QCSP Made Practical by Virtue of Restricted Quantification

Marco Benedetti*, **Arnaud Lallouet, and Jérémie Vautard**
LIFO - University of Orléans
BP 6759 – F45067 Orléans Cedex 2, France
name.surname@univ-orleans.fr

Abstract

The QCSP$^+$ language we introduce extends the framework of Quantified Constraint Satisfaction Problems (QCSPs) by enabling us to neatly express *restricted quantifications* via a chain of nested CSPs to be interpreted as alternately conjuncted and disjuncted. Restricted quantifiers turn out to be a convenient solution to the crippling modeling issues we encounter in QCSP and—surprisingly—they help to reuse propagation technology and to prune the search space. Our QCSP$^+$ solver—which also handles arithmetic and global constraints—exhibits state-of-the-art performances.

1 Introduction

We extend the QCSP (Quantified Constraint Satisfaction Problem) framework by introducing a new language, called QCSP$^+$. Such extension is motivated by the difficulties we experienced in modelling and solving most non-trivial problems as plain QCSPs. So, let us start by describing (Q)CSP, in order to point out some of its modeling 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, and countless applications exist indeed.

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 [Bordeaux and Monfroy, 2002]: What if (some of) the variables are quantified *universally*? For example, what means for

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

to be true? The most intuitive way of entering the new scenario is by thinking of it as a *game* between two players. One player is associated to the existential quantifier—we call him the $\exists$-player—the other is related to the universal quantifier: the $\forall$-player. The goal of the $\exists$-player is to satisfy each constraint, hence to satisfy the whole CSP. The goal of the $\forall$-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 $\exists$-player plays first, and he is given the chance to choose a value for x. Then, it is time for the $\forall$-player to assign a value to the variable y. Finally, the $\exists$-player assigns z and the game terminates: The satisfaction of the set of constraints is evaluated. We say that such *quantified* CSP (QCSP) is true if the $\exists$-player has a winning strategy—i.e. if 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 $\forall x \exists y \forall z$. As opposed to the purely existential CSP case, 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 underlying 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*, i.e. forward inference rules that are vital to cut the search space.

*This author is supported by "Le STUDIUM®" (France).

[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 random models that obey merely syntactical generation patterns.

This limitation discourages or prevents people from investing in the production of realistic models. However, we have recently built a full-fledged QCSP solver [Benedetti *et al.*, 2006b], 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 $\exists$ and $\forall$, 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 life-span in relation to the moves already occurred, so to comply with an underlying *game discipline*. The observance of such discipline is precisely what plain QCSPs find difficult to enforce.

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 $\exists$-player: We consider the membership of the move to the set of legal moves as just an additional constraint. If the $\exists$-player cheats, he makes such constraint false, he looses the game.

No similar expedient can be used against the $\forall$-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 $\forall$-*discipline* problem we should slightly modify the whole formalization. This reformulation-based approach has been presented in [Ansótegui *et al.*, 2005] for the case of two-players QBF games but, for the reasons discussed in Section 4, it comes out to be an unsatisfactory remedy to our 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)

$$\forall X_1 \exists Y_1 \forall X_2 \exists Y_2 . C(X_1, X_2, Y_1, Y_2) \qquad (2)$$

Let uppercase letters denote *sets* of variables rather than a single variable (domains are not shown). Suppose the legal opening moves for $\forall$-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, $\exists$-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 $\forall$-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. To capture this scenario, we introduce *restricted quantifiers* $qX[L]$, with $q \in \{\exists, \forall\}$: They quantify over X yet only span over those assignments to X which comply with the CSP-based *rule* L. With this compact notation, our example is written:

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

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...*". We can reshape the above formula as a prenex QCSP as we realize that the "such that" connective stays for a conjunction when $\exists$-player is involved, and for an implication when $\forall$-player is concerned. So, we actually ask whether

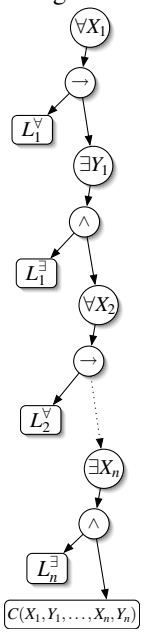
$$\forall X_1 (L_1^{\forall} \rightarrow \exists Y_1 (L_1^{\exists} \wedge \forall X_2 (L_2^{\forall} \rightarrow \exists Y_2 (L_2^{\exists} \wedge C)))) \qquad (3)$$

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

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

Statement (4) shows that the most natural way to express restricted quantifiers would be 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. We would fall back into the inability to exercise CSP tools on real problems, by having a nice formalism with just 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 express all the disjunctions originating from the use of restricted (universal) quantifications, but is (ii) still capable of inheriting and exercising the reasoning core of standard (Q)CSP solvers.



