Family Trip Problem Solver

A couple of Prolog based implementations of a solver for the *family trip* problem, which is stated in the `statement.txt` file.

The `swipl` and `picosat` packages are required to run the solvers. In order for the scripts to run, both programs should be accessible from the `PATH` environment variable.

The problem

A family wants to plan their next holidays trip. They have gathered a list of cities they would like to visit, which comprises the first predicate of the input data:

```
cities([paris,bangkok,montevideo,windhoek,male,delhi,reunion,lima,banff]).
```

Besides the list of cities, the family has also written the interests they would like to fulfill along their holidays:

```
interests([landscapes,culture,ethnics,gastronomy,sport,relax]).
```

Finally, for each of the cities, they know which attractions it offers them:

```
attractions( paris,     [culture,gastronomy]        ).
attractions( bangkok,   [landscapes,relax,sport]    ).
attractions( montevideo,[gastronomy,relax]          ).
attractions( windhoek,  [ethnics,landscapes]        ).
attractions( male,      [landscapes,relax,sport]    ).
attractions( delhi,     [culture,ethnics]           ).
attractions( reunion,   [sport,relax,gastronomy]    ).
attractions( lima,      [landscapes,sport,culture] ).
attractions( banff,     [sport,landscapes]          ).
```

With the given data, they want to minimize the number of cities to be visited, while still fulfilling all their interests.

Prolog

This implementation, under the `prolog` folder is "Prolog-pure". A shell script is provided to aid the compilation and execution of the solver.

Files: * `family-trip.pl`: the solver's main logic code * `run.sh`: the shell script to compile and execute the solver with a given instance * `trip.pl`: an instance file for the solver

Script:

To compile the solver with the `trip.pl` instance file, simply execute:

```
./run.sh trip.pl
```

Generated files: * `solve`: the standalone executable program, compiled from the solver's code * `solver.pl`: the solver's code, with the instance file attached

SAT

The second implementation, under the `sat` folder, uses Prolog to write a CNF encoding of the problem constraints, to be then fed as input to the SAT solver (the `picosat` package).

A shell script is also provided to ease the compilation and execution of the solver.

Files: * `display-solution.pl`: the solver's solution display code * `family-trip.pl`: the solver's main logic code * `run.sh`: the shell script to compile and execute the solver with a given instance * `trip.pl`: an instance file for the solver

Script:

To compile and run the solver with the `trip.pl` instance file, simply execute:

```
./run.sh trip.pl
```

To compile the solver and retrieve the symbolic output (the generated symbolic CNF file to be fed to the `picosat` SAT solver), execute:

```
./run.sh -s 3 symbolic-output.txt trip.pl
```

where 3 is the maximum number of cities (give it the number you desire!) and `symbolic-output.txt` is the file where the symbolic CNF output will be written.

Generated files: * `solution.txt`: the result of the execution of the solver * `solve`: the standalone executable program, compiled from the solver's code * `solver.pl`: the solver's code, with the instance file attached and all the remaining necessary predicates, written by the script