Checking the validity of rule-based arguments grounded in cases: a computational approach

Heng ZHENG [a,1], Minghui XIONG [b] and Bart VERHEIJ [a]
[a] *Artificial Intelligence, University of Groningen, The Netherlands*
[b] *Institute of Logic and Cognition, Sun Yat-sen University, Guangzhou, China*

Abstract. Legal justice needs judges' decisions to be rational and reasonable in the sense that arguments are based on rules and grounded in cases. One puzzle studied in AI & Law is how arguments, rules and cases are formally connected. Recently a formal theory was proposed formalizing how the validity of arguments based on rules can be grounded in cases. Three kinds of argument validity were distinguished: coherence, presumptive validity and conclusiveness. In this paper the theory is implemented in a Prolog program, used to evaluate a previously developed model of Dutch tort law. We also test the theory and its implementation with a new case study modeling Chinese copyright infringement law. In this way we illustrate that by the use of the implementation the process of modeling becomes more efficient and less error-prone.

Keywords. Artificial Intelligence and Law, Rule-based Reasoning, Case-based Reasoning, Argumentation Modeling, Prolog

1. Introduction

In the field of AI and law—going at least back to the 1970s [1]—, scholars usually follow three approaches to develop legal reasoning systems, which are rule-based reasoning, case-based reasoning and argument-based reasoning. In the 1980s, the British Nationality Act (BNA) was implemented in Prolog [2], in a successful attempt to develop rule-based reasoning systems in the field of law. Case-based reasoning was modeled in the systems HYPO, BankXX, CATO and IBP [3, 4, 5, 6, 7]. CABARET [8] and GREBE [9] are hybrid systems not only use case-based reasoning, but also other kinds of reasoning. Argumentation models of legal rule-based and case-based reasoning [10, 11] are connected to later systems based on abstract argumentation [12], inspiring for instance ASPIC+ [13], ABA [14] and DeLP [15].

The recent case model formalism [16] is a hybrid theory showing connections between cases, rules and arguments [17]. The formalism defines different ways in

[1]Corresponding Author: Artificial Intelligence, University of Groningen, The Netherlands; E-mail:zhengh48@mail2.sysu.edu.cn.

which rule-based arguments can be valid in cases: arguments can be coherent, conclusive or presumptive. The formalism has been applied to model Dutch tort law, showing how a rule-based legal domain can be grounded in legal cases. In this way, a formal connection is established between the civil law tradition focusing on rules and the common law tradition focusing on cases. The formalism has also been applied to the modeling of a series of New York tort cases (as studied by [18, 19]), analyzing value-guided teleological reasoning [20].

The present paper provides a computational version of the case model formalism. A Prolog program is presented that can computationally check whether a case model is correct, whether rule-based arguments are valid (in the three kinds of validity coherence, conclusiveness and presumptiveness), and whether defeating circumstances are rebutting, undercutting or undermining. The computational tool can be used to support the manual modeling of a complex legal domain, making that more manageable. As an example, we provide a new domain model, namely Chinese copyright infringement law, both formally (as a case model) and computationally (in Prolog).

2. The case model formalism

The case model formalism was introduced in [16]. The formalism uses a classical formal logical language L generated from a finite set of propositional constants in a standard way writing $\neg$ for negation, $\wedge$ for conjunction, $\vee$ for disjunction, $\leftrightarrow$ for equivalence, $\top$ for a tautology, and $\bot$ for a contradiction. The associated classical, deductive, monotonic consequence relation is denoted $\vDash$.

Case models formalize cases and their ordering. The cases in a case model must be logically consistent, mutually incompatible and different; and the comparison relation must be total and transitive. Here follow the core definitions.

Definition 2.1: A *case model* is a pair $(C, \geq)$ with finite $C \in L$, such that the following hold, for all φ, ψ and $\chi \in C$:

1. $\nvDash \neg\varphi$;
2. If $\nvDash \varphi \leftrightarrow \psi$, then $\vDash \neg\ (\varphi \wedge \psi)$;
3. If $\vDash \varphi \leftrightarrow \psi$, then $\varphi = \psi$;
4. $\varphi \geq \psi$ or $\psi \geq \varphi$;
5. If $\varphi \geq \psi$ and $\psi \geq \chi$, then $\varphi \geq \chi$.

