Generative Policies for Coalition Systems – A Symbolic Learning Framework

Elisa Bertino*, Alessandra Russo†, Mark Law†, Seraphin Calo‡, Irene Manotas‡, Dinesh Verma‡, Amani Abu Jabal*, Daniel Cunnington§, Geeth de Mel§, Graham White§, Jorge Lobo¶, John Ingham‖, Gregory H. Cirincione**

*Department of Computer Science, Purdue University, West Lafayette, IN, USA
†Imperial College, London,UK
‡IBM TJ Watson Research Center, Yorktown Heights, NY, USA
§IBM Research UK
¶ICREA - Universitat Pompeo Fabra, Spain
‖Dstl, UK **Army Research Lab, Adelphi, MD, USA
Email: *{bertino,aabujaba}@purdue.edu, †{arusso,mark.law09}@imperial.ac.uk,
‡{scalo, irene.manotas, dverma}@us.ibm.com, §{dancunnington,geeth.demel,gwhite}@uk.ibm.com,
¶jorge.lobo@upf.edu, ‖jmingham@mail.dstl.gov.uk, **gregory.h.cirincione.civ@mail.mil

***Abstract*—Policy systems are critical for managing missions and collaborative activities carried out by coalitions involving different organizations. Conventional policy-based management approaches are not suitable for next-generation coalitions that will involve not only humans, but also autonomous computing devices and systems. It is critical that those parties be able to generate and customize policies based on contexts and activities. This paper introduces a novel approach for the autonomic generation of policies by autonomous parties. The framework combines context free grammars, answer set programs, and induction-based learning. It allows a party to generate its own policies, based on a grammar and some semantic constraints, by learning from examples. The paper also outlines initial experiments in the use of such a symbolic approach and outlines relevant research challenges, ranging from explainability to quality assessment of policies.**

***Index Terms*—Intelligent Systems, Context Awareness, Logic, System Management, Coalitions**

I. Introduction

Many next-generation collaborative activities and missions will be carried out by coalitions that will include autonomous groups of devices and systems with a large variety of cognitive capabilities. These devices and systems will have to operate in environments characterized by uncertainty, insecurity (both physical and cyber), variability, and instability. In such environments, communications may be fragmented. It is thus critical that coalition devices and systems have the capability to be self-adaptive and evolve. In addition, those devices and systems will collaborate with humans for a variety of coalition tasks, and thus may have to be able to explain their actions/decisions; and, depending on the specific application domain, understand boundaries that may limit their actions. Such boundaries are particularly crucial for applications in which safety is a critical requirement [1]. They may also have to comply with regulations and legal directives.

Addressing such challenges requires proper policy-based management of coalition parties - be these parties devices, systems, or humans. Policies can simplify the complex task of managing decentralized coalitions. According to Bertino, et al. [2], policies can be seen as directives given by a managing party to one or more managed parties in order to guide their behavior in coalition missions and collaborative activities. Different types of policies can be identified, including: (i) *Constraint policies* that impose constraints on the activities that the managed parties execute – notable examples being represented by access control policies [3] and firewall policies; (ii) *Goal-based policies* that direct the managed parties to achieve a specific goal, e.g., maintain a minimum threshold of utilization or try to finish a task before a specific deadline; and (iii) *Utility-based policies* that direct the managed parties to produce the best consequence according to some value function, such as for example maximizing the usage of certain resources [2].

Policies are usually expressed as technology independent rules aiming to enhance the hard-coded functionality of the managed parties by the introduction of an interpreted logic that can be dynamically changed without modifying the underlying implementation. Policy-based management thus significantly increases the self-managing aspects of coalition operations. Because of the usefulness of policy-based management, policy models, languages, formalisms, and systems have been widely investigated and applied to many different domains, including access control [3], firewall systems, and more recently software defined networks [4]. Policy standards have been developed, most notably in the areas of access control (e.g., the eXtensible Access Control Markup Language (XACML) and RBAC standards).

However a critical issue in policy-based management is represented by the specification of the proper policies to be used as input for the policy enforcement mechanisms. Conventional approaches to policy specification are typically based on a top-down approach by which policies are specified through multiple refinement steps by some centralized policy administrator. However such an approach is human-intensive and in addition reduces the autonomy and flexibility of the managed parties – autonomy and flexibility are critical for coalitions operating in distributed and dynamic settings. To address such a challenge, Verma, et al. [5] proposed the notion of a generative policy architecture, by which managed parties are provided an initial policy specification. Each managed party can then (dynamically) generate its own policies from the initial specification, possibly evolve them over time, and based on its own "customized" policies take decisions about its own actions. The generative policy architecture addresses the requirements of autonomous management and flexibility for coalitions [2].

A generative policy architecture, however, requires approaches by which the managed parties can generate and evolve their own policies. There are two main approaches that can be adopted. Both leverage recent advances in big data and learning technologies for data analysis. The first approach is based on the use of statistical learning techniques by which a classifier is learned from data concerning past policy decisions. An earlier example of such an approach is in the area of access control [6]. The second approach is based on the use of symbolic learning techniques by which logic rules are learned from a set of examples through the use of induction [7]. The latter approach has the major advantage of providing a high-level representation of the learned policies, which in turn supports accountability and explainability of policy decisions. In addition, because policies are expressed according to a symbolic formalism, it is easy to support similarity-based policy adaptation, especially when additional semantic knowledge is available in the form of ontologies and other conceptual representations of domains of interest.

In this paper we put forward a novel framework to generate and evolve policies based on:

- *The notion of an answer set grammar (ASG).* A novel type of formal grammar combining context-free grammars with answer set programs (ASP) [8], declarative programs oriented toward solving search problems.
- *ILASP.* An inductive learner that automatically learns ASPs from examples [9].

Context-free grammars specify the syntactical components of a given class of policies, whereas the ASPs represent semantic constraints imposed on the generation of the policies from the grammar. Such constraints allow one, for example, to prevent the generation of policies that, even though syntactically correct, would not be acceptable in certain contexts. The use of such novel grammars in combination with the ability to automatically learn ASPs from examples allows one to easily generate, adapt and evolve policies by providing context-specific examples.

The full design, development and deployment of our framework, referred to as the AGENP (An ASGrammar-based GENerative Policy) framework, require, however, addressing several challenges. In this paper, we aim at sharing our vision and initial experience and results on the use of our approach and develop a research roadmap based also on initial experimental applications.

The rest of the paper is organized as follows. Section II provides details about the notion of an ASG, whereas Section III describes an initial architecture that implements our framework. Section IV reports results from on-going applications of AGENP to learn policies in different areas, some of which are still in an early stage. Throughout those sections we outline research issues. Sections V and VI conclude the paper by discussing additional research directions and outlining a few conclusions.

II. Formalizing Generative Policy Models

In this section we describe our new approach to formalizing and learning generative policy models (GPMs), which is based on the notion of ASG which uses ASP rules to annotate a context-free grammar, yielding a context-sensitive grammar.

