A semantic tableau version of first-order quasi-classical logic

Anthony Hunter

Department of Computer Science
University College London
Gower Street
London WC1E 6BT
UK

Abstract. Quasi-classical logic (QC logic) allows the derivation of non-trivial classical inferences from inconsistent information. A paraconsistent, or non-trivializable, logic is, by necessity, a compromise, or weakening, of classical logic. The compromises on QC logic seem to be more appropriate than other paraconsistent logics for applications in computing. In particular, the connectives behave in a "classical manner" at the object level so that important proof rules such as modus tollens, modus ponens, and disjunctive syllogism hold. Here we develop QC logic by presenting a semantic tableau version for first-order QC logic.

1 Introduction

Paraconsistent reasoning is important in handling inconsistent information, and there have been a number of proposals for paraconsistent logics (for a review see [Hun98]). However, developing non-trivializable, or paraconsistent logics, necessitates some compromise, or weakening, of classical logic. Key paraconsistent logics such as C_ω [dC74] achieve this by weakening the classical connectives, particularly negation. However this results in useful proof rules such as disjunctive syllogism failing, and intuitive equivalences such as $\neg\alpha \vee \beta \equiv \alpha \rightarrow \beta$ failing.

An alternative, called quasi-classical (or QC) logic, is to restrict the proof theory [BH95,Hun00a]. In this restriction, compositional proof rules (for example, disjunction introduction) cannot be followed by decompositional rules (for example, resolution). Whilst this gives a logic that is weaker than classical logic, it does mean that the connectives behave classically at the object level. We believe the logic is appealing for reasoning with inconsistencies arising in applications such as systems development [HN98], and for reasoning with structured text [Hun00b].

In this paper, we present a first-order version of paraconsistent logic. First we give the semantics for the first-order language and then give a semantic tableau version of the proof theory.

2 First-order QC logic

First-order QC logic is a development of QC logic as defined in [Hun00a]. We assume the usual classical definitions for the language including definitions for a free variable, a bound variable, a ground term, and a ground formula.

Definition 1. *The language of first-order QC logic is that of classical first-order logic. We let $\mathcal{L}$ denote a set of formulae formed in the usual way from a set of predicate symbols, a set of function symbols, a set of variable symbols, and the connectives*[1] $\{\neg, \vee, \wedge\}$*. We also assume there is at least one zero-place function symbol in the set of functions symbols.*

Definition 2. *Let α be an atom, and let $\sim$ be a complementation operation such that $\sim\alpha$ is $\neg\alpha$ and $\sim(\neg\alpha)$ is α. The $\sim$ operator is not part of the object language, but it makes some definitions clearer.*

Definition 3. *Let $\alpha_1 \vee .. \vee \alpha_n$ be a clause that includes a literal disjunct α_i. The focus of $\alpha_1 \vee .. \vee \alpha_n$ by α_i, denoted $\otimes(\alpha_1 \vee .. \vee \alpha_n, \alpha_i)$ is defined as the clause obtained by removing α_i from $\alpha_1 \vee .. \vee \alpha_n$. In the case of a clause with just one disjunct, we assume $\otimes(\alpha_1, \alpha_1) = \bot$.*

Example 1. Let $\alpha \vee \beta \vee \gamma$ be a clause where α, β, and γ are literals. Hence, $\otimes(\alpha \vee \beta \vee \gamma, \beta) = \alpha \vee \gamma$.

The notion of a model in first-order QC logic is based on a form of Herbrand interpretation.

Definition 4. *The* **Herbrand universe** *of $\mathcal{L}$ is the set of ground terms in $\mathcal{L}$ and is denoted $\mathcal{U}(\mathcal{L})$. The* **Herbrand base** *of $\mathcal{L}$ is the set of ground atoms in $\mathcal{L}$ formed using the Herbrand universe of $\mathcal{L}$ and is denoted $\mathcal{B}(\mathcal{L})$.*

Definition 5. *Let $\mathcal{B}(\mathcal{L})$ be the Herbrand Base of $\mathcal{L}$. Let $\mathcal{O}(\mathcal{L})$ be the set of objects defined as follows, where $+\alpha$ is a positive object, and $-\alpha$ is a negative object.*

$$\mathcal{O}(\mathcal{L}) = \{+\alpha \mid \alpha \in \mathcal{B}(\mathcal{L})\} \cup \{-\alpha \mid \alpha \in \mathcal{B}(\mathcal{L})\}$$

We call any $E \in \wp(\mathcal{O}(\mathcal{L}))$ a model. So E can contain both $+\alpha$ and $-\alpha$ for some ground atom α.

