A POLICY REASONER FOR POLICY-BASED DYNAMIC SPECTRUM ACCESS[1]

Behnam Bahrak, Amol Deshpande and Jung-Min "Jerry" Park
Department of Electrical and Computer Engineering
Virginia Polytechnic Institute and State University, Blacksburg, VA 24061
Email: {bahrak, amoldesh, jungmin}@vt.edu

ABSTRACT

Policy-based dynamic spectrum access is one of the spectrum access models being considered by regulators and researchers for regulating the behavior of cognitive radios. This approach to spectrum access decouples the policy-related components (i.e., policy management, provisioning, and reasoning) from the radio platform. In policy-based spectrum access, the policy reasoner plays a critical role—it assists in policy enforcement and carries out a number of tasks related to policy analysis and processing. One of the most crucial tasks performed by the policy reasoner is evaluating radio transmission requests in relation to a set of active policies. This paper describes the design process and architecture of a policy reasoner. Key features of the proposed policy reasoner include: (1) policy conflict detection and resolution; and (2) ability to process under-specified transmission requests and compute the corresponding constraints.

1. INTRODUCTION

In current radios, policies are programmed or hardwired into the radio as part of its firmware. This approach has a lot of drawbacks, for example any change in policies requires reimplementation and reaccreditation of radio's firmware. Further, it is not scalable or flexible enough to deal with policies that are written by different authorities.

Cognitive radios [1] need to consider a wide variety of operating dimensions such as frequencies, waveforms, and power levels. On the other hand they should be designed and implemented to work with the ever-changing nature of regulatory environments and application requirements. To make these requirements feasible, a flexible mechanism must be provided to support spectrum sharing in addition to regulatory policy issues. The DARPA's neXt Generation (XG) communication program [2] has proposed a policy-based solution that is able to adapt to changes in policies, applications and radio technology.

In cognitive radio technology, policies are used to describe the preferences and constraints on parameters. For instance what level of detected spectral density is considered as presence of primary users or in which frequency bands transmission is allowed? Each XG radio is equipped with a Policy Engine that is able to understand machine-readable policies and apply them to particular problem domain. The Policy Reasoner (PR) is the main inference component of a policy engine. The System Strategy Reasoner (SSR) part of a cognitive radio provides the PR with facts about the cognitive radio and its environment and the PR tells the radio whether or not it can transmit. Two policy reasoners have been proposed in [4, 5].

There are many benefits in using a policy-based approach to cognitive radios: certification effort delays are significantly reduced, policy engines and radios can be accredited separately, radio behavior can quickly adapt to a changing situation, policy changes can be done easily and so forth.

In this paper, we propose a new policy reasoner based on Multi Terminal Binary Decision Diagrams (MTBDDs), which detects conflicts and resolves them in addition to processing under-specified transmission requests and computing the corresponding constraints. Our policy reasoner uses new algorithms to compute appropriate constraints for denied or incomplete transmission requests in a more systematic and guided way (in comparison with other policy reasoners).

The rest of this paper is organized as follows: we provide technical background knowledge in Section 2. In Section 3, we describe the architecture of our proposed policy reasoner and the algorithms that we use for reasoning. Finally, Section 4 concludes the paper.

2. TECHNICAL BACKGROUND

In this section, we introduce some background knowledge for subsequent discussions.

[1] This research was supported in part by the National Science Foundation under grants CNS-0627436, CNS-0716208, and CNS-0746925.

