Automated inference of rules with exception from past legal cases using ASP

Duangtida Athakravi, Mark Law,
Krysia Broda, Alessandra Russo
Imperial College London, UK

Ken Satoh
National Institute of Informatics, Japan

Overview

Imperial College London NII

- Introduction and Motivation
- Formalisation
- ASP Workflow
- Evaluation
- Summary and Future Work

Introduction

- In legal reasoning, we use written rules to make judgements
- Cases are very important since they can sometimes reveal exceptional situations not considered in written rules
- And these cases might have exceptions revealed by future cases

The purpose of this research is to:

- Formalise case rules in terms of general rules/exceptions.
- Investigate how to infer case-rules from previously judged cases

Motivation

	Factors	Factors	Judgement
			¬DepriveBuyerOfEntitlement (default)

Motivation

	Factors	Factors	Judgement
			¬DepriveBuyerOfEntitlement (default)
1	DeliveredOnTime (dot)		¬DepriveBuyerOfEntitlement

Motivation

	Factors	Factors	Judgement
			¬DepriveBuyerOfEntitlement (default)
1	DeliveredOnTime (dot)		¬DepriveBuyerOfEntitlement
2	DeliveredOnTime (dot) ItemsWereDamanged (ooo)		DepriveBuyerOfEntitlement

Motivation

	Factors	Factors	Judgement
			¬DepriveBuyerOfEntitlement (default)
1	DeliveredOnTime (dot)		¬DepriveBuyerOfEntitlement
2	DeliveredOnTime (dot) ItemsWereDamanged (ooo)		DepriveBuyerOfEntitlement
3	DeliveredOnTime (dot) ItemsWereDamaged (ooo)	DamageIsRepairable (rpl) BuyerFixedRepairTime (far) ItemsRepairedInTime (ria)	¬DepriveBuyerOfEntitlement

Motivation

	Factors	Factors	Judgement
			¬DepriveBuyerOfEntitlement (default)
1	DeliveredOnTime (dot)		¬DepriveBuyerOfEntitlement
2	DeliveredOnTime (dot) ItemsWereDamanged (ooo)		DepriveBuyerOfEntitlement
3	DeliveredOnTime (dot) ItemsWereDamaged (ooo)	DamageIsRepairable (rpl) BuyerFixedRepairTime (far) ItemsRepairedInTime (ria)	¬DepriveBuyerOfEntitlement
4	DeliveredOnTime (dot) ItemsWereDamaged (ooo)	DamageIsRepairable (rpl) BuyerFixedRepairTime (far)	?

Motivation

Should judgement of case 4 be
DepriveBuyerOfEntitlement
because case 2?
or
Should the judgement of case 4 be
¬DepriveBuyerOfEntitlement
because case 3?

	Factors		ment
			riveBuyerOfEntit ent (default)
1	Delivered		riveBuyerOfEntit lement
2	DeliveredOnTime (dot) ItemsWereDamanged (ooo)		DepriveBuyerOfEntit lement
3	DeliveredOnTime (dot) ItemsWereDamaged (ooo)	DamageIsRepairable (rpl) BuyerFixedRepairTime (far) ItemsRepairedInTime (ria)	¬DepriveBuyerOfEntit lement
4	DeliveredOnTime (dot) ItemsWereDamaged (ooo)	DamageIsRepairable (rpl) BuyerFixedRepairTime (far)	?

Case Base

- A case is a subset of a set of factors *F*.
- A case with judgement is cj = <c, j>, where $j \in \{+,–\}$

  case(cj) = c and *judgement*(cj)= j
- A case base *CB* is a set of cases with judgements

All casebases include $<\emptyset, j_0>$

- $\emptyset$ is the empty case and
- j_0 is the default judgement, assumed in the absence of any factor

For all cases cj_1 and cj_2 in CB,

if *case*(cj_1)=*case*(cj_2) then *judgement*(cj_1)=*judgement*(cj_2)

Raw Attack Relation (RA)

The raw attack relation

$$RA = \{< cj_1, \; cj_2 > \mid case(cj_1) \supset case(cj_2) \text{ and } judgement(cj_1) \neq judgement(cj_2) \}$$