From a FOL perspective, the syntactic shape of the problems we introduce—the QCSP$^+$ problems, generalizing Example (3-4)—is a chain of nested alternated quantifications with restrictions (depicted aside). This language is cognitively adequate to represent game-with-rules scenarios (*modeler's viewpoint*, Section 2), and at the same time is amenable to be decided by reusing existing technology (*solver's viewpoint*, Section 3). Indeed, each restriction rule is a standard CSP, inside which the usual propagation procedures can be capitalized. What is to be designed 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. As a final note, let us observe that the "game-with-rules" scenario we target is definitely not a marginal modeling case. On the contrary, the usage of restricted quantifiers is so natural (see Section 2) that one ends up wondering which non-trivial problems can be modeled as plain QCSPs! The relation between our approach and recent contributions in the close field of QBF decision procedures is discussed in Section 4. The implementation of a full-fledged solver and experiments on quantified models are presented in Section 5. In Section 6 we summarize our contributions and present directions for future work.

2 Motivating Examples

The concepts involved in the modeling of the following common 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 formulation:* $\exists X[P(X)].\, \forall Y[P(Y)].\, Y \preceq X$.

In the following example [Cadoli *et al.*, 1997], solutions to the CSP represent sets, and the $\subseteq$ relation creates preferences.

Example 1 (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 represented 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 $\langle C', c\rangle \in Contr$, if $C' \subseteq S$ then $c \in S$. 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 expressed in propositional logic. We adopted such restricted version to enable a direct comparison with boolean reasoners (Section 5). 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 2 (Game strategy) *Let a set of variables X_i describe the state at step i of a system evolving after the alternate moves of two opponents $p_\forall$ and $p_\exists$, selected 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 $\mathsf{I}(X)$ and $\mathsf{G}(P, X)$ respectively (with $P \in \{p_\forall, p_\exists\}$). Let $\overline{\forall}$ denote $\exists$ and vice-versa. The first player $p_\exists$ wins the game in at most k rounds if $\exists X_0[I(X_0)].\mathsf{WS}(\exists, 1)$, where*

$$\mathsf{WS}(\mathsf{q}, i) := \mathsf{q}X_i, M_i[\ \mathsf{PA}(X_{i-1}, M_i) \wedge \overline{G}(p_{\overline{q}}, X_{i-1}) \wedge \mathsf{SSA}(X_{i-1}, X_i, M_i)].\mathsf{WS}(\overline{\mathsf{q}}, i{+}1)$$

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

The next example shows QCSP$^+$ escaping the intricacies of game formalization we find in e.g. [Gent and Rowley, 2003].

Example 2 (Connect n×m−k) *Let $B = (b_{ij}) \in \{0, p_\forall, p_\exists\}^{n \times m}$ 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 $\mathsf{PA}(B, x) := b_{nx} \neq 0$, where $x \in [1..m]$ is the column of the current move, and using*

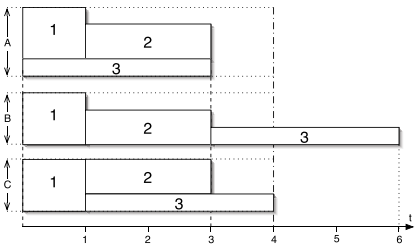
$$\wedge_{i \in [1..n] j \in [1..m]}(b_{ij} \neq b'_{ij}) \leftrightarrow (j = x \wedge b_{ij} = 0 \wedge b'_{ij} = P \wedge b_{i-1\,j} \neq 0)$$

as $\mathsf{SSA}(B, B', x)$, where $P \in \{p_\forall, p_\exists\}$ is the current player. The initial condition $\mathsf{I}(B)$ is $\wedge_{ij} b_{ij} = 0$. The $\overline{\mathsf{G}}$ CSP is a conjunction of allDifferent *constraints excluding each alignment.*

