KNOWLEDGE REPRESENTATION, REASONING AND DECLARATIVE PROBLEM SOLVING

By Jicheng Zhao
based on the notes of others

Class 10: Course projects and more declarative problem solving problems

Possible Course Projects

Omitted.

The Tiles Problem

- Problem description: We are given an $n \times n$ grid of squares, like a chess board, except the upper left and lower right squares are missing. We also have a tile pattern of two adjacent squares. We seek to repeatedly place the tile on the grid so that the whole grid is filled.
- It can be proved that there is no-covering on tiles: construct a proper coloring of the tiles using two colors (a proper coloring has no tile adjacent to a tile of the same color). Argue that the number of tiles with one color exceeds the number of tiles with the other color; every application of the covering tile covers exactly one tile of each color. Therefore, no covering can cover a board with unequal numbers of each color.
- Using answer set programming to reason about this, the first step is to represent the problem.

- One could use $tile(x, y, u, v)$ to indicate that squares (x, y) and (u, v) are covered by the same tile. But the program is too large after grounding.
- Alternatively, use a predicate like $tile(x, y, r)$, indicating that a tile is placed at (x, y), extending to the right.

- Predicates of high arity lead to a large number of rules when the program is grounded. Therefore, sometimes efficiency requires the use of predicates with fewer variables as arguments. To estimate efficiency of a smodels rule is to count the total number of variables in a rule – and the number of different values that each variable takes.

The Zebra Problem

- Problem description: Five houses are in a row. Each house has a color, drink, cigarettes, pet, and country of the occupant.

  For the sake of efficiency, we should not use a predicate with many variables, such as the following.

- Coming up with a good representation is again key, since that determines whether or not it blows up under grounding.
  - A poor representation would be:

    $$P(1, color, drinks, cig, pet, country)$$

    6 variables, maybe more, over all values present.

– Breaking up into smaller sub-rules is far more efficient. Instead of this, use two-places predicates:

$$color(House, Color), drink(House, Drink), cig(House, Cig),$$
$$pet(House, Pet), country(House, Country)$$

– Another way is to introduce a new predicate *associate*, which means that a house and one property associated with it.

$$associated(House, Value)$$

AnsProlog as a general problem solving method

It is easy to map constraint satisfaction, even dynamic constraint satisfaction, problems to AnsProlog with answer set semantics.
If you can map all of these different domains to the same language, then the focus of implementation is on that one language, and less duplication of effort may takes place.