cj_1 raw attacks cj_2 is denoted as $cj_1 \rightarrow_r cj_2$

Example

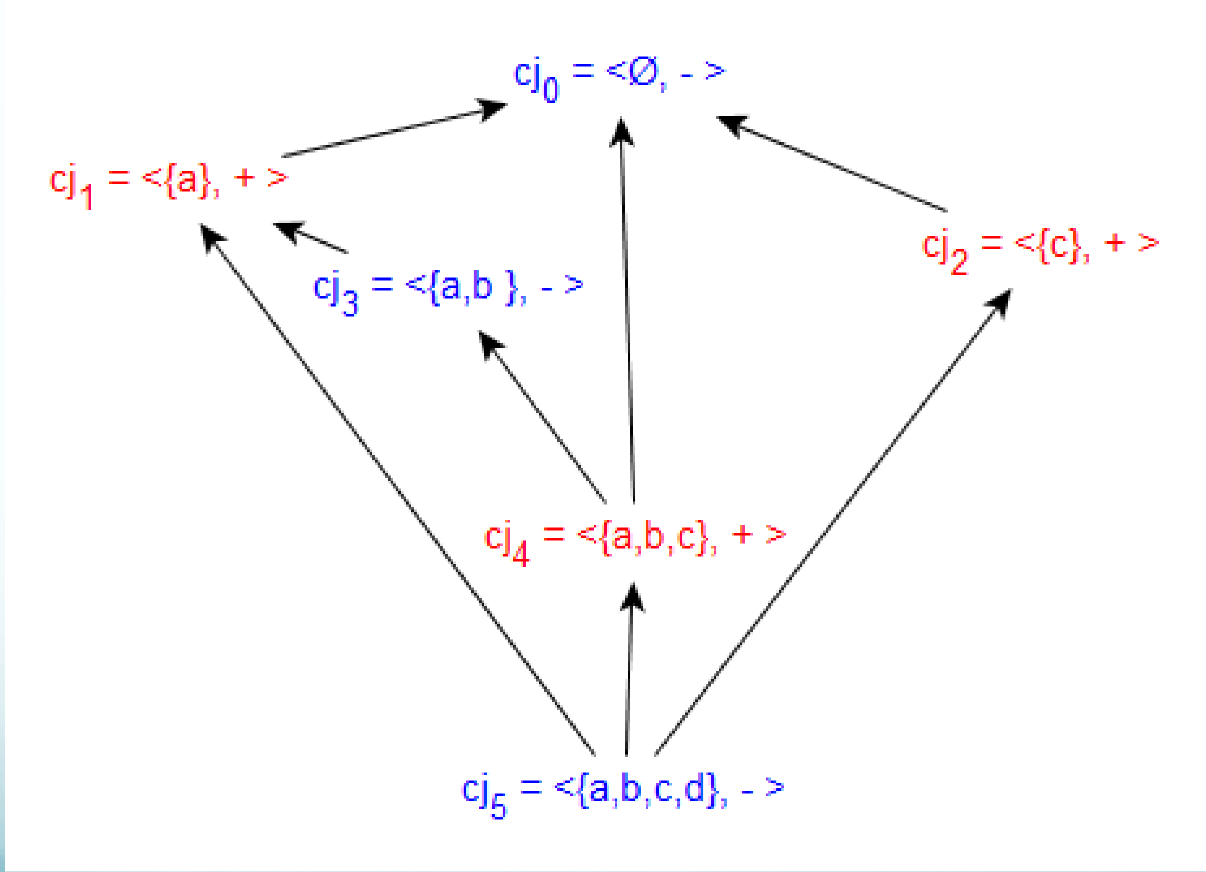
$F = \{a,b,c,d,e,f\}$

$CB = \{ cj_0 = <\emptyset, -> ,\quad cj_1 = <\{a\}, +> ,\quad cj_2 = <\{c\}, +> ,$
$cj_3 = <\{a,b\}, -> ,\quad cj_4 = <\{a,b,c\}, +> ,\quad cj_5 = <\{a,b,c,d\}, -> \}$

$$RA = \{ cj_1 \rightarrow_r cj_0, cj_2 \rightarrow_r cj_0, cj_4 \rightarrow_r cj_0, cj_3 \rightarrow_r cj_1, cj_5 \rightarrow_r cj_1, cj_4 \rightarrow_r cj_3, cj_5 \rightarrow_r cj_2, cj_5 \rightarrow_r cj_4 \}$$

Example: Raw Attack Relation

$$RA = \{\langle cj_1, cj_2 \rangle \mid case(cj_1) \supset case(cj_2) \text{ and } judgement(cj_1) \neq judgement(cj_2)\}$$



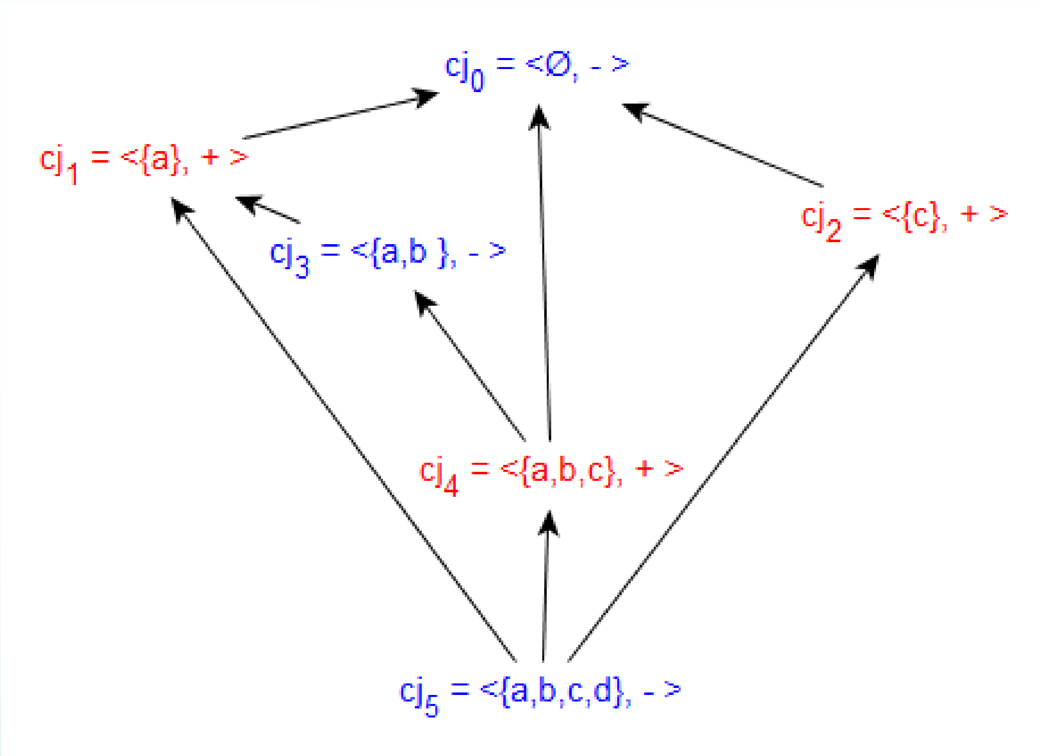
Attack Relation

The Attack relation $AT \subseteq RA$ is defined by:

- $< cj_1, \; cj_2 > \in AT$ if *case*(cj_2) = $\emptyset$ and there is no $cj_3 \rightarrow_r cj_2 \in RA$ such that $case(cj_1) \supset case(cj_3)$
- $< cj_1, \; cj_2 > \in AT$ if there exists $< cj_2, \; cj_4 > \in AT$ and no $cj_5 \rightarrow_r cj_2 \in RA$ such that $case(cj_1) \supset case(cj_5)$
- nothing else is in AT

