On the Decomposition of non-Binary Constraint into Equivalent Binary Constraints

Achref El Mouelhi
Aix-Marseille Université, LSIS UMR 7296
13397 Marseille Cedex 20 - France
achref.elmouelhi@lsis.org

***Abstract*—Considerable research effort has been focused on the translation of non-binary CSP into an equivalent binary CSP. Most of this work has been devoted to studying the binary encoding of non-binary CSP. Three encodings have been proposed, namely dual encoding, hidden variable encoding and double encoding. Unfortunately, such encodings do not allow to use some properties and interesting results defined only for the binary case. Another approach consists in the transformation of each non-binary constraint into a set of binary constraints: the CSP obtained is called primal. Unfortunately, this transformation does not preserve satisfiability.**

In this paper, we propose some conditions under which a non-binary constraint can be decomposed into a set of binary constraints while preserving satisfiability. An experimental study proves that our approach is not artificial since some ternary benchmarks can be transformed into equivalent binary instances and effectively solved by MAC.

I. Introduction

Many real-world problems can be expressed as non-binary constraint satisfaction problems [1] (**CSP** for short). It is well known that a non-binary instance can be translated in polynomial-time into a binary instance. This conversion permits us to use nice properties defined exclusively for binary instance. Moreover, from a solving viewpoint, much more is known about solving binary instances, more tractable classes are defined, more mergings and substitutions can be applied, more useful heuristics and algorithms are known, etc (see [2]).

To translate a CSP instance, there are mainly two general approaches: either by using a binary encoding such as dual encoding [3], hidden variable encoding [4] or double encoding [5], or else by converting (we also say decomposing) each non-binary constraint into a set of binary constraints [3]. The first approach consists in defining new binary constraints without converting the original constraints. It uses the binary encodings, based on some known graphical representation of non-binary instance to obtain an equivalent binary instance without decomposing any constraint (or its associated relation). Unfortunately, none of them can explicitly use certain interesting properties, such as substitution and interchangeability [6] or others on tractability and consistency defined exclusively for binary instances, and this is because of the non-conversion of non-binary constraints. Indeed, enforcing some consistency [7] or proving the tractability [8] of non-binary instances is often NP-hard. The second approach aims to decompose all non-binary constraints into a set of binary constraints. Unfortunately, the instance obtained after decomposing its constraints, called primal, is not equivalent to the initial.

Recently, some research has shown the interest of using the binary encoding from a graphical viewpoint [9]. Indeed, the microstructure[1] of a non-binary instance requires logically the use of hypergraphs while binary instance may be represented by simple graphs. We also know that the literature of Graph Theory is more developed than that of Hypergraph Theory. In addition, several properties, like acyclicity and degree, are more simple to be checked on graphs than hypergraphs. On top of that, other works have proved that extending some tractable classes to non-binary CSP can be efficiently realized thanks to the microstructure based on binary encodings [11]. This taskremains difficult to achieve according to the definition of Cohen [12], which is based on the hypergraph complement and which does not refer to binary encodings and it has not been used beyond the binary case.

In this paper, we describe a theoretical study on the decomposition of non-binary constraint into a set of equivalent binary constraints. So we will identify conditions on the constraint relations in order to guarantee equivalence between initial and modified problems. Firstly, we will refer to Relational Database Theory to describe the first value combinations which allows or forbids the decomposition of a constraint into a set of binary constraints. For the ternary case (a constraint having arity three), we will show that such a constraint can be decomposed into two binary constraints. After that, we propose a second condition which allows, if it holds, to decompose a non-binary constraint into a set of binary constraints. For the ternary case, such a constraint can be decomposed into three constraints.

In the next section, we will recall some definitions and notation which will be used throughout this paper. In sections three and four, we introduce two rules to decompose a non-binary constraint into a set of binary constraints (or not binary constraints but of arity less than that of the original constraint) without loss of satisfiability. In addition to some theoretical results, we check the applicability of our study in practice on a set of benchmarks which are classically used to compare solvers. Moreover, we show that our two rules are incomparable. Before concluding, we establish the link

[1]The micro-structure [10] of a binary instance I, denoted $\mu(I)$, is the graph whose the vertices are the (variable, value) pairs and the edges represent the compatibilities between the vertices.

between our rules and some previous works notably on global constraints.

II. Background

Constraint Satisfaction Problems provide an important tool to express and solve many real problems in Artificial Intelligence and Operational Research. Formally, a CSP instance can be defined as below:

Definition 1 (CSP instance): A CSP instance is a triple $I = (V, D, C)$, where $V = \{V_1, ..., V_n\}$ is a set of n **variables**, $D = \{D_1, ..., D_n\}$ is a set of finite **domains** containing at most d **values**, one for each variable and $C = \{C_1, ..., C_e\}$ is a set of e **constraints**. Each constraint C_i is a couple $(S(C_i), R(C_i))$ where:

- $S(C_i) = \{V_{i_1}, ..., V_{i_{a_i}}\} \subseteq V$, is the **constraint scope**,
- $R(C_i) \subseteq D_{i_1} \times ... \times D_{i_{a_i}}$, is the **constraint relation** which allows r tuples.

We assume that each variable is at least in the scope of one constraint and there is no two constraints with the same scope. $|S(C_i)|$ is the *arity* of the constraint C_i (i.e. the number of variables in the constraint scope) and it will be denoted a_i. An instance is said to be *binary* if the arity of each constraint is two (in this case, we denote C_{ij} the constraint with $S(C_{ij}) = \{V_i, V_j\}$), otherwise it is *non-binary* (*n-ary* or *of arbitrary arity*). An instance is *ternary* instance when the arity of each constraint is less than or equal to three.

We continue with the following notations which are necessary for the rest.

Notation 1 (tuple and relation projection): Given a constraint C_i, a tuple $t \in R(C_i)$ and a set of variables $\{V_{i_1}, ..., V_{i_k}\} \subseteq S(C_i)$: $t[\{V_{i_1}, ..., V_{i_k}\}] = (v_j \in t \mid V_j \in \{V_{i_1}, ..., V_{i_k}\})$ is the **projection** of the tuple t onto $\{V_{i_1}, ..., V_{i_k}\}$. $R(C_i)[\{V_{i_1}, ..., V_{i_k}\}] = \{t[\{V_{i_1}, ..., V_{i_k}\}] \mid t \in R(C_i)\}$ is the projection of $R(C_i)$ on $\{V_{i_1}, ..., V_{i_k}\}$.

