Solving the Wise Mountain Man Riddle with Answer Set Programming

Marcello Balduccini
Intelligent Systems, OCTO
Eastman Kodak Company
Rochester, NY 14650-2102 USA
marcello.balduccini@gmail.com

Abstract

This paper describes an exercise in the formalization of common-sense with Answer Set Programming aimed at solving an interesting riddle, whose solution is not obvious to many people. Solving the riddle requires a considerable amount of common-sense knowledge and sophisticated knowledge representation and reasoning techniques, including planning and adversarial reasoning. Most importantly, the riddle is difficult enough to make it unclear, at a first analysis, whether and how Answer Set Programming or other formalisms can be used to solve it.

Introduction

This paper describes an exercise in the formalization of common-sense with Answer Set Programming (ASP), aimed at solving the riddle:

"A long, long time ago, two cowboys where fighting to marry the daughter of the OK Corral rancher. The rancher, who liked neither of these two men to become his future son-in-law, came up with a clever plan. A horse race would determine who would be allowed his daughter's hand. Both cowboys had to travel from Kansas City to the OK Corral, and the one whose horse arrived LAST would be proclaimed the winner.

The two cowboys, realizing that this could become a very lengthy expedition, finally decided to consult the Wise Mountain Man. They explained to him the situation, upon which the Wise Mountain Man raised his cane and spoke four wise words. Relieved, the two cowboys left his cabin: They were ready for the contest!

Which four wise words did the Wise Mountain Man speak?"

This riddle is interesting because it is easy to understand, but not trivial, and the solution is not obvious to many people. The story can be simplified in various ways without losing the key points. The story is also entirely based on common-sense knowledge. The amount of knowledge that needs to be encoded is not large, which simplifies the encoding; on the other hand, as we will see in the rest of the paper, properly dealing with the riddle requires various sophisticated capabilities, including modeling direct and indirect effects of actions, encoding triggers, planning, dealing with defaults and their exceptions, and concepts from multi-agent systems such as adversarial reasoning. The riddle is difficult enough to make it unclear, at a first analysis, whether and how ASP or other formalisms can be used to formalize the story and underlying reasoning.

In the course of this paper we will discuss how the effects of the actions involved in the story can be formalized, and how to address the main issues of determining that "this could be a lengthy expedition" and of answering the final question.

We begin with a brief introduction on ASP. Next, we show how the knowledge about the riddle is encoded and how reasoning techniques can be used to solve the riddle. Finally, we draw conclusions.

Background

ASP (Marek and Truszczynski 1999) is a programming paradigm based on language A-Prolog (Gelfond and Lifschitz 1991) and its extensions (Balduccini and Gelfond 2003; Brewka, Niemela, and Syrjanen 2004; Mellarkod, Gelfond, and Zhang 2008). In this paper we use the extension of A-Prolog called CR-Prolog (Balduccini and Gelfond 2003), which allows, among other things, simplified handling of exceptions, rare events. To save space, we describe only the fragment of CR-Prolog that will be used in this paper.

Let Σ be a signature containing constant, function and predicate symbols. Terms and atoms are formed as usual. A literal is either an atom a or its strong (also called classical or epistemic) negation $\neg a$.

A *regular rule* (rule, for short) is a statement of the form:

$$h_1 \ \vee \ \dots \ \vee \ h_k \leftarrow l_1, \dots, l_m, \text{not } l_{m+1}, \dots, \text{not } l_n$$

where h_i's and l_i's are literals and *not* is the so-called *default negation*.[1] The intuitive meaning of a rule is that a reasoner, who believes $\{l_1, \dots, l_m\}$ and has no reason to believe $\{l_{m+1}, \dots, l_n\}$, has to believe one of h_i's.

A *consistency restoring rule* (cr-rule) is a statement of the form:

$$h_1 \ \vee \ \dots \ \vee \ h_k \stackrel{+}{\leftarrow} l_1, \dots, l_m, \text{not } l_{m+1}, \dots, \text{not } l_n$$

where h_i's and l_i's are as before. The informal meaning of a cr-rule is that a reasoner, who believes $\{l_1, \dots, l_m\}$ and

[1] We also allow the use of SMODELS style choice rules, but omit their formal definition to save space.

has no reason to believe $\{l_{m+1}, \ldots, l_n\}$, may believe one of h_i's, but only if strictly necessary, that is only if no consistent set of beliefs can be formed otherwise.

