ELSEVIER

Artificial Intelligence 135 (2002) 199–234

Artificial Intelligence

www.elsevier.com/locate/artint

Consistency restoration and explanations in dynamic CSPs—Application to configuration

Jérôme Amilhastre [a], Hélène Fargier [b,*], Pierre Marquis [c]

[a] *Access Commerce, BP 555, 31674 Labege Cedex, France*
[b] *Institut de Recherche en Informatique de Toulouse, 118 route de Narbonne, 31062 Toulouse Cedex, France*
[c] *Centre de Recherche en Informatique de Lens, Rue de l'Université, 62300 Lens, France*

Received 17 February 2001

Abstract

Most of the algorithms developed within the Constraint Satisfaction Problem (CSP) framework cannot be used as such to solve interactive decision support problems, like product configuration. Indeed, in such problems, the user is in charge of assigning values to variables. Global consistency maintaining is only one among several functionalities that should be offered by a CSP-based platform in order to help the user in her task; other important functionalities include providing explanations for some user's choices and ways to restore consistency.

This paper presents an extension of the CSP framework in this direction. The key idea consists in considering and handling the user's choices as assumptions. From a theoretical point of view, the complexity issues of various computational tasks involved in interactive decision support problems are investigated. The results cohere with what is known when Boolean constraints are considered and show all the tasks intractable in the worst case. Since interactivity requires short response times, intractability must be circumvented some way. To this end, we present a new method for compiling configuration problems, that can be generalized to valued CSPs. Specifically, an automaton representing the set of solutions of the CSP is first computed off-line, then this data structure is exploited so as to ensure both consistency maintenance and computation of maximal consistent subsets of user's choices in an efficient way. © 2001 Published by Elsevier Science B.V.

Keywords: CSPs; Assumptions; Explanations; Restorations; Interactive configuration; Compilation

* Corresponding author.
E-mail address: fargier@irit.fr (H. Fargier).

1. Introduction

Constraint programming techniques are widely used to model and solve decision problems. Many algorithms developed in this area aim at solving automatically some families of CSPs. Accordingly, they do not help solving decision support problems that are interactive in essence. For such problems, the user herself is in charge of the choice of values for variables and the role of the system is not to solve a CSP, but to *help the user* in this task. Product configuration [34,40] is a typical example of such problems: a configurable product is defined by a finite set of components, options, or more generally by a set of attributes, the values of which have to be chosen by the user. These values must satisfy a finite set of configuration constraints that encode the feasibility of the product, the compatibility between components, their availability, etc. At a first glance, [1] a configurable product can be represented by means of a CSP, the solutions of which represents the catalog, i.e., all the variants of the product that are feasible.

When configuring a product, the user specifies her requirements by interactively giving values to variables or more generally by stating some unary constraints that restrict the possible values of the decision variables. An important feature of interactive configuration is that such constraints do not have the same status than initial configuration constraints but can be removed during the configuration process because they lead to a solution that is judged not acceptable by the user. Furthermore, all the user's choices do not necessarily have the same importance, but may be subject to preferences (for instance, when configuring a car, requirements dealing with the type of engine can be more important than those concerning the color of the car). Now each time a new choice is made, the domains of the variables must be pruned so as to ensure that the values available for the further variables can lead to a feasible product (i.e., a product satisfying all the initial configuration constraints). Finally, if the current set of choices becomes inconsistent with the constraints, or if the user is not happy with some derived consequences of these choices, she has to backtrack and relax some of them. To sum up, a decision support system should be able to fulfill the following requirements:

- *Maintain consistency*: The system has to ensure at each time that the current set of user's choices is consistent with the CSP modeling the configurable product, ideally globally consistent, or at least to detect inconsistency as soon as possible. It has also to compute the consequences of the user's choices by deleting (respectively restoring) all the values that are incompatible (respectively compatible) with the current set of choices: in other words, the current domains should obey the property of "global consistency" or at least a property of local consistency like arc-consistency, restricted path consistency, etc.
- *Guide relaxation on user's requirements by providing restorations*: The system has to help the user backtracking by answering questions like: "Which choices should I relax

[1] This is obviously an approximation: a configurable product is often structured into sub-components, the existence of which depends on the values given to some of the variables of the upper component. In this context, the set of variables of the problem cannot be defined *a priori*. Several authors have proposed to extend the classical CSP framework in order to handle such structural characteristics [22,27,32,35,39]. However, these works do not address interactivity in the configuration task since they assume that the whole set of user's requirements is given at start.

in order to recover consistency?” or “Which choices should I relax in order to render such a value available for such a variable?”. The problem here is to identify consistent subsets of the current choices, possibly maximal consistent subsets or consistent subsets minimizing a cost function.
- *Provide the user with explanations of the conflicts*: The system has to answer questions like “From which subsets of current choices did inconsistency follow?” or “Why is this value not available any longer for this variable?”. The problem here is to identify (minimal) inconsistent subsets of the current set of choices.

Classical filtering algorithms, and specifically their dynamical versions (cf. [4,5,19]) may address the first functionality. Nevertheless, these algorithms cannot guarantee global consistency nor help in the computation of restorations. In order to fill this gap, this paper presents a new approach to interactive constraint solving that addresses both functionalities. From a formal point of view, this approach leads to an extension of the CSP framework to assumptions, that is presented in Section 2. This new framework can also be viewed as a generalization of the ATMS one [17] to general constraints. It enables various functionalities required by interactive constraint solving (as discussed above) to be formally specified. A complexity analysis of these computational tasks is reported. Not surprisingly, all of them are intractable in the worst case. This hardly contrasts with the practical requirements imposed by interactivity: to be viable, the response time of the algorithms must not exceed a few seconds.

In order to circumvent intractability from the practical side, our approach relies on a *compilation* of the original problem into a data structure from which much better performances can be obtained. We actually follow [41] and use an automaton that represents the set of solutions of the CSP. We show that the set of computational tasks that can be tractably achieved from such an automaton is not limited to consistency checking (as well as validity and equivalence as shown in [41]) but also includes more sophisticated tasks. Accordingly, the principles of Vempaty’s compilations are briefly recalled in Section 3. In Section 4, we explain how the automaton can be exploited to achieve in an efficient way other computational tasks considered in this paper. Some experimental results on a real, large scale, application are provided in Section 5. Section 6 concludes the paper. Proofs are reported in Appendix A.

2. Assumption-based CSPs

This section presents an extension of the standard CSP framework to the handling of a distinguished set of dynamic constraints. For the sake of generality, we do not restrict our framework to configuration problems and give in the following more generic definitions than strictly needed for this purpose. For instance, we will not assume that dynamic constraints are always unary ones.

2.1. Preliminary definitions and notations

A CSP is classically defined by a triplet $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ where $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$ is a finite set of variables, each X_i taking its values in a finite domain D_{X_i}, and a finite set of constraints $\mathcal{C}$. We note $\mathcal{D} = \{D_{X_1}, D_{X_2}, \ldots, D_{X_n}\}$. A constraint C in $\mathcal{C}$ is defined on a set of variables $V(C) \subseteq \mathcal{X}$ and restricts the combinations of values that can be taken by the variables of $V(C)$. Thus, a relation $R(C)$ on $V(C)$ can be associated with each C: it is the set of tuples that satisfy the constraint. In the following, $\neg C$ will denote the constraint on $V(C)$ that is satisfied by any tuple that violates C and violated by any tuple that satisfies C. An assignment s is an element of the cartesian product of domains, noted $\mathcal{D}^n$; it is a solution of the CSP if it satisfies all the constraints, i.e., if for any constraint C, the projection of s on $V(C)$ is an element of $R(C)$ (otherwise, s is said to falsify C). If such a solution exists, the CSP is said to be consistent, otherwise it is inconsistent.

Let us now distinguish a set of dynamic constraints, defining thus "Assumption-based CSPs" (A-CSPs):

Definition 1. An *A-CSP* Π is a 4-uple $\langle \mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{H} \rangle$ where $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ is a CSP and $\mathcal{H}$ a finite set of constraints on variables of $\mathcal{X}$.

In configuration problems, $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ represents a configurable product and $\mathcal{H}$ is the current set of user's choices (we also call them user's restrictions).

$\Pi' = \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup \mathcal{H} \rangle$ is the (classical) CSP associated with Π.

Definition 2. An assignment s is a solution of an A-CSP $\Pi = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{H} \rangle$ iff s is a solution of $\Pi' = \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup \mathcal{H} \rangle$.

$\mathcal{S}(\Pi)$ denotes the set of all solutions of Π. Π is consistent (respectively inconsistent) iff $\mathcal{S}(\Pi) \neq \emptyset$ (respectively $= \emptyset$).

At any time, the solutions of Π' correspond to the feasible products that obey the user's requirements. For each X_i, P_{X_i} denotes the projection of $\mathcal{S}(\Pi)$ on X_i: it is the set of all values of X_i the choice of which allows the definition of a feasible product that satisfies the requirements $\mathcal{H}$.

2.2. Conflicts and consistent environments

In the following, we refer to subsets $E \subseteq \mathcal{H}$ of user's choices as environments, whether or not they are consistent:

Definition 3. Let $\Pi = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{H} \rangle$ be an A-CSP.

A subset $E \subseteq \mathcal{H}$ is called an *environment*.

E is *consistent* (respectively *inconsistent*) given Π iff $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup E \rangle$ is a consistent (respectively inconsistent) CSP.

When no ambiguity is possible, slightly abusing words, we simply say that E is consistent (respectively inconsistent) whenever it is consistent (respectively inconsistent) given Π.

Any inconsistent environment is called a *conflict* for Π (or a conflict on $\mathcal{H}$ for $\mathcal{C}$). When $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ is consistent, a conflict can be understood as a cause of the inconsistency of the CSP $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup \mathcal{H} \rangle$ (i.e., of the incompatibility between the user's choices and the

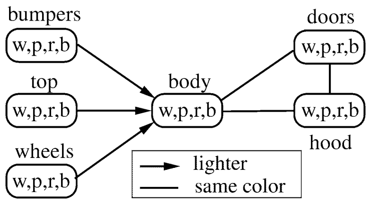


Fig. 1. Constraint graph of our toy example.

configuration constraints). Any consistent environment induces a subset of $\mathcal{H}$ that can be relaxed so as to recover consistency: if E is a consistent environment, relaxing $\mathcal{H} \setminus E$ is sufficient to recover consistency.

For instance, if $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ is consistent and $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup \mathcal{H} \rangle$ is inconsistent, $\mathcal{H}$ is a trivial conflict, $\emptyset$ a consistent environment and a dummy solution is obtained by relaxing all the constraints of $\mathcal{H}$. As this example shows, all the environments are not equally interesting in practice. More formally, in lack of additional information, consistent (respectively inconsistent) environments which are maximal (respectively minimal) with respect to set-theoretic inclusion are preferred:

Definition 4. A *nogood* for $\mathcal{C}$ on $\mathcal{H}$ (or nogood of Π) is a minimal inconsistent environment, i.e., a conflict E for Π such that no conflict E' for Π is such that $E' \subsetneq E$.

An *interpretation* of Π is an environment which is maximal consistent for Π, i.e., a consistent environment E for Π s.t. no consistent environment E' for Π is such that $E \subsetneq E'$.

Example 1.

- Initial CSP: $\mathcal{X} = \{bumpers, top, wheels, body, hood, doors\}$ with all the variable sharing the same initial domain: $\{white, pink, red, black\}$ (see Fig. 1). The constraints of $\mathcal{C}$ are the following:

$$\begin{aligned}
V(C_1) &= \{body, doors\}, \quad V(C_2) = \{hood, doors\},\\
V(C_3) &= \{body, hood\}, \quad V(C_4) = \{bumpers, body\},\\
V(C_5) &= \{top, body\}, \quad V(C_6) = \{wheels, body\},\\
R(C_1) &= R(C_2) = R(C_3) = \big\{(white, white), (pink, pink), (red, red),\\
&\qquad (black, black)\big\},\\
R(C_4) &= R(C_5) = R(C_6) = \big\{(white, pink), (white, red), (white, black),\\
&\qquad (pink, red), (pink, black), (red, black)\big\}.
\end{aligned}$$

- Set of assumptions: $\mathcal{H} = \{H_{bumper}, H_{top}, H_{wheels}, H_{body}, H_{hood}, H_{doors}\}$ with:

$$\begin{aligned}
&R(H_{bumpers}) = R(H_{top}) = \{white, pink\}, \quad R(H_{wheels}) = \{red\},\\
&R(H_{body}) = \{pink, red\}, \quad R(H_{doors}) = \{red, black\},\\
&R(H_{hood}) = \{pink, black\}.
\end{aligned}$$

- Conflicts: $\{H_{body}, H_{doors}, H_{hood}, H_{wheels}\}$, $\{H_{body}, H_{wheels}, H_{top}\}$, $\{H_{body}, H_{doors}, H_{hood}\}$, $\{H_{body}, H_{wheels}\}$, etc.
- Nogoods: $\{H_{body}, H_{doors}, H_{hood}\}$ and $\{H_{body}, H_{wheels}\}$.
- Consistent environments: $\{H_{bumpers}, H_{top}, H_{wheels}, H_{hood}\}$, $\{H_{bumpers}, H_{top}, H_{wheels}, H_{hood}, H_{doors}\}$, $\{H_{bumpers}, H_{top}, H_{body}\}$, $\{H_{bumpers}, H_{top}, H_{body}, H_{doors}\}$, $\{H_{bumpers}, H_{top}, H_{body}, H_{hood}\}$, etc.
- Interpretations: $\{H_{bumpers}, H_{top}, H_{wheels}, H_{hood}, H_{doors}\}$, $\{H_{bumpers}, H_{top}, H_{body}, H_{doors}\}$, $\{H_{bumpers}, H_{top}, H_{body}, H_{hood}\}$.

