Verification Techniques for Policy based Systems

Erisa Karafili
Imperial College London
180 Queen's Gate, SW7 2AZ, London, UK
Email: e.karafili@imperial.ac.uk

Stephen Pipes
IBM UK
Hursley Park, Winchester
Email: pipessd@uk.ibm.com

Emil C. Lupu
Imperial College London
180 Queen's Gate, SW7 2AZ, London, UK
Email: e.c.lupu@imperial.ac.uk

***Abstract*—Verification techniques are applied to policy based systems to ensure design correctness and to aid in the discovery of errors at an early stage of the development life cycle. A primary goal of policy verification is to evaluate the policy's validity. Other analyses on policy based systems include the identification of conflicting policies and policy efficiency evaluation and improvement. In this work, we present a discussion and classification of recent research on verification techniques for policy based systems. We analyse several techniques and identify popular supporting verification tools. An evaluation of the benefits and drawbacks of the existing policy analyses is made. Some of the common identified problems were the significant need of computational power, the limitation of the techniques to particular policy model, which restrict their extension to other policy models and the lack of efficient conflicts resolution methods. We use the evaluation results for discussing the further challenges and future research directions that will be faced by policy verification techniques. In particular, we discuss specific requirements concerning verification techniques for coalition policies systems and autonomous decision making.**

***Keywords*-Verification techniques, policy analysis, generative polices, policy refinement, efficient analysis, policy conflict resolution, autonomous decision making.**

I. Introduction

Policies govern the behaviour of systems and their components. Different policy models together with their languages have been developed for different purposes, e.g., access control, usage control, web access control and data sharing. Verification techniques are needed to reason about correctness of policy and the underlying management functions. This is important because there are pertinent challenges to system design, such as the need for reliable assurances of system operation and the need for consistent operation across disparate information, organisational and security domains. Furthermore, some policy languages can lead to verbose specifications which increase the likelihood of overlooking errors introduced by human policy makers and other users. Tool automation can help in reducing the likelihood of user error and thus can help to improve the reliability of specification and analysis.

Specifications of system requirements are expressed in a notation which is then interpreted by a verification method. The latter are applied to the system to ensure validity of system requirements. A number of verification tools and techniques have been developed together with supporting specification languages that serve a broad variety of needs.

In this work, we review the current state of research in verification approaches for policy systems. This work classifies several contemporary verification techniques and identifies supporting verification tooling that are relevant for policy systems. Where appropriate, techniques that support the analysis and resolution of policy conflicts are highlighted. We give an evaluation of some analysis techniques, then review their benefits and drawbacks and discuss future challenges.

Reasoning techniques and tools can be limited in the properties that they are designed to analyse or may fail to provide a sufficient granularity. A principal role of verification is to identify conflicting policies. Whilst some efforts have been made to capture complex conflicting policies, there are few techniques that provide conflict resolution. This ensures that conflict resolution is typically left to the human user which increases the likelihood of error. A second significant part of verification seeks to improve the efficiency of a given set of policies by performing redundancy and gap analysis. Redundancy analysis identifies the presence of superfluous policy, whilst gap analysis identifies the absence of instances that are not covered by a given set of policies.

Some verification techniques come with drawbacks, such as the need for significant computational power, which may limit their use to offline policy optimisation or the necessity for a translator to interpret the results from reasoning, which can imply a lost of precision and expressivity. Analysis techniques are often specific to a particular policy model, which may limit options to extend to other policy models.

In this work, we analyse some of the most important policy analysis techniques and categorise them by the type of formal approach that is utilised. Verification techniques based on model checking are mainly used for access control and policy validation. Other approaches include the application of concurrency techniques. Constructed concurrency-based analysis identifies inconsistencies and possible conflicting policies before their enforcement.

SMT solvers together with SAT solvers are well known techniques also used for policy verification, in particular for policies defined with XACML. In this case, a translator is usually defined for extracting relevant information and building the internal model from the XACML policies. The policy analysis performs policy validation, comparison and querying.

