Translating Simple Legal Text to Formal Representations

Shruti Gaur(✉), Nguyen H. Vo, Kazuaki Kashihara, and Chitta Baral

Arizona State University, Tempe, AZ, USA
{shruti.gaur,nguyen.h.vo,kkashiha,chitta}@asu.edu

Abstract. Various logical representations and frameworks have been proposed for reasoning with legal information. These approaches assume that the legal text has already been translated to the desired formal representation. However, the approaches for translating legal text into formal representations have mostly focused on inferring facts from text or translating it to a single representation. In this work, we use the NL2KR system to translate legal text into a wide variety of formal representations. This will enable the use of existing logical reasoning approaches on legal text (English), thus allowing reasoning with text.

Keywords: Natural language processing · Natural language understanding · Natural language translation

1 Introduction and Motivation

One of the tasks of the Competition on Legal Information Extraction and Entailment [1] consists of finding whether a given statement is entailed by the given legal article(s) or not. This is similar to the Recognition of Textual Entailment (RTE) challenge [3]. It has been observed by Bos and Markert [7] that classification based on shallow features alone performs better than theorem proving, for RTE. Androutsopoulos and Malakasiotis [3] state that most approaches for RTE do not focus on converting natural language to its formal representation. However, we believe that approaches using statistical or machine learning methods on shallow features do not offer much explanation about why a certain sentence is entailed or not, hence providing little insight into the cause of entailment. In this respect, we consider the approaches based on logical reasoning to be more promising.

There have been several works that propose logical representations and logics for representing and reasoning with legal information [11–14,21]. Reasoning rules and frameworks assume that the information given in the form of natural language can somehow be understood and represented in the required form. However, current methods to convert legal text to formal representations [4,8,17,18] either focus on extracting important facts or are not generalizable to a wide variety of representations. Currently, there is no consensus on a single representation to express legal information. Therefore, a system that can translate

T. Murata et al. (Eds.): JSAI-isAI 2014 Workshops, LNAI 9067, pp. 259–273, 2015.
DOI: 10.1007/978-3-662-48119-6_19

natural language to a wide variety of formal languages, depending on the application, is desired. In this paper we show how our NL2KR system can be used for translation of simple legal sentences in English to various formal representations. This will facilitate reasoning with various frameworks.

2 Related Work

Some approaches to translate text into formal representations focus on extraction of specific facts from the text. For example, Lagos et al. [17] present a semi-automatic method to extract specific information such as events, characters, roles, etc. from legal text by using the Xerox Incremental Parser (XIP) [2]. The XIP performs preprocessing, named entity extraction, chunking and dependency extraction, and combination of dependencies to create new ones. Bajwa et al. [4] propose an approach to automatically translate specification of business rules in English to Semantic Business Vocabulary and Rules (SBVR). Their method is essentially a rule-based information-extraction approach, which identifies SBVR elements from text. The goal of other approaches like the work by McCarty [18] is to obtain a semantic interpretation of the complete sentence. This approach uses the output from the state-of-the-art statistical parser to obtain a semantic representation called Quasi-Logical Form (QLF). QLF is a rich knowledge representation structure which is considered an intermediate step towards a fully logical form.

There have been similar efforts in other languages. Nakamura et al. [20] present a rule-based approach to convert Japanese legal text into a logical representation conforming to Davidsonian style. They ascertain the structure of legal sentences and identify cue phrases that indicate this structure, by manually analyzing around 500 sentences. They also define transformation rules for some special occurrences of nouns and verbs. In their subsequent work [16], they propose a method to resolve references that point to other articles or itemized lists, by replacing them with the relevant content.

As mentioned in the previous section, different legal reasoning frameworks expect input in different logical representations. Even though Legal Knowledge Interchange Format (LKIF) [15] was an attempt to standardize the representation of legal knowledge in the semantic web, currently, no single representation has been unanimously considered the de-facto standard for legal text. Therefore, we need a system that can translate natural language to a particular representation depending on the application.

3 The NL2KR Framework

NL2KR is a framework to develop translation systems that translate natural language to a wide variety of formal language representations. It is easily adaptable to new domains according to the training data supplied. It is based on the algorithms presented in Baral et al. [5]. The workflow using the NL2KR systems consists of two phases: (1.) learning and (2.) translation, as shown in Fig. 1.

