Reasoning About Partial Contracts

Shaun AZZOPARDI [a], Albert GATT [b] and Gordon PACE [a]
[a] *Department of Computer Science, University of Malta, Malta*
[b] *Institute of Linguistics, University of Malta, Malta*

Abstract. Natural language techniques have been employed in attempts to automatically translate legal texts, and specifically contracts, into formal models that allow automatic reasoning. However, such techniques suffer from incomplete coverage, typically resulting in parts of the text being left uninterpreted, and which, in turn, may result in the formal models failing to identify potential problems due to these unknown parts. In this paper we present a formal approach to deal with partiality, by syntactically and semantically permitting unknown subcontracts in an action-based deontic logic, with accompanying formal analysis techniques to enable reasoning under incomplete knowledge.

Keywords. Deontic Logic, Partial Logic, Contracts, Automated Reasoning

1. Introduction

Many different formalisations have been given for contracts, all somehow encoding deontic norms (at least obligation and prohibition). Such logics allow us to reason about contracts in different ways, for example given a concurrent obligation and prohibition to do the same action we can conclude from most representations that there is a conflict.

Automatically extracting a formal description of a contract from a natural language text is not trivial. Several approaches utilizing natural language techniques have been presented in the literature that enable the identification of the deontic details (such as the agent of a clause, whether it is an obligation or prohibition, and the directed action or state) of each clause e.g. [11, 5, 14]. Using such approaches, one can construct a formal model of the contract with one caveat: on many texts the process may not produce a complete model of the contract due to limitations in the natural language processing techniques used, which can miss information (such as the agent of a sentence) or identify it incorrectly.

One way of addressing this problem is to have the parsing process be able to identify which parts of the text it can confidently formalise, and which parts might be problematic. This can be done in a variety of ways, from a simple strategy of only tagging as confident parts of the text which match one of a set of templates, to more complex approaches such as associating each part of the text with a confidence factor and choosing a threshold, below which clauses are tagged as indeterminate. Such a threshold can also be dynamically set depending on how critical a part of the text is, e.g. a clause which includes the text "€1,000,000"

would require higher confidence to accept, since misinterpretation of such a clause might have serious effects.

These confidence tags can be used in the translation to a formal notation, supporting more dependable reasoning processes. In this paper we propose a simple action-based deontic logic that supports a notion of explicitly tagging parts of a contract description as unknown or undetermined. The semantics of the deontic logic supports reasoning about these partially parsed contracts by identifying monotonic operators in the logic, which can be used to reason about unknown terms. For instance, we show how conflict identification results effectively in a three-valued predicate in our formalism, with the indeterminism leading to three possible outcomes: (i) conflict certainly present; (ii) conflict might be present; (iii) certainly no conflict present. Our approach is illustrated on a use case, and has been implemented in a tool (in the form of a Microsoft Word plugin) for contract analysis.

2. Similar and Related Work

Our approach allows for syntactically representing uninterpreted parts of a contract. To reason about such parts, one effectively has to consider them not simply as satisfied or violated clauses in a contract, but parts which have an unknown value. Although we do not take a three-valued logic approach, our approach shares much with such logical systems. Classical logic is a bivalent logic, with the only possible truth values being true and false. However, much work has gone into developing three-valued logics to support unknown or undefined values. Kleene [7] introduced a third truth-value to boolean logic, **U**, with the truth tables for boolean operators ($\neg$, $\wedge$, $\vee$, $\Longrightarrow$) extended accordingly. For example, $A \wedge \mathbf{U} = \mathbf{U}$, but $\mathbf{T} \vee \mathbf{U} = \mathbf{T}$, which is consistent with interpreting **U** as an *unknown*, or rather *undecided*, state, i.e. as a state that is either **T** or **F** but is unknown to us. Łukasiewicz [9] similarly uses a third truth-value, but handles implication in a different manner in that an unknown value implies itself. Blamey [4] also illustrates such a logic that considers truth-gaps and non-denoting sentences. These works are related to our approach in that we introduce an *unknown* value into an established semantics and then generalise the operators to deal with it.

