A Formalization of HIPAA for a Medical Messaging System

Peifung E. Lam[1], John C. Mitchell[1], and Sharada Sundaram[1,2]

[1] Stanford University, Stanford, CA
[2] Tata Research Development and Design, Pune, India
{pflam, mitchell, shas}@cs.stanford.edu

Abstract. The complexity of regulations in healthcare, financial services, and other industries makes it difficult for enterprises to design and deploy effective compliance systems. We believe that in some applications, it may be practical to support compliance by using formalized portions of applicable laws to regulate business processes that use information systems. In order to explore this possibility, we use a stratified fragment of Prolog with limited use of negation to formalize a portion of the US Health Insurance Portability and Accountability Act (HIPAA). As part of our study, we also explore the deployment of our formalization in a prototype hospital Web portal messaging system.

1 Introduction

In regulated sectors such as healthcare, finance, and accounting, laws such as the U.S. Health Insurance Portability and Accountability Act (HIPAA), the Gramm-Leach-Bliley Act, the Sarbanes-Oxley Act, and related European laws have been enacted over the past decade to establish new or enhanced standards. The length of these laws, the opacity of the legal language, and the complexity of these acts make it very difficult for practitioners to determine whether they are in compliance [17]. This complexity becomes even more significant if computer programmers and information technology professionals wish to build and configure automated systems to help business professionals comply with applicable laws. The HIPAA regulation, in particular, appears complex for non-experts to follow for a number of reasons. To give one example, the law generally allows protected information to be shared between appropriate entities for the purpose of treatment. However, clause *164.508.a.2* [27] apparently contradicts this by stating that if the protected information is a *psychotherapy note* then a covered entity, i.e., a health plan, a health care provider or a clearinghouse, must obtain an *authorization* before disclosure. Thus simple reasoning based on actions allowed by one portion of the law, without accounting for prohibitions in other portions of the law, may give erroneous results.

Motivated by issues encountered in connection with the Myhealth@Vanderbilt system [28, 15] developed by the Vanderbilt University Medical Center, we decided to see if we could produce a formalization of applicable parts of the HIPAA

regulation in a form that could be used as part of the MyHealth system. Through Web access to a centralized system, MyHealth allows patients and medical professionals to exchange messages and potentially request and view information such as prescriptions or lab test results. We also envision future use of such systems to respond to requests from other hospitals and clinics, law enforcement, insurers, and other organizations. Our intent is to construct a compliance module that can decide, as messages are composed or entered into the system, whether a message complies with HIPAA. Further, we expect that HIPAA could be augmented by additional constraints, in accordance with Vanderbilt University Medical Center policy decisions.

Starting with a view of privacy policy, business processes, and compliance developed in previous work [5, 6], we chose to experiment with a stratified fragment of the logic programming language Prolog [18] with limited use of negation. In addition to representing HIPAA precisely enough to determine whether any particular action within the scope of the messaging system would comply with law, we also hope to produce a formalization that could be verifiable by lawyers, medical and computer professionals alike. For this reason, we have tried as much as possible to formalize the law so that the Prolog presentation can be read and understood section by section, with the meaning of the entire presentation determined in a systematic way from the meaning of its parts. In addition to supporting outside review and audit, this approach also helps make it possible to combine the HIPAA formalization with additional policies adopted by regulated enterprises. While there have been previous general studies on formulating laws in logical formalisms, e.g. [23, 9], our effort is distinguished by our focus on a specific privacy law, identification of a specific fragment of stratified Datalog that appears appropriate to the task, and our reliance on a general theory of privacy previously articulated for a more expressive but less commonly implemented logical framework [5, 6].

The main contribution of this paper is threefold: First we have identified a specific fragment of stratified Datalog with one alternation of negation, which we refer to for simplicity as pLogic, which suits our approach and supports a certain degree of policy compositionality. Secondly we use this framework to formalize the part of the HIPAA law that regulates information sharing in a healthcare provider environment. The structure of logic programming with predicates, query, and facts correspond to the legal clauses, actions being performed, and relations like roles defined in the law. Being a subset of logic programming, pLogic makes it easy to add cross references as present in the law. Finally we have implemented a prototype compliance checker and message system that is based on the Vanderbilt Medical Center MyHealth web portal [24]. This prototype is used to decide if a message that a practitioner is about to send is in compliance with the HIPAA regulation. We have also used our formalization to examine conflicts in the HIPAA regulation. While this paper focuses on the formalization of HIPAA, our approach appears to apply generally to a broad class of privacy regulations, such as those consistent with Nissenbaum's theory of Contextual Integrity [19] as formalized in [5].

