An Abstraction for the Analysis of Secure Policy Interoperability

Javier Baliosian

November 4, 2016

Seminario de Probabilidad y Estadística 2016, CMAT

Instituto de Computación,
Facultad de Ingeniería, UdelaR,
Montevideo, Uruguay.
Email: baliosian@fing.edu.uy

UNIVERSIDAD
DE LA REPÚBLICA
URUGUAY

Policy-based Management Basics

Policy-based (Management) Systems

Policies are sets of rules governing the behaviour of a system. They are often used as a means of implementing flexible and adaptive systems for the management of Internet services, distributed systems, and security systems.

- Permissions

  `permission(p1,subject,action,object,context)`

- Prohibitions

  `prohibition(p2,subject,action,object,context)`

- Obligations

  `obligation(p1,subject,action,object,activation_context,violation_context)`

Policy-based Admission Control (RFC 2753)

“

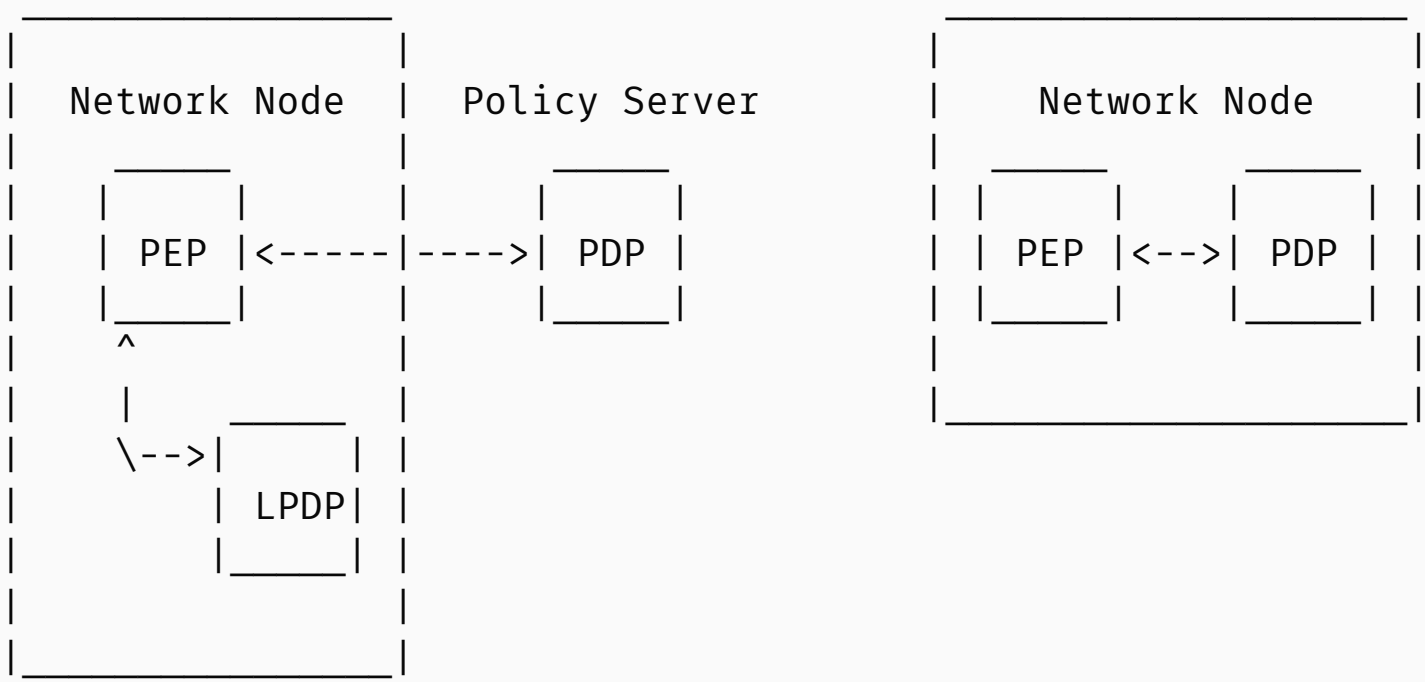


Figure 2: Two other possible configurations of policy control architecture components. The configuration on the left shows a local decision point at a network node and the configuration on the right shows PEP and PDP co-located at the same node.

”

Policy Conflicts

Rule 1: All users are forbidden to reboot virtual servers.

Rule 2: System administrators are authorised to manage any virtual device.

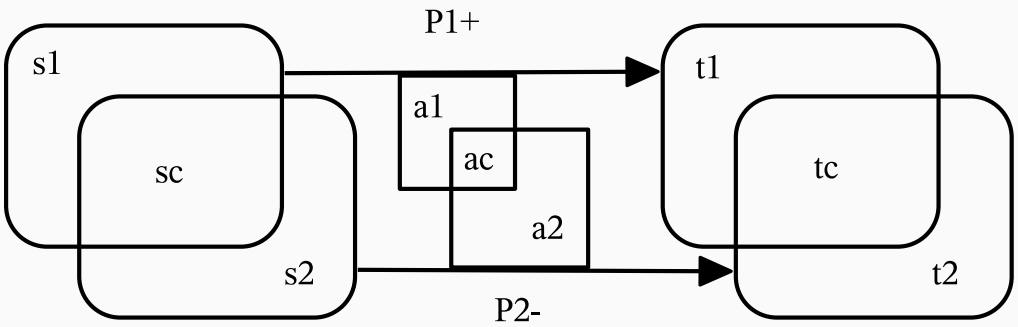


Figure 1: Overlapping Subjects, Objects and Actions (source [5]).

Policy Conflicts II

*"**Policy conflict:** Occurs when the actions of two rules (that are both satisfied simultaneously) contradict each other. The entity implementing the policy would not be able to determine which action to perform. The implementers of policy systems must provide conflict detection and avoidance or resolution mechanisms to prevent this situation. 'Policy conflict' is contrasted with 'policy error'."*

[RFC 3198]

How Policy Based Systems Manage Conflicts?

Negative policies always have priority. A forbidden action will never be permitted.

Assigning explicit priorities. An administrator assigns explicit priority-values to policies in order to define a precedence ordering.

Distance between a policy and the managed objects. Priority is given to the policy applying to the closest class in the inheritance hierarchy when evaluating access to an object referenced in a query.

FSTs for Policy Modelling

Rule 3: Every time a client asks for a virtualized web-server, he or she has to provide a domain name registration.

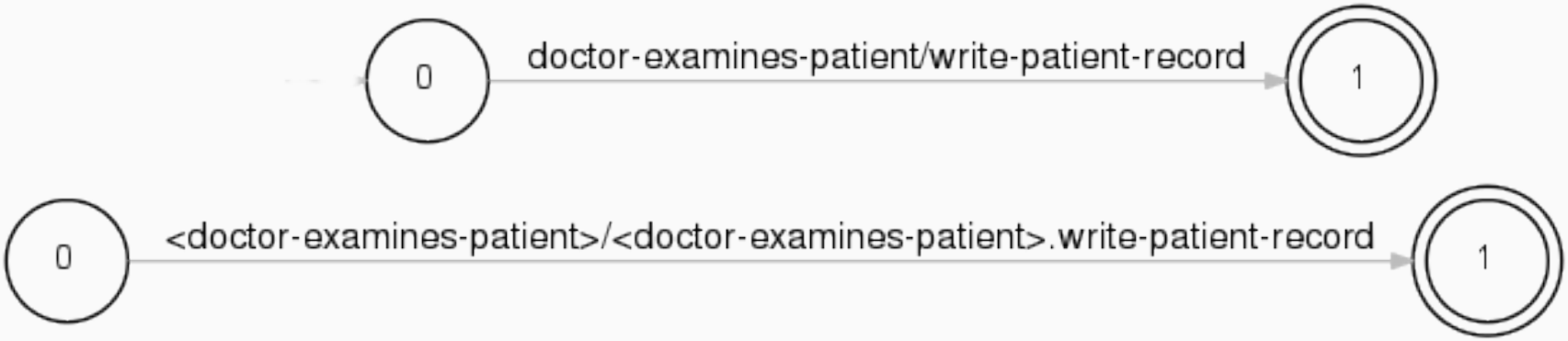


Figure 2: Two possible correspondences between Rule 3's constituents and FST elements.

