'Jealous Husbands' Puzzle

The 'Jealous Husbands' puzzle is a logic puzzle that is not easily solvable by humans and is useful as an AI 'toy' problem. It is a type of river-crossing puzzle like the 'Missionaries and Cannibals' puzzle. The puzzle consists of a few rules. There are m-couples (usually 3 couples) of husband and wives that must cross a river; however there is only one boat. The boat can carry at most n passengers (usually 2). Since the husbands are jealous, they will not leave their wife alone on either side of the river if there are any other men there. The boat also cannot cross the river by itself, since there would be no one to control it. The problem is to find a sequence of river crossings that follows the rules of the puzzle and ends with all couples on the opposite bank. There are some variations of the puzzle, such as altering the number of husband and wife pairs, or changing the capacity of the boat, as well as adding an island in the middle that they may stop on temporarily. Answer set programming should be able to easily solve this puzzle. There are also some variations of the 'Missionaries and Cannibals' puzzle from John McCarty's “Elaboration Tolerance” that can be used for the 'Jealous Husbands' puzzle as well to benchmark how “elaboration tolerant” a formalism is; that is, whether an elaboration be added to the existing formalization just by adding more rules and not changing previous ones. The variations I have considered are adding an island in the middle of the river, a person that takes up the entire room of the boat, a person that can walk on water, having only one husband and one wife know how to operate the boat, and having a wife that is also jealous.

My approach to solving the puzzle in ASP was to convert the rules of the puzzle into logical rules using the “generate-define-test” strategy. First, three variable domains needed to be defined: one for time, one for identifying which husband-wife pair, and one to flag whether the person is the husband or wife.

$$\text{Time} = \{1, \ldots, t\}, \quad \text{Couple} = \{1, \ldots, c\}, \quad \text{HusbandWife} = \{\text{husband}, \text{wife}\}$$

Generating the solution consisted of creating a set of moves such that at every turn at least one person

was moved (since the boat cannot move itself) and at most the capacity of the boat. Or in logical notation:

$$\forall\, T \in \text{Time} \quad 1 \leq \{\, \text{move}(T, C, H) : C \in \text{Couple}, H \in \text{HusbandWife}\}^{c} \leq \text{cap}$$

Next was defining that all people started at the first bank:

$$\forall\, C \in \text{Couple}, \forall\, H \in \text{HusbandWife}\ (\ \text{shore1}\ (1, C, H)\ )$$

Then the rules for moving between the two shores had to be defined. If a person was at the first shore at time T and was moved during time T, then that person had to be on the opposite shore at time T+1. Similarly for anybody moved from the opposite shore they had to end up on the first shore. Expressing this in logical notation:

$$\text{shore1}\ (T, C, H) \wedge \text{move}\ (T, C, H) \rightarrow \sim\text{shore1}(T+1, C, H)$$

$$\sim\text{shore1}\ (T, C, H) \wedge \text{move}\ (T, C, H) \rightarrow \text{shore1}(T+1, C, H)$$

After this, strong negation had to be used to define that if someone is not moved then they must stay where they are at currently for the next turn. If a person is at the first shore at time T and there is no evidence that they are not at the opposite shore at time T+1, then they must have remained at the first shore and are still there at time T+1. Similarly for someone on the opposite shore, if there is no evidence they are on the first shore at the next move, then they must still be on the opposite shore. Implementing these rules into mathematical logic:

$$\text{shore1}\ (T, C, H) \wedge \neg(\sim\text{shore1}(T+1, C, H)\ ) \rightarrow \text{shore1}\ (T+1, C, H)$$

$$\sim\text{shore1}\ (T, C, H) \wedge \neg(\text{shore1}(T+1, C, H)\ ) \rightarrow \sim\text{shore1}\ (T+1, C, H)$$

