A Non-Monotonic Logic for Distributed Access Control

Marcos Cramer ¹  Diego Agustín Ambrossio ¹

¹University of Luxembourg

LAP 2016
23 - Sept - 2016
Introduction

Who has access to what resource?
Who has access to what resource?
Who has access to what resource?
Who has access to what resource?

Many says-based logics.
Who has access to what resource?

Many says-based logics.

“A says \( \varphi \)".
Who has access to what resource?

Many says-based logics.

“A says ϕ”.

“Principal A supports statement ϕ”.

**Introduction**

- **Who** has access to **what** resource?
- Many **says**-based logics.
  - “A **says** ϕ”.
  - “Principal A **supports** statement ϕ”.
- Access is granted iff it is logically entailed by the access control policy.
Consider the following example:

\[
T_A = \left\{ \begin{array}{l}
access(C, r) \land B\ says\ access(C, s) \Rightarrow access(C, o) \\
access(C, r)
\end{array} \right\}
\]

\[
T_B = \left\{ \begin{array}{l}
access(C, s) \\
\neg access(C, s) \land A\ says\ access(C, o) \Rightarrow access(C, o)
\end{array} \right\}
\]
Example

Consider the following example:

\[
T_A = \begin{cases}
  \text{access}(C, r) \land B \text{ says access}(C, s) \implies \text{access}(C, o) \\
  \text{access}(C, r)
\end{cases}
\]

\[
T_B = \begin{cases}
  \text{access}(C, s) \\
  \neg \text{access}(C, s) \land A \text{ says access}(C, o) \implies \text{access}(C, o)
\end{cases}
\]

- The says-statement is irrelevant.
Example

Consider the following example:

\[ T_A = \left\{ \begin{array}{l}
access(C, r) \land B \text{ says } access(C, s) \Rightarrow access(C, o) \\
access(C, r)
\end{array} \right\} \]

\[ T_B = \left\{ \begin{array}{l}
access(C, s) \\
\neg access(C, s) \land A \text{ says } access(C, o) \Rightarrow access(C, o)
\end{array} \right\} \]

- The says-statement is irrelevant.
- Communication Overload!
Monotonicity
Monotonicity

- New statements cannot lead to less access.
Monotonicity

- New statements cannot lead to less access.

Non-Monotonic!
Monotonicity

- New statements cannot lead to less access.

Non-Monotonic!
- Modeling Denial.
Introduction - Cont.

- **Monotonicity**
  - New statements cannot lead to less access.

- **Non-Monotonic!**
  - Modeling Denial.
  - $\neg B \text{ says } \neg \text{access}(C, r) \rightarrow \text{access}(C, r)$
The statements issued by a principal completely characterize what a principal supports.
The statements issued by a principal completely characterize what a principal supports.

Similar to the motivation for autoepistemic logic:
The statements issued by a principal completely characterize what a principal supports.

Similar to the motivation for autoepistemic logic:

“An agent’s knowledge base completely characterizes what the agent knows”
The statements issued by a principal completely characterize what a principal supports.

Similar to the motivation for autoepistemic logic:

“An agent’s knowledge base completely characterizes what the agent knows”

We use autoepistemic logic with well-founded semantics
We adapt autoepistemic logic to the multi-agent case.
We adapt autoepistemic logic to the multi-agent case.

Need to specify how the agents’ “knowledge” interacts.
We adapt autoepistemic logic to the multi-agent case.
Need to specify how the agents’ “knowledge” interacts.
Standard says-based logic:
We adapt autoepistemic logic to the multi-agent case.

Need to specify how the agents’ “knowledge” interacts.

Standard says-based logic:

- Mutual positive introspection:
  \[ k \text{ says } \varphi \Rightarrow j \text{ says } k \text{ says } \varphi \]
We adapt autoepistemic logic to the multi-agent case.

Need to specify how the agents’ “knowledge” interacts.

Standard *says*-based logic:

- Mutual positive introspection:
  \[ k \text{ says } \varphi \Rightarrow j \text{ says } k \text{ says } \varphi \]

- For denial, we need:
We adapt autoepistemic logic to the multi-agent case. 

Need to specify how the agents’ “knowledge” interacts. 

Standard says-based logic:

- Mutual positive introspection:

  \[ k \text{ says } \varphi \Rightarrow j \text{ says } k \text{ says } \varphi \]

- For denial, we need:

  Mutual negative introspection:

  \[ \neg k \text{ says } \varphi \Rightarrow j \text{ says } \neg k \text{ says } \varphi \]
Outline

1. Introduction
2. Syntax
3. Translation
4. Query Procedure
5. Communication Procedure
Introduction Syntax Translation Query Procedure Communication Procedure

Syntax

D-ACL Syntax

t denotes an arbitrary term and x an arbitrary variable:

\[
\phi ::= P(t, \ldots, t) \mid t = t \mid \neg \phi \mid \phi \land \phi \mid \forall x \phi \mid t \text{ says } \phi
\]
**D-ACL Syntax**

