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

Heng ZHENG [a], Minghui XIONG [b] and Bart VERHEIJ [a,1]
[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, presumptiveness and conclusiveness. In this paper the theory is implemented in a Prolog program. We test the theory using two case studies: the first is a model of Dutch tort law developed earlier; the second is a newly developed model of Chinese copyright infringement law. The case studies 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:bart.verheij@rug.nl.

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 $\psi \in L$:

$(C, \geq) \vDash \varphi \Rightarrow \psi$ 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 $\psi \in L$:

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

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

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

Verheij has also defined some kinds of argument attacking in his formalism[17].

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 $(C, \geq) \vDash \varphi \rightsquigarrow \psi \times \chi$. Defeating circumstances are *excluding* when $(\varphi \wedge \chi, \psi)$ is not coherent. A case $\omega \in C$ provides grounding for the attack if $\omega \vDash \varphi \wedge \chi$.

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

The program introduced in this section was written in Prolog. Case models $(C, \geq)$ are used predicate case_model(model_num(N)) to represent. Each model will be given a number N to distinguish. It is implied by a predicate for cases and a predicate for the preferred relation. In this program, cases in the model are represented in the form of Prolog *list*, as well as the preferred relation. Predicate case(model_num(N), case_num(X), List) is used for describing cases. Similarly, each case in the case model will be given a number X, and List contains all the elementary propositions of the case. Predicate case_order has 2 variables, including model_num(N) and the list Case_order_list shows the preferred relation of cases in the model.

Case_order_list lists cases from strong to weak, if one case is as preferred as another case, then these two cases will be placed in a sublist of Case_order_list. For instance, the preferred relation of a case model with 4 cases is: $case1 > case3 \sim case4 > case2$. This order will be listed as [case_num(1), [case_num(3), case_num(4)], case_num(2)]. It is noticed that, predicate case_order must be distinguished with case_list which is a predicate for the set of cases in the case model without the preferred relation.

The program uses predicate case_model_valid which is implied by a set of predicates to check if the inputted models are consistent, incompatible and different follows the formal definition in Verheij's theory (cf. Definition 2.1).

Predicate case_model_consistent is used for judging the consistency of case models. In this process, we use predicate consistent_case to check if there exists a case in the case list that its negation is still in the list. Predicate consistent_list is implied by consistent_case, which is used for checking whether the whole case list exists the situation that happened in consistent_case. Predicate case_model_ incompatible is used to check if the model is incompatible, it is implied by predicate incompatible_case which aims for checking every two cases in one case list is incompatible. Predicate case_model_ different is used for checking the difference of cases in the case model, if there exists two cases the same as each other, the program will return false.

The main function of this program is verifying the validity of the arguments in case models. The Prolog predicates we define are based on the relevant definitions in Verheij's case model formalism.

The program uses predicate coherent to examine coherent arguments. This predicate is implied by coherent_casemade, which is a recursive predicate that uses member_list to check if the *case made* by the argument in queries is logically implied by at least one case in the case list.

```
coherent_casemade(Premise, Conclusion, [Case_num|Other_cases]) :-
        case(Case_num, Case),
        append(Premise, Conclusion, Casemade),
        member_list(Casemade, Case);
        coherent_casemade(Premise, Conclusion, Other_cases).

coherent(argument(Premise, Conclusion)) :-
        case_list(Case_list),
        coherent_casemade(Premise, Conclusion, Case_list).
```

This program uses predicate conclusive to verify the argument (φ, ψ) in the case model $(C, \geq)$ that considered to be conclusive. According to Verheij's formalism, a conclusive argument must be coherent. In addition, if all cases in the model imply the argument's premises also imply its conclusions. In this sense, we use predicate conclusive_casemade to check if a case implies the argument's premises, whether it also implies the *case made* by the argument. For each case in the model, we use predicate conclusive_case_check to examine if it is a case implies both the premises of the argument and the case made by the argument or it doesn't imply the premises of the argument. The implication of this predicate is conclusive_list, a recursive predicate, which uses conclusive_case_check to check every case in the case list. If the result of this predicate is true, then we can say all cases in the case model imply the premises of the argument also imply its conclusions.

```
case_with_casemade(Premise, Conclusion, Case_num) :-
     case(Case_num, Case),
     append(Premise, Conclusion, Casemade),
     member_list(Premise, Case),
     member_list(Casemade, Case).
```

```
conclusive_case_check(Premise,Conclusion,Case_num) :-
    case_with_casemade(Premise,Conclusion,Case_num);
    case(Case_num,Case),
    not(member_list(Premise,Case)).

conclusive_case_list_check(Premise,Conclusion,[Case_num|Other_cases]) :-
      conclusive_case_check(Premise,Conclusion,Case_num),
      conclusive_case_list_check(Premise,Conclusion,Other_cases).

conclusive(argument(Premise,Conclusion)) :-
        coherent(argument(Premise,Conclusion)),
        case_list(Case_list),
        conclusive_case_list_check(Premise,Conclusion,Case_list).
```