We can consider the following meaning for positive and negative objects being in or out of some model E,

$+\alpha \in E$ means α is "satisfiable" in the model
$-\alpha \in E$ means $\neg\alpha$ is "satisfiable" in the model
$+\alpha \notin E$ means α is not "satisfiable" in the model
$-\alpha \notin E$ means $\neg\alpha$ is not "satisfiable" in the model

Since we can allow both an atom and its complement to be satisfiable, we have decoupled, at the level of the model, the link between a formula and its

[1] To provide a succinct presentation, we do not consider implication here.

complement. In contrast, in a classical model, if a model satisfies a literal, then it is forced to not satsify the complement of the literal.

We formalise the notion of satisfiability and extend it to the rest of the language using the following definitions.

Definition 6. *An assignment is a function from the set of variables used in $\mathcal{L}$ to $\mathcal{U}(\mathcal{L})$.*

Definition 7. *Given an assignment A, an X-variant assignment A' is the same as A except perhaps in the assignment for X.*

Definition 8. *For an assignment A, terms in $\mathcal{L}$ are interpreted as follows, where $[.]^A$ is a function from the terms in $\mathcal{L}$ to $\mathcal{U}(\mathcal{L})$.*

$$[c]^A = c \qquad \textit{where } c \textit{ is a constant symbol.}$$

$$[X]^A = A(X) \qquad \textit{where } X \textit{ is a variable symbol.}$$

$$[f(t_1, .., t_n)]^A = f([t_1]^A, .., [t_n]^A) \qquad \textit{where } f \textit{ is a function symbol and } t_1, .., t_n \textit{ are terms.}$$

In this, each ground term in $\mathcal{L}$ is interpreted as the equivalent term in $\mathcal{U}(\mathcal{L})$. Hence the interpretation of terms is independent of the choice of model.

Definition 9. *For an assignment A, an atom $\alpha(t_1, .., t_n)$ in $\mathcal{L}$ is interpreted as follows, where $\models_h$ is a satisfiability relation called* **Herbrand satisfaction**.

$$(E, A) \models_h \alpha(t_1, .., t_n) \quad \textit{iff} \;\; +\alpha([t_1]^A, .., [t_n]^A) \in E$$

$$(E, A) \models_h \neg\alpha(t_1, .., t_n) \quad \textit{iff} \;\; -\alpha([t_1]^A, .., [t_n]^A) \in E$$

Using Herbrand satisfaction, we define two further satisfaction relations, namely strong satisfaction and weak satisfaction, that allow us to define an entailment relation. Essentially, the equivalences in strong satisfaction allow for any formula in $\mathcal{L}$ to be rewritten into a conjunctive normal form, and then into clauses, which can be evaluated with respect to the objects in the model. In addition, the definition for disjunction captures a form of resolution in the semantics. This is needed because the classical relationship between a positive and negative literal has been decoupled.

Definition 10. *Let $\models_s$ be a satisfiability relation called* **strong satisfaction**. *For a model E, and an assignment A, we define $\models_s$ as follows, where $\alpha_1, ..., \alpha_n$ are literals in $\mathcal{L}$, and α is a literal in $\mathcal{L}$.*

$$(E, A) \models_s \alpha \textit{ iff } (E, A) \models_h \alpha$$

$$\begin{array}{l} (E, A) \models_s \alpha_1 \vee ... \vee \alpha_n \\ \quad \textit{iff } [(E, A) \models_s \alpha_1 \textit{ or } ... \textit{ or } (E, A) \models_s \alpha_n] \\ \quad\quad \textit{and } \forall i \textit{ s.t. } 1 \leq i \leq n \\ \quad\quad\quad [(E, A) \models_s \sim\alpha_i \textit{ implies } (E, A) \models_s \otimes(\alpha_1 \vee ... \vee \alpha_n, \alpha_i)] \end{array}$$

For $\alpha, \beta, \gamma \in \mathcal{L}$, we extend the definition as follows,

$(E, A) \models_s \alpha \wedge \beta$ *iff* $(E, A) \models_s \alpha$ *and* $(E, A) \models_s \beta$
$(E, A) \models_s \neg\neg\alpha \vee \gamma$ *iff* $(E, A) \models_s \alpha \vee \gamma$
$(E, A) \models_s \neg(\alpha \wedge \beta) \vee \gamma$ *iff* $(E, A) \models_s \neg\alpha \vee \neg\beta \vee \gamma$
$(E, A) \models_s \neg(\alpha \vee \beta) \vee \gamma$ *iff* $(E, A) \models_s (\neg\alpha \wedge \neg\beta) \vee \gamma$
$(E, A) \models_s \alpha \vee (\beta \wedge \gamma)$ *iff* $(E, A) \models_s (\alpha \vee \beta) \wedge (\alpha \vee \gamma)$
$(E, A) \models_s \alpha \wedge (\beta \vee \gamma)$ *iff* $(E, A) \models_s (\alpha \wedge \beta) \vee (\alpha \wedge \gamma)$

