Detecting Conflicts in Legal Systems

Tingting Li[1] Tina Balke[2,1] Marina De Vos[1] Julian Padget[1]
Ken Satoh[3]

[1]University of Bath
{t.li,mdv,jap}@cs.bath.ac.uk

[2]University of Surrey
t.balke@surrey.ac.uk
Tina Balke is partially supported by the NII and the ePolicy project (FP7, grant n° 288147)

[3]National Institute of Informatics
ksatoh@nii.ac.jp

International Workshop on Juris-informatics 2012







1 Motivation

2 Legal Frameworks

Conflict of Laws

- *Modus Operandi* may be interpreted differently under different jurisdictions
- Different requirements, sequences of actions, additional actions, unnecessary actions and caveats,...
- Problem: detection of possible conflicts
- Main question: Will a particular course of actions result in a conflict?

Conflict of Laws

- *Modus Operandi* may be interpreted differently under different jurisdictions
- Different requirements, sequences of actions, additional actions, unnecessary actions and caveats,. . .
- Problem: detection of possible conflicts
- Main question: Will a particular course of actions result in a conflict?

Conflict of Laws

- *Modus Operandi* may be interpreted differently under different jurisdictions
- Different requirements, sequences of actions, additional actions, unnecessary actions and caveats,...
- Problem: detection of possible conflicts
- Main question: Will a particular course of actions result in a conflict?

Conflict of Laws

- *Modus Operandi* may be interpreted differently under different jurisdictions
- Different requirements, sequences of actions, additional actions, unnecessary actions and caveats,...
- Problem: detection of possible conflicts
- Main question: Will a particular course of actions result in a conflict?

Case Study

- Private International Law
- drafting of contract between Italian and UK company
- liquidated-damage-clause specifying the compensations to be paid in case of insufficient performance

Case Study

- Private International Law
- drafting of contract between Italian and UK company
- liquidated-damage-clause specifying the compensations to be paid in case of insufficient performance

Case Study

- Private International Law
- drafting of contract between Italian and UK company
- liquidated-damage-clause specifying the compensations to be paid in case of insufficient performance

Conflicts

Conflicts:

- Inconsistent state of the world views of different jurisdictions
- An action permitted in one jurisdiction but not in another
- An action empowered in one jurisdiction but temporary disabled in another
- Contradictory states

This paper:

- Participants want be aware of potential conflicts between jurisdictions
- They are not interested in all possible conflicts, but only those ones which might be of interest for them
- Computational mechanism for detecting conflicts between different jurisdictions

Conflicts

Conflicts:

- Inconsistent state of the world views of different jurisdictions
- An action permitted in one jurisdiction but not in another
- An action empowered in one jurisdiction but temporary disabled in another
- Contradictory states

This paper:

- Participants want be aware of potential conflicts between jurisdictions
- They are not interested in all possible conflicts, but only those which might be of interest for them
- Computational mechanism for detecting conflicts between different jurisdictions

Conflicts

Conflicts:

- Inconsistent state of the world views of different jurisdictions
- An action permitted in one jurisdiction but not in another
- An action empowered in one jurisdiction but temporary disabled in another
- Contradictory states

This paper:

- Participants want be aware of potential conflicts between jurisdictions
- They are not interested in all possible conflicts, but only those which might be of interest for them
- Computational mechanism for detecting conflicts between different jurisdictions

Conflicts

Conflicts:

- Inconsistent state of the world views of different jurisdictions
- An action permitted in one jurisdiction but not in another
- An action empowered in one jurisdiction but temporary disabled in another
- Contradictory states

This paper:

- Participants want be aware of potential conflicts between jurisdiction
- They are not interested in all possible conflicts, but only those which might be of interest for them
- Computational mechanism for detecting conflicts between different jurisdictions

Conflicts

Conflicts:

- Inconsistent state of the world views of different jurisdictions
- An action permitted in one jurisdiction but not in another
- An action empowered in one jurisdiction but temporary disabled in another
- Contradictory states

This paper:

- Participants want be aware of potential conflicts between jurisdiction
- They are not interested in all possible conflicts, but only the ones which might be of interest for them
- Computational mechanism for detecting conflicts between different jurisdictions

Conflicts

Conflicts:

- Inconsistent state of the world views of different jurisdictions
- An action permitted in one jurisdiction but not in another
- An action empowered in one jurisdiction but temporary disabled in another
- Contradictory states

This paper:

- Participants want be aware of potential conflicts between jurisdiction
- They are not interested in all possible conflicts, but only the ones which might be of interest for them
- Computational mechanism for detecting conflicts between different jurisdictions

Conflicts

Conflicts:

- Inconsistent state of the world views of different jurisdictions
- An action permitted in one jurisdiction but not in another
- An action empowered in one jurisdiction but temporary disabled in another
- Contradictory states

This paper:

- Participants want be aware of potential conflicts between jurisdiction
- They are not interested in all possible conflicts, but only the ones which might be of interest for them
- Computational mechanism for detecting conflicts between different jurisdictions

Conflicts

Conflicts:

- Inconsistent state of the world views of different jurisdictions
- An action permitted in one jurisdiction but not in another
- An action empowered in one jurisdiction but temporary disabled in another
- Contradictory states

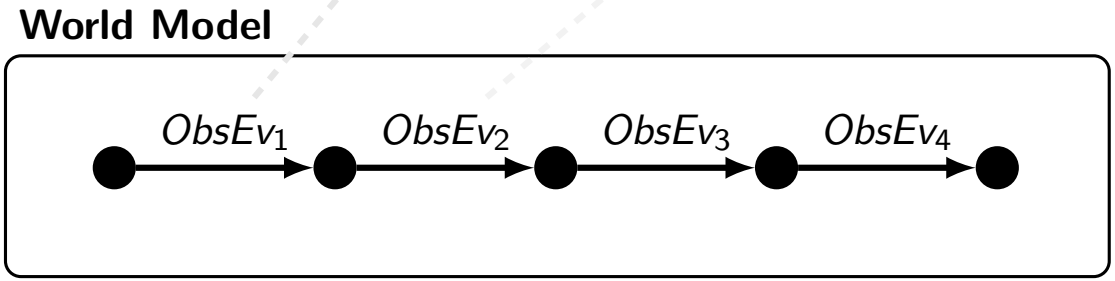
This paper:

- Participants want be aware of potential conflicts between jurisdiction
- They are not interested in all possible conflicts, but only the ones which might be of interest for them
- Computational mechanism for detecting conflicts between different jurisdictions

1 Motivation

2 Legal Frameworks
- Traces
- Mapping to *AnsProlog*

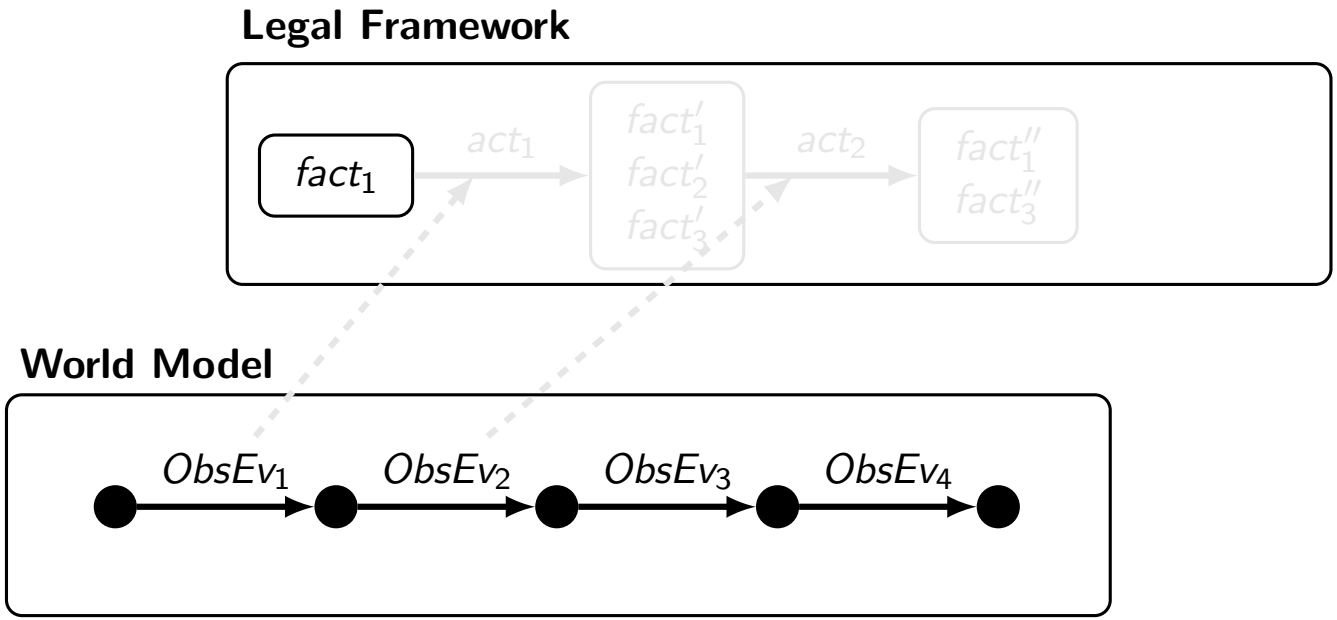
Model sketch



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

Legal Framework: $\mathcal{L} := \langle \mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{C}, \Delta \rangle$

