Multiple Ownership in Access Control

Pooya Mehregan

University of Calgary

March 23, 2016
Problem Statement

Results

Design Patterns

Negotiation Protocol

Conclusion
Problem Statement

Discretionary Access Control (DAC)

- Single owner
- Delegation

<table>
<thead>
<tr>
<th>Subjects</th>
<th>o₁</th>
<th>o₂</th>
<th>o₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>u₁</td>
<td>r, w, own</td>
<td></td>
<td>r</td>
</tr>
<tr>
<td>u₂</td>
<td>r, w</td>
<td>r, own</td>
<td>w</td>
</tr>
<tr>
<td>u₃</td>
<td>r, w</td>
<td>w, own</td>
<td></td>
</tr>
</tbody>
</table>

Social Computing Era

- Multiple owners
Problem Statement

(a) Tagging

(b) Friend-list

Problem Setting

1. Must honour all co-owners’ privacy preferences
2. Must address all co-owners’ sharing requirements
Related Work

Assumptions and Approaches

- All problem instances are the same
  - **Approach**: need general-purpose schemes
- Diverse privacy preferences lead to conflicts
  - **Approach**: need unsupervised conflict resolution

Gaps and Shortcomings

1. Complex black-box general-purpose schemes
   - Have usability/comprehensibility issue (e.g. auction-based, threshold-based, ...)
2. No interaction/collaboration among co-owners
   - Could compromise privacy/sharing
Results
Results

Categories

Annotations
(like, tag, share & comment)

Joint Declaration
.relationship articulation)

Collaborative Authoring
(Google Docs & wikis)

Approaches

Design Patterns

Interactive Negotiation

Multiple Ownership Instances

Pooya Mehregan (UofC)
Design Patterns
We propose

1. Simple annotation
2. Higher-order annotation

Covers
Tag, share and comment

Does not cover
Joint declaration and collaborative authoring
Standard SQL implementation (i.e. using Common Table Expressions)

Anonymized LiveJournal dataset from Stanford Large Network Dataset Collection

- $|V| = 4,847,571$
- $|E| = 2 \times 68,993,773$
- $\text{Pol.: } \{\text{Me, Friends, FOF, Everyone}\}$
- $\text{Rep.} = 1000$
- Confidence Interval

$= 95\%$
Negotiation Protocol
Rationale

- Empirical studies suggest users do negotiate their privacy preferences in absence of tool support
- Users can mitigate/prevent conflicts by negotiation

Context

- Where express consent is mandatory (i.e. for accountability purposes)
- In structured environments where fewer co-owners exist (feasibility)
We propose a novel access control policy negotiation protocol

- Extends ReBAC to ReBAC/MO to support multiple ownership
- Enables co-owners to explicitly consent to the access control policy
- Offers easy and structured policy revision process
- Addresses sharing need via formal availability criteria
- Provides tool support for deciding the availability criteria
Negotiation Protocol

1. Notify
   - intermediate formula $\phi$
   - a boolean flag indicating if $\phi$ satisfies the availability criterion $\kappa_u$

2. Consent/Revise
   - Consent: adopt formula $\phi$ for the co-owned object $o$
   - Revise:
     - the availability criterion $\kappa_u$ and/or
     - the atomic policies ($\alpha$’s and $\beta$’s), user $u$ has contributed

3. Verify
We formally define 3 novel availability criteria

- **Satisfiability**
  - E.g. “Object $o$ shall be accessible to at least 35 users in the current protection state.”

- **Feasibility**
  - E.g. “Object $o$ shall be made accessible to 20 (or more) users if the company hires/fires no more than 10 employees.”

- **Resiliency**
  - E.g. “Even if I were to register/dismiss 10 patients, object $o$ shall remain accessible to at least 5 users”
Availability Criteria

We prove that all three availability criteria are in the second level of the Polynomial Hierarchy.

\[
\begin{align*}
\Sigma_3^P & \uparrow \uparrow \leftarrow \Pi_3^P \\
\Delta_3^P & \downarrow \\
\Sigma_2^P & = NP^{NP} & \Pi_2^P & = coNP^{NP} \\
\Delta_2^P & = P^{NP} \\
\Sigma_1^P & = NP & \Pi_1^P & = coNP \\
\Sigma_0^P & = \Delta_1^P = P = \Pi_0^P
\end{align*}
\]

**Figure:** The Polynomial Hierarchy
Graph

- $\langle V, \{ R_l \}_{l \in \mathcal{L}} \rangle$
- $R_l \subseteq V \times V$

Subgraph

- $G_1 \subseteq G_2$ iff
  $V(G_1) \subseteq V(G_2)$, $L(G_1) \subseteq L(G_2)$, and $R_l(G_1) \subseteq R_l(G_2)$ for every $l \in L(G_1)$.