$(E, A) \models_s (\exists X \alpha) \vee \beta$ *iff for some X-variant assignment* A', $(E, A') \models_s \alpha \vee \beta$
$(E, A) \models_s (\forall X \alpha) \vee \beta$ *iff for all X-variant assignments* A', $(E, A') \models_s \alpha \vee \beta$
$(E, A) \models_s (\neg\exists X \alpha) \vee \beta$ *iff for all X-variant assignments* A', $(E, A') \models_s \neg\alpha \vee \beta$
$(E, A) \models_s (\neg\forall X \alpha) \vee \beta$ *iff for some X-variant assignment* A', $(E, A') \models_s \neg\alpha \vee \beta$

The definition for weak satisfaction is similar to strong satisfaction. The main difference is that the definition for disjunction is less restricted. Note, distributivity is implied by the definition of weak satisfaction.

Definition 11. *Let $\models_w$ be a satisfiability relation called* **weak satisfaction**. *For a model E, and an assignment A, we define $\models_w$ as follows, where $\alpha_1, ..., \alpha_n$ are literals in $\mathcal{L}$ and α is a literal in $\mathcal{L}$.*

$(E, A) \models_w \alpha$ *iff* $(E, A) \models_h \alpha$

$(E, A) \models_w \alpha_1 \vee ... \vee \alpha_n$ *iff* $[(E, A) \models_w \alpha_1$ *or ... or* $(E, A) \models_w \alpha_n]$

For $\alpha, \beta, \gamma \in \mathcal{L}$, we extend the definition as follows,

$(E, A) \models_w \alpha \wedge \beta$ *iff* $(E, A) \models_w \alpha$ *and* $(E, A) \models_w \beta$
$(E, A) \models_w \neg\neg\alpha \vee \gamma$ *iff* $(E, A) \models_w \alpha \vee \gamma$
$(E, A) \models_w \neg(\alpha \wedge \beta) \vee \gamma$ *iff* $(E, A) \models_w \neg\alpha \vee \neg\beta \vee \gamma$
$(E, A) \models_w \neg(\alpha \vee \beta) \vee \gamma$ *iff* $(E, A) \models_w (\neg\alpha \wedge \neg\beta) \vee \gamma$

$(E, A) \models_w (\exists X \alpha) \vee \beta$ *iff for some X-variant assignment* A', $(E, A') \models_w \alpha \vee \beta$
$(E, A) \models_w (\forall X \alpha) \vee \beta$ *iff for all X-variant assignments* A', $(E, A') \models_w \alpha \vee \beta$
$(E, A) \models_w (\neg\exists X \alpha) \vee \beta$ *iff for all X-variant assignments* A', $(E, A') \models_w \neg\alpha \vee \beta$
$(E, A) \models_w (\neg\forall X \alpha) \vee \beta$ *iff for some X-variant assignment* A', $(E, A') \models_w \neg\alpha \vee \beta$

Example 2. Let $\Delta = \{\alpha(a), \gamma(b), \forall X(\neg\alpha(X) \vee \beta(X))\}$,
and let $\mathcal{B}(\mathcal{L}) = \{\alpha(a), \beta(a), \gamma(a), \alpha(b), \beta(b), \gamma(b)\}$.
Now let $A = \{X \mapsto a\}$ and $E = \{+\alpha(a), +\beta(a), +\gamma(b)\}$.
This gives the following.

$$(E, A) \models_s \alpha(a)$$
$$(E, A) \models_s \gamma(b)$$
$$(E, A) \models_s \forall X(\neg\alpha(X) \vee \beta(X))$$

Example 3. Let $\Delta = \{\exists X, Y\ \alpha(X, Y), \beta(f(a))\}$,
and let $\mathcal{B}(\mathcal{L}) = \{\alpha(a, a), \beta(a), \alpha(f(a), a), \beta(f(a)), \alpha(f(f(a)), a), \beta(f(f(a))), ...\}$.

Now let $A = \{X \mapsto a, Y \mapsto a\}$ and $E = \{-\alpha(a,a), +\beta(f(a))\}$.
This gives the following.

$$
\begin{array}{c}
(E,A) \models_s \beta(f(a)) \\
(E,A) \models_s \exists X \beta(X) \\
(E,A) \not\models_s \exists X, Y \alpha(X,Y)
\end{array}
$$

Example 4. Let $\Delta = \{\forall X(\neg\alpha(X), \alpha(a))\}$,
and let $\mathcal{B}(\mathcal{L}) = \{\alpha(a)\}$.
Now let $A = \{X \mapsto a\}$ and $E = \{+\alpha(a), -\alpha(a)\}$.
This gives the following.