Rule 4: When a health professional examines a patient, she/he must write notes in the patient's medical record.

Rule 5: When a mental health professional examines a patient, notes must be recorded in a place other than the patient's medical record.

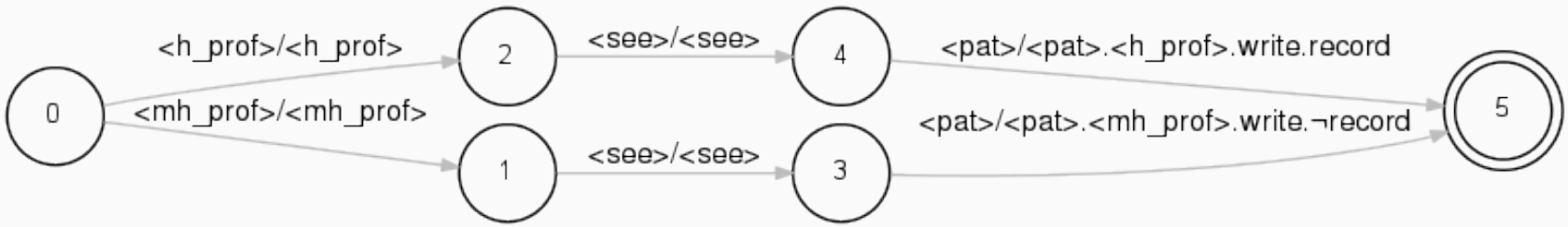


Figure 3: The Union of FSTs modelling rules 6 and 7.

A Little Background on Finite State Transducers

Semirings I

A semiring $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ is a ring that may have no additive inverse. Thus for any a, b, c $\in \mathbb{K}$:

$$a \oplus b = b \oplus a$$

$$a \oplus (b \oplus c) = (a \oplus b) \oplus c$$

$$a \oplus 0 = a$$

$$a \otimes (b \otimes c) = (a \otimes b) \otimes c$$

$$a \otimes 0 = 0 \otimes a = 0$$

$$a \otimes 1 = 1 \otimes a = a$$

$$a \otimes (b \oplus c) = a \otimes b \oplus a \otimes c$$

$$(a \oplus b) \otimes c = a \otimes c \oplus b \otimes c$$

Semirings II

Closed semirings which are semirings with an additional operation called closure ($*$) which satisfies:

$$a^* = 1 \oplus a \otimes a^* = 1 \oplus a^* \otimes a$$

If we have an affine map $x \mapsto ax + b$ in some closed semiring, then $x = a^*b$ is a fixpoint, since $a^*b = (aa^* + 1)b = a(a^*b) + b$.

The regular languages form a closed semiring where $\otimes$ is concatenation, $\oplus$ is union, and * is the Kleene star. Here the infinite geometric series interpretation of * is the most natural: a^* is the union of a^n for all n.

Semirings III

Other semirings:

Semiring	Set	$\oplus$	$\otimes$	$\overline{0}$	$\overline{1}$
Boolean	$\{0, 1\}$	$\vee$	$\wedge$	0	1
Probability	$\mathbb{R}_+$	$+$	$\times$	0	1
Tropical	$\mathbb{R}_+ \cup \{-\infty, +\infty\}$	min	$+$	$+\infty$	0
String	$\sum^* \cup \{\infty\}$	$\wedge$	$\cdot$	∞	ϵ

For strings, $\wedge$ is longest common prefix.

Matrix Semirings

$(S, \oplus, \otimes, 0, 1)$: a semiring

$(\mathbb{M}_n, \oplus, \otimes, J, I)$: the semiring of n × n-matrices over S

$$(A \oplus B)(i,j) = A(i,j) \oplus B(i,j)$$

$$(A \otimes B)(i,j) = \bigoplus_{1 \le q \le n} A(i,q) \otimes B(q,j)$$

$$J(i,j) = 0$$

$$I(i,j) = \begin{cases} 1(if i = j) \\ 0(otherwise) \end{cases}$$

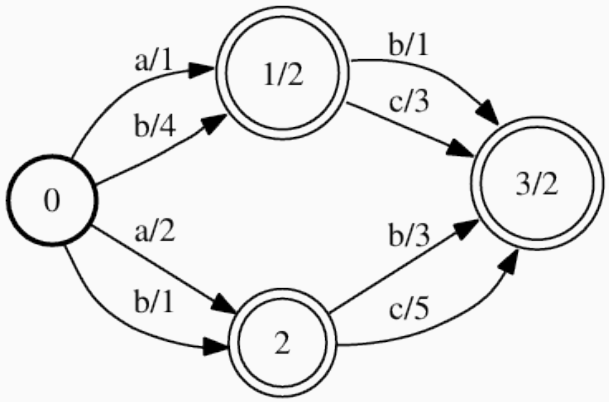


Figure 4: A FST with weights (source [6]).

Probability semiring $(\mathbb{R}_+, +, \times, 0, 1)$	Tropical semiring $(\mathbb{R}_+ \cup \{-\infty, +\infty\}, min, +, +\infty, 0)$
$[\![A]\!](ab) = 14$ $(1 \times 1 \times 2 + 2 \times 3 \times 2 = 14)$	$[\![A]\!](ab) = 4$ $(min(1 + 1 + 2, 3 + 2 + 2) = 4)$

Weighted Transducers

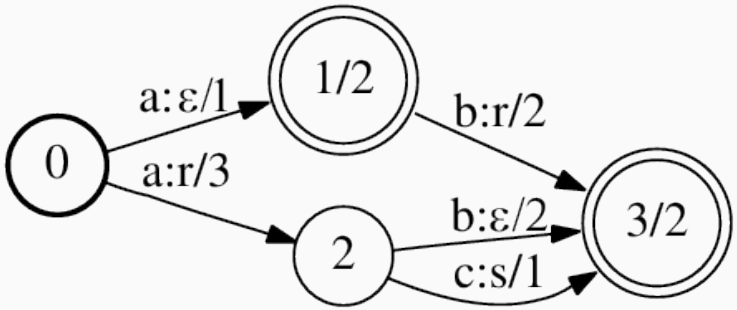


Figure 5: A FST with weights (source [6]).

Probability semiring $(\mathbb{R}_+, +, \times, 0, 1)$	Tropical semiring $(\mathbb{R}_+ \cup \{-\infty, +\infty\}, min, +, +\infty, 0)$
$[\![T]\!](ab, r) = 16$ $(1 \times 2 \times 2 + 3 \times 2 \times 2 = 16)$	$[\![T]\!](ab, r) = 5$ $(min(1 + 2 + 2, 3 + 2 + 2) = 5)$

Definitions and Notation

Alphabets: input Σ, output Δ. States: Q, initial states I, final states F. Transitions: $E \subseteq Q \times (\Sigma \cup \{Q\}) \times (\Delta \cup \{Q\}) \times \mathbb{K} \times Q$. Weight functions: initial weight function $\lambda : I \to \mathbb{K}$ final weight function $\rho : F \to \mathbb{K}$

Automaton $A = (\Sigma, Q, I, F, E, \lambda, \rho)$ with for all $x \in \Sigma^*$:

$$[\![A]\!](x) = \bigoplus_{\pi \in P(I,x,F)} \lambda(p[\pi]) \otimes \omega[\pi] \otimes \rho(n[\pi])$$

Transducer $T = (\Sigma, \Delta, Q, I, F, E, \lambda, \rho)$ with for all $x \in \Sigma^*, y \in \Delta$:

$$[\![T]\!](x, y) = \bigoplus_{\pi \in P(I,x,y,F)} \lambda(p[\pi]) \otimes \omega[\pi] \otimes \rho(n[\pi])$$

Transducers as Weighted Automata

A transducer T is deterministic (or *functional*) iff for each x there exists at most one y such that $[\![T]\!](x, y) = 0$

- An unweighted functional transducer can be seen as a weighted automata over the string semiring $(\sum^* \cup \{\infty\}, \wedge, \cdot, \infty, \epsilon)$.
- A weighted functional transducer over the semiring K can be seen as a weighted automata over the cartesian product of the *string semiring* and $\mathbb{K}$.

