CR-Prolog: Logic Programs with Consistency-Restoring Rules

Marcello Balduccini and Michael Gelfond

August, 21 2003

Talk Outline

- $\Rightarrow$ **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

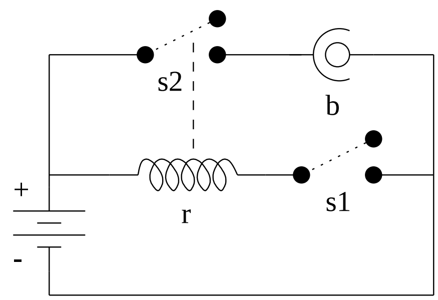
1. observe the world;
2. interpret the observations;
3. select a goal;
4. plan;
5. execute part of the plan.

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



- the *agent actions*:
  - ◇ close s_1
- the *exogenous actions*:
  - ◇ power surge: damages the relay; also damages the bulb if not protected
  - ◇ bulb blow-up

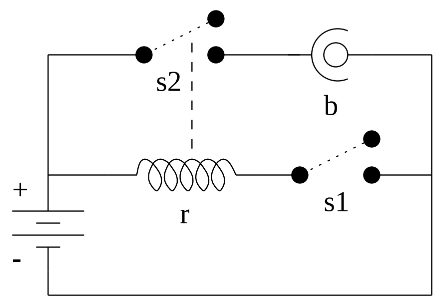
Talk Outline

- Agent architecture

$\Rightarrow$ **Typical scenarios**

- Representing the agent's knowledge
- The agent's behavior and reasoning
- The use of preferences
- On-going research

A Scenario



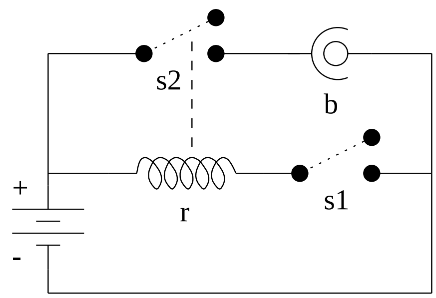
1. Observations: s_1 open; s_2 open; b off; b protected; components are ok.

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

$\Rightarrow$ **Representing the agent's knowledge**

- The agent's behavior and reasoning
- The use of preferences
- On-going research

Domain Signature

Fluents:

$closed(SW)$ – switch SW is closed

$prot(B)$ – bulb B is protected from surges

$on(B)$ – bulb B is on

$ab(C)$ – component C is malfunctioning

Agent Actions:

$close(SW)$ – close switch SW

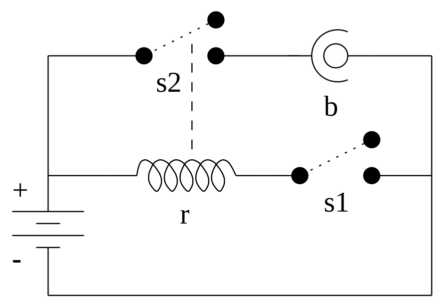
Exogenous Actions:

$blow\_up(B)$ – bulb B blows up

$surge$ – power surge

Causal Relations

Causal relations are described in the **system description**, Π_d.

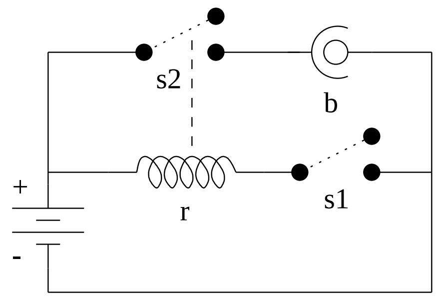


State Constraints:

$$\begin{array}{rcl} h(closed(s_2),T) & \leftarrow & h(closed(s_1),T), \\ & & h(\neg ab(r),T). \end{array}$$

$$\begin{array}{rcl} h(on(b),T) & \leftarrow & h(closed(s_2),T), \\ & & h(\neg ab(b),T). \end{array}$$