Problem 3 (Conformant Scheduling) *Consider n tasks of duration $\delta_1 \ldots \delta_n$ requiring an amount of resource $r_1 \ldots r_n$, 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, (ii) we comply with O, and (iii) the instantaneous resource consumption never exceeds a fixed capacity R. A global constraint $\mathsf{cumulative}(R, t_1, \delta_1, r_1, \ldots, t_n, \delta_n, r_n)$ 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?*

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

Example 3 *Consider three tasks to be completed within time $T = 4$ and without surpassing a capacity $R = 5$, with $\delta_1 = 1$, $\delta_2 = 2$, $\delta_3 = 3$, $r_1 = 3$, $r_2 = 2$, and $r_3 = 1$. Task 1 has to finish before Task 2 starts. We seek an attack-safe schedule, 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, while the solution (C) to the our QCSP$^+$ instance meets all the requirements.*



3 Formalizing and Deciding QCSP$^+$

Let V be a set of variables and $D = (D_v)_{v \in V}$ the family of their domains (hereafter we tacitly assume an underlying global version of V and D). For $W \subseteq V$, we denote by D^W the set $\Pi_{v \in W} D_v$ of tuples on W. The projection of a tuple t (or a set of tuples T) on a variable v (or a set of variables W) is denoted by $t|_v$ (or $T|_W$). A *constraint* is a couple $c = (W, T)$ with $W \subseteq V$ and $T \subseteq D^W$ (W and T are noted $var(c)$ and $sol(c)$). A *CSP* is a set of constraints. For a CSP C, we note by $var(C) = \bigcup_{c \in C} var(c)$ its variables and by $sol(C) = \bowtie_{c \in C} sol(c)$ its solutions, where—for any $W, U \subseteq V$, $A \subseteq D^W$, and $B \subseteq D^U$—it is $A \bowtie B = \{t \in D^{W \cup U} \mid t|_W \in A \wedge t|_U \in B\}$. We name *range* over $W \subseteq V$ any function R with domain W that associates to each $v \in W$ a subset $R(v) \subseteq D_v$.

Definition 1 (Restricted quantifier) *A* restricted quantifier *is a 4-tuple $\mathcal{Q} = (\mathsf{q}, W, R, C)$ where $\mathsf{q} \in \{\exists, \forall\}$, W is a set of variables, R is a range over W, and C is a CSP.*

Definition 2 (QCSP$^+$) *A QCSP$^+$ with free variables F is a couple $(\mathcal{QS}, G)$, where the* quantification structure *$\mathcal{QS}$ is a finite (possibly empty) sequence $[\mathcal{Q}_1, \ldots, \mathcal{Q}_n]$ of restricted quantifiers $\mathcal{Q}_i = (\mathsf{q}_i, V_i, R_i, C_i)$ for which it holds that*

- $\forall i, j \in [1..n], V_i \cap F = \emptyset$ *and* $i \neq j \Rightarrow V_i \cap V_j = \emptyset$
- $var(C_i) \subseteq W_i \cup F$ *with* $W_i = \bigcup_{j \leq i} V_j$

and G is a CSP, called goal*, with $var(G) \subseteq W_n \cup F$. If $F = \emptyset$ the QCSP$^+$ problem is said to be* closed*.*

Consistently with the examples already presented, we compactly denote such QCSP$^+$ by writing:

$$\mathsf{q}_1 V_1 \in R_1\ [C_1].\ \ldots\ .\ \mathsf{q}_n V_n \in R_n\ [C_n].G$$