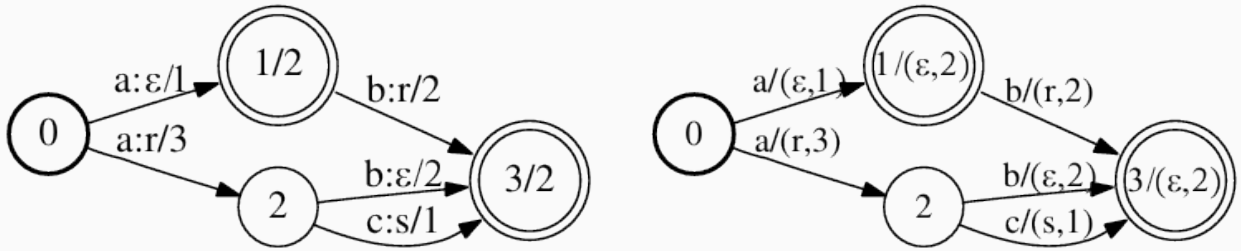


Figure 6: $[\![T]\!](ab, r) = 5, [\![A]\!](ab) = (r, 5)$ on a Tropical semiring $(\mathbb{R}_+ \cup \{-\infty, +\infty\}, min, +, +\infty, 0)$ (source [6]).

Operations: Sum (Union)

$$[\![T_1 \oplus T_2]\!](x, y) = [\![T_1]\!](x, y) \oplus [\![T_2]\!](x, y)$$

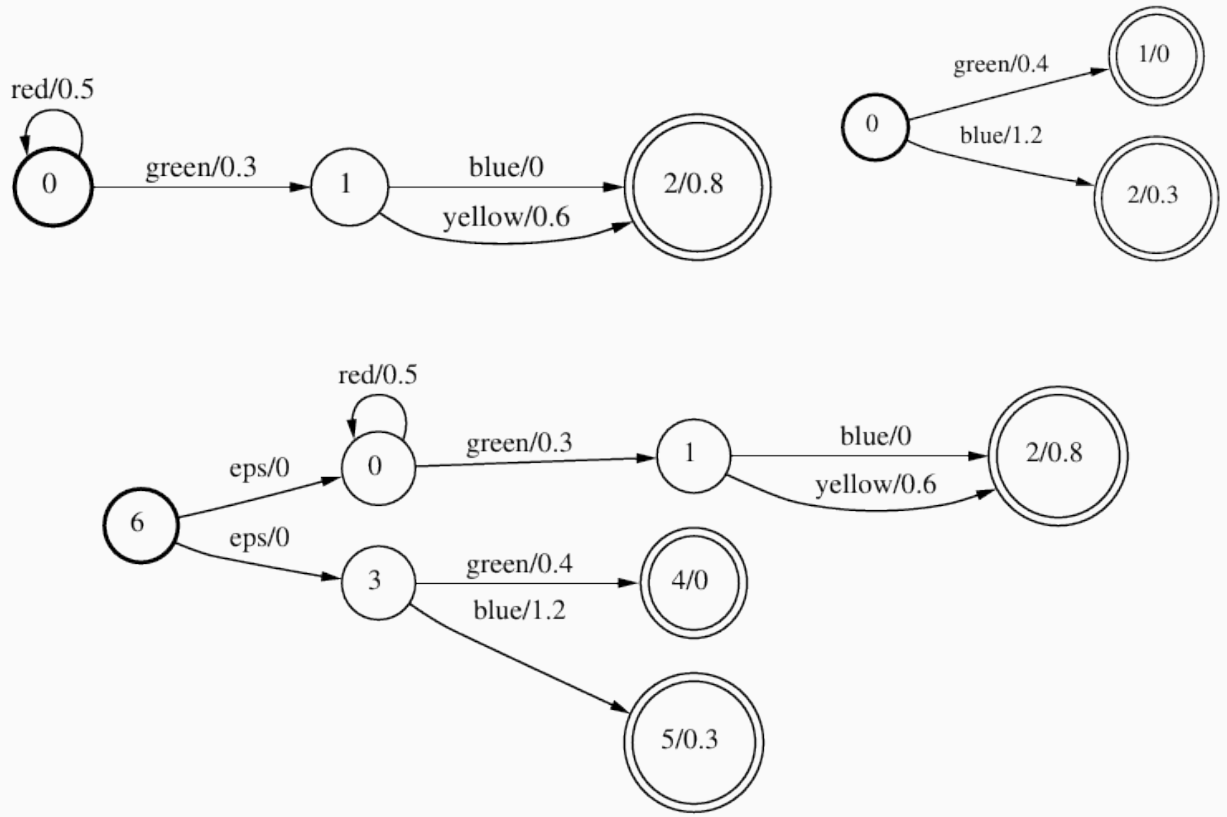


Figure 7: A Union of Weighted FSTs (source [6]).

Operations: Closure

$$[\![T]\!](x, y) = \bigoplus_{n=0}^{\infty} [\![T^n]\!](x, y)$$

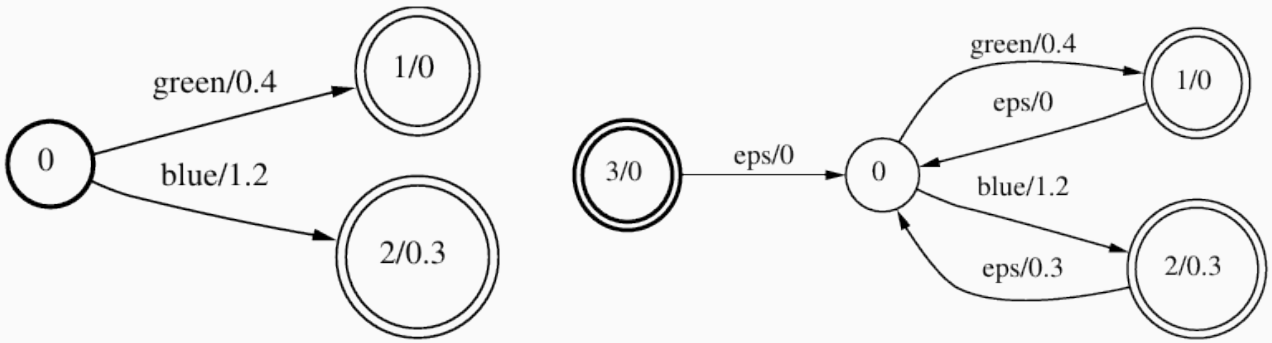


Figure 8: T and T^* (source [6]).

Operations: Composition

$$[\![T_1 \circ T_2]\!](x, y) = \bigoplus_z [\![T_1]\!](x, z) \otimes [\![T_2]\!](z, y)$$

