Conflict Detection in Composite Institutions

Tingting Li
Tina Balke Marina De Vos Ken Satoh Julian Padget

Department of Computer Science
University of Bath

Ample Workshop, 27 September 2012

Institutions

- Collection of policies to govern the actions of the participants
- which when actions are permitted, recognised/empowered and obliged
- based on the current state of the system.
- Designed from a specific point of view.
- Can be combined together to give holistic view of the system
- allow participants to seemingly interact with one institution rather than several ones
- BUT individual institutions can disagree
- resulting in conflicts

Institutions

- Collection of policies to govern the actions of the participants
- which when actions are permitted, recognised/empowered and obliged
- based on the current state of the system.
- Designed from a specific point of view.
- Can be combined together to give holistic view of the system
- allow participants to seemingly interact with one institution rather than several ones
- BUT individual institutions can disagree
- resulting in conflicts

Institutions

- Collection of policies to govern the actions of the participants
- which when actions are permitted, recognised/empowered and obliged
- based on the current state of the system.
- Designed from a specific point of view.
- Can be combined together to give holistic view of the system
- allow participants to seemingly interact with one institution rather than several ones
- BUT individual institutions can disagree
- resulting in conflicts

Institutions

- Collection of policies to govern the actions of the participants
- which when actions are permitted, recognised/empowered and obliged
- based on the current state of the system.
- Designed from a specific point of view.
- Can be combined together to give holistic view of the system
- allow participants to seemingly interact with one institution rather than several ones
- BUT individual institutions can disagree
- resulting in conflicts

Institutions

- Collection of policies to govern the actions of the participants
- which when actions are permitted, recognised/empowered and obliged
- based on the current state of the system.
- Designed from a specific point of view.
- Can be combined together to give holistic view of the system
- allow participants to seemingly interact with one institution rather than several ones
- BUT individual institutions can disagree
- resulting in conflicts

Institutions

- Collection of policies to govern the actions of the participants
- which when actions are permitted, recognised/empowered and obliged
- based on the current state of the system.
- Designed from a specific point of view.
- Can be combined together to give holistic view of the system
- allow participants to seemingly interact with one institution rather than several ones
- BUT individual institutions can disagree
- resulting in conflicts

Institutions

- Collection of policies to govern the actions of the participants
- which when actions are permitted, recognised/empowered and obliged
- based on the current state of the system.
- Designed from a specific point of view.
- Can be combined together to give holistic view of the system
- allow participants to seemingly interact with one institution rather than several ones
- BUT individual institutions can disagree
- resulting in conflicts

Institutions

- Collection of policies to govern the actions of the participants
- which when actions are permitted, recognised/empowered and obliged
- based on the current state of the system.
- Designed from a specific point of view.
- Can be combined together to give holistic view of the system
- allow participants to seemingly interact with one institution rather than several ones
- BUT individual institutions can disagree
- resulting in conflicts

Conflicts

Conflicts:

- Inconsistent state of their world views
- An action permitted in one institution but not in another
- An action empowered in one institution but temporary disabled in another
- Contradictory states

This paper:

1. We want our composite institutions to be conflict free
2. Participants interact with one entity
3. do not need to choose which policies to violate in case of conflict
4. Detecting conflicts when composing institutions

Conflicts

Conflicts:

- Inconsistent state of their world views
- An action permitted in one institution but not in another
- An action empowered in one institution but temporary disabled in another
- Contradictory states

This paper:

1. We want our composite institutions to be conflict free
2. Participants interact with one entity
3. do not need to choose which policies to uphold in case of conflict
4. Detecting conflicts when composing institutions

Conflicts

Conflicts:

- Inconsistent state of their world views
- An action permitted in one institution but not in another
- An action empowered in one institution but temporary disabled in another
- Contradictory states

This paper:

1. We want our composite institutions to be conflict free
2. Participants interact with one entity
3. do not need to choose which policies to uphold in case of conflict
4. Detecting conflicts when composing institutions

Conflicts

Conflicts:

- Inconsistent state of their world views
- An action permitted in one institution but not in another
- An action empowered in one institution but temporary disabled in another
- Contradictory states

This paper:

- We want our composite institutions to be conflict-free
- Participants interact with one entity
- do not need to choose which policies to violate in case of conflict
- Detecting conflicts when composing institutions

Conflicts

Conflicts:

- Inconsistent state of their world views
- An action permitted in one institution but not in another
- An action empowered in one institution but temporary disabled in another
- Contradictory states

This paper:

- We want our composite institutions to be conflict-free
- Participants interact with one entity
- do not need to choose which policies to violate in case of conflict
- Detecting conflicts when composing institutions

Conflicts

Conflicts:

- Inconsistent state of their world views
- An action permitted in one institution but not in another
- An action empowered in one institution but temporary disabled in another
- Contradictory states

This paper:

- We want our composite institutions to be conflict-free
- Participants interact with one entity
- do not need to choose which policies to violate in case of conflict
- Detecting conflicts when composing institutions

Conflicts

Conflicts:

- Inconsistent state of their world views
- An action permitted in one institution but not in another
- An action empowered in one institution but temporary disabled in another
- Contradictory states

This paper:

- We want our composite institutions to be conflict-free
- Participants interact with one entity
- do not need to choose which policies to violate in case of conflict
- Detecting conflicts when composing institutions

Conflicts

Conflicts:

- Inconsistent state of their world views
- An action permitted in one institution but not in another
- An action empowered in one institution but temporary disabled in another
- Contradictory states

This paper:

- We want our composite institutions to be conflict-free
- Participants interact with one entity
- do not need to choose which policies to violate in case of conflict
- Detecting conflicts when composing institutions

Conflicts

Conflicts:

- Inconsistent state of their world views
- An action permitted in one institution but not in another
- An action empowered in one institution but temporary disabled in another
- Contradictory states

This paper:

- We want our composite institutions to be conflict-free
- Participants interact with one entity
- do not need to choose which policies to violate in case of conflict
- Detecting conflicts when composing institutions

Outline

1. Motivation
   - Institutions
   - Conflicts
2. Institutions & Policies
3. Composite Institutions
4. Conflict Detection

Outline

1. Motivation
   - Institutions
   - Conflicts
2. Institutions & Policies
3. Composite Institutions
4. Conflict Detection

