belong to D_n. Obtained contradiction proves the claim that $D_k \cap D_n = \emptyset$. □

4.4. Disjoint Union

In this subsection we show the following result.

Theorem 4.3. *Feasible posets are closed under taking disjoint unions. Also TC-feasible posets are closed under disjoint unions.*

Proof:
Let P_1, P_2 be disjoint feasible posets with complete obstacles φ_1 and φ_2, respectively. Let $P = P_1 \cup P_2$. It is very easy to prove that a P-model Q retracts to P iff there is no path in Q which connects points in P_1 and P_2 and if for $i = 1, 2$, Q_i retracts to P_i, where Q_i is the set of all elements of Q which are connected by a path to an element in P_i. What we have to do is to make this observation into a P-formula.

Let $\sigma(z)$ be any P-formula such that z is its only free variable. For every P-formula φ we define its *relativization* φ^σ as follows.

- If φ is atomic, then φ^σ is φ.
- $(\varphi_1 \wedge \varphi_2)^\sigma$ is $(\varphi_1^\sigma \wedge \varphi_2^\sigma)$, and similar for $\vee$.
- $(\exists x.\ \varphi)^\sigma$ is $\exists x.\ (\sigma(x/z) \wedge \varphi^\sigma)$.
- $((\mu X, \vec{x}.\ \varphi)(\vec{t}))^\sigma$ is $(\mu X, \vec{x}.\ \varphi^\sigma)(\vec{t})$.

In case of transitive closure formulas the relativization of TC is slightly different.

- $(TC(\lambda\vec{x}, \vec{y}.\ \varphi))^\sigma(\vec{t_1}, \vec{t_2})$ is
  $TC(\lambda\vec{x}, \vec{y}.\ \varphi^\sigma \wedge \bigwedge_{i=1}^n \sigma(x_i/z) \wedge \bigwedge_{i=1}^n \sigma(y_i/z))(\vec{t_1}, \vec{t_2})$.

The above construction has the following property. Let Q be a P-model and let $\sigma(z)$ be a P-formula such that z is its only free variable. Let $Q_\sigma = \{a \in Q \mid Q \models \sigma[a]\}$ be a subposet of Q with its partial order being a restriction of that in Q. Let $P_\sigma = P \cap Q_\sigma$. Then for every valuation v in Q_σ and for every P_σ-formula φ the following equivalence holds:

$$Q \models \varphi^\sigma[v] \quad \text{iff} \quad Q_\sigma \models \varphi[v] \tag{7}$$

The proof of (7) is by routine induction on φ.

Now, to conclude the proof of our Theorem, let $Path$ be the following predicate.

$$TC(\lambda x, y.\ x \leq y \vee y \leq x)$$

Let σ be the disjunction

$$\bigvee_{a \in P_1,\ b \in P_2} Path(a, b)$$

and for $i = 1, 2$, let $\sigma_i(x)$ be the formula

$$\bigvee_{a \in P_i} Path(x, a)$$

It follows from the above that

$$\varphi_1^{\sigma_1} \vee \varphi_2^{\sigma_2} \vee \sigma$$

is a complete P-obstacle. □

4.5. Retractions

Let P_1 be an extension of a poset P and let Q be a P-model. We construct a P_1-model Q^+ as follows. Add to Q the new elements in $P_1 - P$, (without loss of generality we may assume that $P_1 - P$ and Q are disjoint). The order relation $\leq_+$ in Q^+ is defined as follows. $x \leq_+ y$ iff one of the following holds.

- $x \leq y$ holds in Q.
- There exists $z \in P$, such that $x \leq z$ holds in Q and $z \leq y$ holds in P_1.
- There exists $z \in P$, such that $x \leq z$ holds in P_1 and $z \leq y$ holds in Q.
- $x \leq y$ holds in P_1.

We have the following easy result.

Proposition 4.5.

(i) $\leq_+$ is a partial order in Q^+.
(ii) P^+ is order-isomorphic to P_1.
(iii) If Q retracts to P, then Q^+ retracts to P_1.
(iv) If P_1 retracts to P, then for every P-model Q, Q retracts to P, iff Q^+ retracts to P_1.