The main requirement of any representation of a GPM is that we must be able to use it to generate the set of policies which are valid in any given context. ASGs consist of two main components: (1) a context-free grammar (CFG), which in our generative policy setting is used to specify the syntax of the underlying policy language; and (2) a set of ASP rules which, when combined with a context, specify which of the syntactically valid policies are appropriate in the current context. This separation of syntactic conditions enforced by the CFG and the more semantic conditions enforced by the ASP rules is important. The underlying syntax of the policy language is fixed, and known (e.g. if our setting requires the generation of XACML policies, then the language of our CFG will be the set of all valid XACML policies), whereas the set of policies which are valid in a particular context is domain dependent and may not be known, or might even change over time. For this reason, our approach to *learning* ASGs does not attempt to learn the production rules of the CFG, as this would be learning the (known) syntax of the policy language, but instead learns the semantic ASP conditions. The reason for choosing ASP to represent the semantic conditions is that we can use an existing inductive learner for ASP [10] to learn these conditions.

A. Answer Set Programming and Answer Set Grammars

In the current version of our framework we use a subset of ASP consisting of normal rules and constraints. Given any atoms $\mathtt{h}$, $\mathtt{b}_1, \ldots, \mathtt{b}_n$, $\mathtt{c}_1, \ldots, \mathtt{c}_m$, a *normal rule* is of the form $\mathtt{h}\texttt{:-}\mathtt{b}_1, \ldots, \mathtt{b}_n$, $\mathtt{not}\ \mathtt{c}_1, \ldots,$ $\mathtt{not}\ \mathtt{c}_m$, where $\mathtt{h}$ is the *head*, $\mathtt{b}_1, \ldots, \mathtt{b}_n$, $\mathtt{not}\ \mathtt{c}_1, \ldots,$ $\mathtt{not}\ \mathtt{c}_m$ (collectively) is the *body* of the rule, and "not" represents negation as failure. Rules without a *head*, i.e., rules of the form $\texttt{:-}\mathtt{b}_1, \ldots, \mathtt{b}_n$, $\mathtt{not}\ \mathtt{c}_1, \ldots,$ $\mathtt{not}\ \mathtt{c}_m$, are called *constraints*. Given any ground (variable free) program P, an interpretation I, i.e., a subset of the atoms in P, is said to be a *model* of P if for every rule R in P whose body is satisfied by I, the head of R is satisfied by I. Note that as constraints have an empty head, this means that no model of P can satisfy the body of any constraint in P. The models of a non-ground

ASP are equal to the models of the ground instantiation of the program where each rule is replaced with its ground instances. Solutions of ASP programs are called *answer sets*, which are a special subset of the models of a program. For a full definition of the answer set semantics, see [8].

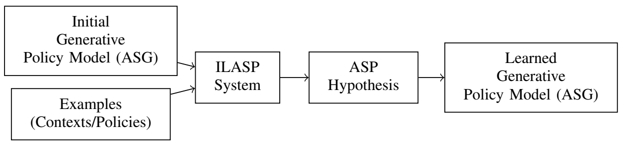
A context-free grammar [11], G, is a tuple $\langle G_N, G_T, G_{PR}, G_S\rangle$ where G_N is a (finite) set of non-terminal nodes, G_T is a (finite) set, disjoint from G_N, of terminal nodes, G_{PR} is a set of production rules of the form $n_0 \rightarrow n_1 \ldots n_k$, where $n_0 \in G_N$ and each $n_i \in G_N \cup G_T$. $G_S \in G_N$ is the start node of G. The terminal nodes of a grammar correspond to the characters of the alphabet that appear in the strings generated by the grammar. The root of a parse tree for any CFG G is the start node G_S. For any node n of a parse tree such that $n \in G_N$, the (ordered) children of n must be the list of nodes on the right hand side of a production rule in G_{PR} whose left hand side is n. The nodes in a parse tree which are in G_T have no children. A grammar G accepts a string s if there is at least one parse tree for which the concatenation of the list of terminal nodes in the parse tree (read depth-first from left to right) is equal to s.

ASGs extend CFGs by allowing each production rule to be given semantic conditions, written in ASP. To allow the semantic conditions to refer to the structure of the CFG, the atoms in these ASP programs have annotations that refer to nodes of the parse tree of a CFG. Specifically, an *annotated ASP program* is an ASP program where some atoms have been *annotated* with a ground term. For instance, the annotated atom `a(1)@2` represents the atom `a(1)` with the annotation 2. When computing the answer sets of an annotated program, annotated atoms are treated as ordinary atoms, where `a@k`, `a@l` and `a` are distinct atoms. We now recall the notions of annotated production rules and ASGs from [12].

Definition 1. *An* annotated production rule *is of the form $n_0 \rightarrow n_1 \ldots n_k$ P where $n_0 \rightarrow n_1 \ldots n_k$ is an ordinary CFG production rule and P is an annotated ASP program, where every annotation is an integer between* 1 *and k.*

Definition 2. *An* answer set grammar *G is a tuple $\langle G_N, G_T, G_{PR}, G_S\rangle$ where G_N is a (finite) set of non-terminal nodes, G_T is a (finite) set, disjoint from G_N of terminal nodes,. G_{PR} is a set of annotated production rules and $G_S \in G_N$ is the start node of G.*

The language of any ASG, $\mathcal{L}(G)$, is a subset of the language of its underlying CFG, G_{CF} (the CFG constructed by removing the annotations from every production rule in G). To decide whether a string $\mathtt{s} \in \mathcal{L}(G_{CF})$ is in the language of G, we must consider the parse trees of G_{CF} for $\mathtt{s}$. We can represent each node n in a parse tree by its trace, $trace(n)$, through the tree. The trace of the root is the empty list $[]$; the i^{th} child of the root is $[i]$; the j^{th} child of the i^{th} child of the root is $[i, j]$, and so on. In [12], a mapping is defined from any parse tree of any ASG to an ASP program. Let G be an ASG and PT be a parse tree. $G[PT]$ is the program $\{rule(n)@trace(n) | n \in PT\}$, where for any production rule $n_0 \rightarrow n_1 \ldots n_k$ P, and any trace t, $PR@t$ is the program constructed by replacing all annotated atoms `a@i` with the atom `a@(t++[i])` and all unannotated atoms `a` with the atom `a@t`. A string `s` is in the language of G, denoted $\mathtt{s} \in \mathcal{L}(G)$, if there is at least one parse tree PT of G for `s` such that the program $G[PT]$ has at least one answer set.



Initial Generative Policy Model (ASG)
Examples (Contexts/Policies)
ILASP System
ASP Hypothesis
Learned Generative Policy Model (ASG)

Fig. 1: The workflow for learning ASGs with ILASP.

B. Learning Generative Policy Models as Answer Set Grammars

In this section we show how the algorithm for learning ASGs from examples of strings [12] can be used to learn ASG-based GPMs from examples of which policies are valid under some contexts. Figure 1 shows the workflow for the learning process. We start with an initial GPM (i.e., ASG), and examples of which policies are valid under which contexts. We then use an ASG learning algorithm, which transforms the learning problem into a task that can be solved by the ILASP [10], [13] system. Each solution of the ILP task then corresponds to an ASG-based GPM that is consistent with the examples. Full details of the ASG-learning algorithm, and the transformation of an ASG learning task into an ILASP task are presented in [12].

