Legal Conflict Detection in Interacting Legal Systems

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

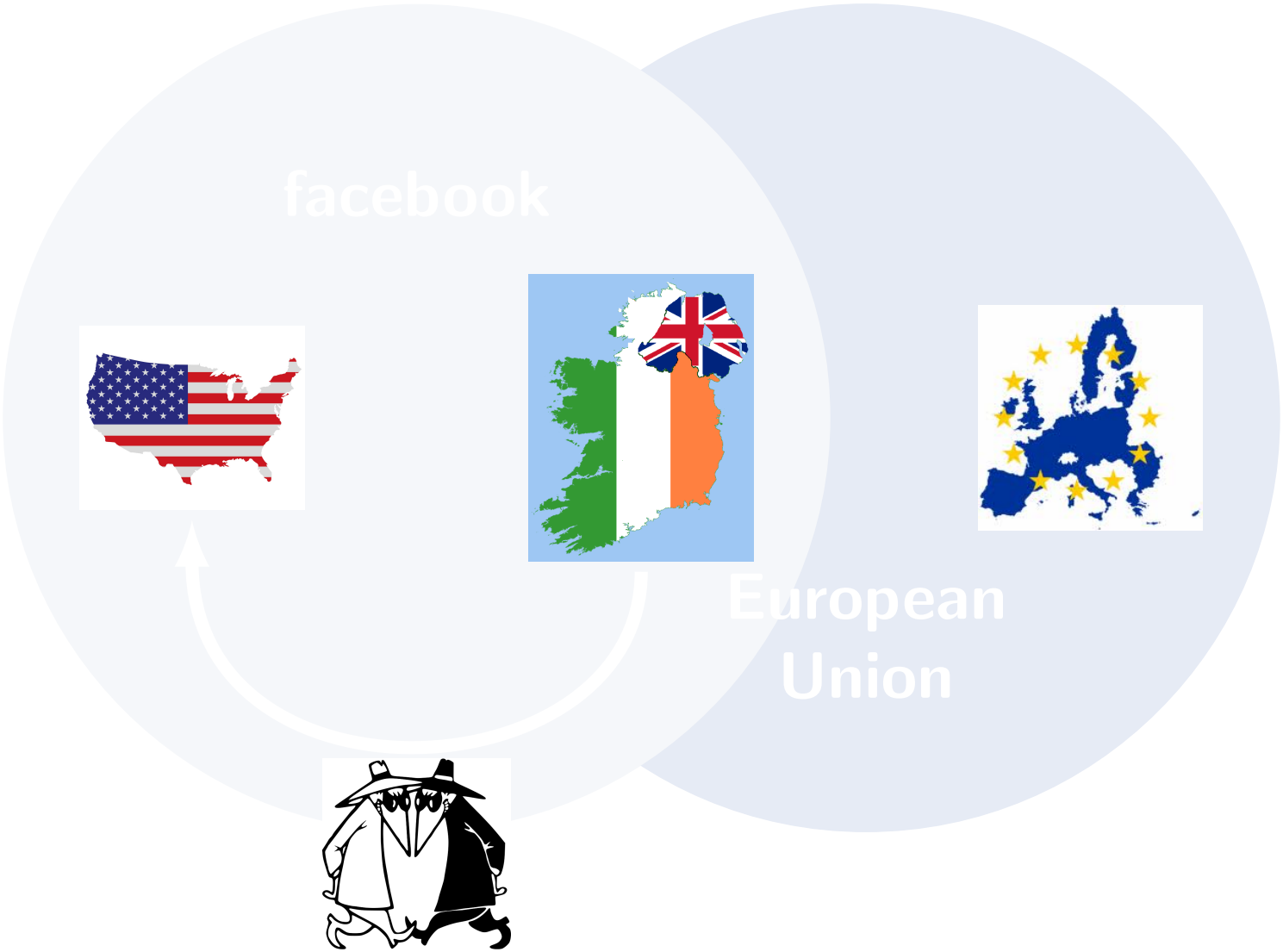
[1]University of Bath
{t.li,mdv,jap}@cs.bath.ac.uk

[2]University of Surrey
t.balke@surrey.ac.uk

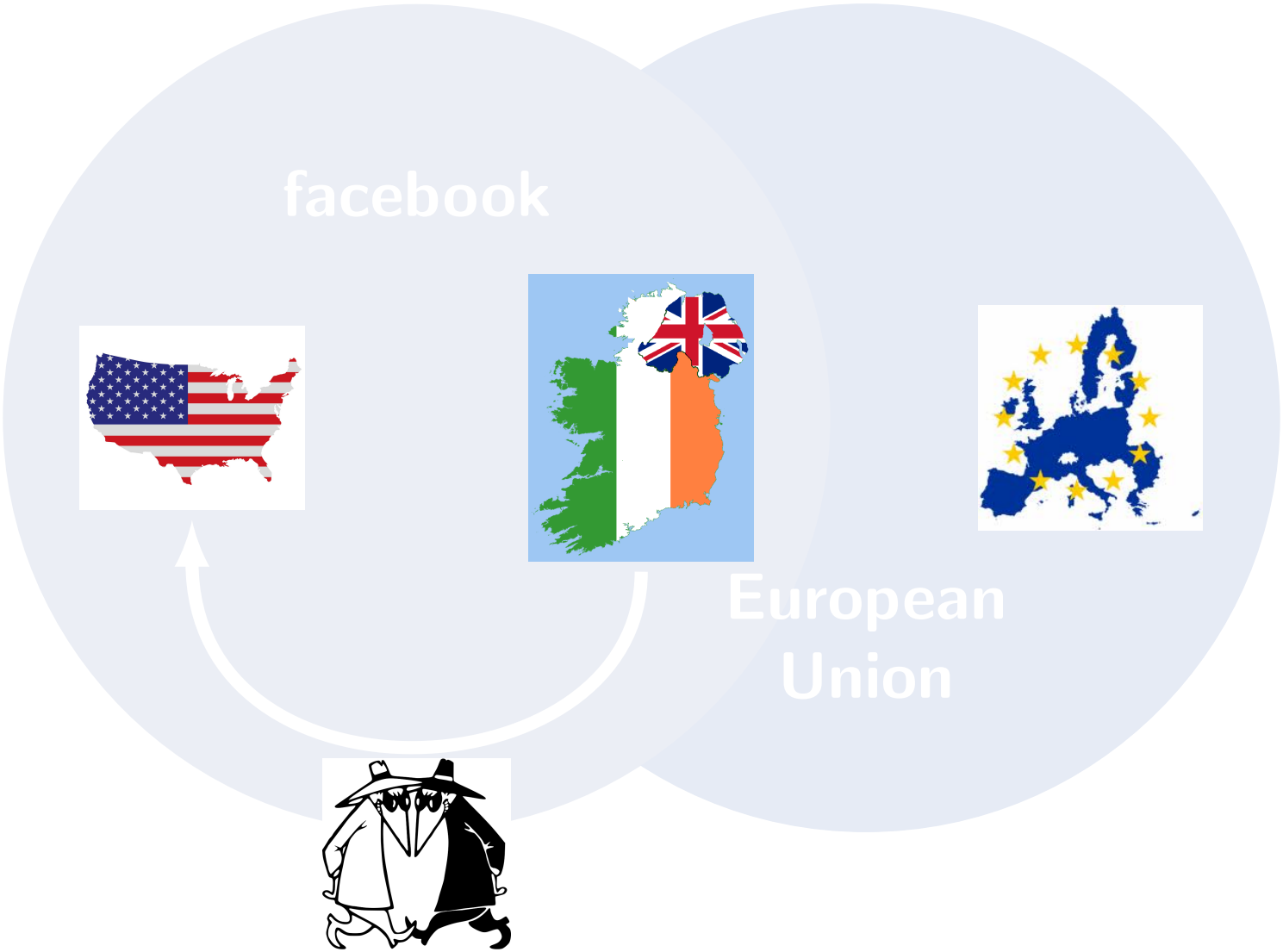
[3]National Institute of Informatics
ksatoh@nii.ac.jp

The 26th International Conference on Legal Knowledge and Information Systems (JURIX 2013)