Only two closed QCSP$^+$ with an empty quantification structure exists: $([], \top)$ and $([], \bot)$, where $\top$ denotes the empty CSP, and $\bot$ stands for any CSP which contains an empty constraint. Given a QCSP$^+$ problem $P = ([(\mathsf{q}_1, V_1, R_1, C_1), \ldots, (\mathsf{q}_n, V_n, R_n, C_n)], G)$ with free variables F, the result of assigning the value $a \in D_v$ to a variable $v \in F$, written $P[v=a]$, is defined as $P = ([(\mathsf{q}_1, V_1, R_1, C_1[v=a]), \ldots, (\mathsf{q}_n, V_n, R_n, C_n[v=a])], G[v=a])$, where the assignment to a CSP is in turn obtained by collecting the result of the assignments to each constraint: Given a constraint $c = (W, T)$, it is $c[v=a] = c$ if $v \notin W$. Otherwise, it is $c' = c[v=a] = (W', T')$ where $W' = W \setminus \{v\}$ and $T' = \{t' \in D^{W'} \mid \exists t \in D^W, t|_{W'} = t' \wedge t|_v = a\}$. This notion extends naturally to tuple-assignments $[W = t]$, with $W \subseteq V$, $t \in D^W$. For any $W \subseteq V$, $t \in D^W$, and any range R over W, let us write $t \in_W R$ to mean $\forall v \in W\ t|_v \in R(v)$.

Definition 3 (Evaluation of a QCSP$^+$) *Any closed QCSP$^+$ $(\mathcal{QS}, G)$ evaluates to a value in $\{\mathsf{true}, \mathsf{false}\}$ as follows.*

base case. *$\mathsf{eval}(([], \top)) = \mathsf{true}$ and $\mathsf{eval}(([], \bot)) = \mathsf{false}$.*

inductive case. *Let it be $\mathcal{QS} = [(\mathsf{q}, W, R, C) \mid \mathcal{QS}']$. It is $\mathsf{eval}((\mathcal{QS}, G)) = \mathsf{true}$ iff: (i) $\mathsf{q} = \forall$ and for all $t \in_W R$ such that $t \in sol(C)$, it is $\mathsf{eval}((\mathcal{QS}', G)[W=t]) = \mathsf{true}$, or (ii) $\mathsf{q} = \exists$ and there exists $t \in_W R$ such that $t \in sol(C)$ and $\mathsf{eval}((\mathcal{QS}', G)[W=t]) = \mathsf{true}$.*

The decision procedure we implement closely mimics the definition of eval, and is based on a classical depth-first recursive search of the and/or tree associated to a quantification structure. As in CSP solvers, a form of forward-inference (called *propagation*) is of key importance to prune the search space, and is exercised in each search node. What is peculiar to QCSP$^+$ is exactly the way this propagation operates, and the way it builds on top of standard CSP propagation.

We model a CSP propagation scheme as parametric function $\mathsf{prop}_P(\cdot)$—where the parameter P is a CSP—whose domain and co-domain are the families of ranges over $var(P)$. The result $R' = \mathsf{prop}_P(R)$ of a propagation over R is meant to *contract* the range of each variable (hence to reduce the branching factor of the search) while preserving the solutions of P. This means that for any variable $v \in var(P)$ it is $R'(v) \subseteq R(v)$, but at the same time for each solution $t \in sol(P)$, if $t|_v \in R(v)$ then $t|_v \in R'(v)$. Propagation is computed by a chaotic iteration as the greatest common fixpoint of the domain-reduction operators associated to each constraint $c \in P$, where the operator for c prunes values that fail a *local consistency check* against c [Apt, 1999].

If R_1 is a range over W_1 and R_2 is a range over W_2 ($W_1 \cap W_2 = \emptyset$), we define $R = R_1 \sqcup R_2$ as $R(v) = R_1(v)$ if $v \in W_1$ and $R(v) = R_2(v)$ if $v \in W_2$. We note projections of ranges onto sub-domains by $|$, so e.g. $R|_{W_1} = R_1$ and $R|_{W_2} = R_2$.

The analogous of propagation in QCSP$^+$ is as follows.