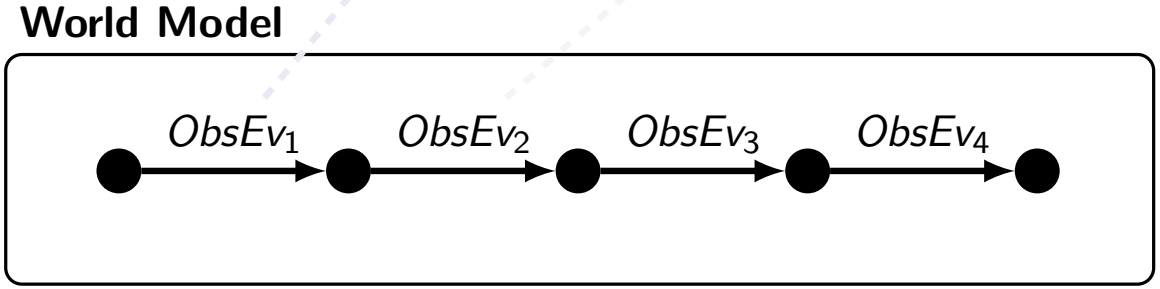
Outline

1. Motivation
   - Institutions
   - Conflicts
2. Institutions & Policies
3. Composite Institutions
4. Conflict Detection

Outline

1. Motivation
   - Institutions
   - Conflicts
2. Institutions & Policies
3. Composite Institutions
4. Conflict Detection

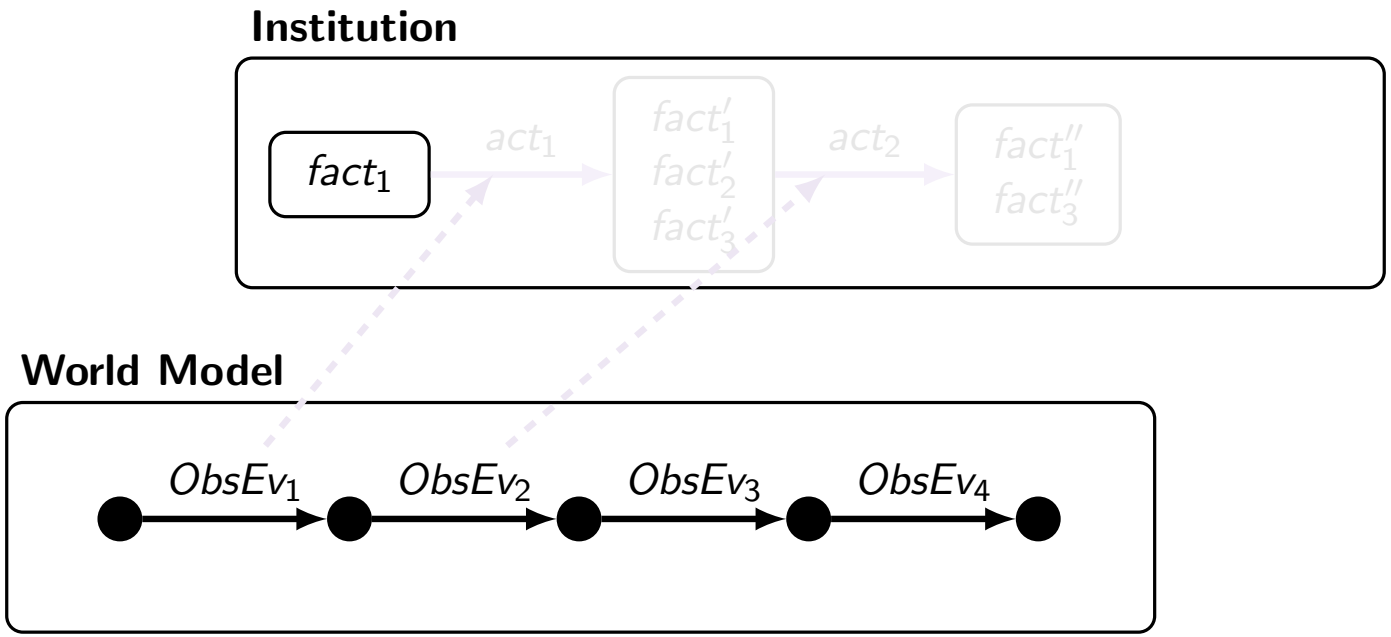
Model sketch



- Model generates ordered traces that show us the evolution of the normative system over time—allows validation, auditing, provenance
- Essential elements of model are

Institution: $\mathcal{I} := \langle \mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{C}, \Delta \rangle$

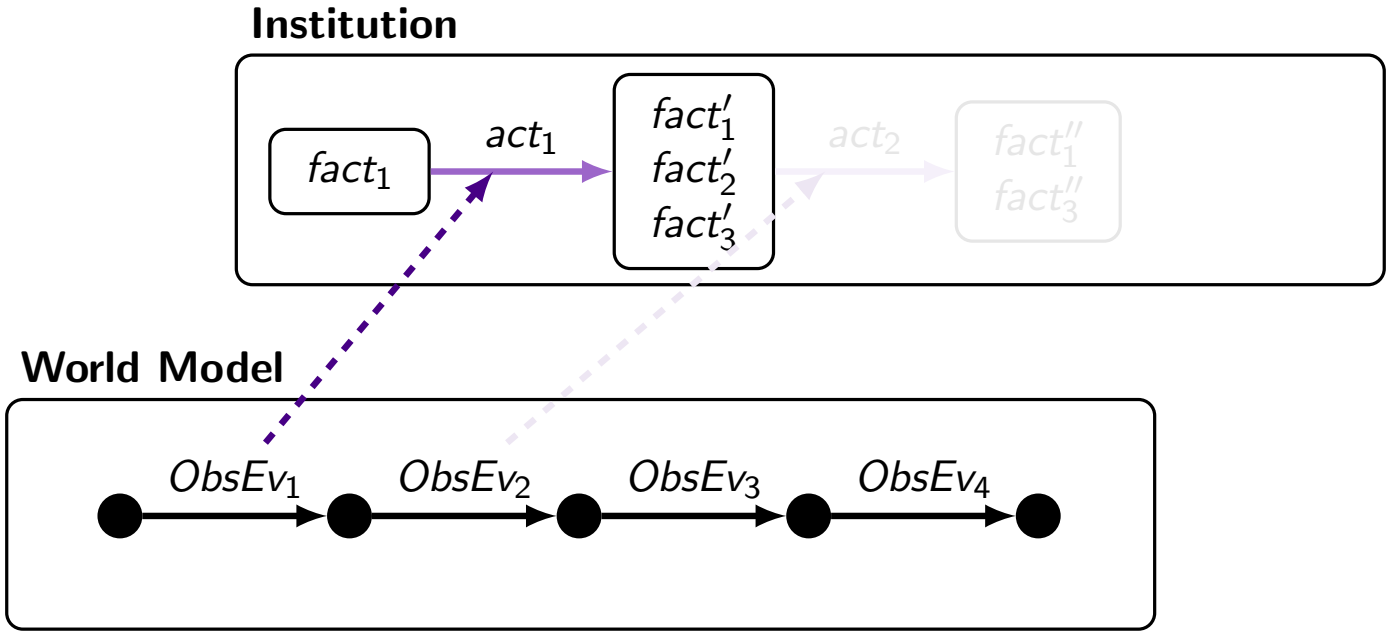
Model sketch



- Model generates ordered traces that show us the evolution of the normative system over time—allows validation, auditing, provenance
- Essential elements of model are:

Institution: $\mathcal{I} := \langle \mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{C}, \Delta \rangle$

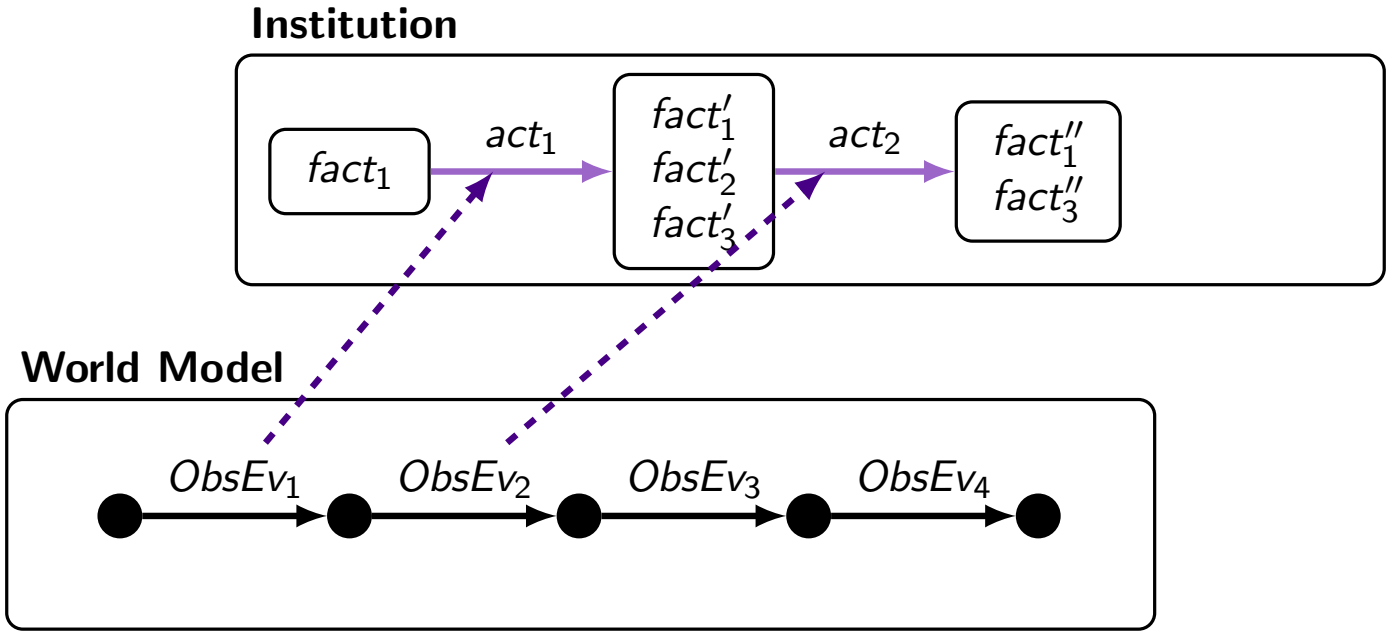
Model sketch



- Model generates ordered traces that show us the evolution of the normative system over time—allows validation, auditing, provenance
- Essential elements of model are:

Institution: $\mathcal{I} := \langle \mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{C}, \Delta \rangle$

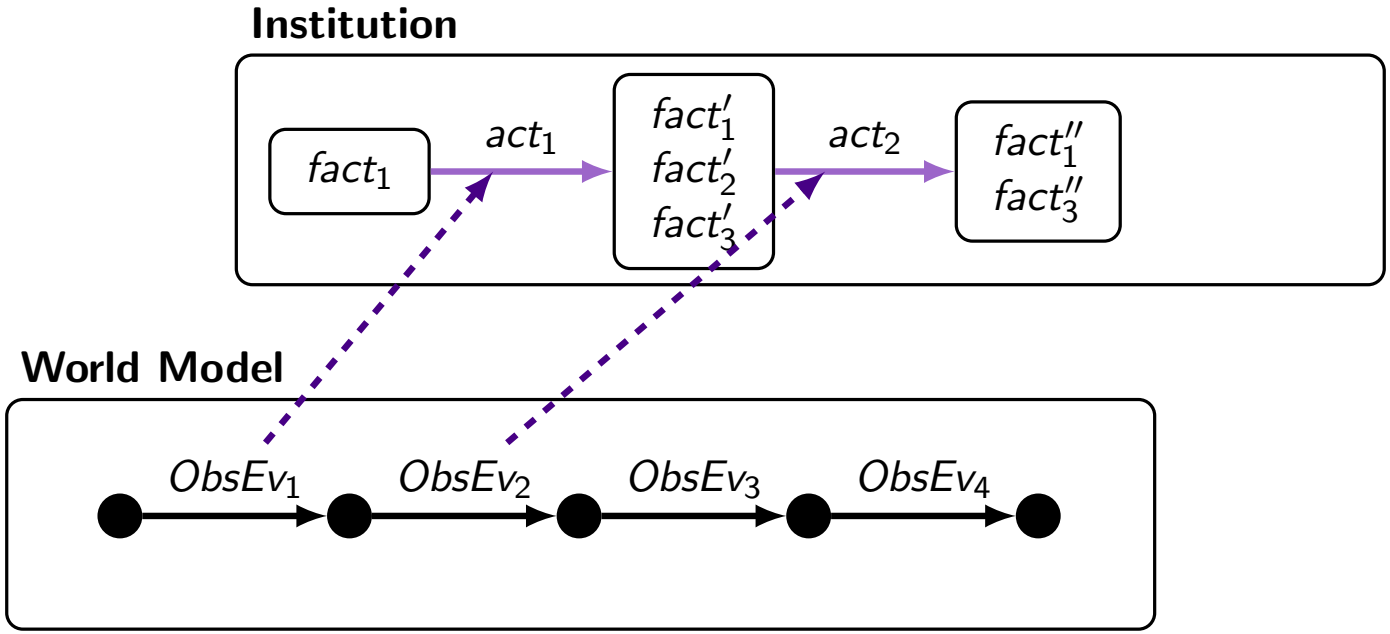
Model sketch



- Model generates **ordered traces** that show us the evolution of the normative system over time—allows validation, auditing, provenance
- Essential elements of model are:
  - events ($\mathcal{E}$): exogenous and institutional
  - fluents ($\mathcal{F}$): power, permission, obligation, norm, ...