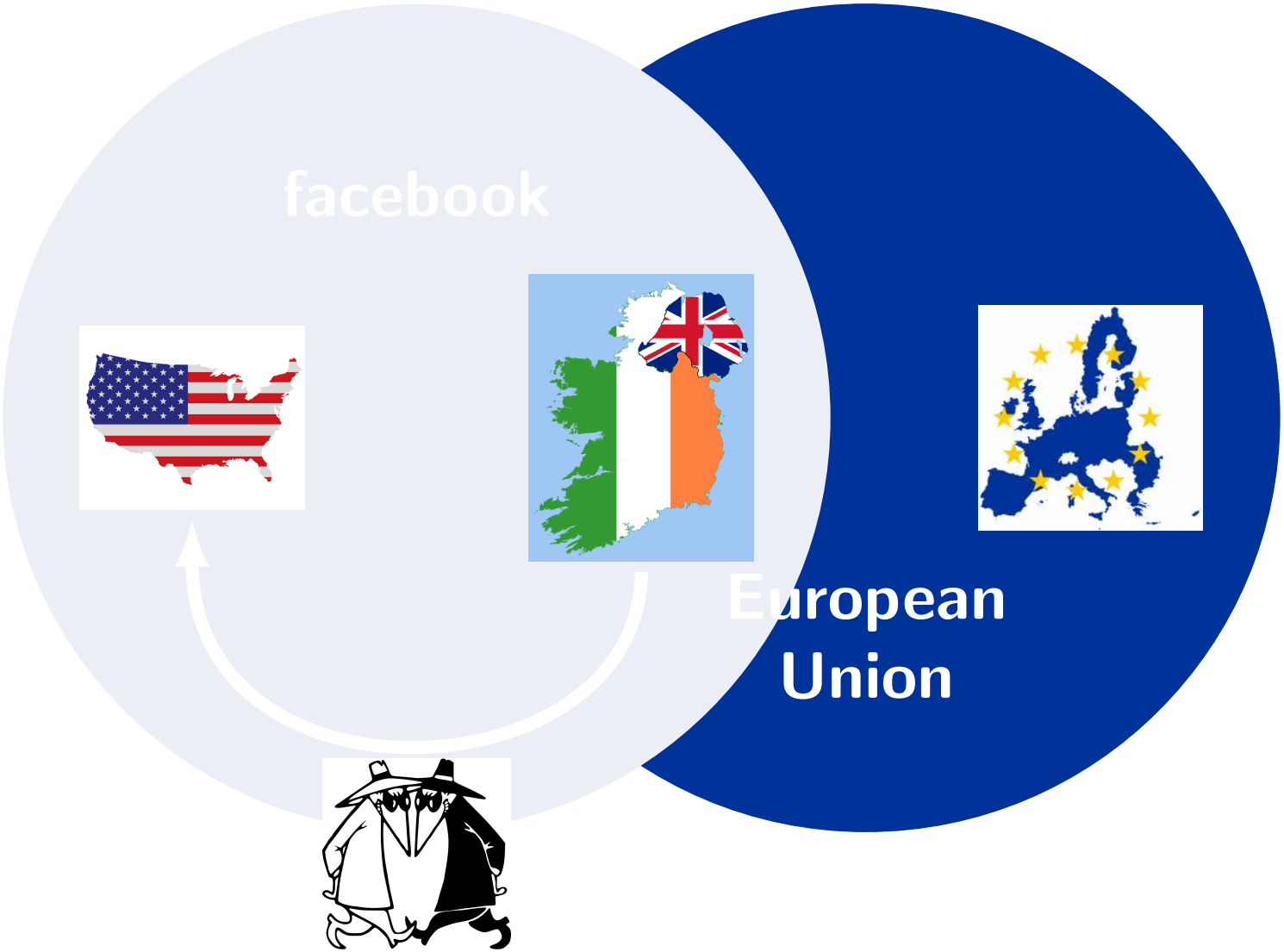
Scenario: digital civil rights



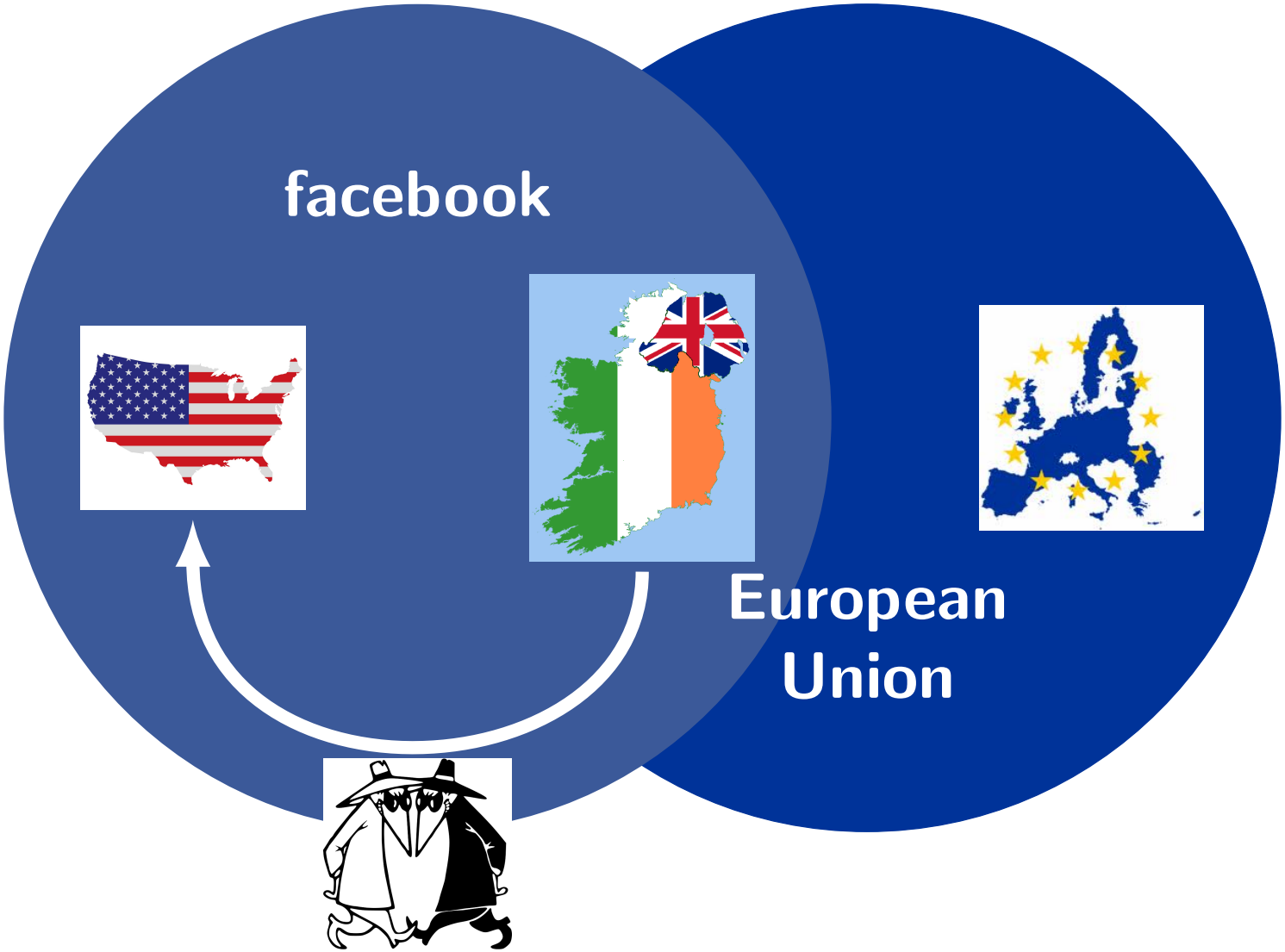
Scenario: digital civil rights



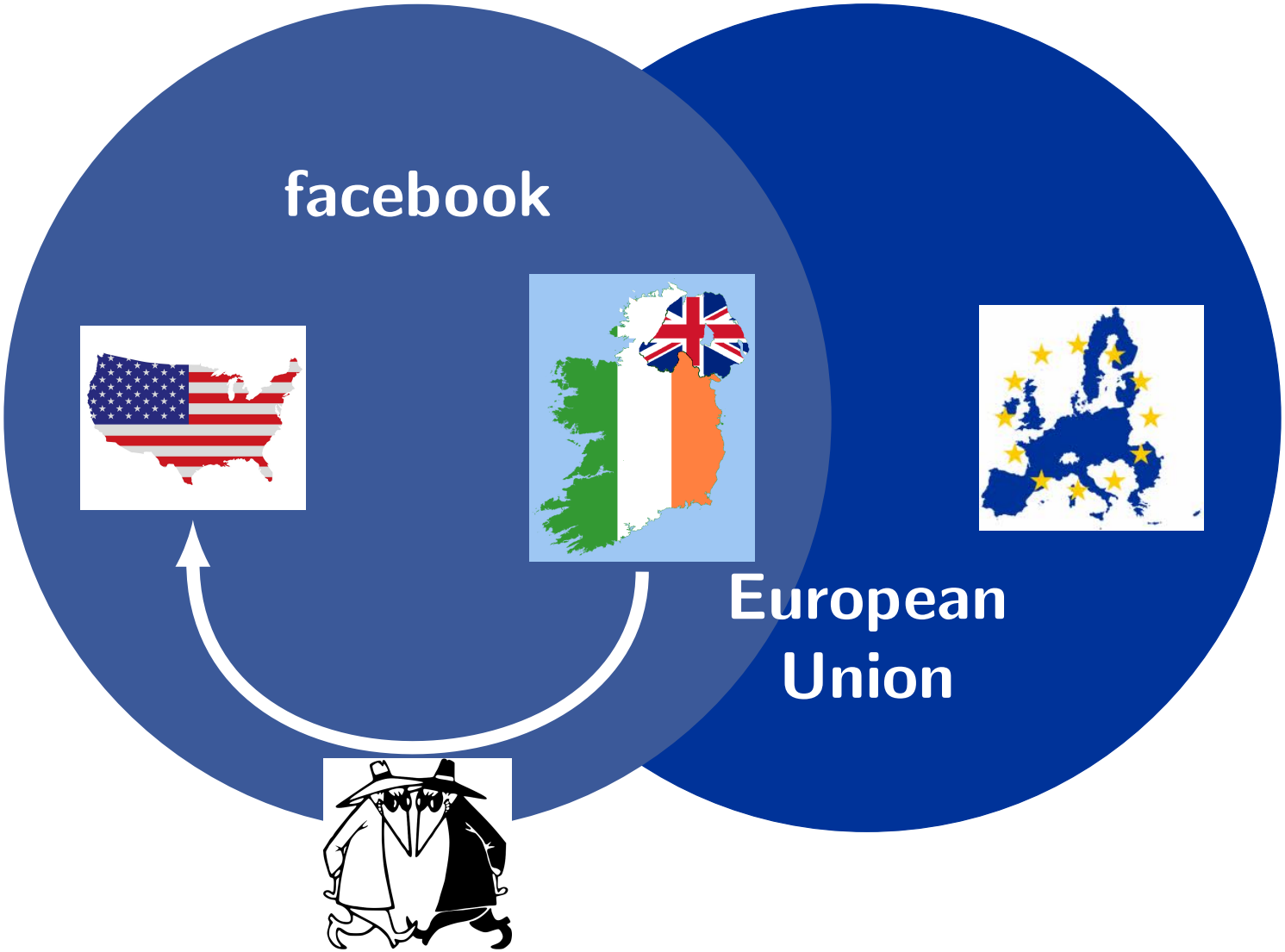
Scenario: digital civil rights



Scenario: digital civil rights



Scenario: digital civil rights



Overview

Contents

Model basics

- idealization of legal systems: LS^A, LS^B
- actors act; (legal) state changes:

$$LS_0^A \xrightarrow{e_0} LS_1^A \xrightarrow{e_1} \cdots LS_{n-1}^A \xrightarrow{e_{n-1}} LS_n^A$$

$$LS_0^B \xrightarrow{e_0} LS_1^B \xrightarrow{e_1} \cdots LS_{n-1}^B \xrightarrow{e_{n-1}} LS_n^B$$

- ... except for some technical details
- want to find
  - weak conflict: action permitted in LS^A, prohibited in LS^B
  - strong conflict: obligation in LS^A, prohibited in LS^B

Model basics

- idealization of legal systems: LS^A, LS^B
- actors act; (legal) state changes:

$$LS_0^A \stackrel{e_0}{\rightarrow} LS_1^A \stackrel{e_1}{\rightarrow} \cdots LS_{n-1}^A \stackrel{e_{n-1}}{\rightarrow} LS_n^A$$

$$LS_0^B \stackrel{e_0}{\rightarrow} LS_1^B \stackrel{e_1}{\rightarrow} \cdots LS_{n-1}^B \stackrel{e_{n-1}}{\rightarrow} LS_n^B$$

- ... except for some technical details
- want to find
  - weak conflict: action permitted in LS^A, prohibited in LS^B
  - strong conflict: obligation in LS^A, prohibited in LS^B

Model basics

- idealization of legal systems: LS^A, LS^B
- actors act; (legal) state changes:

$$LS_0^A \stackrel{e_0}{\rightarrow} LS_1^A \stackrel{e_1}{\rightarrow} \cdots LS_{n-1}^A \stackrel{e_{n-1}}{\rightarrow} LS_n^A$$

$$LS_0^B \stackrel{e_0}{\rightarrow} LS_1^B \stackrel{e_1}{\rightarrow} \cdots LS_{n-1}^B \stackrel{e_{n-1}}{\rightarrow} LS_n^B$$