Isomorphism

- $G_1 \simeq G_2$ iff there exists a bijection $f : V(G_1) \to V(G_2)$ s.t.
  $L(G_1) = L(G_2)$, and for every $l \in L(G_1)$, for every $u, v \in V(G_1)$, we have $(u, v) \in R_l(G_1) \iff (f(u), f(v)) \in R_l(G_2)$. 
Background

Birroected Graph

- $BG = G_{(u, v)}$
- $u$ and $v$ are called the **owner root** and **requester root** respectively.

Birroected Subgraph

- $BG_1 = G_{1(u_1, v_1)}$ is a **(birroected) subgraph** of $BG_2 = G_{2(u_2, v_2)}$ iff $u_1 = u_2$, $v_1 = v_2$, and $G_1 \subseteq G_2$.

Birroected Graph Isomorphism

- $BG_1 \simeq BG_2$ iff there exists bijection $f : V(G_1) \rightarrow V(G_2)$ s.t. $f(u_1) = u_2$, $f(v_1) = v_2$, and $G_1 \simeq f G_2$.

Birroected Subgraph Isomorphism

- $BG_1 \preceq BG_2$ iff there exists $BG_3 \subseteq BG_2$ s.t. $BG_1 \simeq BG_3$
Policy Syntax

- \( \phi = (\alpha_1 \lor \alpha_2 \lor \ldots \lor \alpha_m) \land (\neg \beta_1 \land \neg \beta_2 \land \ldots \land \neg \beta_n), \)
- where \( \alpha_i = \text{acc}(BG_i, u_i) \) and \( \beta_j = \text{acc}(BG_j, u_j) \) are atomic policies.

Policy Semantics

- \( G, v \models \text{acc}(BG, u) \iff BG \preceq G_{(u,v)}. \)
Background

SAT solvers are used as an NP oracle for verifying Satisfiability (MO-SAT).

### Boolean Satisfiability (SAT)

- **CNF formula** $\phi$

  $$\phi = \bigwedge_{i=1}^{n} C_i$$

- $C_i$ is a disjunction of literals $l_j$

- Each literal $l_j$ is either an atom $a_k$ or its negation ($\neg a_k$)

**Decision Question.** Does there exist a truth assignment for a CNF formula $\phi$ which evaluates to true?
QBF solvers are used for verifying Satisfiability (MO-SAT)

### Quantified Boolean Formula Satisfiability (QBF)

- A QBF is formula $\phi$ generated using the following grammar

  $$\phi ::= x \mid \neg \phi \mid \phi \land \phi \mid \exists x. \phi$$

  where $x$ is a boolean variable or atom.

**Decision Question.** Does there exist a truth assignment for a formula $\phi$ which evaluates to true?
Background

ASP solvers are used for verifying all Availability Criteria

Answer Set Programming (ASP)

• Logic Program like Prolog
• An ASP program comprises of rules $r$

$$r : \quad a_0 \leftarrow a_1, \ldots, a_m, \sim a_{m+1}, \ldots, \sim a_n$$

• We define:
  • $\text{head}(r) = a_0$
  • $\text{body}(r) = \{a_1, \ldots, a_m, \sim a_{m+1}, \ldots, \sim a_n\}$

• $r$ semantics:
  
  $\text{head}(r)$ holds if $\text{body}(r)$ holds.

• If $\text{body}(r) = \emptyset$ then $r$ is called $\textbf{fact}$ and it always holds.
• If $\text{head}(r) = \emptyset$ then $r$ is called $\textbf{integrity constraint}$ and it must never hold.
We employ two different approaches for deciding Satisfiability (MO-SAT):

1. Naively reducing MO-SAT to QBF and invoking a QBF solver
2. Invoking a SAT solver as an NP oracle for polynomially many times
   - Vertex Enumerator (VE)
   - Model Enumerator (ME)
Example Graphs and Patterns

(a) Social Graph

Assume $\phi = \alpha \land \neg \beta$, $\text{anch}(\alpha) = 1$ and $\text{anch}(\beta) = 4$
Reducing $\beta$ to SAT