$t$ denotes an arbitrary term and $x$ an arbitrary variable:

$$
\varphi ::= P(t, \ldots, t) \mid t = t \mid \neg \varphi \mid \varphi \land \varphi \mid \forall x \varphi \mid t \text{ says } \varphi
$$

**Inductive Definition**

An D-ACL *inductive definition* $\Delta$ is a finite set of rules of the form $P(t_1, \ldots, t_n) \leftarrow \varphi$, where $P$ is an $n$-ary predicate symbol and $\varphi$ is a D-ACL formula.
Syntax

D-ACL Syntax

\( t \) denotes an arbitrary term and \( x \) an arbitrary variable:

\[
\varphi ::= P(t, \ldots, t) \mid t = t \mid \neg \varphi \mid \varphi \land \varphi \mid \forall x \varphi \mid t \text{ says } \varphi
\]

Inductive Definition

An D-ACL *inductive definition* \( \Delta \) is a finite set of rules of the form \( P(t_1, \ldots, t_n) \leftarrow \varphi \), where \( P \) is an \( n \)-ary predicate symbol and \( \varphi \) is a D-ACL formula.

D-ACL Theory

A D-ACL *theory* is a set that consists of D-ACL formulas and D-ACL inductive definitions.
Example

\[ T_A = \begin{cases} 
\{ p \leftarrow B \text{ says } p \\
p \leftarrow r \} \\
p \land s \land B \text{ says } q \Rightarrow q \\
r \lor \neg r \Rightarrow s \\
B \text{ says } r \lor \neg (B \text{ says } r) \Rightarrow q 
\end{cases} \]

\[ T_B = \begin{cases} 
p \\
C \text{ says } q \Rightarrow q \\
C \text{ says } r \Rightarrow r 
\end{cases} \]

\[ T_C = \begin{cases} 
\neg (B \text{ says } q) \Rightarrow q \\
B \text{ says } r \Rightarrow r 
\end{cases} \]
# Outline

1. Introduction
2. Syntax
3. Translation
4. Query Procedure
5. Communication Procedure
We define a decision procedure for D-ACL.
We define a decision procedure for D-ACL. It coincides with the well-founded semantics.
We define a decision procedure for D-ACL.

- It coincides with the well-founded semantics.
- It minimizes communication.
We define a decision procedure for D-ACL.
- It coincides with the well-founded semantics.
- It minimizes communication.

Implemented in IDP.
We define a decision procedure for D-ACL.
- It coincides with the well-founded semantics.
- It minimizes communication.

Implemented in IDP.

Well-founded semantics uses three truth-values: $t$, $f$ and $u$. 
We define a decision procedure for D-ACL.

- It coincides with the well-founded semantics.
- It minimizes communication.

Implemented in IDP.

Well-founded semantics uses three truth-values: \( t \), \( f \) and \( u \).

Three-valuedness arises only through the modal operator \( \text{says} \).

 três-chaves: 

\( p^+ \) para \( A \text{ says } \phi \) en contexto positivo;
\( p^- \) para \( A \text{ says } \phi \) en contexto negativo.

In inductive definitions, subformulas cannot be meaningfully termed positive or negative.
We define a decision procedure for D-ACL.

- It coincides with the well-founded semantics.
- It minimizes communication.

Implemented in IDP.

Well-founded semantics uses three truth-values: \( t, f \) and \( u \).

Three-valuedness arises only through the modal operator \( \text{says} \).

We use \( p^+_A \text{says} \phi \) for the upper bound for the truth value of \( A \text{ says } \phi \) and \( p^-_A \text{says} \phi \) for the lower bound.
We define a decision procedure for D-ACL.

- It coincides with the well-founded semantics.
- It minimizes communication.

Implemented in IDP.

Well-founded semantics uses three truth-values: \( t, f \) and \( u \).

Three-valuedness arises only through the modal operator \( \text{says} \).

We use \( p^+_A \text{ says } \varphi \) for the upper bound for the truth value of \( A \text{ says } \varphi \) and \( p^-_A \text{ says } \varphi \) for the lower bound.

\( p^+_A \text{ says } \varphi \) is used in positive contexts and \( p^-_A \text{ says } \varphi \) in negative contexts.
We define a decision procedure for D-ACL.

- It coincides with the well-founded semantics.
- It minimizes communication.

Implemented in IDP.

Well-founded semantics uses three truth-values: t, f and u.

Three-valuedness arises only through the modal operator says.

We use $p^+_{A \text{says } \varphi}$ for the upper bound for the truth value of $A \text{says } \varphi$ and $p^-_{A \text{says } \varphi}$ for the lower bound.

$p^+_{A \text{says } \varphi}$ is used in positive contexts and $p^-_{A \text{says } \varphi}$ in negative contexts.

