CR-Prolog: Logic Programs with Consistency-Restoring Rules

Marcello Balduccini and Michael Gelfond

August, 21 2003
Talk Outline

⇒ Agent architecture

• Typical scenarios
• Representing the agent’s knowledge
• The agent’s behavior and reasoning
• The use of preferences
• On-going research
Agent Architecture

**Observe-think-act loop**

1. observe the world;
2. interpret the observations;
3. select a goal;
4. plan;
5. execute part of the plan.

In the loop, simple procedural code fragments connect the various reasoning tasks, which are implemented in (a suitable extension of) A-Prolog.
System Description

The KRAgent is given knowledge about:

- the domain

![Circuit Diagram](image)

- the agent actions:
  - close $s_1$

- the exogenous actions:
  - power surge: damages the relay; also damages the bulb if not protected
  - bulb blow-up
Talk Outline

• Agent architecture

⇒ Typical scenarios
  • Representing the agent’s knowledge
  • The agent’s behavior and reasoning
  • The use of preferences
  • On-going research
A Scenario

![Diagram of a circuit with components labeled as s1, s2, r, b, and a switch.]

1. **Observations:** $s_1$ open; $s_2$ open; $b$ off; $b$ protected; components are ok.

3. **Select Goal:** bulb $b$ on.

4. **Plan:** close $s_1$.

5. **Execute Action:** close $s_1$. 
A Scenario

1. Observations: bulb $b$ is off.

2. Interpret Observations: the observation contradicts the expected effect of “close $s_1$”.

Possible explanations: either power surge or bulb blow-up occurred, unobserved, in the past.

Our goal is to formalize this and similar scenarios.
Talk Outline

• Agent architecture
• Typical scenarios
⇒ Representing the agent’s knowledge
• The agent’s behavior and reasoning
• The use of preferences
• On-going research
Domain Signature

Fluents:

\[ \text{closed}(SW) \] – switch \( SW \) is closed
\[ \text{prot}(B) \] – bulb \( B \) is protected from surges
\[ \text{on}(B) \] – bulb \( B \) is on
\[ \text{ab}(C) \] – component \( C \) is malfunctioning

Agent Actions:

\[ \text{close}(SW) \] – close switch \( SW \)

Exogenous Actions:

\[ \text{blow\_up}(B) \] – bulb \( B \) blows up
\[ \text{surge} \] – power surge
Causal Relations

Causal relations are described in the system description, $\Pi_d$.

State Constraints:

\[
\begin{align*}
  h(\text{closed}(s_2), T) & \leftarrow h(\text{closed}(s_1), T), \\
  & \quad h(\neg \text{ab}(r), T).
\end{align*}
\]

\[
\begin{align*}
  h(\text{on}(b), T) & \leftarrow h(\text{closed}(s_2), T), \\
  & \quad h(\neg \text{ab}(b), T).
\end{align*}
\]
Causal Relations

Dynamic Laws:

\[ h(\text{closed}(s_1), T + 1) \leftarrow o(\text{close}(s_1), T). \]

\[ h(ab(B), T + 1) \leftarrow o(\text{blow\_up}(B), T). \]

Inertia Axiom:

\[ h(L, T + 1) \leftarrow h(L, T), \]
\[ \text{not } h(\overline{L}, T + 1). \]
The Agent’s Observations

Observations are represented by statements:

\[ \text{obs}(L, T) \rightarrow L \text{ was observed at } T \]
\[ \text{hpd}(A, T) \rightarrow A \text{ happened at } T \]

- \text{obs} and \text{hpd} describe observations (which we assume always correct).
- \( h \) and \( o \) are agent’s predictions.
- They are linked, in \( \Pi_d \), by axioms:

\[ o(A, T) \leftarrow \text{hpd}(A, T). \]
\[ h(L, 0) \leftarrow \text{obs}(L, 0). \]

and by the \textbf{Reality Checks}:

\[ \leftarrow \text{obs}(L, T), h(\overline{L}, T). \]
Talk Outline

- Agent architecture
- Typical scenarios
- Representing the agent’s knowledge

⇒ The agent’s behavior and reasoning
- The use of preferences
- On-going research
KR.Agent in Action