The use of algebraic solutions is popular for policy verification techniques that are performed upon enforcement. Different policy analyses based on algebraic solutions are able to not only identify conflicting policies, but also perform efficiency analysis for capturing redundancies and gaps between policies. Algebraic solutions also permit the introduction of conflict resolution, which uses precedences/hierarchies between policies.

More recently, abductive reasoning has taken an important role in policy verification. Abductive constraint logic programming is a well known technique for working under incomplete information. The analyses identify conflicting, redundant and missing policies and enables behavioural simulation. In recent works, abductive reasoning is used together with argumentation-based reasoning. Argumentation-based reasoning – for simplicity, argumentation reasoning – is a well suited technique for implementing decision making mechanisms under conflicting, incomplete and context-related knowledge. The policy analysis based on argumentation and abductive reasoning identifies conflicting, redundant and missing policies. The novelty of this analysis is the identification of conceptual conflicts. The latter are conflicts that are strongly context-dependent. It is the first time that this type of conflict is captured. Abductive reasoning provides useful information for conflict resolution as it provides an explanation for the decisions that are taken. Thus, this technique introduces dynamic conflict resolution based on priorities between policies.

Finally, this work describes a future research direction that seeks to improve policy verification techniques for collaborative decision-making in coalition networks, whereby information networks form dynamically and in response to ad hoc demands for interoperability and sharing. Such systems must be capable of operating in a more independent fashion than is permitted in traditional policy architecture; the decision-making process will need to operate autonomously within an overarching policy framework.

II. Policy verification techniques

The policy language and the system model are closely related to the analysis and verification that can be applied. In this section, we review the current state of research in verification approaches for policy systems. Our focus is on the various analyses and the properties they analyse in the policy systems. We point out the benefits and drawbacks of the different analysis techniques and their relations with the respective policy languages and models and classify these analyses by the implemented formal techniques. The main verification techniques are model checking, algebraic solutions, concurrency, SMT and SAT solver, abductive reasoning and their conjunctions.

A. Model Checking

Model checking is utilised by several verification techniques that are used for policy validation. Techniques used for access control policy validation are based on model checking [1], [2] as well as parameterized model checking [3], or model checking in conjunction with other techniques like *satisfiability modulo theories* (SMT) solvers [4], [5]. Model checking techniques are used also for validating policies during the usage policies process [6].

Symbolic model checking can be used for analysing administrative attribute-based RBAC-policies [5], where the authors present a security analysis process that is automated with the use of existing automated theorem proving techniques. The policies are first translated into a symbolic representation by means of formulae in a decidable fragment of first-order logic. The symbolic representation is used to automate a procedure that solves a reachability problem. Then the security analysis problem of user-role reachability is encoded into a symbolic reachability problem and solved by using symbolic model checking. The analysis can detect and eliminate redundancies, and perform backward reachability refinement for implementing a safety analysis.

Linear time model checking is used for goal oriented policy refinement frameworks and reactive systems analysis as shown in [7]. A model checking approach together with Liner Temporal Logic (LTL) is used to represent and detect conflicts in *self-adaptive systems* [8]. The authors in [8] introduce a classification of conflicting policies for self-adaptive systems, where they not only point out the classic conflicting and redundant policies, but also policies that are unenforceable and policies that never activate or action conflicts that never execute, where the enforcement of a policy makes the action of another one impossible to execute. The different types of conflicts are represented using LTL and detected using a model checking approach. When a property is violated, the model checker generates counterexamples that are helpful in identifying conflicting policies and in determining how to resolve conflicts. The stability of self-adaptive systems is checked by using static analysis techniques.

B. Algebraic Solutions

Using algebraic techniques is another interesting approach for conducting verifications of policy based systems. The algebra proposed in [9] captures both authorisation and obligation policies. The algebra provides language independent mechanisms to manage policies. This work introduces a policy integration together with policy analysis that can be performed upon enforcement. The analysis checks for redundancies, gaps and conflicts between policies. A conflict resolution is defined, which uses the policy precedences that were properly assigned to them.

