Encoding Choice Logics in ASP[⋆]

Michael Bernreiter, Jan Maly, and Stefan Woltran

DBAI, TU Wien, Austria,
{mbernrei,jmaly,woltran}@dbai.tuwien.ac.at

Abstract. Choice Logics are a tool to express and reason about preferences. A choice logic extends classical propositional logic by adding non-classical binary connectives which express a certain form of ordering over interpretations that satisfy the connective; examples in the literature are Qualitative Choice Logic (QCL) which introduces the concept of ordered disjunction and Conjunctive Choice Logic (CCL) where an ordered variant of conjunction has been proposed. These logics have in common that interpretations ascribe a natural number, called satisfaction degree, to formulas; the lower this satisfaction degree, the more preferable the interpretation. In this paper, we present a general framework to capture several such logics and show how choice logics defined in our framework can be encoded using Answer Set Programming. (ASP).

Keywords: Preferences, Choice Logics, Answer Set Programming

1 Introduction

Preferences are of interest in many areas of research such as economics, psychology, philosophy, but also computer science. Artificial intelligence, databases, and other fields are often concerned with analyzing ”human choice behavior” [13].

One formal framework in which preferences can be expressed is Qualitative Choice Logic (QCL) [4]. QCL extends classical propositional logic by a binary connective $\vec{\times}$, called ordered disjunction. Let F and G be formulas. Then the intuitive meaning of $F \vec{\times} G$ is that if possible, F should be satisfied. If this is not possible, then it is still acceptable, but less preferable, to satisfy only G. Satisfying neither F or G is not acceptable. More specifically, the satisfaction relation of QCL ascribes a natural number (called satisfaction degree) to a formula, given some interpretation. We prefer those interpretations that result in the smallest satisfaction degree for a formula. A further logic based on ordered connectives, called Conjunctive Choice Logic (CCL), has later been introduced in [2]. CCL replaces the ordered disjunction of QCL by another binary connective $\vec{\odot}$, called ordered conjunction. The intuitive meaning of $F \vec{\odot} G$ is that if possible, F and G should be satisfied. If this is not possible, then at least F should be satisfied.

⋆ This work was funded by the Austrian Science Fund (FWF) under grant number Y698 and P31890.

In this paper, we will generalize QCL and CCL by defining a framework for logics in which preferences can be expressed by extending propositional logic with additional non-classical connectives. A logic of this framework will be referred to as a choice logic (CL). For instance, our framework allows to combine QCL and CCL into a new logic in a straightforward way. We provide some preliminary results for general CLs in terms of equivalence. The main contribution of the paper, however, is to encode choice logics from our framework by means of Answer Set Programming (ASP).

While there is a significant amount of literature on adding preferences to core ASP (see, e.g. [6, 5, 9]), only a few papers encoded fundamental preference mechanisms in (standard) ASP. Work in this direction includes, for instance, encodings for the voting domain [7, 12], and a translation from ASP with Resources (RASP) to plain ASP [8]. In fact, we opted here also for encodings in standard ASP, since our general framework of choice logics allows for connectives that can be quite different to concrete ASP add-ons.

2 Introducing QCL and CCL

We briefly summarize the definitions of the two choice logics that motivate our general framework, namely QCL, introduced by Brewka et al. [4], and CCL, introduced by Boudjelida and Benferhat [2]. In their original papers, both logics use slightly different notation for the same concepts. For the sake of uniformity, we stick to one notation that is a mixture of the notation of both papers.

QCL is an extension of propositional logic that adds a new connective $\vec{\times}$.

Definition 1. *The set of QCL-formulas $\mathcal{F}_{QCL}$ is defined inductively as follows:*

1. *$x \in \mathcal{F}_{QCL}$, where x is a propositional variable;*
2. *if $F \in \mathcal{F}_{QCL}$, then $(\neg F) \in \mathcal{F}_{QCL}$;*
3. *if $F, G \in \mathcal{F}_{QCL}$, then $(F \circ G) \in \mathcal{F}_{QCL}$ for $\circ \in \{\wedge, \vee, \vec{\times}\}$.*

Observe that every classical propositional formula is also a QCL-formula.

The semantics of QCL is based on satisfaction degrees. Interpretations ascribe a natural number to formulas. The lower this number (satisfaction degree) is, the more preferable this interpretation is for that particular formula. Before we can define the inference relation for satisfaction degrees, we need the concept of optionality, which expresses the number of satisfaction degrees that a formula can possibly have.

Definition 2. *Let x be a propositional variable, and let F and G be QCL formulas. Then the optionality of a QCL formula is defined as follows:*

1. *$opt(F) = 1$ if either $F = x$ or $F = \neg G$;*
2. *$opt(F \circ G) = max(opt(F), opt(G))$ for $\circ \in \{\wedge, \vee\}$;*
3. *$opt(F \vec{\times} G) = opt(F) + opt(G)$.*

Now we can define the notion of satisfaction degree.[1]

[1] Alternative satisfaction relations for QCL formulas have been proposed in [1]. These do not alter the semantics of $\vec{\times}$, but rather change how the classical connectives ($\neg$, $\wedge$, and $\vee$) operate with respect to the satisfaction degree.