Causal Relations



Dynamic Laws:

$$h(closed(s_1), T+1) \leftarrow o(close(s_1), T).$$

$$h(ab(B), T+1) \leftarrow o(blow\_up(B), T).$$

Inertia Axiom:

$$\begin{aligned} h(L, T+1) \leftarrow\ & h(L, T), \\ & \text{not } h(\overline{L}, T+1). \end{aligned}$$

The Agent's Observations

Observations are represented by statements:

$obs(L, T)$ – L was observed at T

$hpd(A, T)$ – A happened at T

- obs and hpd describe observations (which we assume always correct).
- h and o are agent's predictions.
- They are linked, in Π_d, by axioms:

$$\begin{array}{rcl} o(A, T) & \leftarrow & hpd(A, T). \\ h(L, 0) & \leftarrow & obs(L, 0). \end{array}$$

and by the *Reality Checks*:

$$\leftarrow obs(L, T), h(\overline{L}, T).$$

Talk Outline

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

$\Rightarrow$ **The agent's behavior and reasoning**

- The use of preferences
- On-going research

KRAgent in Action

1. Observations:

$obs(\neg on(b), 0), obs(prot(b), 0), obs(\neg closed(s_1), 0), \ldots$

3. Select goal: $on(b)$.

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:

$$
\begin{array}{rcl}
o(A,T) \text{ or } \neg o(A,T) & \leftarrow & n \leq T < n+l, \\
 & & agent\_act(A). \\
 & & \\
goal & \leftarrow & h(on(b),T). \\
 & & \\
 & \leftarrow & \text{not } goal.
\end{array}
$$

KRAgent in Action

1. Observations:

$obs(\neg on(b), 0), obs(prot(b), 0), obs(\neg closed(s_1), 0), \ldots$

3. Select goal: $on(b)$.

4. Plan with $l = 1$. The KRAgent's plan is:

$$o(close(s_1), 0)$$

5. Perform Action: $close(s_1)$. The KRAgent also **records its execution** with a statement

$$hpd(close(s_1), 0).$$

Note that $\Pi_d \cup O \models h(on(b), 1)$

1. Observations: $obs(\neg on(b), 1)$.

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:

$$\{o(blow\_up(b), 0)\}$$
$$\{o(surge, 0)\}$$
$$\{o(blow\_up(b), 0), o(surge, 0)\}$$

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.

$\Rightarrow$ 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:

$$r : \; l \stackrel{+}{\leftarrow}$$

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:

$$\begin{array}{lrl} r_1(T): & o(blow\_up(B),T) & \stackrel{+}{\leftarrow} . \\ r_2(T): & o(surge,T) & \stackrel{+}{\leftarrow} . \end{array}$$

CR-Prolog

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

$$r : h_1 \text{ or } \ldots \text{ or } h_k \stackrel{+}{\leftarrow} l_1, \ldots, l_m, \text{not } l_{m+1}, \ldots, \text{not } l_n$$

and the preference relation between them

$$prefer(r_1, r_2)$$

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

$$\Pi_0 : \begin{cases} a \leftarrow \text{not } b. \\ r_1 : b \stackrel{+}{\leftarrow} . \end{cases}$$

• Π_0 has an answer set $\{a\}$, computed **without** the use of cr-rule r_1.

Example

$$\Pi_0' : \begin{cases} a \leftarrow \text{not } b. \\ \neg a. \\ r_1 : b \stackrel{+}{\leftarrow} . \end{cases}$$

• If r_1 is not used, Π_0' in **inconsistent**.
• The application of r_1 restores consistency, and leads to the answer set $\{\neg a, b\}$.

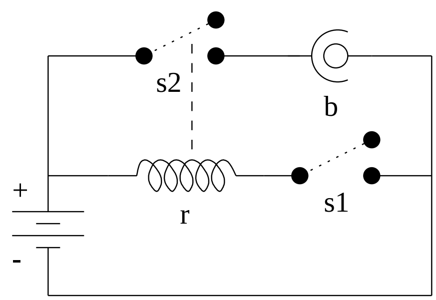
Interpreting Observations

- Suppose now that the agent's knowledge Π_d contains rules $r_1(T)$ and $r_2(T)$.