Notation 2 (constraint restriction): Given a constraint C_i, $C_i[\{V_{i_1}, ..., V_{i_k}\}]$ is the restriction of C_i on $\{V_{i_1}, ..., V_{i_k}\}$ ($\subseteq S(C_i)$). If we denote $C_\ell = C_i[\{V_{i_1}, ..., V_{i_k}\}]$, then $S(C_\ell) = \{V_{i_1}, ..., V_{i_k}\}$ and $R(C_\ell) = R(C_i)[\{V_{i_1}, ..., V_{i_k}\}]$.

Given a CSP instance I, solving I consists of finding a solution, i.e. assigning one value per domain for each variable without violating any constraint. This problem is known to be NP-hard even for the binary case.

A CSP instance can be graphically represented by an hypergraph (a graph in the binary case) where the vertices are the variables and the hyperedges are the constraint scopes. In [3], the authors introduced a new method to convert a non-binary instance into a binary instance in order to represent it by a graph. Unfortunately, this transformation does not preserve satisfiability, i.e. the set of solutions of the initial instance is not necessary equal to the set of the transformed instance. Figure 1 shows that three constraint relations ($R(C_{ij})$,$R(C_{jk})$ and $R(C_{ik})$), obtained after translating the non-binary constraint C_ℓ, allow the assignment (v'_i, v_j, v_k) which is not allowed by $R(C_\ell)$. Thus, this translation does not preserve satisfiability. In the next sections, we propose two conditions to decompose a constraint while preserving satisfiability.

$R(C_\ell)$		
V_i	V_j	V_k
v_i	v_j	v_k
v'_i	v_j	v'_k
v'_i	v'_j	v_k

$R(C_{ij})$	$R(C_{jk})$	$R(C_{ik})$
V_i V_j	V_j V_k	V_i V_k
v_i v_j	v_j v_k	v_i v_k
v'_i v_j	v_j v'_k	v'_i v'_k
v'_i v'_j	v'_j v_k	v'_i v_k

Fig. 1. A non-binary constraint relation $R(C_\ell)$ translating into three constraint relations ($R(C_{ij})$, $R(C_{jk})$ and $R(C_{ik})$).

III. Decomposition based on multivalued dependency

In this part, we will illustrate the first rule, based on *multivalued dependency*, for decomposing constraints while preserving satisfiability[2]. In Relational Database Theory, the concept of multivalued dependency was introduced by Fagin in [13] in order to eliminate some redundancies in relation which are allowed by Boyce-Codd normal form ([14]). If a given relation satisfies multivalued dependency, then it is decomposable into two of its projections without loss of information. Here, we will apply this rule on a non-binary instance in order to obtain a binary instance for the reasons given in the Introduction. First, we will start by the ternary case before generalizing the result to non-binary instances.

A. Decomposition of ternary constraints

We briefly redefine multivalued dependency using the CSP formalism. We only consider ternary instances.

Definition 2 (multivalued dependency [13]): Given a constraint C_ℓ with $S(C_\ell) = \{V_i, V_j, V_k\}$, we say that C_ℓ satisfies the multivalued dependency (MvD) (we also say V_i multidetermines V_j, V_k and we denote $V_i \twoheadrightarrow V_j, V_k$) if $\forall v_i \in D_i, \forall v_j, v'_j \in D_j$ ($v_j \neq v'_j$) and $\forall v_k, v'_k \in D_k$ ($v_k \neq v'_k$) such that

- $(v_i, v_j, v_k) \in R(C_\ell)$
- $(v_i, v'_j, v'_k) \in R(C_\ell)$

then,

- $(v_i, v'_j, v_k) \in R(C_\ell)$ and
- $(v_i, v_j, v'_k) \in R(C_\ell)$

A ternary instance $I = (V, D, C)$ satisfies the multivalued dependency if each ternary constraint $C_\ell \in C$ satisfies MvD.

We now are able to introduce the first rule of decomposition. This rule allows the decomposition of constraint C_ℓ with ($S(C_\ell) = \{V_i, V_j, V_k\}$) with respect to V_i into two binary constraints. The constraint relation of each one is equivalent to the projection of the ternary constraint on $\{V_i, V_j\}$ and $\{V_i, V_k\}$.

Theorem 1: Given a constraint C_ℓ with $S(C_\ell) = \{V_i, V_j, V_k\}$, if $V_i \twoheadrightarrow V_j, V_k$, then C_ℓ can be decomposed into two constraints C_{ij} and C_{ik} while preserving satisfiability.

Proof: (by contradiction) Let C_ℓ be a constraint of arbitrary arity with $S(C_\ell) = \{V_i, V_j, V_k\}$ such that $V_i \twoheadrightarrow V_j, V_k$. We

[2]A decomposition rule preserves satisfiability if each assignment $\mathcal{A}$ of values to variables in the scope of original constraint is consistent if and only if $\mathcal{A}$ does not violate any constraint obtained after decomposition.

will prove by contradiction that MvD preserves satisfiability. Suppose that there is an assignment $\mathcal{A}$ that does not violate C_{ij} and C_{ik} but does not satisfy the original constraint C_ℓ. If we denote $\mathcal{A}$ as (v_i, v_j, v_k), we can obtain the following relations:

- $(v_i, v_j) \in R(C_{ij})$ (i),
- $(v_i, v_k) \in R(C_{ik})$ (ii) but
- $(v_i, v_j, v_k) \notin R(C_\ell)$ (iii).

In this case,

- (i) $\Rightarrow \exists t'_\ell \in R(C_\ell)$ such that $t'_\ell[\{V_i, V_j\}] = (v_i, v_j)$.
- (ii) $\Rightarrow \exists t''_\ell \in R(C_\ell)$ such that $t''_\ell[\{V_i, V_k\}] = (v_i, v_k)$.

So, there is a $v'_j \in D_j$, $v'_k \in D_k$ with $v'_j \neq v_j$, $v'_k \neq v_k$, $t'_\ell[\{V_k\}] = v'_k$ and $t''_\ell[\{V_j\}] = v'_j$. In this case, $t'_\ell = (v_i, v_j, v'_k)$ and $t''_\ell = (v_i, v'_j, v_k)$. But we supposed that $V_i \twoheadrightarrow V_j, V_k$ so we must have $(v_i, v_j, v_k) \in R(C_\ell)$ and $(v_i, v'_j, v'_k) \in R(C_\ell)$ (by definition of MvD), which contradicts our assumption (iii). Thus, this rule preserves satisfiability. □

In practice, this property can be checked in polynomial time and there is an algorithm (Algorithm 1) in $O(er^2)$ ($O(r^2)$ for each constraint) to verify if a given instance could be transformed in binary instance with respect to this rule. To decompose such a ternary constraint, we just need $O(r)$ to achieve it ($O(er)$ to decompose all constraints).