In deontic logic we did not find any example of a logic with an explicit *unknown* value, even if some work adopts many-valued logics. Most approaches use three-valued logic to model other features in deontic logic — for instance such an approach has been adopted to classify actions as good, neutral or bad [8], or by allowing sentences to be both true and false, or neither, to allow inconsistent worlds in a semi-paraconsistent logic [10]. Unlike these approaches, our approach works at the syntactic, rather than at the semantic, level.

3. A Deontic Logic with Undetermined Subexpressions

In this section we present a deontic logic with unknown subexpressions and the semantics behind it. The logic is action-based with obligation, permission and

prohibition deontic modalities over actions and tagged by the party subject to the norm. The syntax of the logic is defined in the following manner.

Definition 3.1. A contract C over action alphabet Σ is defined as follows:[1]

$$C := \top \mid \bot \mid O_p(\alpha) \mid P_p(\alpha) \mid F_p(\alpha) \mid ?_p(\alpha) \mid C \rhd C \mid C \blacktriangleright C \mid C \ \& \ C \mid [e]C \mid ?$$

$$\alpha := \Sigma \mid ?^{act}$$

$$p := n \mid ?^{party}$$

$$e := \alpha \mid 0 \mid \varepsilon \mid e.e \mid e + e \mid e \ \& \ e$$

We call the set of well-formed contracts $\mathcal{C}$.

The core of the logic is similar to other action-based deontic logics e.g. [12, 6], but with the addition of unknown terms ?, $?_p(\cdot)$, $?^{act}$ and $?^{party}$.

The logic is action-based, with Σ representing actions which the system can perform. The simplest contracts, which arise in the semantics, are the trivially satisfied contract $\top$ and the trivially violated contract $\bot$. The underlying deontic modalities are (i) *obligation on party p to perform action* $a \in \Sigma$, written $O_p(a)$; (ii) *prohibition on party p to perform action* $a \in \Sigma$, written $F_p(a)$; and (iii) *permission for party p to perform action* $a \in \Sigma$, written $P_p(a)$. Contracts can be composed using (i) sequential composition (written $C \rhd C'$) which means that contract C' is enacted once contract C is satisfied; (ii) conjunction (written $C \ \& \ C'$) which means that both C and C' should hold; (iii) prefix conditional (written $[e]C$) which means that contract C is enacted after event expression e is fully performed (event expressions are built using standard regular expression operators other than repetition — including sequence $e.f$, choice $e + f$ and conjunction $e \ \& \ f$); and (iv) reparation (written $C \blacktriangleright C'$) which means that if and when C is violated, contract C' is enacted.

In addition to these standard deontic operators, the logic includes explicit placeholders for unknown subexpressions, thus allowing the description of a contract which is wholly or partially unknown. These placeholders are (i) the unknown contract: ?; (ii) the undetermined norm on party p over action $a \in \Sigma$: $?_p(a)$; (iii) the undetermined party: $?^{party}$; and (iv) the undetermined action: $?^{act}$.

In the rest of the paper we will write ? instead of $?^{act}$ and $?^{party}$ when no ambiguity arises.

3.1. Semantics of the Logic

We specify the meaning of a contract using an operational semantics, in the form of a relation of the form $C \xrightarrow{A} C'$ meaning that contract $C \in \mathcal{C}$ is transformed into contract $C' \in \mathcal{C}$ when the system performs action set $A \subseteq \Sigma$. In the rest of the paper we will write $\mathbb{\Sigma}$ to denote the power set of Σ, thus ranging over all tags which may appear on transitions. Note that we use action sets as opposed to single actions since, for instance, we may have different obligations in force at the same time, which require multiple actions in order to be satisfied.

[1] Note that the question mark symbol (?) is part of the syntax of the logic.

