P.A. Bonatti
Datalog 2.0 Workshop

Oxford, March 2010
Main goals of this talk

- Highlighting the (current) advantages of Datalog-based approaches as policy languages
  - Only a small fraction of the “Datalog for security, privacy, and trust” topic
Main goals of this talk

- Highlighting the (current) advantages of Datalog-based approaches as policy languages
  - Only a small fraction of the “Datalog for security, privacy, and trust” topic
  - No time for advertising our Protune trust negotiation framework developed in REWERSE
- Show the need for a formal clean-up of a savagely proliferating area
Privacy and confidentiality policies

- In their simplest form constrain
  - Access to information / knowledge (server's view)
  - Disclosure of information / knowledge (user's view)
    - e.g. when accounts are created, credit card numbers released
Privacy and confidentiality policies

- In their simplest form constrain
  - Access to information / knowledge (server's view)
  - Disclosure of information / knowledge (user's view)
    - e.g. when accounts are created, credit card numbers released

- Based on
  - Properties of the requester
  - Information / knowledge contents
  - The nature of the current transaction / operation
  - Contextual properties (time, place, etc.)
Privacy and confidentiality policies

- In their simplest form constrain
  - Access to information / knowledge (server's view)
  - Disclosure of information / knowledge (user's view)
    - e.g. when accounts are created, credit card numbers released

- Based on
  - Properties of the requester
  - Information / knowledge contents
  - The nature of the current transaction / operation
  - Contextual properties (time, place, etc.)

- Expressiveness needs for policy languages
  - Complex conditions
  - Over all sorts of knowledge and data
Access control & information disclosure depend on metadata such as:
- User profiles
- Relationships between users
  - Friendship
  - Reputation
- Content classification
- etc...

Such metadata are encoded with KR languages
- RDF / Description logics
- Rules
- In perspective, combinations thereof
Policies for enterprise data

- Recent initiatives aimed at applying the LOD paradigm to organization data / knowledge management
- Increasing use of RDF and OWL
Policies *languages* for semantic web, social networks etc

- **KR languages are a natural choice**
  - Uniform representation of usage constraints & support knowledge
- **Existing DL-based proposals**
  - KAoS
  - Rei
- **Existing rule-based proposal**
  - Cassandra (Datalog + constraints)
  - RT family
  - PeerTrust (distributed Datalog)
  - TrustBuilder
  - Protune (Datalog + O.O. syntactic sugar + metalanguage)
Orienteering

- Need for a formal framework for assessing and comparing these policy languages and more
- Exploiting multidisciplinary expertise to highlight strengths and (sometimes serious) weaknesses
Outline

- Some considerations on expressiveness
- Some considerations on reasoning mechanisms
- Conclusions & further needs

- No time for usability, usage control, disclosure minimization and other evolving topics
Outline

- Some considerations on expressiveness
- Some considerations on reasoning mechanisms
- Conclusions & further needs

- No time for usability, usage control, disclosure minimization and other evolving topics
What is a policy in the simplest case?

- In abstract terms, just a mapping...
  - From contexts
    - Database tables, RDF triples, XML documents...
    - Essentially, finite structures (potentially large!)  
  - To authorizations,
  - that can be represented in relational forms
    - Access control matrices *et similia*
    - `<subject, object, action,...>` tuples
What is a policy in the simplest case?

- In abstract terms, just a mapping...
  - From *contexts*
    - Database tables, RDF triples, XML documents...
    - Essentially, finite structures (potentially large!)
  - To *authorizations*,
  - that can be represented in relational forms
    - Access control matrices *et similia*
    - `<subject, object, action,...>` tuples

- Essentially a *query*
Descriptive complexity

A well understood way of measuring the expressiveness of query languages
- A good candidate for policy languages ...