Let $\text{own} \rightarrow 1$ and $\text{req} \rightarrow 2$ in $\beta$

$$
\phi = \\
\left\{ \\
\begin{align*}
(X_{1,1} \lor X_{1,2} \lor X_{1,3} \lor X_{1,4} \lor X_{1,5}) &= 1 \\
(X_{2,1} \lor X_{2,2} \lor X_{2,3} \lor X_{2,4} \lor X_{2,5}) &= 1 \\
\end{align*} \right\} \text{ function} \\
\left\{ \\
\begin{align*}
(X_{1,1} \lor X_{2,1}) &\leq 1 \land (X_{1,2} \lor X_{2,2}) &\leq 1 \\
(X_{1,3} \lor X_{2,3}) &\leq 1 \land (X_{1,4} \lor X_{2,4}) &\leq 1 \\
(X_{1,5} \lor X_{2,1}) &\leq 1 \\
\end{align*} \right\} \text{ one-to-one} \\
\left\{ \\
\begin{align*}
(\neg X_{1,1} \lor X_{2,2}) \land (\neg X_{1,2} \lor X_{2,3} \lor X_{2,5}) \land \\
(\neg X_{1,3}) \land (\neg X_{1,4} \lor X_{2,5}) \land (\neg X_{1,5}) \land \\
(X_{1,1}) \\
\end{align*} \right\} \text{ edge mapping} \\
\left\{ \\
\begin{align*}
\text{mapping the owner} \\
\end{align*} \right\}
$$
We run a series of experiments

- to compare MO-SAT verifiers, and
- to study if they are viable in mid-sized organizations
Verdict: VE significantly outperforms the QBF-based approach.
VerteX Enumerator vs. Model Enumerator

Verdict: ME is more efficient than VE.

Parameters:
- $Rep.: \ 20$; $k_u: \ 1$; $\#atom^+: \ 3$; $\#atom^-: \ 3$; $\#V_{pat^+}$
- $5$; $\#V_{pat^-}: \ 5$; $Prob_G: \ 0.1$; $Prob_{pat}: \ 0.1$; $|\mathcal{L}|: \ 1$
Model Enumerator in Large Graphs

Verdict: ME can be employed for deciding MO-SAT in the mid-sized organizations such as Google and Microsoft (about 100 K users).

Parameters:
• $Rep.$: 20;
• $k_u$: 1;
• $\#atom^+$: 3;
• $\#atom^-$: 3;
• $\#V_{pat^+}$:
• $\#V_{pat^-}$: 5;
• AvgDeg.$G$: 200;
• Prob.$pat$: 0.5;
• $|\mathcal{L}|$: 4;
We employ two different approaches for deciding Feasibility ($\text{MO-Fea}^{\text{MD}}$) and Resiliency ($\text{MO-Res}^{\text{MD}}$)

1. Using ME as subroutine (ME-based)

2. Reducing $\text{MO-Fea}^{\text{MD}}$ and $\text{MO-Res}^{\text{MD}}$ to an Answer Set Program (ASP) and invoking an ASP solver (ASP-based)

More about ASP

- ASP solvers like SAT solvers use intelligent back-tracking (DPLL)
- ASP language is First Order Logic (concise problem encoding)
- ASP can solve problems in $\Sigma_2^p$ (e.g. $\text{MO-Fea}^{\text{MD}}$ and $\text{MO-Res}^{\text{MD}}$)
- Although it is promised to be declarative, clever problem encoding is still crucial
Example Graphs and Patterns

(a) Social Graph

(b) patt(α)

(c) patt(β)

Assume $\phi = \alpha \land \neg \beta$, anch(α) = 1 and anch(β) = 4
Reducing Feasibility to ASP

- \( e(l, 1, 2) \)
- \( e(l, 2, 3) \) (encoding immutable edges)
- \( e(l, 2, 5) \)
- \( e(l, 4, 3) \)

\( \text{mutable}(l, 4, 5) \) (encoding the mutable edge)

0 \{ \text{mutate}(L, U, V) \leftarrow \text{mutable}(L, U, V) \} 1 \} bounding the mutations

1 \{ \text{access}(R) \leftarrow \phi(R) \} \} bounding the requesters

- \( \alpha(R) \leftarrow e(l, 1, X) \land e(l, X, R) \land \sim (1 = X) \land \sim (X = R) \land \sim (1 = R) \) (atoms)
- \( \beta(R) \leftarrow e(l, 4, R), \sim (4 = R) \)

\( \phi(R) \leftarrow \alpha(R) \land \sim \beta(R) \) (encoding the formula)

\( e(l, 4, 5) \leftarrow \sim \text{mutate}(l, 4, 5) \} \) remove the edge if mutate
We run a series of experiments

- to compare $\text{MO-FEA}^{\text{MD}}$ (and $\text{MO-RES}^{\text{MD}}$) verifiers, and
- to study if they are viable in mid-sized organizations
Verdict: ASP-based $\text{MO-FEA}^\text{MD}$ verifier significantly outperforms the ME-based $\text{MO-FEA}^\text{MD}$ verifier.

