# Normative Conflict Detection and Resolution in Cooperating Institutions

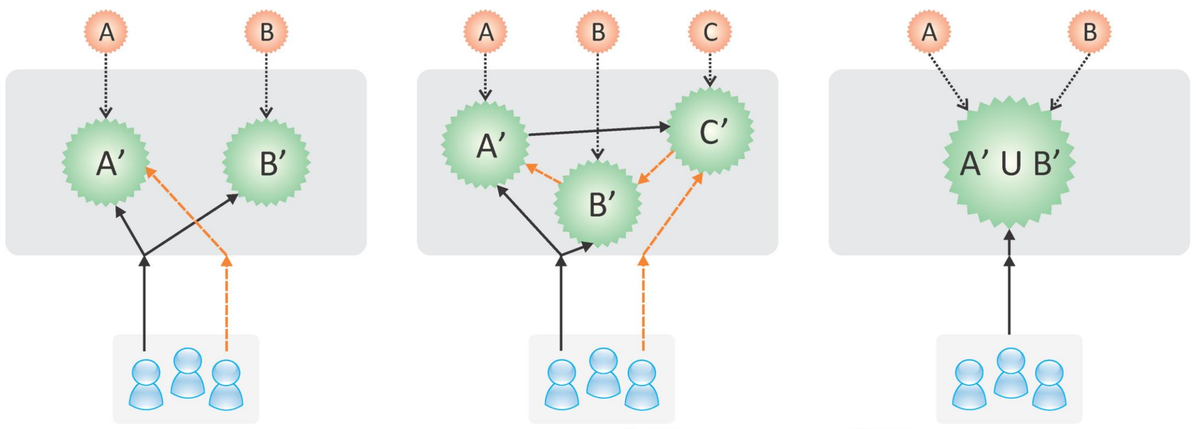
*Tingting Li*
*t.li@cs.bath.ac.uk*
*Department of Computer Science, University of Bath, UK*

## Abstract

***Institutions*** provide an effective mechanism to govern agents in open distributed systems by specifying a set of ***norms*** (in terms of permissions, empowerments and obligations) regarding certain goals. However, when several institutions have to cooperate to govern the same entities simultaneously, ***norm conflicts*** are very likely to occur because institutions are typically designed independently with different goals. In this thesis, we aim to:

- identify the different ways to combine institutions.
- model those ways formally and computationally.
- ***detect conflicts*** in different combinations automatically.
- ***resolve those conflicts*** via automatic ***norm revision*** using an approach based on ***inductive learning***.

