JustAct: Actions Universally Justified by Partial, Dynamic Policies

Christopher Esterhuyse, Tim Müller, Thomas van Binsbergen

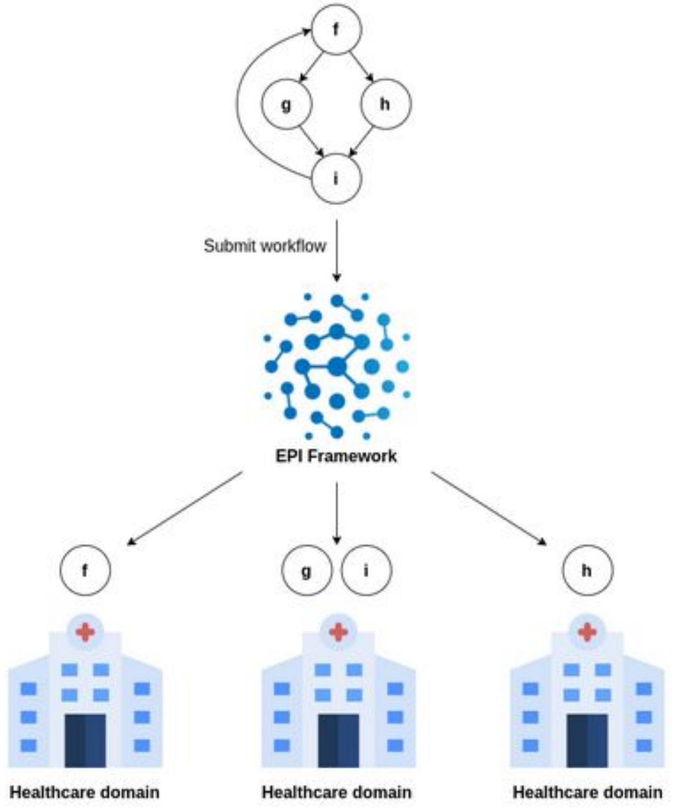
Amsterdam Economic Board
SURF
amsix
Universiteit van Amsterdam
deXes
European Union
European Regional Development Fund
Investing in your future
Provincie Noord-Holland

Data Exchange Systems

Data exchange systems share and process data across organisational boundaries. They are

- inherently distributed, and
- subject to complex requirements.

Example: the Brane system, orchestrating the sharing and processing of medical data.



The Role of Policy

Systems are subject to complex requirements like…

- The General Data Privacy Regulation (GDPR)
- Consortium-level agreements
- Resource-level sharing agreements

Policies capture these requirements. This affords…

- Static analysis (e.g., model-checking)
- Dynamic enforcement (e.g., monitoring)

Example Policy

Noteworthy:

- Built from expressions and rules
  ⟶ modular ⟶ (de)composable
- Models domain-specific concepts
  ⟶ complex and specific
  but not ambiguous

```
/// Stelt dat áls de `totale-commons-waarde-aangeboden` is gegeven,
/// deze waarde direct herleidbaar moet zijn tot de waarde van de
/// aangeboden producten van die Deelnemer.
///
/// Afsprakenstelsel:
/// > Deze Eurowaarde moet direct herleidbaar zijn tot
///   de commerciële waarde van de als Commons
/// > aangeboden producten, diensten en/of data(gebruik)
///   in de Producten en Diensten Catalogus.
///
/// De aanbieder kan aansprakelijk worden gehouden door
/// elke (andere) Deelnemer.

Duty totale-commons-waarde-aangeboden-herleidbaar-van-commerciele-waarde
  Holder aanbieder
  Claimant deelnemer
  Related to totale-commons-waarde-aangeboden

  // De Duty geldt voor elke aanbieder met aangeboden waarde.
  Derived from (Foreach totale-commons-waarde-aangeboden, deelnemer :
    totale-commons-waarde-aangeboden-herleidbaar-van-commerciele-waarde(
      totale-commons-waarde-aangeboden.aanbieder,
      deelnemer,
      totale-commons-waarde-aangeboden
    ) When (totale-commons-waarde-aangeboden.waarde > 0
            && totale-commons-waarde-aangeboden.aanbieder != deelnemer))

  // De Duty is geschonden als er niet genoemt is dat de
  // waarde herleidbaar is tot deze Deelnemer's aangeboden producten.
  Violated when (Exists aanbod :
    aanbod-als-commons(aanbod)
    && aanbod.aanbieder == aanbieder
    && Not(totale-waarde-herleidbaar-tot-aanbod(
                      totale-commons-waarde-aangeboden, aanbod))).
```

Example Policy

Noteworthy:

- Built from expressions and rules
  ⟶ modular ⟶ (de)composable
- Models domain-specific concepts
  ⟶ complex and specific
  but not ambiguous
- There are several useful policy langs.

```
% Statement 's1' by 'consortium' (contents of agreement at time 1)
owns(administrator, Data) :- ctl-accesses(Accessor, Data).
error :- ctl-accesses(Accessor, Data), owns(Owner, Data),
       not ctl-authorises(Owner, Accessor, Data).
```

Datalog$^{\neg}$

The Demands of Policy

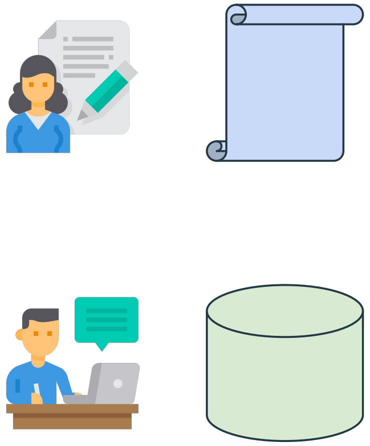
Requirements can impose significant constraints on the runtime system (e.g., data privacy regulations):

1. Policy must determine system behavior
2. Policies may change arbitrarily at runtime
3. Policies themselves may be sensitive

Each data exchange system strikes its own balance.

AMdEX project: develop generic tools for building specialised data exchange systems.

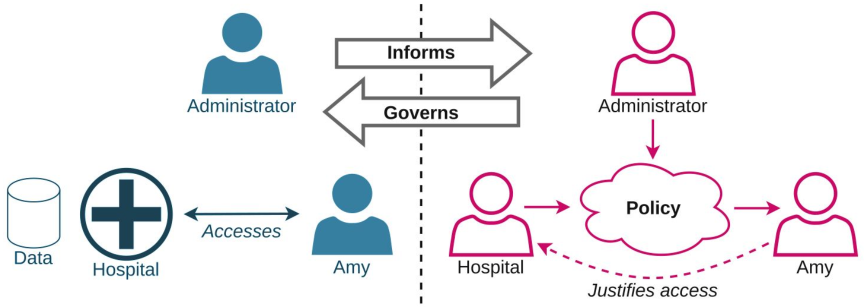
AMdEX



The Contribution

Today, we present the **JustAct** framework, which

- **Defines** the relation between policy and agents' actions and communications, but
- **Leaves undefined** the policy language and to the runtime implementation.

