Generalizing Consistency and other Constraint Properties to Quantified Constraints

LUCAS BORDEAUX
Microsoft Research
MARCO CADOLI
Università di Roma "La Sapienza"
TONI MANCINI
Università di Roma "La Sapienza"

Quantified constraints and Quantified Boolean Formulae are typically much more difficult to reason with than classical constraints, because quantifier alternation makes the usual notion of *solution* inappropriate. As a consequence, basic properties of Constraint Satisfaction Problems (CSP), such as consistency or substitutability, are not completely understood in the quantified case. These properties are important because they are the basis of most of the reasoning methods used to solve classical (existentially quantified) constraints, and one would like to benefit from similar reasoning methods in the resolution of quantified constraints.

In this paper, we show that most of the properties that are used by solvers for CSP can be generalized to quantified CSP. This requires a re-thinking of a number of basic concepts; in particular, we propose a notion of *outcome* that generalizes the classical notion of solution and on which all definitions are based. We propose a systematic study of the relations which hold between these properties, as well as complexity results regarding the decision of these properties. Finally, and since these problems are typically intractable, we generalize the approach used in CSP and propose weaker, easier to check notions based on *locality*, which allow to detect these properties incompletely but in polynomial time.

Categories and Subject Descriptors: F4.1 [**Mathematical Logic and Formal Languages**]: Logic and Constraint Programming

General Terms: Algorithms

Additional Key Words and Phrases: Constraint Satisfaction, Quantified Constraints, Quantified Boolean Formulae

Authors'address: Lucas Bordeaux, Microsoft Research Ltd, Roger Needham Building, J J Thomson Avenue, Cambridge CB3 0FB, United Kingdom.
Marco Cadoli and Toni Mancini, Università di Roma *La Sapienza*, Dipartimento di Informatica e Sistemistica, Via Salaria 113, 00198 Rome, Italy.

A preliminary version of this paper appears in *Proc. of the 20th National Conf. on Artificial Intelligence*, published by the American Association of Artificial Intelligence [Bordeaux et al. 2005]. The current paper is a revised and extended version that includes proofs of all results.

Permission to make digital/hard copy of all or part of this material without fee for personal or classroom use provided that the copies are not made or distributed for profit or commercial advantage, the ACM copyright/server notice, the title of the publication, and its date appear, and notice is given that copying is by permission of the ACM, Inc. To copy otherwise, to republish, to post on servers, or to redistribute to lists requires prior specific permission and/or a fee.
© 2008 ACM 1529-3785/08/0400-0001 $5.00

1. INTRODUCTION

1.1 Quantified Constraints

Quantified Constraint Satisfaction Problems (QCSP) have recently received increasing attention from the Artificial Intelligence community [Bordeaux and Monfroy 2002; Börner et al. 2003; Chen 2004a; 2004b; Mamoulis and Stergiou 2004; Gent et al. 2004; Gent et al. 2005; Verger and Bessière 2006; Benedetti et al. 2007; Bordeaux and Zhang 2007]. A large number of solvers are now available for Quantified Boolean Formulae (QBF), which represent the particular case of QCSP where the domains are Boolean and the constraints are clauses, see e.g., [Buening et al. 1995; Cadoli et al. 1999; Cadoli et al. 2002; Rintanen 1999] for early papers on the subject, and [Benedetti 2004; Zhang 2006; Samulowitz et al. 2006; Samulowitz and Bacchus 2006] for descriptions of state-of-the-art techniques for QBF. The reason behind this trend is that QCSP and QBF are natural generalizations of CSP and SAT that allow to model a wide range of problems not directly expressible in these formalisms, and with applications in Artificial Intelligence and hardware and software verification.

1.2 Reasoning with Quantified Constraints

Quantified constraints are typically much more difficult to reason with than classical constraints. To illustrate this difficulty, let us start by an example of property we would like to characterize formally, and let us suggest why a number of naive attempts to define this property are not suitable. Consider the formula:

$$\phi : \quad \forall x \in [3,10].\ \exists y \in [1,15].\ x = y.$$

We would like to "deduce" in a sense that $y \in [1,10]$ or, in other words, that the values $[11,15]$ are *inconsistent* for y. Such a property will in particular be useful to a search-based solver: if this inconsistency is revealed, then the solver can safely save some effort by skipping the branches corresponding to the values $y \in [11,15]$.

A first attempt to define this notion of consistency would be to use an implication and to say, for instance, that value a is consistent for y iff $\phi \rightarrow (y = a)$. But there is clearly a problem with this approach since the occurrence of y on the right-hand-side of the implication is unrelated to its occurrences in formula ϕ, which fall under the scope of a quantifier. One may attempt to circumvent this problem by putting the implication under the scope of the quantifiers, and to say, for instance, that a is consistent for y iff $\forall x \in [3,10].\ \exists y \in [1,15].\ (x = y) \rightarrow (y = a)$. But with this definition any value would in fact be consistent, even $y = 17$. This is because for every x, we have a value for y that falsifies the left-hand side of the implication, thereby making the implication true.

Another approach that looks tempting at first but is also incorrect is to say that a is inconsistent for y iff the formula obtained by fixing the domain of y to $\{a\}$ is false. With this definition we would deduce that all values $a \in [1,15]$ are inconsistent w.r.t. variable y, since the formula $\forall x \in [3,10].\ \exists y \in [a,a].\ x = y$ is false in each and every case. Other variants of these definitions can be considered, but one quickly gets convinced that there is simply no natural way to define consistency, or any other property like *interchangeability*, using implications or instantiations. To define these notions properly in the case of quantified constraints, we need a new framework, which is what this paper proposes.

1.3 Overview of our Contributions

This paper shows that the definitions of consistency, substitutability, and a wider range of CSP properties can be generalized to quantified constraints. Note that all our definitions and results also hold for the particular case of Quantified Boolean Formulas. These definitions, presented in Section 3, are based on a simple game-theoretic framework and in particular on the new notion of *outcome*, which we identify as a key to define and understand all QCSP properties. We then classify these properties in Section 4 by studying the relationships between them (e.g., some can be shown to be stronger than others). We investigate the simplifications allowed by these properties in Section 5, and we characterize the complexity of their associated decision problem in Section 6. Since, as these complexity results show, determining whether any property holds is typically intractable in general, we investigate the use of the same tool which is used in classical CSP, namely *local reasoning*, and we propose in Section 7 local versions of these properties that can be decided in polynomial time. Concluding comments follow in Section 8. We start (Section 2) by introducing some material on QCSP.

2. QUANTIFIED CONSTRAINT SATISFACTION PROBLEMS

In this section, we present all the definitions related to QCSP, as well as some "game-theoretic" material.

2.1 Definition of QCSP

Let $\mathbb{D}$ be a finite set. Given a finite set V of variables, a V-tuple t with components in $\mathbb{D}$, is a mapping that associates a value $t_x \in \mathbb{D}$ to every $x \in V$; a V-*relation* over $\mathbb{D}$ is a set of V-tuples with components in $\mathbb{D}$.

Definition 1. A *Quantified Constraint Satisfaction Problem* (QCSP) is a tuple $\phi = \langle X, Q, D, C \rangle$ where: $X = \{x_1, \ldots, x_n\}$ is a linearly ordered, finite set of *variables*; Q associates to each variable $x_i \in X$ a *quantifier* $Q_{x_i} \in \{\forall, \exists\}$; D associates to every variable $x_i \in X$ a *domain* $D_{x_i} \subseteq \mathbb{D}$; and $C = \{c_1, \ldots c_m\}$ is a finite set of *constraints*, each of which is a V-relation with components in $\mathbb{D}$ for some $V \subseteq X$.

Note that we are using a definition where Quantified Constraint Satisfaction Problems are in *prenex* form, relying on the fact that non-prenex formulae can easily be put into prenex form. This form is best-suited for the game-theoretic material on which the definitions proposed in this paper rely. Another direction, explored in [Benedetti et al. 2007], is to develop reasoning methods that directly deal with non-prenex formulae. This other approach essentially presents practical advantages, both for problem modeling and in algorithmic terms.

2.1.1 *Notation*

—The notation $\prod_{x \in V} D_x$, where $V \subseteq X$ is a subset of variables, will denote a *Cartesian product* of domains, i.e., the set of V-tuples t that are such that $t_x \in D_x$ for each $x \in V$.

—The notation $t[x := a]$, where t is an X-tuple, $x \in X$ is a variable and $a \in \mathbb{D}$ is a value, will be used for *instantiation*, i.e., it denotes the tuple t' defined by $t'_x = a$ and $t'_y = t_y$ for each $y \in X \setminus \{x\}$.

—The notation $t|_U$, where t is a V-tuple and $U \subseteq V$ is a subset of its variables, will denote the *restriction* of t to U, i.e., the U-tuple t' such that $t'_x = t_x$ for each $x \in U$. (Note that t is undefined on every $y \in V \setminus U$.)

We use the following shorthands to denote the set of existential (resp. universal) variables, the set of variables of index $\leq j$, and the sets of existential/universal variables of index $\leq j$:

$$\begin{array}{ll} & X_j = \{x_i \in X \mid i \leq j\} \\ E = \{x_i \in X \mid Q_{x_i} = \exists\} & E_j = E \cap X_j \\ A = \{x_i \in X \mid Q_{x_i} = \forall\} & A_j = A \cap X_j \end{array}$$

2.1.2 *Satisfaction, Solutions and Truth of a QCSP.* Given a QCSP $\phi = \langle X, Q, D, C \rangle$ as in Definition 1, an X-tuple t is said to *satisfy* the set of constraints C if $t|_V \in c$ for each V-relation $c \in C$. The set of X-tuples satisfying all constraints of ϕ is called the set of *solutions* to C and is denoted by sol^{ϕ}.