The presumptively valid argument (φ, ψ) in the case model $(C, \geq)$ will use predicate presumptively_valid to verify. The program uses coherent to check if the argument is coherent. Predicate best_case_casemade is going to select the most preferred case in the model which contains the case made by the argument. This predicate uses best_case_casemade_basic to check whether the case implies the case made by the argument, if so, variable Best_case_casemade will be assigned a value of the number of this case, i.e., Case_num.

```
best_case_casemade_basic(Premise,Conclusion,Case_num,Best_case_casemade) :-
    case(Case_num,Case),
    append(Premise,Conclusion,Casemade),
    member_list(Casemade,Case),
    Best_case_casemade = Case_num.

best_case_casemade(Premise,Conclusion,[Case_num|Other_cases],Best_case_casemade)
    :-
    best_case_casemade_basic(Premise,Conclusion,Case_num,Best_case_casemade);
    best_case_casemade(Premise,Conclusion,Other_cases,Best_case_casemade).
```

About the most preferred case implied the case made, according to the definition of case order, there is a situation must be considered, that is some cases in the model are as preferred as this case, which means these cases are in a Prolog *sublist*. Predicate best_case_premise is used to compare the best case implied the case made by the argument (i.e. Best_case_casemade) with the cases implied the premises of the argument. In terms of the definition of *presumptively valid arguments*, if the argument is presumptively valid, then the case represented by Best_case_casemade should be at least as preferred as all cases implied the premises of the argument. As mentioned above, in best_case_premise, there are 3 situations can let the program return true,

1. The most preferred case implied the premises is the same as the best case implied the case made, and this case is an element of the case order list;
2. The most preferred case implied the premises is the same or as preferred as the best case implied the case made, and these cases are in a sublist of the case order list;
3. All other cases implied the premises are less preferred than the best case implied the case made.

Predicate best_case_premise_sublist, best_case_premise_element and their implication best_case_premise_basic are used to deal with the third situation:

- If the Element is not the best case implied the case made, best_case_premise_element will check if it contains the premises of the argument;
- If the Element is a sublist, and the best case implied the case made is not a member of it, best_case_premise_sublist will make sure that all cases in this sublist do not contain the premises of the argument.

```
comparison_premise_sublist(Premise,[Case_num|Other_cases]) :-
        is_list([Case_num|Other_cases]),
        case(Case_num,Case),
        not(member_list(Premise,Case)),
        comparison_premise_sublist(Premise,Other_ cases).

comparison_premise_case(Premise,Case_num) :-
        case(Case_num,Case),
        not(member_list(Premise,Case)).

comparison_premise_basic(Premise,Element) :-
   comparison_premise_sublist(Premise,Element);
   comparison_premise_case(Premise,Element).

comparison_premise(Best_case_casemade,Premise,[Element|Other_cases]) :-
     Best_case_casemade = Element;
     member(Best_case_casemade,Element);
     comparison_premise_basic(Premise,Element),
     comparison_premise(Best_case_casemade,Premise,Other_cases).
```

Predicate presumptively_valid is implied by a series of predicates. Through this predicate, we can use coherent to check the coherence of the argument, and then use best_case_casemade to select the most preferred case implied the case made by the argument, and compare this case with other cases implied the premises which can be implemented by comparison_premise.

```
presumptively_valid(argument(Premise,Conclusion)) :-
         coherent(argument(Premise,Conclusion)),
         case_order(Case_order),
         case_list(Case_list),
         best_case_casemade(Premise,Conclusion,Case_list,Best_case_casemade),
         comparison_premise(Best_case_casemade,Premise,Case_order).
```

