Towards Formalizing Statute Law as Default Logic through Automatic Semantic Parsing

Marcos Pertierra
MIT, CSAIL
marcosp@mit.edu

Sarah Lawsky
Northwestern Pritzker School of Law
sarah.lawsky@law.northwestern.edu

Erik Hemberg
MIT, CSAIL
hembergerik@csail.mit.edu

Una-May O'Reilly
MIT, CSAIL
unamay@csail.mit.edu

ABSTRACT

Tax regulations and statutes are long, complex, and difficult to understand, and thus present the opportunity for undetectable legal avoidance. Our project goal is to facilitate a new approach to statute composition wherein a logic representation of existing law would be extended and checked *before* its translation to natural language. We envision a software pipeline that would automatically parse a requested section of the Internal Revenue Code (IRC) and accurately express it with a *default* logic representation. Herein, we evaluate the effectiveness of an end to end assembly of existing software tools. This pipeline uses regular expression search on the Code's common structural text patterns and conducts semantic parsing with various open-source natural language parsers. Using IRC Section 163(h) which we have manually expressed in default logic, we evaluate the resulting intermediate logic representations. We observe that the semantic complexity of tax regulations overwhelms the parsers' capabilities. Their shortcomings will have to be addressed as a prerequisite to a component that will, starting from the intermediate logic, automatically express the default logic.

KEYWORDS

tax, semantics, parsing, default logic

1 INTRODUCTION

United States tax law, as represented in the Internal Revenue Code (IRC) and its accompanying regulations, is notoriously complicated. This complexity increases the cost of tax compliance. Even more alarming, both individuals and corporations take advantage of the law's complexity to reduce their taxes by engaging in *legal avoidance*. Legal avoidance describes actions that are technically legal but which do not fall within legislators' intentions. Fundamentally we are interested in the way in which tax law fails to prevent legal avoidance. Our central research question is how formulations of statutes and regulations can be improved to reduce legal avoidance.

Tax avoidance occurs because taxpayers are able exploit ambiguities within the law or take advantage of disparate legal treatment of similar concepts (i.e., engage in regulatory arbitrage) to reduce their tax owed in a way that is legal but is unintended by the law. Because it is almost impossible to foresee avoidance by manually examining tax regulations and statutes, multiple law and technology projects have modeled the law and used artificial intelligence to enlist computers for automatic reasoning, see [1, 17, 19, 24, 25]. Approaches vary in whether they manually or automatically interpret the meaning of the IRC text and whether they resort to an ad hoc representation of it or different logical formalisms. Modeling tax law is challenging because tax law is represented in the IRC with natural language. Natural language is obviously useful for humans to read and interpret. However, it is very difficult for computers. Thus, a critical step is to convert the law from its natural language representation to some formal representation that a computer can read and use for inference, i.e. *semantically parse* it.

Our project proposes to: (1) **Employ a version of default logic (DL) to represent a portion of tax law relevant to the study of avoidance.** This encompasses the work of Lawsky et. al. [14], which advocates formalizing tax law, and philosophical logic [7] presenting a version of DL that may be particularly well suited to formalizing the tax law. (2) **Provide a software system that collaboratively helps a legal expert to reason about legal logic in this new formalism.** We envision designing software that translates and represents relevant sections of the IRC or proposed new statutes as DL to whatever extent that is feasible. In infeasible circumstances, the system teams up with its expert to accomplish tasks more efficiently and with improved ease. The expert is expected to query the logic system they have set up to determine whether avoidance is possible under its meaning.

We defer our rationale and description of our DL to prior contributions [14, 15]. Here, we focus on automation technology. To date we have conceptually designed an end-to-end `IRC-to-default-logic`[1] pipeline that translates IRC code in XML into supernormal DL, see Figure 1. We have interfacing components which allow us to focus on two stages independently. The goal of the first stage is to automatically parse the text of the relevant regulations into an intermediate logic representation. In this contribution we report our progress in implementing the first stage. The aim of the second stage is to accurately transform the intermediate logic representation to background theory and the ordered default rules that are supported by a theorem prover, allowing queries and propositions to be tested around taxation concepts such as deductibility. To date, we have assembled a solver and set it up to handle simple examples of our DL, but have deferred the most challenging task of accurate transformation.

In: Proceedings of the Second Workshop on Automated Semantic Analysis of Information in Legal Text (ASAIL 2017), June 16, 2017, London, UK.