Although QCSPs are defined in a form that closely follows the traditional definition of CSPs, the most immediate way to define their semantics is to use rudimentary logic with equality. (We shall see in the next section that we can in a second step forget about the logic and think alternatively in terms of tuples and functions when this is more convenient.) A QCSP $\langle X, Q, D, C \rangle$ represents a logical formula whose vocabulary includes n names for the variables (for convenience, we simply denote these names as $x_1 \dots x_n$) and m names for the constraints $(c_1 \dots c_m)$. The formula is defined as:

$$F: \quad Q_{x_1} x_1 \in D_{x_1} \dots Q_{x_n} x_n \in D_{x_n} \ (F_1 \wedge \dots \wedge F_m).$$

where each F_i is obtained from the corresponding V-relation c_i: let $\{y_1, .., y_p\} = V$, then F_i is simply the formula $c_i(y_1, .., y_p)$, i.e., we apply the name of the constraint to the right argument list. Each D_{x_i} explicitly lists the values specified in the QCSP definition, for instance $\forall x \in \{a, b\}.\phi$ is a shorthand for $\forall x.(x = a \vee x = b) \rightarrow \phi$.

Let I be the interpretation function that associates to each constraint name the corresponding relation; the QCSP is said to be *true* if formula F is true in the domain $\mathbb{D}$ and w.r.t. the interpretation I, i.e., iff $\langle \mathbb{D}, I \rangle \models F$.

2.2 Game-Theoretic Material

Quantifier alternation is best understood using an "adversarial" or "game-theoretic" viewpoint, where two players interact. One of them is allowed to choose the values for the existential variables, and its aim is to ultimately make the formula true, while the other assigns the universal variables and aims at falsifying it. We introduce several definitions leading to our central notion of *outcome*, which will be shown to shed light on the definition of properties in the next section. Our presentation of the basic game-theoretic material is inspired from [Chen 2004b], who uses a similar notion of winning strategy.

The following QCSP (written using the usual, self-explanatory logical notation rather than in the form of a tuple $\langle X, Q, D, C \rangle$) will be used to illustrate the notions throughout this sub-section:

$$\begin{aligned} &\exists x_1 \in [1,10].\ \forall x_2 \in [1,10].\ \exists x_3 \in [1,10]. \\ &\qquad \forall x_4 \in [1,10].\ \exists x_5 \in [1,10].\ \ x_1 + x_2 + x_3 + x_4 + x_5 = 30 \end{aligned} \tag{1}$$

This formula can be thought of as a game between two players assigning, respectively, the odd and even variables. The game is in 5 moves and the players draw in turn between 1 and 10 sticks from a heap containing originally 30 sticks; the existential player wins if she manages to empty the heap during her last move.

2.2.1 *Strategies.* The first notion we need is the notion of *strategy*:

Definition 2. A strategy is a family $\{s_{x_i} \mid x_i \in E\}$ where each s_{x_i} is a function of signature $\left(\prod_{y \in A_{i-1}} D_y\right) \to D_{x_i}$.

In other words, a strategy defines for each existential variable x_i a function that specifies which value to pick for x_i depending on the values assigned to the universal variables that precede it. Note in particular that, if the first k variables of the problem are quantified existentially, we have for every $i \leq k$ a constant $s_{x_i} \in D_{x_i}$ which defines which value should directly be assigned to variable x_i.

Example 1. *A strategy for the QCSP (1) can be defined by* $s_{x_1}() = 8$*;* s_{x_3} *associates to every* $\{x_2\}$*-tuple* t *the value* $s_{x_3}(t) = 11 - t_{x_2}$ *and* s_{x_5} *associates to every* $\{x_2, x_4\}$*-tuple* t *the value* $s_{x_5}(t) = 11 - t_{x_4}$. *This strategy specifies that we first draw 8 sticks, then for the next moves we shall draw 11 minus what the opponent just drew.*

2.2.2 *Scenarios.* The tuple of values that will eventually be assigned to the variables of the problem depends on two things: 1) the strategy we have fixed *a priori*, and 2) the sequence of choices of the "adversary", i.e., the values that are assigned to the universal variables. Given a particular strategy, a number of potential *scenarios* may therefore arise, depending on what the adversary will do. These scenarios are defined as follows:

Definition 3. The set of scenarios of a strategy s for a QCSP ϕ, denoted $\mathsf{sce}^{\phi}(s)$, is the set of tuples $t \in \prod_{x \in X} D_x$ such that, for each $x_i \in E$, we have:

$$t_{x_i} = s_{x_i}(t|_{A_{i-1}})$$

In other words, the values for the existential variables are determined by the strategy in function of the values assigned to the universal variables preceding it. There is no restriction, on the contrary, on the values assigned to universal variables: this reflects the fact that we model the viewpoint of the existential player, and the adversary may play whatever she wishes to play.

Example 1. *(Ctd.) An example of scenario for the strategy defined previously is the tuple defined by* $x_1 = 8, x_2 = 4, x_3 = 7, x_4 = 1, x_5 = 10$. *On the contrary, the tuple* $x_1 = 8, x_2 = 4, x_3 = 7, x_4 = 1, x_5 = 5$ *is not a scenario since the value 5 for* x_5 *does not respect what is specified by* s_{x_5}.

2.2.3 *Winning Strategies.* Of particular interest are the strategies whose scenarios are all solutions. We call them *winning strategies*:

Definition 4. A strategy s is a winning strategy for the QCSP ϕ if every scenario $t \in \mathsf{sce}^{\phi}(s)$ satisfies the constraints of ϕ (in other words: if $\mathsf{sce}^{\phi}(s) \subseteq \mathsf{sol}^{\phi}$).

We denote by WIN^{ϕ} the set of winning strategies of the QCSP ϕ.

Example 1. *(Ctd.) In the strategy s defined in Example 1, any scenario t is of the form $x_1 = 8, x_2 = a, x_3 = 11 - a, x_4 = b, x_5 = 11 - b$. As a result the sum always evaluates to $8 + a + 11 - a + b + 11 - b = 30$ and s is therefore a winning strategy. In fact, this strategy is the only winning one; one can check, for instance, that the strategy s' defined by $s'_{x_1}() = 7$; $s'_{x_3}(t) = 7$ and $s'_{x_5}(t) = 7$ is not winning.*

The following proposition is essential in that it justifies the use of the game-theoretic approach[1]:

Proposition 1. *A QCSP is true (as defined in Section 2.1.2) iff it has a winning strategy.*

2.2.4 *Outcome.* Whereas the preceding material is well-known and is used, for instance, in [Chen 2004b], we introduce the following new notion:

Definition 5. The set of outcomes of a QCSP ϕ, denoted out^{ϕ}, is the set of all scenarios of all its winning strategies, i.e., it is defined as:

$$\mathsf{out}^{\phi} \;=\; \bigcup_{s \in \mathsf{WIN}^{\phi}} \mathsf{sce}^{\phi}(s)$$

Example 1. *(Ctd.) Since our example has a unique winning strategy it is easy to characterise its set of outcomes: these are all the tuples of the form $x_1 = 8, x_2 = a, x_3 = 11 - a, x_4 = b, x_5 = 11 - b$, with $a, b \in [1, 10]$.*

Outcomes are related to the classical notion of solution in the following way: in general any outcome satisfies the set of constraints C, so we have $\mathsf{out}^{\phi} \subseteq \mathsf{sol}^{\phi}$, and the equality $\mathsf{out}^{\phi} = \mathsf{sol}^{\phi}$ holds if all variables are existential. On the other hand let us emphasize the fact that not all solutions are necessarily outcomes in general: in our example the tuple $x_1 = 6, x_2 = 6, x_3 = 6, x_4 = 6, x_5 = 6$ is for instance a solution as it satisfies the unique constraint $(x_1 + x_2 + x_3 + x_4 + x_5 = 30)$. But there is no winning strategy whose set of scenarios includes this particular tuple, and it is therefore not an outcome.

The notion of outcome is a generalization of the notion of solution that takes into account the quantifier prefix of the constraints. Our claim in the following is that *outcomes play a role as central for QCSP as the notion of solution does in CSP, and that most definitions can be based on this notion.*

2.2.5 *Summary of the notions and notations.* To summarize, we have defined 3 sets of tuples (sol^{ϕ}: the set of solutions, $\mathsf{sce}^{\phi}(s)$: the set of scenarios of strategy s, and out^{ϕ}: the set of outcomes) and one set of strategies (WIN^{ϕ}: the set of winning

[1]Proofs of all propositions can be found in the online Appendix.

strategies). All the game-theoretic notions we have introduced are illustrated in Fig. 1, where we consider the QCSP represented by the logical formula:

$$\exists x_1 \in [2, 3]\ \forall x_2 \in [3, 4]\ \exists x_3 \in [3, 6].\ x_1 + x_2 \leq x_3. \tag{2}$$

And and *or* labels on the nodes correspond to universal and existential quantifiers, respectively. The solutions are all triples $\langle x_1, x_2, x_3\rangle$ s.t. $x_1 + x_2 \leq x_3$. The only two winning strategies assign x_1 to 2: one (s_1) systematically assigns x_3 to 6 while the 2nd one (s_2) assigns it to $x_2 + 2$ (note that each strategy is constrained to choose one unique branch for each existential node). The scenarios of s_1 and s_2 are therefore those indicated, while the set of outcomes of the QCSP is the union of the scenarios of s_1 and s_2 (also shown in bold line).

