A Model-based Approach to the Automatic Revision of Secondary Legislation

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

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

The 14th International Conference on AI and Law (ICAIL 2013)







1 Motivation

2 Modelling of Legal Specification

3 Automatic Conflict Detection

4 Conflict Resolution via Automatic Revision

5 Future Work

Conflict of Laws

- Law is dynamic: adding new laws or amending existing ones.
- Existing inferior law (or set of regulations) may no longer be consistent with the new superior law: *an action permitted(obliged) in one but prohibited in another.*
- possibly resulting in (unintended) illegal behaviour.
- **Problem**: detection and resolution of such conflicts automatically.
- In particular: Will a specific scenario (a course of actions) result in a conflict? How to revise existing laws to remove this conflict?

Conflict of Laws

- Law is dynamic: adding new laws or amending existing ones.
- Existing inferior law (or set of regulations) may no longer be consistent with the new superior law: *an action permitted(obliged) in one but prohibited in another.*
- possibly resulting in (unintended) illegal behaviour.
- **Problem**: detection and resolution of such conflicts automatically.
- In particular: Will a specific scenario (a course of actions) result in a conflict? How to revise existing laws to remove this conflict?

Case Study

- Recent change in UK immigration law about student visa regulation [1].
- **UK Immigration Law** : the permitted working hours of overseas students are reduced: *up to 20 hours per week*.
- **Studentship Regulations**: minimum number of teaching hours the student has to deliver: *at least 30 hours per week*.

[1] Detailed changes can be found on the UK Home Office website under `http://www.ukba.homeoffice.gov.uk/sitecontent/documents/news/sop4.pdf`.

Case Study

- Recent change in UK immigration law about student visa regulation [1].
- **UK Immigration Law** : the permitted working hours of overseas students are reduced: *up to 20 hours per week*.
- **Studentship Regulations**: minimum number of teaching hours the student has to deliver: *at least 30 hours per week*.

[1] Detailed changes can be found on the UK Home Office website under `http://www.ukba.homeoffice.gov.uk/sitecontent/documents/news/sop4.pdf`.

1 Motivation

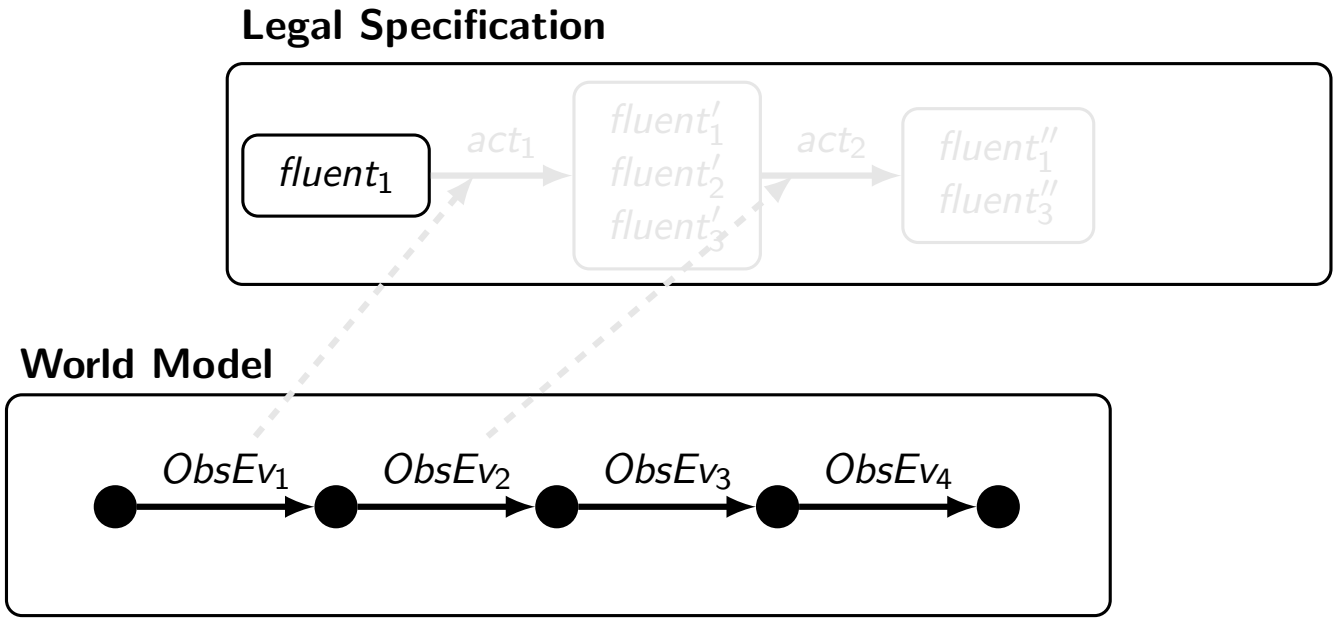
2 Modelling of Legal Specification
- Formal Model Sketch
- Mapping to *AnsProlog*

