Conditional Constraint Satisfaction: Logical Foundations and Complexity

Georg Gottlob
Computing Laboratory
Oxford University
OX1 3QD Oxford, UK
georg.gottlob@comlab.ox.ac.uk

Gianluigi Greco
Dip. di Matematica
Università della Calabria
I-87030 Rende, Italy
ggreco@mat.unical.it

Toni Mancini
Dip. di Informatica e Sistemistica
Università di Roma "La Sapienza"
I-00198 Roma, Italy
tmancini@dis.uniroma1.it

Abstract

Conditional Constraint Satisfaction Problems (CCSPs) are generalizations of classical CSPs that support conditional activation of variables and constraints. Despite the interest emerged for CCSPs in the context of modelling the intrinsic dynamism of diagnosis, structural design, and product configuration applications, a complete characterization of their computational properties and of their expressiveness is still missing. In fact, the aim of the paper is precisely to face these open research issues. First, CCSPs are formally characterized in terms of a suitable fragment of first-order logic. Second, the complexity of some basic reasoning tasks for CCSPs is studied, by establishing completeness results for the first and the second level of the polynomial hierarchy. Finally, motivated by the hardness results, an island of tractability for CCSPs is identified, by extending structural decomposition methods originally proposed for CSPs.

1 Introduction

Constraint satisfaction is a *static* framework in its original formulation that cannot be used to model and solve decision problems in scenarios where uncertainty and dynamism come into play. Indeed, a Constraint Satisfaction Problem (CSP) instance (e.g., [Dechter, 2003]) is basically a triple $(Var, U, \mathcal{C})$, where Var is a finite set of variables, U is a finite domain of values, and $\mathcal{C} = \{C_1, C_2, \ldots, C_q\}$ is a finite set of constraints, where each C_i is a pair (S_i, r_i), in which $S_i \subseteq Var$ is called the *constraint scope*, and $r_i \subseteq U^{|S_i|}$ is called the *constraint relation*. Then, by solving a CSP instance, we simply mean determining whether there is a substitution $\theta : Var \mapsto U$ that satisfies all constraints.

To remove the assumption that all the variables, the domains of values, and the constraints are known beforehand and do not change over time, several generalized CSP frameworks have been already proposed in the literature (see, e.g., [Verfaillie and Jussien, 2005] and the references therein). In this paper, we focus on the core of these extensions, by considering the framework for Conditional Constraint Satisfaction Problems (CCSPs).

Indeed, CCSPs (proposed with the name of Dynamic CSPs in [Mittal and Falkenhainer, 2000]) are CSPs tailored to support conditional activation of variables and constraints, which have been used in model diagnosis, structural design, and configuration problems. As an example application, one may think of an automated personal computer (PC) configurator devoted to assemble several PC components. Conditional constraints may require other components to be added to the current configuration, depending on the choices made for the others. E.g., if the motherboard does not embed a video-card, then a compatible one should be added.

Despite the interest emerged for the CCSPs framework and the efforts spent in the design of specialized algorithms for their efficient solution (e.g., [Soininen *et al.*, 1999; Bowen and Bahler, 1991; Sabin *et al.*, 2003; Gelle and Faltings, 2003; Keppens and Shen, 2004]), many aspects have not yet been investigated and several research questions are, in fact, still open. For instance, both the expressiveness and the complexity issues arising in this setting have only partially been studied in the literature [Soininen *et al.*, 1999; Bowen and Bahler, 1991; Mittal and Falkenhainer, 2000]. And, more importantly, differently from the case of classical CSPs where several classes of instances that are efficiently solvable have been singled out (e.g., [Pearson and Jeavons, 1997; Gottlob *et al.*, 2000]), no island of tractability is known for CCSPs, whose identification is instead crucial for applications in real world scenarios.

The aim of this paper is precisely to shed lights on these aspects and to face the above research questions. To make our analysis formal, we shall exploit the logic-based characterization of a CSP instance (cf. [Kolaitis and Vardi, 1998]) as a pair $(\phi, \mathcal{D})$, where $\mathcal{D}$ is the *constraint database*, i.e., the set of all the constraint relations r_i, for each constraint $C_i = (S_i, r_i)$, and ϕ is a $\exists FO_{\wedge,+}$ sentence, i.e., an existentially quantified first-order formula with no negations or disjunctions, over the relational vocabulary consisting of the atoms $r_i(S_i)$. Within this framework, we investigate the class of those constraint satisfaction problems build over the $\exists FO_{\rightarrow,\wedge,+}$ fragment of the first-order logic, which consists of all the formulas obtained extending $\exists FO_{\wedge,+}$ with the implication connective "$\rightarrow$". The rationale of this approach is that enhancing $\exists FO_{\wedge,+}$ with the implication connective is a mathematically well-founded approach to model conditional activation of variables, as for it is next exemplified.

Example 1. Consider again the PC configuration scenario. Then, the $\exists FO_{\rightarrow,\wedge,+}$ formula:

$$\exists M\ motherboard(M) \wedge (hasNoVideo(M) \rightarrow \\ \exists V, C\ fit(M,V) \wedge cost(V,C))$$

states, intuitively, that a motherboard M is needed to assemble a PC; moreover, if M has no embedded video, then a video-card V fitting the requirements of M has to be chosen (and, C is the additional cost). $\triangleleft$

In fact, we show that $\exists FO_{\rightarrow,\wedge,+}$ formulas characterize (in a precise, logical sense) the class of CCSPs. Based on this result, we then embark on a systematic study of $\exists FO_{\rightarrow,\wedge,+}$. In summary:

$\triangleright$ We investigate the expressive power of CCSPs by depicting, in Section 3, a clear picture of its relationships with other logic-based CSP formalisms. The results of this comparison are graphically summarized in Figure 1.

$\triangleright$ In Section 4, we study the computational complexity of some basic reasoning tasks for CCSPs. We considered both the case where one is interested in finding a generic solution and the case where solutions are refined by means of some pro-orders. From our analysis, summarized in Figure 2, it emerges that reasoning with CCSPs is intractable, with hardness and membership results being established for various classes at the first and the second level of the polynomial hierarchy.

$\triangleright$ Motivated by the above complexity results, in Section 5, we turn to identify tractable classes of $\exists FO_{\rightarrow,\wedge,+}$ formulas, by issuing restrictions on the constraints interactions. As a bad news, we show that *decomposition methods* for traditional CSPs do not guarantee the tractability of CCSPs. Indeed, the NP-hardness of the SATISFIABILITY problem also holds for the class of CCSPs whose constraints interactions can be modelled with an acyclic *primal graph* (see, e.g., [Dechter, 2003]).