Algorithm 1: check whether an instance is MvD

```
function CHECK_DECOMPOSITION(I = (V, D, C): CSP instance): Boolean
    foreach C_ℓ ∈ C with S(C_ℓ) = {V_i, V_j, V_k} do
        if (not Check_Constraint(C_ℓ,V_i, V_j, V_k)) and
    (not Check_Constraint(C_ℓ,V_j, V_i, V_k)) and (not
    Check_Constraint(C_ℓ,V_k, V_j, V_i)) then
            return false
    return true
end function
function CHECK_CONSTRAINT( C_ℓ: Constraint,
V_i, V_j, V_k: Variable): Boolean
    for t_ℓ, t'_ℓ ∈ R(C_ℓ) with t_ℓ[{V_i}] = t'_ℓ[{V_i}] do
        if ((t_ℓ[{V_j}] ≠ t'_ℓ[{V_j}]) and
    (t_ℓ[{V_k}] ≠ t'_ℓ[{V_k}])) then
            if (((t_ℓ[{V_i}], t'_ℓ[{V_j}], t_ℓ[{V_k}]) ∉ R(C_ℓ))
        or ((t_ℓ[{V_i}], t_ℓ[{V_j}], t'_ℓ[{V_k}]) ∉ R(C_ℓ))) then
                return false
    return true
end function
```

As a consequence, a new hidden [15] tractable class of bivalent[3] instances can be defined:

Property 1: The class of ternary bivalent CSP instances which satisfy the multivalued dependency is tractable.

Proof: By definition [16], to define a tractable class we need to determine two polynomial algorithms, one for recognition and the second for solving. As mentioned above, $O(er^2)$ is sufficient to check whether a given ternary instance satisfies multivalued dependency. For resolution, we firstly need $O(er)$ to decompose all ternary constraints and the obtained binary bivalent instance is tractable [17] ($O(n^3d^3)$ to enforce path consistency [18] which will imply a globally consistent instance [17]). □

We performed an experimental study on 701 ternary benchmarks of the CP competition[4] to show the applicability of our approach in practice. We obtained in total results for 581 instances. We must point out that our library does not cover global constraints and we set one hour for each instance to achieve the test. 72 ternary benchmarks were detected as being able to be converted into binary instances. The families of these benchmarks are: `pseudo/primeDimacs`, `pseudo/par`, `pseudo/garden` and `pseudo/aim`. We also note that all of them belong to the tractable class defined in Proposition 1. Furthermore, MAC [19] applied to these converted benchmarks (obtained after decomposition) was more efficient than MAC applied to the original version (decomposition time included)[5].

B. Decomposition of constraints with arbitrary arity

We now generalize this rule and we start intially with a first definition allowing the decomposition of a non-binary constraint C_ℓ into two constraints of arity less than a_ℓ. To do so, we need to introduce the following notation:

Notation 3: We denote by V_I a subset of a_I variables of V ($\{V_{i_1}, ..., V_{i_{a_I}}\} \mid \forall 1 \leq k \leq a_I, V_{i_k} \in V\}$). v_I denotes a tuple $(v_{i_1}, ..., v_{i_{a_I}})$ containing one value for each variable in V_I.

This notation is necessarily to extend the previous result to the general case.

Definition 3 (general multivalued dependency [13]): Given a constraint C_ℓ with arbitrary arity, we say that C_ℓ satisfies general multivalued dependency (GMvD) if there are three disjoint subsets $\{V_I, V_J, V_K\}$ with $S(C_\ell) = V_I \cup V_J \cup V_K\}$ such that $\forall v_I \in R(C_\ell)[V_I], \forall v_J, v'_J \in R(C_\ell)[V_J]$ and $\forall v_K, v'_K \in R(C_\ell)[V_K]$ if

- $(v_I, v_J, v_K) \in R(C_\ell)$
- $(v_I, v'_J, v'_K) \in R(C_\ell)$

Then,

- $(v_I, v'_J, v_K) \in R(C_\ell)$ and
- $(v_I, v_J, v'_K) \in R(C_\ell)$

A non-binary instance $I = (V, D, C)$ satisfies the general multivalued dependency if each non-binary constraint $C_\ell \in C$ satisfies GMvD.

Semantically, a constraint C_ℓ satisfies GMvD if there are two non-disjoint subsets of variables $X, Y \subsetneq S(C_\ell)$ such that $R(C_\ell)$ is the join of $R(C_\ell)[X]$ and $R(C_\ell)[Y]$. Clearly, satisfying GMvD allows the decompostion of the constraint into two constraints $C_\ell[X]$ and $C_\ell[Y]$ while preserving satisfiability.

[3] A CSP instance is said to be bivalent (respectively trivalent) if the size of all variable domains is less than or equal to two (resp. three).

[4] see http://www.cril.univ-artois.fr/CPAI08 for more details.

[5] Some of these results were verified with Abscon (for more details see http://www.cril.univ-artois.fr/∼lecoutre/software.html).

Property 2: Given a constraint C_ℓ with arbitrary arity, if there are three disjoint subsets V_I, V_J and V_K with $S(C_\ell) = V_I \cup V_J \cup V_K$ such that $V_I \twoheadrightarrow V_J, V_K$, then C_ℓ can be decomposed into two constraints $C_\ell[V_I \cup V_J]$ and $C_\ell[V_I \cup V_K]$ while preserving satisfiability.

Proof: (by contradiction) Let C_ℓ be a constraint of arbitrary arity which satisfies GMvD. We will prove by contradiction that GMvD preserves satisfiability. We suppose that there is an assignment $\mathcal{A}$ which does not violate any new constraint (obtained after decomposition) but does not satisfy the original constraint C_ℓ. As C_ℓ is GMvD, there are three disjoint subsets V_I, V_J and V_K such that $S(C_\ell) = V_I \cup V_J \cup V_K$ and $V_I \twoheadrightarrow V_J, V_K$. In this case, we can express $\mathcal{A}$ as (v_I, v_J, v_K). Since $\mathcal{A}$ violates C_ℓ, there are two tuples t_1 and t_2 in $R(C_\ell)$ such that

- $t_1[V_I] = v_I$, $t_1[V_J] = v_J$ and $t_1[V_K] = v'_K$,
- $t_2[V_I] = v_I$, $t_2[V_J] = v'_J$ and $t_2[V_K] = v_K$.

We can point out that v_K must be different than v'_K (the same is true for v_J and v'_J) otherwise $\mathcal{A}$ does not violate C_ℓ. In this case, we have

- $(v_I, v_J, v'_K) \in R(C_\ell)$
- $(v_I, v'_J, v_K) \in R(C_\ell)$

and by definition of GMvD, we necessarily have

- $(v_I, v_J, v_K) \in R(C_\ell)$ (1) and
- $(v_I, v'_J, v'_K) \in R(C_\ell)$ (2).