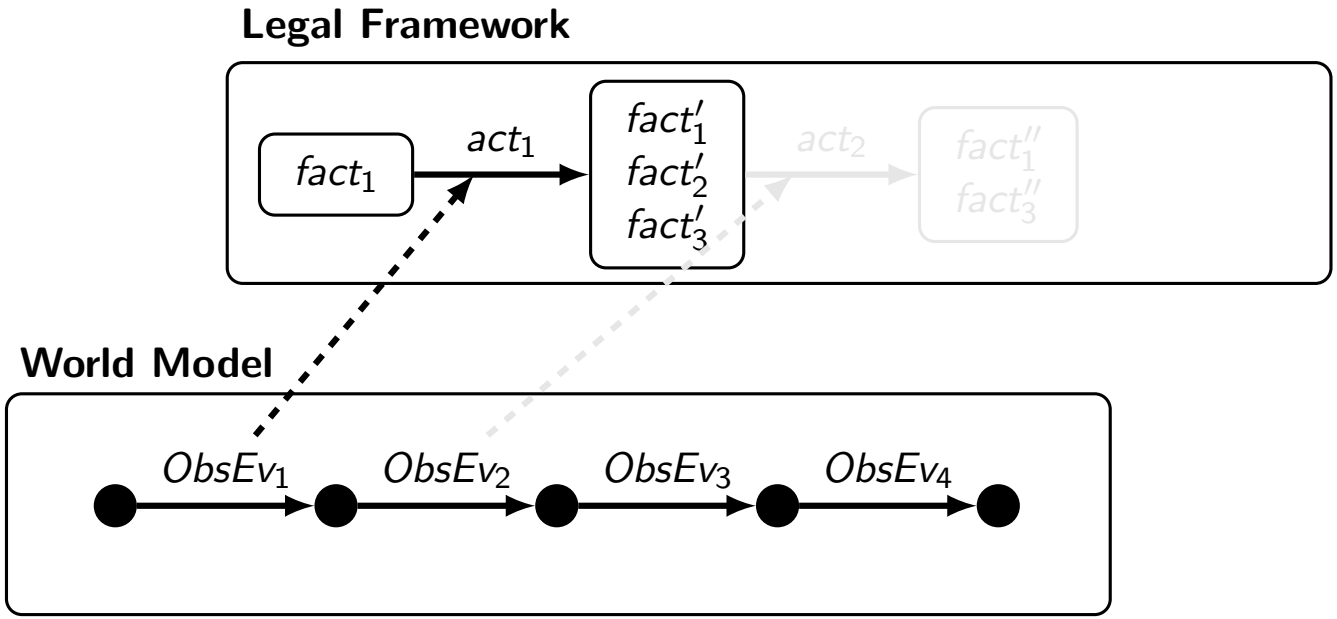
Model sketch



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

Legal Framework: $\mathcal{L} := \langle \mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{C}, \Delta \rangle$