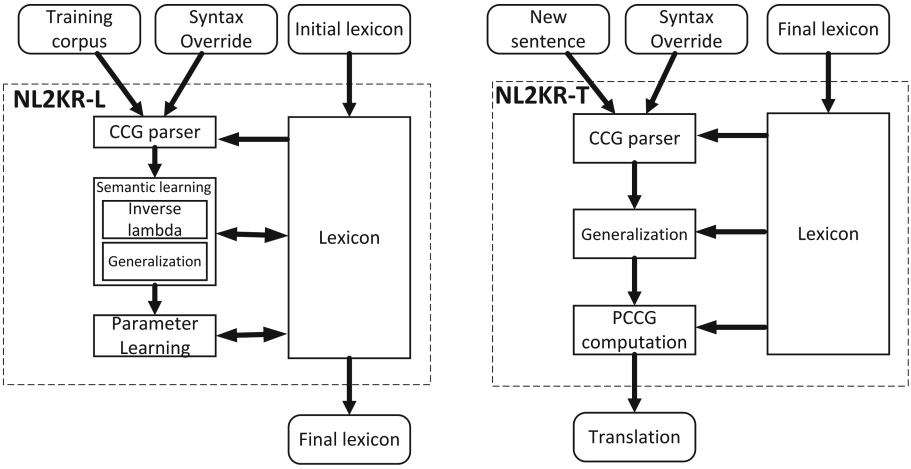

Fig. 1. The NL2KR system showing learning (left) and translation (right)

In the learning phase, the system takes training data, an initial dictionary and any optional syntax overrides, as inputs. The training data consists of a number of natural language sentences along with their formal representations in the desired target language. The initial dictionary (or lexicon) contains meanings of some words. The dictionary is manually supplied to the system. Using these inputs, NL2KR tries to learn the meanings of as many words as possible. Thus, the output of the learning phase is an updated dictionary which includes the meanings of all newly learned words. The translation phase uses the dictionary created by the learning phase to translate previously unseen sentences.

At the core of NL2KR are two very elegant algorithms, Inverse Lambda and Generalization, which are used to find meanings of unknown words in terms of lambda (λ) expressions. NL2KR is inspired[1] by Montague's approach [19]. Every word has a λ expression meaning. The meaning of a sentence is successively built from the combination of the λ expressions of words according to the rules of combination in Lambda (λ) calculus [9]. The order in which the words should be combined is given by the parse tree of the sentence according to a given Combinatory Categorial Grammar (CCG) [22]. As an example illustrating this approach, consider the sentence "John loves Mary" shown in Table 1. The CCG category of "loves" is (S\NP)/NP. This means that this word

Table 1. Example

John	loves	Mary
NP	$(S\backslash NP)/NP$	NP
$john$	$\#y.\#x.loves(x,y)$	$mary$
	$S\backslash NP$	
	$\#x.loves(x,mary)$	
	S	
	$loves(john,mary)$	

[1] NL2KR cannot be said to be based on Montague Semantics as it does not use intensional semantics. The translation of natural language to formal language with the use of lambda calculus, however, is in the same spirit as Montague's approach.

takes arguments of type NP (noun-phrase) from the left and the right, to form a complete sentence. From the CCG parse, we observe that "loves" and "Mary" combine first and then their combination combines with "John" to form a complete parse. The λ expression corresponding to "loves" is $\#y.\#x.loves(x,y)$[2], which means that this word takes two inputs, $\#x$ and $\#y$ as arguments and the application of this word to the arguments results in a λ expression of the form $loves(x,y)$.

The close correspondence between CCG syntax and λ calculus semantics is very helpful in applying this method. In the first step, the λ expression for "loves" is applied to "Mary", with the former as the function and the latter as the argument, in accordance with CCG categories. This application, denoted as $\#y.\#x.loves(x,y)@mary$ results in $\#x.loves(x,mary)$. Proceeding this way, the meaning of the sentence is generated in terms of λ expressions. This is a very elegant way to model semantics and has been widely used [5,6,10,23]. The problem, however, is that for longer sentences, λ expressions become too complex for even humans to figure out. This problem is addressed by NL2KR by employing the Inverse Lambda and Generalization algorithms to automatically formulate λ expressions from words whose semantics are known.

Learning Algorithms: The two algorithms used to learn λ semantics of new words are the Inverse Lambda and Generalization algorithms. When the λ expressions of a phrase and that of one of its sub-parts (children in the CCG parse tree) are known, we can use this knowledge to find the λ expression of the unknown sub-part. The Inverse Lambda operation computes a λ expression F such that $H = F@G$ or $H = G@F$ given H and G. These are called Inverse-L and Inverse-R algorithms, respectively. For example, if we know the meaning of the sentence "John loves Mary" (Table 1) as $loves(john,mary)$ and the meaning of John as $john$, we can find the meaning of "loves Mary" using Inverse Lambda, as $\#x.loves(x,mary)$. Going further, if we know the meaning of "Mary" as $mary$, we can find the meaning of "loves" using Inverse Lambda.

The Generalization algorithm is used to learn meanings of unknown words from syntactically similar words with known meanings. It is used when Inverse Lambda algorithms alone are not enough to learn new meanings of words or when we need to learn meanings of words that are not even present in the training data set. For example, we can generalize the meaning of the word "likes" with CCG category $(S\backslash NP)/NP)$, from the meaning of "loves", which we already know from the previous example. The meaning of "likes" thus generated will be $\#y.\#x.likes(x,y)$. We will illustrate learning in later sections with the help of examples.

