On Broken Triangles (CP 2014)

Martin Cooper, Achref El Mouelhi, Cyril Terrioux, Bruno Zanuttini

▶ **To cite this version:**

Martin Cooper, Achref El Mouelhi, Cyril Terrioux, Bruno Zanuttini. On Broken Triangles (CP 2014). International Conference on Principles and Practice of Constraint Programming (CP 2014), Association for Constraint Programming, Sep 2014, Lyon, France. pp.9-24, 10.1007/978-3-319-10428-7_5 . hal-01095895

HAL Id: hal-01095895
https://hal.science/hal-01095895

Submitted on 16 Dec 2014

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

On Broken Triangles*

Martin C. Cooper† Achref El Mouelhi Cyril Terrioux‡
Bruno Zanuttini§

October 9, 2014

Abstract

A binary CSP instance satisfying the broken-triangle property (BTP) can be solved in polynomial time. Unfortunately, in practice, few instances satisfy the BTP. We show that a local version of the BTP allows the merging of domain values in arbitrary instances of binary CSP, thus providing a novel polynomial-time reduction operation. Extensive experimental trials on benchmark instances demonstrate a significant decrease in instance size for certain classes of problems. We show that BTP-merging can be generalised to instances with constraints of arbitrary arity and we investigate the theoretical relationship with resolution in SAT. A directional version of the general-arity BTP then allows us to extend the BTP tractable class previously defined only for binary CSP.

1 Introduction

At first sight one could assume that the discipline of constraint programming has come of age. On the one hand, efficient solvers are regularly used to solve real-world problems in diverse application domains while, on the other hand, a rich theory has been developed concerning, among other things, global constraints, tractable classes, reduction operations and symmetry. However, there often remains a large gap between theory and practice which is perhaps most evident when we look at the large number of deep results concerning tractable classes which have yet to find any practical application. The research reported in this paper is part of a long-term project to bridge the gap between theory and practice. Our aim is not only to develop new tools but also to explain why present tools work so well.

*supported by ANR Project ANR-10-BLAN-0210.
†IRIT, University of Toulouse III, 31062 Toulouse, France, cooper@irit.fr
‡LSIS, Aix-Marseille University, 13397 Marseille, France, {achref.elmouelhi, cyril.terrioux}@lsis.org
§GREYC, University of Caen Basse-Normandie, 14032 Caen, France, bruno.zanuttini@unicaen.fr

Most research on tractable classes has been based on classes defined by placing restrictions either on the types of constraints or on the constraint hyper-graph whose vertices are the variables and whose hyper-edges are the constraint scopes. Another way of defining classes of binary CSP instances consists in imposing conditions on the microstructure, a graph whose vertices are the possible variable-value assignments with an edge linking each pair of compatible assignments [9, 12]. If each vertex of the microstructure, corresponding to a variable-value assignment $\langle x, a\rangle$, is labelled by the variable x, then this so-called coloured microstructure retains all information from the original instance. The broken-triangle property (BTP) is a simple local condition on the coloured microstructure which defines a tractable class of binary CSP [5]. Inspired by the BTP, investigation of other forbidden patterns in the coloured microstructure has led to the discovery of new tractable classes [1, 4, 6, 8] as well as new reduction operations based on variable elimination [2].

For simplicity of presentation we use two different representations of constraint satisfaction problems. In the binary case, our notation is fairly standard, whereas in the general-arity case we use a notation close to the representation of SAT instances. This is for presentation only, though, and our algorithms do *not* need instances to be represented in this manner.

Definition 1 *A* binary CSP instance I *consists of*

- *a set* X *of* n *variables,*
- *a domain* $\mathcal{D}(x)$ *of possible values for each variable* $x \in X$*,*
- *a relation* $R_{xy} \subseteq \mathcal{D}(x) \times \mathcal{D}(y)$*, for each pair of distinct variables* $x, y \in X$*, which consists of the set of compatible pairs of values* (a, b) *for variables* (x, y)*.*

A partial solution *to* I *on* $Y = \{y_1, \ldots, y_r\} \subseteq X$ *is a set* $\{\langle y_1, a_1\rangle, \ldots, \langle y_r, a_r\rangle\}$ *such that* $\forall i, j \in [1, r]$, $(a_i, a_j) \in R_{y_i y_j}$*. A* solution *to* I *is a partial solution on* X*.*

For simplicity of presentation, Definition 1 assumes that there is exactly one constraint relation for each pair of variables. The number of constraints e is the number of pairs of variables x, y such that $R_{xy} \neq \mathcal{D}(x) \times \mathcal{D}(y)$. An instance I is *arc consistent* if for each pair of distinct variables $x, y \in X$, for each value $a \in \mathcal{D}(x)$, there is a value $b \in \mathcal{D}(y)$ such that $(a, b) \in R_{xy}$.

In our representation of general-arity CSP instances, we require the notion of *tuple* which is simply a set of variable-value assignments. For example, in the binary case, the tuple $\{\langle x, a\rangle, \langle y, b\rangle\}$ is *compatible* if $(a, b) \in R_{xy}$ and *incompatible* otherwise.

Definition 2 *A* (general-arity) CSP instance I *consists of*

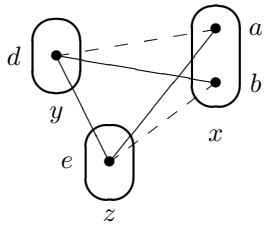

Figure 1: A broken triangle on two values a, b for a given variable x.

- *a set X of n variables,*
- *a domain $\mathcal{D}(x)$ of possible values for each variable $x \in X$,*
- *a set NoGoods(I) consisting of incompatible tuples.*

A partial solution *to I on $Y = \{y_1, \ldots, y_r\} \subseteq X$ is a tuple $t = \{\langle y_1, a_1\rangle, \ldots, \langle y_r, a_r\rangle\}$ such that no subset of t belongs to NoGoods(I). A* solution *is a partial solution on X.*

2 Value merging in binary CSP based on the BTP

In this section we consider a method, based on the BTP, for reducing domain size while preserving satisfiability. Instead of eliminating a value, as in classic reduction operations such as arc consistency or neighbourhood substitution, we merge two values. We show that the absence of broken-triangles [5] on two values for a variable x in a binary CSP instance allows us to merge these two values in the domain of x while preserving satisfiability. This rule generalises the notion of virtual interchangeability [11] as well as neighbourhood substitution [10].

It is known that if for a given variable x in an arc-consistent binary CSP instance I, the set of (in)compatibilities (known as a broken-triangle) shown in Figure 1 occurs for no two values $a, b \in \mathcal{D}(x)$ and no two assignments to two other variables, then the variable x can be eliminated from I without changing the satisfiability of I [5, 2]. In figures, each bullet represents a variable-value assignment, assignments to the same variable are grouped together within the same oval and compatible (incompatible) pairs of assignments are linked by solid (broken) lines. Even when this variable-elimination rule cannot be applied, it may be the case that for a given pair of values $a, b \in \mathcal{D}(x)$, no broken triangle occurs. We will show that if this is the case, then we can perform a domain-reduction operation which consists in merging the values a and b.