Definition 4 (Cascade Propagation) *Given a propagation scheme prop, a QCSP$^+$ problem P, and a range R^F for the free variables F of P, the QCSP$^+$ problem $\mathsf{cascade}(P)$ obtained by* cascade propagation *from P is defined as follows.*

base case. *It is $\mathsf{cascade}(([], G)) = ([], \bot)$ if, for any $v \in F$, we have $R'(v) = \emptyset$, where $R' = \mathsf{prop}_G(R^F)$. Otherwise, $\mathsf{cascade}(([], G)) = ([], G)$.*

inductive case. *Let it be $P = ([(\mathsf{q}, W, R, C) | \mathcal{QS}], G)$ and $R' = \mathsf{prop}_C(R^F \sqcup R)$. If for any $v \in F \cup W$ it is $R'(v) = \emptyset$, then $\mathsf{cascade}(P) = ([], \top)$ if $\mathsf{q} = \forall$, and $\mathsf{cascade}(P) = ([], \bot)$ if $\mathsf{q} = \exists$. Otherwise (no empty range in R'), it is $\mathsf{cascade}(P) = ([(\mathsf{q}, W, R'|_W, C) | \mathcal{QS}'], G')$ where $(\mathcal{QS}', G') = \mathsf{cascade}((\mathcal{QS}, G))$ is computed using the new range R' for the free variables $F \cup W$ of $(\mathcal{QS}, G)$.*

The intuition is that propagation at the level of each restricted quantifier (q, W, R, C) is the "authoritative" source of information about the new ranges of W. Conversely, for variables in $var(C) \backslash W$ a private temporary version of the ranges is employed where all the prunings realized by dominating scopes are cumulated. Such collective result flows through the chain of authoritative propagations to boost them, then vanishes.

The alternated structure of QCSP$^+$ endows cascade propagation with an intrinsic advantage over the usual quantified arc-consistency of QCSP: It turns the global consistency check for values in universal domains into a *local* property, decidable at the level of single constraints in the CSP rule of restricted quantifiers. This way, we expunge values from both universal and existential domains, symmetrically, by just (re)using quantification-unaware CSP propagators.

We call *validity preserving* any mapping $\mathsf{op} : QCSP^+ \to QCSP^+$ such that $\forall P \in QCSP^+$, $\mathsf{eval}(P) = \mathsf{eval}(\mathsf{op}(P))$, and we say that op is a *contractor* if $\forall P = (\mathcal{QS}, G) \in QCSP^+$, $P' = (\mathcal{QS}', G') = \mathsf{op}(P)$ it is (i) $|\mathcal{QS}'| \leq |\mathcal{QS}|$, (ii) $var(P') \subseteq var(P)$, and (iii) for every $(\mathsf{q}, W, R, C) \in \mathcal{QS}$, $(\mathsf{q}', W', R', C') \in \mathcal{QS}'$, if $v \in W \cap W'$ then $R'(v) \subseteq R(v)$.

Theorem 1 *Cascade propagation is a validity-preserving contractor for QCSP$^+$ problems.*

The above property guarantees soundness and termination of the decision procedure, which performs cascade propagation at each search node. For details see [Benedetti *et al.*, 2006a].

4 Discussion and Related Works

Concerns about the unsuitability of QCSP to model real cases have not been raised so far[2]: Current random models do not clash with the issues addressed in this paper. QCSP$^+$ has been designed in fact to provide crucial benefits in the upcoming development of more realistic models.

The existence of a "$\forall$-discipline" issue in quantified conjunctive languages has been recently identified by the QBF community [Ansótegui *et al.*, 2005]. A static version of the same question has been discussed for the case of QBF encodings of multi-valued QCSP variables whose boolean mapping needs to avoid "illegal" combinations [Gent *et al.*, 2004].

This problem impacts 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 write or understand a QBF specification. Such QBF is obtained through a computer-assisted *compilation* which starts from

[2]Such weakness is being quickly realized, as witnessed by the independent proposal of a framework focusing on simplifying the QCSP modeling of two-player games [Bessiere and Verger, 2006].

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 [Plaisted and Greenbaum, 1986]. A natural workaround to the discipline problem comes out of this mechanism: We reduce to a conjunctive matrix the *entire meaning* of e.g., (4). This trick has since ever been adopted—often tacitly—in QBF models, and is the reason why a library of real-world instances exists [Giunchiglia *et al.*, 2001].

A more structured QBF encoding technique—applied to two-player games—has been proposed in [Ansótegui *et al.*, 2005], where *indicator variables* are introduced to connect in a suited QBF the SAT-PLAN propositional encoding of pre-conditions, effects, etc [Kautz and Selman, 1992]. However, this approach is shown to lead search-based solvers to explore artificially enlarged search spaces, an effect which can be mitigated either by employing encodings tuned to trigger a synergy between top-down search and boolean inference mechanisms, or by feeding the solver with information on the special role of indicator variables. More recently, DNF representations for boolean constraints have been adopted to reduce the search space [Sabharwal *et al.*, 2006; Zhang, 2006].