As in propositional logic, nogoods can be generated from interpretations through the computation of hitting sets, and the converse also holds:

Proposition 1. *Let $\mathcal{N}$ (respectively $\mathcal{I}$) be the set of the nogoods (respectively interpretations) of an A-CSP $\Pi = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{H} \rangle$:*

- *E is a consistent environment of $\Pi \Leftrightarrow \forall E_i \in \mathcal{N}, (\mathcal{H} \setminus E) \cap E_i \neq \emptyset$.*
- *E is a conflict of $\Pi \Leftrightarrow \forall E_i \in \mathcal{I}, E \cap (\mathcal{H} \setminus E_i) \neq \emptyset$.*

Corollary 1.

- *E is an interpretation of $\Pi \Leftrightarrow \mathcal{H} \setminus E$ is a hitting set of $\mathcal{N}$ minimal w.r.t. $\subseteq$.*
- *E is a nogood of $\Pi \Leftrightarrow E$ is a hitting set of $\bar{\mathcal{I}} = \{\mathcal{H} \setminus I \mid I \in \mathcal{I}\}$ minimal w.r.t. $\subseteq$.*

2.3. Explanations and restorations

Even if the current set of user's restrictions is not inconsistent given the CSP, these restrictions can lead to reject some values for other variables that the user would prefer as feasible. The system must thus provide the user with an explanation of these prohibitions. Let us generally define the notion of explanation as follows:

Definition 5. Let $\Pi = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{H} \rangle$ be an A-CSP and L a constraint on some variables of $\mathcal{X}$.

An *explanation* of L on Π is an environment E such that $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup E \rangle$ is a consistent CSP and $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup E \cup \{\neg L\} \rangle$ is an inconsistent one.

An explanation E of L on Π is *minimal* iff there exists no explanation E' of L on Π such that $E' \subsetneq E$.

Note that if $\langle \mathcal{X}, \mathcal{D}, \{L\} \rangle$ is inconsistent, then no explanation of L on Π exists. However, this situation is not very relevant since self-contradictory constraints are typically not considered.

In configuration problems, explanations of *unary* constraints are computed: determining why a set of values V is not available any longer for a given variable X_i amounts to compute the explanations of the constraint $L = $ "$X_i \in D_i \setminus V$".

Now, the user not only needs to understand why some values are forbidden for a variable, but also which previous choices should be kept and which previous choices should be relaxed in order to make these values available again. This is formalized thanks to the notion of restoration:

Definition 6. Let $\Pi = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{H} \rangle$ be an A-CSP and L a constraint on some variables of $\mathcal{X}$.

A *restoration* of L on Π is an environment E such that $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup E \cup \{L\} \rangle$ is a consistent CSP.

A restoration E of L on Π is maximal iff there is no restoration E' of L on Π such that $E \subsetneq E'$.

A restoration of L is thus a maximal subset of constraints of $\mathcal{H}$ that can be kept if one wishes to be consistent with L: L will be restored as soon as the constraints of $\mathcal{H} \setminus E$ are relaxed from the A-CSP.

Example 1 (*Continued*). Consider the CSP given in Example 1 with the following set of assumptions: $\mathcal{H} = \{H_{bumpers}, H_{top}, H_{wheels}, H_{doors}, H_{Hood}\}$ with:

$$R(H_{bumpers}) = R(H_{top}) = \{white, pink\}, \quad R(H_{wheels}) = \{red\},$$
$$R(H_{doors}) = \{white, red, black\}, \quad R(H_{hood}) = \{pink, black\}.$$

- The constraint L_1 such that $V(L_1) = \{body\}$ and $R(L_1) = \{red, white, black\}$ meaning "The body of the car cannot be pink" has two minimal explanations: $\{H_{wheels}\}$ and $\{H_{doors}\}$.
- The constraint L_2 such that $V(L_2) = \{body\}$ and $R(L_2) = \{black\}$ ("The body of the car must be black") has two minimal explanations: $\{H_{wheels}\}$ and $\{H_{hood}, H_{doors}\}$.
- To restore the value pink, one must at least relax both H_{wheels} and H_{doors}: $\{H_{bumpers}, H_{top}, H_{hood}\}$ and obviously $\{H_{bumpers}\}$ and $\{H_{top}\}$ are restorations of $L_3 = \neg L_1$ such that $V(L_3) = \{body\}$ and $R(L_3) = \{pink\}$, $\{H_{bumpers}, H_{top}, H_{hood}\}$ is maximal.

Restorations, explanations, interpretations and nogoods are related in the following way:

Proposition 2. *Let $\Pi = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{H} \rangle$ be an A-CSP.*
- *When $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup \mathcal{H} \rangle$ is consistent, E is an explanation* (*respectively minimal explanation*) *of L on Π iff E is a conflict* (*respectively nogood*) *of $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup \{\neg L\}, \mathcal{H} \rangle$.*
- *E is a restoration* (*respectively maximal restoration*) *of L on Π iff E is a consistent environment* (*respectively an interpretation*) *of $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup \{L\}, \mathcal{H} \rangle$.*

Accordingly, when $\mathcal{C} \cup \mathcal{H}$ is a consistent set of contraints, the restorations of L can be computed from the explanations of $\neg L$ and the converse also holds.

Proposition 3. *Let $\mathcal{E}_{\neg L}$ be the set of all minimal explanations of the constraint $\neg L$ on $\Pi = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{H} \rangle$ and let $\mathcal{R}_L$ be the set of all maximal restorations of L on Π. If $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup \mathcal{H} \rangle$ is consistent, then*:
- *E is a restoration of L on $\Pi \Leftrightarrow \forall E_i \in \mathcal{E}_{\neg L}, (\mathcal{H} \setminus E) \cap E_i \neq \emptyset$.*
- *E is an explanation of $\neg L$ on $\Pi \Leftrightarrow \forall E_i \in \mathcal{R}_L, E \cap (\mathcal{H} \setminus E_i) \neq \emptyset$.*

Corollary 2.

E is a restoration of L on Π $\Leftrightarrow$ $\mathcal{H} \setminus E$ is a hitting set of $\mathcal{E}_{\neg L}$ minimal w.r.t. $\subseteq$.

E is an explanation of $\neg L$ on Π $\Leftrightarrow$ E is hitting set of $\bar{\mathcal{R}}_L = \{\mathcal{H} \setminus I \mid I \in \mathcal{R}_L\}$ minimal w.r.t. $\subseteq$.

2.4. Preferences between assumptions

In many real applications, all the user's choices do not have the same importance but are subject to preferences. For instance, a requirement dealing with the type of engine can be more important than a requirement concerning the color of the body of the car. To allow the handling of such preferences, the framework of Valued CSP (VCSP) [36] associates with each constraint $H \in \mathcal{H}$ a degree of importance, or "valuation" $\phi(H)$, belonging to a totally ordered scale, e.g., in $\mathbb{N}^+ \cup \{+\infty\}$: the higher the valuation, the more important the constraint.

In the following, we suppose that every constraint has a positive importance (otherwise, it can be *a priori* discarded from $\mathcal{H}$). In this situation, the best consistent environments as well as the best restorations are those that relax as less important constraints as possible, i.e., that minimize the sum of valuations associated with the constraints they relax. In particular, when ϕ is the constant function 1, the best consistent environments are those that relax the lowest number of constraints. More formally:

Definition 7. Let $\Pi = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{H} \rangle$ be an A-CSP, ϕ a valuation of the assumptions, i.e., a positive application from $\mathcal{H}$ to $\mathbb{N}^+$ and for any $E \in \mathcal{H}$, let $\phi(E) = \sum_{H \in \mathcal{H} \setminus E} \phi(H)$.

A V-interpretation[2] of Π is an environment consistent given Π such that there is no environment E' consistent given Π satisfying $\phi(E') < \phi(E)$.

A *V-restoration* of L on Π is a restoration E of L on Π such that there is no restoration E' of L on Π satisfying $\phi(E') < \phi(E)$.

Clearly enough, the A-CSP framework enriched with such valuation functions is a natural framework for cost-based abduction [11,23] (and for probabilistic abduction [30]—through a *log* transformation and under some independence assumptions).

In some applications, it could be interesting to use other valuation functions as in the general VCSP model. In this model, any valuation structure $\langle \mathcal{L}, \succ, \oplus \rangle$ where $\mathcal{L}$ is a set totally ordered by $\succ$, with a minimal (respectively maximal) element noted $\bot$ (respectively $\top$) and $\oplus$ is a commutative, associative, closed binary operation that satisfies the properties of monotonicity and identity (i.e., a T-conorm) is acceptable. For simplicity, we restrict ourselves to the structure $\langle \mathbb{N} \cup \{+\infty\}, >, + \rangle$ with ϕ giving valuations different from $\top = +\infty$ and $\bot = 0$. This choice allows the representation of several policies for characterizing preferred environments, such as those based on cardinality or lexicographic ordering. It ensures that the valuation of each environment can be computed in polynomial time.

[2] When the elements of $\mathcal{H}$ are unary constraints and ϕ is the constant function 1, we recover here the notion of "minimal revision" first proposed by Dechter and Dechter [15].

Example 1 (*Continued*). Let us step back to Example 1 and suppose the following valuations of assumptions:

$$\phi(H_{wheels}) = \phi(H_{bumper}) = \phi(H_{top}) = \phi(H_{hood}) = 1,$$
$$\phi(H_{body}) = 2, \qquad \phi(H_{doors}) = 2.$$

We thus get for each interpretation:

$$\phi(\{H_{bumpers}, H_{top}, H_{wheels}, H_{hood}, H_{doors}\}) = 2,$$
$$\phi(\{H_{bumpers}, H_{top}, H_{body}, H_{doors}\}) = 2,$$
$$\phi(\{H_{bumpers}, H_{top}, H_{body}, H_{hood}\}) = 3.$$

This yields two V-interpretations:

$$\{H_{bumpers}, H_{top}, H_{wheels}, H_{hood}, H_{doors}\} \quad \text{and} \quad \{H_{bumpers}, H_{top}, H_{body}, H_{doors}\}.$$

Since ϕ assigns valuations different from 0 and $+\infty$ to assumptions, the pre-ordering it induces on environments is consistent with (and refines) the pre-ordering based on set inclusion:

Proposition 4. *Let $\Pi = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{H}\rangle$ be an A-CSP. Any V-interpretation of Π* (*respectively V-restoration of L on Π*) *is an interpretation* (*respectively a maximal restoration*).

As a consequence of Propositions 3 and 4, it is possible to compute the V-interpretations of Π from its nogoods and, when $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup \mathcal{H}\rangle$ is consistent, to compute the V-restorations of L on Π from its explanations through the computation of hitting sets (minimizing the sum of the valuations of the constraints kept in the hitting set rather that minimizing the set with respect to $\subseteq$). [3] But this will be of a poor help, since as we will see in Section 2.6, computing the nogoods of Π is not cheaper than computing its V-interpretations.

As suggested by Lobjois and Verfaillie [25], a more tractable approach to consistency restoration, and more generally, to constraint restoration, would be to use the machinery provided by VCSP, thanks to the following property:

Proposition 5. *Let $\Pi = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{H}\rangle$ be an A-CSP and ϕ be a valuation function of its assumptions, i.e., a positive application from $\mathcal{H}$ to $\mathbb{N}^+$. Let ϕ' be an application from $\mathcal{C} \cup \mathcal{H}$ to $\mathbb{N} \cup \{+\infty\}$ defined by*: *$\forall C \in \mathcal{H}$, $\phi'(C) = \phi(C)$ and $\forall C \in \mathcal{C}$, $\phi'(C) = +\infty$. The following statements are equivalent*:

- *E is a V-interpretation of Π.*
- *$\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup E\rangle$ is minimal relaxation of the valued CSP $(\mathcal{X}, \mathcal{D}, \mathcal{C} \cup \mathcal{H}, \langle \mathbb{N} \cup \{+\infty\}, >, +\rangle, \phi')$ and its valuation differs from $+\infty$.*

Thus, the computation of V-interpretations (and, thanks to Proposition 2, of V-restorations) can be achieved as a VCSP optimization problem: it leads to the computation of all the solutions of a VCSP.

[3] However, every nogood of Π cannot be computed from its V-interpretations, only: it is not sufficient to cover every V-interpretation to cover all the interpretations.

Another way of computing V-interpretations is provided by Dechter and Dechter [14, 15]. This approach is restricted to tree-structured CSPs (it relies on the hypothesis that $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup \mathcal{H} \rangle$ is a hypertree of constraints) and proceeds by a forward propagation of the valuations followed by a top-down (and backtrack free) search of the best interpretations.

2.5. Application to configuration

In the context of a constraint-based interactive approach to product configuration, an A-CSP Π can be used to represent both the feasible products (as the solutions of the CSP $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$) and the current user's choices as a set $\mathcal{H}$ of dynamic constraints/assumptions. This A-CSP has the following noticeable properties:

- Since each solution of $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ corresponds to a feasible product, we can reasonably suppose that $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ is consistent.
- Since the user's elementary choices typically consist in assigning one value to one variable, each element of $\mathcal{H}$ is a *unary constraint*.
- The user's choices are not self-contradictory, i.e., $\langle \mathcal{X}, \mathcal{D}, \mathcal{H} \rangle$ is a consistent CSP. Obviously, this does not mean that the user's choices always lead to a feasible product ($\mathcal{C}$ is not taken into account here).
- The preferences of the user are expressed by means of a valuation function, i.e., an application ϕ from $\mathcal{H}$ to $\mathbb{N}^+$. Accordingly, every dynamic constraint has a positive importance (otherwise, it can be *a priori* discarded from the set of constraints) and none of the dynamic constraints is mandatory.
- It is always possible to realize any kind of pre-computing or compilation on $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$, off-line, i.e., before the configuration phase.

In order to help the user in his configuration task, our aim is to develop a system that achieves the following functionalities:

- Detection of inconsistency: at any time, the system must tell whether there is a feasible product that satisfies all the user's requirements, i.e., it must be able to check whether $\Pi = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{H} \rangle$ is consistent or not.
- Maintenance of global consistency: at any time, the system must discard from the domains of available values those that cannot lead to a feasible product, i.e., it must be able to compute P_{X_i} for any i.
- In case of inconsistency, computation of nogoods and, more importantly, of interpretations and V-interpretations must be offered.
- When some interesting values become forbidden for a variable, computation of minimal explanations and, more importantly, computation of V-restorations of these values must be possible.