- ... except for some technical details
- want to find
  - weak conflict: action permitted in LS^A, prohibited in LS^B
  - strong conflict: obligation in LS^A, prohibited in LS^B

Model basics

- idealization of legal systems: LS^A, LS^B
- actors act; (legal) state changes:

$$LS_0^A \stackrel{e_0}{\rightarrow} LS_1^A \stackrel{e_1}{\rightarrow} \cdots LS_{n-1}^A \stackrel{e_{n-1}}{\rightarrow} LS_n^A$$

$$LS_0^B \stackrel{e_0}{\rightarrow} LS_1^B \stackrel{e_1}{\rightarrow} \cdots LS_{n-1}^B \stackrel{e_{n-1}}{\rightarrow} LS_n^B$$

- ... except for some technical details
- want to find
  - weak conflict: action permitted in LS^A, prohibited in LS^B
  - strong conflict: obligation in LS^A, prohibited in LS^B

Comparative Legal Specifications

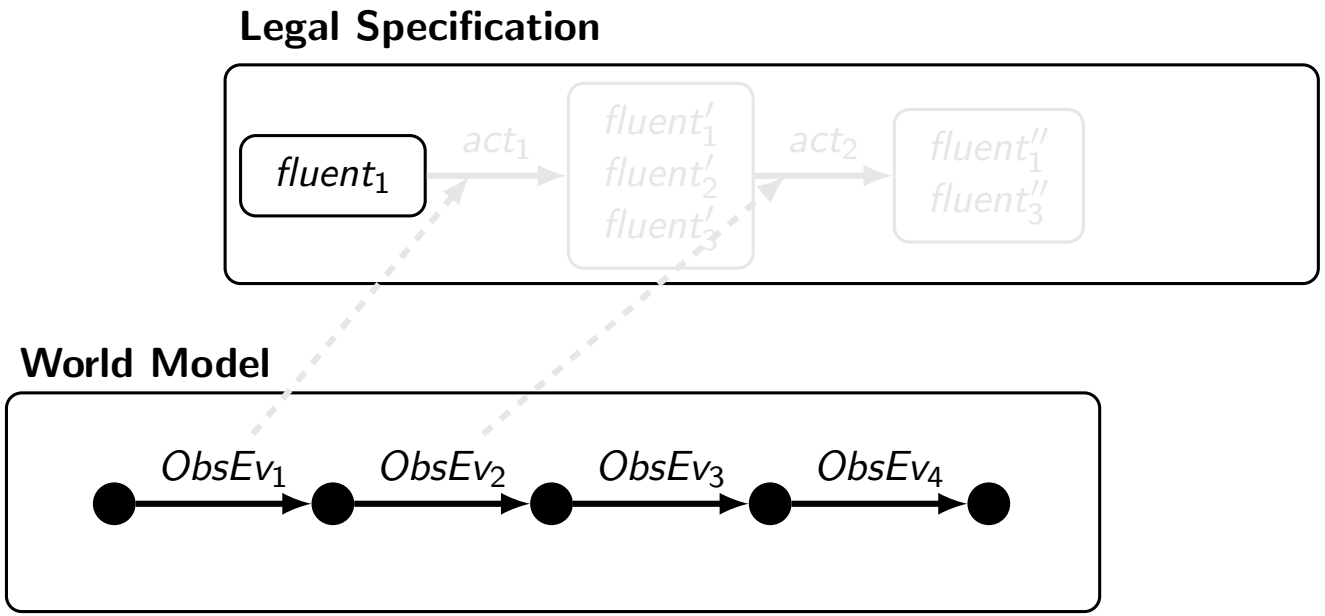
- ICAIL: Automatic revision of secondary legislation
  - Two sets of legislation/regulation
  - $LS^A \succ LS^B$ form a composite specification
  - but are independent of one another
  - conflict detection $\rightsquigarrow$ revision of LS^B
  - interoperation is intrinsic to design/intention of LS^B

Interacting Legal Specifications

- JURIX: Interacting legal specifications
  - LS^A and LS^B are peers
  - an event in one can trigger an event in the other
  - change of state in one can affect the other
  - revision is not an option
  - conflict detection

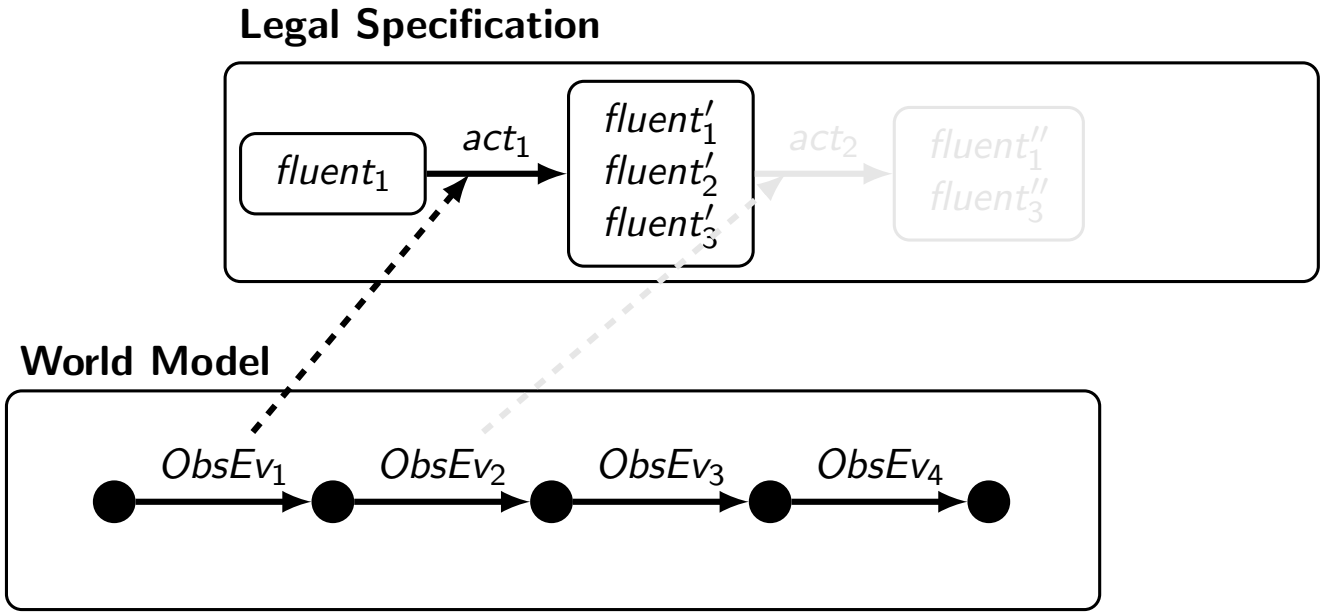
Contents

Single LS: Sketch



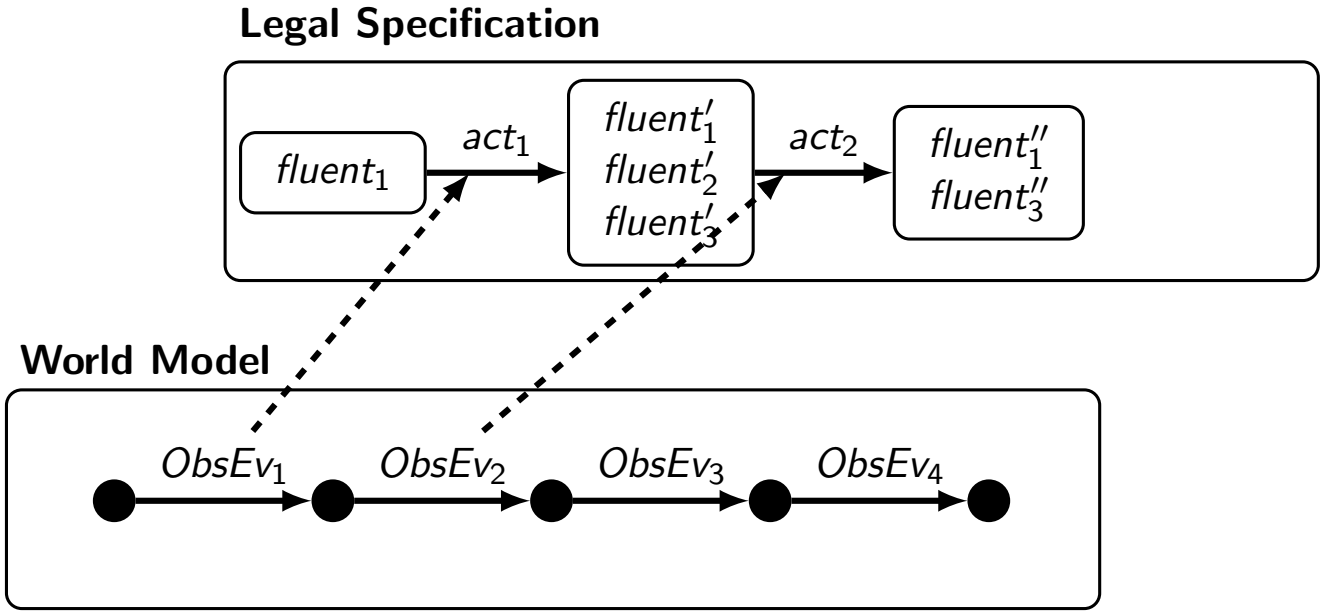
- Model generates **ordered traces** that show us the evolution of the legal specification over time—allows validation and verification.
- Essential elements of model are:
  - events ($\mathcal{E}$): exogenous and legal
  - fluents ($\mathcal{A}$): power, Permission, Obligation, Norm
- Legal Specification: $\Gamma = (\mathcal{E}, \mathcal{A}, \Gamma_0, \Gamma_t, \Delta)$