Definition 3. *Let $\mathcal{I}$ be an interpretation, x be a propositional variable, and F and G be QCL formulas. Then the satisfaction degree $k \in \mathbb{N} \cup \{\infty\}$ of a QCL formula under $\mathcal{I}$ is defined as follows:*

1. $\mathcal{I} \mathrel{|\!\!\sim}_k^{QCL} x$ *with* $k = 1$ *if* $x \in \mathcal{I}$ *and* $k = \infty$ *otherwise;*
2. $\mathcal{I} \mathrel{|\!\!\sim}_k^{QCL} \neg F$ *with* $k = 1$ *if* $\mathcal{I} \mathrel{|\!\!\sim}_\infty^{QCL} F$ *and* $k = \infty$ *otherwise;*
3. $\mathcal{I} \mathrel{|\!\!\sim}_k^{QCL} F \wedge G$ *with* $k = max(m, n)$ *for* $\mathcal{I} \mathrel{|\!\!\sim}_m^{QCL} F$ *and* $\mathcal{I} \mathrel{|\!\!\sim}_n^{QCL} G$*:*
4. $\mathcal{I} \mathrel{|\!\!\sim}_k^{QCL} F \vee G$ *with* $k = min(m, n)$ *for* $\mathcal{I} \mathrel{|\!\!\sim}_m^{QCL} F$ *and* $\mathcal{I} \mathrel{|\!\!\sim}_n^{QCL} G$*;*
5. $\mathcal{I} \mathrel{|\!\!\sim}_k^{QCL} F \vec{\times} G$ *iff* $\begin{cases} \mathcal{I} \mathrel{|\!\!\sim}_k^{QCL} F \text{ for } k < \infty \text{ or} \\ \mathcal{I} \mathrel{|\!\!\sim}_\infty^{QCL} F, \mathcal{I} \mathrel{|\!\!\sim}_m^{QCL} G, \text{ and } k = m + opt(F). \end{cases}$

We observe that the satisfaction degree of a formula under a given interpretation is unique. If there is a $k \in \mathbb{N}$ such that $\mathcal{I} \mathrel{|\!\!\sim}_k^{\text{QCL}} F$, we say that $\mathcal{I}$ satisfies F with a degree of k. Otherwise, we say that F is not satisfied under $\mathcal{I}$. As an example, let us consider the formula $F = ((x \wedge y) \vec{\times} x)$. Intuitively, F expresses that it is preferable to satisfy both x and y. If this is not possible, then it is still acceptable to satisfy only x. Formally, this means $\{x, y\} \mathrel{|\!\!\sim}_1^{\text{QCL}} F$ and $\{x\} \mathrel{|\!\!\sim}_2^{\text{QCL}} F$, but $\emptyset \mathrel{|\!\!\sim}_\infty^{\text{QCL}} F$.

Since the purpose of QCL is to express preference, we are often interested in those models that satisfy a formula with minimal satisfaction degree, i.e. we are interested in the preferred models of a formula.

Definition 4. *Let $\mathcal{I}$ be an interpretation and let F be a QCL-formula. Then $\mathcal{I}$ is a preferred model of F if $\mathcal{I} \mathrel{|\!\!\sim}_k^{QCL} F$ with $k \neq \infty$ and for all other interpretations $\mathcal{J}$ we have $\mathcal{J} \mathrel{|\!\!\sim}_m^{QCL} F$ with $k \leq m$.*

Now let us consider CCL. Syntactically, QCL and CCL are equivalent, but we write $\vec{\odot}$ instead of $\vec{\times}$ for the choice connective. The semantics of CCL only differs in the definition of the satisfaction degree of the choice connective. That means, the optionality of a formula in CCL is defined the same way as in QCL and the inference relation for the classical connectives remains unchanged between CCL and QCL. The semantics of the choice connective is as follows:

Definition 5. *Let $\mathcal{I}$ be an interpretation and let F and G be CCL formulas. Then the satisfaction degree of the CCL formula $F \vec{\odot} G$ under $\mathcal{I}$ is k where*

$$k = \begin{cases} m & \text{if } \mathcal{I} \mathrel{|\!\!\sim}_1^{CCL} F \text{ and } \mathcal{I} \mathrel{|\!\!\sim}_m^{CCL} G \\ m + opt(G) & \text{if } \mathcal{I} \mathrel{|\!\!\sim}_m^{CCL} F,\ m \neq 1, \text{ and } m \neq \infty \\ \infty & \text{otherwise} \end{cases}$$

The intuitive meaning of a formula $A \vec{\odot} B$ is that A and B is the most preferred outcome but, if that is not possible, only A is also acceptable but less preferred.

3 A Framework for Choice Logics

QCL and CCL are two logics that only differ in the definition of the semantics of the choice operator. Furthermore, the choice operators considered in QCL and

CCL are not the only sensible choice operators. Consider, for example a choice connective that expresses that one prefers A to B but satisfying both or neither is not acceptable. This connective can express preferences between options that can not be chosen together, for example events at the same time slot. Therefore, we propose a general framework for choice logics (CL) that captures QCL and CCL and can also express other interesting choice connectives.

Syntactically, our framework closely resembles QCL and CCL, with the main difference that we allow, in general, arbitrary many choice connectives.

Definition 6. *The set of choice connectives $\mathcal{C}_\mathcal{L}$ of a choice logic $\mathcal{L}$ is a finite set of binary connectives such that $\mathcal{C}_\mathcal{L} \cap \{\neg, \wedge, \vee\} = \emptyset$. The set $\mathcal{F}_\mathcal{L}$ of formulas of a choice logic $\mathcal{L}$ is defined inductively as follows:*

1. *$x \in \mathcal{F}_\mathcal{L}$, where x is a propositional variable;*
2. *if $F \in \mathcal{F}_\mathcal{L}$, then $(\neg F) \in \mathcal{F}_\mathcal{L}$;*
3. *if $F, G \in \mathcal{F}_\mathcal{L}$, then $(F \circ G) \in \mathcal{F}_\mathcal{L}$ for $\circ \in \{\wedge, \vee\} \cup \mathcal{C}_\mathcal{L}$.*

For $\mathcal{C}_{\mathrm{QCL}} = \{\vec{\times}\}$ we retrieve the syntax of QCL and for $\mathcal{C}_\mathcal{L} = \{\}$ we retrieve the syntax of propositional logic.