2.6. Complexity issues

Two factors influence the computational complexity of searching for nogoods, interpretations, explanations and restorations in assumption-based CSPs:

- The *number* of objects, i.e., the size of the *result*. The number of nogoods, explanations, interpretations and restorations is exponential in the size of $\mathcal{H}$ in the worst case.

Table 1
Complexity results (Π as input stands for $\langle \mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{H} \rangle$)

Input	Question	Complexity
Π, E	is E a conflict?	coNP-complete
Π, E	is E a nogood?	BH_2-complete
Π	does there exist a conflict? (respectively a nogood)	coNP-complete
Π, E	is E a consistent environment?	NP-complete
Π, E	is E an interpretation?	BH_2-complete
Π	does there exist a consistent environment? (respectively an interpretation)	NP-complete
Π, E, L	is E an explanation of L?	BH_2-complete
Π, E, L	is E a minimal explanation of L?	BH_2-complete
Π, L	does there exist an explanation of $\neg L$? (respectively a minimal explanation of $\neg L$)	Σ_2^p-complete
Π, E, L	is E a restoration of L?	NP-complete
Π, E, L	is E a maximal restoration of L?	BH_2-complete
Π, L	does there exist an a restoration of L? (respectively a maximal restoration of L)	NP-complete

- The complexity of recognizing these objects (e.g., testing whether a given environment E is a nogood for a given A-CSP) or of testing their existence (e.g., does there exist a nogood for a given A-CSP?).

Table 1 presents some results related to this second source of complexity (see the proofs in Appendix A). These results slightly generalize corresponding results obtained so far in the ATMS framework (some of them are close to results given in [20]). These results argue in favor of hard search problems in the worst case; this is not surprising since one of the key issues (consistency) is already intractable. Note that the complexity results that are reported are concerned with the general case, i.e., no specific assumption related to the configuration issue is made.

One should notice that the computation of interpretations is in the worst case no more expensive than the computation of nogoods. This means that computing interpretations from nogoods through hitting sets is not a good approach, unless nogoods can be obtained at cheap cost.

In the restricted situation where the A-CSP framework is applied to interactive configuration, some additional assumptions can be made (see Section 2.5): $\mathcal{C}$ is known to be consistent, the constraints of $\mathcal{H}$ are unary, as well as the constraints that must be explained. In this case, all the complexity proofs remain valid, except those concerning existence problems which become trivial. Indeed, since the consistency restoration feature as well as the explanation of inconsistency cannot be invoked unless inconsistency has been detected, there always is a consistent environment ($\emptyset$) and a conflict ($\mathcal{H}$). Similarly, searching for explanations of a unary constraint $\neg L$ or for restorations of L cannot be invoked when $\neg L$ is not a consequence of the current set of choices: $\mathcal{H}$ is a trivial explanation of $\neg L$ and $\emptyset$ is a trivial restoration of L. However, this does not decrease the difficulty of the problem of searching for good or optimal environments, nor the theoretical complexity of the other decision problems, except the one concerning the test of E as an explanation of a given L. This problem becomes "only" coNP-complete: since $\mathcal{C} \cup \mathcal{H}$ is known to be consistent, so is also $\mathcal{C} \cup E$, whatever $E \subseteq \mathcal{H}$.

Finally, while the existence of preferences modeled by a valuation function ϕ restricts in practice the number of preferred assumptions (cf. Proposition 4), this number remains exponential in the input size in the worst case. Besides, its exploitation does not make the tasks tractable: as a consequence of Proposition 5, the corresponding problems are at least as difficult as decision problems associated to the optimisation of valued CSPs, i.e., they are NP-hard (in the sense of Cook reduction).

3. Compilation of an A-CSP

According to the complexity results given in the previous section, the computational tasks an A-CSP must achieve are highly combinatorial even under the simplifying assumptions associated to configuration problems. On the other hand, interactivity impose some contraints over the response time, which must not exceed a few seconds. That is why we suggest to push the computational effort required by the handling of the on-line requests into an off-line compilation phase.

The compilation of propositional knowledge bases has already been intensively investigated (see [8] for a survey) and many compilation functions have been proposed so far (see, e.g., [6,13,18,26,31,37,38]). Compilation has already been applied to CSPs as well [16,28,41,42]. Following these works, we propose to compile the set of solutions of the CSP under a form from which the computational tasks can be achieved more quickly than from the original formulation.

The data structure used here is the one proposed by Vempaty [41], namely a finite-state automaton that represents the set of solutions of $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$. This concise representation of the solutions can be logically understood as a compact, structured Disjunctive Normal Form of the CSP. Vempaty [41] shows how consistency, validity and equivalence of CSPs can be tested efficiently when represented as automata. For our purpose, the main interest of this compiled form is that both interpretations and restorations can be generated from the compiled form in time linear *in its size*. Notice that this compilation is done only once, although several successive requests will be typically addressed in an interactive configuration situation, and that the system can be used in several configuration situations without requiring any re-compilation.

This approach relies on a few hypotheses that are satisfied in practice by configuration problems. First, the persistence of $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ and the possibility of performing off-line any computation on it, before the introduction of dynamic constraints; then, the restriction of the dynamic part of the problem, $\mathcal{H}$, to unary constraints; finally, the structure of the set of solutions of $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ that allows its practical representation under the form of an automaton—its size, exponential in the worst case, can be reasonable in practice. In the case of configurable products, the satisfaction of this last requirement is due to two facts: the interchangeability of many values and the structure of the product into subproducts that are more or less independent from each other.

3.1. Compilation of a CSP in the form of an automaton

Let us recall the keys points of the method introduced by Vempaty [41] to represent the set of solutions of a CSP as an automaton.

Automata. We consider here the *state diagrams* of automata: a finite-state automaton (FA) $\mathcal{A}$ on an alphabet Σ is a oriented digraph the edges of which are labeled by elements of Σ (the *transitions* of the FA). The finite set of its nodes (or *states* of the FA) is denoted Q. It has one *initial state* denoted I and at least one *final state* F and is such that any transition and any state belongs to a least one path for the initial state to a final state. A word $m = a_1a_2a_3 \dots a_n$ is recognized by the automaton if there exists a path from the initial state I to a final state F with the label m: $I \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \cdots \xrightarrow{a_n} F$. The language recognized by $\mathcal{A}$ is the set $\mathcal{L}(\mathcal{A})$ of the words it recognizes. A FA is deterministic (DFA) iff all the transitions coming from the same state have different labels.

Associating FA with CSP. Let $\Pi = \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ be a CSP. Given a permutation $O = [X_1, X_2, \dots, X_n]$ of $\mathcal{X}$, any solution of Π defines a word of length n over the alphabet $\mathcal{D}$. Hence, the set of solutions of Π defines a language over $\mathcal{D}$. This language, called the solution language of Π w.r.t. O and denoted $\mathcal{S}_O(\Pi)$, is a rational language. It is thus possible to represent $\mathcal{S}_O(\Pi)$ by a FA $\mathcal{A}$. This automaton has only one final state (noted F) and is such that the length of any path from I to F is n.

For any path $p = (I = q_1 \xrightarrow{a_1} q_2 \xrightarrow{a_2} q_3 \xrightarrow{a_3} \cdots \xrightarrow{a_n} F = q_{n+1})$ from I to F, for any state q_i in p, we write $\mathit{var}(q_i) = i$ (as soon as q_i is any state but F, $\mathit{var}(q_i)$ is the index of a variable in $\mathcal{X}$). For any transition a of p, $\mathit{in}(a)$ is its initial node and $\mathit{out}(a)$ is its terminal node; $\mathit{var}(a)$ is the variable $X_{\mathit{var}(\mathit{in}(a))}$ and $\mathit{val}(a)$ the label of a. Using these notations, $(\mathit{val}(a_1), \mathit{val}(a_2), \dots, \mathit{val}(a_n))$ is thus an assignment of $(\mathit{var}(a_1), \mathit{var}(a_2), \dots, \mathit{var}(a_n))$ and a solution of Π.

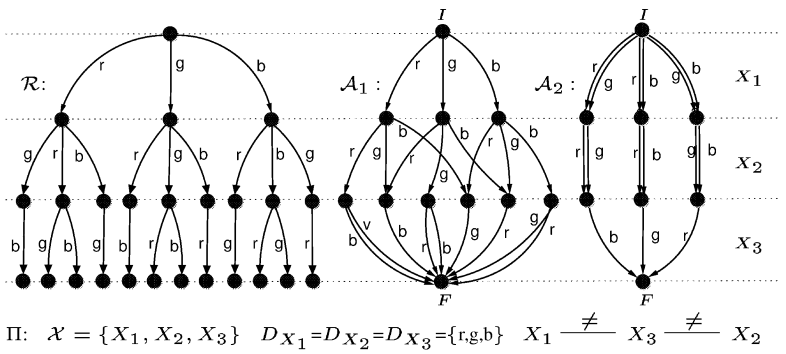
Definition 8. Let Π be a CSP, and $\mathcal{A}$ an automaton representing its set of solutions.

A transition a of $\mathcal{A}$ is said to *support* the value d for X_i (i.e., the assignment $X_i := d$) iff $\mathit{var}(a) = X_i$ and $\mathit{val}(a) = d$.

A transition a of $\mathcal{A}$ is said to support a unary constraint L on X_i iff it supports at least one of the values belonging to $R(L)$.

A path of $\mathcal{A}$ supports a value for a variable (respectively a unary constraint) iff it contains a transition supporting this value (respectively this unary constraint).

Computing the automaton associated with a CSP. According to [41], the automaton can be generated by using standard operators on automata: either by any CSP algorithm that enumerates the set of solutions and adds them successively to the automaton, or by composition operators on (small) automata representing the configuration constraints. The



$\mathcal{R}$:
$\mathcal{A}_1$:
$\mathcal{A}_2$:
I
F
X_1
X_2
X_3
Π: $\mathcal{X} = \{X_1, X_2, X_3\}$ $D_{X_1}=D_{X_2}=D_{X_3}=\{r,g,b\}$ $X_1 \xrightarrow{\neq} X_3 \xrightarrow{\neq} X_2$

Fig. 2. DFA $\mathcal{A}_1$ and FA $\mathcal{A}_1$ associated with Π for $O = [X_1, X_2, X_3]$ are concise representations of the set of solutions.

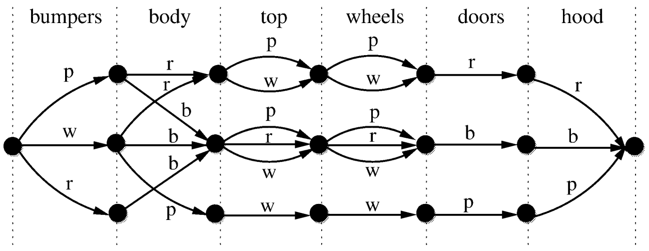


Fig. 3. An automaton associated with the CSP of Fig. 1.

first method is typically not tractable when configuration problems are considered, because of the huge number of configurable products (about 10^{12} in the application presented in Section 5). That is why the second method has been preferred here.

The complexity of the computation of the automaton obviously depends on its size (which is, in the worst case, exponential in the size of the original CSP). This size is mainly influenced by (i) the order with which the variables of the CSP are taken into account and (ii) the algorithm used to built it. Vempaty proposed to build a DFA that minimizes the number of states (MDFA). It is also possible to consider other minimization algorithms generating smaller automata (e.g., non-deterministic ones).

3.2. A-CSP with valuations and weighted automata

In configuration problems, $\mathcal{H}$ represents the user's choices. Each element of $\mathcal{H}$ is a unary constraint H_{X_i} on a variable X_i and its importance is given by $\phi(H_{X_i})$; taking this constraint into account leads to remove some assignments of X_i, and thus some of the transitions of the automaton, namely any transition a such that $var(a) = X_i$ and $val(a) \notin R(H_{X_i})$. The principle of our approach is to associate a cost to each transition: $\phi(a) = \phi(H_{X_i})$ if a corresponds to an assignment forbidden by a restriction H_{X_i}, $\phi(a) = 0$ otherwise. We can thus define the cost of a path as follows:

Definition 9.

- $cost(p) = \sum_{a \in p} \phi(a)$ is the global cost of path p;
- $cost(a)$ is the cost of a best (i.e., minimal cost) path from I to F through transition a;
- $cost(q)$ is the cost of a best path from I to F through state q.

Now, since every constraint of $\mathcal{H}$ has a positive valuation, any path from I to F of cost zero does not violate any of the constraints of $\mathcal{H}$ and thus defines a feasible product satisfying all user's requirements. Conversely, any path of positive cost corresponds to a product that violates at least one of the restrictions of $\mathcal{H}$. Knowing the set of paths with a zero cost is thus equivalent to knowing the set of solutions of $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup \mathcal{H} \rangle$. More formally:

Proposition 6. *Let $\Pi = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{H} \rangle$ be an A-CSP, ϕ a (positive) valuation function, $\mathcal{A}$ a weighted automaton representing Π and L a unary constraint. It holds that*:

(a) *Any path p from I to F corresponds to a consistent environment E such that $\phi(E) = cost(p)$.*
(b) *If E is a consistent environment, then there exists a path p from I to F such that $cost(p) \leqslant \phi(E)$.*
(c) *Any path p from I to F that supports L corresponds to a restoration E of L on Π such that $\phi(E) = cost(p)$.*
(d) *If E is a restoration of L on Π, then there exists a path p from I to F that supports L such that $cost(p) \leqslant \phi(E)$.*

Proposition 6 gives a formulation in terms of optimal paths for most of the requests identified in Section 2.5. It shows that:

Proposition 7.

