Automated Reasoning in Deontic Logic[*]

Ulrich Furbach[1], Claudia Schon[1] and Frieder Stolzenburg[2]

[1] Universität Koblenz-Landau, Koblenz, Germany
{uli,schon}@uni-koblenz.de
[2] Harz University of Applied Sciences, Wernigerode, Germany
fstolzenburg@hs-harz.de

Abstract. Deontic logic is a very well researched branch of mathematical logic and philosophy. Various kinds of deontic logics are discussed for different application domains like argumentation theory, legal reasoning, and acts in multi-agent systems. In this paper, we show how standard deontic logic can be stepwise transformed into description logic and DL-clauses, such that it can be processed by Hyper, a high performance theorem prover which uses a hypertableau calculus. Two use cases, one from multi-agent research and one from the development of normative system are investigated.

Keywords: Deontic Logic, Automated Theorem Proving, Description Logics.

1 Introduction

Deontic logic is a very well researched branch of mathematical logic and philosophy. Various kinds of deontic logics are discussed for different application domains like argumentation theory, legal reasoning, and acts in multi-agent systems [11]. Recently there also is growing interest in modelling human reasoning and testing the models with psychological findings. Deontic logic is an obvious tool to this end, because norms and licenses in human societies can be described easily with it. For example in [9] there is a discussion of some of these problems including solutions with the help of deontic logic. There, the focus is on using deontic logic for modelling certain effects, which occur in human reasoning, e.g. the Wason selection task or Byrne's suppression task.

This paper concentrates on automated reasoning in standard deontic logic (SDL). Instead of implementing a reasoning system for this logic directly, we rather rely on existing methods and systems. Taking into account that SDL is just the modal logic K with a seriality axiom, we show that deontic logic can be translated into description logic $\mathcal{ALC}$. The latter can be transformed into so called DL-clauses, which is a special normal form with clauses consisting of implications where the body is, as usual, a conjunction of atoms and the head is a disjunction of literals. These literals can be atoms or existential quantified expressions.

DL-clauses can be decided by the first-order reasoning system Hyper [22], which uses the hypertableau calculus from [4]. In Sections 2 and 3 we shortly depict this workflow, and in Section 4 we demonstrate the use of our technique with the help of two problems from the literature, one from multi-agent research and the other one from testing normative systems. We choose these examples, because they hopefully document the applicability of reasoning of SDL in various areas of AI research.

2   Deontic Logic as Modal Logic KD

We consider a simple modal logic which consists of propositional logic and the additional modal operators $\Box$ and $\Diamond$. Semantics are given as possible world semantics, where the modal operators $\Box$ and $\Diamond$ are interpreted as quantifiers over possible worlds. Such a possible world is an assignment, which assigns truth values to the propositional variables. An interpretation connects different possible worlds by a (binary) reachability relation R. The $\Box$-operator states that a formula has to hold in all reachable worlds. Hence if v and w are worlds, we have

$$w \models \Box P \qquad \text{iff} \qquad \forall v : R(w,v) \to v \models P$$

Standard deontic logic (SDL) is obtained from the well-known modal logic K by adding the seriality axiom D:

$$\mathsf{D} : \quad \Box P \to \Diamond P$$

In this logic, the $\Box$-operator is interpreted as 'it is obligatory that' and the $\Diamond$ as 'it is permitted that'. The $\Diamond$-operator can be defined by the following equivalence:

$$\Diamond P \equiv \neg \Box \neg P$$

The additional axiom D: $\Box P \to \Diamond P$ in SDL states that, if a formula has to hold in all reachable worlds, then there exists such a world. With the deontic reading of $\Box$ and $\Diamond$ this means: Whenever the formula P ought to be, then there exists a world where it holds. In consequence, there is always a world, which is ideal in the sense, that all the norms formulated by 'the ought to be'-operator hold.