Definition 2.2 (*Arguments*) An *argument* is a pair (φ, ψ) with φ and $\psi \in L$. The sentence φ expresses the argument's premises, the sentence ψ its conlusions, and the sentence $\varphi \wedge \psi$ the *case made* by the arguments. Generalizing, a sentence $\chi \in L$ is a *premise* of the argument when $\varphi \vDash \chi$, a *conclusion* when $\psi \vDash \chi$, and a *position* in the case made by the argument when $\varphi \wedge \psi \vDash \chi$. An argument (φ, ψ) is *(properly) presumptive* when $\varphi \nvDash \psi$; otherwise *not-presumptive*. An argument (φ, ψ) is a *presumption* when $\vDash \varphi$, i.e., when its premises are logically tautologous.

Definition 2.3 *(Coherent arguments)* Let $(C, \geq)$ be a case model. Then we define, for all φ and $\psi \in L$:

$(C, \geq) \vDash (\varphi, \psi)$ if and only if $\exists \omega \in C : \omega \in \varphi \wedge \psi$.

We then say that the argument from φ to ψ is *coherent* with respect to the case model.

Definition 2.4 (*Conclusive arguments*) Let $(C, \geq)$ be a case model. Then we define, for all φ and $\psi \in L$:

$(C, \geq) \models \varphi \Rightarrow \psi$ if and only if $\exists \omega \in C : \omega \in \varphi \wedge \psi$ and $\forall \omega \in C$: if $\omega \in \varphi$, then $\omega \in \varphi \wedge \psi$.

We then say that the argument from φ to ψ is *conclusive* with respect to the case model.

Definition 2.5 (*Presumptively valid arguments*) Let $(C, \geq)$ be a case model. Then we define, for all φ and $\psi \in L$:

$(C, \geq) \models \varphi \rightsquigarrow \psi$ if and only if $\exists \omega \in C$:

1. $\omega \models \varphi \wedge \psi$; and
2. $\forall \omega' \in C$: if $\omega' \models \varphi$, then $\omega \geq \omega'$.

We then say that the argument from φ to ψ is *(presumptively) valid* with respect to the case model.

Definition 2.6 (*Successful attack*) Let $(C, \geq)$ be a case model, and (φ, ψ) a presumptively valid argument. Then circumstances χ are *defeating* or *successful attacking* the argument when $(\varphi \wedge \chi, \psi)$ is not presumptively valid. We write $(C, \geq) \models \varphi \rightsquigarrow \psi \times \chi$. Defeating circumstances are *excluding* when $(\varphi \wedge \chi, \psi)$ is not coherent. A case $\omega \in C$ provides grounding for the attack if $\omega \models \varphi \wedge \chi$.

Definition 2.7 (*Rebutting attack*) When circumstance χ successfully attack presumptively valid argument (φ, ψ), the circumstances are rebutting when $(\varphi \wedge \chi, \neg\psi)$ is presumptively valid.

Definition 2.8 (*Undercutting attack*) When circumstances χ successfully attack presumptively valid argument (φ, ψ), and are not rebutting, the circumstances are *undercutting*.

Definition 2.9 (*Undermining attack*) When circumstances χ successfully attack a presumption $(\top, \varphi)$, the circumstances are *undermining*.

3. Implementation in Prolog

We have implemented the case model formalism in Prolog. We use the previously developed model of Dutch tort law [17] as an illustration. Cases are represented as Prolog lists, of which the elements consist of strings and their negations (represented using `not/1`). For instance, `[not(dut),dmg,unl,imp,not(cau)]` represents the case in which there is no duty (`not(dut)`) to repair the damages (`dmg`) although the act is unlawful (`unl`) and imputable (`imp`]), but there is no causality (`not(cau)`). Case models are represented as lists of cases and their ordering, where case models and cases are referred to using identifiers.