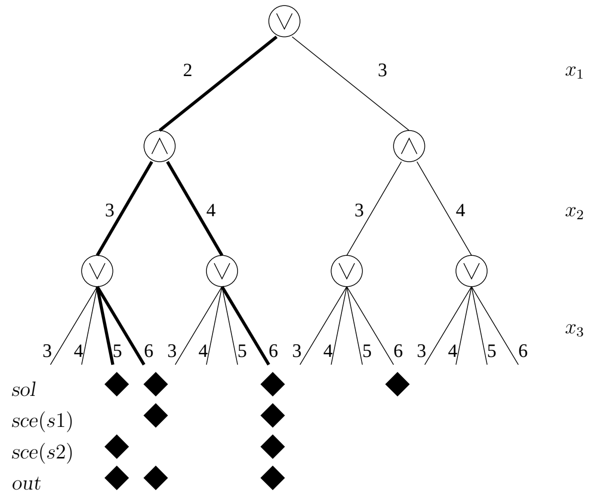

Fig. 1. A summary of the game-theoretic notions used in this paper.

3. DEFINITIONS OF THE CSP PROPERTIES

3.1 Informal Definitions of the Properties

A major part of the CSP literature aims at identifying properties of particular values of some variables. The goal is typically to simplify the problem by ruling out the possibility that a variable x_i can be assigned to a value a. This can be done when one of the following properties holds, with respect to variable x_i:

—Value a is guaranteed not to participate in any solution: a is *inconsistent* for x_i [Mackworth 1977].

—Another value b can replace a in any solution involving it: a is *substitutable* to b for x_i [Freuder 1991].

—All solutions involving a can use another value instead: a is *removable* for x_i [Bordeaux et al. 2004].

On the contrary, some other properties give an indication that instantiating x_i to a is a good idea:

—All solutions assign value a to variable x_i: a is *implied* for x_i [Monasson et al. 1999];

—We have the guarantee that if in any solution we replace the value assigned to a variable x_i by a particular value a we still keep a solution: a is said to be *fixable* for x_i [Bordeaux et al. 2004].

While all the preceding are properties of particular *values*, related properties of *variables* are also of interest [Bordeaux et al. 2004]:

—The value assigned to a variable x_i is forced to a unique possibility: x_i is *determined.*

—The value of variable x_i is a function of the values of other variables: x_i is *dependent.*

—Whether a tuple is a solution or not does not depend on the value assigned to variable x_i: x_i is *irrelevant.*

In this section, we propose generalizations of the definitions of the main CSP properties to quantified constraints. For the sake of homogeneity, we adopt the terminology used in the paper [Bordeaux et al. 2004] for the names of the properties.

We adopt a predicate notation and write, e.g., $p^{\phi}(x_i, a)$ for the statement "value a has property p for variable x_i (in QCSP ϕ)". The superscript ϕ will be omitted in order to simplify the notation whenever there is no ambiguity regarding which QCSP is considered.

We present our definitions in two steps: Section 3.2 introduces the basic definitions, which we call *deep* definitions, for reasons that will become clear in the rest of this section. We then notice in Section 3.3 that the properties can be made more general, leading to our *shallow* definitions.

3.2 Basic Definitions

The first definitions we propose are identified by a d prefix and qualified as "deep" when an ambiguity with the definitions in forthcoming Section 3.3 is possible. They are based on directly rephrasing the original CSP definitions, but using the notion of outcomes in place of solutions:

Definition 6. We define the properties of inconsistency, implication, deep fixability, deep substitutability, deep removability, deep interchangeability, determinacy, deep irrelevance and dependency, as follows, for all $x_i \in X$, $a, b \in D_{x_i}$, $V \subseteq X$:

$$\begin{aligned}
\textit{inconsistent}(x_i, a) &\equiv \forall t \in \mathsf{out}.\ \ t_{x_i} \neq a \\
\textit{implied}(x_i, a) &\equiv \forall t \in \mathsf{out}.\ \ t_{x_i} = a \\
\\
\textit{d-fixable}(x_i, a) &\equiv \forall t \in \mathsf{out}.\ \ t[x_i := a] \in \mathsf{out} \\
\\
\textit{d-substitutable}(x_i, a, b) &\equiv \forall t \in \mathsf{out}.\ \ (t_{x_i} = a) \rightarrow (t[x_i := b] \in \mathsf{out}) \\
\\
\textit{d-removable}(x_i, a) &\equiv \forall t \in \mathsf{out}.\ \ (t_{x_i} = a) \rightarrow (\exists b \neq a.\ \ t[x_i := b] \in \mathsf{out}) \\
\\
\textit{d-interchangeable}(x_i, a, b) &\equiv \textit{d-substitutable}(x_i, a, b) \wedge \textit{d-substitutable}(x_i, b, a) \\
\\
\textit{determined}(x_i) &\equiv \forall t \in \mathsf{out}.\ \ \forall b \neq t_{x_i}.\ \ t[x_i := b] \notin \mathsf{out} \\
\textit{d-irrelevant}(x_i) &\equiv \forall t \in \mathsf{out}.\ \ \forall b \in D_{x_i}.\ t[x_i := b] \in \mathsf{out} \\
\\
\textit{dependent}(V, x_i) &\equiv \forall t, t' \in \mathsf{out}.\ \ (t|_V = t'|_V) \rightarrow (t_{x_i} = t'_{x_i})
\end{aligned}$$

We note that the definition of consistency is equivalent to the one proposed in [Bordeaux and Monfroy 2002]; it is nevertheless expressed in a simpler and more elegant way that avoids explicitly dealing with And/Or trees. All other definitions are new.

EXAMPLE 2. *Consider the QCSP:*

$$\exists x_1 \in [2, 3]\ \forall x_2 \in [3, 4]\ \exists x_3 \in [3, 6].\ x_1 + x_2 \leq x_3$$

(cf. Fig. 1). We have: $\textit{inconsistent}(x_1, 3)$*,* $\textit{inconsistent}(x_3, 3)$*,* $\textit{inconsistent}(x_3, 4)$*,* $\textit{d-substitutable}(x_3, 5, 6)$*,* $\textit{d-fixable}(x_3, 6)$*,* $\textit{d-removable}(x_3, 5)$*, and* $\textit{implied}(x_1, 2)$*.*

A choice we made in Definition 6 requires a justification: if we consider, for instance, fixability, one may think that a more general definition could be obtained if we wrote $\forall t \in \mathsf{out}.\ t[x_i := a] \in \underline{\mathsf{sol}}$ instead of $\forall t \in \mathsf{out}.\ t[x_i := a] \in \mathsf{out}$. Similarly, the question arises whether the other definitions that involve the set out in the right-hand side of an implication (either implicitly or explicitly) could be strengthened be using the set sol instead. This is not the case: except for one property, namely *determinacy*, the modified definitions would actually be strictly equivalent:

PROPOSITION 2. *Deep fixability could equivalently be defined by the condition* $\forall t \in \mathsf{out}. t[x_i := a] \in \mathsf{sol}$*; Deep substitutability could be equivalently defined by* $\forall t \in \mathsf{out}.\ (t_{x_i} = a) \rightarrow (t[x_i := b] \in \mathsf{sol})$*; deep removability by* $\forall t \in \mathsf{out}.(t_{x_i} = a) \rightarrow (\exists b \neq a.t[x_i := b] \in \mathsf{sol})$*; and deep irrelevance by* $\forall t \in \mathsf{out}.\forall b \in D_{x_i}.\ t[x_i := b] \in \mathsf{sol}$*.*

This proposition will play a role in the proof of other results. Defining determinacy by $\forall t \in \mathsf{out}.\forall b \neq t_{x_i}.t[x_i := b] \notin \mathsf{sol}$, instead of the definition we used. i.e., $\forall t \in \mathsf{out}.\forall b \neq t_{x_i}.t[x_i := b] \notin \mathsf{out}$, would on the contrary give a slightly different notion: we note that in this case (because of the negation implicitly on the right-hand side of the implication, i.e., $t[x_i := b] \notin \mathsf{out}$), the definition would become *weaker*. For instance, in Fig. 1, we would not have $\textit{determined}(x_1, 2)$ because the tuple $t = \langle 2, 3, 6\rangle$ is such that $t[x_1 := 3] \in \mathsf{sol}$.

3.3 Generalization: Shallow Definitions

The previous definitions are correct in a sense that will be made formal in Section 5. They are nevertheless overly restrictive in some cases, as the following example shows:

Example 3. *Consider the QCSP:*

$$\forall x_1 \in [1,2]\ \exists x_2 \in [3,4]\ \exists x_3 \in [4,6].\ x_1 + x_2 = x_3.$$

The winning strategies can make arbitrary choices for x_2 as long as they set x_3 to the value $x_1 + x_2$, and the outcomes are the triples $\langle 1,3,4\rangle$, $\langle 1,4,5\rangle$, $\langle 2,3,5\rangle$, $\langle 2,4,6\rangle$. Note that for variable x_2, neither values 3 nor 4 are deep-fixable, and none is deep-substitutable to the other. This somehow goes against the intuition that we are indeed free to choose the value for x_2.