(a) *Every minimal path from I to F represents a V-optimal interpretation of Π and conversely, to any V-optimal interpretation corresponds at least one minimal path.*
(b) *$\mathcal{H}$ is a conflict for $\Pi \Leftrightarrow$ there is no path of cost zero from I to F.*
(c) *Let P_{X_i} denote the projection of the solutions set $\mathcal{S}(\Pi)$ on variable X_i: $d \in P_{X_i} \Leftrightarrow$ there is a transition a such that $var(a) = X_i$, $val(a) = d$ and a belongs to a path of cost zero from I to F.*
(d) *Let L be a unary constraint on $X_i \in \mathcal{X}$, A_L the set of transitions that support it and A_L^* the cheapest of them ($A_L^* = \{a \in A_L \mid cost(a) = Min_{a' \in A_L} cost(a')\}$). Each transition in A_L^* corresponds to a V-restoration of L on Π and to any V-restoration of L on Π corresponds at least one transition in A_L^*.*

Example 1 (*Continued*). Let us step back to Example 1:

- Initial CSP: $\mathcal{X} = \{bumpers, top, wheels, body, hood, doors\}$ with all the variable sharing the same initial domain: $\{white, pink, red, black\}$. The constraints of $\mathcal{C}$ are the following:

$$\begin{aligned} V(C_1) &= \{body, doors\},\ V(C_2) = \{hood, doors\},\ V(C_3) = \{body, hood\}, \\ V(C_4) &= \{bumpers, body\},\ V(C_5) = \{top, body\},\ V(C_6) = \{wheels, body\}, \end{aligned}$$

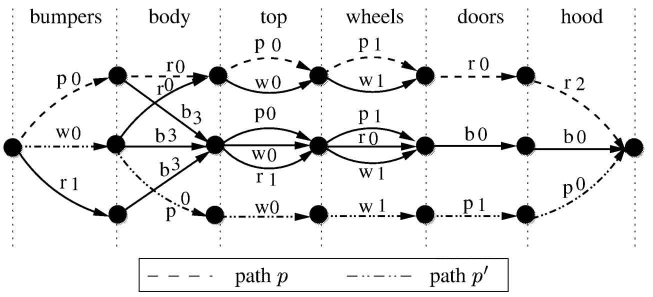


Fig. 4. A weighted automaton associated with the CSP of Fig. 1.

$$\begin{aligned} R(C_1) &= R(C_2) = R(C_3) = \{(\mathit{white}, \mathit{white})\}, \\ R(C_4) &= R(C_5) = R(C_6) = \big\{(\mathit{white}, \mathit{pink}), (\mathit{white}, \mathit{red}), (\mathit{white}, \mathit{black}), \\ &\qquad (\mathit{pink}, \mathit{red}), (\mathit{pink}, \mathit{black}), (\mathit{red}, \mathit{black})\big\}. \end{aligned}$$

- Set of assumptions: $\mathcal{H} = \{H_{bumper}, H_{top}, H_{wheels}, H_{body}, H_{hood}, H_{doors}\}$ with:

$$\begin{aligned} &R(H_{bumpers}) = R(H_{top}) = \{\mathit{white}, \mathit{pink}\}, && R(H_{wheels}) = \{\mathit{red}\}, \\ &R(H_{body}) = \{\mathit{pink}, \mathit{red}\}, && R(H_{doors}) = \{\mathit{red}, \mathit{black}\}, \\ &R(H_{hood}) = \{\mathit{pink}, \mathit{black}\}, \end{aligned}$$

with the following valuations:

$$\begin{aligned} &\phi(H_{hood}) = 2, \qquad \phi(H_{body}) = 3, \\ &\phi(H_{bumpers}) = \phi(H_{top}) = \phi(H_{wheels}) = \phi(H_{doors}) = 1. \end{aligned}$$

The weighted automaton encoding the set of solutions of the initial CSP and the set of dynamical constraints $\mathcal{H}$ is depicted Fig. 4.

The path p with $cost(p) = 3$ corresponds to the assignment $(\mathit{bumpers} = \mathit{pink},\ \mathit{body} = \mathit{red},\ \mathit{top} = \mathit{pink},\ \mathit{wheels} = \mathit{pink},\ \mathit{doors} = \mathit{red},\ \mathit{hood} = \mathit{red})$. It represents the interpretation $E = \{H_{bumpers}, H_{top}, H_{body}, H_{door}\}$ $(\phi(E) = 3)$ where H_{wheels} and H_{hood} are relaxed.

The path p' is minimal. It corresponds to the assignment $(\mathit{bumpers} = \mathit{white},\ \mathit{body} = \mathit{pink},\ \mathit{top} = \mathit{white},\ \mathit{wheels} = \mathit{white},\ \mathit{doors} = \mathit{pink},\ \mathit{hood} = \mathit{pink})$ that has a cost of 2. It represents the interpretation $E' = \{H_{bumpers}, H_{top}, H_{body}, H_{hood}\}$ $(\phi(E') = 2)$ where H_{doors} and H_{wheels} are relaxed.

Eight paths represent the preferred restorations of the value *red* for the variable *body*. They share the same cost (3) and each of them corresponds to the restoration $E'' = \{H_{bumpers}, H_{top}, H_{body}, H_{doors}\}$ $(\phi(E'') = 3)$.

4. Algorithms

Let us now explain how we can take advantage of such an automaton to achieve in an efficient way the various computational tasks we are interested in. As a consequence of

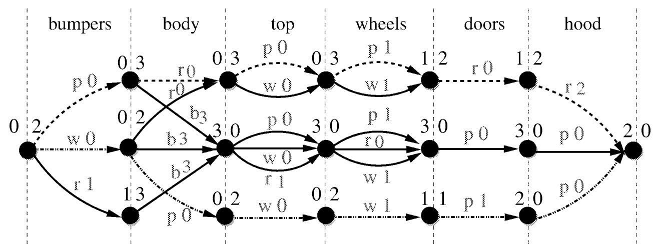


Fig. 5. Left and right costs associated to the nodes of the automaton.

the previous propositions, these tasks amount to the search of some minimal paths in the automaton. In order to allow an efficient computation of these minimal paths, we attach to any state q of the automaton a left cost $c_l(q)$ and a right cost $c_r(q)$ defined as follows (see also Fig. 5):

Definition 10.

- Let $a = (q', q)$. $c_l(q) = Min_{a,out(a)=q} c_l(q') + \phi(a)$ is the minimal cost of the paths from I to q;
- Let $a = (q, q')$. $c_r(q) = Min_{a,in(a)=q} c_r(q') + \phi(a)$ is the minimal cost of the paths from q to F;
- $c_l(I) = c_r(F) = 0$.

The cost $cost(q)$ of a minimal path from I to F through q, and the cost $cost(a)$ of a minimal path from I to F through a can be directly deduced from these two scores:

Proposition 8.

- $cost(q) = c_l(q) + c_r(q)$;
- $cost(a) = c_l(in(a)) + \phi(a) + c_r(out(a))$.

Finally, in order to ensure an efficient computation of the sets P_{X_i}, we maintain a counter $cnt(X_i, d)$ for any pair variable-value (X_i, d). $cnt(X_i, d)$ is the number of transitions that (i) support the assignment and (ii) belong to a path of cost 0 from I to F:

Definition 11.

$$cnt(X_i, d) = Card(\{a \mid var(a) = X_i \text{ and } val(a) = d \text{ and } cost(a) = 0\}).$$

4.1. Adding and deleting restrictions

Suppose that the user adds to $\mathcal{H}$ a unary constraint H_i on X_i with the valuation $\phi(H_i)$. Some of the transitions may become penalized, namely the transitions a such that $var(a) = X_i$ and $val(a) \notin R(H_i)$. It is thus necessary to update the costs of these transitions: for instance, when no other constraint of $\mathcal{H}$ restricts X_i, the cost of these transitions must rise from 0 to $\phi(H_i)$. Conversely, if the user relaxes H_i and deletes it from $\mathcal{H}$, some of the penalized transitions become fully allowed, namely the transitions

a such that $var(a) = X_i$ and $val(a) \notin R(H_i)$. Especially, when no other constraint of $\mathcal{H}$ restricts X_i, their costs come down to 0. In both cases, the modification of the cost of a transition a has to be propagated back (to update $c_r(\,)$) and forward (to update $c_l(\,)$). This can be done in three steps:

(1) Determination of the transitions such that $var(a) = X_i$ and labeled by a value $val(a)$ that does not belong to $R(H_i)$.
(2) Backward propagation from the right state of each of these transitions to the initial state: this is done through a breadth-first search strategy, so as to update the right costs $c_r(\,)$ of the traversed states knowing the right costs of their successors.
(3) Forward propagation from the left states of these transitions to the final state: it is also done with a breadth-first search strategy, so as to update the left costs $c_l()$ of the traversed states knowing the left costs of their successors.

The *cnt*() counters are maintained at steps (2) and (3): for any transition a encountered, $cnt(var(a), val(a))$ is incremented if $cost(a)$ rises from 0 to a positive value (a is no more one of the transition of cost 0 that supports the assignment $var(a) = val(a)$) and conversely, it is decremented if $cost(a)$ comes down to 0.

This algorithm runs in time polynomial in the size of the automaton. Linear implementations can be achieved, relying on a judicious choice of the data structure that encodes the automaton. The practical efficiency of this kind of algorithm can obviously be enhanced, for instance by propagating the modifications only (in practice, the propagation has seldom to reach the extremities of the automaton).

4.2. Detection of inconsistency

Maintaining the costs in the automaton allows us to determine at any time whether the current set of assumptions (i.e., the current user's choices) is consistent with the initial CSP. Indeed:

Proposition 9.
(a) *$c_r(I) = c_l(F)$ is the cost of a minimal path from I to F.*
(b) *$\mathcal{H}$ is a conflict iff $c_r(I) > 0$ (iff $c_l(F) > 0$).*

4.3. Maintenance of global consistency

According to Proposition 7(c), P_{X_i} can be derived from the automaton. It can actually be computed in linear time (linear in the size of D_{X_i}) using the counters *cnt*( ):

Proposition 10. $P_{X_i} = \{d \mid cnt(X_i, d) \neq 0\}$.

4.4. Computing V-interpretations

Whenever an inconsistency occurs, the system must provide the user with V-interpretations: they correspond to the minimal paths from I to F (cf. Proposition 7(a)). These optimal paths are easily obtained, going from I to F through "optimal" states, thanks to the following property:

Proposition 11.

(a) *A path p from q to F is of minimal cost among the paths from q to F iff for any transition a in p, $c_r(in(a)) = \phi(a) + c_r(out(a))$.*

(b) *A path p from I to q is of minimal cost among the paths from I to q iff for any transition a in p, $c_l(out(a)) = \phi(a) + c_l(in(a))$.*

Corollary 3. *A path p from I to F is of minimal cost iff, for any transition a in p, $c_r(in(a)) = \phi(a) + c_r(out(a))$ and $c_l(out(a)) = \phi(a) + c_l(in(a))$.*

Computing a *unique* V-optimal interpretation can be done in polynomial time: since it amounts to following an optimal path from I to F according to the marks $c_r()$, it is bounded by the number of variables of the original CSP and the maximal number of successors of a state, i.e., the size of the domains. The search for *all* V-interpretations is more or less equivalent to the enumeration of all minimal paths of the automaton, the difficulty being that a given interpretation can be represented by more than one path. Two different methods can be proposed for the enumeration of the V-interpretations. The first one is an adaptation of the DPI algorithm [9] that develops the tree of the V-interpretations *without any backtrack due to a failure*. Indeed, the costs of the states are used to determine whether a branch is optimal or not. The sketch of algorithm that follows is a simplified version that assumes that at most one constraint of $\mathcal{H}$ restricts a given variable X_i.

```
// In(a) (respectively Out(a)) is the initial node (respec-
tively the terminal node) of transition a.
function optimal($a$)
  return ($c_r(In(a)) == \phi(a) + c_r(Out(a))$)

procedure Develop($i$, $list$, $E$)
  if ($list = \emptyset$)
    return false
  else if $i > n$
    memorize $E$ (it is a $V$-optimal interpretation)
    return false                                          (1)
  else
    $Q_{KeepH_i} \longleftarrow \{\}$
    $Q_{RelaxH_i} \longleftarrow \{\}$
    forall state $s$ in $list$
      forall transition $a \in Out(s)$ such as optimal($a$) do
        if $\phi(a) > 0$
          add $Out(a)$ to $Q_{RelaxH_i}$
        else
          add $Out(a)$ to $Q_{KeepH_i}$
   // Search for the interpretations that do not relax $H_i$
  if Develop($i+1$, $Q_{KeepH_i}$, $E \cup \{H_i\}$)            (2)
    return true
  else
   // Search for the interpretations that relax $H_i$
    return Develop($i+1$, $Q_{RelaxH_i}$, $E$)                (3)
```

The search for all interpretations is invoked by Develop(1, (I), ∅). Several steps of the algorithm deserve some comments. At line (1), the search can be halted as soon as some criteria are fulfilled, for instance when a given number of V-interpretations have been reached: it is enough to return the value "true" instead of "false". At lines (2) and (3), the two branches of the search tree are developed. A more interactive solution should be to ask the user whether the branch "Relax H_i" should be developed or not. The important point is that a branch is entered (or proposed to the user) iff it is guaranteed that it leads to a V-interpretation.

The other possible method is breadth searching the automaton, from I to F through the optimal paths. The principle is to label every state met with the sets of elements of $\mathcal{H}$ which must be relaxed to reach the state: we thus get on F the complementary sets (to $\mathcal{H}$) of the V-interpretations. The most expensive operation here is updating the sets attached to states, since unioning sets is necessary at each update operation.

Anyway, for both methods, our representation by an automaton allows a computation of all V-interpretations that is polynomial in the size of the result.

4.5. Computing V-restorations

The automaton also provides a way to efficiently compute the V-optimal restorations of a unary constraint L: such restorations correspond to the cheapest paths among those that go from I to F and that support L. If the user is interested in a *unique* restoration of a set of values for a variable it is sufficient to find, among those supporting these values, a transition $a = (q, q')$ that minimizes *cost*( ). The corresponding V-optimal restoration is obtained going backward from q to I and then forward from q' to F through an optimal path, thanks to the following corollary of Proposition 11:

Corollary 4. *A path p from I to F that contains transition a is of minimal cost among the paths from I to F that contains this transition iff*

- *for any a' in p such that $var(a') \leqslant var(a)$,*

$$c_l(out(a')) = c_l(in(a')) + \phi(a');$$

- *for any a' in p such that $var(a) \geqslant var(a')$,*

$$c_r(in(a)) = c_r(out(a')) + \phi(a').$$

Computing a unique V-optimal restoration can thus be done in polynomial time. Again, a judicious choice of the encoding of the automaton can lead to a more efficient implementation (with a running time linearly bounded by the number of transitions that support the unary constraint plus the product of the number of variables of the CSP by the maximal number of successors of a state, i.e., the size of the domains).