Listing 1 provides a part of the representation of the case model for Dutch tort law (`model_num(1)`). The representation of the case model starts with a series of definitions of cases followed by the ordering. Here `case_num(104)` is the non-causality case just discussed. The full model consists of the 16 cases discussed in [17]. The ordering is represented as a list of cases and lists of cases, each representing an equivalence class of the total preorder, in decreasing level of preference. Hence the first element of this list represents the case or cases that are maximal in the total preorder. Here there is one maximal case `case_num(101)`: `not(dmg)`, representing that there are no damages. When an equivalence class consists of

```
case(model_num(1),case_num(101),[not(dmg)]).
...
case(model_num(1),case_num(104),[not(dut),dmg,unl,imp,not(cau)]).
case(model_num(1),case_num(105),[dut,dmg,unl,imp,cau,vrt,not(vst),not(vun),ift
    ,not(ila),not(ico),not(jus),prp]).
...
case(model_num(1),case_num(114),[not(dut),dmg,not(unl),vrt,not(vst),jus]).
...
case_order(model_num(1),[case_num(101),case_num(102),case_num(103),case_num
    (104),[case_num(105),case_num(106),...,case_num(113)],...]).
```

Listing 1: Definition of the Dutch tort law case model in Prolog

several cases, such as [case_num(105),...,case_num(113)], it is represented as a list of cases (actually: of case identifiers).

The predicate case_model_valid/1 (with a case identifier as single argument) checks whether a case model fulfills the definition, i.e. whether all cases are consistent, incompatible and different (cf. Definition 2.1 above). Consistency is determined by checking whether a case contains an element and its negation. For instance, a case [not(dmg), dmg] would not be consistent. This straightforward way of consistency checking is allowed by our use of a language of elementary literals instead of a full propositional language. Incompatibility is checked by determining for every pair of cases whether there is an element in one case of which the negation is in the other. For instance, cases case_num(101): not(dmg) and case_num(104): not(dut),dmg,... can be distinguished by not(dmg) and dmg, hence are incompatible. Difference is checked by determining for every pair of cases whether there is an element in one case that is not in the other. Note that a pair of incompatible cases implies that they are different. Checking elements 4 and 5 in the formal definition of case models is not directly representd since we use an explicit representation of the total preorder as a list of equivalence classes. In this representation another validity check is helpful, namely whether each case of which an identifier appears in the ordering is defined and whether each case identifier of a defined case appears in the ordering exactly once. This check has been implemented using the predicate ordering_valid/1.

Arguments are represented by a list of premises and a list of conclusions. For instance, argument([dmg,unl,imp,cau],[dut]) represents the argument to the one-element conclusion that there is a duty to repair the damages (dut) from the four-element premises that there are damages (dmg) by an unlawful act (unl) that is imputable (imp) to the actor and that has caused the damages (cau).

The three kinds of validity of arguments can be checked using the predicates coherent/1, conclusive/1 and presumptively_valid/1, each taking an argument (represented using argument/2) as input. Coherence is checked by first determining the case made by an argument, found by appending the list of premises to the list of conclusions, and then checking whether there is a case in the case model that contains all elements of the case made by the argument. For instance, argument([dmg,unl,imp,cau],[dut])) is coherent by the case with identifier 105 whereas argument([dmg],[not(dmg)]) is not as its case made is not consistent.

The conclusiveness of an argument is checked by first checking whether the argument is coherent and then checking, if all cases in the case model that con-

tain the argument's premises also contain its conclusions. For instance, argument([dmg],[dut]) is coherent (e.g., by case 105) but not conclusive (by case 104). The argument argument([dut],[dmg]) is conclusive since it is coherent (e.g., by 105) and indeed all cases of dut are also cases of dmg.

Presumptive validity of an argument is checked by first determining a maximally preferred case that witnesses the argument's coherence (i.e., finding a case in which both the premises and the conclusions of the argument hold and that is maximal in the ordering with this property) and then checking for each case in which the argument's premises hold whether such a witnessing case is of equal or lower ordering. For instance, argument([vrt],[unl]) is presumptively valid since for instance case 105 is a case of maximal ordering that shows the arguments coherence, and there is no case of the premises that is higher in the ordering. The coherent argument argument([vrt],[not(unl)]) is not presumptively valid since maximally ordered cases that show the coherence of the argument (such as case 114) are not maximal among the cases in which the premises hold (e.g., case 105 is higher in the ordering).