The rules of the jealous husbands also need to be included. Any sequence of moves that ends up with the wife from couple C1 and a husband from couple C2 on the same shore at the same time T with the husband from couple C1 on the opposite shore at time T could not be a valid solution. Since a person cannot be on two different shores at the same time, it does not matter whether C2 is the same as C1 or not so that does not have to be checked. Creating these rules in mathematical logical expressions

yields:

← shore1(T, C1, wife) ^ shore1(T, C2, husband) ^ ~shore1(T, C1, husband)

← ~shore1(T, C1, wife) ^ ~shore1(T, C2, husband) ^ shore1(T, C1, husband)

The behavior of the boat also needed to be defined. The boat can only travel one direction at a time, so it cannot move someone from the first shore and someone from the second shore during the same turn. It also can only start from where it ended at the previous turn, meaning if it ended at the first shore on turn T, then it must leave the first shore on turn T+1, meaning that everyone it moves must be at first shore. These two rules can be combined into one rule, since if it ended at one of the shores at time T, it can only move people from that shore at turn T+1, so there is no way it could move people from two different shores anyways. Since the boat starts at the first shore during the first turn and alternates every turn, then it must go from the first shore to the second shore on odd turns and from the second shore to the first shore during even turns. So therefore any sequence of moves that violates the rules must be thrown out. Expressing this in mathematical logic:

← move(T, C, H) ^ ~shore1(T, C, H) ^ T % 2 == 1

← move(T, C, H) ^ shore1(T, C, H) ^ T % 2 == 0

Finally, any sequence of moves that does not end with all of the people on the other shore after the last move t must also be thrown out:

← not ~shore1(t+1, C, H)

Converting these logical expressions into ASP was fairly straight forward, and the resulting code is contained in Appendix A1. The solution for the traditional problem with 3 husband-wife pairs and a boat capacity of 2 is also shown in Appendix A2.

The first variation I considered was the puzzle that includes an island between the two shores. My original formalization was not elaboration tolerant for this particular elaboration. Originally I had only a single predicate that determined whether the person was on the first shore or the ending shore at

a particular time. There was no way to include information about the island without changing my previous rules. Also, previously the boat only had one direction to go at each turn, which was the opposite direction of where it moved at the previous turn. Now the boat has two possible destinations at each turn. This also meant I had to change even more rules to formalize this variation. First I had to add a new domain for the possible positions of a person:

Pos = {shore1, island, shore2}

Then, a person can only be at one position at every turn:

$\forall T \in Time, \forall C \in Couple, \forall H \in HusbandWife, 1 \leq \{ppos(T,C,H,P) : P \in Pos\}^c \leq 1$

The predicate for moving also had to be changed to include where the person was moving to, since in the previous formalization when a person was moved they were just changed sides. Now a person has more than one possible destination when they are moved:

$\forall T \in Time \quad 1 \leq \{ move(T, C, H, P) : C \in Couple, H \in HusbandWife, P \in Pos\}^c \leq 2$

Next, everyone had to be defined to be at the first shore on the first turn, and to not include any answer sets that did not have everyone on the second shore on the final state t+1:

ppos(T,C,H,shore1)

$\leftarrow$ not ppos(t+1, C, H, shore2)

The rules for moving had to be defined as well. To avoid redundancy, a person cannot be moved to where they are already are at. Two people also cannot be moved at the same time to two different places. Finally, a person cannot be moved if they are not where the boat ended up from the previous turn. Expressing these rules in logical notation:

$\leftarrow$ move(T,C,H,P) ^ ppos(T,C,H,P).

$\leftarrow$ move(T,C1,H1,P1) ^ move(T,C2,H2,P2) ^ $P1 \neq P2$.

$\leftarrow$ move(T,C1,H1,P1) ^ ppos(T,C1,H1,P2) ^ move(T-1, C2,H2,P3) ^ $P2 \neq P3$.

Next was defining what the move action does. If a person is moved to position P, then the next turn

they will be at that position. Also, a person cannot change positions from one turn to the next if they were not moved.

ppos (T+1,C,H, P) ← move(T,C,H,P).