Now, in order to compute the set of all V-optimal restorations of the unary constraint L, we can re-use the principles presented in the previous section. The idea is to mark, among those supporting L, all the transitions a that minimize *cost*(). The optimal paths can be marked using Corollary 4. It is then possible to take advantage of any of the methods given in Section 4.4, running it not on the whole automaton but on the set of marked paths only. The worst computational cost for the generation of all V-restorations is again polynomially bounded in the size of the result.

5. An empirical evaluation

The algorithms given in the present paper have been implemented and tested on a quite huge benchmark coming from a real application in car configuration. All experiments have been done on a Sun Sparc5 with 128 Mb of RAM using C++.

5.1. The test problem

The test problem has been provided by Renault DVI, a french car manufactoring company, and it deals with the configuration of a specific family of cars, called Renault Megane. In this problem, decision variables represent the type of engine, the country, options like air cooling, etc. The full characteristics of the problem are the following ones:[4]

- There are 101 variables. The sizes of their domains vary from 2 to 43 (there are actually 5 Boolean variables, 32 variables with a domain size between 3 and 5 and 13 variables with a domain size greater than 5).
- Expressed in a brute form, the CSP involves 858 constraints. Some of them can be merged and a more compact set of 113 constraints is obtained.
- When the 113 constraints of the CSP are expressed by extensive relations (sets of tuples), the file that describes the problem needs 6.6 Mb.
- The number of solutions of this CSP is 1 418 701 950 016.

5.2. The automaton

Our strategy to generate the automaton mainly follows the principles described by Vempaty by combination of (small) automata representing the constraints. The automaton has been computed in 2h01mn, which is actually acceptable since compilation is an off-line process. As a variable ordering heuristic, the first variables are those constrained by the most restrictive constraints. The size of the resulting automaton is also very satisfying: 236 160 states and 306 809 transitions, which can be described in a file of 3.4 Mb. Interestingly, this size is lower than the 6.6 Mb needed to store the original CSP (i.e., the 113 extensive relations).

As a matter of comparison, the same test problem has been compiled under the form of an OBDD starting from a Boolean constraints representation. Bryant's package [1,2] has been used with the same variable heuristic as previously. The resulting OBDD is significantly larger than the automaton: it contains 4 104 576 states and thus 8 209 148 transitions (which represents a file of 29.5 Mb).

[4] This problem is available at ftp://ftp.irit.fr/pub/IRIT/RPDMP/Configuration/.

5.3. The experimental protocol

In order to test the efficiency of the algorithms proposed in Section 4, we have "simulated" the behaviour of a user configuring a car. This simulation proceeds in several steps:

- A value is assigned to each variable following a random ordering of the variables. In any case, the valuation associated to the new user's restriction is $\phi = 1$. Global consistency is maintained after each assignment (see Sections 4.1 and 4.3) and the CPU time used to this updating of the automaton is measured. Notice that the value assigned to a variable is always (randomly) chosen among the values that are not ruled out by the propagation process. When only one value is available, the automaton does not need to be updated and no measurement is made.
- Once the car has been fully specified, we test the computation of the V-optimal restorations (i.e., the restorations of minimal cardinality, since $\phi = 1$): for each variable, for each of the values ruled out by propagation, the system is asked for all the optimal restorations and the CPU time is measured.
- Finally, the global consistency maintenance is tested upon the unassignment of variables: the constraints previously posted are deleted backward according to another (random) ordering of the variables and the CPU time used for updating the automaton is measured at each deletion. When the variable is not constrained, the automaton does not need to be updated and no measurement is made.

The results presented in the next section have been obtained from a set of 20,000 simulations, i.e., 1,051,950 calls to the global consistency maintaining algorithm (525,975 calls by assignment + 525,975 calls by deletion) and 5,021,098 computations of restoration.

5.4. Experimental results

Concerning the maintainance of global consistency, updating the weights of the automaton needed an average CPU time of 0.13 sec (standard deviation 0.163 sec, median 0.05 sec). Interestingly, the worst case observed only needed 0.68 sec. The cumulative distribution function of the time needed to update the automaton is given in Table 2. Notice that the performance does not depend on the event that calls the updating (i.e., assignment or unassignment of variables).

The computation of restorations also revealed itself as really efficient: the call for the computation of the set of optimal restorations needed an average CPU time of 0.015 sec (standard deviation 0.023 sec, median 0.01 sec). The worst case observed took 1.05 sec only. The cumulative distribution function of the time needed to compute the sets of restoration is given in Table 3.

Another question that may be addressed is the size of output, i.e., the size of the restorations provided at each call and above all, the number of restorations provided (recall that this number may be theoretically exponential). On our sample set, the average number of optimal restorations is 1.31 (standard deviation 0.7, median 1) and the maximal number of restorations provided is 32 (the minimum is obviously 1). The restorations provided have an average cardinality of 3 constraints (standard deviation 2.35, median 2, minimum 1,

Table 2
Cumulative distribution function of the time needed to update the automaton

X (sec)	% (CPU time $\leqslant$ X)
0.01	31%
0.02	38%
0.03	43%
0.04	47%
0.05	50%
0.2	76%
0.3	84%
0.4	88%
0.5	95%
0.68	100%

Table 3
Cumulative distribution function of the time needed to compute a restoration set

X (sec)	% (CPU time $\leqslant$ X)
0.01	90%
0.02	94%
0.1	99%
1.05	100%

Table 4
Cumulative distribution function of the full size of the restoration set

X (# of constraints)	% ($size(S) \leqslant$ X)
1	30%
2	51%
3	62%
4	71%
5	76%
10	92%
15	96%
192	100%

maximum 21). To summarize, let us define the full size of a set S of restorations as the sum of the cardinality of the restorations it contains ($Size(S) = \sum_{R \in S} Card(R)$). On our experiment set, the average size of the sets S is 4.31 constraints (standard deviation 5.39, median 2). The bigger S involved 192 constraints (the cumulative distribution function of the size of the restoration set is given in Table 4).

6. Conclusion

Several authors [22,27,32,39] have proposed to extend the CSP framework so as to handle configuration problems—the extension dealing mainly with the difficulties that are inherent to the structure of configuration problems. The handling of another salient feature of configuration problems, namely their interactivity, has prevailed on us to extend the framework in another direction and to define "Assumption-based CSPs". This led to a direct extension to non Boolean domains of definitions and properties that are well-known in the ATMS framework [17], e.g., nogoods, interpretations, explanations, etc. Of course, a Boolean encoding of (finite domains) CSPs can be easily achieved in polynomial time. However reducing a CSP to a CSP with Boolean constraints is definitely a bad strategy in the general case. On the one hand, it is not viable from a computational point of view. On the other hand, the Boolean encoding is difficult to understand and to maintain by the user—a drawback that is prehibitory when user-driven processes, like the interactive solving of a CSP, are considered. Moreover, the performances obtained on a real configuration problem shows it practically useful, especially when compared with the performances achieved via a Boolean encoding.

The requests considered in interactive configuration obviously correspond to problems that are highly combinatorial. From a practical point of view, our approach consists in pushing this cost into an off-line pre-computation step: the system works on a compilation of the CSP under the form of an automaton (the size of which may be theoretically exponential) but uses algorithms that are really efficient on this data structure (the worst case complexity is linear in the size of the automaton). This approach takes advantage of the fact that, in configuration problems, only a small set of constraints is subject to dynamicity, and that these constraints are unary. It finds a justification in the fact that real configurable products can actually be described concisely by automata. This is mainly due to two factors: the frequent interchangeability of values and the fact that complex products are typically structured into sub-components that are more or less independent from each other.

The idea of compiling CSPs is not new. For instance, tree clustering [16] can be viewed as an early proposal for CSP compilation. More recent works include Vempaty's automata [41], Moller's arrays [28] and Weigel and Falting's synthesis trees [42]. The data structures and the requests offered in these approaches are different but in any case, the key idea is to compile the set of solutions of a CSP and to run algorithms that require time polynomial in the size of the compiled form. Our contribution here is not a new compilation technique for CSPs since we used Vempaty's automata in our work. What is new is the way in which we then used the compiled form. Especially, we have shown that the set of computational tasks that can be achieved in a tractable way from such automata is not limited to consistency checking but includes more sophisticated tasks, like some of those supported by ATMSs.

The way we use Vempaty's automaton [41] is actually close to the use of OBDD-like structures in the handling of prime implicant/implicate and interpretations of a formula (see [12] for seminal work, and [7,10] for some approaches close to ours). The principles pertaining to the generation of a good (small) automaton are out of the scope of this paper (see [41] for more details) but it should be noticed that this problem is close to the generation of small OBDD-like structures. For instance, determining an ordering of the variables that minimizes the size of the automaton is an NP-hard problem.

The algorithms given in the present paper have been implemented and tested on a quite huge benchmark coming from a real application in car configuration. Concerning the off-line phase, the time consumed by the compilation process (a few hours) is reasonable, as well as the size of the resulting automaton (a few Mb). More importantly, concerning the on-line step, the algorithms described in this paper have been successfully tested: maintaining global consistency as well as computing restorations is immediate (less than 0.2 sec in more than 75% of the cases, less than 1.05 sec in any case). Accordingly, these empirical results show the practical value of a compilation-based approach to configuration. They also cohere with the experimental results reported in [6,37] showing that compilation can prove helpful in practice for many instances of a class of problems, even if this is not the case for the class of problems itself (i.e., in the worst case situation).

Further work will follow three main directions:

Empirical evaluation: Our approach successfully addresses our application to car configuration, but it is quite difficult to draw general conclusions from a few instances. An important issue lies in the design of a protocol for the empirical evaluation of our algorithm, that (even less close to real problems) could enlighten the limits of the approach. Such a protocol involves the definition of a generator of instances that satisfies our work hypothesis (structuration and interchangeability) or more ideally that takes a degree of structuration and a degree of interchangeability as input. Such a statistical tool would also be useful for comparing the main forms of CSP compilation (automata, tree clustering, synthesis trees).

Compilation of strongly structured CSPs: Our approach finds a justification in the fact that real configurable products can be described concisely by an automaton. This is due to two main factors: the frequent interchangeability of values and the fact that complex products are structured into sub-components that are more or less independent from each other. A more complete exploitation of the structural features of configurable products should include the combination of the representation by automata with composite CSPs. The second direction of research for the compilation of strongly structured CSPs consists in mixing Vempaty's automata with compilation procedures that are more oriented toward the structure of the constraint graph, namely tree clustering and synthesis trees. The next step in the compilation of CSP is undoubtedly the cross fertilization of existing approaches.

Computation of explanations: In this paper, we did not present algorithms for computing explanations and nogoods since this kind of information is generally less attractive than restorations/interpretations in the context of a configuration task,[5] unless the number of explanations is small and the number of restorations is large: in this case, it would be interesting for the user to generate the restoration by herself through the selection of one constraint per explanation. Anyway, the computation of nogoods and explanations can be attractive for other applications, e.g., the design of configuration knowledge bases [21,35] or constraint-based diagnosis [33]. If we accept to relax the requirements of minimality and of completeness, an efficient approach can be based on the information gathered by consistency enforcing algorithms, e.g., a trace of a filtering algorithm as in [24] or the justifications maintained by dynamic filtering algorithms like DN-AC4, DN-AC6 [4,19].

[5] Their main interest is generally ... the generation of restorations/interpretations through the computation of hitting sets, whereas the direct generation is generally cheaper as explained in Section 2.6.

Acknowledgements

The authors would like to thank the anonymous reviewers for their helpful comments. They are also grateful to S. Coste-Marquis, B. Pargamin and M. Veron for their help. The third author has been partly supported by the Région Nord/Pas de Calais, the IUT de Lens, and the European Communities.

Appendix A

A.1. Proofs of Propositions 1–11

Proof of Proposition 1.

(i) $\Rightarrow$ Let E be a consistent environment of $\mathcal{C}$. Suppose that there exists a nogood E_i such that E does not relax any constraint of E_i (i.e., $(\mathcal{H} \setminus E) \cap E_i = \emptyset$): all the constraints of E_i are in E. Since E_i is inconsistent with C, so is also E: contradiction. So, E being a consistent environment of $\mathcal{C}$ implies that E relaxes at least one constraint per nogood.

(i) $\Leftarrow$ Suppose that for every $E_i \in \mathcal{N}$, $(\mathcal{H} \setminus E) \cap E_i \neq \emptyset$ and that E is not a consistent environment: E is a conflict and thus a nogood is included in E. This is not possible since E relaxes at least one constraint per nogood. Thus E is a consistent environment.

(ii) $\Rightarrow$ Let E be a conflict of $\mathcal{C}$. Suppose that there exists an interpretation E_i such that E does not include any constraint of $\mathcal{H} \setminus E_i$ (i.e., $E \cap (\mathcal{H} \setminus E_i) = \emptyset$): $E \subseteq E_i$. Since E_i is a consistent environment, so is each of its subsets, and so is E. This contradicts the hypothesis "E is a conflict". So, for any interpretation E_i, $E \cap (\mathcal{H} \setminus E_i) \neq \emptyset$.

(ii) $\Leftarrow$ Suppose that for every $E_i \in \mathcal{I}$, $E \cap (\mathcal{H} \setminus E_i) \neq \emptyset$ and that E is not a conflict: E is a consistent environment and is thus a subset of an interpretation E_i. Since $E \cap (\mathcal{H} \setminus E_i) \neq \emptyset$, we get $E \cap (\mathcal{H} \setminus E) \neq \emptyset$. From this contradiction, we conclude that E is a conflict. $\square$

Proof of Proposition 2. Let $\Pi = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{H} \rangle$ be an A-CSP.