Single LS: Sketch



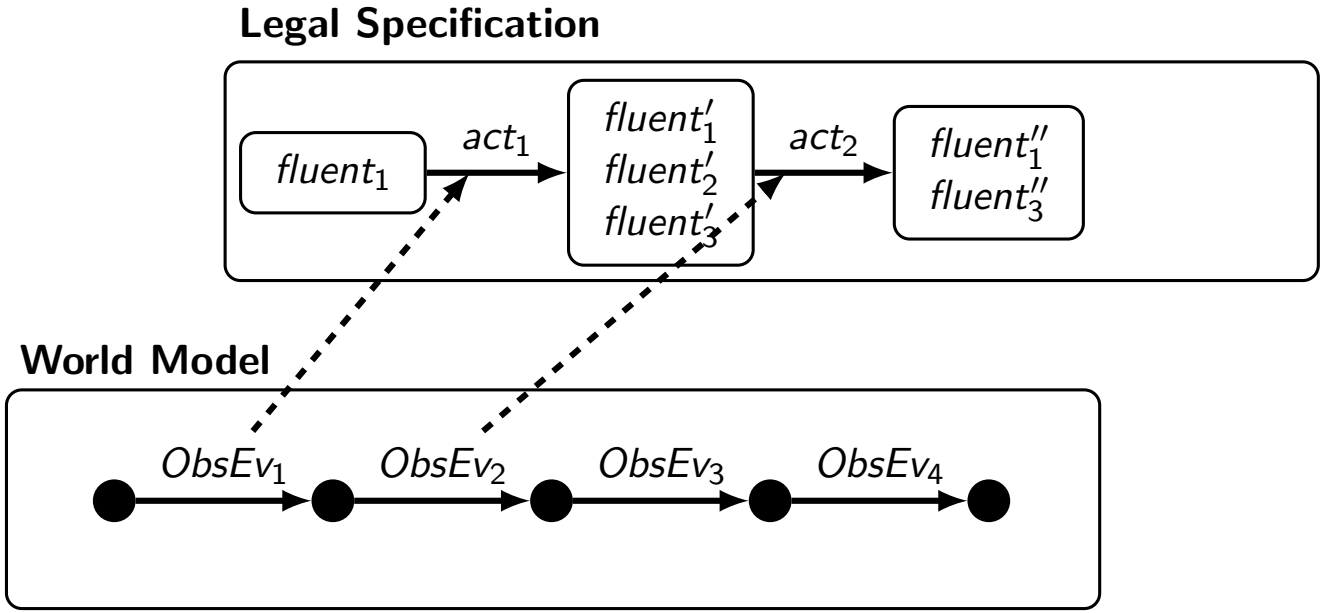
- Model generates **ordered traces** that show us the evolution of the legal specification over time—allows validation and verification.
- Essential elements of model are:
  - events ($\mathcal{E}$): exogenous and legal
  - fluents ($\mathcal{F}$): *pOWer* ∪ *Permission* ∪ *Obligation* ∪ *Domain*
- Legal Specification: $LS = (\mathcal{E}, \mathcal{F}, I_0, \tau, A)$

Single LS: Sketch



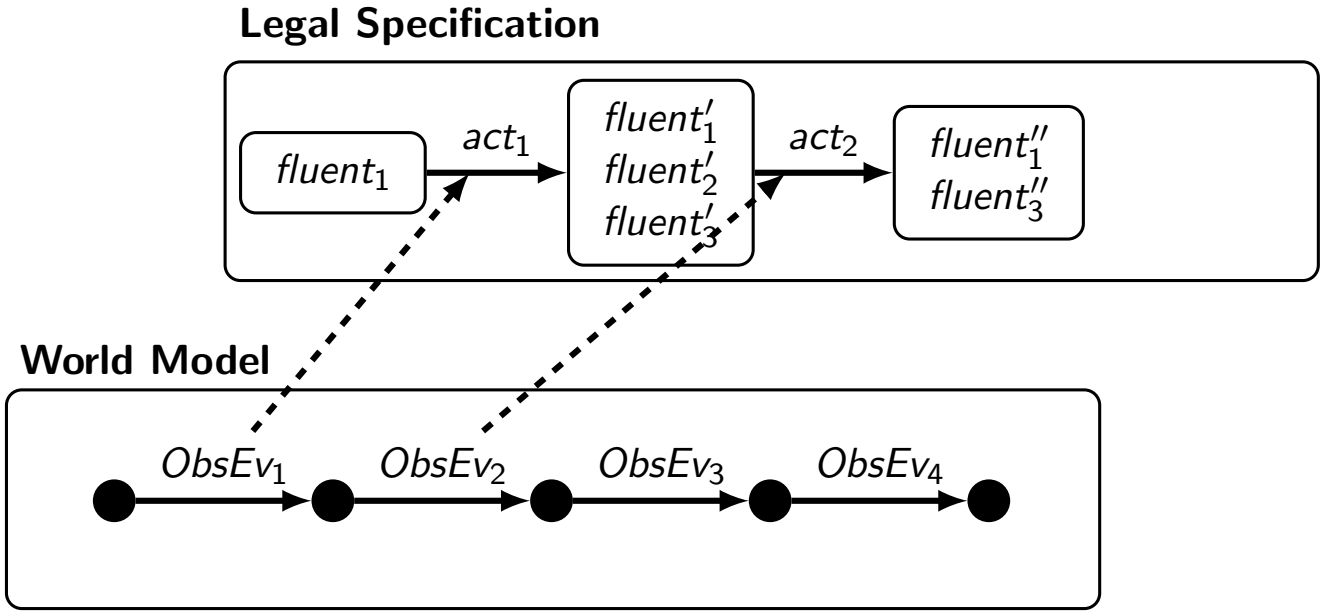
- Model generates **ordered traces** that show us the evolution of the legal specification over time—allows validation and verification.
- Essential elements of model are:
  - events ($\mathcal{E}$): exogenous and legal
  - fluents ($\mathcal{F}$): po$\mathcal{W}$er $\cup$ $\mathcal{P}$ermission $\cup$ $\mathcal{O}$bligation $\cup$ $\mathcal{D}$omain
- **Legal Specification**: $\mathcal{L} := \langle \mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{C}, \Delta \rangle$

Single LS: Sketch



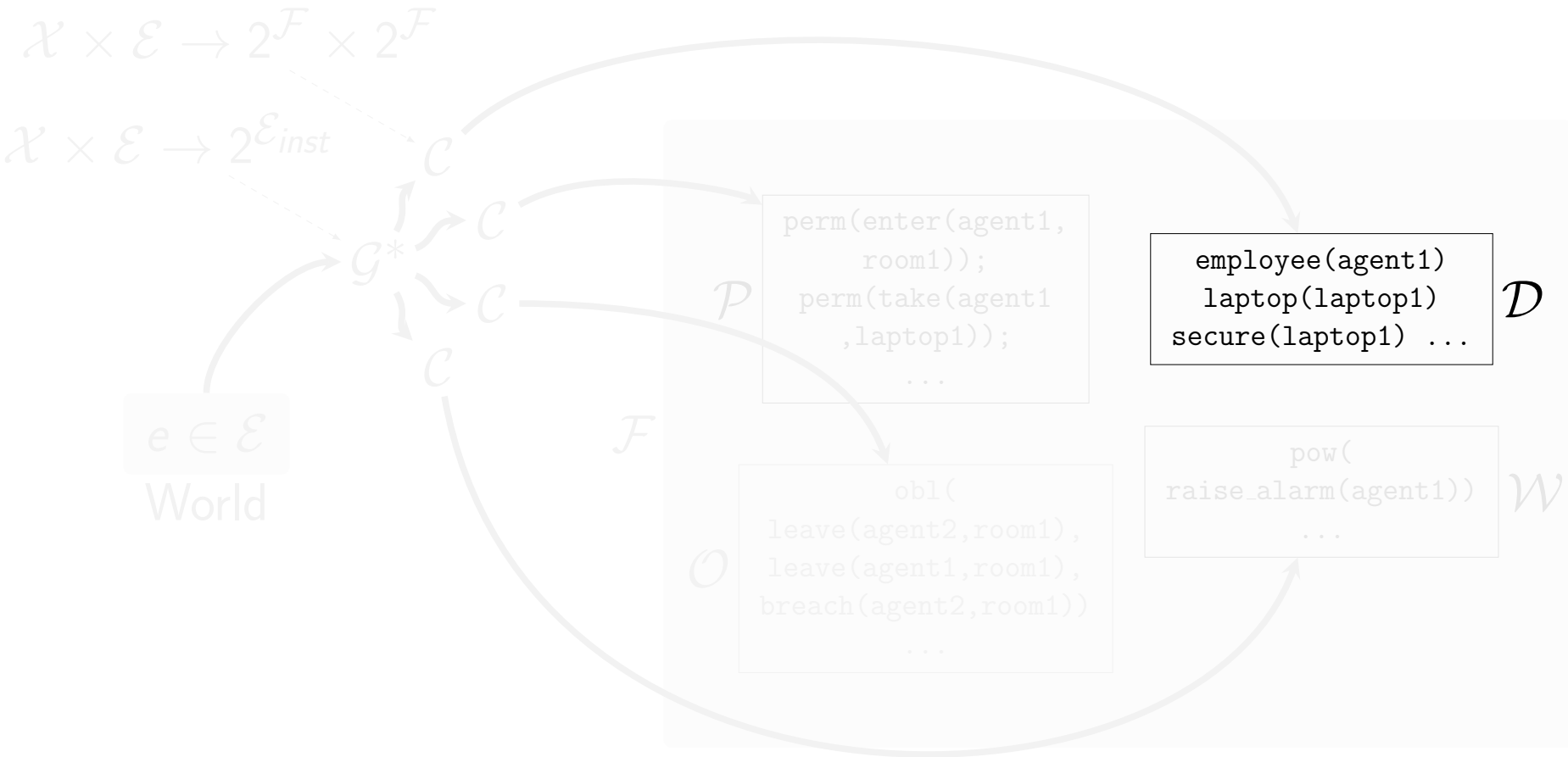
- Model generates **ordered traces** that show us the evolution of the legal specification over time—allows validation and verification.
- Essential elements of model are:
  - events ($\mathcal{E}$): exogenous and legal
  - fluents ($\mathcal{F}$): po$\mathcal{W}$er $\cup$ $\mathcal{P}$ermission $\cup$ $\mathcal{O}$bligation $\cup$ $\mathcal{D}$omain
- **Legal Specification**: $\mathcal{L} := \langle \mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{C}, \Delta \rangle$

