4 rue Léonard de Vinci
BP 6759
F-45067 Orléans Cedex 2
FRANCE

Rapport de Recherche

http://www.univ-orleans.fr/lifo

Problem Modeling and Solving in QCSP made Practical

Marco Benedetti, Arnaud Lallouet, Jérémie Vautard
Université d'Orléans, LIFO

Rapport N^o **2006-11**
18/07/2006

Problem Modeling and Solving in QCSP made Practical

Marco Benedetti, Arnaud Lallouet, Jérémie Vautard

Université d'Orléans — LIFO
BP 6759 — F-45067 Orléans cedex 2
{Marco.Benedetti | Arnaud.Lallouet | Jeremie.Vautard}@univ-orleans.fr

Abstract. Quantified Constraint Satisfaction Problems (QCSPs) extend classical CSPs by allowing universally quantified variables. This is done by adding an alternating prefix of quantified variables to the set of constraints of a CSP. But this formulation is far from being intuitive from the modeling point of view because many problems make use of preconditions or guards that safely restrict the possible values of the variables. In this paper, we introduce a new language called QCSP+ which allows to express restricted quantification like in "*forall X, Y such that X > Y, ...*" and we show that it can model many quantified problems more naturally than classical QCSPs. Formally, it amounts to handle a sequence of CSPs connected by an alternation of conjunction and disjunction. But in contrast with CSPs, disjunction reflects the intrinsic structure of quantified problems and allows to extract useful properties to prune the search space. Our solver has competitive performance over the state-of-the-art. In addition, it allows to reuse arithmetic and global constraints with their native propagator.

1 Introduction

We extend the QCSP (Quantified Constraint Satisfaction Problem) framework by introducing a new language, called QCSP$^+$. Such extension is motivated by the large difficulties we experience in modelling and solving most non-trivial problems as a plain QCSP. So, let us start by introducing (Q)CSP and pointing out its weaknesses.

A CSP is a search problem established by giving a set of variables ranging over finite domains, and a conjunction of constraints mentioning such variables. For example, given $x \in \{1,2,3\}$, $y \in \{3,4\}$, and $z \in \{4,5,6\}$, the CSP

$$x < y \wedge x + y = z \wedge z \neq 3x$$

is solved by selecting (if possible) one value for each variable in so as to satisfy the three constraints at once. For example, $x{=}1$, $y{=}4$, $z{=}5$ is a valid solution, while $x{=}2$, $y{=}4$, $z{=}6$ is not. CSP formulations are naturally suited to model real-world problems.

For the sake of this paper, a CSP problem is best viewed as a *decision* problem in which all the variables are quantified *existentially* (i.e. the existence of one single consistent assignment suffices to answer the problem positively):

$$\exists x{\in}\{1,2,3\}\exists y{\in}\{3,4\}\exists z{\in}\{4,5,6\}.\ x < y \wedge x + y = z \wedge z \neq 3x$$

The seemingly inconsequential amendment of making quantifiers explicit leads us to play in an entirely new field [7]: What if (some of) the variables are quantified *universally*? For example, what means for

$$\exists x{\in}\{1,2,3\}\forall y{\in}\{3,4\}\exists z{\in}\{4,5,6\}.\ x < y \wedge x + y = z \wedge z \neq 3x \tag{1}$$

to be true? The most intuitive way of entering the new scenario is by thinking of it as a *game*. We have two players now: one is associated to the existential quantifier—we call him the ∃-player—the other is related to the universal quantifier: the ∀-player. The goal of the ∃-player is to satisfy each constraint, hence to satisfy the whole CSP. The goal of the ∀-player is to violate at least one constraint, thus overcoming the opponent's effort. The two players play against each other in turn, for a finite and fixed number of rounds. The moves they do consist in assigning values to variables. Which variables get assigned at each step is statically decided by the left-to-right order in the *prefix* of the problem (in our example, the prefix is $\exists x \in \{1,2,3\} \forall y \in \{3,4\} \exists z \in \{4,5,6\}$).

In (1), the ∃-player plays first, and he is given the chance to choose a value for x. Then, it is time for the ∀-player to assign a value to the variable y. Finally, the ∃-player assigns z and the game terminates: the satisfaction of the set of constraints is evaluated. We say that such *quantified* CSP (or QCSP) is true if the ∃-player has a winning strategy—i.e. he can manage to satisfy every constraint whatever the universal opponent does—and is false otherwise. For example, (1) is true, while it becomes false if we change the prefix to $\forall x \exists y \forall z$. The order of quantifiers is important: by flipping y and z in (1) we play under the prefix $\exists x \exists z \forall y$, and the existential player looses the game.

QCSP is widely believed to be strictly more "powerful" than CSP: An entire hierarchy of QCSP problems exists for which no equivalent CSP formulation can be written "compactly" (QCSP is PSPACE-complete, CSP is NP-complete). Some of these problems are openly perceived as games by humans (i.e. board games), while in others the fundamental game structure is camouflaged (see Section 2).

Despite the expectation engendered by the strength of the language, we find in the literature *no single account* of a real-world model[1] actually solved by a QCSP solver. Why?

A partial explanation is that QCSP solvers are in their infancy and miss most *quantified constraint propagators.*This limitation discourages or prevents people from investing in the production of realistic QCSP models. However, we have recently built a full-fledged QCSP solver [5], which manages several constraints. Even so, models of many "natural" cases stay surprisingly difficult to devise.

To pinpoint the problem, let us take a step back to QCSP as a game between ∃ and ∀, and let us observe that with games almost invariably come *rules*: Some choices could be precluded *as a function of previous choices by the same player or by the opponent.* An elementary example is the prohibition in most board games to play in a cell already occupied by someone. In the QCSP game, this means that players cannot in general assign variables in arbitrary ways. The set of *legal choices* is dynamically restricted over a game lifespan in relation to previous agents' moves. The self-limitation to legal moves comes out to be precisely *the* problem with QCSP.

No support for dynamic ranges of variables is provided by the prefix: In (1) z ranges over $\{4,5,6\}$ whichever the values chosen for x and y. So, we have to embed the *game discipline* in the constraints. It is a matter of stating that if a player chooses a forbidden value (or combination of values) he looses immediately. Such threat is promptly posed to the ∃-player: We consider the membership of the move to the set of legal moves as just an additional constraint. If the ∃-player cheats, he makes such constraint false, he looses the game.

No similar expedient can be used against the ∀-player: The game is a loss for him when all the constraints are satisfied, a thing which simply cannot be imposed by just *conjuncting* whatever additional constraint. Rather, to solve such ∀*-discipline*

[1] By real-world model we mean any model designed to capture the pre-existing and human-intelligible semantics of some problem—e.g. the ones in Section 2—as opposed to randomly generated models that obey *purely syntactical* generation rules.

problem we should slightly modify the whole formalization. For reasons that will be discussed in Section 5, this comes out to be an appalling remedy to the problem.

So, let us introduce a different solution, based on handling the game rules explicitly. As we shall see shortly, QCSP is not enough to plainly state the new formalization as it lacks support for *disjunctions* between constraint sets. Consider the following problem (in which rules are provisionally absent)

$$\forall X_1 \exists Y_1 \forall X_2 \exists Y_2 . C(X_1, X_2, Y_1, Y_2) \tag{2}$$

Let uppercase letters denote *sets* of variables rather than a single variable (domains are not shown). Suppose the legal opening moves for ∀-player are characterized by a set of constraints (a CSP) $L_1^\forall(X_1)$, i.e. an assignment to the variables in X_1 is a legal opening move iff it is a solution to $L_1^\forall$ in the classical CSP sense. Next, ∃-player's reply at second step is constrained by some CSP $L_1^\exists(X_1, Y_1)$: once the choices over X_1 from ∀-player's side are known, an assigment over Y_1 is to be considered only if it solves $L_1^\exists$. Likewise, rules $L_2^\forall(X_1, Y_1, X_2)$ and $L_2^\exists(X_1, Y_1, X_2, Y_2)$ are provided. A compact syntax we use to state the whole thing is:

$$\begin{aligned} \forall X_1 [L_1^\forall(X_1)].\ \exists Y_1 [L_1^\exists(X_1, Y_1)].\ \forall X_2 [L_2^\forall(X_1, Y_1, X_2)].\\ \exists Y_2 [L_2^\exists(X_1, Y_1, X_2, Y_2)].\ C(X_1, X_2, Y_1, Y_2) \end{aligned}$$

which reads "*for all the assignments to* X_1 *such that* $L_1^\forall$ *holds, exists an assignment to* Y_2 *such that* $L_1^\exists$ *holds and for all...*". As discussed above, the "such that" particle stays for a conjunction when ∃-player is involved, and for an implication when ∀-player is concerned. So, we actually ask whether

$$\forall X_1 (L_1^\forall \rightarrow \exists Y_1 (L_1^\exists \wedge \forall X_2 (L_2^\forall \rightarrow \exists Y_2 (L_2^\exists \wedge C)))) \tag{3}$$

or, in an equivalent prenex form with explicit disjunction, if

$$\forall X_1 \exists Y_1 \forall X_2 \exists Y_2 . (\overline{L_1^\forall} \vee (L_1^\exists \wedge (\overline{L_2^\forall} \vee (L_2^\exists \wedge C)))) \tag{4}$$