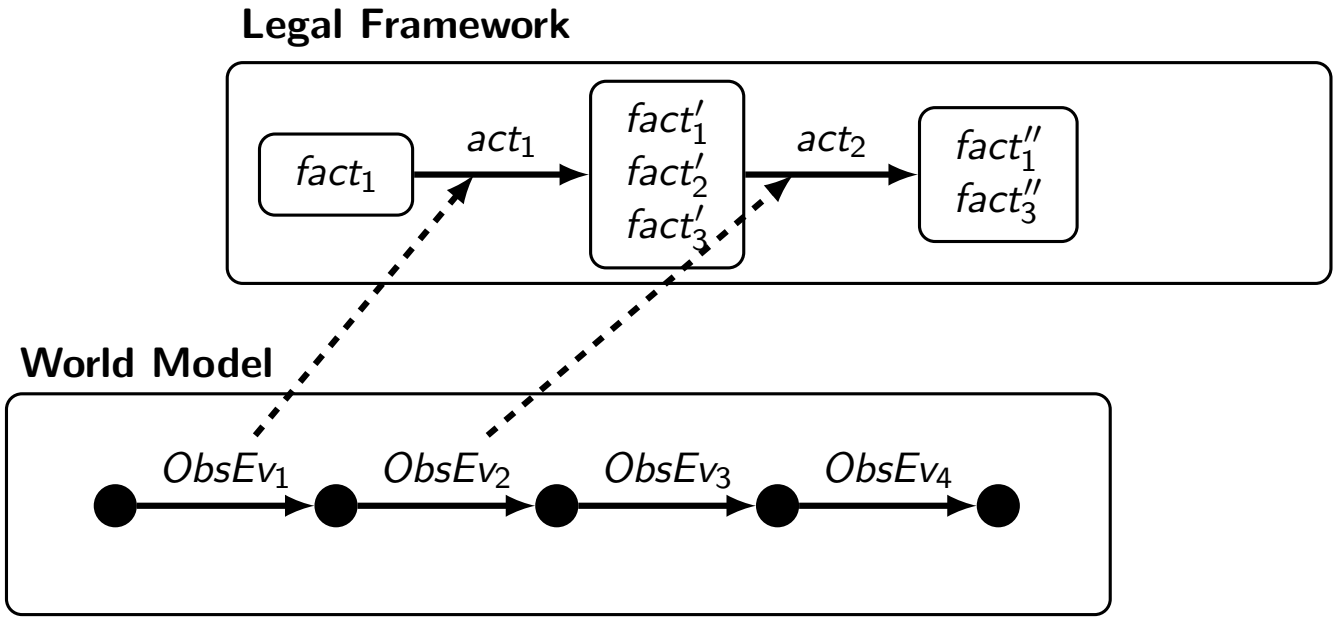
Model sketch



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

Legal Framework: $\mathcal{L} := \langle \mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{C}, \Delta \rangle$

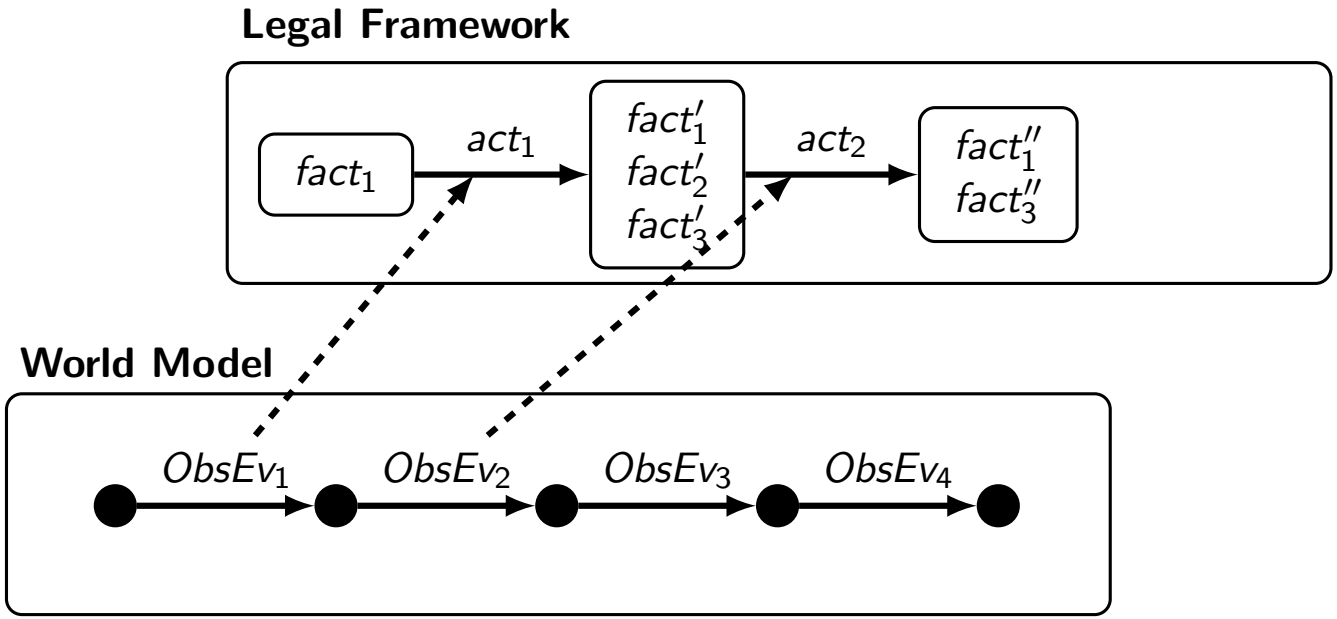
Model sketch



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

Legal Framework: $\mathcal{L} := \langle \mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{C}, \Delta \rangle$