Attack of arguments has been implemented using the predicates successful_attack/2, rebutting_attack/2, undercutting_attack/2 and undermining_attack/2. The predicate successful_attack/2 takes an argument and defeating circumstances (as a list) as input. The predicate checks whether the argument is presumptively valid and whether the argument to the same conclusions but from the premises with the defeating circumstances appended is also presumptively valid. For instance, since argument([vrt],[unl]) is presumptively valid, while argument([vrt, jus],[unl]) is not (it is not even coherent), the Prolog query ?– successful_attack(argument([vrt],[unl]), [jus]) returns true as an answer.

The three kinds of attack—rebutting, undercutting and undermining—are defined in terms of successful attack as follows. Rebutting attack requires a successful attack of an argument such that also the argument from the premises to the opposite of the argument's conclusion is presumptively valid. Note that this only makes sense for arguments with a single element as conclusion as we use no Prolog expression for the negation of a series of conclusions. For instance, rebutting_attack(argument([vrt],[unl]), [jus]) holds since it represents a successful attack and also argument([vrt],[not(unl)]) is presumptively valid. Successful attacks that are not rebutting attacks are undercutting. Undermining attack is a special kind of successful attack, namely an attack of an argument with a tautology as premise. In the program, a tautology is represented as an empty Prolog list [ ].

The Prolog program can be used to validate hand-made domain models, such as the case model for Dutch tort law of [17]. In Figure 1, we show the argument structure that is valid in the hand-made formal case model of that paper (left). On the right, we show Prolog clauses that all evaluate to true given the Prolog version of that case model (partially shown in Listing 1). Each supporting arrow (with a normal arrow head) evaluates as a presumptively valid argument, as expected, some even more strongly as a conclusive argument. Each attacking arrow (ending in a cross) evaluates as a successful attack, in fact all attacks in this model are rebutting attacks. In other words, the model is computationally validated.

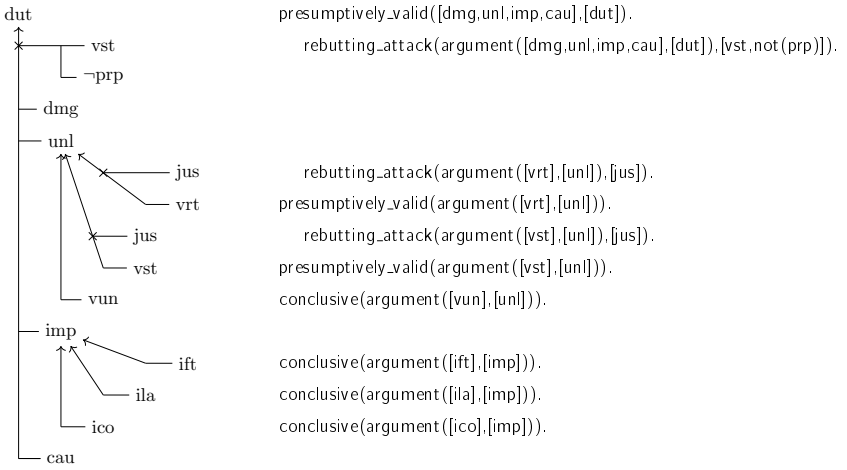


Figure 1. The Dutch tort law model: argument structure (left); in Prolog (right)

4. Case study: Copyright infringement in Chinese Criminal Law

The article of Copyright Infringement in Chinese Criminal Law [21] is below:

> **Article 217** Whoever, for the purpose of making profits, commits any of the following acts of infringement on copyright shall, if the amount of illegal gains is relatively large, or if there are other serious circumstances, be sentenced to fixed-term imprisonment of not more than three years or criminal detention and shall also, or shall only, be fined; if the amount of illegal gains is huge or if there are other especially serious circumstances, he shall be sentenced to fixed-term imprisonment of not less than three years but not more than seven years and shall also be fined:
> (1) copying and publishing a written work, musical work, motion picture, television programme or other visual works, computer software or other works without permission of the copyright owner;
> (2) publishing a book of which another person has the exclusive publishing right;
> (3) copying and publishing audio or video recording without permission of the producer; or
> (4) producing or selling an artwork where the signature of the author is forged.