$$\frac{}{\top \xrightarrow{A} \top} \qquad \frac{}{\bot \xrightarrow{A} \bot}$$

$$\frac{}{O_p(a) \xrightarrow{A} \top}\ a \in A \qquad \frac{}{O_p(a) \xrightarrow{A} \bot}\ a \notin A \qquad \frac{}{F_p(a) \xrightarrow{A} \top}\ a \notin A \qquad \frac{}{F_p(a) \xrightarrow{A} \bot}\ a \in A$$

$$\frac{}{P_p(a) \xrightarrow{A} \top}\ a \in \text{OUTGOING} \qquad \frac{}{P_p(a) \xrightarrow{A} \bot}\ a \notin \text{OUTGOING}$$

$$\frac{C_1 \xrightarrow{A} C_1',\ C_2 \xrightarrow{A} C_2'}{C_1\ \&\ C_2 \xrightarrow{A} C_1'\ \&\ C_2'} \qquad \frac{C_1 \xrightarrow{A} C_1'}{C_1 \rhd C_2 \xrightarrow{A} C_1' \rhd C_2} \qquad \frac{C_1 \xrightarrow{A} C_1'}{C_1 \blacktriangleright C_2 \xrightarrow{A} C_1' \blacktriangleright C_2}$$

$$\frac{C \xrightarrow{A} C'}{[e]C \xrightarrow{A} C'}\ \text{CONTAINSEMPTY}(e) \qquad \frac{}{[e]C \xrightarrow{A} [e']C}\ e' = \text{RESIDUAL}(A, e)$$

Figure 1. Core semantics

The semantics of the core language (without unknowns) is rather standard and is given in Fig. 1. Obligation and prohibition reduce to the satisfied contract or the violated one depending on the actions performed. Permission though does not simply mean that an action can be done, but implies a promise of non-interference from one party to another, as suggested by Von Wright [13] and used in contract automata [3]. It is dealt with using an oracle predicate which allows us to check whether an outgoing permitted action *could* have been taken, and similarly reduces to $\top$ or $\bot$ — we assume that if an action $a \notin$ outgoing, then it cannot appear in any outgoing transition action set.

Conjunction is handled by progressing along both conjunct contracts. Sequential composition ($C \rhd C'$) and reparation ($C \blacktriangleright C'$) allow progress along the first operand. In order to progress further, we require additional formal machinery in the form of a reducing equivalence relation which is applied between standard transitions from left to right until no further reductions are possible:

$$\begin{array}{lll} C\ \&\ \top \equiv C & \bot\ \&\ C \equiv \bot & \top \rhd C \equiv C \\ \top\ \&\ C \equiv C & C\ \&\ \bot \equiv \bot & \bot \blacktriangleright C \equiv C \end{array}$$

This approach of adding syntactic equivalence allows us to simplify the presentation of the semantics in an approach akin to that typically used in process calculi. We will write $C \rightsquigarrow C'$ if C reduces to C' when applying the equivalence relation from left to right on subexpressions of the formula, and no further reductions are possible. It is easy to show that these reductions always terminate and are confluent, thus ensuring that $\rightsquigarrow$ is well-defined.

Finally, prefix guard contracts trigger the contract continuation if the expression includes the empty string, and also allow progress by updating the prefix guard accordingly. The semantics use standard regular expression residuals, where the residual of an expression e with respect to action set A (written residual(A, e)) is the expression e' such that a trace $A.t$ matches expression e if and only if trace

$$\frac{}{? \xrightarrow{A} ?} \qquad \frac{}{?_p(a) \xrightarrow{A} ?}$$

$$\frac{}{O_p(?) \xrightarrow{A} \top}\ A = \Sigma \qquad \frac{}{O_p(?) \xrightarrow{A} \bot}\ A = \varnothing \qquad \frac{}{O_p(?) \xrightarrow{A} ?}\ \text{OTHERWISE}$$

$$\frac{}{F_p(?) \xrightarrow{A} \bot}\ A = \Sigma \qquad \frac{}{F_p(?) \xrightarrow{A} \top}\ A = \varnothing \qquad \frac{}{F_p(?) \xrightarrow{A} ?}\ \text{OTHERWISE}$$

$$\frac{}{P_p(?) \xrightarrow{A} \top}\ A = \Sigma \qquad \frac{}{P_p(?) \xrightarrow{A} ?}\ \text{OTHERWISE}$$

Figure 2. The semantics for terms with unknown contracts

t matches expression e'. For completeness we need to enrich the reduction relation to deal with reduction of prefix guards which can no longer trigger and may thus never lead to a violation again:

$$[e]C \equiv \top \text{ if } isEmpty(e)$$

The semantics for the fragment of the logic which deals with unknown terms is given in Fig. 2. The conservative semantics reduce all unknowns into ?, with special cases for when the the action set is either empty or the full alphabet.

