VL Logikorientierte Programmierung

Answer-Set Programming: Exercise 1

Uwe Egly, Michael Fink, and Hans Tompits

Institut für Informationssysteme,
Arbeitsbereich Wissensbasierte Systeme 184/3,
Technische Universität Wien
Summer term 2009

1 Introduction

This exercise deals with problems associated with organising scientific conferences. The task is to construct, for each problem, suitable *extended logic programs* (ELPs) such that the solutions of a given problem are determined by the answer sets of the programs. The specification of the problems should be in such a way that the *guess-and-check* feature of answer-set programming is taken into account, by using *default negation*, *disjunctions*, and *integrity constraints* as essential formalisation tools. The evaluation of the constructed programs shall be carried out with the system DLV (`http://www.dlvsystem.com`), which is available in the lab accounts. Special features of DLV, which extend standard answer-set programming language constituents, like *weak constraints* and *aggregate functions*, shall **not** be employed for constructing the programs in this example—these features will be the subject of the next exercise.

2 General Problem Description

One of the integral parts of scientific work is the distribution of scientific results. There are different ways to do this, e.g., through conferences, workshops, symposia, journals, or book contributions. The usual way for publishing a scientific paper is by means of a *peer-review procedure*, where referees—usually other scientists—decide about the acceptance of a paper. Depending on the scientific area, there are different policies for the publication process; here, we are interested in the problem of assigning referees for submissions to a conference as it is typical for the area of computer science.

Conferences are held about specialised topics in periodic intervals, usually annually or bi-annually. For submitting a paper, there are *deadlines* until this can be done. A conference has a *program committee* (PC) and an *organising committee*. The program committee is responsible for the selection of submitted papers; the organising committee is responsible for carrying out the actual conference. The PC has a *program chair* (PC-chair), which usually consists of one or two members from the PC.

For determining the acceptance or rejection of a submitted paper, the latter are assigned to members of the PC which in turn choose suitable referees for them. The referees, then, give recommendations about the acceptance or rejection of a paper, and the members of the PC eventually decide about the final outcome. Accepted papers are then published in a *conference proceedings* and must be presented at the conference (usually in a 20 to 30 minutes talk).

In this exercise, the problem of assigning the submitted papers to the members of the PC shall be modeled in terms of logic programs under the answer-set semantics. The precise problem description is as follows:

- The assignment of the submitted papers among the members of the PC is based in terms of *keywords*:

  - Each paper must select exactly one keyword which characterises its area, and analogously each member of the PC must select exactly three keywords which characterise his or her area of competence.
  - The keywords—both for the submissions and the members of the PC—are selected from a given list of keywords.

- The assignment of the submissions among the PC-members should be chosen in such a way that each PC-member obtains at most four papers and each paper must be from his or her area of competence. Moreover, each paper must be assigned to four different PC-members.
- Each paper is associated with its authors. PC-members are allowed to submit papers but not more than one. If a paper was authored by a PC-member, it has to be assigned to a PC-member different from that author.
- PC-members may state a conflict of interest with a paper, e.g., for the reason of an ongoing collaboration between the PC-member and one of the paper's authors. In case of such a conflict, the paper under consideration will not be assigned to that PC-member.

We call this problem the *referee problem* in what follows. We will solve this problem by means of two subtasks.

3 Subtask 1

3.1 Definition of the Check-Program

In the first part of the task, construct a program `check.dl` which checks, given an assignment of submissions to members of the PC, whether the following conditions hold:

(1) each PC-member is assigned with at most four submissions;

(2) for every PC-member M, no submission assigned to M has a keyword which is not one of the keywords of M (i.e., which is not one of the areas of competence of M);

(3) no submission and no PC-member lists a keyword which is not taken from the given list of keywords;

(4) no PC-member has submitted more than one paper;

(5) no submission authored by a PC-member is assigned to that PC-member;

(6) no submission is assigned to a PC-member who stated a conflict with that paper.

The given assignment of submissions to referees is assumed to be stored in some input files (described below) containing the following data:

- the names of the PC-members;
- the submitted papers;
- the list of keywords;
- the authors of each paper (each paper has at least one author);
- the keywords assigned with each submission (every submission must have exactly one keyword) and the keywords assigned with each member of the PC (every PC-member must be associated with exactly three keywords);

- the PC-members together with their conflicting papers
- the assignment of submissions to the members of the PC.

The input must be specified such that these conditions are met. Furthermore, you must use the following predicates for the construction of program `check.dl` as well as for the input data:

- `pc(M)`: `M` is a member of the PC;
- `paper(P)`: `P` is a submitted paper;
- `keyword(K)`: `K` is a keyword;
- `author(A,P)`: `A` is author of the submitted paper `P`;
- `conflict(M,P)`: PC-member `M` has a conflict with paper `P`;
- `associated(O,K)`: object `O` (either a member of the PC or a submission) has an associated keyword `K`;
- `assigned(P,M)`: the submission `P` is assigned to PC-member `M`;

The program `check.dl` should satisfy the following condition:

- `check.dl`, together with the input data, possesses an answer set precisely when Conditions (1)–(6) are met.

Important: Do not use any aggregate functions for constructing the program `check.dl`!

3.2 Testing the Check-Program

Test the functionality of your program `check.dl` by constructing **five test cases**. To this end, for each test case n ($1 \leq n \leq 5$), safe your data in the following files:

- `pc`n`.dl`: contains the names of the PC-members, i.e., facts of form `pc(M)`, the keywords, i.e., facts of form `associated(M,K)`, and the conflicts, i.e., facts of form `conflict(M,P)`, where `M` is a PC-member, `K` is a keyword, and `P` is a paper.
- `submissions`n`.dl`: contains the submitted papers, i.e., facts of form `paper(P)`, as well as authors and keywords of the papers, i.e., facts of form `author(A,P)` and `associated(P,K)`, where `A` is a author, `P` is a paper, and `K` is a keyword;
- `kw`n`.dl`: contains the list of keywords, i.e., facts of form `keyword(K)`.
- `asgn`n`.dl`: contains the assignment of papers to members of the PC, i.e., facts of form `assigned(P,M)`.

For testing your program, note that it is beneficial to use both *positive tests* (corresponding to a correct behaviour) and *negative tests* (violating the specification).

4 Subtask 2

4.1 Definition of the Guess-Program

Now construct a program `guess.dl` which assigns, given a collection of submissions and a given PC, the submissions to the members of the PC in such a way that the following condition is satisfied:

(∗) each submission is assigned to exactly four members of the PC;

Use the above defined predicates

`pc(M)`, `paper(P)`, and `assigned(P,M)`.

The program `guess.dl` should be constructed in such a way that the following property is met:

- the answer sets of `guess.dl`, together with the input data, yield all assignments of submissions to the members of the PC which satisfy Condition (∗).

4.2 Testing the Overall Program

Use your test files from Subsection 3.2—*but without the files* `asgn`n`.dl` ($1 \leq n \leq 5$)—to test the overall program consisting of `guess.dl` and `check.dl`, solving the referee problem.

Important:

1. Be sure that you follow the naming conventions as pointed out above.
2. All of the files mentioned above should be contained in **one directory**!
3. Do not include the command `#maxint` in your files—rather, execute DLV with the corresponding command-line option `-N` to specify the known numbers.
4. Do not use any weak constraints or aggregate functions for specifying your programs.

In case of questions, please ask the teaching assistants or send an email to

`logprog-fragen-ss09@kr.tuwien.ac.at`.