Abstract

We present CR-Prolog, an extension of A-Prolog with cr-rules and ordered disjunction. CR-rules can be used to formalize various types of common-sense knowledge and reasoning, that, to the best of our knowledge, have no formalization in A-Prolog. The use of ordered disjunction often allows for a very concise, easy to read, representation of knowledge. We also show how CR-Prolog can be used to represent preferences intended both as binding preferences, and as desires.

Introduction

In recent years, A-Prolog – the language of logic programs with the answer set semantics (Gelfond & Lifschitz 1991) – was shown to be a useful tool for knowledge representation and reasoning (Gelfond 2002). The language is expressive and has a well understood methodology of representing defaults, causal properties of actions and fluents, various types of incompleteness, etc. The development of efficient computational systems (Cholewinski, Marek, & Truszczynski 1996; Niemela, Simons, & Soininen 2002; Calimeri et al. 2002; Lin & Zhao 2002; Pontelli, Balduccini, & Bermudez 2003) has allowed the use of A-Prolog for a diverse collection of applications. Some of the applications include: a decision support system for space shuttle flight controllers. The system is planned to be used by flight controllers to find plans for the operation of the Reactive Control System of the space shuttle, as well as checking correctness of existing plans (Nogueira et al. 2001; Nogueira 2003); detection of deadlocks using a reduction of the problem to the computation of stable models of logic programs (Heljanko 1999). Other important applications are in planning, product configuration, bounded model checking, wire routing and modeling in hybrid systems etc.

It seems however that A-Prolog lacks the ability to gracefully perform the reasoning needed for certain types of conflict resolution, e.g. for finding the best explanations of unexpected observations. To solve the problem, in (Balduccini & Gelfond 2003b) the authors introduced CR-Prolog – an extension of A-Prolog by consistency-restoring rules (cr-rules) with preferences.

In this paper we present CR-Prolog, a variant of CR-Prolog with an improved semantics, and allowing ordered disjunction (Brewka 2002; Brewka, Niemela, & Syrjanen 2002) in the head of both regular rules and consistency-restoring rules. The new semantics yields intuitive conclusions in cases when CR-Prolog would give unintuitive results. The use of ordered disjunction, when the preference order on a set of alternatives is total, allows for a more concise, easier to read, representation of knowledge. The flexibility of the preference relation in CR-Prolog is such that meta-preferences from LPOD (Brewka, Niemela, & Syrjanen 2002) can be encoded in CR-Prolog using directly its preference relation, rather than requiring the definition of a new type of preference. We show how CR-Prolog can be used to represent preferences intended both as binding preferences (like in CR-Prolog), and as desires (like in LPOD).

The paper is structured as follows. We start with the syntax and semantics of CR-Prolog. Next, we compare the new language with CR-Prolog and LPOD, and show how the new language can be used to represent complex knowledge and to perform fairly sophisticated reasoning tasks. Finally, we summarize the paper and draw conclusions.

Syntax and Semantics

Let \( \Sigma \) be a signature containing symbols for constants, variables, functions, and predicates (denoted by \( const(\Sigma), var(\Sigma), func(\Sigma) \) and \( pred(\Sigma) \), respectively). Terms, atoms, and literals are defined as usual. Literals and terms not containing variables are called ground. The sets of ground terms, atoms and literals over \( \Sigma \) will be denoted by \( terms(\Sigma), atoms(\Sigma), \) and \( lit(\Sigma) \).

Definition 1 A head expression is either an epistemic disjunction of literals \( (h_1 \vee h_2 \vee \ldots \vee h_k) \), with \( k \geq 0 \) or an ordered disjunction of literals \( (h_1 \times h_2 \times \ldots \times h_k) \), with \( k > 1 \).