Parameters:

- $\text{Rep.}: 20$;
- $k_u: 2$;
- $\#V_{\text{pat}^-}: 5$;
- $\delta: 2$.
Verdict: ASP-based MO-\text{FEA}^{MD} verifier can be employed in the mid-sized organizations such as Google and Microsoft (about 100 K users).

Parameters:
- $Rep.: 20$;
- $k_u: 1$;
- $\#atom^+: 3$;
- $\#atom^-: 3$;
- $\#V_{pat^+}: 5$;
- $\#V_{pat^-}: 5$;
- $\delta: 5$;
- $AvgDeg.G: 200$;
- $Prob.pat: 0.5$;
- $|L|: 4$;
Conclusion

Multiple Ownership
Instances

Categories

Annotations
(like, tag, share & comment)

Joint Declaration
(relationship articulation)

Collaborative Authoring
(Google Docs & wikis)

Approaches

Design Patterns

Interactive Negotiation
Future Work and Open Problems

Studying NoSQL implementation of design patterns

Extending the negotiation protocol to social computing

Conducting user study for the negotiation protocol

Examining if the negotiation protocol always terminates


The codes for ReBAC/MO availability verifiers have been published under MIT license.

https://github.com/pooyamehregan/ReBAC-Availability-Verifiers

The green manuscripts/papers are included in this thesis.
Supplementary Slides
Design Principle (1)

Every co-owner of a content shall have a say on its access control policy.

Design Principle (2)

Each component of a content with different set of co-owners must be protected by a separate access control policy.

Design Principle (3)

Every co-owner of an annotated content is a co-owner of its annotations.
Design Pattern Details

Design Decision (4)

Every co-owner $u$ of content $c$ has a preferred policy $p_{u,c}$ for that content.

- If $u$ is the principle co-owner then $p_{u,c}$ is explicitly specified.
- Otherwise, $u$ is an inherited co-owner of $c$. $c$ is an annotation for another content $c'$. We have $p_{u,c} = p_{u,c'}$

Design Decision (5)

Suppose $S_c = \{u_1, \ldots, u_k\}$ is the set of $c$’s co-owners, and $p_{u_1,c}, \ldots, p_{u_k,c}$ are their preferred policies for $c$. Then the access control policy of $c$ is $\bigwedge_{u \in S_c} p_{u,c}$.
Graph, Subgraph and Isomorphism

Definition

Given a finite, non-empty set $\mathcal{L}$ of relation identifiers, a graph $G$ is a pair $\langle V, \{R_l\}_{l \in \mathcal{L}} \rangle$, where $V$ is the set of vertices, and $R$ is an indexed family of binary relations over the vertex set, such that each $R_l \subseteq V \times V$ is a binary relation denoting the edges of type $l$.

Definition

- A graph $G_1$ is a subgraph of $G_2$ iff $V(G_1) \subseteq V(G_2)$, $L(G_1) \subseteq L(G_2)$, and $R_l(G_1) \subseteq R_l(G_2)$ for every $l \in L(G_1)$. In that case, we write $G_1 \subseteq G_2$.

- A bijection $f : V(G_1) \rightarrow V(G_2)$ is a graph isomorphism iff $L(G_1) = L(G_2)$, and for every $l \in L(G_1)$, for every $u, v \in V(G_1)$, we have $(u, v) \in R_l(G_1) \iff (f(u), f(v)) \in R_l(G_2)$. In such a case, we write $G_1 \cong_f G_2$. $G_1$ and $G_2$ are isomorphic, and we write $G_1 \cong G_2$, if the above bijection $f$ exists.
Birooted Subgraph Isomorphism

Definition

A birooted graph $BG$ is a triple $G(u,v)$, where $G$ is a graph, and $u, v \in V(G)$ are called the roots of $BG$. More specifically, $u$ and $v$ are called the owner root and requester root respectively.