Published at http://ceur-ws.org

[1] https://github.com/mpertierra/irc_to_default_logic

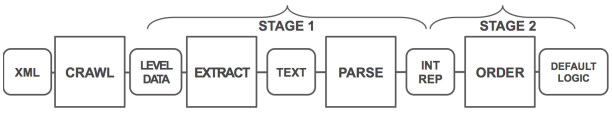


Figure 1: Project pipeline, script: `pipeline.py`.

For the first stage, parsing the regulations, we have taken two specific strategies: (1) We exploit the style guidelines for drafting legislation [18] to extract definitions and rules from the IRC text by pattern matching with *regular expressions*, a relatively simple and a semantically superficial approach. (2) We leverage existing natural language processing semantic parsers to extract formal representations from text. These currently expect relatively simple input sentences. As a result, we must evaluate their capacity. Our evaluation is on the definitions, rules and DL representation of elements of Section 163(a), previously identified by an expert [15].

The contributions of this paper are: (1) The introduction of our project with its goals and approach, encompassing a version of DL supporting the expression of tax statues relevant to our focus on legal avoidance. (2) The introduction of `IRC-to-default-logic`, open source software that automates our approach. (3) A demonstration and evaluation of `IRC-to-default-logic`. This has revealed the merits and open issues arising from stepping off with existing software tools.

In § 2, we discuss related work. In § 3, we present our method. In § 4 we present the results. In § 5 we conclude.

2 RELATED WORK

To express natural language completely and precisely in a formal representation upon which a computer can meaningfully act, we need to capture its semantics. This can be partially accomplished by pattern matching and by semantic parsing. We also face the question of what formalism is best suited as a target output representation. We review: § 2.1 text extraction and ad hoc representations. § 2.2 formalisms for representing law, as well as the version of DL we use. § 2.3 relevant existing semantic parsers.

2.1 Extraction & Ad Hoc Representation

Previous work has focused on pattern-based rule extraction from law, see e.g. [25]. These efforts have typically focused on extracting higher level elements from text, such as exception phrases, rather than translating to a formal representation. A tax avoidance project named Stealth does manual formalization and translation [11] and uses an ad hoc rule-based representation that supports tax calculations. The Tax Knowledge Adventure [1] ontology reuses the WordNet and LKIF-Core ontologies for a set of terms extracted from in the "open-text" from IRC and Tax resources. IRC sections 301, 302, 317 are represented as concepts in the ontology. Rules are "too complicated" for OWL assertions; instead, rules are class member functions in an object oriented programming language.

2.2 Formal Represention of Statute Law

Standard formal logic is not the best representation to accommodate statutory reasoning. One better choice is defeasible reasoning, i.e. reasoning that may result in conclusions that can be defeated by subsequent information [14]. This reasoning is modeled by DL in [12], a non-monotonic logic. A metarule is required in the DL system to indicate how to reason about apparently conflicting statutory rules. The DL formalization fits with the IRC structure making it easier to accurately express the statutory meaning. Much of this meaning can be found by paying attention to the level-based style or structure of the IRC, e.g. general rules are followed by exceptions.[2] There may be a variety of different interpretations of law, depending on the precise question one is asking. DL's formalization provides appropriately different answers by depending on the priority the formalizer gives to the various rules.

There have been various attempts to formalize legal text, whether that be via some programmable representation, an ontological representation, or some other semi-formal representation that is not tied to any implementation. For example, Sergot et. al. [19] translated the British Nationality Act into Prolog [8]. This entailed manually extracting the meaning from the Act then programming the Prolog rules. Using the logic of Prolog presented difficulty because the British Nationality Act expresses non-monotonic logic [19]. Other work has explored the use of non-monotonic logics, e.g. Defeasible Logic [24], to express the law, which is tested on a selection of Section 8.2 of Australia's Telecommunications Consumer Protections Code (2012) on complaint management.