2.1. XG Radio Architecture

The neXt Generation (XG) program is a technology development project sponsored by DARPA's Strategic Technology Office, with the goal of developing both the enabling technologies and system concepts to dynamically redistribute allocated spectrum [2]. An XG radio has four

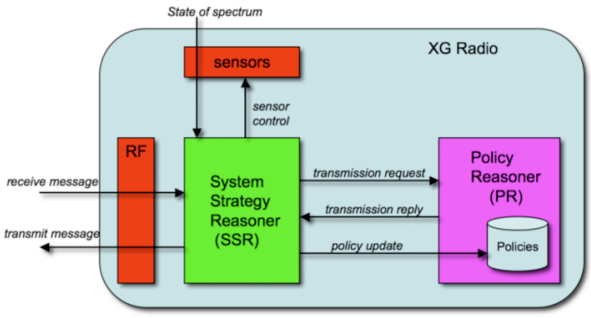

Figure 1. XG architecture [4].

main components [6]:

- Sensors: An XG radio needs sensors to sense its environment and discover the unused spectrum.
- Radio Frequency (RF): The RF part of an XG radio is used to transmit and receive various signals.
- System Strategy Reasoner (SSR): The SSR controls the radio's transmission. It builds optimal transmission request based on sensor data and its current strategies.
- Policy Reasoner (PR): The PR loads and stores policies. It receives the transmission requests that the SSR builds and evaluates them using the currently active set of policies, checking them for policy conformance.

These components are shown in Figure 1. Our focus in this paper is policy reasoner component. XG needs that radios be able to vacate channels within 500 ms, which results in the requirement that the PR must evaluate transmission request in less than 500 ms [2], so we have designed our policy reasoner to operate in a fast and efficient way. A more detailed description of XG radio architecture and implementation efforts can be found in [2-6].

2.2. Multi Terminal Binary Decision Diagrams

A Binary Decision Diagram (BDD) is a data structure that is used to represent a Boolean function [10]. A Boolean function can be represented as a rooted, directed, and acyclic graph, which consists of decision nodes and two terminal nodes called 0-terminal and 1-terminal that have zero out-degree. Each decision node u is labeled by a Boolean variable ($var(u)$) and has two child nodes called low child ($low(u)$) and high child ($high(u)$). An edge from a node to a low (high) child represents an assignment of the variable to 0 (1). In a figurative representation of a BDD, these are shown as dotted and solid lines, respectively. Such a BDD is called "ordered" if different variables appear in the same order on all paths from the root. A BDD is said to be "reduced" if the following two rules have been applied to its graph:

- Uniqueness: no two distinct nodes u and v have the same variable name and low- and high-successor, i.e., $low(u)=low(v)$, $high(u)=high(v)$, $var(u)=var(v)$ implies that $u=v$.
- Non-redundant tests: no variable node u has identical low- and high-successor, i.e. $high(u)\neq low(u)$

In popular usage, the term BDD almost always refers to Reduced Ordered Binary Decision Diagram (ROBDD). The advantage of an ROBDD is that it is unique for a particular functionality. The size of the BDD is determined both by the function being represented and the chosen ordering of the variables.

Multi terminal BDDs [8] (a.k.a algebraic decision diagrams) extend BDDs such that they can represent functions of an arbitrary range, while their domain is still a multidimensional Boolean space, i.e., an MTBDD provides a compact representation of functions of the form $P : B^n \rightarrow E$ which maps bit vectors over a set of variables B^n to a finite set of results (E). So the structure of an MTBDD is the same as BDDs with the difference that MTBDDs have more than two terminal nodes. In this paper, we use MTBDDs with three terminal nodes.

We also use the following explicit representation of MTBDDs. Nodes will be represented as numbers 0, 1, 2, … with 0, 1 and 2 reserved for terminal nodes 'N', 'Y' and 'NA'. The variables in the ordering $x_0 < x_1 < ... < x_n$ are represented by their indices 0, 1, 2, …, n. The MTBDD is stored in a table $T : u \rightarrow (i,l,h)$ which maps a node u to its three attributes $var(u)=i$, $Low(u)=l$ and $high(u)=h$.

2.3. XML Policy Algebra

In this paper, we use the XML policy algebra from [7]. Before we introduce the algebra, we need to find a suitable definition of cognitive radio policy semantics and transmission requests that the SSR sends to a cognitive radio. We assume the existence of a finite set A of names. Each attribute, characterizing an object in policies, has a name **a** in A. For cognitive radio policies, some of these attributes can be power level, frequency band, peak sensed power, location and time. All transmission requests are formally represented by definition 1.