In inductive definitions, subformulas cannot be meaningfully termed positive or negative.
For every modal atom $A$ says $\varphi$ occurring in the body of an inductive definition in theory $T$,

- replace $A$ says $\varphi$ by the propositional variable $w_{A\text{ says } \varphi}$
- add to $t(T)$ the two formulae $w_{A\text{ says } \varphi} \Rightarrow A$ says $\varphi$ and $A$ says $\varphi \Rightarrow w_{A\text{ says } \varphi}$.
Let $T$ be a D-ACL theory. $\tau(T)$ is constructed from $t(T)$ by performing the following replacements for every says-atom $A \text{ says } \varphi$ occurring in $t(T)$ that is not the sub-formula of another says-atom:

- Replace every positive occurrence of $A \text{ says } \varphi$ in $T$ by $p^+_{A \text{ says } \varphi}$.
- Replace every negative occurrence of $A \text{ says } \varphi$ in $T$ by $p^-_{A \text{ says } \varphi}$.
Example - Translation

\[ \mathcal{I}_A = \{ \begin{cases} \{ p \leftarrow w_{B\text{\_says}} p \\ p \leftarrow r \} \\ w_{B\text{\_says}} p \Rightarrow p_{B\text{\_says}}^+ \\ p_{B\text{\_says}}^- \Rightarrow w_{B\text{\_says}} p \\ p \land s \land p_{B\text{\_says}} q \Rightarrow q \\ r \lor \neg r \Rightarrow s \\ p_{B\text{\_says}} q \lor \neg p_{B\text{\_says}} q \Rightarrow q \end{cases} \} \]

\[ \mathcal{I}_B = \{ \begin{cases} p \\ p_{C\text{\_says}} q \Rightarrow q \\ p_{C\text{\_says}} r \Rightarrow r \end{cases} \} \]

\[ \mathcal{I}_C = \{ \begin{cases} \neg p_{B\text{\_says}} q \Rightarrow q \\ p_{B\text{\_says}} r \Rightarrow r \end{cases} \} \]
We work with partial structures: They are like standard first-order structures, but with missing information.

**Partial Model**

We say $S$ is a *partial model* for $\mathcal{T}$ if and only if there exists a total structure $S' \supseteq S$ such that $S' \models \mathcal{T}$.

**Minimal Inconsistent Set**

Let $\mathcal{T}$ a theory such that $S \not\models \mathcal{T}$. We define $\text{min}_\text{incons}_\text{set}(\mathcal{T}, S)$ as the set of minimal (under set inclusion) partial structure $S' \subseteq S$ such that the theory $\mathcal{T}$ has no models that expand $S$. 
Theories and Structures

Set $\mathcal{S}$

We define $\mathcal{S}$ to be the set containing every partial structure $S$ such that:

- For every symbol $\sigma \in \Sigma'$, if $\sigma \neq p^+_A \text{ says } \varphi$ or $\sigma \neq p^-_A \text{ says } \varphi$, then $(\sigma)^I = u$
- For every says-atom $A$ says $\varphi$ occurring in $T$:
  - $(p^+_A \text{ says } \varphi)^I \neq t$.
  - $(p^-_A \text{ says } \varphi)^I \neq f$.
- For no says-atom $A$ says $\varphi$, $(p^+_A \text{ says } \varphi)^I = f$ and $(p^-_A \text{ says } \varphi)^I = t$. 

M. Cramer, D. A. Ambrossio

A Non-Monotonic Logic for Distributed Access Control
<table>
<thead>
<tr>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>1 Introduction</td>
</tr>
<tr>
<td>2 Syntax</td>
</tr>
<tr>
<td>3 Translation</td>
</tr>
<tr>
<td>4 Query Procedure</td>
</tr>
<tr>
<td>5 Communication Procedure</td>
</tr>
</tbody>
</table>
Query Minimization Procedure.

Input: theory $\mathcal{T}$, D-ACL query $\alpha$

Output: set $\mathbb{L}$ of sets of modal atoms

1: $\mathbb{L} := \emptyset$
2: $\mathcal{T} := \tau(\mathcal{T} \cup \{\neg \alpha\})$
3: for each $S \in \mathbb{S}$ do
4: if $S$ is not a partial model of $\mathcal{T}$ then
5: pick a partial structure $S_{\text{min}}$ from min_incons_set($\mathcal{T}, S$)
6: $\mathbb{L} := \mathbb{L} \cup \{L_{S_{\text{min}}}\}$
7: return $\mathbb{L}$
Example - Query Procedure

Query: “q”

\[ T_A = \begin{cases} 
\{ p \leftarrow w_{B\text{-says}\cdot p} \\
  \{ p \leftarrow r \} \\
  w_{B\text{-says}\cdot p} \Rightarrow p^+_{B\text{-says}\cdot p} \\
  p^+_{B\text{-says}\cdot p} \Rightarrow w_{B\text{-says}\cdot p} \\
  p \land s \land p^+_{B\text{-says}\cdot q} \Rightarrow q \\
  r \lor \neg r \Rightarrow s \\
  p^+_{B\text{-says}\cdot r} \lor \neg p^+_{B\text{-says}\cdot r} \Rightarrow q 
\end{cases} \]
Example - Query Procedure

Query: “q”

\[ \mathcal{I}_A = \left\{ \begin{array}{l} \{ p \leftarrow w_{B \text{ says } p} \\ p \leftarrow r \} \\ w_{B \text{ says } p} \Rightarrow p_B^{+} \\ p_B^{+} \Rightarrow w_{B \text{ says } p} \\ p \land s \land p_B^{+} \Rightarrow q \\ r \lor \neg r \Rightarrow s \\ p_B^{+} \lor \neg p_B^{+} \Rightarrow q \end{array} \right\} \]