$$
\begin{array}{cc}
(E,A) \models_s \alpha(a) & (E,A) \models_s \neg\alpha(a) \\
(E,A) \not\models_s \alpha(a) \vee \beta(b) & (E,A) \models_w \alpha(a) \vee \beta(b)
\end{array}
$$

Definition 12. *We polymorphically extend strong satisfaction and weak satisfaction as follows*

$$
\begin{array}{c}
E \models_s \alpha \textit{ iff for all assignments } A, (E,A) \models_s \alpha \\
E \models_w \alpha \textit{ iff for all assignments } A, (E,A) \models_w \alpha
\end{array}
$$

In the following definition, we can see that QC entailment is of the same form as classical entailment except we use strong satisfaction for the assumptions and weak satisfaction for the inference.

Definition 13. *Let $\models_Q$ be an entailment relation, called the QC entailment relation, such that $\models_Q \subseteq \wp(\mathcal{L}) \times \mathcal{L}$, and defined as follows,*

$$
\begin{array}{l}
\{\alpha_1, .., \alpha_n\} \models_Q \beta \\
\qquad\qquad \textit{iff for all } E \textit{ if } E \models_s \alpha_1 \textit{ and ... and } E \models_s \alpha_n \textit{ then } E \models_w \beta
\end{array}
$$

We can consider the strong satisfaction relation as capturing the decomposition of the set of asumptions. Strong satisfaction forces each resolvent β of a clause $\alpha \vee \beta$ to hold if $\sim\alpha$ holds. In contrast, we can consider the weak satisfaction relation as capturing the composition of formulae from resolvents, allowing disjuncts to be introduced.

Example 5. The following illustrate QC entailment.

$$
\begin{array}{c}
\{\alpha(a), \forall X(\neg\alpha(X) \vee \neg\alpha(X))\} \models_Q \neg\alpha(a) \\
\{\alpha(a), \forall X(\neg\alpha(X) \vee \beta(X))\} \models_Q \beta(a) \\
\{\alpha(a)\} \models_Q \exists X \alpha(X)
\end{array}
$$

We can show that $\models_Q$ is non-trivializable in the sense that when Δ is classically inconsistent, it is not the case every formula in $\mathcal{L}$ is entailed by Δ.

Example 6. Let $\beta, \neg\beta$ and α be ground literals in $\mathcal{L}$. So $\{\beta \wedge \neg\beta\}$ is classically inconsistent. However it is not the case that $\{\beta \wedge \neg\beta\} \models_Q \alpha$ holds, since $E = \{+\beta, -\beta\}$ is a model where $E \models_s \beta \wedge \neg\beta$, but $E \not\models_w \alpha$.

However, many classical tautologies do not follow with $\models_Q$. In particular, the classical tautologies do not follow from an empty set.

Example 7. Let $\Delta = \emptyset$. Now consider the classical tautology $\alpha \vee \neg\alpha$. Here $\Delta \models_Q \alpha \vee \neg\alpha$ does not hold. Since $E = \emptyset$ strongly satisfies every formula in Δ, but E does not weakly satisfy $\alpha \vee \neg\alpha$.

This is one illustration of how QC logic is weaker than classical logic. A number of further features of classical logic such as left logical equivalence, conditionalization, cut, and right weakening also fail [Hun00a].

3 The QC semantic tableau

In order to provide an automated proof procedure, we adapt the tableau approach for classical logic that was developed by Smullyan [Smu68]. For this adaptation, we need the following definitions.

Definition 14. *The set of signed formulae of $\mathcal{L}$ is denoted $\mathcal{L}^*$ and is defined as $\mathcal{L} \cup \{\alpha^* \mid \alpha \in \mathcal{L}\}$.*

We will regard the formulae in $\mathcal{L}^*$ without the $*$ symbol as satisfiable and the formulae in $\mathcal{L}^*$ with the $*$ symbol as unsatisfiable.

Definition 15. *We further extend the weak satisfaction and strong satisfaction relations as follows where $\alpha \in \mathcal{L}$.*

$$E \models_s \alpha^* \text{ iff } E \not\models_s \alpha$$
$$E \models_w \alpha^* \text{ iff } E \not\models_w \alpha$$

Definition 16. *For a formula $\alpha \in \mathcal{L}$ with free variable X, and a term $t \in \mathcal{U}(\mathcal{L})$, we let $\alpha[X/t]$ denote the substitution of all occurrences of X in α by t.*

In the definition of QC semantic tableau, there are two types of decomposition rule. The first type is represented by the S-rules given in Definition 17 and the second type is represented by the U-rules given in Definition 18. All the S-rules assume the formula above the line is satisfiable, and all the U-rules assume the formula above the line is unsatisfiable.

