International Journal on Artificial Intelligence Tools
Vol. 21, No. 1 (2012) 1250002 (34 pages)

DOI: 10.1142/S0218213012500029

MoDiSo: A TOOL FOR BUILDING CONTRACT DISPUTE RESOLUTION SYSTEMS

NGUYEN DUY HUNG*, PHAN MINH THANG† and PHAN MINH DUNG‡

Department of Computer Science, Asian Institute of Technology
GPO Box 4, Klong Luang, Pathumthani 12120, Thailand
**hung.nd9@gmail.com*
†*thangfm@gmail.com*
‡*dung.phanminh@gmail.com*

Received 29 April 2010
Accepted 10 October 2011

Real-world dispute resolution should be guided by laws, even if such disputes may be resolved by bodies other than the court of laws. Hence in order to build contract dispute resolution systems we need a tool capable of representing, reasoning and programming with contract laws. In this paper we present such a tool called MoDiSo (MOdular Argumentation for DIspute ReSOlution[a]) which combines the strengths of state-of-the-art argumentation-based techniques for different aspects of law, to propose: first, a modular architecture for contract dispute resolution systems with an edit-compile-dispute loop facilitating incremental system developments; and second, a methodology to represent and reason with legal doctrines in contract laws in the formal language of assumption-based argumentation. We demonstrate the tool with several legal doctrines for performance relief in common law of contracts. As a by-product, we obtain a dispute resolution system capable of explaining legal outcomes by automatically generating relevant arguments.

Keywords: Modular argumentation; dispute resolution; legal doctrines; common law.

1. Motivation, Background, and System Introduction

Contracts are often specified only partially. A contract dispute arises when contract parties have *conflicting* interpretations of their contract on their legal rights and obligations. Since commerce is not possible if contract parties can arbitrarily adopt their *preferred* interpretations, contract dispute resolution is indispensable in any commercial world.

Real-world contract dispute resolution should be guided by contract laws, even if such disputes may be resolved by bodies other than the court of laws. Hence in order to build support systems for contract dispute resolution, we need a tool capable of representing, reasoning and programming with contract laws. Such a tool would

[a]To our best knowledge, MoDiSo is the first tool of its kind in the literature. A demo of a prototype could be found at http://cs.ait.ac.th/∼dung/modiso/About.html.

be of great interest to, for example, the stakeholders of Online Alternative Dispute Resolution (ODR), a widely accepted form of dispute resolution whereby disagreeing parties, facilitated by technologies, come to an agreement short of litigation while are still able to seek legal redress in actual courts.

In many countries, contract laws are regulated by common law, the kind of law developed through decisions in courts. Common law has case-by-case basis in that it generally proceeds by distilling from a particular case the legal principle on which it is decided, and that legal principle is then generally applied to similar dispute contexts.[47] Hence case-based reasoning could be viewed as a key method of reasoning in legal domains. A case-based system may attempt to perform several tasks: generating legal principles from decided cases, representing those principles and reasoning with them in the concerned dispute context. An appealing methodology for generating legal principles presented in Refs. 5, 7–9 based on the idea that competitive generated legal principles should be decided by taking into account the preference over the social values they advance. However case-based systems (e.g. TAXMAN,[33] HYPO,[3,46] CATO,[2] CABERET,[44] CATO's reconstruction in Carneades[25]) seem not to scale up well since the interpretation of cases and extracting the legal principles on which they were decided is an extremely difficult task even for legal experts, because the vast and increasing number of cases lead to many conflicting decisions and an increased uncertainty in common law. Moreover, none of these systems deal with contract dispute resolution.

In the real world, to address this problem Restatements (First and Second) of Contracts have been proposed to "restate" clearly and precisely the principles and rules of common law[50] to make the interpretation of cases much easier and less arbitrary. Restatements of Contracts are accepted widely in America. In England codifying Acts have been used to impose coherence on specific areas of common law, for example Sale of Goods Acts 1893 and 1979.[1] Legal doctrines in these legal sources could be viewed as a widely accepted interpretation of the principles, guidelines and rules for dispute resolution. For illustration consider the following hypothetical E-commerce dispute.

Example 1.1. A GIS company has a contract with a fishing company to map areas rich of fish around a sea port from images of Landsat satellites. The GIS company uses images of Landsat 7, but after the contract signing the Scan Line Corrector of this satellite fails unexpectedly, causing about 22 percent of any scene lost.[b] The supplied map hence blanks out a significant part. Without reliable information about the fishing ground, the fishing company suffers a bad season. It sues the GIS company.

The GIS company asks for relief of performance, citing *the doctrine of impossibility* (analysed thoroughly in this paper), on grounds that it cannot render the contracted service because Landsat 7 functioned improperly.

[b]The Scan Line Corrector of Landsat 7 failed on May 31, 2003 (http://landsat.usgs.gov).

The court prefers accepting legal doctrines that have been well established in stable legal sources to restating the law for new doctrines. This preference holds even more strongly in civil-law judiciary systems where the court only follows predetermined legal rules. This suggests that a promising approach to build contract dispute resolution systems is to model well-established legal doctrines. Though less ambitious than approaches that also attempt to extract legal doctrines from decided cases, our approach seems easier to accomplish and run lower risk of making hypothetically legal arguments. In our related papers,[18,19] we have showed that the approach can be formally modelled by a two-level modular argumentation process: at object level factors of the case are established from argumentation modules representing the concerned dispute context while at meta-level argumentation modules representing legal doctrines and principles combine these factors to derive arguments for/against legal outcomes. In this paper, we present an accompanied tool of the above formalism called MoDiSo, providing a programming environment to build support systems for contract dispute resolution, for example, systems supporting consumers to structure their complaints, supporting business organizations to construct arguments to address consumers's complaints, or supporting disputants to prepare arguments to be presented before the court. Note that MoDiSo is not geared towards systems that attempt to substitute a judge or lawyer in deciding which contract party wins the dispute presented before the court. From this perspective, MoDiSo would be of great interest to stakeholders of Online Alternative Dispute Resolution.

Combining the strengths of state-of-the-art argumentation techniques for different aspects of law, MoDiSo also makes two practical contributions: first, a modular architecture for contract dispute resolution systems with an edit-compile-dispute loop facilitating incremental system developments; and second, a methodology to represent and reason with legal doctrines in contract law (e.g. Restatements First and Second) in the formal language of assumption-based argumentation. For space limitation in this paper we demonstrate MoDiSo mainly with two doctrines: the first is the doctrine of absolute contracts which insists on the literal execution of contractual obligations regardless of what happens after the contract making; and the second is the doctrine of impossibility which says that contractual obligations may be discharged by supervening events where these have brought about a change of circumstances so significant as to destroy a basic assumption which the parties had made when they entered into the contract.[29] As a by-product of the demonstration, we obtain an executable and scalable dispute resolution system capable of explaining legal outcomes in contract disputes concerning performance relief. The system automatically generates relevant arguments by simulating the exchanges of arguments between contract parties by building and exploring a dialectical structure of the plaintiff's initial argument for a claim, and defendant's counter-arguments attacking this argument, and the plaintiff's arguments attacking all the defendant's arguments, and so on.

To our best knowledge, there has not been much work done on contract dispute resolution. Exceptions are the formalism of Refs. 53 and 54 using meta-level rules in first order logic to deduce contractual obligations and the rule-based system of Ref. 24 supporting decision makers, both in the offer-and-acceptance area of contract. In general these formalisms do not model possible outcomes of a contract dispute. We hence believe that MoDiSo is the first of its kind.

In AI much work has been done to study computational models for many aspects of law. One of the earliest legal reasoning systems is the rule-based system of Sergot *et al.*[49] determining whether an applicant is eligible for British citizenship based on logic rules encoding the British Nationality Act. Though representing a major application of logic programming as a tool for constructing legal expert systems, the proposal was criticised as jurisprudentially not convincing,[32] partly because it does not consider the case-by-case basis of the common law. Case-based reasoning involves more than deducing the consequences from a precisely stated set of facts and legal rules.[43] McCarty[33] recognised that the main task for a disputant in a "hard case" is to construct a theory of the disputed rules that produces the desired legal result, and then to persuade the judge that this theory is preferable to any theories offered by an opponent. More concretely, Berman and Hafner[10] pointed out that disputants build arguments for or against legal theories from factors of decided cases, and justify them by appealing to the values their acceptance would advance. Prakken[35] illustrated a method for expressing such values in the system of Prakken and Sartor.[40] Sartor[48] modelled reasoning with cases as dialectical theory construction via a set of operators directed by teleology. Jaap Hage[30] developed a reason-based logic to resolve disputes about the applicability of legal rules to a case. More recent research in case-based reasoning utilises abstract argumentation[14] as a common platform for a number of reasoning tasks. The first in this line of work is Prakken and Sartor's logic[38] instantiating abstract argumentation for reasoning with conflicting rules. In order to model how a judge decides on competitive theories taking into account the preference over their advanced social values, abstract argumentation is extended into value-based abstract argumentation.[5,7–9] The preference over the coherence of theories[9,31] have been also studied. In evidential reasoning,[6] Bex *et al.*[11] argues for the relevance of argumentation schemes, and the combination of argumentation schemes with the scenario-based approach.[12] Attention has been made to analyse how a judge considers a case factor as relevant[45] via argumentation. To model various types of burden of proof, Atkinson and Bench Capon[4] have used argumentation semantics while Gordon *et al.*,[27] Prakken and Sartor[42,41] have used assumptions. Systems capable of reasoning with cases include among others TAX-MAN,[33] HYPO,[3,46] CATO,[2] CABERET,[44] CATO's reconstruction in Carneades.[25] In these systems, to facilitate the generation of legal theories, a case is summarised by a set of factors, which are at an abstract, propositional level of granularity. The establishment of these factors seems to be assumed to have been done elsewhere. Further, none of these systems deal with contract dispute resolution.

Our work could be seen as an attempt to combine the features of Refs. 3–5, 7, 9, 24, 41 and 42 to offer a programming environment and tools for contract dispute resolution. Instead of using abstract argumentation, we use assumption-based argumentation to represent legal doctrines and cases at a concrete, inference-rule level of granularity, allowing the argumentation about factors and the argumentation about legal outcomes to interleave finely. This allows case analysis at multiple levels of detail. For example, one may check whether a new evidential item can change the acceptability of another evidential item, then change the acceptability of a factor, and finally reverse a legal outcome. However, forcing case formulation in the assumption-based argumentation's language with inference rules has a drawback that MoDiSo cannot be used yet when such inference rules are to be established (e.g. at legal proceedings). In principle, instead of assumption-based argumentation we can use any concrete instance[23,25,28,37,39,52] of abstraction argumentation, as long as the instance provides a modularity feature together with relevant argumentation semantics[14] such as the credulous semantics and the sceptical preferred semantics.

The paper is organised as follows. Section 2 presents a modular architecture for legal doctrines-based contract dispute resolution systems with an edit-compile-dispute loop facilitating incremental system developments. Section 3 elaborates on features and properties of MoDiSo along the line of a methodology to represent and reason with legal doctrines. We discuss future work and conclude in Section 4.