The remainder of the paper is organized as follows. Section 2 introduces the key features and structure of the HIPAA policy and our information sharing model. Section 3 depicts the language pLogic that we use to model the HIPAA policy and our rule composition approach. Section 4 reviews related work. Finally Section 5 concludes.

2 Modeling HIPAA

2.1 HIPAA Overview

The U.S. Health Insurance Portability and Accountability Act (HIPAA) title II was enacted in 1996. As stated on an explanatory U.S. Government web site [26], HIPAA both explicitly permits certain transfers of personal health information, and prohibits some disclosures: "The Privacy Rule provides federal protections for personal health information held by covered entities and gives patients an array of rights with respect to that information." HIPAA Administrative Simplification, Regulation Text: 45 CFR Parts 160, 162, and 164 [27] regulate the use and disclosure of personal health information.

In HIPAA terminology, a *covered entity* is a health plan, a health care clearinghouse, or a health care provider who transmits health information in electronic form and *protected health information* is individually identifiable health information that is transmitted or maintained in electronic or other media.

The main focus of this paper is *section 164 of HIPAA,* which regulates the security and privacy issues in the health care industry. It covers general provisions, security standards for the protection of electronic health information, and privacy of individually identifiable health information. We are especially concerned with subpart 164.502, which covers the general rules for uses and disclosures of protected health information. Of the many subparts it refers to we consider subpart 164.506, which covers uses and disclosures to carry out treatment, payment, or health care operations, and subpart 164.508, which covers uses and disclosures requiring an authorization.

2.2 Actions

In our motivating application, patients or professionals enter a message into a centralized message system that can "deliver" the message by making it visible to other users. Messages may be simple questions from a patient, or may contain lab test results or other forms of protected medical information. Given information about the message, and other information such as the roles of the sender and receiver in the hospital, the HIPAA compliance module must decide whether delivery of the message complies with HIPAA. While portions of HIPAA regulate how data may be used after it is disclosed, or specify notifications that must be given after the disclosure, we currently focus only on whether to allow a message from a sender to a recipient.

Based on our understanding of HIPAA and the information available to the MyHealth system, our compliance engine is designed to make compliance decisions based on eight message characteristics: *To*, *From*, *About*, *Type*, *Purpose*, *In Reply To*, *Consented By* and *Belief*. The *To* and *From* fields indicate the recipient and sender of the message. The *About* field identifies whose personal health information is contained in the message. The *Type* field defines what kind of information would be passed, such as name or location. The *Purpose* field indicates a reason the message is being sent, such as for medical treatment. When the purpose is needed to determine compliance, we assume that some professional has asserted a purpose, or an asserted purpose is in some way inferred and made available as input to the compliance module. (Our prototype messaging system can infer when a purpose is needed, and supply the sender with a pulldown menu indicating purposes that would allow the message to be sent.) The *In Reply To* field was added to describe a disclosure where the message is sent as a response to some earlier message. The *Consented By* field indicates which people have consented to the message disclosure. The *belief* field contains a collection of assertions about the current situation, such as whether this is a medical emergency, or whether disclosure is (in the opinion of the sender) in the best interest of the health of the patient. Some beliefs may not be indisputable facts in the sense that another person may think differently. However, a sender may assert a belief (e.g., from a pulldown menu) or the sender's belief may be established by some other means. Once a message is allowed based on a belief, this reason may be recorded and later subject to audit. (We considered ways of processing audit logs based on policy in [6].)