← ppos(T,C,H,P1) ^ ppos(T+1,C,H,P2) ^ not move(T,C,H,P2), P1 ≠ P2.

Finally, the rule that a wife cannot be left on the same position with another husband unless her husband is on the same position as well. As in the previous formalization, it does not need to be checked that the husband that the wife is with is her husband or not, since her husband cannot be in two places at once:

← ppos(T,C1,wife, P1) ^ ppos(T,C2,husband,P1) ^ ppos(T,C1,husband,P2) ^ P1 ≠ P2.

From this, it was straight-forward to implement into ASP. However, the multiple positions a person can be at as well as the multiple possible directions for the boat to travel causes the search space to grow enormously. As a result, finding solutions for this variation takes a considerably longer amount of time as compared to the original problem. For the optimal solution for three couples, it required roughly 20 seconds to find a solution, compared to about a fifth of a second previously. When the amount of couples is increased to four, a non-optimal solution for forty turns was found in about 25 seconds, but an optimal solution for sixteen turns did not yield results even after twenty minutes. The completed ASP code is included in Appendix A3, and the results in A4.

The next elaboration I considered was including the rule that people who are very big take up the entire boat, and therefore cannot be in the boat with anyone else. This turned out to be very simple to add to the original formalization, and therefore the original formalization was elaboration tolerant with respect to this elaboration. All that had to be added was one rule to explain that a big person cannot be in the boat with anyone else:

← move(T,C1,H1) ^ move(T,C2,H2) ^ b(C1,H1) ^ C1 ≠ C2

← move(T,C1,H1) ^ move(T,C2,H2) ^ b(C1,H1) ^ H1 ≠ H2

Where b(C,H) is the input predicate explaining which person is big. This variation was still solvable if a wife was the big person, but could not be solved if the husband was the big person. In the first case, it was optimally solvable in fifteen turns. The completed ASP code and results are included in Appendix A5.

The next variation was considering a person that could walk on water, and therefore did not have to ride in the boat. This also turned out to be very simple to add to the original formalization, and therefore the original formalization was also elaboration tolerant with respect to this elaboration. First was to generate the walk actions. At any time, a person can decide to walk across the water to the other side:

$$\forall\, T \in Time \quad \{\, walk(T, C, H) : C \in Couple, H \in HusbandWife\}^{c}$$

However, only people who can walk on water can actually walk over the water:

$$\leftarrow walk\ (T,C,H)\ \wedge\ not\ w(C,H).$$

Where w(C,H) is the input defining which person can walk on water. Finally, the definition for the walk action had to be defined. These rules are similar to the move actions, where if someone walks, they end up on the opposite shore on the next turn:

$$shore1(T+1,C,H) \leftarrow \sim shore1(T,C,H)\ \wedge\ walk(T,C,H).$$

$$\sim shore1(T+1,C,H) \leftarrow shore1(T,C,H)\ \wedge\ walk(T,C,H).$$

This elaboration was optimally solvable in seven turns regardless of whether the person that could walk on water was a husband or a wife. The complete ASP code as well as the results are included in Appendix A6.

The last of John McCarthy's elaborations I considered was the rower elaboration. In this variation, only one wife and one husband know how to row the boat. Therefore anytime the boat is used, one of them must be in it. After considering a few different implementations, I found one that also turned out to be very simple to add to the original formalization, and therefore the original

formalization was also elaboration tolerant with respect to this elaboration. For this variation all that was added to the original formalization was another rule that said that at every turn, at least one rower must be in the boat:

$$\forall\, T \in Time \quad 1 \leq \{\, move(T, C, H) : C \in Couple, H \in HusbandWife, Rower(C,H)\}^{c}$$

However, after adding this rule there turned out to be no solution with just one husband and one wife being able to row. Adding in a third rower did have a solution. If the third rower was a wife, then it was optimally solvable in eleven turns, otherwise it was solvable in thirteen turns. The ASP code and solution are included in Appendix A7.