The reason why our previous definition did not capture this case is that it takes into account the values of the variables occurring *after* the considered variable: values 3 and 4 are interchangeable (for instance) only if the QCSPs resulting from these instantiations can be solved *using the same strategy* for all the subsequent choices—this is why we called these definitions *deep* (with a *d* prefix). On the contrary, we can formulate *shallow* definitions of the properties, which accept value 4 as a valid substitute for 3 because *in any sequence of choices leading to the possibility of choosing 3 for x_2, value 4 is also a valid option.*

Definition 7. We define the properties of shallow fixability, substitutability, removability, interchangeability, and irrelevance, as follows:

$$\begin{aligned}
&\textit{s-fixable}(x_i, a) \equiv \\
&\quad \forall t \in \mathsf{out}.\ \exists t' \in \mathsf{out}.\ \big(t|_{X_{i-1}} = t'|_{X_{i-1}} \wedge \ \ t'_{x_i} = a \big)
\end{aligned}$$

$$\begin{aligned}
&\textit{s-substitutable}(x_i, a, b) \equiv \\
&\quad \forall t \in \mathsf{out}.\ t_{x_i} = a \rightarrow \\
&\qquad \exists t' \in \mathsf{out}.\ \big((t|_{X_{i-1}} = t'|_{X_{i-1}})\ \wedge\ (t'_{x_i} = b) \big)
\end{aligned}$$

$$\begin{aligned}
&\textit{s-removable}(x_i, a) \equiv \\
&\quad \forall t \in \mathsf{out}.\ t_{x_i} = a \rightarrow \\
&\qquad \exists t' \in \mathsf{out}.\ \big(t|_{X_{i-1}} = t'|_{X_{i-1}} \wedge t'_{x_i} \neq a \big)
\end{aligned}$$

$$\begin{aligned}
&\textit{s-interchangeable}(x_i, a, b) \equiv \\
&\quad \textit{s-substitutable}(x_i, a, b) \wedge \textit{s-substitutable}(x_i, b, a)
\end{aligned}$$

$$\begin{aligned}
&\textit{s-irrelevant}(x_i) \equiv \\
&\quad \forall t \in \mathsf{out}.\ \forall b \in D_{x_i}. \\
&\qquad \exists t' \in \mathsf{out}.\ \big((t|_{X_{i-1}} = t'|_{X_{i-1}})\ \wedge\ (t'_{x_i} = b) \big)
\end{aligned}$$

One can check that with these definitions we handle Example 3 as expected:

Example 3. *(Ctd.) Considering again the QCSP:*

$$\forall x_1 \in [1,2]\ \exists x_2 \in [3,4]\ \exists x_3 \in [4,6].\ x_1 + x_2 = x_3,$$

values 3 and 4 are shallow-interchangeable for variable x_2 (both values are also shallow-fixable, shallow-removable, and variable x_2 is in fact shallow-irrelevant). The reason is that for each outcome t that assigns value 3 to x_2, there exists a tuple t' such that $t'_{x_1} = t_{x_1}$ and $t'_{x_2} = 4$ (to $t = \langle 1,3,4 \rangle$ corresponds $t' = \langle 1,4,5 \rangle$; to $\langle 2,3,5 \rangle$ corresponds $\langle 2,4,6 \rangle$), and vice-versa.

This can be seen pictorially in Fig. 2. On the left-hand side, we see why values 3 and 4 are not (for instance) deep-interchangeable for x_2: the outcomes (branches) going through these values are indeed different. Now on the right-hand side we see the viewpoint of the shallow *definitions: the strategy is only considered* up to variable x_2*, and it is clear, then, that values 3 and 4 are interchangeable.*

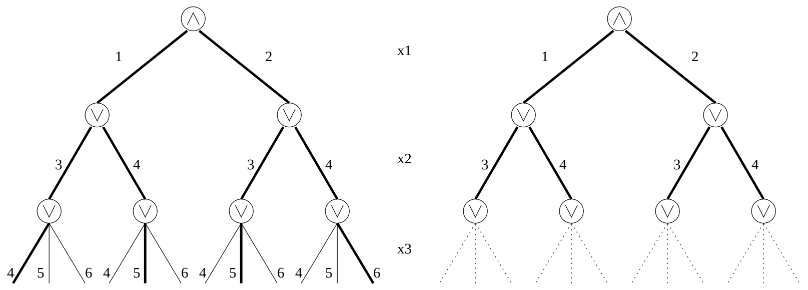

Fig. 2. Illustration of the notion of *shallow* properties, as opposed to the *deep* definitions.

We last remark that the distinction we have introduced between *deep* and *shallow* only makes sense for a subset of the properties. It is easy to see, for instance, that a shallow definition of *inconsistency* would make no difference: this notion is defined by the statement $\forall t \in \mathsf{out}.\ t_{x_i} \neq a$, and this is equivalent to $\forall t \in \mathsf{out}.\ (t|_{X_i})_{x_i} \neq a$.

4. RELATIONS BETWEEN THE PROPERTIES

This section gives a number of results establishing the relations between the classes of properties (e.g., deep, shallow) and between the properties themselves (substitutability, determinacy, etc.). These results will also be used later (Section 5) to prove that our definitions are useful, in that they allow to simplify the considered QCSP while preserving some form of equivalence.

4.1 Relations between Classes of Properties

The basic relations between classical, deep, and shallow definitions, are the following: deep definitions are *more general* than basic, existential ones, and the shallow definitions are *more general* than the deep ones, in a sense that is explained formally in the following.

4.1.1 *Deep definitions vs. classical definitions.* We first note that, in the particular case where the quantifiers are all existential, the deep definitions of the properties (Definition 6) correspond to the classical CSP notions, simply because we have $\mathsf{out} = \mathsf{sol}$ in that case; in other words our definitions truly are *generalizations* of the classical definitions. In the general case, when the quantifiers are not restricted to be existential, we can still ignore the quantifier prefix and apply the classical definitions to the resulting existentially quantified CSP. The relations between the original QCSP and the relaxed CSP are the following:

(1) The deductions made using the classical definitions are *correct*: a property detected on the existentially quantified CSP, using the classical definitions, will also hold for the QCSP.

(2) This reasoning is *incomplete*: if we do not take into account the quantifier prefix as our new definitions do, some properties cannot be detected.

The *correctness* can be stated formally as follows:

PROPOSITION 3. *Let $\phi = \langle X, Q, D, C\rangle$ be a QCSP and let ψ be the same QCSP but in which all quantifiers are existential, i.e., $\psi = \langle X, Q', D, C\rangle$, with $Q'_x = \exists$, for all $x \in X$. We have (forall x_i, a, b, V):*

—*inconsistent*$^\psi(x_i, a) \rightarrow$ *inconsistent*$^\phi(x_i, a)$;
—*d-fixable*$^\psi(x_i, a) \rightarrow$ *d-fixable*$^\phi(x_i, a)$;
—*d-substitutable*$^\psi(x_i, a, b) \rightarrow$ *d-substitutable*$^\phi(x_i, a, b)$;
—*d-removable*$^\psi(x_i, a) \rightarrow$ *d-removable*$^\phi(x_i, a)$;
—*d-interchangeable*$^\psi(x_i, a, b) \rightarrow$ *d-interchangeable*$^\phi(x_i, a, b)$;
—*determined*$^\psi(x_i) \rightarrow$ *determined*$^\phi(x_i)$;
—*d-irrelevant*$^\psi(x_i) \rightarrow$ *d-irrelevant*$^\phi(x_i)$;
—*dependent*$^\psi(V, x_i) \rightarrow$ *dependent*$^\phi(V, x_i)$.

We note that the idea of relaxing universal quantifiers and approximating a QCSP by a classical, existential CSP, has been considered implicitly by several authors: the solver presented in [Benedetti et al. 2007] is built on top of a classical CP solver and its propagation mechanism essentially relies on the classical notion of inconsistency; other authors [Mamoulis and Stergiou 2004; Gent et al. 2005] have investigated the use of substitutability in QCSP; here again the notion they have used was essentially the classical, existential one.

Replacing a universal quantifier by an existential one is but one way to obtain a *relaxation* of a QCSP. In [Ferguson and O'Sullivan 2007], a more comprehensive list of relaxation techniques is studied. Interestingly this work essentially defines a relaxation as a transformation that guarantees that *if the relaxation is false, then so is the original problem.* In other words, the notion of relaxation is based on the truth of the QCSP. Proposition 3 shows that *quantifier relaxation* provides a way to do approximate reasoning on other properties than *truth*.

The *incompleteness* of the reasoning on the existential relaxation is easily seen on an example:

EXAMPLE 2. *(Ctd.) Consider the QCSP:*

$$\exists x_1 \in [2, 3]\ \forall x_2 \in [3, 4]\ \exists x_3 \in [3, 6].\ x_1 + x_2 \leq x_3$$

(See Fig. 1.) Noticeable properties are: inconsistent$(x_1, 3)$*,* implied$(x_1, 2)$*,* d-fixable$(x_1, 2)$*,* d-removable$(x_1, 3)$*,* d-substitutable$(x_1, 3, 2)$*,* determined(x_1)*.*

On the contrary if we apply the classical definition or, equivalently, consider the CSP $\exists x_1 \in [2, 3]\ \exists x_2 \in [3, 4]\ \exists x_3 \in [3, 6].\ x_1 + x_2 \leq x_3$, none *of the properties holds, because of the tuple $\langle 3, 3, 6\rangle$.*

This confirms that the properties we have defined are new notions which do make a difference compared to classical CSP notions, and which allow a finer reasoning taking into account the quantifier prefix as well as the constraints themselves.

