Modeling Knowledge in Alloy

CS195Y Capstone Abstract

Caleb Stanford

1. Introduction and topic

There exists a large class of logic puzzles which make use of the idea of *knowledge*. Understanding these puzzles is important for example in distributed systems, where each system has a different set of things they "know". For example, one system sends some piece of information to the second system, but possibly it was intercepted, and the first system wants to verify that the second system indeed now knows the piece of information.[1]

Underlying the concept of knowledge in computer science is usually one of several systems of epistemic modal logic; the most common and useful one models the knowledge of "ideal rational agents", and is known as S5. In S5, you know something if it is possible for you to deduce it, in theory, from the information you have. This definition of knowledge underlies several well-known logic puzzles. For example, "birthday guessing", "blue-eyed islanders", "surprise exam", and "card guessing", which are described on the next page.

The main purpose of the project was (after background research on epistemic modal logic and some other things) to model knowledge in the Alloy programming language, and to use it to automatically verify the solutions to the aforementioned knowledge puzzles.

2. Possible worlds

To write up a knowledge problem in Alloy, there were two steps:

(i) Specify the set of possible worlds, or the set of possible configurations of all the data involved in the puzzle.

(i) Specify what each person or agent knows, at each moment in time, by specifying in each world what other worlds are consistent with that agent's information. (This is called the accessibility relation.)

The goal with both of these steps is that the end-user of a programming language like Alloy does not have to know too much about epistemic logic or possible worlds in order to state the puzzle or situation and find a solution. In order to properly implement step (i), Alloy is not quite strong enough, and I usually had to input the total number of possible worlds as a parameter in the program, which is not ideal but could be fixed by using a stronger language or an auxilliary program. However, step (ii) works very well in Alloy if you just specify (generally speaking) what information everyone is aware of at the start, and what they learn over time.

The use of Alloy as a model-checker comes in handy particularly when there is nondeterminism involved; for example, if we want to check whether it is *possible*, if the agent acts in some way, for her to know something.

[1] See, for example, *Knowledge and Common Knowledge in a Distributed Environment*, Joseph Y. Halpern and Yoram Moses.

3. The puzzles

Birthday guessing. In the first puzzle, Cheryl's birthday is one of the following dates: 3/4, 3/5, 3/8, 6/4, 6/7, 9/1, 9/5, 12/1, 12/2, 12/8. Albert knows which month it is, and Bernard knows which day it is, and everyone knows that it is one of the above dates. They both want to know the exact date, and they have the following conversation:

Albert: I don't know, and Bernard doesn't know either.
Bernard: I didn't know at first, but now I know.
Albert: Oh, now I know it too.

In Alloy, the set of possible worlds is just the set of birthdays. Alloy verifies that if the above are true, the birthday must have been September 1. The code is found in `dates.als` and the theme settings in `dates.thm`.

Islanders. In this puzzle, residents of an island with no mirrors are not allowed to discuss eye color with anyone else, but each resident knows everyone else's eye color. However, if you know your own eye color, you are required to announce it at exactly noon every day. There are m people with brown eyes, and n people with blue eyes. One day, a new person lands on the island and announces: "I see at least one person with blue eyes!". What happens?

In `islanders.als` (theme files `islanders.thm` and `islanders_basic.thm`), Alloy verifies (for specific values of m and n) that everyone waits n days, and then on the nth day all the blue-eyed people announce they have blue eyes. Then, on day $n+1$, all the brown-eyed people announce that they have brown eyes.

Surprise Exam. A description of the famous surprise exam paradox can be found online, for example http://www-math.mit.edu/ tchow/unexpected.pdf.

Assuming that a "surprise" exam is one where you don't know it will happen the day before it does, I verified in Alloy that (i) a surprise exam is possible, but (ii) it is impossible for the student to *know* the exam will be a surprise. The code is in `surprise-exam.als` and the theme in `surprise-exam.thm`.

Card Guessing. (This one is a bit different, but still about knowledge, if you think about it in the right way.) A cooperative card game is played between two players. In the original problem, 5 cards are drawn from a shuffled deck of 52 diferent cards by the first player. The first player puts 4 of the 5 cards down in some order, and the second player has to guess the last one. Can the pair accomplish this for all possible sets of cards? The answer is yes.

Alloy is not fast enough to handle this, but in a simplified version the deck has 8 cards, 3 are picked, and 2 of the 3 cards are put down in some order. Alloy verifies that the players can always win. The code is in `card-guessing.als`.