Definition 3 Merging *values $a, b \in \mathcal{D}(x)$ in a binary CSP consists in replacing a, b in $\mathcal{D}(x)$ by a new value c which is compatible with all variable-value assignments compatible with at least one of the assignments $\langle x, a\rangle$ or $\langle x, b\rangle$. A* value-merging condition *is a polytime-computable property $P(x, a, b)$ of*

assignments $\langle x, a\rangle$, $\langle x, b\rangle$ in a binary CSP instance I such that when $P(x, a, b)$ holds, the instance I' obtained from I by merging $a, b \in \mathcal{D}(x)$ is satisfiable if and only if I is satisfiable.

We now formally define the value-merging condition based on the BTP.

Definition 4 *A* broken triangle *on the pair of variable-value assignments $a, b \in \mathcal{D}(x)$ consists of a pair of assignments $d \in \mathcal{D}(y)$, $e \in \mathcal{D}(z)$ to distinct variables $y, z \in X \setminus \{x\}$ such that $(a, d) \notin R_{xy}$, $(b, d) \in R_{xy}$, $(a, e) \in R_{xz}$, $(b, e) \notin R_{xz}$ and $(d, e) \in R_{yz}$. The pair of values $a, b \in \mathcal{D}(x)$ is* BT-free *if there is no broken triangle on a, b.*

Proposition 5 *In a binary CSP instance, being BT-free is a value-merging condition. Furthermore, given a solution to the instance resulting from the merging of two values, we can find a solution to the original instance in linear time.*

Proof Let I be the original instance and I' the new instance in which a,b have been merged into a new value c. Clearly, if I is satisfiable then so is I'. It suffices to show that if I' has a solution s which assigns c to x, then I has a solution. Let s_a, s_b be identical to s except that s_a assigns a to x and s_b assigns b to x. Suppose that neither s_a nor s_b are solutions to I. Then, there are variables $y, z \in X \setminus \{x\}$ such that $\langle a, s(y)\rangle \notin R_{xy}$ and $\langle b, s(z)\rangle \notin R_{xz}$. By definition of the merging of a, b to produce c, and since s is a solution to I' containing $\langle x, c\rangle$, we must have $(b, s(y)) \in R_{xy}$ and $(a, s(z)) \in R_{xz}$. Finally, $(s(y), s(z)) \in R_{yz}$ since s is a solution to I'. Hence, $\langle y, s(y)\rangle$, $\langle z, s(z)\rangle$, $\langle x, a\rangle$, $\langle x, b\rangle$ forms a broken-triangle, which contradicts our assumption. Hence, the absence of broken triangles on assignments $\langle x, a\rangle$, $\langle x, b\rangle$ allows us to merge these assignments while preserving satisfiability.

Reconstructing a solution to I from a solution s to I' simply requires checking which of s_a or s_b is a solution to I. □ □

We can see that the BTP-merging rule, given by Proposition 5, generalises neighbourhood substitution [10]: if b is neighbourhood substitutable by a, then no broken triangle occurs on a, b and merging a and b produces a CSP instance which is identical (except for the renaming of the value a as c) to the instance obtained by simply eliminating b from $\mathcal{D}(x)$. BTP-merging also generalises the merging rule proposed by Likitvivatanavong and Yap [11]. The basic idea behind their rule is that if the two assignments $\langle x, a\rangle$, $\langle x, b\rangle$ have identical compatibilities with all assignments to all other variables except concerning at most one other variable, then we can merge a and b. This is clearly subsumed by BTP-merging.

The BTP-merging operation is not only satisfiability-preserving but, from Proposition 5, we know that we can also reconstruct a solution in polynomial time to the original instance I from

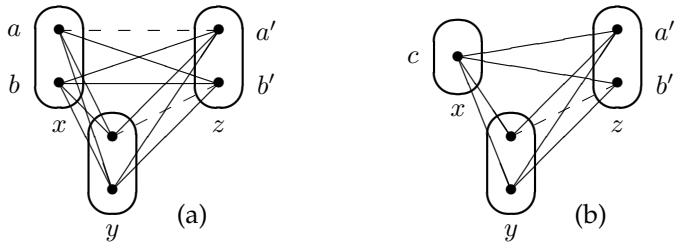

Figure 2: (a) A broken triangle exists on values a', b' at variable z. (b) After BTP-merging of values a and b in $\mathcal{D}(x)$, this broken triangle has disappeared.

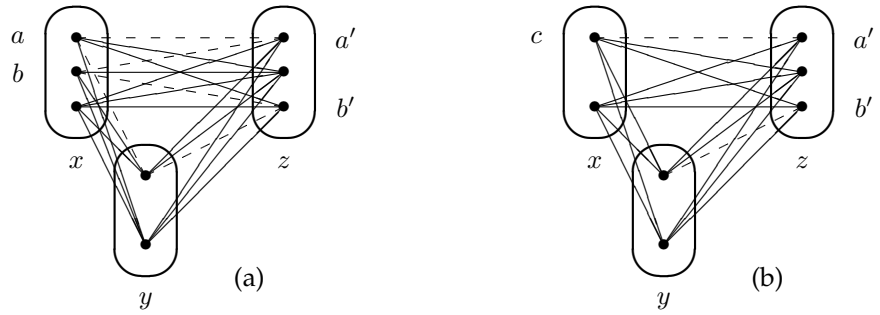

Figure 3: (a) This instance contains no broken triangle. (b) After BTP-merging of values a and b in $\mathcal{D}(x)$, a broken triangle has appeared on values $a', b' \in \mathcal{D}(z)$.

a solution to an instance I^m to which we have applied a sequence of merging operations until convergence. It is known that for the weaker operation of neighbourhood substitutability, all solutions to the original instance can be generated in $O(N(de + n^2))$ time, where N is the number of solutions to the original instance, n is the number of variables, d the maximum domain size and e the number of constraints [3]. We now show that a similar result also holds for the more general rule of BTP-merging.

Proposition 6 *Let I be a binary CSP instance and suppose that we are given the set of all solutions to the instance I^m obtained after applying a sequence of BTP-merging operations. All N solutions to I can then be determined in $O(Nn^2d)$ time.*

Proof Let I' be the CSP instance which results after performing a single BTP-merging operation of values $a, b \in \mathcal{D}(x)$ in I. As we saw in the proof of Proposition 5, given the set of solutions $sol(I')$ to I' we can generate the set of solutions to I by testing for each $s \in Sol(I')$ whether s_a or s_b (or both) are solutions to I. This requires $O(n)$ time per solution to I, since there are at most $n-1$ constraints to be tested involving the variable x, and at least one of s_a or s_b is a solution to I.