Action (Definition 1) For the purpose of determining compliance, a message *action* is represented as an eight-tuple $a = \langle u_{src}, u_{dst}, u_{abt}, m_{typ}, m_{pur}, a_{reply}, c, b\rangle$, where (using underlining to indicate a set)

$$
\begin{array}{lll}
u_{src}, u_{dst}, u_{abt} & & \in U \quad \text{(the set of users or agents)},\\
m_{typ} & & \in T \quad \text{(the set of types of messages)},\\
m_{pur} & & \in P \quad \text{(the set of purposes)},\\
a_{reply} & & \in A \quad \text{(the set of actions)},\\
c = \langle \underline{u_{by}}, \underline{ct_{typ}} \rangle & & \in C \quad \text{(the tuple of consents) } \textit{with}\\
 & \underline{u_{by}} & \in U \quad \textit{(the set of users) and}\\
 & \underline{ct_{typ}} & \in CT \textit{ (the set of consent types)},\\
b = \langle \underline{u_{by}}, \underline{u_{abt}}, \underline{bf} \rangle & & \in B \quad \text{(the tuple of beliefs) } \textit{with}\\
 & \underline{u_{by}} \text{ , } \underline{u_{abt}} & \in U \quad \textit{(the set of users) and}\\
 & \underline{bf} & \in BF \textit{ (the set of beliefs).}
\end{array}
$$

HIPAA Policy (Definition 2). A HIPAA policy is a function from actions to Booleans (true or false), indicating permission or prohibition.

$$U \times U \times U \times T \times P \times A \times C \times B \rightarrow \{T, F\}$$

Categories A category is a set of field values definining the conditions when a legal clause is applicable to a particular action. For example, one common

category of actions are those with type indicating protected health information and purpose indicating medical treatment.

Subcategories Naturally, some field values may indicate that the action belongs to a subcategory of another category of actions. For example *psychotherapy note* is a subtype of *health records*, which implies that policy about *health records* could also affect decisions about *psychotherapy note*, but not vise versa. More generally, the possible values associated with any field may be partially ordered.

Roles While it is possible to express policy about specific individuals, HIPAA policies are written using roles. For example, an individual could be a nurse or a doctor. When an action is considered, our system receives the names of the sender and recipient, for example, and then uses information about the hospital to determine the respective role(s). For patients, similar processing (formalized in Prolog) is used to determine whether the patient is an adult or a minor.

3 Formalization of HIPAA

3.1 Overview

We introduce further concepts for the formalization of HIPAA, using *164.508.a.2* of HIPAA as a running example. As stated in the previous section, 164.508 as a whole governs uses and disclosures of protected health information that require an authorization. Specifically, *164.508.a.2* states, among other things, that a *covered entity* must obtain an authorization for any use or disclosure of *psychotherapy note*, except if it is to be used by the originator of the *psychotherapy note* for treatment.

Requirement An action that falls into the category of a legal clause is allowed only if the *requirement* in the clause is satisfied. For example, *164.508.a.2* states that the specified action is allowed only if an authorization is obtained.

Exception An exception in a legal clause qualifies its category. For example, *164.508.a.2* states that if the purpose of the action is for use by the originator of the *psychotherapy note* for treatment, then the requirement does not apply.

Clause vs. Rule For ease of exposition, we call a labeled paragraph in the HIPAA law a clause, and its translation into logic rules.

To illustrate our terminology, a clause with *category* given by predicate a, *requirement* predicate c and *exceptions* e can be expressed as the following rules:

$$\begin{aligned} permitted\_by_R &\Leftarrow (a \wedge \neg e) \wedge c \\ forbidden\_by_R &\Leftarrow (a \wedge \neg e) \wedge \neg c \\ R\_not\_applicable &\Leftarrow \neg a \vee e \end{aligned}$$

Combination A central concept in our approach is the way that a policy composed of several legal clauses is expressed by a combination of the associated *permitted_by* and $forbidden\_by$ rules. Given rules $R_1 \ldots R_m$, any action is consistent with the policy of these rules if it is permitted by some of the rules

and *not forbidden* by any of them.

$$compliant\_with_{R_1 \ldots R_m} \Leftarrow (permitted\_by_{R_1} \vee \ldots permitted\_by_{R_m}) \wedge \neg(forbidden\_by_{R_1} \vee \ldots forbidden\_by_{R_m})$$

This approach allows each clause to be translated into rules that are then combined in a systematic way to express the requirements of the law.

Cross-Reference Frequently a *requirement* of a clause involves a reference to other clauses of the law. In our formal definition below, we will require an acyclicity condition so that the cross-reference relation among HIPAA clauses forms a directed acyclic graph.

3.2 Expressing policy in pLogic

We have identified a fragment of stratified Datalog with one alternation of negation, which we refer to for simplicity as pLogic, which suits our formalization approach and supports a certain degree of policy compositionality. It is designed so that given an action we can verify whether the action is compliant with the written policy.