Finally (1) contradicts our assumption. Hence decomposition of constraints which satisfy GMvD preserves satisfiability. □

Example 1: Let C_ℓ be a constraint with arity 4 with associated relation $R(C_\ell)$. By considering

- $V_I = \{V_i, V_m\}$,
- $V_J = \{V_j\}$ and
- $V_K = \{V_k\}$,

C_ℓ satisfies GMvD, so it is decomposable into two constraints C'_ℓ and C''_ℓ with arity 3 ($S(C'_\ell) = \{V_i \cup V_m \cup V_j\}$ and $S(C''_\ell) = \{V_i \cup V_m \cup V_k\}$). Tables below represent the relations $R(C_\ell)$, $R(C'_\ell)$ and $R(C''_\ell)$.

$R(C_\ell)$			
V_i	V_m	V_j	V_k
v_i	v_m	v_j	v'_k
v_i	v_m	v'_j	v_k
v_i	v_m	v'_j	v'_k
v_i	v_m	v_j	v_k
v'_i	v'_m	v_j	v_k

$R(C'_\ell)$	$R(C''_\ell)$
V_i V_m V_j	V_i V_m V_k
v_i v_m v_j	v_i v_m v_k
v_i v_m v'_j	v_i v_m v'_k
v'_i v'_m v_j	v'_i v'_m v_k

One can easily observe that the new constraints obtained after decomposition will have an arity less than that of the original constraint.

For ternary instances, when a constraint satisfies the multivalued dependency we deduce that it is decomposable into two binary constraints. Similarly here, we will show that when a constraint C_ℓ (with $S(C_\ell) = V_I \cup V_J \cup V_K$) satisfies the decremental multivalued dependency, then it decomposable into a set of binary constraints.

Definition 4 (decremental multivalued dependency): Given a constraint C_ℓ with $a_\ell \geq 3$, we say that C_ℓ satisfies decremental multivalued dependency (DMvD) if

1) there are three disjoint subsets V_I, V_J and V_K such that $S(C_\ell) = V_I \cup V_J \cup V_K$ and C_ℓ satisfies the GMvD $V_I \twoheadrightarrow V_J, V_K$
2) if $a_I + a_J \geq 3$ then $C_\ell[V_I \cup V_J]$ satisfies DMvD
3) if $a_I + a_K \geq 3$ then $C_\ell[V_I \cup V_K]$ satisfies DMvD.

A non-binary instance $I = (V, D, C)$ satisfies the decremental multivalued dependency if each non-binary constraint $C_\ell \in C$ is DMvD.

We have to show that being DMvD allows a constraint to be replaced by a set of equivalent binary constraints. As mentioned in the previous definition, the constraint C_ℓ will be decomposed with respect to Proposition 2. In other words, we start by decomposing C_ℓ into two constraints and we will recursively decompose the obtained constraints until obtaining binary constraints.

Theorem 2: if a constraint C_ℓ is DMvD, then it can be decomposed into a set of binary constraints while preserving satisfiability.

Proof: (by induction) It is clear that DMvD is defined recursively with respect to GMvD. We will prove by induction for all a_ℓ that if constraint C_ℓ satisfies DMvD, then it can be decomposed into a set of binary constraints while preserving satisfiability.

- **Base case:** for $a_\ell = 3$, we have shown in Theorem 1 that decomposing MvD constraints preserves satisfiability.
- **Induction step:** we suppose that the satisfiability is preserved for $a_\ell \leq p-1$ and we will prove that it remains so when $a_\ell = p$. As C_ℓ is DMvD, it is decomposable into two constraints C'_ℓ and C''_ℓ while preserving satisfiability, and each one has an arity less than p. We supposed that when a constraint is DMvD and its arity is less than p, it is recursively decomposable into a set of binary constraints while preserving satisfiability.

We conclude that DMvD preserves satisfiability whatever the arity of the constraint. □

The constraint C_ℓ of Example 1 is DMvD because it is GMvD and both $C_\ell[V_I \cup V_J]$ and $C_\ell[V_I \cup V_K]$ have arity three and they satisfy DMvD.

We now show that this property can be checked in polynomial-time for a given partition of variables.

Property 3: Given a constraint C_ℓ and three disjoint variable subsets V_I, V_J and V_K such that $S(C_\ell) = V_I \cup V_J \cup V_K$, checking whether $V_I \twoheadrightarrow V_J, V_K$ can be achieved in polynomial-time.

Proof: When we have a three disjoint subsets of variables, we just need $O(a_\ell . r^2 . log(r))$ to check whether a constraint satisfies DMvD:

- r^2 to list all the tuples,
- $log(r)$ to check if each tuple belongs to the constraint relation
- a_ℓ or more precisely $a_\ell - 2$ to apply recursively the test on the new constraints

The complexity of decomposition is $O(r.a_\ell^2)$:

- r to list all the tuples of C_ℓ,
- a_ℓ^2 or more precisely $a_\ell(a_\ell - 1)/2$ which equals the maximal number of binary constraints.

Thus, checking whether a constraint satisfies DMvD and then decomposing it can be achieved in polynomial-time. □

We also can define a strong form of multivalued dependency, as follows:

Definition 5 (strong multivalued dependency): Given a constraint C_ℓ with $a_\ell \geq 3$, we say that C_ℓ satisfies strong multivalued dependency (SMvD) if for all three disjoint subsets V_I, V_J and V_K such that $S(C_\ell) = V_I \cup V_J \cup V_K$, C_ℓ satisfies DMvD. A non-binary instance $I = (V, D, C)$ satisfies the strong multivalued dependency if each non-binary constraint $C_\ell \in C$ is SMvD.

Obviously we can establish the link between the three forms of multivalued dependency.

Corollary 1: If a constraint C_ℓ satisfies SMvD, then it satisfies GMvD and DMvD.

Before concluding this section, make some observations. When a constraint is not decomposable with respect to a subset V_I we cannot deduce anything about the decomposition with respect to V_J or V_K. We also note that if a given constraint is decomposable with respect to V_I and V_J, we cannot deduce anything about decomposition with respect to V_K.

IV. Decomposition based on interdependency

We now introduce the second rule, called *interdependency*, which also guarantees decomposition without loss of satisfiability. This rule will permit us to decompose a ternary constraint into three binary constraints.

A. Decomposition of ternary constraints

We first define interdependency and then briefly review the decomposition of constraints based on this definition.