In Art. 217, there are 4 kinds of act in copyright infringement, if someone violates other people's copyright for the purpose of making profits, then he will be sentenced to the crime of copyright infringement. The judge will sentence the person to one of 4 different kinds of punishment according to the degree of severity of his crime. According to Art. 217's relevant judicial explanations, there are 3 defeating circumstances: 1. The action is not "without permission of the copyright owner"; 2. The defendant is sentenced to the crime of copyright infringement, however, he also satisfies with the conditions of being given a heavier punishment; 3. The defendant is sentenced to the crime of copyright infringement, however, he also satisfies with the conditions of being given a lighter punishment.

In the light of Art. 217 and the judicial explanations related to it, a case model can be built. In our copyright infringement model, we use the elementary propositions in Table 1, shown with their formal abbreviations. The full model has 46 cases. A selection of the cases is shown in its Prolog version in Listing 2. The

Table 1. Elementary propositions in the copyright infringement model; with abbreviations

ifg	there is a copyright infringement
fpp	the act was for the purpose of making profits
pac	the act was publish and copy
ite	the act concerned the items in Art. 217:1
pco	the act was without permission of the copyright owner
pec	the act was not "without permission of the copyright owner"
epr	the act was publishing a book that has another person's exclusive publishing right
avp	the act was audio or video recording of which the producer is someone else
psa	the act was producing or selling an artwork with a forged author signature
ils	the amount of illegal gains is large or other serious circumstances
ihe	the amount of illegal gains is huge or other especially serious circumstances
crc	the person commits the crime of copyright infringement
l3fti	the person shall be sentenced to fixed-term imprisonment of at most three years
cdt	the person shall be sentenced to criminal detention
fin	the person shall be fined
m3fti	the person shall be sentenced to fixed-term imprisonment of not less than three years but not more than seven years
hps	there is a reason which will give the defendant a heavier punishment
lps	there is a reason which will give the defendant a lighter punishment
cpb	the defendant satisfies the conditions of probation
pbt	the defendant will be put on probation

model has identifier model_num(2). In the text below, cases are numbered 1, 2, 3, ... corresponding to cases 201, 202, 203, ... in the Prolog version. Case 1 is built by the principle of "presumption of innocence". Case 2 is for the first defeating circumstance mentioned above. Case 3 shows the scenario that the defendant violated someone else's copyright, but he didn't do it for making profits. From Case 3 to Case 13, different punishments for the defendant's action in Art. 217:1 are listed. In the same way, different punishments for the defendant in Art. 217:2, Art. 217:3 and Art. 217:4 are listed in the full case model. Thus Case 14 to Case 46 have similar components with Case 4 to Case 13, except the acts of infringement on copyright are different. In this model, Case 1 is maximal in the ordering, as the "presumption of innocence" is the starting principle in the process of decision making. Besides, in some situations, the defendant will not be regarded as violating someone else's copyright, for instance, Case 2 and Case 3; hence they appear in second place. The rest of cases in the preferred relation are the specific punishments of copyright infringement, they are less preferred than the cases in first and second place.

From Chinese copyright infringement, we can analyze the argument structure as in the diagram in Figure 2 (left). This argument structure shows multiple rule-based steps and exception-based attacks. The structure is valid in the case model we built, as follows.

