

THE RESEARCH UNIVERSITY PRESS

International
Computational Law Journal

VOLUME 2 / ISSUE 5 / CURATED SOURCE EDITION

A Language For Legal Discourse is All You Need

L. Thorne McCarty

MIT Computational Law Report · Originally published 2023-06-13

Independent reader edition under CC BY 4.0. This attribution cover was added; original source PDF pages follow unchanged. Original authorship and source identity are retained. MIT, Stanford University, Stanford Law School and CodeX have not sponsored or endorsed this republication.

Source, license and version evidence accompany the online article.

MIT Computational Law Report

A Language For Legal Discourse is All You Need

L. Thorne McCarthy

Published on: Feb 01, 2022

URL: https://law.mit.edu/pub/languageforlegaldiscourse

License: Creative Commons Attribution 4.0 International License (CC-BY 4.0)

1. Introduction

My first paper on a *Language for Legal Discourse (LLD)* was published at the International Conference on Artificial Intelligence and Law in 1989 [24]. I used the language subsequently for several small projects: [50] [28] [30] [31], and it motivated much of my theoretical work on Knowledge Representation and Reasoning in those years.

At the time, no one was attempting anything as ambitious as the "Rules as Code" movement, and thus I never wrote an interpreter for the entire language or used it to encode a complete statute. But I think this is a feasible project today. Even without a full-scale implementation, I think the design choices embodied in *LLD* provide useful guidelines for anyone trying to translate legal rules into executable computer code. I will describe these choices in this short paper.

This document was originally prepared as a position paper for two conferences: the first Workshop on Programming Languages and the Law (ProLaLa 2022), which was part of POPL 2022, in Philadelphia, in January, and the conference on Computational Legal Studies 2022, in Singapore, in March. See Section 3, *infra*, for links to the slides and notes for my presentations at these conferences.

2. What is LLD?

My *Language for Legal Discourse (LLD)* is . . .

2.1. An Intuitionistic Logic Programming Language,

Among the several major programming paradigms — imperative, functional, logical, and object-oriented — the best representation for legal rules, in my opinion, is a logic programming language. This is (obviously) the position taken by Robert Kowalski, see [51] and [13], and it is also the position taken by Jason Morris in his recent MS Thesis [44]. One reason for this preference is that it is easy to encode a proof theory for various kinds of legal rules in a logic programming language, such as PROLOG: You get proof search and unification for free in the meta-interpreter. For an example of how this works, see [37], which includes code for several meta-interpreters.

However, I have also taken the position for several decades that the proper setting for logic programming is intuitionistic logic, not classical logic. To see the difference, consider a Horn clause written as an axiom in a sequent calculus, as follows:

$$Q_1 \wedge Q_2 \wedge \ldots \wedge Q_n \ \vdash_{\mathcal{C}} \ P$$

where $\mathcal{C}$ is a *context* containing the free variables appearing in the predicates. If all of our axioms are Horn clauses, then the logic is the same whether it is interpreted in classical or intuitionistic logic. But in

intuitionistic logic, we can write *universally quantified embedded implications*, as follows:

$$Q_1 \wedge \ldots \wedge \forall \mathbf{y} \, (R_1 \wedge R_2 \wedge \ldots \wedge R_k \;\Rightarrow\; Q) \wedge \ldots \wedge Q_n \;\vdash_{\mathcal{C}}\; P$$

An example from John McCarthy is: "**x** is a sterile container if every bug **y** inside **x** is dead." If we allowed this syntactic construction in classical logic, we would have a full first-order logical language that requires an unrestricted resolution theorem prover, but in intuitionistic logic we have a proper subset of a first-order language that retains both the definite-answer-substitution property and the goal-directed proof procedure that we want for logic programs. Motivated by both common sense reasoning and legal reasoning, I proposed such an intuitionistic logic programming language in [22][23]. At about the same time, motivated by applications in programming, generally, Dale Miller proposed a similar language in [41]. For a full account of Dale Miller's work, see [42].

Missing from an intuitionistic logic program, by design, are rules for *disjunctive* and *existential assertions*. We can add these rules separately, as in [37], or we can take the approach advocated in [38]. A statute often includes provisions that state necessary and sufficient conditions for a defined predicate, P. For the sufficient conditions, we can use the proof rules shown above, but in some contexts we need to *assert* P and *expand* the necessary conditions, which means that we need to reason with disjunctive and existential assertions. Ron van der Meyden and I showed, in [38], that we can model this reasoning, semantically, with John McCarthy's circumscription axiom [19], but this sometimes leads to inductive proofs.