For any choice logic, the semantics of the classical connectives for optionality and satisfaction degree are the same as for QCL and CCL. The semantics of a choice connective depends on the intended meaning of the connective. However, we can give reasonable restrictions for the semantics of choice connectives, for example the following upper bound for the optionality of a choice connective:

$$opt_\mathcal{L}(F \circ G) \leq (opt_\mathcal{L}(F) + 1) \cdot (opt_\mathcal{L}(G) + 1).$$

The idea is that F can have at most $opt_\mathcal{L}(F)$ many finite satisfaction degrees, plus the infinite degree ∞, i.e. there are at most $opt_\mathcal{L}(F) + 1$ degrees for F. Analogously for G. Thus, there are $(opt_\mathcal{L}(F) + 1) \cdot (opt_\mathcal{L}(G) + 1)$ possible combinations for how satisfaction degrees can be ascribed to $(F \circ G)$. In addition to giving an upper bound, we can also give the following lower bound:

$$opt_\mathcal{L}(F \circ G) \geq max(opt_\mathcal{L}(F), opt_\mathcal{L}(G)).$$

A choice connective should introduce new ways to distinguish between interpretations, or at least not give less options for doing so.

Lastly, for any reasonable choice connectives, the optionality of $(F \circ G)$ should only depend on the optionality of F and G. No other factors, such as the structure of F or G, should have an influence.

Definition 7. *Let x be a propositional variable, and let F and G be formulas of a choice logic $\mathcal{L}$. Then the optionality of an $\mathcal{L}$-formula is given by the function $opt_\mathcal{L} \colon \mathcal{F}_\mathcal{L} \to \mathbb{N}$ such that*

1. *$opt_\mathcal{L}(F) = 1$ if either $F = x$ or $F = \neg G$;*
2. *$opt_\mathcal{L}(F \circ G) = max(opt_\mathcal{L}(F), opt_\mathcal{L}(G))$ for $\circ \in \{\wedge, \vee\}$;*

3. *for every* $\circ \in \mathcal{C_L}$ *there is a function* $f \colon \mathbb{N}^2 \to \mathbb{N}$ *such that*

$$opt_{\mathcal{L}}(F \circ G) = f(opt_{\mathcal{L}}(F), opt_{\mathcal{L}}(G))$$

with $max(opt_{\mathcal{L}}(F), opt_{\mathcal{L}}(G)) \leq opt_{\mathcal{L}}(F \circ G) \leq (opt_{\mathcal{L}}(F) + 1) \cdot (opt_{\mathcal{L}}(G) + 1)$.

We are now ready to define the notion of satisfaction degrees for an arbitrary choice logic. For the semantics of choice connectives, we impose two crucial restrictions. Firstly, the satisfaction degree of a formula under any given interpretation should never be bigger than its optionality, unless the degree is ∞. After all, the purpose of optionality is to assert the number of satisfaction degrees that a formula can possibly have. Secondly, the satisfaction degree of a formula $F \circ G$ under any given interpretation should only depend on the optionalities and satisfaction degrees of F and G. The structure of the formula or interpretation should have no impact on the satisfaction degree. For example, whether an interpretation assigns an even or uneven number of propositional variables to true should have no influence on the satisfaction degree.

Therefore, the satisfaction degrees in a choice logic are defined as follows. To simplify notation, we write $deg_{\mathcal{L}}(\mathcal{I}, F) = k$ if $\mathcal{I} \mid\!\sim_k^{\mathcal{L}} F$ holds.

Definition 8. *Let $\mathcal{L}$ be a choice logic, $\mathcal{I}$ be an interpretation, x be a propositional variable, and F and G be $\mathcal{L}$-formulas. Then the satisfaction degree of an $\mathcal{L}$-formula under $\mathcal{I}$ is defined as*

1. $\mathcal{I} \mid\!\sim_k^{\mathcal{L}} x$ *with* $k = 1$ *if* $x \in \mathcal{I}$ *and* $k = \infty$ *otherwise:*
2. $\mathcal{I} \mid\!\sim_k^{\mathcal{L}} \neg F$ *with* $k = 1$ *if* $\mathcal{I} \mid\!\sim_\infty^{\mathcal{L}} F$ *and* $k = \infty$ *otherwise;*
3. $\mathcal{I} \mid\!\sim_k^{\mathcal{L}} F \land G$ *with* $k = max(m, n)$ *if* $\mathcal{I} \mid\!\sim_m^{\mathcal{L}} F$ *and* $\mathcal{I} \mid\!\sim_n^{\mathcal{L}} G$*;*
4. $\mathcal{I} \mid\!\sim_k^{\mathcal{L}} F \lor G$ *with* $k = min(m, n)$ *if* $\mathcal{I} \mid\!\sim_m^{\mathcal{L}} F$ *and* $\mathcal{I} \mid\!\sim_n^{\mathcal{L}} G$*;*
5. *for every* $\circ \in \mathcal{C_L}$ *there is a function* $g \colon (\mathbb{N} \cup \{\infty\})^4 \to (\mathbb{N} \cup \{\infty\})$ *such that*

$$deg_{\mathcal{L}}(\mathcal{I}, F \circ G) = g(opt_{\mathcal{L}}(F), opt_{\mathcal{L}}(G), deg_{\mathcal{L}}(\mathcal{I}, F), deg_{\mathcal{L}}(\mathcal{I}, G))$$

with $deg_{\mathcal{L}}(\mathcal{I}, F \circ G) \leq opt_{\mathcal{L}}(F \circ G)$ *or* $deg_{\mathcal{L}}(\mathcal{I}, F \circ G) = \infty$.