The total number of BTP-merging operations performed to transform I into I^m is at most $n(d-1)$. Therefore, the total time to generate all N solutions to I from the set of solutions to I^m is $O(Nn^2d)$. □ □

The weaker operation of neighbourhood substitution has the property that two different con-

domain	no. instances	no. values	no. values deleted	%age deleted
BH-4-13	6	7,334	3,201	44%
BH-4-4	10	674	322	48%
BH-4-7	20	2,102	883	42%
ehi-85	98	2,079	891	43%
ehi-90	100	2,205	945	43%
graph-coloring/school	8	4,473	104	2%
graph-coloring/sgb/book	26	1,887	534	28%
jobShop	45	6,033	388	6%
marc	1	6400	6,240	98%
os-taillard-4	30	2,932	1,820	62%
os-taillard-5	28	6,383	2,713	43%
rlfapGraphsMod	5	14,189	5,035	35%
rlfapScens	5	12,727	821	6%
rlfapScensMod	9	9,398	1,927	21%
others	1919	1,396	28	0.02%

Table 1: Results of experiments on CSP benchmark problems.

vergent sequences of eliminations by neighbourhood substitution necessarily produce isomorphic instances I_1^m, I_2^m [3] . This is not the case for BTP-merging. Firstly, and perhaps rather surprisingly, BTP-merging can have as a side-effect to eliminate broken triangles. This is illustrated in the 3-variable instance shown in Figure 2. The instance in Figure 2(a) contains a broken triangle on values a', b' for variable z, but after BTP-merging of values $a, b \in \mathcal{D}(x)$ into a new value c, as shown in Figure 2(b), there are no broken triangles in the instance. Secondly, BTP-merging of two values in $\mathcal{D}(x)$ can introduce a broken triangle on a variable $z \neq x$, as illustrated in Figure 3. The instance in Figure 3(a) contains no broken triangle, but after the BTP-merging of $a, b \in \mathcal{D}(x)$ into a new value c, a broken triangle has been created on values $a', b' \in \mathcal{D}(z)$.

3 Experimental trials

To test the utility of BTP-merging we performed extensive experimental trials on benchmark instances from the International CP Competition[1]. For each instance not including global constraints, we performed BTP-mergings until convergence with a time-out of one hour. In total, we obtained results for 2,547 instances out of 3,811 benchmark instances. In the other instances the search for all BTP-mergings did not terminate within a time-out of one hour.

All instances from the benchmark-domain `hanoi` satisfy the broken-triangle property and BTP-merging reduced all variable domains to singletons. After establishing arc consistency, 38 instances from diverse benchmark-domains satisfy the BTP, including all instances from the benchmark-domain `domino`. We did not count those instances for which arc consistency detects inconsistency

[1] http://www.cril.univ-artois.fr/CPAI08

by producing a trivial instance with empty variable domains (and which trivially satisfies the BTP). In all instances from the `pigeons` benchmark-domain with a suffix `-ord`, BTP-merging again reduced all domains to singletons. This is because BTP-merging can eliminate broken triangles, as pointed out in Section 2, and hence can render an instance BTP even though initially it was not BTP. The same phenomenon occurred in a 680-variable instance from the benchmark-domain `rlfapGraphsMod` as well as the 3-variable instance `ogdPuzzle`.

Table 1 gives a summary of the results of the experimental trials. We do not include those instances mentioned above which are entirely solved by BTP-merging. We give details about those benchmark-domains where BTP-merging was most effective. All other benchmark-domains are grouped together in the last line of the table. The table shows the number of instances in the benchmark-domain, the average number of values (i.e. variable-value assignments) in the instances from this benchmark-domain, the average number of values deleted (i.e. the number of BTP-merging operations performed) and finally this average represented as a percentage of the average number of values.

We can see that for certain types of problem, BTP-merging is very effective, whereas for others (grouped together in the last line of the table) hardly any merging of values occurred.

4 Generalising BTP-merging to constraints of arbitrary arity

In the remainder of the paper, we assume that the constraints of a general-arity CSP instance I are given in the form described in Definition 2, i.e. as a set of incompatible tuples NoGoods(I), where a tuple is a set of variable-value assignments. For simplicity of presentation, we use the predicate Good(I, t) which is true iff the tuple t is a partial solution, i.e. t does not contain any pair of distinct assignments to the same variable and $\nexists t' \subseteq t$ such that $t' \in$ NoGoods(I). We first generalise the notion of broken triangle and merging to the general-arity case, before showing that absence of broken triangles allows merging.

Definition 7 *A* general-arity broken triangle *(GABT) on values $a, b \in \mathcal{D}(x)$ consists of a pair of tuples t, u (containing no assignments to variable x) satisfying the following conditions:*

1. *Good($I, t \cup u$) $\wedge$ Good($I, t \cup \{\langle x, a\rangle\}$) $\wedge$ Good($I, u \cup \{\langle x, b\rangle\}$)*
2. *$t \cup \{\langle x, b\rangle\} \in$ NoGoods(I) $\wedge$ $u \cup \{\langle x, a\rangle\} \in$ NoGoods(I)*

The pair of values $a, b \in \mathcal{D}(x)$ is GABT-free *if there is no broken triangle on a, b.*

Observe that Good($I, t \cup \{\langle x, a\rangle\}$) entails $t \cup \{\langle x, a\rangle\} \notin$ NoGoods(I). Hence to decide whether there is a GABT on a, b in a CSP instance, one can either explore all pairs $t \cup \{\langle x, b\rangle\}, u \cup \{\langle x, a\rangle\} \in$

NoGoods(I), as suggested by Definition 7, or, equivalently, explore all pairs $t \cup \{\langle x, a\rangle\}, u \cup \{\langle x, b\rangle\}$ of tuples explicitly allowed by the constraints in I. Whatever the representation, a pair t, u can be checked to be a GABT on a, b by evaluating the properties of Definition 7, all of which involve only constraint checks. Hence deciding whether a pair a, b is GABT-free is polytime for constraints given in extension (as the set of satisfying assignments) as well as for those given by nogoods (the set of assignments violating the constraint).

Definition 8 Merging *values $a, b \in \mathcal{D}(x)$ in a general-arity CSP instance I consists in replacing a, b in $\mathcal{D}(x)$ by a new value c which is compatible with all variable-value assignments compatible with at least one of the assignments $\langle x, a\rangle$ or $\langle x, b\rangle$, thus producing an instance I' with the new set of nogoods defined as follows:*

$$\begin{aligned}
\text{NoGoods}(I') \;=\; & \{t \in \text{NoGoods}(I) \mid \langle x, a\rangle, \langle x, b\rangle \notin t\} \\
\cup\; & \{t \cup \{\langle x, c\rangle\} \mid t \cup \{\langle x, a\rangle\} \in \text{NoGoods}(I) \;\wedge \\
& \quad \exists t' \in \text{NoGoods}(I) \text{ s.t. } t' \subseteq t \cup \{\langle x, b\rangle\}\} \\
\cup\; & \{t \cup \{\langle x, c\rangle\} \mid t \cup \{\langle x, b\rangle\} \in \text{NoGoods}(I) \;\wedge \\
& \quad \exists t' \in \text{NoGoods}(I) \text{ s.t. } t' \subseteq t \cup \{\langle x, a\rangle\}\}
\end{aligned}$$

A value-merging condition *is a polytime-computable property $P(x, a, b)$ of assignments $\langle x, a\rangle$, $\langle x, b\rangle$ in a CSP instance I such that when $P(x, a, b)$ holds, the instance I' is satisfiable if and only if I is satisfiable.*

Clearly, this merging operation can be performed in polynomial time whether constraints are represented positively in extension or negatively as nogoods.