SDL can be used in a natural way to describe knowledge about norms or licenses. The use of conditionals for expressing rules which should be considered as norms seems likely, but holds some subtle difficulties. If we want to express that *if P then Q* is a norm, an obvious solution would be to use

$$\Box(P \to Q)$$

which reads *it is obligatory that Q holds if P holds.* An alternative would be

$$P \to \Box Q$$

meaning *if P holds, it is obligatory that Q holds.* In [21] there is a careful discussion which of these two possibilities should be used for conditional norms. The first one has severe disadvantages. The most obvious disadvantage is, that P together with $\Box(P \to Q)$ does not imply $\Box Q$. This is why we prefer the latter method, where the $\Box$-operator is in the conclusion of the conditional. We will come back to this point in Subsection 4.1 where we consider several formalization variants of the well-known problem of contrary-to-duty-obligations. For a more detailed discussion of such aspects we refer to [10].

3   Automated Reasoning for Deontic Logic

Deontic logic is the logic of choice when formalizing knowledge about norms like the representation of legal knowledge. However, there are only few automated theorem provers specially dedicated for deontic logic and used by deontic logicians (see [1,3]). Nonetheless, numerous approaches to translate modal logics into (decidable fragments of) first-order predicate logics are stated in the literature. A nice overview including many relevant references is given in [19].

In this paper, we describe how to use the Hyper theorem prover [22] to handle deontic logic knowledge bases. These knowledge bases can be translated efficiently into description logic formulae. Hyper is a theorem prover for first-order logic with equality. In addition to that, Hyper is a decision procedure for the description logic $\mathcal{SHIQ}$ [6].
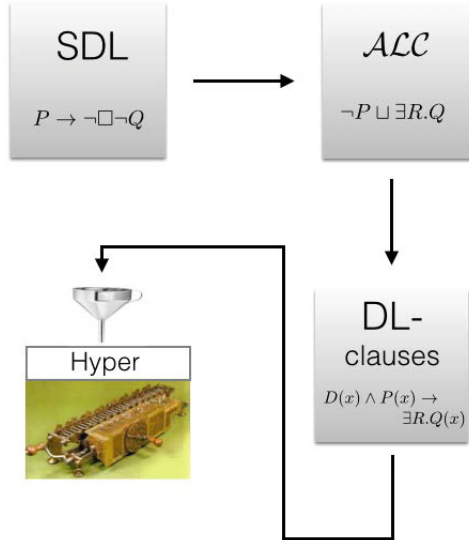

Fig. 1. From SDL to Hyper. Note that concept D occurring in the DL-clauses is an auxiliary concept.

In Figure 1, we depict the entire workflow from a given SDL knowledge base to the final input into the Hyper theorem prover. In the following, we describe these three steps in more detail.

3.1   Transformation from Deontic Logic into $\mathcal{ALC}$

First, we will show how to translate SDL knowledge bases into $\mathcal{ALC}$ knowledge bases. An $\mathcal{ALC}$ knowledge base consists of a TBox and an ABox. The TBox (terminological box) gives information about concepts occurring in the domain of interest and describes concept hierarchies. The ABox (assertional box) introduces individuals and states, to which concepts the individuals belong and how they are interconnected via relations called roles . The ABox contains assertional knowledge and can be seen as the representation of a state of the world. We do not give the syntax and semantics of $\mathcal{ALC}$ here and refer the reader to [2].

There is a strong connection between modal logic and the description logic $\mathcal{ALC}$. As shown in [18], the description logic $\mathcal{ALC}$ is a notational variant of the modal logic K_n. Therefore any formula given in the modal logic K_n can be translated into an $\mathcal{ALC}$ concept and vice versa. Since we are only considering a modal logic as opposed to a multimodal logic, we will omit the part of the translation handling the multimodal part of the logic. Mapping φ translating from modal logic K formulae to $\mathcal{ALC}$ concepts is inductively defined as follows:

$$\varphi(\top) = \top$$
$$\varphi(\bot) = \bot$$
$$\varphi(b) = b$$
$$\varphi(\neg c) = \neg\varphi(c)$$
$$\varphi(c \wedge d) = \varphi(c) \sqcap \varphi(d)$$
$$\varphi(c \vee d) = \varphi(c) \sqcup \varphi(d)$$
$$\varphi(\Box c) = \forall r.\varphi(c)$$
$$\varphi(\Diamond c) = \exists r.\varphi(c)$$

Note that the mapping φ is a one-to-one mapping.