$\triangleright$ The major reason for the failure of traditional decomposition methods is that the primal graph obscures the real nature of the interactions occurring in CCSPs due to the variables activation. Hence, in Section 5.2, we introduce a special graphical representation for CCSPs, called *implication graph*, and show that SATISFIABILITY is feasible in LOGCFL and hence in polynomial time, over instances whose implications graphs are acyclic or have, more generally, *bounded treewidth* [Robertson and Seymour, 1986].

2 $\exists FO_{\rightarrow,\wedge,+}$ Formulas: Syntax and Semantics

In this section, we propose a framework for modelling conditional activation of variables and constraints in CSPs, which is based on an extension of the fragment $\exists FO_{\wedge,+}$ with the logical implication, as formalized next.

An $\exists FO_{\rightarrow,\wedge,+}$ sentence (over a vocabulary $\mathcal{R}$ of constraint relations) is a formula $\exists \vec{X}\ \phi(\vec{X})$, where $\phi(\vec{X})$ belongs to the fragment $FO_{\rightarrow,\wedge,+}$ of first-order logic, defined as follows:

- "true" and "false" are in $FO_{\rightarrow,\wedge,+}$;
- Any atom of the form $r(\vec{X})$ ($r \in \mathcal{R}$) is in $FO_{\rightarrow,\wedge,+}$;
- If $\phi(\vec{X})$ and $\psi(\vec{Z})$ are in $FO_{\rightarrow,\wedge,+}$, then $\phi(\vec{X}) \wedge \psi(\vec{Z})$ is in $FO_{\rightarrow,\wedge,+}$;
- If $\phi(\vec{X})$ is in $FO_{\wedge,+}$ and $\psi(\vec{X},\vec{Y})$ is in $FO_{\rightarrow,\wedge,+}$, then $\phi(\vec{X}) \rightarrow \exists \vec{Y}\ \psi(\vec{X},\vec{Y})$ is in $FO_{\rightarrow,\wedge,+}$.

The set of all the variables in ψ is denoted by $\mathcal{V}(\psi)$.

Example 2. The following formula $\psi_0 \in \exists FO_{\rightarrow,\wedge,+}$ adds some further requirements to the configuration constraints in Example 1 (names have been shortened, for simplicity):

$$\exists M\ m(M) \wedge (h(M) \rightarrow \exists V, C\ f(M,V) \wedge c(V,C)) \\ \wedge (e(M) \rightarrow (b(M) \rightarrow \exists H\ g(M,H)))$$

Here, we have $\mathcal{V}(\psi_0) = \{M, V, C, H\}$. $\triangleleft$

Given a constraint database $\mathcal{D}$ and an $\exists FO_{\rightarrow,\wedge,+}$ formula ψ, we denote by $(\psi, \mathcal{D})$ the constraint satisfaction problem instance of deciding an assignment for variables in ψ that satisfies all the constraints over $\mathcal{D}$.

Note that because of the presence of the implication connective, differently from the standard CSPs, a formula $\psi \in \exists FO_{\rightarrow,\wedge,+}$ can be satisfied by partial assignments over $\mathcal{V}(\psi)$. Formally, a *partial assignment* θ for $(\psi, \mathcal{D})$ is a mapping from a subset of $\mathcal{V}(\psi)$ to values in D. Then, the *θ-reduct* of ψ is the formula $\theta(\psi)$ resulting from ψ by (1) applying all the substitutions in θ to the variables in ψ, and (2) by replacing all the atoms in which a variable in $\mathcal{V}(\psi) \setminus \mathcal{V}(\theta)$ occurs with "false", where $\mathcal{V}(\theta)$ denotes the variables in the domain of θ. A solution for $(\psi, \mathcal{D})$ is a partial assignment θ such that $\theta(\psi)$ evaluates to true over $\mathcal{D}$, denoted by $\mathcal{D} \models \theta(\psi)$.

Example 3. Consider the database $\mathcal{D}_0$ containing the relations: $m = \{(m_1), (m_2)\}$, $h = \{(m_2)\}$, $f = \{(m_2, v_2)\}$, $c = \{(v_2, c_4), (v_1, c_3)\}$, $e = \{(m_1)\}$, $b = \{(m_1)\}$ and $g = \{(m_1, h_4)\}$. Then, the followings assignments are partial:

$$\theta_1 = \{M/m_2, V/v_2, C/c_3\}$$
$$\theta_2 = \{M/m_1, H/h_4\}$$

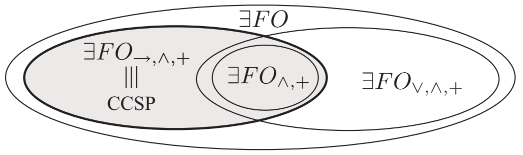
Note that θ_1 is not a solution for $(\psi_0, \mathcal{D}_0)$. On the other hand, the θ_2-reduct of ψ_0 evaluates true over $\mathcal{D}_0$. $\triangleleft$

3 Expressiveness of $\exists FO_{\rightarrow,\wedge,+}$ Formulas

The first crucial questions for CCSPs are how they relate with other approaches to deal with conditional activation of variables and constraints, and whether they are more expressive (in a precise logical sense) than classical CSPs. Both the questions are answered in this section.

Let F be a CSP formalism, i.e., a formalism for modelling constraint satisfaction problems (e.g., a subset of FO), and let $\mathcal{D}$ be a constraint database. For a formula $\phi \in F$, let $\mathtt{S}(\phi, \mathcal{D})$ denote the set of all the solutions, and for a set of variables $\mathcal{V}$, let $\mathtt{S}(\phi, \mathcal{D})_{[\mathcal{V}]}$ denote the set obtained from $\mathtt{S}(\phi, \mathcal{D})$ by restricting each solution over the variables in $\mathcal{V}$.

Let F_1 and F_2 be two CSP formalisms. We say that F_1 *weakly simulates* F_2 if for each $\phi_2 \in F_2$ there exists $\phi_1 \in F_1$ such that for each constraint database $\mathcal{D}$, $\mathtt{S}(\phi_2, \mathcal{D}) = \mathtt{S}(\phi_1, \mathcal{D})_{[\mathcal{V}(\phi_2)]}$. We say that F_1 is *more expressive* than F_2 iff F_1 simulates F_2 but F_2 does not simulate F_1. If F_1 simulates F_2 and viceversa, F_1 and F_2 are *equivalent.* If F_1 does not simulate F_2, and F_2 does not simulate F_1, we say that F_1 and F_2 are *incomparable*.