For every sentence in the training set, we first use the CCG Parser to obtain all possible parse trees. Using the initial dictionary supplied by the user, the system assigns all known meanings to the words (at the leaf) in each parse tree. Moving bottom up, it combines as many words with each other as possible(in the

[2] # is used in place of λ to enable typing into a terminal.

order dictated by the parse tree) by performing λ applications. The meaning of each complete sentence is known from the training corpus. We need to traverse top-down from this known translation, while simultaneously traversing bottom up, by filling in missing word or phrase meanings. Meanings of unknown words and phrases are obtained using Inverse Lambda and Generalization, as applicable, until nothing new can be learned.

Dealing with Ambiguity: To deal with ambiguity of words, a parameter learning method [23] is used to estimate a weight for each word-meaning pair such that the joint probability of the training sentences getting translated to their given formal representation is maximized. However, this method might not work in all cases and more complex approaches, possibly involving word sense identification from context, might have to be used. Completely addressing this problem is a part of future work.

Translation Approach: Given a sentence, we consider all the possible parse trees, consisting of meanings of every word learned by the system or obtained from Generalization algorithm. Then we use Probabilistic CCG (PCCG) [23] to find the most probable tree, according to weights assigned to each word.

Availability: NL2KR is freely available for Windows, Linux and MacOSX systems at http://nl2kr.engineering.asu.edu. It is configurable for different domains and can be adapted to work with a large number of formal representations. A tutorial has also been provided.

4 Translating to Formal Legal Representations

NL2KR can be used to translate sentences into various logical representations, either directly or by using an intermediate language[3]. It can be customized to different domains based on the initial dictionary and training data provided. The quality of these inputs affects NL2KR's performance. A language class can be considered a good analogy of NL2KR. The effectiveness of learning depends on the richness of vocabulary imparted to the students beforehand (similar to initial lexicon) and the sentences chosen to teach the language (training data). In our experiments, we observed that learning simpler sentences before complex ones aided learning. We will also give some guidelines for creating the initial dictionary. Several logics have been proposed in the literature for representing legal information [11–14,21], from which we have selected a few. In this section, we will illustrate the method of creating a good initial lexicon and demonstrate how to use the system to learn new word meanings, with respect to these examples. We will start with simple examples and progress to more complicated ones.

[3] The choice of intermediate language depends on the domain and target languages. Once an intermediate language has been decided, the conversion can be automated.

4.1 Translating to First Order Logic Representations

In this section, we demonstrate translating a sentence from the Competition on Legal Information Extraction and Entailment [1] corpus to a first order logic representation.

Sentence: Possessory rights may be acquired by an agent.
Translation: $rights(X) \wedge type(X, possessory) \wedge agent(Y) > acquirable(X, Y, may)$

Here $>$ is used to denote implication. The form of an action, for e.g., "acquirable" is $action(X, Y, Z)$. It denotes X(possessory rights) is being acquired by Y(agent) and the type of this action is Z. In the given example, *acquiring* is a possibility, not an obligation, which is why we use *may* as its type.

Once we provide this training data and other required inputs to the NL2KR learning interface (Fig. 2), we can start the Learning process. We will describe how to create inputs for learning in the next sub-section. The initial dictionary contains a list of words and their meanings in terms of λ expressions. Even if we do not know meanings of some words, we can use the system to figure them out on its own, using Inverse Lambda or Generalization algorithms. Figure 2 shows that the system learns the meaning of "rights" automatically using Inverse Lambda.

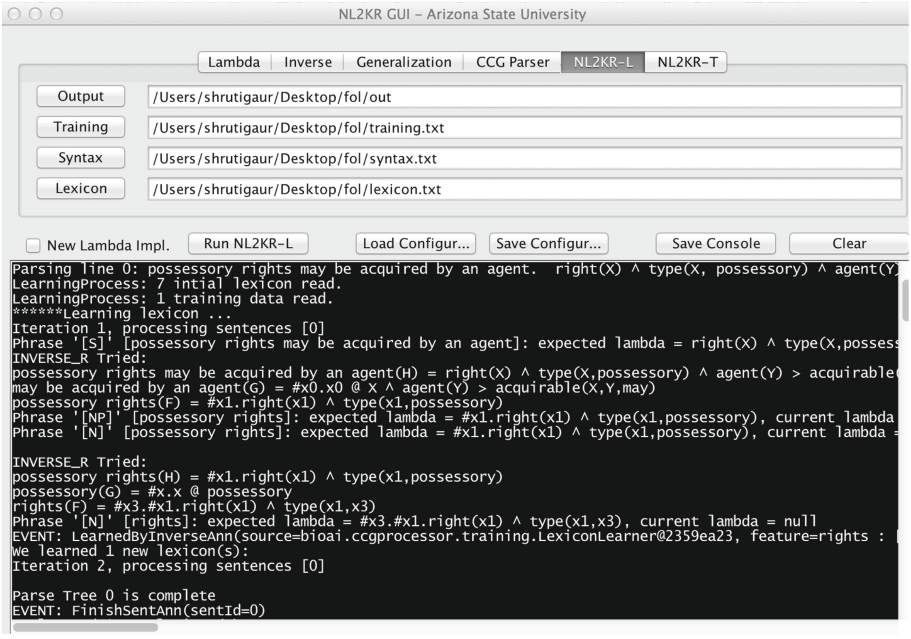

Fig. 2. NL2KR automatically learning the meaning of "rights" using Inverse Lambda Algorithm : feature = rights : [N] : #x3.#x1.right(x1) ∧ type(x1,x3)