Single LS: Sketch

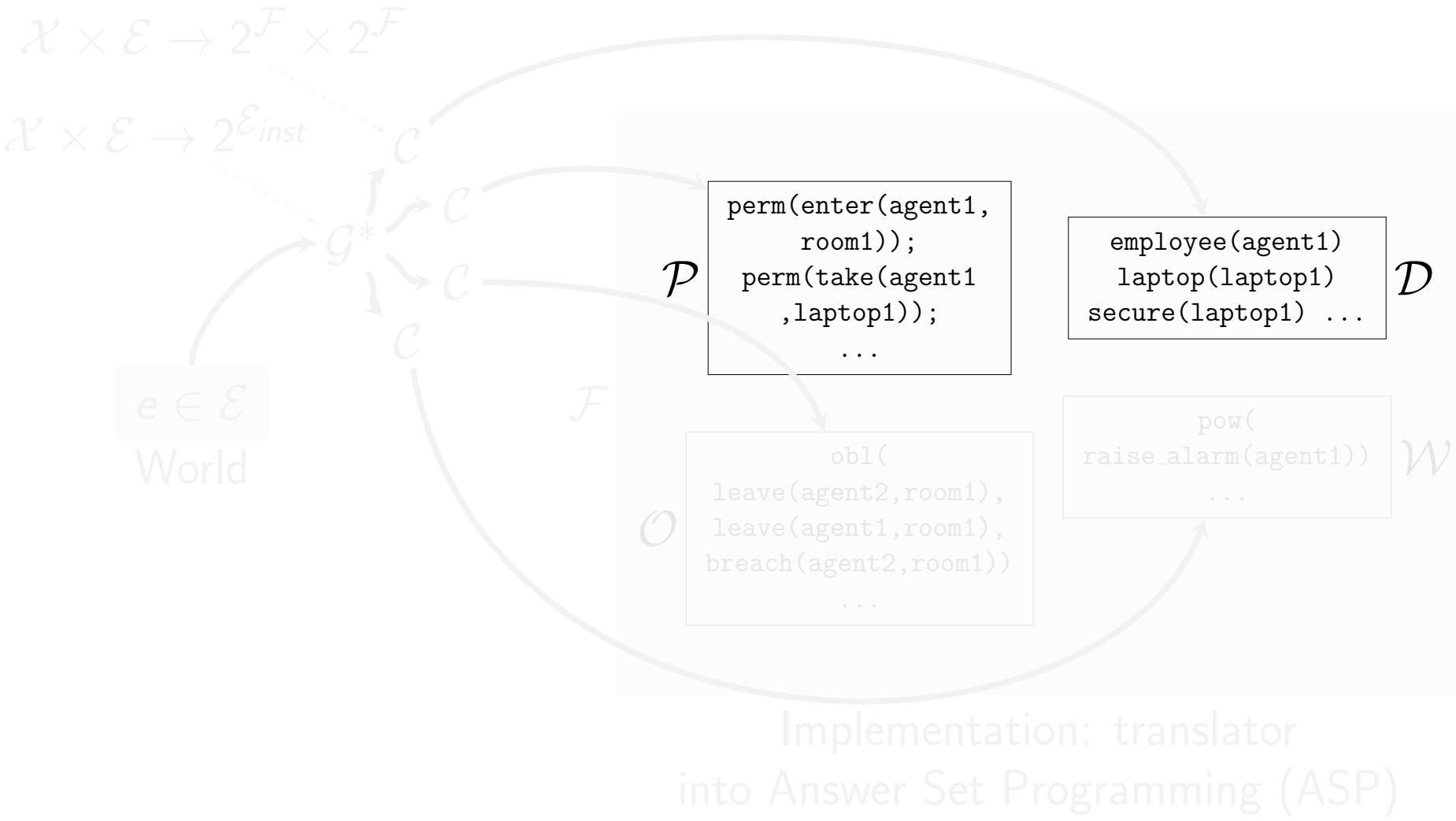


- Model generates **ordered traces** that show us the evolution of the legal specification over time—allows validation and verification.
- Essential elements of model are:
  - events ($\mathcal{E}$): exogenous and legal
  - fluents ($\mathcal{F}$): po$\mathcal{W}$er $\cup$ $\mathcal{P}$ermission $\cup$ $\mathcal{O}$bligation $\cup$ $\mathcal{D}$omain
- **Legal Specification**: $\mathcal{L} := \langle \mathcal{E}, \mathcal{F}, \mathcal{G}, \mathcal{C}, \Delta \rangle$