DEFINITION 1. Let $a_1, a_2, ..., a_k$ be attribute names, and let v_i be a value for attribute a_i, then $r = \{(a_1, v_1), \ldots, (a_k, v_k)\}$ is a request.

DEFINITION 2. Let P be a 2-valued cognitive radio policy. We define the semantics of P as a 2-tuple $\langle R_Y^P, R_N^P \rangle$ where R_Y^P and R_N^P is the set of requests that are permitted (Y) and denied (N) by P respectively, and the requests that are neither in R_Y^P nor in R_N^P, are Not Applicable (NA).

The Fine-grained Integration Algebra (FIA) is given by $\langle \Sigma, P_Y, P_N, +, \&, \neg, \Pi_{dc} \rangle$ where Σ is a vocabulary of attribute names and their domains, P_Y and P_N are two policy constants, + and & are two binary operators, and $\neg$ and Π_{dc} are two unary operators.

We now describe the policy constants and operators in FIA. In what follows, $P_1 = \langle R_Y^{P_1}, R_N^{P_1} \rangle$ and $P_2 = \langle R_Y^{P_2}, R_N^{P_2} \rangle$ denote two policies to be combined, and $P_I = \langle R_Y^{P_I}, R_N^{P_I} \rangle$ denotes the policy obtained from the combination. Operators on policies are described as set operations.

- Permit Policy (P_Y) is a policy constant that permits everything.
- Deny Policy (P_N) is a policy constant that denies everything.
- Addition (+): Addition of policies P_1 and P_2 results in a combined policy P_I in which requests that are permitted by either P_1 or P_2 are permitted, requests that are denied by one policy and is not permitted by the other are denied.
- Intersection (&): Given two policies P_1 and P_2, the intersection operator returns a policy P_I in which requests that are not permitted by neither P_1 nor P_2 are denied, requests that are permitted by one policy and is not denied by the other are permitted.
- Negation (¬): Given a policy P, ¬P returns a policy, which permits (denies) all requests denied (permitted) by P. The negation operator does not affect those requests that are not applicable to the policy.
- Domain projection (Π_{dc}): The domain projection operator takes a parameter, the domain constraint *dc*, and restricts the policy only to the set of requests identified by *dc*.

DEFINITION 3. Domain Constraint (*dc*) takes the form $\{(a_1, range_1), (a_2, range_2), \ldots, (a_k, range_k)\}$, where a_1, a_2, ..., a_k are attribute names, and $range_i$ ($1 \le i \le k$) are sets of values from the vocabulary Σ. Given a request r = $\{(a_{r_1}, v_{r_1}), \ldots, (a_{r_m}, v_{r_m})\}$, We say r satisfies dc if the following condition holds: for each $(a_{r_j}, v_{r_j}) \in$ r ($1 \le j \le m$), if there exists $(a_{r_j}, range_i)$ in *dc*, then v_{r_j} is in $range_i$.

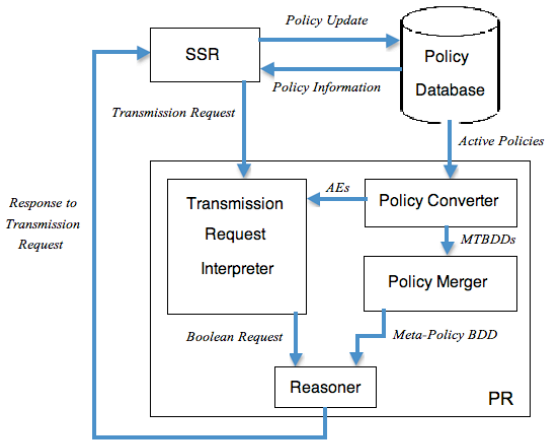

Figure 2. Architecture of the proposed policy reasoner

DEFINITION 4. Given two policies P_1 and P_2, the subtraction operator returns a policy P_I that is obtained by starting from P_1 and limiting the requests that the integrated policy applies only to those that P_2 does not apply to. The subtraction operator is defined as:

$$P_1 - P_2 = (P_Y \ \& \ (\neg(\neg P_1 + P_2 + \neg P_2))) + (P_N \ \& \ (P_1 + P_2 + \neg P_2))$$