Statement (4) shows that the most natural way to state a "quantified game with rules" is to extend the language to handle disjunctions. But, a shift to general non-conjunctive CSPs would hamper the critical opportunity to inherit the huge amount of propagators already implemented for the purely conjunctive case, and the propagation schemes designed for the very same case. We would fall back into the inability to exercise QCSP tools on real problems, by having a nice formalism with a proof-of-concept or no implementation at all.

Our solution is to design a *limited* disjunctive extension of the QCSP formalism, which is (i) enough to capture all the game-with-rule scenarios, but is (ii) still capable of inheriting and exercising the reasoning core of standard (Q)CSP solvers. So, our approach makes a crucial distinction between (i) quantifier-induced disjunctions, which are *not* in the original problem specification but come into existence just to solve the ∀-discipline problem, and (ii) general disjunctions used in modeling problems. The rationale behind this partition is that —as the literature on CSP witnesses with the countless number of models devised— the concepts involved in modelizing CSPs can usually be expressed as conjunctions of constraints, where complex disjunctions are just unnecessary. However, when the *question we ask on the model* requires the introduction of universal quantifiers (see for example Section 2), the ability to express a limited form of disjunction—the quantifier-induced disjunction—becomes unescapable (even in problems where each basic concept needs no disjunction to be expressed). Hence, we provide an ad-hoc mechanism to capture quantifier-induced disjunctions, while no wider support than in standard CSP solvers is given to the general case.

The syntactic shape of the problems we introduce—the QCSP$^+$ problems, generalizing Example (3-4)—is a *chain of nested alternated quantifications with preconditions* (see Figure 1). This language is cognitively adequate to represent rules

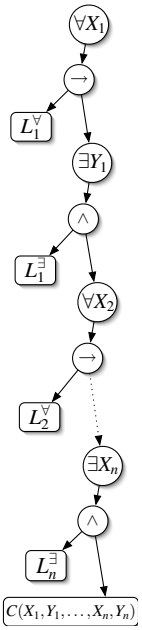

Fig. 1. Shape of the problem we introduce

(*modeler's viewpoint*, Section 2), and at the same time is amenable to be decided by reusing existing technology (*solver's viewpoint*, Section 4). Indeed, each "precondition" part contains a standard CSP, inside which the usual (quantified) propagation procedures can be capitalized. What is to be added is (i) an external search-guiding mechanism that evaluates the truth value of the alternated chain of CSPs as a function of the truth value of the leaves, and (ii) an inter-leaf propagation scheme that reconciles the separate CSPs and makes them share information in a sound way. A thorough discussion on how this approach relates to recent contributions in the close field of QBF decision procedures is presented in Section 5. The implementation of a full-fledged solver and experiments on quantified models are presented in Section 6. In Section 7 we summarize our contributions and present directions for future work.

2 Motivating Examples

The concepts involved in the modelization of the following recurrent problems are naturally captured as a conjunction of constraints. However, the questions we ask require universal quantifiers and disjunctions in exactly the QCSP$^+$ style.

Problem 1. Suppose some (partial) ordering $\preceq$ is established to rank the solutions of a given CSP P. While a CSP is not enough to compactly characterize the *best* solutions of P w.r.t. $\preceq$, QCSP$^+$ gives to the problem an elegant one-move game solution: $\exists X[P(X)].\ \forall Y[P(Y)].\ Y \preceq X$.

In the following example [10], solutions to the CSP represent sets, and the $\subseteq$ relation creates preferences.

Example 2 (Strategic Companies). We are given a collection C of *companies*, a set G of *goods*, and a relation $Prod \subseteq C \times G$ to specify which goods each company produces. Companies have reciprocal financial participations, and a subset $C' \subseteq C$ that owns more than 50% of some $c \in C$ is said to be a *controlling* set for c.

A company may have many controlling sets. They all are captured by a relation $Contr \subseteq 2^C \times C$. A set of companies $S \subseteq C$ is "*production-preserving*"—written $\mathsf{PP}(S)$—if it (i) covers all the goods in G, i.e. by cumulating the goods g such that $\langle c, g\rangle \in Prod$ and $c \in S$ we obtain G; and (ii) is closed under the controlling relation, i.e., for each $\langle S, c\rangle \in Contr$, if $C' \subseteq S$ then $c \in C'$. A *strategic set* is any *subset-minimal* production-preserving set (i.e. the PP property is lost in each proper subset of a strategic set). A company $x \in C$ is *not strategic* (intuition: it can be sold with no impact on either the overall portfolio of goods or the controlled companies) if it belongs to no strategic set, i.e. if

$$\forall S[S \subseteq C \wedge \mathsf{PP}(S) \wedge x \in S].\ \exists S'[\mathsf{PP}(S') \wedge x \notin S'].\ S' \subset S$$

The *Prod* and *Contr* relations as defined above can be easily captured in propositional logic. We adopted such restricted version to enable a direct comparison with boolean reasoners (Section 6). More realistic models—mentioning explicit *amounts* of goods, *capacity* of production and *percentage* of participations—still fit nicely in the CSP vocabulary, but they lay outside the natural reach of propositional logic.

Problem 3 (Game strategy). Let a set of variables X_i describe the state at step i of a system evolving after the moves that two opponents $\forall$ and $\exists$ alternatively execute out of a finite set of possibilities. A *game* is defined by a 4-tuple $\langle \mathsf{PA}, \mathsf{SSA}, \mathsf{G}, \mathsf{I}\rangle$ of CSPs: A move M in a state X is only possible when the *precondition axiom* $\mathsf{PA}(X, M)$ is satisfied and leads to a new state X' defined by the *successor state axiom* $\mathsf{SSA}(X, X', M)$. Initial and winning conditions are recognized by I and G respectively. The first player wins the game in at most k rounds if $\exists X_0[I(X_0)].\mathsf{WS}(\exists, 1)$ is true, where

$$\begin{aligned}\mathsf{WS}(\mathcal{Q}, i) := \mathcal{Q}X_i, M_i[\ &\mathsf{PA}(X_{i-1}, M_i) \wedge \overline{G}(\overline{\mathcal{Q}}, X_{i-1}) \wedge \\ &\mathsf{SSA}(X_{i-1}, X_i, M_i)].\mathsf{WS}(\overline{\mathcal{Q}}, i{+}1)\end{aligned}$$

for $i < 2k$, and as $\mathsf{WS}(\mathcal{Q}, i) := \overline{G}(\overline{\mathcal{Q}}, X_{i-1})$ for $i = 2k$.

The next example shows QCSP$^+$ escaping the intricacies of game formalization we find in e.g. [12].

Example 4 (Connect $n{\times}m{-}k$). Let $B{=}(b_{ij}) \in \{0, \forall, \exists\}^{n\times m}$ be a matrix of variables representing the board state in a generalized Connect-4 game (played on a $n{\times}m$ board in the attempt to align k signs). The existence of a winning strategy for the first player is modeled by posing $\mathsf{PA}(B) := b_{xm} {\neq} 0$, where $x \in [1..m]$ is the current move, and using

$$\wedge_{i\in[1..n]j\in[1..n]}(b_{ij} {\neq} b'_{ij}) \leftrightarrow (j{=}m \wedge b_{ij}{=}0 \wedge b'_{ij}{=}p \wedge b_{i-1j} {\neq} 0)$$

as $\mathsf{SSA}(B, B', m)$, where $p \in \{\forall, \exists\}$ is the current player. The initial condition is $\mathsf{I}(B) \coloneqq \wedge_{ij} b_{ij} = 0$ and the $\overline{\mathsf{G}}$ condition is a simple conjunction of *allDifferent* constraints.

