*ECAI 2008*
*M. Ghallab et al. (Eds.)*
*IOS Press, 2008*
*© 2008 The authors and IOS Press. All rights reserved.*
*doi:10.3233/978-1-58603-891-5-907*

# Redundancy in CSPs

**Assef Chmeiss** and **Vincent Krawczyk** and **Lakhdar Sais** [1]

**Abstract.** In this paper, we propose a new technique to compute irredundant sub-sets of constraint networks. Since, checking redundancy is Co-NP Complete problem, we use different polynomial local consistency entailments for reducing the computational complexity. The obtained constraint network is irredundant *modulo* a given local consistency. Redundant constraints are eliminated from the original instance producing an equivalent one with respect to satisfiability. Eliminating redundancy might help the CSP solver to direct the search to the most constrained (irredundant) part of the network.

## 1 Constraint Satisfaction

Constraint-satisfaction problems (CSPs) involve the assignment of values to variables which are subject to a set of constraints. The modeling and solving phases are known to be heavily interconnected. Indeed, the efficiency of the solver depends on the way the problem instance is modeled. Until recently, these two phases are considered separately. Many improvements have been proposed for the solving side and many other approaches have been suggested to simplify the crucial modeling step [1, 3]. As there exists several ways to model the same problem, this means that the user is not safe from introducing redundancies in such modeling process. Also redundancies might result from an incorrect encoding or merging different parts from several sources. The obtained constraint network (CN), might contain parts that can be removed without losing the information it carries. However, several forms of redundancies can be characterized.

In this paper, we address constraint redundancies. A CSP is redundant if and only if some its constraints can be removed while preserving its set of models. As stated by Paolo Liberatore [4] in the context of propositional clausal formulae, the deletion of redundant constraints is clearly important for several reasons. First, removing redundant constraints can simplify the CN by reducing its size. A large amount of redundancies might obscure the real set of constraints (the irredundant part of the CN). In other cases, redundancy might indicate that some pieces of the CN are more important than the others. Consequently, depending on the application domain, redundancy might be either a positive or a negative concept.

Our main goal is to measure the relationships between constraints redundancies and the efficiency of CSP solvers. As a side effect, our approach can be seen as a possible technique that can be used to check the degree of redundancy of a given CN. On the current available CSP instances, our approach might give a nice way to approximate their irredundant part. However, checking constraint redundancy, meaning that deciding if a given constraint can be deduced from the remaining part of the CN is known to be Co-NP complete [4]. To deal with this main drawback, in this paper, different polynomial local consistency entailments are used for reducing the computational complexity. The obtained CN is irredundant *modulo* a given local consistency.

[1] Université Lille-Nord de France, CNRS UMR 8188, Artois, Rue Jean Souvraz, SP-18, F-62307 Lens, email:{chmeiss, krawczyk, sais}@cril.fr

### 1.1 Definitions and notations

A CSP is defined as a tuple $\mathcal{P} =< \mathcal{X}, \mathcal{C} >$. $\mathcal{X}$ is a finite set of $n$ variables $\{x_1, \ldots, x_n\}$. Each variable $x_i \in \mathcal{X}$ is defined on a finite set of $d_i$ values, denoted $dom(x_i) = \{v_{i_1}, \ldots v_{i_{d_i}}\}$. $\mathcal{C}$ is a finite set of $m$ constraints $\{c_1, \ldots, c_m\}$. Each constraint $c_i \in \mathcal{C}$ of arity $k$ is defined as a couple $(scope(c_i), R_{c_i})$ where $scope(c_i) = \{x_{i_1}, \ldots, x_{i_k}\} \subseteq \mathcal{X}$ is the set of variables involved in $c_i$ and $R_{c_i} \subseteq dom(x_{i_1}) \times \ldots \times dom(x_{i_k})$ the set of allowed tuples i.e. $t \in R_{c_i}$ iff the tuple $t$ satisfies the constraint $c_i$. A CSP $\mathcal{P}$ is called binary iff $\forall c_i \in \mathcal{C}, |scope(c_i)| \leq 2$. A model (solution) is an assignment of a value for each variable $x \in \mathcal{X}$ which satisfies all the constraints.

In this paper, we limit our presentation to binary CSPs. However, our proposed approach can be easily extended to n-ary CSPs.