Expressiveness of a language:
- The class of mappings it can express
- It frequently coincides with a complexity class
- Example:
  - if the descriptive complexity of L1 is PSPACE
  - and the descriptive complexity of L2 is EXPTIME
  - then L1 is “less expressive” than L2
Descriptive complexity

- Many results for rule-based languages
  - When the context is a set of facts...
- Missing results:
  - Descriptive complexity of DLs
- We can't use descriptive complexity to compare DL-based policy languages right away
  - A nice motivation for further work on DLs...
- However some preliminary observations are possible
Easy observations on DL

- DL typically enjoy tree- or forest-model properties
- Therefore DL cannot uniformly express cyclic patterns
  - There exist simple PTIME-computable policies that cannot be expressed with DL
  - We will make an effort to identify *practically relevant* such policies
- Difficulties also with conditions involving 3 or more individuals
  - Basic DLs are fragments of 2-variable logic
  - Only partially relaxed by additional constructs such as generalized quantifiers
Simple policies for complex DL

- Allow access if:
  - medical_record(R), patient(R,P), cures(Doctor,P)
  - user(U), picture_of(Pic,Owner), friend(Owner,U)
  - id(ID), credit_card(CC), owner(ID,User), owner(CC,User)
Simple policies for complex DL

- Allow access if:
  - medical_record(R), patient(R,P), cures(Doctor,P)
  - user(U), picture_of(Pic,Owner), friend(Owner,U)
  - id(ID), credit_card(CC), owner(ID,User), owner(CC,User)

- Ternary formulas! Partial workaround for DLs:
  - Reification: represent context as an individual with 3 attributes
  - $\exists id \land \exists credit\_card \land \exists user$
Simple policies for complex DL

- Allow access if:
  - medical_record(R), patient(R,P), cures(Doctor,P)
  - user(U), picture_of(Pic,Owner), friend(Owner,U)
  - id(ID), credit_card(CC), owner(ID,User), owner(CC,User)

- Ternary formulas! Partial workaround for DLs:
  - Reification: represent context as an individual with 3 attributes
  - $\exists \text{id} \land \exists \text{credit\_card} \land \exists \text{User} \land ???$
Simple policies for complex DL

- Allow access if:
  - medical_record(R), patient(R,P), cures(Doctor,P)
  - user(U), picture_of(Target,Owner), friend(Owner,U)
  - id(ID), credit_card(CC), owner(ID,User), owner(CC,User)

- Ternary formulas! Partial workaround for DLs:
  - Reification: represent context as an individual with 3 attributes
    - $\exists id \land \exists credit\_card \land \exists user \land ???$
  - $\mathbf{ALC, SHIQ}$: No way: tree/forest-model property
KAoS's approach

- Role-value maps + role composition [CCGRID'05]
  \[ \exists \text{id} \land \exists \text{credit_card} \land \exists \text{user} \land \text{id}_{\text{owner}} = \text{credit_card}_{\text{owner}} \]
KAoS's approach

- Role-value maps + role composition [CCGRID'05]
  \[ \exists id \ \land \ \exists credit\_card \ \land \ \exists user \ \land id \text{\_owner} = credit\_card \text{\_owner} \]
- Problem: reasoning becomes undecidable
  - Concept subsumption in \( \mathcal{AL} \) with role-value maps and role composition is undecidable (!)
  - cf. survey in the Handbook of Description Logics, Ch. 5
- Possible consequences:
  - Access control does not terminate
  - Unauthorized access
  - Denial of service (improperly denied access)
  - Some policies are “illegal” (which ones?)
- KAoS's solution: not specified?!?
Datalog policy languages

- A minor difficulty: Only stratified negation is allowed
  - Multiple models undesirable (access control policies are supposed to be unambiguous)
  - Stratified neg. not enough to express all PTIME policies *but*
  - An ordering on the domain is enough (like Prolog's @>) to express *all policies in PTIME*
Datalog policy languages

- A minor difficulty: Only stratified negation is allowed
  - Multiple models undesirable (access control policies are supposed to be unambiguous)
  - Stratified neg. not enough to express all PTIME policies but
  - An ordering on the domain is enough (like Prolog's @>) to express all policies in PTIME