Definition 3 presents a generalization of the ASG learning task presented in [12], which has been upgraded to support the notion of context-dependent examples necessary to learn ASG-based GPMs. For any ASG G and ASP program C, we write $G(C)$ to denote the grammar constructed by adding C to the annotation of every production rule in G. An ASG hypothesis space S_M is the set of rules which can be learned, where each rule in S_M also contains a set of identifiers specifying which production rules it can be added to. Given a hypothesis H, where each element of H is of the form $\langle h, pr_{id}\rangle$ such that $h \in H$ and pr_{id} is the identifier of a production rule in G_{PR}, we write $G : H$ to denote the ASG constructed from G by adding each rule in H to the annotation of the relevant production rule in G_{PR}.

Definition 3. *A context-dependent ASG learning task T is of the form $\langle G, S_M, E^+, E^-\rangle$, where G is an ASG called the* initial grammar, *S_M is an ASG hypothesis space and E^+ and E^- are sets of pairs of the form $\langle \mathtt{s}, C\rangle$, where $\mathtt{s}$ is a string and C is an ASP program, called the* positive *and* negative *examples, respectively. An* inductive solution *of T is a hypothesis $H \subseteq S_M$ such that*

1) $\forall\langle \mathtt{s}, C\rangle \in E^+, s \in \mathcal{L}(G(C) : H)$
2) $\forall\langle \mathtt{s}, C\rangle \in E^-, s \notin \mathcal{L}(G(C) : H)$

III. The AGENP Framework

In this section we present the ASGrammar-based GENerative Policy (AGENP) framework. Specifically, we first describe

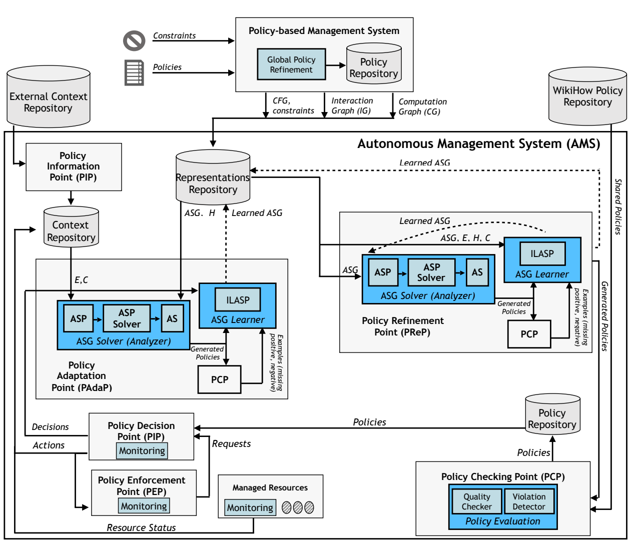


Fig. 2: The Generative Policy-based Model (GPM)

how the key components in the AGENP framework interact to produce the policies for Autonomous Managed Systems, as well as how the Answer Set Grammar (ASG) and Inductive Logic Programming (ILP)-based learning mechanisms presented in Section II are used inside the AGENP components that adjust and generate the policies. We then discuss research directions related to the architecture.

A. *Key Components of AGENP*

Figure 2 shows the AGENP architecture. Its major components include: a Policy Refinement Point (PReP), a Policy Adaptation Point (PAdaP), a Policy Checking Point (PCP), a Policy Information Point (PIP), a Policy Repository, a Policy Decision Point (PDP), and a Policy Enforcement Point (PEP). The overall flow that is represented in the architecture starts with a Policy-based Management System (PBMS) providing a characterization of the policy space within which the Autonomous Managed System (AMS), that is, an autonomous coalition party, will operate in terms of a CFG, goals, and constraints. The AMS is only free to generate policies that are captured in the language of the CFG and comply with the high level constraints. The PReP takes the information provided by the PBMS and produces an ASG that is pertinent to the context within which the AMS is operating. The PReP then uses the ASG to learn its GPM and generates the policies for the AMS which are captured in the Policy Repository.

The PEP, PDP, and Policy Repository operate in a manner similar to conventional PBMS. When the managed parties require a decision to be made regarding their operation, the PDP obtains all the policies pertinent to that decision and uses them to determine the actions that must be performed by the PEP on the managed parties. In the AGENP architecture, however, the operations of the PDP and PEP are monitored to produce a history of the decisions that have been made, the actions that have been taken, and the effects that they have had on the state of the system. This information is used by the PAdaP to update the ASG-based GPM when deemed necessary. Such an update would be triggered if the operation of the system is not meeting the goals set by the global PBMS, or there has been a change in context. The PReP would then get this latest learned ASG and generate policies from the updated policy model.

Below we describe how the PAdaP and PReP components use the ASG representation and ILP learning, described previously, to generate policies and to learn the generative policy model. We also present details about the additional components in the AGENP architecture that allow the Automated Managed System (AMS) to check the quality and consistency of the policies, and to obtain external information to make more informative decisions.

1) Adaptation and Refinement using the ASG Solver and Learner: In the AGENP framework, a GPM is represented as an ASG that when combined with a given context represents a language that corresponds to the set of policies generated by the GPM. An ASG, denoted G, allows sets of facts to be added to the production rule(s) for the start node of G. Depending on which facts are added, the language of G will contain different strings. Valid strings that are in the language of G under a context C are denoted by the language $G(C)$. Thus, $G(C)$ is the grammar constructed by adding the facts in C to the annotation of every production rule for G. These *context-dependent* ASGs provide a natural representation for the GPM.

The PAdaP component, shown in Figure 2, adapts policies according to the effects on the system of previous decisions taken by the PDP component. To adapt policies, the PAdaP analyzes context information (e.g., resources, actions/effects, external conditions), the previous learned policy model (e.g., ASG), and previously selected policies, to generate, validate, and update the ASG. More specifically, the PAdaP is composed of the *ASG Solver (Analyzer)*, and the *ASG learner*. Generated policies from the ASG learner are sent to the PCP component which evaluates their quality and identifies policies that incur violations (e.g., as determined by negative policy examples). Feedback from the PCP is used by the ASG learner, along with information about additional annotated policy examples, context information, and the definition of a hypothesis space — which represents the set of learnable rules— to learn a new generative policy model that represents the current valid AMS ASG. The learned ASG is stored in the representations repository so that the PAdaP can access the latest representation of the ASG-based generative policy model.

2) Quality Assessment and Validation of Policies: The goal of the PCP component of the framework is to check the quality and validity of the generated policies received internally, or of external policies shared by other AMSs in a collaborative environment. Quality of policies can be checked by the *Quality* component inside the PCP by considering different policy metrics and policy quality requirements such as consistency, completeness, relevance, and minimality [14].