2. A Modular Architecture for Contract Dispute Resolution Systems

Theoretically courts could resolve any disputed issue that contract parties bring up. However a certain court, by its functions and roles in its judiciary system, often restricts what can be disputed. Generally in lower courts (e.g. trial courts) dispute parties cannot challenge legal doctrines and principles that are already established in widely accepted legal texts (e.g. Restatements or codifying Acts). Instead, they can present arguments establishing the factors of their case from the dispute context, cite well-established legal doctrines and principles to combine those factors to support their favoured legal outcomes. This insight suggests that a promising approach to build contract dispute resolution systems is to model well-established legal doctrines. A contract dispute resolution system for a specific contract area needs to reason with legal doctrines and principles specific to the area, which often extend, supplement but do not contrast with more general ones that govern a wider contract area. Governing the whole common law of contracts is the *golden* principle that the court resolves a contract dispute by enforcing a contract interpretation, called *complete intended contract*, that reasonably represents the mutual intention of contract parties at the time they signed the contract. Legal doctrines can be seen as instances of this golden principle, providing concrete ways to ascertain the complete intended contract. Because an actual contract is often viewed as representing the clearest mutual intention of contract parties, legal doctrines do not rewrite ex-

plicit contractual terms, instead supplement *implied terms* representing the *shared, but unexpressed intention* of contract parties. For example, legal doctrines for performance relief supplement *implied risk allocation terms*. Parties bearing the risk of an event by explicit or implied terms in the complete intended contract do not have the right to rescind when the risk materialises.[c] For illustration consider a hypothetical dispute the fishing company and the GIS company in Example 1.1:

- Fishing company (Plaintiff): On the ground of the doctrine of contract breach, your failure to render the service constitutes a contract breach which entitles me to claim for damages.
- GIS company (Defendant): It is impossible for me to render the service because Landsat 7 functioned improperly. On the ground of the doctrine of impossibility, I have the right to rescind the contract.
- Fishing company: Being an expert in Remote Sensing, you should know that Landsat 7 could function improperly any time. Hence you should have improved your mapping algorithms so that without proper Landsat 7 images you could still render your service satisfactorily. The failure to do so allocates the risk to you. Thus you do not have the right to rescind on the ground of the doctrine of impossibility.

  Here, if the GIS company does not counter-argue, the court will conclude that the GIS company bears the risk of the event that Landsat 7 functioned improperly. This is an implied risk allocation term, which is not written in the actual contract by the parties, but is added to it to obtain the complete intended contract, which the court relies on to decide that the fishing company wins the dispute.

However, unlike other attempts to case-based reasoning, we do not target at systems giving an automatic verdict, or creating the illusion of a judge tasked to decide which contract party wins the dispute presented before the court. Instead, we target at systems supporting users in thinking about and analysing their cases. These systems support, for example, consumers in legally structuring their complaints, business organizations in constructing arguments to address consumers's complaints, lawyers in finding legal arguments favouring their clients, or general disputants in preparing their arguments to be presented before the court. To capture the above insights into dispute resolution with legal doctrines, we propose in Fig. 1 a modular architecture for legal doctrine-based dispute resolution systems using argumentation techniques. In this architecture, a system is specified by a set of argumentation modules representing the doctrines and principles it covers. Such a system *runs* as long as its specification is loaded into a common runtime engine/environment,

[c]Readers are referred to our related papers[18,19] for the formalisation of the notions *complete intended contract* and *risk allocation*. Roughly: *complete intended contract* = *actual contract* + *implied terms*; a risk allocation term is a rule $\varepsilon \to P$ representing that if event ε occurs after the contract making, then the risk of the occurrence is allocated to party P.

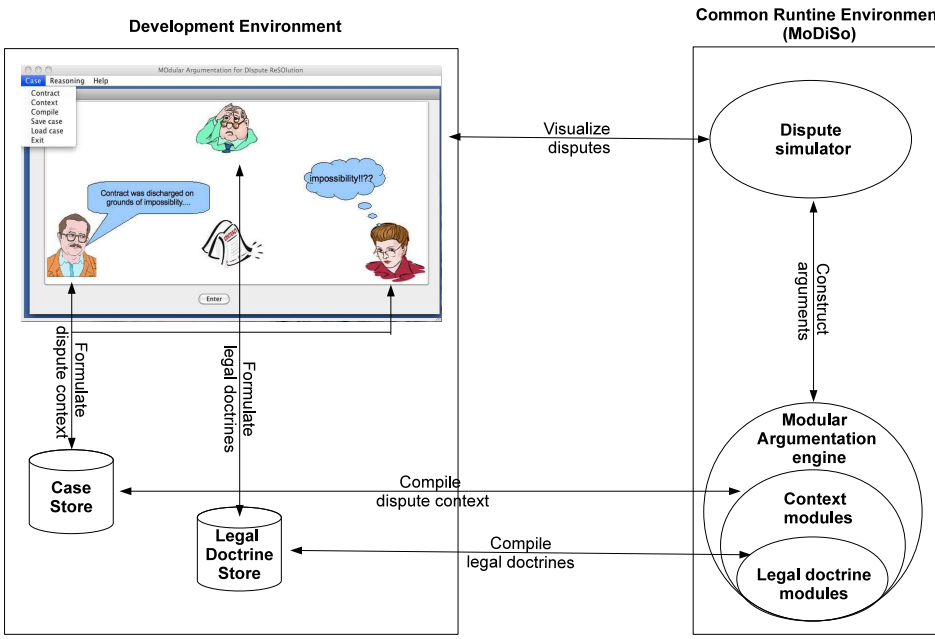

Fig. 1. A modular architecture for contract dispute resolution systems.

the core of the architecture. A running system is ready to try a case as long as argumentation modules representing the case's dispute context are loaded into the system. The common runtime engine in turn runs on a multi-semantic argumentation engine. The former invokes the latter to construct and evaluate arguments about the legal outcomes as well as about factors of the case upon the user's requests. Theoretically the main requirement for a common runtime engine in this architecture is that it has to provide necessary features and properties to represent and reason with legal doctrines and dispute contexts. The next section is devoted to demonstrating, via examples,[d] that MoDiSo satisfies the requirement. To facilitate system developments, the architecture includes a development environment consisting of a dispute simulator, a set of GUIs and persistent knowledge stores. The dispute simulator simulates the exchanges of arguments between two fictitious disputants about a given claim, then having the GUIs to render those arguments graphically, and also texturally in several pre-determined templates. The GUIs also provide interfaces to facilitate the formulation of dispute contexts. The development environment and the common runtime engine forms an edit-compile-dispute loop, allowing the user to build and test his system incrementally.

[d]The ease to craft an example in E-commerce, like Example 1.1, says that dispute resolution systems are necessary for its take-off. However, following the tradition of citing classical cases to signify the very nature of common laws, we will use classical cases in the remaining part of this paper.

3. MoDiSo: Features and Properties

In this section we elaborate on fundamental features and properties of MoDiSo and demonstrate how MoDiSo could be used as a common runtime environment for contract dispute resolution systems with the architecture in Fig. 1. The section also sheds light on a methodology to represent and reason with legal doctrines, principles, and rules of contract laws.

3.1. *Inference rules*

Let us start with a simple example of contract dispute resolution. Consider the doctrine of absolute contracts established in a famous court case Paradine v Jane, 1646[29] in which the Defendant Jane was unable to occupy land leased to him by the Plaintiff Paradine because the territory was under military occupation for a considerable time, however still had to pay the rent, as the court held that: "... when the party of his own contract creates a duty or charge upon himself, he is bound to make it good, if he may, notwithstanding any accident by inevitable necessity, because he might have provided against it by his contract".

For contracts for sale of goods or supply of services for a consideration represented by the contract price, the doctrine could be stated as follows.

- Premise 1: CO (contractor) and CE (contractee) are parties to contract Γ.
- Premise 2: Γ states that CO promised to perform a transaction τ (i.e. to deliver the goods or to render the services) and CE promised to pay a contract price π.
- Conclusion: CO is obliged to perform transaction τ and CE is obliged to pay price π.

Hence, using self-describing sentences, the doctrine could be captured by two inference rules:

- $perform(CO, \tau) \leftarrow contract(CO, CE, \Gamma)$, $transaction(\tau, \Gamma)$
- $pay(CE, \pi) \leftarrow contract(CO, CE, \Gamma)$, $transaction(\tau, \Gamma)$, $price(\pi, \Gamma)$, $perform(CO, \tau)$[e]

These rules are case-independent thus stand for the set of their instances. Specific facts of the contract between Paradine and Jane can be represented by the following additional inference rules, which together with the above two rules establish claims $perform(\boldsymbol{paradine, lease})$, $pay(\boldsymbol{jane, \pi_0})$.[f]

- $contract(\boldsymbol{paradine, jane, pjContract}) \leftarrow$
- $transaction(\boldsymbol{lease, pjContract}) \leftarrow$
- $price(\boldsymbol{\pi_0, pjContract}) \leftarrow$

[e]In these two rules and subsequent rules, we assume a fix, but arbitrary contract dispute, and hence can omit variable Γ in several predicates (e.g. $perform(CO, \tau)$, $pay(CE, \pi)$). An obvious restoration of Γ is needed in order to deal with several contracts at the same time.
[f]Constants are ***bold***-faced.

3.2. *Assumptions as a mechanism for distributing burden of proofs*

The above inference rules are strict because they always draw the claims (or conclusions) stated at their heads whenever the factors stated at their bodies hold. They succinctly represent the undisputed account of the facts of the case and the absolute nature of the doctrine of absolute contracts, however, fail to represent disputed factors, defeasible legal principles, and doctrines with exceptions. To solve this problem, assumptions can be inserted into the premises of those strict inference rules. With assumption premises, resulted inference rules only draw their conclusions from their non-assumption premises when there is no evidence to the contrary of all the inserted assumption premises. To see the appropriateness of inference rules with assumptions, consider the doctrine of contract breach, which can be read as "when a party of a contract creates a duty or charge upon himself, he is bound to make it good *unless there are exceptions for him to rescind the contract*". The following inference rules are obtained from the strict inference rules representing the doctrine of absolute contracts:

- $perform(CO, \tau) \leftarrow contract(CO, CE, \Gamma)$, $transaction(\tau, \Gamma)$, $\neg rescind(CO, \Gamma)$
- $pay(CE, \pi) \leftarrow contract(CO, CE, \Gamma)$, $transaction(\tau, \Gamma)$ $price(\pi, \Gamma)$, $perform(CO, \tau)$, $\neg rescind(CE, \Gamma)$

The inserted assumption $\neg rescind(CX, \Gamma)$ means that "the strict rules should not be applied", or "party $CX \in \{CO, CE\}$ does not have the right to rescind" if the contrary $rescind(CX, \Gamma)$ (representing the right of party CX to rescind) can *not* be shown. Note that the doctrine of contract breach does not purport to explain why the courts abandon the doctrine of absolute contract as it simply asserts that the courts do so. Thus the doctrine justifies the assumption $\neg rescind(CX, \Gamma)$ to model the placement of burden of proving $rescind(CX, \Gamma)$ on party CX who asks for rescission. However the doctrine does not give any rules on how party CX could prove $rescind(CX, \Gamma)$. Such rules are specified by other doctrines, one of the most important of which is the doctrine of impossibility introduced in a famous court case Taylor v Caldwell.

> (Taylor v Caldwell, 1863[20,29]) Taylor (Contractee) hired Caldwell's hall for the purpose of given four grand concerts. Taylor were to pay £100 in the evening of each concert. After signing the contract but before the first concert a fire destroyed the hall. Taylor claimed damages (£58) in respect of the expenses which Taylor had incurred in advertising and preparing for the concerts. The court dismissed the claim, for the reason that Caldwell has the right to rescind the contract, on grounds that the fire, an unexpected event for both parties, destroyed the hall the non-existence of which renders the transaction of contract (i.e. to give the hall) impossible.

Here the court views that a contract may or may not impose an absolute obligation on a contract party. Thus the court would not regard an obligation as absolute if

the parties themselves did not intend it to be absolute.[29] Because Caldwell's promise to give the hall is conditional on the existence of the hall (the subject-matter of the contract) when the time of performance comes, he is not obligated to perform when the hall is destroyed unless he is in fault. "Of course, if he is in fault because his deliberate act has done away with the subject matter of the contract, and perhaps, if he has been negligent, he cannot recover. But *prima facie* he escapes. To make him liable, his fault must be proved by the party which alleges that it destroys his excuse."[g] This is a general rule of English common law, which appears to be based on the principle that the burden of proof is, in general, on the party who alleges either that a contract has been broken, or that the other party has lost the benefit of an "exception"[29] (e.g. a defence for non-performance because of impossibility).