A *program* is a pair $\langle \Sigma, \Pi \rangle$, where Σ is a signature and Π is a set of rules and cr-rules over Σ. Often we denote programs by just the second element of the pair, and let the signature be defined implicitly.

Given a CR-Prolog program Π, we denote the set of its regular rules by Π^r and the set of its cr-rules by Π^{cr}. By $\alpha(r)$ we denote the regular rule obtained from cr-rule r by replacing the symbol $\stackrel{+}{\leftarrow}$ with $\leftarrow$. Given a set of cr-rules R, $\alpha(R)$ denotes the set obtained by applying α to each cr-rule in R. The semantics of a CR-Prolog program is defined in two steps.

Definition 1 *Given a CR-Prolog program Π, a minimal (with respect to set-theoretic inclusion) set R of cr-rules of Π, such that $\Pi^r \cup \alpha(R)$ is consistent is called an* abductive support *of* Π.

Definition 2 *Given a CR-Prolog program Π, a set of literals A is an* answer set *of* Π *if it is an answer set of the program $\Pi^r \cup \alpha(R)$ for some abductive support R of* Π.

To represent knowledge and reason about dynamic domains, we use ASP to encode dynamic laws, state constraints and executability conditions (Gelfond and Lifschitz 1998). The laws are written directly in ASP, rather than represented using an action language (Gelfond 2002), to save space and have a more uniform representation.

The key elements of the representation are as follows; we refer the readers to e.g. (Gelfond 2002) for more details. The evolution of a dynamic domain is viewed as a *transition diagram*, which is encoded in a compact way by means of an *action description* consisting of dynamic laws (describing the direct effects of actions), state constraints (describing the indirect effects), and executability conditions (stating when the actions can be executed). Properties of interest, whose truth value changes over time, are represented by *fluents* (e.g. $on(block_1, block_2)$). A state of the transition diagram is encoded as a consistent and complete set of fluent literals (i.e. fluents and their negations). The truth value of a fluent f is encoded by a statement of the form $h(f, s)$, where s is an integer denoting the step in the evolution of the domain, intuitively saying that f holds at step s. The fact that f is false is denoted by $\neg h(f, s)$. Occurrences of actions are traditionally represented by expressions of the form $o(a, s)$, saying that a occurs at step s.

Formalizing the Riddle

The next step is to encode the knowledge about the domain of the story. To focus on the main issues, we abstract from several details and concentrate on the horse ride. The objects of interest are the two competitors (a, b), the two horses ($h(a)$, $h(b)$), and locations $start$, $finish$, and en_route. Horse ownership is described by relation $owns$, defined by the rule $owns(C, h(C)) \leftarrow competitor(C)$.

The fluents of interest and their informal meanings are: $at(X, L)$, "competitor or horse X is at location L"; $riding(C, H)$, "competitor C is riding horse H"; $crossed(X)$, "competitor or horse X has crossed the finish line."

The actions of interest are $wait$, $move$ (the actor moves to the next location along the race track), and $cross$ (the actor crosses the finish line). Because this domain involves multiple actors, we represent the occurrence of actions by a relation $o(A, C, S)$, which intuitively says that action A occurred, performed by competitor C, at step S.[2]

The formalization of action $move$ deserves some discussion. Typically, it is difficult to predict who will complete a race first, as many variables influence the result of a race. To keep our formalization simple, we have chosen a rather coarse-grained model of the movements from one location to the other. Because often one horse will be faster than the other, we introduce a relation $faster(H)$, which informally says that H is the faster horse. This allows us to deal with both simple and more complex situations: when it is known which horse is faster, we encode the information as a fact. When the information is not available, we use the disjunction $faster(h(a)) \ \vee \ faster(h(b))$. Action $move$ is formalized so that, when executed, the slower horse moves from location $start$ to en_route and from en_route to $finish$. The faster horse, instead, moves from $start$ directly to $finish$.[3] The direct effects of the actions can be formalized in ASP as follows:[4]

- Action $move$:

  % If competitor C is at start and riding the faster horse,
  % action $move$ takes him to the finish line.
  $$\begin{array}{l} h(at(C, finish), S+1) \leftarrow \\ \qquad h(at(C, start), S), \\ \qquad h(riding(C, H), S), \\ \qquad faster(H), \\ \qquad o(move, C, S). \end{array}$$

  % If competitor C is at start and riding the slower horse,
  % action $move$ takes him to location "en route."
  $$\begin{array}{l} h(at(C, en\_route), S+1) \leftarrow \\ \qquad h(at(C, start), S), \\ \qquad h(riding(C, H), S), \\ \qquad \text{not } faster(H), \\ \qquad o(move, C, S). \end{array}$$

---

[2] This simple representation is justified because the domain does not include exogenous actions. Otherwise, we would have to use a more sophisticated representation, such as specifying the actor as an argument of the terms representing the actions.