The syntactic equivalence relation is enriched to deal with unknown subformulae:

$$? \ \& \ ? \equiv ? \qquad ? \rhd C \equiv ? \qquad ? \blacktriangleright C \equiv ?$$

We can now proceed to define the languages over traces which lead to satisfaction, violation or unknown contracts:

Definition 3.2. We define $C \stackrel{w}{\Rightarrow} C'$, where $w \in \Sigma^*$ is a string over subsets of the alphabet Σ to be the transitive closure of $\dot{\rightarrow}$ followed by reductions $\rightsquigarrow$:

$$C \stackrel{\epsilon}{\Rightarrow} C' \stackrel{\text{def}}{=} C \rightsquigarrow C'$$
$$C \stackrel{A.w}{\Longrightarrow} C' \stackrel{\text{def}}{=} \exists C'',\ C''' \cdot C \xrightarrow{A} C'' \rightsquigarrow C''' \stackrel{w}{\Rightarrow} C'$$

The set of *violating traces of a contract* C, written $\mathbb{V}(C)$, is defined to be the set of traces that result in the violation $\bot$. The set of *satisfying traces of a contract* C (resulting in $\top$), written $\mathbb{S}(C)$, and the set of *indeterminate traces of contract* C (resulting in the unknown contract ?), written $\mathbb{U}(C)$ are similarly defined:

$$\mathbb{V}(C) \stackrel{\text{def}}{=} \{w : \Sigma^* \mid C \stackrel{w}{\Rightarrow} \bot\}$$
$$\mathbb{S}(C) \stackrel{\text{def}}{=} \{w : \Sigma^* \mid C \stackrel{w}{\Rightarrow} \top\}$$
$$\mathbb{U}(C) \stackrel{\text{def}}{=} \{w : \Sigma^* \mid C \stackrel{w}{\Rightarrow} ?\}$$

A contract C is said to be *partial*, written partial(C), if it contains at least one indeterminate trace: partial(C) $\stackrel{\text{def}}{=} \mathbb{U}(C) \neq \varnothing$.

3.2. Strictness

With these semantics, we can define a relation which relates two contracts C and C' if C is stricter than C' — effectively when the violation traces of C' also violate C. For example, consider $O_p(\mathit{giveReceipt})$ and $O_p(\mathit{giveReceipt})$ & $O_p(\mathit{giveChange})$, the latter is clearly stricter than the former contract.

One can take different interpretations of strictness in the presence of unknown parts of a contract. For instance, consider the contracts $O_p(a)$ & ? and $O_p(a)$ & $O_p(b)$. Depending on the real (though unknown) subcontract represented by ?, the former can be stricter than the latter (e.g. when its actual value is $O_p(b)$ & $O_p(c)$) or vice-versa (e.g. when the actual value is $\top$). We choose to take a worst-case scenario and consider that unknown traces could be violating.

Definition 3.3. The set of *possibly violating traces* of a contract C, written $\mathbb{V}_?(C)$, is defined as: $\mathbb{V}_?(C) \stackrel{\text{def}}{=} \mathbb{V}(C) \cup \mathbb{U}(C)$.

Contract C is said to be *stricter than contract* C', written $C \geq_{\text{str}} C'$, if and only if the possibly violating traces of C' are violations of C: $C \geq_{\text{str}} C' \stackrel{\text{def}}{=} \mathbb{V}_?(C') \subseteq \mathbb{V}(C)$.

This definition gives a sound interpretation to strictness in that if $C \geq_{\text{str}} C'$, then whatever the actual value of unknowns, the strictness relation holds. On the other hand, we lose completeness, in that there are possible values of unknowns under which C would be stricter than C' but $C \geq_{\text{str}} C'$ does not hold. A number of laws of strictness relation can be proved.

Lemma 3.1. (i) Strictness is a partial order; (ii) $\bot$ is stricter than any contract: $\bot \geq_{\text{str}} C$; (iii) $\top$ is weaker than any contract: $C \geq_{\text{str}} \top$.

Full proofs can be found in [1].