The common law generally proceeds by distilling from a particular case the legal principle on which it is decided, and that legal principle is then generally applied to similar dispute contexts. Demonstratively, the grounds on which the court granted Caldwell the right to rescind was later generalised to become the doctrine of impossibility. The Restatement Second of contracts states:

"Where, after a contract is made, **a party's performance is made impossible** without his fault by the occurrence of an event the non-occurrence of which is a basic assumption on which the contract was made, his duty to render that performance is discharged, unless the language or the circumstances indicate the contrary".

Like many other legal principles for dispute resolution, the doctrine of impossibility is of a dialectical nature as it provides arguments for both dispute parties. Namely a party arguing for rescission on grounds of impossibility needs to show:

(1) that an unexpected event occurred after the contract making, and
(2) that the non-occurrence of the event was a basic assumption on which the contract was made.

And a party arguing against rescission on grounds of risk allocation needs to show:

(3) that the event is the fault of the party asking for rescission, or
(4) that the party asking for rescission bears the risk of that occurrence of the event either under the language of the contract or the surrounding circumstances.

Certainly this interpretation of the doctrine considers that the purpose of the words "without his fault" is to limit the cases in which the doctrine should not be applied, but not to impose on the party seeking to be excused the necessity of proving he is not in fault. This interpretation is consistent with the above general rule of English common law that the burden of proof of fault is on the other party.

[g]Lord Porter in Joseph Constantine SS Ltd v Imperial Smelting Co Ltd (1942).[47]

However, this rule is not applicable in a special group of cases: a person to whom goods have been bailed, and who relies on their destruction as a ground of discharge, must show that the destruction was not due to any breach of his duty as a bailee.[29] Thus it is not surprising that in some countries the burden of proof on the issue of fault is on the party who relies on the unexpected event as a ground of discharge. This shows that interpreting legal doctrines is a complex task which is subject to case-based reasoning as well.

But at the end of a case when all evidence has been presented, is the legal outcome affected by where the burden of proof on the issue of fault lies? In Joseph Constantine SS Ltd v Imperial Smelting Co Ltd (1942)[47] where a ship was prevented by an explosion from performing the services which she was to have rendered under a charter-party, the evidence neither affirmatively shows that the shipowners caused the explosion, nor affirmatively shows that the shipowners did not cause it. The charterers' claim for damages for breach of the charter-party was first awarded by the Court of Appeal on the grounds that shipowners have not established affirmatively that the explosion was not due to its fault. However, it was later rejected by the House of Lords on the grounds that the burden of proving fault is on the charterers.

For the purpose of demonstration, let us stick with the first interpretation. The following inference rule connects the doctrine of impossibility with the doctrine of contract breach.

$$rescind(CX, \Gamma) \leftarrow impossibility(\Gamma), \neg riskAllocatedTo(CX, \varepsilon, \Gamma)$$

where $impossibility(\Gamma)$ covers the above conditions (1) and (2), and $\neg riskAllocatedTo(CX, \varepsilon, \Gamma)$ is an assumption covering conditions (3) and (4), with contrary $riskAllocatedTo(CX, \varepsilon, \Gamma)$ which states that party CX is allocated the risk of event ε.

Why $\neg riskAllocatedTo(CX, \varepsilon, \Gamma)$ should be modelled as an assumption? This comes from the distribution of burden of proof among contract parties: the burden of proving $impossibility(\Gamma)$[h] lies with the party CX who asks for rescission, and then the burden passes to the other party to prove $riskAllocatedTo(CX, \varepsilon, \Gamma)$.[i]

3.3. *Modularisation*

How $impossibility(\Gamma)$ and $riskAllocatedTo(CX, \varepsilon, \Gamma)$ could be formalised? In condition (1), the fire is unexpected for both parties means that at the time of contract making both of them did not believe that the fire could occur. If Caldwell believed that the fire could occur, he should have bargained for a provision against the risk (e.g. by charging a higher price for buying insurance) thus he does not have the

[h]I.e. it is impossible to carry out the contract due to the occurrence of the unexpected event.
[i]I.e. even if unexpected events made it impossible for party CX to carry out the contract, the risk of such events is allocated to CX and hence CX could not rescind.

right to rescind when the risk materializes. In other words, there could not be impossibility due to expected events as parties rationally do not contract to do what is impossible. Note that Caldwell is still not allowed to rescind if the fire is unexpected for him but expected for Taylor. This is because if Taylor could expect the fire, then it may not be impossible for Caldwell to expect the fire as well. So the unexpectedness of the fire for Caldwell constitutes a mistake of Caldwell and in general a mistake by one party does not allow that party to repudiate its signed contract since the opposite would encourage the ignorance of necessary knowledge of the contract domain.

In condition (2), the occurrence of fire destroyed the hall, an essential means without which the promised transaction (i.e. to give the hall) could not be performed. The existence of hall is a basic assumption on which the contract was made because the parties would not enter into the contract if the hall does not exist. Here, the destroyed hall must be specifically referred to by a contractual condition, or at least understood by both parties to be the property that would be used. Thus Caldwell would not have the right to rescind if he has several halls and Taylor contracts to rent "a" hall without saying any specific hall because it is still possible for Caldwell to give a hall that is not destroyed, for the concerts.

In other words, $impossibility(\Gamma)$ means that two parties must have made their bargain on an assumption that an essential means for contract continues to exist. It follows that the contract is put to an end if this basic assumption no longer holds as a result of changes of circumstances brought about by a supervening event. However, a supervening event that simply turns a contract into a bad bargain for a party does not justify rescission because the main purpose of contract as legal and commercial institution is precisely to allocate the risks of such events: once those risks have been so allocated by the parties, they should, as an application of the principle of freedom of contract, not be re-allocated in a different manner by the courts.[29]

But how the courts determine risk allocation made by contract parties? Consider again condition (1). The court infers that if a party believed an event is possible, that party actually intended to bear the risk of the event and the omission of an explicit term in the contract expressing this actual intention merely means that contract parties did not trouble to write it down. In condition (3), there is natural justice in saying that parties who caused the event is allocated the risk, because, as a general principle, one cannot take benefit from his own wrong. In condition (4), the court may infer from "*the circumstances*" implied risk allocation clauses. For the purpose of providing an efficient way to allocate the combined loss, many modern courts and law schools advocate that risk is allocated to a party who is able to *foresee* it but did not guard against it.[34] The position taken here is that these are clauses the parties would have agreed on had they bargained with the unexpected event brought to their attention, assuming that they are *rational* persons.

The above analysis suggests that the formalisation of $impossibility(\Gamma)$ and $riskAllocatedTo(CX, \varepsilon, \Gamma)$ requires extra representation features. To determine that

an event was unexpected for a party, the court cannot rely solely on the party's declaration. Instead, the court reconstructs what the party believed when signing contract in light of the surrounding circumstances, then determines whether the party possibly believed that the event is possible. In contrast, to demonstrate that a party could foresee an event, the court reconstructs the expertise or professional knowledge the party should have known at the time of contract making, by for example looking at his education, training, and experience that may not considered known or available to the general public. The court then determines whether this expertise or professional knowledge allows the party to foresee the event. Common knowledge bases need also to be constructed in order to establish objectively the timing of the event causing the dispute, and whether the event destroys some means for performing the contracted transaction, and whether the unavailability of such a means renders the performance of the transaction impossible.[j]

In other words, for resolving a contract dispute, distinct knowledge bases specifying the beliefs and professional knowledge of contract parties at the time of contract making (often referred to respectively as BX and KX for party $CX \in \{CO, CE\}$[k]), as well as temporal common knowledge and general common knowledge in the contract domain (often referred to respectively as CK_t and CK_d) need to be constructed. The set of knowledge bases forming the context of the case at hand is called its contract context.

In practice, the construction of a contract context is done during legal proceedings by exchanges of arguments between the parties and the judge. The demo system provides a window for entering in the resulted context. The window consists of two areas as illustrated in Fig. 2: the upper area is for declaring inference rules and the lower area is for declaring assumptions. A blank entry for a new inference rule/assumption is added when the user clicks at the end of each area. While the identifier of each entry is generated automatically, the user can optionally document the entry by a short English text. On the right of each entry there are boxes which can be clicked to allocate the inference rule/assumption into different argumentation modules.

For illustration, the contract context of Taylor v Caldwell case is entered as follows:

- Both parties believed (by common sense) at the time of making contract that:
  - — the hall exists when the time of concerts comes. Since parties made their contract on this footing, ***hallExist*** is an assumption in Taylor's belief base (i.e. ***BE***) as well as Caldwell's belief base (i.e. ***BO***).

[j]It can be said that the role of contract parties in bringing about impossibility is limited because the courts could construct common knowledge bases to represent the view of fair and reasonable people, who after all represent the anthropomorphic conception of justice in order to achieve just and reasonable outcomes.

[k]For example, BO and KO are respectively the belief base and the professional knowledge base of the contractor CO.

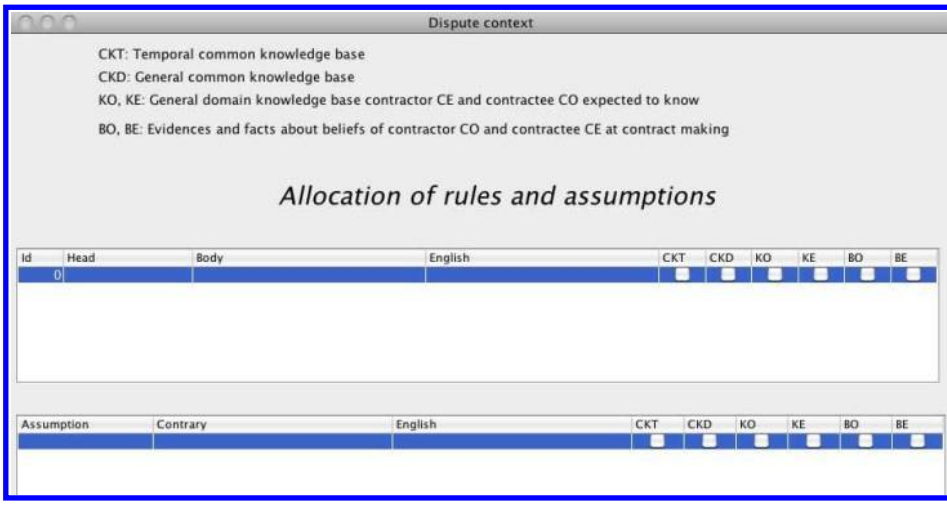

Fig. 2. Blank contract context window.

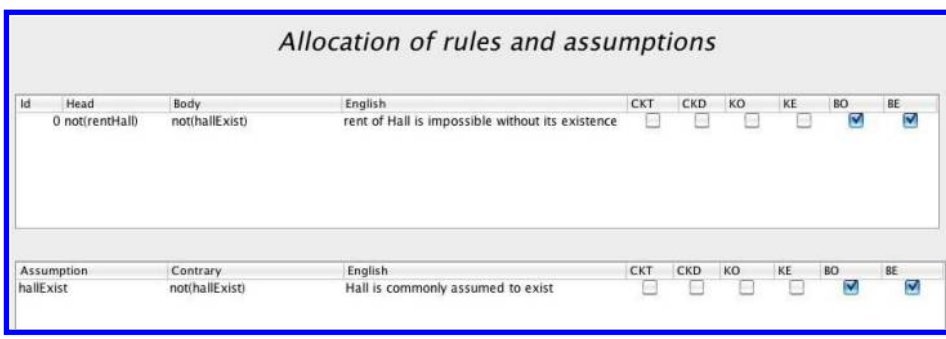

Fig. 3. Typing BO and BE.

— it is impossible to rent the hall without its existence. Hence inference rule

$$r_0 : \neg \boldsymbol{rentHall} \leftarrow \neg \boldsymbol{hallExist}$$

is also in both $\boldsymbol{BE}$ and $\boldsymbol{BO}$.