As before, we say that $\mathcal{I}$ satisfies F with a degree of k. If $\mathcal{I}$ satisfies F with a finite degree, then $\mathcal{I}$ is called a model of F. Furthermore, we can define preferred models analogously to QCL.

Definition 9. *Let $\mathcal{I}$ be an interpretation and let F be a formula of some choice logic $\mathcal{L}$. Then $\mathcal{I}$ is a preferred model of F, written as $\mathcal{I} \in Mod_{\mathcal{L}}(F)$, if $deg_{\mathcal{L}}(\mathcal{I}, F) \neq \infty$ and for all other interpretations $\mathcal{J}$ we have $deg_{\mathcal{L}}(\mathcal{I}, F) \leq deg_{\mathcal{L}}(\mathcal{J}, F)$.*

3.1 Examples for Choice Logics

A CL is characterized by its choice connectives and by the meaning ascribed to said connectives. We have already seen the definition of QCL and CCL and it is evident that they can be expressed within our framework. As discussed above, we would like a connective $\overrightarrow{\oplus}$ that expresses the following situation: It is preferable to satisfy F; if this is not possible, then G should be satisfied, but it is not acceptable to satisfy both F and G. This connective is based on exclusive disjunction (XOR), similar to how $\vec{\times}$ is based on regular disjunction. We name this connective *ordered exclusive disjunction*, and call the corresponding choice logic Exclusive Disjunctive Choice Logic (XCL).

Definition 10. *XCL is the choice logic such that* $\mathcal{C}_{XCL} = \{\vec{\oplus}\}$,

$$opt_{XCL}(F\vec{\oplus}G) = opt_{XCL}(F) + opt_{XCL}(G),$$

and

$$deg_{XCL}(\mathcal{I}, F\vec{\oplus}G) = \begin{cases} deg_{XCL}(\mathcal{I}, F) & \textit{if } deg_{XCL}(\mathcal{I}, F) < \infty \\ & \textit{and } deg_{XCL}(\mathcal{I}, G) = \infty \\ deg_{XCL}(\mathcal{I}, G) + opt_{XCL}(F) & \textit{if } deg_{XCL}(\mathcal{I}, F) = \infty \\ & \textit{and } deg_{XCL}(\mathcal{I}, G) < \infty \\ \infty & \textit{otherwise} \end{cases}$$

We note that this connective can also be expressed in QCL as $((F\vec{\times}G) \wedge \neg(F \wedge G))$. But if this type of preference has to be expressed often in a given system, then a dedicated choice connective can simplify specifications. Other, simple, choice connectives can not directly be expressed in QCL, for example because they ignore the optionality of a formula.

Definition 11. *L_1 is the choice logic such that* $\mathcal{C}_{L_1} = \{\circ\}$,

$$opt_{L_1}(F \circ G) = opt_{L_1}(F) + 1,$$

and

$$deg_{L_1}(\mathcal{I}, F \circ G) = \begin{cases} deg_{L_1}(\mathcal{I}, F) & \textit{if } deg_{L_1}(\mathcal{I}, F) < \infty \textit{ and } deg_{L_1}(\mathcal{I}, G) < \infty \\ deg_{L_1}(\mathcal{I}, F) + 1 & \textit{if } deg_{L_1}(\mathcal{I}, F) < \infty \textit{ and } deg_{L_1}(\mathcal{I}, G) = \infty \\ \infty & \textit{otherwise} \end{cases}$$

The idea behind $F \circ G$ in L_1 is that it is preferable to satisfy both F and G. If this is not possible, at least F should be satisfied. In this sense, the choice connective of L_1 fulfills the same purpose as ordered conjunction in CCL. However, L_1 does not use optionality to penalize less preferable interpretations. Instead, the degree of such interpretations is simply incremented by 1.

The framework also allows for CLs with multiple choice connectives. For example, one could define a CL that combines the choice connectives of QCL and CCL.

Definition 12. *QCCL is the choice logic such that* $\mathcal{C}_{QCCL} = \{\vec{\times}, \vec{\odot}\}$,

$$\begin{aligned} opt_{QCCL}(F\vec{\times}G) &= opt_{QCL}(F\vec{\times}G), \\ opt_{QCCL}(F\vec{\odot}G) &= opt_{CCL}(F\vec{\odot}G), \\ deg_{QCCL}(\mathcal{I}, F\vec{\times}G) &= deg_{QCL}(\mathcal{I}, F\vec{\times}G), \textit{ and} \\ deg_{QCCL}(\mathcal{I}, F\vec{\odot}G) &= deg_{CCL}(\mathcal{I}, F\vec{\odot}G). \end{aligned}$$

Since our framework is not very restrictive, there are infinitely many CLs. This includes more exotic CLs than the ones we have seen, with less intuitive properties.

3.2 Equivalence of formulas

An important question in any logic is, when are two formulas equivalent. There are different ways to formalize equivalence between two choice formulas. The weakest one is to say that F and G are equivalent if they have the same preferred models, i.e. if $Mod_{\mathcal{L}}(F) = Mod_{\mathcal{L}}(G)$. A stronger notion would be the following.

Definition 13. *Let F and G be formulas of some choice logic $\mathcal{L}$. F and G are degree-equivalent, written as $F \equiv_d^{\mathcal{L}} G$, if for all interpretations $\mathcal{I}$ we have that $deg_{\mathcal{L}}(\mathcal{I}, F) = deg_{\mathcal{L}}(\mathcal{I}, G)$.*

It is easy to see that degree-equivalence of two formulas implies that they also have the same preferred models. However, degree-equivalence does not imply that two formulas can be used interchangeably. This property is usually called strong equivalence [11].