Problem 5 (Conformant Scheduling). Consider n tasks of duration $\delta_1 \ldots \delta_n$ requiring an amount of resources $r_1 \ldots r_n$, and subject to the ordering constraints $O \subseteq [1..n] \times [1..n]$, where $\langle i, j\rangle \in O$ means that task i must end before task j starts. We want to schedule the tasks so that (i) we finish by time T_{max}, (ii) we comply with O, and (iii) the overall instantaneous resource consumption never exceeds a fixed capacity r_{max}. A global constraint $\mathsf{cumulative}(t_1, \delta_1, r_1, \ldots, t_n, \delta_n, r_n, r_{max})$ is provided by most CSP solvers to check the latter condition. We have a degree of uncertainty about the actual resources required by each task. An hostile environment

can impact on them, subject to certain constraints $\mathsf{EM}(r_1, \ldots, r_n)$. Is it possible to devise an adversary-safe schedule?

$$\exists t_1, \ldots, t_n [\wedge_{\langle i,j \rangle \in O} t_i + \delta_i \leq t_j].\ \forall r_1, \ldots, r_n [\mathsf{EM}(r_1, \ldots, r_n)].$$
$$\mathsf{cumulative}(t_1, \delta_1, r_1, \ldots, t_n, \delta_n, r_n, r_{max}) \wedge max(t_i + \delta_i) \leq T_{max}$$

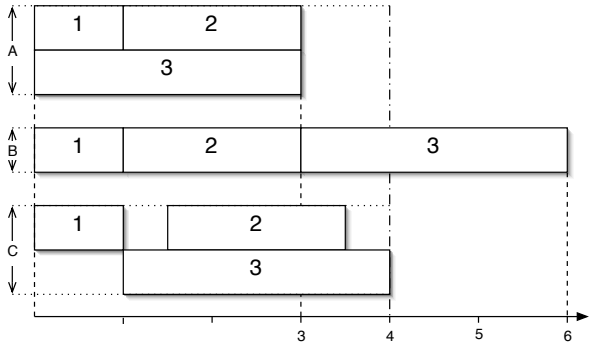

Fig. 2. Conformant scheduling

Example 6. Consider three tasks with $\delta_1 = 1$, $\delta_2 = 2$, $\delta_3 = 3$ and $r_1 = 3$, $r_2 = 2$, $r_3 = 1$ and such that task 1 is before task 2, to be completed within $T_{max} = 4$ and $r_{max} = 5$, under the assumption that the environment can add one unit of cost to up to two tasks. The picture aside shows that the optimal schedule (A) is subject to critical attacks, the linear schedule (B) exceeds T_{max}, while the solution (C) to the proper QCSP$^+$ instance meets all our criteria. Figure 2 gives a representation of the problem. The resource height of the tasks are not depicted.

3 Syntax, notation, evaluation

Let V be a set of variables and $D = (D_X)_{X \in V}$ be their domains. The set of tuples on V is denoted by D^V. A *constraint* $c = (W, T)$ is composed of a set of variables $W \subseteq V$ and a set of tuples $T \subseteq D^W$ (W and T are also noted $var(c)$ and $sol(c)$). A *CSP* C is a set of constraints. We denote by $var(C) = \bigcup_{c \in C} var(c)$ its set of variables. If $Y \notin W$, the extension of a tuple $t \in D^W$ to $t' \in D^{W \cup \{Y\}}$ with $t'|_W = t$ and $t'_Y = a$ is denoted by $t : [Y = a]$. We denote by $\top$ the special case of a CSP made up of no constraint (the empty CSP).

Definition 7 (Q-block). *A* Q-block *is a 4-tuple $b = \langle \mathcal{Q}, V, D, C \rangle$ where $\mathcal{Q} \in \{\exists, \forall\}$, V is a set of variables with domains $D = (D_X)_{X \in V}$, and C is a CSP with $V \subseteq var(C)$.*

We say that a quantifier block $b = \langle \mathcal{Q}, V, D, C \rangle$ *owns* the variables in V. The set V of variables owned by b may be empty. At the same time, C may mention some *free* variables $free(b) := var(C) \setminus V$ which are not owned by b.

Definition 8 (QCSP$^+$). *Given an integer $n > 0$, a QCSP$^+$ of depth n with free variables F is any sequence $[b_1, \ldots, b_n]$ of Q-blocks $b_i = (\mathcal{Q}_i, V_i, D_i, C_i)$ such that:*

- *$\mathcal{Q}_{j+1} \neq \mathcal{Q}_j$ for every $j \in [1..n-1]$;*
- *$F \cap V_j = \emptyset$ for every $j \in [1..n]$;*

– $i \neq j$ *implies* $V_i \cap V_j = \emptyset$*;*
– $var(C_i) \subseteq W_i \cup F$ *with* $W_i = \bigcup_{j \leq i} V_j$*.*

A closed *QCSP*$^+$ *of depth* n*, or simply a QCSP*$^+$ *henceforth, is a QCSP*$^+$ *of depth* n *with an empty set of free variables.*

The intuition is that a QCSP$^+$ is a sequence of Q-blocks with (strictly) alternating quantification type, such that the free variables appearing in the CSP associated to each Q-block are surely owned by some *dominating* Q-block. We say that a Q-block b is dominated by a Q-block b' in a certain QCSP$^+$ P if b' appears to the left of b in P.

As we will see shortly, each Q-block is meant to modelize a *restricted* or *qualified* (universal/existential) quantification in a step of some finite-length two-player game: The free variables represent the moves already occurred in the game, while the CSP embedded in the Q-block characterizes (in a very general sense) the set of moves that are legal in the game step associated to the Q-block itself.

We adopt a compact and intuitive denotation for a QCSP$^+$, which is consistent with the examples given in the introduction. In particular, we write

$$\mathcal{Q}_1 V_1 \in D_1\ [C_1].\ \mathcal{Q}_2 V_2 \in D_2\ [C_2].\ \cdots\ \ \mathcal{Q}_n V_n \in D_n\ [C_n].\ P \tag{5}$$

(where P is a CSP with $var(P) \subseteq \cup_{i=1..n} V_i$) to denote:

– the QCSP$^+$ $[b_1, \ldots, b_n]$ with $b_i = (\mathcal{Q}_i, V_i, D_i, C_i)$ for $i = 1, \ldots, n-1$ and $b_n = (\mathcal{Q}_n, V_n, D_n, C_n \cup P)$, if $\mathcal{Q}_n = \exists$;
– the QCSP$^+$ $[b_1, \ldots, b_n, b_{n+1}]$ with $b_i = (\mathcal{Q}_i, V_i, D_i, C_i)$ for $i = 1, \ldots, n$, and $b_{n+1} = (\exists, \emptyset, (), P)$ if $\mathcal{Q}_n = \forall$.

The slight asymmetry between the way we write a QCSP and the way we represent it aims to simplify the formal treatment of the new problems. Indeed, instead of introducing a special case each time we need to manage the innermost CSP problem P, we can always imagine that such innermost CSP collapses to the empty problem $\top$, and that its semantics is absorbed in the deepest Q-block (possibly introduced on purpose), so that our instances are pure sequences of Q-blocks. Dually, the QCSP$^+$ $[b_1, \ldots, b_n]$ is related to the special case in which P is empty, and is written as

$$\mathcal{Q}_1 V_1 \in D_1\ [C_1].\ \mathcal{Q}_2 V_2 \in D_2\ [C_2].\ \cdots\ \ \mathcal{Q}_n V_n \in D_n\ [C_n].\ \top$$

As we will see after a semantics for QCSP$^+$ is introduced, this *normalization* preserves the intuitive meaning we associate to an instance of the form (5).

Two simple properties of a QCSP$^+$ we will use in the rest of this section are:

– if $[b_1, \ldots, b_n]$ is a QCSP$^+$ of depth n, then $[b_1, \ldots, b_m]$ is a QCSP$^+$ of depth m for every $0 < m < n$.
– if $[b_1, \ldots, b_n]$ is a QCSP$^+$ of depth n and $b_i = (\mathcal{Q}_i, V_i, D_i, C_i)$, then for every $1 < m < n$, $[b_m, \ldots, b_n]$ is a QCSP$^+$ of depth $n - m + 1$ with free variables $\cup_{i=1..m-1} V_i$.