Furthermore, the PCP has a *Violation Detector* component, where policies are checked for validity, i.e., policies nonconforming to the local context, or having inconsistencies, similar to policy conformance mechanisms, such as policy testing and verification proposed for XACML policies [15]. The PCP component analyzes the quality and validity of generated policies from the ASG-based model learned in the PReP, and also of the policies received from an external policy repository.

3) External Context Information and Shared Policies: The Policy Information Point (PIP) component aims to acquire information about any external conditions that affect the operation of the AMS. These are used internally by the PAdaP component in the AMS to adapt the local policies so that they reflect pertinent knowledge about the external environment. These conditions thus influence the set of policies to be considered for local operation. Shared policies could be used by the AMS's GPM to leverage policies learned in different contexts by other trusted AMSs in a collaborative fashion, as is proposed in community-based policy learning (*CASWiki*) by Bertino, et al [16]. In the *CASWiki*, agents, primarily autonomous systems such as Connected and Autonomous Vehicles (CAVs) or Internet of Things (IoT) devices, contribute policies to a shared knowledge base. Policies shared by different agents implicitly contain knowledge learned from the application of policies in different contexts, thus contributing to the joint knowledge of different AMSs working in collaboration.

B. Research Directions

The AGENP framework has been designed with the goal of supporting the generation and evolution of policies for autonomous parties. Although the ASG-based generative policy model representation and learning through ILP help to fulfill this goal, many challenges are still open for investigation. Here we describe some of those challenges. One is *Performance Optimization* - since autonomous parties are in many cases devices that need to respond in real-time, the adaptation and learning of GPMs has to be fast enough to be able to respond in a timely manner to requests made to the AMS. Thus, policy adaptation and learning algorithms need to consider the characteristics of the devices where the GPM is being evolved, and take into account any resource constraints. Another challenge is *Effective Multi-party Collaboration* - AGENP's design enables it to be instantiated for multi-party systems, such as subsets of parties in a coalition, for which efficient mechanisms are required to communicate and share policies. Finding the most appropriate way parties can communicate (e.g., via a communication language and coordination mechanisms, like the ones developed for multi-agent systems [17]) needs to be further explored. *From natural language to grammar-based policies* - policies are initially defined by end users or organizations in natural language, and in terms of goals that they want the system to achieve. These constructs must be transformed into the grammars that are the basis of the generative policy approaches being investigated. Automatically or semi-automatically transforming intents and constraints into grammars that capture the space of admissible policies, would facilitate the interaction of end users with the policy-based management system being developed in AGENP. Previous work in software engineering [18], programming languages [19], and security [20] has investigated similar techniques but empirical analyses and further investigation of alternative approaches is required.

IV. A Sample of Applications

In this section we briefly discuss a few applications in which we have used or plan to use our symbolic framework for generative policies. The variety of applications shows that there are many different domains for which flexible and autonomous policy management is critical.

A. Autonomous Systems

Deploying autonomous systems for enabling or enhancing tasks —especially in support of human activities has become a reality [21], [22]. In such situations, autonomous systems execute specific tasks in a collaborative manner, reducing the cognitive burden on human users. In order to enable such cooperation between humans and machines, autonomous systems require a taxonomy that defines their full range of autonomous capability. One such specification is the *Autonomy Levels for Unmanned Systems (ALFUS) Framework* [23]. This defines various *levels of autonomy* ranging from human remote control (*Level 0*) to full autonomy (*Level 10*), where the system approaches no human interaction—only the resulting output is communicated. Intermediate levels are also described such as *Level 6* where a system can follow directives issued by a human operator that may include goal setting and decision approval.

The Society of Automotive Engineers (SAE) has also outlined their own autonomous specification similar to the ALFUS framework that outlines the full range of autonomy for CAVs [24], enabling CAV manufacturers to assign a Level of Autonomy (LOA) to a given vehicle. However, due to varying and possibly conflicting state and governmental policies, as well as a wide range of possible environmental conditions, assuming a static LOA proposes a challenge for a CAV. Furthermore, in local situations authorities may enforce transient autonomy levels to aid the management of a given situation, such as maintenance works or emergency vehicle scenarios. Therefore, autonomous systems in such environments require means to dynamically adapt their local and global policies and modify their behavior given the varying contexts. Also in a future intelligent transportation environment, CAVs of lower LOA may be able to utilize capabilities or services from nearby CAVs of higher LOA to accomplish tasks such as sensing and monitoring of the local environment if certain constraints and situations prohibit autonomous operation of lower LOA CAVs. The feasibility of these enhanced capabilities will require policy sharing and will also be subject to temporal, spatial, and utility constraints.

Cunnington *et. al* have proposed an ASG based GPM for CAVs [25]. This enables a CAV to learn a policy model that states whether a particular request to execute a driving task should be *accepted* or *rejected*, based on the current environmental conditions and the LOA of the vehicle, region and driving task. The authors have also shown that the ASG based GPM outperforms shallow Machine Learning (ML) techniques when learning complex policy models, as fewer examples are required to achieve a greater accuracy. An important research direction in this context includes utilizing the distributed nature of a coalition of CAVs alongside road-side infrastructure and other entities in the driving environment to support collaborative policy management, dynamic real-time policy adaptation, and decision making whilst adhering to temporal and spatial constraints. As policy adaptation may require generating new policies via the symbolic learner, it is also important to optimize the learner so that it can meet the real-time requirements of this domain.

B. Logistical Resupply

An interesting general military scenario set in the 2035 to 2050 time frame has been recently developed as part of the DAIS-ITA [26]. The scenario includes several examples of collaborative activities undertaken by military coalitions. One of the mission scenarios focuses on logistical resupply activities and describes coalition forces in an urban environment being regularly resupplied. In the scenario, a resupply convoy consisting of delivery vehicles supported by escort vehicles and drones must follow one of a set of route options, at some time of day and under certain assumed or predicted conditions that may take different values when the mission is under way.

Each resupply mission can be broken into phases including planning, reconnaissance, resupply and recovery of vehicles back to base. These phases give rise to two distinct times when policies are needed, a planning phase (planning and reconnaissance) and an execution phase (resupply and recovery). When considering policies, each phase is distinct. The planning phase is in advance of the mission and contains speculative information with varying degrees of accuracy such as the expected weather conditions. The execution phase contains real-time values that are likely to differ from the planning phase due to updated information and other external factors such as the enemy employing disruption tactics.

We recognize that at the start of any engagement, information may be low and the number of training samples will be in short supply. However, as time progresses and missions take place the learning tasks should become easier and more accurate as more training samples become available. As such, the coalition is able to learn from previous experience.

The learning task and hence generative policy for this scenario could take many forms. The policy could focus on areas such which route the convoy should take, at what time the convoy should travel, how the convoy should be made up (ratio of delivery vehicles and the value of supplies to the number of escort vehicles and their strength) and a variety of other similar options. The task is also likely to take into account the military risk appetite at the time of each mission. It may be, for example, that some options that were previously discounted on grounds of risk may later become acceptable due to a variation in risk appetite within the coalition.