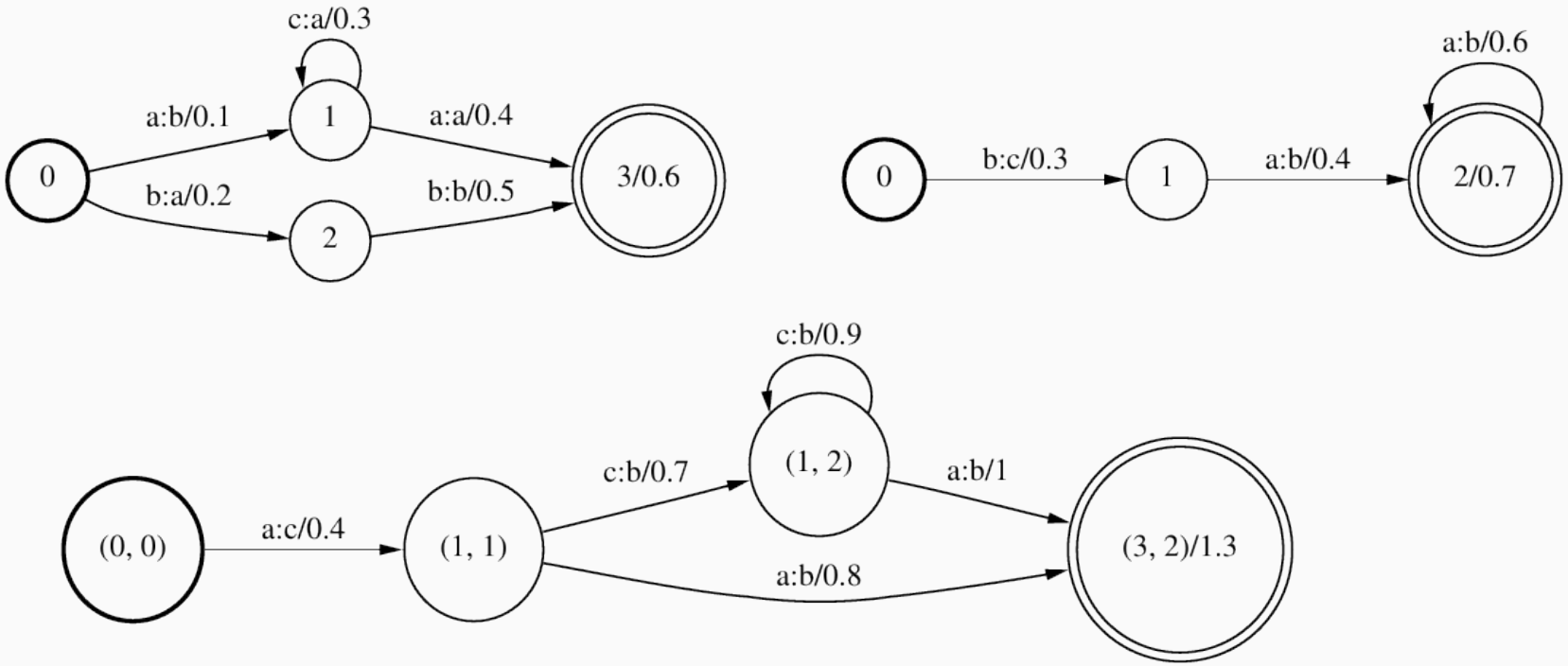


Figure 9: T_1, T_2, and $T_1 \circ T_2$ (source [6]).

Actually it is a matrix multiplication!.

Operations: (Acceptor's) Intersection

$$[\![A_1 \cap A_2]\!](x) = [\![A_1]\!](x) \otimes [\![A_2]\!](x)$$

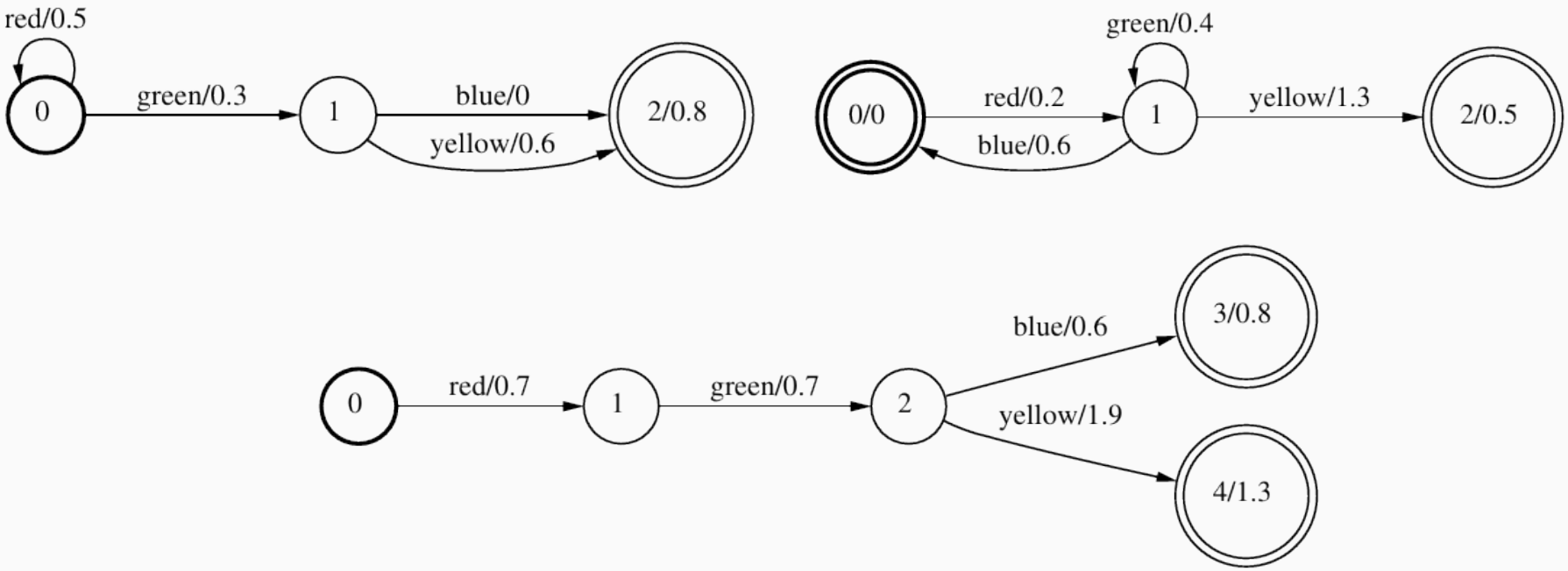
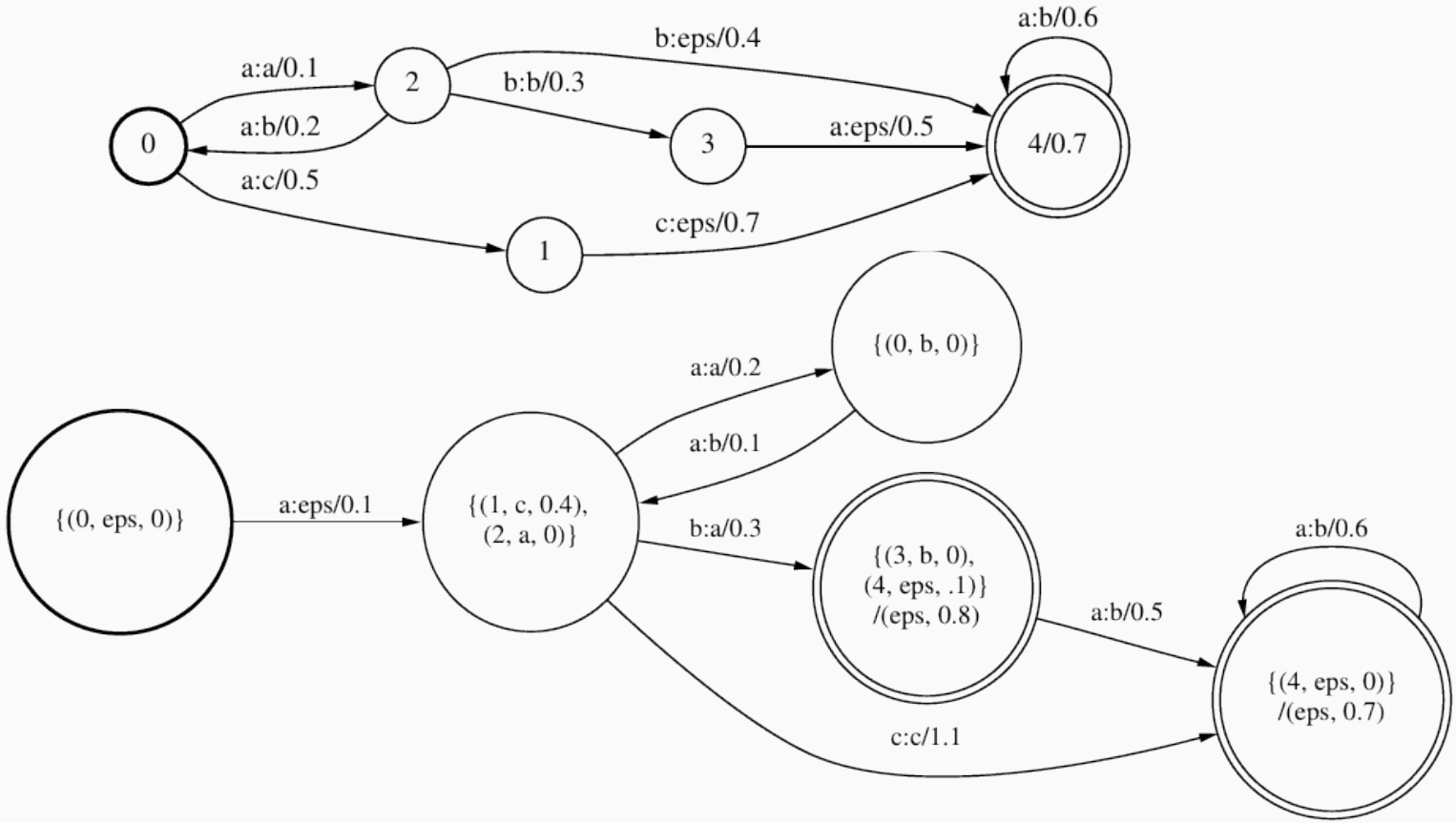


Figure 10: A_1, A_2, and $A_1 \cap A_2$ (source [6]).

