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 allowing 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 allowing 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:

- 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:

- 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:

- 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:

- 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:

- 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:
- 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:

- 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:

- 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:
- 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
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
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

Institution:

\[ I := (E, F, G, C, \Delta) \]

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 (E)**: exogenous and institutional
- **Fluents (F)**: power, permission, obligation, domain

De Vos (University of Bath)
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**: \( I := \langle E, F, G, C, \Delta \rangle \)

---

De Vos  (University of Bath)  
Policy Conflicts  
mdv@cs.bath.ac.uk
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 ($E$): exogenous and institutional
  - fluents ($F$): power, permission, obligation, domain

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

Institution

\[ \text{fact}_1 \xrightarrow{\text{act}_1} \text{fact}'_1 \xrightarrow{\text{act}_2} \text{fact}''_1 \]

World Model

\[ \text{ObsEv}_1 \xrightarrow{} \text{ObsEv}_2 \xrightarrow{} \text{ObsEv}_3 \xrightarrow{} \text{ObsEv}_4 \]

- 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 (\(E\)): exogenous and institutional
  - fluents (\(F\)): power, permission, obligation, domain

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

Institution

\[ \text{Institution: } I := (E, F, G, C, \Delta) \]

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 (\(E\)): exogenous and institutional
- fluents (\(F\)): power, permission, obligation, domain
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**: $\mathcal{I} := \langle \mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{C}, \Delta \rangle$
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.
• 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.
An ordered trace is defined as a sequence of observable events
\[ \langle e_0, e_1, \ldots, e_n \rangle \quad e_i \in E_{ex}, 0 \leq i \leq n \]
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.
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.
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 \( S_{i+1} = \text{TR}(S_i, e_i) \)

Ordered traces and their evaluations allow us to monitor or investigate the evolution of an institution over time.
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
- pos(Event): perm(Event): int(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
Answer set programming used as computational back-end

Conclusion: \(-\text{Conditions}\) 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): fluents that indicate norms
- initiated(Fluent,Instant): fluents to be added to state
- terminated(Fluent,Instant): fluents to be deleted from the state
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*: \( \neg \text{Conditions} \) 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
Answer set programming used as computational back-end

**Conclusion**: \(\neg Conditions\) with use of negation as failure

**Important components:**
- \(\text{observed(Event,Instant)}\): an exogenous event at time \(t\)
- \(\text{occurred(Event,Instant)}\): an institutional event at time \(t\)
- \(\text{holdsat(Fluent,Instant)}\): fluent is true at time \(t\)
- \(\text{pow(Event)}, \text{perm(Event)}, \text{obl(Event,Event,Event)}\): fluents that indicate norms
- \(\text{initiated(Fluent,Instant)}\): fluents to be added to state
- \(\text{terminated(Fluent,Instant)}\): fluents to be deleted from the state
Answer set programming used as computational back-end

**Conclusion**: $\neg$-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
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
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
Answer set programming used as computational back-end

Conclusion: \( \neg \) 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?

De Vos (University of Bath)
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 Institution $C_I$ with $I = I_1, \ldots, I_n$

Wrapper

- Individual states
- Composite trace = sequence of exogenous events from all individual 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 $C_I$ with $I = I_1, \ldots, I_n$
- Wrapper
- Individual states
- Composite trace = sequence of exogenous events from all individual institutions
Unrecognised events in composite trace

could lead individual institutions to have different time lines

So how/when to compare states?

Null events

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
Unrecognised events in composite trace

could lead individual institutions to have different time lines

So how/when to compare states?

Null events

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
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
Given a composite trace, a conflict occurs for any two institutions of the composite if a fluent known to both appears positively in one and negatively in the other at the same point of time. Conflict traces.
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
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
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
- Computing conflict traces
- Not sufficient to add \textit{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)
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)
• Computing conflict traces
• Not sufficient to add *AnsProlog* program together
• Contradiction results in no answer set sets
• Rename each institution to make the unique
  \[ \text{rename}(E, ERE) \]
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

\[
\text{compEvent}(E) : \quad \text{evtype}(\text{ERE}, \text{ex}), \text{evinst}(E, \text{In}), \text{rename}(E, \text{ERE}), \text{instRE}(\text{In}).
\]

\[
\text{compEvent}(E) : \quad \text{evtype}(E, \text{ex}), \text{evinst}(E, \text{In}), \text{inst}(\text{In}).
\]

\[
\{\text{compObserved}(E, I)\} : \quad \text{compEvent}(E), \text{instant}(I), \text{not final}(I).
\]

\[
\text{ev}(I) : \quad \text{compObserved}(E, I), \text{instant}(I).
\]

\[
: \quad \text{not ev}(I), \text{instant}(I), \text{not final}(I).
\]

\[
: \quad \text{compObserved}(E_1, I), \text{compObserved}(E_2, I), E_1! = E_2,
\]

\[
\text{instant}(I), \text{compEvent}(E_1), \text{compEvent}(E_2).
\]
Composite traces