Our method for translating HIPAA into stratified Datalog with one alternation of negation is structured according to the form of *pLogic* rules and *pLogic* policies given below. As is standard in logic programming [18], a predicate is a symbol with an associated arity. Since we are using only Datalog, a term is a variable (starting with an upper-case letter) or an object constant (starting with a lower-case letter). An atom is an n-ary predicate applied to n terms. A literal is an atom. An expression is ground if it contains no variables.

Intuitively, a *pLogic* rule is a translation of a HIPAA clause into permitted and forbidden conditions. Each rule R therefore gives conditions on predicates $permitted\_by_R$ or $forbidden\_by_R$, taking actions as arguments, indicating whether the action should be allowed or denied. *pLogic* facts may be used to define subsidiary predicates or other inputs to the compliance process.

***pLogic* Facts** (Definition 3) A *pLogic* fact is an atom $g_i(a_1, \ldots, a_n)$ written using any relation g_i of arity n.

***pLogic* Rule** (Definition 4). The *pLogic* rules associated with a HIPAA clause R_i possibly cross-referencing clauses $R_j, \ldots, R_k$ have the form:

$$\begin{aligned} permitted\_by_{R_i}(A) \Leftarrow\ & category\_R_i(A) \wedge \neg exception\_R_i(A) \wedge requirement\_R_i(A) \\ & \wedge (permitted\_by_{R_j}(A)\ op_{i,j+1}\ \ldots\ op_{i,k}\ permitted\_by_{R_k}(A)) \\ forbidden\_by_{R_i}(A) \Leftarrow\ & category\_R_i(A) \wedge \neg exception\_R_i(A) \wedge (\neg requirement\_R_i(A) \\ & \vee forbidden\_by_{R_j}(A) \vee \ldots \vee forbidden\_by_{R_k}(A)) \end{aligned}$$

where

- $permitted\_by_{R_i}$, $forbidden\_by_{R_i}$, $category\_R_i$, $exception\_R_i$ and $requirement\_R_i$ are predicates on actions,

- each $op_{i,x}$ is either the $\wedge$ (AND) or the $\vee$ (OR) operator, as specified in the corresponding legal clause in HIPAA,
- $category\_R_i$, $exception\_R_i$ and $requirement\_R_i$ may appear as the head of additional Datalog rules we consider part of the rule expressing the clause,
- Every variable in the body must appear in the head,
- As indicated, $permitted\_by_{R_i}$ may depend on $permitted\_by_{R_j}$ for another clause R_j, but not $forbidden\_by_{R_j}$, and similarly $forbidden\_by_{R_i}$ may depend on another $forbidden\_by_{R_j}$ but not $permitted\_by_{R_j}$.

In the definition given above, the requirements are considered to be both *may* and *must*. However, the definition could easily be generalized to put one requirement in the permit rule and another in the forbid rule.

***pLogic* Policy** (Definition 5). An *pLogic* policy is a set Δ of *pLogic* rules and *pLogic* facts whose dependency graph (defined below) is acyclic.

The dependency graph $\langle V, E\rangle$ of Δ is defined as follows. The vertices V are predicates occurring in Δ and E contains a directed edge from u to v exactly when there is a rule in Δ where the predicate in the head is u and the predicate v appears in the body. The acyclicity condition ensures a nonrecursive stratified Datalog program.

Entailment for *pLogic* is based on the usual stratified semantics from deductive databases and logic programming.

pLogic policy is decidable *pLogic* policy is a nonrecursive logic program with negation and without function constants. Restricting the arity of the predicates to a constant reduces the complexity to polynomial time [18].

3.3 Rule Combination and Conflicts

pLogic is designed so that prohibition takes precedence over permission. However, we have found that some care must be taken in translating HIPAA into *pLogic* when it comes to overlapping clauses. We say that two rules *overlap* if the category and exceptions of the two rules allow them to apply to the same action, and one is a *subcase* of the other if its category and exception make it apply to every action satisfying the category and exceptions of the other. Two overlapping rules *conflict* if one permits an action while another forbids it; two rules are disjoint if there exists no action to which both apply.

Some example relationships between rules are illustrated in *Table 1*. All three rules presented in the table are pairwise overlapping. However, only rule $R_{502a1ii}$ has a category that is a subcategory of another, specifically rule $502a1v$.