Logic programming also offers a simple way to write default rules: *negation-as-failure*. Sarah Lawsky argues persuasively in [16] that most statutes are drafted and should be interpreted as rules with exceptions, and she uses Ray Reiter's *default logic* [49] to illustrate this point with a translation of Section 163(h) of the Internal Revenue Code. I took the following position in [29]: "If we used only stratified negation-as-failure with metalevel references in our representation language, we would have a powerful normal form for statutes and regulations," but I argued in the same paper that a well-drafted statute should never contain *unstratified* default rules. Nevertheless, if unstratified rules happen to occur in a statute, inadvertently, it may be comforting to know that the circumscription of intuitionistic embedded implications offers a reasonable interpretation of the possible inferences [40], with provable relationships to both the (2-valued) stable model semantics [10] and the (3-valued) well-founded semantics [52]. For the general theory, see [25].

2.2. With Sorts and Subsorts,

Here is an atomic formula that appears frequently in my encodings of legal rules:

```
(Own ?o (Actor ?a) (Stock ?s))
```

In this example, an `Actor` can be either a `Person` or a `Corporation`, and `Stock` is a subsort of `Security`. The unification algorithm is required to respect these sorts, although the details will depend on

how the sort hierarchy is defined. Also, `Actor` is a *count term*, while `Stock` is a *mass term* which can have a measure attached to it. For some examples of legal rules that use mass terms with measures, see [30].

One feature of this syntax which is unusual in a first-order logic is the variable `?o`. Think of `?o` as an instance of the ownership relation, so that `Own` can be interpreted as either a predicate or an object, depending on the context, and there is no syntactic distinction between an atomic predicate and an atomic term. Thus, in some contexts,

```
(Own ?o)
```

could be an argument in a higher-order expression. One way to formalize this interpretation is to define a type theory and a categorical logic for *LLD*. We will return to this point in Section 2.7, *infra*.

2.3. Events and Actions,

There are many ways to represent a world that changes over time: Temporal Logic, in various forms [48][46] [7]; Dynamic Logic [47][11]; the Situation Calculus [18][20]; the Event Calculus [14]; and many more.

My current choice in *LLD* is to represent an `Event` as a predicate defined over a linear temporal order, and to represent an `Action` as an `Event` paired with a responsible `Actor`. Let's assume the existence of a set of *basic events* and define a set of *abstract events* using Horn clauses and (optional) order relations. We can then represent *concurrent* and *repetitive* actions, with indefinite information about their order. For the theoretical details and several examples, see [39]. In that paper, to reason about these actions, Ron van der Meyden and I generalized the results in [38] and analyzed two methods for answering queries: (1) an inductive proof procedure for linear recursive definitions, which is sound but not complete; and (2) a decision procedure, which works for a natural class of rules and queries.

The semantics of my action language has always been influenced by the necessity of embedding it inside the deontic modalities, which will be discussed in Section 2.4, *infra*. In the earliest version, in [21], the basic actions were defined on partial states (called *substates*) and sequences of partial states (called *subworlds*) using the notion of *strict satisfaction*. The intention was to construct the *denotation* of an action, recursively, although the logic itself was still classical. Later, in [27], the background logic was intuitionistic, as described in Section 2.1, *supra*, and strict satisfaction was replaced with a construction based on the *principal filters* of a *final Kripke model*. This construction worked, technically, but it was painfully complex. I will suggest a simpler semantics in Section 2.7, *infra*.

2.4. Modalities Over Actions,

I published my first paper on a deontic logic for legal reasoning at IJCAI in 1983 [21], several years before my paper on *LLD*. Today, this system would be described as a dyadic deontic logic with a `Condition` and an `Action`, in which the `Action` is defined in a first-order dynamic logic [11]. I revisited the subject in 1994

[27], swapping out the dynamic logic and replacing it with the action language discussed in [39]. The 1994 paper also incorporated into the deontic language the interpretation of negation-as-failure that had been developed previously in [40].