Definition 14. *Let A and B be formulas of some choice logic $\mathcal{L}$. A and B are strongly equivalent, written as $A \equiv_s^{\mathcal{L}} B$, if $Mod_{\mathcal{L}}(F) = Mod_{\mathcal{L}}(F[A/B])$ for all $\mathcal{L}$-formulas F.*

In general, two formulas that are degree-equivalent are not necessarily strongly equivalent. However, for some logics, like L_1, it can be checked that degree-equivalence and strong equivalence coincide. We can show that the two concepts coincide for logics that completely ignore the optionality of formulas.

Definition 15. *A choice logic $\mathcal{L}$ is called optionality-ignoring if for all $\circ \in \mathcal{C}_{\mathcal{L}}$ it holds that if $deg_{\mathcal{L}}(\mathcal{I}, F) = deg_{\mathcal{L}}(\mathcal{I}, F')$ and $deg_{\mathcal{L}}(\mathcal{I}, G) = deg_{\mathcal{L}}(\mathcal{I}, G')$, then $deg_{\mathcal{L}}(\mathcal{I}, F \circ G) = deg_{\mathcal{L}}(\mathcal{I}, F' \circ G')$.*

Theorem 1. *Let $\mathcal{L}$ be an optionality-ignoring choice logic. Then $A \equiv_s^{\mathcal{L}} B$ iff $A \equiv_d^{\mathcal{L}} B$.*

Proof. Assume first that $A \equiv_d^{\mathcal{L}} B$. By the definition of a choice logic, the satisfaction degree of a formula only depends on the satisfaction degree and the optionality of its subformulas. Now, if a logic is optionality ignoring, then the satisfaction degree of a formula only depends on the satisfaction degree of its subformulas. Therefore, $A \equiv_d^{\mathcal{L}} B$ implies $F \equiv_d^{\mathcal{L}} F[A/B]$ and hence $A \equiv_s^{\mathcal{L}} B$.

Assume now that $A \not\equiv_d^{\mathcal{L}} B$. We want to show that $A \not\equiv_s^{\mathcal{L}} B$, i.e. that there is a formula F such that $Mod_{\mathcal{L}}(F) \neq Mod_{\mathcal{L}}(F[A/B])$.

Since $A \not\equiv_d^{\mathcal{L}} B$, there exists an interpretation $\mathcal{I}$ such that $\mathcal{I} \models_m^{\mathcal{L}} A$ and $\mathcal{I} \models_n^{\mathcal{L}} B$ with $m \neq n$. Let $k = min(m, n)$. We claim that there is a formula G such that the minimum degree that $\mathcal{I}$ satisfies G is k. Assume $k = m$. Then the following formula has the desired property.

$$A \wedge \Big(\bigwedge_{x \in \mathcal{I}} x \Big) \wedge \Big(\bigwedge_{x \in var(A) \setminus \mathcal{I}} \neg x \Big).$$

A similar construction works in the case that $k = n$. By renaming the variables, we can assume that there are formulas G and H and interpretations $\mathcal{I}_G$, $\mathcal{I}_H$ such that $\mathcal{I}_G \models^{\mathcal{L}}_k G$, $\mathcal{I}_H \models^{\mathcal{L}}_k H$, $\mathcal{J} \not\models^{\mathcal{L}}_l G, H$ for any interpretation $\mathcal{J}$ and $l < k$, and G and H are variable disjoint from each other as well as from A and B. Additionally, we can assume that $\mathcal{I} \cap \mathcal{I}_G = \emptyset$, $\mathcal{I} \cap \mathcal{I}_H = \emptyset$, and $\mathcal{I}_G \cap \mathcal{I}_H = \emptyset$. We now construct

$$F = (A \wedge G) \vee (x \wedge H),$$

where x is a fresh variable. Observe that the minimal degree with which F (or $F[A/B]$) can possibly be satisfied is k, as either G or H need to be satisfied. Furthermore, $\mathcal{I}_H \cup \{x\} \models^{\mathcal{L}}_k F$ and $\mathcal{I}_H \cup \{x\} \models^{\mathcal{L}}_k F[A/B]$. This means that any preferred model of F must satisfy F with a degree of k. The same is true for preferred models of $F[A/B]$. Also observe that since x is not contained in $\mathcal{I}$ or $\mathcal{I}_G$ the model $\mathcal{I} \cup \mathcal{I}_G$ does not satisfy $(x \wedge H)$.

Assume $k = m$. Then $\mathcal{I} \models^{\mathcal{L}}_k A$, and therefore $\mathcal{I} \cup \mathcal{I}_G \models^{\mathcal{L}}_k (A \wedge G)$. Thus, $\mathcal{I} \cup \mathcal{I}_G \models^{\mathcal{L}}_k F$, i.e. $\mathcal{I} \cup \mathcal{I}_G \in \mathit{Mod}_{\mathcal{L}}(F)$. Analogously, since $\mathcal{I} \models^{\mathcal{L}}_n B$, we have that $\mathcal{I} \cup \mathcal{I}_G \models^{\mathcal{L}}_n (B \wedge G)$. Therefore $\mathcal{I} \cup \mathcal{I}_G \models^{\mathcal{L}}_n F[A/B]$. Since $n > k$, we have $\mathcal{I} \cup \mathcal{I}_G \notin \mathit{Mod}_{\mathcal{L}}(F[A/B])$. The case $k = n$ is similar. □

In order to guarantee strong equivalence for every choice logic, both the optionality and the satisfaction degree of two formulas must be the same.