Formulae given in SDL can be translated into $\mathcal{ALC}$ concepts using the above introduced φ mapping. For a normative system consisting of the set of deontic logic formulae $\mathcal{N} = \{F_1, \ldots, F_n\}$ the translation is defined as the conjunctive combination of the translation of all deontic logic formulae in $\mathcal{N}$:

$$\varphi(\mathcal{N}) = \varphi(F_1) \sqcap \ldots \sqcap \varphi(F_n)$$

Note that $\varphi(\mathcal{N})$ does not yet contain the translation of the seriality axiom. As shown in [12] the seriality axiom can be translated into the following TBox:

$$\mathcal{T} = \{\top \sqsubseteq \exists r.\top\}$$

with r the atomic role introduced by the mapping φ.

For our application, the result of the translation of a normative system $\mathcal{N}$ and the seriality axiom is an $\mathcal{ALC}$ knowledge base $\Phi(\mathcal{N}) = (\mathcal{T}, \mathcal{A})$, where the TBox $\mathcal{T}$

consists of the translation of the seriality axiom and the ABox $\mathcal{A} = \{(\varphi(\mathcal{N}))(a)\}$ for a new individual a. In description logics performing a satisfiability test of a concept C w.r.t. a TBox is usually done by adding a new individual a together with the ABox assertion $C(a)$. For the sake of simplicity, we do this construction already during the transformation of Φ by adding $(\varphi(\mathcal{N}))(a)$ to the ABox.

An advantage of the translation of deontic logic formulae into an $\mathcal{ALC}$ knowledge base is the existence of a TBox in $\mathcal{ALC}$. This makes it possible to add further axioms to the TBox. For example we can add certain norms that we want to be satisfied in all reachable worlds into the TBox.

3.2   Translation from $\mathcal{ALC}$ into DL-Clauses

Next we transform the $\mathcal{ALC}$ knowledge base into so called DL-clauses introduced in [15] which represent the input format for the Hyper theorem prover.

DL-clauses are constructed from so called *atoms*. An atom is of the form $b(s)$, $r(s,t)$, $\exists r.b(s)$ or $\exists r.\neg b(s)$ for b an atomic concept and s and t individuals or variables. They are universally quantified implications of the form

$$\bigwedge_{i=1}^{m} u_i \rightarrow \bigvee_{j=1}^{n} v_j$$

where the u_i are atoms of the form $b(s)$ or $r(s,t)$ and the v_j may be arbitrary DL-clause atoms, i.e. including existential quantification, with $m, n \geq 0$.

Comparing the syntax of DL-clauses to the syntax of first order logic clauses written as implications, the first obvious difference is the absence of function symbols. The second difference is the fact, that in DL-clauses all atoms are constructed from unary or binary predicates. The most interesting difference however is the fact, that the head of a DL-clause is allowed to contain atoms of the form $\exists r.b(s)$.

The basic idea of the translation of an $\mathcal{ALC}$ knowledge base into DL-clauses is that the subsumption in a TBox assertion is interpreted as an implication from the left to the right side. Further concepts are translated to unary and roles to binary predicates. Depending on the structure of the assertion, auxiliary concepts are introduced. For example the TBox axiom

$$d \sqsubseteq \exists r.b \sqcup \forall r.c$$

corresponds to the following DL-clause

$$d(x) \wedge r(x,y) \rightarrow c(y) \vee \exists r.b(x)$$

For detailed definitions of both syntax and semantics of DL-clauses and the translation into DL-clauses, we refer the reader to [15]. The translation preserves equivalence, avoids an exponential blowup by using a well-known structural transformation [17] and can be computed in polynomial time. In the following, for an $\mathcal{ALC}$ knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, the corresponding set of DL-clauses is denoted by $\omega(\mathcal{K})$.