- \( p_B^{+} \text{ and } p_B^{+} : \{ B \text{ says } p; B \text{ says } q \} \)
Example - Query Procedure

Query: “q”

\[ T_A = \begin{cases} 
\{ p \leftarrow w_{B\text{-}says\cdot p} \\
p \leftarrow r \} \\
w_{B\text{-}says\cdot p} \Rightarrow p^+_{B\text{-}says\cdot p} \\
p^-_{B\text{-}says\cdot p} \Rightarrow w_{B\text{-}says\cdot p} \\
p \land s \land p^-_{B\text{-}says\cdot q} \Rightarrow q \\
r \lor \neg r \Rightarrow s \\
p^-_{B\text{-}says\cdot r} \lor \neg p^+_{B\text{-}says\cdot r} \Rightarrow q \end{cases} \]

- \( p^-_{B\text{-}says\cdot p} \) and \( p^-_{B\text{-}says\cdot q} : \{ B \text{ says } p; B \text{ says } q \} \)
Example - Query Procedure

Query: “q”

\[
\mathcal{I}_A = \begin{cases} 
\{p \leftarrow w_B \text{--says--} p \\
p \leftarrow r \}
\end{cases}
\]

- \( p_B \text{--says--} p \) and \( p_B \text{--says--} q \) : \{B says p; B says q\}
Example - Query Procedure

Query: “q“

\[ \mathcal{I}_A = \{ \{ p \leftarrow w_{B\text{-says }p} \}
\{ p \leftarrow r \}\}
\{ w_{B\text{-says }p} \Rightarrow p^+_B\text{-says }p \}
\{ p^-_B\text{-says }p \Rightarrow w_{B\text{-says }p} \}
\{ p \land s \land p^-_B\text{-says }q \Rightarrow q \}
\{ r \lor \neg r \Rightarrow s \}
\{ p^-_B\text{-says }r \lor \neg p^+_B\text{-says }r \Rightarrow q \}\]
Example - Query Procedure

Query: “q”

\[
\begin{align*}
\mathcal{I}_A &= \{ \\
&\{ p \leftarrow w_{B \text{ says } p} \\
&\{ p \leftarrow r \} \\
&w_{B \text{ says } p} \Rightarrow p^+_B \text{ says } p \\
&p^+_B \text{ says } p \Rightarrow w_{B \text{ says } p} \\
p \land s \land p^-_{B \text{ says } q} \Rightarrow q \\
r \lor \neg r \Rightarrow s \\
p^-_{B \text{ says } r} \lor \neg p^+_B \text{ says } r \Rightarrow q \\
\} \\
\end{align*}
\]

- \( p^-_{B \text{ says } p} \) and \( p^-_{B \text{ says } q} \) : \{ B \text{ says } p; B \text{ says } q \}
- \( p^-_{B \text{ says } r} \) : \{ B \text{ says } r \}
Example - Query Procedure

Query: “q”

\[ \{ p \leftarrow w_{B \text{ says } p} \}
\]

\[ \begin{align*}
&\{ p \leftarrow w_{B \text{ says } p} \\
&p \leftarrow r \\
&w_{B \text{ says } p} \Rightarrow p_{B \text{ says } p}^+ \\
&p_{B \text{ says } p}^- \Rightarrow w_{B \text{ says } p} \\
&p \land s \land p_{B \text{ says } q}^- \Rightarrow q \\
&r \lor \neg r \Rightarrow s \\
&p_{B \text{ says } r}^- \lor \neg p_{B \text{ says } r}^+ \Rightarrow q
\end{align*} \]

- \( p_{B \text{ says } p}^- \) and \( p_{B \text{ says } q}^- \) : \( \{ B \text{ says } p; B \text{ says } q \} \)
- \( p_{B \text{ says } r}^- \) : \( \{ B \text{ says } r \} \)
- \( \neg p_{B \text{ says } r}^+ \) : \( \{ \neg (B \text{ says } r) \} \)
Example - Query Procedure

Query: “q”

\[ T_A = \begin{cases} 
    \{ p \leftarrow w_{B \text{ says } p} \} \\
    p \leftarrow r \\
    w_{B \text{ says } p} \Rightarrow p_{B \text{ says } p}^+ \\
    p_{B \text{ says } p}^- \Rightarrow w_{B \text{ says } p} \\
    p \land s \land p_{B \text{ says } q}^- \Rightarrow q \\
    r \lor \neg r \Rightarrow s \\
    p_{B \text{ says } r}^- \lor \neg p_{B \text{ says } r}^+ \Rightarrow q 
\end{cases} \]

- \( p_{B \text{ says } p}^- \) and \( p_{B \text{ says } q}^- \): \( \{ B \text{ says } p; B \text{ says } q \} \)
- \( p_{B \text{ says } r}^- \): \( \{ B \text{ says } r \} \)
- \( \neg p_{B \text{ says } r}^+ \): \( \{ \neg (B \text{ says } r) \} \)
Example - Query Procedure