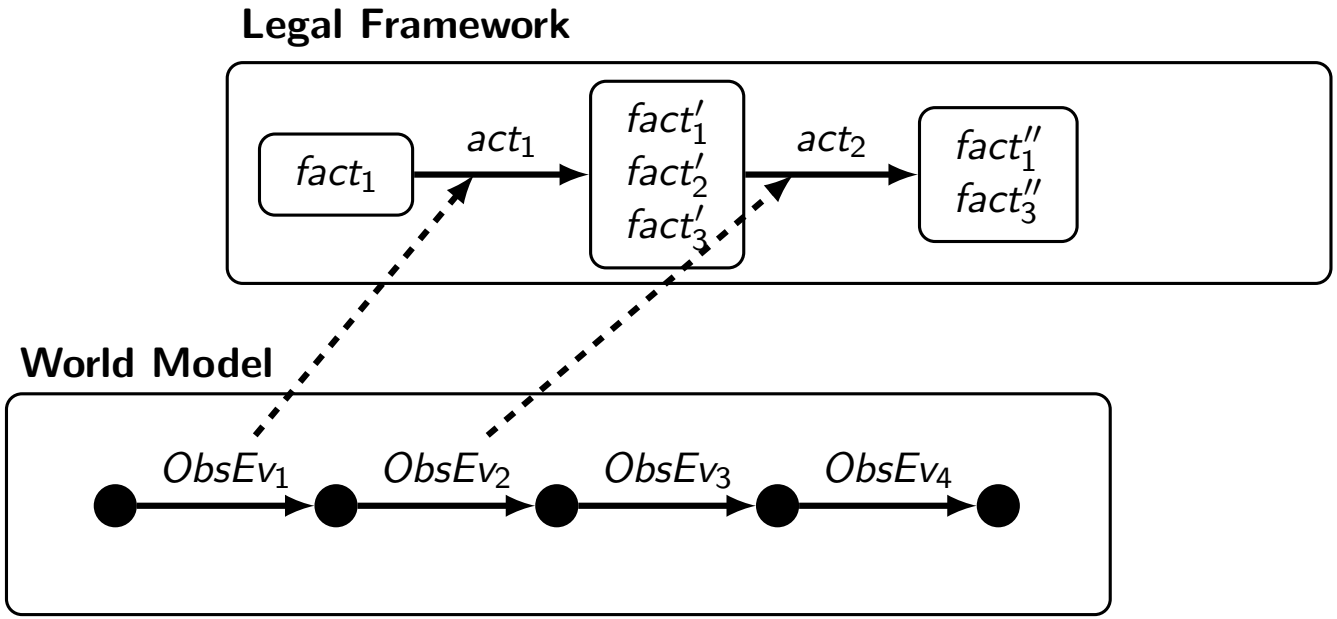
Model sketch



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

Legal Framework: $\mathcal{L} := \langle \mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{C}, \Delta \rangle$