3.3. Instantiation of Unknowns

A contract with unknowns can be concretised by replacing unknown terms with known ones. In this section we present an ordering on contracts based on unknown subcontracts.

Definition 3.4. Given (possibly unknown) actions $a, a' \in \Sigma \cup \{?^{\text{act}}\}$, a is said to be more concrete than a', written $a \geq^{\text{act}}_{\text{conc}} a'$, if they are equivalent or a' is the unknown action: $a \geq^{\text{act}}_{\text{conc}} a' \stackrel{\text{def}}{=} a = a' \vee a' = ?^{\text{act}}$. Similarly, we define what it means for a party to be more concrete than another: $p \geq^{\text{party}}_{\text{conc}} p' \stackrel{\text{def}}{=} p = p' \vee p' = ?^{\text{party}}$. Finally, deontic modalities (possibly unknown) D and D' can be similarly compared: $D \geq^{\text{deon}}_{\text{conc}} D' \stackrel{\text{def}}{=} D = D' \vee D' = ?$

A contract C is said to be *more concrete than contract* C', written $C \geq_{\text{conc}} C'$, if C is syntactically identical to C' but with some unknown subcontracts resolved:

$$\frac{}{C \geq_{\text{conc}} ?} \qquad \frac{p \geq^{\text{party}}_{\text{conc}} p',\ a \geq^{\text{act}}_{\text{conc}} a',\ D \geq^{\text{deon}}_{\text{conc}} D'}{D_p(a) \geq_{\text{conc}} D'_{p'}(a')}$$

This relation is also a partial order.

Lemma 3.2. The concreteness comparison relation $\geq_{\text{conc}}$ is a partial order.

3.4. Conflicts

Analysis of contracts under the possibility of unknown subcontracts allows reasoning about contracts despite that parts of these contracts are either not yet available or were not fully formalised due to limitations in the parsing techniques or otherwise. In this section we present an axiomatisation of conflicts with unknowns.

In order to enrich conflict analysis, in the rest of this section we will assume a mutually exclusive actions relation, where we write $a \bowtie b$ to signify that actions a and b are mutually exclusive and may not appear together in an action set.

Given unknowns, which may possibly, but not necessarily, lead to a conflict, we define two conflict relations between contracts: (i) $C \underset{must}{\maltese} C'$ to signify that there certainly is a conflict between C and C'; and (ii) $C \underset{may}{\maltese} C'$ to signify that a conflict might be present.

Definition 3.5. The relation between contracts $\underset{must}{\maltese} \in \mathcal{C} \leftrightarrow \mathcal{C}$ such that $C \underset{must}{\maltese} C'$ indicating that C *necessarily conflicts with* C' is defined through the following axioms and rules:

$$\begin{aligned}
\vdash P_p(a) &\underset{must}{\maltese} F_p(a) \\
a \bowtie a' \vdash O_p(a) &\underset{must}{\maltese} O_p(a') \\
a \bowtie a' \vdash O_p(a) &\underset{must}{\maltese} P_p(a') \\
C_1 \underset{must}{\maltese} C_2 \vdash C_2 &\underset{must}{\maltese} C_1 \\
C_1 \underset{must}{\maltese} C_2 \text{ and } C_1' \geq_{\text{str}} C_1 \vdash C_1' &\underset{must}{\maltese} C_2
\end{aligned}$$

Similarly, the relation $\underset{may}{\maltese} \in \mathcal{C} \leftrightarrow \mathcal{C}$ indicating that C *may conflict with* C' is defined as follows:

$$\begin{aligned}
C \underset{must}{\maltese} C' \vdash C &\underset{may}{\maltese} C' \\
C \underset{may}{\maltese} C' \vdash C' &\underset{may}{\maltese} C \\
C_1 \underset{may}{\maltese} C_2 \text{ and } C_1' \geq_{\text{str}} C_1 \vdash C_1' &\underset{may}{\maltese} C_2 \\
C_1 \underset{may}{\maltese} C_2 \text{ and } C_1 \geq_{\text{conc}} C_1' \vdash C_1' &\underset{may}{\maltese} C_2
\end{aligned}$$

Consider a contract which obliges party p to board a plane, $O_p(\mathit{boardPlane})$, but also prohibits this action, $F_p(\mathit{boardPlane})$. By Lemma 3.1, it follows that $O_p(\mathit{boardPlane}) \geq_{\text{str}} P_p(\mathit{boardPlane})$ and by the first conflict axiom, then $P_p(\mathit{boardPlane}) \underset{must}{\maltese} F_p(\mathit{boardPlane})$. Therefore, using the fifth conflicts axiom, it follows that $O_p(\mathit{boardPlane}) \underset{must}{\maltese} F_p(\mathit{boardPlane})$.