2.3 Semantic Parsers

Another body of work has focused on automatic translation of tax law to formal representation using semantic parsers. A semantic parser takes text as input and outputs a formal representation, e.g. first-order logic. Semantic parsers are usually initialized with machine learning. They are trained on pairs of sentences and their corresponding logical representations. Much work focuses on training models for specific domains. Others are trained on a variety of corpora to achieve wide coverage. Semantic parsers include:

1. C&C/Boxer Combining C&C tools [6][3] and Boxer [5][4].

2. JAMR A graph-based parser [9] for Abstract Meaning Representation (AMR) [4].

3. Cornell AMR A CCG-based parser for AMR [3].

4. Cornell SPF A semantic parsing framework that uses CCG to implement various algorithms. [2]

5. CAMR A transition-based, AMR parser. [22]

6. NL2KR A platform with a CCG-based parser. [21]

The lack of a large machine learning data set available for law texts (training pairs of input text and output formal representation) makes it difficult to train a semantic parser specific to legal text. In [17], McCarty proposes a semi-supervised learning approach based on word embeddings computed from legal texts that could potentially be used to overcome this problem; however, this is still theoretical and has yet to be implemented.

Some research has taken a different approach by experimenting with wide-coverage semantic parsers. Work by Wyner et. al. [23] shows that this is not entirely unreasonable for short, simple sentences from legal texts, using C&C/Boxer parser. Gaur et al. [10] attempt the same task with their own semantic parser, NL2KR.

[2]The IRC does not necessarily follow the recommended structure.
[3]Consists of a POS-tagger, Named-Entity Recognition, and a CCG [20] parser.
[4]Maps CCG derivations output to Discourse Representation Structure [13]

[16] seek to show that the distinction between logical and statistical approaches is being closed with the development of models that can learn the conventional aspects of natural language meaning from corpora and databases.

3 METHOD

We have designed `IRC-to-default-logic`, an open source software pipeline, see Figure 1. The top level code, `pipeline.py`, takes as input the number of a specific section (or lower level unit) of the IRC to be parsed and a desired final representation. The pipeline consists of 4 functional modules:

1. Crawl: `irc_crawler.py`, Input: XML formatted IRC code. Output: A pipeline level data structure that mimics the level structure of the IRC.

2. Extract: `definition_extractor.py` & `rule_extractor.py`, Output: first-order logic of definitions and text segments of rules.

3. Parse: `candc_boxer_api.py` and `parse_amr.py` Input: a "sentence" aka text segment, Output: an intermediate formal representation.

4. Order: Output: DL.

The modules in `IRC-to-default-logic` are:

1. Crawl This module references the IRC in its XML format and isolates an element at any specified level. We represent the IRC in terms of abstract elements called "levels". Each level can be one of the following: section, subsection, paragraph, subparagraph, clause, sub-clause, item, sub-item, sub-sub-item. These are in hierarchical order; each level can be nested in levels that precede it. Each level can optionally contain any of the following: heading, chapeau, content, sub-levels, continuations. The heading indicates that this paragraph states a general rule. The chapeau sets up the beginning of the rule. The sub-levels, in this case sub-paragraphs, provide conditions on the contract mentioned in the chapeau, and the continuation states the conclusion of that rule.

2. Extract This module pattern matches and extracts (1) defined terms using a single regular expression, (2) definitions given the retrieved terms using a single regular expression (3) "general rules", "exceptions", and "special rules", by searching for levels with headers matching those terms. It outputs defined terms and first-order logic expressions representing ontological relations between defined terms. It outputs text for extracted rules.

The regular expressions are based upon the style guidelines for drafting legislation in a manual entitled "*House Legislative Counsel's Manual on Drafting Style*" [18]: (1) General rule – State the main message. (2) Exceptions – State the persons or things to which the main message does not apply. (3) Special rules – Describe the person or things – (a) to which the main message applies in a different way; or (b) for which there is a different message.

The manual lists three phrases that are generally used to "lead in" a definition: • "*For purposes of this* [provision]" • "*In this* [provision]" • "*As used in this* [provision]", where [provision] is a placeholder for the level type, such as "paragraph". The manual also indicates that drafters should begin a definition (after the lead in) with the phrase "*the term*", preceding it. The word or phrase following "*the term*" will specify the type of definition. There is no explicit guideline in the manual for what this should be. However, through using regular expressions, we have observed that the vast majority of definitions use the word "*means*" after the defined term. Other less common phrases used instead are "*has the meaning*", "*includes*", "*does not include*", "*shall include*", and "*shall not include*".

3. Parse This component parses initially with C&C Boxer. It displays the result in both Discourse Representation Structure (DRS) and First-Order Logic. The parser will fail on sentences that are too long (of which there are a few in the IRC). CAMR is called if C&C/Boxer fails, and outputs the result in AMR.