A new formal model for policy representation that is independent from the enforcement elements is introduced in [10]. A procedure based on an algebraic approach for identifying and easily removing inconsistencies and anomalies is proposed. This analysis technique identifies conflicting policies and performs a gap analysis of policy coverage. The conflict resolution is done with the use of hierarchies between policies, where *lattices* together with the *canonical form* of the set-based policy representation are used. The implementation is made by using the Network Contextualization tool (NCTool) [11]. The task is computationally heavy and can be applied as an offline policy optimization.

C. SMT and SAT Solver Techniques

Satisfiability Modulo Theories (SMT) solvers together with SAT solvers are well known techniques used also in policy analysis. The policy analysis introduced in [12] is a formal framework for the analysis of XACML policies that uses SMT as the reasoning mechanism and support policy refinement. A translator is defined for extracting relevant information and policy structure, for building the internal model, from the XACML policies. The SMT solver uses background theories for reasoning about the rules, and permits the reasoning over non-boolean attributes and functions that appear frequently in XACML policies. This analysis is able to detect and correct errors in the policies, and support other analysis like attribute-hiding.

Another policy analysis that uses SMT solver for XACML policies is introduced in [13]. The authors show how to automate useful policy analysis, where XACML policies are represented as logical formulae using the SMT subset of first-order logic, and the analysis queries are expressed as SMT problems, which are then solved by off-the-shelf SMT solvers. This technique claims to handle many policies that cannot be analysed by existing policy engines. Some of the analyses that can be performed are consistency, completeness and observational equivalence.

An analysis based on SMT solvers for Content-based Protection and Release (CPR) access control model for the NATO infrastructure is presented in [14]. The authors define a declarative language for CPR based on a first-order logical framework based on SMT solvers, whereby theories are used to model the algebraic structure of types. CPR language reuses results and techniques of [15] from SMT field, where policy analysis problems are reduced to SMT problems. In [15], SMT solvers are used for detecting redundancies and inconsistencies between policies in an extension of the Role-Based Access Control (RBAC) model. CPR refines RBAC in different aspects, one of the most important is that the access control policies are divided between those that identify the release conditions of the data, and the protection requirements. The decomposition enables an efficient implementation of the policies, by avoiding conflicting policies. The type of resource is not needed by the Policy Decision Point (PDP) of CPR, but can be delegated to an appropriate module of the NATO infrastructure. An implementation of the refinement in XACML architecture is also presented in [16].

D. Abductive Reasoning

Lately other policy analysis techniques were developed that uses abductive reasoning [17]. Abductive reasoning has mainly been used for policies represented using Event Calculus [18], and permits conflict, redundancies and gaps analyses. The captured conflicting policies are of different types: implicit, explicit, context dependent. Verification techniques based on abductive reasoning provide further information of the performed analyses, e.g., which are the missing cases for gap analysis, the repeated cases for redundant analysis, or the conflicting one for conflict analysis. The latter permits, when it is possible, an efficient conflict resolution.

Abductive reasoning is used alone or in conjunction with other techniques. The verification technique introduced in [19] uses *free variable tableaux* method, which is a sound and complete theorem prover, upon which abductive reasoning is built. This analysis detects conflicts and redundancies between policies. For performing the analysis, the policies are translated, from the used policy notation, such as Ponder, into first-order logic. This analysis identifies conflicting and redundant policies, as well as provides useful information for conflicts resolution.

Free variable tableaux in conjunction with abduction are used in [20] to analyse policies and their conflict detection for access control in web services environments. The authors define the notion of *implicit conflicts*, policies that are in conflict due to the propagation of the action composition and constraint policies that cannot be detected by simple comparing the policies. This method permits the static detection of conflicting policies, such as modality, implicit, and static constraints conflicts. The use of abduction enable the provision of information regarding the conflicting policies, which is very helpful for a future conflict resolution phase.