— The idea of fire did not come up at all for both parties while negotiating for the contract. Hence there are no rules or assumptions referring to fire in both $\boldsymbol{BE}$ and $\boldsymbol{BO}$.

Figure 3 shows the context contract window after entering $\boldsymbol{BE}$ and $\boldsymbol{BO}$. Note that we use $not(\_)$ to represent $\neg\_$.

- As a matter of fact, $\boldsymbol{fire}$ occurred after the signing of $\boldsymbol{contract}$. Hence the inference rule

$$r_1 : happen(\boldsymbol{fire}, \boldsymbol{contract}) \leftarrow$$

is in the temporal common knowledge (i.e. $\boldsymbol{CK_t}$), where predicate $happen(\epsilon, \Gamma)$ states that event ϵ happens after the signing of contract Γ.

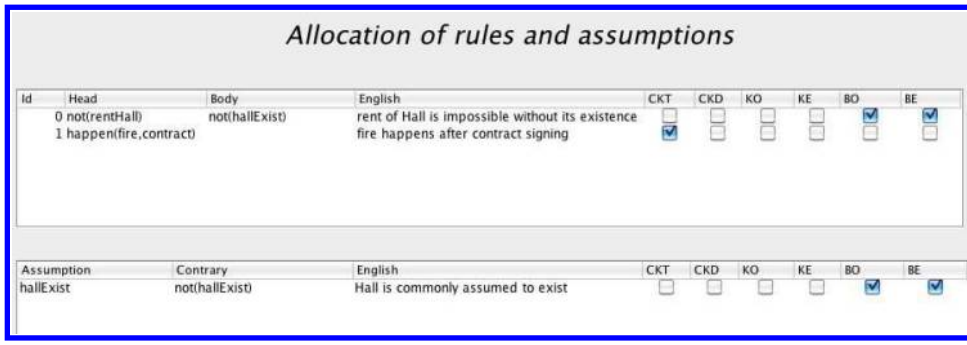

Fig. 4. Typing CK_t.

Figure 4 shows the contract context window after entering $\boldsymbol{CK_t}$.

- As a matter of fact, the fire destroyed the hall, the existence of which is a means for carrying out the contract. Hence

$$r_2 : \neg \boldsymbol{hallExist} \leftarrow \boldsymbol{fire}^{[l]}$$

$$r_3 : means(\boldsymbol{hallExist}, \boldsymbol{contract}) \leftarrow$$

is in the general common knowledge base (i.e. $\boldsymbol{CK_d}$), where predicate $means(m, \Gamma)$ states that m is means for carrying out (the transaction in) contract Γ.

Figure 5 shows the contract context window after entering $\boldsymbol{CK_d}$.

- Both parties did not have any expertise in mitigating the consequences of the fire, hence the professional knowledge base of both parties (i.e. $\boldsymbol{KO}$ and $\boldsymbol{KE}$) is the same as $\boldsymbol{CK_d}$.[m]

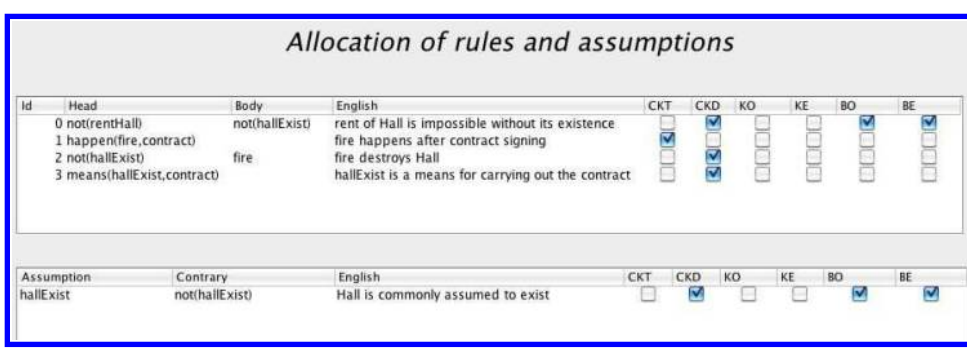

Fig. 5. Typing CK_d.

[l]Strictly speaking the term $\boldsymbol{fire}$ in r_1 reifies the predicate $\boldsymbol{fire}$ in r_2.

[m]If the case happens in our time, the professional knowledge base of Caldwell should contain an inference rule $\neg \boldsymbol{fire} \leftarrow \boldsymbol{springler}$ stating that the fire could be prevented by installing springer and alarm systems.

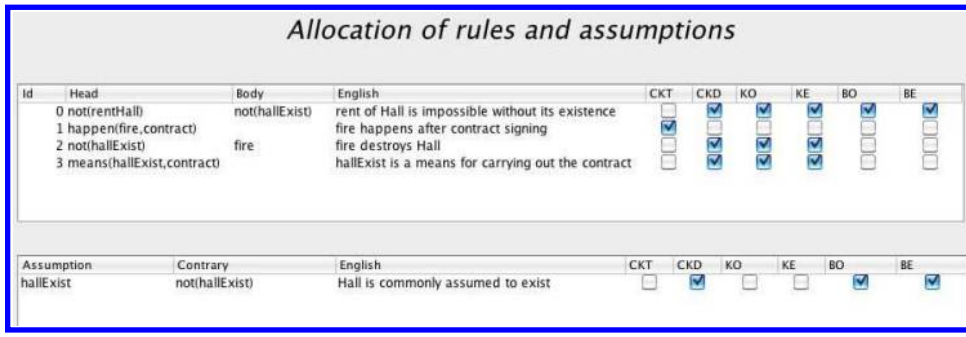

Fig. 6. Typing ***KO*** and ***KE***.

Figure 6 shows the contract context window after entering ***KO*** and ***KE***.

To summarise, recognising contract parties cannot exhaustively bargain for any eventuality, in resolving a contract dispute the court not only looks at terms actually agreed in their contract but also import other terms from the contract context, which consists of a number of distinct knowledge bases.

Definition 3.1. Under the doctrine of impossibility, a contract context Γ between contractor CO and contractee CE consists of knowledge bases BO, BE, CK_t, CK_d, KO, KE where

(1) BO, BE contain the evidences and facts about the relevant beliefs of contractor CO and contractee CE respectively at the time of making the contract.
(2) CK_t describes a body of temporal common knowledge established by the court whose purpose is to establish that the event ε causing the dispute happened after the contract making.
(3) CK_d describes a body of general common domain knowledge held by people with a rational mind in similar situations whose purpose is to establish objectively:
   (a) whether ε destroys or makes some means for performing the transaction τ of Γ unavailable,
   (b) whether the unavailability of such a means renders the performance of τ impossible.
(4) KO, KE describe respectively the professional knowledge or expertise that contractor CO and contractee CE are expected to know at the time of making the contract.

MoDiSo represents these knowledge bases by modular assumption-based argumentation (MABA) frameworks.[17,18] A MABA framework is structured into distinct assumption-based argumentation modules where exactly one of them is considered as the main module while others are called submodules. A module consists of a set of inference rules and a set of assumptions together with their contraries. A module refers to its submodules by means of module calls using different argumentation-

based semantics. The formal definition of a module is recalled from Ref. 18 as follows.

Definition 3.2. An assumption-based argumentation module $\mathcal{F}$ is a triple $(\mathcal{R}, \mathcal{A}, \overline{\ })$ where $\mathcal{A} \subseteq \mathcal{L}$ is a set of assumptions; $\overline{\ }$ is a (total) mapping from $\mathcal{A}$ into $\mathcal{L}$, where $\overline{x}$ is referred to as the *contrary* of x; and $\mathcal{R}$ is set of inference rules of the form $l_0 \leftarrow l_1, \ldots, l_n$ (for $n \geq 0$) where l_i is either a sentence of $\mathcal{L}$ or a module call of the form $call(l, M, t)$ where l is a non-assumption sentence in $\mathcal{L}$, M is a submodule of $\mathcal{F}$ in which l occurs, and t is a semantics of M according to which l is defined.

For example, the belief bases of parties in Taylor v Caldwell case are represented by modules $\boldsymbol{BO} = \boldsymbol{BE} = (R, \mathcal{A}, \overline{\ })$ with $\mathcal{R} = \{r_0\}$ and $\mathcal{A} = \{\boldsymbol{hallExist}\}$ where the contrary of $\boldsymbol{hallExist}$ is $\neg\boldsymbol{hallExist}$; while the temporal common knowledge base is represented by module $\boldsymbol{CK_t} = (\mathcal{R}_t, \mathcal{A}_t, \overline{\ })$ with $\mathcal{R}_t = \{r_1\}$ and $\mathcal{A}_t = \emptyset$; and the general common domain knowledge base is represented by module $\boldsymbol{CK_d} = (\mathcal{R}_d, \mathcal{A}, \overline{\ })$ with $\mathcal{R}_d = \{r_0, r_2, r_3\}$.

All modules in the context of Taylor v Caldwell do not have submodules as they do not contain any module calls. In the following we will develop a main module representing the doctrine of impossibility which refer to these modules via various module calls. For now, let us see how this main module could build an argument for impossibility that the fire, an unexpected event occurring after contract making, destroyed the hall, the non-existence of which renders the contracted transaction (i.e. to give the hall) impossible: since neither of $\boldsymbol{BO}$ and $\boldsymbol{BE}$ contains a rule concluding $\boldsymbol{fire}$, both parties do not have a reason to believe that $\boldsymbol{fire}$ can occur and hence $\boldsymbol{fire}$ must be unexpected for both parties; from $r_1 \in \boldsymbol{CK_t}$, the fire happened after the contract making; from $r_2 \in \boldsymbol{CK_d}$, the fire destroyed the hall; from $r_0 \in \boldsymbol{CK_d}$, the non-existence of hall makes it impossible to give the hall for performance. In general arguments are constructed by applying legal principles and doctrines (representable by inference rules as have been shown) to factors of the case (representable by module calls as will be shown), thus having tree structures as illustrated in Fig. 7 for the above argument.

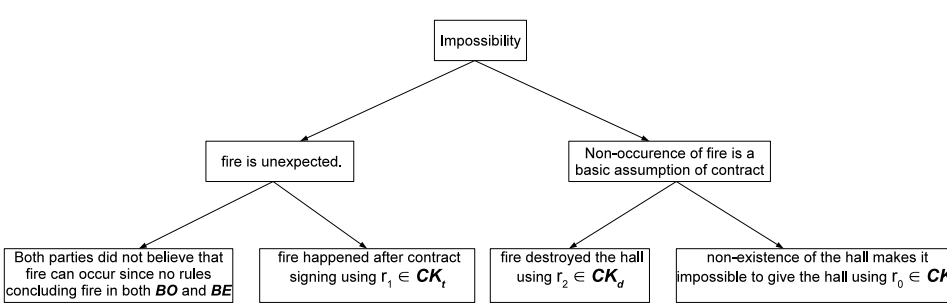

Fig. 7. Argument for impossibility.

3.4. *Multi argumentation-based semantics*

Why do we need module calls with different semantics? One reason is that for different factors to be proved, the court may require different standards of proof. For example, the court may be satisfied an event occurred if it considers that, from a knowledge base representing the temporal evidences of the case, the occurrence of the event was likely (i.e. some rational reasoner accepts the occurrence); or the occurrence of the event was surely (i.e. all rational reasoners accept the occurrence). Often the more serious the allegation, the less likely that it is that the event occurred and hence, the more convincing should be the standards of proof.

MoDiSo can compute consequences of a knowledge base/module according to several argumentation-based semantics but for the purpose of this paper we only present two of them: the *skeptical* semantics defines consequences that are accepted by *all* reasoners while the *credulous* semantics defines consequences that are accepted by only *some* reasoners. In other words a skeptical consequence represents a *consensus* among *all* reasoners while a credulous consequence represent an opinion of a reasoner, which may and may not be shared by other reasoners with the same knowledge base.