Finally, I created an elaboration of my own specifically for the 'Jealous Husbands' puzzle. In this variation, there is one wife that is jealous. Therefore, similar to the husbands, she will not let her husband be alone with any other woman. This also turned out to be very simple to add to the original formalization, and therefore the original formalization was also elaboration tolerant with respect to this elaboration. All that was added was two rules similar to the two rules for defining the jealous husbands:

$$\leftarrow shore1(T,C1,wife) \wedge shore1(T,C2,husband) \wedge \sim shore1(T,C2,wife) \wedge j(C2).$$

$$\leftarrow \sim shore1(T,C1,wife) \wedge \sim shore1(T,C2,husband) \wedge shore1(T,C2,wife) \wedge j(C2).$$

Where j(C) is the input of which wife is jealous. This variation was solvable, and could be optimally solved in thirteen moves. The ASP code and solution for this variation of the puzzle is included in Appendix A8.

Aside from the island elaboration, all the variants were easily added into my original formalization. Therefore, the original formalization was fairly elaboration tolerant. The island variation was more complex to integrate into my existing solution because I had not considered that there would be more than one destination to move to each turn and I determined a person's position by only saying they were either on the starting shore or not at every turn. Therefore, there would not be a

way to add an elaboration that had multiple positions without changing existing rules. If I had considered available destinations for the boat and various positions for a person originally, it might have been more elaboration tolerant.

Appendix

A1. ASP code for original problem.

```
time(1..t).
couple(1..c).
husband(husband;wife).

#domain time(T1).
#domain couple(C1;C2).
#domain husband(H1;H2).

1 {move(T0,C0,H) : couple(C0): husband(H) } b :- time(T0).

shore1(1,C1,H1).
:- not -shore1(t+1,C1,H1).

shore1(T1+1,C1,H1) :- -shore1(T1, C1, H1), move(T1, C1, H1).
-shore1(T1+1,C1,H1) :- shore1(T1, C1, H1), move(T1, C1, H1).

shore1(T1+1,C1,H1) :- shore1(T1, C1, H1), not -shore1(T1+1,C1,H1).
-shore1(T1+1,C1,H1) :- -shore1(T1, C1, H1), not shore1(T1+1,C1,H1).

:- shore1(T1,C1,wife), shore1(T1,C2,husband), -shore1(T1,C1,husband).
:- -shore1(T1,C1,wife), -shore1(T1,C2,husband), shore1(T1,C1,husband).

:- move(T1,C1,H1), T1 mod 2 == 1, -shore1(T1,C1,H1).
:- move(T1,C1,H1), T1 mod 2 == 0, shore1(T1,C1,H1).

hide.

show move(_,_,_).
```

A2. Results from smodels for 3 pairs, capacity of 2, and 11 turns.

```
lparse --true-negation -c c=3 -c t=11 -c b=2 jealous.lp | smodels
smodels version 2.26. Reading...done
Answer: 1
Stable Model: move(1,2,1) move(1,2,2) move(2,2,1) move(3,3,2) move(3,1,2)
move(4,1,2) move(5,3,1) move(5,2,1) move(6,3,1) move(6,3,2) move(7,3,1) move(7,1,1)
move(8,2,2) move(9,2,2) move(9,1,2) move(10,3,1) move(11,3,1) move(11,3,2)
True
Duration 0.171
Number of choice points: 482
Number of wrong choices: 473
Number of atoms: 249
Number of rules: 683
Number of picked atoms: 36710
Number of forced atoms: 1877
Number of truth assignments: 175174
Size of searchspace (removed): 174 (12)
```