4.2 Translating Sentences with Temporal Information

Consider the following sentence and translation, which shows an example of temporal ordering.

Sentence: After the invoice is received the customer is obliged to pay.
Translation: $implies(receipt(invoice, T1) \wedge (T2 > T1),$
$obl(pay(customer, T2)))$

Here $implies(x, y)$ denotes $x \rightarrow y$. The predicate *obl* denotes that the action is an *obligation* (usually marked by words such as obliged to, shall, must, etc.) in contrast to a *possibility* (usually marked by words such as may). T1 and T2 are the instances of time at which the two events occurred.

Words that do not contribute significantly to the meaning of the sentence can be assigned the trivial meaning $\#x.x$ in the dictionary. It is a λ expression that does not affect the meaning of other λ expressions. We can assign it to words such as "is", "to" and "the" since these do not carry much meaning in this example. Next, we can start entering the meanings that are evident from looking at the target representation. Since "invoice" occurs as itself, we can give it the simple meaning *invoice* (similarly for "customer"). From the representation, we observe that "received" is a function called *receipt* with two arguments, hence we can give it the meaning $\#x.\#t.receipt(x, t)$ (similarly for "pay"). *Obliged* is a more complicated function because it takes another function (pay) as its argument and therefore uses @y@t to carry forward the variables in *pay* to the next higher level of the tree, where we obtain the real arguments (customer and T2). Once all these meanings (Table 2) have been supplied, the system can automatically find the meaning of the word "after" using Inverse Lambda algorithm (Fig. 3). This is remarkable from the perspective that the meaning of "after" looks complicated and it might be tedious for users to supply such meanings manually in the initial lexicon. This demonstrates one of the advantages of using NL2KR. The meaning of "after" makes intuitive sense. The λ expression #x12.#x11.implies(x12 @ T1 $\wedge$ T2 >T1,x11 @ T2) means that "after" is a λ function which takes two inputs:x11 and x12, where the first input event (x12) occurs at time T1, T2 >T1, the second input event (x11) occurs at time T2 and x12 implies (or leads to) x11. Hence, we were able to learn a significantly complicated meaning automatically by providing relatively simple λ expressions in the initial dictionary.

4.3 Translating to Temporal Deontic Action Laws

Giordano et al. [14] have defined a Temporal Deontic Action Language for defining temporal deontic action theories, by introducing a temporal deontic extension of Answer Set Programming (ASP) combined with Deontic Dynamic Linear Time Temporal Logic (DDLTL). This language is used for expressing domain description laws, for e.g., action laws, precondition laws, causal laws, etc., which describe the preconditions and effects of actions. It is also used for expressing obligations, for e.g., achievement obligations, maintenance obligations, contrary

Table 2. λ expressions and CCG categories in the initial dictionary for the sentence "After the invoice is received the customer is obliged to pay."

Word	Syntax	Meaning
invoice	N	invoice
is	(S\NP)/NP	#x.x
received	NP	#x.#t.receipt(x,t)
customer	N	customer
obliged	NP/NP	#x.#y.#t.obl(x@y@t)
to	NP/(S\NP)	#x.x
pay	S\NP	#x.#t.pay(x,t)
the	NP/N	#x.x

to duty obligations, etc. We will take examples of several domain description laws from the paper and demonstrate how to translate them automatically from natural language to the Deontic action language, using NL2KR.

Since NL2KR does not support some special symbols used in the Temporal Deontic Action Language, we first use NL2KR to convert the natural language sentences to an intermediate representation which is directly convertible to the Temporal Deontic Action Language. Then using the one-to-one correspondence between the intermediate language and the action language, we obtain the desired representation. In the Intermediate representation shown below, we have defined the predicate $creates(x, y)$, which means x creates y. We have also changed the representation of *until* to have two parameters a and b denoting "a until b" (as defined by Giordano et al. [14]).

