Numerical Reasoning over Legal Contracts via Relational Database

Jiani Huang
University of Pennsylvania
jianih@seas.upenn.edu

Ziyang Li
University of Pennsylvania
liby99@seas.upenn.edu

Ilias Fountalis
Relational AI
ilias.fountalis@relational.ai

Mayur Naik
University of Pennsylvania
mhnaik@seas.upenn.edu

Abstract

Numerical reasoning over text requires deep integration between the semantic understanding of the natural language context and the mathematical calculation of the symbolic terms. However, existing approaches are limited in their ability to incorporate domain-specific knowledge and express mathematical formulas over data structures. Delegating logic reasoning to a relational database is a promising approach to enhance the reasoning complexity. We study the problem of distilling natural language text into a relational database with numerical data structure and querying this database to obtain desired answers. Specifically, given a legal contract and a set of date-related questions in natural language, we utilize pre-trained neural network models to create a relational database to retrieve and generate the target dates. We evaluate our method on the CUAD dataset and demonstrate that our approach has high correct answer coverage and reduces a significant amount of incorrect results even without any labels.

1 Introduction

Performing numerical reasoning in Natural Language (NL) reading comprehension remains an open problem. While large language models like GPT-3 [3], ERNIE3.0 [21], and BERT [8] show great promise and achieve impressive results in many NL benchmarks, it is still challenging to perform logical reasoning and numerical calculations. For example, when GPT-J [23], an open-source implementation of GPT-3, is asked to auto-complete the sentence "The month following January is _", it answers "March". The model is capable of memorizing commonly occurring terms, but it does not possess factual relations between such concepts. Recent works targeting the DROP benchmark [9] focus on numerical reasoning over text. However, even the state-of-the-art approach [5] does not generalize to numerical operations beyond plus and subtract over simple numbers.

We bridge the gap between natural language comprehension and numerical reasoning by representing an NL document in a Datalog-based relational database. Questions can be represented by Datalog queries and the desired answer is drawn from the execution output. This representation offers several benefits. First, the semantic meaning of the NL document can be captured by relations generated using well-known NLP techniques, such as Semantic Role Labeling (SRL), Named Entity Recognition (NER), coreference resolution, and synonym resolution. Second, numerical constructs such as integer, date, and period are native to relational databases, enabling us to do precise computation. Third, the Datalog query is not only interpretable, but the system can also efficiently collect provenance information while executing the query, making the prediction fully explainable. Finally, recent advances in probabilistic and differentiable logic programming [1, 6, 14] can be employed to fine-tune NL models and also synthesize queries.

In this paper, we focus specifically on date-related question answering (QA) over legal contracts. Recent works [4] observe that large language models trained on broader datasets struggle to cope

Challenge	NR	LD	Example
Number Calculation	✓	✗	... sold for $16.3 million, well above its $12 million high estimate.
Date Retrieval	✓	✓	... the Agreement is made as of May 22, 2000 ...
Date Calculation	✓	✓	.. shall remain effective for 2 years from and after the E.D. ...
Domain Specific	✗	✓	expiration date = effective date + term
Multi-hop Reasoning	✗	✓	The Effective Date ... shown by the signatures below.

Table 1: We compare general numerical reasoning required reading comprehension task [9] against date related questions in legal documents. The legal document has several unique challenges: date focused calculation, domain specific knowledge, and multi-hop reasoning.

with a specialized domain such as comprehending legal documents (LD). In addition, this setting poses several unique challenges to traditional NL reasoning systems for numerical reasoning (NR), as shown in Table 1. We demonstrate that our approach can overcome the challenges above by evaluating it on the CUAD dataset [10]. The dataset contains 511 legal documents and 5 kinds of date-related questions (e.g. "the expiration date of this contract"). With 18 Datalog rules in total (3.6 rules per question kind on average), we achieve a 77.7% recall for 1,342 answers of the 5 question kinds combined.

2 Related Work

There have been several attempts to integrate Natural Language with Databases. Natural language interface to Database focuses on using natural language to query existing databases, i.e. interfacing databases with NL. For example, the Spider dataset [24] contains static database schemas along with SQL queries, enabling recent text-to-SQL approaches [7] to zero-shot generate SQL queries on even unseen database schemas. We, on the other hand, focus on transforming a natural language passage into a standalone relational database, rather than focusing on query synthesis.

NeuralDB [22] constructs Natural Language Database (NLDB) and trains a Neural SPJ (Select-Project-Join) operator to perform a database query. However, they construct the NLDB from *"corpora that consist of unordered collections of facts expressed as short natural language sentences"*, which does not apply to legal documents, and in general to natural language passages. General NL passages have rich sentence structures and inter-sentence relations that short sentences cannot capture, and legal documents commonly employ long, complex, and formal prose. On the contrary, our relational database construction takes these into account by employing well-known NLP techniques.

Differentiable and probabilistic reasoning has been studied extensively [18]. DeepProbLog [13], DiffLog [20], and Scallop [1] employ differentiable reasoning on probabilistic databases, which could be used to extend our framework for fine-tuning NLP models. Further, neural-based approaches to rule learning such as NTP [14], GNTP [15], and CTP [16] can be used to automatically generate programmatic queries.