1. Observations:

\[ \text{obs}(\neg \text{on}(b), 0), \text{obs}(\text{prot}(b), 0), \text{obs}(\neg \text{closed}(s_1), 0), \ldots \]

3. Select goal: \( \text{on}(b) \).

4. Plan: \ldots
Planning Algorithm

Planning is reduced to finding an answer set of \( \Pi_d \cup O \cup PM \) where \( O \) consists of the agent’s observations and \( PM \) is a Planning Module with goal \( G \), current time-step \( n \), and plan length \( l \) as parameters.

A simple Planning Module for our agent may consists of rules:

\[
\begin{align*}
o(A, T) \text{ or } \neg o(A, T) & \leftarrow n \leq T < n + l, \\
& \quad \text{agent}_\text{act}(A). \\
goal & \leftarrow h(on(b), T). \\
& \leftarrow \text{not} \ goal.
\end{align*}
\]
KRAgent in Action

1. Observations:
   
   \( \text{obs}(\neg \text{on}(b), 0), \text{obs}(\text{prot}(b), 0), \text{obs}(\neg \text{closed}(s_1), 0), \ldots) \)

3. Select goal: \( \text{on}(b) \).

4. Plan with \( l = 1 \). The KRAgent’s plan is:
   
   \( o(\text{close}(s_1), 0) \)

5. Perform Action: \( \text{close}(s_1) \). The KRAgent also records its execution with a statement
   
   \( \text{hpd}(\text{close}(s_1), 0) \).

Note that \( \Pi_d \cup O \models h(\text{on}(b), 1) \)

1. Observations: \( \text{obs}(\neg \text{on}(b), 1) \).
Interpreting Observations

- The new observation contradicts the agent’s expectation. This will be detected by the Reality Checks. Now $\Pi_d \cup O$ is inconsistent.
- The problem is caused by the agent’s assumption that closing the switch was the only action which occurred at moment 0.
- To explain the inconsistency the agent will assume that some exogenous actions did occur in the past. In our case this may lead to three possible explanations:

\[
\begin{align*}
\{ & o(blow\_up(b), 0) \\
\{ & o(surge, 0) \\
\{ & o(blow\_up(b), 0), o(surge, 0) \}
\end{align*}
\]
Interpreting Observations

[Balduccini & Gelfond, 2003]: explanations can be found by algorithms similar to planning.

This leaves the agent with the problem of selecting best explanations, which proved to be more difficult to address.

• Suppose the agent knows that bulbs may blow up and surges may happen, but both events are rare.
⇒ The agent may justifiably prefer the first two explanations.

• If in addition the agent knows that blow-ups are more frequent than surges, it may prefer the first explanation.

(Of course the selected explanation(s) will be further tested by the agent).
Expanding the Language

To model this type of reasoning we expand A-Prolog by **consistency-restoring rules** (cr-rules) with preferences.

A cr-rule, \( r \):

\[
\begin{align*}
 r : & \quad l \leftarrow^+ \\
\end{align*}
\]

says

“\( l \) may be assumed to be true if such assumption helps to restore consistency of the agent’s beliefs (but this should happen rarely).”

Our agent for instance may have two such rules:

\[
\begin{align*}
 r_1(T) : & \quad o(blow\_up(B),T) \leftarrow^+ . \\
 r_2(T) : & \quad o(surge,T) \leftarrow .
\end{align*}
\]
CR-Prolog

The new language is called CR-Prolog. In addition to standard rules of A-Prolog it allows cr-rules of the form:

\[ r : h_1 \lor \ldots \lor h_k \leftarrow^+ l_1, \ldots, l_m, \text{ not } l_{m+1}, \ldots, \text{ not } l_n \]

and the preference relation between them

\[ \text{prefer}(r_1, r_2) \]

which says

“Explanations based on \( r_1 \) are preferred to those based on \( r_2 \).”