Definition

• Suppose $BG_1 = G_1(u_1,v_1)$ and $BG_2 = G_2(u_2,v_2)$. $BG_1$ is a (birooted) subgraph of $BG_2$ iff $u_1 = u_2$, $v_1 = v_2$, and $G_1 \subseteq G_2$.

• A bijective function $f : V(G_1) \rightarrow V(G_2)$ is an isomorphism of $BG_1$ and $BG_2$ iff $f(u_1) = u_2$, $f(v_1) = v_2$, and $G_1 \simeq_f G_2$. We write $BG_1 \simeq BG_2$ iff such a bijection $f$ exists.

• We write $BG_1 \prec BG_2$ iff there exists a birooted subgraph $BG_3 \subseteq BG_2$ such that $BG_1 \simeq BG_3$. That is, $BG_1$ is isomorphic to some subgraph of $BG_2$. 
Policy Language

Syntax

\[ \phi ::= T | \alpha | \neg \phi | \phi \land \phi \]
\[ \alpha, \beta ::= \text{acc}(BG, u) \]

Semantics

- \( G, v \models \text{acc}(BG, u) \) iff \( BG \preceq G(u, v) \).
- \( G, v \models \neg \phi \) iff it is not the case that \( G, v \models \phi \).
- \( G, v \models \phi_1 \land \phi_2 \) iff \( G, v \models \phi_1 \) and \( G, v \models \phi_2 \).
Policy Negotiation Protocol

Settlement Policy Syntax

\[ \phi = (\alpha_1 \lor \alpha_2 \lor \ldots \lor \alpha_m) \land (\neg \beta_1 \land \neg \beta_2 \land \ldots \land \neg \beta_n) \]

Negotiation Protocol

1. Notify
   - formula \( \phi \) including \( pat^+(\phi, v) \) and \( pat^-(\phi, v) \) of all co-owners \( v \)'s
   - a boolean flag indicating if \( \phi \) satisfies the availability criterion \( \kappa_u \)

2. Consent/Revise
   - Consent: adopt formula \( \phi \) for the co-owned object \( o \)
   - Revise:
     - \( \kappa_u \) and/or
     - the graph pattern sets \( pat^+(\phi, u) \) and \( pat^-(\phi, u) \).

3. Verify
### Satisfiability (MO-SAT)

**Instance:** A positive integer \( k \), a graph \( G \), a policy \( \phi \) for which \( \text{anchors}(\phi) \subseteq V(G) \)

**Question:** Does there exist \( k \) distinct vertices \( v \in V(G) \) for which \( G, v \models \phi \)?

### Feasibility (MO-FEA)

**Instance:** A non-negative integer \( \delta \), a positive integer \( k \), a graph \( G \), and a policy \( \phi \) for which \( \text{anchors}(\phi) \subseteq V(G) \)

**Question:** Is there a graph \( G' \) for which \( L(G') = L(G) \), \( V(G') = V(G) \), and \( d(G, G') \leq \delta \), such that there exists \( k \) distinct vertices \( v \in V(G') \) for which \( G', v \models \phi \)?
Resiliency (MO-RES)

**Instance:** A non-negative integer $\delta$, a positive integer $k$, a graph $G$, and a policy $\phi$ for which $\text{anchors}(\phi) \subseteq V(G)$

**Question:** Is it the case that for every $G'$ such that $L(G') = L(G)$ and $V(G') = V(G)$ and $d(G, G') \leq \delta$, there exists $k$ distinct vertices $v \in V(G')$ for which $G', v \models \phi$?
Availability Criteria Computational Complexity

Theorem (1)

**MO-Sat** is in $\Delta^p_2$, and is both **NP-hard** and **coNP-hard**.

Theorem (2)

**MO-Fea** is $\Sigma^p_2$-complete.

Theorem (3)

**MO-Res** is $\Pi^p_2$-complete.
Let $G = \langle V, \{R_l\}_{l \in L}\rangle$ and $E$ be a set of triples of the form $l(u, v)$ where $l \in L(G)$ and $u, v \in V(G)$. We write $G \oplus E = \langle V, \{R'_l\}_{l \in L}\rangle$ to denote $E$-induced mutation of graph $G$. $R'_l$ in $\{R'_l\}_{l \in L}$ is defined as follows:

$$R'_l = \{(u, v) \in R_l \mid l(u, v) \notin E\} \cup \{(u, v) \notin R_l \mid l(u, v) \in E\}$$
## Reformulation of Availability Verifiers

### Feasibility with Mutation Descriptor ($\text{MO-FEA}^{\text{MD}}$)

**Instance:** a non-negative integer $\delta$, a positive integer $k$, a graph $G$, a mutation descriptor $E$ and a policy $\phi$ for which $\text{anchors}(\phi) \subseteq V(G)$

**Question:** Is there a graph $G' = G \oplus E'$ for which $E' \subseteq E$, and $d(G, G') \leq \delta$, such that there exists $k$ distinct $\nu \in V(G')$ for which $G', \nu \models \phi$?