The deontic semantics itself remained basically the same from 1983 to 1994. There are three modalities: $\mathbf{O}\langle\phi \mid \alpha\rangle$ (obligatory), $\mathbf{F}\langle\phi \mid \alpha\rangle$ (forbidden), $\mathbf{P}\langle\phi \mid \alpha\rangle$ (permitted), and each one can be negated with a monotonic intuitionistic negation. There are no negated actions in the language, which means that $\mathbf{O}$ and $\mathbf{F}$ must be defined separately. $\mathbf{P}$ is a "free choice" permission, or a *strong* permission. $\mathbf{P}\langle\phi \mid \alpha\rangle$ means: "Under the condition ϕ, *all* the ways of performing the action α are permitted." $\mathbf{F}\langle\phi \mid \alpha\rangle$ means: "Under the condition ϕ, all the ways of performing the action α are *not* permitted." Thus $\neg\mathbf{F}\langle\alpha\rangle$ is a *weak* permission. $\mathbf{O}\langle\phi \mid \alpha\rangle$ means: "Under the condition ϕ, *only* the ways of performing the action α are permitted." Formally, the semantics of all three modalities are determined by a single construct, the Grand Permitted Set, $\mathcal{P}$, which designates among all possible denotations of all possible actions those that are permitted and those that are not. Also introduced in [27] is the modality $\mathbf{DO}\langle\alpha\rangle$, which is *veridical* and therefore results in the event α being true in the successor world.

There are two theorems in [27] that have implications for the deontic proof theory in *LLD*. Theorem 4.7 says (roughly) that, in a language without $\mathbf{P}$, all inferences about $\mathbf{O}$ and $\mathbf{F}$ can be reduced to proofs in the action language. Theorem 4.8 says (roughly) that, in a language without $\mathbf{F}$, the inferences about $\mathbf{O}$ are independent from the inferences about $\mathbf{P}$. This means that we can construct simple deontic proofs in the style of a logic program in two restricted versions of the language. We can use $\mathbf{O}$ and $\mathbf{F}$, along with negation-as-failure on $\mathbf{F}$, which is essentially a default rule in the following form: "Every action that is not explicitly forbidden is permitted." Or we can use $\mathbf{O}$ and $\mathbf{P}$, which is essentially a default rule in the following form: "Every action that is not explicitly permitted is forbidden." These are two familiar principles from real legal systems, and there are several examples in [27] that fall into one of these two categories.

There is now an enormous literature on the "paradoxes" of deontic logic, with no consensus. My view is that the semantics for the deontic modalities is very simple, and the paradoxes arise from complexities in the action language and from our problematic formalisms for default inference. For example, see [26] for an analysis of Chisholm's Paradox, based on the Grand Permitted Set, $\mathcal{P}$, which is itself very simple, along with a somewhat more complicated system for default reasoning.

2.5. Epistemic Modalities,

It is a slight overstatement to say that the current version of *LLD* includes the epistemic modalities, but the facilities that we need to model *knowledge* and *belief* are a basic part of the language.

The traditional approach [8] treats knowledge as a modal operator, $\mathbf{K}P$, endowed with a Kripke semantics. But there is a more powerful approach in the recent literature on *justification logics* [3][4]. Based on the work of Sergei Artemov on the Logic of Proofs (LP) [2], a justification logic adds the annotation $t{:}P$ to the

proposition P and interprets this compound term as "P is justified by reason t." Essentially, t is a *proof* of P, and it can be extracted from a provable modal formula, $\mathbf{K}P$, by what is known as a Realization Theorem. This is currently an active area of research, and there are justification logics that correspond to the modal systems $\mathbf{K}$, $\mathbf{T}$, $\mathbf{K4}$, $\mathbf{S4}$, $\mathbf{K45}$, $\mathbf{S5}$ and others.

From my perspective, this work is interesting because it is fairly easy to construct and manipulate proofs in a logic programming language such as *LLD*. For a further note on the structure of proofs in *LLD*, see Section 2.7, *infra*.

2.6. A Natural Language Interface,

The syntax of atomic formulas in *LLD*, as shown in Section 2.2, *supra*, makes it a good target language for natural language processing. In my paper at ICAIL in 2007 [31], I developed a *quasi-logical form*, or *QLF*, to represent the semantic interpretation of a sentence in a legal text, and a *definite clause grammar*, or *DCG*, to compute the correct quasi-logical form from the output of a syntactic parser. The *QLF*s were intended to serve as an intermediate step towards the full representation of a legal case in *LLD*. There are several examples in the paper, but for a larger sample see the *QLF*s for 211 sentences from *Carter v. Exxon Co., USA*, 177 F.3d 197 (3d Cir. 1999), available online at `http://bit.ly/2yhnPdC`. The syntactic analysis was from Michael Collins' statistical parser [5], which is now more than 20 years old. There are better parsers available today.