Definition 2 A regular rule of CR-Prolog is a statement of the form:

\[
\mathcal{H} \leftarrow l_1, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n
\]

(1)

where \( \mathcal{H} \) is a head expression, \( l_1, \ldots, l_n \) are literals, and \( r \) is a term representing the name of the rule. If \( \mathcal{H} \) is an epistemic disjunction, the intuitive reading of the rule is as usual. If \( \mathcal{H} \) is an ordered disjunction, the intuitive meaning of the rule is (Brewka, Niemela, & Syrjanen 2002): if the body of the rule is satisfied by the agent’s beliefs, then the agent must believe...
the first (leftmost) element of \( \mathcal{H} \), if possible; otherwise it must believe the second element, if possible; \ldots otherwise, it must believe the last element of \( \mathcal{H} \).

For example, program

\[
\begin{align*}
r_1 &: p \lor q \leftarrow \neg r. \\
r_2 &: s.
\end{align*}
\]

yields two possible conclusions: \{\( p \}\) and \{\( q \}\}. On the other hand, program

\[
\begin{align*}
r_1 &: p \times q \leftarrow \neg r. \\
r_2 &: s \leftarrow \neg p, \neg q.
\end{align*}
\]

forces the agent to believe \( p \), and program

\[
\begin{align*}
r_1 &: p \times q \leftarrow \neg r. \\
r_2 &: s \leftarrow \neg s, p, \neg r.
\end{align*}
\]

forces the agent to believe \( q \) (since believing \( p \) is made impossible by the second rule).

**Definition 3** A cr-rule is a statement of the form:

\[
r : \ H \leftarrow l_1, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n \tag{2}
\]

where \( r \) is the name of the rule, \( \mathcal{H} \) is a head expression, and \( l_1, \ldots, l_n \) are literals. The rule says that if \( l_1, \ldots, l_m \) belong to a set of agent’s beliefs and none of \( l_{m+1}, \ldots, l_n \) belongs to it then the agent “may possibly” believe one of the elements of the head expression. This possibility is used only if the agent has no way to obtain a consistent set of beliefs using regular rules only. If \( \mathcal{H} \) is an ordered disjunction, the preference order that the agent uses to select an element from the head expression goes from left to right, as for regular rules. If \( \mathcal{H} \) is an epistemic disjunction, then all the elements are equally preferable.

For example, program

\[
\begin{align*}
r_1 &: p \lor q \leftarrow \neg r. \\
r_2 &: s.
\end{align*}
\]

only forces the agent to believe \( s \) (the cr-rule need not be applied, since the program containing only the second rule is consistent). On the other hand, program

\[
\begin{align*}
r_1 &: p \lor q \leftarrow \neg r. \\
r_2 &: s. \\
r_3 &: s \leftarrow \neg p, \neg q.
\end{align*}
\]

forces the agent to believe either \{\( s, p \)\} or \{\( s, q \)\}. If finally we want the agent to prefer conclusion \( p \) to conclusion \( q \) when possible, we write

\[
\begin{align*}
r_1 &: p \times q \leftarrow \neg r. \\
r_2 &: s. \\
r_3 &: \neg s, p, \neg q.
\end{align*}
\]

which yields a unique set of beliefs, \{\( s, p \)\}. Notice though that adding new information to the above program, for example a new rule \( \leftarrow p \), forces the agent to retract the previous conclusions, and believe \{\( s, q \)\}. We will use the term rule to denote both regular rules and cr-rules. As usual, non-ground rules are intended as schemata for their ground counterparts.

**Definition 4** Preferences between cr-rules are expressed by atoms of the form \( \text{prefer}(r_1, r_2) \). The intuitive reading of the atom is “do not consider belief sets obtained using \( r_2 \) unless you have excluded the existence of belief sets obtained using \( r_1 \)”.

We call this type of preference binding. If all preferences in a program are expressed as facts, we say that the program employs static preferences. Otherwise, preferences are dynamic.