Multi-semantics and modularisation can be combined to provide different standards of proof (and as will be shown in the next section, to reason about the risk attitudes of contract parties). To establish that event ε occurred after the signing of contract Γ (denoted by $happen(\varepsilon, \Gamma)$) by a *proof* of such a convincing character that *all* jurors (i.e. reasoners) after careful and impartial consideration of all the temporal evidences in the case, agree so, the main module representing the doctrine of impossibility refers to module CK_t by $call(happen(\varepsilon, \Gamma), CK_t, sk)$ where sk stands for *skeptical*. Here $call(happen(\varepsilon, \Gamma), CK_t, sk)$ is accepted if and only if $happen(\varepsilon, \Gamma)$ is a skeptical consequence of module CK_t. In contrast, to establish that party CX believed that event ε is *possible*, we need a proof for ε from the belief base BX of party CX, of such a convincing character that a rational person given the same belief base BX would *possibly* believed in ε. For example, $call(\varepsilon, BX, cr)$ where cr stands for *credulous* is accepted if and only if ε is a credulous consequence of module BX.

Hence, for determining that event ε occurred after the contract making and is unexpected for both parties (represented by sentence $unExpected(\varepsilon)$), the main module contains the following inference rules, where $\neg expected(\varepsilon)$ is an assumption that could be rejected by proving $expected(\varepsilon)$.

$$unExpected(\varepsilon) \leftarrow happen(\varepsilon, \Gamma), \neg expected(\varepsilon)$$
$$happen(\varepsilon, \Gamma) \leftarrow call(happen(\varepsilon, \Gamma), CK_t, sk)$$
$$expected(\varepsilon) \leftarrow call(\varepsilon, BE, cr)$$
$$expected(\varepsilon) \leftarrow call(\varepsilon, BO, cr)$$

A proof represents an argument. An argument for conclusion (claim) α from a set of assumptions X has a tree structure with α labelling the root and assumptions in

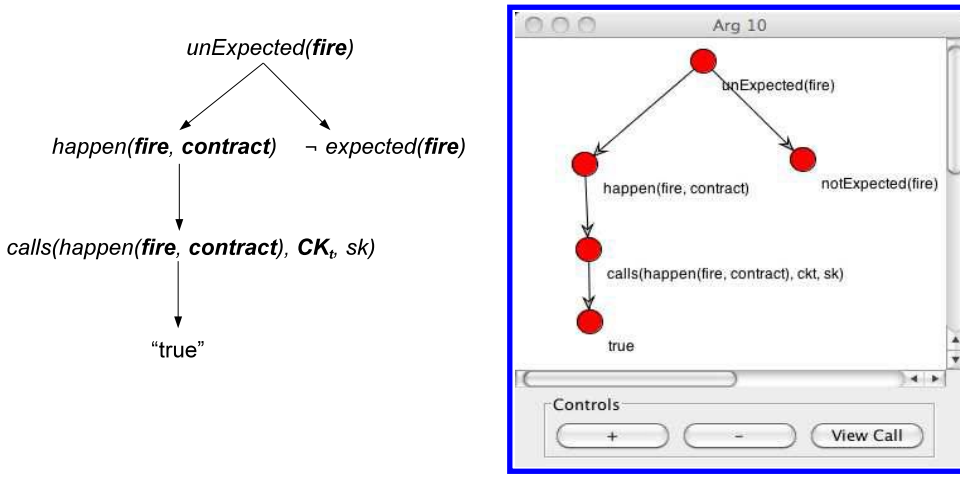

Fig. 8. Argument tree.

X labelling the leaves. Nodes of this tree are connected by the inference rules, with sentences matching the conclusion of an inference rule connected parent nodes to sentences matching the premises of the inference rule as children nodes. The leaves are assumptions or a special sentence "true" representing an accepted module call or an empty set of premises.

Figure 8 illustrates an argument formed from the four above inferences rules applied to the context of Taylor v Caldwell case.

Definition 3.3. In a MABA $\mathcal{F} = (\mathcal{R}, \mathcal{A}, \overline{\ })$, an *argument* for $\alpha \in \mathcal{L}$ (*conclusion* or *claim*) *supported* by $X \subseteq \mathcal{A}$, denoted by (α, X), is a tree with nodes labelled by sentences in $\mathcal{L} \cup \{\text{"}true\text{"}\}$, such that

(1) the root is labelled by α
(2) for every internal node N
   (a) if N is a module $call(l, M, t)$, then l is a consequence of M wrt semantics t and the child of N is labelled by "$true$".
   (b) if N is not a module call and l_N is the label of N, then there is an inference rule $l_N \leftarrow b_1, \ldots, b_m$ in $\mathcal{R}$ and
      - if $m = 0$, then the child of N is labelled by "$true$"
      - if $m > 0$, then N has m children which are labelled by $b_1, \ldots, b_m$ respectively.
(3) X is the set of assumptions labelling the leaves.

An argument (x, X) attacks an argument (y, Y) if x is the contrary of some assumption in Y. For example, if $\boldsymbol{fire}$ is added to $\boldsymbol{BO}$ as an assumption, then argument $(expected(\boldsymbol{fire}), \{\})$ in Fig. 9 attacks argument $(unExpected(\boldsymbol{fire}), \{\neg expected(\boldsymbol{fire})\})$ in Fig. 8.

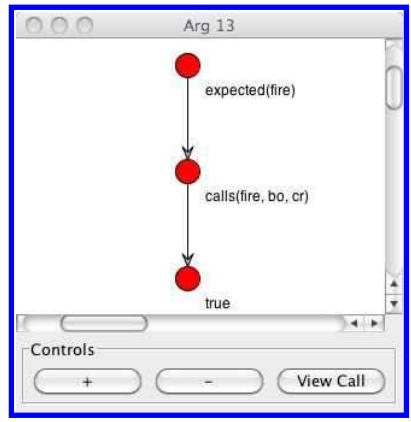

Fig. 9. Fire is expected for Caldwell if ***fire*** is an assumption in ***BO***.

A set of assumptions S attacks an assumption α if there is an argument (y, Y) with $Y \subseteq S$ and $y = \overline{\alpha}$. S attacks another set of assumptions S' if S attacks an assumption in S'; S is conflict-free if it does not attack itself. S is *admissible* if S is conflict-free and S attacks each set of assumptions that attacks S. A maximal (wrt set inclusion) admissible set of assumptions is called a *preferred extension.*

A proposition $\pi \in \mathcal{L}$ is said to be a *credulous consequence* of $\mathcal{F}$, denoted by $\mathcal{F} \vdash_{cr} \pi$ if π is supported by (a subset of) *some* preferred extension of $\mathcal{F}$; π is said to be a *skeptical consequence* of $\mathcal{F}$, denoted by $\mathcal{F} \vdash_{sk} \pi$ if π is supported by (a subset of) *each* preferred extension of $\mathcal{F}$.

Viewing different preferred extensions as representing different reasoners, skeptical consequences are accepted by all reasoners while credulous consequences are accepted by only some reasoners

It could be helpful for dispute parties to view failed proofs representing failed attempts to construct arguments. Figure 10 shows a failed proof in Taylor v Caldwell case, where the node labelled by "$fail$" indicates that $call(\boldsymbol{fire}, \boldsymbol{BO}, cr)$ is not accepted, suggesting that new evidences for ***fire*** in ***BO*** could overturn the current legal outcome. Failed proofs are defined by adding the following exceptions to Definition 3.3: in condition 2, if $M \not\vdash_t l$ or if there is no inference rule with head l_N, then the child of N is labelled by "$fail$"; in condition 3, X is a set of sentences labelling the leaves. A failed proof for claim α supported by premises X that fails to turn into an argument due to failed premises F (the set of sentences labelling the parents of leaves "$fail$") is denoted by $(\alpha, X \cup F)$. Note that a failed proof $(\alpha, X \cup F)$ attacks an argument (β, Y) if α is a contrary of some assumption in Y; $(\alpha, X \cup F)$ is attacked by special arguments $fail(f)$ for $f \in F$ where $fail(f)$ is not attacked by any argument.

3.5. *Programming the doctrine of impossibility*

Now we present a main module representing the doctrine of impossibility.

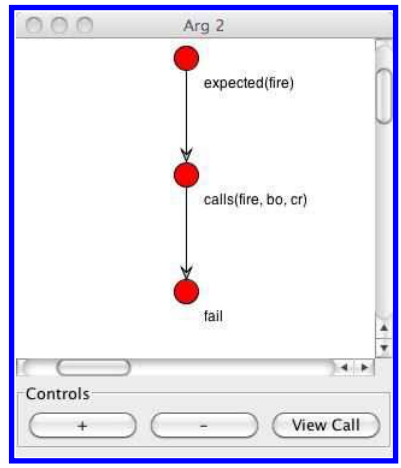

Fig. 10. A failed proof.

- Impossibility means that the occurrence of an unexpected event ε after the contract making event E violates a basic assumption on which the contract was made.[n]

$$impossibility(\Gamma) \leftarrow unExpected(\varepsilon), violateBA(\Gamma)$$

- The occurrence of event ε violates a basic assumption on which the contract was made means that ε destroyed an essential means m (i.e. $CK_d \cup \{\varepsilon\} \vdash_{sk} \neg m$),[o] without which it is impossible to perform the transaction τ of contract Γ according to the contract specification T (i.e. $CK_d \cup T \cup \{\neg m\} \vdash_{sk} \neg\tau$).

$$violateBA(\Gamma) \leftarrow call(\neg m, CK_d \cup \{\varepsilon\}, sk), call(\neg\tau, CK_d \cup T \cup \{\neg m\}, sk)$$

Here, $call(\neg m, CK_d \cup \{\varepsilon\}, sk)$ is accepted only if there is an argument for the unavailability of means m from module $CK_d \cup \{\varepsilon\}$, of such a convincing character that *all* jurors (as general fact triers) after careful and impartial consideration of this module, accepts the unavailability. A similar reading can be given to the second module call. Note that the inclusion of module CK_d in the expression $CK_d \cup T \cup \{\neg m\}$ is just an application of a general principle that a contract must be read in light of the surrounding circumstances.

For example, Caldwell's argument for impossibility assigns $\varepsilon = \boldsymbol{fire}$, $m = \boldsymbol{hallExist}$ and $\tau = \boldsymbol{rentHall}$.

The following rules generate arguments for risk allocation.

[n]Assumptions in this module are represented by negative literals whose contraries are corresponding possible literals.
[o]Note that by $\mathcal{F} \cup X$, we mean the module obtained from $\mathcal{F}$ by adding $\{x \leftarrow | \ x \in X\}$ to its set of inference rules.