We have already encountered the special case of the empty CSP $\top$. The dual case is that of a CSP P in which at least one constraint is *empty*, i.e. the set T of tuples belonging to some constraint $c = (W, T) \in P$ is empty. In the latter case, we say that the CSP P is *inconsistent*, and we denote it by $\bot$. An empty or inconsistent CSP may give rise to special situations if it appears inside a Q-block, so we pose the following

Definition 9. *A Q-block* $b = \langle \mathcal{Q}, V, D, P \rangle$ *in which* $P = \top$ *is called* unconstrained *Q-block (universal or existential depending on* $\mathcal{Q}$*). If* $\mathcal{Q} = \exists$ *and* $P = \bot$ *the block is called* existential dead-end*. If* $\mathcal{Q} = \forall$ *and* $P = \bot$ *it is called* universal dead-end*.*

Unconstrained Q-blocks just represent moves in which the players do not need to comply to any rule: Choices range over the full cartesian product of the domains of variables, like in a standard QCSP. On the contrary, dead-ends are corner cases in which no valid move exists. We want to keep under control the presence of these corner cases in the QCSP$^+$ problems we manage. We also want to avoid universal Q-blocks as tailing blocks, because—intuitively—the last move is redundant if it is made by the universal player with the aim of contradicting the empty problem $\top$. To these aims, we pose the following

Definition 10 (Normal QCSP$^+$). *A QCSP$^+$ $[b_1, \ldots, b_n]$ is* normal *if at the same time (i) b_n is existential, (ii) no Q-block in the problem is a universal dead-end, and (iii) at most one block is an existential dead-end, and it is exactly the last one, i.e. b_n.*

We associate a normal QCSP$^+$ to any (non-normal) QCSP$^+$ as follows.

Definition 11 (Normalized QCSP$^+$). *Given a QCSP$^+$ $P = [b_1, \ldots, b_n]$, let k be the smallest integer such that b_k is a dead-end (or put $k = n$ if no such block exists). The normalized QCSP$^+$ associated to P, denoted as $norm(P)$ is*

- $[b_1, \ldots, b_k]$ *if b_k is existential;*
- $[b_1, \ldots, b_{k-1}]$ *if b_k is universal.*

The intuition behind the normalization of a QCSP$^+$ is tightly related to its nested alternated quantification structure. It captures the essential property that when the CSP characterizing the moves allowed at some intermediate step of the game becomes inconsistent, all the subsequent moves are no longer relevant and can be cut away. The only remaining problem is whether or not the player having such future "impossible" move to perform can be really forced to reach the inconsistent stage and hence to loose the game. The only alternative is—as we will see explicitly after the evaluation function for QCSP$^+$ is introduced—that the player facing a future dead-end can in turn force a dead-end for the other player to occur sooner than his own one. Whether or not this is possible inherently depends on the game rules. What is interesting at this level is that the tails of dead-ended games need not to be represented and can be cut/normalized.

Before giving a semantics to QCSP$^+$, we need to introduce *substitutions* of values for variables in a set of constraints. Given a constraint $c = (W, T)$, let D_v be the domain of a variable $v \in W$, and let $a \in D_v$ be an element in the domain of v. By $c[a/v]$ we denote the constraint obtained from c by substituting the variable v with the element a. The result is a new constraint $c' = c[a/v] = (W', T')$ where $W' = W \setminus \{v\}$ and $T' = \{t \in D^{W'} | t : [v = a] \in D^W\}$. The result of applying more substitutions in sequence does not depend on the order of application, so we can univocally generalize substitutions to tuples. Given a set of variables $V = \{v_1, \ldots, v_m\} \subseteq W$, and a tuple of values $A = \langle a_1, \ldots, a_m \rangle$ such that $a_j \in D_{v_j}$ for $j = 1, .., m$, we write $c[A/V]$ to denote the constraint $c[a_1/v_1] \cdots [a_m/v_m]$. Finally, substitutions can be extended to sets of constraints (CSPs) by defining as c the result of each substitution $c[a/v]$ such that $v \notin var(c)$: Given a CSP $P = \{c_1, \ldots, c_n\}$, it is $P[A/V] = \{c_1[A/V], \ldots, c_n[A/V]\}$.

Definition 12 (Substitution in a Q-block). *Given a Q-block $b = (\mathcal{Q}, V, D, C)$ and a substitution $[a/v]$ with $v \in free(b) = var(C) \setminus V$ and $a \in D_v$, the result of applying the substitution $[a/v]$ to the Q-block b, denoted as $b[a/v]$, is a new Q-block $b' = \langle \mathcal{Q}, V', D', C' \rangle$ where $C' = C[a/v]$, $V' = V \setminus \{v\}$ and $D' = D \setminus D_v$. This definition naturally generalizes to tuple substitutions $[A/V]$.*

Definition 13 (Substitution in QCSP$^+$). *Given a QCSP$^+$ $P = [b_1, \ldots, b_n]$ with free variables F and a substitution $[a/v]$ with $v \in F$, the result of applying $[a/v]$ to P, denoted $P[a/v]$, is defined as $norm([b_1[a/v], \ldots, b_n[a/v]])$. This definition naturally generalizes to tuple substitutions $[A/V]$.*

Definition 14 (QCSP$^+$ evaluation).
The total evaluation function $\mathsf{eval} : QCSP^+ \to \{T, F\}$ for a closed normal QCSP$^+$ $P = [b_1, \ldots, b_n]$ is inductively defined as follows.

base case, n=1 *It is $\mathsf{eval}([\langle \exists, V, D, P \rangle]) = T$ iff for at least one substitution $[A/V]$ of values in D for the variables V the CSP $P[A/V]$ is not inconsistent.*

inductive case, n¿1 *Let b_1 be $\langle \mathcal{Q}, V, D, C \rangle$ and P' be $[b_2, \ldots, b_n]$ (notice that P' is a QCSP$^+$ with free variables $V \cup free(b_1)$). It is $\mathsf{eval}(P) = T$ iff*

- *$\mathcal{Q} = \forall$ and for every substitution $[A/V]$ of values in D for the variables V such that $C[A/V]$ is not inconsistent we have $\mathsf{eval}(P'[A/V]) = T$;*
- *$\mathcal{Q} = \exists$ and for some substitution $[A/V]$ of values in D for the variables V such that $C[A/V]$ is not inconsistent we have $\mathsf{eval}(P'[A/V]) = T$;*

The last definition conveys to a QCSP$^+$ the intuitive meaning we associate to an expression like (5), which we can finally summarize as follows: Each Q-blocks characterizes the admissible moves at some step in a finite-lenght two-player game. In a Q-block $\langle \mathcal{Q}, V, D, C \rangle$ the $\mathcal{Q}$-player chooses one value for each variable V (in their respective domains D) subject to the condition that the CSP C has not to be made inconsistent by such choice. The game then continues with subsequent moves in a left-to-right direction. A move made at some step in the game propagates its effects in (possibly) all the subsequent steps, a thing formally captured by the substitution $P'[A/V]$. The definition of the eval function assigns to the $\forall$-player the goal of finding (legal) moves that make the inductive evaluation of $P'[A/V]$ fail (i.e. be F), and to the $\exists$-player the goal of finding (legal) moves that make the inductive evaluation of $P'[A/V]$ succeed (i.e. be T). Finally, by the time a move in a Q-block is choosen, all the free variables in the CSP of that block have got a value through substitution, so it is always possible to univocally decide whether a move is admissible or not.

To conclude, let us point out two nice properties that stem from Definition 14, and link QCSP$^+$ to standard QCSP problems.