We define $\phi(\mathcal{P})$ as the CSP $\mathcal{P}$ obtained after applying a local consistency $\phi$. For $\phi = AC$ this means that all arc-inconsistent values are removed from $\mathcal{P}$. If there is a variable with an empty domain in $\phi(\mathcal{P})$, we denote $\phi(\mathcal{P}) =\perp$. The sub-network obtained after the assignment of a variable $x$ to a value $v$ is denoted $\mathcal{P}|_{x=v}$.

### 1.2 Tuple Arc Consistency

In this section, we propose a new filtering technique, called Tuple Arc Consistency (TAC). This local consistency is introduced to be exploited in our redundancy framework. The main idea is that instead of fixing one value as for SAC, we fix one tuple of a constraint $c$ i.e. we assign the variables involved in $c$ and we apply AC on the obtained sub-network.

**Definition 1** *Let $\mathcal{P}$ be a CSP. A constraint $c_{ij} \in \mathcal{C}$ is Tuple Arc Consistent (TAC) iff $\forall (a, b) \in R_{c_{ij}}, AC(P|_{x_i=a, x_j=b}) \neq \perp$. $\mathcal{P}$ is TAC iff $\forall c \in \mathcal{C}$, $c$ is TAC.*

## 2 Constraints redundancies

The redundancy, in a CSP, occurs when some informations are present several times, that is, a subset of constraints can be deduced from others. To determine if a constraint $c$ is redundant or not, we need to solve the CSP with the negation of a constraint $c$. This problem is known to be Co-NP complete [4]. However, it's possible to detect in polynomial time some redundant constraints while using entailment *modulo* a given local consistency. In this section, we define formally the notion of constraint redundancy in CSP and we

| | AC, $Red_{AC}$ | | AC, $Red_{TAC}$ | | TAC, $Red_{AC}$ | | TAC, $Red_{TAC}$ | |
|---|---|---|---|---|---|---|---|---|
| instance name (#var, #ctr) | time | % | time | % | time | % | time | % |
| bqwh-15-106 (106, 644) | 0,03 | 572 (11%) | 0,16 | 570 (11%) | 0,13 | 559 (13%) | 0,25 | 555 (14%) |
| domino-1000-800 (1000, 1000) | 123,01 | 0 (100%) | 123,85 | 0 (100%) | 123,06 | 0 (100%) | 123,75 | 0 (100%) |
| driverlogw-02c-sat | 5,5 | 1910 (53%) | 10,36 | 1756 (57%) | 6,78 | 1428 (65%) | 9,58 | 1367 (66%) |
| ehi-85-297-0 (297, 4094) | 0,26 | 4094 (0%) | 12,29 | 776 (81%) | 10,72 | - | 11,25 | - |
| frb30-15-1 (30, 208) | 0,01 | 208 (0%) | 5,87 | 208 (0%) | 0,11 | 208 (0%) | 5,88 | 208 (0%) |
| rlfap-graph1 (200, 1134) | 0,15 | 1134 (0%) | 101,83 | 885 (22%) | 964 | 1134 (0%) | 1042,26 | 522 (54%) |
| rlfapscen11-f10 (680, 4103) | 0,476 | 4103 (0%) | 197,742 | 2954 (28%) | 69,827 | - | 72,548 | - |

**Table 1.** Results on benchmarks from the second international CSPs competition

show how we can use local filterings to detect some redundant constraints. In satisfiability problem redundancy *modulo* unit propagation has been shown very useful in practice namely on real-world instances [2]. A CSP $\mathcal{P}$ is redundant if it contains a subset of redundant constraints otherwise it is called irredundant. For a CSP $\mathcal{P} =< \mathcal{X}, \mathcal{C} >$ and a constraint $c \in \mathcal{C}$ , we define $\mathcal{P} \setminus \{c\}$ as the CSP $\mathcal{P}' =< \mathcal{X}, \mathcal{C} \backslash c >$. We define the negation of a constraint $c$, noted $\neg c$, as the constraint $c'$ such that $scope(c') = scope(c)$ and $R_{c'} = \{t | t \in \prod_{\forall x \in scope(c)} dom(x), t \notin R_c\}$.

**Definition 2** *Let $\mathcal{P}$ be a CSP and $c \in \mathcal{C}$. c is redundant iff $\mathcal{P} \setminus \{c\} \cup \neg c$ is unsatisfiable. $\mathcal{P}$ is redundant iff $\exists c \in \mathcal{C}$ such that c is redundant. Otherwise $\mathcal{P}$ is said to be irredundant.*

To avoid solving the problem $\mathcal{P} \setminus \{c\} \cup \neg c$ to see if $c$ is redundant or not, we consider an incomplete but polynomial time algorithm to detect redundant constraints. We apply a local filtering $\phi$ such AC and TAC. Any other local consistency can be used.