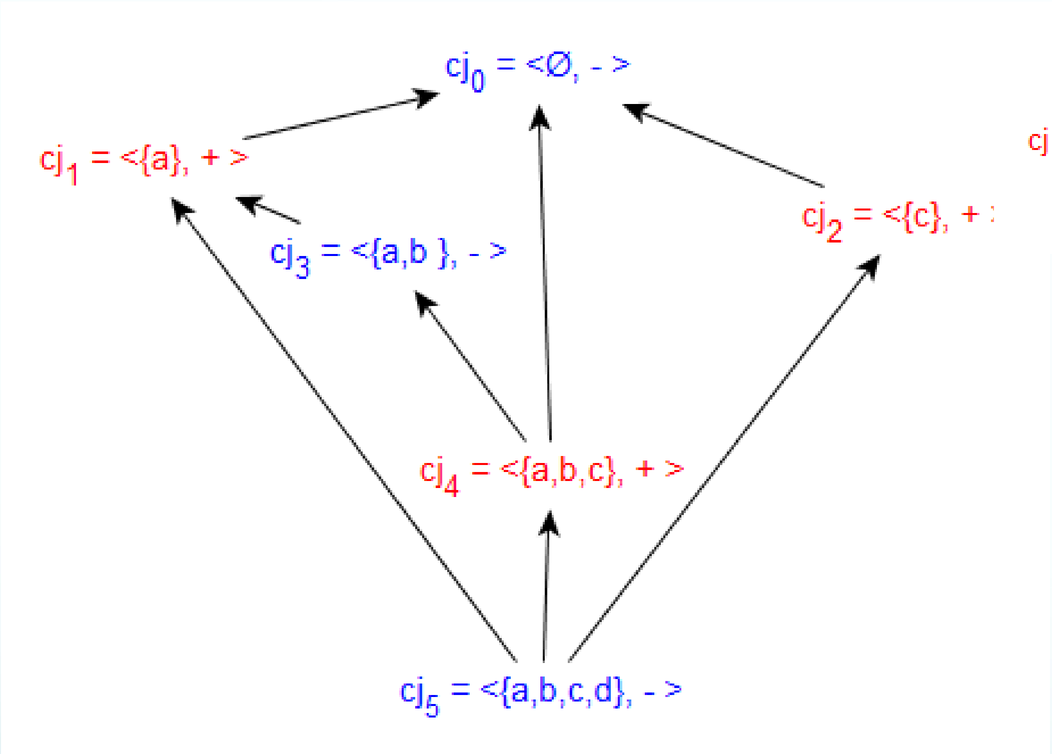
cj_1 attacks cj_2 is denoted $cj_1 \rightarrow cj_2$