$\exists FO$
$\exists FO_{\rightarrow,\wedge,+}$
|||
CCSP
$\exists FO_{\wedge,+}$
$\exists FO_{\vee,\wedge,+}$

Figure 1: Expressiveness of $\exists FO_{\rightarrow,\wedge,+}$.

Armed with the notions above, we compared the expressiveness of $\exists FO_{\rightarrow,\wedge,+}$ with *conditional CSPs* (CCSP)s, with $\exists FO_{\wedge,+}$ and with the positive existential disjunctive fragment $\exists FO_{\vee,\wedge,+}$ of FO. Results are summarized in Figure 1.

3.1 Comparison with Conditional CSPs

A CCSP specification π is a tuple $\langle Var, Var_I, \mathcal{C}_C, \mathcal{C}_A\rangle$, where Var is a set of variables, $Var_I \subseteq Var$ is a set of *initial* variables (intuitively, of the variables that are initially "active"), and where the set of constraints is partitioned into *compatibility* ($\mathcal{C}_C$) and *activity* ($\mathcal{C}_A$) constraints.

Compatibility constraints are constraint atoms (as in traditional CSPs), while activity constraints are of the following kinds: *require variable* (RV): $c(V_1,\ldots,V_k)\overset{\text{RV}}{\rightarrow} V$; *always require variable* (ARV): $\{V_1,\ldots,V_k\}\overset{\text{ARV}}{\rightarrow} V$; *require not* (RN): $c(V_1,\ldots,V_k)\overset{\text{RN}}{\rightarrow} V$; *always require not* (ARN): $\{V_1,\ldots,V_k\}\overset{\text{ARN}}{\rightarrow} V$, with $\{V_1,\ldots,V_k,V\}\subseteq Var$.

A CCSP instance is a pair $(\pi,\mathcal{D})$, where π is a specification and $\mathcal{D}$ is a constraint database. A *legal assignment* for $(\pi,\mathcal{D})$ is a partial mapping θ from variables in Var to domain values in $\mathcal{D}$. A *solution* is a legal assignment such that: *(1)* $Var_I \subseteq \mathcal{V}(\theta)$; *(2)* θ satisfies every *active* compatibility constraints, i.e., those compatibility constraints whose variables are in $\mathcal{V}(\theta)$; and *(3)* θ satisfies every activity constraint:

- $\mathcal{W}\overset{\text{ARV}}{\rightarrow}V$ is satisfied if $\mathcal{W}\subseteq\mathcal{V}(\theta)$ implies $V\in\mathcal{V}(\theta)$;
- $c(V_1,\ldots,V_k)\overset{\text{RV}}{\rightarrow}V$ is satisfied if $\{V_1,\ldots,V_k\}\subseteq\mathcal{V}(\theta)$ and $(\theta(V_1),\ldots\theta(V_k))\in c$ (in $\mathcal{D}$) implies $V\in\mathcal{V}(\theta)$;
- $\mathcal{W}\overset{\text{ARN}}{\rightarrow}V$ is satisfied if $\mathcal{W}\subseteq\mathcal{V}(\theta)$ implies $V\notin\mathcal{V}(\theta)$;
- $c(V_1,\ldots,V_k)\overset{\text{RN}}{\rightarrow}V$ is satisfied if $\{V_1,\ldots,V_k\}\subseteq\mathcal{V}(\theta)$ and $(\theta(V_1),\ldots\theta(V_k))\in c$ (in $\mathcal{D}$) implies $V\notin\mathcal{V}(\theta)$;

As usual, the set of all the solutions is denoted by $\mathtt{S}(\pi,\mathcal{D})$.

We next show that $\exists FO_{\rightarrow,\wedge,+}$ formulas are, in fact, an elegant, mathematically-founded characterization of CCSPs.

Theorem 1. $\exists FO_{\rightarrow,\wedge,+}$ *simulates CCSP.*

Proof (Sketch). Let π be $\langle Var, Var_I, \mathcal{C}_C, \mathcal{C}_A\rangle$. We build an $\exists FO_{\rightarrow,\wedge,+}$ formula $\psi_\pi = \exists Var, Var_I \bigwedge_{c\in\mathcal{C}_C\cup\mathcal{C}_A}\phi_c$, where ϕ_c is obtained as follows:

- if c is the compatibility constraint $c(V_1,\ldots,V_k)$, then $\phi_c = \bigwedge_{V\in\{V_1,\ldots,V_k\}\setminus Var_I} dom(V) \rightarrow c(V_1,\ldots,V_k)$;
- if c is the activity constraint $c(V_1,\ldots,V_k)\overset{\text{RV}}{\rightarrow}V$, ϕ_c is given by $c(V_1,\ldots,V_k)\rightarrow dom(V)$.

Note that other activity constraints can be, in fact, simulated by RV and compatibility constraints (cf. [Mittal and Falkenhainer, 2000]). Moreover, w.l.o.g., we have assumed the existence of a monadic relation dom listing all possible values for the variables — without this relation, the simulation is also possible, but would have required a much intricate construction for ψ_π.

It is possible to show (details omitted for lack of space) that, for every constraint c of π, a legal assignment θ satisfies c iff the corresponding partial assignment θ of ψ_π satisfies the subformula ϕ_c. Hence, $\forall\mathcal{D}$, it is: $\mathtt{S}(\pi,\mathcal{D}) = \mathtt{S}(\psi_\pi,\mathcal{D})$. □

Not surprisingly, CCSPs have been previously simulated by means of different logic-based frameworks. For instance, [Bowen and Bahler, 1991] expressed CCSPs by means of the FOF logic; and, [Bacchus and Walsh, 2005] discussed a rich formalism where implications can be simulated by negations and disjunctions, by introducing a specific propagation method and characterizing when this method enforces generalized arc-consistency. Yet, the precise logical characterization of CCSPs was unknown, which is made clear below.

Theorem 2. *CCSP simulates* $\exists FO_{\rightarrow,\wedge,+}$.