Proof:
(i) is proved by case analysis. (ii) is obvious. The proof of (iii) is also easy. Let $f : Q \to P$ be a retraction and let $g : Q^+ \to P_1$ be the extension of f which is identity on elements in $P_1 - P$. A simple case analysis shows that g is monotone and therefore a retraction on P_1.
(iv) obviously follows from (iii). □

Now, let us assume that P has at least two elements, say $0, 1 \in P$ and let k be the least integer such that $|P_1 - P| \leq 2^k$. Given any P-model Q we encode $a \in Q^+$ by $\hat{a} \in Q^{k+1}$ as follows. Every $a \in Q$ is encoded by $\hat{a} = (0, a, \ldots, a)$ (0 followed by k a's). Every $a \in P_1 - P$ is encoded by $\hat{a} = (1, \varepsilon_1, \ldots, \varepsilon_k)$, where $\varepsilon_1, \ldots, \varepsilon_k$ is a binary encoding of a.

We now give a P-formula D describing the *domain of encoding*, i.e., the set of all $k+1$-tuples of elements of Q which are codes of the elements of Q^+. We define $D(x, y_1, \ldots, y_k)$ to be

$$(x = 0 \wedge y_1 = \ldots = y_k) \vee \bigvee_{a \in P_1 - P} (x = 1 \wedge y_1 = (\hat{a})_1 \wedge \ldots \wedge y_k = (\hat{a})_k).$$

The P-formula LE below translates the definition of $a \leq_+ b$ as defined on Q^+ into the definition of inequality between the codes of a and b, respectively. That is, LE defines a binary relation between $k+1$-tuples of elements of Q each coding an element of Q^+.

Define $LE(x, \vec{y}, x', \vec{y'})$ to be

$$D(x, \vec{y}) \wedge D(x', \vec{y'}) \wedge \text{disjunction of clauses defining } \leq_+ .$$

For example, the first clause in the definition of $\leq_+$ takes the form

$$x = 0 \wedge x' = 0 \wedge y_1 \leq y_1',$$

while the second clause becomes

$$x = 0 \wedge x' = 1 \wedge \bigvee_{(a,b)\in A} (y_1 \leq a \wedge y_1' = (\hat{b})_1 \wedge \ldots \wedge y_k' = (\hat{b})_k)$$

where $A = \{(a,b) \mid a \in P,\ b \in P_1 - P,\ a \leq b\}$.

Next, for every P_1-formula φ with n free variables we construct a P-formula $\hat{\varphi}$ with $n(k+1)$ free variables. The role of $\hat{\varphi}$ when evaluated in Q is to mimic the evaluation of φ in Q^+. Since the variables of $\hat{\varphi}$ are confined to Q, we use $k+1$-tuples of elements of Q to code single elements of Q^+. The coding is accomplished with the help of the formulas D and LE defined earlier.

The construction is by induction on φ. To simplify the notation, for every individual variable x, let $\hat{x}$ denote the $(k+1)$-vector of variables $(x_0, x_1, \ldots, x_k)$. We assume that the variables $(x_0, x_1, \ldots, x_k)$ are all pairwise different and that the assignment of $\hat{x}$ to x is one-to-one. This gives us an obvious transformation which maps every P_1-term t into a $(k+1)$-vector $\hat{t}$ of P-terms, i.e. a P_1 constant a is transformed to $\hat{a}$ and a variable x is transformed to $\hat{x}$. Similarly, for every n-ary predicate variable X, let $\hat{X}$ denote a $n(k+1)$-ary predicate variable. The construction of $\hat{\varphi}$ follows.