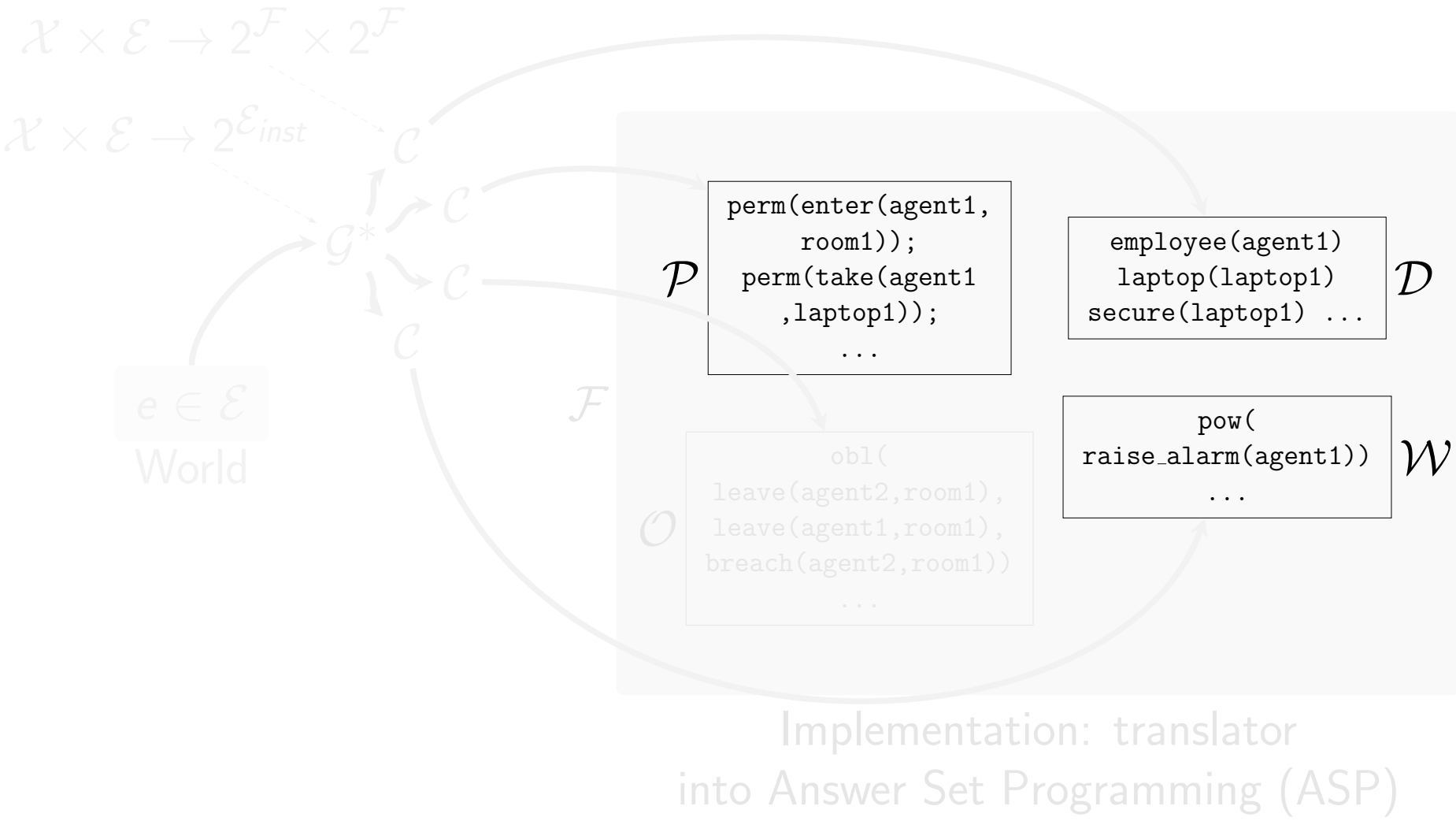
Single LS: Formal mapping



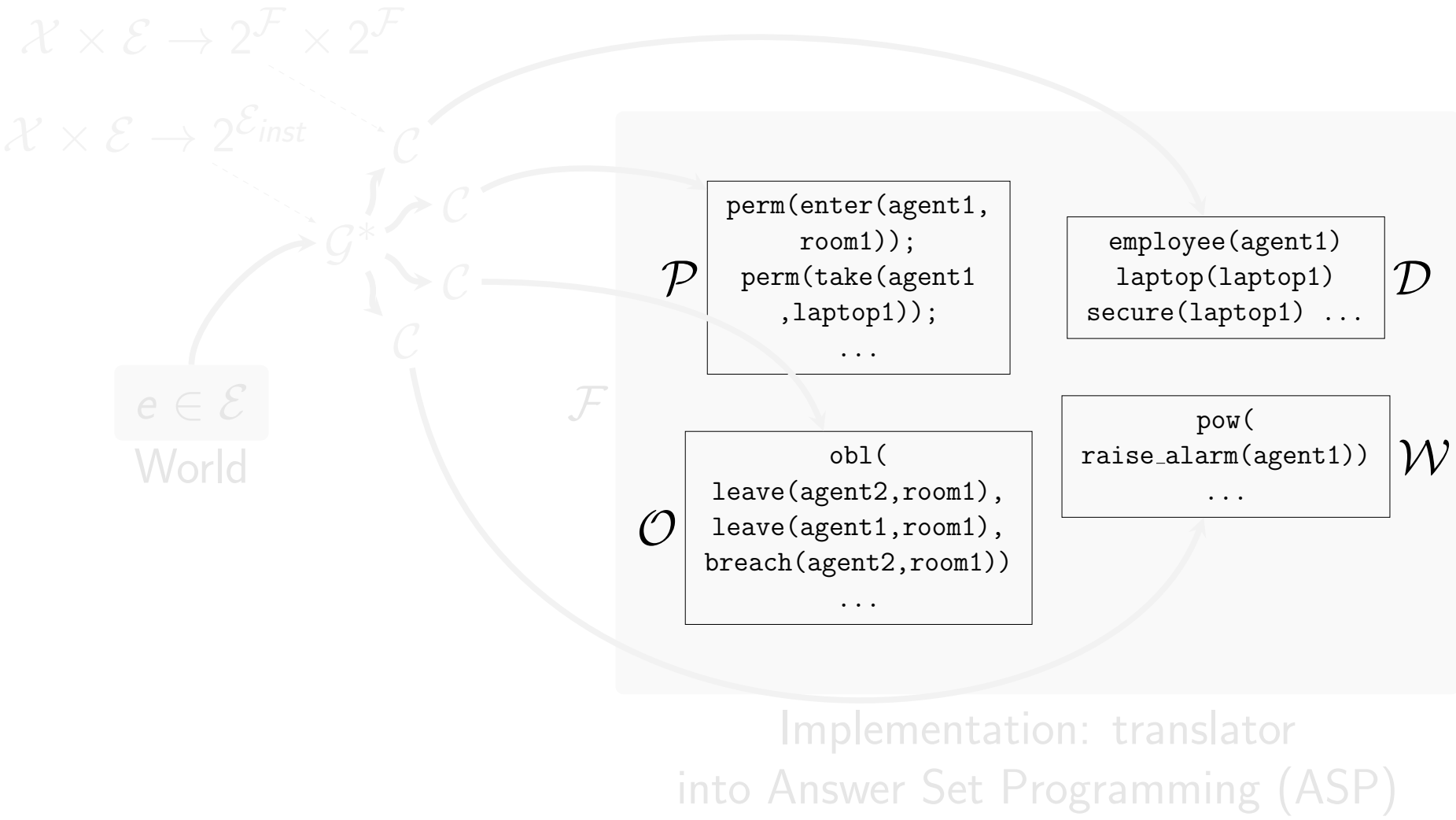
Single LS: Formal mapping



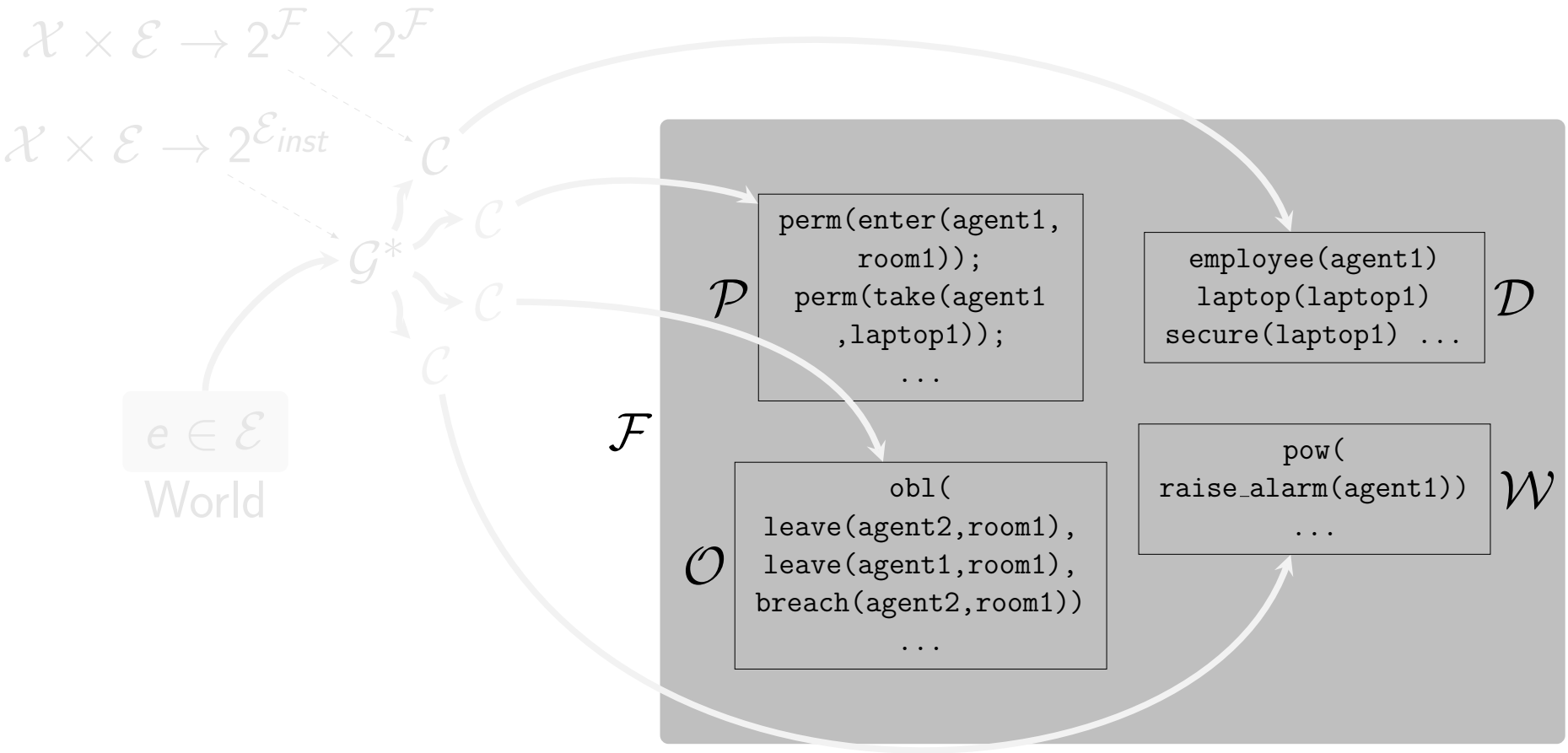
Single LS: Formal mapping



Single LS: Formal mapping

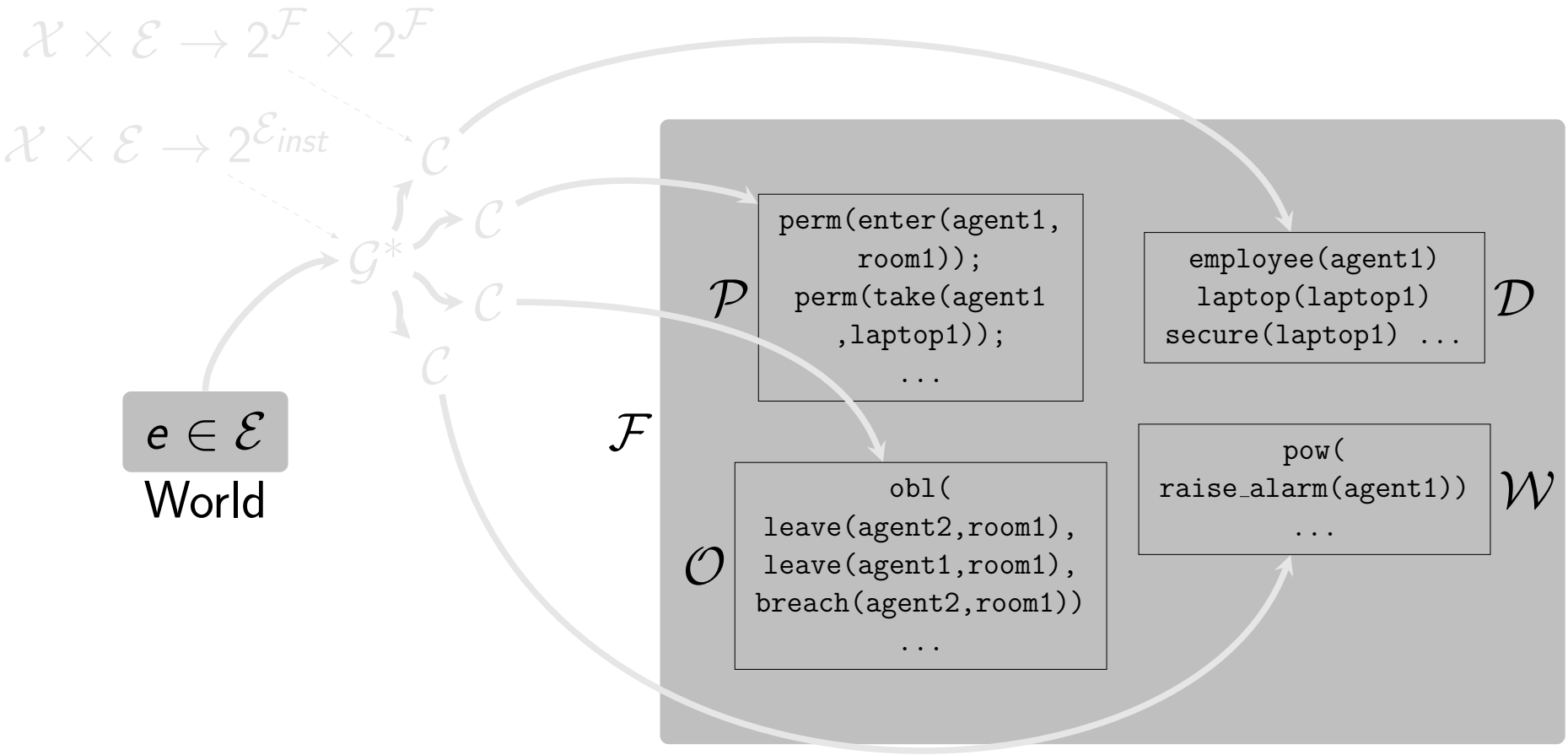


Single LS: Formal mapping



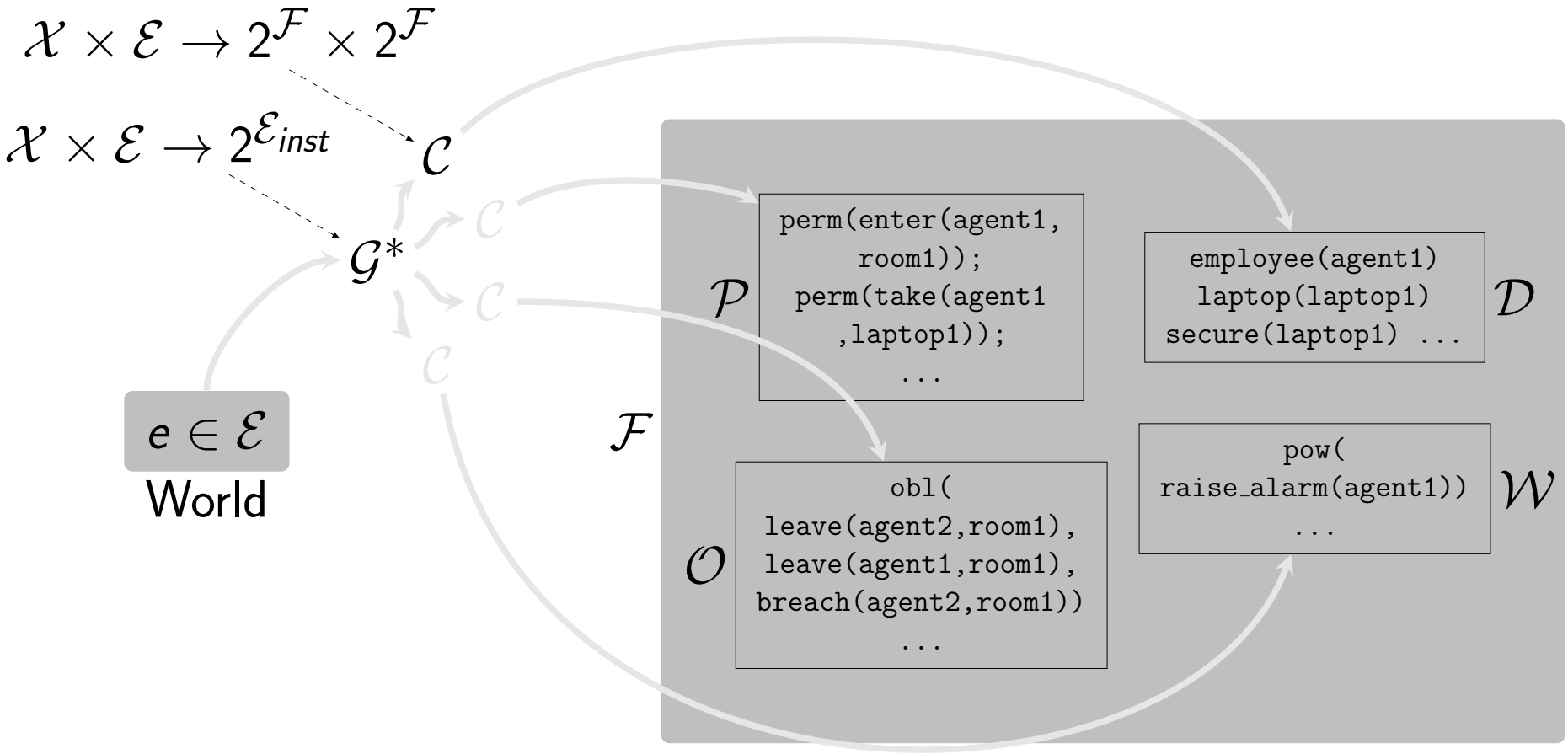
Implementation: translator
into Answer Set Programming (ASP)