Following the definitions of the case model formalism (Section 2), the rule $\text{ifg} \wedge \text{fpp} \Rightarrow \text{crc}$ is indeed valid in this model, in different ways. As the model shows, this argument is coherent, as all cases imply sentence $\text{ifg} \wedge \text{fpp} \wedge \text{crc}$ which is the case made by the argument $(\text{ifg} \wedge \text{fpp}, \text{crc})$. Besides, all the cases which imply the

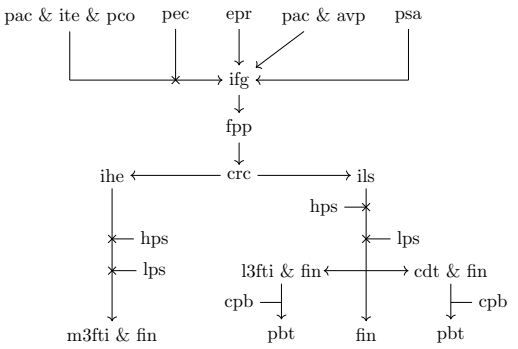
```
case(model_num(2),case_num(201),[not(pac),not(ite),not(pco),not(epr),not(avp),
     not(psa),not(ifg)]).
case(model_num(2),case_num(202),[pac,ite,pco,pec,not(ifg)]).
case(model_num(2),case_num(203),[pac,ite,pco,not(pec),not(epr),not(avp),not(psa)
     ,ifg,not(fpp)]).
case(model_num(2),case_num(204),[pac,ite,pco,not(epr),not(avp),not(psa),ifg,fpp,
     ihe,not(ils),crc,hps,not(lps),not(m3fti),not(l3fti),not(cdt),not(fin)]).
case(model_num(2),case_num(205),[pac,ite,pco,not(epr),not(avp),not(psa),ifg,fpp,
     ihe,not(ils),crc,not(hps),lps,not(m3fti),not(l3fti),not(cdt),not(fin)]).
case(model_num(2),case_num(206),[pac,ite,pco,not(epr),not(avp),not(psa),ifg,fpp,
     ihe,not(ils),crc,not(hps),not(lps),m3fti,not(l3fti),not(cdt),fin]).
case(model_num(2),case_num(207),[pac,ite,pco,not(epr),not(avp),not(psa),ifg,fpp,
     not(ihe),ils,crc,hps,not(lps),not(m3fti),not(l3fti),not(cdt),not(fin)]).
case(model_num(2),case_num(208),[pac,ite,pco,not(epr),not(avp),not(psa),ifg,fpp,
     not(ihe),ils,crc,not(hps),lps,not(m3fti),not(l3fti),not(cdt),not(fin)]).
case(model_num(2),case_num(209),[pac,ite,pco,not(epr),not(avp),not(psa),ifg,fpp,
     not(ihe),ils,crc,not(hps),not(lps),not(m3fti),not(l3fti),not(cdt),fin]).
case(model_num(2),case_num(210),[pac,ite,pco,not(epr),not(avp),not(psa),ifg,fpp,
     not(ihe),ils,crc,not(hps),not(lps),not(m3fti),l3fti,not(cdt),fin,cpb,pbt]).
case(model_num(2),case_num(211),[pac,ite,pco,not(epr),not(avp),not(psa),ifg,fpp,
     not(ihe),ils,crc,not(hps),not(lps),not(m3fti),not(l3fti),cdt,fin,cpb,pbt]).
case(model_num(2),case_num(212),[pac,ite,pco,not(epr),not(avp),not(psa),ifg,fpp,
     not(ihe),ils,crc,not(hps),not(lps),not(m3fti),l3fti,not(cdt),fin,not(cpb),
     not(pbt)]).
case(model_num(2),case_num(213),[pac,ite,pco,not(epr),not(avp),not(psa),ifg,fpp,
     not(ihe),ils,crc,not(hps),not(lps),not(m3fti),not(l3fti),cdt,fin,not(cpb),
     not(pbt)]).
...
case_order(model_num(2),[case_num(201),[case_num(202),case_num(203), ...],[
     case_num(204),case_num(205),case_num(207),case_num(208), ... ,case_num(206)
     ,case_num(209),case_num(210),case_num(211),case_num(212),case_num(213),
     ...]]).
```

Listing 2: The Chinese copyright infringement case model in Prolog (selection)

premises ifg ∧ fpp, also imply the conclusion crc. So argument (ifg ∧ fpp, crc) is also conclusive in the case model, and hence it is presumptively valid. And these arguments are also conclusive in the model:

$(C, \geq) \models \text{crc} \wedge \text{l3fti} \wedge \text{fin} \wedge \text{cpb} \Rightarrow \text{pbt}$
$(C, \geq) \models \text{crc} \wedge \text{cdt} \wedge \text{fin} \wedge \text{cpb} \Rightarrow \text{pbt}$

Argument (pac∧ite∧pco, ifg) is presumptively valid. In the copyright infringement model, this argument is coherent as Case 2 implies pac ∧ ite ∧ pco ∧ ifg, the case made by the argument. As Case 2 is also the strongest case among the cases




```
presumptively_valid(argument([pac,ite,pco],[ifg])).
presumptively_valid(argument([crc],[ils])).

rebutting_attack(argument([pac,ite,pco],[ifg]),[pec]).
rebutting_attack(argument([crc,ils],[l3fti,fin],[hps])).
rebutting_attack(argument([crc,ihe],[m3fti,fin],[lps])).

conclusive(argument([ifg,fpp],[crc])).
conclusive(argument([crc,l3fti,fin,cpb],[pbt])).
conclusive(argument([crc,cdt,fin,cpb],[pbt])).
```

Figure 2. The Chinese copyright infringement model: argument structure (left); in Prolog (right)

which imply the premises of this argument, the argument $(\text{pac} \wedge \text{ite} \wedge \text{pco}, \text{ifg})$ is presumptively valid in the case model. It is not conclusive, as the defeating circumstance pec successfully attacks the argument. Since the case model makes the argument $(\text{pac} \wedge \text{ite} \wedge \text{pco} \wedge \text{pec}, \neg\text{ifg})$ presumptively valid, this defeating circumstance is rebutting. In the same way, the following hold in the case model.

$(C, \subseteq) \models \text{crc} \rightsquigarrow \text{ils}$
$(C, \subseteq) \models \text{crc} \wedge \text{ils} \rightsquigarrow \text{l3fti} \wedge \text{fin} \times \text{hps}$
$(C, \subseteq) \models \text{crc} \wedge \text{ils} \rightsquigarrow \text{cdt} \wedge \text{fin} \times \text{hps}$
$(C, \subseteq) \models \text{crc} \wedge \text{ihe} \rightsquigarrow \text{m3fti} \wedge \text{fin} \times \text{lps}$

The Prolog program confirms the validity of the arguments above. Figure 2 (right) lists Prolog queries evaluated as true, which means the results of the Prolog program correspond to our analysis of the Chinese copyright infringement model.

5. Discussion and Conclusion

The Prolog program we introduced in this paper can support the process of modeling legal domains and its main function is automatically verifying the validity of the arguments implicated in a case model and the case model itself. This program has been shown to be suitable for both the Dutch tort law model presented before ([17]) and the new Chinese copyright infringement model. Compared with the original hand-made modeling way, the program supports the reduction of mistakes made during the process of modeling, as this program can verify the model automatically. In this way, we have presented the first computational implementation for the case model formalism which can support making the theoretical approach more applicable in a practical environment.

A remaining puzzle is to control the number of cases in case models. For those people who are unfamiliar with legal statues, an explicit case model can be helpful for them, which means the model needs to contain as much details as possible, such as the official judicial explanations related to Chinese criminal law. But case models can require large numbers of cases. For instance, there are already 8 scenarios about the elementary proposition "amount of illegal gains is huge or other especially serious circumstances" mentioned in the Chinese copyright infringement model in the light of relevant judicial explanations. If all of such specific scenarios are added into the model, the number of cases in this model quickly increases. According to the way of building a model completely based on the case model formalism, all these scenarios will be treated as elementary propositions and they will replace the position of ihe. The number of cases increase by eight times, if we added these specific scenarios to the model directly, which is error prone when the list of cases is developed manually. In future research, keeping the number of cases in control could be investigated, e.g., by providing computational means that do not require the explicit listing of all possible cases.

The results of this paper shows that an implementation of the case model formalism can be used to support the modeling of a legal domain with a complex argument structure involving combined support and attack. In this way, we have shown a computational connection between cases, rules and arguments, applied

to the civil law system of the Netherlands and to criminal law in the Chinese legal system. In this way, we contribute to AI and legal reasoning technology that is needed to combine rule-based reasoning, case-based reasoning and argumentation together, paving the way for argumentation technology that bridges cases and rules, as it is common in the law.