Can these techniques be applied to statutes and regulations? In 2009, Tim Armstrong and I attempted to compute a semantic interpretation of Articles 2 and 3 of the Uniform Commercial Code (UCC). The results were poor [1]. Other researchers have also reported negative results on parsing statutes [53][43][45]. The explanation seems fairly clear: The syntax of a statute is complex, contorted, and far removed from the syntax of the sentences on which our current parsers have been trained. One alternative is to use a human annotator and a *controlled natural language* interface, such as Robert Kowalski's Logical English [12]. Another alternative is to experiment with a *large language model*, such as BERT [6], which might be able to learn an idiosyncratic syntax from a small number of annotated examples.

2.7. And a Prototypical Perceptual Semantics.

My current work on *deep learning* might appear to be far removed from a *Language for Legal Discourse*, but it is actually part of a broader effort to bridge the gap between machine-learning-based AI and logic-based AI.

My most recent papers [34][35] develop a theory of *clustering* and *coding* that combines a geometric model with a probabilistic model in a principled way. The geometric model is a Riemannian manifold with a Riemannian metric, $g_{ij}(\mathbf{x})$, which is interpreted as a measure of *dissimilarity*. The dissimilarity metric is defined by an equation that depends on the probability density of the sample input data — in an image classification task, for example — and this leads to an optimal lower dimensional encoding of the data, an

object that is referred to as a *prototypical cluster*. The theory is illustrated by a number of examples from the MNIST dataset [17] and the CIFAR-10 dataset [15].

The main thesis of my paper at ICAIL 2015 [32] is that the *theory of differential similarity* also provides a novel semantics for my *Language for Legal Discourse*. Here is an excerpt from the last page of [35], which explains how this works in the image classification tasks:

> There is one operation that appears repeatedly . . . We construct a product manifold consisting of four prototypical clusters, and then construct a submanifold which is itself a prototypical cluster. We can now take a big step: We can use this construction to define the semantics of an *atomic formula* in a logical language, that is, to define a *predicate* with four *arguments*. The general idea is to replace the standard semantics of classical logic, based on sets and their elements, with a semantics based on manifolds and their points. . . . The natural setting for these developments is a logical language based on category theory, or what is known as a *categorical logic*.

Thus, in [32], I presented a categorical logic based on the category of differential manifolds (**Man**), which is weaker than a logic based on the category of sets (**Set**) or the category of topological spaces (**Top**). See [32] for the technical details, or see [33] for a more informal exposition. A comprehensive paper addressed to the computational logic community is currently in preparation [36].

Here are some consequences for the several questions left open from previous sections of this paper:

From Section 2.2: The hierarchy of prototypical clusters in an image classification task is a model for the type hierarchies that we need for predicates like `Own` in *LLD*.

From Section 2.3: How do we represent actions in a sequence of moving images? Suppose we take differential manifolds seriously, and represent an action by the Lie group of *affine isometries* in $\mathbb{R}^3$, also known as *rigid body motions*. See [9], Chapter 18. We can then apply the theory of differential similarity to the manifold of physical actions, and generalize from there to a manifold of abstract actions. This gives us a new semantics for the action language in *LLD*.

From Section 2.5: In a categorical logic, a sequent is a *morphism*, and a proof is a *composition* of morphisms. Thus, in the category **Man**, a proof is a *smooth mapping* of differential manifolds, which means that justifications have much more structure in *LLD* than they do in other languages.

3. Conclusion

I have tried to demonstrate in this short paper that the features of my *Language for Legal Discourse (LLD)* are necessary for a computational representation of statutory and regulatory rules. Are they also sufficient? Perhaps. But there are still some aspects of the language that are subject to revision, and we still need to implement it in full and apply it to a range of real legal examples.

In my presentations at the Workshop on Programming Languages and the Law (ProLaLa 2022) and at the conference on Computational Legal Studies (CLS 2022), I discussed several concrete examples of the abstract concepts in this paper. The title of my talk at CLS 2022 was “Bridging the Gap between Machine Learning and Logical Rules in Computational Legal Studies,” and it covered the main ideas in Section 2.7, *supra*. Here is a video:

> `https://youtu.be/rBPadM9tyNo`

The "gap" is illustrated with an example from Article 3 of the Uniform Commercial Code, which then motivates the discussion of “A Theoretical Synthesis” in the last half of the talk. Here are the slides and notes from CLS 2022:

> `https://bit.ly/3x32x25`
> `https://bit.ly/3OatUhJ`

My talk at ProLaLa 2022 discussed several additional examples from Article 3 of the Uniform Commercial Code. Here are the slides and notes:

> `https://bit.ly/39aHEKd`
> `https://bit.ly/3tf8rMq`

These examples demonstrate the automatic generation of *QLF*s and their translation into *LLD*, following Section 2.6, *supra*, and they cover the deontic modalities from Section 2.4 and the epistemic modalities from Section 2.5. Taken together, my presentations at ProLaLa 2022 and CLS 2022 outline a methodology for translating statutory and regulatory rules into executable computer code, and they suggest a research agenda to help us achieve this goal.