Query: “q”

\[ \mathcal{I}_A = \left\{ \begin{array}{l}
\{ p \leftarrow w_{B\text{-says\_p}} \\
p \leftarrow r \\
w_{B\text{-says\_p}} \Rightarrow p_{B\text{-says\_p}}^+ \\
p_{B\text{-says\_p}}^- \Rightarrow w_{B\text{-says\_p}} \\
p \land s \land p_{B\text{-says\_q}}^- \Rightarrow q \\
r \lor \lnot r \Rightarrow s \\
p_{B\text{-says\_r}}^- \lor \lnot p_{B\text{-says\_r}}^+ \Rightarrow q \end{array} \right\} \]

- \( p_{B\text{-says\_p}}^- \) and \( p_{B\text{-says\_q}}^- \): \{B says p; B says q\}
- \( p_{B\text{-says\_r}}^- \): \{B says r\}
- \( \lnot p_{B\text{-says\_r}}^+ \): \{\lnot (B says r)\}

\[ \mathbb{I}_L = \{\{B says p; B says q\}; \{B says r\}; \{\lnot (B says r)\}\} \]
## Outline

1. **Introduction**
2. **Syntax**
3. **Translation**
4. **Query Procedure**
5. **Communication Procedure**

---

M. Cramer, D. A. Ambrossio
(1) Apply Query Minimization Procedure.
(1) Apply Query Minimization Procedure.

(2) Build *query graph*: 

query vertices: ⟨A: α⟩: {? | t | f | u}.
says vertices: {A says ϕ}; {¬A says ϕ}; ... .
unlabelled edges: from query vertices to says vertices (that make the query true).
labelled edges: from says vertices to query vertices.
(1) Apply Query Minimization Procedure.

(2) Build *query graph*:
   - **query vertices**: \( \langle A : \alpha \rangle : \{? \mid t \mid f \mid u \} \).
   - **says vertices**: \( \{A \text{ says } \varphi\} \); \(\{\neg A \text{ says } \varphi\}\); . . . .
   - **unlabelled edges**: from query vertices to says vertices (that make the query true).
   - **labelled edges**: from says vertices to query vertices.
(1) Apply Query Minimization Procedure.

(2) Build *query graph*:

- **query vertices**: $\langle A : \alpha \rangle : \{? | t | f | u\}$.
- **says vertices**: $\{A \text{ says } \phi\}; \{\neg A \text{ says } \phi\}; \ldots$.
(1) Apply Query Minimization Procedure.

(2) Build *query graph*:

- **query vertices**: \( \langle A : \alpha \rangle : \{? | t | f | u \} \).
- **says vertices**: \( \{ A \ says \ \varphi \}; \ \{ \neg A \ says \ \varphi \}; \ldots \).
- **unlabelled edges**: from *query vertices* to *says vertices* (that make the query true).
(1) Apply Query Minimization Procedure.

(2) Build *query graph*:

- **query vertices**: \( \langle A : \alpha \rangle : \{? \mid t \mid f \mid u \} \).
- **says vertices**: \( \{A \ says \ \varphi\} ; \ \{\neg A \ says \ \varphi\} ; \ldots \).
- unlabelled edges: from **query vertices** to **says vertices** (that make the query true).
- labelled edges: from **says vertices** to **query vertices**.
We query principal $A$ about the truth value of $q$. 

Example Query Graph
Example Query Graph

We query principal $A$ about the truth value of $q$.

$\text{minimize\_query}(A, q)$
Example Query Graph

We query principal $A$ about the truth value of $q$.

\[
\text{minimize_query}(A, q)
\]

\[
\mathbb{L} = \{\{B \text{ says } p; B \text{ says } q\}; \{B \text{ says } r\}; \{\neg(B \text{ says } r)\}\}
\]
Example Query Graph

We query principal $A$ about the truth value of $q$.

\[
\text{minimize}_{\text{query}}(A, q)
\]

\[
\mathbb{L} = \{ \{ B \text{ says } p; B \text{ says } q \}; \{ B \text{ says } r \}; \{ \neg (B \text{ says } r) \} \}
\]

We start building the query graph:
We query principal $A$ about the truth value of $q$.

$$\text{minimize_query}(A, q)$$

$$\mathbb{L} = \left\{ \{ B \text{ says } p; B \text{ says } q \}; \{ B \text{ says } r \}; \{ \neg(B \text{ says } r) \} \right\}$$

We start building the query graph:
Example Query Graph - Complete

\[ \langle A : q \rangle : ? \]
Example Query Graph - Complete

\[
\langle A : q \rangle : ? \quad \rightarrow \quad \{ B \text{ says } p; B \text{ says } q \} \quad \rightarrow \quad \{ B \text{ says } r \} \quad \rightarrow \quad \{ \neg (B \text{ says } r) \}
\]
Example Query Graph - Complete

\[
\langle A : q \rangle : ? \quad \rightarrow \quad \{ B \text{ says } p; B \text{ says } q \} \quad \rightarrow \quad \{ B \text{ says } r \} \quad \rightarrow \quad \{ \neg ( B \text{ says } r) \} \quad \rightarrow \quad \langle B : p \rangle : ?
\]
Example Query Graph - Complete

\[ \langle A : q \rangle :? \]