Single LS: Formal mapping



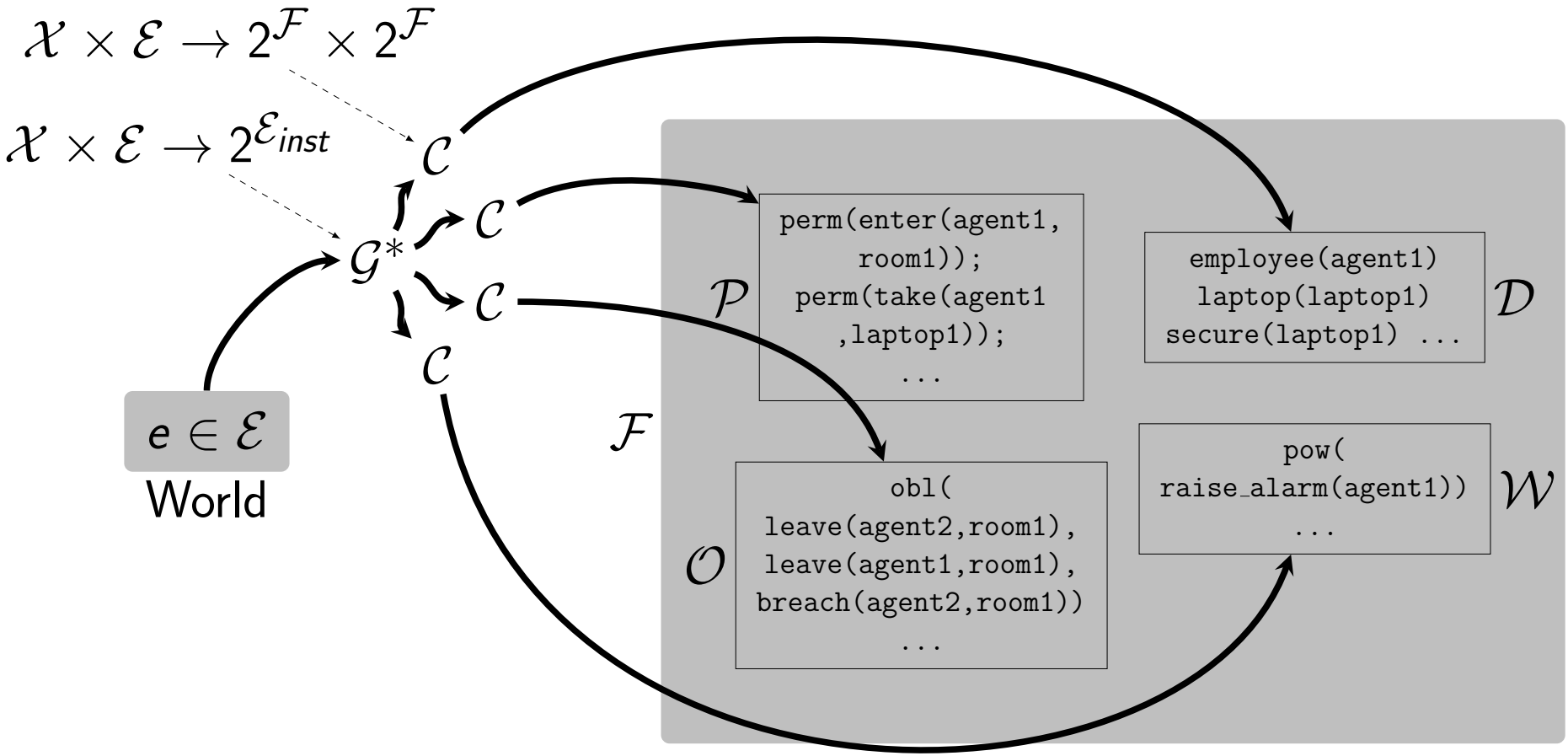
Implementation: translator
into Answer Set Programming (ASP)

Single LS: Formal mapping



Implementation: translator
into Answer Set Programming (ASP)

Single LS: Formal mapping



Implementation: translator
into Answer Set Programming (ASP)

Translation to *AnsProlog*

- ASP rules: *Conclusion* :− *Conditions*.
- Negation as failure
- Key terms:
  - `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 denoting norms
  - `initiated(Fluent,Instant)`: fluents to be added to state
  - `terminated(Fluent,Instant)`: fluents to be deleted from the state

Translation to *AnsProlog*

- ASP rules: *Conclusion* :− *Conditions*.
- Negation as failure
- Key terms:
  - `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 denoting norms
  - `initiated(Fluent,Instant)`: fluents to be added to state
  - `terminated(Fluent,Instant)`: fluents to be deleted from the state

Translation to *AnsProlog*

- ASP rules: *Conclusion* :− *Conditions*.
- Negation as failure
- Key terms:
  - `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 denoting norms
  - `initiated(Fluent,Instant)`: fluents to be added to state
  - `terminated(Fluent,Instant)`: fluents to be deleted from the state

Contents

Requirements for Interaction

- Event in LS^A $\mathcal{G}$enerates an event in LS^B
- Event in LS^A has a $\mathcal{C}$onsequence for state of LS^B
- Cliffe et.al. (2007) added rules to existing specifications
- To preserve the interface, specify separately
- Bridge rules $\Rightarrow$
  - cross-specification generation rules: $\mathcal{G}^x$
  - cross-specification consequence rules: $\mathcal{C}^x$

Requirements for Interaction

- Event in LS^A $\mathcal{G}$enerates an event in LS^B
- Event in LS^A has a $\mathcal{C}$onsequence for state of LS^B
- Cliffe et.al. (2007) added rules to existing specifications
- To preserve the interface, specify separately
- Bridge rules $\Rightarrow$
  - cross-specification generation rules: $\mathcal{G}^x$
  - cross-specification consequence rules: $\mathcal{C}^x$

Requirements for Interaction

- Event in LS^A $\mathcal{G}$enerates an event in LS^B
- Event in LS^A has a $\mathcal{C}$onsequence for state of LS^B
- Cliffe et.al. (2007) added rules to existing specifications
- To preserve the interface, specify separately
- Bridge rules $\Rightarrow$
  - cross-specification generation rules: $\mathcal{G}^x$
  - cross-specification consequence rules: $\mathcal{C}^x$

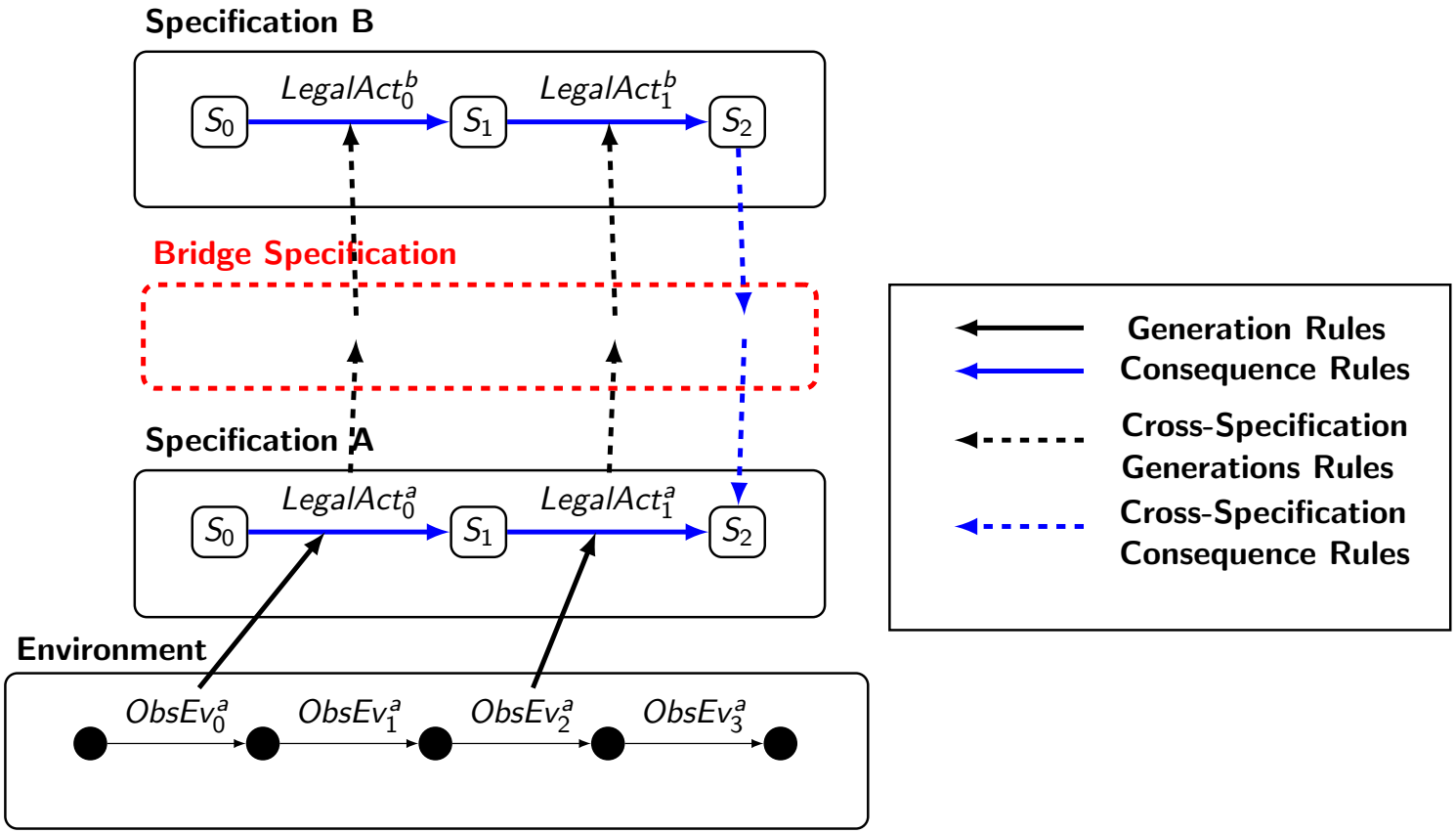
Requirements for Interaction

- Event in LS^A $\mathcal{G}$enerates an event in LS^B
- Event in LS^A has a $\mathcal{C}$onsequence for state of LS^B
- Cliffe et.al. (2007) added rules to existing specifications
- To preserve the interface, specify separately
- Bridge rules $\Rightarrow$
  - cross-specification generation rules: $\mathcal{G}^x$
  - cross-specification consequence rules: $\mathcal{C}^x$