- If every Q-block in a QCSP$^+$ problem is *unconstrained*, then the problem is equivalent to a standard QCSP. In particular, the QCSP$^+$ problem

$$\mathcal{Q}_1 V_1 \in D_1\ [C_1].\ \mathcal{Q}_2 V_2 \in D_2\ [C_2].\ \cdots\ \ \mathcal{Q}_n V_n \in D_n\ [C_n].\ P$$

  is equivalent to the standard QCSP problem

$$\mathcal{Q}_1 V_1 \in D_1 \mathcal{Q}_2 V_2 \in D_2\ \cdots\ \ \mathcal{Q}_n V_n \in D_n.\ P$$

  if the CSP in every C_i is empty: By syntactically dropping away rules from a QCSP$^+$ we fall back in the domain of standard QCSPs.
- Let us define the negation of a QCSP$^+$ problem P as an operation that yields as a result a problem $\overline{P}$ which evaluates to T if and only if P evaluates to F, for just about any problem P. For a QCSP problem, if we pose $\overline{\exists} = \forall$ and $\overline{\forall} = \exists$ the negation of

$$\mathcal{Q}_1 V_1 \in D_1 \mathcal{Q}_2 V_2 \in D_2\ \cdots\ \ \mathcal{Q}_n V_n \in D_n.\ P$$

  happens to be

$$\overline{\mathcal{Q}}_1 V_1 \in D_1 \overline{\mathcal{Q}}_2 V_2 \in D_2 \ \cdots \ \overline{\mathcal{Q}_n} V_n \in D_n.\ \overline{P}$$

where $\overline{P}$ is a set of constraints inconsistent if and only if P is not inconsistent. A nice thing about QCSP$^+$ is that rules need not to be negated, so we can prove that the negation of

$$\mathcal{Q}_1 V_1 \in D_1\ [C_1].\ \mathcal{Q}_2 V_2 \in D_2\ [C_2].\ \cdots \ \mathcal{Q}_n V_n \in D_n\ [C_n].\ P$$

is just

$$\overline{\mathcal{Q}_1} V_1 \in D_1\ [C_1].\ \overline{\mathcal{Q}_2} V_2 \in D_2\ [C_2].\ \cdots \ \overline{\mathcal{Q}_n} V_n \in D_n\ [C_n].\ \overline{P}$$

4 Decision procedure

We devise a decision procedure for QCSP$^+$ which closely mimics Definition 14 for eval except that it enumerates variables of a block one by one. Such procedure consists in a classical depth-first search of the and/or tree associated with the problem, and is essentially the same search-based procedure proposed in the literature to decide standard QCSP problems [9].

A key amendment for our case is that the search must be confined to explore legal moves only, according to the restrictions enforced by each Q-block. An even more important adaptation concerns the role of *forward inference*. As usual in (Q)CSP, a form of forward inference—called *propagation* and aimed at enforcing *local consistency*—is of key importance to prune the search space that remains to be visited, and is exercised in each search node. What is peculiar to QCSP$^+$—more than the search strategy—is exactly the way this propagation operates. We focus on this topic in the next section, then come back to compose a complete search procedure in Section 4.2.

4.1 Propagation in QCSP$^+$

Our notion of propagation in the QCSP$^+$ setting builds on top of the analogous notions developed for standard (Q)CSPs. We give an informal introduction to the latter here, and refer the reader to [2, 9] for a formal treatment.

Consistency checks in (Q)CSPs aim at identifying *irrelevant* values in the domain of variables. A value in a domain is irrelevant when its removal does not change the validity of the overall quantified problem. Thus, the value $a \in D_X$ of an existential (universal) variable X is irrelevant when the problem is guaranteed to be *false* (*true*, respectively) if the value a is chosen for X. Once values are known to be irrelevant, their validity-preserving removal is performed. In some cases this sole operation may lead to solve the entire problem. Even if this is not the case, the problem itself is simplified and the redundancy the search procedure experiences is greatly reduced.

A sufficient condition for an existential value to be irrelevant is that it *locally* contradicts some constraint (*local consistency* check). For plain QCSPs, this form of per-constraint identification and removal of irrelevant existential values is an operation called *quantified arc-consistency* (qac). It has been defined and studied in [7]. Irrelevant universal values are inherently not a local notion, so they cannot be identified by qac. What qac can do is just to identify universal values that contradict some constraint (a circumstance called *failure*, which leads to declare the entire problem as false).

Value removals can lead to identify new inconsistencies, and this may trigger further removals, in a process called *propagation*. To represent the *state* of propagation,

a set of admissible values for each variable is maintained, which over-approximate the set of values taking part in solutions. A common model of constraint propagation is to associate a domain-reduction operator r_c to each constraint c in a given CSP and to compute their greatest common fixpoint by a chaotic iteration [2].

Let us now move on to consider QCSP$^+$: We propose a notion of propagation for QCSP$^+$ based upon the notion of propagation for *non-quantified* CSP. The reuse of the more powerful *quantified* propagators in QCSP$^+$ is possible as well, but we argue in the next section that there are many practical benefits in not doing so. A propagator for a non-quantified CSP is modeled is a function prop that given a CSP P where the variables $V = var(P)$ range over domains $D = (D_X)_{X \in V}$, yields a new family of domains $D' = (D'_X)_{X \in V}$, written $\mathsf{prop}(V, D, P) = D'$, with a few key properies:

contractance: domains are not enlarged, i.e. $D'_v \subseteq D_v$ for every $v \in V$.

correctness: no solution of the CSP P is lost moving from D to D'; this means that if we select any value $a \in D_v \setminus D'_v$ for any variable $v \in V$ such that $D_v \setminus D'_v$ is not empty, then no problem $P[a/v]$ has any solution when the remaining variables $V \setminus \{v\}$ range over domains $D \setminus D_v$;

monotonicity: the inclusion relation among families of domains is preserved; this means that if two families of domains D and E on variables V are such that $D_v \subseteq E_v$ for every $v \in V$, then necessarily $D'_v \subseteq E'_v$ for every $v \in V$ where $D' = \mathsf{prop}(V, D, P)$ and $E' = \mathsf{prop}(V, E, P)$;

singleton completeness: if all the domains are singleton, the propagator is able to say a final word about the consistency of the CSP. Let us consider a family of domains D over variables V such that for every $v \in V$ it is $|D_v| = 1$. If the problem $P[a_1/v_1] \cdots [a_n/v_n]$ is inconsistent (where a_i is the unique element in the domain of v_i) then at least one of the domains in $\mathsf{prop}(V, D, P)$ is empty.

How do we reuse prop inside a QCSP$^+$? There are many CSPs in a single QCSP$^+$ and manys Q-blocks introducing and owning certain variables and domains, so the problem is how to soundly intermix propagation and domain reductions over such a sequence of Q-blocks. We define a function that associate to each QCSP$^+$ instance a "smaller" QCSP$^+$ instance while preserving validity. The notion of "being smaller" is formalized as follows.

Definition 15 (Contracted QCSP$^+$). *Given a QCSP$^+$ $P = [b_1, \ldots, b_n]$, where $b_i = \langle \mathcal{Q}_i, V_i, D_i, C_i \rangle$, a QCSP$^+$ $P' = [b'_1, \ldots, b'_m]$ with $b'_i = \langle \mathcal{Q}'_i, V'_i, D'_i, C'_i \rangle$ is* contracted from P *if $m \leq n$, and for every $i = 1..m$ we have $\mathcal{Q}_i = \mathcal{Q}'_i$, $\mathcal{C}_i = \mathcal{C}'_i$, $\mathcal{V}_i = \mathcal{V}'_i$, and $D'_v \subseteq D_v$ for every $v \in V_i$ with $D_v \in D_i$ and $D'_v \in D'_i$.*

Essentially, a QCSP$^+$ is contrated by possibly cutting the game at some depth and/or by possibly reducing the domain of the variables in each Q-block.

Definition 16 (Validity-preserving QCSP$^+$ propagator). *A validity-preserving QCSP$^+$ propagator is any function $\mathsf{prop} : QCSP^+ \to QCSP^+$ such that for every QCSP$^+$ P the problem $P' = \mathsf{prop}(P)$ is contracted w.r.t. P and is such that $\mathsf{eval}(P) = \mathsf{eval}(P')$.*

The intuition behind the *cascade propagation* we define next is that each Q-block is an "autoritative" source of information about the valid prunings to be performed, in so far as *it talks about variables it owns*. So, a Q-block cannot claim for the global removal of values from domains it does not own. Nevertheless, it is allowed to perform such removal in a *local* sense, as this could be beneficial to further pruning his owned domains. The overall mechanism thus resembles a parallel fixpoint computation in which every Q-block (i) takes for granted the prunings performed

by dominating Q-blocks on their owned variables, which locally are free variables, (ii) applies propagators to prune all the domains of variables locally mentioned—whether free or not, and (iii) estabilishes in a global sense the new domain of every variable it owns (while keeping the results of reductions over non-owned domains private). Such parallel computation is best shaped as a *cascade* of computations as soon as we realize that information cannot soundly back-propagate from a Q-block to any of its dominating Q-blocks. So, the most natural way to reach a global fixpoint is to execute all the local Q-block fixpoint computations just once, in a left to right order, taking care to collect in the final result the autoritative prunings only, and hiding the other ones.