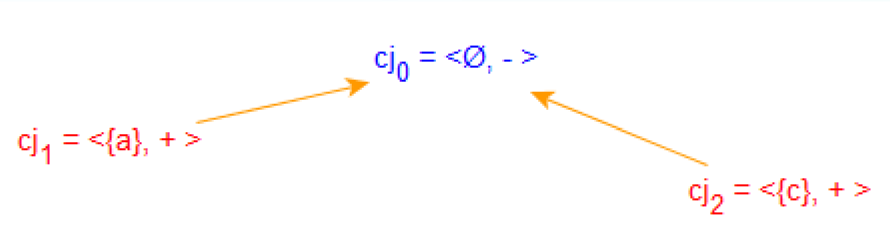
Example: Attack Relation



cj_0 = <∅, - >

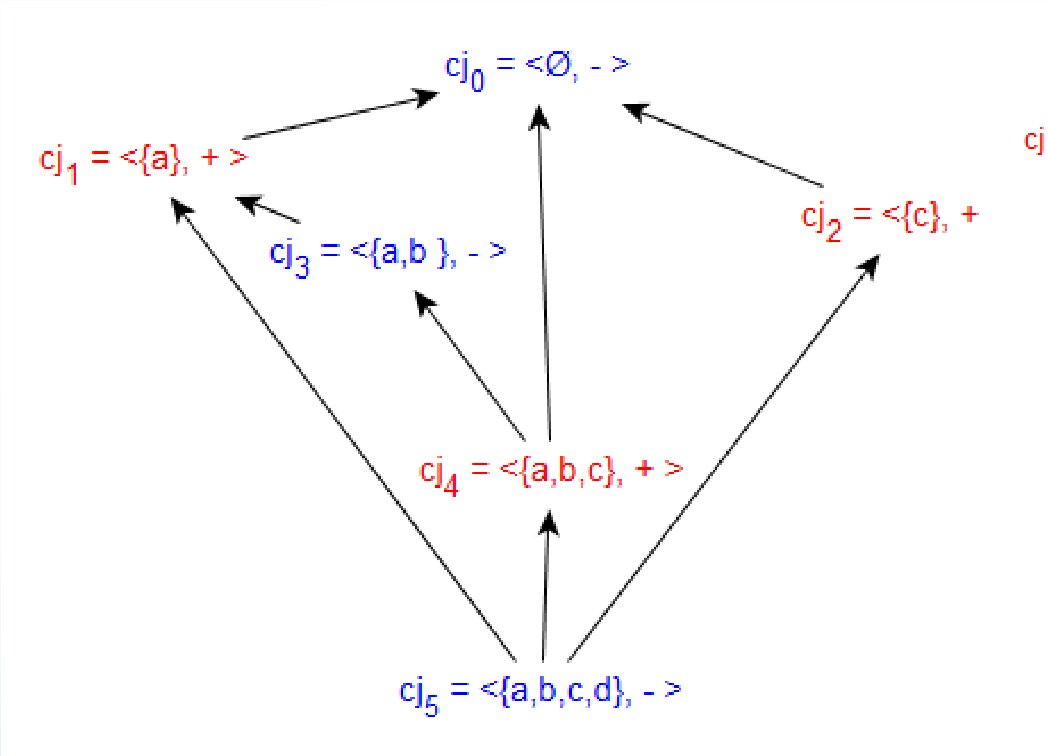
Example: Attack Relation

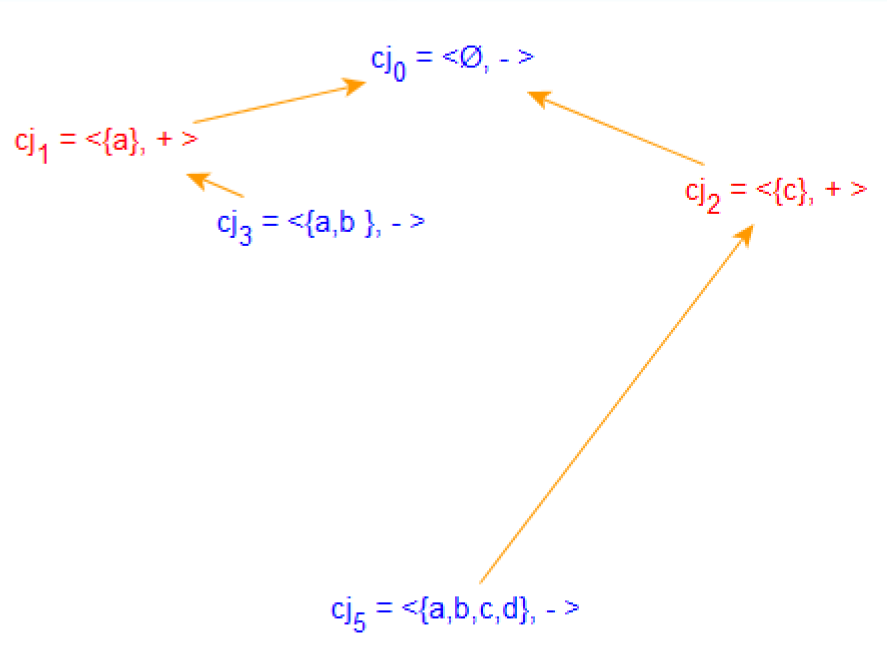




$cj_4 \nrightarrow cj_0$ since *case*(cj_4) = {a,b,c} is not minimal factors to overturn the judgement of cj_0