Abductive reasoning is used not only for giving information regarding conflicting policies but also for providing information for authorisation policies. In [21] is presented an abduction-based policy analysis, for explaining access grants and denials and automated delegation. An algorithm is provided, which works on policies specified in Datalog, and its results are extended to construct a copy of the proof graph during the evaluation. This algorithm is extended in [22] for analysing a rule-based policy framework with administrative policies that controls changes to the rules as well as the facts in the policies.

An expressive policy analysis language that permits the representation of different types of policies, authorisations and obligations is given in [23]. This language, based on event calculus [18], permits the representation of various policy constraints and the domain description predicates. The authors of [23] present a novel policy analysis based on *abductive constraint logic programming* [24] that identifies conflicting policies, redundant ones, coverage of gaps and permits behavioural simulation. The analysis is made directly to the language and no translation is required.

Abductive reasoning is also used in policy conflict analysis for quality of service (QoS) management. In [25] the authors point out that policy-based management has the ability to reconfigure the services networks, so that QoS goals are achieved. Some of the configurations are network provision decisions, admission control, dynamic bandwidth allocation etc. The policy based approach are flexible and adaptable as the policies can change without changing the implementation. Due to the complexity of the system, conflicts and inconsistencies may arise. The authors shows how conflicts are detected using event calculus in conjunction with abductive reasoning techniques. The authors also performs a network dimensioning conflict classification, which extends the work done in [26], [27]. The introduced conflict analysis in addition to detecting the conflicts, e.g., redundancy, mutual exclusion, bandwidth and routing conflicts, provide also an explanation to why the conflict occurred.

Argumentation-based reasoning [28] together with abductive reasoning can be used for policy analysis with incomplete, conflicting and context-dependent knowledge. The policy analysis introduced in [29]–[31] uses argumentation and abductive reasoning. This analysis through the use of abductive reasoning can identify conflicting policies, as well as redundant and missing ones. This analysis is implemented through the use of an abductive constraint logic programming system, A-system[1] [32]. The use of argumentation reasoning permits the analysis to find *context dependent conflicts*, that were not captured before by other types of policy analysis. A conflict resolution is also introduced, which permits the introduction of hierarchies between conflicting policies. The argumentation reasoning based analysis together with the conflict resolution are implemented using GorgiasB[2] [33], a tool for preference-based argumentation. In this case, no translation is needed, as the used policy language is easily implemented with the logic programming language used. A translation of other formalisms into this policy language is offered.

E. Analysis Based on Other Techniques

A controlled natural language that deals with data sharing agreements is proposed in [34]. A data sharing agreement represents the agreements between different parties that describes how the data should be treated. This language is translated into a *process algebra*, POLPA, that permits the execution of an analysis to the policies of the data sharing agreements. An executable specification of the process algebra is developed in Maude, which identifies inconsistencies and possible conflicting policies before their enforcement. The given analysis is strictly related to the proposed language.

Several aspects of knowledge such as causality, defaults and incomplete information are usually not considered by verification techniques[3]. For dealing with the above and providing a compact encoding of complex problems, the authors of [35] decide to introduce a policy analysis framework based on *Answer Set Programming* (ASP). They introduce a logic-based policy management approach for Web access control policies, focusing in XACML. A translation from XACML into ASP is performed and a policy analysis framework that performs policy validation, comparison and querying is constructed. They introduce a tool, called XACM2ASP, that works with ASP solvers, runs the analysis for a set of policies, and give a policy analysis method that identifies constraint violations in XACML-based policies.

F. Discussion of the Verification Techniques

Verification techniques are important for a correct and efficient implementation of the policy based systems. We notice that techniques that use model checking and SMT/SAT solvers are mainly used for performing policy refinement and safety analysis. Some of them are able to perform also redundancy and efficient analysis. Techniques based on algebraic solutions and abductive reasoning focus mainly on performing conflict detection and efficient analysis like completeness check and redundancies discovery.