To simplify the following definition, let us introduce the symbol [] to mean an empty QCSP$^+$ (made up of no Q-blocks), and let us use the $\cup$ operator to join families of domains (e.g. if $D_1 = (D_a, D_b)$ and $D_2 = (D_c, D_d)$ then $D_1 \cup D_2 = (D_a, D_b, D_c, D_d)$.

Definition 17 (Cascade propagation). *Given a CSP propagator* prop*, and a QCSP$^+$ problem $P = [b_1, \ldots, b_n]$ with free variables F ranging over domains $D^F = (D_X)_{X \in F}$, the QCSP$^+$ problem obtained by* cascade propagation *from P, written* cascade(P)*, is inductively defined as follows.*

base case. cascade$([]) = []$

inductive case. *Let b_1 be $\langle \mathcal{Q}, V, D, C \rangle$ and $D' =$ prop$(V \cup F, D \cup D^F, C)$. If some domain $D_v \in D'$ is empty, then* cascade$(P) = []$ *if $\mathcal{Q} = \forall$ and* cascade$(P) = [\bot]$ *if $\mathcal{Q} = \exists$. Otherwise, it is* cascade$(P) = [\langle \mathcal{Q}, V, D', C \rangle, b'_2, \ldots, b'_n]$*, where $[b'_2, \ldots, b'_n]$ is obtained by cascade propagation over $[b_2, \ldots, b_n]$ with the free variables $F \cup V$ ranging over $D^F \cup D'$.*

The desired result we establish about cascade propagation is the following.

Theorem 18. *Cascade propagation is a validity preserving QCSP$^+$ propagator.*

4.2 Search-based decision algorithm

The cascade propagation defined in the previous section gives means to simplify the problem we have to decide, while preserving its validity. To obtain a complete decision algorithm we put together Definition 14 with Theorem 18, and we obtain the following.

Theorem 19. *Algorithm 1 is sound and complete.*

Notice that in the algorithm the actual manipulations over the QCSP$^+$ are performed either in the substitutions applied before the recursive calls (which in turn leverage the normalization operator), or in the cascade propagation excercised at each step. Both these operations possibly cut the problem by removing tailing Q-blocks: This happens when the CSP in some intermediate Q-block is already decided to be a dead-end. Cascade propagation may in addition remove values from (universal or existential) domains, thus reducing the branching factor of the search. Such reduction is guaranteed by Theorem 1 not to change the overall result of the evaluation.

Let us point out that cascade *symmetrically* removes values from existential and *universal* domains, the latter case being interesting for two reasons: (i) quantified arc consistency as defined in QCSP is not able to prune universal domains, so the burden of exploring every children of a universal node is left to the search procedure, and (ii) this result is achieved by only using a standard CSP propagator, which is

not aware of the existence of quantifiers. Universal domains can be pruned as the notion of "irrelevant values" for a universal variable—which is not recognizable in a QCSP by looking locally at one constraint, and as such is beyond what quantified arc consistency can do—is turned into a local property thank to universal Q-blocks: If one constraint in the CSP associated to a universal Q-block is inconsistent under some substitution of a value a for a universal variable owned by that block, such value a can be pruned as it is only related to invalid moves of the universal player (that would make the sub-game true).

So, it comes out that not only QCSP$^+$ is a much more intuitive framework to modelize quantified constraints (thank to the presence of restricted/qualified quantifications), but that the alternated Q-block structure is also able to push the inference power of unquantified propagation beyond what can be achieved by quantified propagation on flat QCSPs. The fact that cascade is based on standard propagators for CSPs also gives crucial practical benefits. Indeed, instead of struggling to extend CSP propagators to a quantified-aware form, we can just effectively reuse the implementation of existing CSP propagators in a QCSP$^+$ solver, as we discuss in the next session.

Algorithm 1 Eval : QCSP+ evaluation algorithm

```
Require: A QCSP+ P
Ensure: truth value of P
  if P = [] then
    RETURN TRUE
  end if
  Let P = [(Q, V, D, C)|P']
  if V = ∅ then
    RETURN Eval(P')
  else
    P ← cascade(P)
    Let P = [(Q, V, D, C)|P']
    if ∃D_X ∈ D such that D_X = ∅ then
      RETURN (Q ⇔ ∀)
    end if
    Let X ∈ V
    if Q = ∀ then
      for all a ∈ D_X do
        if Eval(P[a/X]) is FALSE then
          RETURN FALSE
        end if
      end for
      RETURN TRUE
    else
      for all a ∈ D_X do
        if Eval(P[a/X]) is TRUE then
          RETURN TRUE
        end if
      end for
      RETURN FALSE
    end if
  end if
```

4.3 Propagating in subsequent scopes

Detection of dead-ends can be speeded-up by maintaining additional information in deeper scopes. The key point to consider is that the CSP in each Q-block introduces some new variables that it owns, and these variables are possibly mentioned in the nested Q-blocks. This means we can maintain several domains for the same variable, one for *each* Q-block after the one in which this variable is first defined. This gives the following definition of search state. Let $Q = [b_0, \ldots, b_{n-1}]$ be a QCSP$^+$. Let us denote by $Q[i..n-1]$ the QCSP$^+$ $[b_i, \ldots, b_{n-1}]$.

Definition 20 (QCSP$^+$ state). *A QCSP$^+$* state *is a sequence* $\overline{s} = [s_0, \ldots, s_{n-1}]$ *of CSP search states such that* $\forall i \in I, s_i = (s_{i_X})_{X \in W_i}$ *where* $s_{i_X} \subseteq D_{i_X}$. *It describes for each variable the set of values which are not irrelevant for* $Q[i..n-1]$.

We define the directed intersection between two CSP search states s_i and s_j such that $i > j$ as $s_i \cap^{\triangleleft} s_j = s'$ where $s'_X = s_{i_X} \cap s_{j_X}$ if $X \in W_j$ and $s'_X = s_{i_X}$ otherwise.

Definition 21 (Cascade propagation, optimized). *Let* Q *be a QCSP$^+$. The result of* Cascade propagation *for* Q *on a search state* $\overline{s}$ *is* $\overline{s'}$ *where* $s'_i = prop_i(s_i \cap^{\triangleleft} s'_{i-1})$ *where* $prop_i$ *is the consistency operator associated to the CSP* C_i.

The intuition is that propagation in a QCSP$^+$ amounts to apply propagation in each Q-block, and then to inherit the changes in nested levels, in such a way that the domain of a variable in a Q-block is always a superset of the domains of the same variable in each nested Q-block. The key property of cascade propagation is that it is *correct*, in the sense that it only removes irrelevant values from domains.

Theorem 22. *Cascade propagation is correct.*

Proof. (sketch) Suppose that a value a of a variable $X \in V_i$ is pruned in a CSP C_j with $i \leq j$. If $q_j = \exists$, then $Q[j..n-1]$ corresponds to the formula $\exists V_j \in D_j\ C_j \wedge Q[j+1..n-1]$. Since the main connector is a conjunction, all the subproblem $Q[j..n-1]$ restricted to $X = a$ is false. On the other side, if $q_j = \forall$, $Q[j..n-1]$ corresponds to the formula $\forall V_j \in D_j\ C_j \rightarrow Q[j+1..n-1]$. Then a value which makes C_j false makes $Q[j..n-1]$ true whatever value $Q[j+1..n-1]$ may have. In both cases, the value is irrelevant for $Q[j..n-1]$. Moreover, the truth value of the subproblem depends only on the j-th Q-block's quantifier and not on the variable's quantifier.

5 Discussion and Related Works

Concerns about the unsuitability of QCSP to model real cases have not been raised so far: The field is too young for people to devise applicative models that clash with the issues addressed in this paper. But, the availability of strong implementations is likely to foster the development of rich models based on quantified constraints. QCSP$^+$ has been designed to provide crucial benefits in such upcoming scenario.

The existence of a "∀-discipline" issue in quantified conjunctive languages has been recently identified by the QBF community [1]. Its impact is dramatic on QCSP remarkably more than on QBF: The latter does not bother at all with the *modeler's viewpoint*, as it is not required for a human to be able to write or understand a QBF specification. Such specification is obtained through a computer-assisted *compilation* process which starts from some higher-level language and terminates in a process called *CNF-ization*. This consists in casting the meaning of any structured formula into a CNF by conjuncting the clause-based local meaning of each sub-formula through the help of auxiliary variables [15]. A natural workaround to the discipline problem comes out of this mechanism: we reduce to a conjunctive matrix the *entire meaning* of e.g., (5). This trick has since ever been adopted—often tacitly—in QBF models, and is the reason why a library of real-world instances exists [13].