For an argument (φ, ψ) with defeating circumstances χ in the case model $(C, \geq)$, we use predicate successful_attack to check if it is a successful attack. According to the definition of *successful attack*, argument (φ, ψ) must be presumptively valid. When the defeating circumstances χ are added, the argument $(\varphi \wedge \chi, \psi)$ will become invalid. We can used predicate presumptively_valid to implement this definition.

```
successful_attack(argument(Premise,Conclusion),Defeating_circumstance) :-
           append(Premise,Defeating_circumstance,Premise_and_defeating),
           presumptively_valid(argument(Premise,Conclusion)),
           not(presumptively_valid(argument(Premise_and_defeating,Conclusion))).
```

Predicate rebutting_attack is used to distinguish rebutting attack, which is implied by successful_attack and presumtively_valid. If circumstances χ successfully

attacking the argument (φ, ψ), and argument $(\varphi \wedge \chi, \neg\psi)$ is also presumptively valid, then we can say this is a rebutting attack. In this process, we use negation_list to turn the conclusions into its negation. Predicate undercutting_attack is for undercutting attack, which is the negation of predicate rebutting_attack.

```
rebutting_attack(argument(Premise,Conclusion),Defeating_circumstance) :-
    successful_attack(argument(Premise,Conclusion),Defeating_circumstance),
    negation_list(Conclusion,Neg_Conclusion),
    append(Premise,Defeating_circumstance,Premise_and_defeating),
    presumptively_valid(argument(Premise_and_defeating,Neg_Conclusion)).

undercutting_attack(argument(Premise,Conclusion),Defeating_circumstance) :-
    not(rebutting_attack(argument(Premise,Conclusion),Defeating_circumstance)).
```

Undermining attack is a special kind of successful attack, which attacks presumption $(\top, \varphi)$. In this program, a tautology $\top$ is represented as an empty Prolog *list* [ ], so predicate undermining_attack is not only implied by successful_attack, but also a predicate which is used to check if the list of premises is empty.

```
undermining_attack(argument(Premise,Conclusion),Defeating_circumstance) :-
      presumption(argument(Premise,Conclusion)),
      successful_attack(argument(Premise,Conclusion),Defeating_circumstance).
```

4. Case study: Dutch tort law

The case model of Dutch tort law was built by Verheij in 2017[17]. This model applied two articles 6:162 and 6:163 of the Dutch civil code (in the Netherlands referred to as Art. 6:162 and 6:163 BW, BW for 'Burgerlijk Werboek') govern the handling of wrongful acts. Here follows the translation by [21]:

- **Art. 6:612 BW.** 1. A person who commits an unlawful act toward another which can be imputed to him, must repair the damage which the other person suffers as a consequence thereof.
  2. Except where there is a ground of justification, the following acts are deemed to be unlawful: the violation of a right, an act or omission violating a statutory duty or a rule of unwritten law pertaining to proper social conduct.
  3. An unlawful act can be imputed to its author if it results from his fault or from a cause for which he is answerable according to law or common opinion.
- **Art. 6:613 BW.** There is no obligation to repair damage when the violated norm does not have as its purpose the protection from damage such as that suffered by the victim.

Verheij built a case model for these two articles in his paper presented in ICAIL 2017[17], he gave each elementary proposition in the case model an abbreviation. These propositions will be listed in Table 1.

Table 1. Elementary propositions for the tort law domain

dut	There is a duty to repair someone's damages
dmg	Someone has suffered damages by someone else's act.
unl	The act committed was unlawful
imp	The act can be imputed to the person that committed the act
cau	The act caused the suffered damages
vrt	The act is a violation of someones right.
vst	The act is a violation of a statutory duty
vun	The act is a violation of unwritten law against proper social conduct
jus	There exist grounds of justification
ift	The act is imputable to someone because of the person's fault
ila	The act is imputable to someone because of law
ico	The act is imputable to someone because of common opinion
prp	The violated statutory duty does not have the purpose to prevent the damages