- The risk can be allocated by contract. If $\epsilon \rightarrow CX$ is a risk allocation term in contract Γ (stating that if event ϵ occurs after the contract making then the risk is allocated to party CX), then there is a rule to that effect:

$$riskAllocatedTo(CX, \epsilon, \Gamma) \leftarrow call(happen(\epsilon, \Gamma), CK_t, sk)$$

- The risk is allocated to parties who caused the unexpected situations.

$$riskAllocatedTo(CX, \varepsilon, \Gamma) \leftarrow call(cause(CX, \varepsilon),\ CK_t \cup CK_d, sk)^{\text{p}}$$

- The risk is allocated to efficient risk bearers of the event, i.e. parties who could foresee the event (i.e. $KX \vdash_{cr} \varepsilon$) and do some reasonable action α to prevent it (i.e. $KX \cup \{\alpha\} \vdash_{cr} \neg\varepsilon$) or mitigates its consequence (i.e. $KX \cup \{\varepsilon, \alpha\} \vdash_{cr} \tau$). Note that if the event was not foreseeable by party CX, then CX can hardly be expected to perform any prevention or mitigation actions. However, foreseeability alone does not necessarily prove the allocation of risk of the event to CX since the event may be beyond the capability of CX wrt the value of the contract.

$$riskAllocatedTo(CX, \varepsilon, \Gamma) \leftarrow call(\varepsilon, KX, cr),\ prevent(CX, \varepsilon)$$
$$riskAllocatedTo(CX, \varepsilon, \Gamma) \leftarrow call(\varepsilon, KX, cr),\ mitigate(CX, \varepsilon)$$
$$prevent(CX, \varepsilon) \leftarrow reasonableAction(CX, \alpha),\ call(\neg\varepsilon, KX \cup \{\alpha\}, cr)$$
$$mitigate(CX, \varepsilon) \leftarrow reasonableAction(CX, \alpha),\ call(\tau, KX \cup \{\varepsilon, \alpha\}, cr)$$

  In the last two inference rules, $call(\neg\varepsilon, KX \cup \{\alpha\}, sk)$ or $call(\tau, KX \cup \{\varepsilon, \alpha\}, sk)$ would be rather strong conditions as, for example, practically one may take precaution to prevent fire or mitigate its consequences but fire could still happen and burn down one's hall.
- An action α of party CX is said to be reasonable if its cost is reasonable wrt the price π of contract Γ (i.e. $muchLess(cost(\alpha), \pi)$ where $muchLess(p, q)$ represents a partial order between integers that p is smaller than q by orders of magnitude).

$$reasonableAction(CX, \alpha) \leftarrow action(CX, \alpha), price(\pi, \Gamma), muchLess(cost(\alpha), \pi)$$

Thus if Taylor v Caldwell happened in our time, then Taylor may successfully argue that the risk of fire should be allocated to Caldwell because fire not only is reasonably foreseeable (i.e. $\boldsymbol{KE} \vdash_{cr} \boldsymbol{fire}$) but also can be prevented by installing sprinkler and alarm system (i.e. $\boldsymbol{KE} \cup \{\boldsymbol{sprinkler}\} \vdash_{cr} \neg\boldsymbol{fire}$) at a reasonable cost.

3.6. *Legal doctrines store*

As presented in Section 2, before going to courts contract parties should have determined legal doctrines on which their arguments are based. Thus a collection of legal doctrines that can be loaded dynamically into a contract dispute resolution system

[p] $\neg cause(CO, \varepsilon)$, $\neg cause(CE, \varepsilon)$ are assumptions since common sense assumes a person does not do bad things unless proven contrary.

Fig. 11. Loading a legal doctrine.

is helpful for them. Our demo system includes several legal doctrines for performance relief including the above doctrine of impossibility. To load these doctrines, the user just needs to click on the Reasoning menu, as shown in Fig. 11 (top). Then a window for receiving user queries appears, as shown in Fig. 11 (bottom). The user could type his own claim in the blank entry at the bottom of this window, or just selects a prepared one. Issuing a claim results in another window showing whether the claim is accepted and a simulated dispute between contract parties, as shown in Fig. 12 and explained shortly below.

3.7. *Simulated disputes*

To explain why a claim is accepted, the Dispute simulator (recall Fig. 2) simulates the exchanges of arguments between contract parties by building and exploring a dialectical structure of the plaintiff's argument for the claim, and defendant's counter-arguments attacking the argument, the plaintiff's arguments attacking all the defendant's arguments, and so on. This is done by the following rules and predicates.[q]

(1) A predicate $defArg(\alpha, Arg)$ generates a (complete or incomplete) argument Arg for a claim α for the defendant.
(2) A predicate $plaArg(\alpha, S, Arg)$ generates a (plaintiff's) complete argument Arg for a claim α supported by a set S of assumptions.

[q]Note that module reference is omitted for simplicity.

(3) The following rule generates an initial argument Arg for a given claim α together with an admissible set S of assumptions supporting Arg.

$$initialArg(\alpha, S, Arg) \leftarrow call(\alpha, cr, S),\ plaArg(\alpha, S, Arg)\,.$$

Note that $call(\alpha, cr, S)$ determines an admissible set S of assumptions supporting α.

(4) The following rules generate an argument B for the defendant to attack an given argument A of the plaintiff.

$$attack(A, B) \leftarrow assumption(A, \alpha),\ defArg(\overline{\alpha}, B).$$

$$assumption(A, \alpha) \leftarrow A = (\_, X), \alpha \in X.$$

(5) The following rules generate an argument B supported by a given admissible set S of assumptions for the plaintiff to counter-attack a given argument A of the defendant.

$$counterAtt(A, S, B) \leftarrow A = (\alpha, X \cup F),\ f \in X,\ B = fail(f).$$

$$counterAtt(A, S, B) \leftarrow assumption(A, \beta),\ plaArg(\overline{\beta}, S, B).$$

A dispute can be simulated as follows. The plaintiff starts by putting forwards an initial argument Arg for his claim α by calling $initialArg(\alpha, S, Arg)$, and then the plaintiff and the defendant alternate in attacking each other's previous arguments. By calling $attack(A, B)$, the defendant generates an argument B to attack an argument A previously presented by the plaintiff. By calling $counterAtt(A, S, B)$, the plaintiff generates an argument B to attack an argument A previously presented by the defendant.

A simulated dispute can be rendered sequentially as in Fig. 12 (left). This form offers some space for rendering also claims and premises of arguments in English, but the dispute parties may see it inconvenient to scroll down the dispute. A dispute tree, illustrated in Fig. 12 (right), addresses this inconvenience. The root of such a tree stands for the initial argument of the dispute while each other node stands for an argument presented by a party during the dispute to attack an argument at its parent node previously presented by the other party. For example in the dispute tree in Fig. 12 (right), argument Arg_0 is attacked by several arguments including Arg_1 which in turn is attacked by argument Arg_6.

Combining dispute trees and argument trees offers an interesting way to navigate on simulated disputes. By selecting an argument on a dispute tree and then clicking button "Deduction" (see Fig. 12), one could view its argument tree. By selecting a module call on an argument tree and then clicking "View call" button (see Fig. 10), one could view a dispute tree for this module call.

3.8. *MoDiSo for constructing arguments*

The doctrine of impossibility plays an important role in regulating commerce. Without it there would be two possible cases: either that any party could withdraw its

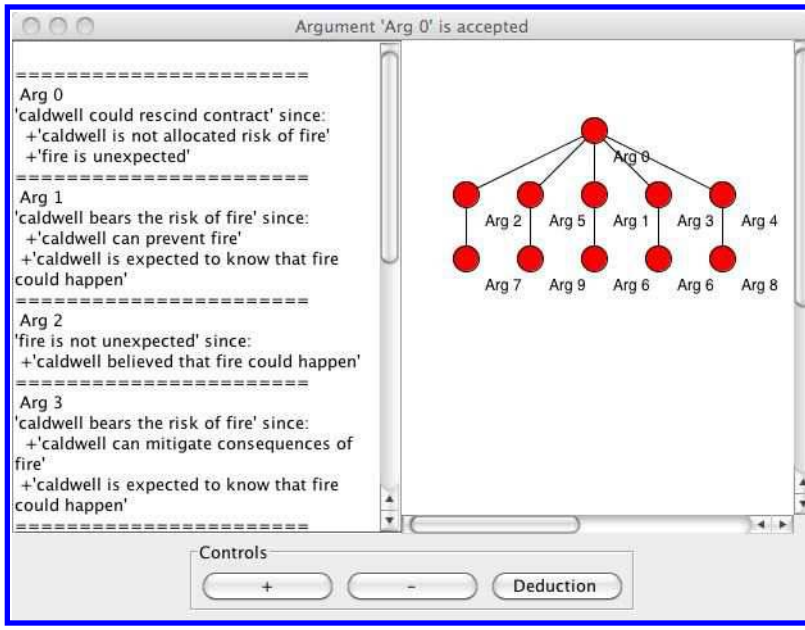

Fig. 12. A simulated dispute.

signed contract arbitrarily, e.g. merely because a supervening change of circumstances had turned the contract, for that party, into a bad bargain; or no parties could withdraw in any circumstances as stipulated by the doctrine of absolute contracts. The first case would clearly make commerce impossible. The second case is not longer considered as good law as it could lead situations opposite to the view that commerce is to increase the wealth of the society, and could lead to riots and social instability in case of large-scale disasters. The doctrine of impossibility, on one hand, by defining an extremely narrow gap for a non-performance of a contractual promise not to be deemed as a contract breach, the doctrine makes it almost impossible or practically impossible for contract parties to break their contractual promises. On the other hand, by discharging contracts in truly impossible situations, the doctrine avoids causing extra distress to contract parties in a situation beyond its capacity to handle, hence supports the above view that commerce is to increase the wealth of a society. For its important role, the doctrine should be included in any dispute resolution system.

An interesting use-case of MoDiSo is to help a dispute party construct arguments. Suppose a modern-day plaintiff MTaylor in a situation similar to Taylor v Caldwell wants to develop a context where the argument in Fig. 10 becomes complete. Further suppose MTaylor knows that other fires occurred before and fortunately the hall was still intact. So MTaylor modifies $\boldsymbol{KO}$ so that $\boldsymbol{fire}, \neg \boldsymbol{fire}$ become assumptions. Because now $call(\boldsymbol{fire}, \boldsymbol{KO}, cr)$ is accepted, the system renders a new argument tree (Fig. 13).

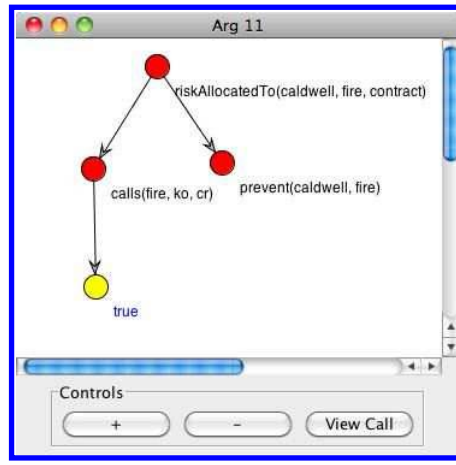

Fig. 13. $calls(\boldsymbol{fire}, KO, cr)$ is now accepted.

However, the argument is still incomplete and as it is attacked by $fail(prevent(\boldsymbol{caldwell}, \boldsymbol{fire}))$. Here the system suggests that MTaylor gives new evidences to show that Caldwell could prevent the fire. Let MTaylor do so with the following rules inserted into $\boldsymbol{CK_d}$, saying Caldwell could install a sprinkler and alarm system at only £1 to prevent the fire.

- $\neg \boldsymbol{fire} \leftarrow \boldsymbol{sprinkler}$
- $action(\boldsymbol{caldwell}, \boldsymbol{sprinkler}) \leftarrow$
- $cost(\boldsymbol{sprinker}, 1) \leftarrow$
- $muchLess(1, 100) \leftarrow$

Now the system finds for MTaylor and accordingly renders a new dispute tree given in Fig. 14. If one selects the argument "Caldwell bears the risk of fire" and

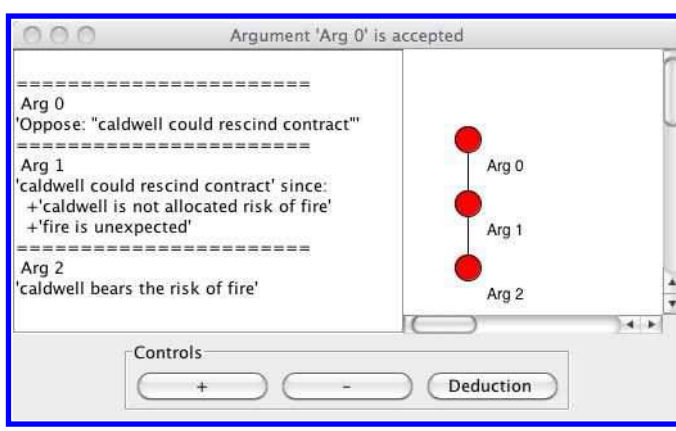

Fig. 14. New dispute tree.