Observation 2 *Let $\mathcal{L}$ be an arbitrary choice logic. If both, $A \equiv^{\mathcal{L}}_d B$ and $\mathit{opt}_{\mathcal{L}}(F) = \mathit{opt}_{\mathcal{L}}(G)$, hold, then $A \equiv^{\mathcal{L}}_s B$.*

Clearly, the reverse of Observation 2 is not always true, as Theorem 1 shows. However, for some logics, for example QCL and QCCL, it can be shown that the reverse holds, i.e. two formulas are strongly equivalent if and only if they are degree equivalent and have the same optionality.

4 ASP Encoding

We will provide a system written in Clingo 5 with which arbitrary choice logics can be encoded easily. Given a formula as input, each answer set of the encoding (composed of Listings 2–6 below) reports an interpretation together with its satisfaction degree. If one wishes to encode a CL that is not implemented in this system yet, then it suffices to specify the semantics of the CL's choice connectives in Listing 5. All other functionalities, such as syntax and the semantics of the classical connectives are fixed and readily provided by the encoding. Based on this core encoding, other problems such as finding preferred models or checking for equivalence are then presented in Listings 7, 8, and 10.

Input formulas will be contained in the predicate *inputformula/1.* Classical connectives are represented by functions *neg/1*, *and/2*, as well as *or/2.* Choice connectives can be described with the function *pref/3.* The first argument of *pref/3* is a string that tells us which choice connective we are dealing with. For example, Listing 1 shows the encoding of $((a \overrightarrow{\times} b) \vee (\neg a \overrightarrow{\odot} c))$.

Listing 1. input.lp

```
inputformula(
        or(pref('qcl',a,b), pref('ccl',neg(a),c))
).
```

Encoding choice connectives as ternary functions allows for adding new choice connectives and specifying their semantics without worrying about the syntactic implications of doing so. To process the input formula syntactically, we dissect it into subformulas and atoms.

Listing 2. base.lp, Part 1

```
subformula(F,F) :- inputformula(F).
subformula(F,G) :- subformula(F,neg(G)).
subformula(F,G) :- subformula(F,and(G,_)).
subformula(F,G) :- subformula(F,and(_,G)).
subformula(F,G) :- subformula(F,or(G,_)).
subformula(F,G) :- subformula(F,or(_,G)).
subformula(F,G) :- subformula(F,pref(_,G,_)).
subformula(F,G) :- subformula(F,pref(_,_,G)).

-atom(F) :- subformula(_,F), F = neg(_).
-atom(F) :- subformula(_,F), F = and(_,_).
-atom(F) :- subformula(_,F), F = or(_,_).
-atom(F) :- subformula(_,F), F = pref(_,_,_).
atom(F) :- subformula(_,F), not -atom(F).
```

The above program will yield the fact atom(x) for every constant x that occurs in the input formula. We can use this to guess all relevant interpretations.

Listing 3. base.lp, Part 2

```
{in(F) : atom(F)}.
out(F) :- atom(F), not in(F).
```

As for semantics, we will compute the optionality and satisfaction degree of every subformula, and therefore also of the input formula. In Listing 4, we can see how this can be done for the classical connectives. Note that we use the constant #sup, which is built into Clingo, for ∞.

Listing 4. base.lp, Part 3

```
opt(F,1) :- subformula(_,F), atom(F).
opt(F,1) :- subformula(_,F), F = neg(_).
opt(F,X) :- subformula(_,F), F = and(G,H), opt(G,X), opt(H,Y),
        X >= Y.
opt(F,Y) :- subformula(_,F), F = and(G,H), opt(G,X), opt(H,Y),
        X < Y.
opt(F,X) :- subformula(_,F), F = or(G,H), opt(G,X), opt(H,Y),
        X >= Y.
opt(F,Y) :- subformula(_,F), F = or(G,H), opt(G,X), opt(H,Y),
        X < Y.
```

```

deg(F,1) :- subformula(_,F), atom(F), in(F).
deg(F,#sup) :- subformula(_,F), atom(F), out(F).
deg(F,1) :- subformula(_,F), F = neg(G), deg(G,#sup).
deg(F,#sup) :- subformula(_,F), F = neg(G), deg(G,K), K < #sup.
deg(F,X) :- subformula(_,F), F = and(G,H), deg(G,X), deg(H,Y),
        X >= Y.
deg(F,Y) :- subformula(_,F), F = and(G,H), deg(G,X), deg(H,Y),
        X < Y.
deg(F,X) :- subformula(_,F), F = or(G,H), deg(G,X), deg(H,Y),
        X < Y.
deg(F,Y) :- subformula(_,F), F = or(G,H), deg(G,X), deg(H,Y),
        X >= Y.
```

To encode the semantics of the choice connectives, we will use the predicates *optIn/3*, *optOut/4*, *degIn/5*, and *degOut/6*. Recall that $opt_{\mathcal{L}}(F \circ G)$ is a function over $opt_{\mathcal{L}}(F)$ and $opt_{\mathcal{L}}(G)$. This function can be encoded via *optIn/3* and *optOut/4*. The first argument in both of these predicates is the string identifying which choice connective we are dealing with. The second and third arguments are the optionalities of F and G respectively. The fourth argument in *optOut/4* is the result of the function call, i.e. the optionality of $F \circ G$. The function for the satisfaction degree of the choice connective can be encoded analogously, except that $deg_{\mathcal{L}}(\mathcal{I}, F \circ G)$ is a function over $opt_{\mathcal{L}}(F)$, $deg_{\mathcal{L}}(\mathcal{I}, F)$, $opt_{\mathcal{L}}(G)$, and $deg_{\mathcal{L}}(\mathcal{I}, G)$. Listing 5 shows the specification of QCCL (cf. Definition 12).