Determinization of Weighted Transducers

An acceptor is deterministic iff for each state q there is at most one transition labeled with a given label. A transducer can be input deterministic (or subsequential) or output deterministic.



Single-Source Shortest-Distance Algorithms

If $M(i, j)$ is the adjacency matrix of a graph, then

$M^k(i, j)$ is the length of the shortest path with k edges from node i to node j, and

M^* is the sum (which in the tropical semiring means "minimum") of M^k for any k. Thus,

$M^*(i, j)$ is the length of the shortest path with any number of edges from node i to node j.

Writing Rules as FSTs

Finite State Transducers for Policies

A FST T over a semiring $(\mathbb{K}, \oplus, \otimes, \overline{0}, \overline{1})$ is a tuple $(Q, E, P, \Pi, S, F, \lambda, \rho)$ where:

Q is a finite set of states,
E is a set of symbols,
P is a set of predicates over E.
Π is a finite set of transitions
$Q \times (P \cup \{\epsilon\}) \times \mathbb{K} \times (P \cup \{\epsilon\} \times \mathbb{K}) \times Q \times \{-1, 0, 1\}$. [1] $S \subseteq Q$ is a set of start states, $F \subseteq Q$ is a set of final states, λ is an initial weight and ρ a final weight function. For all transitions $(p, d, u, r, w, q, 1)$ it must be the case that $d = r \neq \epsilon$.

$\widehat{\Pi} \subseteq Q \times E^* \times E^* \times Q$ is the relation defined by T.

[1] The final component of a transition is a sort of "identity flag" used to indicate when an incoming event must be replicated in the output.

Permissions

In the context of a system in which everything is prohibited by default, the OrBAC rule

`permission(p1,r,a,v,true)`

is modelled as the following FST:

$$T_{P+} = \left(id \left(R_P \right)^* \varepsilon \left(\overline{E^* R_P E^*} \right)^* \right)^*$$

where R_P is the FSR that consumes exactly the event string *rav*, $id(X)$ is the identity FST, $\varepsilon(X)$ is a transducer that consumes language X and produces ϵ always, * is the *Kleene* closure, and E^* is the language of all possible strings of events.

Permissions

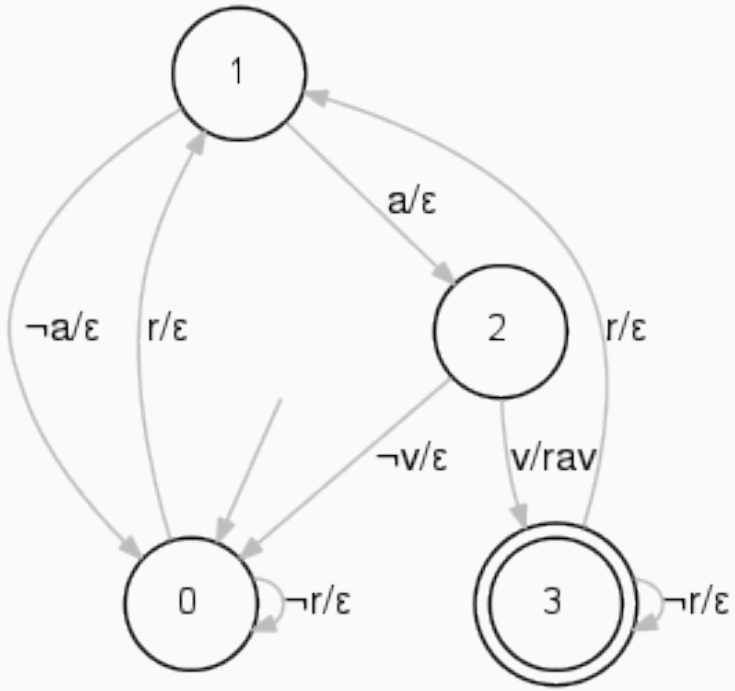


Figure 11: An FST modeling a permission (weights are not included because they have no role at this stage).

Prohibitions

In the context of a system that permits everything by default, the FST that models the rule

`prohibition(p1,r,a,v,TRUE)`

is the following:

$$T_{P-} = \left(\varepsilon \left(R_P\right)^* id \left(\overline{E^* R_P E^*}\right)^*\right)^*$$

Prohibitions (cont'd)

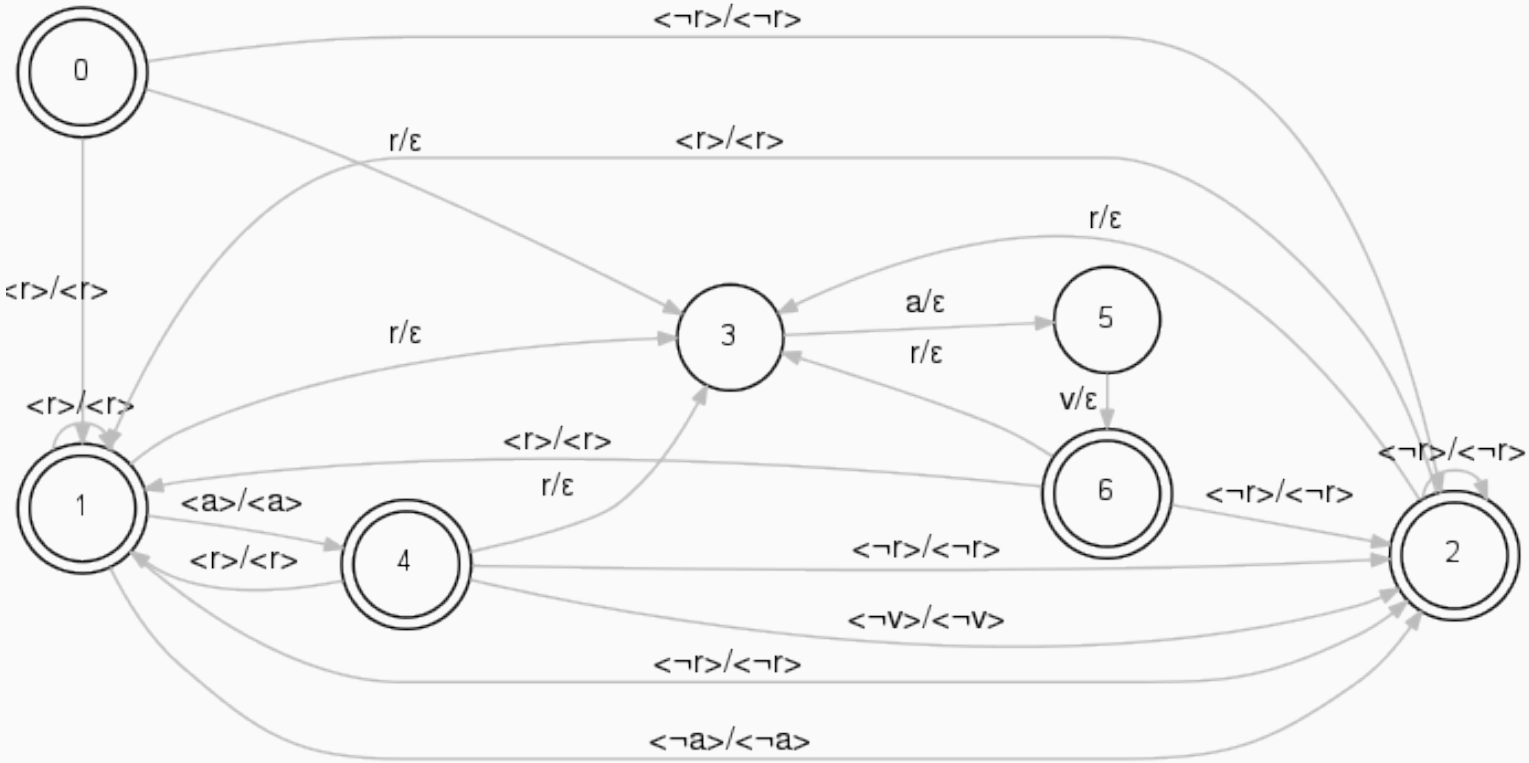


Figure 12: An FST Modeling a Prohibition.