Major obstacles prevent us from porting this approach straight to QCSP. First, the modeler should be made responsible to properly capture the semantics of the alternating structure of (4). This threatens one of the fundamental assumption of (Q)CSP: models have to be human-writable and readable. Furthermore, QCSP constraints cannot undergo arbitrary syntactical manipulations as clauses do. The only way of capturing the meaning of (4) would be to use the *reified version* of each constraint [8]. The reified version of $C(X)$ is $C'(X,t)$ where $t \in \{0,1\}$. The

meaning of the last boolean is to capture the truth value of the constraint itself. Reification allows us to capture, for example, the meaning of the QCSP$^+$ formula $\forall X[C_1(X)].C_2(X)$ by writing $\forall X.C_1'(X,a) \wedge C_2'(Y,b) \wedge (\neg a \vee b)$. This method becomes more involved on CSPs featuring more than one constraint and/or multiple quantifier alternations, but at least it shows that QCSP$^+$ is still in *PSpace*. Inescapable complications exist though: (i) for many constraints, no reified version is provided by the CSP environment, and (ii) the shift to the reified version may be considerably detrimental from the *solver's viewpoint* since in many case less propagation is possible. The negative effects of *dispersing* the rules of the game inside a flat matrix have been pointed out in [1] for QBF. Evidences of such issue arising in QCSP are easy to produce. Suppose $\forall X[C_1(X)].C_2(X)$ is being decided by our QCSP$^+$ solver: It is well possible that the initial propagation over C_1 passes on to C_2 where it triggers further propagation, up to the point that the whole problem is possibly decided by just *forward inference*. It can happen that not even one of these inferences may occur in a reified QCSP version if the truth value of the constraints is not determinable. Also branching on these boolean variables would be inefficient. The other solution of encoding the meaning of disjunction into a high-arity constraint processed by a quantified extension of the GAC-scheme [14] is only possible with table constraints or complex predicates.

Recently, ad-hoc techniques have been proposed to mitigate the discipline problem in QBF [1, 16, 18]. They all move important steps towards treating more explicitly the rules the universal player has to comply with. However, these approaches do not introduce explicitly a new language/semantics, and exclusively focus on modifying search-based boolean solvers by either feeding them with additional information on the CNF conversion or exploiting DNF representations for clausal constraints. Pushed by the lack of similar workarounds in QCSP, we more radically address the issue by introducing a new language which (i) solves the $\forall$-discipline problem, (ii) can be decided by reusing most of the existing technology, and (iii) is of course not intended to be exclusively decided by search-based reasoners. In addition, our approach can be easily "back ported" to the realm of QBF, where it yields the QBF$^+$ language, defined as *the restriction of QCSP$^+$ to boolean variables and clausal constraints*. Such new language differs from all the QBF techniques already proposed in the literature, and can be a very competitive alternative for solving the $\forall$-discipline problem in QBF (see next section).

6 Implementation, Models, and Experiments

We implemented our decision procedure for QCSP$^+$ in a system built around the CSP solver *GeCode* [17]. Our system accepts a wide set of quantified constraints in the input language[2], and is publically available from
`http://www.univ-orleans.fr/lifo/Members/lallouet/research/qecode/qecode.html`
under the name of QeCode. At present, no other QCSP system accepts a constraint language as expressive as QeCode$^+$ does[3], and no suited public domain model exists yet. In order to contribute an initial test suite, we devise QCSP$^+$ generators for the examples described in Section 2. Example 1 and 2 do not rely on non-boolean variables/constraints and have a straight translation into QBF or QBF$^+$. Our generators also produce such equivalent propositional formulations. This enables us to exercise our system against QBF solvers. It is to be noticed that such comparison is of relative significance and biased in favor of QBF solvers, as it does not leverage the strengths of QeCode$^+$ (e.g. expressive constraints over finite-domain variables,

[2] All those GeCode knows, under a possibly weakened form of quantified arc consistency. The way CSP propagators have been extended and reused for QCSP is described in [5].
[3] Even if we do not take into account disjunctions.

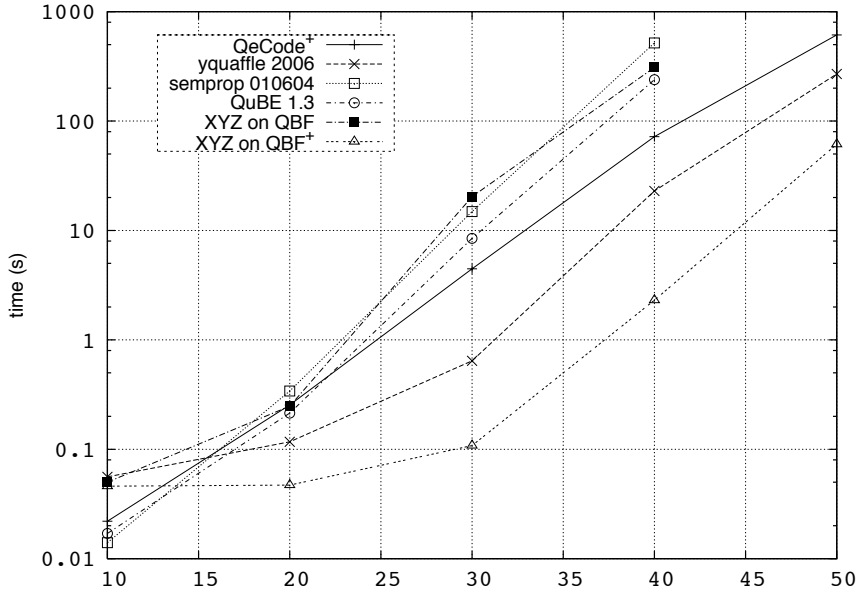

Fig. 3. Running time comparison over the "Strategic Company" family (Example 2, Section 2). The x axis gives the number of companies in the set. We compare QeCode$^+$ with state-of-the-art QBF solvers, and the QBF formulation with the QBF$^+$ one.

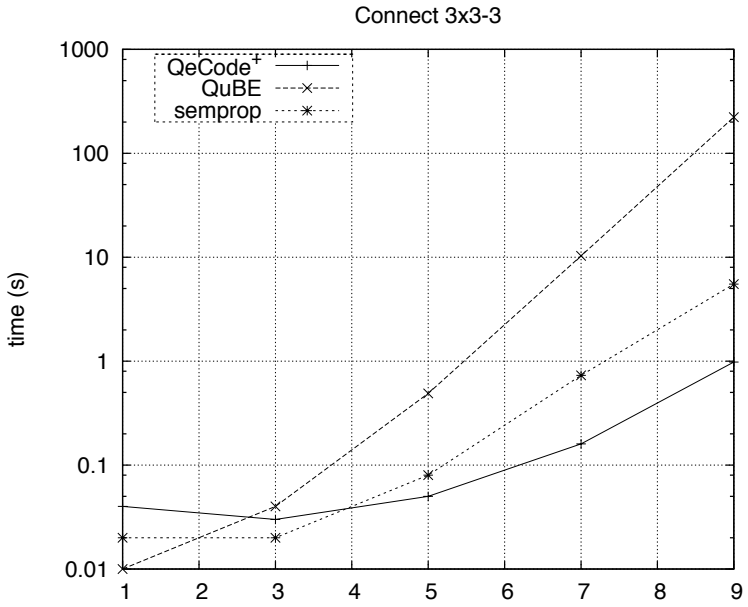

Fig. 4. Comparison over "Connect" $3 \times 3 - 3$ games (Section 2, Example 4). The x axis gives the depth of analysis (number of moves).

non-boolean propagators with a complex semantics), while an important weakness is exposed (i.e. data structures are not tailored to boolean reasoning). Results are shown in Figure 3 and 4, 5, 6. For the "strategic companies" case, it comes out that only one QBF solver, out of the six state-of-the-art ones we have tested[4], can

[4] We also tested Quantor [6] and sKizzo [3], whose performances on these family are weak and are not reported.