The definition of answer set is expanded to capture this intuition.
CR-Prolog

Example

\[ \Pi_0 : \begin{cases} 
  a \leftarrow \text{not} \ b. \\
  r_1 : b \leftarrow. 
\end{cases} \]

- \( \Pi_0 \) has an answer set \{a\}, computed \textbf{without} the use of cr-rule \( r_1 \).

Example

\[ \Pi'_0 : \begin{cases} 
  a \leftarrow \text{not} \ b. \\
  \neg a. \\
  r_1 : b \leftarrow. 
\end{cases} \]

- If \( r_1 \) is not used, \( \Pi'_0 \) is \textbf{inconsistent}.
- The application of \( r_1 \) restores consistency, and leads to the answer set \{\( \neg a \), \( b \)\}.
Interpreting Observations

• Suppose now that the agent’s knowledge $\Pi_d$ contains rules $r_1(T)$ and $r_2(T)$.

⇒ Computing explanations of the unexpected observation is reduced to finding the answer sets of the CR-Prolog program $\Pi_d \cup O$.

• It can be shown that the algorithm works in general.
KRAgent in Action (cont.)

5. Perform Action: \( \text{close}(s_1) \).

1. Observations: \( \text{obs}(\neg \text{on}(b), 1) \).

2. Interpret Observations: the explanations of observations are:

\[
\begin{align*}
\{ o(\text{blow\_up}(b), 0) \} \\
\{ o(\text{surge}, 0) \}
\end{align*}
\]

Since our semantics minimizes the application of cr-rules,

\[
\{ o(\text{blow\_up}(b), 0), o(\text{surge}, 0) \}
\]

is not an explanation.
Talk Outline

- Agent architecture
- Typical scenarios
- Representing the agent’s knowledge
- The agent’s behavior and reasoning

⇒ The use of preferences

- On-going research
Adding Preferences

• Suppose now that the agent’s knowledge base is expanded by the statement

\[ \text{prefer}(r_1(T), r_2(T)) \]

which says

“Explanations based on blow-ups are preferred to explanations based on power surges”.

\[ \Rightarrow \text{The CR-Program } \Pi_d \cup O \text{ has one answer set containing the unique preferred explanation} \]

\[ \{ o(blow\_up(b), 0) \} \]

Now we will consider a more complex preference between our cr-rules.
New Scenario

Consider a new fluent, \textit{storm}, and a new preference: “\textit{Prefer explanations based on bulb blow-ups to those based on power surges, unless there is a storm in the area.}”

\[
pref(r_1(T), r_2(T)) \leftarrow h(\neg storm, T).\]
\[
pref(r_2(T), r_1(T)) \leftarrow h(storm, T).
\]

1. (Initial) Observations:

\[
obs(storm, 0), obs(\neg on(b), 0), obs(prot(b), 0), \ldots
\]

[\ldots]

1. Observations:

\[
obs(\neg on(b), 1).
\]

2. Interpret Obs.:

\[
\{o(surge, 0)\}
\]

1. Observations:

\[
obs(\neg on(b), 1), obs(\neg ab(r), 1).
\]

2. Interpret Obs.:

\[
\{o(blow\_up(b), 0)\}\]
Another Scenario

1. Observations:

\( \text{obs(storm, 0)} \) or \( \text{obs(\neg \text{storm}, 0)} \), \( \text{obs(\neg \text{on}(b), 0)} \), ...  

[...]

5. Execute Action: \( \text{close}(s_1) \).

1. Observations: \( \text{obs(\neg \text{on}(b), 1)} \), \( \text{obs(\neg \text{ab}(b), 1)} \).

2. Interpret Observations: two answer sets:

\[
\{ \text{obs(storm, 0)}, \text{o(surge, 0)}, \ldots \} \\
\{ \text{obs(\neg \text{storm}, 0)}, \text{o(surge, 0)}, \ldots \}
\]
Other Applications of CR-rules:

• Specification and selection of quality plans:
  ◊ Find a plan in which unreliable switches are used only if absolutely necessary.