4. Order We use NLTK's logic package to parse and represent the first-order logic expressions that make up the background theory and default rules of our supernormal DL. We use NLTK's inference package to access the library's theorem prover as well as interface with the Mace4 model builder, which we use to process default rules, and query the supernormal DL.

After the pipeline completes (1–4), it is possible to query and prove the resulting DL with `default_logic.py`. This will reference background theory and default rules. The background theory is a set of first-order logic expressions that express any information that is already established. The default rules are a list of rules, expressed using first-order logic, that are ordered by priority. These default rules are processed, starting with the highest-priority rule, and we extend our theory using these rules. We stop processing default rules once a default rule being processed is inconsistent with our current theory [7]. The query formal representation we use is supernormal DL, a variant of DL, particularly suited for statutory law as detailed in [15].

4 EXPERIMENTS

Our experimental starting point is a set of elements in Section 163 that have been represented as DL in [15]. We worked first to achieve as complete and accurate extraction and parsing as possible of this set, using the expert translation as ground truth. We then applied the regular expression(s) we used in extraction to the rest of the IRC. We are interested in how many relevant rules and definitions can be parsed and represented in the entire IRC. More broadly we are looking for hints as to how much knowledge beyond the IRC we may need. For example, "interest" is not defined in the IRC but its definition is important. For that extra knowledge, we are interested in learning how much may need to be elicited from experts versus from another digital resource. We present our experiments following the `IRC-to-default-logic` pipeline.

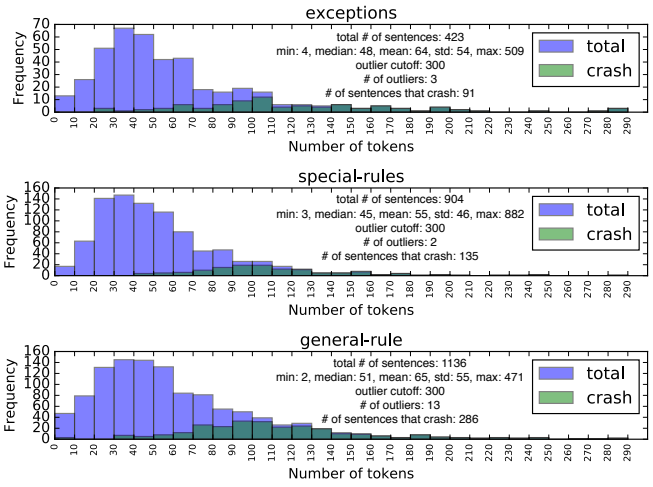
4.1 Definition & Rule Extraction

The pattern-based approach has shown some promising results. Although the lead-ins listed in Section 3 are used in many sections of the IRC, we find that the majority of definitions do not follow the style manual. The pattern: `the term (?:("[^"]+")|(âĂŸ[^âĂŽ]+âĂŽ))`[5] retrieves 4971 matches in the entire IRC (not including notes, repealed sections, omitted sections). Some of these matches are not found in definitions and are instead just references to a defined term. When we refine the pattern:

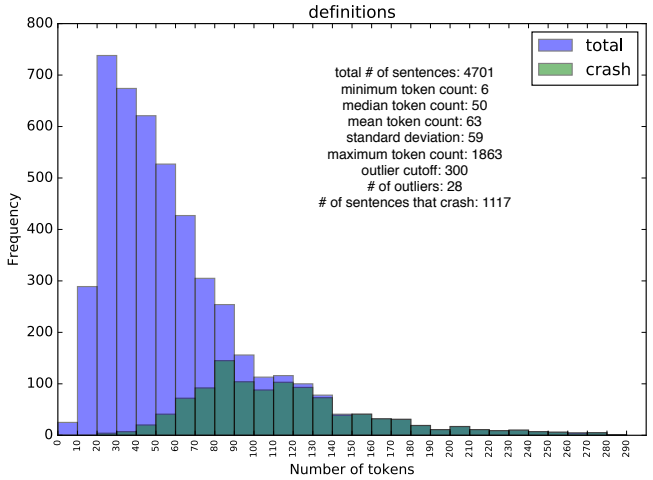
```
the term (?:("[^"]+")|(âĂŸ[^âĂŽ]+âĂŽ)) (?:means|includes|does not include|has the meaning|
shall include|shall not include)
```

we retrieve 4710 matches in all of the IRC, about 94.75% of all occurrences of the first pattern.