Proof (Sketch). Let ψ be an $\exists FO_{\rightarrow,\wedge,+}$ formula. We build a CCSP π_ψ as follows. First, all the atoms are converted into compatibility constraints. Then, for each subformula $\phi'(\vec{X})\rightarrow\exists\vec{Y}\phi''(\vec{X},\vec{Y})$ where $\phi' = c_1(\vec{X})\wedge\cdots\wedge c_k(\vec{X})$, and where $\vec{Y}=\{Y_1,\ldots Y_m\}$, we create $m\times k$ fresh variables Y_i^j $(1\le i\le m, 1\le j\le k)$, $m\times k$ activation constraints of the form $c_j(\vec{X})\overset{\text{RV}}{\rightarrow}Y_1^j,\ldots c_j(\vec{X})\overset{\text{RV}}{\rightarrow}Y_m^j$ $(1\le j\le k)$, and m ARV constraints of the form $\{Y_i^1,\ldots,Y_i^k\}\overset{\text{ARV}}{\rightarrow}Y_i$. It can be shown that for each database $\mathcal{D}$, $\mathtt{S}(\psi,\mathcal{D}) = \mathtt{S}(\pi_\psi,\mathcal{D})_{[\mathcal{V}(\psi)]}$.

Interestingly, if ψ is such that the left-hand side of any implicative formula is a constraint atom only, the encoding can be adjusted not to use fresh variables. Hence, we can show a stronger kind of simulation: $\forall\mathcal{D}$, $\mathtt{S}(\psi,\mathcal{D}) = \mathtt{S}(\pi_\psi,\mathcal{D})$. □

Corollary 1. *CCSP and* $\exists FO_{\rightarrow,\wedge,+}$ *are equivalent.*

Before leaving the section, note that in [Mittal and Falkenhainer, 2000] solutions to CCSPs are also required to be subset-minimal. In fact, the simulation techniques of Theorem 1 and 2 (in the case left sides of implications are constraint atoms) established a one-to-one correspondence between solutions of CCSPs and assignments for $\exists FO_{\rightarrow,\wedge,+}$ formulas. Hence, further properties on the solutions are preserved as well, but formal results are omitted.

3.2 Comparison with Other Logics for CSPs

A comparison between CSPs and CCSPs has been proposed in [Soininen *et al.*, 1999], by focusing on a *knowledge representation* perspective. Indeed, it is shown that CCSPs are not *modular* representable by standard CSPs, i.e., intuitively, that small changes in a CCSP may result in significant changes in any corresponding CSP, which is an encoding for it.

Based on the logical equivalence between CCSPs and $\exists FO_{\rightarrow,\wedge,+}$, we can now formally compare CCSPs with CSPs, and in fact prove that there are CCSPs (formulas in $\exists FO_{\rightarrow,\wedge,+}$) that cannot be "simulated" in $\exists FO_{\wedge,+}$. The argument is that there is no way to encode the "$\rightarrow$" connective by means of conjunctions. In turn, "$\rightarrow$" cannot encode disjunction of atoms. Thus, the following relationships hold[1].

Theorem 3. $\exists FO_{\rightarrow,\wedge,+}$ *is such that:* (1) *it is more expressive than* $\exists FO_{\wedge,+}$*; and,* (2) *it is incomparable with* $\exists FO_{\vee,\wedge,+}$.

[1]Proofs are reported in an extended version of the paper available at `www.unical.it/~ggreco`.

4 The Complexity of Reasoning with CCSPs

Let $(\psi, \mathcal{D})$ be a CCSP instance. The intrinsic complexity of some basic reasoning tasks for $(\psi, \mathcal{D})$ has been studied in [Soininen *et al.*, 1999; Mittal and Falkenhainer, 2000]. In particular, it is known that the SATISFIABILITY problem (does there exists a partial assignment θ such that θ is a solution for $(\psi, \mathcal{D})$?) is NP-complete.

However, there are other important reasoning tasks that often come into play with CSPs, and whose complexity has not been investigated for CCSPs. In this section, we explore these issues, by focusing on the following problems:

- CHECKING: Given a partial assignment θ, is θ a solution for $(\psi, \mathcal{D})$?
- RELEVANCE: Given a variable X, is X contained in $\mathcal{V}(\theta)$ for some solution θ for $(\psi, \mathcal{D})$?
- NECESSITY: Given a variable X, is X contained in $\mathcal{V}(\theta)$ for every solution θ for $(\psi, \mathcal{D})$?

When moving to CCSPs, these problems can be specialized to take care of those solutions that are partial assignments. Indeed, knowing whether there are solutions in which some variable is not "active" may be quite relevant. For instance, in product configuration, one may be interested in finding the configuration with the minimum number of (additional) components that, yet, meets the users' requirements.

In order to model scenarios as above, it is convenient to refine the notion of solutions by means of some pre-order $\preceq$ on the set of all the possible partial assignments. Accordingly, a solution θ for $(\psi, \mathcal{D})$ is said *$\preceq$-minimal* if for each solution θ', it holds: $\theta \preceq \theta'$. In particular, we next consider $\preceq$-minimal solutions, where $\preceq$ is one of the following pre-orders[2]:

subset minimality "$\subseteq$": $\theta_1 \subseteq_v \theta_2$ iff $\mathcal{V}(\theta_1) \subseteq \mathcal{V}(\theta_2)$.
minimum cardinality "$\leq$": $\theta_1 \leq \theta_2$ iff $|\mathcal{V}(\theta_1)| \leq |\mathcal{V}(\theta_2)|$.
weighted cardinality "$\sqsubseteq$": A weight w is attached to each variable, and

$$\theta_1 \sqsubseteq \theta_2 \text{ iff } \sum_{X \in \mathcal{V}(\theta_1)} w(X) \leq \sum_{X \in \mathcal{V}(\theta_2)} w(X).$$

Next, we depict a complete picture of the complexity figures arising in the CCSP setting.

Theorem 4. *The complexity of* CHECKING, RELEVANCE, *and* NECESSITY *and of their variants for $\preceq$-minimal solutions ($\preceq \in \{$"$\subseteq$", "$\leq$", "$\sqsubseteq$"$\}$) is as shown in Figure 2.*

Proof (Sketch). When no pre-order is considered, results easily derive from standard results on CSPs. Due to space limitations, we next only report the proof sketches for the variants of CHECKING.
CHECKING$_\subseteq$ *is* co-NP-*complete: (Membership)* Let $(\psi, \mathcal{D})$ be a CCSP instance, and let θ be a partial assignment such that $\theta \models \psi$. Consider the complementary problem of deciding whether there is an assignment θ' such that *(i)* θ' is a solution for $(\psi, \mathcal{D})$ and *(ii)* $\mathcal{V}(\theta') \subseteq \mathcal{V}(\theta)$ does not hold. Clearly, θ' can be guessed in NP, while *(i)* and *(ii)* can be checked in P.