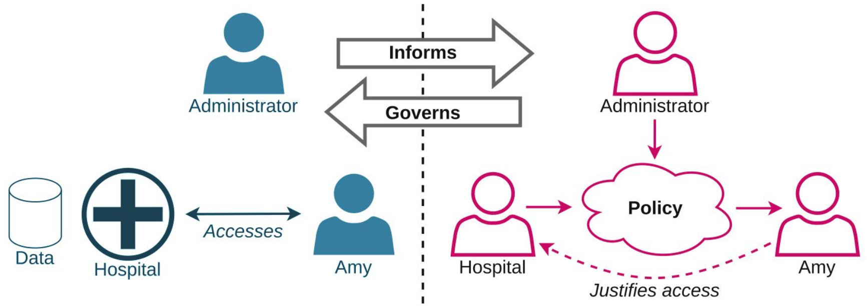


The Idea

The system consists of agents which are autonomous: each independently decides …

- Which policies they create[1] and share
- Which actions they take[2]

…

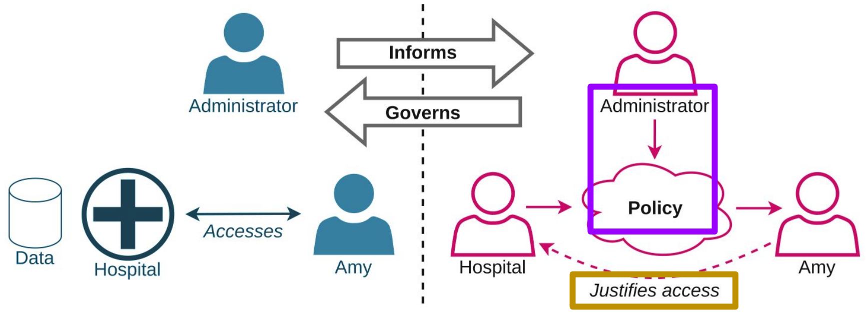


The Idea

The system consists of agents which are autonomous: each independently decides …

- Which policies they create[1] and share
- Which actions they take[2]

[1] but not every agent can **create** every policy
[2] but each action must be **justified** with a policy



Using the Framework

Precisely, our framework is…

1. A relational **abstraction** over the system
2. **Requirements** “realistically” satisfiable
3. **Guarantees** following from the requirements

Using the framework means adopting the abstraction such that the requirements are satisfied.

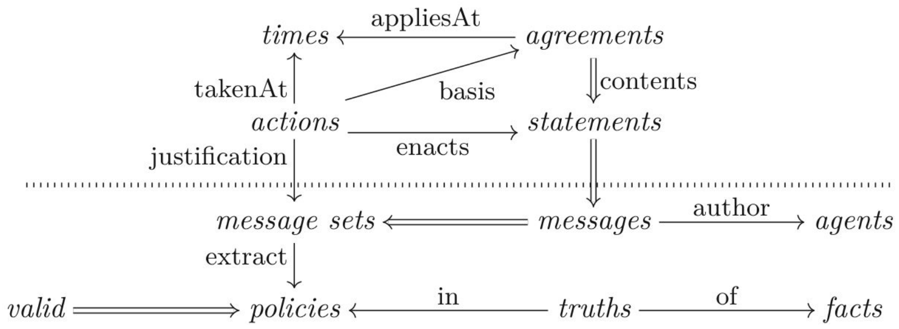
Using the Framework

Precisely, our framework is…

1. A relational **abstraction** over the system
2. **Requirements** “realistically” satisfiable
3. **Guarantees** following from the requirements

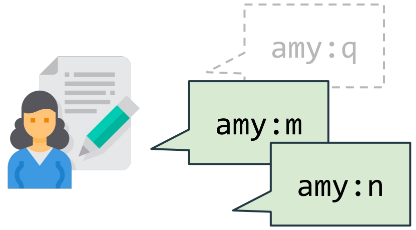
Using the framework means adopting the abstraction such that the requirements are satisfied.

Here is the abstraction, a relational ontology:

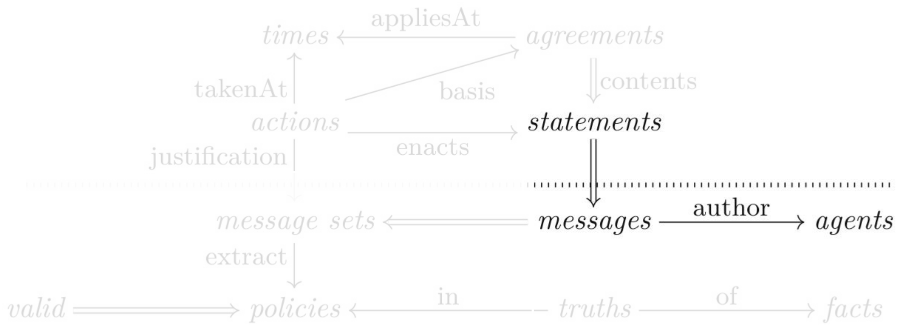


$$\begin{tikzcd}
 & \mathit{times} & \mathit{agreements} \arrow[l, "\text{appliesAt}"'] \arrow[d, Rightarrow, "\text{contents}"] & \\
 & \mathit{actions} \arrow[u, "\text{takenAt}"] \arrow[ur, "\text{basis}"'] \arrow[r, "\text{enacts}"'] \arrow[d, "\text{justification}"'] & \mathit{statements} \arrow[d, Rightarrow] & \\
 & \mathit{message\ sets} \arrow[d, "\text{extract}"'] & \mathit{messages} \arrow[l, Rightarrow] \arrow[r, "\text{author}"] & \mathit{agents} \\
\mathit{valid} \arrow[r, Rightarrow] & \mathit{policies} & \mathit{truths} \arrow[l, "\text{in}"'] \arrow[r, "\text{of}"] & \mathit{facts}
\end{tikzcd}$$

Concepts: 1/7

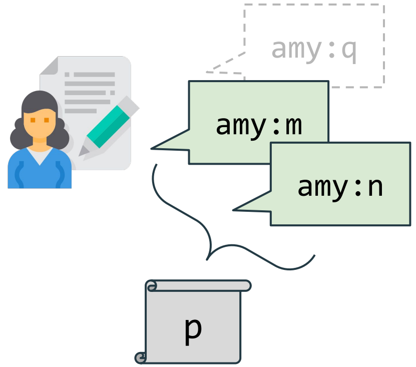


- Agents incrementally unfold the subset of stated messages at runtime.
- The author of each message is evident.