Definition 17. *The following are the S-rules for a QC semantic tableau, where t is in $\mathcal{U}(\mathcal{L})$ and t' is in $\mathcal{U}(\mathcal{L})$ but not occurring in the tableau constructed so far. The $\mid$ symbol denotes the introduction of a branch point in the QC semantic tableau.*

$$\frac{\alpha_1 \vee \ldots \vee \alpha_n}{(\sim \alpha_i)^* \mid \otimes(\alpha_1 \vee \ldots \vee \alpha_n, \alpha_i)} \qquad \textit{where } \alpha_1, .., \alpha_n \textit{ are literals}$$

$$\frac{\alpha_1 \vee \ldots \vee \alpha_n}{\alpha_1 \mid \ldots \mid \alpha_n} \qquad \textit{where } \alpha_1, .., \alpha_n \textit{ are literals}$$

$$\frac{\alpha \wedge \beta}{\alpha, \beta} \qquad \frac{\neg\neg\alpha \vee \gamma}{\alpha \vee \gamma}$$

$$\frac{\neg(\alpha \wedge \beta) \vee \gamma}{\neg\alpha \vee \neg\beta \vee \gamma} \qquad \frac{\neg(\alpha \vee \beta) \vee \gamma}{(\neg\alpha \wedge \neg\beta) \vee \gamma}$$

$$\frac{\alpha \vee (\beta \wedge \gamma)}{(\alpha \vee \beta) \wedge (\alpha \vee \gamma)} \quad \frac{\alpha \wedge (\beta \vee \gamma)}{(\alpha \wedge \beta) \vee (\alpha \wedge \gamma)}$$

$$\frac{(\forall X \alpha) \vee \gamma}{(\alpha[X/t]) \vee \gamma} \quad \frac{(\neg\exists X \alpha) \vee \gamma}{(\neg\alpha[X/t]) \vee \gamma} \quad \frac{(\exists X \alpha) \vee \gamma}{(\alpha[X/t']) \vee \gamma} \quad \frac{(\neg\forall X \alpha) \vee \gamma}{(\neg\alpha[X/t']) \vee \gamma}$$

We will refer to the first two rules as the disjunction S-rules, the next six rules as the rewrite S-rules, and the last four rules as the quantification S-rules.

Definition 18. *The following are the U-rules for a QC semantic tableau, where t is in $\mathcal{U}(\mathcal{L})$ and t' is in $\mathcal{U}(\mathcal{L})$ but not occurring in the tableau so far. The $|$ symbol denotes the introduction of a branch point in the QC semantic tableau.*

$$\frac{(\alpha \vee \beta)^*}{\alpha^*, \beta^*} \quad \frac{(\alpha \wedge \beta)^*}{\alpha^* \mid \beta^*} \quad \frac{(\neg\neg\alpha \vee \gamma)^*}{(\alpha \vee \gamma)^*}$$

$$\frac{(\neg(\alpha \wedge \beta) \vee \gamma)^*}{(\neg\alpha \vee \neg\beta \vee \gamma)^*} \quad \frac{(\neg(\alpha \vee \beta) \vee \gamma)^*}{((\neg\alpha \wedge \neg\beta) \vee \gamma)^*}$$

$$\frac{((\forall X \alpha) \vee \gamma)^*}{((\alpha[X/t']) \vee \gamma)^*} \quad \frac{((\neg\exists X \alpha) \vee \gamma)^*}{((\neg\alpha[X/t']) \vee \gamma)^*} \quad \frac{((\exists X \alpha) \vee \gamma)^*}{((\alpha[X/t]) \vee \gamma)^*} \quad \frac{((\neg\forall X \alpha) \vee \gamma)^*}{((\neg\alpha[X/t]) \vee \gamma)^*}$$

We will refer to the first rule as the disjunction U-rule, the second rule as the conjunction U-rule, the next three rules as the rewrite U-rules, and the last four rules as the quantification U-rules.

Definition 19. *A QC semantic tableau for a database Δ and a query α is a tree such that: (1) the formulae in $\Delta \cup \{\alpha^*\}$ are at the root of the tree; (2) each node of the tree has a set of signed formulae; and (3) the formulae at each node are generated by an application of one of the decomposition rules on a signed formula at ancestors of that node.*

The classical definition for semantic tableau incorporates a similar definition to that of Definition 19. The major difference is that in the classical definition, the root has $\Delta \cup \{\neg\alpha\}$ where $\neg\alpha$ is the negation of the query. The reason QC logic doesn't use this is that we have decoupled the classical relationship between a formula and its complement.