(Hardness) Deciding whether a Boolean formula in conjunctive normal form $\Phi = c_1 \wedge \ldots \wedge c_m$ over the variables

	—	"$\subseteq$"	"$\leq$"	"$\sqsubseteq$"
CHECKING	in P	co-NP-c	co-NP-c	co-NP-c
RELEVANCE	NP-c	Σ_2^P-c	$\mathrm{P}^{\mathrm{NP}[O(\log n)]}$-c	P^{NP}-c
NECESSITY	co-NP-c	co-NP-c	$\mathrm{P}^{\mathrm{NP}[O(\log n)]}$-c	P^{NP}-c

Figure 2: Complexity of CCSPs ('c' stands for complete).

$X_1, \ldots, X_n$ is *not* satisfiable, i.e., deciding whether there exist no truth assignments to the variables making each clause c_j true, is a well-known co-NP-complete problem.

We build a CCSP instance $(\psi(\Phi), \mathcal{D}(\Phi))$ as follows. For each clause c_j, which is w.l.o.g. of the form $t_{j_1} \vee t_{j_2} \vee t_{j_3}$, where t_{j_1} is a variable X_i or its negation $\neg X_i$, $\mathcal{D}(\Phi)$ contains the relation $rc_j = \{(T_1, T_2, T_3) \mid T_i \in \{\text{"true"}, \text{"false"}\}$ s.t. c_j is not satisfied when $t_{j_i} = T_i, \forall i\}$. Then, for each variable X_i in Φ, $\mathcal{D}(\Phi)$ contains the relation $rv_i = \{(\text{"true"}), (\text{"false"})\}$. Finally, $unsat = \{(1)\}$ is in $\mathcal{D}(\Phi)$, and no other relation is in $\mathcal{D}(\Phi)$. The formula $\psi(\Phi)$ is of the form $\exists X'_1, ..., X'_n, Z \bigwedge_{1 \leq i \leq n} rv_i(X'_i) \wedge \bigwedge_{1 \leq j \leq m} (rc_j(X'_{j_1}, X'_{j_2}, X'_{j_3}) \rightarrow unsat(Z))$, where $X'_{j_i} \in \{X'_1, ..., X'_n\}$ is a variable such that X_{j_i} occurs in c_j.

Let $\mathbf{x}$ be a truth value assignment for the variables in Φ. Let $\theta^{\mathbf{x}}$ denote the assignment for $(\psi(\Phi), \mathcal{D}(\Phi))$ such that: X'_i is mapped to "true" (resp., "false") in $\theta(\mathbf{x})$ iff X_i is true (resp., false) $\mathbf{x}$, and such that $Z/1$ is in $\theta^{\mathbf{x}}$ iff $\mathbf{x}$ is not satisfying. Then, any solution θ' for $(\psi(\Phi), \mathcal{D}(\Phi))$ is such that $\mathcal{V}(\theta') \supseteq \{X'_1, ..., X'_n\}$; moreover, Z may not occur in $\mathcal{V}(\theta')$ if and only if there is a satisfying assignment $\mathbf{x}$ for Φ such that $\theta' = \theta^{\mathbf{x}}$. To conclude the proof, we can hence take an assignment $\mathbf{x_u}$ that does not satisfy Φ and notice that $\theta^{\mathbf{x_u}}$ is $\subseteq$-minimal $\Leftrightarrow$ there is no satisfying assignment for Φ.
CHECKING$_\leq$ *and* CHECKING$_\sqsubseteq$ *are* co-NP-*complete:* Memberships can be shown with similar arguments as above. The hardness of CHECKING$_\leq$ can be shown by observing that, in the above reduction, $\theta^{\mathbf{x_u}}$ is in fact $\leq$-minimal $\Leftrightarrow$ $\theta^{\mathbf{x_u}}$ $\subseteq$-minimal $\Leftrightarrow$ there is no satisfying assignment for Φ. Finally, the hardness of CHECKING$_\sqsubseteq$ follows by assigning unitary weights to all the variables of $\psi(\Phi)$. $\square$

5 Restricted Classes of CCSPs

In the light of the intractability results of the previous section, it is relevant to single out classes of CCSPs that can be efficiently solved. To this aim, we next investigate how to adapt the approaches used in traditional CSPs, where tractable classes are identified by *structural decomposition methods* (e.g., [Pearson and Jeavons, 1997; Gottlob *et al.*, 2000]).

5.1 The Hardness of Acyclic Structures

Let ψ be a *FO* formula. The structure of constraints interactions in ψ can be represented by the *primal graph* $\mathrm{G}(\psi) = (\mathcal{V}(\psi), E)$, where two variables occur in some edge of E if they appear together in a constraint atom in ψ. We next focus on classes of primal graphs having bounded treewidth.

A *tree decomposition* of a graph $G = (V, E)$ is a pair $\langle T, \chi \rangle$, where $T = (N, F)$ is a tree, and χ is a function assigning to each vertex $p \in N$ a set of vertices $\chi(p) \subseteq V$, such that the following conditions are satisfied: (1) $\forall b \in V$, $\exists p \in N$ such that $b \in \chi(p)$; (2) $\forall \{b, d\} \in E$, $\exists p \in N$ such

[2]In fact, these pre-orders have been widely used for refining solution concepts in other AI frameworks.

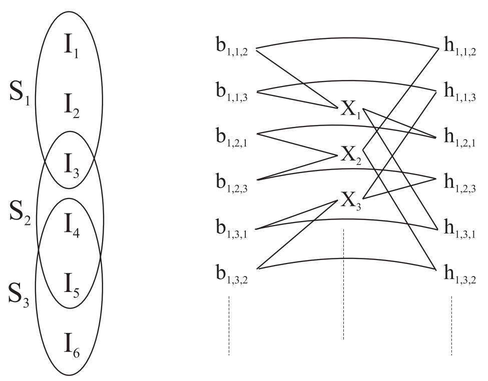


Figure 3: An EXACT-COVER-BY-3-SETS instance.

that $\{b, d\} \subseteq \chi(p)$; (3) $\forall b \in N$, the set $\{p \in N \mid b \in \chi(p)\}$ induces a connected subtree of T. The *width* of $\langle T, \chi\rangle$ is $\max_{p\in N} |\chi(p) - 1|$, and the *treewidth* of G (short: $tw(G)$) is the minimum width over all its tree decompositions.