- Further restrictions on policy languages
  - Policies should be monotonic w.r.t. the digital credentials disclosed (which are part of the context)
  - Rationale: no reliable way to check whether a user does not have a credential
  - Open question: can restricted Datalog-based policy languages express all credential-monotonic policies?
Summary on expressiveness

- Datalog-based languages are much less problematic from the expressiveness point of view
  - well-suited to popular reference applications
  - no expressiveness gaps
Outline

- Expressiveness
- **Reasoning**
- Conclusions & further needs
Reasoning tasks

- Deduction
  - e.g.: is Auth entailed by Policy + Context?
- Highly mature, both in DLs and rule languages
  - Tableaux, optimizations & heuristics
  - Abstract machines, intelligent grounding, ...
Reasoning tasks

- Deduction

  however, more is needed

- Nonmonotonic reasoning
- Abduction
- Policy comparison (query containment)
Reasoning tasks: purposes (I)

- Deduction: access control
  - is authorization $A$ entailed by policy $P$?
- Nonmonotonic reasoning: default decisions
  - open/closed policies
Reasoning tasks: purposes (I)

- Deduction: access control
  - is authorization $A$ entailed by policy $P$?
- Nonmonotonic reasoning: default decisions
  - open/closed policies
  - inheritance *with exceptions* along subject/object/role hierarchies
Reasoning tasks: purposes (I)

- Deduction: access control
  - is authorization $A$ entailed by policy $P$?
- Nonmonotonic reasoning: default decisions
  - open/closed policies
  - inheritance *with exceptions* along subject/object/role hierarchies
  - conflict resolution (e.g. denials/most specific take precedence)
- Note: all these mechanisms have been independently introduced by researchers on security, not AI guys
Reasoning tasks: purposes (II)

- Abduction: credential selection (trust negotiation)
  - Given authorization $A$, a $Policy$, and a portfolio $P$
  - Find a set of credentials $C \subseteq P$ such that
    $$Policy \cup C \models A$$
  - Warning: somebody does not know that this is a classically sound inference... [Kagal et al. POLICY 08]
Reasoning tasks: purposes (III)

- Policy comparison
  - does P1 grant at most the same authorizations as P2
  - in *all contexts*?

  useful for

- P3P-like compliance
  - is X’s policy compatible with Bob's privacy preferences?

- Validation
  - does the last update restrict/enlarge the policy?
Reasoning mechanisms: maturity
Reasoning mechanisms: maturity

- Nonmonotonic reasoning
  - Highly engineered and optimized implementations for rule languages / LP / ASP (negation as failure)
    - and policy models such as FAF (stratified LP+methodology)
  - Only theoretical results for description logics
    - High complexity: up to NexpTime\(^{NP}\) and 3ExpTime
    - More practical approaches are still work in progress:
      - DL-lite, $\mathcal{EL}$ [B., Faella, Sauro IJCAI'09]
      - No implementations
Reasoning mechanisms: maturity

- Abduction
  - Well-established approaches for logic programming
    - Starting with [Eshghi ICLP'88]
    - Several systems exist: ACLP, A-system, CIFF, SCIFF, ABDUAL, ProLogICA, and ASP-based implementations
  - Relatively recent approaches for DLs
    - [Di Noia et al. IJCAI'03] based on concept length / maximality w.r.t. subsumption / number of conjuncts
    - Tableaux algorithm in [Colucci et al. DL'04]
    - More general approaches from [Elsenbroich et al. OWLED'06]
    - No direct support from main DL engines yet
Reasoning mechanisms: maturity

- Policy comparison
  - Naturally supported by DLs
    - Subsumption checking
  - More complex for LP, due to general recursion
    - Equivalent to *Datalog query containment*
      - In general undecidable
      - Highly complex in many cases
  - Low-complexity solution in [POLICY'08]: Restricted recursion
    - Still covering *inheritance hierarchies, certificate chains*
    - Acceptable complexity via:
      - preprocessing + classical algorithm for conjunctive queries
      - Prototypical implementation, positive experimental results
## Reasoning mechanisms: maturity

- **Policy comparison for LP**
  - Experimental evaluation on artificial worst cases