References

[1] B. G. Buchanan and T. E. Headrick. Some speculation about artificial intelligence and legal reasoning. *Stanford Law Review*, 23(1):40–62, 1970.

[2] M. J. Sergot, F. Sadri, R. A. Kowalski, et al. The British nationality act as a logic program. *Communications of the ACM*, 29(5):370–386, 1986.

[3] E. L. Rissland and K. D. Ashley. A case-based system for trade secrets law. In *Proceedings of the First International Conference on Artificial Intelligence and Law (ICAIL 1987)*, pages 60–66. ACM Press, New York (New York), 1987.

[4] K. D. Ashley. Reasoning with cases and hypotheticals in HYPO. *International Journal of Man-Machine Studies*, 34(6):753–796, 1991.

[5] E. L. Rissland, D. B. Skalak, and M. T. Friedman. BankXX: Supporting legal arguments through heuristic retrieval. *Artificial Intelligence and Law*, 4(1):1–71, 1996.

[6] V. Aleven and K. D. Ashley. Evaluating a learning environment for case-based argumentation skills. In *International Conference on Artificial Intelligence and Law*, pages 170–179, 1997.

[7] S. Brüninghaus and K. D. Ashley. Predicting outcomes of case based legal arguments. In *International Conference on Artificial Intelligence and Law*, pages 233–242, 2003.

[8] E. L. Rissland and D. B. Skalak. CABARET: rule interpretation in a hybrid architecture. *International Journal of Man-Machine Studies*, 34(6):839–887, 1991.

[9] L. K. Branting. Building explanations from rules and structured cases. *International Journal of Man-Machine Studies*, 34(6):797–837, 1991.

[10] H. Prakken and G. Sartor. A dialectical model of assessing conflicting arguments in legal reasoning. *Artificial Intelligence and Law*, 4:331–368, 1996.

[11] H. Prakken and G. Sartor. Modelling reasoning with precedents in a formal dialogue game. *Artificial Intelligence and Law*, 6:231–287, 1998.

[12] P. M. Dung. On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games. *Artificial Intelligence*, 77(2):321–357, 1995.

[13] H. Prakken. An abstract framework for argumentation with structured arguments. *Argument and Computation*, 1(2):93–124, 2010.

[14] A. Bondarenko, P. M. Dung, R. A. Kowalski, and F. Toni. An abstract, argumentation-theoretic approach to default reasoning. *Artificial Intelligence*, 93(s 1–2):63–101, 1997.

[15] A. J. García and G. R. Simari. Defeasible logic programming: an argumentative approach. *Theory and Practice of Logic Programming*, 4(1):95–138, 2003.

[16] B. Verheij. Correct grounded reasoning with presumptive arguments. In L. Michael and A. Kakas, editors, *15th European Conference on Logics in Artificial Intelligence, JELIA 2016. Larnaca, Cyprus, November 9–11, 2016. Proceedings (LNAI 10021)*. Springer, Berlin, 2016.

[17] B. Verheij. Formalizing arguments, rules and cases. In *Proceedings of the Seventeenth International Conference on Artificial Intelligence and Law (ICAIL 2017)*, 2017.

[18] D. H. Berman and C. L. Hafner. Understanding precedents in a temporal context of evolving legal doctrine. In *Proceedings of the Fifth International Conference on Artificial Intelligence and Law*, pages 42–51. ACM Press, New York (New York), 1995.

[19] C. L. Hafner and D. H. Berman. The role of context in case-based legal reasoning: Teleological, temporal, and procedural. *Artificial Intelligence and Law*, 10(1–3):19–64, 2002.

[20] B. Verheij. Formalizing value-guided argumentation for ethical systems design. *Artificial Intelligence and Law*, 24(4):387–407, 2016.

[21] The Legislative Affairs Office of the State Council. *Series of Statute of the People's Republic of China in English*. China Legal Publishing House, 2015.