We believe that our framework is well suited for learning policies for logistical resupply applications which are relevant not only for military coalitions but also for smart cities. One important issue that this scenario has highlighted is that decisions concerning logistical resupply may be characterized by large number of factors, and also the contexts may rapidly change. Therefore, techniques to enhance the scalability and real-time generation of policies is critical.

C. Access Control Policies

The specification of access control policies is quite challenging especially for complex policies [27], [28] like the ones that can be expressed in XACML [29]. One approach to address such a challenge is to use symbolic learning for learning policies from examples. For access control, the most suitable form of examples is represented by access requests and corresponding access control responses (i.e., decisions). Such a form of examples is suitable in many real-world situations. For example, when an organization has logs of past decisions taken by administrators, these logs can be used as examples to learn policies so that access control can be automated. In other cases, an organization may have automated access control but based on low-level models, e.g., models that do not support attribute-based access control, and the organization is interested in adopting a richer access control model. Logs of past decisions taken by the low-level mechanism can be used as examples to learn policies expressed into the higher-level model of interest.

To explore the feasibility and issues in using our symbolic learning framework for learning access control policies, we ran a case study on XACML policies. Using a public dataset of requests and responses of XACML policies[1], we generated a set of examples. Each example consists of an access request, in turn consisting of values of attributes of subject, resource, action, and context, and the corresponding decision. Then, we gave this example dataset as input to the ASG learner for learning the corresponding XACML policies. Fig. 3a[2] shows a sample of the policies that were learned correctly.

The experiment has shown that the characteristics of the examples greatly affect the correct learning of the policies. Of course, this was not unexpected. However, the experimental results indicate specific issues in the overall learning process. Namely, these issues are related to *overfitting learning* and *noisy example dataset*. Overfitting refers to the situation when the learned model corresponds too closely (or exactly) to the example dataset and, therefore, fails to predict future observations [30]. In particular, when the generated access control policies are only appropriate for scenarios similar to the ones in the example dataset, such policies are generated by an "overfitted" learned model. Therefore in order to be

[1]https://github.com/att/XACML/tree/master/XACML-TEST/src/test/resources/testsets/conformance/xacml3.0-ct-v.0.4

[2]The policies shown in Fig. 3a are considered correct because they are similar to the original policies in the public dataset.

applicable to other contexts the learned model should learn some "general" insights from the example dataset. The main issue is that the learned mode should be "safe" to minimize risks arising from policies that have un-intended consequences. Therefore, learning with "safe generalization" is essential to balance the trade-off between security and overfitting. Noisy example datasets, that is, datasets including"low quality" examples such as inconsistent responses to similar requests, result in patterns being missed by the learning process. In what follows, we discuss approaches to address these issues.

One method to avoid overfitting is to augment the example dataset with *background knowledge* that provides either statistical observations on the example dataset and/or on the context of the system of interest. Examples of contextual knowledge include role hierarchies and role assignments of subjects to these roles. For example, prior knowledge about the role of a user makes it possible to generate policies that are relevant to the role of the user rather than generating policies fitting only that specific user.

It is, however, important to point out that sometimes, using one piece of background knowledge for generalization may lead to security risks (i.e., to unsafe generalization). For example, suppose that an organization has many users with the *DBA* role while the example dataset shows that only few of these users were granted a specific permission p_i. In this case, it is risky to generate a policy that grants the *DBA* role the permission p_i. Using some statistics of the example dataset (e.g., the number of examples where the users of a specific role was granted a certain permission) can help reducing security risks. Therefore, gathering statistical information on the example dataset and contextual information can help one prioritizing the examples by assigning weights to them or to associate confidence values with the generated policies.

Example: Policy 1 in Fig. 3b shows a case in which the learner generated a policy based on the subject and resource age which might be valid based on the local insights learned from the example dataset. If statistical information were provided, the learner would be able to generate a more general version of Policy 1 more suitable for transfer to other contexts.

Another method to assure safe generalization is to augment the example dataset with *pre-defined restrictions*. The restrictions provide additional conditions that are critical to protect against security threats and vulnerabilities. The pre-defined restrictions may fall into two categories: *domain-based* and *target-based*. Including domain-based constraints implies that the learning process should consider the domain differences between the system of the example dataset and the target system. For example, learning security policies in a small system that has few roles cannot be generalized straightforwardly to a large system having a complex role hierarchy. On the other hand, including the target-based constraints implies that the generated policies should explicitly specify a deterministic target (i.e., subject or resource). In the context of access control policies, explicitly specifying targets enhances security.

Example: Policy 2 in Fig. 3b shows a case in which the learner generated a general policy in which the subject is not well-specified. If a target-based restriction were provided, the learner would be able to generate a policy explicitly specifying the resource and the specific attributes of the subject.

To avoid learning using a noisy example dataset, a straightforward method is to filter the dataset in advance. However, appropriately *filtering the example dataset* requires formal definitions for the "low quality" examples. "Low quality" examples include inconsistent responses to similar requests and requests associated with irrelevant responses which do not reflect appropriate decisions of a policies (i.e., 'not applicable' decision for XACML policies). Formal definitions of "low quality" examples can be adapted from the definitions of "low quality" policies by Bertino et al. [14], [31]. These formal definitions enable analyzing policies [31], [32]. Similarly, adapted definitions for "low quality" examples can enable analyzing and refining the example dataset to obtain a noise-free one.

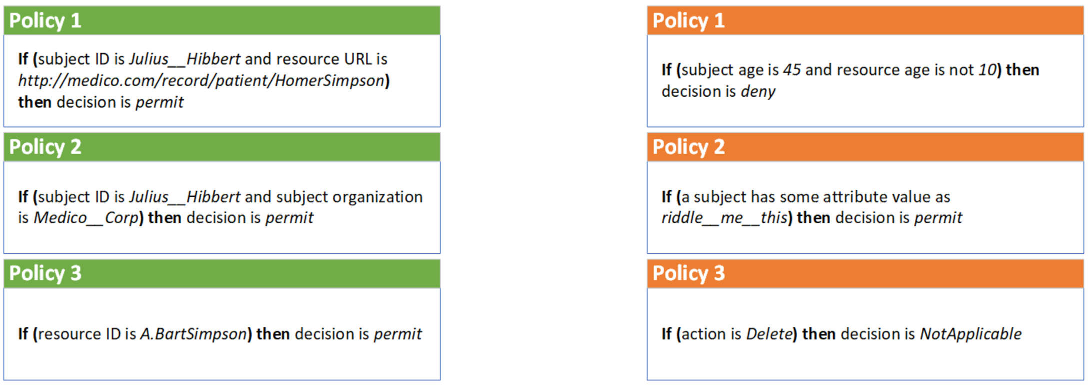
Example: Policy 3 in Fig. 3b shows a case in which the learner has misinterpreted an irrelevant response as a proper decision. In XACML, the decision of a specified policy is either "Deny" or "Permit"; hence "NotApplicable" is not a proper decision. If the example dataset were refined by pruning such irrelevant examples, the learner would be able generate a policy with a proper decision.