Definition 6 (interdependency): Given a constraint C_ℓ with $S(C_\ell) = \{V_i, V_j, V_k\}$, we say that C_ℓ satisfies interdependency (ID) (we also say that V_i, V_j and V_k are interdependent) if $\forall v_i \in D_i, \forall v_j, v'_j \in D_j$ $(v_j \neq v'_j)$ and $\forall v_k, v'_k \in D_k$ $(v_k \neq v'_k)$ such that

- $(v_i, v_j, v_k) \in R(C_\ell)$
- $(v_i, v'_j, v'_k) \in R(C_\ell)$

then, there is no $v'_i \in D_i \setminus \{v_i\}$ such that

- $(v'_i, v'_j, v_k) \in R(C_\ell)$ or
- $(v'_i, v_j, v'_k) \in R(C_\ell)$

A ternary instance $I = (V, D, C)$ satisfies interdependency if each ternary constraint $C_\ell \in C$ satisfies ID.

In other words, a constraint C_ℓ with $S(C_\ell) = \{V_i, V_j, V_k\}$ satisfies ID if all combinations of two couples of values (v_j, v_k) and (v'_j, v'_k) which are compatible with $v_i \in D_i$ depend only on v_i, i.e. (v'_j, v_k) and (v_j, v'_k) must only be compatible with v_i. Similarly for any two couples (v_i, v_k) and (v'_i, v'_k) (respectively (v_i, v_j) and (v'_i, v'_j)) which are compatible with such a v_j (resp. v_k).

Theorem 3 tells us the relationship between interdependency and decomposition.

Theorem 3: Given a constraint C_ℓ with $S(C_\ell) = \{V_i, V_j, V_k\}$, if V_i, V_j and V_k are interdependent then C_ℓ can be decomposed into three binary constraints C_{ij}, C_{ik} and C_{jk} while preserving satisfiability.

Proof: (by contradiction) Let C_ℓ be a constraint with $S(C_\ell) = \{V_i, V_j, V_k\}$ such that V_i, V_j and V_k are interdependent. We will prove by contradiction that ID preserves satisfiability. Suppose that there is an assignment $\mathcal{A}$ which does not violate C_{ij}, C_{jk} and C_{ik} but does not satisfy the original constraint C_ℓ. If we denote $\mathcal{A}$ as (v_i, v_j, v_k), we can obtain the following relations:

- $(v_i, v_j) \in R(C_{ij})$ (a),
- $(v_i, v_k) \in R(C_{ik})$ (b) and
- $(v_j, v_k) \in R(C_{jk})$ (c) but
- $(v_i, v_j, v_k) \notin R(C_\ell)$ (d).

For this,

- (a) $\Rightarrow \exists t_\ell \in R(C_\ell)$ such that $t_\ell[\{V_i, V_j\}] = (v_i, v_j)$.
- (b) $\Rightarrow \exists t'_\ell \in R(C_\ell)$ such that $t'_\ell[\{V_i, V_k\}] = (v_i, v_k)$.
- (c) $\Rightarrow \exists t''_\ell \in R(C_\ell)$ such that $t''_\ell[\{V_j, V_k\}] = (v_j, v_k)$.

So, there are a $v'_i \in D_i$, $v'_j \in D_j$ and $v'_k \in D_k$ with $v'_i \neq v_i$ $v'_j \neq v_j$, $v'_k \neq v_k$ such that $t_\ell[\{V_k\}] = v'_k$, $t'_\ell[\{V_j\}] = v'_j$ and $t''_\ell[\{V_i\}] = v'_i$. In this case,

- $t_\ell = (v_i, v_j, v'_k)$,
- $t'_\ell = (v_i, v'_j, v_k)$ and
- $t''_\ell = (v'_i, v_j, v_k)$.

But we supposed (by definition of ID) that V_i, V_j and V_k are interdependent so we must have $(v'_i, v_j, v_k) \notin R(C_\ell)$ and $(v'_i, v'_j, v'_k) \notin R(C_\ell)$ which is impossible. Thus, this rule preserves satisfiability. □

Contrary to multivalued dependency, interdependency is symmetric, i.e. if a given constraint C_ℓ with $S(C_\ell) = \{V_i, V_j, V_k\}$ satisfies it with respect to V_i, then it also does with respect to V_j and V_k. Thus, we are not required to check the decomposability of a given constraint with respect to each variable in the scope, we have only to check if it is decomposable with respect to a single variable.

In practice, we can check whether the variables in the scope of a given constraint are interdependent in $O(dr^2)$ and if the property holds we need a time complexity $O(r)$ to decompose this constraint. This leads us to the following proposition.

Property 4: The class of ternary bivalent instances which satisfy interdependency is tractable.

Proof: Similar to the proof of Proposition 1. □

From an experimental viewpoint, all benchmarks which are decomposable based on multivalued dependency are also decomposable based on interdependence in addition to some other instances from families `primes-20` and `primes-30`. All these instances satisfy either the tractable class described

in Proposition 4 or BTP[6] [20] or DBTP[7] [11]. Moreover, the obtained binary instances are effectively resolved by MAC.

For some other instances belonging to `dubois`, `pret` and `primes`, all the ternary constraints except one were successfully converted into binary constraints. Overall, we managed to convert 86 benchmarks of all 581 considered ternary benchmarks (which represents nearly 14% of the total).

B. Decomposition of constraints with arbitrary arity

In this section, we will generalize the definition of interdependency and obviously the decomposition concept related to this definition.

Definition 7 (general interdependency): Given a constraint C_ℓ with arbitrary arity, we say that C_ℓ satisfies general interdependency (GID) if there are three disjoint subsets $\{V_I, V_J, V_K\}$ with $S(C_\ell) = V_I \cup V_J \cup V_K$ and if $\forall v_I \in R(C_\ell)[V_I], \forall v_J, v'_J \in R(C_\ell)[V_J]$ and $\forall v_K, v'_K \in R(C_\ell)[V_K]$ such that

- $(v_I, v_J, v_K) \in R(C_\ell)$
- $(v_I, v'_J, v'_K) \in R(C_\ell)$

then, there is no v'_I such that

- $(v'_I, v'_J, v_K) \in R(C_\ell)$ or
- $(v'_I, v_J, v'_K) \in R(C_\ell)$

A non-binary instance $I = (V, D, C)$ satisfies general interdependency if each non-binary constraint $C_\ell \in C$ satisfies GID.

As for general multivalued dependency, a constraint C_ℓ whose scope can be divided into three disjoint interdependent subsets can be decomposed into three constraints whose arities are less than that of C_ℓ.

Theorem 4: Given a constraint C_ℓ, if there are three disjoint subsets of variables V_I, V_J and V_K such that $S(C_\ell) = V_I \cup V_J \cup V_K$ and V_I,V_J and V_K are interdependent, C_ℓ is decomposable into the constraints $C_\ell[V_I \cup V_J]$, $C_\ell[V_J \cup V_K]$ and $C_\ell[V_I \cup V_K]$ while preserving satisfiability.

