Role-Based Access Control via JASP

Roberta Costabile[1], Alessio Fiorentino[1], Nicola Leone[1,2],
Marco Manna[1], Kristian Reale[1,2], and Francesco Ricca[1]

[1]University of Calabria, Italy – `<lastname>@mat.unical.it`
[2]DLVSystem s.r.l, Italy

Abstract. In this paper, we answer the Role-Based Access Control (RBAC) challenge by showcasing the solution of RBAC components by using JASP, a flexible framework integrating ASP with Java. In JASP the programmer can simply embed ASP code in a Java program without caring about the interaction with the underlying ASP system. This way, it is possible solve seamlessly both tasks suitable for imperative and declarative specification as required by RBAC.

1 Introducing JASP

Answer Set Programming (ASP) [1] is a fully-declarative logic programming paradigm proposed in the area of knowledge representation and non-monotonic reasoning. Its idea is to represent a given computational problem by a logic program whose answer sets correspond to solutions, and use a solver to find them. After many years of research, the formal properties of ASP are well-understood; notably, ASP is expressive: it can solve problems of complexity beyond NP [4, 5]. Moreover, the availability of robust and efficient solvers [8] made ASP an effective powerful tool for advanced applications, and stimulated the development of many interesting applications [6]. Recently, we have employed ASP for developing some industrial application, such as: building systems for workforce management [10], e-tourism [3], and solving complex industry-relevant problems [2].

The experience we gained has confirmed the viability of the industrial exploitation of ASP. However, it has evidenced the strong need of integrating ASP technologies (i.e., ASP programs and solvers) with well-assessed software-development processes and platforms, which are tailored for modern imperative object-oriented programming languages [9]. Indeed, the lesson we have learned, while building real-world ASP-based applications, confirms that complex business-logic features can be developed in ASP at a lower (implementation) price than in traditional imperative languages. Indeed, from a software engineering viewpoint, the employment of ASP brings many advantages not only in terms of readability, but also in flexibility, extensibility, and ease of maintenance. However, since ASP is not a fully general-purpose language, declarative specifications have to be "embedded", at some point, inside imperative modules that are necessary to develop user-friendly applications making use, for example, of visual user-interfaces. To this end, we have introduced a new programming framework integrating ASP with Java [7]. The framework is based on a hybrid language, called JASP, that transparently supports a bilateral interaction between ASP and Java. The programmer can simply embed ASP code in a Java program without caring about the interaction with the underlying ASP system. The logical ASP program can access Java variables, and the answer sets, resulting from the execution of the ASP code, are automatically stored in Java objects, possibly populating Java collections, transparently.

2 Modeling the RBAC Challenge with JASP

In this section, we sketch a JASP-based solution to the Role-Based Access Control (RBAC) challenge.[1] According to JASP's philosophy, we associate a Java class to each RBAC-set occurring in the five components. Then, we define a class `Manager` to implement all methods performing updates and queries from checking to planning. In what follows, we report the implementation of method `trans` —computing the transitive closure of role hierarchy (encoded via relation `rh`) unioned with the reflexive role (encoded via relation `role`) pairs— to appreciate the succinctness and elegance of the approach:

```
public List<RH> trans() {
     List<RH> jtr = new ArrayList<RH>();
     <#
        IN = jrole::role;
        IN = jrh::rh;
        OUT = jtr::tr;
        tr(R, R) :- role(R).
        tr(R1, R2) :- rh(R1, R2).
        tr(R1, R3) :- tr(R1, R2), tr(R2, R3).
     #>
    return jtr;
}
```

In particular, we create the Java object `jtr` and state, via the keyword `OUT`, that it will host —after the subsequent three ASP rules will have been evaluated— all tuples of predicate `tr`, which occurs in the head of those ASP rules. Then, we use the keyword `IN` to specify that all roles in the Java object `jrole` (resp., `jrh`) will populate the EDB predicate `role` (resp., `rh`) used in the body of the first (resp., second) ASP rule to "feed" the answer set computation. Other functions of the challenge can be developed similarly.

The same applies to getRolesShortestPlan, the hardest function in the challenge. In particular, we drafted an encoding that combines: (*i*) *functional terms* to encode in a uniform way different kind of updating actions given as input; (*ii*) *arithmetic operators* to design a "temporal" encoding involving at most 2^n updating steps, where n is the number of possible actions; (*iii*) *disjunction* to guess, at each step, the next action; (*iv*) *aggregates*, *negation* and *strong constraints*

[1] See `http://lpop.cs.stonybrook.edu/preparing-your-position-paper`

to guarantee consistency after each update; and (*v*) *weak constraints* to minimize the number of steps. A more complete description of all solutions can be downloaded from `www.mat.unical.it/ricca/downloads/LPOP-RBAC-18.zip`. Finally, database-oriented features as well as data persistence can be also added by integrating a standard RDBMS in the application. Indeed, update functions can be implemented via DML statement in SQL, and the evaluation of the prescribed constraints can be specified as ASP rules inside Java function and the execution (if needed) can be delegated to the DBMS integrating DLV^{DB} [11].

Concluding remarks. JASP allows to combine in the same environment mainstream technologies for developing applications and ASP. This allowed us to model RBAC Challenge problems and obtain quite rapidly a prototype system.[2] Of course, obtaining a "real", complete and also efficient solution, which takes into account also other real-world nonfunctional and functional requirements, would very likely require to adopt more advanced coding strategies and additional tools (e.g., to develop a graphical interface or a WEB service).

References

1. Brewka, G., Eiter, T., Truszczynski, M.: Answer set programming at a glance. Commun. ACM **54**(12), 92–103 (2011)
2. Dodaro, C., Gasteiger, P., Leone, N., Musitsch, B., Ricca, F., Schekotihin, K.: Combining answer set programming and domain heuristics for solving hard industrial problems. TPLP **16**(5-6), 653–669 (2016)
3. Dodaro, C., Leone, N., Nardi, B., Ricca, F.: Allotment problem in travel industry: A solution based on ASP. In: In proc. of RR'15. pp. 77–92 (2015)
4. Eiter, T., Gottlob, G.: Expressiveness of stable model semantics for disjuncitve logic programs with functions. J. Log. Program. **33**(2), 167–178 (1997)
5. Eiter, T., Gottlob, G., Mannila, H.: Disjunctive datalog. ACM Trans. Database Syst. **22**(3), 364–418 (1997)
6. Erdem, E., Gelfond, M., Leone, N.: Applications of answer set programming. AI Magazine **37**(3), 53–68 (2016)
7. Febbraro, O., Leone, N., Grasso, G., Ricca, F.: JASP: A framework for integrating answer set programming with java. In: In proc. of KR'12 (2012)
8. Gebser, M., Maratea, M., Ricca, F.: The sixth answer set programming competition. J. Artif. Intell. Res. **60**, 41–95 (2017)
9. Leone, N., Ricca, F.: Answer set programming: A tour from the basics to advanced development tools and industrial applications. In: In proc. of Reasoning Web'15. pp. 308–326 (2015)
10. Ricca, F., Grasso, G., Alviano, M., Manna, M., Lio, V., Iiritano, S., Leone, N.: Team-building with answer set programming in the gioia-tauro seaport. TPLP **12**(3), 361–381 (2012)
11. Terracina, G., Leone, N., Lio, V., Panetta, C.: Experimenting with recursive queries in database and logic programming systems. TPLP **8**(2), 129–165 (2008)

[2] Two 1st year PhD students dedicated about two weeks to the project, including the time to study the challenge and the JASP framework. On the technical side they were advised by maintainer of JDLV.