References

[1] T. J. Armstrong and L. T. McCarty. Parsing the text of the Uniform Com- mercial Code. In *Workshop on Natural Language Engineering of Legal Argu- mentation (NaLELA), at ICAIL*, 2009.

[2] S. N. Artemov. Explicit provability and constructive semantics. *Bulletin of Symbolic Logic*, 7(1):1–36, 2001.

[3] S. N. Artemov. The logic of justification. *The Review of Symbolic Logic*, 1(4):477–513, 2008.

[4] S. N. Artemov and M. Fitting. *Justification Logic: Reasoning with Reasons*. Cambridge University Press, 2019.

[5] M. Collins. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, 1999.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings, 2019 Conference of the NAACL: Human Language Technologies*, pages 4171– 4186, 2019.

[7] E. A. Emerson and J. Y. Halpern. Decision Procedures and Expressiveness in the Temporal Logic of Branching Time. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing*, pages 169–180. ACM, 1982.

[8] R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995.

[9] J. Gallier. *Geometric methods and applications: For computer science and engineering. 2nd ed*. Texts in Applied Mathematics 38. Springer, 2011.

[10] M. Gelfond and V. Lifschitz. The stable model semantics for logic program- ming. In *Proceedings, Fifth International Conference and Symposium on Logic Programming*, pages 1070–1080, 1988.

[11] D. Harel. *First-Order Dynamic Logic*. Springer-Verlag, 1979. LNCS No. 68.

[12] R. Kowalski and A. Datoo. Logical English meets legal English for swaps and derivatives. *Artificial Intelligence and Law*, 2021. https://doi.org/10.1007/s10506-021-09295-3.

[13] R. A. Kowalski. Legislation as logic programs. In G. Comyn, N. E. Fuchs, and M. J. Ratcliffe, editors, *Logic Programming in Action*, pages 203–230. Springer Berlin Heidelberg, 1992.

[14] R. A. Kowalski and M. J. Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–95, 1986.

[15] A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Department of Computer Science, University of Toronto, 2009.

[16] S. B. Lawsky. A logic for statutes. *Florida Tax Review*, 21:60, 2017.

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[18] J. McCarthy. Situations, actions, and causal laws. In M. Minsky, editor, *Semantic Information Processing*, pages 410–417, 1968.

[19] J. McCarthy. Circumscription: A form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980.

[20] J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969.

[21] L. T. McCarty. Permissions and obligations. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 287–294, 1983.

[22] L. T. McCarty. Clausal intuitionistic logic. I. Fixed-point semantics. *Journal of Logic Programming*, 5(1):1–31, 1988.

[23] L. T. McCarty. Clausal intuitionistic logic. II. Tableau proof procedures. *Journal of Logic Programming*, 5(2):93–132, 1988.

[24] L. T. McCarty. A language for legal discourse. I. Basic features. In *Proceedings of the Second International Conference on Artificial Intelligence and Law*, pages 180–189. ACM Press, 1989.

[25] L. T. McCarty. Circumscribing embedded implications (without stratifications). *Journal of Logic Programming*, 17:323–364, 1993.

[26] L. T. McCarty. Defeasible deontic reasoning. *Fundamenta Informaticae*, 21(1–2):125–148, 1994.

[27] L. T. McCarty. Modalities over actions, I. Model theory. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR94)*, pages 437–448. Morgan Kaufmann, 1994.

[28] L. T. McCarty. An implementation of Eisner v. Macomber. In *Proceedings of the Fifth International Conference on Artificial Intelligence and Law*, pages 276–286. ACM Press, 1995.

[29] L. T. McCarty. Some arguments about legal arguments. In *Proceedings of the Sixth International Conference on Artificial Intelligence and Law*, pages 215–224. ACM Press, 1997.

[30] L. T. McCarty. Ownership: A case study in the representation of legal concepts. *Artificial Intelligence and Law*, 10(1-3):135–161, 2002.