Table 2. The case list of the Dutch tort law model

1	¬dmg
2	¬dut, dmg, ¬unl, ¬vrt, ¬vst, ¬vun
3	¬dut, dmg, unl, ¬imp, ¬ift, ¬ila, ¬ico
4	¬dut, dmg, unl, imp, ¬cau
5	dut, dmg, unl, imp, cau, vrt, ¬vst, ¬vun, ift, ¬ila, ¬ico, ¬jus, prp
6	dut, dmg, unl, imp, cau, vrt, ¬vst, ¬vun, ¬ift, ila, ¬ico, ¬jus, prp
7	dut, dmg, unl, imp, cau, vrt, ¬vst, ¬vun, ¬ift, ¬ila, ico, ¬jus, prp
8	dut, dmg, unl, imp, cau, ¬vrt, vst, ¬vun, ift, ¬ila, ¬ico, ¬jus, prp
9	dut, dmg, unl, imp, cau, ¬vrt, vst, ¬vun, ¬ift, ila, ¬ico, ¬jus, prp
10	dut, dmg, unl, imp, cau, ¬vrt, vst, ¬vun, ¬ift, ¬ila, ico, ¬jus, prp
11	dut, dmg, unl, imp, cau, ¬vrt, ¬vst, vun, ift, ¬ila, ¬ico, ¬jus, prp
12	dut, dmg, unl, imp, cau, ¬vrt, ¬vst, vun, ¬ift, ila, ¬ico, ¬jus, prp
13	dut, dmg, unl, imp, cau, ¬vrt, ¬vst, vun, ¬ift, ¬ila, ico, ¬jus, prp
14	¬dut, dmg, ¬unl, vrt, ¬vst, jus
15	¬dut, dmg, ¬unl, ¬vrt, vst, jus
16	¬dut, dmg, unl, imp, cau, vst, ¬prp
Order	case 1 > case 2 > case 3 > case 4 > case 5 ∼ case 6 ∼ case 7 ∼ case 8 ∼ case 9 ∼ case 10 ∼ case 11 ∼ case 12 ∼ case 13 > case 14 ∼ case 15 ∼ case 16

There are 16 cases in this model which can be found in table 2.

Figure 1 shows the arguments extracted from this model, according to the formalism's definitions, a series of arguments can be generated:

- $(C, \geq) \models dmg \wedge unl \wedge imp \wedge cau \rightsquigarrow dut \times vst \wedge \neg prp$
- $(C, \geq) \models vrt \rightsquigarrow unl \times jus$
- $(C, \geq) \models vst \rightsquigarrow unl \times jus$
- $(C, \geq) \models vun \rightsquigarrow unl$
- $(C, \geq) \models ift \rightsquigarrow imp$
- $(C, \geq) \models ila \rightsquigarrow imp$
- $(C, \geq) \models ico \rightsquigarrow imp$

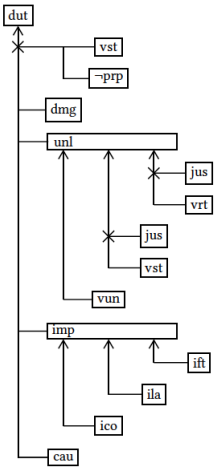

Figure 1. Arguments and their attacks in the case model of Dutch tort law

The Dutch tort law model in the Prolog program is represented as model_num(1). Through the verification given by predicate case_model_valid, this model is a valid model.

```
?- case_model_valid(model_num(1)).
true.
```

In this program, we use predicate successful_attack to check the validity of the successful attack $dmg \wedge unl \wedge imp \wedge cau \rightsquigarrow dut \times vst \wedge \neg prp$ in the model. As the program shows, this attack is a rebutting attack, which means it is not undercutting. This judgement is corresponded to the analysis about the Dutch tort law model above.

```
?- successful_attack(argument([dmg,unl,imp,cau],[dut]),[vst,not(prp)]).
true.

?- rebutting_attack(argument([dmg,unl,imp,cau],[dut]),[vst,not(prp)]).
true.

?- undercutting_attack(argument([dmg,unl,imp,cau],[dut]),[vst,not(prp)]).
false.
```