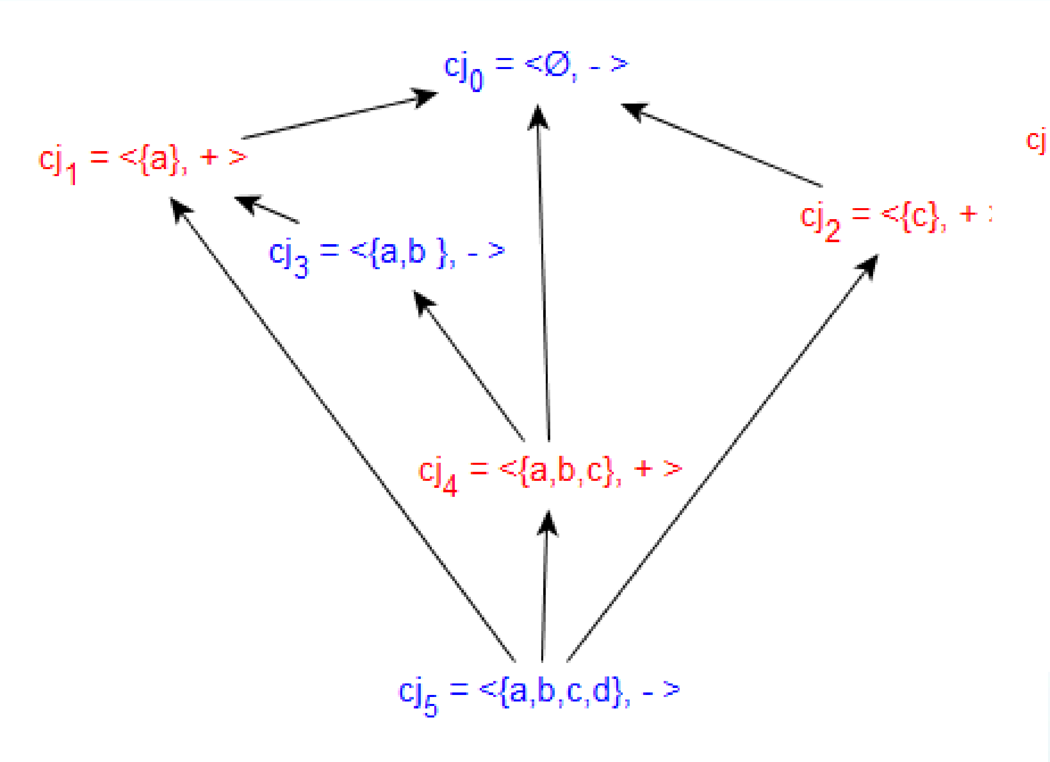
Example: Attack Relation

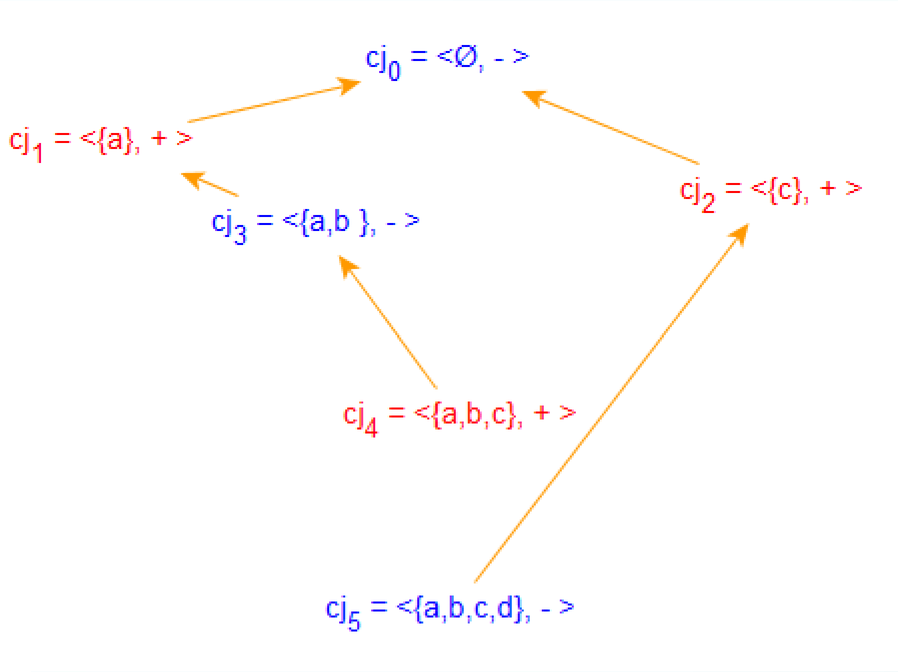




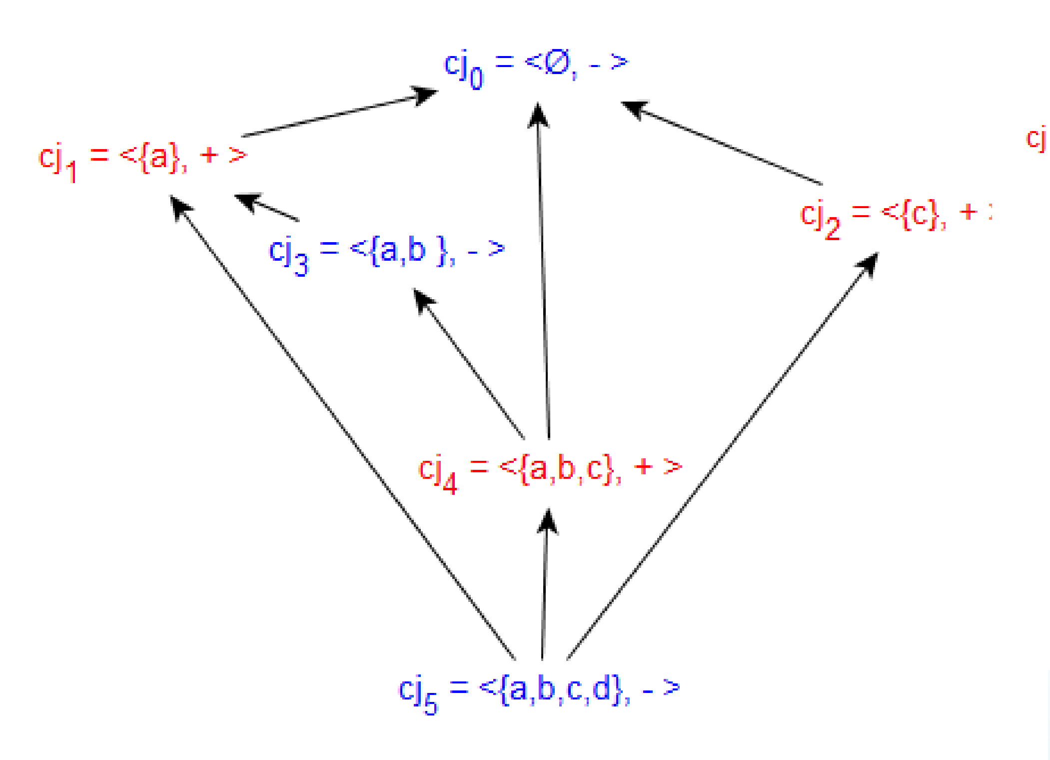
$cj_5 \nrightarrow cj_1$ since there is an intermediate overturning a judgement between cj_5 and cj_1

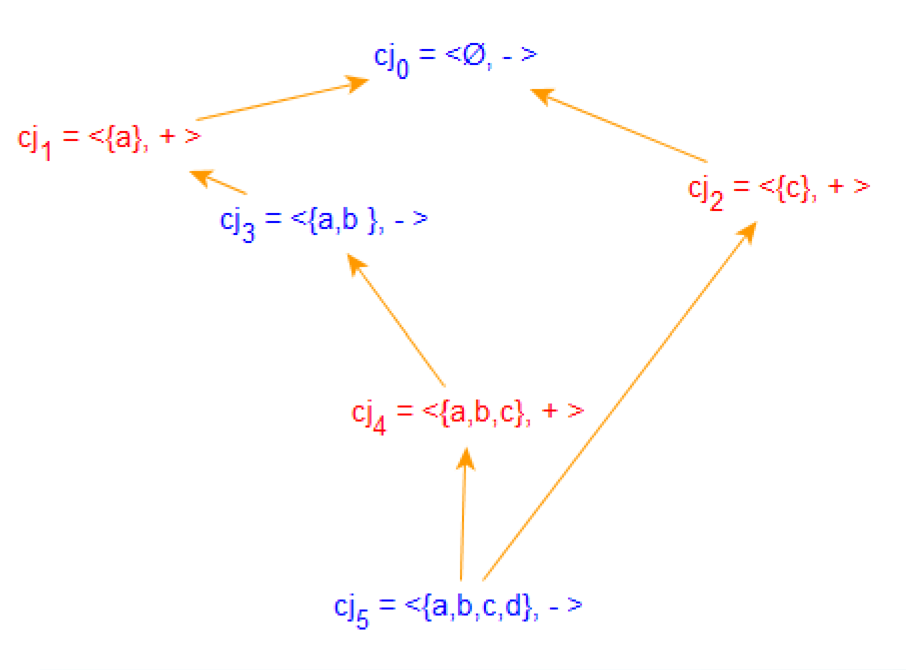
Example: Attack Relation





Example: Attack Relation





Argument

The set of factors responsible for overturning a judgement is called *an argument*.