A3. ASP code for island variation

```
turn(1..t).
time(1..t+1).
pos(shore1;island;shore2).
couple(1..c).
husband(husband;wife).

#domain time(T1).
#domain couple(C1;C2).
#domain husband(H1;H2).
#domain pos(P1;P2;P3).


1 {move(T0,C0,H, P) : couple(C0): husband(H):pos(P) } 2 :- turn(T0).
1 {ppos(T0,C0,H0,P) : pos(P)}1 :- time(T0), couple(C0), husband(H0).

ppos(1,C1,H1,shore1).
:- not ppos(t+1,C1,H1,shore2).

:- move(T1,C1,H1,P1), ppos(T1,C1,H1,P1).
:- move(T1,C1,H1,P1), move(T1,C2,H2,P2), P1 != P2.
:- move(T1,C1,H1,P1), ppos(T1,C1,H1,P2), move(T1-1,C2,H2,P3), P2 != P3.


ppos(T1+1,C1,H1,P1) :- move(T1,C1,H1,P1).

:- ppos(T1,C1,H1,P1), ppos(T1+1,C1,H1,P2), not move(T1,C1,H1,P2), P1 != P2.

:- ppos(T1,C1,wife,P1), ppos(T1,C2,husband,P1), ppos(T1,C1,husband,P2), P1 != P2.

hide.

show move(_,_,_,_).
```

A4. Results from island variation

Three couples, optimal solution:

```
lparse -c c=3 -c t=11 jealous_island.lp | smodels
smodels version 2.26. Reading...done
Answer: 1
Stable Model: move(1,3,husband,island) move(1,3,wife,island)
move(2,3,husband,shore1) move(3,1,husband,shore2) move(3,1,wife,shore2)
move(4,1,husband,shore1) move(5,3,husband,shore2) move(5,1,husband,shore2)
move(6,3,husband,shore1) move(7,3,husband,shore2) move(7,2,husband,shore2)
move(8,3,husband,island) move(9,3,husband,shore2) move(9,3,wife,shore2)
move(10,1,wife,shore1) move(11,2,wife,shore2)
move(11,1,wife,shore2)
True
Duration 21.827
Number of choice points: 2847
Number of wrong choices: 2836
Number of atoms: 687
Number of rules: 6075
Number of picked atoms: 1060890
Number of forced atoms: 46643
Number of truth assignments: 13520182
Size of searchspace (removed): 360 (0)
```

Four couples, non-optimal solution:

```
lparse -c c=4 -c t=40 jealous_island.lp | smodels
smodels version 2.26. Reading...done
Answer: 1
Stable Model: move(1,4,husband,shore2) move(1,4,wife,shore2)
move(2,4,husband,shore1) move(3,3,wife,shore2) move(3,1,wife,shore2)
move(4,3,wife,shore1) move(5,4,husband,island) move(5,1,husband,island)
move(6,1,husband,shore1) move(7,1,husband,island) move(8,4,husband,shore2)
move(8,1,husband,shore2) move(9,1,husband,island) move(9,1,wife,island)
move(10,1,husband,shore2) move(11,4,husband,shore1) move(11,1,husband,shore1)
move(12,3,wife,shore2) move(12,2,wife,shore2) move(13,3,wife,shore1)
move(14,4,husband,island) move(14,1,husband,island) move(15,1,husband,shore1)
move(15,1,wife,shore1) move(16,3,husband,island) move(16,3,wife,island)
move(17,4,husband,shore1) move(18,1,wife,shore2) move(19,2,wife,shore1)
move(20,4,husband,shore2) move(20,1,husband,shore2) move(21,1,husband,island)
move(21,1,wife,island) move(22,3,husband,shore2) move(22,3,wife,shore2)
move(23,4,husband,island) move(23,4,wife,island) move(24,1,husband,shore1)
move(24,1,wife,shore1) move(25,1,husband,island) move(25,1,wife,island)
move(26,4,husband,shore2) move(26,1,husband,shore2) move(27,3,husband,shore1)
move(27,3,wife,shore1) move(28,3,husband,shore2) move(28,2,husband,shore2)
move(29,4,husband,island) move(2
9,1,husband,island) move(30,1,husband,shore2) move(30,1,wife,shore2)
move(31,2,husband,island) move(32,4,husband,shore2) move(32,2,husband,shore2)
move(33,1,wife,shore1) move(34,3,wife,shore2) move(34,2,wife,shore2)
move(35,4,husband,island) move(36,4,wife,shore1) move(37,4,wife,island)
move(38,4,husband,shore2) move(38,4,wife,shore2) move(39,3,wife,shore1)
move(40,3,wife,shore2) move(40,1,wife,shore2)
True
Duration 26.937
```

```
Number of choice points: 1170
Number of wrong choices: 1109
Number of atoms: 2945
Number of rules: 32451
Number of picked atoms: 713216
Number of forced atoms: 26983
Number of truth assignments: 10890023
Size of searchspace (removed): 1872 (0)
```