- $t \leq t'$ is translated into $LE(\hat{t}, \hat{t'})$.
- $X(t_1, \ldots, t_n)$ is translated into $\hat{X}(\hat{t_1}, \ldots, \hat{t_n}) \wedge D(\hat{t_1}) \wedge \ldots \wedge D(\hat{t_n})$.
- $(\varphi \vee \psi)$ is translated into $\hat{\varphi} \vee \hat{\psi}$, and similarly $(\varphi \wedge \psi)$ is translated into $\hat{\varphi} \wedge \hat{\psi}$.
- $\exists x.\ \varphi$ is translated into $\exists \hat{x}.\ (D(\hat{x}) \wedge \hat{\varphi})$.
- Finally, $(\mu X, z_1 \ldots, z_n.\ \varphi)(t_1, \ldots, t_n)$ is translated into $(\mu \hat{X}, \hat{z_1} \ldots, \hat{z_n}.\ \hat{\varphi})(\hat{t_1}, \ldots, \hat{t_n})$.

For the case of transitive closure formulas we have the following defining condition.

- $TC(\lambda x_1, \ldots, x_n, y_1, \ldots, y_n.\ \varphi)(t_1, \ldots, t_n, s_1, \ldots, s_n)$ is translated into $TC(\lambda \hat{x_1}, \ldots, \hat{x_n}, \hat{y_1}, \ldots, \hat{y_n}.\ \hat{\varphi} \wedge \bigwedge_{i=1}^n D(\hat{x_i}) \wedge \bigwedge_{i=1}^n D(\hat{y_i}))(\hat{t_1}, \ldots, \hat{t_n}, \hat{s_1}, \ldots, \hat{s_n})$.

Theorem 4.4. *Feasible posets are closed under retractions. Also TC-feasible posets are closed under retractions.*

Proof:
One first proves by induction on φ the following property. For every P_1-formula φ, for every P-model Q and every valuation v in Q^+,

$$Q^+ \models \varphi[v] \quad \text{iff} \quad Q \models \hat{\varphi}[\hat{v}] \tag{8}$$

where $\hat{v}$ is any valuation which satisfies the following two conditions for every free variable x in φ and free predicate variable X in φ,

$$v(x) = a \quad \text{iff} \quad (\hat{v}(x_0), \ldots, \hat{v}(x_k) = \hat{a})$$

where $\hat{x} = x_0, \ldots, x_k$ is the $(k+1)$-tuple of variables associated with x, and

$$\text{if } (a_1, \ldots, a_n) \in v(X), \quad \text{then } (\hat{a_1}, \ldots, \hat{a_n}) \in v(\hat{X})$$

The proof of (8) is routine and we omit the details. Now, to complete the proof let's assume that P_1 retracts to P and let φ be a complete P_1-obstacle. It follows from (8) and Proposition 4.5. (iv) that $\hat{\varphi}$ is a complete P-obstacle. □

5. Bipartite Posets

Call a poset P *bipartite* if for all $p, q, r \in P$, if $p \leq q \leq r$, then either $p = q$ or $q = r$, i.e. these are the posets with all proper chains having at most two elements.

Let us observe that every n-crown C_n is a bipartite poset. Posets P_1, P_2 and P_3 introduced in Section 3.1. are also bipartite. In this section we give a complete classification of bipartite posets with respect to the retractability problem.

Theorem 5.1. *Let P be a bipartite poset.*
(i) If for some $n \geq 2$, P contains a crown C_n as a subposet, then P-SAT is NP-complete.
(ii) If P contains no crown, then it is TC-feasible, and therefore P-SAT is in NLOGSPACE.

Proof:
For the proof of (i) let us assume that P contains a crown C_n, and let n be the smallest number with this property. We proceed just the same as in the proof of NP-completeness for C_n-SAT (see Theorem 2.1.). The property that P is bipartite ensures that there are exactly two retractions of the "double crown" of variables. Since C_n is minimal, there are no shortcuts in P which would make the distance between two points of C_n smaller. Thus the "locking mechanism" for expressing negation will work in this encoding.

Also the encoding of clauses works here correctly. For $n = 2$ the encoding works (see Fig. 2.6) since the forbidden assignment: $x_{12} = 1$, $y_{30} = 3$ and $z_{20} = 2$ for this clause, yields a contradiction with the assumption that P is bipartite (retractability under this assignment implies that there is an element w which is below 1 and 3 and above 0 and 2). For $n > 2$ the encoding works as well, though for a slightly different reason. Take, for example, the poset of Fig. 2.5 which encodes a clause for case $n = 5$. Retractability under the assignment $x_{12} = 1$, $y_{90} = 9$ and $z_{53} = 5$ would imply that there exists an element w which is below 1 and 9. It cannot be 0 since the distance conditions prevent this possibility. Hence $\{0, w, 1, 9\}$ is a crown $C_2 \subseteq P$. This yields a contradiction with the assumption that P does not contain C_2.