$$\begin{tikzcd}
& \textit{times} & \textit{agreements} \arrow[l, "\text{appliesAt}"'] \arrow[d, Rightarrow, "\text{contents}"] & \\
& \textit{actions} \arrow[u, "\text{takenAt}"] \arrow[ur, "\text{basis}"'] \arrow[r, "\text{enacts}"'] \arrow[d, "\text{justification}"'] & \textit{statements} \arrow[d, Rightarrow] & \\
& \textit{message sets} \arrow[d, "\text{extract}"'] & \textit{messages} \arrow[l, Rightarrow] \arrow[r, "\text{author}"] & \textit{agents} \\
\textit{valid} \arrow[r, Rightarrow] & \textit{policies} & \textit{truths} \arrow[l, "\text{in}"'] \arrow[r, "\text{of}"] & \textit{facts}
\end{tikzcd}$$

Concepts: 2/7



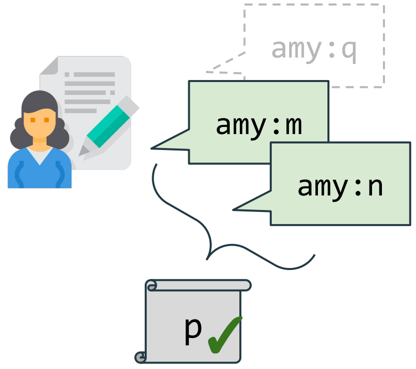
- Messages carry policies.
  We suggest: extract sensitive to author

- Each message set carries one policy
  We suggest: composed message-policies

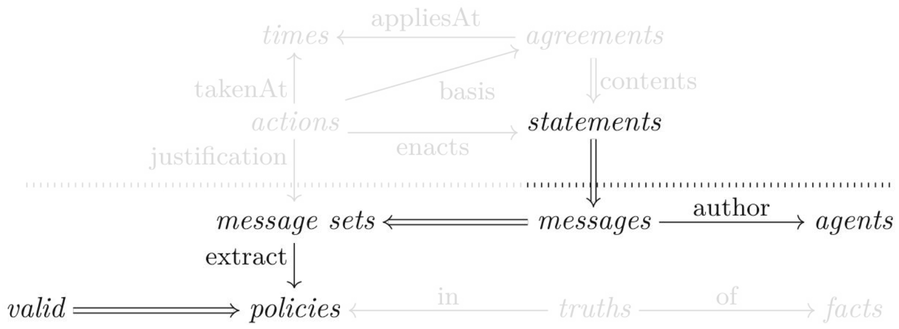
- → Available policies grow with statements.

$$\begin{tikzcd}
& \mathit{times} & \mathit{agreements} \arrow[l, "\text{appliesAt}"'] \arrow[d, Rightarrow, "\text{contents}"] & \\
& \mathit{actions} \arrow[u, "\text{takenAt}"] \arrow[ur, "\text{basis}"'] \arrow[r, "\text{enacts}"'] \arrow[d, "\text{justification}"'] & \mathit{statements} \arrow[d, Rightarrow] & \\
& \mathit{message\ sets} \arrow[d, "\text{extract}"'] & \mathit{messages} \arrow[l, Leftarrow] \arrow[r, "\text{author}"] & \mathit{agents} \\
\mathit{valid} \arrow[r, Rightarrow] & \mathit{policies} & \mathit{truths} \arrow[l, "\text{in}"'] \arrow[r, "\text{of}"] & \mathit{facts}
\end{tikzcd}$$

Concepts: 3/7

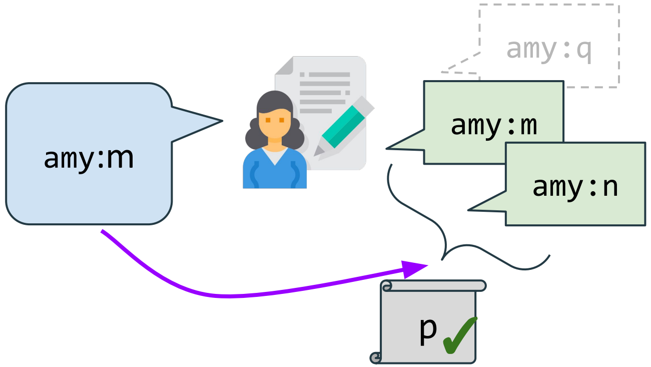


- Not every policy is valid (“useful”).
- We suggest: wrong message by wrong author reflected as invalidity



$$\begin{tikzcd}
& times & agreements \arrow[l, "appliesAt"'] \arrow[d, Rightarrow, "contents"] & \\
& actions \arrow[u, "takenAt"] \arrow[ur, "basis"'] \arrow[r, "enacts"'] \arrow[d, "justification"] & statements \arrow[d, Rightarrow] & \\
& message\ sets \arrow[d, "extract"] & messages \arrow[l, Rightarrow] \arrow[r, "author"] & agents \\
valid \arrow[r, Rightarrow] & policies & truths \arrow[l, "in"'] \arrow[r, "of"] & facts
\end{tikzcd}$$

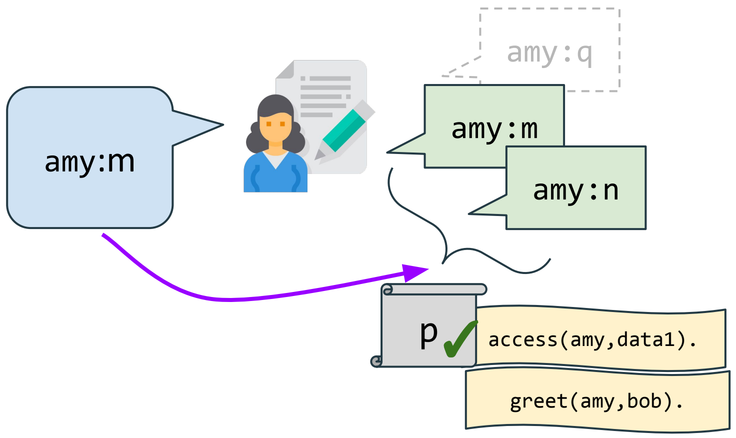
Concepts: 4/7



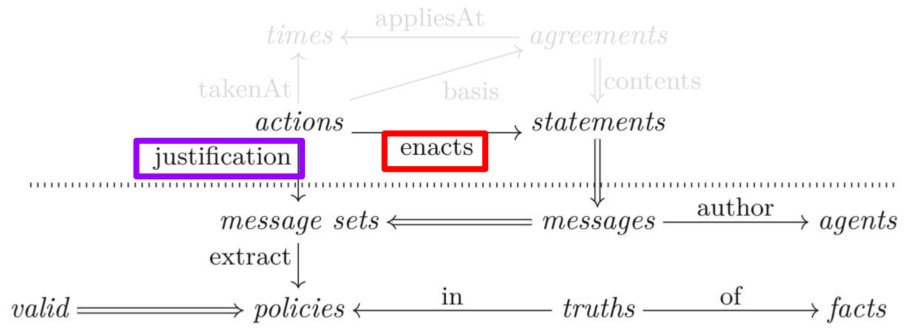
- The set of actions grows at runtime.
- The author of each action is evident: the author of the statement it enacts.
- Each action is justified by a message set. Its extracted policy must be valid. ⟶ Valid policies determine justified actions.