Definition 20. *A QC tableau is closed iff every branch is closed. A branch is closed iff there is a formula β for which β and β^* belong to that branch. A tableau is open if there is an open branch. A branch is open if there are no more rules that can be applied, and it is not closed.*

The classical form of tableau also incorporates the definitions given in Definition 20. In Proposition 8, below, we show that a database Δ implies a query α, by QC logic, if and only if a QC tableau for a database Δ and query α is closed. First we consider some examples in Figures 1, 2, and 3.

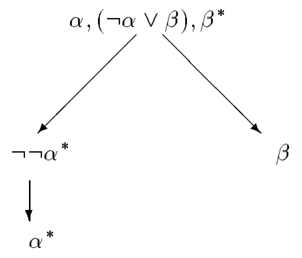

Fig. 1. Let Δ be $\{(\neg\alpha \vee \beta), \alpha\}$, and the query be β. This gives the root $\{\alpha, (\neg\alpha \vee \beta), \beta^*\}$, and the tableau is closed. In this tableau, $(\neg\alpha \vee \beta)$ is decomposed to $\sim\neg\alpha^*$ in the left branch, and $\otimes(\neg\alpha \vee \beta, \alpha)$ in the right branch, where $\sim\neg\alpha^* = \neg\neg\alpha^*$, and $\otimes(\neg\alpha \vee \beta, \neg\alpha) = \beta$. The final step in the left branch is to obtain α^* from $\neg\neg\alpha^*$.

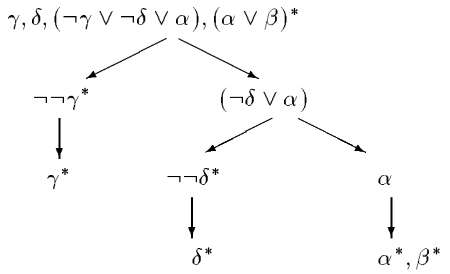

Fig. 2. Let Δ be $\{\gamma, \delta, (\neg\gamma \vee \neg\delta \vee \alpha)\}$ and let the query be $\alpha \vee \beta$. This gives the root $\{\gamma, \delta, ((\neg\gamma \vee \neg\delta) \vee \alpha), (\alpha \vee \beta)^*\}$, and the tableau is closed.

4 Some properties of the QC semantic tableau

Each branch in a QC semantic tableau delineates a class of pairs of (E, A) where E is a model and A is an assignment. As we decompose the formulae in a branch, we refine the class of pairs (E, A).

$$
\begin{tikzcd}
 & \alpha(g(a)), \forall X(\neg\alpha(X) \vee \beta(f(X))), \beta(f(g(a)))^* \arrow[d] & \\
 & \neg\alpha(g(a)) \vee \beta(f(g(a))) \arrow[dl] \arrow[dr] & \\
\neg\neg\alpha(g(a))^* \arrow[d] & & \beta(f(g(a))) \\
\alpha(g(a))^* & &
\end{tikzcd}
$$

Fig. 3. Let Δ be $\{\alpha(g(a)), \forall X(\neg\alpha(X) \vee \beta(f(X)))\}$ and the query be $\beta(f(g(a)))$. This gives the root $\{\alpha(g(a)), \forall X(\neg\alpha(X) \vee \beta(f(X))), \beta(f(g(a)))^*\}$, and the tableau is closed.

Proposition 1. *Each tableau rule given in Definition 17 is sound in the following sense: If $\phi \in \mathcal{L}^*$ is the formula above the line, and $\psi \in \mathcal{L}^*$ is the formula below the line, and E is a model such that $E \models_s \phi$, then $E \models_s \psi$.*

Proof. For any formula $\phi \in \mathcal{L}$ at the root, if we assume that the formula is satisfiable according to the $\models_s$ relation, then we can also assume that any formula resulting from the decomposition of ϕ using the S-rules is also satisfiable using the $\models_s$ relation. We justify this for each of the S-rules as follows: (First disjunction S-rule) According to Definition 10, for all E,A, if $(E, A) \models_s \alpha_1 \vee .. \vee \alpha_n$, then for each α_i either $(E, A) \not\models_s {\sim}\alpha_i$, or $(E, A) \models_s \otimes(\alpha_1 \vee .. \vee \alpha_n, \alpha_i)$. (Second disjunction S-rule) According to Definition 10, for all E,A, if $(E, A) \models_s \alpha_1 \vee .. \vee \alpha_n$, then $(E, A) \models_s \alpha_1$ or ... or $(E, A) \models_s \alpha_n$. (Quantification S-rules) Consider the rule with $\forall X\alpha$ above the line. Here, for all E, A, if $(E, A) \models_s \forall X\alpha$, then for any A' that differs at most in X, and hence for any $t \in \mathcal{U}(\mathcal{L})$, $(E, A') \models_s \alpha[X/t]$. Now consider the rule with $\exists X\alpha$ above the line. Here, for all (E, A), if $(E, A) \models_s \exists X\alpha$, then for some A' that differs at most in X, and here for some t', $(E, A') \models_s \alpha[X/t']$. However, as we do not know which t', we remain impartial, and we select some t' that has not yet been used in the tableau so far. The other two quantification rules follow similarly. (Rewrite S-rules) The soundness for these follow directly from Definition 10.