[3] More refined modeling is possible, but is out of the scope of the present discussion. However, we would like to mention the possibility of using the recent advances in integrating ASP and constraint satisfaction (Mellarkod, Gelfond, and Zhang 2008) to introduce numerical distances, speed, and to take into account parameters such as stamina in their computation.

[4] Depending on the context, executability conditions might be needed stating that each competitor must be riding in order to perform the $move$ or $cross$ actions. Because the story assumes that the competitors are riding at all times, we omit such executability conditions to save space.

% Performing $move$ while "en route" takes the actor
% to the finish line.

$$h(at(C, finish), S+1) \leftarrow h(at(C, en_route), S), o(move, C, S).$$

% $move$ cannot be executed while at the finish line.

$$\leftarrow o(move, C, S), h(at(C, finish), S).$$

- Action $cross$:

% Action $cross$, at the finish line, causes the actor to
% cross the finish line.

$$h(crossed(C), S+1) \leftarrow o(cross, C, S), h(at(C, finish), S).$$

% $cross$ can only be executed at the finish line.

$$\leftarrow o(cross, C, S), h(at(C, L), S), L \neq finish.$$

% $cross$ can be executed only once by each competitor.

$$\leftarrow o(cross, C, S), h(crossed(C), S).$$

No rules are needed for action $wait$, as it has no direct effects. The state constraints are:

- "Each competitor or horse can only be at one location at a time."

$$\neg h(at(X, L_2), S) \leftarrow h(at(X, L_1), S), L_1 \neq L_2.$$

- "The competitor and the horse he is riding on are always at the same location."

$$h(at(H, L), S) \leftarrow h(at(C, L), S), h(riding(C, H), S).$$

$$h(at(C, L), S) \leftarrow h(at(H, L), S), h(riding(C, H), S).$$

It is worth noting that, in this formalization, horses do not perform actions on their own (that is, they are viewed as "vehicles"). Because of that, only the first of the two rules above is really needed. However, the second rule makes the formalization more general, as it allows one to apply it to cases when the horses can autonomously decide to perform actions (e.g. the horse suddenly moves to the next location and the rider is carried there as a side-effect).

- "Each competitor can only ride one horse at a time; each horse can only have one rider at a time."

$$\neg h(riding(X, H2), S) \leftarrow h(riding(X, H1), S), H1 \neq H2.$$

$$\neg h(riding(C2, H), S) \leftarrow h(riding(C1, H), S), C1 \neq C2.$$

- "The competitor and the horse he is riding on always cross the finish line together."

$$h(crossed(H), S) \leftarrow h(crossed(C), S), h(riding(C, H), S).$$

$$h(crossed(C), S) \leftarrow h(crossed(H), S), h(riding(C, H), S).$$

As noted for the previous group of state constraints, only the first of these two rules is strictly necessary, although the seconds increases the generality of the formalization.

The action description is completed by the law of inertia (Hayes and McCarthy 1969), in its usual ASP representation (e.g. (Gelfond 2002)):

$$h(F, S+1) \leftarrow h(F, S), \text{not } \neg h(F, S+1).$$

$$\neg h(F, S+1) \leftarrow \neg h(F, S), \text{not } h(F, S+1).$$

Reasoning About the Riddle

Let us now see how action description $\mathcal{AD}$, consisting of all of the rules from the previous section, is used to reason about the riddle.