$$\begin{tikzcd}
& \mathit{times} & \mathit{agreements} \arrow[l, "\text{appliesAt}"'] \arrow[d, Rightarrow, "\text{contents}"] & \\
& \mathit{actions} \arrow[u, "\text{takenAt}"] \arrow[ur, "\text{basis}"'] \arrow[r, "\text{enacts}"'] \arrow[d, "\text{justification}"'] & \mathit{statements} \arrow[d, Rightarrow] & \\
& \mathit{message\ sets} \arrow[d, "\text{extract}"'] & \mathit{messages} \arrow[l, Rightarrow] \arrow[r, "\text{author}"] & \mathit{agents} \\
\mathit{valid} \arrow[r, Rightarrow] & \mathit{policies} & \mathit{truths} \arrow[l, "\text{in}"'] \arrow[r, "\text{of}"] & \mathit{facts}
\end{tikzcd}$$

Concepts: 5/7



- Each policy is a model of the domain (e.g., policies = deterministic logic programs).
- → All agents agree on a given action's
  - effects
  - justifiaction and validity



$$\begin{tikzcd}
& \textit{times} & \textit{agreements} \arrow[l, "\text{appliesAt}"'] \arrow[d, Rightarrow, "\text{contents}"] & \\
& \textit{actions} \arrow[u, "\text{takenAt}"] \arrow[ur, "\text{basis}"] \arrow[r, "\boxed{\text{enacts}}"'] \arrow[d, "\boxed{\text{justification}}"'] & \textit{statements} \arrow[d, Rightarrow] & \\
& \textit{message sets} \arrow[d, "\text{extract}"'] & \textit{messages} \arrow[l, Rightarrow] \arrow[r, "\text{author}"] & \textit{agents} \\
\textit{valid} \arrow[r, Rightarrow] & \textit{policies} & \textit{truths} \arrow[l, "\text{in}"'] \arrow[r, "\text{of}"] & \textit{facts}
\end{tikzcd}$$

A Usage Example

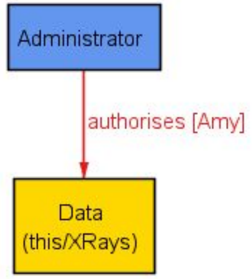
Model of domain relations in extract($\{s_1\}$)

A priori agreement to empower the admin.

```
% Statement 's1' by 'consortium' (contents of agreement at time 1)
owns(administrator, Data) :- ctl-accesses(Accessor, Data).
error :- ctl-accesses(Accessor, Data), owns(Owner, Data),
      not ctl-authorises(Owner, Accessor, Data).
```

A Usage Example

Model of domain relations in extract($\{s_1,s_2\}$)



A priori agreement to empower the admin.

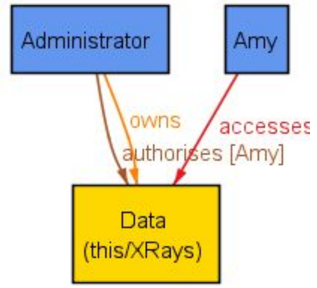
```
% Statement 's1' by 'consortium' (contents of agreement at time 1)
owns(administrator, Data) :- ctl-accesses(Accessor, Data).
error :- ctl-accesses(Accessor, Data), owns(Owner, Data),
        not ctl-authorises(Owner, Accessor, Data).
```

Administrator authorises a particular data-access.

```
% Statement 's2' by 'administrator'
ctl-authorises(administrator, amy, x-rays).
```

A Usage Example

Model of domain relations in extract($\{s_1, s_2, s_3\}$)



A priori agreement to empower the admin.

```
% Statement 's1' by 'consortium' (contents of agreement at time 1)
owns(administrator, Data) :- ctl-accesses(Accessor, Data).
error :- ctl-accesses(Accessor, Data), owns(Owner, Data),
       not ctl-authorises(Owner, Accessor, Data).
```

Administrator authorises a particular data-access.

```
% Statement 's2' by 'administrator'
ctl-authorises(administrator, amy, x-rays).
```

Amy accesses X-rays

```
% Statement 's3' authored by 'amy'
ctl-accesses(amy, x-rays).
```

A Usage Example

A priori agreement to empower the admin.

```
% Statement 's1' by 'consortium' (contents of agreement at time 1)
owns(administrator, Data) :- ctl-accesses(Accessor, Data).
error :- ctl-accesses(Accessor, Data), owns(Owner, Data),
      not ctl-authorises(Owner, Accessor, Data).
```

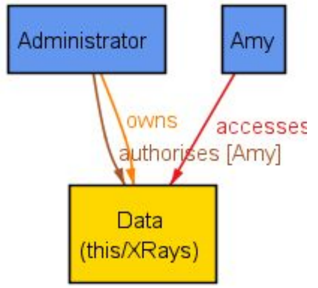
Administrator authorises a particular data-access.

```
% Statement 's2' by 'administrator'
ctl-authorises(administrator, amy, x-rays).
```

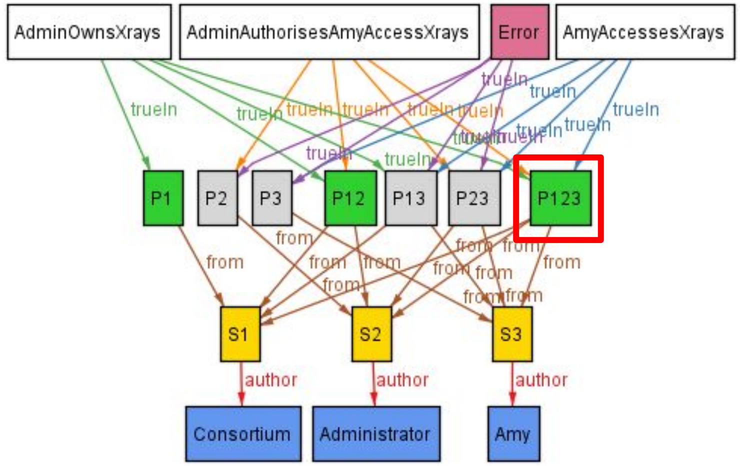
Amy accesses X-rays

```
% Statement 's3' authored by 'amy'
ctl-accesses(amy, x-rays).
```

Model of domain relations in extract($\{s_1,s_2,s_3\}$)

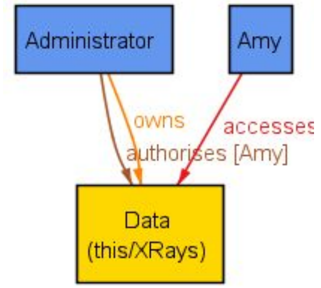


Model of framework-level relations



A Usage Example

Model of domain relations in extract($\{s_1,s_2,s_3\}$)