$\Rightarrow$ **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.)



$[\ldots]$

5. Perform Action: $close(s_1)$.

1. Observations: $obs(\neg on(b), 1)$.

2. Interpret Observations: the explanations of observations are:

$$\{o(blow\_up(b), 0)\}$$
$$\{o(surge, 0)\}$$

Since our semantics minimizes the application of cr-rules,

$$\{o(blow\_up(b), 0), o(surge, 0)\}$$

is not an explanation.

Talk Outline

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

$\Rightarrow$ **The use of preferences**

- On-going research

Adding Preferences

- Suppose now that the agent's knowledge base is expanded by the statement

$$prefer(r_1(T), r_2(T))$$

which says
"*Explanations based on blow-ups are preferred to explanations based on power surges*".

⇒ The CR-Program $\Pi_d \cup O$ 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, $storm$, and a new preference: "*Prefer explanations based on bulb blow-ups to those based on power surges, unless there is a storm in the area.*"

$$\begin{array}{lcl} prefer(r_1(T), r_2(T)) & \leftarrow & h(\neg storm, T). \\ prefer(r_2(T), r_1(T)) & \leftarrow & h(storm, T). \end{array}$$

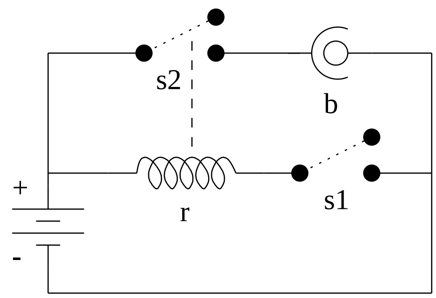
1. (Initial) Observations:

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

[...]

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:

$obs(storm, 0)$ or $obs(\neg storm, 0), obs(\neg on(b), 0), \ldots$

$[\ldots]$

5. Execute Action: $close(s_1)$.

1. Observations: $obs(\neg on(b), 1)$, $obs(\neg ab(b), 1)$.

2. Interpret Observations: two answer sets:

$$\{obs(storm, 0), o(surge, 0), \ldots\}$$
$$\{obs(\neg storm, 0), o(surge, 0), \ldots\}$$

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:

$$\{obs(storm, 0), o(surge, 0), \ldots\}$$

Talk Outline

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

$\Rightarrow$ **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$_2$

(with Veena Mellarkod)

CR-Prolog$_2$ is an extension of CR-Prolog which allows ordered disjunction in the head of both regular rules and cr-rules.

$\Rightarrow$ More concise, easier to read representation of knowledge

Recall the rules we used earlier to interpret observations:

$$\begin{cases} r_1(T): & o(blow\_up(B),T) \stackrel{+}{\leftarrow} . \\ r_2(T): & o(surge,T) \stackrel{+}{\leftarrow} . \\ & prefer(r_1,r_2). \end{cases}$$

Compare them with the equivalent CR-Prolog$_2$ rule:

$$o(blow\_up(B),T) \times o(surge,T) \stackrel{+}{\leftarrow} .$$

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:

$$
\left\{
\begin{array}{l}
\begin{array}{ll}
r_r : run \stackrel{+}{\leftarrow} . & r_s : swim \stackrel{+}{\leftarrow} . \\
r_b : play\_ball \stackrel{+}{\leftarrow} . & r_w : lift\_weights \stackrel{+}{\leftarrow} .
\end{array} \\
\\
full\_body\_exercise \leftarrow run, lift\_weights. \\
full\_body\_exercise \leftarrow swim, play\_ball. \\
\leftarrow \text{not } full\_body\_exercise. \\
\\
prefer(r_r, r_s) \leftarrow \text{not } ignore\_prefs. \\
prefer(r_b, r_w) \leftarrow \text{not } ignore\_prefs. \\
r_z : ignore\_prefs \stackrel{+}{\leftarrow} .
\end{array}
\right.
$$