D. Data Sharing in Coalitions

In coalition operations, a wide variety of data is collected by each of the coalition members. This data can be collected using their Intelligence, Surveillance and Reconnaissance (ISR) assets deployed in the field which include videos and images from cameras, audio and conversations from microphones, seismic and vibration data, radio spectrum data, infrared imagery etc; information collected using satellites or unmanned aerial vehicles, or it may be a collection of documents they may have found when conducting operations. In many cases, coalition members are willing to share that data with each others so that the insights that are obtained from the data can be mined by partners.

Because the trust among partners is not absolute, there are restrictions both on what can be sent to the other partners, as well as what can be received from the partners. These are usually expressed as policies on data sharing. In some cases, these policies can be defined manually.

In many cases, e.g. when the data is used for an exercise like machine learning to create an AI model, the definition of manual policies becomes difficult. In these cases, policies need to be generated automatically using a system that can analyze the data provided by the partner; such an analysis may take into account many different factors. For example, data provided by different partners may have different levels of quality, the partner may not be completely trusted, and the value of the data provided may be different. Because of the large number of factors and types of policy, policies for data sharing will include Boolean combinations of conditions -such as for example testing whether the value of some data items is


Policy 1

If (subject ID is *Julius__Hibbert* and resource URL is *http://medico.com/record/patient/HomerSimpson***)** **then** decision is *permit*

Policy 2

If (subject ID is *Julius__Hibbert* and subject organization is *Medico__Corp***)** **then** decision is *permit*

Policy 3

If (resource ID is *A.BartSimpson***)** **then** decision is *permit*

(a) Correctly Learned Policies

Policy 1

If (subject age is *45* and resource age is not *10***)** **then** decision is *deny*

Policy 2

If (a subject has some attribute value as *riddle__me__this***)** **then** decision is *permit*

Policy 3

If (action is *Delete***)** **then** decision is *NotApplicable*

(b) Incorrectly Learned Policies

Fig. 3: Examples of Policies Learned by the ASG learner

above a certain threshold. Manually specifying such policies is not feasible and a generative policy framework is thus required. An initial approach has been developed by Verma et al. [33]. A notable feature of the approach is the use of "helper" microservices for generating values used to evaluate the policy conditions for specific data items and based on this evaluation decide whether to use the data. As one may have multiple services that can be used in different contexts, an interesting direction is to use the symbolic learner to learn which microservice to use for which context and data.

E. Federated Learning

In some coalition environments, sharing of raw data may not be possible. The limitation on the sharing of data may arise because of a lack of required bandwidth or reliable network connectivity to exchange data, or because of a lack of trust in the partner's ability to assure data security, or because of regulations or policy requirements that restrict data sharing. In these cases, coalition members may choose to share insights that they get from their sets of local data, and then share those insights instead. The insights can be shared and combined across all of the coalition partners.

This concept of transferring insights (or AI models) instead of raw data leads to the approach for federated learning [34]. Federated learning is useful in many different contexts, e.g. sharing insights among different Government agencies [35], distributed electronic health records [36], machine translation [37], and intelligence in wireless networks [38].

When insights or models are received from other parties, that are only partially trusted, the receiving party needs to make decisions regarding how to incorporate those insights together, e.g. by adapting those models, by combining those models, or by training a new model that is equivalent to those that are received. The policies for these are hard to generate manually, and generating them dynamically using ASG provides one mechanism to combine insights from many different sources together. Research is, however, needed to identify the types of policy that are required to govern federated learning, relevant attributes to be used in policies, and how to generate examples to be used for learning.

V. Research Directions

In this section we elaborate on additional open research directions to complement the discussion in the previous sections.

A. Policy Assessment

In many coalition settings, contexts and activities to be executed by coalition parties may rapidly change. It is difficult to anticipate beforehand whether a set of generated policies is suitable for a context and/or set of activities. It is thus critical to support the ability to dynamically evolve policies. However such evolution needs to be based on specific policy "quality" requirements (also referred to as policy metrics). Initial work for access control policies has identified four key requirements [14]:

- *Consistency* - it requires that the policy set does not include two or more policies that contradict each other. An example would be a policy that allows a subject to perform an action on an object and another policy that prohibits this subject to perform the same action on the same object.
- *Relevance* - it requires that the policy set does not contain policies that do not apply to any action or activity executed in the context of interest.
- *Minimality* - it requires making sure that the policy set does not include redundant policies as redundancy increases the policy maintenance costs and for some policy domains, such as security, it may introduce vulnerabilities.
- *Completeness* - it requires that for any action or activity that needs to be controlled/guided, there is at least one corresponding policy. The lack of policies may require involving human administrator which may be expensive and not possible in some contexts.

Previous research has focused on methods and tools for analyzing policies with respect to those four requirements (see [4] for a detailed survey of the state of the art). The use of our symbolic learning framework for generating policies helps in such an analysis. As our framework generates policies expressed as rules, it is possible to apply various reasoning methods to detect inconsistencies, redundancies and so forth as several of these methods are defined to work on logical representations. In addition, depending on the specific quality requirement, our framework may be able to directly generate policies that comply with the requirement, e.g. are free of conflicts.

Assessing policies in distributed and dynamic coalition settings entails, however, addressing a few challenges. The first is which requirements are critical and must thus be assessed for a given set of policies depend on several factors including the policy type, the domain, and the context. So case studies are needed to better understand which factors are the most critical and how to prioritize the requirements depending on these factors. The second is that assessment results may vary depending on the context with respect to which policies are assessed, especially when dealing with attribute-based policies (such as XACML policies [29]). For example, whether two policies conflict may depend on the context. Consider a policy specifying that "Any member of the Crypto project can modify the new crypto libraries" and another policy specifying that "A postdoc cannot modify the new crypto libraries". Whether those two policies conflict in a given context depends on whether there are subjects who are both members of the Crypto project and postdocs – which is context dependent. To address such problem one approach is to use a static analysis to identify potential conflicts and then at run-time use a conflict resolution algorithm to solve conflicts. However as several conflict resolution strategies are possible (see for example the XACML conflict resolution algorithms), one may need to decide which strategy to adopt depending on the context. Approaches like learning from human decisions about conflict resolutions can be adopted or one can specify additional policies that indicate which conflict resolution strategy to adopt based on the context. The third is related to the need of deploying logging tools able to capture requests, actions and activities by the managed parties to identify various situations, such as situations in which there was no policy covering certain actions, or situations in which the managed parties did not comply with the policies. In such situations collecting comprehensive information about the context may help in understanding the research for the lack of compliance.

Identifying additional requirements that are specific to coalition settings is also important. Two such requirements are: "enforceability" and risk. Enforceability requires that a policy can actually be enforced by a managed party in a certain context. For example, a policy may require contextual information be acquired in real time – which may be challenging in certain contexts – and it is crucial to provide indicators about the feasibility of the policy enforcement. The risk related requirement focuses on possible risks that may result from the application of a policy (or set of policies). For example, a restrictive access control policy may prevent the delivery of relevant information needed by a party, thus affecting the outcomes of activities, tasks, and actions. Supporting the assessment of such requirements requires the ability to use different enforceability and risk models for different contexts and coalition missions.