3.3   Reasoning Tasks

With the help of Hyper, we can solve several interesting reasoning tasks:

– **Consistency checking of normative systems:** In practice, normative systems can be very large. Therefore it is not easy to see, if a given normative system is consistent. The Hyper theorem prover can be used to check consistency of a normative system $\mathcal{N}$. We first translate $\mathcal{N}$ into an $\mathcal{ALC}$ knowledge base $\Phi(\mathcal{N})$, then translate $\Phi(\mathcal{N})$ into the set $\omega(\Phi(\mathcal{N}))$ of DL-clauses. Then we can check the consistency of $\omega(\Phi(\mathcal{N}))$ using Hyper.
– **Evaluation of normative systems:** Given several normative systems, we use Hyper to find out for which normative system guarantees a desired outcome is guaranteed.
– **Independence checking:** Given a normative system $\mathcal{N}$ and a formula F representing a norm, we can check whether F is independent from $\mathcal{N}$. If F is independent from $\mathcal{N}$, then F is not a logical consequence of $\mathcal{N}$.

In Section 4, we will give detailed examples for those tasks. Subsection 4.1 gives an example for a consistency check of a normative system and illustrates how the independence of a formula from a normative system can be decided. In Subsection 4.2, we use an example from multi-agent systems to show how to evaluate normative systems.

4   Applications

The literature on deontic logic deals with numerous small but nonetheless interesting examples. They are mostly used to show typical problems or special features of the logic under consideration (cf. [10]). In Subsection 4.1, we deal with one of these examples. In Subsection 4.2, we formalize a 'real-life' problem from multi-agent research.

4.1   Contrary-to-duty Obligations

Let us now consider consistency testing of normative systems and independence checking. As an example, we examine the well-known problem of *contrary-to-duty obligations* introduced in [8]:

(1) *a* ought not steal.
(2) *a* steals.
(3) If *a* steals, he ought to be punished for stealing.
(4) If *a* does not steal, he ought not be punished for stealing.

Table 1 shows three different formalizations of this problem. Those formalizations are well-known from the literature [5,14,13,21]:

Table 1. Formalizations of the *contrary-to-duty obligation* introduced in [8]

	$\mathcal{N}_1$	$\mathcal{N}_2$	$\mathcal{N}_3$
(1)	$\Box\neg s$	$\Box\neg s$	$\Box\neg s$
(2)	s	s	s
(3)	$s \rightarrow \Box p$	$\Box(s \rightarrow p)$	$s \rightarrow \Box p$
(4)	$\Box(\neg s \rightarrow \neg p)$	$\Box(\neg s \rightarrow \neg p)$	$\neg s \rightarrow \Box\neg p$

Table 2. Translation of the normative system $\mathcal{N}_1$ into $\varphi(\mathcal{N}_1)$

$\mathcal{N}_1$ (in Deontic Logic)	$\varphi(\mathcal{N}_1)$
$\Box\neg s$	$\forall r.\neg s$
s	s
$s \rightarrow \Box p$	$\neg s \sqcup \forall r.p$
$\Box(\neg s \rightarrow \neg p)$	$\forall r.(s \sqcup \neg p)$

Consistency Testing of Normative Systems. The contrary-to-duty obliga-
tion formalized above is a very small example. In practice, normative systems
can be rather complex. This makes it difficult to see if a normative system is
consistent. We will show how to use the Hyper theorem prover to check the
consistency of a given normative system.

As an example, we consider formalization $\mathcal{N}_1$ given in Table 1 which, according
to [21], is inconsistent. We will use Hyper to show this inconsistency. For this, we
first translate normative system $\mathcal{N}_1$ into an $\mathcal{ALC}$ knowledge base $\Phi(\mathcal{N}_1)$. Table 2
shows $\varphi(\mathcal{N}_1)$.

To perform the satisfiability test, we transform the description logic represen-
tation $\Phi(\mathcal{N}_1)$ into a set of DL-clauses $\omega(\Phi(\mathcal{N}_1))$. Hyper constructs a hypertableau
for $\omega(\Phi(\mathcal{N}_1))$. This hypertableau is closed and therefore we can conclude that
$\mathcal{N}_1$ is inconsistent.