**Definition 5** A CR-Prolog2 program, \( \Pi \), is a pair \((\Sigma, R)\) consisting of signature \( \Sigma \) and a set \( R \) of rules of form (1) or (2). We require that \( \text{func}(\Sigma) \) does not contain choice, and that \( \text{pred}(\Sigma) \) contains \text{prefer} and does not contain \text{appl}, \text{fired}, and \text{is_preferred}. Signature \( \Sigma \) is denoted by \( \text{sig}(\Pi) \); \( \text{const}(\Pi) \), \( \text{func}(\Pi) \), \( \text{pred}(\Pi) \), \( \text{atoms}(\Pi) \) and \( \text{lit}(\Pi) \) are shorthands for \( \text{const}(\text{sig}(\Pi)) \), \( \text{func}(\text{sig}(\Pi)) \), \( \text{pred}(\text{sig}(\Pi)) \), \( \text{atoms}(\text{sig}(\Pi)) \) and \( \text{lit}(\text{sig}(\Pi)) \), respectively.

Let \( P \) be a set of predicate symbols from \( \Sigma \). By \( \text{atoms}(\Pi, P) \) we denote the set of all atoms from \( \text{atoms}(\Pi) \) formed by predicate symbols from \( P \). (Whenever possible we drop the first argument and simply write \( \text{atoms}(P) \)). The set of rules of \( \Pi \) is denoted by \( \text{rules}(\Pi) \). If \( \rho \) is a rule of \( \Pi \) then \( \text{head}(\rho) = \mathcal{H} \), and \( \text{body}(\rho) = \{l_1, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n\} \).

Programs of CR-Prolog2 are closely related to abductive logic programs.

**Definition 6** An abductive logic program (Kakas & Mancarella 1990; Gelfond 1991) is a pair \((\Pi, A)\), where \( \Pi \) is a program of A-Prolog and \( A \) is a set of atoms, called abducibles.

The semantics of an abductive program is given by the notion of generalized answer set – an answer set \( M(\Delta) \) of \( \Pi \cup \Delta \) where \( \Delta \subseteq A \); \( M(\Delta_1) < M(\Delta_2) \) if \( \Delta_1 \subset \Delta_2 \). We refer to an answer set as minimal if it is minimal with respect to this ordering.

The semantics of CR-Prolog2 is based on a transformation \( hr(\Pi) \) of its programs into abductive programs.

**Definition 7** The hard reduct \( hr(\Pi) = \langle H_{\Pi}, \text{atoms}(H_{\Pi}, \{\text{appl}_{cr}, \text{appl}_{x}\}) \rangle \) is defined as follows:

1. let \( N \) be a set of new constant symbols, such that every element of \( N \) is uniquely associated with an atom from \( \text{sig}(\Pi) \). We will denote the constant associated with atom \( a \) by \( n_a \);
2. \( \text{sig}(H_{\Pi}) \) extends \( \text{sig}(\Pi) \) so that: \( \text{const}(H_{\Pi}) = \text{const}(\Pi) \cup N \), \( \text{func}(H_{\Pi}) = \text{func}(\Pi) \cup \{\text{choice}\} \), and \( \text{pred}(H_{\Pi}) = \text{pred}(\Pi) \cup \{\text{appl}_{cr}, \text{appl}_{x}, \text{fired}, \text{is_preferred}, \text{bodytru}\} \);
3. let \( R_{\Pi} \) be set of rules obtained from \( H_{\Pi} \) by replacing every cr-rule, \( \rho \), with:

\[
r : \text{head}(\rho) \leftarrow \text{body}(\rho), \text{appl}_{cr}(r).
\]

\[
\text{bodytru}(r) \leftarrow \text{body}(\rho).
\]

where \( r \) is the name of \( \rho \). Notice that \( R_{\Pi} \) contains only regular rules;
4. the set of rules of \( H_{\Pi} \) is obtained from \( R_{\Pi} \) by replacing every rule, \( \rho \), such that \( \text{head}(\rho) = h_1 \times h_2 \times \ldots \times h_k \), with the following rules: (\( r \) is the name of rule \( \rho \))
Let us compute the hard reduct of the following program:

\[ R \]

\[ \mathcal{C}^{\prime} \triangleq \{ \text{hr} \} \]

\[ \Pi_1 \]

\[ \Pi_1 \]

\[ H_{\Pi_1} = H_{\Pi_1}^1 \cup \Pi_p, \text{ where } H_{\Pi_1}^1 \text{ is:} \]

\[ \Pi_1 \]

\[ \Pi_1 \]

\[ C_1 = \{ \text{preference}(r_6, r_4, r_5, r_6) \} \]

\[ C_2 = \{ \text{preference}(r_7, r_5, r_6) \} \]

\[ C_3 = \{ \text{preference}(r_6, r_5, r_7, t, s) \} \]

\[ C_4 = \{ \text{preference}(r_6, r_5, r_7, t, s) \} \]

\[ C_5 = \{ \text{preference}(r_6, r_5, r_7, t, s) \} \]

Intuitively, not all the generalized answer sets appear equally appealing w.r.t. the preferences expressed in the program. The following definition formalizes this idea.

**Definition 8** Let \( \Pi \) be a CR-Prolog program, and \( C, C' \) be generalized answer sets of \( \text{hr}(\Pi) \). We say that \( C \) dominates \( C' \) (and write \( C \succ C' \)) if:

\[ \exists \text{app}(r_1) \in C, \text{app}(r_2) \in C' \text{ s.t. } \text{is_preferred}(r_1, r_2) \in C \cap C'. \]
weight lifting and running, or combine swimming and ball playing, but each option is at the same time better and worse than the other according to different points of view. If preferences are ignored, both combinations are acceptable.

Statement (4) can be encoded by the following program, \( \Pi_4 \):\

\[
\begin{align*}
& r_r : run \leftarrow. \\
& r_s : swim \leftarrow. \\
& r_p : play \leftarrow. \\
& r_w : lift \leftarrow.
\end{align*}
\]

\[
\begin{align*}
& \text{full body exercise} \leftarrow \text{swim, play}.
\end{align*}
\]

\[
\begin{align*}
& \text{prefer}(r_r, r_s) \leftarrow \text{not ignore prefs}.
\end{align*}
\]

The generalized answer sets of \( hr(\Pi_4) \) are: (we show only the atoms formed by run, swim, play, lift, ignore prefs, and prefer)

\[
\begin{align*}
& G_1 : \{lift, run, prefer(r_r, r_s), prefer(r_p, r_w)\} \\
& G_2 : \{swim, play, lift, prefer(r_r, r_s), prefer(r_p, r_w)\} \\
& G_3 : \{ignore prefs, lift, run\} \\
& G_4 : \{ignore prefs, swim, play\} \\
& G_5 : \{ignore prefs, run, swim\} \\
& G_6 : \{ignore prefs, lift, run, play\} \\
& G_7 : \{ignore prefs, swim, play, lift\} \\
& G_8 : \{ignore prefs, lift, run, swim, play\}
\end{align*}
\]

Under the semantics of CR-Prolog, \( G_1 \) and \( G_2 \) are the only minimal generalized answer sets. Since \( G_1 \gg G_2 \) and \( G_2 \gg G_1 \), \( \Pi_4 \) has no answer sets.

Under the semantics of CR-Prolog2, \( G_1 \) and \( G_2 \) dominate each other, which leaves only \( G_3 \) to \( G_8 \) as candidate answer sets. Since \( G_3 \) and \( G_4 \) are both minimal w.r.t. the admissible present in each candidate answer set, they are both answer sets of \( \Pi_4 \), like intuition suggested.

The reason for this difference is that, in the semantics of CR-Prolog, set-theoretic minimization occurs before the comparison of belief sets w.r.t. the preferences. In CR-Prolog2, on the other hand, generalized answer sets are first of all compared w.r.t. the preference relation, and only later set-theoretic minimization is applied. In our opinion, giving higher relevance to the preference relation is a better choice (as confirmed by the previous example), since preferences are explicitly given by the programmer.