Proposition 2. *Each tableau rule given in Definition 18 is sound in the following sense: If $\phi \in \mathcal{L}^*$ is the formula above the line, and $\psi \in \mathcal{L}^*$ is the formula below the line, and E is a model such that $E \models_w \phi$, then $E \models_w \psi$.*

Proof. For any formula ϕ^* at the root, if we assume that the formula is unsatisfiable according to the $\models_w$ relation, then we also assume that any formula resulting from the decomposition of ϕ^* using the U-rules is also unsatisfiable using the $\models_w$ relation. We justify this for each of the U-rules as follows: (Disjunction U-rule) According to Definition 11, for all E,A, if $(E, A) \not\models_w \alpha \vee \beta$, then

$(E, A) \not\models_w \alpha$ and $(E, A) \not\models_w \beta$. (Conjunction U-rule) According to Definition 11, for all E,A, if $(E, A) \not\models_w \alpha \wedge \beta$, then $(E, A) \not\models_w \alpha$ or $(E, A) \not\models_w \beta$. (Quantification U-rules) Consider the rule with $(\exists X \alpha)^*$ above the line. Here, for all (E, A), if $(E, A) \not\models_w \exists X \alpha$, then for any A' that differs at most in X, and hence for any $t \in \mathcal{U}(\mathcal{L})$, $(E, A') \not\models_w \alpha[X/t]$. Now consider the rule with $(\forall X \alpha)^*$ above the line. Here, for all (E, A), if $(E, A) \not\models_w \forall X \alpha$, then there is a A' that differs at most in X, and hence a $t' \in \mathcal{U}(\mathcal{L})$, such that $(E, A') \not\models_w \alpha[X/t']$. However, we do not know which t', and so to remain impartial, we select some t' that has not yet been used in the tableau so far. The other two quantification rules follow similarly. (Rewrite U-rules) The soundness for these follow directly from Definition 11.

Proposition 3. *The set of decomposition rules given in Definition 17 is complete in the following sense: If $\phi \in \mathcal{L}$ is a formula in a branch of a QC semantic tableau, and there is a pair (E, A) such that $(E, A) \models_s \phi$, and according to Definition 10 there is a derivation of the form $(E, A) \models_s \phi$ implies $(E, A) \models_s \psi$, then ψ can be obtained as a formula in the branch using the S-rules in Definition 17.*

Proof. The strong satisfaction relation in Definition 10 is defined for non-literal formulae by eleven equivalences. The first, which is for disjunction, is captured by the two disjunction S-rules. The next six are captured by the six rewrite S-rules. The last four, which are the quantification rules, are captured by the four quantification S-rules.

Proposition 4. *The set of decomposition rules given in Definition 18 is complete in the following sense: If ϕ^* is a formula in a branch of a QC semantic tableau, and there is a pair (E, A) such that $(E, A) \models_w \phi^*$, and according to Definition 11 there is a derivation of the form $(E, A) \models_w \phi^*$ implies $(E, A) \models_w \psi^*$, then ψ^* can be obtained as a formula in the branch using the U-rules in Definition 18.*

Proof. The weak satisfaction relation in Definition 11 is defined for non-literal formulae by nine equivalences. The first, which is for disjunction, is captured by the disjunction U-rule. The second, which is for conjunction, is captured by the conjunction U-rule. The next three are captured by the three rewrite U-rules. The last four, which are the quantification rules, are captured by the four quantification U-rules.

Definition 21. *Let B be a branch of a QC semantic tableau where no further decomposition rules can be applied. $F(B) \subseteq \mathcal{L}^*$ is the set of all the formulae at the nodes in the branch. Also let $S(B) = F(B) \cap \mathcal{L}$ and let $U(B) = F(B) - S(B)$.*

Proposition 5. *For any set of formulae $\Delta \in \wp(\mathcal{L})$, and any formula $\alpha \in \mathcal{L}$, and any branch B of a QC semantic tableau for a database Δ and a query α, the branch B is closed iff there is no model E such that $E \models_s \phi$ for all $\phi \in S(B)$ and $E \models_w \phi^*$ for all $\phi^* \in U(B)$.*

Definition 22. *If $\Delta \in \wp(\mathcal{L})$ is of the form $\{\alpha_1, .., \alpha_n\}$, then $\bigwedge \Delta$ denotes the formula $\alpha_1 \wedge .. \wedge \alpha_n \in \mathcal{L}$.*