3. THE POLICY REASONER

There are several different algorithms for combining policies in FIA; we introduce two of them in here. For cognitive radio policies, the '*Deny-overrides*' combining algorithm has been chosen by BBN technologies [2]. This algorithm states that the combined result is *deny* if any policy evaluates to *deny*, regardless of the evaluation result of other policies. If no policy evaluates to deny and at least one policy evaluates to *permit*, the combined result is *permit*.

The proposed policy reasoner consists of four different components: policy converter, transmission request interpreter, policy merger and reasoner. Figure 2 illustrates the architecture of our policy reasoner. In what follows, we will describe these components and their operation.

3.1. Policy Converter

The Policy Converter converts the XML cognitive radio policies into MTBDDs. As mentioned in Section 2, we can define policy P as a function $P : R \rightarrow E$ from the domain of requests R onto the domain of effects E, where E={*Y, N, NA*}. A cognitive radio policy in XML format can be transformed into a Boolean expression. A Boolean expression is composed of atomic Boolean expressions (AE) combined using the logical operations $\wedge$ and $\vee$.

EXAMPLE 1. Consider the following cognitive radio

policy. P1: Allow transmission in [255, 328] or [2200, 2290] frequency bands, if the mode is "Day to Day" or "Special Event" and the power level is less than 115 dB. Deny transmission if the mode is "Day to Day", between 11 a.m. and 2 p.m. Policy P1 can be defined as a function:

$$P_1 = \begin{Bmatrix} N \ \ if \ \ (Mode = D2D) \wedge (11 < time < 14) \\ Y \ \ if \ \ ((255 < f < 328) \vee (2200 < f < 2290)) \wedge \ldots \end{Bmatrix}$$

We now encode each unique atomic Boolean expression AE_i in the policy into a Boolean variable x_i such that: $x_i = 0$ if AE_i is false; $x_i = 1$ if AE_i is true:

$x_0 : Mode = Special\ Event$

$x_1 : Mode = Day\ to\ Day$

$x_2 : Power < 115\,dB$

$x_3 : 11 < time < 14$

$x_4 : 255MHz < f < 328MHz$

$x_5 : 2200MHz < f < 2290MHz$

Using the above Boolean encoding, a policy P can be transformed into a function $P : B^n \rightarrow E$, over a vector of Boolean variables, $\vec{x} = (x_0, x_1, \ldots, x_n)$, onto the finite set of effects $E = \{Y, N, NA\}$, where n is the number of unique atomic Boolean expressions in policy P. A request r corresponds to an assignment of the Boolean vector $\vec{x} = (x_0, x_1, \ldots, x_n)$, which is derived by evaluating the atomic Boolean expressions with attribute values specified in the request. After Boolean encoding, the policy P1 is transformed into the following function:

$$P_1(\vec{x}) = \begin{cases} Y \ \ if \ \ (x_0 \vee x_1) \wedge x_2 \wedge (x_4 \vee x_5) \\ N \ \ if \ \ x_1 \wedge x_3 \end{cases}$$

PROPERTY 1. *In a Boolean representation of a cognitive radio policy, each literal appears only once, and all literals are positive, i.e. their complements do not appear in the Boolean expression.*

A straightforward Boolean representation of an arbitrary cognitive radio policy may not satisfy Property 1. However, the proposed policy reasoner was designed under the assumption that the Boolean representation of any policy satisfies it. Hence any representation of a policy needs to be translated into a Boolean representation satisfying Property 1 before the PR can begin processing the policies.