Contract	**Pattern**
Norm	⟨obligation⟩ | ⟨permission⟩ | ⟨prohibition⟩
Obligation	⟨agent⟩ should ⟨action⟩.
Permission	⟨agent⟩ may ⟨action⟩.
Prohibition	⟨agent⟩ should not ⟨action⟩.
Concurrency	⟨contract⟩ and ⟨contract⟩.
Sequential Composition	⟨norm⟩, after which ⟨norm⟩.
Conditional	If ⟨action⟩, ⟨contract⟩.
Reparation	⟨contract⟩. If this is violated, ⟨contract⟩.
	⟨contract⟩, otherwise ⟨contract⟩.

Figure 3. Some possible rules for an English-to-DL algorithm.

4. Case Study

The logic we have presented has been implemented in a proof-of-concept tool which supports natural language contract analysis [2]. The tool has been implemented as a Word plugin and uses natural language techniques to attempt to parse English contracts into the logic we have presented in this paper. The parsing algorithms are conservative in that we preferred to err on the side of leaving parts of a contract unparsed (i.e. formalised as the unknown contract ?) rather than parse parts of the contract incorrectly. The logic and conflict analysis presented in this paper were required to enable us to be able to analyse contracts which were not fully parsed. In this section we present a small use case to show how a contract in English, and which is parsed (using a simple set of heuristics) into our logic with unknown (unparsed) parts, can still be analysed for conflicts.

In English, norms can be specified in several ways. We consider an algorithm that formalises a natural language text by defining pattern matching rules, e.g. ⟨*agent*⟩ ⟨**norm-specifier**⟩ ⟨action⟩, would match each unit specified in the following sentence as marked: '*passenger* **should** have their boarding pass'. Some rules that can be considered are illustrated in Fig 3.

Given some set of such rules, consider the contract shown in Fig 4. Our unknowns can be used when sentences do not match any of the rules (or match them partially) of a certain algorithm, but more deeply, unknowns can be used to represent those cases when we cannot determine the semantics of a sentence, or it is ambiguous. For example, looking at the first clause, it is not specified exactly what the local regulations are, since these would depend on which country the passenger is catching the plane. An algorithm could thus overlook this clause, by representing it as the unknown contract, but still parse the rest of the unambiguous clauses, allowing partial analysis of the contract.

Using unknowns in this case allows us to maintain enough of the contract structure to be able to analyse it for conflicts. In fact note how if we assume that the two actions *checkInDesk* and *leave* are mutually exclusive, then we can conclude that the clauses presented must conflict, since the first argument to each reparation operator may be violated at the same time, reaching a state where both $O_{passenger}(leave)$ and $O_{passenger}(checkInDesk)$ hold, using the second conflicts axiom. The contract may also conflict, since after the passenger arrives at the boarding gate, the formalised contract representation cannot parse the first

1. *Provided the passenger satisfies any local airport regulations and checks, when at the boarding gate, the passenger should board the plane. If this is violated, they should return to the check-in desk.*
2. *If the passenger is at the boarding gate, the passenger should not be carrying any weapons, otherwise the passenger should leave the airport.*

Formalisation:

1. $([atGate]? \rhd O_p(boardPlane)) \blacktriangleright O_p(checkInDesk)$
2. $([atGate]O_p(noWeapons)) \blacktriangleright O_p(leave)$

Figure 4. Airline contract example [3], and formalisation given patterns in Fig 3.

obligation (i.e. it cannot elicit what the normative sentence requires), meaning it has to assume, for soundness, that it may conflict with any other norms active at the same time (in this case with $O_p(noWeapons)$).

This case illustrates how our logic enables imperfect algorithms to still give useful output that is amenable to logical analysis.