These approaches involve intricate details, or do not introduce explicitly a language/semantics. They demand search-based solvers, or exploit boolean-only concepts (e.g. DNF), or focus on (human-level) games rather than on the general issue of modeling concepts through quantified constraints.

Conversely, restricted quantification is a fairly general and neat solution, inspired by analogous forms of quantification found in other modeling languages under the name of *qualified* (Description Logic), or *bounded* (Logic Programming), or *restricted* quantification (Logic and Semantics Theory).

Restricted quantifiers—which allow us to reuse propagation technology and are not exclusively designed for search-based reasoners—can be fully "back ported" to the realm of QBF, where they yield the QBF$^+$ language, defined as *the restriction of QCSP$^+$ to boolean variables and clausal constraints*. QBF$^+$ will be studied in a forthcoming paper, but a first application is presented in the next section.

The QBF-inspired techniques mentioned above can be interpreted as workarounds to bypass restricted quantifiers in QBF. While first evidences suggest (cfr. next section) that restricted quantifiers may be equally rewarding in QBF, their introduction in QCSP is supported by compelling arguments, as major obstacles prevent us from adapting other solutions: First, the modeler should be made responsible for capturing the semantics of the alternating structure in (4). This threatens one of the fundamental assumptions of (Q)CSP: Models have to be human-writable and readable. Second, QCSP constraints cannot undergo arbitrary syntactic manipulations as clauses do. To illustrate the problem, let us consider the QCSP$^+$ formula $\exists X[C_1(X)]\forall Y[C_2(X,Y)].C_3(X,Y)$. One way to attain a QCSP version of this formula would be to define a (single) constraint over $X \cup Y$ whose meaning is $C_1(X) \wedge (\overline{C_2(X,Y)} \vee C_3(X,Y))$, to be processed by a quantified extension of the GAC-scheme [Nightingale, 2005]. Though heavily demanding (in space or time, depending on the technique employed), this is still feasible for small, explicitly defined table constraints. Yet no viable solution exists if the C_{1-3} are (as they usually happen to be) sets of possibly arithmetic/global constraints. Another option is to use the *reified version* of each constraint [Bordeaux, 2005]. The reified version of $C(X)$ is an always consistent constraint $C'(X,y)$, where $y \in \{0,1\}$, which is satisfied as $C'(X,1)$ if $C(X)$ is consistent, and as $C'(X,0)$ otherwise. So, we can write $\exists X.\forall Y.\exists a,b.C_1(X) \wedge C'_2(X,Y,a) \wedge C'_3(X,Y,b) \wedge (\neg a \vee b)$ (a generalized version of such rewrite shows that QCSP$^+$ is still in PSPACE). Inescapable complications arise in practice, as (i) no reified version is provided by the CSP environment for many constraints, and (ii) the shift to the reified version may be considerably detrimental from the *solver's viewpoint*, since reified constraints cannot prune values from domains so long as they don't know the value of their last parameter.

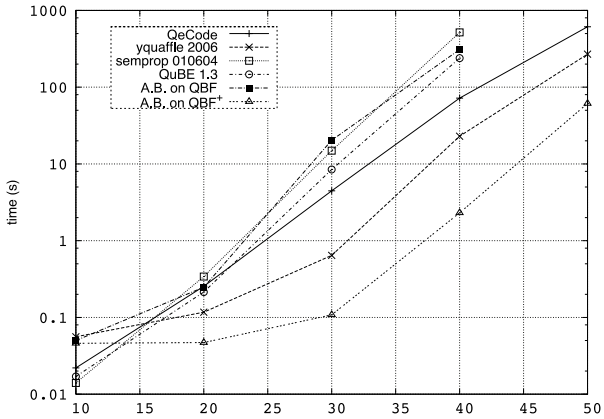


Figure 1: Running time comparison over the "Strategic Company" family (Example 1, Section 2). The x axis gives the number of companies in the set. We compare QeCode with state-of-the-art QBF solvers, and the QBF formulation with the QBF$^+$ one.