Independence Checking. Normative System $\mathcal{N}_2$ given in Table 1 is consistent.
However it has another drawback: The different formulae in this formalization
are not independent from another. Formula (3) is a logical consequence of (1),
because $\Box(s \rightarrow p) \equiv \Box(\neg s \vee p)$ (definition of $\rightarrow$) which clearly is implied by the
(subsuming) formula (1) $\Box\neg s$. We can use Hyper to show this by transforming
the problem into a satisfiability test. For this, we remove formula (3) from $\mathcal{N}_2$
and add its negation $\neg\Box(s \rightarrow p)$ to $\mathcal{N}_2$. If the resulting normative system is
inconsistent, we can conclude, that formula (3) is not independent from the
other formulae in $\mathcal{N}_2$.

The problem of independence of formulae given in a normative system is in-
teresting in practice as well. If an existing normative system is extended with
some new formulae, it is interesting to know, whether the new formulae are inde-
pendent from the original normative system. This can be checked automatically
using Hyper as described above.

In the same way, we can show, that formula (4) is not independent from $\mathcal{N}_3$. Note that only this normative system is both consistent and represents all conditionals carefully, i.e. with formulae of the form $P \to \Box Q$ (cf. Section 2). Only for this formalization we have: If a steals in the actual world, a will be punished in the corresponding reachable ideal world.

4.2   An Example from Multi-agent Systems

In multi-agent systems, there is a relatively new area of research, namely the formalization of 'robot ethics'. It aims at defining formal rules for the behavior of agents and to prove certain properties. As an example consider Asimov's laws, which aim at regulating the relation between robots and humans. In [7] the authors depict a small example of two surgery robots obeying ethical codes concerning their work. These codes are expressed by means of MADL, which is an extension of standard deontic logic with two operators. In [16] an axiomatization of MADL is given. Further it is asserted, that MADL is not essentially different from standard deontic logic. This is why we use SDL to model the example.

Formalization in SDL. In our example, there are two robots *ag1* and *ag2* in a hospital. For sake of simplicity, each robot can perform one specific action: *ag1* can terminate a person's life support and *ag2* can delay the delivery of pain medication. In [7] four different ethical codes J, $J^\star$, O and $O^\star$ are considered:

– "If ethical code J holds, then robot *ag1* ought to take care, that life support is terminated." This is formalized as:

$$J \to \Box act(ag1, term)$$

– "If ethical code $J^\star$ holds, then code J holds, and robot *ag2* ought to take care, that the delivery of pain medication is delayed." This is formalized as:

$$J^\star \to J \wedge J^\star \to \Box act(ag2, delay)$$

– "If ethical code O holds, then robot *ag2* ought to take care, that delivery of pain medication is not delayed." This is formalized as:

$$O \to \Box \neg act(ag2, delay)$$

– "If ethical code $O^\star$ holds, then code O holds, and robot *ag1* ought to take care, that life support is not terminated." This is formalized as:

$$O^\star \to O \wedge O^\star \to \Box \neg act(ag1, term)$$

Further we give a slightly modified version of the evaluation of the robot's acts given in [7], where $(+!!)$ describes the most and $(-!!)$ the least desired outcome. Note that terms like $(+!!)$ are just propositional atomic formulae here.

$$act(ag1, term) \wedge \ \ act(ag2, delay) \to (-!!) \tag{1}$$
$$act(ag1, term) \wedge \neg act(ag2, delay) \to (-!) \tag{2}$$
$$\neg act(ag1, term) \wedge \ \ act(ag2, delay) \to (-) \tag{3}$$
$$\neg act(ag1, term) \wedge \neg act(ag2, delay) \to (+!!) \tag{4}$$

Table 3. Translation of the normative system $\mathcal{N}$ into $\varphi(\mathcal{N})$