Proof: (by contradiction) Let C_ℓ be a constraint of arbitrary arity which satisfies GID. We will prove by contradiction that GID preserves satisfiability. Suppose that there is an assignment $\mathcal{A}$ which does not violate any new constraint (obtained after decomposition) but does not satisfy the original constraint C_ℓ. As C_ℓ is GID, there are three disjoint subsets V_I, V_J and V_K with $S(C_\ell) = V_I \cup V_J \cup V_K$ which are interdependent and in this case we can express $\mathcal{A}$ as (v_I, v_J, v_K). $\mathcal{A}$ violates C_ℓ, so there are three tuples t_1, t_2 and t_3 belonging to $R(C_\ell)$ such that

- $t_1[V_I] = v_I$, $t_1[V_J] = v_J$ and $t_1[V_K] = v'_K$ (1),
- $t_2[V_I] = v_I$, $t_2[V_J] = v'_J$ and $t_2[V_K] = v_K$ (2),
- $t_3[V_I] = v'_I$, $t_3[V_J] = v_J$ and $t_3[V_K] = v_K$ (3).

Note that v_I must be different to v'_I (similarly for v_J, v_K and v'_J, v'_K) otherwise $\mathcal{A}$ does not violate C_ℓ. In this case, we have

- $(v_I, v_J, v'_K) \in R(C_\ell)$
- $(v_I, v'_J, v_K) \in R(C_\ell)$

and by definition of GID, there is no $v'_I \neq v_I$ such that

- $(v'_I, v_J, v_K) \in R(C_\ell)$ (a)

Finally (a) is violated by (3). Hence decomposition of constraints which satisfy GID preserves satisfiability. □

Example 2: Let C_ℓ be a constraint of arity four and $R(C_\ell)$ its associated relation. By considering

- $V_I = \{V_i\}$, $V_J = \{V_j\}$ and
- $V_K = \{V_m, V_k\}$,

C_ℓ satisfies GID, so it is decomposable into three constraints C'_ℓ, C''_ℓ et C'''_ℓ. Tables below represent their associated relations $R(C_\ell)$, $R(C'_\ell)$, $R(C''_\ell)$ et $R(C'''_\ell)$.

$R(C_\ell)$			
V_i	V_j	V_m	V_k
v_i	v_j	v_m	v_k
v_i	v'_j	v'_m	v'_k
v'_i	v'_j	v'_m	v_k

$R(C_{ij})$	$R(C'_\ell)$	$R(C''_\ell)$
$V_i\ V_j$	$V_i\ V_m\ V_k$	$V_j\ V_m\ V_k$
$v_i\ v_j$	$v_i\ v_m\ v_k$	$v_j\ v_m\ v_k$
$v_i\ v'_j$	$v_i\ v'_m\ v'_k$	$v'_j\ v'_m\ v'_k$
$v'_i\ v'_j$	$v'_i\ v'_m\ v_k$	$v'_j\ v'_m\ v_k$

Unfortunately GID does not guarantee the decomposition of a non-binary constraint into a set of binary constraints. For this, we introduce Decremental Interdependency.

Definition 8 (decremental interdependency): Given a constraint C_ℓ with $a_\ell \geq 3$, we say that C_ℓ satisfies decremental interdependency (DID) if

1) there are three disjoint subsets V_I, V_J and V_K such that $S(C_\ell) = V_I \cup V_J \cup V_K$ and C_ℓ satisfies GID
2) if $a_I + a_J \geq 3$ then $C_\ell[V_I \cup V_J]$ satisfies DID
3) if $a_I + a_K \geq 3$ then $C_\ell[V_I \cup V_K]$ satisfies DID
4) if $a_J + a_K \geq 3$ then $C_\ell[V_J \cup V_K]$ satisfies DID.

A non-binary instance $I = (V, D, C)$ satisfies decremental interdependency if each non-binary constraint $C_\ell \in C$ is DID.

We now prove that being DID is sufficient to decompose a non-binary constraint into a set of equivalent binary constraints. As mentioned in the above definition, the constraint C_ℓ will be decomposed according to Proposition 4. In other words, we start by decomposing C_ℓ into three constraints and we recursively decompose the new constraints until we obtain binary constraints.

Theorem 5: if a constraint C_ℓ is DID, then it can be decomposed into a set of binary constraints while preserving satisfiability.

Proof: (by induction) It is clear that DID is defined recursively with respect to GID. We will prove by induction for all a_ℓ if a constraint C_ℓ satisfies DID, it can be decomposed into a set of binary constraints while preserving satisfiability.

- **Base case:** for $a_\ell = 3$, we have shown in Theorem 3 that decomposing ID constraints preserves satisfiability.
- **Induction step:** we suppose that satisfiability is preserved when $a_\ell \leq p - 1$ and we prove that it remains so when $a_\ell = p$. As C_ℓ is DID, it is decomposable into three

[6]A binary instance $I = (V, D, C)$ satisfies the *Broken Triangle Property* (BTP) with respect to the variable ordering $<$ if, for all triples of variables (V_i, V_j, x_k) such that $V_i < V_j < V_k$, such that $(v_i, v_j) \in R(C_{ij})$, $(v_i, v_k) \in R(C_{ik})$ and $(v_j, v'_k) \in R(C_{jk})$, then either $(v_i, v'_k) \in R(C_{ik})$ or $(v_j, v_k) \in R(C_{jk})$. Let BTP be the set of the instances for which BTP holds with respect to some variable ordering.

[7]An instance I with arbitrary arity satisfies the *Dual Broken Triangle Property* (DBTP) with respect to the constraint ordering $\prec$ iff the dual of I satisfies BTP with respect to $\prec$.

constraints $C_\ell[V_I \cup V_J]$, $C_\ell[V_J \cup V_K]$ and $C_\ell[V_I \cup V_K]$ while preserving satisfiability, and each one has arity less than p. We supposed that when a constraint is DID and its arity is less than p, it is recursively decomposable into a set of binary constraints while preserving satisfiability.

We conclude that DID preserves satisfiability whatever the arity of the constraint. □

Returning to Example 2, the constraint C_ℓ satisfies DID because it satisfies GID and C'_ℓ and C''_ℓ are of arity three and satisfy DID.

As for testing the above rule, given a partition of the variables in the constraint scope $S(C_\ell)$ into three disjoint subsets, checking whether this partition makes the three subsets V_I, V_J and V_K decremental interdependent can be realized in polynomial time. Decomposing C_ℓ can also be done in polynomial-time.

Property 5: Given a constraint C_ℓ and three disjoint subsets V_I, V_J and V_K such that $S(C_\ell) = V_I \cup V_J \cup V_K$, checking whether V_I, V_J and V_K are decremental interdependent can be achieved in polynomial-time.

Now consider the strong form of interdependency.