Proposition 6. *For any set of formulae $\Delta \in \wp(\mathcal{L})$, and any formula $\alpha \in \mathcal{L}$, there is a QC semantic tableau for a database Δ and a query α that is closed iff there is no model E such that $E \models_s \bigwedge \Delta$ and $E \models_w \alpha^*$.*

Proof. First, according to Propositions 1 and 2, each application of a decomposition rule is sound. Second, according to Propositions 3 and 4, the application of the decomposition rules is complete. Now let us consider a particular Δ and α. There is a QC semantic tableau for a database Δ and a query α that is closed $\Leftrightarrow$ Every branch of the semantic tableau with root $\Delta \cup \{\alpha^*\}$ is closed $\Leftrightarrow$ Every branch of the semantic tableau with root $\Delta \cup \{\alpha^*\}$ contains β and β^* for some ground literal $\beta \Leftrightarrow$ There is no model for each branch of the semantic tableau with root $\Delta \cup \{\alpha^*\} \Leftrightarrow$ There is no model E such that $E \models_s \bigwedge \Delta$ and $E \models_w \alpha^*$.

Proposition 7. *For any set of formulae $\Delta \in \wp(\mathcal{L})$, and any formula $\alpha \in \mathcal{L}$, there is an open branch B of a QC semantic tableau for a database Δ and a query α iff there is a model E such that $E \models_s \bigwedge \Delta$ and $E \models_w \alpha^*$.*

Proposition 8. *For any set of formulae $\Delta \in \wp(\mathcal{L})$, and any formula $\alpha \in \mathcal{L}$, a QC tableau for a database Δ and a query α is closed iff $\Delta \models_Q \alpha$ holds.*

Proof. This follows directly from Proposition 6. Let us consider a particular Δ and α. There is a QC semantic tableau for a database Δ and a query α that is closed $\Leftrightarrow$ There is no model E_i such that $E_i \models_s \bigwedge \Delta$ and $E_i \models_w \alpha^* \Leftrightarrow$ For all models E_j, $E_j \not\models_s \bigwedge \Delta$ or $E_j \models_w \alpha \Leftrightarrow$ For all models E_j, if $E_j \models_s \bigwedge \Delta$, then $E_j \models_w \alpha$. $\Leftrightarrow \Delta \models_Q \alpha$ holds

Proposition 9. *The QC semantic tableau collapses to a classical semantic tableau if the following rules are added to the decomposition rules,*

$$\frac{\alpha}{(\neg\alpha)^*} \qquad \frac{\neg\alpha}{\alpha^*} \qquad \frac{\alpha^*}{\neg\alpha} \qquad \frac{(\neg\alpha)^*}{\alpha}$$

and we can use the classical definition for closure of a branch (i.e. the branch contains both β and $\neg\beta$ for some ground atom).

5 Discussion

Developing a non-trivializable, or paraconsistent logic, necessitates some compromise, or weakening, of classical logic. The compromises imposed to give QC logic seem to be more appropriate than other paraconsistent logics for applications in computing. QC logic provides a means to obtain all the non-trivial resolvents from a set of formulae, without the problem of trivial clauses also following.

QC logic exhibits the nice feature that no attention needs to be paid to a special form that the formulae in a set of premisses should have, as long as each formula in the set is individually consistent and not a tautology. This is in contrast with other paraconsistent logics where two formulae identical by definition of a connective in classical logic may not yield the same set of conclusions.

Acknowledgements

The author would like to thank Ralph Miarka and the anonymous referees for some helpful comments.

References

[BH95] Ph Besnard and A Hunter. Quasi-classical logic: Non-trivializable classical reasoning from inconsistent information. In C Froidevaux and J Kohlas, editors, *Symbolic and Quantitative Approaches to Uncertainty*, volume 946 of *Lecture Notes in Computer Science*, pages 44–51, 1995.

[dC74] N C da Costa. On the theory of inconsistent formal systems. *Notre Dame Journal of Formal Logic*, 15:497–510, 1974.

[HN98] A Hunter and B Nuseibeh. Managing inconsistent specifications: Reasoning, analysis and action. *ACM Transactions on Software Engineering and Methodology*, 7:335–367, 1998.

[Hun98] A Hunter. Paraconsistent logics. In *Handbook of Defeasible Reasoning and Uncertainty Management Systems*, volume 2, pages 11–36. Kluwer, 1998.

[Hun00a] A Hunter. Reasoning with conflicting information using quasi-classical logic. *Journal of Logic and Computation*, 10:677–703, 2000.

[Hun00b] A Hunter. Reasoning with inconsistency in structured text. *Knowledge Engineering Review*, 15:317–337, 2000.

[Smu68] R Smullyan. *First-order Logic*. Springer, 1968.