B. *Policy Explainability*

The ability of explaining decisions and recommendations taken by intelligent systems is today an important requirement for humans to trust these systems [39], [40]. Policy-based management systems are no exceptions, as these are systems that return decisions and recommendations and thus they may need to provide explations. In the context of our symbolic learning framework explainability is required at two different levels: policy learning, and policy enforcement. For the former, explainability is required to explain why certain policies are generated and why others are not. Explanations are crucial to indicate how generalization from examples and contexts has occurred and specifically which examples, example weights – if provided for the example dataset – and contextual information have resulted in the generation of certain policies and/or generalizations. For the latter, explanations are required when enforcing complex attribute-based access control policies, such as XACML, that often include multiple rules. When enforcing such policies, access requests include attributes on the subject, the object, and the context. However, depending on the specific conditions in policies, not all attributes may be relevant for the request. Therefore explanations may have to clarify which rules within a policy were the ones that were applied to the request. An important requirement for explainability approaches is that they must be easily understood by humans. An interesting approach to address such requirement is based on the notion of counterfactual explanation [41], that has been suggested in different contexts, such as for example to support the "right to explanation" in the General Data Protection Regulation (GDPR) of the EU. An example of a counterfactual explanation is the statement (slightly adapted from an example in [41]) "You were denied a loan because your annual income was $40,000. If your income had been $45,000, you would have been offered a loan". As counterfactual explanations are considered to be quite effective in communicating with human users, these approaches are being investigated in different domains (see [42] for examples). An interesting research direction is thus to explore such an approach, possibly combined with other approaches for AI systems [43] for policy explanations in our framework at both levels mentioned in the previous discussion.

C. *Integration of Symbolic Learning and Statistical Learning*

In spite of the tremendous success that statistical machine learning has had over the past 20 years, what these systems do is limited to the implementation of a specialized function-fitting algorithm [44] that does not provide any information on causality [45]. Take, for example, a purely statistically learned policy to decide the data quality provided by two partners in a 3 partner coalition. Assume one of the partners leaves the coalition, and the second one knows that the data they share could be partially verified by the partner that left. Now the incentive caused by the presence of the other partner to provide accurate data changes, and the learned function becomes useless without warning. Environment conditions are very dynamic in coalition systems, hence making pure statistical machine learning to learn policies hard to apply.

Symbolic learning, on the other hand, provides a door to causal information through rules. With this information tools can be built for an analyst to spot missing co-dependencies that were not captured during learning. Such tools could also provide information about potential effects when circumstances change. In other words, rules also open the door for explainability, as discussed in Subsection V.B. We have already mentioned counterfactuals. Counterfactuals allow one to see consequences in hypothetical situations, and this is possible only if causal relations are known: "Do I get a loan if my income were $50,000?." Learning causal rules, though, is not easy. There is usually a very large hypothesis space to search. Here is one place where statistical machine learning can complement, in a supporting role, symbolic learning. One can learn strategies to best search the hypothesis space. This has been proved successful in other areas where statistical machine learning is also hard to apply like planning (see, for example, [46]). Nevertheless, causal rules most be rigorously verified and tested by data analysis and certainty values should be associated with rules. At the end both approaches should co-exist, where statistical machine learned functions are used to detect "atomic" concepts such as edges and shapes from the pixels of an image, and a rule model of causation can be used to identify more complex concepts like the image shows a car parked inside the parking garage of a house.

VI. Conclusions

In this paper we have introduced an approach based on symbolic learning for the autonomic generation of policies for distributed dynamic coalitions, that may include robots, drones, self-driving vehicles, and IoT devices. We have developed several case studies that assess the use of generative policies and identify several challenges. As final remark we would like to emphasize that the development of a comprehensive management system based on symbolic learning requires techniques from different areas of computer science, including agent technologies, risk-based assessment techniques, statistical machine learning techniques, service-oriented architectures, edge computing.

Acknowledgment

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. Jorge Lobo was partially supported by the Spanish Ministry of Economy and Competitiveness under Grant Numbers: TIN201681032P, MDM20150502.

References

[1] S. Calo, D. Verma, E. Bertino, J. Ingham, and G. H. Cirincione, "How to prevent skynet from forming (a perspective from policy-based autonomic device management)," in *The IEEE 38th International Conference on Distributed Computing Systems (ICDCS)*, July 2018, pp. 1369–1376.

[2] E. Bertino, S. Calo, M. Touma, D. Verma, C. Williams, and B. Rivera, "A cognitive policy framework for next-generation distributed federated systems: Concepts and research directions," in *The IEEE 37th International Conference on Distributed Computing Systems (ICDCS)*, June 2017, pp. 1876–1886.

[3] E. Bertino, G. Ghinita, and K. Ashish, "Access control for databases: Concepts and systems," *Foundations and Trends in Databases*, vol. 3, no. 1-2, pp. 1–148, Feb. 2011.

[4] A. Abu Jabal, M. Davari, E. Bertino, C. Makaya, S. Calo, D. Verma, A. Russo, and C. Williams, "Methods and tools for policy analysis," *ACM Computing Survey, in print.*

[5] D. Verma, S. Calo, S. Chakraborty, E. Bertino, C. Williams, J. Tucker, and B. Rivera, "Generative policy model for autonomic management," in *IEEE SmartWorld, Ubiquitous Intelligence Computing, Advanced Trusted Computed, Scalable Computing Communications, Cloud Big Data Computing, Internet of People and Smart City Innovation*, 2017, pp. 1–6.

[6] Q. Ni, J. Lobo, S. Calo, R. Pankaj, and E. Bertino, "Automating role-based provisioning by learning from examples." in *The 13th ACM Symposium on Access Control Models and Technologies (SACMAT)*. ACM, pp. 75–84.

[7] S. Muggleton, "Inductive logic programming," *New Generation Computing*, vol. 8, no. 4, pp. 295–318, Feb. 1991.

[8] T. Eiter, G. Ianni, and T. Krennwallner, "Answer set programming: A primer," in *Reasoning Web International Summer School*. Springer, 2009, pp. 40–110.

[9] M. Law, A. Russo, and K. Broda, "The complexity and generality of learning answer set programs," *Artificial Intelligence*, vol. 259, no. 3, pp. 110–146, Jun. 2018.

[10] ——, "The ILASP system for learning answer set programs," https://www.ilasp.com, 2015.

[11] M. Sipser, *Introduction to the Theory of Computation*. PWS Publishing, 1997.

[12] M. Law, A. Russo, B. Elisa, B. Krysia, and L. Jorge, "Representing and learning grammars in answer set programming," in *The Thirty-Third AAAI Conference on Artificial Intelligence (AAAI-19)*, 2019.

[13] M. Law, A. Russo, and K. Broda, "Inductive learning of answer set programs," in *European Conference on Logics in Artificial Intelligence*. Springer, 2014, pp. 311–325.