3 Automatic Conflict Detection

4 Conflict Resolution via Automatic Revision

5 Future Work

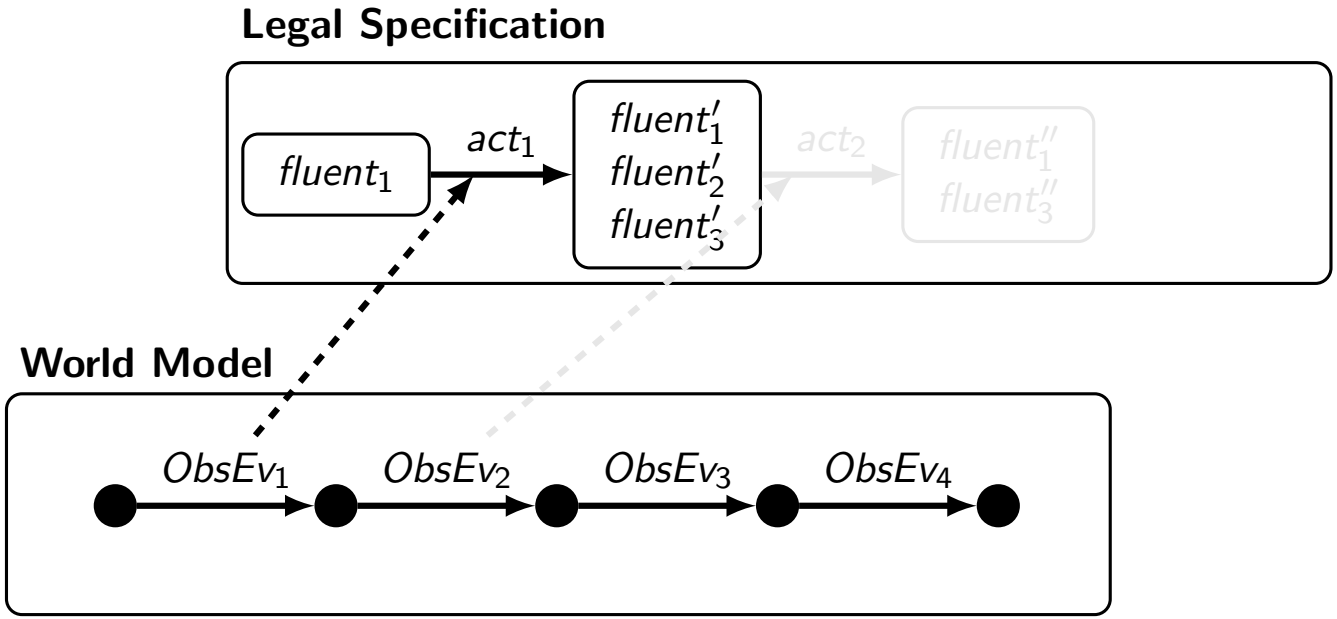
Model sketch



- Model generates **ordered traces** that show us the evolution of the legal specification over time—allows validation and verification.
- Essential elements of model are:

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

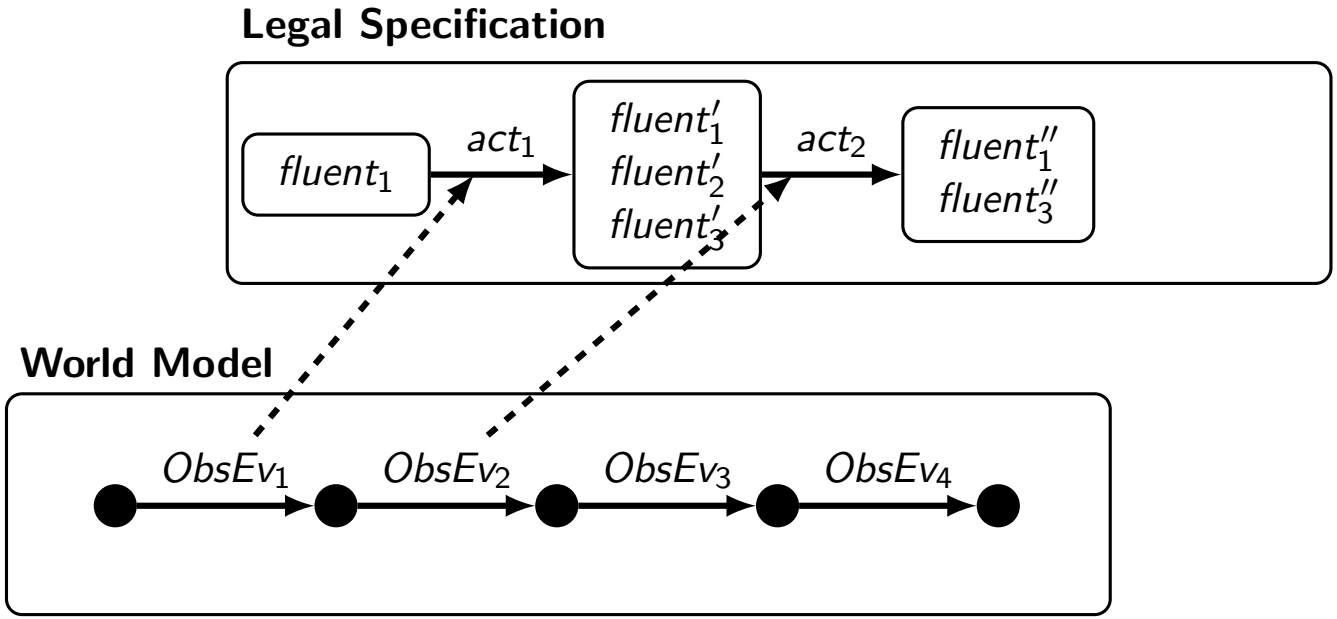
Model sketch



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

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

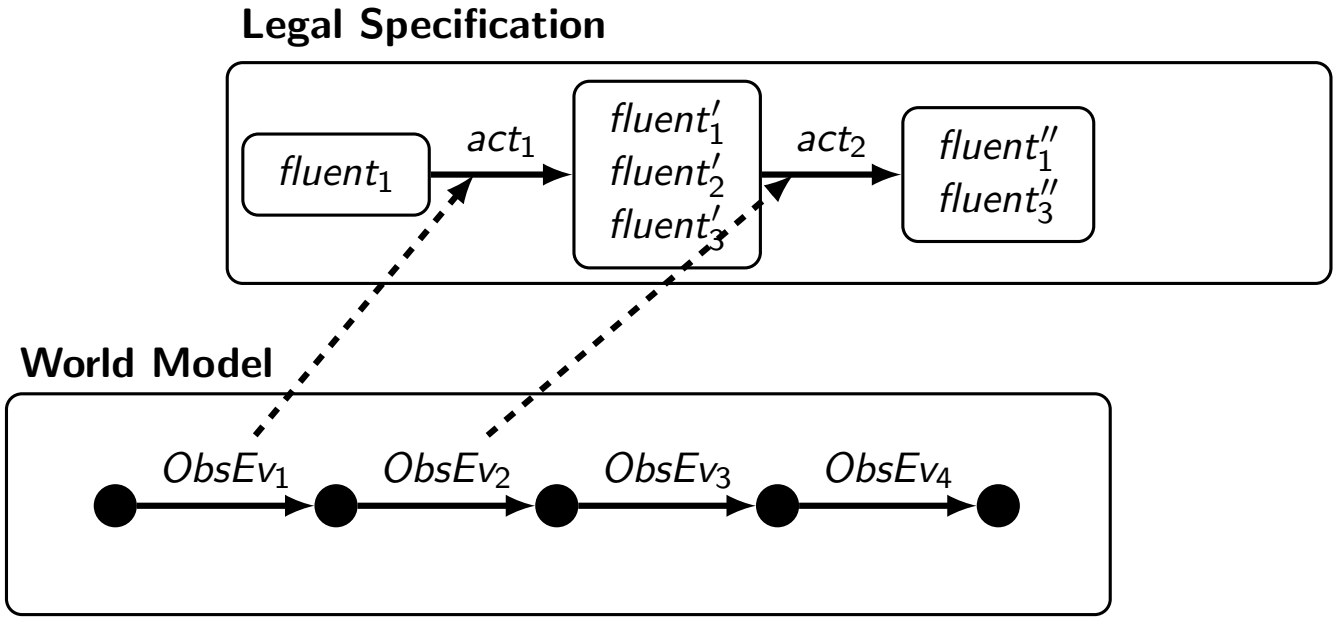
Model sketch



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

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

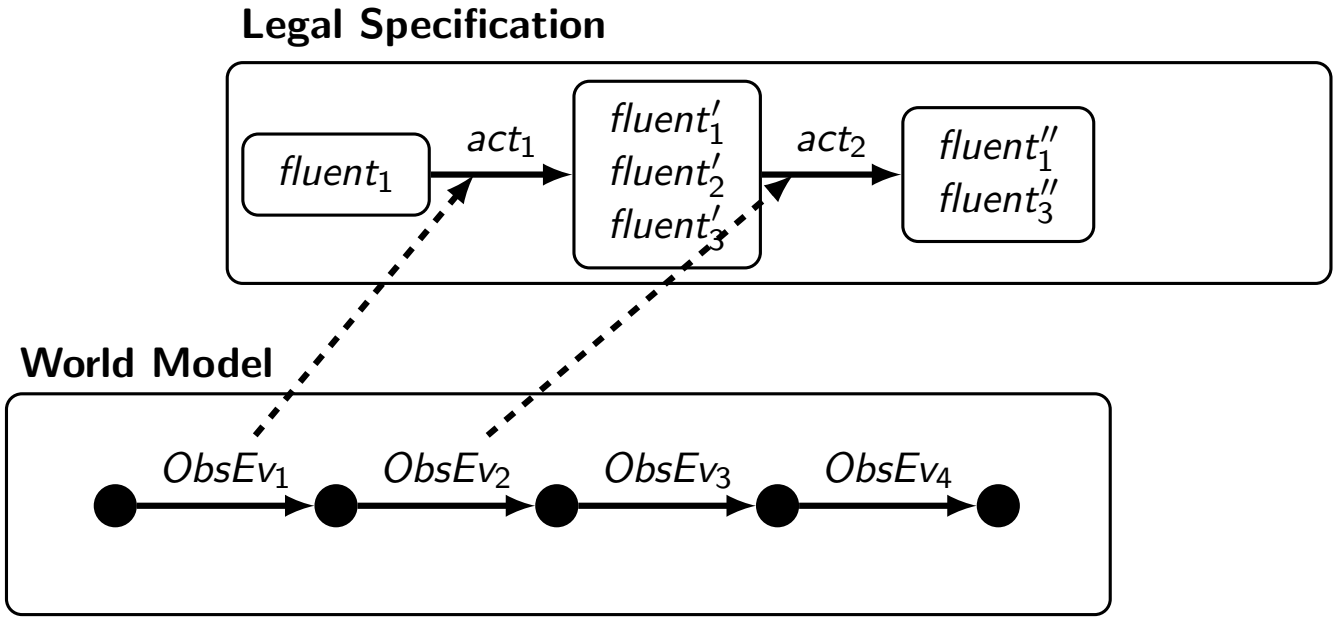
Model sketch



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

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

Model sketch



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

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

Framework translation to *AnsProlog*

- Answer set programming used as computational back-end
- *Conclusion* : −*Conditions*. with use of negation as failure
- Important components:
  - `observed(Event,Instant)`: an exogenous event at time t
  - `occurred(Event,Instant)`: a legal action at time t
  - `holdsat(Fluent,Instant)`: fluent is true at time t
  - `pow(Event)`,`perm(Event)`,`obl(Event,DueEvent,VioEvent)`: 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 action at time t
  - `holdsat(Fluent,Instant)`: fluent is true at time t
  - `pow(Event)`,`perm(Event)`,`obl(Event,DueEvent,VioEvent)`: 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 action at time t
  - `holdsat(Fluent,Instant)`: fluent is true at time t
  - `pow(Event)`,`perm(Event)`,`obl(Event,DueEvent,VioEvent)`: 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 Modelling of Legal Specification

3 Automatic Conflict Detection

4 Conflict Resolution via Automatic Revision

5 Future Work

Composite Legal Specification

- Composite Legal Specification $C_{\mathcal{L}}$ with $\mathcal{L} = \{\mathcal{L}_1, \ldots, \mathcal{L}_n\}$
- Wrapper for independent legal specification. Individual state transition.
- Composite trace = sequence of exogenous events from all individual legal specification.
- Synchronised Traces: Null events (`enull`) to fill up the unrecognised events in individual traces. DOES NOT change the states.