[5] See https://docs.python.org/2/library/re.html



(a) Frequency of tokens in IRC different rules.



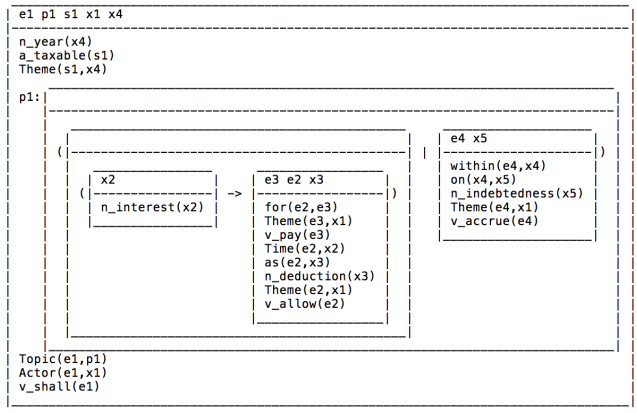
(b) Frequency of tokens in IRC definitions

Figure 2: Output from our analysis scripts for IRC rules and definitions.

Because the number of tokens in a sentence is a limit for the parsers we use, we also count the number of tokens we retrieve for exceptions, special and general rules. Figures 2a and 2b show histograms of the number of tokens found in each rule (Figure 2a) and definition (Figure 2b), as well as some statistics of token counts for the extracted rules and definitions[6].

4.2 Semantic Parsing

Subsection 163(a) is a general rule, as indicated by its header "GENERAL RULE". It states "*There shall be allowed as a deduction all interest paid or accrued within the taxable year on indebtedness.*". This sentence is one of the shortest sentences in the entire section, and C&C/Boxer and CAMR were unable to correctly parse it. They generated different outputs. The DRS outputs generated by C&C/Boxer are shown in Figure 3a and the converted first-order



(a) C&C/Boxer DRS output

```
exists e1 p1 s1 x1 x4.(n_year(x4) & a_taxable(s1) & Theme(s1,x4) & (all x2.(n_interest(x2) ->
exists e3 e2 x3.(for(e2,e3) & Theme(e3,x1) & v_pay(e3) & Time(e2,x2) & as(e2,x3) & n_deduction(x3)
& Theme(e2,x1) & v_allow(e2))) | exists e4 x5.(within(e4,x4) & on(x4,x5) & n_indebtedness(x5) &
Theme(e4,x1) & v_accrue(e4))) & Topic(e1,p1) & Actor(e1,x1) & v_shall(e1))
```

(b) C&C/Boxer FOL output

```
(x4 / allow-01
    :ARG1 (x7 / deduction)
    :ARG1 (x11 / or
                :op2 (x9 / interest-01
                        :quant (x8 / all))
                :op1 (x10 / pay-01)
                :op2 (x12 / accrue))
    :time (x13 / within
                :op1 (x18 / indebtedness
                        :time (x15 / thing
                                :name (n / name
                                        :op1 "taxable")))))
```

(c) CAMR AMR output

```
exists x4.(allow_01(x4) & exists x7.(deduction(x7) & ARG1(x4,x7)) & exists x11.(OR(x11) & exists
x9.(interest_01(x9) & exists x8.(ALL(x8) & quant(x9,x8)) & op2(x11,x9)) & exists x10.(pay_01(x10) &
op1(x11,x10)) & exists x12.(accrue(x12) & op2(x11,x12)) & ARG1(x4,x11)) & exists x13.(within(x13) &
exists x18.(indebtedness(x18) & exists x15.(thing(x15) & exists n.(name(n) & op1(n,"taxable") &
name(x15,n)) & time(x18,x15)) & op1(x13,x18)) & time(x4,x13)))
```

(d) CAMR FOL output

Figure 3: Parser outputs for section 163, subsection a.

logic in Figure 3b. From Figure 3a, we see that the operands of the "*or*" were wrongly parsed into "*There shall be allowed as a deduction all paid*" and "*accrued within the taxable year on indebtedness*", as indicated by the two boxes separated by the | symbol. However, the two innermost boxes separated by the -> symbol bear some semblance to a simpler statement of the rule that *if x is an interest then x is deductible*. Figure 3c shows CAMR outputs in AMR representation and Figure 3d the converted first-order logic. From Figure 3c, we see that CAMR's output is an invalid AMR graph, as the x4 node contains two edges with the same relation ARG1. It also misrepresents the "*or*" operation, as its operands have concepts "*all interest*", "*pay*", and "*accrue*". The "*or*" operation should have operands "*pay*" and "*accrue*", and should be modifying "*all interest*". However, it did capture the fact that a deduction should be allowed, and the representation is less convoluted than that of C&C/Boxer.