Checking if a constraint is redundant can be done using a refutation procedure. Namely, a constraint $c \in \mathcal{C}$ is redundant iff the constraint network in which $c$ is substituted by its negation is unsatisfiable. This is clearly intractable. That's why, we define weaker form of refutation inducing a weaker form of redundancy.

**Definition 3** *Let $\mathcal{P}$ be a CSP and $\phi$ a local consistency. A constraint $c \in \mathcal{C}$ is $\phi$-redundant iff $\phi(\mathcal{P} \backslash \{c\} \cup \{\neg c\}) = \perp$.*

A CSP $\mathcal{P}$ is called $\phi$-redundant (respectively $\phi$-irredundant) iff it (respectively does not) contains $\phi$-redundant constraints.

**Algorithm 1**: Computing a $\phi$ irredundant constraint network

```
  Input: $\mathcal{P} =< \mathcal{X}, \mathcal{C} >$
  Output: A $\phi$-irredundant CSP $\mathcal{P}$
1 for each $c \in \mathcal{C}$ do
2     $\mathcal{P}' \leftarrow \mathcal{P} \setminus \{c\} \cup \neg c$;
3     if $\phi(\mathcal{P}') = \perp$ then
4         $\mathcal{C} \leftarrow \mathcal{C} \backslash c$;
```

In algorithm 2, $\phi$ can be replaced by any local consistency filtering like AC and TAC. The complexity of Algorithm 1 is polynomial. If we use an AC filtering whose the time complexity is $O(md^2)$, then the time complexity of the algorithm 1 is bounded by $\mathcal{O}(m^2 d^2)$.

Let us note that using different constraint orderings in the algorithm 1, might lead to different $\phi$-irredundant constraints sub-networks.

## 3 Preliminary experiments

In this section, we show the practical interest of our approach. We present the reduction power in terms of the percentage of deleted $\phi$-redundant constraints with $\phi$ instantiated to $AC$ and $TAC$.

As a CSP solver, we used the MAC algorithm with *dom/WDeg*. In table 1, which presents results on some instances, we provide the percentage of $\phi$-redundant constraints. In the four double-columns, the results obtained by applying a $\phi$ consistency as a preprocessing and $\phi$-redundancy checking are given. For example, in the second double column ($AC$, $Red_{TAC}$) means that we apply AC on the original problem then we check constraints redundancy using $TAC$. For each case, we give the run time (in seconds), the number of remaining constraints and the percentage of $\phi$-redundant constraints. Instances solved in the preprocessing step, are indicated with a dash "-".

On the *domino-1000-1000* instance, we can see that all constraints are deleted. In fact, all the constraints become redundant since, after the preprocessing, there is one value for each domain and the instance is proven satisfiable. On the contrary, for some instances like *frb30-15-1*, the technique does not detect any redundant constraint. For the *bqwh-15-106* and *driverlogw-02c-sat* instances, we remark that a stronger filtering like *TAC* detects more redundant constraints than *AC*. This remark is confirmed for other classes like *ehi-85* and *rlfap-graph1* where the detection of redundant constraints by TAC is more significant than with AC. For classes *ehi-85* and *rlfap-scen11*, the filtering technique *TAC* prove inconsistency during the preprocessing step.

## 4 Conclusions

In this paper, a new approach to compute irredundant sub-sets of CN is proposed. Using polynomial time local consistency techniques for redundancy checking, significant reductions in the size of the have been obtained on many classes of CSP instances. The obtained sub-network is irredundant *modulo* a local consistency entailment. The new filtering TAC we propose is clearly powerful for detecting redundant constraints. Used as a preprocessing some classes of instances are solved without search.

## REFERENCES

[1] C. Bessière, R. Coletta, B. O'Sullivan, and M. Paulin, 'Query-driven constraint acquisition', in *IJCAI'2007*, pp. 50–55, (2007).

[2] O. Fourdrinoy, E. Grégoire, B. Mazure, and L. Saïs, 'Reducing hard sat instances to polynomial ones', in *IEEE-IRI'07*, pp. 18–23, (2007).

[3] A. M. Frisch, C. Jefferson, B. Martínez Hernández, and I. Miguel, 'The rules of constraint modelling', in *Ijcai'2005*, pp. 109–116, (2005).

[4] P. Liberatore, 'Redundancy in logic i: Cnf propositional formulae', *Artif. Intell.*, **163**(2), 203–232, (2005).