Security Policies in Constraint Handling Rules

Position Paper

Thom Frühwirth
Ulm University, Germany

http://www.informatik.uni-ulm.de/pm/fileadmin/pm/home/fruehwirth/

Abstract

Role-Based Access Control (RBAC) is a popular security policy framework. User access to resources is controlled by roles and privileges. Constraint Handling Rules (CHR) is a rule-based constraint logic language. CHR has been used to implement security policies for trust management systems like RBAC for more than two decades.

Constraint Handling Rules (CHR) [Frü09, Frü18, Frü15, FR18] is a logical rule-based language and framework employing constraints. In this short paper, we describe work on RBAC that is implemented in CHR. We just cite the main works for each group of authors. Further references may be found in the cited papers and/or by googling.

Based on [BFM02], Bistarelli et al. [BMS10, BMS12] apply an extension of Datalog by weighted facts to model role-based trust management. Deduction can validate access requests. Abduction can compute missing credentials if the access is denied and it can compute the level of preference that would grant the access. Both deduction and abduction are expressed in Weighted Datalog and translated into CHR for execution. [BCMS14] show how this deductive and abductive reasoning can be efficiently ported to Android enabling distributed authorization. Both deduction and abduction are implemented as programs in a version of CHR that is embedded into Java (JCHR).

Ribeiro et al. [RG99] present a static analyzer that automatically verifies consistency of workflow specifications written in WPDL (Workflow Process Definition Language) and of specifications in a security policy language (SPL). The analyzer is implemented with CHR embedded in SICstus Prolog. [RZFG00] further describes this Policy Consistency Verifier (PVC). It now includes constraints automatically annotated with temporal information. [RF07] presents further work on the security policy language (SPL). It can express the concepts of permission and prohibition, and some restricted forms of obligation as well as history-based approaches. Given a SPL specification, it is verified using CHR and then compiled to Java into a corresponding security access monitor. The current CHR verifier has about 300 rules and is able to solve all SPL constraints, including the constraints implicitly qualified with time.
The Object Constraint Language (OCL) is a declarative text language describing rules applying to Unified Modeling Language (UML) models. OCL provides constraint and object query expressions on models that cannot be expressed by diagrammatic notation. OCL is now a key component of the new OMG standard recommendation for transforming models. Model finders automatically verify UML/OCL models by checking satisfiability (consistency) of models using example instances. The work of [DTVH16] presents oclCHR https://uet.vnu.edu.vn/~hanhdd/oclCHR/, a verifier implemented in CHR embedded in Eclipse Prolog. It is of interest here, because the authors use an UML model of RBAC as their main example.

Finally, we would like to cite two approaches of RBAC in logical languages that can be readily translated into CHR. [BS03] show how a range of role-based access control (RBAC) models may be usefully represented as executable logical specifications in constraint logic programs (CLP). Like Weighted Datalog, CLP clauses can be translated to CHR [Fri09].

[OPR18] presents a declarative interpretation of Access Permissions (APs) as Linear Concurrent Constraint Programs (LCC). By interpreting LCC programs as formulae in intuitionistic linear logic, they can verify properties of AP programs such as deadlock detection, correctness, and concurrency. CHR also admits a linear logic interpretation [Bet14] and is closely related to the more recent LCC language. Translations between LCC and CHR are given in [Mar10].

Concluding, CHR is a often used language to build reasoning services. In this paper, we showed this surveying shortly work on the problem of security policies, i.e. access control. We would like to thank the anonymous referees for their helpful comments.

References