Definition 9 (strong interdependency): Given a constraint C_ℓ with $a_\ell \geq 3$, we say that C_ℓ satisfies strong interdependency (SID) if for all three disjoint subsets V_I, V_J and V_K such that $S(C_\ell) = V_I \cup V_J \cup V_K$, we have C_ℓ satisfies DID. A non-binary instance $I = (V, D, C)$ satisfies strong interdependency if each non-binary constraint $C_\ell \in C$ is SID.

In the rest of this section, we want to define when a constraint is decomposable whatever the three disjoint subsets V_I, V_J and V_K.

Definition 10 (perfect decomposition): Given a constraint C_ℓ, C_ℓ is *perfectly decomposable* if for all three disjoint subsets V_I, V_J and V_K such that $S(C_\ell) = V_I \cup V_J \cup V_K$, V_I, V_J and V_K are general interdependent.

We show that any perfectly decomposable constraint has an important property which will be used later.

Lemma 1: Given a constraint C_ℓ, all three disjoint subsets V_I, V_J and V_K such that $S(C_\ell) = V_I \cup V_J \cup V_K$ are interdependent, if and only if, for all $V_i \in V_I$, $V_j \in V_J$ and $V_k \in V_K$, V_i, V_j and V_k are interdependent.

Proof: ($\Rightarrow$) Suppose for a given constraint C_ℓ such that all three disjoint subsets V_I, V_J and V_K such that $S(C_\ell) = V_I \cup V_J \cup V_K$ and C_ℓ satisfies GID but for some $V_i \in V_I$, $V_j \in V_J$ and $V_k \in V_K$, V_i, V_j and V_k are not interdependent. So there is a $v_i, v'_i \in D_i$, $v_j, v'_j \in D_j$ and $v_k, v'_k \in D_k$ such that

- $(v_i, v_j, v_k) \in R(C_\ell)[\{V_i, V_j, V_k\}]$
- $(v_i, v'_j, v'_k) \in R(C_\ell)[\{V_i, V_j, V_k\}]$ and
- $(v'_i, v'_j, v_k) \in R(C_\ell[\{V_i, V_j, V_k\}])$ (or $(v'_i, v_j, v'_k) \in R(C_\ell)[\{V_i, V_j, V_k\}]$).

In this case, it is imperative that v_i of the first tuple (v_i, v_j, v_k) belongs to a v_I and v_i of the second tuple (v_i, v'_j, v'_k) belongs to v'_I which is different from v_I, otherwise V_I, V_J and V_K are not general interdependent. For this, there are $v_m, v'_m \in D_m$ ($V_m \in V_I$) such that $v_I[\{V_i, V_m\}] = (v_i, v_m)$ and $v'_I[\{V_i, V_m\}] = (v'_i, v_m)$. If we consider another partition, i.e. $V_I = V_I - \{V_m\}$ and $V_J = V_J \cup \{V_m\}$, V_I, V_J and V_K are not general interdependent which is impossible because it contradicts our assumption.

($\Leftarrow$) Suppose for a given constraint C_ℓ that there are three disjoint subsets V_I, V_J and V_K (with $S(C_\ell) = V_I \cup V_J \cup V_K$) which are not general interdependent although V_i, V_j and V_k are interdependent for all $V_i, V_j, V_k \in S(C_\ell)$. Then, $\exists v_I, v'_I \in R(C_\ell)[V_I]$, $v_J, v'_J \in R(C_\ell)[V_J]$, v_K, $v'_K \in R(C_\ell)[V_K]$ in such a way that

- $(v_I, v_J, v_K) \in R(C_\ell)$,
- $(v_I, v'_J, v'_K) \in R(C_\ell)$ and
- $(v'_I, v'_J, v_K) \in R(C_\ell)$ (or $(v'_I, v_J, v'_K) \in R(C_\ell)$)

$v_I \neq v'_I \Longrightarrow \exists V_i \in V_I$ such that $v_I[\{V_i\}] \neq v'_I[\{V_i\}]$, we denote $v_i = v_I[\{V_i\}]$ and $v'_i = v'_I[\{V_i\}]$. Similarly, we obtain $v_j = v_J[\{V_j\}]$, $v'_j = v'_J[\{V_j\}]$ and $v_k = v_K[\{V_k\}]$ and $v'_k = v'_K[\{V_k\}]$. This implies that

- $(v_i, v_j, v_k) \in R(C_\ell)[\{V_i, V_j, V_k\}]$,
- $(v_i, v'_j, v'_k) \in R(C_\ell)[\{V_i, V_j, V_k\}]$ and
- $(v'_i, v'_j, v_k) \in R(C_\ell)[\{V_i, V_j, V_k\}]$.

Thus, V_i, V_j and V_k are not interdependent which contradicts our assumption. □

V. Comparison between the two rules

Here we prove that our two rules are different and there is implication between these two properties.

Property 6: Interdependency and multivalued dependency are incomparable.

Proof: To show that ID and MvD are incomparable, it suffices to illustrate an example of a constraint which satisfies these two rules and two others such that each one satisfies only one rule.

$R(C_\ell)$		
V_i	V_j	V_k
v_i	v_j	v_k
v_i	v'_j	v'_k
v'_i	v_j	v''_k
v''_i	v''_j	v_k

$R(C'_\ell)$		
V_i	V_j	V_k
v_i	v_j	v_k
v_i	v'_j	v'_k
v_i	v_j	v'_k
v_i	v'_j	v_k
v'_i	v'_j	v_k

$R(C''_\ell)$		
V_i	V_j	V_k
v_i	v_j	v_k
v_i	v'_j	v'_k
v_i	v_j	v'_k
v_i	v'_j	v_k

In the tables above, the constraint C_ℓ satisfies ID but not MvD ($V_i \not\twoheadrightarrow V_j, V_k$, $V_j \not\twoheadrightarrow V_i, V_k$ and $V_k \not\twoheadrightarrow V_i, V_j$), C'_ℓ satisfies MvD but not ID and C''_ℓ satisfies both MvD and ID. □

Even when a given constraint C_ℓ is decomposable by the multivalued dependency rule and with respect to all variables in the constraint scope, we cannot deduce anything about decomposing by the interdependency rule.

VI. What about global constraints?

On the one hand, we can see that the link with tractability mentioned in our paper can be compared to the notion of transformation used in [15] but this notion does not allow constraint decomposition. Previously, several works (like [21], [22]) have studied the decomposition of non-binary constraints and they are based on some other rules which sometimes

are inspired from Relational Data Base Theory. On the other hand, several works have studied decomposition methods and specially for *global constraints*. For reasons of space, we will treat only the case of the *All-Different constraint* [23]. As in most of the cases, we consider that variable domains are all the same and therefore $v_i = v_j = v_k$ and $v'_i = v'_j = v'_k$, etc. Previously, some works (like [24]) have proved that an All-Different constraint can be decomposed into a clique of binary ones. Here, we show that All-Different constraints are not decomposable neither by using the multivalued dependency rule nor by considering the interdependency rule except when all the variables in the scope are trivalent.