Action Law:
Sentence: The action *accept_price* creates an obligation to pay.
Translation: $[accept\_price]\mathbf{O}(\top U < pay > \top)$
Intermediate: creates(action(accept_price),O(until(a(T),b(pay,T))))

The NL2KR learning component can be used to make the system learn words and meanings from this sentence. The iterative learning process is depicted in the screenshot in Fig. 4. We start by giving meanings of simple words first. We give trivial meanings $\#x.x$ to "the","an" and "to", because they do not significantly affect the meaning of the sentence. Next, we guess the meanings of words from the target representation. Since "action" is a function that accepts a single argument, we give it the meaning $\#x.action(x)$. Similarly, "accept_price" which occurs as itself is given the meaning *accept_price*. Similarly, Obligation, O is also a function, but it contains more structure, which can be obtained from the target representation. We interpret an obligation to also have an implicit notion of time by having "until" embedded in its meaning. However, the verb "pay" should be replaceable, because there can be other sentences such as "The action *accept_price* creates an obligation to ship". Therefore, we leave it as a

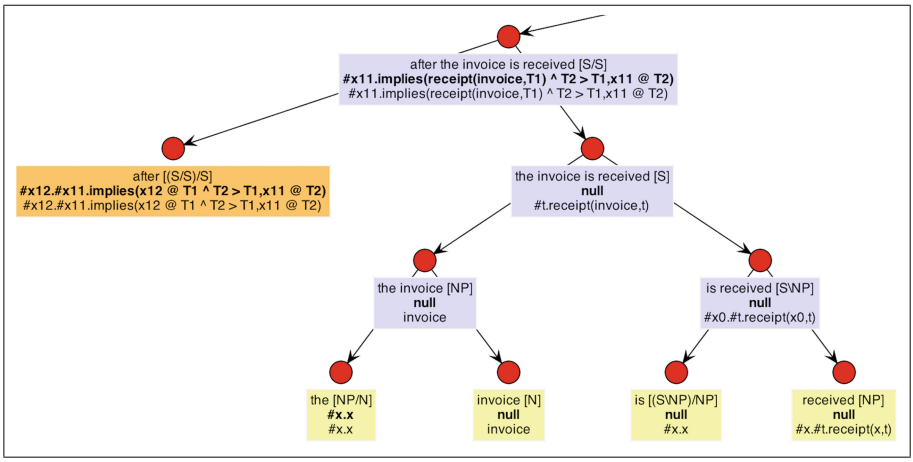

Fig. 3. NL2KR automatically learning the meaning of "after" using Inverse Lambda Algorithm : feature = after : [(S/S)/S] : #x12.#x11.implies(x12 @ T1 ∧ T2 >T1,x11 @ T2)

variable input. The word "pay" could have been simply *pay* but we assign it the meaning $\#x.x@pay$. This is because the node, "an obligation", expects "to pay" to be an argument to it, but their CCG categories dictate otherwise. In cases where there is such inconsistency, we use meanings prefixed with $\#x.x@$ for the function (according to CCG categories), so that their role is *flipped* to that of arguments[4]. The λ expressions and CCG categories of the constituent words are shown in Table 3. After giving these meanings, we find that the meaning of "creates" is obtained automatically by the system using the Inverse Lambda algorithm (Fig. 4).

Once the learning process is complete, we can use the Translation component of NL2KR to translate a new sentence. In this case, we use NL2KR to translate the following action law.

Action Law:
Sentence: The action *cancel_payment* cancels the obligation to pay.
Translation: $[cancel\_payment]\neg\mathbf{O}(\top U < pay > \top)$
Intermediate: cancels(action(cancel_payment),O(until(a(T),b(pay,T))))

The screenshot of the translation process is shown in Fig. 5. We observe that the sentence was automatically translated by the system successfully. This was done by generating the meanings of unknown words ("cancel_payment"

[4] Let the required function be A and the required argument be B. Let the CCG-determined function be B and the CCG-determined argument be A. Recall that @ denotes λ application. By giving a meaning of the form $\#x.(x@b)$ to B, and performing application as determined by CCG, we obtain the result as $(\#x.(x@b))@a$ or $a@b$.

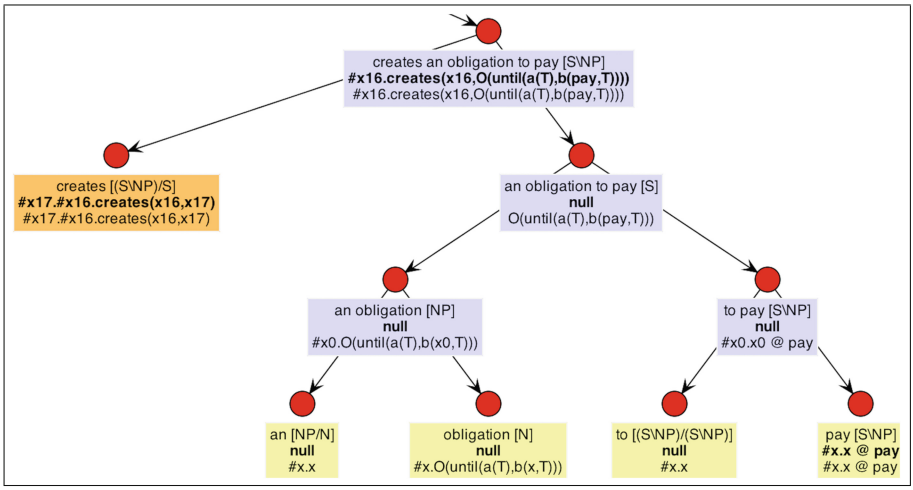

Fig. 4. Screenshot of the Learning Process in NL2KR for the sentence "The action *accept_price* creates an obligation to pay." The meaning of "creates" is obtained automatically by the system using the Inverse Lambda algorithm

and "cancels") using Generalization (Fig. 6) on the words learned from the first action law.