A priori agreement to empower the admin.

```
% Statement 's1' by 'consortium' (contents of agreement at time 1)
owns(administrator, Data) :- ctl-accesses(Accessor, Data).
error :- ctl-accesses(Accessor, Data), owns(Owner, Data),
      not ctl-authorises(Owner, Accessor, Data).
```

Administrator authorises a particular data-access.

```
% Statement 's2' by 'administrator'
ctl-authorises(administrator, amy, x-rays).
```

Amy accesses X-rays

```
% Statement 's3' authored by 'amy'
ctl-accesses(amy, x-rays).
```

← Amy can enact this, as justified by $\{s_1,s_2,s_3\}$

All observers agree:

- That it is permitted
- On the effects

A Usage Example

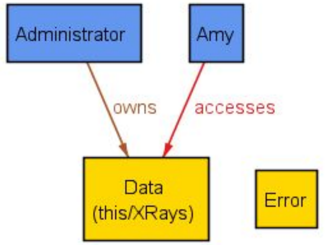
A priori agreement to empower the admin.

```
% Statement 's1' by 'consortium' (contents of agreement at time 1)
owns(administrator, Data) :- ctl-accesses(Accessor, Data).
error :- ctl-accesses(Accessor, Data), owns(Owner, Data),
         not ctl-authorises(Owner, Accessor, Data).
```

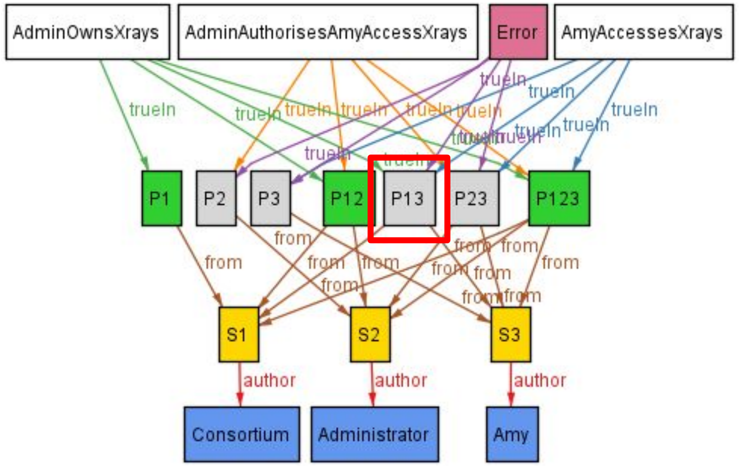
Amy accesses X-rays

```
% Statement 's3' authored by 'amy'
ctl-accesses(amy, x-rays).
```

Model of domain relations in extract($\{s_1,s_3\}$)



Model of framework-level relations



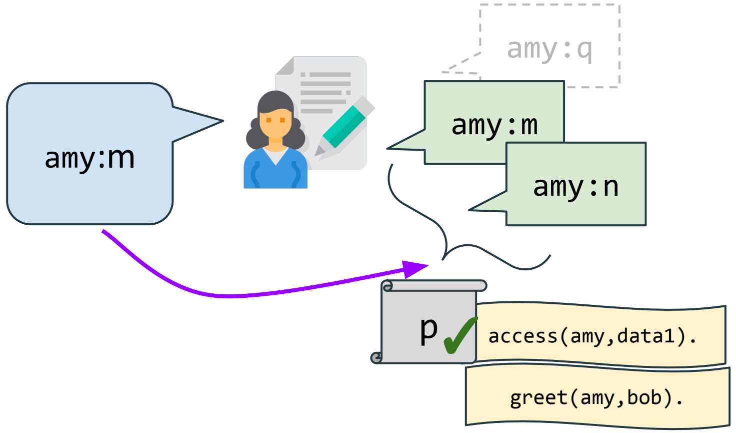
A Usage Example

Amy accesses X-rays

```
% Statement 's3' authored by 'amy'
ctl-accesses(amy, x-rays).
```

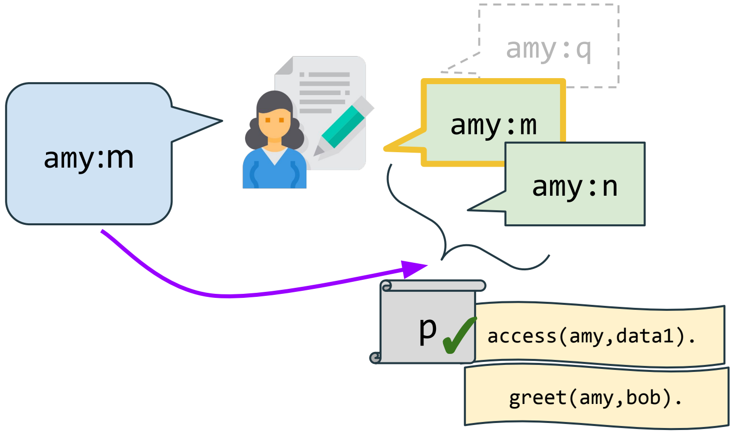
← Amy can enact this as justified by $\{s_3\}$?

Concepts: 5/7



$$\begin{tikzcd}
 & times & agreements \arrow[l, "appliesAt"'] \arrow[d, Rightarrow, "contents"] & \\
 & actions \arrow[u, "takenAt"] \arrow[ur, "basis"'] \arrow[r, "enacts"'] \arrow[d, "justification"'] & statements \arrow[d, Rightarrow] & \\
 & message\ sets \arrow[d, "extract"'] & messages \arrow[l, Rightarrow] \arrow[r, "author"] & agents \\
valid \arrow[r, Rightarrow] & policies & truths \arrow[l, "in"'] \arrow[r, "of"] & facts
\end{tikzcd}$$

Concepts: 6/7

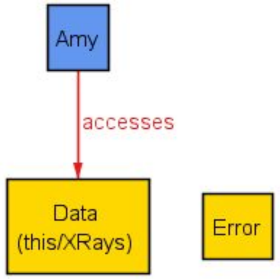


- Agreements are special statements.
  That a statement is an agreement is evident.
- Each action is based on some agreement.
  The agreement must be in the justification.
- → Agreements determine justifications