4.3 Default logic representation

Figure 4 shows the DL representation for Section 163 (h), used for determining whether interest, such as personal interest or qualified residence interest, is deductible. The background theory includes relations between defined terms, extracted by `definition_extractor.py`. Not all of these expressions in the background theory are necessary for determining whether personal interest and qualified residence

[6]Note that crashes in `IRC-to-default-logic` comes from C&C/Boxer calls.

```
# Background Theory:
all x.(qualified_SPACE_residence_SPACE_interest(x) -> -personal_SPACE_interest(x))
all x.(qualified_SPACE_residence(x) -> -personal_SPACE_interest(x))
all x.(acquisition_SPACE_indebtedness(x) -> qualified_SPACE_residence_SPACE_interest(x))
all x.(home_SPACE_equity_SPACE_indebtedness(x) -> qualified_SPACE_residence_SPACE_interest(x))
all x.(qualified_SPACE_residence(x) -> qualified_SPACE_residence_SPACE_interest(x))
all x.(qualified_SPACE_residence(x) -> acquisition_SPACE_indebtedness(x))
all x.(acquisition_SPACE_indebtedness(x) -> -home_SPACE_equity_SPACE_indebtedness(x))
all x.(qualified_SPACE_residence(x) -> -home_SPACE_equity_SPACE_indebtedness(x))
all x.(qualified_SPACE_residence(x) ->
pre_DASH_October_SPACE_13_COMMA__SPACE_1987_COMMA__SPACE_indebtedness(x))
all x.(personal_SPACE_interest(x) -> interest(x))
personal_SPACE_interest(y)
# Default Rules:
all x.(qualified_SPACE_residence_SPACE_interest(x) -> deductible(x))
all x.(personal_SPACE_interest(x) -> -deductible(x))
all x.(interest(x) -> deductible(x))
```

Figure 4: DL representation's background and default rules, from `pipeline.py`, for Section 163, subsection (h).

interest are deductible. Also, the expression `personal_SPACE_interest(y)` is one that we hand-coded, as this represents the query that one might inject into the background theory to query the DL theory about whether or not "personal interest" is deductible or not. However, the expression `all x.(personal_SPACE_interest(x) -> interest(x))` also had to be hand-coded; the term "*interest*" is not a defined term, and so this relation was not extracted by `definition_extractor.py`. The default rules shown were also hand-coded, as our semantic parser approach was not able to parse the sentences corresponding to these rules.

5 CONCLUSIONS & FUTURE WORK

In this paper we have presented software tools that make use of existing work in the field of semantic parsing, as well as having shown how we exploit the drafting style of the law to make use of regular expressions for extracting rules and definitions. We take our first step towards automatically converting the law from its natural language representation to a DL, that a computer can easily read and use for inference.

We extract formal representations from definitions and rules, by exploiting common structural patterns in the IRC. We are able to extract some simple formal representations from definitions in certain sections. However, extracting rules is not straightforward, most rules do not have an easily exploitable pattern and require deeper analysis.

We make use of existing natural language parsing algorithms that are capable of extracting formal representations from text. These parsers will only work adequately when the input text consists of short, simple sentences. Unfortunately, tax law in its textual representation consists of long, complex sentences that are difficult even for humans to understand. As a result, this approach has been only marginally successful.

As future work, we will investigate whether software could handle a large number of cases while flagging ambiguities it encounters. These ambiguities could be passed to an expert for assistance. This task could be sourced to law students or a pool of experts.

REFERENCES

[1] Yoo Jung An and Ned Wilson. 2016. Tax Knowledge Adventure: Ontologies that Analyze Corporate Tax Transactions. In *Proceedings of the 17th International Digital Government Research Conference on Digital Government Research*. ACM, 303–311.

[2] Yoav Artzi. 2016. Cornell SPF: Cornell Semantic Parsing Framework. (2016). arXiv:arXiv:1311.3011

[3] Yoav Artzi, Kenton Lee, and Luke Zettlemoyer. 2015. Broad-coverage CCG Semantic Parsing with AMR. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Lisbon, Portugal, 1699–1710. http://aclweb.org/anthology/D15-1198

[4] Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2012. Abstract meaning representation (AMR) 1.0 specification. In *Parsing on Freebase from Question-Answer Pairs. In Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing. Seattle: ACL*. 1533–1544.