4.4 Translating to Temporal Object Logic in REALM

Regulations Expressed as Logical Models (REALM) [13] is a system that models regulatory rules in temporal object logic. The concepts and relationships occurring in this rule are mapped to predefined types and relationships in a Unified Modeling Language (UML) model. Using some examples from this paper, we will show how NL2KR can be used to translate rules specified in natural language to this temporal object logic representation.

Table 3. λ expressions and CCG categories for the words in the action law "The action *accept_price* creates an obligation to pay."

Word	Syntax	Meaning
the	NP/N	#x.x
action	N/N	#x.action(x)
accept_price	N	accept_price
an	NP/N	#x.x
obligation	N	#x.O(until(a(T),b(x,T)))
to	(S\NP)/(S\NP)	#x.x
pay	S\NP	#x.x@pay

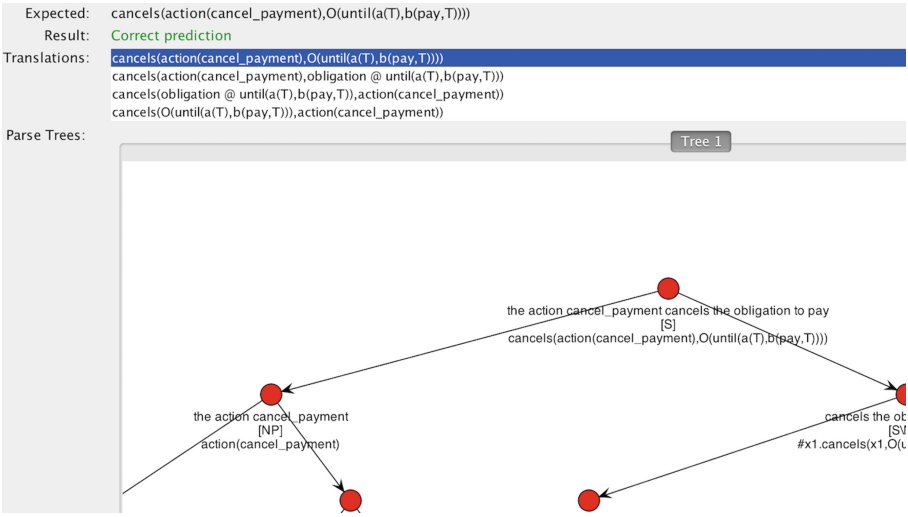

Fig. 5. Screenshot of the Translation Process in NL2KR for the sentence "The action *cancel_payment* cancels the obligation to pay."

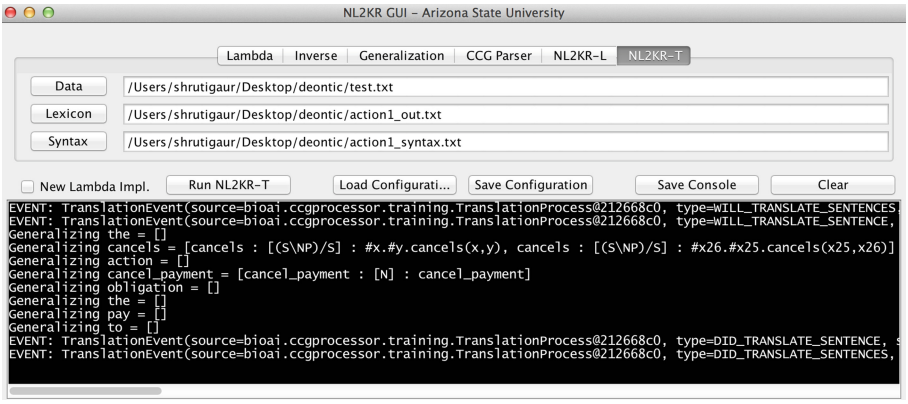

Fig. 6. Generating the meanings of unknown words (cancel_payment and cancels) using Generalization during the Translation Process in NL2KR for the sentence "The action *cancel_payment* cancels the obligation to pay."

As in the previous section, we have created an intermediate representation which can directly be converted to the desired temporal object logic representation. This is needed due to unavailability of certain symbols in NL2KR's vocabulary. We also assume that coreference in sentences has been resolved. For example, in the following sentence, the second occurrence of "bank" has replaced the pronoun "it".