5 Implementation, Models, and Experiments

We implemented our decision procedure for QCSP$^+$ in a system built around the CSP solver *GeCode* [Schulte and Tack, 2005]. Our solver—called QeCode—accepts a wide set of constraints in the input language[3], and is publically available from [Benedetti *et al.*, 2006c]. At present, no other system accepts a constraint language as expressive as QeCode does[4], and no suited public domain model exists yet.

We contribute an initial test suite by devising QCSP$^+$ generators for the examples 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. non-boolean variables, 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 1 and 2. For the "strategic companies" case, it comes out that only one QBF solver, out

[3]Actually, all those (the latest version of) *GeCode* knows.

[4]Even if we eschew restricted quantifiers in the modeling.

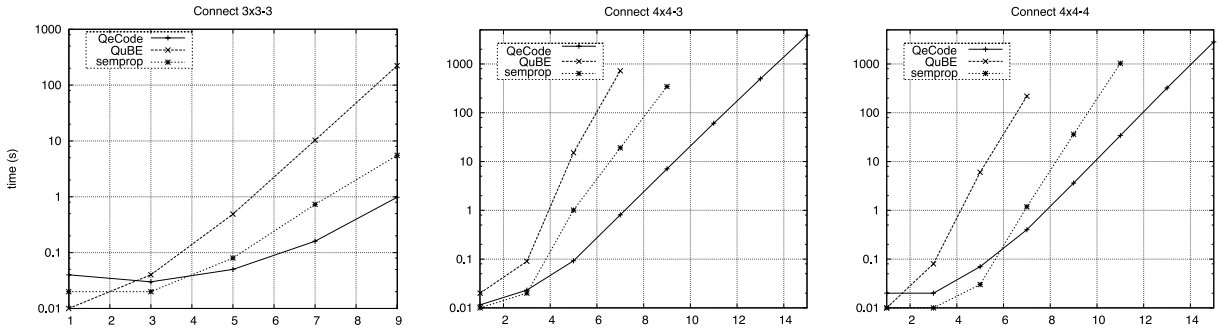


Figure 2: Comparison over some "Connect" games (Section 2, Example 2). The x axis gives the depth of analysis (number of moves).

of the six state-of-the-art ones we have tested[5], can compete with QeCode. This result is impressive in the light of a recent work [Gent *et al.*, 2006] estimating in two to three orders of magnitude the gain (Q)CSP solvers can achieve by using 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 described in [Benedetti, 2006] to parse and solve QBF$^+$ specifications[6]: The results—labeled "A.B. on QBF$^+$" in Figure 1—confirm an improvement of one/two orders of magnitude over the best QBF solver and the base solver respectively.

The existence of strategies in board games is a combinatorial problem with a large number of quantifier alternations known to be hard for general purpose reasoners. For Connect n×m-k results are quite favorable to QeCode (Figure 2, only the two best performing QBF solvers are shown). Our models do not include symmetry breaking or auxiliary predicates.

6 Conclusions and Future Work

Major obstacles complicate the modeling of real-world problems in purely-conjunctive quantified constraint languages, like QCSP or QBF. The "∀-discipline" issue has been identified as the main culprit. We radically overcome such difficulty by introducing the QCSP$^+$ language, where *restricted quantifiers* are allowed. This language enrichment is obtained through a controlled integration of disjunctive operators, and is purposely designed to facilitate the inheritance of existing solving and propagation technology: We implement the full-fledged QCSP$^+$ solver QeCode on top of *GeCode*. Incidentally, QeCode (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.

We devise multi-language generators to establish an initial suite of instances with restricted quantification. Experimental comparisons against QBF solvers on some boolean models are quite favorable. However, our main contribution lays elsewhere: QCSP$^+$ offers the formerly absent possibility to write and solve real-world models based on quantified constraints.

We are currently working on the modeling of applicative problems in QCSP$^+$, and in designing decision procedures for QCSP$^+$ and QBF$^+$ that are not based on search.

[5]We also tested Quantor [Biere, 2004] and sKizzo [Benedetti, 2005] but they are not competitive on these families.

[6]The current implementation is limited to ∀∃ prefixes, but the general case equally follows from the results in Section 3.