Based on our experience with HIPAA, we believe that when two rules are disjoint or overlapping, but neither is a subcase of the other, the general approach described in sections 3.1 and 3.2 gives the correct results: an action is permitted if it is permitted by at least one rule and not forbidden by any. However, when one clause addresses a subcase of another, it often appears to be the expressed intent of the law to have the more specific clause take precedence over the other clause. In other words, it appears correct to disregard both the permitted and forbidden conditions of the less specific clause, and use only the more specific

Table 1. This table shows some examples of overlapping rules in HIPAA.

		Category			Requirement	
		A_{from}	A_{type}	$A_{purpose}$	$A_{consent}$	
$R_{502a1ii}$	+	*covered entity*	*health records*	*treatment*	*	$permitted\_by_{R_{506}}$(A)
	-	*covered entity*	*health records*	*treatment*	*	$forbidden\_by_{R_{506}}$(A)
R_{502a1v}	+	*covered entity*	*health records*	*	*	$permitted\_by_{R_{510}}$(A)
	-	*covered entity*	*health records*	*	*	$forbidden\_by_{R_{510}}$(A)
R_{508}	+	*	*psy-therapy note*	*	⟨ x, authz ⟩	
	-	*	*psy-therapy note*	*	¬⟨ x, authz ⟩	

clause. Fortunately, we can handle this correctly within pLogic, by using exceptions to narrow the scope of the less specific rule so that it is not applied in the conflicting subcase.

Generally, disregarding the added complexity of cross-references and exceptions, conflicts happen when the *category* of an action matches two or more rules, the *requirement* for one rule is satisfied and the requirement for the other is violated. In the above example, an action like ⟨ *from: covered entity*, *type: health records*, *for: treatment*, *requirement*:- as satisfying R_{506}⟩ is permitted by $R_{502a1ii}$ but forbidden by R_{502a1v}. Because pLogic is designed to give precedence to parts of the law that forbid an action, an action that is permitted by one of two overlapping rules and forbidden by the other will be considered forbidden.

In cases where one rule specifies a category that is a proper subset of the category of another rule, giving precedence to denial may be incorrect because the more specific clause of the law was intended to have higher priority. A simple way to modify the translation of the law into rules is to add exceptions to the more generic rule to make the two rules disjoint. In the example illustrated above (in the table), we add an exception to rule $R_{502a1ii}$ specifying *for:* ¬ *treatment*. This causes rule $R_{502a1ii}$ not to be applied when the purpose is treatment, eliminating the problematic conflict. Another solution suggested in [12] is to assign priorities and split all the overlapping rules to make them disjoint. If applied throughout, the alternative approach could produce a more efficient compliance checker, but we believe that it require substantial effort to properly split all rules, as many HIPAA rules are overlapping. Additionally, our approach has the advantage that it better preserves the correspondence between the logic rules and the corresponding legal clauses. To elucidate the structure of HIPAA and its translation in logic we have included an example in the appendix.

3.4 Extensions to the Model

Three extensions to the work described in this paper involve audit, implicit information, and obligations (as considered in [5, 6], for example).

As mentioned earlier in connection with beliefs asserted by the sender of a message, compliance decisions depend on the accuracy of the information provided by the users. It seems natural to generally assume that the users of a hospital medical system are professional practitioners who will provide correct

information. However, there may be some instances in which faulty or questionable information is entered. To provide accountability, auditing systems can be added to provide trace logs of how decisions are made and when beliefs or other potentially questionable input is used. Some related discussion involving non-compliant actions and audit appears in [6].

Another enhancement could infer relevant information, to reduce the amount of information the user has to provide to send a message. This could be achieved by extracting information from the message itself or by reasoning about the context of an action, information in previous messages, and so on.

Since obligations to perform future actions arise in many privacy contexts [5, 6], it may be useful to extend our *pLogic* approach to support such obligations. However, Prolog and Datalog do not inherently have any concept of past or future. While we represent the past explicitly through the in-reply-to field of messages, which produces linked structures of relevant past actions, future obligations requires an additional approach. One method may be to periodically run a scheduled process which looks through the log and checks whether, for any particular action, any further action is required and notify the concerned person. In fact, this corresponds to the current manual process at Vanderbilt, where a staff person checks every Friday afternoon to make sure that messages requiring a response are addressed.