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

\begin{verbatim}
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).
\end{verbatim}
• 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 \texttt{compObserved} based on available exogenous events
- used to generate corresponding renamed observed event

\begin{verbatim}
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).
\end{verbatim}
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

\[
\begin{align*}
\text{compEvent}(E) & : - \quad \text{evtype}(\text{ERE}, \text{ex}), \text{evinst}(E, \text{In}), \text{rename}(E, \text{ERE}), \text{instRE}(\text{In}). \\
\text{compEvent}(E) & : - \quad \text{evtype}(E, \text{ex}), \text{evinst}(E, \text{In}), \text{inst}(\text{In}). \\
\{\text{compObserved}(E, I)\} & : - \quad \text{compEvent}(E), \text{instant}(I), \textbf{not} \ \text{final}(I). \\
\text{ev}(I) & : - \quad \text{compObserved}(E, I), \text{instant}(I). \\
\quad & : - \quad \textbf{not} \ \text{ev}(I), \text{instant}(I), \textbf{not} \ \text{final}(I). \\
\quad & : - \quad \text{compObserved}(E_1, I), \text{compObserved}(E_2, I), E_1 \neq E_2, \text{instant}(I), \text{compEvent}(E_1), \text{compEvent}(E_2).
\end{align*}
\]
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:

- \( \text{conflict} : \neg \text{holdsat}(F, I), \neg \text{holdsat}(\text{FRE}, I), \text{rename}(F, \text{FRE}), \text{instant}(I). \)

- \( \text{conflict}(F, \text{FRE}, I) : \neg \text{holdsat}(F, I), \neg \text{holdsat}(\text{FRE}, I), \text{rename}(F, \text{FRE}), \text{instant}(I). \)

- \( \text{conflict}(\text{FRE}, F, I) : \neg \text{holdsat}(\text{FRE}, I), \neg \text{holdsat}(F, I), \text{rename}(F, \text{FRE}), \text{instant}(I). \)
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).
```
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:

\[
\text{conflict} : - \text{holdsat}(F, I), \text{not holdsat}(\text{FRE}, I), \text{rename}(F, \text{FRE}), \text{instant}(I).
\]

\[
\text{conflict} : - \text{holdsat}(\text{FRE}, I), \text{not holdsat}(F, I), \text{rename}(F, \text{FRE}), \text{instant}(I).
\]

: - not conflict.

\[
\text{conflict}(F, \text{FRE}, I) : - \text{holdsat}(F, I), \text{not holdsat}(\text{FRE}, I), \text{rename}(F, \text{FRE}), \text{instant}(I).
\]

\[
\text{conflict}(\text{FRE}, F, I) : - \text{holdsat}(\text{FRE}, I), \text{not holdsat}(F, I), \text{rename}(F, \text{FRE}), \text{instant}(I).
\]
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:

- \(\text{conflict} : \neg \text{holdsat}(F, I), \text{not holdsat}(F, I), \text{rename}(F, \text{FRE}), \text{instant}(I).\)

- \(\text{conflict} : \neg \text{holdsat}(\text{FRE}, I), \text{not holdsat}(F, I), \text{rename}(F, \text{FRE}), \text{instant}(I).\)

- \(\text{not conflict}.\)

- \(\text{conflict}(F, \text{FRE}, I) : \neg \text{holdsat}(F, I), \text{not holdsat}(\text{FRE}, I), \text{rename}(F, \text{FRE}), \text{instant}(I).\)

- \(\text{conflict}(\text{FRE}, F, I) : \neg \text{holdsat}(\text{FRE}, I), \text{not holdsat}(F, I), \text{rename}(F, \text{FRE}), \text{instant}(I).\)
Future Work

- Conflict Resolution
- Different types of Composition
Future Work

- Conflict Resolution
- Different types of Composition
Thank you for your attention

Questions??
Thank you for your attention

Questions