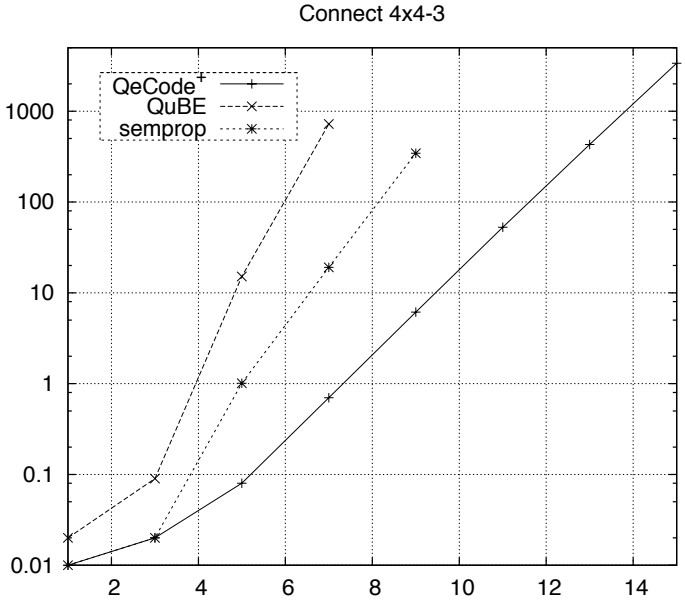

Fig. 5. Comparison over "Connect" $4 \times 4 - 3$games (Section 2, Example 4). The x axis gives the depth of analysis (number of moves).

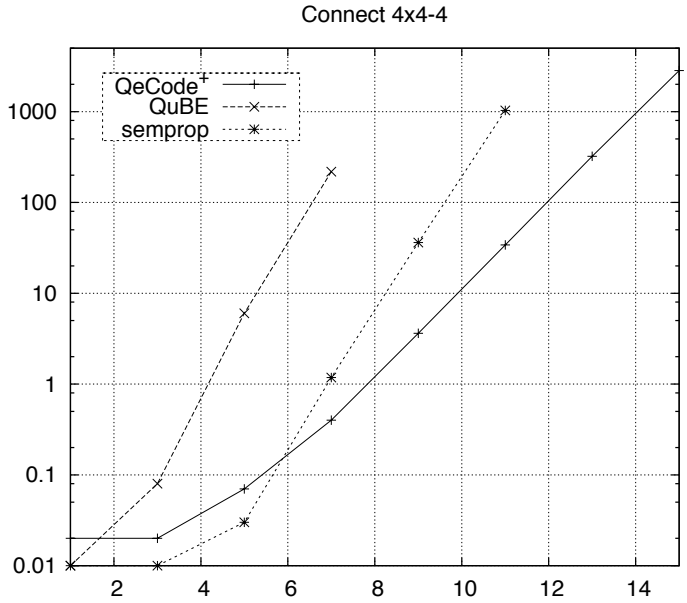

Fig. 6. Comparison over "Connect" $4 \times 4 - 4$games (Section 2, Example 4). The x axis gives the depth of analysis (number of moves).

compete with QeCode$^+$. This result is impressive in the light of a recent work [11] estimating in two to three orders of magnitude the gain (Q)CSP solvers can achieve by reusing the light data structures of SAT/QBF solvers (which GeCODE, hence QeCode$^+$, are not using). To cross-check this hypothesis of gain we modified the QBF decision procedure sKizzo [4] to parse and solve QBF$^+$ specifications[5]: The

[5] The current implementation is limited to $\forall\exists$ prefixes, but the general case equally follows from the results in Section 4.

results labeled "sKizzo on QBF$^+$" in Figure 3 confirm an improvement of one/two orders of magnitude over the best QBF solver and the base sKizzo solver respectively. In the Connect n×m-k games results are quite favorable to QeCode$^+$(Figure 4, 5, 6; only the two best performing QBF solvers are shown). The existence of strategies in board games is a boolean combinatorial problem featuring a large number of quantified alternations, a case known to be quite hard for general purpose reasoners. The basic models we use do not include any symmetry breaking or auxiliary predicate.

7 Conclusions and Future Work

Many obstacles complicate the modelization of real-world problems in purely conjunctive quantified constraint languages, like QCSP. We proposed the QCSP$^+$ language to radically overcome such difficulties: A controlled integration of disjunctive operators allows us to face naturally all the "game discipline" issues that might arise, as we exemplified in different modeling cases. Most importantly, our decision procedure for QCSP$^+$ is designed to inherit the existing conjunction-oriented solving technologies. This allows us to devise the full-fledged solver QeCode$^+$ which knows about dynamic restrictions over quantifications. Incidentally, this solver—which can decide plain QCSPs as special cases—happens to be the first QCSP solver to accept a wide set of constraints in the input language, including global constraints, and one of the first to be available publically. The multi-language generators we devised to establish an initial suite of instances with explicit quantification rules are available as well. Experimental comparisons against QBF solvers on some purely boolean models are quite favorable. However, our main contribution lays elsewhere: we provide the formerly absent possibility to solve real-world models based on quantified constraints.

We are currently working on the modelization of applicative problems in QCSP$^+$ (process scheduling on devices with bounded resources), and in designing decision procedures for QCSP$^+$ and QBF$^+$ that are *not* based on search.

References

1. Carlos Ansótegui, Carla P. Gomes, and Bart Selman. The Achilles' Heel of QBF. In Manuela M. Veloso and Subbarao Kambhampati, editors, *Proc. of AAAI*. AAAI Press AAAI Press / The MIT Press, 2005.
2. K.R. Apt. The essence of constraint propagation. *Theoretical Computer Science*, 221(1-2):179–210, 1999.
3. M. Benedetti. Evaluating QBFs via Symbolic Skolemization. In *Proc. of LPAR'04*. 2005.
4. Marco Benedetti. skizzo: A suite to evaluate and certify qbfs. In Robert Nieuwenhuis, editor, *CADE*, volume 3632 of *Lecture Notes in Computer Science*, pages 369–376. Springer, 2005.
5. Marco Benedetti, Arnaud Lallouet, and Jérémie Vautard. Reusing csp propagators for qcsps. In Francois Fages Francisco Azevedo, Pedro Barahona and Francesca Rossi, editors, *Joint Annual Workshop of the ERCIM Working Group on Constraints and the CoLogNET area on Constraint and Logic Programming*, Lisbon, Portugal, June 26-28 2006.
6. A. Biere. Resolve and Expand. In *Proc. of SAT'04*, 2004.
7. L. Bordeaux and E. Monfroy. Beyond NP: Arc-consistency for quantified constraints. *Proc. of CP*, 2002.
8. Lucas Bordeaux. Boolean and interval propagation for quantified constraints. In Ian Gent, Enrico Giunchiglia, and Kostas Stergiou, editors, *Workshop on Quantification in Constraint Programming*, Barcelona, Spain, 2005.

9. Lucas Bordeaux and Eric Monfroy. Beyond NP: Arc-consistency for quantified constraints. In Pascal Van Hentenryck, editor, *Principles and Practice of Constraint Programming*, volume 2470 of *LNCS*, pages 371–386, Ithaca, NY, USA, 2002. Springer.
10. Marco Cadoli, Thomas Eiter, and Georg Gottlob. Default logic as a query language. *IEEE Transactions on Knowledge and Data Engineering*, 9(3):448–463, 1997.
11. I. Gent, C. Jefferson, and I. Miguel. Minion:Lean, Fast Constraint Solving. *Proc. of ECAI*, 2006.
12. I. P. Gent and A. G. Rowley. Encoding Connect-4 using Quantified Boolean Formulae. *Proc. of Workshop on Modelling and Reform. CSP, Ireland.*, pages 78—93, 2003.
13. E. Giunchiglia, M. Narizzano, and A. Tacchella. Quantified Boolean Formula satisfiability library (QBFLIB), `www.qbflib.org`, 2001, 2001.
14. Peter Nightingale. Consistency for quantified constraint satisfaction problems. In Peter van Beek, editor, *CP*, volume 3709 of *Lecture Notes in Computer Science*, pages 792–796. Springer, 2005.
15. D. A. Plaisted and S. Greenbaum. A Structure-preserving Clause Form Translation. *Journal of Symbolic Computation*, pages 293–304, 1986.
16. A. Sabharwal, C. Ansotegui2, C. P. Gomes, J. W. Hart, and Bart Selman. QBF Modeling: Exploiting Player Symmetry for Simplicity and Efficiency. *Proc. of SAT*, 2006.
17. Christian Schulte and Guido Tack. Views and iterators for generic constraint implementations. *Proc. of CP*, 2005.
18. L. Zhang. Solving QBF with Combined Conjunctive and Disjunctive Normal Form. *Proc. of AAAI06*, 2006.