4 Related Work

Access control mechanisms have been widely studied and deployed. Discretionary access control [10, 14], for example, allows the owner of a resource to specify access conditions, while mandatory access control [8] enforces policy of an organization. Role based access control [21] provides simplified policy entry and maintenance.

Regulations have been formalized using a variety of policy languages [3, 4, 2]. Privacy APIs [16], based on HRU access control, have been used to express some privacy legislation but are not tailored to compliance analysis or conflicts in policy. One approach to eliminating conflicts is shown in [12].

One widely available system for expressing and checking privacy policy is P3P [11]. In [1], the authors examine ways to use P3P to enforce policies in database systems where the type of information is known explicitly. P3P has limited expressiveness [25].

The EPAL language [22] privacy policy designed by IBM has been used to enforce privacy enterprise policies but it is not currently supported. XACML [20] is a prominent authorization algorithm, with some advantages and limiations [7].

The Logic of Privacy and Utility is based on the privacy language CI [5], a formalization of contextual integrity's transmission norms which has received some recent media attention [13]. The idea of contextual integrity was first proposed by Helen Nissenbaum [19].

5 Conclusions and Future Work

In this paper, we identified a fragment of stratified Datalog with limited use of negation, and developed a specific format for compostionally representing clauses of a law as Datalog rules, which we refer to for simplicity as pLogic. We used this framework to formalize the part of the US Health Insurance Portability and Accountability Act (HIPAA) that regulates information sharing in a healthcare provider environment. We tested this executable formalization of legal regulation by implementing a prototype web-based message system and compliance checker based on the Vanderbilt Medical Center MyHealth web portal [28]. This prototype is publicly accessible, allowing anyone to try the system and view sample Prolog HIPAA source [24].

We have also used our formalization to examine conflicts in the HIPAA regulation. By querying the logic program to return all the possible agents who could gain access to patient information, we found some anomalies regarding lack of regulation of government employees who are granted access to medical data, for example. While this paper focuses on the formalization of HIPAA, our approach appears to apply generally to a broad class of privacy regulations, such as those consistent with Nissenbaum's theory of Contextual Integrity [19] as formalized in [5].

A number of possible future directions seem promising. While we have not formalized all parts of HIPAA, it is possible to continue the effort to other portions of the law, if desired. We also look forward to collaborating with others, in hopes that there could be an open-source HIPPA formalization process. By sharing a formal presentation of HIPAA among many researchers and healthcare organizations, it may be possible to develop confidence in the formal presentation and use it widely across many enterprises. Another promising direction involves generating meaningful annotated audit logs, as by logging messages with semantic information about each action and compliance issues associated with it. In addition to automating compliance tasks, we also believe that a formal presentation of HIPAA (or other regulations) could also be useful in training medical personnel about the consequences and non-consequencs of the law.

References

1. Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. An XPath-based preference language for P3P. In *Proceedings of the Twelfth International Conference on World Wide Web*, pages 629–639. ACM Press, 2003.
2. Annie I. Antón, Julia Brande Earp, and Angela Reese. Analyzing website privacy requirements using a privacy goal taxonomy. In *Requirements Engineering 2002*, pages 23–31, 2002.
3. Annie I. Anton, Julia B. Eart, Matthew W. Vail, Neha Jain, Carrie M. Gheen, and Jack M. Frink. Hipaa's effect on web site privacy policies. *IEEE Security and Privacy*, 5(1):45–52, 2007.
4. Annie I. Antón, Qingfeng He, and David L. Baumer. Inside JetBlue's privacy policy violations. *IEEE Security and Privacy*, 2(6):12–18, 2004.