[31] L. T. McCarty. Deep semantic interpretations of legal texts. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Law*, pages 217–224. ACM Press, 2007.

[32] L. T. McCarty. How to ground a language for legal discourse in a prototypical perceptual semantics. In *Proceedings of the Fifteenth International Conference on Artificial Intelligence and Law*, pages 89–98. ACM Press, 2015.

[33] L. T. McCarty. How to ground a language for legal discourse in a prototypical perceptual semantics. *Michigan State Law Review*, 2016:511–538, 2016.

[34] L. T. McCarty. Clustering, coding, and the concept of similarity (Version 2.0). *Preprint*, arXiv:1401.2411v2 [cs.LG], 2018.

[35] L. T. McCarty. Differential similarity in higher dimensional spaces: Theory and applications (Version 3.0). *Preprint*, arXiv:1902.03667v3 [cs.LG, stat.ML], 2021.

[36] L. T. McCarty. Manifold logic and the theory of differential similarity. Forthcoming, 2023.

[37] L. T. McCarty and L. A. Shklar. A PROLOG interpreter for first-order intu- itionistic logic (abstract). In *Proceedings, 1994 International Logic Program- ming Symposium*, page 685. MIT Press, 1994.

[38] L. T. McCarty and R. van der Meyden. Indefinite reasoning with definite rules. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 890–896, 1991.

[39] L. T. McCarty and R. van der Meyden. Reasoning about indefinite actions. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR92)*, pages 59–70. Morgan Kaufmann, 1992.

[40] L. T. McCarty and R. van der Meyden. An intuitionistic interpretation of finite and infinite failure. In L. M. Pereira and A. Nerode, editors, *Logic Programming and NonMonotonic Reasoning: Proceedings of the Second International Workshop*, pages 417–436. MIT Press, 1993.

[41] D. Miller. A logical analysis of modules in logic programming. *Journal of Logic Programming*, 6(1):79–108, 1989.

[42] D. Miller and G. Nadathur. *Programming with Higher-Order Logic*. Cambridge University Press, 2012.

[43] L. Morgenstern. Toward automated international law compliance monitoring (TAILCM). Technical Report AFRL-RI-RS-TR-2014-206, Leidos, Inc., 2014.

[44] J. P. Morris. Spreadsheets for legal reasoning: The continued promise of declarative logic programming in law. Master’s thesis, University of Alberta, 2020.

[45] M. A. Pertierra, S. Lawsky, E. Hemberg, and U. O’Reilly. Towards formalizing statute law as default logic through automatic semantic parsing. In *2nd Workshop on Automated Semantic Analysis of Information in Legal Texts (ASAIL), at ICAIL*, 2017.

[46] A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 46–57. IEEE Com- puter Society, 1977.

[47] V. Pratt. Semantical considerations on Floyd-Hoare logic. In *Proceedings, 17th IEEE Symposium on Foundations of Computer Science*, pages 109–121, 1976.

[48] A. N. Prior. *Past, present and future*. Oxford University Press, London, 1967.

[49] R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1-2):81–132, 1980.

[50] D. A. Schlobohm and L. T. McCarty. EPS-II: Estate planning with prototypes. In *Proceedings of the Second International Conference on Artificial Intelligence and Law*, pages 1–10. ACM Press, 1989.

[51] M. J. Sergot, F. Sadri, R. A. Kowalski, F. Kriwaczek, P. Hammond, and H. T. Cory. The British Nationality Act as a logic program. *Communications of the ACM*, 29:370–386, 1986.

[52] A. Van Gelder, K. A. Ross, and J. S. Schlipf. Unfounded sets and well-founded semantics for general logic programs. In *Proceedings, Seventh ACM Symposium on the Principles of Database Systems*, pages 221–230, 1988.

[53] A. Wyner and G. Governatori. A study on translating regulatory rules from natural language to defeasible logic. In *Proceedings of RuleML 2013*. CEUR, 2013.