Proposition 9 *In a general-arity CSP instance, being GABT-free is a value-merging condition. Furthermore, given a solution to the instance resulting from the merging of two values, we can find a solution to the original instance in linear time.*

Proof In order to prove that satisfiability is preserved by this merging operation, it suffices to show that if s is a solution to I' containing $\langle x, c\rangle$, then either $s_a = (s \setminus \{\langle x, c\rangle\}) \cup \{\langle x, a\rangle\}$ or $s_b = (s \setminus \{\langle x, c\rangle\}) \cup \{\langle x, b\rangle\}$ is a solution to I. Suppose, for a contradiction that this is not the case. Then there are tuples $t, u \subseteq s \setminus \{\langle x, c\rangle\}$ such that $t \cup \{\langle x, b\rangle\} \in$ NoGoods(I) and $u \cup \{\langle x, a\rangle\} \in$ NoGoods(I). Since t, u are subsets of the solution s to I' and t, u contain no assignments to x, we have Good($I, t \cup u$). Since $t \cup \{\langle x, c\rangle\}$ is a subset of the solution s to I', we have $t \cup \{\langle x, c\rangle\} \notin$ NoGoods(I'). By the definition of NoGoods(I') given in Definition 8, and since $t \cup \{\langle x, b\rangle\} \in$ NoGoods(I), we know that $\nexists t' \in$ NoGoods(I) such that $t' \subseteq t \cup \{\langle x, a\rangle\}$. But then Good($I, t \cup \{\langle x, a\rangle\}$). By a symmetric

argument, we can deduce Good($I, u \cup \{\langle x, b\rangle\}$). This provides the contradiction we were looking for, since we have shown that a general-arity broken triangle occurs on t, u, $\langle x, a\rangle$, $\langle x, b\rangle$.

Reconstructing a solution to the original instance can be achieved in linear time, since it suffices to verify which (or both) of s_a or s_b is a solution to I. □ □

Relationship with Resolution in SAT

We now show that in the case of Boolean domains, there is a close relationship between merging two values a, b on which no GABT occurs and a common preprocessing operation used by SAT solvers. Given a propositional CNF formula φ in the form of a set of clauses (each clause C_i being represented as a set of literals) and a variable x occurring in φ, recall that *resolution* is the process of inferring the clause $(C_0 \cup C_1)$ from the two clauses $(\{\bar{x}\} \cup C_0), (\{x\} \cup C_1)$. Define the formula $Res(x, \varphi)$ to be the result of performing all such resolutions on φ, removing all clauses containing x or $\bar{x}$, and removing subsumed clauses:

$$Res(x, \varphi) = \min_{\subset}(\{C \mid C \in \varphi; x, \bar{x} \notin C\} \cup \{(C_0 \cup C_1) \mid (\{\bar{x}\} \cup C_0), (\{x\} \cup C_1) \in \varphi\})$$

It is a well-known fact that $Res(x, \varphi)$ is satisfiable if and only if φ is.

Eliminating variables in this manner from SAT instances, to get an equisatisfiable formula with less variables, is a common preprocessing step in SAT solving, and is typically performed provided it does not increase the size of the formula [7]. A particular case is when it amounts to simply removing all occurrences of x, which is the case, for instance, if x or $\bar{x}$ is unit or pure in φ, or if all resolutions on x yield a tautological clause.

Definition 10 *A variable x is said to be* erasable *from a CNF φ if*

$$Res(x, \varphi) \subseteq \{C \mid C \in \varphi; x, \bar{x} \notin C\} \ \cup \ \{C_0 \mid (\{\bar{x}\} \cup C_0) \in \varphi\} \ \cup \ \{C_1 \mid (\{x\} \cup C_1) \in \varphi\}$$

A CNF φ can be seen as the CSP instance I_φ on the set X of variables occurring in φ, with $\mathcal{D}(x) = \{\top, \bot\}$ for all $x \in X$, and NoGoods(I_φ) $= \{\overline{C} \mid C \in \varphi\}$, where $\overline{(\{x_1, \cdots x_p, \bar{x}_{p+1}, \cdots, \bar{x}_q\})} = \{\langle x_1, \bot\rangle, \ldots, \langle x_p, \bot\rangle, \langle x_{p+1}, \top\rangle, \ldots, \langle x_q, \top\rangle\}$.

Proposition 11 *Assume that no GABT occurs on values $\bot, \top$ for x in I_φ. Assume moreover that no clause in φ is subsumed by another one*[2]*. Then x is erasable from φ.*

Proof Rephrasing Definition 7 in terms of clauses, for any two clauses $(\{\bar{x}\} \cup C_0), (\{x\} \cup C_1) \in \varphi$

[2]This is without loss of generality since such clauses can be removed in polytime and such removal preserves logical equivalence.

we have one of (i) $\exists C \in \varphi, C \subseteq (C_0 \cup C_1)$, (ii) $\exists C' \in \varphi, C' \subseteq (C_0 \cup \{x\})$, or (iii) $\exists C'' \in \varphi, C'' \subseteq (C_1 \cup \{\bar{x}\})$. Moreover, in Case (ii) C' must contain x, for otherwise the clause $(\{\bar{x}\} \cup C_0)$ would be subsumed in φ, contradicting our assumption. Similarly, in Case (iii) C'' must contain $\bar{x}$.

In Case (i) the resolvent $(C_0 \cup C_1)$ of $(\{\bar{x}\} \cup C_0), (\{x\} \cup C_1)$ is subsumed by C in $Res(x, \varphi)$, and hence does not occur in it. Similarly, in the second case $(C_0 \cup C_1)$ is subsumed by the resolvent of $(\{\bar{x}\} \cup C_0)$ and C', which is precisely C_0. The third case is dual. We finally have that the only resolvents added are of the form C_0 (resp. C_1) for some clause $(\{\bar{x}\} \cup C_0)$ (resp. $(\{x\} \cup C_1)$) of φ, as required. □ □

We can show the converse is also true provided that a very reasonable property holds.