Institution: $\mathcal{I} := \langle \mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{C}, \Delta \rangle$

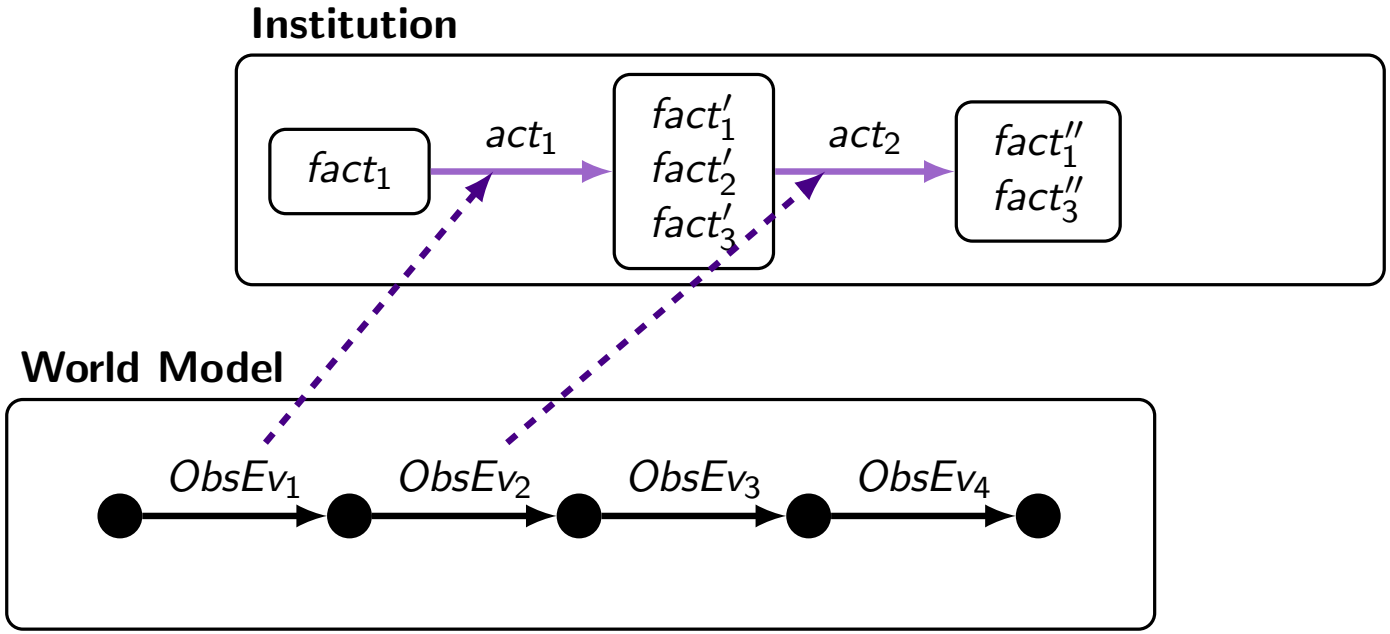
Model sketch



- Model generates **ordered traces** that show us the evolution of the normative system over time—allows validation, auditing, provenance
- Essential elements of model are:
  - events ($\mathcal{E}$): exogenous and institutional
  - fluents ($\mathcal{F}$): power, permission, obligation, domain

Institution: $\mathcal{I} := \langle \mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{C}, \Delta \rangle$

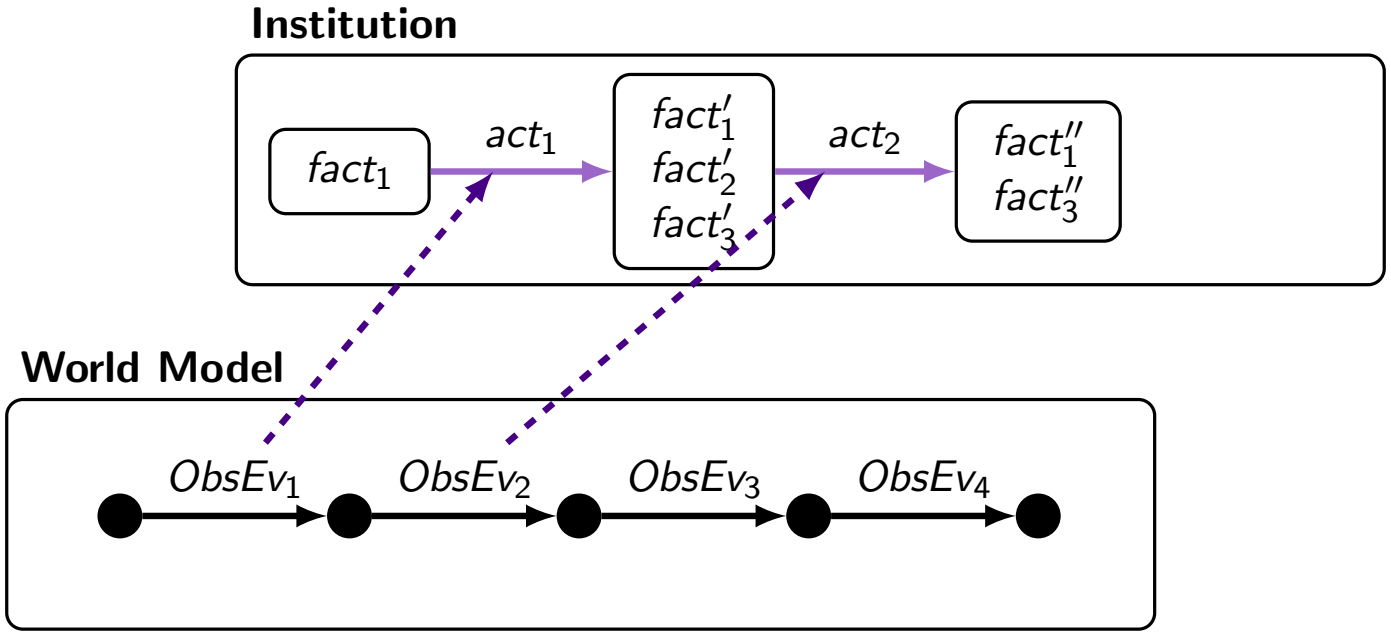
Model sketch



- Model generates **ordered traces** that show us the evolution of the normative system over time—allows validation, auditing, provenance
- Essential elements of model are:
  - events ($\mathcal{E}$): exogenous and institutional
  - fluents ($\mathcal{F}$): power, permission, obligation, domain

Institution: $\mathcal{I} := \langle \mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{C}, \Delta \rangle$