In the same way, other arguments and attacks shown above can be successfully verified in the Prolog program.

```
?- successful_attack(argument([vrt],[unl]),[jus]).
true.

?- successful_attack(argument([vst],[unl]),[jus]).
true.

?- presumptively_valid(argument([vun],[unl])).
```

```
true.

?- presumptively_valid(argument([ift],[imp])).
true.

?- presumptively_valid(argument([ila],[imp])).
true.

?- presumptively_valid(argument([ico],[imp])).
true.
```

```
case(model_num(1),case_num(101),C) :- C = [not(dmg)].
case(model_num(1),case_num(102),C) :- C = [not(dut),dmg,not(unl),not(vrt),not(
    vst),not(vun)].
case(model_num(1),case_num(103),C) :- C = [not(dut),dmg,unl,not(imp),not(ift),
    not(ila),not(ico)].
case(model_num(1),case_num(104),C) :- C = [not(dut),dmg,unl,imp,not(cau)].
case(model_num(1),case_num(105),C) :- C = [dut,dmg,unl,imp,cau,vrt,not(vst),not(
    vun),ift,not(ila),not(ico),not(jus),prp].
case(model_num(1),case_num(106),C) :- C = [dut,dmg,unl,imp,cau,vrt,not(vst),not(
    vun),not(ift),ila,not(ico),not(jus),prp].
case(model_num(1),case_num(107),C) :- C = [dut,dmg,unl,imp,cau,vrt,not(vst),not(
    vun),not(ift),not(ila),ico,not(jus),prp].
case(model_num(1),case_num(108),C) :- C = [dut,dmg,unl,imp,cau,not(vrt),vst,not(
    vun),ift,not(ila),not(ico),not(jus),prp].
case(model_num(1),case_num(109),C) :- C = [dut,dmg,unl,imp,cau,not(vrt),vst,not(
    vun),not(ift),ila,not(ico),not(jus),prp].
case(model_num(1),case_num(110),C) :- C = [dut,dmg,unl,imp,cau,not(vrt),vst,not(
    vun),not(ift),not(ila),ico,not(jus),prp].
case(model_num(1),case_num(111),C) :- C = [dut,dmg,unl,imp,cau,not(vrt),not(vst)
    ,vun,ift,not(ila),not(ico),not(jus),prp].
case(model_num(1),case_num(112),C) :- C = [dut,dmg,unl,imp,cau,not(vrt),not(vst)
    ,vun,not(ift),ila,not(ico),not(jus),prp].
case(model_num(1),case_num(113),C) :- C = [dut,dmg,unl,imp,cau,not(vrt),not(vst)
    ,vun,not(ift),not(ila),ico,not(jus),prp].
case(model_num(1),case_num(114),C) :- C = [not(dut),dmg,not(unl),vrt,not(vst),
    jus].
case(model_num(1),case_num(115),C) :- C = [not(dut),dmg,not(unl),not(vrt),vst,
    jus].
case(model_num(1),case_num(116),C) :- C = [not(dut),dmg,unl,imp,cau,vst,not(prp)
    ].

case_order(model_num(1),Case_order) :- Case_order = [case_num(101),case_num(102)
    ,case_num(103),case_num(104),[case_num(105),case_num(106),case_num(107),
    case_num(108),case_num(109),case_num(110),case_num(111),case_num(112),
    case_num(113)],[case_num(114),case_num(115),case_num(116)]].
```

5. Case study: Copyright infringement in Chinese Criminal Law

The article of Copyright Infringement in Chinese Criminal Law[22] 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 order to recognize the propositions in the case model easily, each elementary proposition in this model has been given an abbreviation. These abbreviations are shown in table 3.

Table 3. Elementary propositions in the case model of copyright infringement

ifg	copyright infringement
fpp	for the purpose of making profits
pac	publish and copy
ite	the items in Art. 217:1
pco	without permission of the copyright owner
pec	the action is not belong to "without permission of the copyright owner"
epr	publishing a book of which another person has the exclusive publishing right
avp	the audio or video recording which the producer is someone else
psa	producing or selling an artwork where the signature of the author is forged
ils	amount of illegal gains is large or other serious circumstances
ihe	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 not more than 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

In Art. 217, there are 4 kinds of situation 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 sentenced him to 4 different kinds of punishment according to the degree of severity of his crime. Above all, several rules about copyright infringement can be extracted.

If the defendant has following actions:

1. *publish or copied something which can be considered as one of the items shown in Art. 217:1 without permission of the copyright owner*;
2. *publish a book of which another person has the exclusive publishing right*;

3. *publish or copied the audio or video resording which the producer is someone else*;
4. *produce or sell an artwork where the signature of the author if forged*,

Then the defendant will be regarded as violating someone else's copyright.

If the defendant was regarded as violating someone else's copyright for the purpose of making profits, then the defendant will be sentenced to the crime of copyright infringement. This rule can be represented as $ifg \wedge fpp \Rightarrow crc$.