Proposition 12 *Assume that φ satisfies: $\forall(\{x\} \cup C) \in \varphi, \nexists C' \subseteq C, (\{\bar{x}\} \cup C') \in \varphi$ and dually $\forall(\{\bar{x}\} \cup C) \in \varphi, \nexists C' \subseteq C, (\{x\} \cup C') \in \varphi$. If x is erasable from φ, then no GABT occurs on values $\bot, \top$ for x in I_φ.*

Proof Assume for a contradiction that there is a GABT on values $\bot, \top$ for x in I_φ, let t, u be witnesses to this, and write $t \cup \{\langle x, \top\rangle\} = \overline{(\{\bar{x}\} \cup C_0)}$, $u \cup \{\langle x, \bot\rangle\} = \overline{(\{x\} \cup C_1)}$. Then the clause $(C_0 \cup C_1)$ is produced by resolution on x. Since x is erasable, $(C_0 \cup C_1)$ is equal to or subsumed by a clause $C \in Res(x, \varphi)$, where (applying Definition 10 in reverse) either C, or $(\{x\} \cup C)$, or $(\{\bar{x}\} \cup C)$ is in φ. The first case contradicts Good($I_\varphi, t \cup u$), so assume by symmetry $(\{x\} \cup C) \in \varphi$. From $C \notin \varphi$ and $C \in Res(x, \varphi)$ we get $\exists C' \subseteq C, (\{\bar{x}\} \cup C') \in \varphi$. But then the pair of clauses $(\{x\} \cup C), (\{\bar{x}\} \cup C') \in \varphi$ violates the assumption of the claim. □ □

5 A tractable class of general-arity CSP

In binary CSP, the broken-triangle property defines an interesting tractable class when broken-triangles are forbidden according to a given variable ordering. Unfortunately, the original definition of BTP was limited to binary CSPs [5]. Section 4 described a general-arity version of the broken-triangle property whose absence on two values allows these values to be merged while preserving satisfiability. An obvious question is whether GABT-freeness can be adapted to define a tractable class. In this section we show that this is possible for a fixed variable ordering, but not if the ordering is unknown.

Definition 7 defined a general-arity broken triangle (GABT). What happens if we forbid GABTs according to a given variable ordering? Absence of GABTs on two values a, b for the last variable x in the variable ordering allows us to merge a and b while preserving satisfiability. It is possible to show that if GABTs are absent on all pairs of values for x, then we can merge all values in

the domain $D(x)$ of x to produce a singleton domain. This is because (as we will show later) merging a and b, to produce a merged value c, cannot introduce a GABT on c, d for any other value $d \in \mathcal{D}(x)$. Once the domain $D(x)$ becomes a singleton $\{a\}$, we can clearly eliminate x from the instance, by deleting $\langle x, a\rangle$ from all nogoods, without changing its satisfiability. It is at this moment that GABTs may be introduced on other variables, meaning that forbidding GABTs according to a variable ordering does not define a tractable class.

Nevertheless, we will show that strengthening the general-arity BTP allows us to avoid this problem. The resulting directional general-arity version of BTP (for a known variable ordering) then defines a tractable class which includes the binary BTP tractable class as a special case.

Note that the set of general-arity CSP instances whose dual instance satisfies the BTP also defines a tractable class which can be recognised in polynomial time even if the ordering of the variables in the dual instance is unknown [8]. This DBTP class is incomparable with the class we present in the present paper (which is equivalent to BTP in binary CSP) since DBTP is known to be incomparable with the BTP class already in the special case of binary CSP [8].

5.1 Directional general-arity BTP

We suppose given a total ordering $<$ of the variables of a CSP instance I. We write $t^{<x}$ to represent the subset of the tuple t consisting of assignments to variables occurring before x in the order $<$, and $Vars(t)$ to denote the set of all variables assigned by t.

Definition 13 *A* directional general-arity (DGA) broken triangle *on assignments a, b to variable x in a CSP instance I is a pair of tuples t, u (containing no assignments to variable x) satisfying the following conditions:*

1. $t^{<x}$ *and* $u^{<x}$ *are non-empty*
2. *Good(*$I, t^{<x} \cup u^{<x}$*)* $\wedge$ *Good(*$I, t^{<x} \cup \{\langle x, a\rangle\}$*)* $\wedge$ *Good(*$I, u^{<x} \cup \{\langle x, b\rangle\}$*)*
3. $\exists t'$ *s.t.* $Vars(t') = Vars(t)$ $\wedge$ $(t')^{<x} = t^{<x}$ $\wedge$ $t' \cup \{\langle x, a\rangle\} \notin$ *NoGoods(I)*
4. $\exists u'$ *s.t.* $Vars(u') = Vars(u)$ $\wedge$ $(u')^{<x} = u^{<x}$ $\wedge$ $u' \cup \{\langle x, b\rangle\} \notin$ *NoGoods(I)*
5. $t \cup \{\langle x, b\rangle\} \in$ *NoGoods(I)* $\wedge$ $u \cup \{\langle x, a\rangle\} \in$ *NoGoods(I)*

I satisfies the directional general-arity broken-triangle property (DGABTP) *according to the variable ordering $<$ if no directional general-arity broken triangle occurs on any pair of values a, b for any variable x.*

We will show that any instance I satisfying the DGABTP can be solved in polynomial time by repeatedly alternating the following two operations: (i) merge all values in the last remaining variable (according to the order $<$); (ii) eliminate this variable when its domain becomes a singleton. We will give the two operations (merging and variable-elimination) and show that both operations preserve satisfiability and that neither of them can introduce DGA broken triangles. Moreover, as for GABT-freeness, the DGABTP can be tested in polynomial time for a given order whether constraints are given as tables of satisfying assignments or as nogoods. Indeed, in the former case, using items (3) and (4) in Definition 13 we can restrict the search for a DGA broken triangle to pairs of tuples satisfying some constraint (there must be a constraint with scope $Vars(t' \cup \{x\})$ since there is a nogood on these variables by item (5), and similarly for u'). This is sufficient to define a tractable class.

5.2 Merging

Let x be the last variable according to the variable order $<$. When values a, b in the domain of variable x do not belong to any DGA broken triangle, we can replace a, b by a new value c to produce an instance I' with the new set of nogoods given by Definition 8. Since x is the last variable in the ordering $<$, DGA broken triangles on $a, b \in \mathcal{D}(x)$ are GA broken triangles (and vice versa). Thus, from Proposition 9 we can deduce that satisfiability is preserved by this merging operation. What remains to be shown is that merging two values in the domain of the last variable cannot introduce the forbidden pattern.