Expected conclusions:

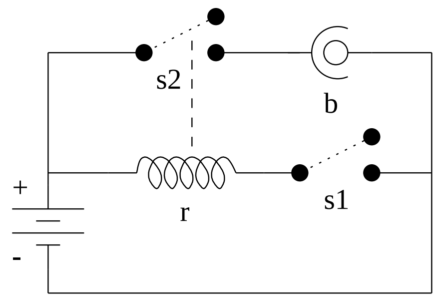
$$
\begin{array}{l}
\{ignore\_prefs, run, lift\_weights\} \\
\{ignore\_prefs, swim, play\_ball\}
\end{array}
$$

- 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$_2$.

System Description



% **State Constraints**

$$h(closed(s_2), T) \leftarrow h(closed(s_1), T), h(\neg ab(r), T).$$

$$h(on(b), T) \leftarrow h(closed(s_2), T), h(\neg ab(b), T).$$

% **Dynamic Laws**

$$h(closed(s_1), T+1) \leftarrow o(close(s_1), T).$$

$$h(ab(B), T+1) \leftarrow o(blow\_up(B), T).$$

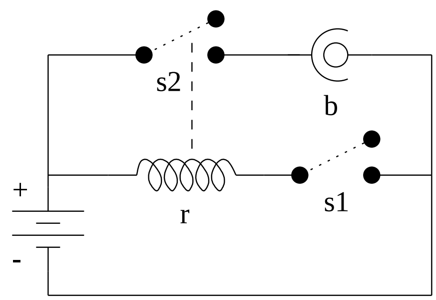
$$h(ab(r), T+1) \leftarrow o(surge, T).$$

$$h(ab(B), T+1) \leftarrow h(\neg prot(B), T), o(surge, T).$$

% **Inertia**

$$h(L, T+1) \leftarrow h(L, T), \text{not } h(\overline{L}, T+1).$$

System Description in D.L.



% **State Constraints**

$$\frac{h(closed(s_1),T) \wedge h(\neg ab(r),T):}{h(closed(s_2),T)}$$

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

% **Dynamic Laws**

$$\frac{o(close(s_1),T):}{h(closed(s_1),T+1)} \qquad \frac{o(blow\_up(B),T):}{h(ab(B),T+1)}$$

$$\frac{o(surge,T):}{h(ab(r),T+1)} \qquad \frac{h(\neg prot(B),T) \wedge o(surge,T):}{h(ab(B),T+1)}$$

% **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 $\langle \Pi, \mathcal{A} \rangle$ where Π is a program of A-Prolog and $\mathcal{A}$ is a set of atoms, called *abducibles*.

- The semantics of an abductive program, Π, is given by the notion of *generalized answer set* – an answer set $M(\Delta)$ of $\Pi \cup \Delta$ where $\Delta \subseteq \mathcal{A}$

- $M(\Delta_1) < M(\Delta_2)$ 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 Π belongs to H_Π.

2. For every cr-rule ρ of Π, with name r, the following belongs to H_Π:
$$head(\rho) \leftarrow \ body(\rho), appl(r).$$

3. If $prefer$ occurs in Π, H_Π contains the following set of rules, denoted by Π_p:

$$\left\{ \begin{array}{l} \text{\% transitive closure of predicate prefer} \\ \quad is\_preferred(R1, R2) \ \leftarrow \ prefer(R1, R2). \\ \quad is\_preferred(R1, R2) \ \leftarrow \ prefer(R1, R3), \\ \qquad\qquad\qquad\qquad\qquad\qquad is\_preferred(R3, R2). \\ \text{\% no circular preferences} \\ \quad \leftarrow is\_preferred(R, R). \\ \text{\% prohibits application of a lesser rule if} \\ \text{\% a better rule is applied} \\ \quad \leftarrow appl(R1), appl(R2), is\_preferred(R1, R2). \end{array} \right.$$

(R_1, R_2, R_3 are variables for names of rules.)