Obligations

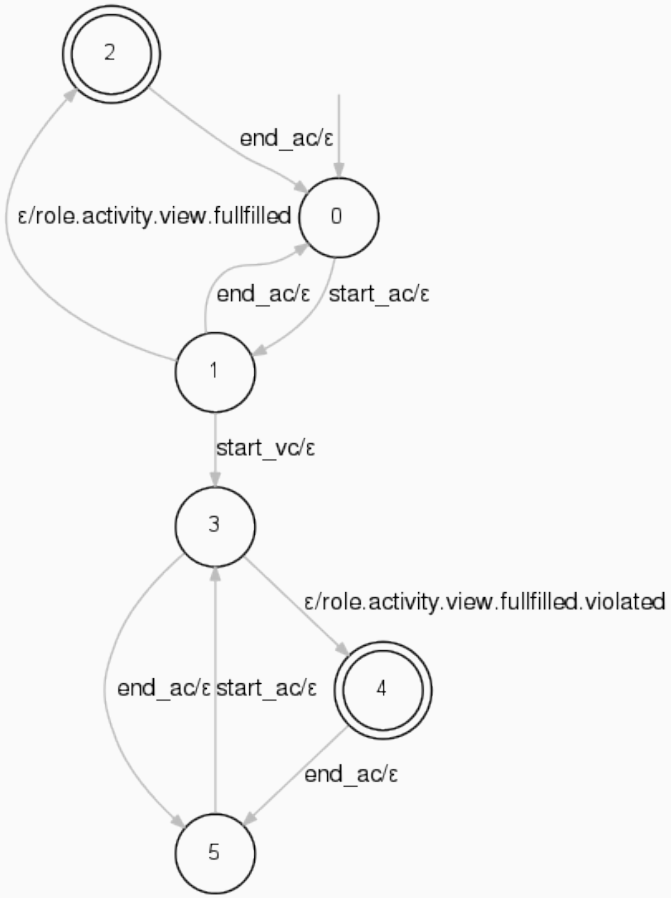


Figure 13: The obligation `obligation(p1, r, a, v, ac, vc)`.

Policy Conflict Resolution I

Rule 6: When a health professional examines a patient, she/he must write notes in the patient's medical record.

Rule 7: When a mental health professional examines a patient, notes must be recorded in a place other than the patient's medical record.

Policy Conflict Resolution II

A transducer is *sequential* when, standing in any state and consuming any possible input symbol, there is only one edge with a matching input label. A transducer is *deterministic* when it produces at most one output string for each possible input string.

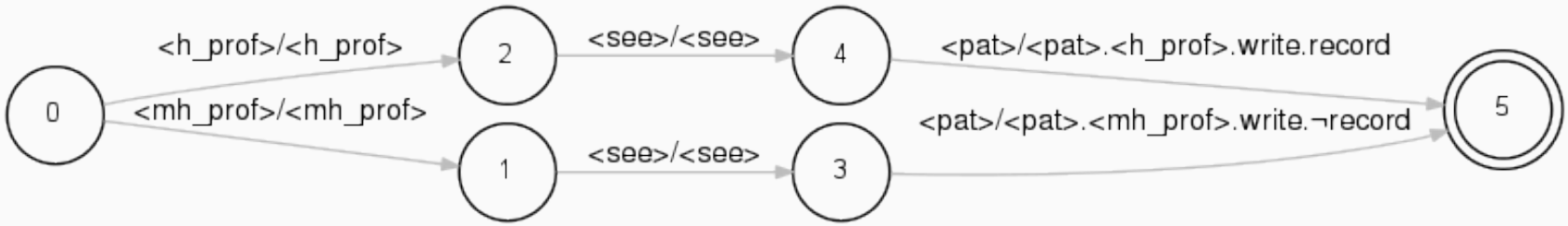


Figure 14: The Union of FSTs modelling rules 6 and 7.

This transducer is neither sequential, nor deterministic.

Policy Conflict Resolution III

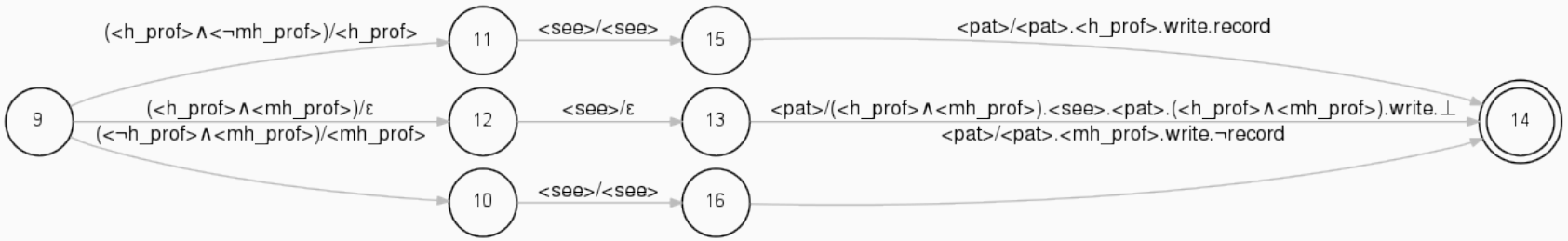


Figure 15: Previous FST determinized.

"Shortest"-Path for Conflict Resolution

- Once we have a policy transducer T, it is possible to compute the shortest path on the Cartesian product of the semiring $\{0,1\} \times \mathbb{R}^+ \times \{0,1\} \times \mathbb{R}^+$ computing the closure T^*.
- Submodules can be used to model action contradictions and remove them from the valid paths.

Interoperability Analysis

Interoperability for Policy-based Management

Interoperability: "The capability to communicate, execute programs, or transfer data among various functional units in a manner that requires the user to have little or no knowledge of the unique characteristics of those units" [3]

What we do analyze when we analyze Interoperability?

1. what is permitted if we apply the security rules of two organizations at the same time?
2. are the obligated actions consistent between them and with the prohibitions of both organizations?
3. is the emerging policy in line with their cooperation objectives?

Interoperability Analysis I

- subjects, objects and actions at different organizations may have different names, classifications, and meanings.
- we need to draw a correspondence map between entities and procedures.
- this means to translate the entity names and,
- to translate their interaction manners.

FST Composition for Policy Mapping

Lets consider two hospitals with two similar rules:

Rule 8: `Physicians can manage any medical record.`

Rule 9: `Health Professionals can manage any medical record.`

managing a medical record includes actions such as *reading*, *writing*, and *signing* one of its entries.

for H' signing and writing a medical record are things completely different and for H" to sign a medical record is a particular case of writing it.

FST Composition for Policy Mapping

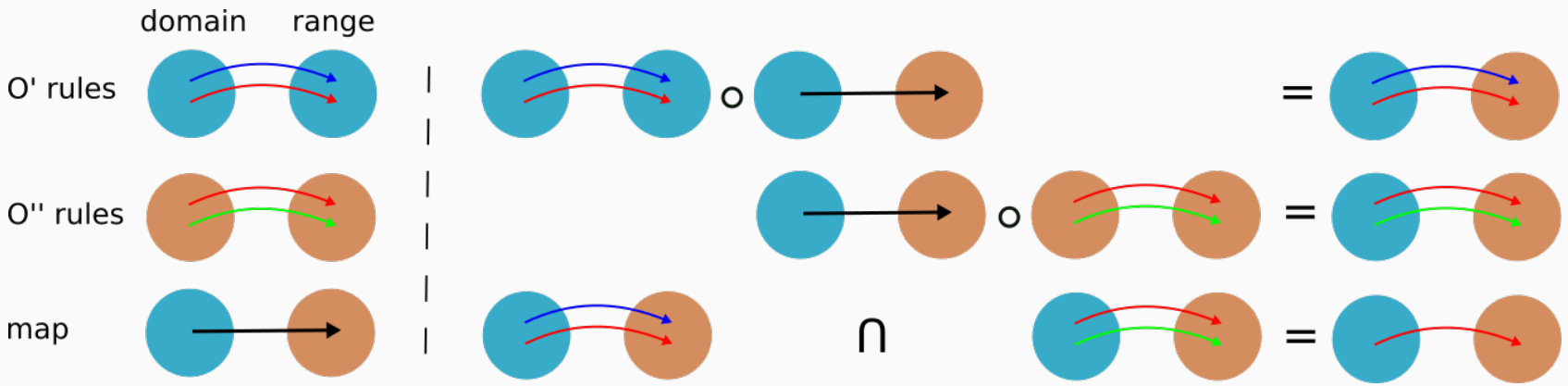


Figure 16: General interoperability analysis process.