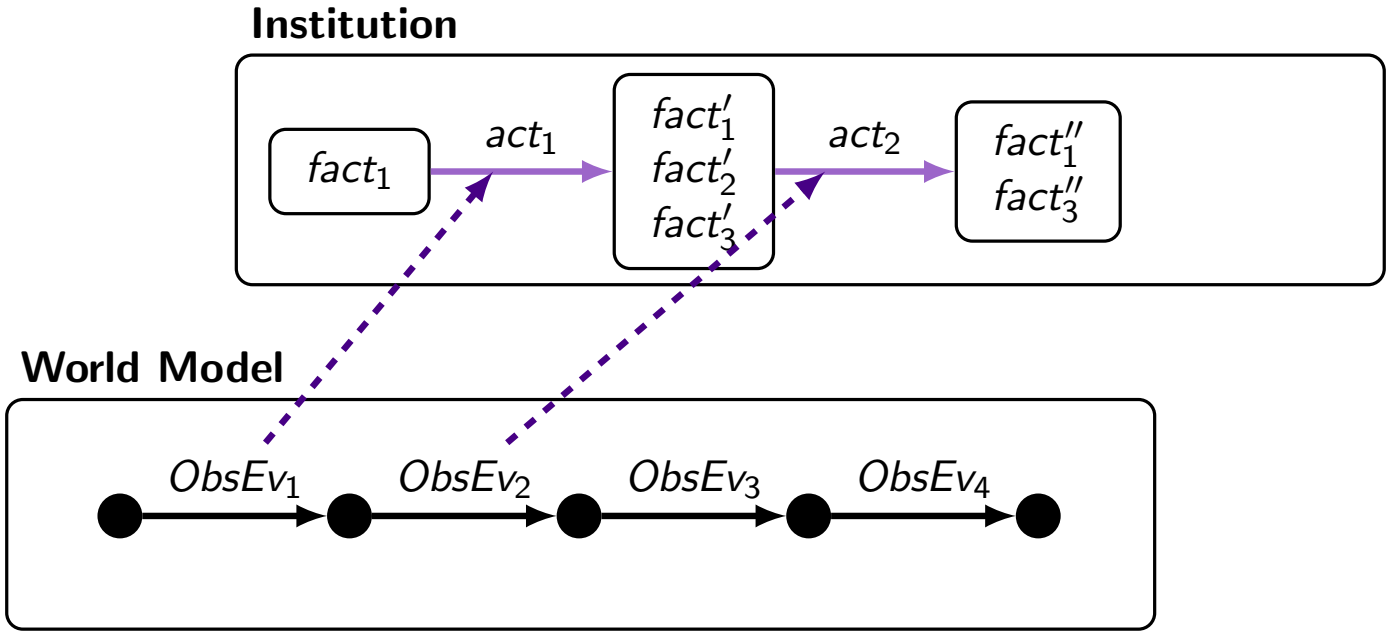
Model sketch



- Model generates **ordered traces** that show us the evolution of the normative system over time—allows validation, auditing, provenance
- Essential elements of model are:
  - events ($\mathcal{E}$): exogenous and institutional
  - fluents ($\mathcal{F}$): power, permission, obligation, domain

Institution: $\mathcal{I} := \langle \mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{C}, \Delta \rangle$

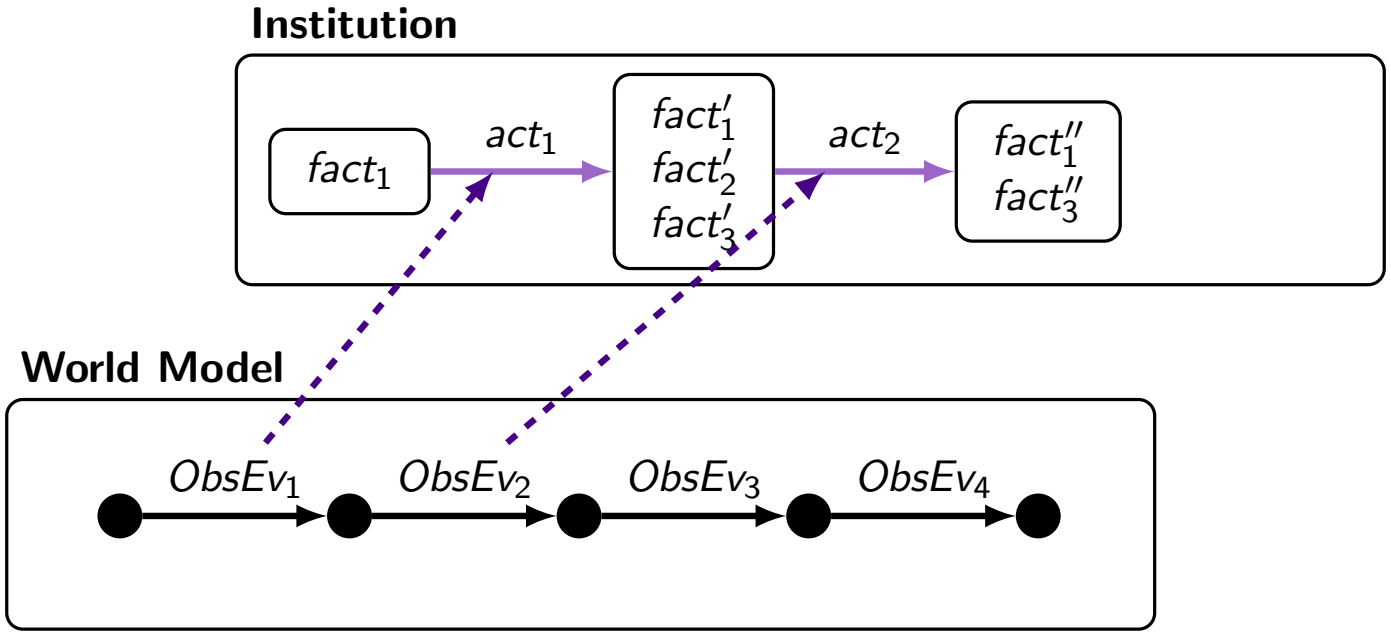
Model sketch



- Model generates **ordered traces** that show us the evolution of the normative system over time—allows validation, auditing, provenance
- Essential elements of model are:
  - events ($\mathcal{E}$): exogenous and institutional
  - fluents ($\mathcal{F}$): power, permission, obligation, domain

Institution: $\mathcal{I} := \langle \mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{C}, \Delta \rangle$

Model sketch



- Model generates **ordered traces** that show us the evolution of the normative system over time—allows validation, auditing, provenance
- Essential elements of model are:
  - events ($\mathcal{E}$): exogenous and institutional
  - fluents ($\mathcal{F}$): power, permission, obligation, domain

Institution: $\mathcal{I} := \langle \mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{C}, \Delta \rangle$

Traces

- An ordered trace is defined as a sequence of observable events

$$\langle e_0, e_1, \ldots, e_n \rangle \quad e_i \in \mathcal{E}_{ex}, 0 \leq i \leq n$$

- The evaluation of an ordered trace for a given starting state S_0 is a sequence $\langle S_0, S_1, \ldots S_{n+1} \rangle$ such that $S_{i+1} = TR(S_i, e_i)$
- Ordered traces and their evaluations allow us to monitor or investigate the evolution of an institution over time

Traces

- An ordered trace is defined as a sequence of observable events

$$\langle e_0, e_1, \ldots, e_n \rangle \quad e_i \in \mathcal{E}_{ex}, 0 \leq i \leq n$$

- The evaluation of an ordered trace for a given starting state S_0 is a sequence $\langle S_0, S_1, \ldots S_{n+1} \rangle$ such that $S_{i+1} = TR(S_i, e_i)$
- Ordered traces and their evaluations allow us to monitor or investigate the evolution of an institution over time

Traces

- An ordered trace is defined as a sequence of observable events

$$\langle e_0, e_1, \ldots, e_n \rangle \quad e_i \in \mathcal{E}_{ex}, 0 \leq i \leq n$$

- The evaluation of an ordered trace for a given starting state S_0 is a sequence $\langle S_0, S_1, \ldots S_{n+1} \rangle$ such that $S_{i+1} = \mathrm{TR}(S_i, e_i)$
- Ordered traces and their evaluations allow us to monitor or investigate the evolution of an institution over time.