Deontic Logic	$\mathcal{ALC}$
$J \rightarrow \Box act(ag1, term)$	$\neg J \sqcup \forall r.act(ag1, term)$
$J^\star \rightarrow J \wedge J^\star \rightarrow \Box act(ag2, delay)$	$(\neg J^\star \sqcup J) \sqcap (\neg J^\star \sqcup \forall r.act(ag2, delay))$
$O \rightarrow \Box \neg act(ag2, delay)$	$\neg O \sqcup \forall r.\neg act(ag2, delay)$
$O^\star \rightarrow O \wedge O^\star \rightarrow \Box \neg act(ag1, term)$	$(\neg O^\star \sqcup O) \sqcap (\neg O^\star \sqcup \forall r.\neg act(ag1, term))$
$act(ag1, term) \wedge \quad act(ag2, delay) \rightarrow (-!!)$	$\neg(act(ag1, term) \sqcap \quad act(ag2, delay)) \sqcup (-!!)$
$act(ag1, term) \wedge \neg act(ag2, delay) \rightarrow (-!)$	$\neg(act(ag1, term) \sqcap \neg act(ag2, delay)) \sqcup (-!)$
$\neg act(ag1, term) \wedge \quad act(ag2, delay) \rightarrow (-)$	$\neg(\neg act(ag1, term) \sqcap \quad act(ag2, delay)) \sqcup (-)$
$\neg act(ag1, term) \wedge \neg act(ag2, delay) \rightarrow (+!!)$	$\neg(\neg act(ag1, term) \sqcap \neg act(ag2, delay)) \sqcup (+!!)$

These formulae evaluate the outcome of the robots' actions. It makes sense to assume, that this evaluation is effective in all reachable worlds. This is why we add formulae stating that formulae (1)–(4) hold in all reachable worlds. For example, for (1) we add:

$$\Box(act(ag1, term) \wedge act(ag2, delay) \rightarrow (-!!)) \tag{5}$$

Since our example does not include nested modal operators, the formulae of the form (5) are sufficient to spread the evaluation formulae to all reachable worlds. The normative system $\mathcal{N}$ formalizing this example consists of the formalization of the four ethical codes and the formulae for the evaluation of the robots actions.

Reduction to a Satisfiability Test A possible query would be to ask, if the most desirable outcome $(+!!)$ will come to pass, if ethical code $O^\star$ is operative. This query can be translated into a satisfiability test: If

$$\mathcal{N} \wedge O^\star \wedge \Diamond \neg(+!!)$$

is unsatisfiable, then ethical code $O^\star$ ensures outcome $(+!!)$.

Translation into Description Logic. As described in Section 3.1, we translate normative system $\mathcal{N}$ given in the previous section into an $\mathcal{ALC}$ knowledge base $\Phi(\mathcal{N}) = (\mathcal{T}, \mathcal{A})$. Table 3 shows the result of translating $\mathcal{N}$ into $\varphi(\mathcal{N})$.

We further add the following two assertions to the ABox $\mathcal{A}$:

$$O^\star(a)$$
$$\exists r.\neg(+!!)(a)$$

Next we translate the knowledge base into DL-clauses and use Hyper to test the satisfiability of the resulting set of DL-clauses. Using further satisfiability tests, we can show, that ethical codes J, $J^\star$ or O are not sufficient to guarantee the most desired outcome $(+!!)$.

Formalization in Description Logic Using a TBox. In the formalization given in the previous subsection, we added formulae stating that the evaluation

of the agents' actions holds in all worlds, which are reachable in one step, see (5) for an example. In our case it is sufficient to add formulae of the form (5) because the formalization does not include nested modal operators. In general it is desirable to express that those formulae hold in *all* reachable worlds including worlds reachable in more than one step. However this would mean to either add infinitely many formulae or to use a universal modality, i.e. the reflexive-transitive closure of the respective simple modality.