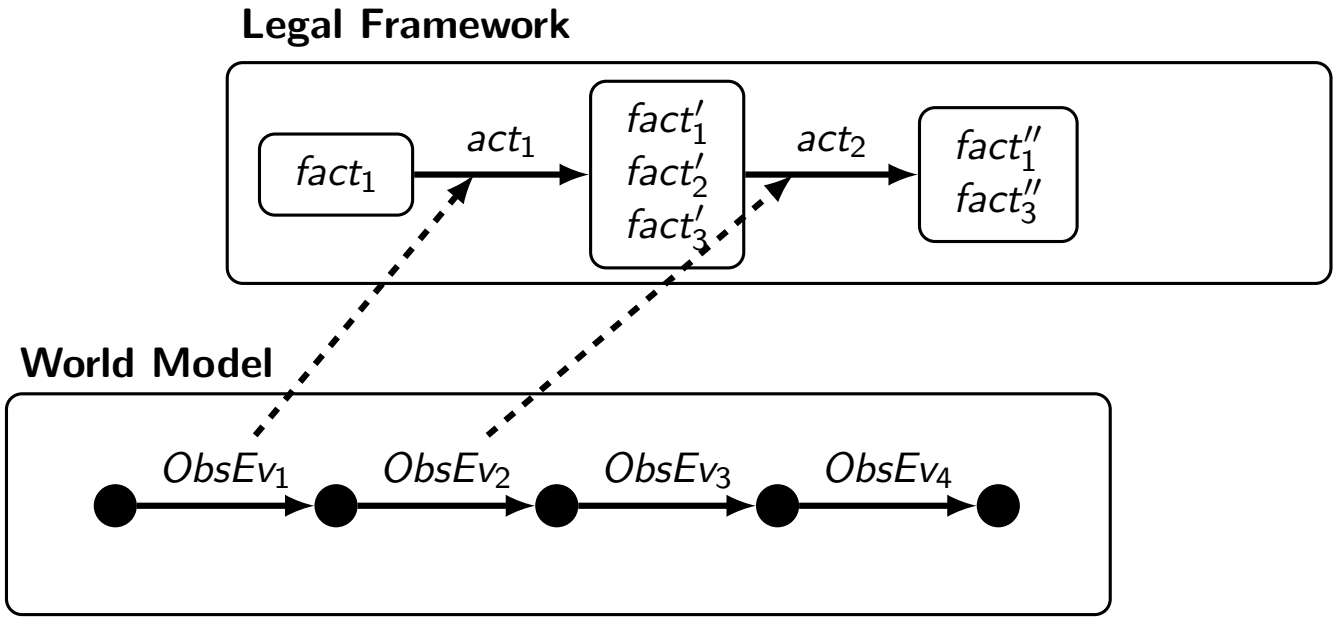
Model sketch



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

Legal Framework: $\mathcal{L} := \langle \mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{C}, \Delta \rangle$

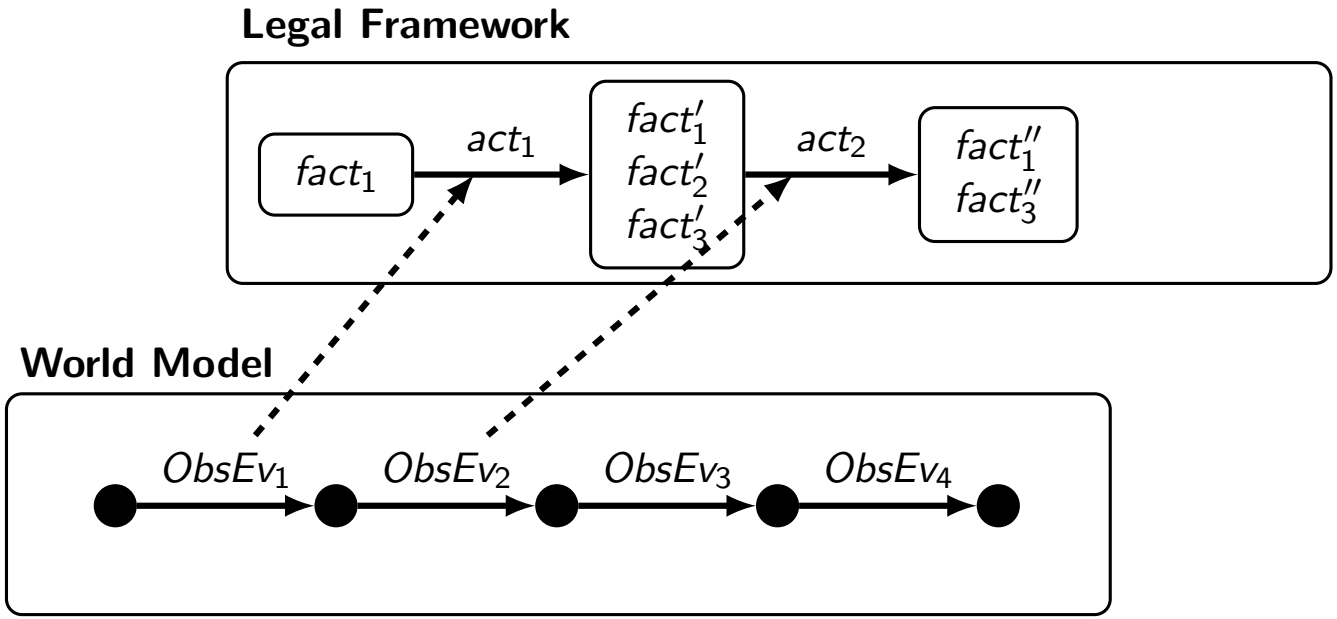
Model sketch



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

Legal Framework: $\mathcal{L} := \langle \mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{C}, \Delta \rangle$

