Solution to the Open Problem: AnsProlog encoding of win and lose (Chs 2,4)

The goal here is to develop an AnsProlog program that has the same characterization of winning and losing as the following logic program with respect to the well-founded semantics.

$win(X) \leftarrow move(X,Y), \textbf{not}\ win(Y).$
an arbitrary set of 'move' facts.

We will now give another characterization of this which does not appeal to the well-founded semantics.

1. We have a set of nodes.

2. We are given a set of facts about a binary predicate *move*. Intuitively, $move(a,b)$ means that there is an available move from node a to node b.

3. A strategy S is a function from nodes to nodes such that $S(X) = Y$ only if move(X,Y) is true.

4. Given two strategies S_1 and S_2, and a node a we define the trajectory followed by alternatively applying S_1 and S_2 as: $traj(a, S_1, S_2) = X_0^{a,S_1,S_2} X_1^{a,S_1,S_2} \ldots$, where

   $$\begin{aligned} X_0^{a,S_1,S_2} &= a \\ X_{k+1}^{a,S_1,S_2} &= S_1(X_k^{a,S_1,S_2}) \text{ if } k \text{ is even} \\ &= S_2(X_k^{a,S_1,S_2}) \text{ if } k \text{ is odd} \end{aligned}$$

5. The length of a trajectory $X_0 X_1 \ldots X_k$ is k.

6. A node a is said to be a winning node if there exists S_1 such that for all S_2 the sequence $traj(a, S_1, S_2)$ terminates and its length is odd.

7. A node a is said to be a losing node if for all S_1 there exists S_2 such that the sequence $traj(a, S_1, S_2)$ terminates and its length is even.

Question 1: Write an AnsProlog program Π whose answer set semantics corresponds to the *win* and *lose* above. (The solution to this is known.)

Question 2: Write an AnsProlog program Π which has a unique answer set corresponding to the *win* and *lose* above. (To the best of my knowledge, this is an open problem.)

Acknowledgement: Bertram Ludaescher first posed this question to me in July 2002. Later in August 2002 I discussed this with Vladimir Lifschitz and his group in Austin. The above formulation was developed during my presentation in Austin.

A solution to Question 2 has been given by Carlos Damasio (cd@di.fct.unl.pt) on 1/22/04. See the next page for the solution.

The Solution

The solution is based on simulating the well-founded semantics. Recall that answer sets (stable models) are defined as fixpoints of an operator F. The well-founded semantics is given as the pair $\langle lfp(F^2), HB \setminus gfp(F^2)\rangle$, and $lfp(F^2)$ can be computed iteratively by applying F^2 repeatedly starting from $\emptyset$ until a fixpoint is reached. $gfp(F^2)$ is obtained by applying F to $lfp(F^2)$.

The Solution consists of the following:

1. Defining *node*:

   $node(X) \leftarrow move(X,Y).$
   $node(Y) \leftarrow move(X,Y).$

2. Defining *int*:

   $int(0). \ldots int(max).$

   where max is an even number large enough that F^2 reaches the least fixpoint by then.

3. Defining *win_aux*: This predicate computes *win* at different iterations of F.

   $win\_aux(X, S+1) \leftarrow int(S), move(X,Y),$ **not** $win\_aux(Y,S).$

4. Defining *win*, $\neg win$, and *draw*:

   $win(X) \leftarrow win\_aux(X, max).$

   $\neg win(X) \leftarrow node(X),$ **not** $win\_aux(X, max-1).$

   $draw(X) \leftarrow node(X),$ **not** $win(X),$ **not** $\neg win(X).$

Following is the solution in Smodels.

```
#maxint=20.

node(a;b;c;d;e;f;g)

move(a,b).

move(b,c).

move(b,d).

move(b,g).

move(c,c).

move(d,f).

move(f,d).

winaux( X, S ) :- #int(S), move(X,Y), #succ(S1,S), not winaux(Y,S1).

win(X)  :- winaux(X,#maxint).

-win(X)  :- node(X), #succ(S1,#maxint), not winaux(X, S1).

draw(X) :- node(X), not win(X), not -win(X).
```

It is fundamental that #maxint is an even number. When S is odd steps we compute F by predicate winaux(_,S) and in even steps F^2, and thats it.

This technique can be used in general for computing the WFM by ASP.

You can also include some rules for detecting the fixpoint of F^2, like the ones below. In this way you do not need anymore to enforce #maxint to be an even number.

It just has to be big enough. I think that it is possible to change winaux/2 in order to stop after end. That is not difficult, after having end.

```
win(X)  :- end(S), winaux(X,S).

-win(X)  :- end(S), #succ(S1,S), node(X), not winaux(X, S1).

draw(X) :- node(X), not win(X), not -win(X).

end(S) :- #int(S1), S1 >= 1, S = 2 * S1, not change(S), not
ended_before(S).

ended_before (S) :- end(S1), #int(S), S1 < S .

change(S) :- winaux(X,S), #succ(S1,S), #succ(S2,S1), not
winaux(X,S2).
```