The first task that we want to be able to perform is determining the winner of the race, based on the statement from the riddle "the one whose horse arrived LAST would be proclaimed the winner." In terms of the formalization developed so far, arriving last means being the last to cross the finish line. Encoding the basic idea behind this notion is not difficult, but attention must be paid to the special case of the two horses crossing the finish line together. Commonsense seems to entail that, if the two horses cross the line together, then they are both first. (One way to convince oneself about this is to observe that the other option is to say that both horses arrived last. But talking about "last" appears to imply that they have been preceded by some horse that arrived "first.") The corresponding definition of relations $first\_to\_cross$ and $last\_to\_cross$ is:[5]

% $first\_to\_cross(H)$: horse H crossed the line first.

$$first\_to\_cross(H_1) \leftarrow h(crossed(H_1), S_2), \neg h(crossed(H_2), S_1), S_2 = S_1 + 1, horse(H_2), H_1 \neq H_2.$$

% $last\_to\_cross(H)$: horse H crossed the line last.

$$last\_to\_cross(H_1) \leftarrow h(crossed(H_1), S_2), \neg h(crossed(H_1), S_1), S_2 = S_1 + 1, h(crossed(H_2), S_1), horse(H_2), H_1 \neq H_2.$$

Winners and losers can be determined from the previous relations, and from horse ownership:

% C wins if his horse crosses the finish line last.

$$wins(C) \leftarrow owns(C, H), last\_to\_cross(H).$$

[5] To save space, the definitions of these relations are given for the special case of a 2-competitor race. Extending the definitions to the general case is not difficult, but requires some extra rules.

$\%\ C$ loses if his horse crosses the finish line first.
$loses(C) \leftarrow owns(C,H), first\_to\_cross(H).$

Let $\mathcal{W}$ be the set consisting of the definitions of $last\_to\_cross$, $first\_to\_cross$, $wins$, and $loses$. It is not difficult to check that, given suitable input about the initial state, $\mathcal{AD} \cup \mathcal{W}$ entails intuitively correct conclusions. For example, let σ_0 denote the intended initial state of the riddle, where each competitor is at the start location, riding his horse:

$$h(at(a,start),0). \quad h(at(b,start),0).$$

$$\begin{array}{l} h(riding(C,H),0) \leftarrow \\ \qquad owns(C,H), \\ \qquad \text{not } \neg h(riding(C,H),0). \end{array}$$

$$\neg h(F,0) \leftarrow \text{not } h(F,0).$$

The rule about fluent $riding$ captures the intuition that normally one competitor rides his own horse, but there may be exceptions. Also notice that the last rule in σ encodes the Closed World Assumption, and provides a compact way to specify the fluents that are false in σ. Also, notice that it is not necessary to specify explicitly the location of the horses, as that will be derived from the locations of their riders by state constraints of $\mathcal{AD}$. Assuming that a's horse is the faster, let $F^a = \{faster(h(a))\}$. Let also O^0 denote the set $\{o(a,move,0), o(b,move,0)\}$. It is not difficult to see that $\sigma \cup F^a \cup O^0 \cup \mathcal{AD} \cup \mathcal{W}$ entails:

$$\{h(at(a,finish),1), h(at(b,en\_route),1)\},$$

meaning that a is expected to arrive at the finish, and b at location "en route." Similarly, given

$$O^1 = \left\{ \begin{array}{ll} o(a,move,0). & o(b,move,0). \\ o(a,wait,1). & o(b,move,1). \\ o(a,wait,2). & o(b,cross,2). \\ o(a,cross,3). & \end{array} \right.$$

the theory $\sigma \cup F^a \cup O^1 \cup \mathcal{AD} \cup \mathcal{W}$ entails:

$$\begin{array}{l} \{h(at(a,finish),1),\ h(at(b,finish),2), \\ \ h(crossed(a),4),\ h(crossed(b),3), \\ \ last(h(a)),\ first(h(b)), \\ \ wins(a),\ loses(b)\}, \end{array}$$

meaning that both competitors crossed the finish line, but b's horse crossed it first, and therefore b lost the race.

The next task of interest is to use the theory developed so far to determine that the race "could become a very lengthy expedition." Attention must be paid to the interpretation of this sentence. Intuitively, the sentence refers to the fact that none of the competitors might be able to end the race. However, this makes sense only if interpreted with commonsense. Of course sequences of actions exist that cause the race to terminate. For example, one competitor could ride his horse as fast as he can to the finish line and then cross, but that is likely to cause him to lose the race.

We believe the correct interpretation of the sentence is that we need to check if the two competitors *acting rationally* (i.e. selecting actions in order to achieve their own goal) will ever complete the race. In the remainder of the discussion, we call this the *completion problem*. Notice that, under the assumption of rational acting, no competitor will just run as fast as he can to the finish line and cross it, without paying attention to where the other competitor is.

In this paper, we will focus on addressing the completion problem from the point of view of one of the competitors. That is, we are interested in the reasoning that one competitor needs to perform to solve the problem. So, we will define a relation me, e.g. $me(a)$. In the rest of the discussion, we refer to the competitor whose reasoning we are examining as "our competitor," while the other competitor is referred to as the "adversary."