Traces

- An ordered trace is defined as a sequence of observable events

$$\langle e_0, e_1, \ldots, e_n \rangle \quad e_i \in \mathcal{E}_{ex}, 0 \leq i \leq n$$

- The evaluation of an ordered trace for a given starting state S_0 is a sequence $\langle S_0, S_1, \ldots S_{n+1} \rangle$ such that $S_{i+1} = \mathrm{TR}(S_i, e_i)$
- Ordered traces and their evaluations allow us to monitor or investigate the evolution of an institution over time.

Traces

- An ordered trace is defined as a sequence of observable events

$$\langle e_0, e_1, \ldots, e_n \rangle \quad e_i \in \mathcal{E}_{ex}, 0 \leq i \leq n$$

- The evaluation of an ordered trace for a given starting state S_0 is a sequence $\langle S_0, S_1, \ldots S_{n+1} \rangle$ such that $S_{i+1} = \mathrm{TR}(S_i, e_i)$
- Ordered traces and their evaluations allow us to monitor or investigate the evolution of an institution over time.

Framework translation to *AnsProlog*

- Answer set programming used as computational back-end
- *Conclusion* : −*Conditions* with use of negation as failure
- Important components:
  - observed(Event, Instant): an exogenous event at time t
  - occurred(Event, Instant): an institutional event at time t
  - holdsat(Fluent, Instant): fluent is true at time t
  - pow(Event), perm(Event), obl(Event, Event, Event): fluents that indicate norms
  - initiated(Fluent, Instant): fluents to be added to state
  - terminated(Fluent, Instant): fluents to be removed from the state

Framework translation to *AnsProlog*

- Answer set programming used as computational back-end
- *Conclusion* : −*Conditions* with use of negation as failure
- Important components:
  - `observed(Event,Instant)`: an exogenous event at time t
  - `occurred(Event,Instant)`: an institutional event at time t
  - `holdsat(Fluent,Instant)`: fluent is true at time t
  - `pow(Event),perm(Event),obl(Event,Deadline,Breach)`: fluents that indicate norms
  - `initiated(Fluent,Instant)`: fluents to be added to create the state
  - `terminated(Fluent,Instant)`: fluents to be removed from the state

Framework translation to *AnsProlog*

- Answer set programming used as computational back-end
- *Conclusion* : − *Conditions* with use of negation as failure
- Important components:
  - `observed(Event,Instant)`: an exogenous event at time t
  - `occurred(Event,Instant)`: an institutional event at time t
  - `holdsat(Fluent,Instant)`: fluent is true at time t
  - `pow(Event),perm(Event),obl(Event,Event,Event)`: fluents that indicate norms
  - `initiated(Fluent,Instant)`: fluents to be added to state
  - `terminated(Fluent,Instant)`: fluents to be deleted from the state

Framework translation to *AnsProlog*

- Answer set programming used as computational back-end
- *Conclusion* : − *Conditions* with use of negation as failure
- Important components:
  - `observed(Event,Instant)`: an exogenous event at time t
  - `occurred(Event,Instant)`: an institutional event at time t
  - `holdsat(Fluent,Instant)`: fluent is true at time t
  - `pow(Event)`,`perm(Event)`,`obl(Event,Event,Event)`: fluents that indicate norms
  - `initiated(Fluent,Instant)`: fluents to be added to state
  - `terminated(Fluent,Instant)`: fluents to be deleted from the state

Framework translation to *AnsProlog*

- Answer set programming used as computational back-end
- *Conclusion* : − *Conditions* with use of negation as failure
- Important components:
  - `observed(Event,Instant)`: an exogenous event at time t
  - `occurred(Event,Instant)`: an institutional event at time t
  - `holdsat(Fluent,Instant)`: fluent is true at time t
  - `pow(Event),perm(Event),obl(Event,Event,Event)`: fluents that indicate norms
  - `initiated(Fluent,Instant)`: fluents to be added to state
  - `terminated(Fluent,Instant)`: fluents to be deleted from the state

Framework translation to *AnsProlog*

- Answer set programming used as computational back-end
- *Conclusion* : − *Conditions* with use of negation as failure
- Important components:
  - `observed(Event,Instant)`: an exogenous event at time t
  - `occurred(Event,Instant)`: an institutional event at time t
  - `holdsat(Fluent,Instant)`: fluent is true at time t
  - `pow(Event)`,`perm(Event)`,`obl(Event,Event,Event)`: fluents that indicate norms
  - `initiated(Fluent,Instant)`: fluents to be added to state
  - `terminated(Fluent,Instant)`: fluents to be deleted from the state

Framework translation to *AnsProlog*

- Answer set programming used as computational back-end
- *Conclusion* : − *Conditions* with use of negation as failure
- Important components:
  - `observed(Event,Instant)`: an exogenous event at time t
  - `occurred(Event,Instant)`: an institutional event at time t
  - `holdsat(Fluent,Instant)`: fluent is true at time t
  - `pow(Event),perm(Event),obl(Event,Event,Event)`: fluents that indicate norms
  - `initiated(Fluent,Instant)`: fluents to be added to state
  - `terminated(Fluent,Instant)`: fluents to be deleted from the state

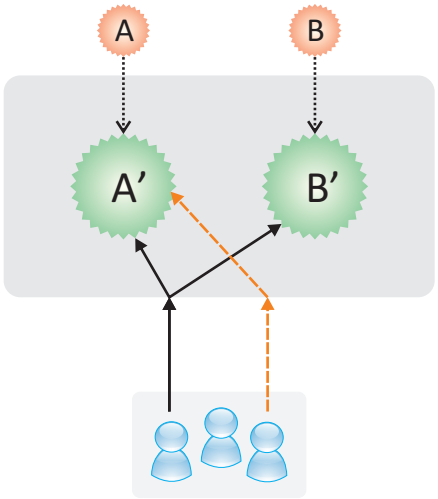
Framework translation to *AnsProlog*

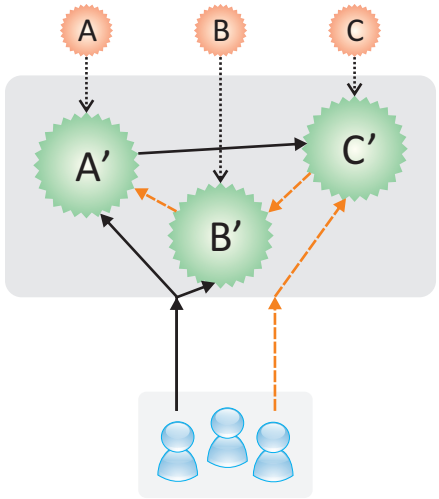
- Answer set programming used as computational back-end
- *Conclusion* : − *Conditions* with use of negation as failure
- Important components:
  - `observed(Event,Instant)`: an exogenous event at time t
  - `occurred(Event,Instant)`: an institutional event at time t
  - `holdsat(Fluent,Instant)`: fluent is true at time t
  - `pow(Event),perm(Event),obl(Event,Event,Event)`: fluents that indicate norms
  - `initiated(Fluent,Instant)`: fluents to be added to state
  - `terminated(Fluent,Instant)`: fluents to be deleted from the state