References

1. L. T. McCarty. A language for legal discourse. I. Basic features. *In Proceed-ings of the Second International Conference on Artificial Intelligence and Law*, pages 180–189. ACM Press, 1989. ↩
2. D. A. Schlobohm and L. T. McCarty. EPS-II: Estate planning with prototypes. In *Proceedings of the Second International Conference on Artificial Intelligence and Law*, pages 1–10. ACM Press, 1989. ↩
3. L. T. McCarty. An implementation of Eisner v. Macomber. In *Proceedings of the Fifth International Conference on Artificial Intelligence and Law*, pages 276–286. ACM Press, 1995. ↩
4. L. T. McCarty. Ownership: A case study in the representation of legal concepts. *Artificial Intelligence and Law*, 10(1-3):135–161, 2002. ↩
5. L. T. McCarty. Deep semantic interpretations of legal texts. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Law*, pages 217–224. ACM Press, 2007. ↩
6. M. J. Sergot, F. Sadri, R. A. Kowalski, F. Kriwaczek, P. Hammond, and H. T. Cory. The British Nationality Act as a logic program. *Communications of the ACM*, 29:370–386, 1986. ↩
7. R. A. Kowalski. Legislation as logic programs. In G. Comyn, N. E. Fuchs, and M. J. Ratcliffe, editors, *Logic Programming in Action*, pages 203–230. Springer Berlin Heidelberg, 1992. ↩
8. J. P. Morris. Spreadsheets for legal reasoning: The continued promise of declarative logic programming in law. Master's thesis, University of Alberta, 2020. ↩
9. L. T. McCarty and L. A. Shklar. A PROLOG interpreter for first-order intuitionistic logic (abstract). In *Proceedings, 1994 International Logic Programming Symposium*, page 685. MIT Press, 1994. ↩

10. L. T. McCarty. Clausal intuitionistic logic. I. Fixed-point semantics. *Journal of Logic Programming*, 5(1):1–31, 1988. ↩

11. L. T. McCarty. Clausal intuitionistic logic. II. Tableau proof procedures. *Journal of Logic Programming*, 5(2):93–132, 1988. ↩

12. D. Miller. A logical analysis of modules in logic programming. *Journal of Logic Programming*, 6(1):79–108, 1989. ↩

13. D. Miller and G. Nadathur. *Programming with Higher-Order Logic*. Cambridge University Press, 2012. ↩

14. L. T. McCarty and R. van der Meyden. Indefinite reasoning with definite rules. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 890–896, 1991. ↩

15. L. T. McCarty and R. van der Meyden. Indefinite reasoning with definite rules. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, pages 890–896, 1991. ↩

16. J. McCarthy. Circumscription: A form of non-monotonic reasoning. *Artificial Intelligence*, 13:27–39, 1980. ↩

17. S. B. Lawsky. A logic for statutes. *Florida Tax Review*, 21:60, 2017. ↩

18. R. Reiter. A logic for default reasoning. *Artificial Intelligence*, 13(1-2):81–132, 1980. ↩

19. L. T. McCarty. Some arguments about legal arguments. In *Proceedings of the Sixth International Conference on Artificial Intelligence and Law*, pages 215– 224. ACM Press, 1997. ↩

20. L. T. McCarty and R. van der Meyden. An intuitionistic interpretation of finite and infinite failure. In L. M. Pereira and A. Nerode, editors, *Logic Programming and NonMonotonic Reasoning: Proceedings of the Second International Workshop*, pages 417–436. MIT Press, 1993. ↩

21. M. Gelfond and V. Lifschitz. The stable model semantics for logic programming. In *Proceedings, Fifth International Conference and Symposium on Logic Programming*, pages 1070–1080, 1988. ↩

22. A. Van Gelder, K. A. Ross, and J. S. Schlipf. Unfounded sets and well-founded semantics for general logic programs. In *Proceedings, Seventh ACM Symposium on the Principles of Database Systems*, pages 221–230, 1988. ↩

23. L. T. McCarty. Circumscribing embedded implications (without stratifications). *Journal of Logic Programming*, 17:323–364, 1993. ↩

24. A. N. Prior. *Past, present and future*. Oxford University Press, London, 1967. ↩

25. A. Pnueli. The temporal logic of programs. In *Proceedings of the 18th Annual Symposium on Foundations of Computer Science*, pages 46–57. IEEE Computer Society, 1977. ↩

26. E. A. Emerson and J. Y. Halpern. Decision Procedures and Expressiveness in the Temporal Logic of Branching Time. In *Proceedings of the 14th Annual ACM Symposium on Theory of Computing*, pages 169–180. ACM, 1982. ↩

27. V. Pratt. Semantical considerations on Floyd-Hoare logic. In *Proceedings, 17th IEEE Symposium on Foundations of Computer Science*, pages 109–121, 1976. ↩

28. D. Harel. *First-Order Dynamic Logic*. Springer-Verlag, 1979. LNCS No. 68. ↩

29. J. McCarthy. Situations, actions, and causal laws. In M. Minsky, editor, *Semantic Information Processing*, pages 410–417, 1968. ↩