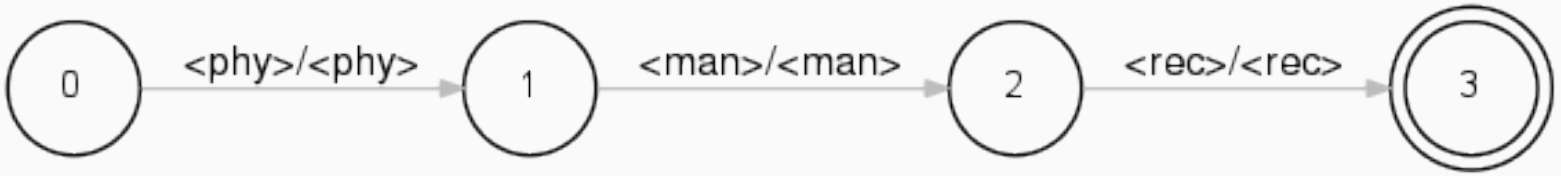


Figure 17: FST T_{Rule8} modelling Rule 8. The label *phy* represents the set of *physicians*, *man* represents the class of actions *management* and *rec* the class of objects *medical records*.

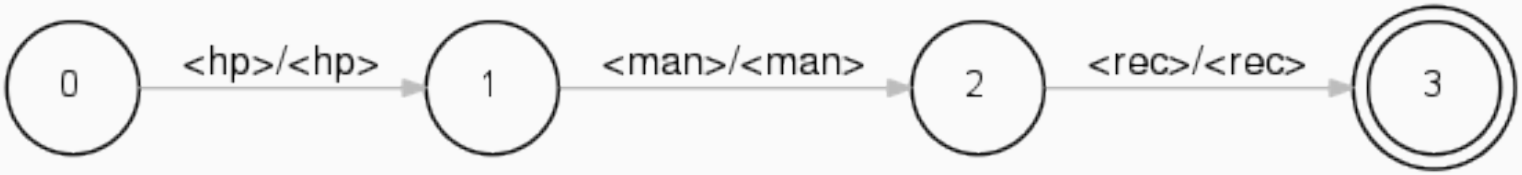


Figure 18: FST T_{Rule9} modelling Rule 9. The label *hp* represents the set of subjects *health professionals*, *man* represents the class of actions *management* and *rec* the class of objects *medical records*.

FST Composition for Policy Mapping

Physicians maps to *Health Professionals*, but not when writting a health record.

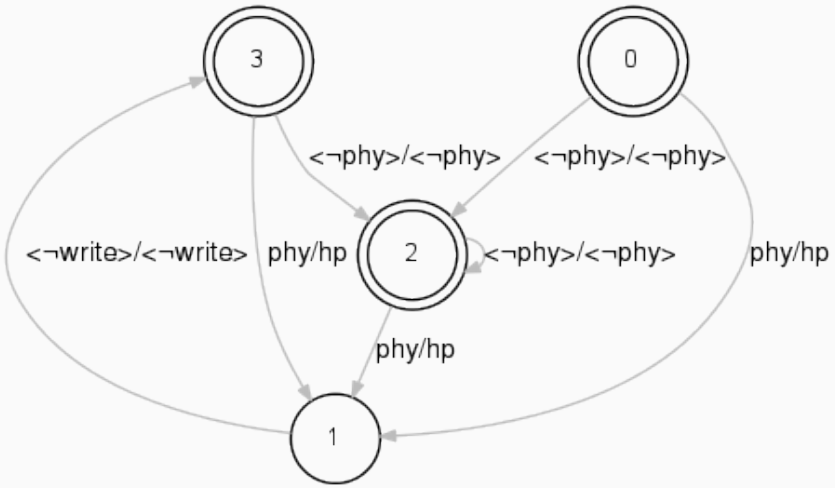


Figure 19: FST T_{map} which defines a map between the entities of two simple permissions.

FST Composition for Policy Mapping

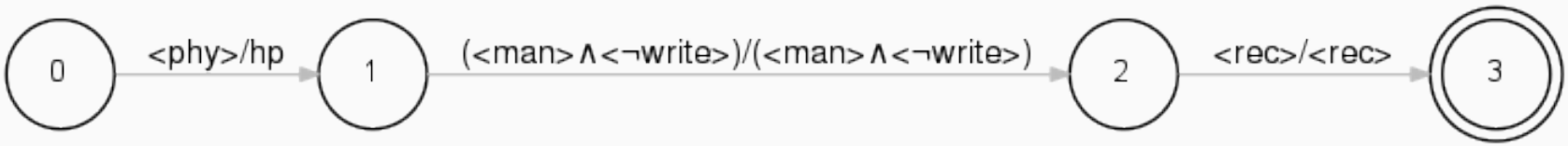


Figure 20: The FST $T_{R8\_map} = T_{Rule\ 8} \circ T_{map}$. This models *"Physicians can manage any medical record."* with its output actions written in terms of the entities of H" policy.

This computation is based on the presumption that $read \rightarrow man$, $man \rightarrow \neg rec$, $man \rightarrow \neg phy$, and $rec \rightarrow \neg phy$.

FST Composition for Policy Mapping

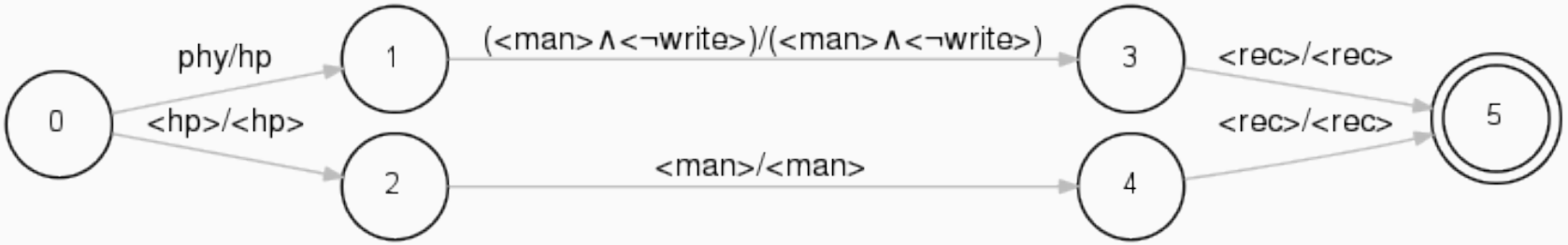


Figure 21: The FST $T_{map\_R9} = T_{map} \circ T_{Rule\ 9}$. This models *"Health Professionals can manage any medicalrecord."* with its input events written in terms of the entities of H' policy.

This computation is based on the presumption that $read \rightarrow man$, $man \rightarrow \neg rec$, $man \rightarrow \neg phy$, $man \rightarrow \neg hp$, $rec \rightarrow \neg hp$, $rec \rightarrow \neg phy$, $hp \rightarrow \neg phy$.

FST Composition for Policy Mapping

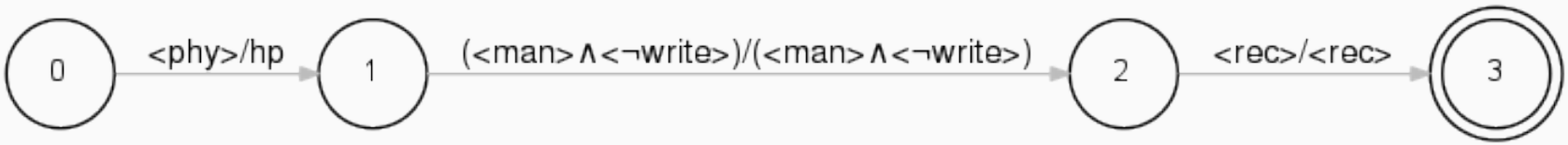


Figure 22: The τ-FST $T_{P^+_{int}} = T_{R8\_map} \cap T_{map\_R9}$. T_{int} is the transducer representing what a subject of H' can do on H" objects when the policies of both organization are applied at the same time.