In description logics we can use a more elegant way to formalize that all worlds are supposed to fulfill certain formulae. Description logic knowledge bases contain a TBox including the terminological knowledge. Every individual is supposed to fulfill the assertions given in the TBox. Hence, we can add the formulae stating the evaluation of the agents' actions into the TBox. For this, we reformulate implication ($\rightarrow$) by subsumption ($\sqsubseteq$). We model the deontic logic formulae given in Table 3 by the following TBox $\mathcal{T}$:

$$\top \sqsubseteq \exists r.\top$$
$$J \sqsubseteq \forall r.act(ag1, term)$$
$$J^\star \sqsubseteq J$$
$$J^\star \sqsubseteq \forall r.act(ag2, delay)$$
$$O \sqsubseteq \forall r.\neg act(ag2, delay)$$
$$O^\star \sqsubseteq O$$
$$O^\star \sqsubseteq \forall r.\neg act(ag1, term)$$
$$act(ag1, term) \sqcap act(ag2, delay) \sqsubseteq (-!!)$$
$$act(ag1, term) \sqcap \neg act(ag2, delay) \sqsubseteq (-!)$$
$$\neg act(ag1, term) \sqcap act(ag2, delay) \sqsubseteq (-)$$
$$\neg act(ag1, term) \sqcap \neg act(ag2, delay) \sqsubseteq (+!!)$$

Reduction to a Satisfiability Test Like in the previous section, we now want to know, if the most desirable outcome $(+!!)$ will come to pass, if ethical code $O^\star$ is operative. We perform this test by checking the satisfiability of the description logic knowledge base $\mathcal{K} = (\mathcal{T}, \mathcal{A})$, with $\mathcal{T}$ as given above and $\mathcal{A}$ given as:

$$\mathcal{A} = \{O^\star(a), \exists r.\neg(+!!)(a)\}$$

If this knowledge base is unsatisfiable, we can conclude, that $(+!!)$ will come to pass, if $O^\star$ is operative. Again we can perform this satisfiability test, by translating the TBox and the ABox into DL-clauses and using Hyper to check the satisfiability. We obtain the desired result, namely that (only) ethical code $O^\star$ leads to the most desirable behavior $(+!!)$.

4.3   Experiments

We formalized the examples introduced in this section and tested it with the Hyper theorem prover as described above. Since all formalizations are available

Table 4. Time in seconds Pellet needed to show the unsatisfiability of the introduced examples. Time in seconds Hyper needed to show the unsatisfiability of the DL-clauses for the examples (the second number includes the translation into DL-clauses).

	Multi-agent Systems	Multi-agent Systems (with TBox)	Contrary-to-duty Obligations
Pellet	2.548	2.468	2.31
Hyper	0.048 / 2.596	0.048 / 2.102	0.03 / 1.749

in $\mathcal{ALC}$, we used the description logic reasoner Pellet [20] to show the unsatisfiability of the formalizations as well. Table 4 shows the results of our experiments. In the first column we see the time in seconds the two reasoners needed to show the unsatisfiability of the formalization of the example from multi-agent systems. For Hyper we give two different numbers. The first number is the time Hyper needs to show the unsatisfiability given the set of DL-clauses. In addition to that the second number contains the time needed to transform the $\mathcal{ALC}$ knowledge base into DL-clauses. The second column gives the runtimes for the example from multi-agent systems using the formalization with a TBox. And in the last column we present the runtimes for the consistency test of normative system $\mathcal{N}_1$ from the example on contrary-to-duty obligations.

For the examples we considered, the runtimes of Pellet and Hyper are comparable. Further investigation and comparison with other modal and/or description logic reasoning tools is required and subject of future work. In order to use Hyper to perform the satisfiability tests, we first have to translate the examples into DL-clauses. Our experiments show, that this translation is not harmful.

5   Conclusion

In this paper, we have demonstrated that by means of deontic logic complex normative systems can be formalized easily. These formalizations can be checked effectively with respect to consistency and independence from additional formulae. For normative systems described with deontic logic, there is a one-to-one translation into description logic formulae. These formula can be checked automatically by automated theorem provers, which is in our case Hyper.

We are aware that deontic logic has several limitations. This is why future work aims at using more recent formalisms. For example we want to apply deontic logic in the context of natural-language question-answering systems. There the normative knowledge in large databases often leads to inconsistencies, which motivates us to consider combinations of deontic with defeasible logic.