Let $\mathcal{L}$ be a subset of FO. We denote by $\mathcal{CSP}_{\mathtt{btw}}(\mathcal{L})$ the class of all the CSP instances $(\phi, \mathcal{D})$, where ϕ is in $\mathcal{L}$ and where $tw(\mathrm{G}(\phi))$ is bounded by a fixed constant. It is well-known (see, e.g., [Gottlob *et al.*, 2000]) that SATISFIABILITY can be solved in polynomial time on $\mathcal{CSP}_{\mathtt{btw}}(\exists FO_{\wedge,+})$. We next show that this is not the case for $\mathcal{CSP}_{\mathtt{btw}}(\exists FO_{\rightarrow,\wedge,+})$, even when further restricted on the class $\mathcal{CSP}_{\mathtt{ac}}(\exists FO_{\rightarrow,\wedge,+})$ of *acyclic* instances (i.e., those having treewidth 1).

Theorem 5. SATISFIABILITY *is* NP-*hard, even when restricted on the class* $\mathcal{CSP}_{\mathtt{ac}}(\exists FO_{\rightarrow,\wedge,+})$.

Proof. Let $\mathcal{I} = \{I_1, ..., I_n\}$ be a set of elements, and let $\mathcal{S} = \{S_1, ..., S_m\}$ be a number of sets each one containing exactly three elements in $\mathcal{I}$. Recall that the EXACT-COVER-BY-3-SETS problem of deciding whether there exists a set $\mathcal{C} \subseteq \{S_1, ..., S_m\}$ such that $\bigcup_{S_i\in\mathcal{C}} S_i = \{I_1, ..., I_n\}$ and $S_i \cap S_j = \emptyset$, for each $S_i, S_j \in \mathcal{C}$ with $i \neq j$, is NP-complete.

As an example instance, Figure 3 shows (on the left) the sets $S_1 = \{I_1, I_2, I_3\}$, $S_2 = \{I_4, I_5, I_6\}$, and $S_3 = \{I_3, I_4, I_5\}$. A solution consists of the set $\{S_1, S_3\}$.

We build a CCSP instance $(\psi(\mathcal{I}, \mathcal{S}), \mathcal{D}(\mathcal{I}, \mathcal{S}))$ as follows. For each item I_i, $\mathcal{D}(\mathcal{I}, \mathcal{S})$ contains the relation $rI_i = \{(s_j) \mid S_j \in \mathcal{S} \text{ and } I_i \in S_j\}$. For each set S_j and item $I_i \in S_j$, $\mathcal{D}(\mathcal{I}, \mathcal{S})$ contains the relation $rS_{ji} = \{(s_j)\}$. And, no other relation is in $\mathcal{D}(\mathcal{I}, \mathcal{S})$. Moreover, the formula $\psi(\mathcal{I}, \mathcal{S})$ is of the form: $\exists X_1, ..., X_n, \bigwedge_i rI_i(X_i) \wedge \bigwedge_{i,i',j|\{I_i,I_{i'}\}\subseteq S_j}(rS_{ji}(X_i) \rightarrow rS_{ji'}(X_{i'}))$. Importantly, given that constraint relations in $\psi(\mathcal{I}, \mathcal{S})$ are unary, $\mathrm{G}((\psi(\mathcal{I}, \mathcal{S}), \mathcal{D}(\mathcal{I}, \mathcal{S})))$ does not contain any edge and, hence, it is trivially acyclic.

We now claim that: there exists an exact cover of $\mathcal{I}$ with sets from $\mathcal{S} \Leftrightarrow (\psi(\mathcal{I}, \mathcal{S}), \mathcal{D}(\mathcal{I}, \mathcal{S}))$ admits a solution.

($\Rightarrow$) Let $\mathcal{C}$ be an exact cover. Consider the assignment $\theta^{\mathcal{C}}$ build as follows: (1) $\mathcal{V}(\psi(\mathcal{I}, \mathcal{S})) = \mathcal{V}(\theta^{\mathcal{C}})$; and (2) X_i is mapped to s_j if and only if $I_i \in S_j$ and $S_j \in \mathcal{C}$. Note that the conjunction $\bigwedge_i rI_i(X_i)$ is satisfied by $\theta^{\mathcal{C}}$, by construction of the relation rI_i in $\mathcal{D}(\mathcal{I}, \mathcal{S})$. Hence, it remains to show that each implication of the form $(rS_{ji}(X_i) \rightarrow rS_{ji'}(X_{i'}))$ is satisfied as well. We distinguish two cases. If X_i is not mapped to s_j, then $rS_{ji}(X_i)$ evaluates false over the database, and the implication is trivially satisfied. Otherwise, i.e., if X_i is mapped to s_j, we must have that $X_{i'}$ with $I_{i'} \in S_j$ is mapped to s_j as well, which in fact holds by construction of $\theta^{\mathcal{C}}$.

($\Leftarrow$) Assume there is a solution θ for $(\phi(\mathcal{I}, \mathcal{S}), \mathcal{D}(\mathcal{I}, \mathcal{S}))$. We first notice that the following properties hold on θ: (P_1) $\mathcal{V}(\theta) = \{X_1, ..., X_n\}$; and (P_2) if X_i is mapped to s_j in θ, then $X_{i'}$ is mapped to s_j, for each $I_{i'} \in S_j$. Then, we can build the set $\mathcal{C}^{\theta} = \{S_j \mid \exists X_i \text{ that is mapped to } s_j \text{ in } \theta\}$, and notice that it is an exact cover. $\square$

5.2 Tractable Classes

Since classical approaches fail in isolating classes of tractable CCSPs, we next propose and study a decomposition strategy specific for $\exists FO_{\rightarrow,\wedge,+}$ formulas. We shall focus on *plain* formulas, where nesting of the implication connective is not allowed. Formally, a *plain* $\exists FO_{\rightarrow,\wedge,+}$ formula is of the form:

$$\exists \vec{X}\ \phi(\vec{X}) \wedge \bigwedge_{1\leq i\leq n} (b_i(\vec{X}) \rightarrow \exists \vec{Y_i} h_i(\vec{X}, \vec{Y_i}))$$

where $n \geq 0$, and ϕ, b_i, h_i are in $FO_{\wedge,+}$.

In fact, arbitrary formulas can be made plain by iteratively substituting each subformula $(\phi \rightarrow ((\phi' \rightarrow \phi'') \wedge \phi'''))$ with $(\phi \wedge \phi' \rightarrow \phi'') \wedge (\phi \rightarrow \phi''')$; this transformation preserves the set of all the solutions, and is feasible in quadratic time.