Requirements for Interaction

- Event in LS^A $\mathcal{G}$enerates an event in LS^B
- Event in LS^A has a $\mathcal{C}$onsequence for state of LS^B
- Cliffe et.al. (2007) added rules to existing specifications
- To preserve the interface, specify separately
- Bridge rules $\Rightarrow$
  - cross-specification generation rules: $\mathcal{G}^x$
  - cross-specification consequence rules: $\mathcal{C}^x$

Interacting LS: sketch



- Formal model: in paper along with ASP translation rules

Translation to *AnsProlog*

- Mostly unchanged except ...
- ... LS^A and LS^B may have events/fluents with the same name
- the potential conflict we want to discover
- but letting this happen is not helpful
- Extend key terms to tag events/fluents with originating LS:
  - `occurred(Event,LS,Instant)`
  - `holdsat(Fluent,LS,Instant)`
  - `initiated(Fluent,LS,Instant)`
  - `terminated(Fluent,LS,Instant)`

Translation to *AnsProlog*

- Mostly unchanged except ...
- ... LS^A and LS^B may have events/fluents with the same name
- the potential conflict we want to discover
- but letting this happen is not helpful
- Extend key terms to tag events/fluents with originating LS:
  - `occurred(Event,LS,Instant)`
  - `holdsat(Fluent,LS,Instant)`
  - `initiated(Fluent,LS,Instant)`
  - `terminated(Fluent,LS,Instant)`

Translation to *AnsProlog*

- Mostly unchanged except ...
- ... LS^A and LS^B may have events/fluents with the same name
- the potential conflict we want to discover
- but letting this happen is not helpful
- Extend key terms to tag events/fluents with originating LS:
  - `occurred(Event,LS,Instant)`
  - `holdsat(Fluent,LS,Instant)`
  - `initiated(Fluent,LS,Instant)`
  - `terminated(Fluent,LS,Instant)`

Translation to *AnsProlog*

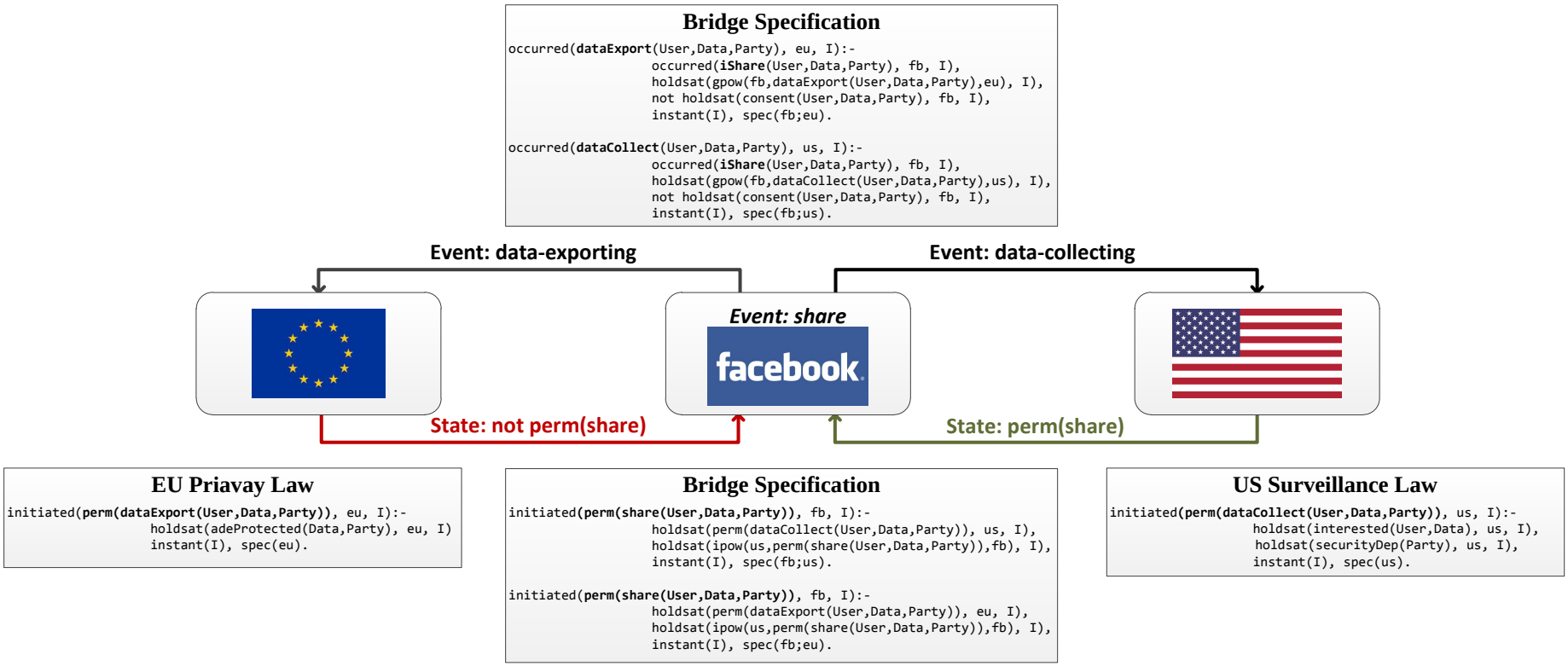
- Mostly unchanged except ...
- ... LS^A and LS^B may have events/fluents with the same name
- the potential conflict we want to discover
- but letting this happen is not helpful
- Extend key terms to tag events/fluents with originating LS:
  - `occurred(Event,LS,Instant)`
  - `holdsat(Fluent,LS,Instant)`
  - `initiated(Fluent,LS,Instant)`
  - `terminated(Fluent,LS,Instant)`

Translation to *AnsProlog*

- Mostly unchanged except ...
- ... LS^A and LS^B may have events/fluents with the same name
- the potential conflict we want to discover
- but letting this happen is not helpful
- Extend key terms to tag events/fluents with originating LS:
  - `occurred(Event,LS,Instant)`
  - `holdsat(Fluent,LS,Instant)`
  - `initiated(Fluent,LS,Instant)`
  - `terminated(Fluent,LS,Instant)`

Contents

Case Study


Bridge Specification
occurred(dataExport(User,Data,Party), eu, I):-
occurred(iShare(User,Data,Party), fb, I),
holdsat(gpow(fb,dataExport(User,Data,Party),eu), I),
not holdsat(consent(User,Data,Party), fb, I),
instant(I), spec(fb;eu).
occurred(dataCollect(User,Data,Party), us, I):-
occurred(iShare(User,Data,Party), fb, I),
holdsat(gpow(fb,dataCollect(User,Data,Party),us), I),
not holdsat(consent(User,Data,Party), fb, I),
instant(I), spec(fb;us).
Event: data-exporting
Event: data-collecting
Event: share
facebook
State: not perm(share)
State: perm(share)
EU Priavay Law
initiated(perm(dataExport(User,Data,Party)), eu, I):-
holdsat(adeProtected(Data,Party), eu, I)
instant(I), spec(eu).
Bridge Specification
initiated(perm(share(User,Data,Party)), fb, I):-
holdsat(perm(dataCollect(User,Data,Party)), us, I),
holdsat(ipow(us,perm(share(User,Data,Party)),fb), I),
instant(I), spec(fb;us).
initiated(perm(share(User,Data,Party)), fb, I):-
holdsat(perm(dataExport(User,Data,Party)), eu, I),
holdsat(ipow(us,perm(share(User,Data,Party)),fb), I),
instant(I), spec(fb;eu).
US Surveillance Law
initiated(perm(dataCollect(User,Data,Party)), us, I):-
holdsat(interested(User,Data), us, I),
holdsat(securityDep(Party), us, I),
instant(I), spec(us).

Conflict detection

- Adding

```
1  weakConflict(X, Y, I, F) :-
2      holdsat(F, X, I), % F is true in X
3      not  holdsat(F, Y, I), % F is false in Y
4      instant(I), spec(X;Y).
```

- and

```
1  strongConflict(X, Y, I, E) :-
2      holdsat(obl(E, D, V), X, I), % obligation
3      not  holdsat(perm(E),Y, I), % prohibition
4      spec(X;Y), instant(I).
```

- $\rightsquigarrow$ answer sets with above terms... if conflicts exist

Animating the scenario

1. Add some actions (events) that characterise the situation to examine
2. NSA requests Facebook to share the data of Bob and Alice

```
1  shareRequest(bob, bob_data, nsa)
2  shareRequest(alice, alice_data, nsa)
```

3. Alice has given consent to share
4. Bob has not

```
1  requestConsent(alice, alice_data, nsa)
2  requestConsent(bob, bob_data, nsa)
3  approveConsent(alice,alice_data,nsa)
```

5. Facebook shares data for both because NSA is a FB trusted party

```
1  share(bob,bob_data,nsa)
2  share(alice,alice_data,nsa)
```

In consequence

- Providing the initial facts

```
1  trusted(NSA) % in NSA we trust
2  interested(bob, bob_data) % subject of interest to NSA
3  interested(alice, alice_data) % Alice likewise
```

- and solving gives:

```
1  weakConflict(us, eu, 5, perm(share(bob, bob data, nsa)))
2  weakConflict(fb, eu, 5, perm(share(bob, bob data, nsa)))
3  strongConflict(us, eu, 5, share(bob, bob data, nsa))
```

- First two show EU disagrees with US and Facebook, resp.
- Third shows that Facebook is obliged by US law to share Bob's data but EU privacy law does not permit it
- (5 is the time instant when the conflict occurred)

Contents

Summary

- Contribution:
  - Formal model of bridge specifications
  - Computational model too
  - Allows connection of cooperating legal specifications
  - Allows conflict detection
- Future work:
  - Conflict resolution for cooperating LSs
  - Detection and resolution for merged specifications
  - Dynamic (on-line) conflict detection and resolution

Summary

- Contribution:
  - Formal model of bridge specifications
  - Computational model too
  - Allows connection of cooperating legal specifications
  - Allows conflict detection
- Future work:
  - Conflict resolution for cooperating LSs
  - Detection and resolution for merged specifications
  - Dynamic (on-line) conflict detection and resolution