Inspire at Inductive Logic Programming Competition: Fine-grained Cost-based Hypothesis Generation*

Peter Schüller

Computer Engineering Department
Faculty of Engineering
Marmara University, Turkey
peter.schuller@marmara.edu.tr

Abstract. The Inspire system is an Inductive Logic Programming system based on an ASP encoding for generating hypotheses with a cost from the mode bias, and a transformation of hypotheses and examples to an ASP optimization problem that has the smallest hypothesis covering all examples as solutions. Inspire attempts to learn a hypothesis on single examples while increasing maximum hypothesis cost. As underlying ASP solver we use Clingo.

Disclaimer: for space reasons, this note only describes design decisions and the core idea of the Inspire system. Literature review, comparison and analysis of the system with other systems, and details about hypothesis cost, are omitted.

1 Design Decisions

Our system uses Answer Set Programming (ASP) [4–6, 9] for generating, optimizing, and verifying hypotheses.

Hypotheses are learned on an example-by-example basis, never from multiple examples. This is due to (i) low available resources (2 GB memory and 30 sec time), (ii) simplified handling of infinite relations like $time/1$ in the background knowledge, and (iii) because single given examples in the given example datasets were complex enough to fully specify a hypothesis.

Algorithm 1 shows the main idea of the Inspire system: the mode bias is used to blindly generate hypotheses using an ASP program, based on a fine-grained cost function and an incrementally increasing cost limit. Finding a hypothesis for a single example triggers an (efficient) test of the hypothesis over all examples and in case more examples are covered than by prior hypotheses, produces an output attempt for all test traces (unlimited attempts were permitted in the competition). The cost of hypotheses is parameterized as indicated in Figure 1.

Inspire is implemented in Python and uses ASP systems Gringo [7] and Clasp [8]. We use both unsat-core optimization [2] and stratification [1, 3].

* This work has been supported by Scientific and Technological Research Council of Turkey (TUBITAK) Grants 114E430 and 114E777.

Algorithm 1: INSPIRE-ILP(KB bk, Examples e, Mode-bias m, Test-traces t)

Sort e by length of example trace `// smaller examples first`
$bestquality := 0$
for $climit \in \{4, \ldots, 15\}$ **do**
  $hcand$:= hypothesis rule candidates obtained from $\mathcal{AS}(P_h(costlimit = climit)$
  for $\langle trace, label \rangle \in e$ **do** `// go over sorted examples`
    Find optimal $hr \subseteq hcand$ s.t. $label \in \mathcal{AS}(bk \cup hr \cup trace)$
    if hr *does not exist (unsat)* **then** `// cannot cover example`
      Exit loop and continue with next $climit$
    else
      $quality := \big|\langle etrace, elabel \rangle \in e \mid elabel \in \mathcal{AS}(bk \cup hr \cup etrace)\big|$
      if $quality > bestquality$ **then**
        $bestquality := quality$
        Output `#attempt` `// best-effort output`
        for $ttrace \in t$ **do**
          if $valid \in \mathcal{AS}(bk \cup hr \cup ttrace)$ **then** Output `VALID`
          else Output `INVALID`
        if $quality = |e|$ **then**
          Exit program `// on e we predict correctly`

2 Properties

The main idea of our approach is to obtain a fine-grained hypothesis spaces of slowly increasing complexity. Simple hypotheses are found fast, while all hypotheses have a chance to be found eventually.

The approach has the advantage that there is no need to make abduction of required facts, then induction of potential rules, then generalization of these rules, then a search for the smallest hypothesis (as done in the XHAIL [11] system) while the obvious disadvantage is, that hypothesis search is blind (similar as in the ILASP [10] system).

References

1. Alviano, M., Dodaro, C., Marques-Silva, J., Ricca, F.: Optimum stable model search: algorithms and implementation. Journal of Logic and Computation exv061 (2015)
2. Andres, B., Kaufmann, B., Matheis, O., Schaub, T.: Unsatisfiability-based optimization in clasp. In: International Conference on Logic Programming (ICLP), Technical Communications. pp. 212–221 (2012)
3. Ansótegui, C., Bonet, M.L., Levy, J.: SAT-based MaxSAT algorithms. Artificial Intelligence 196, 77–105 (2013)
4. Baral, C.: Knowledge Representation, Reasoning, and Declarative Problem Solving. Cambridge University Press (2004)
5. Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming at a glance. Communications of the ACM 54(12) (2011)

```
% rule restrictions
#const maxliterals = 4.
#const maxuseppred = 2.   % positive predicate use
#const maxusenpred = 2.   % negative predicate use
#const maxvars = 4.       % overall variables

% predicate invention restrictions
#const maxinventpred = 1.   % number of inventable predicates
#const invent_minarity = 2. % (arity 1 can be represented in arity 2
#const invent_maxarity = 2.

% costs for distinct items in one rule
#const cost_distinctvariable = 1.
#const free_distinct_variables = 2.         % free = used once
#const cost_variable_boundmorethantwice = 2.
#const cost_type_usedmorethantwice = 2.     % mode bias types
#const cost_posbodyliteral = 1.
#const cost_negbodyliteral = 2.
#const cost_pred_multi = 2. % using predicate multiple times
% if same variable occurs twice in binary predicate
#const cost_binary_reflexive = 5.
% if variable occurs only in rule head (safe because of type)
#const cost_variable_onlyheadbound = 5.
% if variable is not used in two distinct body predicates
#const cost_variable_onlyoncebodybound = 5.

#const cost_invent_predicate = 2.     % base cost for inventing
#const cost_invented_predicate = 2.   % cost per usage in rule
#const cost_invented_head_and_body = 3.
#const cost_invented_body_multi = 5.  % multiple in body
% if invented predicates are not 'stratified' across rules
#const cost_invented_head_body_order = 5.
```

Fig. 1. Costs used in Inspire ILP system.

6. Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Answer Set Solving in Practice. Morgan Claypool (2012)
7. Gebser, M., Kaminski, R., König, A., Schaub, T.: Advances in gringo series 3. In: International Conference on Logic Programming and Non-monotonic Reasoning (LPNMR). pp. 345–351 (2011)
8. Gebser, M., Kaufmann, B., Schaub, T.: Conflict-driven answer set solving: From theory to practice. Artificial Intelligence 187-188, 52–89 (2012)
9. Gelfond, M., Lifschitz, V.: The Stable Model Semantics for Logic Programming. In: International Conference and Symposium on Logic Programming (ICLP/SLP). pp. 1070–1080 (1988)
10. Law, M., Russo, A., Broda, K.: Inductive Learning of Answer Set Programs. In: European Conference on Logics in Artificial Intelligence (JELIA). pp. 311–325 (2014)
11. Ray, O.: Nonmonotonic abductive inductive learning. Journal of Applied Logic 7, 329–340 (2009)