Let ψ be a plain $\exists FO_{\rightarrow,\wedge,+}$. Constraint interactions in ψ can be represented with the *implication graph* $\mathrm{IG}(\psi) = (V, E)$, where $V = \mathcal{V}(\psi) \cup \{b_1, ..., b_n\} \cup \{h_1, ..., h_n\}$ and edges have the form: (X, Y) between pairs of variables occurring in the same atom; (X, b_i) if X occurs in some atom of b_i; (X, h_i) if X occurs in some atom of h_i; and (b_i, h_i), for each subformula $(b_i(\vec{X}) \rightarrow \exists \vec{Y_i} h_i(\vec{X}, \vec{Y_i}))$. E.g., a portion of the implication graph for the EXACT-COVER-BY-3-SETS instance in the proof of Theorem 5, which reveals the "hidden" intricacy of the problem, is reported in Figure 3 ($b_{j,i,i'}$ and $h_{j,i,i'}$ refer to the implication $rS_{ji}(X_i) \rightarrow rS_{ji'}(X_{i'})$).

Then, the class $\mathcal{CSP}_{\mathtt{[i]btw}}(\exists FO_{\rightarrow,\wedge,+})$ is defined as the set of all the CCSPs instances $(\psi, \mathcal{D})$, where ψ is a plain formula in $\exists FO_{\rightarrow,\wedge,+}$ such that $tw(\mathrm{IG}(\psi))$ is bounded by a fixed constant. An algorithm, called DecideSolution$_k$, deciding whether an instance in $\mathcal{CSP}_{\mathtt{[i]btw}}(\exists FO_{\wedge,+})$ has a solution is reported in Figure 4.

The algorithm works on an tree decomposition of $\mathrm{IG}(\phi)$ of width k, whose shape is $T = (N, F)$. Roughly, it is based on a recursive procedure *findSolution* that receives in input a vertex $v \in N$. Each vertex is equipped with some nodes associated with variables of ϕ, denoted by $\mathcal{V}(v)$, as well as with nodes from $\{b_1, ..., b_n\}$ and $\{h_1, ..., h_n\}$. The sets of these nodes are denoted by $\mathcal{B}(v)$ and $\mathcal{H}(v)$, respectively.

Basically, *findSolution* guesses for each child c of v (in F), a partial assignment θ_c for $\mathcal{V}(c)$ (conforming with the assignment for θ_v), and two "states" $\mathbf{b}_c$ and $\mathbf{h}_c$. The state $\mathbf{b}_c$ (resp., $\mathbf{h}_c$) is a mapping from $\mathcal{B}(c)$ (resp., $\mathcal{H}_c$) to the values $\{\mathtt{T}, \mathtt{F}, \mathtt{willF}, \mathtt{wasF}\}$. The mapping $\mathbf{b}_c(b_i) = \mathtt{T}$ states that the conjunct b_i of ϕ must be true. In fact, when $\mathbf{b}_c(b_i) = \mathtt{T}$, it must be the case that the atoms in b_i restricted over the assignment for v are satisfied in $\mathcal{D}$ (short: $\mathcal{D} \models \pi_{b_i}(v)$ in $B1$). Note that the information that $\mathbf{b}_c(b_i) = \mathtt{T}$ is propagated to all

```
Input: $(\phi, \mathcal{D})$, and a $k$-width tree decomposition $\langle T, \mathcal{V}, \mathcal{H}, \mathcal{B} \rangle$
                    of IG$((\phi, \mathcal{D}))$ with $T = (N, F)$;
Output: true iff there is a solution $\theta$ for $(\psi, \mathcal{D})$;
--------------------------------------------------------------------
Boolean Function $findSolution(v$ : vertex in $N$;
  $\theta_v$ : partial assignment; $\mathbf{b}_v$ : state for $\mathcal{B}(v)$; $\mathbf{h}_v$ : state for $\mathcal{H}(v))$;
begin
 for each $c$ s.t. $(v, c) \in F$ guess:
    a partial assignment $\theta_c$ for $\mathcal{V}(c)$;
    a "state" $\mathbf{b}_c$ for $\mathcal{B}(c)$, and
    a "state" $\mathbf{h}_c$ for $\mathcal{H}(c)$;
 check that the following conditions hold:
   $V1 : \mathcal{D} \models \pi_\phi(v)$;
   $V2$ : for each $c \in N$ s.t. $(v, c) \in F$
            $\theta_v$ conforms with $\theta_c$;
   $B1$ : for each $b_i \in \mathcal{B}(v)$
            $\mathbf{b}_v(b_i) = \mathtt{T} \Rightarrow \mathcal{D} \models \pi_{b_i}(v)$;
            $\mathbf{b}_v(b_i) = \mathtt{F} \Leftrightarrow \mathcal{D} \not\models \pi_{b_i}(v)$;
   $B2$ : for each $b_i \in \mathcal{B}(p)$ s.t. $\mathbf{b}_p(b_i) = \mathtt{T}$
            $\forall (v, c) \in F, b_i \in \mathcal{B}(c) \Rightarrow \mathbf{b}_p(b_i) = \mathtt{T}$;
   $B3$ : for each $b_i \in \mathcal{B}(p)$ s.t. $\mathbf{b}_p(b_i) = \mathtt{F} \vee \mathtt{wasF}$
            $\forall (v, c) \in F, b_i \in \mathcal{B}(c) \Rightarrow \mathbf{b}_p(b_i) = \mathtt{wasF}$;
   $B4$ : for each $b_i \in \mathcal{B}(p)$ s.t. $\mathbf{b}_p(b_i) = \mathtt{willF}$
            $\forall (v, c) \in F, b_i \in \mathcal{B}(c) \Rightarrow \mathbf{b}_c(b_i) \neq \mathtt{T}$;
            $\exists (v, c) \in F$ s.t. $b_i \in \mathcal{B}(c) \wedge (\mathbf{b}_c(b_i) = \mathtt{willF} \vee \mathtt{F})$;
   $H1$ : for each $h_i \in \mathcal{H}(v)$
            $\mathbf{h}_v(h_i) = \mathtt{T} \Rightarrow \mathcal{D} \models \pi_{h_i}(v)$;
            $\mathbf{h}_v(h_i) = \mathtt{F} \Leftrightarrow \mathcal{D} \not\models \pi_{h_i}(v)$;
   $H2$ : for each $h_i \in \mathcal{H}(p)$ s.t. $\mathbf{h}_p(h_i) = \mathtt{T}$
            $\forall (v, c) \in F, h_i \in \mathcal{H}(c) \Rightarrow \mathbf{h}_p(h_i) = \mathtt{T}$;
   $H3$ : for each $h_i \in \mathcal{B}(p)$ s.t. $\mathbf{h}_p(b_i) = \mathtt{F} \vee \mathtt{wasF}$;
            $\forall (v, c) \in F, h_i \in \mathcal{H}(c) \Rightarrow \mathbf{h}_p(h_i) = \mathtt{wasF}$;
   $H4$ : for each $h_i \in \mathcal{H}(p)$ s.t. $\mathbf{h}_p(h_i) = \mathtt{willF}$
            $\forall (v, c) \in F, h_i \in \mathcal{H}(c) \Rightarrow \mathbf{h}_c(h_i) \neq \mathtt{T}$;
            $\exists (v, c) \in F$ s.t. $h_i \in \mathcal{H}(c) \wedge (\mathbf{h}_c(b_i) = \mathtt{willF} \vee \mathtt{F})$;
   $I1$ : for each $(b_i, h_i) \in \mathcal{B}(v) \times \mathcal{H}(v)$
            $\mathbf{b}_v(b_i) = \mathtt{T} \Rightarrow \mathbf{h}_v(h_i) = \mathtt{T}$;
   $R1$ : for each $c \in N$ s.t. $(v, c) \in F$,
              $findSolution(c, \theta_c, \mathbf{b}_c, \mathbf{h}_c)$;
 if this check fails then return false;
 else return true;
end;
--------------------------------------------------------------------
  begin (* MAIN *)
    let $v$ be the root of $T$;
    guess the sets $\theta_v$, $\mathbf{b}_v$, and $\mathbf{h}_v$;
    check conditions $(V1)$, $(B1)$, $(H1)$, and $(I1)$;
    if this check fails then return false;
    else return $findSolution(r, \theta_v, \mathbf{b}_b, \mathbf{h}_v)$;
  end.
```

