SAT Planning in Description Logics: Solving the Classical “Wolf Goat Cabbage” Riddle

Michael Wessel
2014 – 06 - 30
Wolf – Goat (Sheep) – Cabbage Riddle

A shepherd (= “ferryman” in the following), wolf, goat, and cabbage want to cross the river.

The boat only hosts two.

The wolf and goat (sheep?) cannot be left alone together.

The cabbage and goat cannot be left alone.

How can they safely cross the river?
Sat Plan

• Is a planning problem
• Reduction of planning to SAT (propositional case)

  – SAT ( plan \( \land \) start \( \land \) goal \( \land \) actions \( \land \) … ) = true
    iff
    Plan = <step_1 = start, step_2, …, step_n = goal>

    (see Russell & Norvig’s AIMA book for full details)

  – Problem: length n of plan unknown (try all…)
  – Propositional logic: proliferation of symbols

• Here – more “symbol” efficient reduction to modal logic / description logics
Logical Signature

• Next is a functional relation (feature)
• Each “logical symbol” can be in 4 different states
  – goat-A (goat on A river bank)
  – goat-B (goat on B river bank)
  – goat-on-boat-from-A-to-B
  – goat-on-boat-from-B-to-A
Axioms to Constrain Possible Worlds

- Need axioms that describe the possible next states
Successor State Axioms

• If the boat is on riverbank A, the next it is going from A to B
• If the boat is going from A to B, then next it is on riverbank B, etc.

\[
\begin{align*}
(\text{implies boat-a} & \quad (\text{all next boat-from-a-to-b})) \\
(\text{implies boat-from-a-to-b} & \quad \forall x : \text{boat-from-a-to-b}(x) \Rightarrow \\
& \quad (\text{all next boat-b})) \\
(\text{implies boat-b} & \quad \forall y : \text{next}(x, y) \Rightarrow \text{boat-b}(y)) \\
(\text{implies boat-from-b-to-a} & \quad (\text{all next boat-from-b-to-a})) \\
(\text{implies boat-from-b-to-a} & \quad (\text{all next boat-a}))
\end{align*}
\]
Some More Constraints

• If the boat is going from A to B, then
  – the boat cannot go alone (the ferryman has to go with it)
  – It should not go with only the ferryman – however, it can go back from B to A with only the ferryman!
  – the boat has capacity for ferryman and one other object

(implies boat-from-a-to-b
  (and ferryman-boat-from-a-to-b
    (or cabbage-boat-from-a-to-b
      wolf-boat-from-a-to-b
      goat-boat-from-a-to-b)
    (not (and cabbage-boat-from-a-to-b
      wolf-boat-from-a-to-b))
    (not (and cabbage-boat-from-a-to-b
      goat-boat-from-a-to-b))
    (not (and wolf-boat-from-a-to-b
      goat-boat-from-a-to-b)))))
Further Axioms

- Ensure cabbage and goat, and wolf and goat, are not alone
- Ensure every object can only be at one place at a time
- Make sure objects don’t disappear – every “state” specifies that goat, boat, etc. exists “somewhere”

\[
(\text{implies } \text{and } \text{wolf}-\text{a goat}-\text{a} \text{ ferryman}-\text{a}) \\
(\text{implies } \text{and } \text{wolf}-\text{b goat}-\text{b} \text{ ferryman}-\text{b}) \\
(\text{implies } \text{and } \text{cabbage}-\text{a goat}-\text{a} \text{ ferryman}-\text{a}) \\
(\text{implies } \text{and } \text{cabbage}-\text{b goat}-\text{b} \text{ ferryman}-\text{b}) \\
(\text{disjoint goat}-\text{a goat}-\text{b goat-boat-from}-\text{a-to-b goat-boat-from}-\text{b-to-a}) \\
(\text{implies goat} \\
\text{(or goat}-\text{a goat}-\text{b goat-boat-from}-\text{a-to-b goat-boat-from}-\text{b-to-a}))
\]
States, Start, Goal

- Every state is either a goal or has some next state
- Every state specifies the state of all objects
- Start state = all objects on riverbank A
- Goal state = all objects on riverbank B
Verifying a “Solution”

• Check satisfiability of formula

\[
\begin{aligned}
&\text{(and start} \\
&\quad (\text{some next} \\
&\quad \quad (\text{some next} \\
&\quad \quad \quad (\text{some next} \\
&\quad \quad \quad \quad \ldots \\
&\quad \quad \quad \quad (\text{some next goal}))))
\end{aligned}
\]

• There is a solution for \( n = 14 \)
  – and also a surprising one
  – however, we need an ABox to read off the solution!
Create ABox to Read off 
Entailed Individual Types via Abox Query

(instance s1 start) 
(related s1 s2 next) 
(related s2 s3 next) 
(related s3 s4 next) 
... 
(related s13 s14 next) 
(related s14 s15 next) 
(instance s15 goal) 