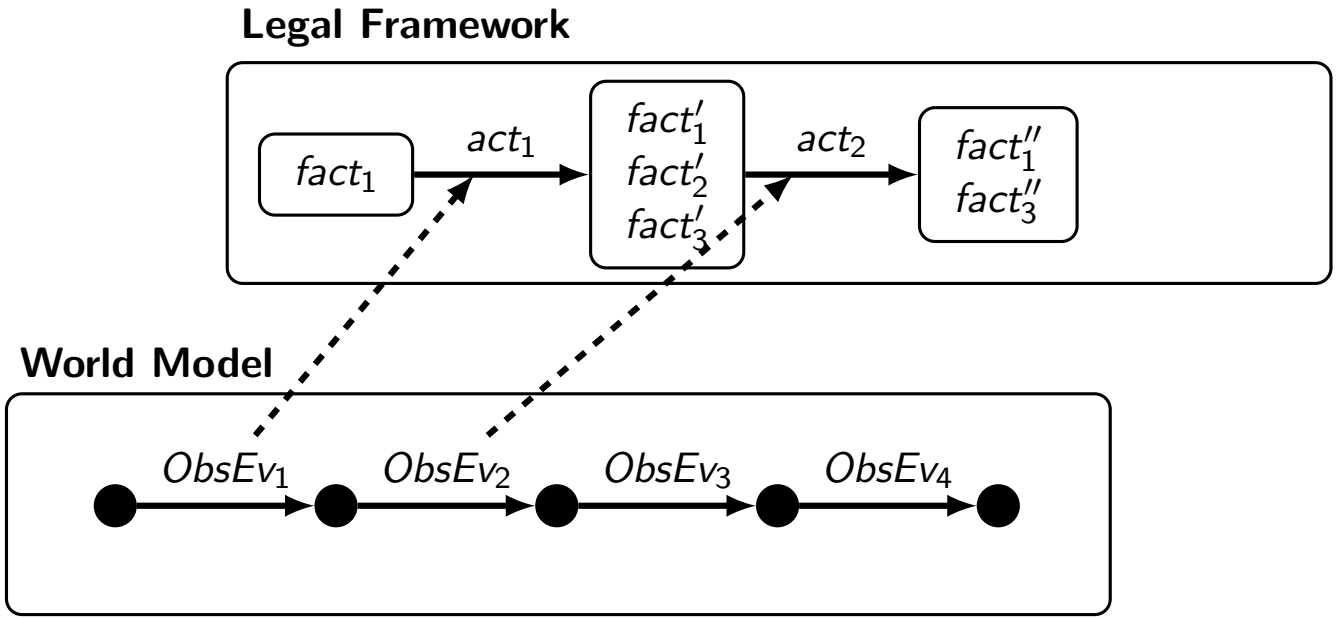
Model sketch



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

Legal Framework: $\mathcal{L} := \langle \mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{C}, \Delta \rangle$

Model sketch



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

Legal Framework: $\mathcal{L} := \langle \mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{C}, \Delta \rangle$

Framework translation to *AnsProlog*

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

1. Motivation
2. Legal Frameworks

Comparative Legal Framework

- Comparative Legal Framework $C_{\mathcal{I}}$ with $\mathcal{L} = \{\mathcal{L}_1, \ldots, \mathcal{L}_n\}$
- Wrapper
- Individual states
- Comparative trace = sequence of exogenous events from all individual legal frameworks

Comparative Legal Framework

- Comparative Legal Framework $C_{\mathcal{I}}$ with $\mathcal{L} = \{\mathcal{L}_1, \ldots, \mathcal{L}_n\}$
- Wrapper
- Individual states
- Comparative trace = sequence of exogenous events from all individual legal frameworks

Comparative Legal Framework

- Comparative Legal Framework $C_{\mathcal{I}}$ with $\mathcal{L} = \{\mathcal{L}_1, \ldots, \mathcal{L}_n\}$
- Wrapper
- Individual states
- Comparative trace = sequence of exogenous events from all individual legal frameworks

Comparative Legal Framework

- Comparative Legal Framework $C_{\mathcal{I}}$ with $\mathcal{L} = \{\mathcal{L}_1, \ldots, \mathcal{L}_n\}$
- Wrapper
- Individual states
- Comparative trace = sequence of exogenous events from all individual legal frameworks