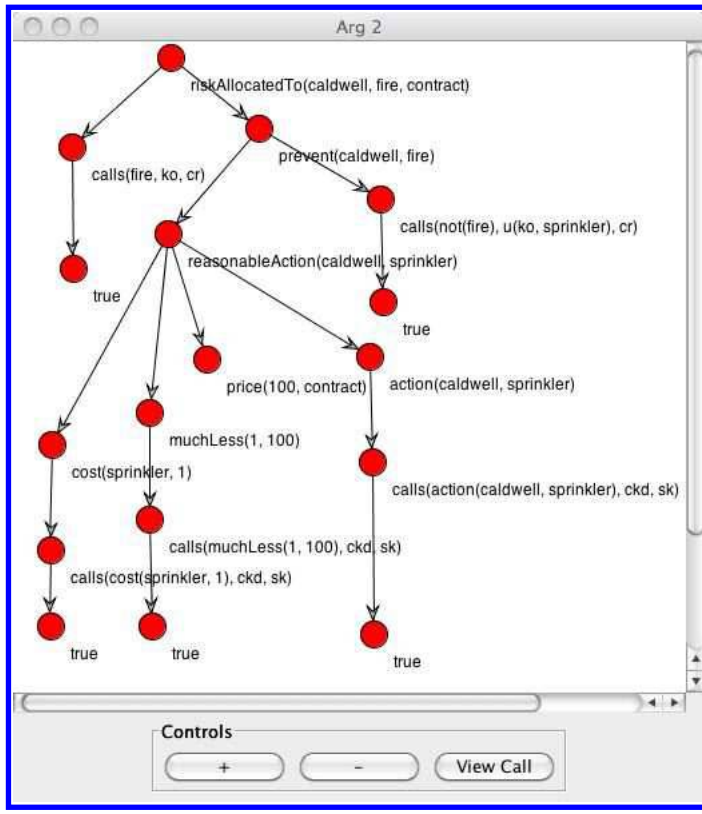

Fig. 15. Caldwell bears the risk of fire.

then clicks "Deduction" button, the system will display the tree of this argument as in Fig. 15. Note that we use $u(\mathcal{F}, X)$ to represent $\mathcal{F} \cup X$, the module obtained from $\mathcal{F}$ by adding $\{x \leftarrow | \ x \in X\}$.

3.9. *Other legal doctrines for performance relief*

Common law of contracts adopt other legal doctrines for performance relief. If a basic assumption of a contract concerns a continued state of things (e.g. the continued existence of a means for performance) when the time of performance comes, then naturally that assumption can be violated by supervening events that bring about changes of the state. In general a basic assumption of a contract could be violated by events other than supervening ones. Under the doctrine of mutual mistake, a basic assumption of a contract concerns a fact existing at the time of contract making, thus could be violated by a mistaken belief made by both parties about that fact. Party arguing for its right to rescind on the grounds of this doctrine needs to show that the mistake nullifies what parties intended to agree on and has a major impact on the fairness of the contract. Party arguing against rescission needs

to show that the risk of this type of mistake is allocated to the other party. For illustration of the doctrine, we recall famous court cases below.[20]

(Sherwood v Walker, Michigan, 1887) Walker, a cattle breeder, agreed to sell Sherwood, a banker, a cow (Rose 2nd of Aberlone) which both parties believe to be barren. The price was 80 USD. Prior to the delivery, Walker discovered that Rose 2nd is pregnant and refused to deliver her. The market price of a pregnant cow was around 800 USD. Sherwood sued, prevailed in trial court but lost in appeal. The appeal court based its decision on mutual mistake.

(Wood v Boyton, Wisconsin, 1885) Clarissa Wood found a colourful stone. She was told it could possibly be a topaz. She asked Boyton, a jewellery dealer. Boyton was not sure either and offered to buy it for one dollar. Wood declined. But later she needed money and returned to sell it to Boyton for one dollar. Later it turned out to be a rough diamond worth around 700 dollars. Wood brought a court action for the return of the stone citing mutual mistake. The court agreed that there was a mutual mistake but still ruled in favor of Boyton though not quite clear reasons had been given.

Analyzing Wood v Boyton case under the doctrine of mutual mistake, modern courts and scholars agree with the ruling for the reason of *conscious ignorance* meaning that Wood had known that there was a risk that the stone was not a topaz, i.e. it could be more or less valuable, but she still decided to sell it. Hence she should be allocated the risk of her decision. This principle of *conscious ignorance* can be captured (roughly) by the following rule:

$$riskAllocatedTo(\boldsymbol{wood}, \boldsymbol{diamond}, \boldsymbol{\Gamma}) \leftarrow call(\boldsymbol{topaz}, \boldsymbol{BO}, cr), \\ \neg\ call(\boldsymbol{topaz}, \boldsymbol{BO}, sk)$$

where $\boldsymbol{BO} = (\mathcal{R}, \mathcal{A}, \overline{\ })$ with $\mathcal{A} = \{\boldsymbol{topaz}, \neg\boldsymbol{topaz}\} : \overline{\boldsymbol{topaz}} = \neg\boldsymbol{topaz}, \overline{\neg\boldsymbol{topaz}} = \boldsymbol{topaz}$ and $\mathcal{R} = \{price(1) \leftarrow \boldsymbol{topaz}\}$ is Wood's belief base at the contract making which represents that Wood was not sure whether the stone is a topaz or not, but agreed to trade it for the price of one dollar. Note that since Wood was aware that the stone could possibly be a topaz but may be not, it is not possible that $\mathcal{A} = \{\}$. The idea that the stone could be a diamond does not come up at all at the time of making the deal. Hence no contract party could assume that it could be a diamond. Therefore it is not possible that $\mathcal{A} = \{\boldsymbol{topaz}, \neg\boldsymbol{topaz}, \boldsymbol{diamond}, \neg\boldsymbol{diamond}\}$. Readers are referred to Ref. 18 for a detailed formalisation of the doctrine.

3.10. *MoDiSo argumentation engine*

MoDiSo needs a modular assumption-based argumentation engine that can compute the credulous semantics as well as the skeptical semantics. CaSAPI,[22] the first assumption-based argumentation engine, can compute the credulous semantics and

the ideal semantics, an approximation of the skeptical semantics.[r] Moreover, CaSAPI does not support modularity hence it cannot directly be used for MoDiSo. For this reason we developed a genuine MoDiSo argumentation engine. As in CaSAPI, to compute the credulous semantics we implemented the dialectical notion in Refs. 15 and 16, called dispute derivations. To compute skeptical semantics, we adapted base derivations, the dialectical notion introduced in Ref. 51 for abstract argumentation, by handling the construction of arguments and attacks as in the dispute derivations.[15,16] The engine (in version 0.5 at the time of writing) is restricted to stratified MABA frameworks where the modules names are ranked (by ordinals) such that all module calls in rules belonging to a module of rank k refer to modules of ranks lower than k, like the MABA in Example 3.1 below.

Example 3.1.

- Submodule M_0 consists of:
  - — Assumptions: $\neg p$ and $\neg q$ with $\overline{\neg p} = p$ and $\overline{\neg q} = q$.
  - — Inference rules: $p \leftarrow \neg q$ and $q \leftarrow \neg p$.

  M_0 has two preferred extensions: $\{\neg q\}$ supports p (but not q) and $\{\neg p\}$ supports q (but not p).
  Hence both p and q are credulous consequences of M_0, however neither are its skeptical consequences.
  Hence $call(q, M_0, cr)$ and $call(p, M_0, cr)$ are accepted while $call(q, M_0, sk)$ and $call(p, M_0, sk)$ are not.
- Main module M_1 consists of:
  - — Assumption $\neg r$ with $\overline{\neg r} = r$.
  - — Inference rules:
    - ∗ Case 1: $h \leftarrow call(p, M_0, cr), call(q, M_0, cr), \neg r$
    - ∗ Case 2: $h \leftarrow call(p, M_0, cr), call(q, M_0, sk), \neg r$.

  M_1 has an unique preferred extension $\{\neg r\}$ which supports h in the first case but not in the second case.
  Hence h is a skeptical consequence of M_1 in the first case but not in the second case.

A MoDiSo program defines an MABA by means of several self-explaining predicates. The following program defines the above MABA (Note *calls* is used to represent module calls because *call* is already a Prolog meta predicate).

```
%% iCon(A,C) defines an assumption A with contrary C
iCon(not(q),q).
iCon(not(p),p).
iCon(not(r),r).
```

[r]If a claim is accepted under the ideal semantics, then it is also accepted under the skeptical semantics, but not vice versa

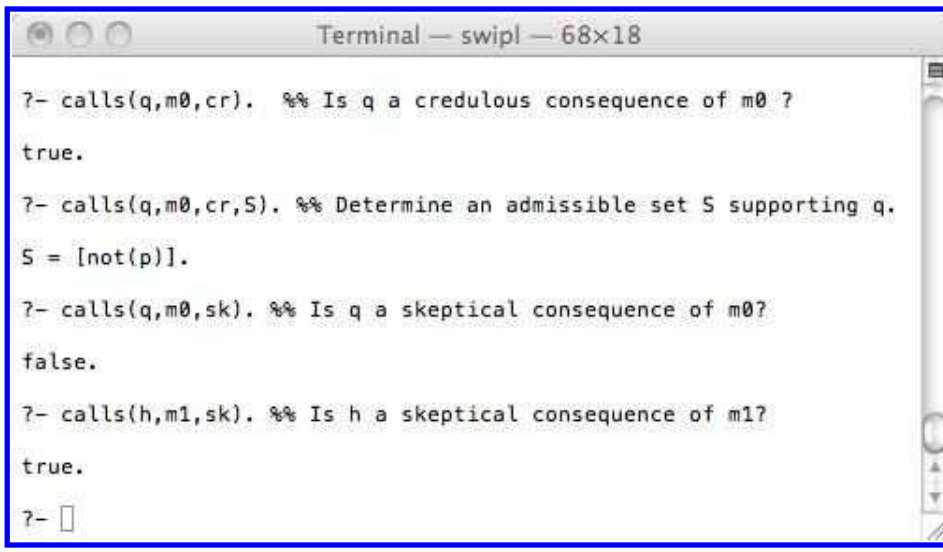

Fig. 16. MoDiSo engine front end.

```
%% iAss(M,As) allocates assumptions in list As into module M.
iAss(m0,[not(p),not(q)]).
iAss(m1,[not(r)]).

%% iRule(Id,H,B) defines a rule with head H, body B, and identifer Id.
iRule(r1,p,[not(q)]).
iRule(r2,q,[not(p)]).
iRule(r3,h,[calls(p,m0,cr),calls(q,m0,cr),not(r)]).

%% iRules(M,Ids) allocates rules in list Ids into module M.
iRules(m0,[r1,r2]).
iRules(m1,[r3]).
```

MoDiSo engine can run as an independent web service accessible by a Prolog front end. After invoking a Prolog process and loading the front end, users load files containing MoDiSo programs. Figure 16 shows a working session after loading the above program.