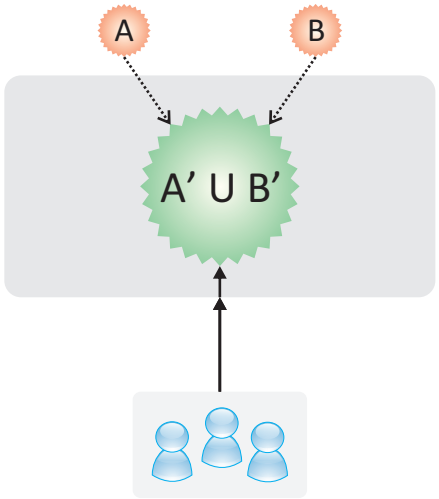
Framework translation to *AnsProlog*

- Answer set programming used as computational back-end
- *Conclusion* : − *Conditions* with use of negation as failure
- Important components:
  - `observed(Event,Instant)`: an exogenous event at time t
  - `occurred(Event,Instant)`: an institutional event at time t
  - `holdsat(Fluent,Instant)`: fluent is true at time t
  - `pow(Event)`,`perm(Event)`,`obl(Event,Event,Event)`: fluents that indicate norms
  - `initiated(Fluent,Instant)`: fluents to be added to state
  - `terminated(Fluent,Instant)`: fluents to be deleted from the state

How to compose?







Composite Institutions

- Composite Institution $C_{\mathcal{I}}$ with $\mathcal{I} = \mathcal{I}_1, \ldots, \mathcal{I}_n$
- Wrapper
- Individual states
- Composite trace = sequence of exogenous events from all individual institutions

Composite Institutions

- Composite Institution $C_{\mathcal{I}}$ with $\mathcal{I} = \mathcal{I}_1, \ldots, \mathcal{I}_n$
- Wrapper
- Individual states
- Composite trace = sequence of exogenous events from all individual institutions

Composite Institutions

- Composite Institution $C_{\mathcal{I}}$ with $\mathcal{I} = \mathcal{I}_1, \ldots, \mathcal{I}_n$
- Wrapper
- Individual states
- Composite trace = sequence of exogenous events from all individual institutions

Composite Institutions

- Composite Institution $C_{\mathcal{I}}$ with $\mathcal{I} = \mathcal{I}_1, \ldots, \mathcal{I}_n$
- Wrapper
- Individual states
- Composite trace = sequence of exogenous events from all individual institutions

Synchronised Traces

- Unrecognised events in composite trace
- could lead individual institutions to have different time lines
- So how/when to compare states?
- Null events
- Synchronised traces

Synchronised Traces

- Unrecognised events in composite trace
- could lead individual institutions to have different time lines
- So how/when to compare states?
- Null events
- Synchronised traces

Synchronised Traces

- Unrecognised events in composite trace
- could lead individual institutions to have different time lines
- So how/when to compare states?
- Null events
- Synchronised traces

Synchronised Traces

- Unrecognised events in composite trace
- could lead individual institutions to have different time lines
- So how/when to compare states?
- Null events
- Synchronised traces

Synchronised Traces

- Unrecognised events in composite trace
- could lead individual institutions to have different time lines
- So how/when to compare states?
- Null events
- Synchronised traces

Conflicts

- Given a composite trace
- a conflict occurs for any two institutions of the composite
- a fluent known to both appears
- positively in one and
- negatively in the other
- at the same point of time
- Conflict traces

Conflicts

- Given a composite trace
- a conflict occurs for any two institutions of the composite
- a fluent known to both appears
- positively in one and
- negatively in the other
- at the same point of time
- Conflict traces

Conflicts

- Given a composite trace
- a conflict occurs for any two institutions of the composite
- a fluent known to both appears
- positively in one and
- negatively in the other
- at the same point of time
- Conflict traces

Conflicts

- Given a composite trace
- a conflict occurs for any two institutions of the composite
- a fluent known to both appears
- positively in one and
- negatively in the other
- at the same point of time
- Conflict traces

Conflicts

- Given a composite trace
- a conflict occurs for any two institutions of the composite
- a fluent known to both appears
- positively in one and
- negatively in the other
- at the same point of time
- Conflict traces

Conflicts

- Given a composite trace
- a conflict occurs for any two institutions of the composite
- a fluent known to both appears
- positively in one and
- negatively in the other
- at the same point of time
- Conflict traces

Conflicts

- Given a composite trace
- a conflict occurs for any two institutions of the composite
- a fluent known to both appears
- positively in one and
- negatively in the other
- at the same point of time
- Conflict traces

Renaming

- Computing conflict traces
- Not sufficient to add *AnsProlog* program together
- Contradiction results in no answer set sets
- Rename each institution to make the unique
- `rename(E, ERE)`

Renaming

- Computing conflict traces
- Not sufficient to add *AnsProlog* program together
- Contradiction results in no answer set sets
- Rename each institution to make the unique
- `rename(E, ERE)`

Renaming

- Computing conflict traces
- Not sufficient to add *AnsProlog* program together
- Contradiction results in no answer set sets
- Rename each institution to make the unique
- `rename(E, ERE)`

Renaming

- Computing conflict traces
- Not sufficient to add *AnsProlog* program together
- Contradiction results in no answer set sets
- Rename each institution to make the unique
- `rename(E, ERE)`

Renaming

- Computing conflict traces
- Not sufficient to add *AnsProlog* program together
- Contradiction results in no answer set sets
- Rename each institution to make the unique
- `rename(E, ERE)`

Composite traces

- previously exactly one `observed` event per time instance
- now exactly one `compObserved` based on available exogenous events
- used to generate corresponding renamed `observed` event

```
compEvent(E)      :- evtype(ERE, ex), evinst(E, In), rename(E, ERE), instRE(In).
compEvent(E)      :- evtype(E, ex), evinst(E, In), inst(In).
{compObserved(E, I)} :- compEvent(E), instant(I), not final(I).
ev(I)             :- compObserved(E, I), instant(I).
                  :- not ev(I), instant(I), not final(I).
                  :- compObserved(E1, I), compObserved(E2, I), E1 != E2,
                     instant(I), compEvent(E1), compEvent(E2).
```