Composite Legal Specification

- Composite Legal Specification $C_{\mathcal{L}}$ with $\mathcal{L} = \{\mathcal{L}_1, \ldots, \mathcal{L}_n\}$
- Wrapper for independent legal specification. Individual state transition.
- Composite trace = sequence of exogenous events from all individual legal specification.
- Synchronised Traces: Null events (`enull`) to fill up the unrecognised events in individual traces. DOES NOT change the states.

Composite Legal Specification

- Composite Legal Specification $C_{\mathcal{L}}$ with $\mathcal{L} = \{\mathcal{L}_1, \ldots, \mathcal{L}_n\}$
- Wrapper for independent legal specification. Individual state transition.
- Composite trace = sequence of exogenous events from all individual legal specification.
- Synchronised Traces: Null events (`enull`) to fill up the unrecognised events in individual traces. DOES NOT change the states.

Conflict traces

- Conflicts occur when any two legal specifications disagree on a fluent known to both of them at the same time instant:

$$\begin{aligned}\texttt{conflict}(\texttt{FX},\texttt{FY},\texttt{I}) :- \texttt{holdsat}(\texttt{FX},\texttt{I}),\ &\textbf{not}\ \texttt{holdsat}(\texttt{FY},\texttt{I}),\\ &\texttt{rename}(\texttt{F},\texttt{FX},\texttt{X}),\texttt{rename}(\texttt{F},\texttt{FY},\texttt{Y}),\\ &\texttt{ifluent}(\texttt{FX}),\texttt{ifluent}(\texttt{FY}),\\ &\texttt{instant}(\texttt{I}),\texttt{inst}(\texttt{X};\texttt{Y}).\end{aligned}$$

- Renaming: Contradiction results in no answer set.

Conflict traces

- Conflicts occur when any two legal specifications disagree on a fluent known to both of them at the same time instant:

$$\begin{aligned}\texttt{conflict(FX, FY, I)} :- &\texttt{holdsat(FX, I)}, \textbf{not } \texttt{holdsat(FY, I)},\\ &\texttt{rename(F, FX, X)}, \texttt{rename(F, FY, Y)},\\ &\texttt{ifluent(FX)}, \texttt{ifluent(FY)},\\ &\texttt{instant(I)}, \texttt{inst(X; Y)}.\end{aligned}$$

- Renaming: Contradiction results in no answer set.

Conflicts Detection

$$CTR = \langle ..., sendOfferLetter, acceptOffer, askStudentship, applyVisa, ... \rangle$$

University Regulations:

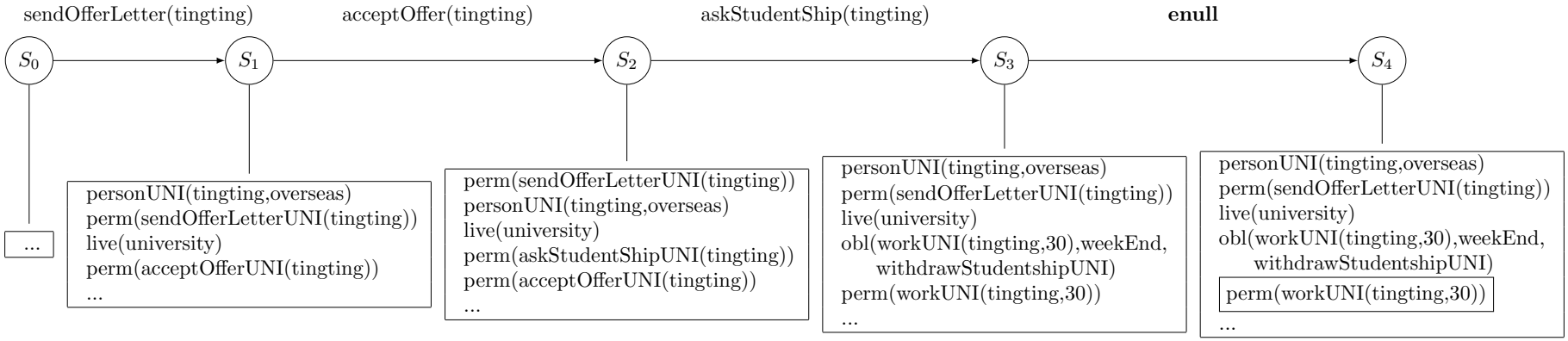
UK Immigration Law:

$$\texttt{conflict}(\texttt{perm}(\texttt{workUNI}(\texttt{tingting}, 30)), \texttt{permVI}(\texttt{tingting}), 30), 4)$$
$$\texttt{conflict}(\texttt{perm}(\texttt{workVI}(\texttt{tingting}, 20)), \texttt{permUNI}(\texttt{tingting}), 20), 4)$$

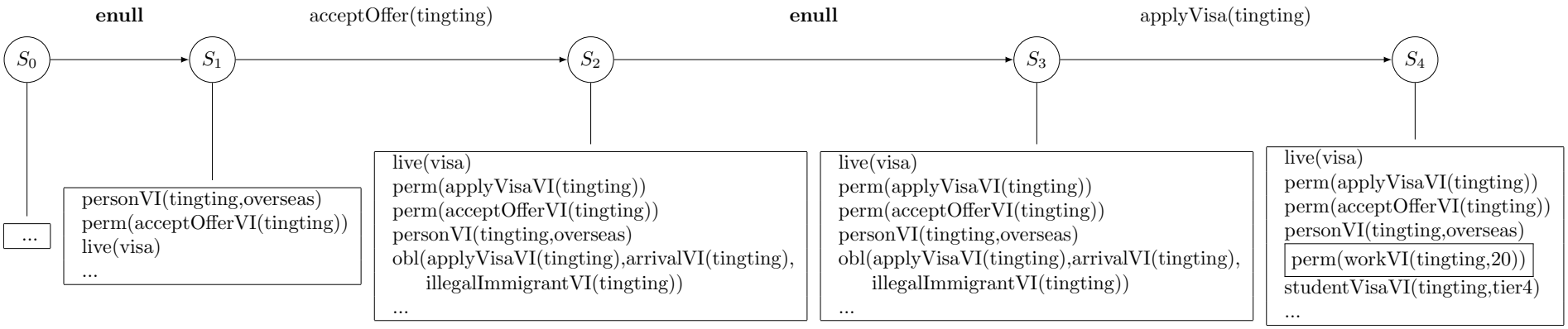
Conflicts Detection

$$CTR = \langle ..., sendOfferLetter, acceptOffer, askStudentship, applyVisa, ... \rangle$$

University Regulations:



UK Immigration Law:

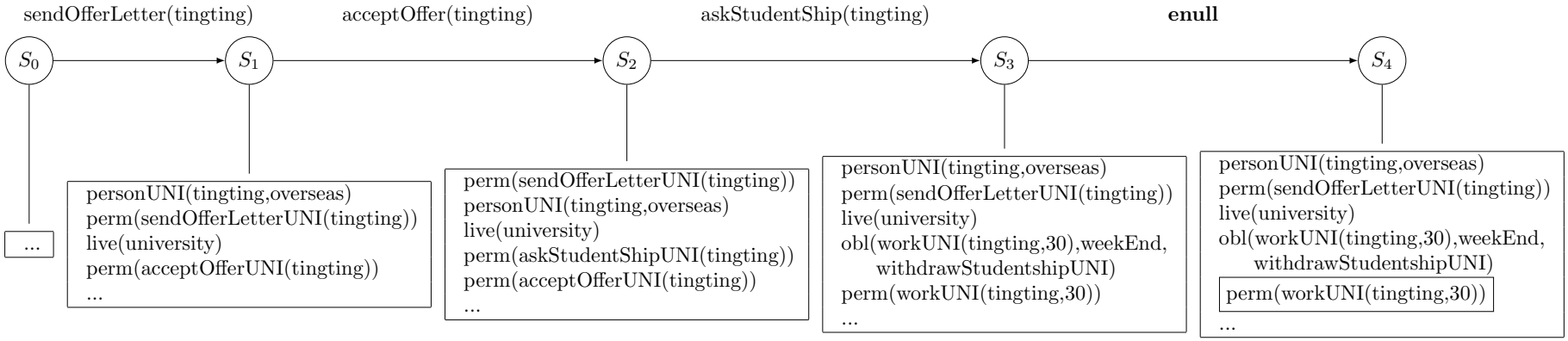


$$conflict(perm(workUNI(tingting, 30)), permVI(tingting), 30), 4)$$
$$conflict(perm(workVI(tingting, 20)), permUNI(tingting), 20), 4)$$