\{ B says p; B says q \}

\{ B says r \}

\{ \neg (B says r) \}

\langle B : p \rangle :? \}

\{ \}
Example Query Graph - Complete

\[
\begin{align*}
\langle A \vdash q \rangle &? \rightarrow \\
\{\text{B says } p; \text{B says } q\} &\quad \{\text{B says } r\} &\quad \{\neg(\text{B says } r)\} \\
\langle B \vdash p \rangle &\rightarrow t \\
\{\} &
\end{align*}
\]
Example Query Graph - Complete

\[
\begin{align*}
\langle A : q \rangle : ? & \quad \{ B \text{ says } p; B \text{ says } q \} \\
\langle A : q \rangle : ? & \quad \{ B \text{ says } r \} \\
\{ B \text{ says } r \} & \quad \{ \neg (B \text{ says } r) \} \\
\langle B : p \rangle : t & \\
\langle B : q \rangle : ? & \\
\{ \} &
\end{align*}
\]
Example Query Graph - Complete

\[
\langle A : q \rangle : ?
\]

\[
\{ B \text{ says } p; B \text{ says } q \}
\]

\[
\{ B \text{ says } r \}
\]

\[
\{ \neg (B \text{ says } r) \}
\]

\[
\langle B : p \rangle : t
\]

\[
\{ \}
\]

\[
\langle B : q \rangle : ?
\]

\[
\{ C \text{ says } q \}
\]
Example Query Graph - Complete

\[
\begin{align*}
\langle A: q \rangle :? & \quad \{ B \text{ says } r \} \quad \{ \neg (B \text{ says } r) \} \\
\langle B : p \rangle : t & \quad \langle B : q \rangle :? \quad \{ B \text{ says } q \} \\
\{ \} & \quad \{ C \text{ says } q \} \quad \rightarrow \quad \langle C : q \rangle :?
\end{align*}
\]
Example Query Graph - Complete

\[
\begin{align*}
\langle A : q \rangle : ? & \\
\{ B says p; B says q \} & \\
\{ B says r \} & \\
\{ \neg (B says r) \} & \\
\langle B : p \rangle : t & \\
\{ C says q \} & \\
\langle C : q \rangle : ? & \\
\{ \neg (B says q) \} & \\
\end{align*}
\]
Example Query Graph - Complete

\[
\begin{align*}
\langle A : q \rangle : ? & \quad \{ B \text{ says } p; B \text{ says } q \} \\
\langle B : p \rangle : t & \quad \{ \} \\
\langle B : q \rangle : ? & \quad \{ C \text{ says } q \} \\
\langle C : q \rangle : ? & \quad \{ \neg (B \text{ says } q) \} \\
\langle B : q \rangle : ? & \quad \{ \neg (B \text{ says } r) \} \\
\end{align*}
\]
Example Query Graph - Complete

\[\langle A : q \rangle : ?\]

\{B says p; B says q\}

\{B says r\}

\{¬(B says r)\}

\langle B : p \rangle : t

\{\}

\langle B : q \rangle : ?

\{C says q\}

\langle C : q \rangle : ?

\{¬(B says q)\}

\langle B : q \rangle : u

loop!

(over negation)
Example Query Graph - Complete

\[ \langle A : q \rangle : ? \]

\begin{align*}
\{ B \text{ says } p ; B \text{ says } q \} & \quad \{ B \text{ says } r \} & \quad \{ \neg ( B \text{ says } r ) \} \\
\langle B : p \rangle : t & \quad \langle B : q \rangle : ? & \\
\{} & \quad \{ C \text{ says } q \} & \\
\} & \quad \neg & \\
\langle C : q \rangle : u & \quad \langle B : q \rangle : u & \text{loop!} \\
\} & \quad \{ \neg ( B \text{ says } q ) \} & \\
\} & \quad \neg & \\
\langle B : q \rangle : u & \text{(over negation)} & \\
\end{align*}
Example Query Graph - Complete

\[ \langle A : q \rangle : ? \]

\{ B says p; B says q \} \rightarrow \{ B says r \} \rightarrow \{ \neg (B says r) \}

\langle B : p \rangle : t \rightarrow \{ \} \rightarrow \{ C says q \} \rightarrow \neg \rightarrow \langle C : q \rangle : u \rightarrow \{ \neg (B says q) \} \rightarrow \rightarrow \langle B : q \rangle : u \rightarrow \text{loop! (over negation)}

M. Cramer, D. A. Ambrossio
Example Query Graph - Complete

\[ \langle A : q \rangle : ? \]

\{ \langle B : p \rangle : t \}
\rightarrow
\{ B says p; B says q \}
\rightarrow
\{ \langle B : q \rangle : u \}
\rightarrow
\{ C says q \}
\rightarrow
\{ \langle C : q \rangle : u \}
\rightarrow
\{ \neg(B says q) \}
\rightarrow
\{ \neg\neg(B says q) \}
\rightarrow
\{ \langle B : q \rangle : u \}

loop!