The action selection performed by our competitor can be formalized using the well-known ASP planning technique (e.g. (Gelfond 2002)) based on a generate-and-test approach, encoded by the set $\mathcal{P}_{me}$ of rules:

$$me(a).$$

$$\begin{array}{l} 1\{\ o(A,C,S) : relevant(A)\ \}1 \leftarrow me(C). \\ \leftarrow \text{not } wins(C), me(C), selected\_goal(win). \end{array}$$

$$relevant(wait).\ relevant(move).\ relevant(cross).$$

where the first rule informally states that the agent should consider performing any action relevant to the task (and exactly one at a time), while the second rule says that sequences of actions that do not lead our competitor to a win should be discarded (if our competitor's goal is indeed to win). Relation $relevant$ allows one to specify which actions are relevant to the task at hand, thus reducing the number of combinations that the reasoner considers.

Our competitor also needs to reason about his adversary's actions. For that purpose, our competitor possesses a model of the adversary's behavior.[6] The model is based on the following heuristics:

- Reach the finish line;
- At the finish line, if the opponent has already crossed, cross (as the race is over anyway);
- At the finish line, if riding the opponent's horse, cross right away;
- Otherwise, wait.

This model of the adversary's behavior could be more sophisticated – for example, it could include some level of non-determinism – but the simple model shown above is sufficient to solve the completion problem for this simple riddle. The heuristics are encoded by the set $\mathcal{P}_{adv}$ of triggers:[7]

$$my\_adversary(C_2) \leftarrow me(C_1), C_1 \neq C_2.$$

$$\begin{array}{l} o(move,C,S) \leftarrow \\ \qquad my\_adversary(C), \\ \qquad \neg h(at(C,finish\_line),S). \end{array}$$

[6] The model here is hard-coded, but could be learned, e.g. (Sakama 2005; Balduccini 2007).

[7] A discussion on the use of triggers can be found in the Conclusions section.

$$o(cross, C_1, S) \leftarrow$$
$$\quad my\_adversary(C_1),$$
$$\quad h(at(C_1, finish), S),$$
$$\quad \neg h(crossed(C_1), S),$$
$$\quad h(riding(C_1, H), S),$$
$$\quad owns(C_2, H), C1 \neq C_2.$$

$$o(cross, C_1, S) \leftarrow$$
$$\quad my\_adversary(C_1),$$
$$\quad h(at(C_1, finish), S),$$
$$\quad \neg h(crossed(C_1), S),$$
$$\quad h(crossed(C_2), S),$$
$$\quad competitor(C_2), C_1 \neq C_2.$$

$$\neg o(A_2, C, S) \leftarrow$$
$$\quad my\_adversary(C),$$
$$\quad o(A_1, C, S),$$
$$\quad A_2 \neq A_1.$$

$$o(wait, C, S) \leftarrow$$
$$\quad my\_adversary(C),$$
$$\quad \text{not } \neg o(wait, C, S).$$

Now let us see how the theory developed so far can be used to reason about the completion problem. Let $\mathcal{P}$ denote the set $\mathcal{P}_{me} \cup \mathcal{P}_{adv}$. It is not difficult to see that the theory

$$\sigma \cup F^a \cup \mathcal{AD} \cup \mathcal{W} \cup \mathcal{P}$$

is inconsistent. That is, a has no way of winning if his horse is faster. Let us now show that the result does not depend upon the horse's speed. Let $F^\vee$ denote the rule

$$faster(h(a)) \ \vee \ faster(h(b)).$$

which informally says that it is not known which horse is faster. The theory

$$\sigma \cup F^\vee \cup \mathcal{AD} \cup \mathcal{W} \cup \mathcal{P}$$

is still inconsistent. That is, a cannot win no matter whose horse is faster. Therefore, because our competitor is acting rationally, he is not going to take part in the race. Because the domain of the race is fully symmetrical, it is not difficult to see that b cannot win either, and therefore we will refuse to take part in the race as well.