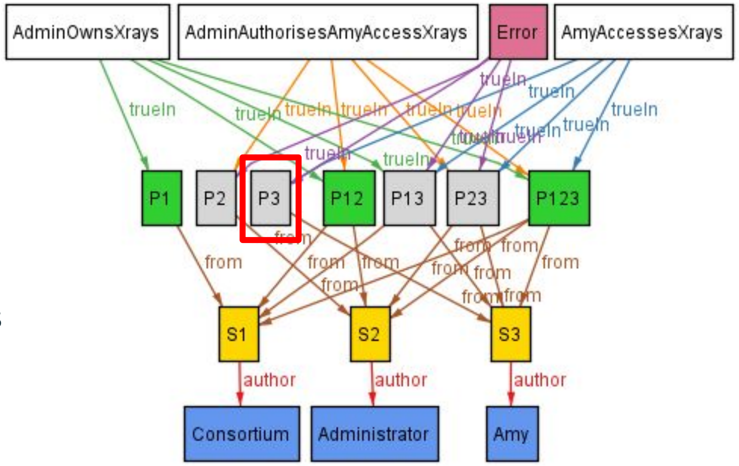
$$\begin{tikzcd}
 & \mathit{times} & \mathit{agreements} \arrow[l, "\text{appliesAt}"'] \arrow[d, Rightarrow, "\text{contents}"] & \\
 & \mathit{actions} \arrow[u, "\text{takenAt}"] \arrow[ur, "\text{basis}"'] \arrow[r, "\text{enacts}"'] \arrow[d, "\text{justification}"'] & \mathit{statements} \arrow[d, Rightarrow] & \\
 & \mathit{message\ sets} \arrow[d, "\text{extract}"'] & \mathit{messages} \arrow[l, Rightarrow] \arrow[r, "\text{author}"] & \mathit{agents} \\
\mathit{valid} \arrow[r, Rightarrow] & \mathit{policies} & \mathit{truths} \arrow[l, "\text{in}"'] \arrow[r, "\text{of}"] & \mathit{facts}
\end{tikzcd}$$

A Usage Example

Model of domain relations in extract($\{s_3\}$)



Model of framework-level relations



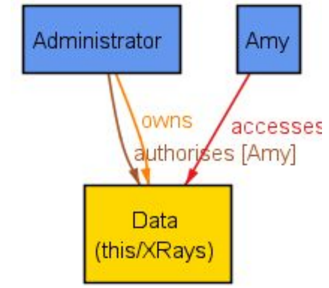
Amy accesses X-rays

```
% Statement 's3' authored by 'amy'
ctl-accesses(amy, x-rays).
```

← Amy can enact this but $\{s_3\}$ is invalid

A Usage Example

Model of domain relations in extract($\{s_1,s_2,s_3\}$)



A priori agreement to empower the admin.

```
% Statement 's1' by 'consortium' (contents of agreement at time 1)
owns(administrator, Data) :- ctl-accesses(Accessor, Data).
error :- ctl-accesses(Accessor, Data), owns(Owner, Data),
       not ctl-authorises(Owner, Accessor, Data).
```

Administrator authorises a particular data-access.

```
% Statement 's2' by 'administrator'
ctl-authorises(administrator, amy, x-rays).
```

Amy accesses X-rays

```
% Statement 's3' authored by 'amy'
ctl-accesses(amy, x-rays).
```

A Usage Example

A priori agreement to empower the admin.

```
% Statement 's1' by 'consortium' (contents of agreement at time 1)
owns(administrator, Data) :- ctl-accesses(Accessor, Data).
error :- ctl-accesses(Accessor, Data), owns(Owner, Data),
        not ctl-authorises(Owner, Accessor, Data).
```

Amy impersonates the administrator?

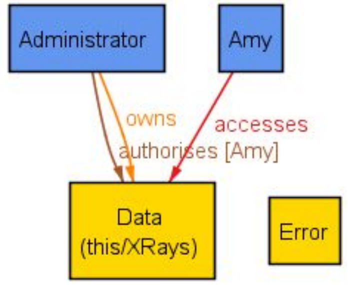
```
% Statement 's2’' by 'amy'
ctl-authorises(administrator, amy, x-rays).
```

Amy accesses X-rays

```
% Statement 's3' authored by 'amy'
ctl-accesses(amy, x-rays).
```

A Usage Example

Model of domain relations in extract({s_1,s'_2,s_3})



A priori agreement to empower the admin.

```
% Statement 's1' by 'consortium' (contents of agreement at time 1)
owns(administrator, Data) :- ctl-accesses(Accessor, Data).
error :- ctl-accesses(Accessor, Data), owns(Owner, Data),
      not ctl-authorises(Owner, Accessor, Data).
```

Amy impersonates the administrator?

```
% Statement 's2'' by 'amy'
ctl-authorises(administrator, amy, x-rays).
```

Amy accesses X-rays

```
% Statement 's3' authored by 'amy'
ctl-accesses(amy, x-rays).
```

A Usage Example

The consortium “takes back” the agreement?

```
% Statement 's1' by 'consortium' (contents of agreement at time 1)
owns(administrator, Data) :- ctl-accesses(Accessor, Data).
error :- ctl-accesses(Accessor, Data), owns(Owner, Data),
       not ctl-authorises(Owner, Accessor, Data).
```

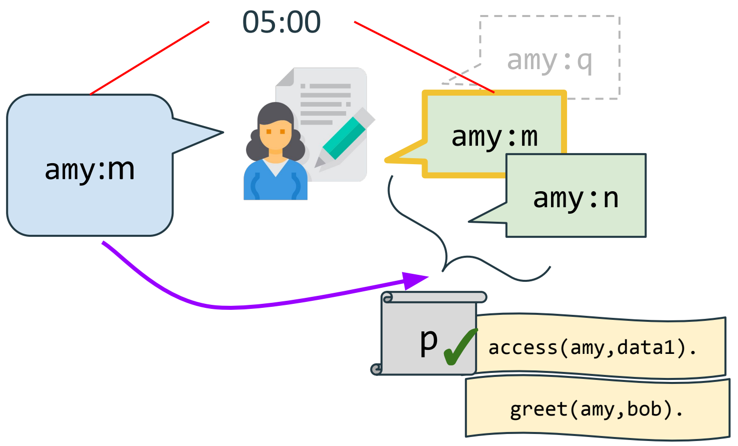
Administrator authorises a particular data-access.

```
% Statement 's2' by 'administrator'
ctl-authorises(administrator, amy, x-rays).
```

Amy accesses X-rays

```
% Statement 's3' authored by 'amy'
ctl-accesses(amy, x-rays).
```

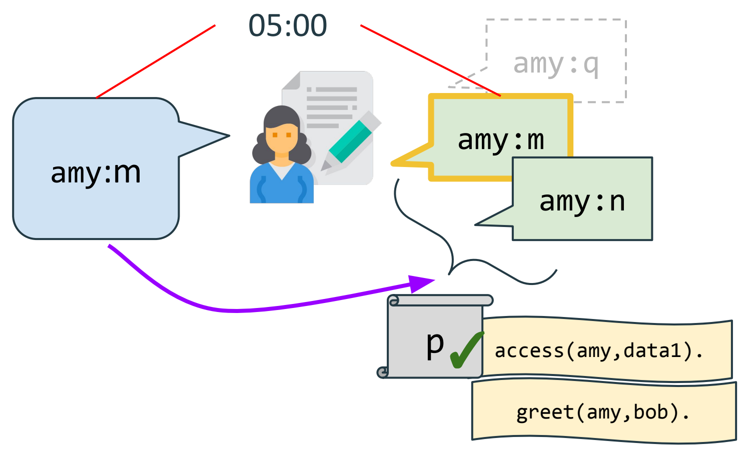
Concepts: 6/7



$$\begin{tikzcd}
 & \mathit{times} & \mathit{agreements} \arrow[l, "\text{appliesAt}"'] \arrow[d, Rightarrow, "\text{contents}"] & \\
 & \mathit{actions} \arrow[u, "\text{takenAt}"] \arrow[ur, "\text{basis}"'] \arrow[r, "\text{enacts}"'] \arrow[d, "\text{justification}"'] & \mathit{statements} \arrow[d, Rightarrow] & \\
 & \mathit{message\ sets} \arrow[d, "\text{extract}"'] & \mathit{messages} \arrow[l, Rightarrow] \arrow[r, "\text{author}"] & \mathit{agents} \\
\mathit{valid} \arrow[r, Rightarrow] & \mathit{policies} & \mathit{truths} \arrow[l, "\text{in}"'] \arrow[r, "\text{of}"] & \mathit{facts}
\end{tikzcd}$$