A5. Complete ASP code including “big” people.

```
time(1..t).
couple(1..3).
husband(husband;wife).

#domain time(T1).
#domain couple(C1;C2).
#domain husband(H1;H2).

1 {move(T0,C0,H) : couple(C0): husband(H) } 2 :- time(T0).

shore1(1,C1,H1).
:- not -shore1(t+1,C1,H1).

shore1(T1+1,C1,H1) :- -shore1(T1, C1, H1), move(T1, C1, H1).
-shore1(T1+1,C1,H1) :- shore1(T1, C1, H1), move(T1, C1, H1).

shore1(T1+1,C1,H1) :- shore1(T1, C1, H1), not -shore1(T1+1,C1,H1).
-shore1(T1+1,C1,H1) :- -shore1(T1, C1, H1), not shore1(T1+1,C1,H1).

:- shore1(T1,C1,wife), shore1(T1,C2,husband), -shore1(T1,C1,husband).
:- -shore1(T1,C1,wife), -shore1(T1,C2,husband), shore1(T1,C1,husband).

:- move(T1,C1,H1), T1 mod 2 == 1, -shore1(T1,C1,H1).
:- move(T1,C1,H1), T1 mod 2 == 0, shore1(T1,C1,H1).

:- move(T1,C1,H1), move(T1,C2,H2), b(C1,H1), C1 != C2.
:- move(T1,C1,H1), move(T1,C2,H2), b(C1,H1), H1 != H2.

hide.

show move(_,_,_).
```

Solution, where input1.txt consists of single predicate b(1,wife):

```
lparse --true-negation -c t=15 input1.txt jealous_big.lp | smodels
smodels version 2.26. Reading...done
Answer: 1
Stable Model: move(1,2,husband) move(1,2,wife) move(2,2,husband) move(3,1,wife)
move(4,2,wife) move(5,3,wife) move(5,2,wife) move(6,2,wife) move(7,3,husband)
move(7,1,husband) move(8,3,husband) move(8,3,wife) move(9,3,husband)
move(9,2,husband) move(10,1,wife) move(11,3,wife) move(11,2,wife) move(12,2,wife)
move(13,1,wife) move(14,3,wife) move(15,3,wife) move(15,2,wife)
True
Duration 0.405
Number of choice points: 502
Number of wrong choices: 494
Number of atoms: 334
Number of rules: 1029
Number of picked atoms: 78134
Number of forced atoms: 3990
Number of truth assignments: 509769
Size of searchspace (removed): 234 (12)
```