Lemma 14 *Merging two values a, b into a value c in the domain of the last variable x (according to the variable order $<$) in an instance I cannot introduce a directional general-arity broken triangle (DGABT) in the resulting instance I'.*

Proof We first claim that this operation cannot introduce a DGABT on a variable $y < x$. Indeed, assume there is a DGABT on $d, e \in \mathcal{D}(y)$ in I', that is, that there are tuples v, w such that

1. $v^{<y}$ and $w^{<y}$ are non-empty
2. Good$(I', v^{<y} \cup w^{<y})$ $\wedge$ Good$(I', v^{<y} \cup \{\langle y, d\rangle\})$ $\wedge$ Good$(I', w^{<y} \cup \{\langle y, e\rangle\})$
3. $\exists v'$ $Vars(v') = Vars(v)$ $\wedge$ $(v')^{<y} = v^{<y}$ $\wedge$ $v' \cup \{\langle y, d\rangle\} \notin$ NoGoods(I')
4. $\exists w'$ $Vars(w') = Vars(w)$ $\wedge$ $(w')^{<y} = w^{<y}$ $\wedge$ $w' \cup \{\langle y, e\rangle\} \notin$ NoGoods(I')
5. $v \cup \{\langle y, e\rangle\} \in$ NoGoods(I') $\wedge$ $w \cup \{\langle y, d\rangle\} \in$ NoGoods(I')

If v' contains the assignment $\langle x, c\rangle$ then, by construction of NoGoods(I') (Definition 8), $\exists v'' \in \{(v' \setminus \langle x, c\rangle) \cup \{\langle x, a\rangle\}, (v' \setminus \langle x, c\rangle) \cup \{\langle x, b\rangle\}\}$ such that $v'' \cup \{\langle y, d\rangle\} \notin$ NoGoods(I). If v' does not contain $\langle x, c\rangle$ then let $v'' = v'$. Define w'' in a similar way. Now considering the last item, if v contains $\langle x, c\rangle$ then by construction of NoGoods(I') there is v''' assigning a or b to x and otherwise equal to v, such that $v''' \cup \{\langle y, e\rangle\}$ was in NoGoods(I), and if $v \not\ni \langle x, c\rangle$ we let $v''' = v$. We define w''' similarly. Then:

1. $(v''')^{<y} = v^{<y}$ and $(w''')^{<y} = w^{<y}$ are non-empty
2. Good($I, (v''')^{<y} \cup (w''')^{<y}$) $\wedge$ Good($I, (v''')^{<y} \cup \{\langle y, d\rangle\}$) $\wedge$ Good($I, (w''')^{<y} \cup \{\langle y, e\rangle\}$) (since x is the last variable, $(v''')^{<y} = v^{<y}$ and $(w''')^{<y} = w^{<y}$)
3. $Vars(v'') = Vars(v''')$ $\wedge$ $(v'')^{<y} = (v''')^{<y}$ $\wedge$ $v'' \cup \{\langle y, d\rangle\} \notin$ NoGoods(I)
4. $Vars(w'') = Vars(w''')$ $\wedge$ $(w'')^{<y} = (w''')^{<y}$ $\wedge$ $w'' \cup \{\langle y, e\rangle\} \notin$ NoGoods(I))
5. $v''' \cup \{\langle y, e\rangle\} \in$ NoGoods(I) $\wedge$ $w''' \cup \{\langle y, d\rangle\} \in$ NoGoods(I)

that is, there was a DGABT on d, e in I, contradicting our assumption.

We now show that a broken triangle cannot be introduced on x. Observe that since x is the last variable, for all tuples t not containing an assignment to x, $t^{<x} = t$ holds. We use this tacitly in the rest of the proof. Suppose for a contradiction that I contained no DGABT, but that after merging $a, b \in D(x)$ in I to produce the instance I', in which a, b have been replaced by a new value c, we have a DGABT on c, d. Then there is a pair of non-empty tuples t, u (containing no assignments to variable x) satisfying in particular the following conditions:

(1) Good($I', t \cup u$)
(2) Good($I', t \cup \{\langle x, c\rangle\}$)
(3) Good($I', u \cup \{\langle x, d\rangle\}$)
(4) $t \cup \{\langle x, d\rangle\} \in$ NoGoods(I')
(5) $u \cup \{\langle x, c\rangle\} \in$ NoGoods(I')

We show that there was a DGABT in I either on a, d, on b, d or on a, b.

Since merging only affects tuples containing $\langle x, a\rangle$ or $\langle x, b\rangle$, (1) implies that Good($I, t \cup u$) and hence Good($I, t \cup u'$) for all $u' \subseteq u$. Similarly, (3) implies that Good($I, u \cup \{\langle x, d\rangle\}$) and hence Good($I, u' \cup \{\langle x, d\rangle\}$) for all $u' \subseteq u$. Similarly, (4) implies that $t \cup \{\langle x, d\rangle\} \in$NoGoods($I$).
There are three possible cases to consider:

(a) Good($I, t \cup \{\langle x, a\rangle\}$),

(b) Good($I, t \cup \{\langle x, b\rangle\}$),

(c) $\exists t_1, t_2 \subseteq t$ such that $t_1 \cup \{\langle x, a\rangle\}$, $t_2 \cup \{\langle x, b\rangle\} \in$ NoGoods(I).

case (a): By Definition 8 of the creation of nogoods during merging, (5) implies that $\exists u' \subseteq u$ such that $u' \cup \{\langle x, a\rangle\} \in$ NoGoods(I). We know that u' is non-empty since $u' \cup \{\langle x, a\rangle\} \in$ NoGoods(I) but Good($I, t \cup \{\langle x, a\rangle\}$) (and hence Good($I, \{\langle x, a\rangle\}$)). We have Good($I, t \cup u'$), Good($I, t \cup \{\langle x, a\rangle\}$) (and hence $t \cup \{\langle x, a\rangle\} \notin$ NoGoods(I)), Good($I, u' \cup \{\langle x, d\rangle\}$) (and hence $u' \cup \{\langle x, d\rangle\} \notin$ NoGoods(I)), $t \cup \{\langle x, d\rangle\} \in$ NoGoods(I), $u' \cup \{\langle x, a\rangle\} \in$ NoGoods(I) and hence there was a DGABT on a, d in I.

case (b): Symmetrically to case (a), there was a DGABT on b, d in I.

case (c): We claim that Good($I, t_1 \cup \{\langle x, b\rangle\}$). If not, then we would have $\exists t_3 \subseteq t_1$ such that $t_3 \cup \{\langle x, b\rangle\} \in$ NoGoods(I) which would imply $t_1 \cup \{\langle x, c\rangle\} \in$ NoGoods(I') which is impossible since, by (2) above, we have Good($I', t \cup \{\langle x, c\rangle\}$). By a symmetrical argument, we can deduce Good($I, t_2 \cup \{\langle x, a\rangle\}$). Since Good($I, t \cup u$) and $t_1, t_2 \subseteq t$, we have Good($I, t_1 \cup t_2$). Since $t_1 \cup \{\langle x, a\rangle\} \in$ NoGoods(I) and Good($I, t_2 \cup \{\langle x, a\rangle\}$) (and hence Good($I, \{\langle x, a\rangle\}$)), we must have $t_1 \neq \emptyset$. By a symmetric argument, $t_2 \neq \emptyset$. We therefore have non-empty tuples t_1, t_2 such that Good($I, t_1 \cup t_2$), Good($I, t_1 \cup \{\langle x, b\rangle\}$ (and hence $t_1 \cup \{\langle x, b\rangle\} \notin$ NoGoods(I)), Good($I, t_2 \cup \{\langle x, a\rangle\}$) (and hence $t_2 \cup \{\langle x, a\rangle\} \notin$ NoGoods(I)), $t_1 \cup \{\langle x, a\rangle\} \in$ NoGoods(I), $t_2 \cup \{\langle x, b\rangle\} \in$ NoGoods(I) and hence we have a DGABT in I on a, b.