One of the main drawback that these techniques suffer from is the significant need for computational power that causes them to generally run offline. Another drawback is that these techniques are bound to particular policy models and consequently have limited extensions to other policy models. On the one hand different techniques offer the policies conflicts detection, but on the other hand we notice a lack of efficient conflict resolution. For the latter, techniques like abductive reasoning with argumentation are offering policy conflict resolution, thanks to their nonmonotonic nature and the provided decision making explanations.

III. Future Research

A future research direction in policy verification is motivated by the need to manage systems that are becoming increasingly autonomous. Such systems are designed to operate with a degree of freedom from centralised control and to make decisions independently of a policy authority whilst retaining high-level management oversight to ensure that strategic goals and constraints are enforced. Our work aims to develop, or adapt, verification techniques that evaluate operational correctness whilst enabling support for decision-making autonomy in challenging or unforeseen conditions,

[1]A-system http://dtai.cs.kuleuven.be/krr/Asystem/

[2]GorgiasB http://gorgiasb.tuc.gr/

[3]Abductive-based verification techniques take into account incomplete information and notions of causality.

such as those encountered in scenarios that experience a rapid change in operational tempo.

As part of our planned research, we will introduce an approach called generative policy management that seeks to enable a model of policy management whereby computing resources can make their own policy decisions whilst adhering to higher-level rules administered by a human controller. As modern mobile computing systems gain processing capacity, their ability to reason with situational awareness improves, which enables a greater potential for reacting to operational change with a freedom of action. This autonomy introduces additional flexibility to systems that can make decisions and cope with unexpected situations.

It is in this context of autonomous decision-making that we plan to explore gaps in present-day verification techniques. One area of interest is in considering plausible approaches for constructing verification models in a modular fashion towards maximising the likelihood of verification completion, or minimising verification complexity. We aim to better understand if current verification techniques are amenable to modularisation and the steps needed to address any shortfalls. A further motivation aims to address the challenge of parallelising the verification process such that parallel parts could be processed simultaneously in a collaboration, towards improving the robustness (and scalability) of verification in an agile fashion. As part of this work, we will assess if contemporary techniques for parallel processing could be adopted and applied to verification methods. We aim to extend our research in the context of scenarios involving coalition operations and in doing so anticipate cross-cutting concerns with topics such as trust establishment and secure information management.

Finally, we view verification as a supporting role in the system assurance process. Traditionally, assurance depends on techniques to analyse and model systems both at design-time, in static form, and at run-time, with defined behaviour. The assurance process is conducted on systems that operate within well-defined bounds on their possible actions at run-time. Assurance is possible if the states of the system can be defined and the system is operationally predictable. On the other hand, a system the operates with some degree of autonomy is able to alter its run-time state in ways that may not have been predicted. This ability introduces uncertainty into the assessment process since it may not be possible to reason about every plausible state of operation by simply examining system specification. An improvement to the assurance process may be possible by extending the process of evaluating an agile system to examine changes in run-time state. By being part of the run-time control loop, this extension would have the facility to observe how the system adapts operationally and to assess the case for assurance based on these observations even where the system takes actions that were previously difficult to predict in advance. Another advantage of this extended approach is the potential for significantly reducing the requirements for state space evaluation, which is known to be a challenge for contemporary verification techniques such as model checking at design time. Certain verification techniques may be well-suited for evaluating correctness of autonomous systems, particularly those that are sufficiently flexible to cope with behavioural uncertainty. How assurance strategy might evolve given the emergent challenges of decision-making autonomy is a focus of our planned research.