After transforming an XML policy into a Boolean function, the policy can be represented as an MTBDD [8]. The internal (or non-terminal) nodes represent Boolean variables and the terminal nodes represent values in a finite set. Thus when a policy is represented using an MTBDD, the non-terminal nodes correspond to the unique atomic Boolean expressions and the terminal nodes correspond to the effects. Each path in the MTBDD represents an assignment for the Boolean variables along the path, thus representing a request *r*.

The terminal node on a path represents the effect of the policy for the request represented by that path. Note that a different ordering on the variables may result in a different MTBDD representation, possibly of different size. In this paper, we use the variable ordering $x_0 < x_1 < \ldots < x_n$. The MTBDD of policy P1 is shown in Figure 3, where the dashed lines are 0-edges and solid lines are 1-edges.

The reasons for using MTBDDs for representing policies are: (i) MTBDDs have proven to be a simple and efficient representation for XML policies [7] and (ii) operators in FIA can be mapped to efficient operations on the underlying policy MTBDDs.

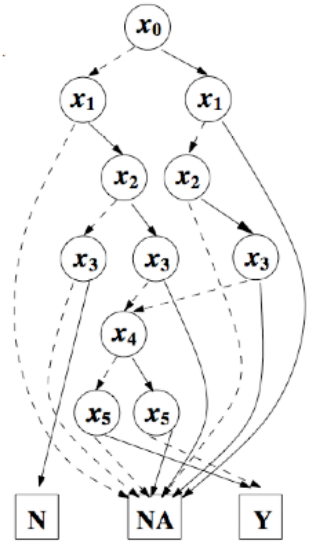

Figure 3. The MTBDD representation of Policy P1.

3.2. Transmission Request Interpreter

The PR assumes that the transmission request submitted by the SSR is in the format of Definition 1. For example, a request that conforms to policy P1 of Example 1 may take the form $\{(\text{frequency}, v_1), (\text{mode}, v_2), (\text{power}, v_3), (\text{time}, v_4)\}$. In order to respond to the transmission request, the Reasoner requires the transmission request to be compatible with the meta-policy MTBDD (the output of Policy Merger that is the combination of all active policy MTBDDs). The transmission request interpreter evaluates the transmission request and assigns a value to each of the Boolean variables of the meta-policy BDD such that the reasoner can use these values to respond to the transmission request. Each complete request corresponds to one path from the source node (the first node in the ordering) to one of the terminal nodes. The Reasoner will use this path to give an appropriate response to the transmission request.

EXAMPLE 2. Consider policy P1 and the following transmission request: R1 = {(mode, D2D), (power, 80),

(time, 6), (frequency, 300)}. The variables will have the following values:
$x_0 = 0, x_1 = 1, x_2 = 1, x_3 = 0, x_4 = 1, x_5 = 0$. The path for this request is shown in Figure 4 with blue lines. It can be seen that the response to this request is "Yes".

3.3. Policy Merger

Policy merger combines all the active policy MTBDDs that policy converter has made to build a single meta-policy MTBDD. All the binary and unary operations on policies

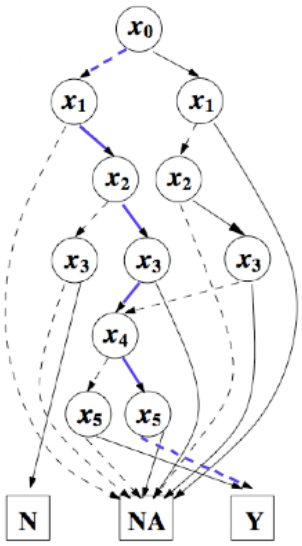

Figure 4. The path corresponding to a transmission request.

can be expressed as operations on the corresponding policy MTBDDs. There are many efficient operations for MTBDDs in the literature. In this paper, we use the **Apply** operation [8] to perform the FIA binary operations {+, -, &} and the **not** operation [7] for the FIA unary negation operation ¬. The **Apply** operation combines two MTBDDs by using a specified binary arithmetic operation. The Apply operation traverses each of the MTBDDs simultaneously starting from the root node. When the terminal nodes of both MTBDDs are reached, the operation is applied on the terminal nodes to obtain the terminal node for the resulting combined MTBDD.