For each pair $< cj_1, cj_2 >$ in the set of attacks AT

$$\alpha(cj_1, cj_2) = \mathit{case}(cj_1) - \mathit{case}(cj_2)$$

is the argument of the attack from cj_1 to cj_2

$CB = \{ cj_0 = <\emptyset, -\ >, cj_1 = <\{a\}, +\ >, cj_2 = <\{c\}, +\ >, cj_3 : <\{a,b\}, -\ >,$
$cj_4 = <\{a,b,c\}, +\ >, \quad cj_5 = <\{a,b,c,d\}, -\ >\}$

$AT = \{cj_1 \rightarrow cj_0, cj_2 \rightarrow cj_0, cj_3 \rightarrow cj_1, cj_4 \rightarrow cj_3, cj_5 \rightarrow cj_2, cj_5 \rightarrow cj_4\}$

$\alpha(cj_1, cj_0) = \{a\}, \ \alpha(cj_2, cj_0) = \{c\}, \ \alpha(cj_3, cj_1) = \{b\},$
$\alpha(cj_4, cj_3) = \{c\}, \ \alpha(cj_5, cj_2) = \{a,b,d\}, \alpha(cj_5, cj_4) = \{d\}$

Active Case with Judgement

To predict the judgement of a new case, we look at judgements of similar past cases that have not been overturned.

Given *CB*, corresponding *AT*, and a new case *c*,
A cj $\in$ CB is *active* w.r.t. c iff *case*(cj) $\subseteq$ c and for all $<cj_n, cj> \in$ AT, either cj_n is not active w.r.t. c or *case*(cj) $\not\subseteq$ c.

CB = { $cj_0 = <\emptyset, -\ >$, $cj_1 = <\{a\}, +\ >$, $cj_2 = <\{c\}, +\ >$, $cj_3: <\{a,b\}, -\ >$,
$cj_4 = <\{a,b,c\}, +\ >$, $cj_5 = <\{a,b,c,d\}, -\ >$}

AT = {$cj_1 \rightarrow cj_0$, $cj_2 \rightarrow cj_0$, $cj_3 \rightarrow cj_1$, $cj_4 \rightarrow cj_3$, $cj_5 \rightarrow cj_2$, $cj_5 \rightarrow cj_4$}

Let c_{new} = {a,b,d} be a new case.
The active cases with judgment w.r.t. c_{new} are cj_3, cj_0.
Note: cj_1 is not active w.r.t. c_{new} since $<cj_3,cj_1> \in$ AT and *case*(cj_3) $\subseteq$ c

Predicted Judgement

Given a CB and corresponding AT and unseen case c,
The unique predicted judgement of c is defined as

$$pj(c) = \text{default } j_0 \text{ iff } \langle \emptyset, j_0 \rangle \text{ is active w.r.t. } c$$

$CB = \{ cj_0 = \langle \emptyset, - \rangle, cj_1 = \langle \{a\}, + \rangle, cj_2 = \langle \{c\}, + \rangle, cj_3 : \langle \{a,b\}, - \rangle,$
$cj_4 = \langle \{a,b,c\}, + \rangle, \quad cj_5 = \langle \{a,b,c,d\}, - \rangle \}$

$AT = \{cj_1 \rightarrow cj_0, cj_2 \rightarrow cj_0, cj_3 \rightarrow cj_1, cj_4 \rightarrow cj_3, cj_5 \rightarrow cj_2, cj_5 \rightarrow cj_4\}$

For case $c_{new} = \{a,b,d\}$, cj_0 is active w.r.t. c_{new}

pj(c) = default judgement

Judgement Theory

Our aim is to generate a judgement theory from given CB and default j_0 to enable us to infer judgements of new cases *c*.

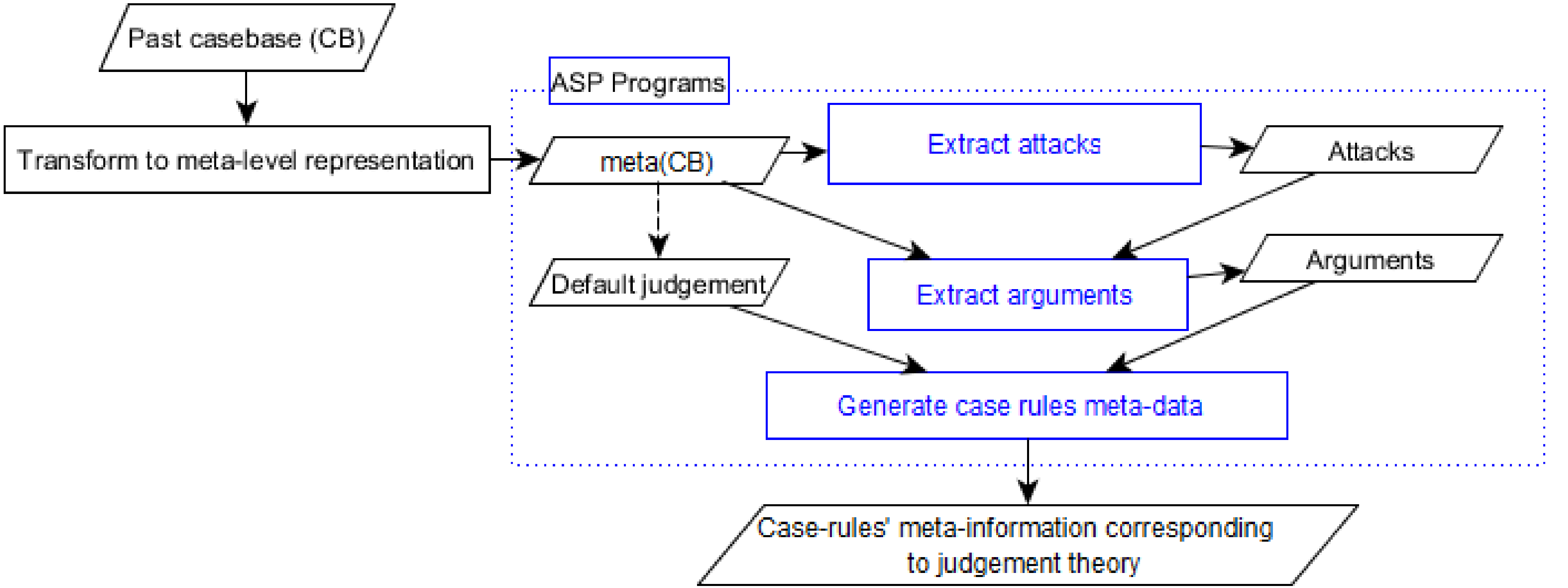
For the working example the judgement theory is:

```
def_j0 :- not ab0, not ab1.
ab0 :- a, not ab2.
ab1 :- c, not ab5.
ab2 :- b, not ab3.
ab3 :- c, not ab4.
ab4 :- d.
```

The predicted judgement of $c_{new} = \{a,b,d\}$ is j_0.