(over negation)
Example Query Graph - Complete

\begin{align*}
\langle A : q \rangle : ? &\quad \{ B \text{ says } r \} \\
\langle B : p \rangle : \text{t} &\quad \{ B \text{ says } q \} \\
\langle B : q \rangle : \text{u} &\quad \{ C \text{ says } q \} \\
\langle C : q \rangle : \text{u} &\quad \{ \neg (B \text{ says } q) \} \\
\langle B : q \rangle : \text{u} &\quad \{ \neg (B \text{ says } r) \} \\
\langle B : r \rangle : ? &\quad \{ B \text{ says } r \} \\
\end{align*}

M. Cramer, D. A. Ambrossio

A Non-Monotonic Logic for Distributed Access Control
Example Query Graph - Complete

\[
\langle A : q \rangle : ?
\]

\[
\{ B \text{ says } p ; B \text{ says } q \}
\]

\[
\{ B \text{ says } r \}
\]

\[
\{ \neg (B \text{ says } r) \}
\]

\[
\langle B : p \rangle : t
\]

\[
\langle B : q \rangle : u
\]

\[
\langle B : r \rangle : ?
\]

\[
\{ C \text{ says } q \}
\]

\[
\{ C \text{ says } r \}
\]

\[
\langle C : q \rangle : u
\]

\[
\langle C : r \rangle : ?
\]

\[
\{ \neg (B \text{ says } q) \}
\]

\[
\{ \neg (B \text{ says } r) \}
\]

\[
\langle B : q \rangle : u
\]

loop!

(over negation)
Example Query Graph - Complete

\[\langle A : q \rangle : ? \]

\{ \{B \text{ says } p; B \text{ says } q\}\}
\{ \{B \text{ says } r\}\}
\{ \{\neg (B \text{ says } r)\}\}

\langle B : p \rangle : t
\langle B : q \rangle : u
\langle B : r \rangle : ?

\{\}
\{C \text{ says } q\}
\{C \text{ says } r\}

\langle C : q \rangle : u
\langle C : r \rangle : ?

\{\neg (B \text{ says } q)\}
\{B \text{ says } r\}

\langle B : q \rangle : u

loop!
(over negation)
Example Query Graph - Complete

\[ \langle A : q \rangle : ? \]

\{ B says p; B says q \} \rightarrow \{ B says r \} \rightarrow \{ \neg (B says r) \}

\langle B : p \rangle : t \rightarrow \{ \} \rightarrow \{ C says q \} \rightarrow \{ \neg (B says q) \} \rightarrow \{ B says r \} \rightarrow \langle B : r \rangle : ?

\langle B : q \rangle : u \rightarrow \{ C says r \} \rightarrow \langle C : r \rangle : ?

\langle B : r \rangle : ?

loop!

(over negation)
Example Query Graph - Complete

\[\langle A : q \rangle : ?\]

\{B says p; B says q\}

\{B says r\}

\{\neg (B says r)\}

\langle B : p \rangle : t\n
\langle B : q \rangle : u

\langle B : r \rangle : ?

\langle C : q \rangle : u

\langle C : r \rangle : ?

\langle C : q \rangle : u

\langle C : r \rangle : ?

\langle B : q \rangle : u

\langle B : r \rangle : f

loop! (over negation)
Example Query Graph - Complete

\[ \langle A : q \rangle : ? \]
\[ \{ B \text{ says } p; B \text{ says } q \} \]
\[ \langle B : p \rangle : t \]
\[ \{ C \text{ says } q \} \]
\[ \langle C : q \rangle : u \]
\[ \{ \neg (B \text{ says } q) \} \]
\[ \langle B : q \rangle : f \]

\[ \langle B : r \rangle : ? \]
\[ \{ B \text{ says } r \} \]
\[ \langle C : r \rangle : f \]
\[ \{ B \text{ says } r \} \]

\[ \langle B : r \rangle : f \]

loop! (over negation)
Example Query Graph - Complete

\[
\begin{align*}
\langle B : p \rangle &: t \\
\langle B : q \rangle &: u \\
\langle C : q \rangle &: u \\
\langle B : q \rangle &: u
\end{align*}
\]

\[
\begin{align*}
\langle B : p \rangle &: t \\
\langle B : q \rangle &: u \\
\langle C : q \rangle &: u \\
\langle B : q \rangle &: u
\end{align*}
\]

\[
\begin{align*}
\langle B : r \rangle &: f \\
\langle B : r \rangle &: f \\
\langle B : r \rangle &: f
\end{align*}
\]

\[
\begin{align*}
\langle B : r \rangle &: f \\
\langle B : r \rangle &: f \\
\langle B : r \rangle &: f
\end{align*}
\]

\[
\begin{align*}
\langle A : q \rangle &: ? \\
\{ B \text{ says } p; B \text{ says } q \} \\
\{ B \text{ says } q \} \\
\{ C \text{ says } q \} \\
\{ C \text{ says } q \}
\end{align*}
\]

\[
\begin{align*}
\{ B \text{ says } r \} \\
\{ B \text{ says } r \} \\
\{ C \text{ says } r \} \\
\{ C \text{ says } r \}
\end{align*}
\]