• Specification of unlikely events:
  ◊ In some rare circumstances loading the gun may fail if the agent is in a hurry.
Related Work: DLV

DLV extends A-Prolog by weak constraints, i.e. statements of the form:

\[ \sim l_1, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n : \text{weight.} \]

- Intuitively, these constraints may be violated but this should happen as rarely as possible.
- **Weight**: the cost of violating the weak constraint.
- This intuition is captured by the notion of answer set which minimizes the sum of the weights of the constraints that it violates.

In the previous scenario, DLV-based reasoning algorithm computes the answer set:

\[ \{ \text{obs(storm, 0), o(surge, 0), \ldots} \} \]
Talk Outline

- Agent architecture
- Typical scenarios
- Representing the agent’s knowledge
- The agent’s behavior and reasoning
- The use of preferences

⇒ On-going research
Implementation of the Agent Loop

A very preliminary version of the agent loop, with graphical interface and loop both implemented in Java, can be downloaded from:

http://krlab.cs.ttu.edu/~marcy/APLAgent/
CR-Prolog\textsubscript{2} (with Veena Mellarkod)

CR-Prolog\textsubscript{2} is an extension of CR-Prolog which allows ordered disjunction in the head of both regular rules and cr-rules.

⇒ More concise, easier to read representation of knowledge

Recall the rules we used earlier to interpret observations:

\[
\begin{align*}
    r_1(T) & : \ o(\text{blow\_up}(B),T) \leftarrow. \\
    r_2(T) & : \ o(\text{surge},T) \leftarrow. \\
    \text{prefer}(r_1,r_2).
\end{align*}
\]

Compare them with the equivalent CR-Prolog\textsubscript{2} rule:

\[
o(\text{blow\_up}(B),T) \times o(\text{surge},T) \leftarrow.
\]
CR-Prolog$_2$ (cont.)

The semantics of CR-Prolog$_2$ is an improvement over the semantics of CR-Prolog.

⇒ Solves some problems present in the semantics of CR-Prolog

For example, consider:

\[
\begin{align*}
  r_r : \text{run} & \leftarrow. \\
  r_s : \text{swim} & \leftarrow. \\
  r_b : \text{play\_ball} & \leftarrow. \\
  r_w : \text{lift\_weights} & \leftarrow. \\
  \text{full\_body\_exercise} & \leftarrow \text{run, lift\_weights}. \\
  \text{full\_body\_exercise} & \leftarrow \text{swim, play\_ball}. \\
  & \leftarrow \text{not full\_body\_exercise}. \\
  \text{prefer}(r_r, r_s) & \leftarrow \text{not ignore\_prefs}. \\
  \text{prefer}(r_b, r_w) & \leftarrow \text{not ignore\_prefs}. \\
  r_z : \text{ignore\_prefs} & \leftarrow. 
\end{align*}
\]

Expected conclusions:

\[
\{\text{ignore\_prefs, run, lift\_weights}\} \\
\{\text{ignore\_prefs, swim, play\_ball}\}
\]

- Under the CR-Prolog semantics: no answer sets.
- Under the CR-Prolog$_2$ semantics, answer sets match intuition.
Future Work

• Finding an efficient inference algorithm for CR-Prolog$_2$.

• Integrating cr-rules in existing A-Prolog engines.

• Extending the mathematical properties of A-Prolog to CR-Prolog$_2$. 
% State Constraints
\[ h(\text{closed}(s_2), T) \leftarrow h(\text{closed}(s_1), T), \]
\[ h(\neg \text{ab}(r), T). \]
\[ h(\text{on}(b), T) \leftarrow h(\text{closed}(s_2), T), \]
\[ h(\neg \text{ab}(b), T). \]

% Dynamic Laws
\[ h(\text{closed}(s_1), T + 1) \leftarrow o(\text{close}(s_1), T). \]
\[ h(\text{ab}(B), T + 1) \leftarrow o(\text{blow\_up}(B), T). \]
\[ h(\text{ab}(r), T + 1) \leftarrow o(\text{surge}, T). \]
\[ h(\text{ab}(B), T + 1) \leftarrow h(\neg \text{prot}(B), T), \]
\[ o(\text{surge}, T). \]

% Inertia
\[ h(L, T + 1) \leftarrow h(L, T), \]
\[ \text{not } h(L, T + 1). \]
System Description in D.L.