**Comparison with LPOD**

In (Brewka 2002), the author introduces logic programs with ordered disjunction (LPOD). The semantics of LPOD is based on the notion of preferred answer sets. In a later paper (Brewka, Niemela, & Syrjanen 2002), the authors introduce the notion of Pareto-preference between belief sets and show that this criterion gives more intuitive results that the other criteria described in (Brewka 2002; Brewka, Niemela, & Syrjanen 2002). In this section, we compare LPOD (under Pareto-preference) and CR-Prolog2.

Consider program \( \Pi_{s1} \) from (Brewka, Niemela, & Syrjanen 2002):

\[
\begin{align*}
& \text{% Have ice cream, if possible; otherwise, have tea.} \\
& r_1 : \text{ice cream} \times \text{cake}. \\
& \text{% Have coffee if possible, otherwise, have tea.} \\
& r_2 : \text{coffee} \times \text{tea}. \\
& \text{% It is impossible to have ice cream and cake together.} \\
& \text{← ice cream, coffee.}
\end{align*}
\]

The preferred answer sets of \( \Pi_{s1} \) in LPOD are:

\[
\{\text{ice cream, tea}\} \text{ and } \{\text{cake, coffee}\}.
\]

There are no answer sets of \( \Pi_{s1} \) according to the semantics of CR-Prolog2. The difference between the two semantics depends on the fact that Pareto optimality was introduced to satisfy desires and it looks for a set of solutions that satisfy as many desires as possible. On the other hand, our preference criterion corresponds to a stricter reading of the preferences.

In order to make it easy to understand the relationship between the two types of preference, we restate the Pareto criterion in the context of CR-Prolog2.

**Definition 11** Let \( \Pi \) be a CR-Prolog2 program, and \( C, C' \) be generalized answer sets of \( hr(\Pi) \). We say that \( C \) Pareto-dominates \( C' \) (written as \( C \gg_p C' \)), if

\[
\exists \text{ appl}(r_1) \in C, \text{ appl}(r_2) \in C' \text{ s.t.} \\
\text{is preferred}(r_1, r_2) \in C \cap C', \text{ and}
\]

\[
\neg \exists \text{ appl}(r_3) \in C, \text{ appl}(r_4) \in C' \text{ s.t.} \\
\text{is preferred}(r_4, r_3) \in C \cap C'.
\]

**Definition 12** Let \( \Pi \) be a CR-Prolog2 program, \( C \) be a generalized answer set of \( hr(\Pi) \). We say that \( C \) is a Pareto-candidate answer set of \( \Pi \) if there exists no generalized answer set, \( C' \), of \( hr(\Pi) \) such that \( C' \gg_p C \).

(Notice that Pareto-domination is essentially a restatement of the Pareto criterion, in the context of CR-Prolog2. Also, Pareto-candidate answer sets essentially correspond to preferred answer sets.)

Now, to see the difference between Definitions 8 and 11, consider a program, \( \Pi \), and generalized answer sets, \( C_1 \) and \( C_2 \), such that \( C_1 \) dominates \( C_2 \) and vice-versa. Notice that they do not Pareto-dominate each other. Under our semantics, none of them is a candidate answer set of \( \Pi \). However, using Pareto-domination, \( C_1 \) and \( C_2 \) are incomparable and thus, both are eligible as Pareto-candidate answer sets (whether they really are Pareto-candidate answer sets, depends on the other generalized answer sets).

In a sense, Definition 8 enforces a clearer representation of knowledge and of preferences. However, that does not rule
out the possibility of representing desires in CR-Prolog.