Semantics of CR-Prolog (cont.)

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

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

$$\begin{array}{l} \exists appl(r_1) \in C \quad \exists appl(r_2) \in D \\ is\_preferred(r_1, r_2) \in C \cap D. \end{array} \qquad (1)$$

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

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

Semantics of CR-Prolog
– Examples –

Example

Let us compute the answer sets of:

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

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

The hard reduct of Π_1 is given by (Π_p is omitted):

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

- $\{p, r, s, appl(r_3)\}$ 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 Π_1.

Example

$$
\Pi_2 \left\{ \begin{array}{llll}
r_1: & p & \leftarrow & \text{not } q. \\
r_2: & r & \leftarrow & \text{not } s. \\
r_3: & q & \leftarrow & t. \\
r_4: & s & \leftarrow & t. \\
 & & & \\
r_5: & & \leftarrow & p, r. \\
 & & & \\
r_6: & q \stackrel{+}{\leftarrow}. & & \\
r_7: & s \stackrel{+}{\leftarrow}. & & \\
r_8: & t \stackrel{+}{\leftarrow}. & & \\
 & & & \\
r_9: & prefer(r_6, r_7). & &
\end{array} \right.
$$

The hard reduct of Π_2 is given by:

$$
H'_{\Pi_2} \left\{ \begin{array}{llll}
r_1: & p & \leftarrow & \text{not } q. \\
r_2: & r & \leftarrow & \text{not } s. \\
r_3: & q & \leftarrow & t. \\
r_4: & s & \leftarrow & t. \\
 & & & \\
r_5: & & \leftarrow & p, r. \\
 & & & \\
r'_6: & q & \leftarrow & appl(r_6). \\
r'_7: & s & \leftarrow & appl(r_7). \\
r'_8: & t & \leftarrow & appl(r_8). \\
 & & & \\
r_9: & prefer(r_6, r_7). & &
\end{array} \right.
$$

- The candidate answer sets of Π_2 are ($is\_preferred$ is omitted):

$$
\begin{array}{l}
C_1 = \{prefer(r_6, r_7), appl(r_6), q, r\} \\
C_2 = \{prefer(r_6, r_7), appl(r_7), s, p\} \\
C_3 = \{prefer(r_6, r_7), appl(r_8), t, q, s\}
\end{array}
$$

- Since $C_1 \prec C_2$, $\widehat{C}_2$ is not an answer set of Π_2, while $\widehat{C}_1$ and $\widehat{C}_3$ are.

Example

$$\Pi_3 \left\{ \begin{array}{lcl} r_1: & a & \leftarrow \; p. \\ r_2: & a & \leftarrow \; r. \\ r_3: & b & \leftarrow \; q. \\ r_4: & b & \leftarrow \; s. \\ \\ r_{5a}: & \leftarrow \text{not} & a. \\ r_{5b}: & \leftarrow \text{not} & b. \\ \\ r_6: & p \stackrel{+}{\leftarrow} . & \\ r_7: & q \stackrel{+}{\leftarrow} . & \\ r_8: & r \stackrel{+}{\leftarrow} . & \\ r_9: & s \stackrel{+}{\leftarrow} . & \\ \\ r_{10}: & prefer(r_6, r_7). & \\ r_{11}: & prefer(r_8, r_9). & \end{array} \right.$$

The candidate answer sets of Π_3 are:

$$\begin{array}{l} C_1 = \{\, prefer(r_6, r_7), prefer(r_8, r_9), \\ \qquad\qquad appl(r_6), appl(r_9), p, s, a, b\} \\ C_2 = \{\, prefer(r_6, r_7), prefer(r_8, r_9), \\ \qquad\qquad appl(r_8), appl(r_7), r, q, a, b\} \end{array}$$

Since $C_1 \prec C_2$ and $C_2 \prec C_1$, Π_3 has no answer set.