5. Adam Barth, Anupam Datta, John C. Mitchell, and Helen Nissenbaum. Privacy and contextual integrity: Framework and applications. In *IEEE Symposium on Security and Privacy*, pages 184–198. IEEE Computer Society, 2006.
6. Adam Barth, John Mitchell, Anupam Datta, and Sharada Sundaram. Privacy and utility in business processes. *Computer Security Foundations Symposium, IEEE*, pages 279–294, 2007.
7. Adam Barth and John C. Mitchell. Enterprise privacy promises and enforcement. In *Workshop on Issues in the Theory of Security*, pages 58–66. ACM Press, 2005.
8. David Elliott Bell and Leonard J. La Padula. Secure computer systems: Mathematical foundations. Technical Report 2547, MITRE Corporation, 1973.
9. Marc A. Borrelli. Prolog and the law: using expert systems to perform legal analysis in the United Kingdom. *Softw. Law J.*, 3(4):687–715, 1990.
10. Jason Crampton. On permissions, inheritance and role hierarchies. In *Proceedings of the 10th ACM Conference on Computer and Communication Security*, pages 85–92. ACM Press, 2003.
11. Lorrie Faith Cranor, Marc Langheinrich, Massimo Marchiori, Martin Presler-Marshall, and Joseph Reagle. The platform for privacy preferences 1.0 (P3P1.0) specification. `http://www.w3.org/TR/P3P/`, 2002.
12. Nora Cuppens-Boulahia, Frédéric Cuppens, Diala Abi Haidar, and Hervé Debar. Negotiation of prohibition: An approach based on policy rewriting. In *IFIP International Federation for Information Processing*, volume 278/2008, pages 173–187. Springer Boston, 2008.
13. Christine Evans-Pughe. The logic of privacy. *The Economist*, 382(8510):65–66, 2007.
14. Sushil Jajodia, Pierangela Samarati, Maria Luisa Sapino, and V. S. Subrahmanian. Flexible support for multiple access control policies. *ACM Trans. Database Syst.*, 26(2):214–260, 2001.
15. D. Masys. Electronic medical records and secure patient portals as an application domain for team research in ubiquitous secure technologies. `http://dbmi.mc.vanderbilt.edu/trust/TRUST_for_patient_portals.pdf`, 2005.
16. Michael J. May, Carl A. Gunter, and Insup Lee. Privacy APIs: Access control techniques to analyze and verify legal privacy policies. In *IEEE Workshop on Computer Security Foundations*, pages 85–97. IEEE Computer Society, 2006.
17. Roberta B. Ness. A year is a terrible thing to waste: early experience with HIPAA. *Annals of Epidemiology*, 15(2):85–86, 2005.
18. Ulf Nilsson and Jan Maluszynski. *Logic, Programming and Prolog (2nd ed.)*. Wiley, 1995.
19. Helen Nissenbaum. Privacy as contextual integrity. *Washington Law Review*, 79(1):119–158, 2004.
20. OASIS. eXtensible Access Control Markup Language (XACML) 2.0. `http://docs.oasis-open.org/xacml/2.0/access_control-xacml-2.0-core-spec-os.pdf`.
21. R. S. Sandhu, E.J. Coyne, H.L. Feinstein, and C.E. Youman. Role-based access control models. *IEEE Computer 29(2)*, pages 38–47, 1996.
22. Matthias Schunter, Paul Ashley, Satoshi Hada, Günter Karjoth, Calvin Powers, and Matthias Schunter. Enterprise privacy authorization language (EPAL 1.1). `http://www.zurich.ibm.com/security/enterprise-privacy/epal/Specification/`, 2003.
23. D. M. Sherman. A prolog model of the income tax act of Canada. In *ICAIL '87: Proceedings of the 1st international conference on Artificial intelligence and law*, pages 127–136, 1987.

24. Stanford Privacy Group. HIPAA Compliance Checker. `http://crypto.stanford.edu/privacy/HIPAA`.
25. William H. Stufflebeam, Annie I. Antón, Qingfeng He, and Neha Jain. Specifying privacy policies with P3P and EPAL: lessons learned. In *WPES '04: Proceedings of the 2004 ACM workshop on Privacy in the electronic society*, pages 35–35, New York, NY, USA, 2004. ACM.
26. U.S. Department of Health and Human Services. Understanding HIPAA privacy. `http://www.hhs.gov/ocr/privacy/hipaa/understanding/index.html`.
27. U.S. Department of Health and Human Services. HIPAA administrative simplification. `http://www.hhs.gov/ocr/privacy/hipaa/administrative/privacyrule/adminsimpregtext.pdf`, 2006.
28. Vanderbilt Medical Center. MyHealthAtVanderbilt. `https://www.myhealthatvanderbilt.com/`.