[5] Johan Bos. 2008. Wide-coverage semantic analysis with boxer. In *Proceedings of the 2008 Conference on Semantics in Text Processing*. Association for Computational Linguistics, 277–286.

[6] Johan Bos, Stephen Clark, Mark Steedman, James R Curran, and Julia Hockenmaier. 2004. Wide-coverage semantic representations from a CCG parser. In *Proceedings of the 20th international conference on Computational Linguistics*. Association for Computational Linguistics, 1240.

[7] Gerhard Brewka and Thomas Eiter. 2000. Prioritizing default logic. In *Intellectics and computational logic*. Springer, 27–45.

[8] William Clocksin and Christopher S Mellish. 2003. *Programming in PROLOG*. Springer Science & Business Media.

[9] Jeffrey Flanigan, Sam Thomson, Jaime G Carbonell, Chris Dyer, and Noah A Smith. 2014. A discriminative graph-based parser for the abstract meaning representation. (2014).

[10] Shruti Gaur, Nguyen H Vo, Kazuaki Kashihara, and Chitta Baral. 2014. Translating simple legal text to formal representations. In *JSAI International Symposium on Artificial Intelligence*. Springer, 259–273.

[11] Erik Hemberg, Jacob Rosen, Geoffrey Warner, Sanith Wijesinghe, and Una-May O'Reilly. 2015. Tax Non-Compliance Detection Using Co-Evolution of Tax Evasion Risk and Audit Likelihood. In *ICAIL*.

[12] John F Horty. 2012. *Reasons as defaults*. Oxford University Press.

[13] Hans Kamp, Josef Van Genabith, and Uwe Reyle. 2011. Discourse representation theory. In *Handbook of philosophical logic*. Springer, 125–394.

[14] Sarah Lawsky. 2017. Formalizing the Code. *Tax Law Review* (2017).

[15] Sarah Lawsky. forthcoming 2017. A Logic for Statutes, 21 Florida Tax Review. (forthcoming 2017).

[16] Percy Liang and Christopher Potts. 2015. Bringing machine learning and compositional semantics together. *Annu. Rev. Linguist.* 1, 1 (2015), 355–376.

[17] L Thorne McCarty. 2016. Discussion Paper: On Semi-Supervised Learning of Legal Semantics. (2016). https://www.researchgate.net/profile/L_Thorne_Mccarty2/publication/304742441_Discussion_Paper_On_Semi-Supervised_Learning_of_Legal_Semantics/links/5778bfc108ae4645d61182cf.pdf

[18] US HOUSE OF REPRESENTATIVES. 1995. HOUSE LEGISLATIVE COUNSEL'S MANUAL ON DRAFTING STYLE. (1995).

[19] Marek J. Sergot, Fariba Sadri, Robert A. Kowalski, Frank Kriwaczek, Peter Hammond, and H Terese Cory. 1986. The British Nationality Act as a logic program. *Commun. ACM* 29, 5 (1986), 370–386.

[20] Mark Steedman and Jason Baldridge. 2011. Combinatory categorial grammar. *Non-Transformational Syntax: Formal and Explicit Models of Grammar. Wiley-Blackwell* (2011).

[21] Nguyen Ha Vo, Arindam Mitra, and Chitta Baral. 2015. The NL2KR Platform for building Natural Language Translation Systems.. In *ACL (1)*. 899–908.

[22] Chuan Wang, Nianwen Xue, and Sameer Pradhan. 2015. A Transition-based Algorithm for AMR Parsing. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Denver, Colorado, 366–375. http://www.aclweb.org/anthology/N15-1040

[23] Adam Wyner, Johan Bos, Valerio Basile, and Paulo Quaresma. 2012. An empirical approach to the semantic representation of laws. In *The 25th International Conference on Legal Knowledge and Information Systems*.

[24] Adam Wyner and Guido Governatori. 2013. A Study on Translating Regulatory Rules from Natural Language to Defeasible Logics.. In *RuleML (2)*. Citeseer.

[25] Adam Wyner and Wim Peters. 2011. On Rule Extraction from Regulations.. In *JURIX*, Vol. 11. 113–122.