If *the amount of defendant's illegal gains was large or existed other serious circumstances*, the defendant shall be sentenced to 3 kinds of punishments, *fixed-term imprisonment of not more than 3 years and fined* ($crc \wedge ils \rightsquigarrow l3fti \wedge fin$); *criminal detention and fined* ($crc \wedge ils \rightsquigarrow cdt \wedge fin$) and only *fined* ($crc \wedge ils \rightsquigarrow fin$).

If *the amount of defendant's illegal gains was huge or existed other especially serious circumstances*, then the defendant shall be sentenced to *fixed-term imprisonment of not less that three years but not more than seven years and fined*, this can be represented as $crc \wedge ihe \Rightarrow m3fti \wedge fin$.

There are two kinds of punishments which are possible to be put on probation, if the defendant satisfied the conditions of probation: 1. fixed-term imprisonment of not more than 3 years and fined ($crc \wedge l3fti \wedge fin \wedge cpb \Rightarrow pbt$); 2. criminal detention and fined ($crc \wedge cdt \wedge fin \wedge cpb \Rightarrow pbt$).

According to Art. 217's relevant judicial explanations, there are 3 defeating circumstance: 1. The action is not belong to "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.

If we add these defeating circumstance to the rules we listed above, then some of the rules will be changed. For example,

- $pac \wedge ite \wedge pco \rightsquigarrow ifg \times pec$
- $crc \wedge ils \rightsquigarrow l3fti \wedge fin \times hps$
- $crc \wedge ihe \Rightarrow m3fti \wedge fin \times lps$

In the light of Art. 217 and the judicial explanations related to it, a case model can be built. The model has 46 cases. Case 1 is built by the principle of "presumption of innocence". Case 2 shows the scenario that although the defendant has published and copied the items shown in Art. 217:1 without the permission of the copyright owner, he still will not be considered as copyright infringement because his action is not belong to "without permission of the copyright owner". Case 3 shows the scenario that the defendant violated someone else's copyright, but he didn't do it for making profits, so he will not be judge through Art. 217.

From Case 4 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 will also be listed into the case model. Table 4 lists cases from Case 1 to Case 13. 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 the most preferred case, as we consider the "presumption of innocence" is the most important 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, the defendant's action is not belong to "without permission of the copyright owner" or his purpose is not for making profits. In this model, these situations are also very important. So we put them in second place. The rest of cases in the preferred relation are the specific punishments of copyright infringement, we put them in third place.

Table 4. The case list of the case model

1	¬pac, ¬ite, ¬pco, ¬epr, ¬avp, ¬psa, ¬ifg
2	pac, ite, pco, pec, ¬ifg
3	pac, ite, pco, ¬pec, ¬epr, ¬avp, ¬psa, ifg, ¬fpp
4	pac, ite, pco, ¬epr, ¬avp, ¬psa, ifg, fpp, ihe, ¬ils, crc, hps, ¬lps, ¬m3fti, ¬l3fti, ¬cdt, ¬fin
5	pac, ite, pco, ¬epr, ¬avp, ¬psa, ifg, fpp, ihe, ¬ils, crc, ¬hps, lps, ¬m3fti, ¬l3fti, ¬cdt, ¬fin
6	pac, ite, pco, ¬epr, ¬avp, ¬psa, ifg, fpp, ihe, ¬ils, crc, ¬hps, ¬lps, m3fti, ¬l3fti, ¬cdt, fin
7	pac, ite, pco, ¬epr, ¬avp, ¬psa, ifg, fpp, ¬ihe, ils, crc, hps, ¬lps, ¬m3fti, ¬l3fti, ¬cdt, ¬fin
8	pac, ite, pco, ¬epr, ¬avp, ¬psa, ifg, fpp, ¬ihe, ils, crc, ¬hps, lps, ¬m3fti, ¬l3fti, ¬cdt, ¬fin
9	pac, ite, pco, ¬epr, ¬avp, ¬psa, ifg, fpp, ¬ihe, ils, crc, ¬hps, ¬lps, ¬m3fti, ¬l3fti, ¬cdt, fin
10	pac, ite, pco, ¬epr, ¬avp, ¬psa, ifg, fpp, ¬ihe, ils, crc, ¬hps, ¬lps, ¬m3fti, l3fti, ¬cdt, fin, cpb, pbt
11	pac, ite, pco, ¬epr, ¬avp, ¬psa, ifg, fpp, ¬ihe, ils, crc, ¬hps, ¬lps, ¬m3fti, ¬l3fti, cdt, fin, cpb, pbt
12	pac, ite, pco, ¬epr, ¬avp, ¬psa, ifg, fpp, ¬ihe, ils, crc, ¬hps, ¬lps, ¬m3fti, l3fti, ¬cdt, fin, ¬cpb, ¬pbt
13	pac, ite, pco, ¬epr, ¬avp, ¬psa, ifg, fpp, ¬ihe, ils, crc, ¬hps, ¬lps, ¬m3fti, ¬l3fti, cdt, fin, ¬cpb, ¬pbt
14	... ...
Order	case 1 > case 2 = case 3 = case 14 = case 25 = case 36 > case 4 = case 5 = case 7 = case 8 = case 15 = case 16 = case 18 = case 19 = case 26 = case 27 = case 29 = case 30 = case 37 = case 38 = case 40 = case 41 = case 6 = case 9 = case 10 = case 11 = case 12 = case 13 = case 17 = case 20 = case 21 = case 22 = case 23 = case 24 = case 28 = case 31 = case 32 = case 33 = case 34 = case 35 = case 39 = case 42 = case 43 = case 44 = case 45 = case 46