4.1.2 *Shallow properties vs. deep properties.* To complete the picture, we have the following relations between deep and shallow notions (the deep ones are more restrictive):

PROPOSITION 4. *For all variables x_i and values a and b, we have:*

—*d-fixable*$(x_i, a) \rightarrow$ *s-fixable*(x_i, a);
—*d-removable*$(x_i, a) \rightarrow$ *s-removable*(x_i, a);
—*d-substitutable*$(x_i, a, b) \rightarrow$ *s-substitutable*(x_i, a, b);
—*d-interchangeable*$(x_i, a, b) \rightarrow$ *s-interchangeable*(x_i, a, b);
—*d-irrelevant*$(x_i) \rightarrow$ *s-irrelevant*(x_i).

Note that whether a property holds is always dependent on the quantification order. In the case of shallow definitions, this is even more true, because the ordering matters even within a block of variables *of the same nature*, for instance when the quantifiers are all existential. To see that, consider the QCSP:

$$\exists x_1 \in [1,2]\ \exists x_2 \in [3,4]\ \exists x_3 \in [4,6].\ x_1 + x_2 = x_3.$$

Value 1 is shallow-substitutable to 2 for x_1, and x_1 is shallow-irrelevant, while 1 is not deep-substitutable to 2 for x_1 (i.e., substitutable in the classical sense), nor is x_1 deep-irrelevant. The intuition behind this is that here we consider that x_1 is assigned first, and *at this step* the two choices are equivalent. In other words, the property holds *because we are considering the ordering* x_1, x_2, x_3.

Interestingly, shallow properties, and shallow substitutability in particular, provide a new, general form of properties even for the case of classical CSP. These properties are more general because they take into account information on a particular variable ordering. An interesting question is to determine the variable ordering that allows to detect the highest number of substitutability properties in a given CSP.

4.2 Relations between Properties

As in the classical case [Bordeaux et al. 2004], we also have relations between the properties, for instance a value that is implied is also deep-fixable (and therefore also shallow fixable); a variable that is (deep/shallow) irrelevant is also (deep/shallow) fixable to any value, etc. We list the most remarkable of these relations in the next proposition:

PROPOSITION 5. *The following relations hold between the properties (forall x_i, a and b):*

—*inconsistent*$(x_i, a) \rightarrow \forall b \in D_{x_i}.$ *d-substitutable*(x_i, a, b);
—*implied*$(x_i, a) \leftrightarrow \forall b \in D_{x_i} \setminus \{a\}.$ *inconsistent*(x_i, b);
—*implied*$(x_i, a) \rightarrow$ *d-fixable*(x_i, a);
—*inconsistent*$(x_i, a) \rightarrow$ *d-removable*(x_i, a);

—$\exists b \in D_{x_i} \setminus \{a\}.$ *d-substitutable*$(x_i, a, b) \rightarrow$ *d-removable*(x_i, a)*;*
—$\exists b \in D_{x_i} \setminus \{a\}.$ *s-substitutable*$(x_i, a, b) \rightarrow$ *s-removable*(x_i, a)*;*
—*d-fixable*$(x_i, b) \leftrightarrow \forall a \in D_{x_i}.$ *d-substitutable*(x_i, a, b)*;*
—*s-fixable*$(x_i, b) \leftrightarrow \forall a \in D_{x_i}.$ *s-substitutable*(x_i, a, b)*;*
—*d-irrelevant*$(x_i) \leftrightarrow \forall a \in D_{x_i}.$ *d-fixable*(x_i, a)*;*
—*s-irrelevant*$(x_i) \leftrightarrow \forall a \in D_{x_i}.$ *s-fixable*(x_i, a)*.*

5. SIMPLIFICATIONS ALLOWED WHEN THE PROPERTIES HOLD

The goal of reasoning on the properties of a QCSP is typically to simplify the problem. In the cases we are interested in, this can be done in two ways: (1) by removing an element from the list of values to consider for one of the variables, or (2) by instantiating a variable to a particular value. Such simplifications are helpful for backtrack search algorithms, which are typically considered when solving QCSP.

We now show that the properties we defined allow simplifications that are *correct*, in the sense that they do not alter the truth of the QCSP:

—If a value is removable for a given existential variable, then removing the value from the domain of that variable does not change the truth of the problem.
—If a value is fixable to a particular value for a given existential variable, then instantiating the variable to this value does not change the truth of the problem.

The interest of the other properties lies essentially in their relation with the two fundamental properties of removability and fixability, as expressed by Prop. 5. For instance, an implied value is of interest essentially because it is fixable, and an irrelevant variable is of interest essentially because it is fixable to any value of its domain. Similarly, the interest of, e.g., inconsistent and substitutable values is that they are removable. We therefore focus on proving the correctness of the two notions of removability and fixability, and we will consider their shallow forms: recall that, by Prop. 4, the shallow definitions are the stronger ones; a value which is deep-removable or deep-fixable is also shallow-removable or shallow-fixable, respectively.

5.1 Simplifying Existental Variables

Our whole game-theoretic approach is naturally biased towards existential variables: the notion of strategy considers that the values for the universal variables can be arbitrary, and specifies the values that should be taken for the existential ones. As a consequence, the approach is more naturally fitted to make deductions on the existential variables, and we first focus on this case.

The simplifications allowed for an existential variable when the removability property holds rely on the following proposition:

PROPOSITION 6. *Let $\phi = \langle X, Q, D, C \rangle$ be a QCSP in which value $a \in D_{x_i}$ is shallow-removable for an existential variable x_i, and let ϕ' denote the same QCSP in which value a is effectively removed (i.e., $\phi' = \langle X, Q, D', C \rangle$ where $D'_{x_i} = D_{x_i} \setminus \{a\}$ and $D'_{x_j} = D_{x_j}, \forall j \neq i$). Then ϕ is true iff ϕ' is true.*

The simplifications allowed for an existential variable when the fixability property holds rely on the following proposition:

PROPOSITION 7. *Let $\phi = \langle X, Q, D, C\rangle$ be a QCSP in which value $a \in D_{x_i}$ is shallow-fixable for an existential variable x_i, and let ϕ' denote the same QCSP in which value a is effectively fixed (i.e., $\phi' = \langle X, Q, D', C\rangle$ where $D'_{x_i} = \{a\}$ and $D'_{x_j} = D_{x_j}, \forall j \neq i$). Then ϕ is true iff ϕ' is true.*

5.2 Simplifying Universal Variables

To allow a proper, symmetric treatment of all variables of QCSPs it is necessary to also develop deduction techniques for universal variables. A first type of reasoning is enabled by the following observation, which allows to detect some simple cases of false QCSPs:

PROPOSITION 8. *Let $\phi = \langle X, Q, D, C\rangle$ be a QCSP in which a value $a \in D_{x_i}$ is inconsistent for a universal variable x_i. Then ϕ is false.*

Richer deductions can be made following an approach that has been suggested by several authors in the literature (e.g., workshop version of [Ansótegui et al. 2005]): to reason on the universal variables, which represent the "moves of the opponent", we can reason on the negation of the formula, which captures the "winning strategies of the opponent". The rest of this section briefly develops this approach.

5.2.1 *Dual versions of the properties.* We say that a value is *dual*-shallow-removable if it is shallow-removable in the negation of the considered QCSP, i.e.,

$$\textit{dual-s-removable}^{\phi}(x_i, a) \;\equiv\; \textit{s-removable}^{\neg\phi}(x_i, a)$$

We can define a "dual" version of all other properties in a similar fashion: notably, a value is said to be *dual*-shallow-fixable if it is shallow-fixable in the negation and *dual*-inconsistent if it is inconsistent in the negation. The simplifications allowed for a universal variable when the removability property holds rely on the following proposition:

PROPOSITION 9. *Let $\phi = \langle X, Q, D, C\rangle$ be a QCSP in which value $a \in D_{x_i}$ is dual-shallow-removable for a universal variable x_i, and let ϕ' denote the same QCSP in which value a is effectively removed (i.e., $\phi' = \langle X, Q, D', C\rangle$ where $D'_{x_i} = D_{x_i} \setminus \{a\}$ and $D'_{x_j} = D_{x_j}, \forall j \neq i$). Then ϕ is true iff ϕ' is true.*

The simplifications allowed for a universal variable when the fixability property holds rely on the following proposition:

PROPOSITION 10. *Let $\phi = \langle X, Q, D, C\rangle$ be a QCSP in which value $a \in D_{x_i}$ is dual-shallow-fixable for an universal variable x_i, and let ϕ' denote the same QCSP in which value a is effectively fixed (i.e., $\phi' = \langle X, Q, D', C\rangle$ where $D'_{x_i} = \{a\}$ and $D'_{x_j} = D_{x_j}, \forall j \neq i$). Then ϕ is true iff ϕ' is true.*

5.2.2 *The rôle of purity and fixability for universal variables.* Dual forms of consistency, removability or fixability are present in the literature under two prominent forms, which can be briefly presented as follows. A first approach, represented for instance by [Bordeaux and Zhang 2007], is to take the definitions literally: to reason on existential and universal variables, the idea is simply to maintain two formulas during the search—the original one and its negation. Inconsistencies detected on the "primal" formula, for instance, allow to simplify the existential variables; and inconsistencies detected on the "dual", or negated formula allow to simplify the universal variables.