Synchronised Traces

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

Synchronised Traces

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

Synchronised Traces

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

Synchronised Traces

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

Synchronised Traces

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

Conflicts

- Given a comparative trace
- a conflict occurs for any two legal frameworks of $C_{\mathcal{I}}$ if
- 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 comparative trace
- a conflict occurs for any two legal frameworks of $C_{\mathcal{I}}$ if
- 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 comparative trace
- a conflict occurs for any two legal frameworks of $C_{\mathcal{I}}$ if
- 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 comparative trace
- a conflict occurs for any two legal frameworks of $C_{\mathcal{I}}$ if
- 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 comparative trace
- a conflict occurs for any two legal frameworks of $C_{\mathcal{I}}$ if
- 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 comparative trace
- a conflict occurs for any two legal frameworks of $C_{\mathcal{I}}$ if
- 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 comparative trace
- a conflict occurs for any two legal frameworks of $C_{\mathcal{I}}$ if
- a fluent known to both appears
- positively in one and
- negatively in the other
- at the same point of time
- Conflict traces

1. Motivation
2. Legal Frameworks

Renaming

- Computing conflict traces
- Not sufficient to add *AnsProlog* program together
- Contradiction results in no answer set sets
- Rename each legal framework to make the unique
- `rename(F,FInst,Inst)`

Renaming

- Computing conflict traces
- Not sufficient to add *AnsProlog* program together
- Contradiction results in no answer set sets
- Rename each legal framework to make the unique
- `rename(F,FInst,Inst)`

Renaming

- Computing conflict traces
- Not sufficient to add *AnsProlog* program together
- Contradiction results in no answer set sets
- Rename each legal framework to make the unique
- `rename(F,FInst,Inst)`

Renaming

- Computing conflict traces
- Not sufficient to add *AnsProlog* program together
- Contradiction results in no answer set sets
- Rename each legal framework to make the unique
- `rename(F,FInst,Inst)`

Renaming

- Computing conflict traces
- Not sufficient to add *AnsProlog* program together
- Contradiction results in no answer set sets
- Rename each legal framework to make the unique
- `rename(F, FInst, Inst)`

Conflict traces

- Conflicts occur when two legal frameworks disagree on a fluent known to both of them

```
conflict :- holdsat(FInX, I), not holdsat(FInY, I),
            rename(F, FInX, InX), rename(F, FInY, InY),
            ifluent(FInX), ifluent(FInY),
            instant(I), inst(InX; InY).
:- not conflict.
conflict(FInX, FInY, I) :- holdsat(FInX, I), not holdsat(FInY, I),
            rename(F, FInX, InX), rename(F, FInY, InY),
            ifluent(FInX), ifluent(FInY),
            instant(I), inst(InX; InY).
```

Conflict traces

- Conflicts occur when two legal frameworks disagree on a fluent known to both of them

```
conflict :- holdsat(FInX, I), not holdsat(FInY, I),
            rename(F, FInX, InX), rename(F, FInY, InY),
            ifluent(FInX), ifluent(FInY),
            instant(I), inst(InX; InY).
:- not conflict.
conflict(FInX, FInY, I) :- holdsat(FInX, I), not holdsat(FInY, I),
            rename(F, FInX, InX), rename(F, FInY, InY),
            ifluent(FInX), ifluent(FInY),
            instant(I), inst(InX; InY).
```

Conflict traces

- Conflicts occur when two legal frameworks disagree on a fluent known to both of them

```
conflict :- holdsat(FInX, I), not holdsat(FInY, I),
            rename(F, FInX, InX), rename(F, FInY, InY),
            ifluent(FInX), ifluent(FInY),
            instant(I), inst(InX; InY).
:- not conflict.
conflict(FInX, FInY, I) :- holdsat(FInX, I), not holdsat(FInY, I),
            rename(F, FInX, InX), rename(F, FInY, InY),
            ifluent(FInX), ifluent(FInY),
            instant(I), inst(InX; InY).
```

1. Motivation
2. Legal Frameworks

Future Work

Future Work

- Extension Conflict Definition
- Conflict Resolution
- Ontology
- Interfaces & Tools

Future Work

Future Work

- Extension Conflict Definition
- Conflict Resolution
- Ontology
- Interfaces & Tools

Future Work

Future Work

- Extension Conflict Definition
- Conflict Resolution
- Ontology
- Interfaces & Tools

Future Work

Future Work

- Extension Conflict Definition
- Conflict Resolution
- Ontology
- Interfaces & Tools

Thank you for your attention!
Any questions?