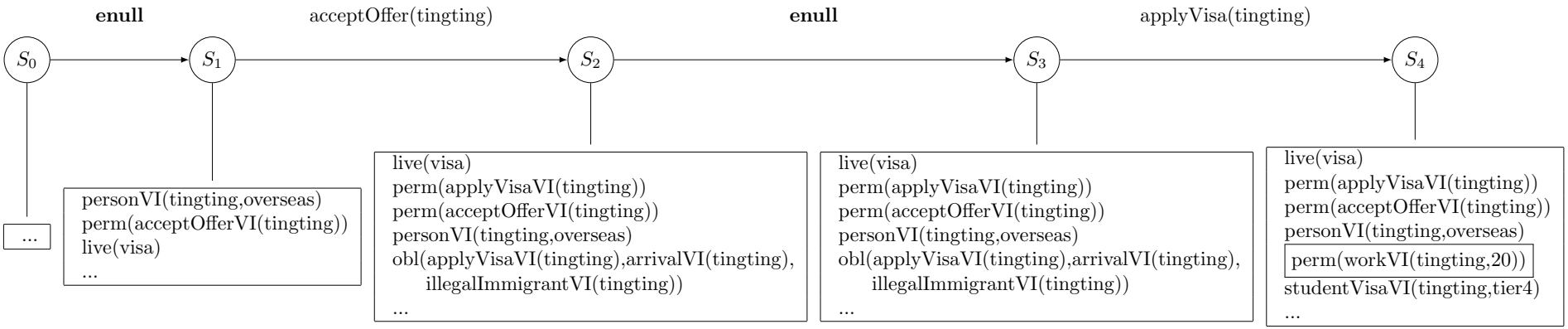
Conflicts Detection

$$CTR = \langle ..., sendOfferLetter, acceptOffer, askStudentship, applyVisa, ... \rangle$$

University Regulations:



UK Immigration Law:



$$\texttt{conflict}(\texttt{perm}(\texttt{workUNI}(\texttt{tingting}, 30)), \texttt{permVI}(\texttt{tingting}), 30), 4)$$
$$\texttt{conflict}(\texttt{perm}(\texttt{workVI}(\texttt{tingting}, 20)), \texttt{permUNI}(\texttt{tingting}), 20), 4)$$

1 Motivation

2 Modelling of Legal Specification

3 Automatic Conflict Detection

4 Conflict Resolution via Automatic Revision

5 Future Work

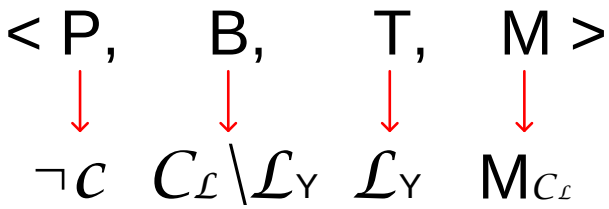
Conflict Resolution via Theory Revision[2]

- Theory Revision Task is characterized by a tuple $\langle P, B, T, M \rangle$ and T', called *revised theory*, is a *solution* to the task with cost $c(T, T')$, iff
  1. $T' \subseteq s(M)$,
  2. $B \cup T' \models P$,
  3. $c(T, T')$ is minimal.
- Given a trace CTR which admits a conflict $c = conflict(FX, FY, I)$ and assume $\mathcal{L}_X \succ \mathcal{L}_Y$:
- Inductive Logic Programming produces all possible revisions $\mathcal{L}'_Y$: $\mathcal{L}'_Y \cup (C_{\mathcal{L}} \setminus \mathcal{L}_Y) \cup CTR \models \neg c$

[2] Corapi, Domenico, et al. "Normative design using inductive learning." Theory and Practice of Logic Programming 11.4-5 (2011): 783-799.