Since in each of the three possible cases, we produced a contradiction, this completes the proof. □ □

5.3 Tractability of DGABTP for a known variable ordering

Theorem 15 *A CSP instance I satisfying the DGABTP on a given variable ordering can be solved in polynomial time.*

Proof Suppose that I satisfies the DGABTP for variable ordering $<$ and that x is the last variable according to this ordering. Lemma 14 tells us that DGA broken triangles cannot be introduced by merging all elements in $\mathcal{D}(x)$ to form a singleton domain $\{a\}$. At this point it may be that $\{\langle x, a\rangle\}$ is a nogood. In this case the instance is clearly unsatisfiable and the algorithm halts returning this result. If not then we simply delete $\langle x, a\rangle$ from all nogoods in which it occurs. This operation of variable elimination clearly preserves satisfiability. It is polynomial time to recursively apply this merging and variable elimination algorithm until a nogood corresponding to a singleton domain is discovered or until all variables have been eliminated (in which case I is satisfiable).

To complete the proof of correction of this algorithm, it only remains to show that elimination of the last variable x cannot introduce a DGA broken triangle on another variable y. For all

tuples t, u and all values $c, d \in D(y)$, none of Good($I, t^{<y} \cup u^{<y}$), Good($I, t^{<y} \cup \{\langle y, c\rangle\}$) and Good($I, u^{<y} \cup \{\langle y, d\rangle\}$) can become true due to the variable elimination operation described above. On the other hand it is possible that $t \cup \{\langle y, d\rangle\}$ or $u \cup \{\langle y, c\rangle\}$ becomes a nogood due to variable elimination. Without loss of generality, suppose that $t \cup \{\langle y, d\rangle\}$ becomes a nogood and that $t' \cup \{\langle y, d\rangle\}$ is not a nogood for some t' such that $(t')^{<y} = t^{<y}$. Then by construction there was a nogood $t \cup \{\langle y, d\rangle\} \cup \{\langle x, a\rangle\}$ before the variable x (with singleton domain $\{a\}$) was eliminated, and $t' \cup \{\langle y, d\rangle\} \cup \{\langle x, a\rangle\}$ was not a nogood. But then there was a DGA broken triangle (given by tuples $t \cup \{\langle x, a\rangle\}$, u on values $c, d \in D(y)$) before elimination of x. This contradiction shows that variable elimination cannot introduce DGA broken triangles. □ □

5.4 Finding a DGABTP variable ordering is NP-hard

An important question is the tractability of the recognition problem of the class DGABTP when the variable order is not given, i.e. testing the existence of a variable ordering for which a given instance satisfies the DGABTP. In the case of binary CSP, this test can be performed in polynomial time [5]. Unfortunately, as the following theorem shows, the problem becomes NP-complete in the general-arity case.

Theorem 16 *Testing the existence of a variable ordering for which a CSP instance satisfies the DGABTP is NP-complete (even if the arity of constraints is at most 5).*

Proof The problem is in NP since verifying the DGABTP is polytime for a given order, so it suffices to give a polynomial-time reduction from the well-known NP-complete problem 3SAT. Let I_{3SAT} be an instance of 3SAT with variables $X_1, \ldots, X_N$ and clauses $C_1, \ldots, C_M$. We will create a CSP instance I_{CSP} which has a DGABTP variable-ordering if and only if I_{3SAT} is satisfiable. For each variable X_i of I_{3SAT}, we add two variables x_i, y_i to I_{CSP}. To complete the set of variables in I_{CSP}, we add three special variables v, w, z. We add constraints to I_{CSP} in such a way that each DGABTP ordering of its variables corresponds to a solution to I_{3SAT} (and vice versa). The role of the variable z is critical: a DGABTP ordering $>$ of the variables of I_{CSP} corresponds to a solution to I_{3SAT} in which $X_i = \texttt{true} \Leftrightarrow x_i > z$. The variables y_i are used to code $\overline{X_i}$: $y_i > z$ in a DGABTP ordering if and only if $X_i = \texttt{false}$ in the corresponding solution to I_{3SAT}. The variables v, w are necessary for our construction and will necessarily satisfy $v, w < z$ in a DGABTP ordering. Each clause $C = l_1 \vee l_2 \vee l_3$, where l_1, l_2, l_3 are literals in I_{3SAT}, is imposed in I_{CSP} by adding constraints which force one of $\overline{l_1}$, $\overline{l_2}$, $\overline{l_3}$ to be false. To give a concrete example, if $C = X_1 \vee X_2 \vee X_3$, then constraints are added to I_{CSP} to force $y_1 < z$ or $y_2 < z$ or $y_3 < z$ in a DGABTP ordering. If the clause C contains a negated variable $\overline{X_i}$ instead of X_i, it suffices to replace y_i by x_i.

We now give in detail the necessary gadgets in I_{CSP} to enforce each of the following properties in a DGABTP ordering:

1. $v, w < z$
2. $y_i < z \Leftrightarrow x_i > z$
3. $y_i < z$ or $y_j < z$ or $y_k < z$

We introduce broken triangles in order to impose these properties. However, it is important not to inadvertently introduce other broken triangles. This can be avoided by making all pairs of assignments $\langle x, a\rangle$, $\langle x', a'\rangle$ from two different gadgets incompatible (i.e. $\{\langle x, a\rangle, \langle x', a'\rangle\} \in$ NoGoods(I_{CSP})). We also assume that two gadgets which use the same variable x use distinct domain values in $\mathcal{D}(x)$. To avoid creating a trivial instance in which the gadgets disappear after establishing arc consistency, we can also add extra values in each domain which are compatible with all variable-value assignments in the gadgets.

We give the details of the three types of gadget:

1. The gadget to force $v, w < z$ in a DGABTP ordering consists of values $a_0 \in \mathcal{D}(z)$, $b_0, b_1 \in \mathcal{D}(v)$, $c_0, c_1 \in \mathcal{D}(w)$ and three nogoods $\{\langle z, a_0\rangle, \langle v, b_0\rangle\}$, $\{\langle z, a_0\rangle, \langle w, c_0\rangle\}$, $\{\langle v, b_1\rangle, \langle w, c_1\rangle\}$. The only way to satisfy the DGABTP on this triple of variables is to have $v, w < z$ since there are broken triangles on variables v and w.

2. To force $y_i < z \Leftrightarrow x_i > z$ in a DGABTP ordering we use two gadgets, the first to force $y_i > z \vee x_i > z$ and the second to force $y_i < z \vee x_i < z$.

   The first gadget is a broken triangle consisting of values $a_1, a_2 \in \mathcal{D}(z)$, $d_0 \in \mathcal{D}(x_i)$, $e_0 \in \mathcal{D}(y_i)$ and two nogoods $\{\langle z, a_1\rangle, \langle x_i, d_0\rangle\}$, $\{\langle z, a_2\rangle, \langle y_i, e_0\rangle\}$. In a DGABTP ordering we must have $y_i > z \vee x_i > z$.