After integrating cognitive radio policies into a single MTBDD, the policy reasoner only needs to follow the path that the transmission request specifies to determine a response to the transmission request. If this path ends in the 'Y' terminal node, the policy reasoner should permit the transmission, but if it ends in the 'N' or the 'NA' nodes, the policy reasoner should deny the transmission.

The policy merger in our policy reasoner uses the *deny-override* rule to resolve conflicts. A conflict occurs when the respective responses of two different policies to a single request are different. *Deny-override* rule denies a request if the request is denied by any of the currently active policies. For instance, if the path of a request ends with the N terminal node in one policy MTBDD, then the path of the same request in the meta-policy MTBDD will also end with the N terminal node (of the meta-policy MTBDD); note that this meta-policy MTBDD is the integration of all the MTBDDs of the currently active policies.

To combine cognitive radio policies, we need 'NA' terminal nodes because in cases when a policy permits a transmission request and a second policy neither prohibits nor permits that request, the final response to the request is a Yes, but when the second policy prohibits the transmission request the response is a 'No'; we need to separate these two cases. However, when we integrate all active policies into a single meta-policy, there is no difference between NA and N terminal nodes because the paths of all the transmission requests that end with either N or NA will be denied by the policy reasoner. Hence, we can combine the N and NA nodes when creating a representation of the meta-policy. This means that we can use a BDD instead of a MTBDD to represent a meta-policy. The use of a BDD as opposed to a MTBDD is advantageous because it enables us to decrease the size of the meta-policy representation (i.e., smaller number of nodes) and makes available a number of algorithms that work only on BDDs.

3.4. The Reasoner

To respond to a transmission request, the Reasoner uses the meta-policy BDD and the Boolean interpretation of the transmission request. The Reasoner sends one of the following responses to the SSR:

- **Yes:** The transmission is allowed. The SSR cannot transmit unless it receives such a message.
- **No:** The transmission is not allowed, but if the SSR applies a set of appropriate changes to its transmission request, the transmission will be allowed. These changes are called *opportunity constraints*.
- **Incomplete:** The transmission request is incomplete, but will be allowed if the SSR applies the appropriate opportunity constraints to its transmission request.

If the transmission request is incomplete, the transmission request interpreter will assign values to AEs that are specified in the transmission request and the reasoner simplifies the meta-policy BDD via the **RESTRICT** algorithm [10] and those values. After simplifying the meta-policy BDD, the Reasoner uses the FindPath function to find an opportunity constraint such that if the SSR satisfies that constraint in its transmission request, the policy reasoner will permit the transmission request. Algorithm 1 provides the pseudo code of the FindPath algorithm.

The **FindPath** algorithm is a direct result of the observation that in BDDs, if a node is not the N terminal node, it has at least one path leading to the Y terminal node.

Algorithm 1 FindPath Algorithm

```
1:  t = Source Node
2:  FindPath(t)
3:  Add node t to array Path
4:  if ( high(t) = 1 or low(t) = 1) then
5:      return True
6:  else
7:      if ( low(t) = 0 ) then
8:          return FindPath( high(t) )
9:      end if
10: else
11:     return FindPath( low(t) )
12: end if
```

When the transmission request is denied, the policy reasoner needs to compute an opportunity constraint that is returned to the SSR. The policy reasoner computes the opportunity constraint by finding a path in the meta-policy BDD that satisfies two requirements: (1) it terminates with the Y terminal node and (2) shares the maximum number of edges with the transmission request's path (X^{Tx}). The second requirement is needed because the PR needs to be cognizant of the fact that the transmission request produced by the SSR is already optimized for optimal radio performance. Therefore, the PR needs to make as few changes as possible to the transmission request's path when computing the opportunity constraint. The **FindBestPath** algorithm, shown as pseudo code in Algorithm 2, is used to compute the opportunity constraint.