<table>
<thead>
<tr>
<th>Body len</th>
<th>N rules</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
<th>100</th>
<th>150</th>
<th>200</th>
<th>250</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>.05</td>
<td>.08</td>
<td>.17</td>
<td>.27</td>
<td>.39</td>
<td>1.44</td>
<td>3.22</td>
<td>5.57</td>
<td>8.64</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td>.12</td>
<td>.33</td>
<td>.60</td>
<td>1.01</td>
<td>1.54</td>
<td>6.14</td>
<td>13.70</td>
<td>23.84</td>
<td>37.33</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td>.25</td>
<td>.76</td>
<td>1.61</td>
<td>2.88</td>
<td>4.45</td>
<td>16.99</td>
<td>39.00</td>
<td>68.80</td>
<td>108.17</td>
<td></td>
</tr>
<tr>
<td>40</td>
<td>.43</td>
<td>1.59</td>
<td>3.47</td>
<td>6.10</td>
<td>9.32</td>
<td>37.46</td>
<td>84.36</td>
<td>150.98</td>
<td>234.45</td>
<td></td>
</tr>
<tr>
<td>50</td>
<td>.76</td>
<td>2.88</td>
<td>6.49</td>
<td>11.50</td>
<td>17.63</td>
<td>70.63</td>
<td>161.92</td>
<td>279.65</td>
<td>442.37</td>
<td></td>
</tr>
</tbody>
</table>

Worst case performance
(in seconds)
Summary and conclusions
Summary and conclusions

Today

- Datalog-based policy languages can generally rely on more mature
  - foundations,
  - methodologies,
  - implementations

This may change in the future,

- as progress is being made on DL extensions and reasoning
  - nonmonotonic extensions
  - abduction
  - explanations (that we have not touched today)
Summary and conclusions

- Further opportunities for interesting work
  - Incomplete contexts (due to ontologies)
    - Old relevant work on querying disjunctive databases [B. & Eiter TCS 1996]
    - The standard stable model semantics has limitations
Summary and conclusions

Further opportunities for interesting work
- Incomplete contexts (due to ontologies)
  - Old relevant work on querying disjunctive databases [B. & Eiter TCS 1996]
  - The standard stable model semantics has limitations
- Hybrid approaches (DL + rules, perhaps DL queries)
  - Enhanced expressiveness
    - Full integration of policies and domain ontologies
  - Inherit problems
    - Undecidable policy comparison
    - Maturity (explanations, abduction, advanced implementations)
Summary and conclusions

- Further opportunities for interesting work
  - Incomplete contexts (due to ontologies)
    - Old relevant work on querying disjunctive databases [B. & Eiter TCS 1996]
    - The standard stable model semantics has limitations
  - Hybrid approaches (DL + rules, perhaps DL queries)
    - Enhanced expressiveness
      - Full integration of policies and domain ontologies
    - Inherit problems
      - Undecidable policy comparison
      - Maturity (explanations, abduction, advanced implementations)
  - More results on comparison of rule-based policies
    - Extending the class of comparable policies
    - With practical algorithms
Summary and conclusions

- Further opportunities for interesting work include three topics we have not touched today:
  - Large scale policy reasoning, using billions of RDF triples...
Further opportunities for interesting work include three topics we have not touched today:

- **Large scale** policy reasoning, using billions of RDF triples...
- **Usage control**: say what to do with your information after you disclose it
  - Dynamic aspects, delegation, obligations
    - Multimodal, dynamic logics
  - Enforcement problems (voluntary?)
  - Expressiveness criteria / techniques?
Summary and conclusions

Further opportunities for interesting work include three topics we have not touched today:

- **Large scale** policy reasoning, using billions of RDF triples...
- **Usage control**: say what to do with your information after you disclose it
  - Dynamic aspects, delegation, obligations
    - Multimodal, dynamic logics
  - Enforcement problems (voluntary?)
  - Expressiveness criteria / techniques?
- **The BIG, BAD open problem**: usability
  - Esp. ability of writing correct policies
  - Strong negative experimental results (CMU)
  - Explanation facilities, what-if scenarios, auto documentation (see also ProtuneX)
QUESTIONS/DISCUSSION?
A less formal view of expressiveness