? (retrieve (?x (types ?x)) (?x state))
> (((?X S1)  ((BOAT-A) (CABBAGE-A) (FERRYMAN-A) (GOAT-A) (START) (WOLF-A))))
((?X S3)  ((BOAT-B) (CABBAGE-A) (FERRYMAN-B) (GOAT-B) (WOLF-A))))
((?X S4)  ((BOAT-FROM-B-TO-A FERRYMAN-BOAT-FROM-B-TO-A) (CABBAGE-A) (GOAT-B) (WOLF-A))))
((?X S5)  ((BOAT-A) (CABBAGE-A) (FERRYMAN-A) (GOAT-B) (WOLF-A))))
((?X S6)  ((BOAT-FROM-A-TO-B FERRYMAN-BOAT-FROM-A-TO-B) (GOAT-B) ))
((?X S7)  ((BOAT-B) (FERRYMAN-B) (GOAT-B) ))
((?X S8)  ((BOAT-FROM-B-TO-A FERRYMAN-BOAT-FROM-B-TO-A) (GOAT-BOAT-FROM-B-TO-A) ))
((?X S9)  ((BOAT-A) (FERRYMAN-A) (GOAT-A) ))
((?X S10)  ((BOAT-FROM-A-TO-B FERRYMAN-BOAT-FROM-A-TO-B) (GOAT-A) ))
((?X S11)  ((BOAT-B) (CABBAGE-B) (FERRYMAN-B) (GOAT-A) (WOLF-B))))
((?X S12)  ((BOAT-FROM-B-TO-A FERRYMAN-BOAT-FROM-B-TO-A) (CABBAGE-B) (GOAT-A) (WOLF-B))))
((?X S13)  ((BOAT-A) (CABBAGE-A) (FERRYMAN-A) (GOAT-A) (WOLF-B))))
((?X S15)  ((BOAT-B) (CABBAGE-B) (FERRYMAN-B) (GOAL) (GOAT-B) (WOLF-B))))
Two Possible Solutions!

? (retrieve (?x (types ?x)) (?x state))
> (((?X S1) ((BOAT-A) (CABBAGE-A) (FERRYMAN-A) (GOAT-A) (START) (WOLF-A)))
    (WOLF-A)))
  ((?X S3) ((BOAT-B) (CABBAGE-A) (FERRYMAN-B) (GOAT-B) (WOLF-A)))
  ((?X S4) ((BOAT-FROM-B-TO-A FERRYMAN-BOAT-FROM-B-TO-A) (CABBAGE-A) (GOAT-B) (WOLF-A)))
  ((?X S5) ((BOAT-A) (CABBAGE-A) (FERRYMAN-A) (GOAT-B) (WOLF-A)))
((?X S6) (CABBAGE-BOAT-FROM-A-TO-B) …)
((?X S7) (CABBAGE-B) (GOAT-B) …)
((?X S8) (GOAT-BOAT-FROM-B-TO-A) …)
((?X S9) (FERRYMAN-A) …)
((?X S10) (WOLF-BOAT-FROM-A-TO-B) …)
((?X S11) (BOAT-B) (CABBAGE-B) (FERRYMAN-B) (GOAT-A) (WOLF-B)))
((?X S12) (BOAT-FROM-B-TO-A FERRYMAN-BOAT-FROM-B-TO-A) (CABBAGE-B) (GOAT-A) (WOLF-B)))
((?X S13) (BOAT-A) (CABBAGE-B) (FERRYMAN-A) (GOAT-A) (WOLF-B)))
  (WOLF-B)))
((?X S15) (BOAT-B) (CABBAGE-B) (FERRYMAN-B) (GOAL) (GOAT-B) (WOLF-B)))
The Official Solution
To load a file named `wzk-via-porter.racer`, follow these steps:

1. **Open RacerPorter**:
   - Launch the RacerPorter application.

2. **Navigate to the Profiles Tab**:
   - Select the `Profiles` tab within the RacerPorter interface.

3. **Select a Profile**:
   - Choose a profile that matches your needs. In this case, select the profile labeled `Localhost / Big TBoxes, Big ABoxes`.
   - Ensure that the profile is configured correctly:
     - **Server**: UPCase
     - **Host**: localhost
     - **Port**: 8988
     - **Version**: 2.0
     - **Auto Connect**: YES
     - **Unsafe**: YES

4. **Load the File**:
   - Click on the `Load...` button.
   - Navigate to the directory containing `wzk-via-porter.racer`.
   - Select `wzk-via-porter.racer` and click `Open`.

5. **Verify the Load**:
   - Confirm that the file `wzk-via-porter.racer` is loaded as intended.

By following these steps, you can successfully load the `wzk-via-porter.racer` file into RacerPorter.
[*] ? Cannot find Racer Executable! Please specify path to Racer using "Edit Profile -> Racer Executable"!
[*] > :ERROR

[*] ? Automatically connected to Racer 2.0 running on localhost:8080 (case: UPCASE)
[*] > (:OKAY "Racer 2.0 running on localhost:8080 (case: UPCASE)"

[1] ? (full-reset)
[1] > :OKAY-FULL-RESET

"C:/Users/wessel/Desktop/Lunch and Learn/Hand-Authoried Racer HT/wak-via-porter.racer"
(FULL-RESET) --> :OKAY-FULL-RESET


[3] ? (abox-consistent?)
[3] > T

[4]?
User this button to get all types on a node.

Change options as shown here... computation of graph takes about 5 seconds.