   The second gadget consists of values $a_3, a_4 \in \mathcal{D}(z)$, $b_2 \in \mathcal{D}(v)$, $c_2 \in \mathcal{D}(w)$, $d_1 \in \mathcal{D}(x_i)$, $e_1 \in \mathcal{D}(y_i)$ and four nogoods $\{\langle z, a_3\rangle, \langle v, b_2\rangle, \langle x_i, d_1\rangle\}$, $\{\langle z, a_4\rangle, \langle v, b_2\rangle, \langle x_i, d_1\rangle\}$, $\{\langle z, a_4\rangle, \langle w, c_2\rangle, \langle y_i, e_1\rangle\}$, $\{\langle z, a_3\rangle, \langle w, c_2\rangle, \langle y_i, e_1\rangle\}$. We assume that we have forced $v, w < z$ using the gadget described in point (1). The tuples $t = \{\langle v, b_2\rangle, \langle x_i, d_1\rangle\}$, $u = \{\langle w, c_2\rangle, \langle y_i, e_1\rangle\}$ then form a DGA broken triangle on assignments $a_3, a_4 \in \mathcal{D}(z)$ if $x_i, y_i > z$. If either $x_i < z$ or $y_i < z$ then there is no DGA broken triangle; for example, if $x_i < z$, then we longer have Good(I_{CSP},$t^{<z} \cup \{\langle z, a_3\rangle\}$) since $t^{<z} \cup \{\langle z, a_3\rangle$ is precisely the nogood $\{\langle z, a_3\rangle, \langle v, b_2\rangle, \langle x_i, d_1\rangle\}$. Thus this gadget forces $y_i < z \vee x_i < z$ in a DGABTP ordering.

3. The gadget to force $y_i < z$ or $y_j < z$ or $y_k < z$ in a DGABTP ordering consists of values $a_5, a_6 \in \mathcal{D}(z)$, $b_3 \in \mathcal{D}(v)$, $c_3 \in \mathcal{D}(w)$, $e_2 \in \mathcal{D}(y_i)$, $e_3 \in \mathcal{D}(y_j)$, $e_4 \in \mathcal{D}(y_k)$ and five nogoods, namely $\{\langle z, a_6\rangle, \langle v, b_3\rangle, \langle y_i, e_2\rangle, \langle y_j, e_3\rangle, \langle y_k, e_4\rangle\}$, $\{\langle z, a_5\rangle, \langle w, c_3\rangle\}$, $\{\langle z, a_5\rangle, \langle y_i, e_2\rangle\}$, $\{\langle z, a_5\rangle, \langle y_j, e_3\rangle\}$, $\{\langle z, a_5\rangle, \langle y_k, e_4\rangle\}$. The tuples $t = \{\langle v, b_3\rangle, \langle y_i, e_2\rangle, \langle y_j, e_3\rangle, \langle y_k, e_4\rangle\}$, $u = \{\langle w, c_3\rangle\}$ form a DGA broken triangle on $a_5, a_6 \in \mathcal{D}(a)$ if $y_i, y_j, y_k > z$. If $y_i < z$ or $y_j < z$ or $y_k < z$, then there is no DGA broken triangle; for example, if $y_i < z$, then we longer have Good(I_{CSP},$t^{<z} \cup \{\langle z, a_5\rangle\}$) since $\{\langle z, a_5\rangle, \langle y_i, e_2\rangle\}$ is a nogood. Thus this gadget forces $y_i < z$ or $y_j < z$ or $y_k < z$ in a DGABTP ordering.

The above gadgets allow us to code I_{3SAT} as the problem of testing the existence of a DGABTP ordering in the corresponding instance I_{CSP}. To complete the proof it suffices to observe that this reduction is clearly polynomial. □ □

Our proof of Theorem 16 used large domains. The question still remains whether it is possible to detect in polynomial time whether a DGABTP variable ordering exists in the case of domains of bounded size, and in particular in the important case of SAT.

6 Conclusion

This paper described a novel reduction operation for binary CSP, called BTP-merging, which is strictly stronger than neighbourhood substitution. Experimental trials have shown that in several benchmark-domains applying BTP-merging until convergence can significantly reduce the total number of variable-value assignments. We gave a general-arity version of BTP-merging and demonstrated a theoretical link with resolution in SAT. From a theoretical point of view, we then went on to define a general-arity version of the tractable class defined by the broken-triangle property for a known variable ordering. Further research is required to find optimal algorithms for BTP-merging and to investigate the tractability of applying BTP-merging in instances containing global constraints.

References

[1] David A. Cohen, Martin C. Cooper, Páidí Creed, Dániel Marx, and András Z. Salamon. The tractability of CSP classes defined by forbidden patterns. *Journal of Artificial Intelligence Research*, 45, 47–78, 2012.

[2] David A. Cohen, Martin C. Cooper, Guillaume Escamocher and Stanislav Živný, Variable elimination in binary CSP via forbidden patterns, *Proceedings of IJCAI*, 517–523, 2013.

[3] Martin C. Cooper. Fundamental properties of neighbourhood substitution in constraint satisfaction problems. *Artificial Intelligence*, 90(1-2), 1–24, 1997.

[4] Martin C. Cooper and Guillaume Escamocher. A dichotomy for 2-constraint forbidden CSP patterns. In *Proceedings of AAAI*, 464–470, 2012.

[5] Martin C. Cooper, Peter G. Jeavons, and András Z. Salamon. Generalizing constraint satisfaction on trees: Hybrid tractability and variable elimination. *Artificial Intelligence*, 174(9-10), 570–584, 2010.

[6] Martin C. Cooper and Stanislav Živný, Tractable Triangles and Cross-Free Convexity in Discrete Optimisation, *Journal of Artificial Intelligence Research*, 44, 455–490, 2012.

[7] Niklas Eén and Armin Biere, Effective Preprocessing in SAT Through Variable and Clause Elimination, In *Proceedings of SAT*, 61–75, 2005.

[8] Achref El Mouelhi, Philippe Jégou and Cyril Terrioux. A Hybrid Tractable Class for Non-Binary CSPs. In *Proceedings of ICTAI*, 947–954, 2013.

[9] Philippe Jégou, Decomposition of Domains Based on the Micro-Structure of Finite Constraint-Satisfaction Problems In *Proceedings of AAAI*, 731–736, 1993.

[10] Eugene C. Freuder. Eliminating interchangeable values in constraint satisfaction problems. In *Proceedings of AAAI*, 227–233, 1991.

[11] Chavalit Likitvivatanavong and Roland H.C. Yap. Eliminating redundancy in csps through merging and subsumption of domain values. *ACM SIGAPP Applied Computing Review*, 13(2), 2013.

[12] András Z. Salamon and Peter G. Jeavons. Perfect Constraints Are Tractable. In *Proceedings of CP*, *LNCS* 5202, 524–528, 2008.