### Resiliency with Mutation Descriptor ($\text{MO-RES}^{\text{MD}}$)

**Instance:** a non-negative integer $\delta$, a positive integer $k$, a graph $G$, a mutation descriptor $E$ and a policy $\phi$ for which $\text{anchors}(\phi) \subseteq V(G)$

**Question:** Is it the case that for every $G' = G \oplus E'$ such that $E' \subseteq E$ and $d(G, G') \leq \delta$, there exists $k$ distinct vertices $\nu \in V(G')$ for which $G', \nu \models \phi$?
Parameters

- **Rep.**: number of times an experiment is repeated for a set of parameters;
- **$k_u$**: number of requesters needed to satisfy policy;
- **$\#atom^+$**: number of positive atoms ($\alpha$) in policy ($\phi$);
- **$\#atom^-$**: number of negative atoms ($\beta$) in policy ($\phi$);
- **$\#V_{pat^+}$**: number of vertices in the graph patterns of positive atoms;
- **$\#V_{pat^-}$**: number of vertices in the graph patterns of negative atoms;
- **Prob.$_G$**: the edge probability by which social graph $G$ is generated;
- **Prob.$_{pat}$**: the edge probability by which graph patterns are generated;
- **AvgDeg.$_G$**: the average degree of vertices (for outgoing edges) that we require social graph $G$ to possess;
- **AvgDeg.$_{pat}$**: the average degree of vertices (for outgoing edges) that we require the patterns to possess;
- **$|\mathcal{L}|$**: number of edge labels used in graphs and graph patterns.
- **$\delta$**: the maximum distance between the original and mutated graphs.
Theorem

Let \( m \) and \( n \) be the number of vertices for the graph patterns and the social graph, respectively. Suppose we fix the average degree of the vertices. The space complexity of the adjacency preservation clauses in Reduce Algorithm is \( \Theta(m^2 n) \) and in the Torán reduction is \( \Theta(m^2 n^2) \).

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Out of Memory Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Torán reduction</td>
<td>10 out of 50</td>
</tr>
<tr>
<td>Reduce</td>
<td>0 out of 50</td>
</tr>
</tbody>
</table>

Table: Ratio of `OutOfMemoryError` in Reductions
Time-Outs in ME

(a) Model Enumerator

(b) Time-out Percentages

(c) No. of Positive Models Tried
Comparing ASP and ME Verifiers for Satisfiability

Figure: ASP-Based vs. ME Satisfiability

Parameters:
- $Rep.: 20$
- $k_u: 1$
- $\#V_{\text{pat}}^+: 5$
- $\#V_{\text{pat}}^- : 5$
- $\#\text{atom}^+: 3$
- $\#\text{atom}^- : 3$
- $\#V_{\text{pat}}^+$
- $\text{AvgDeg}_G : 200$
- $\text{Prob}_{\text{pat}} : 0.5$
- $|\mathcal{L}|$
No Time-outs in ASP-Based Verifiers

(a) Feasibility

(b) Resiliency
Proportionate (Denser) Graphs

Figure: Feasibility with Proportionate Degrees

Parameters:

- \( \text{Rep.}: 20; \)
- \( k_u: 1; \)
- \( \#V_{\text{pat}^-}: 5; \)
- \( \delta: 5. \)
- \( \#atom^+ : 3; \)
- \( \#atom^- : 3; \)
- \( \#V_{\text{pat}^+}; \)
- \( \text{Prob.}_G: 0.01; \)
- \( \text{Prob.}_{\text{pat}}: 0.5; \)
- \( |\mathcal{L}|: 4; \)
System Level Analyses of ASP Verifiers

(a) clasp (solver)  
(b) gringo (grounder)

Parameters:
- $Rep.$: 20;  
- $k_u$: 1;  
- $\#V_{pat}^-$: 5;  
- $\delta$: 5.  
- $\#atom^+$: 3;  
- $\#atom^-$: 3;  
- $Prob_G$: 0.01;  
- $Prob_{pat}$: 0.5;  
- $|\mathcal{L}|$: 4;