Theorem 6: All-Different constraints are not decomposable by considering the multivalued dependency rule.

Proof: Given a constraint C_ℓ with $S(C_\ell) = \{V_i, V_j, V_k\}$, if we have $(v_i, v'_j, v''_k) \in R(C_\ell)$ and $(v_i, v''_j, v'_k) \in R(C_\ell)$, we must have $(v_i, v'_j, v'_k) \in R(C_\ell)$ and $(v_i, v''_j, v''_k) \in R(C_\ell)$ in order that C_ℓ is decomposable. But we know that this is impossible because of the assignment of the same value to two different variables in the same tuple (by definition of the All-Different constraint). □

Theorem 7: Any ternary trivalent All-Different constraint is decomposable by considering the interdependency rule.

Proof: Any constraint C_ℓ with $S(C_\ell) = \{V_i, V_j, V_k\}$ is perfectly decomposable because if we have $(v_i, v'_j, v''_k) \in R(C_\ell)$ and $(v_i, v''_j, v'_k) \in R(C_\ell)$ then it is impossible that a value $v'''_i \in D_i$ exists such that $(v'''_i, v'_j, v'_k) \in R(C_\ell)$ and $(v'''_i, v''_j, v''_k) \in R(C_\ell)$ because $\mid D_i \mid = \mid D_j \mid = \mid D_k \mid = 3$, v'_j and v'_k are equal and they cannot appear in the same tuple (by definition of the All-Different constraint). □

Except this case, All-Different constraints does not satisfy interdependency. To prove it, we have just to consider a ternary All-Different constraint s.t. each variable in its scope have a domain of size 4. This constraint must allow $(0, 1, 2)$, $(0, 2, 3)$ and $(0, 1, 3)$ which makes it non-interdependent.

VII. Conclusion

In this paper, we have introduced some new rules for checking if a non-binary constraint is decomposable. The first rule, called multivalued dependency, is based on a notion introduced previously in Relational Database Theory by Fagin. The second, called interdependency, is introduced for the first time and defines a relation of dependency between the combination of values allowed by the constraint. For the ternary case, we have shown that both multivalued dependency and interdependency are interesting from a practical point of view since the number of converted instances is large and all such instances belong to known tractable classes such as BTP and DBTP. There are many perspectives for further research. The first is to check if our study could be used to decompose global constraints. It could also be interesting to define the case when a partition of the variables in the scope for which a given constraint is decomposable could be found in polynomial-time? Do these converted CSP instances belong to known tractable classes or do they define new tractable classes?

Acknowledgments

I would like to warmly thank Martin Cooper, Philippe Jégou and Cyril Terrioux for their useful comments and suggestions on the language of this paper.

References

[1] U. Montanari. Networks of Constraints: Fundamental Properties and Applications to Picture Processing. *Artificial Intelligence*, 7:95–132, 1974.

[2] Fahiem Bacchus and Peter van Beek. On the conversion between non-binary and binary constraint satisfaction problems. In *AAAI/IAAI*, pages 310–318, 1998.

[3] R. Dechter and J. Pearl. Network-based heuristics for constraint satisfaction problems. *Artificial Intelligence*, 34:1–38, 1987.

[4] Francesca Rossi, Charles J. Petrie, and Vasant Dhar. On the equivalence of constraint satisfaction problems. In *ECAI*, pages 550–556, 1990.

[5] K. Stergiou and T. Walsh. Encodings of Non-Binary Constraint Satisfaction Problems. In *AAAI*, pages 163–168, 1999.

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

[7] Yuanlin Zhang and Roland H. C. Yap. Arc consistency on *n*-ary monotonic and linear constraints. In *Principles and Practice of Constraint Programming - CP 2000, 6th International Conference*, pages 470–483, 2000.

[8] Martin C. Cooper, Achref El Mouelhi, Cyril Terrioux, and Bruno Zanuttini. On broken triangles. In *Principles and Practice of Constraint Programming - 20th International Conference, CP 2014*, pages 9–24, 2014.

[9] Achref El Mouelhi, Philippe Jégou, and Cyril Terrioux. Microstructures for csps with constraints of arbitrary arity. In *SARA*, 2013.

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

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

[12] David A. Cohen. A New Classs of Binary CSPs for which Arc-Constistency Is a Decision Procedure. In *CP*, pages 807–811, 2003.

[13] Ronald Fagin. Multivalued dependencies and a new normal form for relational databases. *ACM Trans. Database Syst.*, 2(3):262–278, 1977.

[14] Edgar Frank Codd. Further normalization of the data base relational model. *IBM Research Report, San Jose, California*, RJ909, 1971.

[15] Achref El Mouelhi, Philippe Jégou, and Cyril Terrioux. Hidden Tractable Classes. In *26th IEEE International Conference on Tools with Artificial Intelligence, ICTAI 2014, November 10-12, 2014*, pages 437–445, 2014.

[16] G. Gottlob, N. Leone, and F. Scarcello. A Comparison of Structural CSP Decomposition Methods. *Artificial Intelligence*, 124:343–282, 2000.

[17] R. Dechter. Constraint Networks. In *Encyclopedia of Artificial Intelligence*, volume 1, pages 276–285. John Wiley & Sons, Inc., second edition, 1992.

[18] C. Han and C. Lee. Comments on Mohr and Henderson's path consistency algorithm. *Artificial Intelligence*, 36:125–130, 1988.

[19] D. Sabin and E. Freuder. Contradicting Conventional Wisdom in Constraint Satisfaction. In *Proc. of ECAI*, pages 125–129, 1994.

[20] M. Cooper, Peter Jeavons, and Andras Salamon. Generalizing constraint satisfaction on trees: hybrid tractability and variable elimination. *Artificial Intelligence*, 174:570–584, 2010.

[21] M. Gyssens, P. Jeavons, and D. Cohen. Decomposing constraint satisfaction problems using database techniques. *Artificial Intelligence*, 66:57–89, 1994.

[22] Peter Jeavons, David A. Cohen, and Martin C. Cooper. Constraints, consistency and closure. *Artificial Intelligence*, 101(1-2):251–265, 1998.

[23] Jean-Louis Laurière. A language and a program for stating and solving combinatorial problems. *Artificial Intelligence.*, 10(1):29–127, 1978.

[24] Christian Bessiere, George Katsirelos, Nina Narodytska, Claude-Guy Quimper, and Toby Walsh. Decompositions of all different, global cardinality and related constraints. In *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, July 11-17, 2009*, pages 419–424, 2009.