A6. ASP Code for water walking variation

```
time(1..t).
couple(1..3).
husband(husband;wife).

#domain time(T1).
#domain couple(C1;C2).
#domain husband(H1;H2).

1 {move(T0,C0,H) : couple(C0): husband(H) } 2 :- time(T0).
{walk(T0,C0,H) : couple(C0): husband(H) } :- time(T0).

:- walk(T1,C1,H1), not w(C1,H1).

shore1(T1+1,C1,H1) :- -shore1(T1, C1, H1), walk(T1, C1, H1).
-shore1(T1+1,C1,H1) :- shore1(T1, C1, H1), walk(T1, C1, H1).

shore1(1,C1,H1).
:- not -shore1(t+1,C1,H1).

shore1(T1+1,C1,H1) :- -shore1(T1, C1, H1), move(T1, C1, H1).
-shore1(T1+1,C1,H1) :- shore1(T1, C1, H1), move(T1, C1, H1).

shore1(T1+1,C1,H1) :- shore1(T1, C1, H1), not -shore1(T1+1,C1,H1).
-shore1(T1+1,C1,H1) :- -shore1(T1, C1, H1), not shore1(T1+1,C1,H1).

:- shore1(T1,C1,wife), shore1(T1,C2,husband), -shore1(T1,C1,husband).
:- -shore1(T1,C1,wife), -shore1(T1,C2,husband), shore1(T1,C1,husband).

:- move(T1,C1,H1), T1 mod 2 == 1, -shore1(T1,C1,H1).
:- move(T1,C1,H1), T1 mod 2 == 0, shore1(T1,C1,H1).

hide.
show move(_,_,_).
show walk(_,_,_).
```

Solution, with Wife #1 able to water walk:

```
lparse --true-negation -c t=7 input2.txt jealous_walk.lp | smodels
smodels version 2.26. Reading...done
Answer: 1
Stable Model: walk(2,1,wife) walk(7,1,wife) move(1,2,wife) move(7,2,wife)
move(7,3,wife) move(3,1,husband) move(1,2,husband) move(3,2,husband)
move(5,2,husband) move(5,3,husband) move(6,1,wife) move(4,2,wife) move(2,2,husband)
move(4,2,husband)
True
Duration 0.061
Number of choice points: 11
Number of wrong choices: 4
Number of atoms: 208
Number of rules: 570
Number of picked atoms: 1098
Number of forced atoms: 36
Number of truth assignments: 4350
Size of searchspace (removed): 109 (12)
```

A7. Rower variant

```
time(1..t).
couple(1..3).
husband(husband;wife).

#domain time(T1).
#domain couple(C1;C2).
#domain husband(H1;H2).

1 {move(T0,C0,H) : couple(C0): husband(H) } 2 :- time(T0).

1 {move(T0,C0,H0) : couple(C0):husband(H0):r(C0,H0)} :- time(T0).

shore1(1,C1,H1).
:- not -shore1(t+1,C1,H1).

shore1(T1+1,C1,H1) :- -shore1(T1, C1, H1), move(T1, C1, H1).
-shore1(T1+1,C1,H1) :- shore1(T1, C1, H1), move(T1, C1, H1).

shore1(T1+1,C1,H1) :- shore1(T1, C1, H1), not -shore1(T1+1,C1,H1).
-shore1(T1+1,C1,H1) :- -shore1(T1, C1, H1), not shore1(T1+1,C1,H1).

:- shore1(T1,C1,wife), shore1(T1,C2,husband), -shore1(T1,C1,husband).
:- -shore1(T1,C1,wife), -shore1(T1,C2,husband), shore1(T1,C1,husband).

:- move(T1,C1,H1), T1 mod 2 == 1, -shore1(T1,C1,H1).
:- move(T1,C1,H1), T1 mod 2 == 0, shore1(T1,C1,H1).

hide.

show move(_,_,_).
```

Solution, Rowers: Husband #1, Wife #2, Wife #3

```
lparse --true-negation -c t=11 input3.txt jealous_rower.lp | smodels
smodels version 2.26. Reading...done
Answer: 1
Stable Model: move(1,1,husband) move(1,1,wife) move(2,1,husband) move(3,3,wife)
move(3,2,wife) move(4,3,wife) move(5,2,husband) move(5,1,husband) move(6,1,husband)
move(6,1,wife) move(7,3,husband) move(7,1,husband) move(8,2,wife) move(9,3,wife)
move(9,2,wife) move(10,1,husband) move(11,1,husband) move(11,1,wife)
True
Duration 0.625
Number of choice points: 3037
Number of wrong choices: 3030
Number of atoms: 263
Number of rules: 719
Number of picked atoms: 201584
Number of forced atoms: 8185
Number of truth assignments: 941058
Size of searchspace (removed): 174 (12)
```