Conflict Resolution via Theory Revision[2]

- Theory Revision Task is characterized by a tuple $\langle P, B, T, M \rangle$ and T', called *revised theory*, is a *solution* to the task with cost $c(T, T')$, iff
  1. $T' \subseteq s(M)$,
  2. $B \cup T' \models P$,
  3. $c(T, T')$ is minimal.
- Given a trace CTR which admits a conflict $c = conflict(FX, FY, I)$ and assume $\mathcal{L}_X \succ \mathcal{L}_Y$:



- Inductive Logic Programming produces all possible revisions $\mathcal{L}'_Y$: $\mathcal{L}'_Y \cup (C_{\mathcal{L}} \setminus \mathcal{L}_Y) \cup CTR \models \neg c$

[2] Corapi, Domenico, et al. "Normative design using inductive learning." Theory and Practice of Logic Programming 11.4-5 (2011): 783-799.

Conflict Resolution via Theory Revision[2]

- Theory Revision Task is characterized by a tuple $\langle P, B, T, M \rangle$ and T', called *revised theory*, is a *solution* to the task with cost $c(T, T')$, iff
  1. $T' \subseteq s(M)$,
  2. $B \cup T' \models P$,
  3. $c(T, T')$ is minimal.
- Given a trace CTR which admits a conflict $c = conflict(FX, FY, I)$ and assume $\mathcal{L}_X \succ \mathcal{L}_Y$:

$$< \mathrm{P}, \quad \mathrm{B}, \quad \mathrm{T}, \quad \mathrm{M} >$$
$$\downarrow \quad \downarrow \quad \downarrow \quad \downarrow$$
$$\neg c \quad C_{\mathcal{L}} \backslash \mathcal{L}_{\mathrm{Y}} \quad \mathcal{L}_{\mathrm{Y}} \quad \mathrm{M}_{C_{\mathcal{L}}}$$

- Inductive Logic Programming produces all possible revisions $\mathcal{L}'_Y$:

$$\mathcal{L}'_Y \cup (C_{\mathcal{L}} \setminus \mathcal{L}_Y) \cup CTR \models \neg c$$

[2] Corapi, Domenico, et al. "Normative design using inductive learning." Theory and Practice of Logic Programming 11.4-5 (2011): 783-799.

Case Study

Rule 2 : $\mathtt{initiated}(\mathtt{perm}(\mathtt{work}(\mathtt{Student}, 30)), \mathtt{I}) :-$
$\mathtt{occurred}(\mathtt{askStudentShip}(\mathtt{Student}), \mathtt{I}),$
$\mathtt{holdsat}(\mathtt{availability}, \mathtt{I}),$
$\mathbf{not}\ \mathtt{holdsat}(\mathtt{person}(\mathtt{Student}, \mathtt{overseas}), \mathtt{I}).$

Rule 3 : $\mathtt{initiated}(\mathtt{obl}(\mathtt{work}(\mathtt{Student}, 30), \mathtt{weekEnd},$
$\mathtt{withdrawStudentship}), \mathtt{I}) :-$
$\mathtt{occurred}(\mathtt{askStudentShip}(\mathtt{Student}), \mathtt{I}),$
$\mathtt{holdsat}(\mathtt{availability}, \mathtt{I}),$
$\mathbf{not}\ \mathtt{holdsat}(\mathtt{person}(\mathtt{Student}, \mathtt{overseas}), \mathtt{I}).$

Case Study

Rule 2 : $\mathtt{initiated}(\mathtt{perm}(\mathtt{work}(\mathtt{Student}, 30)), \mathtt{I}) :-$
$\mathtt{occurred}(\mathtt{askStudentShip}(\mathtt{Student}), \mathtt{I}),$
$\mathtt{holdsat}(\mathtt{availability}, \mathtt{I}),$
not $\mathtt{holdsat}(\mathtt{person}(\mathtt{Student}, \mathtt{overseas}), \mathtt{I}).$

Rule 3 : $\mathtt{initiated}(\mathtt{obl}(\mathtt{work}(\mathtt{Student}, 30), \mathtt{weekEnd},$
$\mathtt{withdrawStudentship}), \mathtt{I}) :-$
$\mathtt{occurred}(\mathtt{askStudentShip}(\mathtt{Student}), \mathtt{I}),$
$\mathtt{holdsat}(\mathtt{availability}, \mathtt{I}),$
not $\mathtt{holdsat}(\mathtt{person}(\mathtt{Student}, \mathtt{overseas}), \mathtt{I}).$

1 Motivation

2 Modelling of Legal Specification

3 Automatic Conflict Detection

4 Conflict Resolution via Automatic Revision

5 Future Work

Future Work

Future Work

- Measurement Mechanism of Revision Cost
- Ontology Alignment
- Interfaces & Tools

Thank you for your attention!
Any questions?