- E is an explanation of L on $\langle \mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{H} \rangle \Rightarrow \langle \mathcal{X}, \mathcal{D}, (\mathcal{C} \cup \{\neg L\}) \cup E \rangle$ is inconsistent and $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup E \rangle$ is consistent $\Rightarrow E$ is a conflict of $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup \{\neg L\}, \mathcal{H} \rangle$.
- E is a conflict of $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup \{\neg L\}, \mathcal{H} \rangle$ and $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup \mathcal{H} \rangle$ is consistent $\Rightarrow \langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup \{\neg L\} \cup E \rangle$ is inconsistent and $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup E \rangle$ is consistent $\Rightarrow E$ is an explanation of L on $\langle \mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{H} \rangle$.
- E is a minimal explanation of L on $\langle \mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{H} \rangle \Rightarrow \langle \mathcal{X}, \mathcal{D}, (\mathcal{C} \cup \{\neg L\}) \cup E \rangle$ is inconsistent and no $E' \subset E$ is an explanation of $L \Rightarrow E$ is a conflict of $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup \{\neg L\}, \mathcal{H} \rangle$ and ($\forall E' \subset E$, $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup \{\neg L\} \cup E' \rangle$ is consistent or $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup E' \rangle$ is inconsistent) $\Rightarrow E$ is a conflict for $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup \{\neg L\}, \mathcal{H} \rangle$ and $\forall E' \subset E$, $\langle \mathcal{X}, \mathcal{D}, (\mathcal{C} \cup \{\neg L\}) \cup E' \rangle$ is consistent $\Rightarrow E$ is a conflict for $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup \{\neg L\}, \mathcal{H} \rangle$ and $\forall E' \subset E$, E' is not a conflict for $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup \{\neg L\} \rangle \Rightarrow E$ is a nogood for $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \cup \{\neg L\} \rangle$.

- E is a nogood $\langle\mathcal{X},\mathcal{D},\mathcal{C}\cup\{\neg L\}$ and $\langle\mathcal{X},\mathcal{D},\mathcal{C}\cup\mathcal{H}\rangle$ is consistent $\Rightarrow\langle\mathcal{X},\mathcal{D},\mathcal{C}\cup\{\neg L\}\cup E\rangle$ is inconsistent and there is no $E'\subset E$ such as $\langle\mathcal{X},\mathcal{D},\mathcal{C}\cup\{\neg L\}\cup E'\rangle$ is inconsistent and $\langle\mathcal{X},\mathcal{D},\mathcal{C}\cup E\rangle$ is consistent $\Rightarrow E$ is a minimal explanation of L.
- E is a restoration of L on $\langle\mathcal{X},\mathcal{D},\mathcal{C},\mathcal{H}\rangle\Rightarrow\langle\mathcal{X},\mathcal{D},(\mathcal{C}\cup\{L\})\cup E\rangle$ is consistent $\Leftrightarrow E$ is a consistent environment of $\langle\mathcal{X},\mathcal{D},\mathcal{C}\cup\{L\},\mathcal{H}\rangle$.
- E is a minimal restoration of L on $\langle\mathcal{X},\mathcal{D},\mathcal{C},\mathcal{H}\rangle\Rightarrow\langle\mathcal{X},\mathcal{D},(\mathcal{C}\cup\{L\})\cup E\rangle$ is consistent and $\forall E'\supset E\ \langle\mathcal{X},\mathcal{D},(\mathcal{C}\cup\{L\})\cup E'\rangle$ is inconsistent $\Rightarrow E$ is an interpretation of $\langle\mathcal{X},\mathcal{D},\mathcal{C}\cup\{L\},\mathcal{H}\rangle$. □

Proof of Proposition 3. Easy consequence of Propositions 2 and 1. □

Proof of Proposition 4. Easy from the fact that $E'\subsetneq E\Rightarrow\phi(E')>\phi(E)$ when (i) $\phi(A)=\sum_{H\in\mathcal{H}\setminus A}\phi(H)$ and (ii) ϕ differs from 0 and from $+\infty$ (since $\phi(H)$ is a non-null integer for every assumption H of $\mathcal{H}$). □

Proof of Proposition 5. Easy from the three following remarks:

- $\forall E,\ \phi(E)=\sum_{H\in\mathcal{H}\setminus E}\phi(H)=\phi'(\langle\mathcal{X},\mathcal{D},\mathcal{C}\cup E\rangle)$.
- E is a consistent environment of Π iff $\langle\mathcal{X},\mathcal{D},\mathcal{C}\cup E\rangle$ is consistent iff $\langle\mathcal{X},\mathcal{D},\mathcal{C}\cup E\rangle$ is a relaxation of the valued CSP $(\mathcal{X},\mathcal{D},\mathcal{C}\cup\mathcal{H},\langle\mathbb{N}\cup\{+\infty\},>,+\rangle,\phi')$.
- $\phi'(\langle\mathcal{X},\mathcal{D},\mathcal{C}\cup E\rangle)\neq+\infty$ since it relaxes only constraints on $\mathcal{H}$, i.e., constraints such that $\phi'\neq+\infty$. □

Proof of Proposition 6.

(a) A path p from I to F defines a assignment s of $\mathcal{X}$ that is a solution of $\langle\mathcal{X},\mathcal{D},\mathcal{C}\rangle$. Let us denote E the subset of $\mathcal{H}$ satisfied by s. Since s satisfies E and $\mathcal{C}$, E is a consistent environment. Now, since the cost of p is equal to sum of the valuations of the subset of constraints in $\mathcal{H}$ that are violated by s, i.e., of the constraints of $\mathcal{H}\setminus E$, it holds that $\phi(E)=\mathit{cost}(p)$.

(b) Conversely, if E is a consistent environment, then there is a solution s of $\langle\mathcal{X},\mathcal{D},\mathcal{C}\rangle$ that satisfies at least all the constraints in E. Since all the solutions of $\langle\mathcal{X},\mathcal{D},\mathcal{C}\rangle$ are represented by the automaton, to s corresponds a path p. Let us denote F the subset of constraints of $\mathcal{H}$ violated by s. It holds that (i) $F\subseteq\mathcal{H}\setminus E$ and (ii) $\mathit{cost}(p)=\sum_{H\in F}\phi(H)$. Since $F\subseteq\mathcal{H}\setminus E$, $\sum_{H\in F}\phi(H)\leqslant\sum_{H\in\mathcal{H}\setminus E}\phi(H)$, i.e., $\mathit{cost}(p)\leqslant\phi(E)$.

(c) A path p from I to F that supports L defines a complete assignment of $\mathcal{X}$, that is a solution s of $\langle\mathcal{X},\mathcal{D},\mathcal{C}\rangle$ that satisfies L. Let F be the set of constraints in $\mathcal{H}$ that are violated by s. Since s satisfies $\mathcal{C}$ and $\mathcal{H}\setminus F$, $E=\mathcal{H}\setminus F$ is a restoration of L on Π. Since $\mathit{cost}(p)\sum_{H\in F}\phi(H)$, $\mathit{cost}(p)=\phi(E)$.

(d) If E is a restoration of L, then there is a solution s of $\langle\mathcal{X},\mathcal{D},\mathcal{C}\rangle$ that satisfies L and all the constraints in E. Since all the solutions of $\langle\mathcal{X},\mathcal{D},\mathcal{C}\rangle$ are represented by the automaton, to s corresponds a path p which supports L. Let us denote F the subset of constraints of $\mathcal{H}$ violated by s. It holds that (i) $F\subseteq\mathcal{H}\setminus E$ and (ii) $\mathit{cost}(p)=\sum_{H\in F}\phi(H)$. Since $F\subseteq\mathcal{H}\setminus E$, $\sum_{H\in F}\phi(H)\leqslant\sum_{H\in\mathcal{H}\setminus E}\phi(H)$, i.e., $\mathit{cost}(p)\leqslant V(E)$. □

Proof of Proposition 7.

(a) $\Rightarrow$ Let p be a minimal path from I to F: for any path p', $cost(p) \leqslant cost(p')$. From Proposition 6, there exists a consistent environment E such that $cost(p) = \phi(E)$. Let us prove that E is V-optimal. Consider any consistent environment E'. From Proposition 6, there exists a path p' from I to F with $cost(p') \leqslant \phi(E')$. Thus $\phi(E) = cost(p) \leqslant cost(p') \leqslant \phi(E')$: for any E', $\phi(E) \leqslant \phi(E')$, i.e., E is a V-interpretation.
$\Leftarrow$ Let E be a V-consistent environment. From Proposition 6, there exists a path p from I to F with cost $cost(p) \leqslant \phi(E)$. Let us prove that it is minimal: suppose that there is another path p' such that $cost(p') < cost(p)$. From Proposition 6, this means that there is a consistent environment E' such that $cost(p') = \phi(E')$, i.e., $\phi(E') = cost(p') < cost(p) \leqslant \phi(E)$; hence, E is not V-optimal. Hence, such a path p' does not exist: p is minimal.

(b) Comes from Proposition 6: if there is a path from I to F with cost 0, then there exists a consistent environment E such that $\phi(E) = 0$, i.e., $\sum_{\mathcal{H}\setminus E} \phi(H) = 0$. Since it is assumed that no constraint has a null valuation, it holds that $E = \mathcal{H}$, i.e., $\mathcal{H}$ is not a conflict. Conversely, if $\mathcal{H}$ is not a conflict, then it is a consistent environment. Thus, from Proposition 6, there exists a path p from I to F such that $cost(p) \leqslant \sum_{\mathcal{H}\setminus\mathcal{H}} \phi(H) = 0$: $cost(p) = 0$.

(c) If the assignment d of X_i is supported by an edge involved in a path of cost zero, then this assignment is involved in a solution of $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ that satisfies all the constraints of $\mathcal{H}$ (assuming that no constraint has a null valuation), i.e., $d \in P_{X_i}$. Conversely, the automaton represents all the solutions of $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$. Hence, if every path that supports the assignment has a positive cost, then every solution of $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ violates at least one constraint of $\mathcal{H}$, i.e., d does not belong to P_{X_i}.

(d) Similar to the proof of item (a) (based on items (c) and (d) of Proposition 6 rather than of items (a) and (b)). □

Proof of Proposition 8.

- $cost(q)$ is the cost of the minimal path from I to F through q. It is thus equal to the cost of the minimal path from I to q plus the cost of the minimal path from q to F, i.e., to $c_l(q) + c_r(q)$;
- $cost(a)$ is the cost of the minimal path from I to F through a. It is thus equal to the cost of the minimal path from I to the origin of a plus the cost of the minimal path from the extremity of a plus $\phi(a)$, i.e. to $c_l(in(a)) + \phi(a) + c_r(out(a))$. □

Proof of Proposition 9.

(a) Direct from Definition 10, since $c_l(F)$ is the minimal cost of the paths from I to F, and $c_r(I)$ is the minimal cost of the paths from I to F.

(b) Easy from (a) and Proposition 7(b). □

Proof of Proposition 10. $cnt[X_i, d]$ is the number of edges that (i) support the assignment $X_i := d$ and (ii) belong to a path of cost 0 from I to F. So there is a path of cost 0 that support d for X_i iff $cnt(X_i, d) > 0$.

Since $d \in P_{X_i}$ iff there is an edge a that supports d for X_i and a belongs to a path of cost zero from I to F (Proposition 7(c)), we have $d \in P_{X_i}$ iff $cnt(X_i, d) > 0$. □

Proof of Proposition 11. We prove only (a) (the proof of (b) is similar). Since $c_r(F) = 0$ by definition, the proposition holds when the path reduces only to one edge a: $c_r(q) = Min_{a,in(a)=q}(c_r(F) + \phi(a)) = Min_{a,in(a)=q}(\phi(a))$. Thus $c_r(q) = \phi(a)$ iff $\phi(a) = cost((q, F))$ is minimal.

Suppose now that proposition (a) holds for the paths of k edges. Consider a path p of $k + 1$ edges for a q to F, let a_1 be the first of these edges and p' be the subpath from $out(a_1)$ to F.

$\Rightarrow$ If p is of minimal cost among the paths from q to F, $Min_{a',in(a')=q}(c_r(out(a')) + \phi(a') = c_r(out(a_1)) + \phi(a_1)$, thus $c_r(in(a_1)) = \phi(a_1) + c_r(out(a_1))$. Since p' is minimal among the paths from $out(a)$ to F, the property $c_r(in(a)) = \phi(a) + c_r(out(a))$ holds by induction.

$\Leftarrow$ Suppose that, for each edge a of p, $c_r(in(a)) = \phi(a) + c_r(out(a))$. From $c_r(q) = Min_{a,in(a)=q}(c_r(out(a)) + \phi(a)) = c_r(out(a_1)) + \phi(a_1)$ we deduce that the minimal path from q to F begins by edge a_1. Now, by induction, we know that p' is of minimal cost among the paths from $out(a_1)$ to F. So p is minimal among the paths from q to F. □

A.2. Complexity proofs

A.2.1. Background

We assume familiarity with basic notions of computational complexity theory (see, e.g., [29] for a survey).

The class of all languages (encoding decision problems) that can be recognized in polynomial time by a nondeterministic Turing machine is denoted by NP. Among all the problems in NP, the hardest ones are those from which every problem in NP can be *polynomially many-one reduced*: such problems are referred to as NP-*complete*. If any of them has a polynomial algorithm, then P = NP holds. Accordingly, it is believed that it is impossible to solve NP-complete problems in polynomial time. SAT, the problem in determining whether a propositional formula is satisfiable, is the prototypical NP-complete problem. Its complementary problem UNSAT (consisting of determining whether a propositional formula is unsatisfiable) is not necessarily in NP (in contrast to P, NP is not known to be closed under complementation). It is assigned to the class coNP that contains the complementary problems to problems of NP.

To go further into the classification of non-efficiently solvable problems, an important tool is the notion of Turing machine (deterministic or nondeterministic) *with oracle*. Let X be a class of decision problems. P^X (respectively NP^X) is the class of all decision problems that can be solved in polynomial time using a deterministic (respectively nondeterministic) Turing machine that can use an oracle for deciding a problem $Q \in X$ for "free" (i.e., within a constant, unit time).