Composite traces

- previously exactly one `observed` event per time instance
- now exactly one `compObserved` based on available exogenous events
- used to generate corresponding renamed `observed` event

```
compEvent(E)          :- evtype(ERE, ex), evinst(E, In), rename(E, ERE), instRE(In).
compEvent(E)          :- evtype(E, ex), evinst(E, In), inst(In).
{compObserved(E, I)}  :- compEvent(E), instant(I), not final(I).
ev(I)                 :- compObserved(E, I), instant(I).
                      :- not ev(I), instant(I), not final(I).
                      :- compObserved(E1, I), compObserved(E2, I), E1 != E2,
                         instant(I), compEvent(E1), compEvent(E2).
```

Composite traces

- previously exactly one `observed` event per time instance
- now exactly one `compObserved` based on available exogenous events
- used to generate corresponding renamed `observed` event

```
compEvent(E)   :- evtype(ERE, ex), evinst(E, In), rename(E, ERE), instRE(In).
compEvent(E)   :- evtype(E, ex), evinst(E, In), inst(In).
{compObserved(E, I)}   :- compEvent(E), instant(I), not final(I).
ev(I)   :- compObserved(E, I), instant(I).
        :- not ev(I), instant(I), not final(I).
        :- compObserved(E1, I), compObserved(E2, I), E1 != E2,
           instant(I), compEvent(E1), compEvent(E2).
```

Composite traces

- previously exactly one `observed` event per time instance
- now exactly one `compObserved` based on available exogenous events
- used to generate corresponding renamed `observed` event

```
compEvent(E)   :- evtype(ERE, ex), evinst(E, In), rename(E, ERE), instRE(In).
compEvent(E)   :- evtype(E, ex), evinst(E, In), inst(In).
{compObserved(E, I)}   :- compEvent(E), instant(I), not final(I).
ev(I)   :- compObserved(E, I), instant(I).
        :- not ev(I), instant(I), not final(I).
        :- compObserved(E1, I), compObserved(E2, I), E1 != E2,
           instant(I), compEvent(E1), compEvent(E2).
```

Composite traces

- previously exactly one `observed` event per time instance
- now exactly one `compObserved` based on available exogenous events
- used to generate corresponding renamed `observed` event

```
compEvent(E)  :- evtype(ERE, ex), evinst(E, In), rename(E, ERE), instRE(In).
compEvent(E)  :- evtype(E, ex), evinst(E, In), inst(In).
{compObserved(E, I)}  :- compEvent(E), instant(I), not final(I).
ev(I)  :- compObserved(E, I), instant(I).
       :- not ev(I), instant(I), not final(I).
       :- compObserved(E1, I), compObserved(E2, I), E1! = E2,
          instant(I), compEvent(E1), compEvent(E2).
```

Selecting Conflict traces

- Conflicts occur when two institutions disagree on a fluent known to both of them
- We are only interested in traces that result in conflicts
- Two `conflict` atoms for efficiency and clarification

```
conflict    :- holdsat(F, I), not holdsat(FRE, I), rename(F, FRE),
               instant(I).
conflict    :- holdsat(FRE, I), not holdsat(F, I), rename(F, FRE),
               instant(I).
            :- not conflict.
conflict(F, FRE, I)    :- holdsat(F, I), not holdsat(FRE, I), rename(F, FRE),
                          instant(I).
conflict(FRE, F, I)    :- holdsat(FRE, I), not holdsat(F, I), rename(F, FRE),
                          instant(I).
```

Selecting Conflict traces

- Conflicts occur when two institutions disagree on a fluent known to both of them
- We are only interested in traces that result in conflicts
- Two `conflict` atoms for efficiency and clarification

```
conflict    :- holdsat(F, I), not holdsat(FRE, I), rename(F, FRE),
               instant(I).
conflict    :- holdsat(FRE, I), not holdsat(F, I), rename(F, FRE),
               instant(I).
            :- not conflict.
conflict(F, FRE, I)    :- holdsat(F, I), not holdsat(FRE, I), rename(F, FRE),
                          instant(I).
conflict(FRE, F, I)    :- holdsat(FRE, I), not holdsat(F, I), rename(F, FRE),
                          instant(I).
```

Selecting Conflict traces

- Conflicts occur when two institutions disagree on a fluent known to both of them
- We are only interested in traces that result in conflicts
- Two `conflict` atoms for efficiency and clarification

```
conflict :- holdsat(F, I), not holdsat(FRE, I), rename(F, FRE),
            instant(I).
conflict :- holdsat(FRE, I), not holdsat(F, I), rename(F, FRE),
            instant(I).
         :- not conflict.
conflict(F, FRE, I) :- holdsat(F, I), not holdsat(FRE, I), rename(F, FRE),
                       instant(I).
conflict(FRE, F, I) :- holdsat(FRE, I), not holdsat(F, I), rename(F, FRE),
                       instant(I).
```

Selecting Conflict traces

- Conflicts occur when two institutions disagree on a fluent known to both of them
- We are only interested in traces that result in conflicts
- Two `conflict` atoms for efficiency and clarification

```
conflict :- holdsat(F, I), not holdsat(FRE, I), rename(F, FRE),
            instant(I).
conflict :- holdsat(FRE, I), not holdsat(F, I), rename(F, FRE),
            instant(I).
         :- not conflict.
conflict(F, FRE, I) :- holdsat(F, I), not holdsat(FRE, I), rename(F, FRE),
                       instant(I).
conflict(FRE, F, I) :- holdsat(FRE, I), not holdsat(F, I), rename(F, FRE),
                       instant(I).
```

Selecting Conflict traces

- Conflicts occur when two institutions disagree on a fluent known to both of them
- We are only interested in traces that result in conflicts
- Two `conflict` atoms for efficiency and clarification

```
conflict :- holdsat(F, I), not holdsat(FRE, I), rename(F, FRE),
            instant(I).
conflict :- holdsat(FRE, I), not holdsat(F, I), rename(F, FRE),
            instant(I).
         :- not conflict.
conflict(F, FRE, I) :- holdsat(F, I), not holdsat(FRE, I), rename(F, FRE),
                       instant(I).
conflict(FRE, F, I) :- holdsat(FRE, I), not holdsat(F, I), rename(F, FRE),
                       instant(I).
```

Future Work

- Conflict Resolution
- Different types of Composition

Future Work

- Conflict Resolution
- Different types of Composition

Thanks

$\sim\sim$ Thank you for your attention $\sim\sim$

?? Questions ??

Thanks

$\sim\sim$ Thank you for your attention $\sim\sim$

?? Questions ??