4. Conclusions and Future Work

Real-world contract dispute resolution must be guided by contract laws, even if such disputes may be resolved by bodies other than the court of laws. Hence to build support systems for contract dispute resolution, we need a tool capable of representing, reasoning and programming with contract laws. Such a tool would be of great interest to stakeholders of Online Alternative Dispute Resolution whereby

disagreeing parties, facilitated by technologies, come to an agreement short of litigation while are still able to seek legal redress in actual courts. In this paper we presented a tool called MoDiSo which combines the strengths of state-of-the-art argumentation-based formalisms for different aspects of law, to propose: first, a modular architecture for contract dispute resolution systems with an edit-compile-dispute loop facilitating incremental system developments; second a methodology to represent and reason with legal doctrines in the formal language of assumption-based argumentation. We demonstrate the tool with several legal doctrines for performance relief. As a by-product, we obtain a dispute resolution system capable of explaining legal outcomes by automatically generating relevant arguments. As work in progress, MoDiSo currently just fulfils the first step in what we envisage as a three-step development of an tool/environment to support practical contract dispute resolution. The steps are:

- First, establish a proof of concept, determining whether it is *possible at all* to formally represent legal doctrines in contract laws. A pragmatic way to argue for a proposed tool is to show that legal doctrines as stated in Restatements First and Second could be formulated and captured successfully within the tool. To this end, we have applied MoDiSo to several legal doctrines: the doctrine of impossibility with Taylor v Caldwell case (1863), Opera Company v Wolf Trap foundation case (1987); the doctrine of frustration of purpose with Krell v Henry (1903) case, Herne Bay Steamboat Co v Hutton case (1903)[17,19]; the doctrine of mutual mistake with Sherwood v Walker case (1887), Wood v Boyton case (1885)[18]; and many hypothetical E-commerce cases motivated by these classical cases.
- Second, establish a proof of feasibility/technology. Analogous to the fact that it is theoretically possible, but not always feasible to apply the language of Turing machines to write algorithms, it is theoretically possible, but not always feasible to apply the formal language of a tool with an established proof of concept in the first step to formulate dispute contexts in practical settings. Here to establish a proof of feasibility for such a tool, we need to show that its formal language can naturally formulate many real court cases. The formal language of MoDiSo can naturally formulate developed cases whose dispute contexts have been summarised. To argue for this point we plan to use MoDiSo to build a database of decided cases. For cases whose dispute contexts are still being developed in legal proceedings by exchanges of arguments between the parties and the judge, MoDiSo cannot be used yet. Dialogue protocols for such exchanges in a huge body of research[7,13,26,27,36,38] could be of great help in extending MoDiSo to these cases.
- Third, establish a proof of usability. Practices in Online Alternative Dispute Resolution show that simplicity, adaptability, and interoperability are essential usability features of a dispute resolution system.[21] Simplicity here implies that the dispute resolution process must be easily understood, and that the users are in a position to easily follow the progress of their case. Adaptability requires the

system to adjust to the profiles of users and their capabilities to interact with the system or with other parties using the system. Interoperability requires that the system to connect well with other legal systems, for example case databases. We believe that human interfaces using some form of natural language to allow users to formulate their problems, and language translators between this form of natural language and the formal system's language are essential means to develop these usability features.

References

1. *Chitty On Contracts*. Sweet and Maxwell, PO Box 2000, Andover, SP10 9AH, UK, 29 edition, 2004.
2. V. Aleven and K. D. Ashley. Evaluating a learning environment for case-based argumentation skills. In *Proceedings of the 6th international conference on Artificial intelligence and law*, ICAIL '97, pages 170–179, New York, NY, USA, 1997. ACM.
3. K. D. Ashley. Reasoning with cases and hypotheticals in HYPO. *International Journal of Man-Machine Studies*, 34(6):753–796, 1991.
4. K. Atkinson and T. Bench-Capon. Argumentation and standards of proof. In *Proceedings of the 11th international conference on Artificial intelligence and law*, ICAIL '07, pages 107–116, New York, NY, USA, 2007. ACM.
5. K. Atkinson and T. J. M. Bench-Capon. Legal case-based reasoning as practical reasoning. *Artificial Intelligence and Law*, 13(1):93–131, 2005.
6. T. Bench-Capon and H. Prakken. Introducing the logic and law corner. *Journal of Logic and Computation*, 18(1):1–12, 2008.
7. T. J. M. Bench-Capon, K. Atkinson, and A. Chorley. Persuasion and value in legal argument. *Journal of Logic and Computation*, 15(6):1075–1097, 2005.
8. T. J. M. Bench-Capon and S. Modgil. Case law in extended argumentation frameworks. In *Proceedings of the 12th International Conference on Artificial Intelligence and Law*, pages 118–127, Barcelona, Spain, June 2009. ACM Press.
9. T. J. M. Bench-Capon and G. Sartor. A model of legal reasoning with cases incorporating theories and values. *Artificial Intelligence*, 150(1-2):97–143, 2003.
10. D. H. Berman and C. D. Hafner. Representing teleological structure in case-based legal reasoning: the missing link. In *ICAIL '93: Proceedings of the 4th international conference on Artificial intelligence and law*, pages 50–59, New York, NY, USA, 1993. ACM.
11. F. Bex, H. Prakken, C. Reed, and D. Walton. Towards a formal account of reasoning about evidence: argumentation schemes and generalisations. *Artificial Intelligence and Law*, 11(2):125–165, 2003.
12. F. Bex, S. Van den Braak, H. Van Oostendorp, H. Prakken, B. Verheij, and G. Vreeswijk. Sense-making software for crime investigation: how to combine stories and arguments? *Law, Probability and Risk*, 6(1-4):145–168, 2007.
13. P. M. Dung. Logic programming as dialog-game. Technical report, AIT, 1993.
14. 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):321–257, 1995.
15. P. M. Dung, R. Kowalski, and F. Toni. Dialectic proof procedures for assumption-based, admissible argumentation. *Artificial Intelligence*, 170(2):114–159, 2006.
16. P. M. Dung, P. Mancarella, and F. Toni. Computing ideal skeptical argumentation. *Artificial Intelligence*, 171(10-15):642–674, July 2007.

17. P. M. Dung and P. M. Thang. Modular argumentation for modelling legal doctrines in common law of contract. In *Proceedings of the 21st Annual Conference, JURIX 2008*, volume 189 of *Frontiers in Artificial Intelligence and Applications*, pages 108–117, Florence, Italy, Dec. 2008. IOS Press.
18. P. M. Dung and P. M. Thang. Modular Argumentation For Modelling Legal Doctrines in Common Law of Contract. *Artificial Intelligence and Law*, DOI: 10.1007/s10506-009-9076-x, Springer Verlag, June 2009.
19. P. M. Dung, P. M. Thang, and N. D. Hung. Modular argumentation for modelling legal doctrines of performance relief. In *ICAIL*, pages 128–136, 2009.
20. E. A. Farnsworth, W. F. Young, and C. Sanger. *Contracts: Cases and Materials*. ISBN: 1587780577. Foundation Press, 6 edition, June 2001.
21. T. S. Gabrielle Kaufmann-Kohler. *Online dispute resolution : challenges for contemporary justice*. Kluwer Law International, 2004.
22. D. Gaertner and F. Toni. Casapi - a system for credulous and sceptical argumentation. In *First International Workshop on Argumentation and Nonmonotonic Reasoning*, Arizona, USA, 2007.
23. A. J. García and G. R. Simari. Defeasible logic programming: An argumentative approach. *TPLP*, 4(1-2):95–138, 2004.
24. A. v. d. L. Gardner. *An Artificial Intelligence Approach to Legal Reasoning*. Mit Press Series Of Artificial Intelligence And Legal Reasoning. MIT Press, Cambridge, MA, USA, 1987.
25. T. Gordon. *Dialectics, dialogue and argumentation. An examination of Douglas Walton's theories of reasoning and argument*, chapter An Overview of the Carneades Argumentation Support System, pages 145–156. King's College London, 2010.
26. T. F. Gordon. The pleadings game: an exercise in computational dialectics. *Artificial Intelligence and Law*, 2(4):239–292, 1994.
27. T. F. Gordon, H. Prakken, and D. Walton. The Carneades model of argument and burden of proof. *Artificial Intelligence*, 171(10–15):875–896, 2007.
28. G. Governatori, M. J. Maher, G. Antoniou, and D. Billington. Argumentation semantics for defeasible logic. *Journal of Logic and Computation*, 14(5):675–702, 2004.
29. Guenter Freitel. *Frustration and Force Majeure*. ISBN: 9780421778207. Sweet and Maxwell, PO Box 2000, Andover, SP10 9AH, UK, 2nd edition, 2004.
30. J. Hage. A theory of legal reasoning and a logic to match. *Artificial Intelligence and Law*, 4:199–273, 1996. 10.1007/BF00118493.
31. J. C. Hage. Formalizing legal coherence. In *Proceedings of the 8th international conference on Artificial intelligence and law*, ICAIL '01, pages 22–31, New York, NY, USA, 2001. ACM.
32. A. Lodder and J. Zelznikow. Developing an online dispute resolution environment: Dialogue tools and negotiation support systems in a three-step model. *Harvard Negotiation Law Review*, 10:287–337, 2005.
33. L. T. McCarty. An implementation of eisner v. macomber. In *ICAIL '95: Proceedings of the 5th international conference on Artificial intelligence and law*, pages 276–286, New York, NY, USA, 1995. ACM.
34. R. A. Posner. *Economic Analysis of Law*. Wolters Kluwer Law and Business Press, Feb. 2007.
35. H. Prakken. An exercise in formalising teleological case-based reasoning. *Artificial Intelligence and Law*, 10(1-3):113–133, 2002.
36. H. Prakken. Formal systems for persuasion dialogue. *The Knowledge Engineering Review*, 1:163–188, 2006.

37. H. Prakken. An abstract framework for argumentation with structured arguments. *Argument & Computation*, 1(2):93–124, 2010.
38. H. Prakken and G. Sartor. A dialectical model of assessing conflicting arguments in legal reasoning. *Artificial Intelligence and Law*, 4(3-4):331–368, 1996.
39. H. Prakken and G. Sartor. Argument-based extended logic programming with defeasible priorities. *Journal of Applied Non-Classical Logics*, 7(1), 1997.
40. H. Prakken and G. Sartor. Modelling reasoning with precedents in a formal dialogue game. *Artificial Intelligence and Law*, 6(2-4):231–287, 1998.
41. H. Prakken and G. Sartor. More on presumptions and burdens of proof. In *Proceeding of the 2008 conference on Legal Knowledge and Information Systems: JURIX 2008: The Twenty-First Annual Conference*, pages 176–185, Amsterdam, The Netherlands, The Netherlands, 2008. IOS Press.
42. H. Prakken and G. Sartor. A Logical Analysis of Burdens of Proof. *LEGAL EVIDENCE AND PROOF: STATISTICS, STORIES, LOGIC*, pp. 223-253, H. Kaptein, H. Prakken, B. Verheij, eds., Aldershot, 2010.
43. I. Rahwan and G. Simari, editors. *Argumentation in AI*, chapter Argumentation in legal reasoning. Springer-Verlag, 2009.
44. E. Rissland and D. Skalak. Cabaret: rule interpretation in a hybrid architecture. *International Journal of Man-Machine Studies archive*, 34(6):839–887, 1991.
45. B. Roth and B. Verheij. Cases and dialectical arguments - an approach to case-based reasoning. In *OTM Workshops*, volume 3292 of *Lecture Notes in Computer Science*, pages 634–651. Springer, 2004.
46. K. D. A. S. Bruninghaus. Predicting outcomes of case-based legal arguments. In *ICAIL 2003*, NewYork, USA, 2003. ACM Press.
47. Samuel Geoffrey. *Source book on Obligations and Legal Remedies*. Cavendish Publishing Limited, 2000.
48. G. Sartor. Teleological arguments and theory-based dialectics. *Artificial Intelligence and Law*, 10(1):95–112, 2002.
49. 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.
50. J. B. Steven and A. E. Melvin. *Contract Law: Selected Source Materials*. Foundation Press, 2007 edition, 2007.
51. P. M. Thang, P. M. Dung, and N. D. Hung. Towards a common framework for dialectical proof procedures in abstract argumentation. *J. Log. Comput.*, 19(6):1071–1109, 2009.
52. B. Verheij. Deflog: on the logical interpretation of prima facie justified assumptions. *Journal of Logic and Computation*, 13(3):319–346, 2003.
53. H. Yoshino. The systematization of legal meta-inference. In *ICAIL*, pages 266–275, 1995.
54. H. Yoshino. Logical structure of contract law system — for constructing a knowledge base of the united nations convention on contracts for the international sale of goods. *JACIII*, 2(1):2–11, 1998.