A An Example of the Encoding of the Law

We consider the part of clause $R_{164.502}$ which states that a *covered entity* can give out *health records* if it adheres to R_{502b} or R_{506a2} and satisfies additional conditions. We begin with 164.502.b.

```
164.502.b  Standard: Minimum necessary
 164.502.b.1Minimum necessary applies.
     When using or disclosing protected health information or when
     requesting protected health information from another covered
     entity, a covered entity must make reasonable efforts to limit
     protected health information to the minimum necessary to accomplish
     the intended purpose of the use, disclosure, or request.
 164.502.b.2 Minimum necessary does not apply.
  This requirement does not apply to:
   (i) Disclosures to or requests by a health care provider for treatment;
```

In short R_{502b} implies that when the *covered entity* is giving out the information to another *covered entity*, it should ensure that it is *minimal* information except for the purposes of treatment. Thus the *category* for this clause is *from: covered entity* and *to: covered entity* and*type: health records*. The requirement is *belief: minimal*. The exception is *for: treatment*.

```
164.502 Uses and disclosure of protected health information
 164.502.a Standard: A covered entity may not use or disclose protected
  health information, except as permitted or required by this
  subpart or by subpart C of part 160 of this subchapter.
  164.502.a.1 Permitted uses and disclosures. A covered entity
    is permitted to use or disclose protected health
    information as follows:
    (ii) For treatment, payment, or health care operations,
      as permitted by and in compliance with 164.506;
```

The clause R_{502aii} implies that when a *covered entity* is sending *health records* for the purposes of *treatment* then it should also comply with R_{506}. Here the *category* is *from:*

covered entity and *type: health records* and *purpose: treatment*. The *requirement* is to comply with R_{506}.

Thus the logic translation of the two clauses would look like:

Rules:-

$permitted\_by_{R_{502b}}(A) \Leftarrow$
$(A_{from} = \textit{covered entity}) \wedge (A_{type} = \textit{health records}) \wedge (A_{to} = \textit{covered entity}) \wedge$
$\neg(A_{purpose} = \textit{treatment}) \wedge (A_{belief} = \textit{minimal})$

$forbidden\_by_{R_{502b}}(A) \Leftarrow$
$(A_{from} = \textit{covered entity}) \wedge (A_{type} = \textit{health records}) \wedge (A_{to} = \textit{covered entity}) \wedge$
$\neg(A_{purpose} = \textit{treatment}) \wedge \neg(A_{belief} = \textit{minimal})$

$permitted\_by_{R_{502aii}}(A) \Leftarrow$
$(A_{from} = \textit{covered entity}) \wedge (A_{type} = \textit{health records}) \wedge (A_{purpose} = \textit{treatment}) \wedge$
$permitted\_by_{R_{506}}(A)$

$forbidden\_by_{R_{502aii}}(A) \Leftarrow$
$(A_{from} = \textit{covered entity}) \wedge (A_{type} = \textit{health records}) \wedge (A_{purpose} = \textit{treatment}) \wedge$
$forbidden\_by_{R_{506}}(A)$

Policy:-

$compliant\_with_{HIPAA} \Leftarrow$
$permitted\_by_{R_{502b}} \vee permitted\_by_{R_{502aii}} \wedge \neg(forbidden\_by_{R_{502b}} \vee forbidden\_by_{R_{502aii}})$

Attributes:-

We can define attributes and relations. Consider a relation called *inRole* that identify a particular individual and their role. It is simple to consider an example from the sitcom *Scrubs* where *dr_cox*, a doctor and *carla*, a nurse work for the *Sacred Heart Hospital*.

$inRole(carla, nurse)$
$inRole(dr\_cox, doctor)$
$inRole(doctor, \textit{covered entity})$
$inRole(nurse, \textit{covered entity})$
$inRole(sacredHeart, \textit{covered entity})$
$employeeOf(sacredHeart, dr\_cox)$
$employeeOf(sacredHeart, carla)$

We can also have a transitive closure of these rules which would imply that *carla* and *dr_cox* are *covered entity*.

Given this policy and the list of attributes, assuming *dr_cox* and *carla* work for the same hospital and R_{506} is satisfied, an action that would be allowed with this particular rule system is:

$$(from : carla, to : dr\_cox, type : \textit{health records}, for : \textit{treatment})$$

The policy would permit this action because of the rule R_{502aii}
An action like

$$(from : carla, to : xyz, type : \textit{health records}, for : \textit{treatment})$$

would not be allowed as there is no relation stating that *xyz* is some kind of *covered entity* and there is no other rule in the policy permitting this action.