Sentence: Whenever a bank opens an account *bank* must verify customers identity within two_days
Translation: $\Box t_{open}(DoOn_F(bank, open, a) \rightarrow$
$\Diamond t_{verify}(DoInput_F(bank, verify, a.customer.record) \wedge t_{verify} - t_{open} \leq 2_{[day]})$
Intermediate: $implies(g(do(bank, open, a, T1)),$
$f(do(bank, verify, a\_customer\_record, T2)$
$\wedge equals(difference(T2, T1), two\_days)))$

Similar to the previous examples, we use NL2KR to learn unknown words from these sentences. We do not give the meaning of "verify" for the second sentence (which is different from its meaning in the first sentence) but the system is able to figure it out on its own. Moreover, it also generalizes the correct meaning of three_days using the meaning of two_days from the previous sentence.

Sentence: Whenever a bank can not verify an identity bank has to close the account within three_days

Table 4. Initial Lexicon containing λ expressions and CCG categories for both REALM examples

Word[Syntax]	Meaning
whenever [(S/S)/S]	#y.#x.implies(g(y@T1),f((x@T1)@T2))
a [NP/N]	#x.x
bank [N]	bank
opens [(S\NP)/NP]	#y.#x.#t1.do(x,open,y,t1)
an [NP/N]	#x.x
account [N]	a
must [(S\NP)/(S\NP)]	#x.x
verify [(S\NP)/NP]	#x.x@#x1.#x2.#x3.#x4.#x5.(do(x3,verify,x1,x5)
	$\wedge$ x2@x4@x5)
customers [NP/N]	#x.x
identity [N]	a_customer_record
within [(NP\NP)/NP]	#z.#y.#x.x@y@#t1.#t2.equals(difference(t2,t1),z)
has [(S\NP)/(S\NP)]	#x.x
to [(S\NP)/(S\NP)]	#x.x
the [NP/N]	#x.x
can [(S\NP)/(S\NP)]	#x.x
close [(S\NP)/NP]	#x.x@#x1.#x2.#x3.#x4.#x5.(do(x3,close,x1,x5)
	$\wedge$ x2@x4@x5)
not [(S\NP)/(S\NP)]	#y.#x.#t1.(y @ x @ t1 $\wedge$ isfalse)
two_days [N]	two_days

Translation: $\Box t_{open}(DoOn_F(bank, open, a) \rightarrow$
$\Diamond t_{verify}(DoInput_F(bank, verify, a.customer.record) \wedge t_{verify} - t_{open} \leq 2_{[day]})$
Intermediate: $implies(g(do(bank, verify, a\_customer\_record, T1) \wedge isfalse),$
$f(do(bank, close, a, T2) \wedge equals(difference(T2, T1), three\_days)))$

We observe that the initial dictionary for this case (Table 4) looks more complicated than the one in Sect. 4.3. This is because the target language in this case is such that the functions which would have been intuitive according to their natural language meanings, for e.g., "opens", "verify", etc. are not functions but arguments of an artificially created function, "do". It is obvious that the language of REALM was designed for different purposes than that of translation, which is why such a situation exists. Our motivation here is to give the reader an explanation of why some languages are easy for NL2KR to translate, while others are more difficult.

5 Conclusion and Future Work

Although legal text is written in natural language, one needs to do some kind of formal reasoning with it to draw conclusions. The first step to do that is to translate legal text to an appropriate logical language. At present there is no consensus on a single logical language to represent legal text. Therefore, one cannot develop a translation system targeted to a single language. Thus, a platform that can translate legal text to the desired logical language depending on the application, is needed. We have developed such a system called NL2KR. In this paper, we showed how NL2KR is useful in translating sentences from legal texts in English to various formal representations defined in various works, thereby bridging the gap from language to logical representation and enabling the use of various logical frameworks over the information contained in such texts.

So far we have experimented with a few small sentences picked from the literature on logical representation of legal texts. However, we need to expand this approach to capture nuances of legal texts used in real laws and statutes. Further enhancements are needed in NL2KR to equip it to deal with longer and more complicated sentences. One approach that can be used would involve breaking the sentence into smaller parts and subsequently dealing with each part separately. Such a parser, called L-Parser is available at http://bioai8core.fulton.asu.edu/lparser. We also plan to combine statistical and logical methods in the future. In particular, we are considering using a combination of distributional semantics and hand curated linguistic knowledge to characterize content words (especially, noun, verbs and adjectives) and use logical characterization for grammatical words (prepositions, articles, quantifiers, negation, etc.).

Acknowledgements. We thank Arindam Mitra and Somak Aditya for their work in developing the L-Parser. We thank NSF for the DataNet Federation Consortium grant OCI-0940841 and ONR for their grant N00014-13-1-0334 for partially supporting the development of NL2KR.