30. J. McCarthy and P. J. Hayes. Some philosophical problems from the standpoint of artificial intelligence. In *Machine Intelligence 4*, pages 463–502. Edinburgh University Press, 1969. ↩

31. R. A. Kowalski and M. J. Sergot. A logic-based calculus of events. *New Generation Computing*, 4(1):67–95, 1986. ↩

32. L. T. McCarty and R. van der Meyden. Reasoning about indefinite actions. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Third International Conference (KR92)*, pages 59–70. Morgan Kaufmann, 1992. ↩

33. L. T. McCarty. Permissions and obligations. In *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, pages 287–294, 1983. ↩

34. L. T. McCarty. Modalities over actions, I. Model theory. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR94)*, pages 437–448. Morgan Kaufmann, 1994. ↩

35. L. T. McCarty. Modalities over actions, I. Model theory. In *Principles of Knowledge Representation and Reasoning: Proceedings of the Fourth International Conference (KR94)*, pages 437–448. Morgan Kaufmann, 1994. ↩

36. L. T. McCarty. Defeasible deontic reasoning. *Fundamenta Informaticae*, 21(1– 2):125–148, 1994. ↩

37. R. Fagin, J. Y. Halpern, Y. Moses, and M. Y. Vardi. *Reasoning About Knowledge*. MIT Press, 1995. ↩

38. S. N. Artemov. The logic of justification. *The Review of Symbolic Logic*, 1(4):477–513, 2008. ↩

39. S. N. Artemov and M. Fitting. *Justification Logic: Reasoning with Reasons*. Cambridge University Press, 2019. ↩

40. S. N. Artemov. Explicit provability and constructive semantics. *Bulletin of Symbolic Logic*, 7(1):1–36, 2001. ↩

41. L. T. McCarty. Deep semantic interpretations of legal texts. In *Proceedings of the Eleventh International Conference on Artificial Intelligence and Law*, pages 217–224. ACM Press, 2007. ↩

42. M. Collins. *Head-Driven Statistical Models for Natural Language Parsing*. PhD thesis, University of Pennsylvania, 1999. ↩

43. T. J. Armstrong and L. T. McCarty. Parsing the text of the Uniform Commercial Code. In *Workshop on Natural Language Engineering of Legal Argumentation (NaLELA), at ICAIL*, 2009. ↩

44. A. Wyner and G. Governatori. A study on translating regulatory rules from natural language to defeasible logic. In *Proceedings of RuleML 2013*. CEUR, 2013. ↩

45. L. Morgenstern. Toward automated international law compliance monitoring (TAILCM). Technical Report AFRL-RI-RS-TR-2014-206, Leidos, Inc., 2014. ↩

46. M. A. Pertierra, S. Lawsky, E. Hemberg, and U. O'Reilly. Towards formalizing statute law as default logic through automatic semantic parsing. In *2nd Workshop on Automated Semantic Analysis of Information in Legal Texts (ASAIL), at ICAIL*, 2017. ↩

47. R. Kowalski and A. Datoo. Logical English meets legal English for swaps and derivatives. *Artificial Intelligence and Law*, 2021. https://doi.org/10.1007/s10506-021-09295-3. ↩

48. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings, 2019 Conference of the NAACL: Human Language Technologies*, pages 4171– 4186, 2019. ↩

49. L. T. McCarty. Clustering, coding, and the concept of similarity (Version 2.0). *Preprint*, arXiv:1401.2411v2 [cs.LG], 2018. ↩

50. L. T. McCarty. Differential similarity in higher dimensional spaces: Theory and applications (Version 3.0). *Preprint*, arXiv:1902.03667v3 [cs.LG, stat.ML], 2021. ↩

51. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. ↩

52. A. Krizhevsky. Learning multiple layers of features from tiny images. Technical report, Department of Computer Science, University of Toronto, 2009. ↩

53. L. T. McCarty. How to ground a language for legal discourse in a prototypical perceptual semantics. In Proceedings of the Fifteenth International Conference on Artificial Intelligence and Law, pages 89–98. ACM Press, 2015. ↩

54. L. T. McCarty. How to ground a language for legal discourse in a prototypical perceptual semantics. Michigan State Law Review, 2016:511–538, 2016. ↩

55. L. T. McCarty. Manifold logic and the theory of differential similarity. Forthcoming, 2023. ↩

56. J. Gallier. *Geometric methods and applications: For computer science and engineering. 2nd ed.* Texts in Applied Mathematics 38. Springer, 2011. ↩