More interesting perhaps are approaches that reveal dual forms of removability in an indirect way, using less general properties that can be detected using specialized algorithms. An interesting such property is the *pure value* property, proposed by [Gent et al. 2005] and studied further in [Nightingale 2007; Bacchus and Stergiou 2007]. It can be defined as follows:

$$pure(x_i, a) \;\equiv\; \forall t \in \Pi_{y \in X} D_y . \left((t_{x_i} = a) \rightarrow t \in \mathsf{sol} \right)$$

In other words, value a is pure for x_i if we are guaranteed that any tuple that assigns the value a to x_i satisfies the constraints. The pure value property is used in the solver *QCSP-Solve* as a main tool for reasoning on values from the domains of universal variables. Purity is a strong property whose place in our classification is characterized by the following proposition:

PROPOSITION 11. *The pure value property allows to detect values that are dual-inconsistent, i.e.,*

$$pure(x_i, a) \;\rightarrow\; dual\text{-}inconsistent(x_i, a)$$

Since dual-inconsistent values are also dual-removable, Prop. 9 justifies the use of purity for simplifying universal quantifiers. Purity is probably the way of detecting dual-inconsistent values that appears to be best-used in practice. Algorithms for detecting pure values, as well as an analysis of the role played by this property on some problem encodings, can be found in [Nightingale 2007].

A related way of simplifying universal variables that is important, in particular, in QBF solvers, is based on fixability. In a propositional formula in Conjunctive Normal Form, one way of detecting fixability is by using the so-called *monotone literal* rule: if a literal occurs only with the same sign in all active clauses of a formula, then it is clearly fixable to the corresponding polarity. (Although this rule is sometimes also called *pure literal* rule, note that it does not mean that the value is *pure* in the sense of [Gent et al. 2005]—purity is a strictly stronger property, and a monotone literal corresponds indeed to fixability.) In QBF the monotone literal rule is especially important because it allows to prune the universal variables: we can fix the variable to the opposite polarity [Cadoli et al. 2002]. In general, the role of fixability for the pruning of universal variables is stated by the following proposition, which was first reported by [Nightingale 2007]:

PROPOSITION 12. *Let $\phi = \langle X, Q, D, C \rangle$ be a QCSP in which value $a \in D_{x_i}$ is deep-fixable for a universal variable x_i, and let ϕ' denote the same QCSP in which value a is effectively* removed *(i.e., $\phi' = \langle X, Q, D', C \rangle$ where $D'_{x_i} = D_{x_i} \setminus \{a\}$ and $D'_{x_j} = D_{x_j}, \forall j \neq i$). If $D'_{x_i} \neq \emptyset$ (i.e., a was not the only element in D_{x_i}), then ϕ is true iff ϕ' is true.*

It was noticed in [Nightingale 2007] (Theorem 3.2.18) that a pure value is also deep-fixable which, as noticed by the author, is another way of justifying the use of this property for the simplification of universal variables.

6. COMPLEXITY RESULTS

In this section, we study the complexity of the problem of determining whether the properties defined in Definitions 6 and 7 hold. As was to be expected, our results show that the problem is in general intractable, and we essentially obtain PSPACE-completeness results. In other words the complexity of checking one of the properties is typically the same as the complexity of determining whether the QCSP is true [Papadimitriou 1994; Stockmeyer and Meyer 1973].

6.1 Encoding Issues

To analyze the complexity, a few words are needed on the encoding of the QCSP $\langle X, Q, D, C\rangle$. Def. 1 did not specify anything on this issue, because the encoding did not have any consequence on the results of previous sections. We assume that X and Q are encoded in the natural way, i.e., as a list. For the set of domains D, two choices may be considered: a domain can be encoded as a list of allowed values or as an interval, in which case its two bounds need to be encoded. Our results will hold independently of whether the interval or domain representation is chosen. The main question is how the constraints are defined. Some examples of representation formalisms are the following:

- I The domain is Boolean, i.e., $B = \{0, 1\}$, and C is defined as a Boolean circuit.
- II The domain is Boolean, i.e., $B = \{0, 1\}$, and C is put in Conjunctive Normal Form, i.e., it is a conjunction of clauses (disjunctions of literals, each of which is a variable or its negation).
- III C is a conjunction of constraints, each of which is represented in extension as a table (e.g., binary) which lists all tuples that are accepted.
- IV C is a conjunction of constraints, each of which is represented by a numerical (linear or polynomial) equality or inequality.
- V C is a polynomial-time *program* (written in any universal language, for instance the Turing machine) which, given a tuple t, determines whether $t \in \mathsf{sol}$.

In all cases we impose the restriction that testing whether $t \in \mathsf{sol}$ be feasible in polynomial time. The fifth encoding represents the most general possible encoding satisfying this restriction: we shall consider it when we want to check that a result holds for any encoding in which testing whether $t \in \mathsf{sol}$ can be done in polynomial time.

Using encoding (V) to capture the notion of "most general encoding" is therefore convenient, but an important point is that the 4 other formalisms are essentially as concise as formalism (V). If the domain is Boolean, then if sol can be represented by a program P (in the sense that $P(t) = 1$ iff $t \in \mathsf{sol}$) and if the execution of P requires a memory bounded by S and a time bounded by L, then the set sol can be also represented by a Boolean circuit of size polynomial in S, L, and the length of the text of the program P, using the technique used by Cook in proving that SAT is NP-complete. In other words, for Boolean domains, formalism (I) is as expressive as formalism (V). Now the relations between formalism (I) and formalisms (II) to (IV) are well-known: we can reduce a circuit to a CNF involving only clauses of size at most three (3CNF) by introducing existential variables, and it is straightforward to reduce a 3CNF to formalism (III) or formalism (IV). The complexities of our problems for (I) to (V) will therefore be equivalent except for minor refinements occurring at intermediate levels of the polynomial hierarchy (Prop 16), where introducing existential variables makes a little difference.

6.2 A Common Upper Bound: PSPACE

The most difficult side of our complexity characterizations is to prove *membership* in PSPACE. It is indeed not completely obvious at first that the properties we have studied can be verified in polynomial space. The key point is to notice that a polynomial space algorithm exists to recognize the set of outcomes. To explain why this result holds, we show that deciding whether a tuple is an outcome of a QCSP ϕ can be done by solving another QCSP obtained by making simple changes to ϕ. Considering representation (V), we have the following:

PROPOSITION 13. *Let $\phi = \langle X, Q, D, C\rangle$ be a QCSP. Given a tuple $t \in \prod_{x\in X} D_x$, we denote by B the conjunction of constraints:*

$$\bigwedge_{x_i \in E} \left(\left(\bigwedge_{y \in A_{i-1}} y = t_y \right) \rightarrow (x_i = t_{x_i}) \right) \tag{3}$$

The QCSP $\psi = \langle X, Q, D, B \cup C\rangle$ is true iff $t \in \mathsf{out}^{\phi}$.

Note that $B \cup C$ can be expressed concisely in formalism (V). The conjunction of constraints added in (3) makes sure that any winning strategy of ψ contains t as a scenario.

A direct corollary of Prop. 13 is that checking whether a particular tuple t belongs to the set of outcomes of a QCSP ϕ can be done in polynomial space, simply by solving ψ. This is true for any representation of the constraints that respects the restriction that testing whether $t \in \mathsf{sol}$ be feasible in polynomial time[2]. Now being able to test in polynomial space whether a tuple is an outcome, the membership in PSPACE of all properties becomes clear: for instance if we consider inconsistency ($\forall t \in \mathsf{out}.\ t_{x_i} \neq a$) we can enumerate all tuples in lexicographical order, determine whether each of them is an outcome, and whether it satisfies the implication $t \in \mathsf{out}.\ t_{x_i} \neq a$. The precise list of results will be given in the next section, where we state completeness results (including both hardness and membership for the considered class).

EXAMPLE 4. *Let us illustrate the idea of Prop. 13 on a simple example. Consider the QCSP $\exists x_1.\ \forall y_1.\ \exists x_2.\ \forall y_2.\ \exists x_3.\ C$, where the domain of each variable is, for instance, $\{0,1\}$. We want to determine whether the tuple $\langle x_1 = 0, y_1 = 0, x_2 = 0, y_2 = 0, x_3 = 0\rangle$ is an outcome of the QCSP. This can be done by solving the QCSP in which the constraints of (3) are added:*

$$\exists x_1.\forall y_1.\exists x_2.\forall y_2.\exists x_3.\ C \wedge (x_1 = 0 \wedge (y_1 = 0 \rightarrow x_2 = 0) \wedge ((y_1 = 0 \wedge y_2 = 0) \rightarrow x_3 = 0)).$$

It might be useful to mention a possible source of confusion: it is the case that our PSPACE membership results hold for formalism (4), since it respects our restriction. This is true even if the domains D_x are represented by intervals: even

[2]In fact this condition could itself be considerably relaxed: the PSPACE membership result holds under the very general condition that testing whether $t \in \mathsf{sol}$ be feasible in polynomial *space*.

though an interval whose bounds are n-bit integers represents in general a set of values of cardinality exponential in n, we can always iterate on these values using polynomial space. This should be contrasted with classical complexity results related to arithmetics: in general deciding the truth of quantified linear constraints is extremely complex (hard for $\mathsf{NDTIME}(2^{2^n})$ by the Fischer-Rabin theorem [Fischer and Rabin 1974], and therefore provably not in PSPACE $\subseteq$ EXPTIME), and if we consider quantified polynomial constraints the problem becomes undecidable (Gödel's theorem). The key point is that in these cases the values of the variables can grow extremely large; as long as we bound the domains explicitly this problem does not arise, which is why we remain within PSPACE.