Algorithm 2 FindBestPath Algorithm

```
    t = p[m-1]
2:  FindBestPath(t)
    Add node t to array Path
4:  if ( high(t) = 1 or low(t) = 1) then
        return True
6:  else if X^Tx_var(t) = 0 then
        if ( low(t) = 0 ) then
8:          return FindBestPath( high(t) )
        else
10:         return FindBestPath( low(t) )
        end if
12: else if ( high(t) = 0 ) then
            return FindBestPath( low(t) )
14: else
        return FindBestPath(high(t) )
16: end if
```

The **FindBestPath** algorithm cannot have more than n iterations, where n is the number of variables. So the complexity of this algorithm is $O(n)$.

4. CONCLUSION & FUTURE WORK

In this paper, we described the architecture of a new policy reasoner. Our device-independent policy reasoner processes policies represented as MTBDDs. It detects and resolves policy conflicts. The policy reasoner also has the ability to process under-specified or invalid transmission requests—i.e., it computes the appropriate opportunity constraints for such transmission requests. We have contrived two new algorithms for computing the opportunity constraints.

As part of our future work, we plan to extend the policy reasoner architecture to include additional modules and functionalities. One module we plan to add is a policy preprocessing module that removes invalid paths within the meta-policy BDD. Invalid paths within the meta-policy can lead to illogical opportunity constraints. We also plan to devise an algorithm for computing opportunity constraints that can process *weighted* transmission request parameters. The implementation of our policy reasoner is ongoing and we plan to carry out a quantitative analysis of the policy reasoner once the implementation is finished.

5. REFERENCES

[1] J. Mitola III, Cognitive radio: an integrated agent architecture for software defined radio, Ph.D Thesis, KTH Royal Institute of Technology, 2000.

[2] DARPA XG Working Group, "The XG Vision, Request for Comments," prepared by BBN Technologies, July 2003.

[3] F. Perich, R. Foster, P. Tenhula, and M. McHenry, "Experimental Field Test Results on Feasibility of Declarative Spectrum Management", 3rd IEEE International Symposium on New Frontiers in Dynamic Spectrum Access Networks (DySPAN 2008), Chicago, October 2008.

[4] D. Elenius, G. Denker, M.O. Stehr, R. Senanayake, C. Talcott, D. Wilkins, "CoRaL - Policy Language and Reasoning Techniques for Spectrum Policies", Policies for Distributed Systems and Networks, 13-15 June 2007, pp. 261-265

[5] F. Perich, M. McHenry, Policy-based spectrum access control for dynamic spectrum access network radios,Web Semantics: Science, Services and Agents on the World Wide Web, 2008.

[6] D. Wilkins, G. Denker, M.-O. Stehr, D. Elenius, R. Senanayake "Policy-based cognitive radios," IEEE Wireless Communications Magazine, August 2007.

[7] IP. Rao, D. Lin, E. Bertino, N. Li, and J. Lobo, "An Algebra for Fine-Grained Integration of XACML Policies" 14th ACM symposium on Access control models and technologies, 2009.

[8] M. Fujita, P. C. McGeer, and J. C.-Y. Yang. "Multi-terminal binary decision diagrams: An efficient datastructure for matrix representation." Formal Methods in System Design, 10(2-3):149-169, 1997.

[9] K. Fisler, S. Krishnamurthi, L. A. Meyerovich, and M. C. Tschantz. "Verification and change-impact analysis of access-control policies." In Proceedings of the 27th International Conference on Software Engineering (ICSE), pages 196-205, 2005.

[10] H. R. Andersen "An Introduction to Binary Decision Diagrams", Lecture Notes, 1999, IT University of Copenhagen.