In order to prove (ii) let us assume that P contains no crown. Call a set $A \subseteq P$ *decomposable* if there is a sequence $P_1 \subseteq P_2 \subseteq \ldots P_n$ such that $A = P_n$, P_1 has exactly one element, and for every $1 < i \leq n$, there exists $p \in P_i$ such that $P_i - P_{i-1} = \{p\}$ and p is comparable to exactly one element of P_{i-1}.

Clearly every decomposable set, treated as a poset, is connected. Also, by a routine induction on the number of elements one proves that every decomposable set, treated as a poset, is a generalized zigzag. The routine argument is left for the reader. Thus, by Corollary 4.1. and Proposition 4.4. every decomposable set is TC-feasible.

Now, let $A \subseteq P$ be a maximal decomposable subset. If A is properly contained in a connected component of P which contains it, then it follows that there is an element p in that component such that $A \cup \{p\}$ is not decomposable. Hence, p must be comparable to at least two elements $a_1, a_2 \in A$. Let $a_1, q_1, \ldots, q_n, a_2$ be a path in A which connects a_1 and a_2. Then $a_1, q_1, \ldots, q_n, a_2, p$ is a crown. The obtained contradiction proves that A must be equal to a connected component of P. Thus P is a disjoint union of TC-feasible posets and therefore (see Theorem 4.3.) is TC-feasible as well. □

References

[Benke93] Marcin Benke. Efficient Type Reconstruction in the Presence of Inheritance. In: A. M. Borzyszkowski, S. Sokołowski (Eds.) *MFCS'93: Mathematical Foundations of Computer science, Proc. 18th Intern. Symp.*, pages 272–280, Springer Verlag, LNCS 711, 1993.

[Benke95] Marcin Benke. Subtyping and Alternation. To appear as Technical Report, Institute of Informatics, Warsaw University, 1995.

[FederVardi93] T.A. Feder and M. Vardi. Monotone monadic SNP and constraint satisfaction. In *Proc. 25th ACM Symp. on Theory of Computing*, pages 612–622, 1993.

[HoangMitchell95] My Hoang and John C. Mitchell. Lower Bounds on Type Inference with Subtypes. In *Conf. Rec. 22nd ACM Symposium on Principles of Programming Languages*, pages 176–185, 1995.

[Mitchell84] John C. Mitchell. Coercion and Type Inference (summary). In *Conf. Rec. 11th ACM Symposium on Principles of Programming Languages*, pages 175–185, 1984.

[MitchellLincoln92] Patrick Lincoln and John C. Mitchell. Algorithmic Aspects of Type Inference with Subtypes. In *Conf. Rec. 19th ACM Symposium on Principles of Programming Languages*, pages 293–304, 1992.

[NevermannRival85] P. Nevermann and I. Rival. Holes in ordered sets. *Graphs and Combinatorics*, pages 339–350, 1985.

[OKeefeWand89] Mitchell Wand and Patrick M. O'Keefe. On the Complexity of Type Inference with Coercion. In *Conf. on Functional Programming Languages and Computer Architecture*, 1989.

[Tiuryn92] Jerzy Tiuryn. Subtype Inequalities. In *Proc. 7th IEEE Symposium on Logic in Computer Science*, pages 308–315, 1992.

[TiurynWand93] Jerzy Tiuryn and Mitchell Wand. Type reconstruction with recursive types and atomic subtyping. In: M.-C. Gaudel and J.-P. Jouannaud (Eds.) *TAPSOFT'93: Theory and Practice of Software Development, Proc. 4th Intern. Joint Conf. CAAP/FASE*, Springer-Verlag LNCS 668, 1993, pp.686-701.