6.3 Complexity Characterizations

We now list the complexity results we obtain. These results hold for any of the 5 representations we have mentioned.

PROPOSITION 14. *Given a QCSP $\phi = \langle X, Q, D, C\rangle$, the problems of deciding whether:*

—*value $a \in D_{x_i}$ is d-fixable, d-removable, inconsistent, implied for variable $x_i \in X$,*

—*value $a \in D_{x_i}$ is d-substitutable to or d-interchangeable with $b \in D_{x_i}$ for variable $x_i \in X$,*

—*variable $x_i \in X$ is dependent on variables $V \subseteq X$, or is d-irrelevant*

are PSPACE-complete.

An analogous result holds for the shallow properties:

PROPOSITION 15. *Given a QCSP $\phi = \langle X, Q, D, C\rangle$, the problems of deciding whether:*

—*value $a \in D_{x_i}$ is s-fixable, s-removable for variable $x_i \in X$,*

—*value $a \in D_{x_i}$ is s-substitutable to or s-interchangeable with $b \in D_{x_i}$ for variable $x_i \in X$,*

—*variable $x_i \in X$ is s-irrelevant*

are PSPACE-complete.

As usual when considering quantified constraints, the complexity increases with the number of quantifier alternations, more precisely each additional alternation brings us one level higher in the Polynomial Hierarchy [Stockmeyer 1976]. The precise level that is reached is dependent on the considered property and on many details, including the formalism used for the encoding of the QCSP. We shall not list all results but instead we characterize, as an example, the complexity obtained in a particular setting, i.e., for the "deep" definitions of the properties, in the case where the QCSP starts with an existential quantifiers, and where its constraints are encoded as a Boolean circuit.

We call Σ_kQCSPs the QCSPs with at most k quantifier alternations and whose first variables are existential. We have the following results:

PROPOSITION 16. *Given a Σ_k QCSP $\phi = \langle X, Q, D, C\rangle$ encoded using Formalism (I), the problems of deciding whether:*

—*value $a \in D_{x_i}$ is deep-fixable, deep-removable, inconsistent, implied for variable $x_i \in X$,*

—*value $a \in D_{x_i}$ is deep-substitutable to or deep-interchangeable with $b \in D_{x_i}$ for variable $x_i \in X$,*

—*variable $x_i \in X$ is dependent on variables $V \subseteq X$, or is deep-irrelevant,*

are Π^p_k-hard and belong to Π^p_{k+1}. Moreover, for deep inconsistency, implication, determinacy and dependence, the problems are more precisely Π^p_k-complete.

In particular, it was reported in [Bordeaux et al. 2004] that these problems are coNP-complete for purely existential QCSPs.

Why the precise results are less regular than in previous cases is because the precise number of quantifier alternations is impacted by many factors. For instance, if we consider a Quantified Boolean Formula $\exists X.\ \forall Y.\ F(X,Y)$, where X and Y are vectors of Boolean variables and F is a Boolean circuit, then putting F into CNF will produce a formula of the form $\exists X.\ \forall Y.\ \exists Z.\ G(X,Y,Z)$, and this sometimes incurs a difference of one level in the polynomial hierarchy between Formalism (I) and Formalisms (II) to (IV). Similarly, there is a difference between shallow and deep properties in that shallow properties are themselves usually stated with more quantifier alternations, a typical form being "forall outcomes, there exists an outcome". What is obviously true for all properties in any case, however, is if we consider QCSPs with a limited number of quantifier alternations, the level reached in the polynomial hierarchy is also bounded.

7. LOCAL REASONING

The previous section shows that all of the properties we are interested in are computationally difficult to detect—in fact as difficult as the resolution of the QCSP problem itself. There are nonetheless particular cases where a property can be cheaply revealed. In CSP solvers the most widely used way of detecting properties cheaply is by using *local* reasoning: instead of analysing the whole problem at once, thereby facing its full complexity, we analyse it bit by bit (typically constraint by constraint). Depending on the property we know how deductions made on the bits generalize to the whole CSP. For instance:

—In the case of inconsistency, a deduction made on one single constraint generalizes to the whole CSP. For instance, if we have a CSP $\exists x \in [0,5].\ y \in [0,5].\ x > y \wedge C$, we can deduce from the constraint $x > y$ that value 0 is inconsistent for x, without having to worry of which other constraints are present in C.

—In the case of substitutability, a deduction is valid for the whole CSP if it can be checked independently for each and every constraint. For instance if we have the CSP $\exists x \in [0,5].\ y \in [0,5].\ x > 1 \wedge x \leq y$, we can deduce that value value 3 is substitutable to 2 for x. This is the case because the substitutability property holds for both constraints $x > 1$ and $x \leq y$. If, however, there were a third constraint, we would have to make sure that the property holds for it as well before deducing that it holds for the whole CSP. The situation is slightly less advantageous than for inconsistency because we have to consider each constraint before making a deduction, but it is nevertheless of interest—analysing the constraints one by one is typically much cheaper than analysing the whole CSP at once.

Following the classical CSP approach, we investigate the use of local reasoning as a means to cheaply detect the properties we have proposed.

7.1 Positive Results

Our first result is that using local reasoning allows to detect the deep properties except removability. Depending on the property one of the two forms of generalization mentioned before is correct.

PROPOSITION 17. *Let $\phi = \langle X, Q, D, C \rangle$ be a QCSP where $C = \{c_1, \ldots, c_m\}$. We denote by ϕ_k the QCSP $\langle X, Q, D, \{c_k\} \rangle$ in which only the k-th constraint is considered. We have, for all $x_i \in X$, $V \subseteq X$, and $a, b \in D_{x_i}$:*

—$\left(\bigvee_{k \in 1..m} \textit{inconsistent}^{\phi_k}(x_i, a) \right) \rightarrow \textit{inconsistent}^{\phi}(x_i, a)$;

—$\left(\bigvee_{k \in 1..m} \textit{implied}^{\phi_k}(x_i, a) \right) \rightarrow \textit{implied}^{\phi}(x_i, a)$;

—$\left(\bigwedge_{k \in 1..m} \textit{d-fixable}^{\phi_k}(x_i, a) \right) \rightarrow \textit{d-fixable}^{\phi}(x_i, a)$;

—$\left(\bigwedge_{k \in 1..m} \textit{d-substitutable}^{\phi_k}(x_i, a, b) \right) \rightarrow \textit{d-substitutable}^{\phi}(x_i, a, b)$;

—$\left(\bigwedge_{k \in 1..m} \textit{d-interchangeable}^{\phi_k}(x_i, a, b) \right) \rightarrow \textit{d-interchangeable}^{\phi}(x_i, a, b)$;

—$\left(\bigvee_{k \in 1..m} \textit{determined}^{\phi_k}(x_i) \right) \rightarrow \textit{determined}^{\phi}(x_i)$;

—$\left(\bigwedge_{k \in 1..m} \textit{d-irrelevant}^{\phi_k}(x_i) \right) \rightarrow \textit{d-irrelevant}^{\phi}(x_i)$;

—$\left(\bigvee_{k \in 1..m} \textit{dependent}^{\phi_k}(V, x_i) \right) \rightarrow \textit{dependent}^{\phi}(V, x_i)$.

7.2 Negative Results

It was noticed in [Bordeaux et al. 2004] that, even in the non-quantified case, deep removability is not as well-behaved as the other deep properties since it is not possible to detect it using local reasoning. This was seen on an example, which we borrow from that paper:

EXAMPLE 5. *Consider the CSP*

$$\exists x \in \{1, 2, 3\}.\ \exists y \in \{1, 2, 3\}.\ (x \le y, y \le x, x \ne 1, x \ne 3)$$

If we consider each of the four constraints, then we find that value 2 is removable for x. But obviously value 2 is not *removable for the CSP as the only solution is indeed $x = 2, y = 2$.*

A similar problem occurs when we consider the shallow definitions: it is incorrect, in general, to use local reasoning to detect these versions of the properties[3]. Here again this can be seen on a simple example:

EXAMPLE 6. *Consider the (Q)CSP*

$$\exists x_1 \in \{0, 1\}.\ \exists x_2 \in \{0, 1\}.\ (x_1 = x_2 \land x_2 = 1)$$

[3]This corrects an error in [Bordeaux et al. 2005], where we wrongly stated that local reasoning is valid for all properties.

It is the case that variable x_1 is shallow-fixable to value 0 w.r.t. constraint $x_1 = x_2$; and variable x_1 is also shallow-fixable to value 0 w.r.t. constraint $x_2 = 1$. Despite of that, x_1 is not shallow-fixable to 0 in the QCSP, as there is simply no solution with $x_1 = 0$.

The shallow definitions therefore have to be considered carefully: they are more general than the deep properties, but they have to be detected by other means than local reasoning. What this means is that we cannot, in general, try to detect that the property holds by having a look at each constraint one by one, or by doing other forms of reasoning that would not consider the problem globally. This is somewhat reminiscent of what happens with the removability property, whose generality comes at the price of being a less well-behaved property than substitutability or inconsistency.