However, that is not exactly what statement of the completion problem talks about. The statement in fact seems to suggest that, were the competitors to take part in the race (for example, because they hope for a mistake by the opponent), they would not be able to complete the race. To model that, we allow our competitor to have two goals with a preference relation among them: the goal to win, and the goal to at least not lose, where the former is preferred to the second. The second goal formalizes the strategy of waiting for a mistake by the adversary. To introduce the second goal and the preference, we obtain $\mathcal{P}'$ from $\mathcal{P}$ by adding to it the rules:

$$selected\_goal(win) \leftarrow$$
$$\quad \text{not } \neg selected\_goal(win).$$
$$\neg selected\_goal(win) \leftarrow$$
$$\quad selected\_goal(not\_lose).$$
$$\leftarrow lose(C), me(C), selected\_goal(not\_lose).$$
$$selected\_goal(not\_lose) \overset{+}{\leftarrow} .$$

The first rule says that our competitor's goal is to win, unless otherwise stated. The second rule says that one exception to this is if the selected goal is to not lose. The constraint says that, if the competitor's goal is to not lose, all action selections causing a loss must be discarded. The last rule says that our competitor may possibly decide to select the goal to just not lose, but only if strictly necessary (that is, if the goal of winning cannot be currently achieved).

Now, it can be shown that the theory

$$\sigma \cup F^\vee \cup \mathcal{AD} \cup \mathcal{W} \cup \mathcal{P}'$$

is consistent. One of its answer sets includes for example the atoms:

$$\begin{array}{ll}
\{faster(h(a)), & \\
\ o(wait, a, 0), & o(move, b, 0), \\
\ o(wait, a, 1), & o(move, b, 1), \\
\ o(move, a, 2), & o(wait, b, 2), \\
\ o(wait, a, 3), & o(wait, b, 3), \\
\ o(wait, a, 4), & o(wait, b, 4)\ \}
\end{array}$$

which represent the possibility that, if a's horse is faster, a and b will reach the finish line, and then wait there indefinitely. To confirm that the race will not be completed, let us introduce a set of rules $\mathcal{C}$ containing the definition of completion, together with a constraint that requires the race to be completed in any model of the underlying theory:

$$completed \leftarrow h(crossed(X), S).$$
$$\leftarrow \text{not } completed.$$

The first rule states that the race has been completed when one competitor has crossed the finish line (the result of the race at that point is fully determined). Because the theory

$$\sigma \cup F^\vee \cup \mathcal{AD} \cup \mathcal{W} \cup \mathcal{P}' \cup \mathcal{C}$$

is inconsistent, we can conclude formally that, if the competitors act rationally, they will not complete the race.

The last problem left to solve is answering the question "Which four wise words did the Wise Mountain Man speak?" In terms of our formalization, we need to find additional information, to be included in the theory developed so far, that allows to entail the completion of the race. Notice that, often, to solve a riddle one needs to revisit assumptions that were initially taken for granted. From a knowledge representation perspective, that means revisiting the defaults used in the encoding of the theory, and allowing the reasoner to select appropriate exceptions to the defaults.

The simple formalization given so far contains only one default, the rule for fluent $riding$ in σ:

$$h(riding(C, H), 0) \leftarrow$$
$$\quad owns(C, H),$$
$$\quad \text{not } \neg h(riding(C, H), 0).$$

To allow the reasoner to consider exceptions to this default, we add a cr-rule stating that a competitor may possibly ride the opponent's horse, although that should happen only if strictly necessary.

$$h(riding(C, H2), 0) \overset{+}{\leftarrow}$$
$$\quad owns(C, H1), horse(H2), H1 \neq H2.$$

We use a cr-rule, instead of a regular rule, to capture the intuition that the competitors will not normally switch horses. *Although for simplicity here we focus on a specific default, it is important to stress that this technique can be extended to the general case by writing the knowledge base so that each default is accompanied by a cr-rule allowing the reasoner to consider unexpected exceptions (but only if strictly necessary).* Let σ' be obtained from σ by adding to it the new cr-rule. It can be shown that the theory[8]

$$\sigma' \cup F^{\vee} \cup \mathcal{AD} \cup \mathcal{W} \cup \mathcal{P}$$

is consistent and its unique answer set contains:

$$\begin{array}{l}\{faster(h(b)),\\ \;h(riding(a,h(b)),0),\;\; h(riding(b,h(a)),0),\\ \;o(move,a,0),\quad o(move,b,0),\\ \;o(cross,a,1),\quad o(move,b,1),\\ \;o(wait,a,2),\quad o(cross,b,2),\\ \;o(wait,a,3),\quad o(wait,b,3),\\ \;o(wait,a,4),\quad o(wait,b,4)\;\}\end{array}$$

which encodes the answer that, if the competitors switch horses and the horse owned by b is faster, then a can win by immediately reaching the finish line and crossing it. In agreement with common-sense, a does not expect to win if the horse b owns is slower. On the other hand, it is not difficult to see that b will win in that case. That is, the race will be completed no matter what.