IV. Conclusion

Verification techniques are an important component for policy based systems. They ensure the correct implementation of the policy based systems, by finding errors at an early stage and performing policy refinement. Verification techniques are able to evaluate the policy's validity and to identify conflicting policies as well as to perform evaluation and improvement of efficiency. In this work, we review some of the most significant verification techniques and discuss future research directions. An interesting direction is the use of verification techniques in autonomous systems, where we expect an increase of autonomous decision-making. Important future work includes the construction of verification techniques for generative policies. Assurance is another aspect where verification plays an important role: future work is needed to establish a clearer understanding of what assurance is and how policy verification techniques may be used to assure correct system operation.

Acknowledgments

Supported by EPSRC Project CIPART grant no. EP/L022729/1. This research was sponsored by the U.S. Army Research Laboratory and the U.K. Ministry of Defence under Agreement Number W911NF-16-3-0001. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the U.S. Army Research Laboratory, the U.S. Government, the U.K. Ministry of Defence or the U.K. Government. The U.S. and U.K. Governments are authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation hereon.

References

[1] N. Zhang, M. Ryan, and D. P. Guelev, "Evaluating access control policies through model checking," in *ISC*, 2005, pp. 446–460.

[2] S. Kikuchi, S. Tsuchiya, M. Adachi, and T. Katsuyama, "Policy verification and validation framework based on model checking approach," in *ICAC'07*, 2007, pp. 1–9.

[3] S. Ranise, A. T. Truong, and R. Traverso, "Parameterized model checking for security policy analysis," *STTT*, vol. 18, no. 5, pp. 559–573, 2016.

[4] A. Armando and S. Ranise, “Scalable automated symbolic analysis of administrative role-based access control policies by SMT solving,” *Journal of Computer Security*, vol. 20, no. 4, pp. 309–352, 2012.

[5] S. Ranise, A. T. Truong, and A. Armando, “Boosting model checking to analyse large ARBAC policies,” in *STM*, 2012, pp. 273–288.

[6] M. Bartoletti, P. Degano, G. L. Ferrari, and R. Zunino, “Model checking usage policies,” *Mathematical Structures in Computer Science*, vol. 25, no. 3, pp. 710–763, 2015.

[7] J. Rubio-Loyola, J. Serrat, M. Charalambides, P. Flegkas, G. Pavlou, and A. Lluch-Lafuente, “Using linear temporal model checking for goal-oriented policy refinement frameworks,” in *POLICY*, 2005, pp. 181–190.

[8] N. Khakpour, R. Khosravi, M. Sirjani, and S. Jalili, “Formal analysis of policy-based self-adaptive systems,” in *ACM SAC*, 2010, pp. 2536–2543.

[9] H. Zhao, J. Lobo, and S. M. Bellovin, “An algebra for integration and analysis of ponder2 policies,” in *POLICY*, 2008, pp. 74–77.

[10] C. Basile, A. Cappadonia, and A. Lioy, “Network-level access control policy analysis and transformation,” *IEEE/ACM Trans. Netw.*, vol. 20, no. 4, pp. 985–998, 2012.

[11] C. Basile, A. Lioy, and M. Vallini, “Towards a network-independent policy specification,” in *PDP*, 2010, pp. 649–653.

[12] F. Turkmen, J. den Hartog, S. Ranise, and N. Zannone, “Analysis of XACML policies with SMT,” in *POST, ETAPS*, 2015, pp. 115–134.

[13] K. Arkoudas, S. Loeb, R. Chadha, C. J. Chiang, and K. Whittaker, “Automated policy analysis,” in *POLICY*, 2012, pp. 1–8.

[14] A. Armando, S. Oudkerk, S. Ranise, and K. S. Wrona, “Formal modelling of content-based protection and release for access control in NATO operations,” in *FPS*, 2013, pp. 227–244.

[15] A. Armando and S. Ranise, “Automated and efficient analysis of role-based access control with attributes,” in *DBSec*, 2012, pp. 25–40.

[16] A. Armando, M. Grasso, S. Oudkerk, S. Ranise, and K. S. Wrona, “Content-based information protection and release in NATO operations,” in *SACMAT*, 2013, pp. 261–264.