Concepts: 7/7

- Actions and agreements are contextualised by time (instants). Each action must be contemporary with its basis agreement.
- → Changing the time effectively changes the agreements, i.e., this models mutability.



$$\begin{tikzcd}
 & \textit{times} & \textit{agreements} \arrow[l, "\text{appliesAt}"'] \arrow[d, Rightarrow, "\text{contents}"] & \\
 & \textit{actions} \arrow[u, "\text{takenAt}"] \arrow[ur, "\text{basis}"'] \arrow[r, "\text{enacts}"'] \arrow[d, "\text{justification}"'] & \textit{statements} \arrow[d, Rightarrow] & \\
 & \textit{message sets} \arrow[d, "\text{extract}"'] & \textit{messages} \arrow[l, Rightarrow] \arrow[r, "\text{author}"] & \textit{agents} \\
\textit{valid} \arrow[r, Rightarrow] & \textit{policies} & \textit{truths} \arrow[l, "\text{in}"'] \arrow[r, "\text{of}"] & \textit{facts}
\end{tikzcd}$$

A Usage Example

The consortium “takes back” the agreement?

```
% Statement 's11' by 'consortium' (contents of agreement at time 1)
~~owns(administrator, Data) :- ctl-accesses(Accessor, Data).~~
error :- ctl-accesses(Accessor, Data), owns(Owner, Data),
      not ctl-authorises(Owner, Accessor, Data).
```

Administrator authorises a particular data-access.

```
% Statement 's2' by 'administrator'
ctl-authorises(administrator, amy, x-rays).
```

Amy accesses X-rays

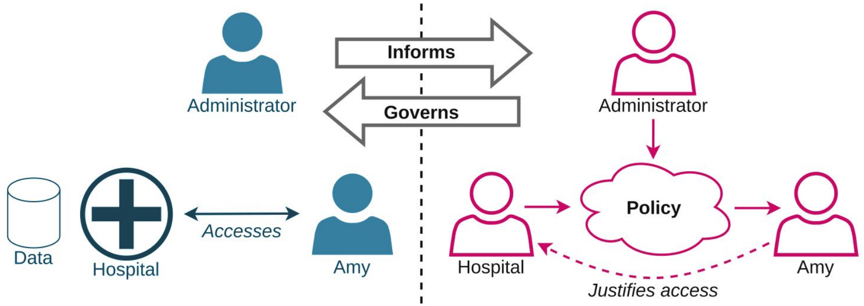
```
% Statement 's3' authored by 'amy'
ctl-accesses(amy, x-rays).
```

Characteristics

amy:m ✓
amy:m
amy:n

In systems implementing the framework…

- Agents can make statements and take actions autonomously, lossily, asynchronously.
- It suffices for the relevant policies to reach the relevant actors.
- Only agreements must be synchronised.
- All agents (e.g., an auditor) can decide whether a given action is permitted.



Looking Forward

Work continues to develop and use the framework:

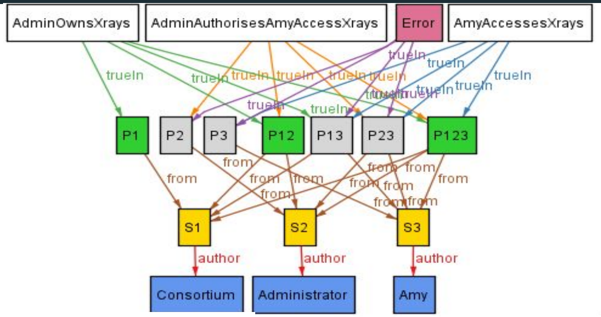
1. We **develop policy languages** for this
   a. We adapt existing languages (see PLNL!)

2. We experiment with **implementations**.

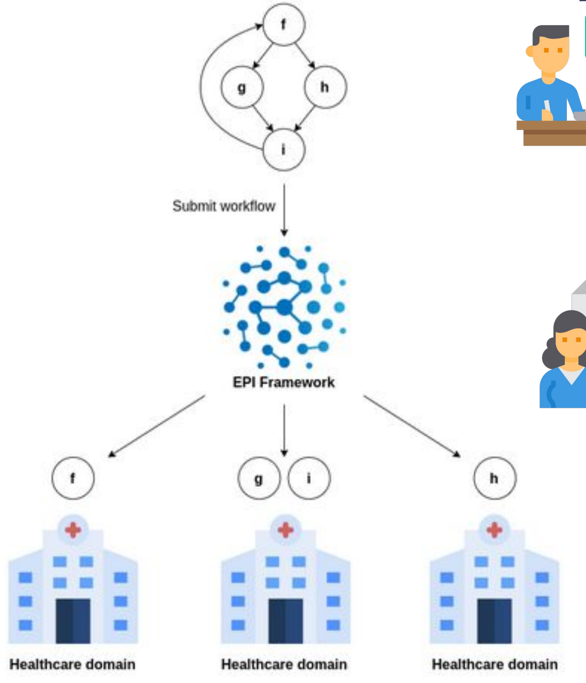
3. We **automate agent work**:
   a. Policy analysis and search via ASP

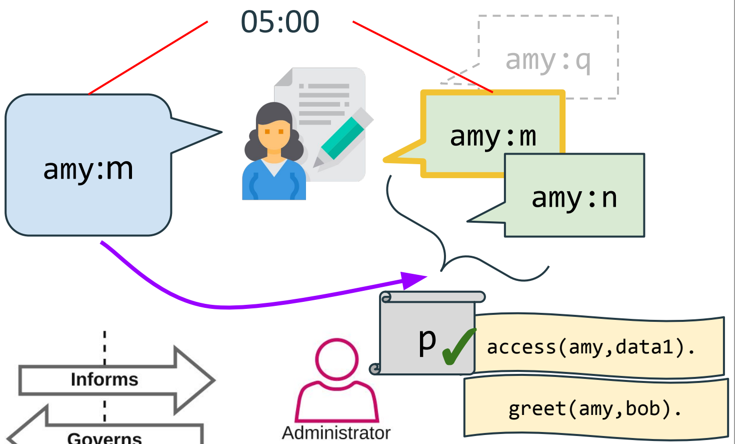
4. Extend framework to explicitly **treat privacy**.