The defeasible nature of desires is represented by means of

cr-rules. For example, the program $\Pi_{a_1}$ can be rewritten as

follows, $\Pi_{a_2}$:

$$
\begin{align*}
\{ & r_{1a} : ice\_cream \leftarrow . \\
 & r_{1b} : cake \leftarrow . \\
 & r_{2a} : coffee \leftarrow . \\
 & r_{2b} : tea \leftarrow . \\
 & r_{3} : prefer(r_{1a}, r_{1b}) \leftarrow \neg\ prefer(r_{1a}, r_{1b}). \\
 & r_{4} : prefer(r_{2a}, r_{2b}) \leftarrow \neg\ prefer(r_{2a}, r_{2b}). \\
 & r_{5} : \neg prefer(r_{1a}, r_{1b}) \leftarrow . \\
 & r_{6} : \neg prefer(r_{2a}, r_{2b}) \leftarrow . \\
\end{align*}
$$

solid \leftarrow ice\_cream.

solid \leftarrow cake.

liquid \leftarrow coffee.

liquid \leftarrow tea.

\neg solid. \\
\neg liquid. \\
\leftarrow liquid, coffee.

In $\Pi_{a_2}$, the desire to have ice\_cream over cake is represented by:

(1) a cr-rule, $r_5$, that says that “the agent may possibly give up his preference for ice\_cream over cake”;

(2) a default, $r_6$, saying that “the agent normally prefers ice\_cream over cake.”

In a similar way, we represent the
desire for coffee over tea. The answer sets of the above

program are (we show only the atoms from cr-rules $r_{1a}$-$r_{2b}$)

$\{ ice\_cream, tea \}$ and $\{ cake, coffee \}$, which correspond to (5).

There are also some programs for which the semantics of

LPOD seems to yield unintuitive results, while the semantics of CR-Prolog gives results that agree with the intuition.

Consider the following example:

Example 1 “A television show conducts a game where the first

winner is offered a prize of $200,000 and the second winner is offered a prize of $100,000. John wants to play, if possible.

Otherwise he will give up. If he plays he wants to

gain $200,000 if possible; otherwise, $100,000. He is told

that he cannot win the first prize.”

$$
\begin{align*}
% John wants to play, if possible. Otherwise, give up. \\
% If he plays, he prefers gaining $200,000 over $100,000. \\
% He is told that, he cannot gain $200,000. \\
% turn, unless there is a failure with some equipment responsible for starting the engine. There can be electrical failures, such as battery down or fuse burnt; or mechanical failures, such as clutch sensor stuck or belt loose. In general, the electrical failures are more likely than the mechanical failures. Among the electrical failures, battery down is more likely than fuse burnt. Among the mechanical failures, clutch sensor stick is more likely than belt loose.”

The knowledge contained in this story can be represented by

the following action description, $\Pi_{s}$:

$$
\begin{align*}
% normally, a car’s engine starts when the start key is turned, unless there is a failure with some equipment responsible for starting the engine. \\
% battery being down causes failure in start equipment. \\
% fuse being burnt causes failure in start equipment. \\
% clutch sensor stuck causes failure in start equipment. \\
% belt being loose causes failure in start equipment. \\
% sometimes, battery is down or fuse is burnt, \\
% the former being more likely than the latter \\
% r_{elec}(T) : h(battery\_down, T) \times h(fuse\_burnt, T) \leftarrow . \\
% sometimes, clutch sensor is stuck or \\
% belt is loose, the former being more \\
% likely than the latter \\
% r_{mech}(T) : h(sensor\_stuck, T) \times h(belt\_loose, T) \leftarrow . \\
% electrical failures are more likely \\
% than mechanical failures
\end{align*}
$$

We thank Richard Watson for pointing out the behavior of

LPOD in this case.
% INERTIA
\[ h(F, T + 1) \leftarrow h(F, T), \neg h(F, T + 1). \]
\[ \neg h(F, T + 1) \leftarrow h(F, T), \not h(F, T + 1). \]

% REALITY CHECKS
\[ \left\{ \begin{array}{l}
\text{\texttt{obs}}(F, T), \not h(F, T).
\text{\texttt{obs}}(\neg F, T), \not \neg h(F, T).
\end{array} \right. \]