FST Composition for Policy Mapping

The procedure to find out what are the actions that a subject of a provider O' can do on the objects of another provider O", follows the steps below:

1. to model the policies of organization O' as the transducer $T_{O'}$,
2. to model the policies of organization O" as the transducer $T_{O''}$,
3. to model the entity map between O' and O" as the transducer T_{map},
4. to compute $T_{O'\_map} = T_{O'} \circ T_{map}$,
5. to compute $T_{map\_O''} = T_{map} \circ T_{O''}$, and,
6. to compute $T_{int} = T_{O'\_map} \cap T_{map\_O''}$

T_{int} is the transducer which models the set of interoperability policy-rules between O' and O".

Conclusion

- We used the extensive FST and Semirings theory to *operate* with policies from different organizations and study their interrelations,
- we are able to compute the security policy that emerges from the combination of all the organization-level policies and,
- the higher-level interoperability policy when two organizations cooperate.
- This work was partially described in the paper:
  J. Baliosian and A. Cavalli, “An Abstraction for the Interoperability Analysis of Security Policies”, in proceedings of **NSS 2015**, 9th International Conference on Network and System Security, November 3-5, 2015,New York City, USA.

Further and Future Work

Further work:

- Writing rules back from FSTs, (made but too many slides)
- integration with Orbac policy manger and MotOrBac policy editor, (started)
- objective-oriented conflict resolution, (to be made)
- Self-Management (control loops for network management systems), (lots made on opportunistic networks, etc.)
- Human-walk modelling. This is starting just now.

“Frequently Asked Questions”

- Why using FSTs instead of FSRs?
- Why using FST instead of a logic programming approach?
- How weights are selected?
- How do I know that they are right?
- What is new? (your first work with FSTs is from 2004)
- How do I know that the model is complete?
- How do I know that the model is right?

Real Questions?

Writing Policies Back from FSTs I

The answer to the third question

is the emerging policy in line with their cooperation objectives?

worth a complete research work on itself, but, at least

we can write policy rules back from its FST model.

Writing Policies Back from FSTs II

Writing back a set of policies P_{int} in a high-level specification language such as OrBAC from T_{int}, is analogous to find

$$\widehat{\Pi}_{P^{+}_{int}} \subseteq \widehat{\Pi}_{P_{int}}$$

which corresponds to the permissions P^{+}_{int} and

$$\widehat{\Pi}_{P^{o}_{int}} \subseteq \widehat{\Pi}_{P_{int}}$$

for the obligations P^{o}_{int}.

(Re-writing the set of prohibitions does not have a meaning in a context where everything that is not permitted is prohibited)

Writing Policies Back from FSTs III

$\widehat{\Pi}_{P^+_{int}}$ is the subset of the relation that maps strings from the language of events L_{P^+}, where each string has the form *"context, role, activity, view"*, with exactly the same string.

$$T_{P^+_{int}} = id(R_{P^+}) \cap T_{P_{int}}$$

where R_{P^+} is the FSR that accepts L_{P^+}.

Writing Policies Back from FSTs IV

$\widehat{\Pi}_{P^o_{int}}$ is the subset of the relation that maps *activation context* starting events (start_ac) to strings of the form *"role, activity, view, fulfilled"*, or strings with an *activation context* starting event and a *violation context* starting event (start_vc) to strings of the form *"role, action, view, fulfilled, violated"*.

$\widehat{\Pi}_{P^o_{int}}$ is defined by the FST

$$T_{P^o_{int}} = T_{P^o} \cap T_{P_{int}}$$

Writing Policies Back from FSTs V

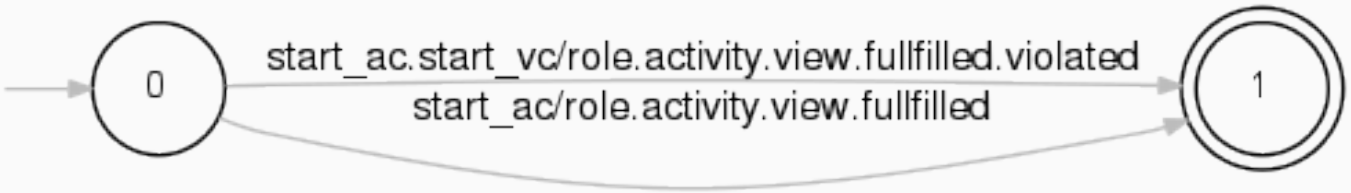


Figure 23: The transducer T_{po} used to compute $T_{po_{int}}$

All the FST must be determinised and in its minimal representation.

Writing Policies Back from FSTs VI

Each valid path in $T_{P^+_{int}}$ corresponds to a permission in P^+_{int}, and, each valid path in $T_{P^o_{int}}$ corresponds to an obligation in P^o_{int}.

For example, if the path in Figure 51 bellongs to $T_{P^+_{int}}$ the policy

`permission(p,doctor,write,medical_record,context_1)`

is in the set P^+_{int}.

Writing Policies Back from FSTs VII

bellongs to $T_{P^+_{int}}$, then, the policy
`permission(p,doctor,write,medical_record,context_1)`
is in the set P^+_{int}.

Writing Policies Back from FSTs VIII

1. compute T_{int} (as seen before),
2. compute $T_{P^+_{int}} = id(R_{P^+}) \cap T_{P_{int}}$,
3. compute $T_{P^o_{int}} = T_{P^o} \cap T_{P_{int}}$,
4. find each valid path in $T_{P^+_{int}}$,
5. write a permission for each path using the *context*, *role*, *activity*, and *view* in the path,
6. find each valid path in $T_{P^o_{int}}$, and,
7. write an obligation for each path using the *activation context*, *violation context*, *role*, *activity*, and *view* in the path.

Further Reading I


J. Baliosian and D. Wonsever, “Finite State Transducers,” in Handbook of Finite State Based Models and Applications, Chapman and Hall/CRC, 2012, pp. 45–68.


Elrakaiby, Y., Cuppens, F., Cuppens-Boulahia, N.: Formal enforcement and management of obligation policies. Data & Knowledge Engineering 71(1), 127–147 (Jan 2012), http://linkinghub.elsevier.com/retrieve/pii/S0169023X11001248


ISO: ISO\slash IEC 2382-1:1993 Information technology — Vocabulary — Part 1: Fundamental terms. International Organization for Standardization, Geneva, Switzerland (1993), http://www.iso.ch/cate/d7229.html


Kalam, A., Baida, R., Balbiani, P., Benferhat, S., Cuppens, F., Deswarte, Y., Miege, A., Saurel, C., Trouessin, G.: Organization based access control. In: Proceedings POLICY 2003. IEEE 4th International Workshop on Policies for Distributed Systems and Networks. pp. 120–131. IEEE Comput. Soc (2003), http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1206966


Lupu, E.C., Sloman, M.: Conflicts in Policy-based Distributed Systems Management . IEEE Transactions on Software Engineering 25(6), 852–869 (1999)


Mohri, M.: Weighted automata algorithms. In: Handbook of weighted automata, pp. 213–254 (2009), http://www.springerlink.com/index/P872G5Q565H44544.pdf

Further Reading II

- van Noord, G., Gerdemann, D.: Finite State Transducers with Predicates and Identities . Grammars 4(3), 263–286 (Dec 2001)
- Roche, E., Schabes, Y.: Finite-State Language Processing . Tech. rep., MIT Press, Cambridge, Massachusetts. (1997)
- Sloman, M.: Policy Driven Management for Distributed Systems . Journal of Network and Systems Management 2, 333 (1994)
- Veanes, M., Hooimeijer, P., Livshits, B., Molnar, D., Bjorner, N.: Symbolic finite state transducers. ACM SIGPLAN Notices 47(1), 137 (Jan 2012), http://dl.acm.org/citation.cfm?doid=2103621.2103674

More Questions?