Listing 5. qccl.lp

```
% Definition of QCL connective

optOut('qcl',Opt1, Opt2, Z) :-
        optIn('qcl', Opt1, Opt2), Z = Opt1 + Opt2.

degOut('qcl', Opt1, Deg1, Opt2, Deg2, Deg1) :-
        degIn('qcl', Opt1, Deg1, Opt2, Deg2),
        Deg1 < #sup.
degOut('qcl', Opt1, Deg1, Opt2, Deg2, Deg2 + Opt1) :-
        degIn('qcl', Opt1, Deg1, Opt2, Deg2),
        Deg1 = #sup, Deg2 < #sup.
degOut('qcl', Opt1, Deg1, Opt2, Deg2, #sup) :-
        degIn('qcl', Opt1, Deg1, Opt2, Deg2),
        Deg1 = #sup, Deg2 = #sup.

% Definition of CCL connective

optOut('ccl',X, Y, Z) :- optIn('ccl', X, Y), Z = X + Y.

degOut('ccl', Opt1, Deg1, Opt2, Deg2, Deg2) :-
        degIn('ccl', Opt1, Deg1, Opt2, Deg2),
        Deg1 = 1.
degOut('ccl', Opt1, Deg1, Opt2, Deg2, Deg1 + Opt2) :-
```

```
        degIn('ccl', Opt1, Deg1, Opt2, Deg2),
        Deg1 > 1, Deg1 < #sup.
degOut('ccl', Opt1, Deg1, Opt2, Deg2, #sup) :-
        degIn('ccl', Opt1, Deg1, Opt2, Deg2),
        Deg1 = #sup.
```

We can use this encoding of the optionality and degree functions to compute the values of *opt/2* and *deg/2* for the choice connectives. This ensures that the custom defined choice connectives are integrated into the base system.

Listing 6. base.lp, Part 4

```
optIn(CL, X, Y) :- subformula(_,F), F = pref(CL,G,H),
        opt(G,X), opt(H,Y).
opt(F,Z) :- subformula(_,F), F = pref(CL,G,H),
        opt(G,X), opt(H,Y), optOut(CL, X, Y, Z).

degIn(CL, Opt1, Deg1, Opt2, Deg2) :- subformula(_,F),
        F = pref(CL,G,H), opt(G,Opt1), opt(H,Opt2),
        deg(G,Deg1), deg(H,Deg2).
deg(F,Z) :- subformula(_,F), F = pref(CL,G,H),
        opt(G,Opt1), opt(H,Opt2), deg(G,Deg1), deg(H,Deg2),
        degOut(CL, Opt1, Deg1, Opt2, Deg2, Z).
```

4.1 Computing Preferred Models

To compute all models of the input formula, one has to simply exclude those answer sets that result in an infinite satisfaction degree. Note that, for the sake of a cleaner output, we introduce a predicate *deg/1*, which holds the satisfaction degree of the input formula.

Listing 7. models.lp

```
deg(K) :- inputformula(F), deg(F,K).
:- deg(#sup).
#show in/1.
#show deg/1.
```

Execute the following bash command to find all models of a formula:

```
$ clingo base.lp qccl.lp input.lp models.lp 0
```

Preferred models of the input formula can be computed with Clingo's #minimize statement. Instead of minimizing over *deg/1*, we minimize over the newly introduced predicate *p/1*, which only holds finite satisfaction degrees. This is simply to avoid warning messages, since the #minimize statement is not designed to work with #sup.

Listing 8. pref_models.lp

```
deg(K) :- inputformula(F), deg(F,K).
```

```
:- deg(#sup).
p(K) :- deg(K), K < #sup.
#minimize {X:p(X)}.
#show in/1.
#show deg/1.
```

This bash command gives the preferred model of the given formula:

```
$ clingo --opt-mode=optN --quiet=1 base.lp qccl.lp input.lp
    pref_models.lp
```

4.2 Checking for Equivalence

Some problems, i.e. checking whether two formulas are equivalent, require more than one input formula. Recall, for example, that two QCL formulas are strongly equivalent if they are degree equivalent and have the same optionality (see Observation 2 and the remark following it). In what follows we discuss how to test strong equivalence in QCL using our encoding. We introduce the predicates *inputformula1/1* and *inputformula2/1*. In Listing 9, we see the encoding of $F_1 = (a \overrightarrow{\times} (b \overrightarrow{\times} c))$ and $F_2 = ((a \overrightarrow{\times} b) \overrightarrow{\times} c)$. Note that F_1 and F_2 are strongly equivalent, since ordered disjunction is associative.

Listing 9. input2.lp

```
inputformula1(
        pref('qcl',a,pref('qcl',b,c))
).
inputformula2(
        pref('qcl',pref('qcl',a,b),c)
).
```

Instead of directly checking whether F_1 and F_2 are strongly equivalent, we solve the complementary problem: If we find an interpretation (i.e. an answer set) where F_1 and F_2 are ascribed different satisfaction degrees, or show that the optionalities of F_1 and F_2 are not equal, then F_1 and F_2 are not strongly equivalent. See Listing 10 for how this can be implemented. The two input formulas are strongly equivalent if and only if the program is unsatisfiable.