ASP Workflow

Imperial College London NII



The judgement theory is computed using 3 ASP programs

Example

	Factors	Factors	Judgement
cj_0			¬dwe
cj_1	dot		¬dwe
cj_2	dot, ooo		dwe
cj_3	dot, ooo	rpl, far, ria	¬dwe

Past cases with judgement can be seen as “defeasible” rules:

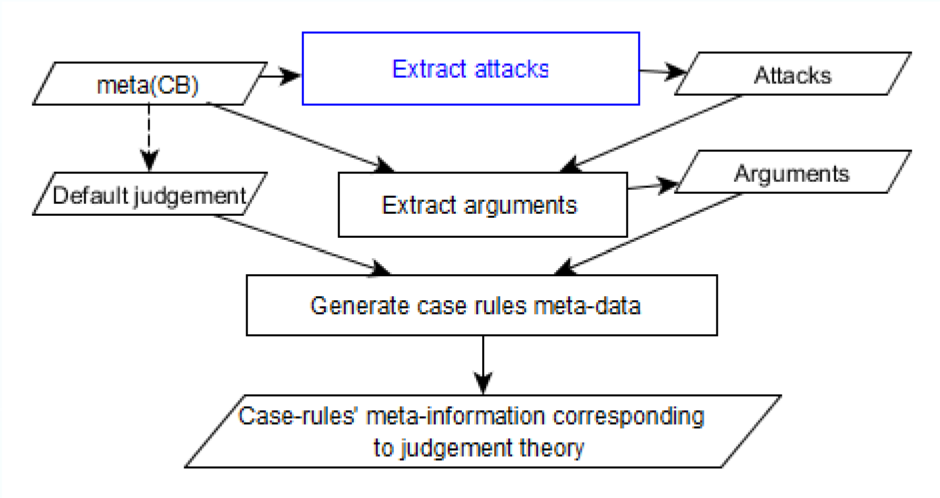
- ¬dwe.
- ¬dwe ←dot.
- dwe ←dot, ooo.
- ¬dwe ←dot, ooo, rpl, far, ria.

Each rule can be encoded in a program as a a set of meta-level information. For example, for ¬dwe ←dot, ooo, rpl, far, ria:

```
cb_id(cj3).
is_rule(cj3,neg_dwe).
in_rule(cj3,neg_dwe,dot).
in_rule(cj3,neg_dwe,ooo).
in_rule(cj3,neg_dwe,rpl).
in_rule(cj3,neg_dwe,far).
in_rule(cj3,neg_dwe,ria).
```

Example

	Factors	Factors	Judgement
cj_0			¬dwe
cj_1	dot		¬dwe
cj_2	dot, ooo		dwe
cj_3	dot, ooo	rpl, far, ria	¬dwe

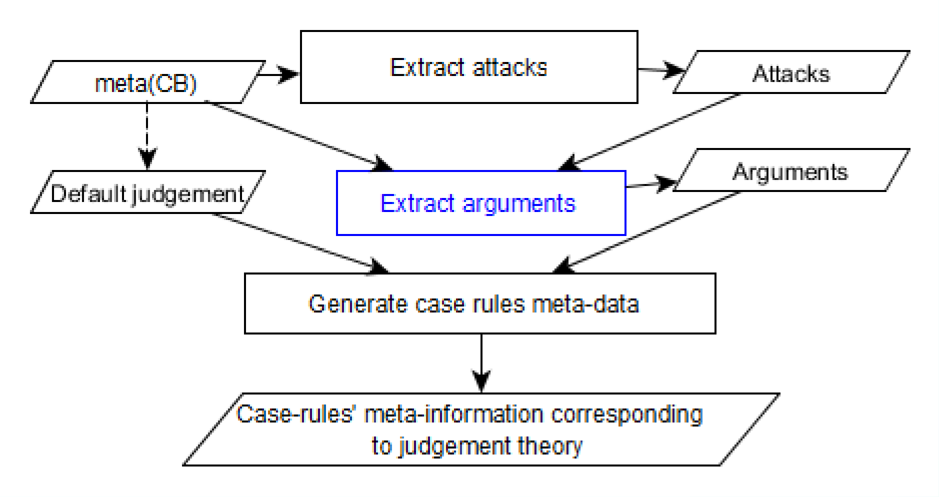


The first program use the casebase to compute the answer set containing all attacks and raw attacks:

{ raw_attack(cj2,c0), raw_attack(cj2,cj1),
raw_attack(cj3,c2), attack(cj2,cj0),
attack(cj3,cj2) }

Example

	Factors	Factors	Judgement
cj_0			¬dwe
cj_1	dot		¬dwe
cj_2	dot, ooo		dwe
cj_3	dot, ooo	rpl, far, ria	¬dwe



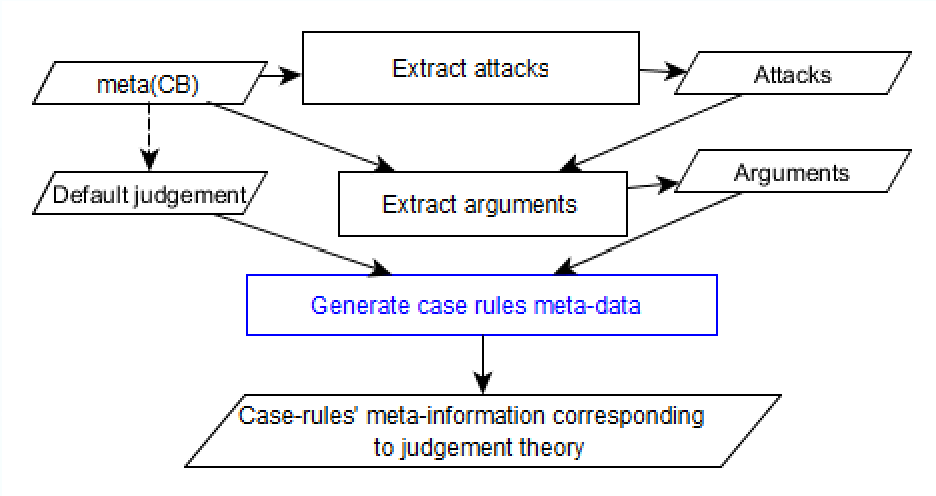
The second program finds the answer set with all arguments:

{ argument(cj2,cj0,dot), argument(cj2,cj0,ooo),
argument(cj3,cj2,rpl), argument(cj3,cj2,far),
argument(cj3,cj2, ria) }

Example



	Factors	Factors	Judgement
cj_0			¬dwe
cj_1	dot		¬dwe
cj_2	dot, ooo		dwe
cj_3	dot, ooo	rpl, far, ria	¬dwe



The final program generates the meta-level representation of the case rules:

{ is_rule(r0,neg_dwe), in_rule(r0,neg_dwe,not_ab0),
in_rule(r1,ab0), in_rule(r1,ab0,dot),
in_rule(r1,ab0,ooo), in_rule(r1,ab0,not_ab1),
is_rule(r2,ab1), in_rule(r2,ab1,rpl),
in_rule(r2,ab1,far), in_rule(r2,ab1,ria) }

Example

The meta-level representation of the judgement theory:

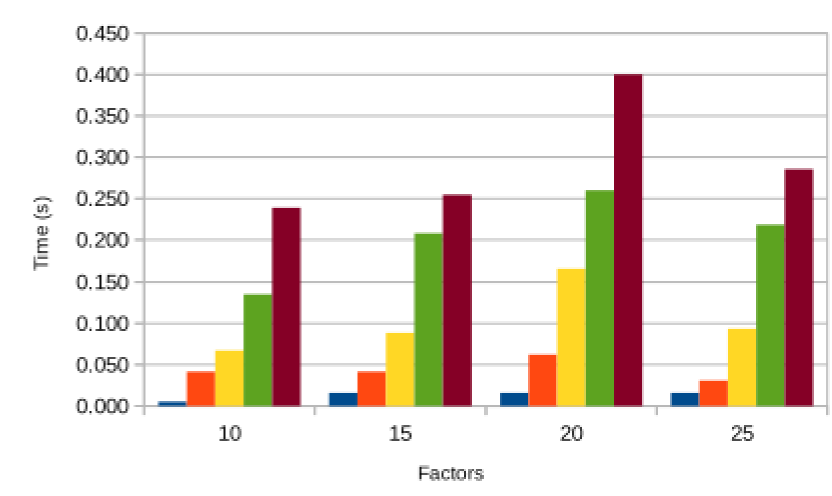
{ is_rule(r0,neg_dwe), in_rule(r0,neg_dwe,not_ab0),
in_rule(r1,ab0), in_rule(r1,ab0,dot),
in_rule(r1,ab0,ooo), in_rule(r1,ab0,not_ab1),
is_rule(r2,ab1), in_rule(r2,ab1,rpl),
in_rule(r2,ab1,far), in_rule(r2,ab1,ria) }

Corresponds to the judgement theory:

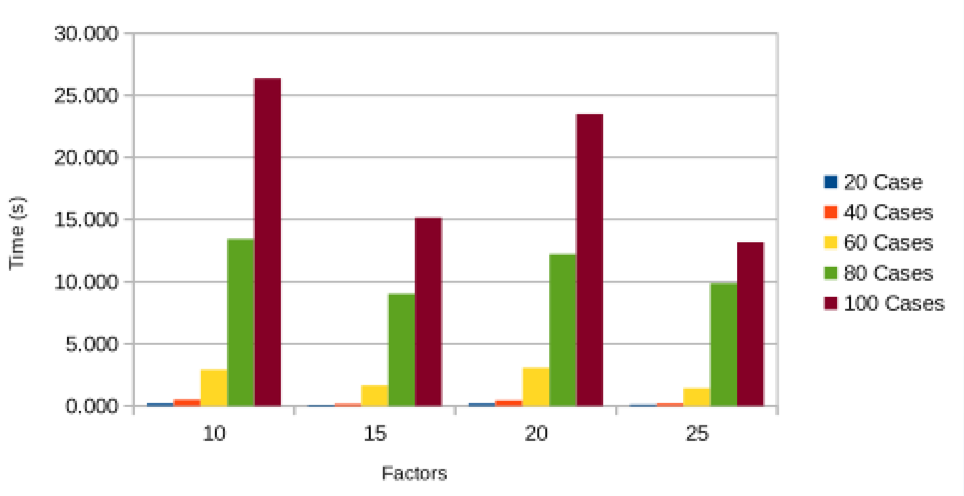
$\neg dwe \leftarrow$ not ab0.
ab0 $\leftarrow$ dot, ooo, not ab1.
ab1 $\leftarrow$ rpl, far, ria.

The predicted judgement of {dot, ooo, rpl, far} is dwe.

Evaluation



Time taken to generate the attacks and arguments



Time taken to generate meta-level representation of judgement theory

Correctness Theorem

Given a casebase *CB, and* associated judgement theory *T*, and a new case *c*. Let *A* = AS(T $\cup$ c).

$$j_0 \in A \text{ if and only if } pj(c) = j_0.$$

Conclusion and Future Work

- We presented a method for reasoning about, and extracting information from, past cases in a casebase to infer the arguments and attacks present in the casebase

- We defined notion of attack to identify factors relevant to judgement of cases and used ASP to infer rules for modelling judgements that can be applied to new cases.

For future work we would like to consider
- how to revise an existing judgement theory
- how to deal with inconsistency among cases
- other possible representations of a casebase