\[
\begin{align*}
\{ \neg (B \text{ says } r) \} \\
\{ \neg (B \text{ says } q) \} \\
\{ \neg (B \text{ says } q) \} \\
\{ B \text{ says } r \} \\
\{ B \text{ says } r \}
\end{align*}
\]

\[
\begin{align*}
\{ \neg (B \text{ says } r) \} \\
\{ \neg (B \text{ says } q) \} \\
\{ B \text{ says } r \} \\
\{ B \text{ says } r \}
\end{align*}
\]

\[
\begin{align*}
\{ \neg (B \text{ says } r) \} \\
\{ \neg (B \text{ says } q) \} \\
\{ B \text{ says } r \} \\
\{ B \text{ says } r \}
\end{align*}
\]

\[
\begin{align*}
\{ \neg (B \text{ says } r) \} \\
\{ \neg (B \text{ says } q) \} \\
\{ B \text{ says } r \} \\
\{ B \text{ says } r \}
\end{align*}
\]
Example Query Graph - Complete

\[\langle A : q \rangle : ? \]

\{B says p; B says q\} \rightarrow \{B says r\} \rightarrow \{\neg(B says r)\}

\langle B : p \rangle : t \rightarrow \{\}\rightarrow \langle B : q \rangle : u \rightarrow \{C says q\} \rightarrow \langle C : q \rangle : u \rightarrow \{\neg(B says q)\} \rightarrow \langle B : q \rangle : u \rightarrow \text{loop!}

\langle B : r \rangle : f \rightarrow \{\}\rightarrow \langle B : r \rangle : f \rightarrow \{C says r\} \rightarrow \langle C : r \rangle : f \rightarrow \{B says r\} \rightarrow \langle B : r \rangle : f \rightarrow \text{loop!}
Example Query Graph - Complete

\[\langle A : q \rangle : ?\]

\{B says p; B says q\}

\(\langle B : p \rangle : t\)

\(\langle B : q \rangle : u\)

\{C says q\}

\(\langle C : q \rangle : u\)

\{\neg (B says q)\}

\(\langle B : q \rangle : u\)

loop!

(over negation)

\{B says r\}

\(\langle B : r \rangle : f\)

\(\langle C : r \rangle : f\)

\{B says r\}

\(\langle B : r \rangle : f\)

loop!
Example Query Graph - Complete

\[\langle A : q \rangle : ? \]

\[\{ B \text{ says } p; B \text{ says } q \}\]
\[\langle B : p \rangle : true \]
\[\} \]
\[\langle B : q \rangle : true \]
\[\} \]
\[\langle C : q \rangle : false \]
\[\} \]
\[\{ \neg (B \text{ says } q) \}\]
\[\langle B : q \rangle : true \]
\[\text{loop!} \]

\[\{ B \text{ says } r \}\]
\[\langle B : r \rangle : true \]
\[\} \]
\[\langle C : r \rangle : true \]
\[\} \]
\[\{ B \text{ says } r \}\]
\[\langle B : r \rangle : true \]
\[\text{loop!} \]

\[\{ \neg (B \text{ says } r) \}\]
\[\langle B : r \rangle : true \]
\[\} \]
\[\langle C : r \rangle : true \]
\[\} \]
\[\{ B \text{ says } r \}\]
\[\langle B : r \rangle : true \]
\[\text{loop!} \]

\[\{ \text{over negation} \}\]
Example Query Graph - Complete

\[ \langle A : q \rangle : t \]
\[ \langle A : q \rangle : t \]
\[ \{ B \text{ says } p; B \text{ says } q \} \]
\[ \langle B : p \rangle : t \]
\[ \{ C \text{ says } q \} \]
\[ \langle C : q \rangle : u \]
\[ \{ \neg (B \text{ says } q) \} \]
\[ \langle B : q \rangle : u \]

\[ \langle B : q \rangle : u \]
loop! (over negation)

\[ \langle B : q \rangle : u \]
\[ \langle B : q \rangle : u \]
\[ \{ B \text{ says } r \} \]
\[ \langle B : r \rangle : f \]
\[ \{ \neg (B \text{ says } r) \} \]
\[ \langle B : r \rangle : f \]

\[ \{ C \text{ says } r \} \]
\[ \langle C : r \rangle : f \]
\[ \{ B \text{ says } r \} \]
\[ \langle B : r \rangle : f \]
\[ \{ \neg (B \text{ says } r) \} \]
\[ \langle B : r \rangle : f \]

\[ \{ C \text{ says } q \} \]
\[ \langle C : q \rangle : u \]
\[ \{ \neg (B \text{ says } q) \} \]
\[ \langle B : q \rangle : u \]
loop!

\[ \{ B \text{ says } r \} \]
\[ \langle B : r \rangle : f \]
\[ \{ \neg (B \text{ says } r) \} \]
\[ \langle B : r \rangle : f \]
\[ \{ C \text{ says } q \} \]
\[ \langle C : q \rangle : u \]
\[ \{ \neg (B \text{ says } q) \} \]
\[ \langle B : q \rangle : u \]
loop! (over negation)
Thanks!

Questions?