% AUXILIARY AXIOMS
\[ \begin{array}{l}
a(A, T) \leftarrow \text{\texttt{hpd}}(A, T).
\text{\texttt{obs}}(F, 0) \leftarrow \text{\texttt{obs}}(F, 0).
\neg h(F, 0) \leftarrow \text{\texttt{obs}}(\neg F, 0).
\end{array} \]

Let us add the following history to the action description:
\[ \left\{ \begin{array}{l}
\text{\texttt{obs}}(\neg \text{\texttt{engine}}_\text{\texttt{gm}}, 0), \% \text{CWA on initial observations}
\text{\texttt{hp}}(\text{\texttt{turn}}_\text{\texttt{key}}, 0), \% \text{\texttt{obs}}(\neg F, 0) \not \text{\texttt{obs}}(F, 0).
\text{\texttt{obs}}(\neg \text{\texttt{engine}}_\text{\texttt{gm}}, 1).
\end{array} \right. \]

The observation at time 1 is unexpected, and causes the program to be inconsistent (because of the reality checks), unless at least one cr-rule is applied; because of the preference encoded by \( r_p(T) \), the preferred way to restore consistency is by applying \( r_{elec}(0) \); of the two options contained in the head of \( r_{elec}(0) \), \( h(\text{\texttt{battery}}_\text{\texttt{down}}, 0) \) is the preferred one. Clearly, adding the belief \( h(\text{\texttt{battery}}_\text{\texttt{down}}, 0) \) restores consistency of the program, and explains the unexpected observation.

It is worth noticing that the statements encoded by rules \( r_{elec}(T), r_{mech}(T) \) and \( r_p(T) \) of \( \Pi_c \) can be also represented without ordered disjunction. The three rules are replaced by:
\[ \begin{array}{l}
r_{\text{\texttt{batt}}}(T) : \ h(\text{\texttt{battery}}_\text{\texttt{down}}, T) \leftarrow h_{\text{\texttt{elec}}}(T).

r_{\text{\texttt{fuse}}}(T) : \ h(\text{\texttt{fuse}}_\text{\texttt{burnt}}, T) \leftarrow h_{\text{\texttt{elec}}}(T).

r_{\text{\texttt{sens}}}(T) : \ h(\text{\texttt{sensor}}_\text{\texttt{stuck}}, T) \leftarrow h_{\text{\texttt{mech}}}(T).

r_{\text{\texttt{belt}}}(T) : \ h(\text{\texttt{belt}}_\text{\texttt{loose}}, T) \leftarrow h_{\text{\texttt{mech}}}(T).

\text{\texttt{pref}}(r_{\text{\texttt{batt}}}(T), r_{\text{\texttt{fuse}}}(T)).
\text{\texttt{pref}}(r_{\text{\texttt{sens}}}(T), r_{\text{\texttt{belt}}}(T)).
\end{array} \]

The new program has (essentially) the same answer sets as the previous one. This shows that the rules with ordered disjunction allow for a more concise and elegant representation of knowledge.

Conclusions
In this paper, we extended CR-Prolog by ordered disjunction and an improved semantics, gave the semantics of the new language, and demonstrated how it differs from CR-Prolog and LPOD. We also showed how CR-Prolog can be used to formalize various types of common-sense knowledge and reasoning. We could not find natural A-Prolog formalizations for some of the examples in the paper, and formalizations in CR-Prolog were often less elegant and concise (besides giving sometimes unintuitive results). In comparison with CR-Prolog, we believe that the new features of CR-Prolog\(^2\) make it possible to write formalizations that are more natural, and reasonably elaboration tolerant. In comparison with LPOD, CR-Prolog\(^2\) appears more expressive (because of the availability of cr-rules and epistemic disjunction), and, in some cases, yields more intuitive results than LPOD.

References