References

1. Artosi, A., Cattabriga, P., Governatori, G.: Ked: A deontic theorem prover. In: On Legal Application of Logic Programming, ICLP 1994, pp. 60–76 (1994)

2. Baader, F., Nutt, W.: Basic description logics. In: Baader, F., Calvanese, D., McGuinness, D., Nardi, D., Patel-Schneider, P.F. (eds.) The Description Logic Handbook: Theory, Implementation, and Applications, pp. 43–95. Cambridge University Press (2003)
3. Bassiliades, N., Kontopoulos, E., Governatori, G., Antoniou, G.: A modal defeasible reasoner of deontic logic for the semantic web. Int. J. Semant. Web Inf. Syst. 7(1), 18–43 (2011)
4. Baumgartner, P., Furbach, U., Niemelä, I.: Hyper tableaux. In: Alferes, J.J., Pereira, L.M., Orlowska, E. (eds.) JELIA 1996. LNCS, vol. 1126, pp. 1–17. Springer, Heidelberg (1996)
5. Beirlaen, M.: Tolerating normative conflicts in deontic logic. PhD thesis, Ghent University (2012)
6. Bender, M., Pelzer, B., Schon, C.: System description: E-KRHyper 1.4 - Extensions for unique names and description logic. In: Bonacina, M.P. (ed.) CADE 2013. LNCS (LNAI), vol. 7898, pp. 126–134. Springer, Heidelberg (2013)
7. Bringsjord, S., Arkoudas, K., Bello, P.: Toward a general logicist methodology for engineering ethically correct robots. IEEE Intelligent Systems 21(4), 38–44 (2006)
8. Chisolm, R.M.: Contrary-to-duty imperatives and deontic logic. Analysis 23, 33–36 (1963)
9. Furbach, U., Schon, C.: Deontic logic for human reasoning. CoRR, abs/1404.6974 (2014)
10. Gabbay, D., Horty, J., Parent, X., van der Meyden, R., van der Torre, L. (eds.): Handbook of Deontic Logic and Normative Systems. College Publications (2013)
11. Horty, J.F.: Agency and Deontic Logic. Oxford University Press, Oxford (2001)
12. Klarman, S., Gutiérrez-Basulto, V.: Description logics of context. Journal of Logic and Computation (2013)
13. McNamara, P.: Deontic logic. In: Zalta, E.N. (ed.) The Stanford Encyclopedia of Philosophy. Stanford University (2010)
14. McNamara, P., Prakken, H.: Norms, Logics and Information Systems: New Studies in Deontic Logic and Computer Science. Frontiers in artificial intelligence and applications. IOS Press (1999)
15. Motik, B., Shearer, R., Horrocks, I.: Optimized Reasoning in Description Logics using Hypertableaux. In: Pfenning, F. (ed.) CADE 2007. LNCS (LNAI), vol. 4603, pp. 67–83. Springer, Heidelberg (2007)
16. Murakami, Y.: Utilitarian deontic logic. In: Proceedings of the Fifth International Conference on Advances in Modal Logic (AiML 2004), pp. 288–302 (2004)
17. Plaisted, D.A., Greenbaum, S.: A structure-preserving clause form translation. J. Symb. Comput. 2(3), 293–304 (1986)
18. Schild, K.: A correspondence theory for terminological logics: Preliminary report. In: Proc. of IJCAI 1991, pp. 466–471 (1991)
19. Schmidt, R.A., Hustadt, U.: First-order resolution methods for modal logics. In: Voronkov, A., Weidenbach, C. (eds.) Programming Logics. LNCS, vol. 7797, pp. 345–391. Springer, Heidelberg (2013)
20. Sirin, E., Parsia, B., Grau, B.C., Kalyanpur, A., Katz, Y.: Pellet: A practical OWL-DL reasoner. Web Semantics: Science, Services and Agents on the World Wide Web 5(2), 51–53 (2007)
21. von Kutschera, F.: Einführung in die Logik der Normen, Werte und Entscheidungen. Alber (1973)
22. Pelzer, B., Wernhard, C.: System description: E- KRHyper. In: Pfenning, F. (ed.) CADE 2007. LNCS (LNAI), vol. 4603, pp. 508–513. Springer, Heidelberg (2007)