5. Conclusions

We have presented a logic that introduces the concept of unknowns in a deontic context. These unknowns are used both to represent clauses or parts of these whose deontic meaning was not parsed or is not yet available. The applicability of such an approach has been shown through a use case of a contract between an airline company and its passengers — using unknowns to allow reasoning under imperfect knowledge (e.g. conflict analysis). The formal logic analysis algorithms presented in this paper have also been implemented in a tool [2].

The treatment of unknown contracts in our approach is rather coarse-grained and can be used to represent any contract. We envisage building more fine-grained formalisation and reasoning tools, for instance enabling reasoning about unknown contracts which can range over a limited set, or enable negatively tagged unknown parts (e.g. this contract certainly contains no obligation). Another limitation of our approach is that there is no way of relating unknown parts of a contract e.g. the same unparseable clause may appear twice in a contract, in which case our concretisation relation allows for refining the unknown parts in different ways, even if we know that they have to be identical. On the other hand, such finer-grained analysis will mean more computationally expensive analysis and more room for error when parsing natural language contracts.

References

[1] Shaun Azzopardi. Intelligent contract editing. Master's thesis, Department of Computer Science, University of Malta, 2015.

[2] Shaun Azzopardi, Albert Gatt, and Gordon J. Pace. Integrating natural language and formal analysis for legal documents. In *10th Conference on Language Technologies and Digital Humanities 2016*, 2016.

[3] Shaun Azzopardi, Gordon J. Pace, and Fernando Schapachnik. Contract automata with reparations. In *Legal Knowledge and Information Systems - JURIX 2014: The Twenty-Seventh Annual Conference, Jagiellonian University, Krakow, Poland, 10-12 December 2014*, pages 49–54, 2014.

[4] Stephen Blamey. Partial logic. In D. Gabbay and F. Guenthner, editors, *Handbook of Philosophical Logic*, volume 166 of *Synthese Library*, pages 1–70. Springer Netherlands, 1986.

[5] Xibin Gao and Munindar P. Singh. Extracting normative relationships from business contracts. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '14, pages 101–108, Richland, SC, 2014. International Foundation for Autonomous Agents and Multiagent Systems.

[6] Guido Governatori and Antonino Rotolo. Logic of violations: A gentzen system for reasoning with contrary-to-duty obligations. *Australasian Journal of Logic*, 4:193–215, 2006.

[7] S. C. Kleene. *Introduction to Metamathematics*. D. Van Nostrand, 1950.

[8] Piotr Kulicki and Robert Trypuz. Doing the right things – trivalence in deontic action logic. In Paul Egre and David Ripley, editors, *Trivalent Logics and their applications, Proceedings of the ESSLLI 2012 Workshop*, pages 53–63, 2012.

[9] Gregorz Malinowski. Many-valued logic and its philosophy. In Dov M. Gabbay and John Woods, editors, *The Many Valued and Nonmonotonic Turn in Logic*, volume 8 of *Handbook of the History of Logic*, pages 13 – 94. North-Holland, 2007.

[10] Casey McGinnis. Semi-paraconsistent deontic logic. In W.A. Carnielli J.Y. Beziau, editor, *Paraconsistency with No Frontiers*. Elsevier, 2006.

[11] Wim Peters and Adam Wyner. Extracting hohfeldian relations from text. In *Legal Knowledge and Information Systems - JURIX 2015: The Twenty-Eighth Annual Conference, Braga, Portual, December 10-11, 2015*, pages 189–190, 2015.

[12] Cristian Prisacariu and Gerardo Schneider. A formal language for electronic contracts. In *Formal Methods for Open Object-Based Distributed Systems, 9th IFIP WG 6.1 International Conference, FMOODS 2007, Paphos, Cyprus, June 6-8, 2007, Proceedings*, pages 174–189, 2007.

[13] Georg Henrik Von Wright. Deontic logic: A personal view. *Ratio Juris*, 12(1):26–38, 1999.

[14] Adam Wyner and Wim Peters. On rule extraction from regulations. In *Legal Knowledge and Information Systems - JURIX 2011: The Twenty-Fourth Annual Conference, University of Vienna, Austria, 14th-16th December 2011*, pages 113–122, 2011.