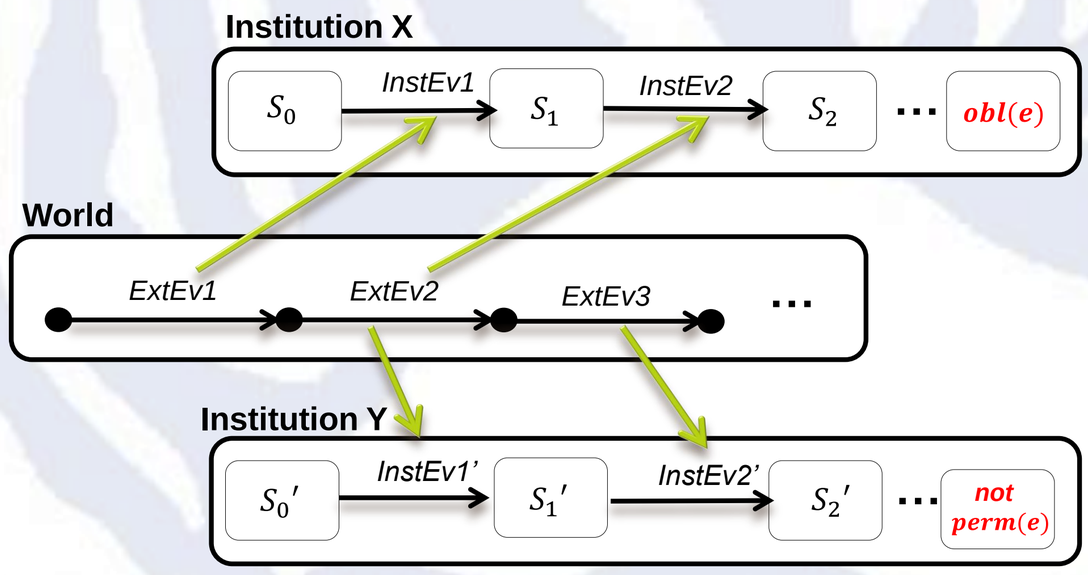
## Cooperating Institutions




### *Three Forms of Cooperation*

**(a) Composite Institution:** a set of peer institutions combined together to form a common governance scope.
**(b) Multi-institution**: a hierarchical structure of interlinked institutions, in which an institution might be governed or influenced by other institutions.
**(c) Merged institution:** all the norms of each institutions are merged to form a completely new institution.

## Normative Conflicts




- formalize institutions: event-driven action language ***InstAL*** [1].
- translate to computational model based on ***Answer Set Programming.***
- automatic procedure to detect conflicting normative positions: ***permission vs. prohibition, obligation vs. prohibition.***
- examine conflicts in all three combinations differently.

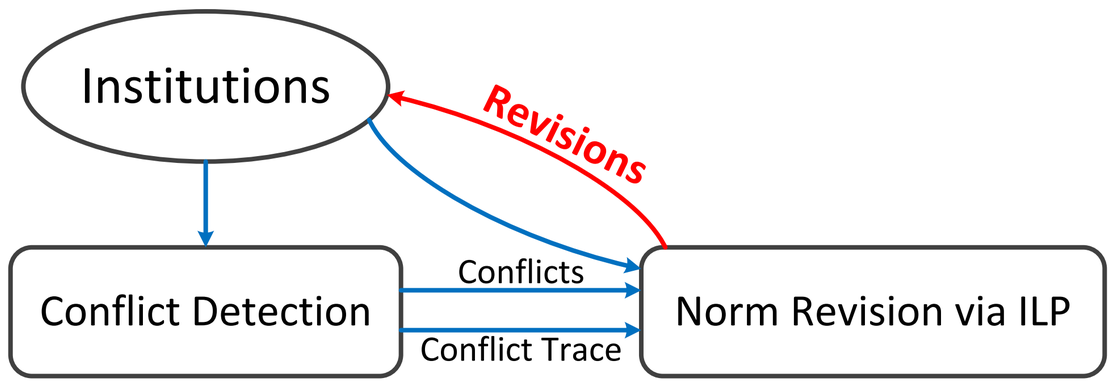
## Conflict Detection Mechanism

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

```
conflict(InstX, InstY,I, F):-
            holdsat(FX, I), not holdsat(FY, I),
            rename(F, FX, InstX), rename(F, FY, InstY),
            ifluent(FX), ifluent(FY),
            instant(I), inst(InstX; InstY):
```

## Conflict Resolution Mechanism

- input conflicts and conflict traces.
- automatic norm revision via ***Inductive Logic Programming (ILP***) [2].
- generate revision suggestions to resolve conflicts.
- select revision that minimizes the number of changes.




## Example

***UK Immigration Law*** *: the permitted working hours of overseas students are reduced to 20 hours per week.*
***Studentship Regulations****: minimum number of teaching hours the student has to deliver is 30 hours per week.*

***Studentship Regulations:***

```
initiated(obl(work(Student, 30), I) :-
        occurred(askForStudentShip(Student), I),
        holdsat(availability, I),
        not holdsat(person(Student, overseas), I).
```

- Conflicts arise for overseas students who are also granted by a studentship.
- ***Red part*** is suggested by the conflict resolution system.
- More details about this example can be found in [3].

For a digital copy of this poster, please scan this QR Code.

## *References*

**[1]** O. Cliffe, M. De Vos, and J. A. Padget. Answer set programming for representing and reasoning about virtual institutions. Computational Logic in Multi-Agent Systems, volume 4371 of LNCS, pages 60–79. Springer, 2007.
**[2]** D. Corapi, A. Russo, M. De Vos, J. A. Padget, and K. Satoh. Normative Design using Inductive Learning. TPLP, 11(4-5):783–799, 2011.
**[3]** T. Li, T. Balke, M. De Vos, J. A. Padget, and K. Satoh. A Model-based Approach to the Automatic Revision of Secondary Legislation. Proceedings of the 14th International Conference on AI and Law, pages 202-206, ACM, 2013.