[17] A. C. Kakas, R. A. Kowalski, and F. Toni, “Abductive logic programming,” *J. Log. Comput.*, vol. 2, no. 6, pp. 719–770, 1992.

[18] R. Kowalski and M. Sergot, “A logic-based calculus of events,” *New Gen. Comput.*, vol. 4, no. 1, pp. 67–95, 1986.

[19] H. Kamoda, M. Yamaoka, S. Matsuda, K. Broda, and M. Sloman, “Access control policy analysis using free variable tableaux,” *Information Processing Society of Japan (IPSJ)Digital Courier*, vol. 2, pp. 207–221, 2006.

[20] H. Kamoda and K. Broda, “Policy conflict analysis using free variable tableaux for access control in web services environments,” in *In Policy Man. for the Web*, 2005, pp. 5–12.

[21] M. Y. Becker and S. Nanz, “The role of abduction in declarative authorization policies,” in *PADL*, P. Hudak and D. S. Warren, Eds., 2008, pp. 84–99.

[22] P. Gupta, S. D. Stoller, and Z. Xu, “Abductive analysis of administrative policies in rule-based access control,” in *ICISS*, S. Jajodia and C. Mazumdar, Eds., 2011, pp. 116–130.

[23] R. Craven, J. Lobo, J. Ma, A. Russo, E. C. Lupu, and A. K. Bandara, “Expressive policy analysis with enhanced system dynamicity,” in *ASIACCS*, 2009, pp. 239–250.

[24] A. C. Kakas, R. A. Kowalski, and F. Toni, “Abductive logic programming,” *J. Log. Comput.*, vol. 2, no. 6, pp. 719–770, 1992.

[25] M. Charalambides, P. Flegkas, G. Pavlou, A. K. Bandara, E. C. Lupu, A. Russo, N. Dulay, M. Sloman, and J. Rubio-Loyola, “Policy conflict analysis for quality of service management,” in *POLICY ’05*, 2005, pp. 99–108.

[26] E. Lupu and M. Sloman, “Conflict analysis for management policies,” in *IFIP/IEEE International Symposium on Integrated Network Management*, 1997.

[27] J. D. Moffett and M. S. Sloman, “Policy conflict analysis in distributed system management,” *Journal of Organizational Computing*, vol. 4, no. 1, pp. 1–22, 1994.

[28] A. Bondarenko, P. M. Dung, R. A. Kowalski, and F. Toni, “An abstract, argumentation-theoretic approach to default reasoning,” *Artif. Intell.*, vol. 93, pp. 63–101, 1997.

[29] E. Karafili, A. C. Kakas, N. I. Spanoudakis, and E. C. Lupu, “Argumentation-based security for social good,” 2017. [Online]. Available: https://arxiv.org/abs/1705.00732

[30] E. Karafili and E. C. Lupu, “Enabling data sharing in contextual environments: Policy representation and analysis,” in *SACMAT ’17*. ACM, 2017, pp. 231–238.

[31] E. Karafili, E. C. Lupu, S. Arunkumar, and E. Bertino, “Argumentation-based policy analysis for drone systems,” in *DAIS Workshop, IEEE SmartWorld Congr.*, 2017 (to appear).

[32] B. V. Nuffelen and A. C. Kakas, “A-system: Declarative programming with abduction,” in *LPNMR*, 2001, pp. 393–396.

[33] A. Kakas and P. Moraitis, “Argumentation based decision making for autonomous agents,” in *AAMAS*, 2003, pp. 883–890.

[34] I. Matteucci, M. Petrocchi, and M. L. Sbodio, “CNL4DSA: a controlled natural language for data sharing agreements,” in *ACM SAC*, 2010, pp. 616–620.

[35] G. Ahn, H. Hu, J. Lee, and Y. Meng, “Representing and reasoning about web access control policies,” in *COMPSAC*, 2010, pp. 137–146.