[14] E. Bertino, A. Abu Jabal, S. Calo, D. Verma, and C. Williams, "The challenge of access control policies quality," *J. Data and Information Quality*, vol. 10, no. 2, pp. 6:1–6:6, 2018.

[15] V. C. Hu, E. Martin, J. Hwang, and T. Xie, "Conformance checking of access control policies specified in xacml," in *Annual International Computer Software and Applications Conference (COMPSAC 2007)*, vol. 2, July 2007, pp. 275–280.

[16] E. Bertino, G. de Mel, A. Russo, S. Calo, and D. Verma, "Community-based self generation of policies and processes for assets: Concepts and research directions," in *The IEEE International Conference on Big Data (Big Data)*. IEEE, 2017, pp. 2961–2969.

[17] R. Bordini, M. Dastani, J. Dix, and A. Seghrouchni, "Multi-agent programming: Languages, platforms and applications," vol. 15, 01 2010.

[18] X. Xiao, A. Paradkar, S. Thummalapenta, and T. Xie, "Automated extraction of security policies from natural-language software documents," in *Proceedings of the ACM SIGSOFT 20th International Symposium on the Foundations of Software Engineering*, ser. FSE. New York, NY, USA: ACM, 2012, pp. 12:1–12:11.

[19] J. B. Michael, V. L. Ong, and N. C. Rowe, "Natural-language processing support for developing policy-governed software systems," in *Proceedings 39th International Conference and Exhibition on Technology of Object-Oriented Languages and Systems. TOOLS 39*, 2001, pp. 263–274.

[20] S. Jero, M. L. Pacheco, D. Goldwasser, and C. Nita-Rotaru, "Leveraging textual specifications for grammar-based fuzzing of network protocols. corr abs/1810.04755," 2018.

[21] J. A. Stankovic, "Research directions for the internet of things," *IEEE Internet of Things Journal*, vol. 1, no. 1, pp. 3–9, 2014.

[22] P. Kodeswaran, R. Kokku, M. Mallick, and S. Sen, "Demultiplexing activities of daily living in iot enabled smarthomes," in *INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications, IEEE.* IEEE, 2016, pp. 1–9.

[23] H.-M. Huang, "Autonomy levels for unmanned systems (alfus) framework: safety and application issues," in *Proceedings of the 2007 Workshop on Performance Metrics for Intelligent Systems*. ACM, 2007, pp. 48–53.

[24] "Taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," *Available at https://www.sae.org/standards/content/j3016_201806/*, SAE International, Standard, June 2018.

[25] D. Cunnington, I. Manotas, M. Law, G. de Mel, S. Calo, E. Bertino, and A. Russo, "An answer set grammar-based generative policy model for connected and autonomous vehicles," 2019, submitted for publication.

[26] G. White, S. Pierson, B. Rivera, M. Touma, P. Sullivan, and D. Braines, "Dais-ita scenario," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, vol. 11006. International Society for Optics and Photonics, 2019.

[27] N. Sadeh, J. Hong, L. Cranor, I. Fette, P. Kelley, M. Prabaker, and J. Rao, "Understanding and capturing people's privacy policies in a mobile social networking application," *Personal and Ubiquitous Computing*, vol. 13, no. 6, pp. 401–412, 2009.

[28] R. A. Maxion and R. W. Reeder, "Improving user-interface dependability through mitigation of human error," *International Journal of Human-Computer Studies*, vol. 63, no. 1-2, pp. 25–50, 2005.

[29] Oasis extensible access control markup language (xacml) tc. [Online]. Available: https://www.oasis-open.org/committees/tc_home.php?wg_abbrev=xacml

[30] R. Kohavi and D. Sommerfield, "Feature subset selection using the wrapper method: Overfitting and dynamic search space topology." in *KDD*, 1995, pp. 192–197.

[31] E. Bertino, A. Abu Jabal, S. Calo, C. Makaya, M. Touma, D. Verma, and C. Williams, "Provenance-based analytics services for access control policies," in *Proceedings of the 2017 IEEE World Congress on Services (SERVICES), Honolulu, HI, USA, June 25-30, 2017*. IEEE, 2017, pp. 94–101.

[32] A. Abu Jabal, M. Davari, E. Bertino, C. Makaya, S. Calo, D. Verma, and C. Williams, "Profact: A provenance-based analytics framework for access control policies," 2018, manuscript submitted for publication.

[33] D. Verma, S. Calo, S. Witherspoon, E. Bertino, A. A. Jabal, G. Cirincione, A. Swami, G. Pearson, G. D. Mel, and I. Manotas, "Self generating policies for training data curation in coalition environments," in *Policies for Autonomic Data Governance at ESORICS*, 2018.

[34] B. McMahan and D. Ramage, "Federated learning: Collaborative machine learning without centralized training data," *Google Research Blog*, 2017.

[35] D. Verma, S. Julier, and G. Cirincione, "Federated ai for building ai solutions across multiple agencies," in *AAAI FSS-18: Artificial Intelligence in Government and Public Sector, Arlington, VA, USA*, 2018.

[36] T. S. Brisimi, R. Chen, T. Mela, A. Olshevsky, I. C. Paschalidis, and W. Shi, "Federated learning of predictive models from federated electronic health records," *International journal of medical informatics*, vol. 112, pp. 59–67, 2018.

[37] M. A. Thomas, D. S. Abraham, and D. Liu, "Federated machine learning for translational research," 2018.

[38] J. Park, S. Samarakoon, M. Bennis, and M. Debbah, "Wireless network intelligence at the edge," *arXiv preprint arXiv:1812.02858*, 2018.

[39] M. T. Ribeiro, S. Singh, and C. G. Guestrin, "Why should i trust you?": Explaining the predictions of any classifier," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD-16)*, 2016.

[40] B. Kuipers, "How can we trust a robot?" *Commun. ACM*, vol. 61, no. 3, pp. 86–95, Feb. 2018.

[41] S. Wachter, B. Mittelstadt, and C. Russell, "Counterfactual explanations without opening the black box: Automated decisions and the gdpr," *Harward Journal of Law & Technology*, vol. 31, no. 2, Feb. 2018.

[42] A. Dhurandhar, P.-Y. Chen, R. Luss, C.-C. Tu, P.-S. Ting, K. Shanmugam, and P. Das, "Explanations based on the missing: Towards contrastive explanations with pertinent negatives," in *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018*, 2018, pp. 10 317–10 327.

[43] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, "A survey of methods for explaining black box models," *ACM Computing Survey*.

[44] A. Darwiche, "Human-level intelligence or animal-like abilities?" *arXiv preprint arXiv:1707.04327*, 2017.

[45] J. Pearl, "Theoretical impediments to machine learning with seven sparks from the causal revolution," *arXiv preprint arXiv:1801.04016*, 2018.

[46] S. Yoon, A. Fern, and R. Givan, "Learning control knowledge for forward search planning," *Journal of Machine Learning Research*, vol. 9, no. Apr, pp. 683–718, 2008.