On this ground, the complexity class Σ_2^p is defined by $\Sigma_2^p = \mathsf{NP}^{\mathsf{NP}}$. The hardest problems of this class are referred to as Σ_2^p-complete problems. Among them is the validity problem 2-$QBF_\exists$ that consists in determining whether a given quantified boolean formula of the form $\exists A\ \forall B\ \Sigma$ is valid (where Σ is any formula from $PROP_{PS}$ or even a formula in disjunctive normal form—DNF—and $\{A, B\}$ is a partition of $Var(\Sigma)$).

In order to discriminate further among the problems from $\Delta_2^p = \mathsf{P}^{\mathsf{NP}}$, one can focus on the number of calls to an NP oracle that are used. Thus, the complexity class BH_2 is defined by $\mathrm{BH}_2 = \{L_1 \cap \bar{L}_2 \mid L_1 \in \mathsf{NP},\ L_2 \in \mathsf{NP}\}$. The membership to any language of BH_2 can be determined using only two calls to an NP oracle. The hardest problems of this class are referred to as BH_2-complete problems. Among them is the SAT-UNSAT problem that consists in determining, given a pair $\langle \Phi, \Psi \rangle$ of propositional formulas in conjunctive normal form (CNF), whether Φ is satisfiable and Ψ is unsatisfiable.

A.2.2. How the proofs work

The membership proofs only assume that checking whether a given assignment s is a solution of a given CSP can be done in time polynomial in the input size (this is the unique restriction that we put on the finite set of constraints C over X).

The hardness proofs rely on two principles:

- First, the transformation of a k-CNF Σ into a binary CSP $CSP(\Sigma) = \langle X, D, C \rangle$ [3]: the idea is to associate to each clause c_i of Σ a variable x_i such that the domain D_i of x_i is the set of literals of c_i. A constraint $C_{i,j}$ between two variables x_i and x_j associated to two clauses c_i and c_j is created if the clause c_i contains a literal and c_j is complement. The relation $R(C_{i,j})$ is defined by the cartesian product $D_i \times D_j$ minus the tuples (l_i, l_j) such that l_i is the negation of l_j. Bennaceur [3] has shown that this transformation is polynomial and that Σ is satisfiable iff $CSP(\Sigma)$ is consistent.
- Secondly, the definition of an A-CSP from a CNF Σ and a set of literals $\Theta = \{l_1, \ldots, l_m\}$. Σ is the static part of the A-CSP and Θ its dynamic part. First, we generate a trivial CNF Λ from the literals of Θ: for each of l_i of Θ, the clause $\neg l_i \vee l_i$ is considered in Λ. Obviously, Σ is equivalent to $\Sigma \cup \Lambda$. Then, define the A-CSP $Acsp(\Sigma, \Theta) = \langle \mathcal{X}, \mathcal{D}, \mathcal{C}, \mathcal{H} \rangle$ as follows: $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle = CSP(\Sigma \cup \Lambda)$ is Bennaceur's encoding and $\mathcal{H}$ is a set $\{H(l_1), \ldots, H(l_m)\}$ of unary constraints such that, for each literal l_i of Θ, $V(H(l_i))$ is the variable encoding the clause $\neg l_i \vee l_i$ of Λ and $R(H(l_i)) = \{l_i\}$. Let $\{l_1, \ldots, l_k\}$ be any subset of Θ. It holds that: $\Sigma \wedge l_1 \wedge \cdots \wedge l_k$ is satisfiable (respectively unsatisfiable) iff $E = \{H(l_1), \ldots, H(l_k)\}$ is a consistent environment (respectively a conflict) for $Acsp(\Sigma, \Theta)$.

Notice also that:

- $\langle \mathcal{X}, \mathcal{D}, \mathcal{H} \rangle$ involves unary constraints only.
- $\langle \mathcal{X}, \mathcal{D}, \mathcal{H} \rangle$ is consistent.
- Since $\Sigma \cup \Lambda$ is equivalent to Σ, $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ is consistent iff Σ is satisfiable.

As a consequence, each of the following proofs remain valid under the first two assumptions related to configuration problems: $\langle \mathcal{X}, \mathcal{D}, \mathcal{H} \rangle$ is consistent and $\mathcal{H}$ contains only unary contraints. All of them are valid when the constraints L to restore or to explain are unary. Most of them also remain valid when $\langle \mathcal{X}, \mathcal{D}, \mathcal{C} \rangle$ is assumed to be consistent (except (6)).

A.2.3. Complexity proofs: Conflicts

(1) Given $\Pi = \langle \mathcal{X}, D, C, H \rangle$ and $E \subseteq H$, is E a conflict for Π?
   - Membership to coNP: We consider the complementary problem. E is not a conflict for P iff $\langle X, D, C \cup E \rangle$ is consistent. This can be easily checked in nondeterministic polynomial time (it is sufficient to guess an assignment over X and to check in polynomial time that it is a solution).

– Hardness: Reduction from UNSAT. Let Φ be a CNF formula. We define the formula $M = (new \vee \Phi)$ where *new* is a new variable (not belonging to $Var(\Phi)$). M is satisfiable and it can be polynomially transformed into an equivalent CNF M'. It can be checked that Φ is unsatisfiable iff $\neg new \wedge M'$ is unsatisfiable. Consider the A-CSP $Acsp(M', \{\neg new\})$: $\neg new \wedge M'$ is unsatisfiable iff $\{H(\neg new)\}$ is a conflict for $Acsp(M', \{\neg new\})$. Since M' is satisfiable, the proof also holds under the restriction of a consistent set of constraints $\mathcal{C}$.

(2) Given $\Pi = \langle X, D, C, H\rangle$ and $E \subseteq H$, is E a nogood for Π?

– Membership to BH_2: We check that $\langle X, D, C \cup E\rangle$ is inconsistent (one inconsistency check) and independently, we check that $\langle X, D, C \cup (E \setminus \{e\})\rangle$ is consistent for every $e \in E$. All these consistency checks can be encoded in polynomial time into a single consistency check. Indeed, $C \cup (E \setminus \{e\})$ is consistent for every $e \in E$ iff the constraint $\bigwedge_{e\in E} rename(C \cup (E \setminus \{e\}))$ is inconsistent (here, each $rename(C \cup (E \setminus \{e\}))$ is a uniform renaming of the variables occurring in $C \cup (E \setminus \{e\})$ into new variables). Hence, one inconsistency check plus one consistency check are sufficient to test whether E is a nogood for Π, which shows the membership to BH_2.

– Hardness: Reduction from SAT-UNSAT. Let Φ, Ψ be two CNF formulas. We define the formula $M = (\neg new_1 \vee \Phi) \wedge (\neg new_1 \vee \neg new_2 \vee rename(\Psi))$ where new_1 and new_2 are new variables (not belonging to $Var(\Phi) \cup Var(rename(\Psi))$), and $rename(\Psi)$ is a uniform renaming of the variables of Ψ (into new variables, not occurring in $Var(\Phi)$). M is satisfiable and it can be polynomially transformed into an equivalent CNF M'. It can be checked that $new_2 \wedge M'$ is satisfiable. Moreover, Φ is satisfiable and Ψ is unsatisfiable iff $new_1 \wedge new_2 \wedge M'$ is unsatisfiable and $new_1 \wedge M'$ is satisfiable. Consider the A-CSP $Acsp(M', \{new_1, new_2\})$. It holds that $new_1 \wedge new_2 \wedge M'$ is unsatisfiable and $new_1 \wedge M'$ is satisfiable iff $\{H(new_1), H(new_2)\}$ is a nogood for $Acsp(M', \{new_1, new_2\})$. Since M' is satisfiable, the proof holds under the restriction of a consistent set of constraints $\mathcal{C}$.

(3) Given $\Pi = \langle X, D, C, H\rangle$, is there a conflict for Π?

– Membership to coNP: We consider the complementary problem. There is no conflict for Π iff $\langle X, D, C \cup H\rangle$ is consistent, which can be checked easily in nondeterministic polynomial time.

– Hardness: Reduction from UNSAT. Let Φ be a CNF formula. We define the formula $M = (new \vee \Phi)$ where *new* is a new variable (not belonging to $Var(\Phi)$). M is satisfiable and it can be polynomially transformed into an equivalent CNF M'. It can be checked that Φ is unsatisfiable iff $\neg new \wedge M'$ is unsatisfiable. Consider the A-CSP $Acsp(M', \{\neg new\})$: since M' is satisfiable, the static part of the A-CSP is consistent; thus $\{\neg new\}$ is the only potential conflict. Hence $\neg new \wedge M'$ is unsatisfiable iff there is a conflict for $Acsp(M', \{\neg new\})$. Since M' is satisfiable, the proof also holds under the restriction of a consistent set of constraints $\mathcal{C}$.

A.2.4. Complexity proofs: Consistent environments

(4) Given $\Pi = \langle X, D, C, H\rangle$ and $E \subseteq H$, is E a consistent environment for Π?
- Membership to NP: Trivial consequence of the fact that testing whether E is a restoration of L on Π is in NP since E is a consistent environment for Π iff E is a restoration of *true* (the constraint always satisfied) on Π.
- Hardness: Direct consequence of the fact that E is a consistent environment for Π iff E is not a conflict for Π and the coNP-hardness proof above (1).

(5) Given $\Pi = \langle X, D, C, H\rangle$ and $E \subseteq H$, is E an interpretation of Π?
- Membership to BH_2: It is sufficient to check that E is consistent for Π (one consistency check), and independently to check that $\langle X, D, C \cup E \cup \{h\}\rangle$ is inconsistent for every $h \in \mathcal{H} \setminus E$. All these inconsistency checks can be encoded in polynomial time into a single inconsistency check. Indeed, $C \cup E \cup \{h\}\rangle$ is inconsistent for every $h \in H \setminus E$ iff the constraint $\bigvee_{h \in H\setminus E} \mathit{rename}(C \cup E \cup \{h\})$ is inconsistent (here, each $\mathit{rename}(C \cup E \cup \{h\})$ is a uniform renaming of the variables occurring in $C \cup E \cup \{h\}$ into new variables). Hence, one consistency check plus one inconsistency check are sufficient to test whether E is a maximal consistent environment of Π, which shows the membership to BH_2.
- Hardness: Reduction from SAT-UNSAT. Let Φ, Ψ be two CNF formulas. We define the formula $M = (\neg \mathit{new}_1 \vee \Phi) \wedge (\neg \mathit{new}_2 \vee \mathit{rename}(\Psi))$ where new_1 and new_2 are new variables (not belonging to $\mathit{Var}(\Phi) \cup \mathit{Var}(\mathit{rename}(\Psi))$), and $\mathit{rename}(\Psi)$ is a uniform renaming of the variables of Ψ (into new variables, not occurring in $\mathit{Var}(\Phi)$). M is satisfiable and it can be polynomially transformed into an equivalent CNF M'. It can be checked that Φ is satisfiable and Ψ is unsatisfiable iff $\mathit{new}_1 \wedge M'$ is satisfiable and $\mathit{new}_1 \wedge \mathit{new}_2 \wedge M'$ is unsatisfiable. Consider the A-CSP $\mathit{Acsp}(M', \{\mathit{new}_1, \mathit{new}_2\})$; $\mathit{new}_1 \wedge M'$ is satisfiable and $\mathit{new}_1 \wedge \mathit{new}_2 \wedge M'$ is unsatisfiable iff $\{H(\mathit{new}_1)\}$ is an interpretation of $\mathit{Acsp}(M', \{\mathit{new}_1, \mathit{new}_2\})$. Since M' is satisfiable, the proof also holds under the restriction of a consistent set of constraints $\mathcal{C}$.

(6) Given $\Pi = \langle X, D, C, H\rangle$, is there a consistent environment for Π?
- The problem is trivial if $\mathcal{C}$ is assumed to be consistent (in this case, $\emptyset$ is a trivial consistent environment). It is NP-complete otherwise:
- Membership to NP: A consistent environment for Π exists iff $\langle X, D, C\rangle$ is consistent, which can be easily checked in nondeterministic polynomial time.
- Hardness: Reduction from SAT. Let Σ be a CNF. Σ is a satisfiable CNF iff the encoding of Σ by a CSP is consistent iff $\mathit{Acsp}(\Sigma, \{\})$ has a consistent environment.

A.2.5. Complexity proofs: Explanations

(7) Given $\Pi = \langle X, D, C, H\rangle$, $E \subseteq H$, and L, is E an explanation of L on Π (general case)?
- Membership to BH_2: E is an explanation of L on Π iff $\langle X, D, C \cup E\rangle$ is consistent and $\langle X, D, C \cup E \cup \{\neg L\}\rangle$ is inconsistent. Hence, one inconsistency check plus one consistency check are sufficient to test whether E is an explanation of L on Π, which shows the membership to BH_2.
- Hardness: Reduction from SAT-UNSAT. Let Φ, Ψ be two CNF formulas. We define the formula $M = (\neg \mathit{new}_1 \vee \Phi) \wedge (\neg \mathit{new}_1 \vee \neg \mathit{new}_2 \vee \mathit{rename}(\Psi))$

where new_1 and new_2 are two new variables (not belonging to $Var(\Phi) \cup Var(rename(\Psi)))$, and $rename(\Psi)$ is a uniform renaming of the variables of Ψ (into new variables, not occurring in $Var(\Phi)$). M is satisfiable and it can be polynomially transformed into an equivalent CNF M'. It can be checked that Φ is satisfiable and Ψ is unsatisfiable iff $new_1 \wedge M$ is satisfiable and $new_1 \wedge M \wedge new_2$ is unsatisfiable. Consider the A-CSP $Acsp(M', \{new_1\})$, and a variable X_i of this CSP such that $\neg new_2$ is a value of its domain. $new_1 \wedge M$ is satisfiable and $new_1 \wedge M \wedge new_2$ is not iff $H(new_1)$ is an explanation of the unary constraint "$X_i = \neg new_2$".

(7b) Given $\mathcal{C}$, $E \subseteq H$, and L, is E an explanation of L on Π?
This problem is "only" coNP-complete when $\langle X, D, C \cup H\rangle$ is known as consistent.