Listing 10. strong_equiv.lp

```
inputformula(F) :- inputformula1(F).
inputformula(F) :- inputformula2(F).

same_degree :- inputformula1(F), inputformula2(G),
        deg(F,K), deg(G,L), K=L.
same_optionality :- inputformula1(F), inputformula2(G),
        opt(F,K), opt(G,L), K=L.
:- same_degree, same_optionality.

#show in/1.
```

Note that Listing 10 also includes the rules inputformula(F) :− inputformula1(F) and inputformula(F) :− inputformula2(F). If this were not the case, then *inputformula1/1* and *inputformula2/1* could not be processed correctly by the base program. The following program call has to be made to check whether two formulas are degree-equivalent and have the same optionality:

```
$ clingo base.lp qccl.lp input.lp strong_equiv.lp
```

To check directly for strong equivalence, instead of solving the complementary problem, one could employ the saturation technique described by Egly et al. [10]. This would require modifications to the base program, as default negation can not be used with this technique.

Of course, one could also check for degree-equivalence. This can be done analogously to Listing 10, except that we do not need to verify whether the two formulas have the same optionality or not.

5 Conclusion

In this paper, we introduced a framework for choice logics that encompasses both QCL and CCL. Furthermore, we showed how logics in this framework can be encoded via ASP, and how certain problems, such as finding preferred models for a given formula, or checking whether two formulas are strongly equivalent, can be solved using this encoding.

The encodings as provided in this paper are available under

https://github.com/mbernr/choice-logics-asp.

There we also provide specifications for further variants of choice logics.

The described ASP system can also be used in conjunction with other systems. As an alternative to encoding the choice connectives of a CL in pure ASP, the optionality- and degree functions could be described externally. For example, in Clingo 5, Python or Lua can be used to encode functions. Specifying the semantics of a CL in this way might be more convenient for some users. Furthermore, it is possible to make calls to Clingo 5 programs from other environments, such as Python. This allows for a system in which the business logic runs in Python, but when reasoning about preferences is needed, the ASP program can be employed.

A further possible field of study is to investigate the CL framework more closely, by looking at some of the properties of choice logics. Since the definition of a CL is not very restrictive, one could examine certain subgroups of choice logics that exhibit certain, possibly desirable, properties.

Lastly, instead of encoding choice logics within ASP, it would also be possible to extend ASP itself by choice connectives. This has already been done with ordered disjunction, resulting in Logic Programming with Ordered Disjunction (LPOD) [3, 6].

References

1. Benferhat, S., Sedki, K.: Two alternatives for handling preferences in qualitative choice logic. Fuzzy Sets Syst. 159(15), 1889–1912 (2008), https://doi.org/10.1016/j.fss.2008.02.014
2. Boudjelida, A., Benferhat, S.: Conjunctive choice logic. In: International Symposium on Artificial Intelligence and Mathematics, ISAIM 2016, Fort Lauderdale, Florida, USA, January 4-6, 2016 (2016), http://isaim2016.cs.virginia.edu/papers/ISAIM2016_Boudjelida_Benferhat.pdf
3. Brewka, G.: Answer sets and qualitative decision making. Synthese 146(1-2), 171–187 (2005), https://doi.org/10.1007/s11229-005-9084-7
4. Brewka, G., Benferhat, S., Berre, D.L.: Qualitative choice logic. Artif. Intell. 157(1-2), 203–237 (2004), https://doi.org/10.1016/j.artint.2004.04.006
5. Brewka, G., Delgrande, J.P., Romero, J., Schaub, T.: asprin: Customizing answer set preferences without a headache. In: Bonet, B., Koenig, S. (eds.) Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA. pp. 1467–1474. AAAI Press (2015), http://www.aaai.org/ocs/index.php/AAAI/AAAI15/paper/view/9535
6. Brewka, G., Niemelä, I., Syrjänen, T.: Logic programs with ordered disjunction. Comput. Intell. 20(2), 335–357 (2004), https://doi.org/10.1111/j.0824-7935.2004.00241.x
7. Charwat, G., Pfandler, A.: Democratix: A declarative approach to winner determination. In: Walsh, T. (ed.) Algorithmic Decision Theory - 4th International Conference, ADT 2015, Lexington, KY, USA, September 27-30, 2015, Proceedings. LNCS, vol. 9346, pp. 253–269. Springer (2015), https://doi.org/10.1007/978-3-319-23114-3_16
8. Costantini, S., Formisano, A., Petturiti, D.: Extending and implementing RASP. Fundam. Inform. 105(1-2), 1–33 (2010), https://doi.org/10.3233/FI-2010-356
9. Delgrande, J.P., Schaub, T., Tompits, H.: A framework for compiling preferences in logic programs. Theory Pract. Log. Program. 3(2), 129–187 (2003), https://doi.org/10.1017/S1471068402001539
10. Egly, U., Gaggl, S.A., Woltran, S.: Answer-set programming encodings for argumentation frameworks. Argument & Computation 1(2), 147–177 (2010), https://doi.org/10.1080/19462166.2010.486479
11. Faber, W., Truszczynski, M., Woltran, S.: Abstract preference frameworks - a unifying perspective on separability and strong equivalence. In: desJardins, M., Littman, M.L. (eds.) Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence, July 14-18, 2013, Bellevue, Washington, USA. pp. 297–303. AAAI Press (2013), http://www.aaai.org/ocs/index.php/AAAI/AAAI13/paper/view/6294
12. de Haan, R., Slavkovik, M.: Answer set programming for judgment aggregation. In: Kraus, S. (ed.) Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019. pp. 1668–1674. ijcai.org (2019), https://doi.org/10.24963/ijcai.2019/231
13. Pigozzi, G., Tsoukiàs, A., Viappiani, P.: Preferences in artificial intelligence. Ann. Math. Artif. Intell. 77(3-4), 361–401 (2016), https://doi.org/10.1007/s10472-015-9475-5