% State Constraints
$$\frac{h(\text{closed}(s_1),T) \land h(\neg \text{ab}(r),T)}{h(\text{closed}(s_2),T)}$$

$$\frac{h(\text{closed}(s_2),T) \land h(\neg \text{ab}(b),T)}{h(\text{on}(b),T)}$$

% Dynamic Laws
$$\frac{o(\text{close}(s_1),T)}{h(\text{closed}(s_1),T+1)} \quad \frac{o(\text{blow\_up}(B),T)}{h(\text{ab}(B),T+1)}$$

$$\frac{o(\text{surge},T)}{h(\text{ab}(r),T+1)} \quad \frac{h(\neg \text{prot}(B),T) \land o(\text{surge},T)}{h(\text{ab}(B),T+1)}$$

% Inertia
$$\frac{h(L,T) : h(L,T+1)}{h(L,T+1)}$$
Semantics of Abductive Logic Programs

• Used to define the semantics of CR-Prolog.

• Abductive logic programs are pairs $\langle \Pi, A \rangle$ where $\Pi$ is a program of A-Prolog and $A$ is a set of atoms, called abducibles.

• The semantics of an abductive program, $\Pi$, 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.
Semantics of CR-Prolog

Definition 1 The hard reduct \( hr(\Pi) = \langle H_\Pi, atoms(\{appl\}) \rangle \) transforms CR-Prolog programs into abductive programs. It is defined as follows:

1. Every regular rule of \( \Pi \) belongs to \( H_\Pi \).

2. For every cr-rule \( \rho \) of \( \Pi \), with name \( r \), the following belongs to \( H_\Pi \):
   \[
   \text{head}(\rho) \leftarrow \text{body}(\rho), \text{appl}(r).
   \]

3. If \( prefer \) occurs in \( \Pi \), \( H_\Pi \) contains the following set of rules, denoted by \( \Pi_p \):

   \[
   \begin{cases}
   \text{\% transitive closure of predicate prefer} \\
   \text{is\_preferred}(R_1, R_2) \leftarrow prefer(R_1, R_2). \\
   \text{is\_preferred}(R_1, R_2) \leftarrow prefer(R_1, R_3), \text{is\_preferred}(R_3, R_2). \\
   \text{\% no circular preferences} \\
   \text{\% prohibits application of a lesser rule if} \\
   \text{\% a better rule is applied} \\
   \text{\% appl}(R_1), \text{appl}(R_2), \text{is\_preferred}(R_1, R_2). \\
   \end{cases}
   \]

\( (R_1, R_2, R_3 \text{ are variables for names of rules.}) \)
Semantics of CR-Prolog (cont.)

**Definition 2** A set of literals, $C$, is a candidate answer set of $\Pi$ if $C$ is a minimal generalized answer set of $hr(\Pi)$.

**Definition 3** Let $C$, $D$ be candidate answer sets of $\Pi$. $C$ is better than $D$ ($C \prec D$) if

$$\exists appl(r_1) \in C \quad \exists appl(r_2) \in D$$

$$\text{is\_preferred}(r_1, r_2) \in C \cap D.$$  \hspace{1cm} (1)

(In the following definition, $\text{atoms}(\{p, q\})$ denotes the set of atoms formed by predicates $p$ and $q$.)

**Definition 4** Let $C$ be a candidate answer set of $\Pi$, and $\bar{C}$ be $C \setminus \text{atoms}(\{appl, \text{is\_preferred}\})$. $\bar{C}$ is an answer set of $\Pi$ if there exists no candidate answer set, $D$, of $\Pi$ which is better than $C$. 