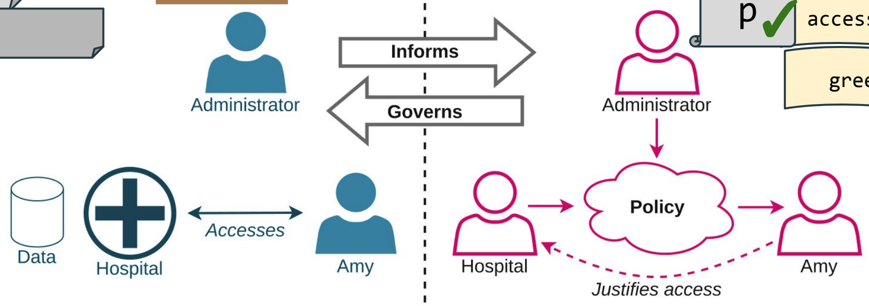
End.



$$\begin{tikzcd}
 & times & agreements \arrow[l, "appliesAt"'] \arrow[d, Rightarrow, "contents"] & \\
 & actions \arrow[u, "takenAt"] \arrow[ur, "basis"] \arrow[r, "enacts"'] \arrow[d, "justification"'] & statements \arrow[d, Rightarrow] & \\
 & message\ sets \arrow[d, "extract"'] & messages \arrow[l, Rightarrow] \arrow[r, "author"] & agents \\
valid \arrow[r, Rightarrow] & policies & truths \arrow[l, "in"'] \arrow[r, "of"] & facts
\end{tikzcd}$$







Graveyard

A Usage Example

A priori agreement to empower the admin.

```
% Statement 's1' by 'consortium' (contents of agreement at time 1)
owns(administrator, Data) :- ctl-accesses(Accessor, Data).
error :- ctl-accesses(Accessor, Data), owns(Owner, Data),
      not ctl-authorises(Owner, Accessor, Data).
```

Administrator authorises a particular data-access.

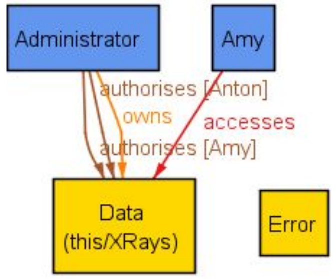
```
% Statement 's2' by 'administrator'
ctl-authorises(administrator, amy, x-rays).
```

Anton "the antagonist" considers misbehaving.

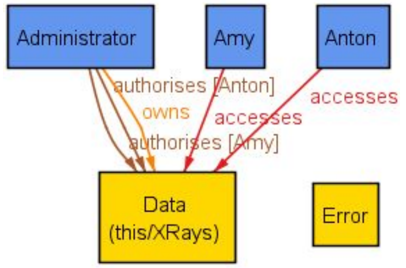
```
% Statement 's4' authored by 'anton'
ctl-authorises(administrator, anton, x-rays).
```

```
% Statement 's5' authored by 'anton'
owns(anton, x-rays).
```

Model of domain relations in extract({s_1,s_2,s_4})



Model of domain relations in extract({s_1,s_2,s_4,s_5})



A Usage Example

Model of domain relations in extract($s_{6\text{-}8}$)

A priori agreement to empower the admin.

```
% Statement 's6' authored by 'administrator'
ctl-authorises(administrator, bob, x-rays)
        :- ctl-authorises(h1, bob, x-rays),
           ctl-authorises(h2, bob, x-rays).
```

Hospitals h_1 and h_2 condition their authorisations.

```
% Statement 's7' authored by 'h1'
ctl-authorises(h1, Accessor, x-rays)
:- ctl-authorises(h2, Accessor, x-rays).
```

```
% Statement 's8' authored by 'h2'
ctl-authorises(h2, Accessor, x-rays)
  :- ctl-accesses(Accessor, x-rays),
 not ctl-accesses(anton   , x-rays).
```

A Usage Example

Model of domain relations in extract($s_{6\text{-}8}$)

A priori agreement to empower the admin.

```
% Statement 's6' authored by 'administrator'
ctl-authorises(administrator, bob, x-rays)
        :- ctl-authorises(h1, bob, x-rays),
           ctl-authorises(h2, bob, x-rays).
```

Hospitals h_1 and h_2 condition their authorisations.

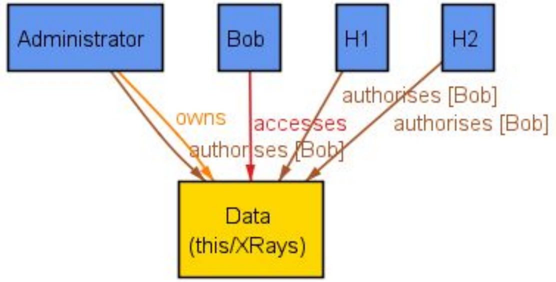
```
% Statement 's7' authored by 'h1'
ctl-authorises(h1, Accessor, x-rays)
:- ctl-authorises(h2, Accessor, x-rays).
```

```
% Statement 's8' authored by 'h2'
ctl-authorises(h2, Accessor, x-rays)
  :- ctl-accesses(Accessor, x-rays),
 not ctl-accesses(anton    , x-rays).
```

Bob accesses data (justifiably!)

```
% Statement 's9' authored by 'bob'
ctl-accesses(bob, x-rays).
```

Model of domain relations in extract($\{s_6,s_7,s_8,s_9\}$)



A Usage Example

Model of domain relations in extract(s_{6-8})

A priori agreement to empower the admin.

```
% Statement 's6' authored by 'administrator'
ctl-authorises(administrator, bob, x-rays)
        :- ctl-authorises(h1, bob, x-rays),
           ctl-authorises(h2, bob, x-rays).
```

Hospitals h_1 and h_2 condition their authorisations.

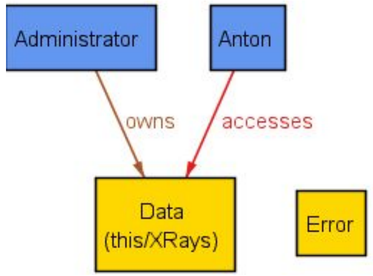
```
% Statement 's7' authored by 'h1'
ctl-authorises(h1, Accessor, x-rays)
:- ctl-authorises(h2, Accessor, x-rays).
```

```
% Statement 's8' authored by 'h2'
ctl-authorises(h2, Accessor, x-rays)
  :- ctl-accesses(Accessor, x-rays),
 not ctl-accesses(anton    , x-rays).
```

Anton accesses data (unjustifiably!)

```
% Statement 's10' authored by 'anton'
ctl-accesses(anton, x-rays).
```

Model of domain relations in extract($\{s_6, s_7, s_8, s_{10}\}$)


Administrator
Anton
owns
accesses
Data (this/XRays)
Error