- Easy for DLs, hard for rules:
  - Asserting the existence of anonymous individuals
    \[ \exists \text{mother}.\text{Human} \text{(John)} \]
  - Rule skolemization makes reasoning undecidable, in general
    - but see *finitary* and *FDNC* logic programs (ASP)

- Easy for rules, hard for DLs:
  - Conditions involving 3 or more individuals
  - Cyclic patterns
    because
  - DLs are frequently fragments of 2-variable logic
  - and frequently enjoy tree- or forest-model properties
Looking for a solution

- “Features” and concrete domains [Lutz, KR'02]
  - Concrete domains: consist of distinguished nonstructured elements (numbers, etc.)
  - Feature paths: compositions of functional roles, ending with a “concrete role” (whose range is a concrete dom.)
  - $\forall fp_1, fp_2$. = similar to role-value map ($fp_i$ are feat.paths)

- Current limitation
  - Decidability results cover inverse and/or nonfunctional roles $R$ only if $fp_1 = R \circ g_1$ and $fp_2 = g_2$, with $g_1$ and $g_2$ concrete features
Grant access to “abc.pdf” to owner's friends

- Typically non functional
- Inverse role
- \( target = subject \circ friend \circ owns \)
Outline

- Expressiveness
- Reasoning
- Usability
- Conclusions & further needs
Usability facets

- Formulating policies
- Understanding policies
  - static
- Understanding transaction outcomes
  - dynamic, context dependent
- No assumption on user's background
Usability facets: maturity

- Formulating policies
  - GUI for simple languages (Cranor and Sadeh @ CMU)
    - and machine learning
  - Controlled Natural Language (mainly Attempto)
  - Same level of (im)maturity for both DL and rules

- Understanding policies

- Understanding transaction outcomes
  - Explanation facilities
  - Discussed in the next slides
Explanation facilities

- History
  - Introduced since pioneering work on expert systems
  - Today: second generation explanation facilities
  - DL approaches started in [McGuinness, Borgida IJCAI'95]
  - However the benchmark is not a generic approach...

- Protune-X: second generation explanations
  - [ECAI'06] B., Olmedilla, Peer + Sauro
  - Tailored to trust negotiation to obtain
  - Generic heuristics
  - Deployment ease
Second generation features and Protune-X

- User-oriented navigation (proof tree not enough)
- Departure from engine behavior / tracing
Second generation features and Protune-X

- User-oriented navigation (proof tree not enough)
  - All proof attempts, local + global information

![Diagram showing REVERSE and PROTUNE Why-Not Explanations]

- including failures
- directly applicable rules
- true
- fail
- final answer
Second generation features and Protune-X

- Focus on user's interests (I): removing irrelevant information

2.
- J. Smith is authenticated
- but the following conditions cannot be simultaneously satisfied:
  - J. Smith subscribed some Subscription [Subscription = basic computer pubs] [Subscription = basic law pubs]
  - paper_0123.pdf is available for the Subscription [Subscription = complete computer pubs] [Subscription = gold subscription]
Second generation features and Protune-X

- Focus on user's interests (I): removing irrelevant information

Generic heuristics: auto-generated meta-annotations (blurring)
Second generation features and Protune-X

- Focus on user's interests (II): responsibilities
  - ad-hoc for trust negotiation, extendible to other app.s

Responsibilities automatically identified through dependency analysis based on independently motivated meta-information about actions.
Second generation features and Protune-X

- Key attributes, or denoting structured objects
  - Pre-specified in classical approaches
  - Dynamic in Protune-X

![Protune-X example]

aggregation of multiple literals (dynamically selected) that uniquely identify an object

Partial mismatch better explains failure
Summary of Protune-X's queries

- Static:
  - How-to

- Dynamic, context dependent
  - Why / why not
  - What-if
    - Simulated scenarios