Solution, Rowers: Husband #1, Wife #2, Husband #3.

```
lparse --true-negation -c t=13 input4.txt jealous_rower.lp | smodels
smodels version 2.26. Reading...done
Answer: 1
Stable Model: move(1,1,husband) move(1,1,wife) move(2,1,husband) move(3,3,wife)
move(3,2,wife) move(4,2,wife) move(5,3,husband) move(5,1,husband) move(6,1,husband)
move(6,1,wife) move(7,2,husband) move(7,2,wife) move(8,3,husband) move(8,3,wife)
move(9,3,husband) move(9,1,husband) move(10,2,wife) move(11,3,wife) move(11,2,wife)
move(12,2,wife) move(13,2,wife) move(13,1,wife)
True
Duration 0.281
Number of choice points: 1056
Number of wrong choices: 1047
Number of atoms: 307
Number of rules: 845
Number of picked atoms: 80004
Number of forced atoms: 3086
Number of truth assignments: 366531
Size of searchspace (removed): 210 (12)
```

A8. Jealous wife variation

```
time(1..t).
couple(1..3).
husband(husband;wife).

#domain time(T1).
#domain couple(C1;C2).
#domain husband(H1;H2).

1 {move(T0,C0,H) : couple(C0): husband(H) } 2 :- time(T0).

shore1(1,C1,H1).
:- not -shore1(t+1,C1,H1).

shore1(T1+1,C1,H1) :- -shore1(T1, C1, H1), move(T1, C1, H1).
-shore1(T1+1,C1,H1) :- shore1(T1, C1, H1), move(T1, C1, H1).

shore1(T1+1,C1,H1) :- shore1(T1, C1, H1), not -shore1(T1+1,C1,H1).
-shore1(T1+1,C1,H1) :- -shore1(T1, C1, H1), not shore1(T1+1,C1,H1).

:- shore1(T1,C1,wife), shore1(T1,C2,husband), -shore1(T1,C1,husband).
:- -shore1(T1,C1,wife), -shore1(T1,C2,husband), shore1(T1,C1,husband).

:- shore1(T1,C1,wife), shore1(T1,C2,husband), -shore1(T1,C2,wife), j(C2).
:- -shore1(T1,C1,wife), -shore1(T1,C2,husband), shore1(T1,C2,wife), j(C2).

:- move(T1,C1,H1), T1 mod 2 == 1, -shore1(T1,C1,H1).
:- move(T1,C1,H1), T1 mod 2 == 0, shore1(T1,C1,H1).

hide.

show move(_,_,_).
```

Solution, Wife #1 is jealous:

```
lparse --true-negation -c t=13 input5.txt jealous_wife.lp | smodels
smodels version 2.26. Reading...done
Answer: 1
Stable Model: move(1,2,husband) move(1,2,wife) move(2,2,husband) move(3,3,wife)
move(3,1,wife) move(4,1,wife) move(5,3,husband) move(5,2,husband) move(6,2,husband)
move(6,2,wife) move(7,1,husband) move(7,1,wife) move(8,3,husband) move(8,3,wife)
move(9,3,husband) move(9,2,husband) move(10,1,wife) move(11,2,wife) move(11,1,wife)
move(12,3,husband) move(13,3,husband) move(13,3,wife)
True
Duration 0.999
Number of choice points: 4114
Number of wrong choices: 4105
Number of atoms: 292
Number of rules: 882
Number of picked atoms: 326232
Number of forced atoms: 14782
Number of truth assignments: 1523918
Size of searchspace (removed): 210 (12)
```