Semantics of CR-Prolog
– Examples –

Example

Let us compute the answer sets of:

$$\Pi_1 = \begin{cases} 
  r_1 : p & \leftarrow r, \text{not } q. \\
  r_2 : r. \\
  r_3 : s & \leftarrow r.
\end{cases}$$

(Notice that $\Pi_1 \setminus \{r_3\}$ is consistent.)

The hard reduct of $\Pi_1$ is given by ($\Pi_p$ is omitted):

$$H'_{\Pi_1} = \begin{cases} 
  r_1 : p & \leftarrow r, \text{not } q. \\
  r_2 : r. \\
  r'_3 : s & \leftarrow r, \text{appl}(r_3).
\end{cases}$$

- $\{p, r, s, \text{appl}(r_3)\}$ is a generalized answer set of $hr(\Pi_1)$, but it is not minimal.
- The only minimal generalized answer set of $hr(\Pi_1)$ is $C = \{p, r\}$.
- $C$ is the only answer set of $\Pi_1$. 
Example

\[\Pi_2 = \begin{cases} r_1 & : p \leftarrow \neg q. \\ r_2 & : r \leftarrow \neg s. \\ r_3 & : q \leftarrow t. \\ r_4 & : s \leftarrow t. \\ r_5 & : p, r. \end{cases}\]

The hard reduct of \(\Pi_2\) is given by:

\[H'_{\Pi_2} = \begin{cases} r_1 & : p \leftarrow \neg q. \\ r_2 & : r \leftarrow \neg s. \\ r_3 & : q \leftarrow t. \\ r_4 & : s \leftarrow t. \\ r_5 & : p, r. \end{cases}\]

\[r'_6 : q \leftarrow \text{appl}(r_6).\]

\[r'_7 : s \leftarrow \text{appl}(r_7).\]

\[r'_8 : t \leftarrow \text{appl}(r_8).\]

\[r_9 : \text{prefer}(r_6, r_7).\]

- The candidate answer sets of \(\Pi_2\) are (\textit{is\_preferred} is omitted):

  \[C_1 = \{\text{prefer}(r_6, r_7), \text{appl}(r_6), q, r\}\]

  \[C_2 = \{\text{prefer}(r_6, r_7), \text{appl}(r_7), s, p\}\]

  \[C_3 = \{\text{prefer}(r_6, r_7), \text{appl}(r_8), t, q, s\}\]

- Since \(C_1 \prec C_2\), \(\hat{C}_2\) is not an answer set of \(\Pi_2\), while \(\hat{C}_1\) and \(\hat{C}_3\) are.
Example

\[ \Pi_3 = \begin{cases} 
 r_1 : & a \leftarrow p. \\
 r_2 : & a \leftarrow r. \\
 r_3 : & b \leftarrow q. \\
 r_4 : & b \leftarrow s. \\
 r_{5a} : & \leftarrow \text{not } a. \\
 r_{5b} : & \leftarrow \text{not } b. \\
 r_6 : & p \leftarrow. \\
 r_7 : & q \leftarrow. \\
 r_8 : & r \leftarrow. \\
 r_9 : & s \leftarrow. \\
 r_{10} : & \text{prefer}(r_6, r_7). \\
 r_{11} : & \text{prefer}(r_8, r_9). 
\end{cases} \]

The candidate answer sets of \( \Pi_3 \) are:

\[ C_1 = \{ \text{prefer}(r_6, r_7), \text{prefer}(r_8, r_9), \text{appl}(r_6), \text{appl}(r_9), p, s, a, b \} \]
\[ C_2 = \{ \text{prefer}(r_6, r_7), \text{prefer}(r_8, r_9), \text{appl}(r_8), \text{appl}(r_7), r, q, a, b \} \]

Since \( C_1 \prec C_2 \) and \( C_2 \prec C_1 \), \( \Pi_3 \) has no answer set.