The conclusion obtained formally here agrees with the accepted solution of the riddle: "Take each other's horse."

Conclusions

In this paper we have described an exercise in the use of ASP for common-sense knowledge representation and reasoning, aimed at formalizing and reasoning about an easy-to-understand, but non-trivial riddle. One reason why we have selected this particular riddle, besides its high content of common-sense knowledge, is the fact that upon an initial analysis, it was unclear whether and how ASP or other formalisms could be used to solve it. Solving the riddle has required the combined use of some of the latest ASP techniques, including using consistency restoring rules to allow the reasoner to select alternative goals and to consider exceptions to the defaults in the knowledge base as a last resort, and has shown how ASP can be used for adversarial reasoning by employing it to encode a model of the adversary's behavior.

Another possible way of solving the riddle, not shown here for lack of space, consists in introducing a $switch\_horses$ action, made not relevant by default, but with the possibility to use it if no solution can be found otherwise. Such action would be *cooperative*, in the sense that both competitors would have to perform it together. However, as with many actions of this type in a competitive environment, rationally acting competitors are not always expected to agree to perform the action. An interesting continuation of our exercise will consist of an accurate formalization of this solution of the riddle, which we think may yield useful results in the formalization of sophisticated adversarial reasoning.

One last note should be made regarding the use of triggers to model the adversary's behavior. We hope the present paper has shown the usefulness of this technique and the substantial simplicity of implementation using ASP. This technique has limits, however, due to the fact that an a-priori model is not always available. Intuitively, it is possible to use ASP to allow a competitor to "simulate" the opponent's line of reasoning (e.g. by using choice rules). However, an accurate execution of this idea involves solving a number of non-trivial technical issues. We plan to expand on this topic in a future paper.

[8]The same answer is obtained by replacing $\mathcal{P}$ by $\mathcal{P}'$. However, doing that would require specifying preferences between the cr-rule just added and the cr-rule in $\mathcal{P}'$. To save space, we use $\mathcal{P}$ to answer the final question of the riddle.

References

Balduccini, M., and Gelfond, M. 2003. Logic Programs with Consistency-Restoring Rules. In Doherty, P.; McCarthy, J.; and Williams, M.-A., eds., *International Symposium on Logical Formalization of Commonsense Reasoning*, AAAI 2003 Spring Symposium Series, 9–18.

Balduccini, M. 2007. Learning Action Descriptions with A-Prolog: Action Language C. In Amir, E.; Lifschitz, V.; and Miller, R., eds., *Procs of Logical Formalizations of Commonsense Reasoning, 2007 AAAI Spring Symposium*.

Brewka, G.; Niemela, I.; and Syrjanen, T. 2004. Logic Programs wirh Ordered Disjunction. 20(2):335–357.

Gelfond, M., and Lifschitz, V. 1991. Classical negation in logic programs and disjunctive databases. *New Generation Computing* 365–385.

Gelfond, M., and Lifschitz, V. 1998. Action Languages. *Electronic Transactions on AI* 3(16).

Gelfond, M. 2002. Representing Knowledge in A-Prolog. In Kakas, A. C., and Sadri, F., eds., *Computational Logic: Logic Programming and Beyond, Essays in Honour of Robert A. Kowalski, Part II*, volume 2408, 413–451. Springer Verlag, Berlin.

Hayes, P. J., and McCarthy, J. 1969. Some Philosophical Problems from the Standpoint of Artificial Intelligence. In Meltzer, B., and Michie, D., eds., *Machine Intelligence 4*. Edinburgh University Press. 463–502.

Marek, V. W., and Truszczynski, M. 1999. *Stable models and an alternative logic programming paradigm*. The Logic Programming Paradigm: a 25-Year Perspective. Springer Verlag, Berlin. 375–398.

Mellarkod, V. S.; Gelfond, M.; and Zhang, Y. 2008. Integrating Answer Set Programming and Constraint Logic Programming. *Annals of Mathematics and Artificial Intelligence*. (to appear).

Sakama, C. 2005. Induction from answer sets in nonmonotonic logic programs. *ACM Transactions on Computational Logic* 6(2):203–231.