References

[Ansótegui *et al.*, 2005] C. Ansótegui, C. Gomes, and B. Selman. The Achilles' Heel of QBF. In *Proc. of AAAI*, 2005.

[Apt, 1999] K.R. Apt. The essence of constraint propagation. *Theoretical Computer Science*, 221(1-2):179–210, 1999.

[Benedetti *et al.*, 2006a] M. Benedetti, A. Lallouet, and J. Vautard. Problem Modeling and Solving in QCSP made Practical. Tech. Rep. 11-06, LIFO, Univ. Orleans, 2006.

[Benedetti *et al.*, 2006b] M. Benedetti, A. Lallouet, and J. Vautard. Reusing CSP propagators for QCSPs. In *Proc. of the Annual ERCIM Workshop on Constraint Solving and Constraint Logic Programming (CSCLP-06)*, Portugal, 2006.

[Benedetti *et al.*, 2006c] M. Benedetti, A. Lallouet, and J. Vautard. `QeCode`'s web page, `www.univ-orleans.fr/lifo/members/vautard/qecode`. 2006.

[Benedetti, 2005] M. Benedetti. Evaluating QBFs via Symbolic Skolemization. In *Proc. of LPAR'04*. 2005.

[Benedetti, 2006] M. Benedetti. Abstract Branching for Quantified Formulas. In *Proc. of AAAI*, 2006.

[Bessiere and Verger, 2006] C. Bessiere and G. Verger. Strategic constraint satisfaction problems. In *Proc. of CP'06 Workshop on Modelling and Reformulation*, Nantes, France, 2006.

[Biere, 2004] A. Biere. Resolve and Expand. In *Proc. of SAT*, 2004.

[Bordeaux and Monfroy, 2002] L. Bordeaux and E. Monfroy. Beyond NP: Arc-consistency for quantified constraints. In *Proc. of CP*, 2002.

[Bordeaux, 2005] L. Bordeaux. Boolean and Interval Propagation for Quantified Constraints. In *Proc. of Workshop on Quantification in Constraint Programming*, Barcelona, Spain, 2005.

[Cadoli *et al.*, 1997] M. Cadoli, T. Eiter, and G. Gottlob. Default logic as a query language. *IEEE Trans. on Knowledge and Data Engineering*, 9(3):448–463, 1997.

[Gent and Rowley, 2003] I. P. Gent and A. G. Rowley. Encoding Connect-4 using Quantified Boolean Formulae. *Proc. of 2^{nd} Int. Workshop on Modelling and Reformulating CSPs, Ireland*, 2003.

[Gent *et al.*, 2004] I. P. Gent, P. Nightingale, and A. Rowley. Encoding quantified CSPs as Quantified Boolean Formulae. In *Proc. of ECAI*, 2004.

[Gent *et al.*, 2006] I. P. Gent, C. Jefferson, and I. Miguel. Minion: Lean, Fast Constraint Solving. *Proc. of ECAI*, 2006.

[Giunchiglia *et al.*, 2001] E. Giunchiglia, M. Narizzano, and A. Tacchella. Quantified Boolean Formula library, `www.qbflib.org`, 2001.

[Kautz and Selman, 1992] H. A. Kautz and B. Selman. Planning as satisfiability. In *Proc. of ECAI*, 1992.

[Nightingale, 2005] P. Nightingale. Consistency for Quantified Constraint Satisfaction Problems. In *Proc. of CP*, 2005.

[Plaisted and Greenbaum, 1986] D. A. Plaisted and S. Greenbaum. A Structure-preserving Clause Form Translation. *Journal of Symbolic Computation*, pages 293–304, 1986.

[Sabharwal *et al.*, 2006] A. Sabharwal, C. Ansotegui, C. P. Gomes, J. W. Hart, and B. Selman. QBF Modeling: Exploiting Player Symmetry for Simplicity and Efficiency. *Proc. of SAT*, 2006.

[Schulte and Tack, 2005] C. Schulte and G. Tack. Views and iterators for generic constraint implementations. In *Proc. of CP*, 2005.

[Zhang, 2006] L. Zhang. Solving QBF with Combined Conjunctive and Disjunctive Normal Form. In *Proc. of AAAI*, 2006.