- Membership to coNP: We consider the complementary problem. E is not an explanation of L on Π iff $C \cup E$ is inconsistent or $C \cup E \cup \{\neg L\}$ is consistent. Since $C \cup H$ is assumed consistent, it is also the case that $C \cup E$ is consistent since $E \subseteq H$. Accordingly, E is not an explanation of L on Π iff $C \cup E \cup \{\neg L\}$ is consistent, which can be easily checked in nondeterministic polynomial time (just guess an assignment and check that it is a solution of $C \cup E \cup \{\neg L\}$).
- Hardness: Reduction from UNSAT. Let Φ be a CNF formula. We define the formula $M = (new \vee \Phi)$ where new is a new variable (not belonging to $Var(\Phi)$). M is satisfiable and it can be polynomially transformed into an equivalent CNF M'. It can be checked that Φ is unsatisfiable iff $\neg new \wedge M'$ is unsatisfiable. Consider the A-CSP $\mathcal{C} = Acsp(M', \{\})$. Since M' is satisfiable, $\langle X, D, C \cup H\rangle$ is consistent. Consider a variable X_i of this CSP such that new is a value of its domain. $\neg new \wedge M'$ is unsatisfiable iff $\{\}$ is an explanation of the constraint "$X_i = \neg new$".

(8) Given $\Pi = \langle X, D, C, H\rangle$, $E \subseteq H$, and L, is E a minimal explanation of L on Π?

- Membership to BH_2: E a minimal explanation of L on Π iff $C \cup E$ is consistent, $C \cup E \cup \{\neg L\}$ is inconsistent and for every $e \in E$, $C \cup (E \setminus \{e\}) \cup \{\neg L\}$ is consistent. In order to check whether $C \cup E$ is consistent and for every $e \in E$, $C \cup (E \setminus \{e\}) \cup \{\neg L\}$ is consistent, it is sufficient to check the consistency of the constraint $C \wedge E \wedge \bigwedge_{e \in E} rename(C \cup (E \setminus \{e\}) \cup \{\neg L\})$ (here, each $rename(C \cup (E \setminus \{e\}) \cup \{\neg L\})$ is a uniform renaming of the variables occurring in $C \cup (E \setminus \{e\}) \cup \{\neg L\}$ into new variables). Hence, one inconsistency check plus one consistency check are sufficient to test whether E is a minimal explanation of L on Π, which shows the membership to BH_2.
- Hardness: The hardness proof given above (7) still holds here since $\emptyset$ is an explanation of L on Π iff it is a minimal explanation of L on Π.

(9) Given $\Pi = \langle X, D, C, H\rangle$ and L, is there an explanation of L on Π?

- Membership to Σ_2^p: Here is a polynomial time nondeterministic algorithm using an NP oracle that is sufficient to determine whether an explanation of L on Π exists. Guess a subset E of H, and check that E is an explanation of L on Π. As shown above (7), this can be achieved using only two calls to an NP oracle (one consistency check plus one inconsistency check).

– Hardness: Reduction from 2-$QBF_{\exists}$. Let $\exists A\ \forall B\ \Sigma$ be a quantified boolean formula s.t. $\{A, B\}$ is a partition of $Var(\Sigma)$ and Σ is a DNF formula.
Let $\Delta = \{a_1, \neg a_1, \ldots, a_n, \neg a_n\}$ be the set of all the literals built up on A. $\exists A\ \forall B\ \Sigma$ holds iff $\exists \Delta' \subset \Delta$ such that $\Delta' \wedge \neg\Sigma$ is unsatisfiable and Δ' is satisfiable. Now, let $M = new \vee \neg\Sigma$ where *new* is a new variable (not belonging to $Var(\Sigma)$). M is satisfiable. Since Σ is a DNF and thus $\neg\Sigma$ a CNF, M can be polynomially transformed into an equivalent CNF M'. It holds that $\Delta' \wedge \neg\Sigma$ is unsatisfiable and Δ' is satisfiable iff $\Delta' \wedge M \wedge \neg new$ is unsatisfiable and $\Delta' \wedge M$ is satisfiable. Consider the A-CSP $Acsp(M', \Delta)$, and a variable X_i of this CSP such that *new* is a value of its domain. $\Delta' \wedge M \wedge \neg new$ is unsatisfiable and $\Delta' \wedge M$ satisfiable iff there is an explanation of the unary constraint "$X_i = new$". Thus Φ is satisfiable and Ψ is unsatisfiable iff there is an explanation of the unary constraint "$X_i = new$".

A.2.6. Complexity proofs: Restorations

(10) Given $\Pi = \langle X, D, C, H\rangle$, $E \subseteq H$, and L, is E a restoration of L on Π?
– Membership to NP: Here is a polynomial time nondeterministic algorithm for checking whether E is a restoration of L on Π. Guess an assignment over X and check in polynomial time that it is a solution of $C \cup E \cup \{L\}$.
– Hardness: Direct consequence of the NP-hardness of the restricted case of checking whether E is a consistent environment for Π (item (4)).

(11) Given $\Pi = \langle X, D, C, H\rangle$, $E \subseteq H$, and L, is E a maximal restoration of L on Π?
– Membership to BH_2: It is sufficient to check that E is a restoration of L on Π (one consistency check), and independently to check that $\langle X, D, C \cup E \cup \{L\} \cup \{h\}\rangle$ is inconsistent for every $h \in H \setminus E$. All these inconsistency checks can be encoded in polynomial time into a single inconsistency check. Indeed, $C \cup E \cup \{L\} \cup \{h\}\rangle$ is inconsistent for every $h \in H \setminus E$ iff the constraint $\bigvee_{h \in H\setminus E} rename(C \cup E \cup \{L\} \cup \{h\})$ is inconsistent (here, each $rename(C \cup E \cup \{L\} \cup \{h\})$ is a uniform renaming of the variables occurring in $C \cup E \cup \{L\} \cup \{h\}$ into new variables). Hence, one consistency check plus one inconsistency check are sufficient to test whether E is a maximal restoration of L on Π, which shows the membership to BH_2.
– Hardness: Direct consequence of the BH_2-hardness of the restricted case of checking whether E is an an interpretation for Π (item (5)).

(12) Given $\Pi = \langle X, D, C, H\rangle$ and L, is there a restoration of L on Π?
– The problem is trivial when $\langle X, D, C \cup \{L\}\rangle$ is assumed to be consistent ($\emptyset$ is a restoration of L). It is NP-complete otherwise:
– Membership to NP: A restoration of L on Π exists iff $\langle X, D, C \cup \{L\}\rangle$ is consistent, which can be easily checked in nondeterministic polynomial time.
– Hardness: Reduction from SAT. Let Φ be a CNF formula. We define the formula $M = (new \vee \Phi)$ where *new* is a new variable (not belonging to $Var(\Phi)$). M is satisfiable and it can be polynomially transformed into an equivalent CNF M'. It can be checked that Φ is satisfiable iff $\neg new \wedge M'$ is satisfiable. Consider the A-CSP $Acsp(M', \{\})$, and a variable X_i of this CSP such that *new* is a value of

its domain. $\neg new \wedge M'$ is satisfiable iff there is a restoration of the constraint "$X_i \neq new$".

References

[1] K.S. Brace, R.L. Rudell, R.E. Bryant, Efficient implementation of a BDD package, in: Proc. 27th ACM/IEEE Design Automation Conference, Orlando, FL, 1990, pp. 40–45.

[2] R.E. Bryant, Graph based algorithms for boolean function manipulation, IEEE Trans. Comput. 35 (8) (1986) 677–692.

[3] H. Bennaceur, The satisfiability problem regarded as a constraint satisfaction problem, in: Proc. ECAI-96, Budapest, Hungary, 1996, pp. 155–159.

[4] C. Bessière, Arc-consistency for dynamic constraint satisfaction problems, in: Proc. AAAI-91, Anaheim, CA, 1991, pp. 221–227.

[5] C. Bessière, Arc-consistency for non-binary dynamic CSPs, in: Proc. ECAI-92, Vienna, Austria, 1992, pp. 23–27.

[6] Y. Boufkhad, E. Grégoire, P. Marquis, B. Mazure, L. Saïs, Tractable cover compilations, in: Proc. IJCAI-97, Nagoya, Japan, 1997, pp. 122–127.

[7] F. Bouquet, P. Jégou, Solving over-constrained CSP using weighted OBDDs, in: Proc. Over-Constrained Systems, Lecture Notes in Computer Science, Vol. 1106, Springer, Berlin, 1996, pp. 293–308.

[8] M. Cadoli, F.M. Donini, A survey on knowledge compilation, AI Comm. 10 (1997) 137–150.

[9] T. Castell, Computation of prime implicates and prime implicants by a variant of the Davis and Putnam procedure, in: Proc. ICTAI-96, Toulouse, France, 1996, pp. 428–429.

[10] C. Cayrol, M.C. Lagasquie-Schiex, T. Schiex, Nonmonotonic reasoning: From complexity to algorithms, Ann. Math. Artificial Intelligence 22 (3–4) (1998) 207–236.

[11] E. Charniak, S.E. Shimony, Probabilistic semantics for cost-based abduction, in: Proc. AAAI-90, Boston, MA, 1990, pp. 106–111.

[12] O. Coudert, J.-C. Madre, A logically complete reasoning maintenance system based on a logical constraint solver, in: Proc. IJCAI-91, Sydney, Australia, 1991, pp. 294–299.

[13] A. Darwiche, Compiling devices into decomposable negation normal form, in: Proc. IJCAI-99, Stockholm, Sweden, 1999, pp. 284–289.

[14] R. Dechter, A. Dechter, Belief maintenance in dynamic constraint networks, in: Proc. AAAI-88, St. Paul, MN, 1988, pp. 37–42.

[15] R. Dechter, A. Dechter, Structure-driven algorithms for truth maintenance, Artificial Intelligence 82 (1996) 1–20.

[16] R. Dechter, J. Pearl, Tree clustering schemes for constraint-processing, Artificial Intelligence 38 (3) (1989) 353–366.

[17] J. de Kleer, An assumption-based TMS, Artificial Intelligence 28 (1986) 127–167.

[18] A. del Val, Tractable databases: How to make propositional unit resolution complete through compilation, in: Proc. KR-94, Bonn, Germany, 1994, pp. 551–561.

[19] R. Debruyne, Arc-consistency in dynamic CSPs is no more prohibitive, in: Proc. ICTAI-96, Toulouse, France, 1996, pp. 299–306.

[20] T. Eiter, G. Gottlob, The complexity of logic-based abduction, J. ACM 42 (1995) 3–42.

[21] A. Falfernig, G. Friedrich, D. Jannach, M. Stumptner, Consistency-based diagnosis of configuration knowledge bases, in: Proc. ECAI-2000, Berlin, 2000, pp. 146–150.

[22] E. Gelle, R. Weigel, Interactive configuration using constraint satisfaction techniques, in: Proc. Artificial Intelligence and Manufacturing Research Planning Workshop, AAAI Technical Report FS-96-03, 1996, pp. 37–44.

[23] J.R. Hobbs, M.E. Stickel, D.E. Appelt, P. Martin, Interpretation as abduction, Artificial Intelligence 63 (1993) 69–142.

[24] N. Jussien, O. Lhomme, Dynamic domain splitting for numeric CSP, in: Proc. ECAI-98, Brighton, UK, 1998, pp. 224–228.

[25] L. Lobjois, G. Verfaillie, Problèmes incohérents: Expliquer l'incohérence, restaurer la cohérence, in: Actes de JNPC-99, 1999, pp. 111–120.

[26] P. Marquis, Knowledge compilation using theory prime implicates, in: Proc. IJCAI-95, Montreal, Quebec, 1995, pp. 837–843.

[27] S. Mittal, F. Frayman, Dynamic constraint satisfaction problems, in: Proc. AAAI-90, Boston, MA, 1990, pp. 25–32.

[28] G. Moller, On the technology of array based logic, Ph.D. Dissertation, Technical University of Denmark, 1995.

[29] C.H. Papadimitriou, Computational Complexity, Addison-Wesley, Reading, MA, 1994.

[30] D. Poole, Probabilistic Horn abduction and Bayesian networks, Artificial Intelligence 64 (1993) 81–129.

[31] R. Reiter, J. de Kleer, Foundations of assumption-based truth maintenance systems: Preliminary report, in: Proc. AAAI-87, Seattle, WA, 1987, pp. 183–188.

[32] D. Sabin, E.C. Freuder, Configuration as composite constraint satisfaction, in: Proc. Artificial Intelligence and Manufacturing Research Planning Workshop, AAAI Technical Report FS-96-03, 1996, pp. 28–36.

[33] D. Sabin, M. Sabin, R.D. Russell, E.C. Freuder, A constraint-based approach to diagnosing software problems in computer networks, in: Proc. CP-95, Lecture Notes in Computer Science, Vol. 976, 1995, pp. 463–480.

[34] D. Sabin, R. Weigel, Product configuration frameworks—A survey, IEEE Intelligent Systems Appl. 13 (4) (1998) 42–49.

[35] M. Sabin, E.C. Freuder, Detecting and resolving inconsistency in conditional constraint satisfaction problems, in: Proc. AAAI-99 Workshop on Configuration, Orlando, FL, 1999, pp. 90–94.

[36] T. Schiex, H. Fargier, G. Verfaillie, Valuated constraint satisfaction problems: Hard and easy problems, in: Proc. IJCAI-95, Montreal, Quebec, 1995, pp. 631–637.

[37] R. Schrag, Compilation for critically constrained knowledge bases, in: Proc. AAAI-96, Portland, OR, 1996, pp. 510–515.

[38] B. Selman, H.A. Kautz, Knowledge compilation and theory approximation, J. ACM 43 (1996) 193–224.

[39] T. Soininen, E. Gelle, Dynamic constraint satisfaction in configuration, in: Proc. AAAI-99 Workshop on Configuration, Orlando, FL, 1999, pp. 95–100.

[40] M. Stumptner, An overview of knowledge-based configuration, AI Comm. 10 (1997) 111–125.

[41] N.R. Vempaty, Solving constraint satisfaction problems using finite state automata, in: Proc. AAAI-92, San Jose, CA, 1992, pp. 453–458.

[42] R. Weigel, Compiling constraint satisfaction problems, Artificial Intelligence 115 (1999) 257–287.