Figure 4: **Algorithm** DecideSolution$_k$.

the children $(B2)$, so that all the atoms in b_i are eventually satisfied. On the other hand, when $\mathbf{b}_c(b_i) = \mathtt{F}$ the algorithm must find an atom which is not true in the database $(B1)$. Moreover, all the children are advertised that b_i is false, by putting them into the state `wasF` in $(B3)$, which is basically a do not care state — this is also done when $\mathbf{b}_p(b_i) = \mathtt{wasF}$, to propagate this knowledge in the tree. Note that, it may happen that b_i will be made false during the computation but not in the current node v. For these cases, we use the state `willF`, which forces $(B4)$ that none of its children are in state `T` and that there is at least a children with state `willF` or `F`. Similar considerations apply to the mapping $\mathbf{h}_v$. Eventually, $(I1)$ checks whether the implications are satisfied, while $(V1)$ checks whether the pure conjunctive part is satisfied.

The proof of correctness of DecideSolution$_k$ is omitted. For the running time, we can adapt the techniques in [Gottlob *et al.*, 2002] to show the following.

Theorem 6. *On* $\mathcal{CSP}_{\mathtt{[i]btw}}(\exists FO_{\rightarrow,\wedge,+})$, SATISFIABILITY *is in* LOGCFL, *hence, tractable and highly-parallelizable.*

Acknowledgments

This work was supported by a Royal Society Wolfson Research Merit Award, allowing G. Gottlob to invite G. Greco and T. Mancini to Oxford.

In addition, G. Greco's work was supported by M.I.U.R. under project "tocai.it: Tecnologie Orientate alla Conoscenza per Aggregazioni di Imprese in Internet".

References

[Bowen and Bahler, 1991] J. Bowen and D. Bahler. Conditional existence of variables in generalised constraint networks. In *Proc. of AAAI'91*, pages 215–220, 1991.

[Bacchus and Walsh, 2005] F. Bacchus and T. Walsh. Propagating Logical Combinations of Constraints. In *Proc. of IJCAI'05*, pages 35–40, 1995.

[Dechter, 2003] R. Dechter. *Constraint Processing*. Morgan Kaufmann, Los Altos, 2003.

[Gelle and Faltings, 2003] E. Gelle and B. Faltings. Solving mixed and conditional constraint satisfaction problems. *Constraints*, 8(2):107–141, 2003.

[Gottlob *et al.*, 2000] G. Gottlob, N. Leone, and F. Scarcello. A comparison of structural csp decomposition methods. *Artif. Intell.*, 124(2):243–282, 2000.

[Gottlob *et al.*, 2002] G. Gottlob, N. Leone, and F. Scarcello. Computing logcfl certificates. *TCS*, 270(1-2):761–777, 2002.

[Keppens and Shen, 2004] J. Keppens and Q. Shen. Compositional model repositories via dynamic constraint satisfaction with order-of-magnitude preferences. *JAIR*, 21:499–550, 2004.

[Kolaitis and Vardi, 1998] P. G. Kolaitis and M. Y. Vardi. Conjunctive-query containment and constraint satisfaction. *JCSS*, 61(2):302–332, 1998.

[Mittal and Falkenhainer, 2000] S. Mittal and B. Falkenhainer. Dynamic Constraint Satisfaction Problems. In *Proc. of AAAI'90*, pages 25–32, 2000.

[Pearson and Jeavons, 1997] J.K. Pearson and P.G. Jeavons. A survey of tractable constraint satisfaction problems. CSD-TR-97-15, Royal Holloway, University of London, July 1997.

[Robertson and Seymour, 1986] N. Robertson and P. D. Seymour. Graph minors. II. algorithmic aspects of treewidth. *J. of Algorithms*, 7(3):309–322, 1986.

[Sabin *et al.*, 2003] M. Sabin, E. C. Freuder, and R. J. Wallace. Greater efficiency for conditional constraint satisfaction. In *Proc. of CP 2003*, pages 649–663, 2003.

[Soininen *et al.*, 1999] G. Soininen, T. Esther, and I. Niemelä. A fixpoint definition of Dynamic Constraint Satisfaction. In *Proc. of CP'99*, pages 419–433, 1999.

[Verfaillie and Jussien, 2005] G. Verfaillie and N. Jussien. Constraint solving in uncertain and dynamic environments: A survey. *Constraints*, 10:253–281, 2005.