From this copyright infringement model, we can get the argument diagram illustrated in Figure 2. This argument has multi-steps, and it is corresponding to the model we built.

According to the definitions of Verheij's case model formalism, the rule $epr \Rightarrow ifg$ is valid in this model. As the model shows, case 14 and case 15 implicate sentence $epr \wedge ifg$ which is the case made by the argument (epr, ifg), so this argument is coherent. Besides, all the cases which imply the premise epr, also imply the conclusion ifg. So argument (epr, ifg) is conclusive in the case model. And these arguments are also conclusive in the model:

- $(C, \geq) \models pac \wedge avp \Rightarrow ifg$
- $(C, \geq) \models ifg \wedge fpp \Rightarrow crc$

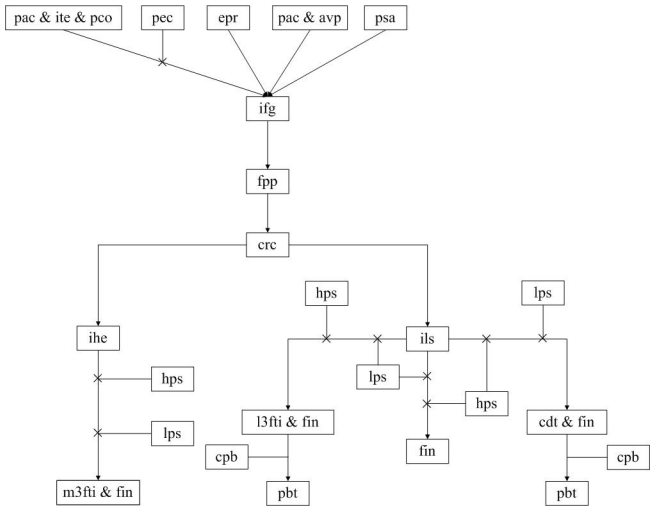

Figure 2. Arguments and their attacks in the case model of Chinese copyright infringement

- $(C, \geq) \models crc \wedge l3fti \wedge fin \wedge cpb \Rightarrow pbt$

According to the definitions of Verheij's case model formalism, the rule $pac \wedge ite \wedge pco \rightsquigarrow ifg \times pec$ is also valid in the copyright infringement model. The attack from pec successfully attacked the presumptively valid argument $(pac \wedge ite \wedge pco, ifg)$, and made the argument $(pac \wedge ite \wedge pco \wedge pec, \neg ifg)$ presumptively valid. In the light of the copyright infringement model, Case 2 has implied the case made by the argument $(pac \wedge ite \wedge pco \wedge pec, \neg ifg)$, so this argument is coherent. Furthermore, Case 2 is the strongest case in the cases which implied the premise of this argument. So we say the argument $(pac \wedge ite \wedge pco \wedge pec, \neg ifg)$ is presumptively valid in the case model. The arguments below are also presumptively valid:

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

The Chinese copyright infringement model is represented as model_num(2) in the program. The following query shows that it is a valid model. Also, the arguments and attacks shown above can be verified by this program.

```
?- case_model_valid(model_num(2)).
true.

?- successful_attack(argument([pac,ite,pco],[ifg]),[pec]).
true.

?- rebutting_attack(argument([pac,ite,pco],[ifg]),[pec]).
true.

?- undercutting_attack(argument([pac,ite,pco],[ifg]),[pec]).
false.

?- successful_attack(argument([crc,ils],[l3fti,fin]),[hps]).
true.

?- successful_attack(argument([crc,ils],[cdt,fin]),[hps]).
true.

?-successful_attack(argument([crc,ihe],[m3fti,fin]),[lps]).
true.
```

```
?- conclusive(argument([epr],[ifg])).
true.

?- conclusive(argument([pac,avp],[ifg])).
true.

?- conclusive(argument([ifg,fpp],[crc])).
true.

?- conclusive(argument([crc,l3fti,fin,cpb],[pbt])).
true.
```

As the queries shows, the program has verified that circumstance *pec* successfully attacks argument $pac \wedge ite \wedge pco \rightsquigarrow ifg$, and it is a rebutting attack, as well as the other arguments with defeating circumstances above which are also considered to be successful. The program also finds argument $epr \Rightarrow ifg$ and other three arguments to be conclusive. These results are corresponded to the analysis about Chinese copyright infringement model above.

6. Discussion and Conclusion

In this paper, two case models with different backgrounds based on Verheij's case model formalism have been discussed. As the arguments implied in these case models are corresponded to the rules in statutes, Verheij's theory has been proved that it is suitable for Chinese legal system. The preferred relation established by Verheij solves the priority relationship among the cases in case model, so that the presumptive arguments in legal reasoning can be concluded by this theory. In other words, some legal issues can be solved, such as the rules used for legal reasoning, which are beyond the statutes, for instance, "presumption of innocence" in the background of Chinese legal system. Although it is not in any statute, every judge will think about it during the process of making decisions. So we can place this rule as several cases in the preferred relation of a case model properly to solve this issue. Therefore, we believe arguments embedded values are more suitable for dealing with practical issues in legal reasoning.

The program we build in this paper intends to make the modeling process with less mistakes 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 proved that it is completely suitable for those models based on Verheij's theory, even these models have different backgrounds. Compare with the original hand-made modeling way, it can reduce mistakes during the process of modeling significantly, as this program can verify the model automatically. So, it is a successful attempt as a computational implementation for Verheij's case model formalism which can make Verheij's theory more useable in practical environment.

However, there is still room for improvement. 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 this action can bring a

problem cannot be ignored. For instance, there are 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 these specific scenarios are added into the model, the number of cases in this model will become huge. According to the way of building a model completely based on Verheij's formalism, all these scenarios will be treated as elementary propositions and they will replace the position of *ihe*. The number of cases will increase by eight times, if we added these specific scenarios to the model by Verheij's theory directly, which also lets the process of modeling easily to make mistakes. It will be a hard job for the people who wants to build a model. This issue needed to be solved in the future research.

The results of this paper shows that Verheij's case model formalism can be used to model the object with complex argument structure, and the program developed in this paper is feasible for the models based on Verheij's theory. It also proves that the Prolog program we built can well combine cases, rules and arguments. It is not only applicable to the civil law system but also to the Chinese legal system. AI and legal reasoning technology needs to combine rule-based reasoning, case-based reasoning and argumentation together, and argumentation technology can be the bridge of both cases and rules.

References

[1] Bruce G. Buchanan and Thomas 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, F Kriwaczek, P Hammond, and H. T Cory. 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*, pages 60–66. ACM Press, New York (New York), 1987.

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

[5] Edwina L. Rissland, David B. Skalak, and M. Timur Friedman. Bankxx: Supporting legal arguments through heuristic retrieval. *Artificial Intelligence and Law*, 4(1):1–71, 1996.

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

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

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

[9] L. Karl 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] Phan Minh 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] Henry 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] Alejandro J. García and Guillermo R. Simari. Defeasible logic programming: an argumentative approach. *Theory and Practice of Logic Programming*, 4(1):95–138, 2003.

[16] Bart. Verheij. *Correct Grounded Reasoning with Presumptive Arguments*. Springer International Publishing, 2016.

[17] Bart Verheij. Formalizing arguments, rules and cases. In *Seventeenth International Conference on Artificial Intelligence and Law*, 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] Bart. Verheij. Formalizing value-guided argumentation for ethical systems design. *Artificial Intelligence and Law*, 24(4):387–407, 2016.

[21] Gerrit Betlem. Civil liability for transfrontier pollution. *Graham and Trotman, London*, 1993.

[22] 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.