References

1. Jurisin legal information extraction and entailment competition (2014). http://webdocs.cs.ualberta.ca/miyoung2/jurisin_task/index.html
2. Aït-Mokhtar, S., Chanod, J.P., Roux, C.: Robustness beyond shallowness: incremental deep parsing. Nat. Lang. Eng. **8**(3), 121–144 (2002)
3. Androutsopoulos, I., Malakasiotis, P.: A survey of paraphrasing and textual entailment methods. J. Artif. Int. Res. **38**(1), 135–187 (2010)
4. Bajwa, I.B., Behzad, L.M.: SBVR business rules generation from natural language specification. In: AAAI 2011 Spring Symposium AI for Business Agility, San Francisco, USA, pp. 2–8 (2011)
5. Baral, C., Dzifcak, J., Gonzalez, M.A., Zhou, J.: Using Inverse lambda and Generalization to Translate English to Formal Languages. CoRR abs/1108.3843 (2011)
6. Blackburn, P., Bos, J.: Representation and Inference for Natural Language: A First Course in Computational Semantics. Center for the Study of Language and Information, Stanford (2005)
7. Bos, J., Markert, K.: Recognising textual entailment with logical inference. In: Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing, HLT 2005, pp. 628–635. Association for Computational Linguistics, Stroudsburg (2005)
8. Brüninghaus, S., Ashley, K.D.: Improving the representation of legal case texts with information extraction methods. In: Proceedings of the 8th International Conference on Artificial Intelligence and Law, ICAIL 2001, pp. 42–51. ACM, New York (2001)
9. Church, A.: An unsolvable problem of elementary number theory. Am. J. Math. **58**(2), 345–363 (1936)
10. Costantini, S., Paolucci, A.: Towards translating natural language sentences into ASP. In: Faber, W., Leone, N. (eds.) CILC, CEUR Workshop Proceedings, vol. 598. CEUR-WS.org (2010)
11. De Vos, M., Padget, J., Satoh, K.: Legal modelling and reasoning using institutions. In: Bekki, D. (ed.) JSAI-isAI 2010. LNCS, vol. 6797, pp. 129–140. Springer, Heidelberg (2011)
12. Distinto, I., Guarino, N., Masolo, C.: A well-founded ontological framework for modeling personal income tax. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Law, ICAIL 2013, pp. 33–42. ACM, New York (2013)
13. Giblin, C., Liu, A.Y., Müller, S., Pfitzmann, B., Zhou, X.: Regulations expressed as logical models (REALM). In: Proceedings of the 2005 Conference on Legal Knowledge and Information Systems, JURIX 2005, The Eighteenth Annual Conference, pp. 37–48. IOS Press, Amsterdam (2005)
14. Giordano, L., Martelli, A., Dupré, D.T.: Temporal deontic action logic for the verification of compliance to norms in ASP. In: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Law, ICAIL 2013, pp. 53–62. ACM, New York (2013)
15. Hoekstra, R., Breuker, J., Bello, M.D., Boer, E.: The LKIF core ontology of basic legal concepts. In: Proceedings of the Workshop on Legal Ontologies and Artificial Intelligence Techniques, LOAIT 2007 (2007)
16. Kimura, Y., Nakamura, M., Shimazu, A.: Treatment of legal sentences including itemized and referential expressions – towards translation into logical forms. In: Hattori, H., Kawamura, T., Idé, T., Yokoo, M., Murakami, Y. (eds.) JSAI 2008. LNCS, vol. 5447, pp. 242–253. Springer, Heidelberg (2009)

17. Lagos, N., Segond, F., Castellani, S., O'Neill, J.: Event extraction for legal case building and reasoning. In: Shi, Z., Vadera, S., Aamodt, A., Leake, D. (eds.) IIP 2010. IFIP AICT, vol. 340, pp. 92–101. Springer, Heidelberg (2010)
18. McCarty, L.T.: Deep semantic interpretations of legal texts. In: Proceedings of the 11th International Conference on Artificial Intelligence and Law, ICAIL 2007, pp. 217–224. ACM, New York (2007)
19. Montague, R.: English as a formal language. In: Thomason, R.H. (ed.) Formal Philosophy: Selected Papers of Richard Montague, pp. 188–222. Yale University Press, New Haven (1974)
20. Nakamura, M., Nobuoka, S., Shimazu, A.: Towards translation of legal sentences into logical forms. In: Satoh, K., Inokuchi, A., Nagao, K., Kawamura, T. (eds.) JSAI 2007. LNCS (LNAI), vol. 4914, pp. 349–362. Springer, Heidelberg (2008)
21. Riveret, R., Rotolo, A., Contissa, G., Sartor, G., Vasconcelos, W.: Temporal accommodation of legal argumentation. In: Proceedings of the 13th International Conference on Artificial Intelligence and Law, ICAIL 2011, pp. 71–80. ACM, New York (2011)
22. Steedman, M.: The Syntactic Process. MIT Press, Cambridge (2000)
23. Zettlemoyer, L.S., Collins, M.: Learning to map sentences to logical form: structured classification with probabilistic categorial grammars. In: UAI, pp. 658–666. AUAI Press (2005)