8. CONCLUDING REMARKS

8.1 Related Works

A number of works related to Quantified CSP have considered particular cases of the properties we have attempted to study systematically in this paper. Most of these works have been mentioned throughout the paper, notably [Mamoulis and Stergiou 2004] for their use of substitutability; we also note the work done by Peter Nightingale in his thesis, which devotes large parts to the consistency property [Nightingale 2005]. The notions considered in these works are related to our proposals but typically less general, because our definitions finely take into account the quantifiers. For substitutability for instance, the definition used in [Mamoulis and Stergiou 2004] was essentially the classical (existential) definition. For consistency, our definition is equivalent to the notions proposed by [Bordeaux and Monfroy 2002] or [Nightingale 2005]. Our general definition nevertheless leaves open the question of how to efficiently detect inconsistent values, and these papers propose practical ways of using local reasoning to detect inconsistent values. This situation is quite closely related to works in CSP, where many notions of local consistency can be defined. These notions have different merits that can be evaluated experimentally, but they all share the basic property of being ways to detect (globally) inconsistent values, which explains why they are correct.

We also note that more advanced studies are available for the particular case of Boolean quantified constraints. In these works some techniques have been proposed that specifically take into account the quantifier prefix. However, contrary to ours, these proposals are restricted to Boolean domains. For instance in [Rintanen 1999; Cadoli et al. 2002], several techniques are proposed to fix and remove values. These works have shown that detecting properties is essential and can lead to a consistent pruning of the search space, but no clear and general framework to understand these properties was available.

An interesting, recent related work is [Audemard et al. 2007], which initiates the study of *symmetries* in Quantified Boolean Formulae. Symmetries are related to the notion of interchangeability but are in a sense a more general concept. Our feeling is that the idea of using the notion of outcome to define constraint properties may be applicable to this class of properties as well. Symmetries are a complex and fascinating topic; an interesting perspective for future work will be to see if our framework can help understanding them in the general context of quantified CSP.

8.2 Conclusion

A primary goal of our work was to state the definitions in a way that is formal and amenable to proofs. In previous QCSP literature, it is fair to say that formal proofs were scarce, probably because facts that are trivial to prove in CSP tend to become complex to write formally when quantifiers come into play. Quantifiers can be complex to reason with, and it is sometimes easy to make wrong assumptions on some properties, as we saw ourselves when finding the error we made in the preliminary version of this paper (Section 6). Because of this difficulty, we wanted in this work to build solid foundations on which the deductions made in QCSP solvers can rely.

ACKNOWLEDGMENTS

Detailed comments from our anonymous reviewers helped significantly improve the paper and we are grateful to them. Section 5.2.2 and Proposition 12 were, in particular, added based entirely on their comments and suggestions. Useful discussions with Horst Samulowitz are also acknowledged.

REFERENCES

Ansótegui, C., Gomes, C. P., and Selman, B. 2005. The Achilles' heel of QBF. In *Proc. of Amer. Conf. on Artificial Intelligence (AAAI)*. AAAI Press, 275–281.

Audemard, G., Jabbour, S., and Saïs, L. 2007. Symmetry breaking in Quantified Boolean Formulae. In *Proc. of Int. Joint. Conf. on Artificial Intelligence (IJCAI)*. Morgan Kaufmann, 2262–2267.

Bacchus, F. and Stergiou, K. 2007. Solution directed backjumping for QCSP. In *Proc. of Int. Conf. on Principles and Practice of Constraint Programming (CP)*. Springer, 148–163.

Benedetti, M. 2004. Evaluating QBF via symbolic skolemization. In *Proc. of Int. Conf. on Logic for Programming, Artificial Intelligence and Reasoning (LPAR)*. Springer, 285–300.

Benedetti, M., Lallouet, A., and Vautard, J. 2007. QCSP made practical by virtue of restricted quantification. In *Proc. of Int. Joint. Conf. on Artificial Intelligence (IJCAI)*. Morgan Kaufmann, 38–43.

Bordeaux, L., Cadoli, M., and Mancini, T. 2004. Exploiting fixable, removable and determined values in constraint satisfaction problems. In *Proc. of Int. Conf. on Logic for Programming, Artificial Intelligence and Reasoning (LPAR)*. Springer, 270–284.

Bordeaux, L., Cadoli, M., and Mancini, T. 2005. CSP properties for quantified constraints: Definitions and complexity. In *Proc. of Amer. Conf. on Artificial Intelligence (AAAI)*. AAAI Press, 360–365.

Bordeaux, L. and Monfroy, E. 2002. Beyond NP: Arc-consistency for quantified constraints. In *Proc. of Int. Conf. on Principles and Practice of Constraint Programming (CP)*. Springer, 371–386.

Bordeaux, L. and Zhang, L. 2007. A solver for quantified Boolean and linear constraints. In *Proc. of Int. Symp. on Applied Computing (SAC)*. ACM, To appear.

Börner, F., Bulatov, A., Jeavons, P., and Krokhin, A. 2003. Quantified constraints: Algorithms and complexity. In *Proc. of Int. Conf. on Computer Science Logic (CSL)*. Springer, 58–70.

Buening, H. K., Karpinski, M., and Flogel, A. 1995. Resolution for Quantified Boolean Formulas. *Information and Computation 117,* 1, 12–18.

Cadoli, M., Giovanardi, A., and Schaerf, M. 1999. An algorithm to evaluate quantified boolean formulae. In *Proc. of Amer. Conf. on Artificial Intelligence (AAAI)*. AAAI/MIT Press, 262–267.

Cadoli, M., Schaerf, M., Giovanardi, A., and Giovanardi, M. 2002. An algorithm to evaluate Quantified Boolean Formulae and its experimental evaluation. *J. of Automated Reasoning 28,* 2, 101–142.

Chen, H. 2004a. Collapsibility and consistency in quantified constraint satisfaction. In *Proc. of Amer. Conf. on Artificial Intelligence (AAAI).* AAAI Press, 155–160.

Chen, H. 2004b. Quantified constraint satisfaction and bounded treewidth. In *Proc. of Euro. Conf. on Artificial Intelligence (ECAI).* IOS Press, 161–165.

Ferguson, A. and O'Sullivan, B. 2007. Quantified Constraint Satisfaction Problems: from relaxations to explanations. In *Proc. of Int. Joint. Conf. on Artificial Intelligence (IJCAI).* Morgan Kaufmann, 74–79.

Fischer, M. J. and Rabin, M. O. 1974. Super-exponential complexity of Presburger Arithmetics. In *Complexity of Computation*, R. Karp, Ed. 27–41.

Freuder, E. C. 1991. Eliminating interchangeable values in constraint satisfaction problems. In *Proc. of Amer. Conf. on Artificial Intelligence (AAAI).* AAAI Press, 227–233.

Gent, I., Nightingale, P., and Stergiou, K. 2005. QCSP-Solve: A solver for quantified constraint satisfaction problems. In *Proc. of Int. Joint. Conf. on Artificial Intelligence (IJCAI).* Morgan Kaufmann, 138–143.

Gent, I. P., Nightingale, P., and Rowley, A. 2004. Encoding quantified CSPs as Quantified Boolean Formulae. In *Proc. of Euro. Conf. on Artificial Intelligence (ECAI).* IOS Press, 176–180.

Mackworth, A. 1977. Consistency in networks of relations. *Artificial Intelligence 8*, 99–118.

Mamoulis, N. and Stergiou, K. 2004. Algorithms for Quantified Constraint Satisfaction Problems. Tech. Rep. APES-79-2004, Apes research group.

Monasson, R., Zecchina, R., Kirkpatrick, S., Selman, B., and Troyansky, L. 1999. Determining computational complexity from characteristic 'phase transitions'. *Nature 400*, 133–137.

Nightingale, P. 2005. Consistency for quantified constraint satisfaction problems. In *Proc. of Int. Conf. on Principles and Practice of Constraint Programming (CP).* Springer, 792–796.

Nightingale, P. 2007. Consistency and the quantified constraint satisfaction problem. Ph.D. thesis, University of St Andrews.

Papadimitriou, C. H. 1994. *Computational Complexity.* Addison Wesley.

Rintanen, J. 1999. Improvements to the Evaluation of Quantified Boolean formulae. In *Proc. of Int. Joint. Conf. on Artificial Intelligence (IJCAI).* Morgan Kaufmann, 1192–1197.

Samulowitz, H. and Bacchus, F. 2006. Binary clause reasoning in QBF. In *Proc. of Int. Conf. on Theory and Applications of Satisfiability Testing (SAT).* Springer, 353–367.

Samulowitz, H., Davies, J., and Bacchus, F. 2006. Preprocessing QBF. In *Proc. of Int. Conf. on Principles and Practice of Constraint Programming (CP).* Springer, 514–529.

Stockmeyer, L. J. 1976. The polynomial-time hierarchy. *Theoretical Computer Science (TCS) 3,* 1, 1–22.

Stockmeyer, L. J. and Meyer, A. R. 1973. Word problems requiring exponential time: Preliminary report. In *Proc. of Symp. on Theory of Computing (STOC).* ACM, 1–9.

Verger, G. and Bessière, C. 2006. A bottom-up approach for solving quantified CSPs. In *Proc. of Int. Conf. on Principles and Practice of Constraint Programming (CP).* Springer, 635–649.

Zhang, L. 2006. Solving QBF by combining conjunctive and disjunctive normal forms. In *Proc. of Amer. Conf. on Artificial Intelligence (AAAI).*

Received May 2007; revised December 2007; accepted April 2008