3 Framework

Constructing the Relational Database The first part of our framework concerns constructing a relational database from the NL document. Our database captures unitary entities such as a verb (e.g. "continue"), phrase (e.g. "this content license agreement (agreement)") and date (e.g. "May 22, 2020"). Additionally, it contains relationships between entities that capture the semantic information. We employ pre-trained NLP models for tokenizing, NER, SRL, synonym resolution, and coreference resolution. With the full extensional database (EDB) schema shown in Table 2, we describe the four stages of constructing our relational database:

1. **Tokenizing.** We first convert the plain text into tokens using an off-the-shelf tokenizer [11]. The tokenizer not only provides the token with its position in the text but also the token's part-of-speech (POS) information. We store the verbs and nouns identified by the tokenizer in relations such as `verb` and `noun`, respectively.
2. **Extracting Numbers.** Given the sequence of tokens generated by the tokenizer, the NER system can help us identify numbers and dates in the document. All numerical tokens, such as "Forty" and "May 20, 2000", are stored into the `num` relation with their respective number type (e.g. "cardinal", "date", "period"). For the numbers with type "date" and "period", we further utilize a date parser

Preprint. Under review.

Relation	Arguments	Generated By
verb	(<verb_id|keyword_id>, String)	Tokenizer
noun	(<noun_id|keyword_id>, String)	Tokenizer
phrase	(<phrase_id>, String)	SRL
num	(<num_id>, String, <num_type>)	NER
date	(<num_id>, Date)	NER, DateParser
period	(<num_id>, Period)	NER, DateParser
arg	(<verb_id>, <phrase_id>, <arg_type>)	SRL
synonym	(<keyword_id>, <noun_id|verb_id>)	Similarity Network
coref	(<*_id>, <*_id>, String)	Corefence Resolution, Regex
overlap	(<*_id>, <*_id>)	Tokenizer

Table 2: Extensional Database (EDB) Schema. Each row shows the name of the relation, the argument types, and the method(s) used to generate such relation. Note that we use `*_id`s to represent entities in the database, while keeping their raw string form in relations such as `verb`, `noun`, and `phrase`.

to convert the phrase into a unified data structure of `Date` and `Period` in our relational database, before storing them in the `date` and `period` relations.

3. **Entity Relations within Sentence.** We adopt SRL models to link the entities within a single sentence. SRL aims to capture the semantic meaning by identifying the arguments associated with each verb in the sentence. For example, in the sentence "Tom kicks the ball", "Tom" is the actor (`arg0`) of the verb "kicks", while "ball" is the receiver (`arg1`). We specifically care about the *temporal modifier* argument (`argm_tmp`): in the sentence "... continues for two (2) years", the phrase "two years" is a *temporal modifier* for the verb "continues". It produces the `arg(<"continues">, <"two years">, "argm_tmp")` relation in our database. In addition, we store an `overlap` relation between two entities if their positions in the original document have overlap.
4. **Cross-Sentence Entity Relations.** To capture relations across sentences, we utilize models recognizing synonyms and coreferences. We employ a similarity neural network to detect if two entities are synonyms (e.g. `synonym(<"cease", "stop">)`), and a coreference resolution network to detect if two entities are referring to the same concept. As an example, in the sentence "During the period of three years from the date hereof...", we have "the date" referencing the effective date defined in the previous sentence.

All of the relations extracted are associated with probabilities (the softmax output of each model). Although not explored in the scope of this paper, they allow subsequent differentiable and probabilistic reasoning systems to rank answers by probabilities and to back-propagate loss for data-efficient training.

Constructing Programmatic Queries We now describe how we construct programmatic queries for each of the 5 kinds of date-related questions in the CUAD dataset. In general, each query contains two parts of reasoning: *equivalence* and *association*. First, we want to find the words and phrases that are *equivalent* to the keyword in the question (e.g. "effective date"), using relations such as `overlap`, `synonym` and `coref`. Then, we can start from such words and phrases to find the *associated* dates, using SRL relations such as `arg`. Figure 1 shows a concrete example of a query applied to the text to derive the expected answer. The full set of queries and the helper rules are provided in appendix A.


```
eff_date_id(N) :- eff_date_verb(V), num(N, _, "DATE"), arg(V, N, "argm_tmp").
```

Figure 1: This rule extracts the effective date id, asking for the date that is associated to a verb which is equivalent to the keyword "effective date". Since the verb "is made" is a relaxed synonym of the keyword "effective date", and it has a temporal modifier "May 22, 2000", executing the rule yields the id of the phrase "May 22, 2000".

Domain knowledge such as "expiration date = effective date + term period" can be incorporated in the queries as well. With native support for the `Date` and `Period` data types, our relational database system enables arithmetic operations such as `M = D + P`, that performs addition between a period `P`


```
term_id(N)  :- term_verb(V), num(N, _, "PERIOD"), argm(V, N, "argm_tmp").
exp_date(M) :- eff_date_id(E), term_id(T), date_id(E, D), period_id(T, P), M = D + P.
```

Figure 2: The expiration date is calculated from the effective date plus the term length. The term period, "two (2) years", is extracted by the first rule, similar to Figure 1. Then, according to the second rule, expiration date is calculated with a native plus function operating over the effective date and the term length. Specifically, the date "May 22, 2000" plus the period "two (2) years" gives us the result "May 22, 2002".

and a date `D`. We employ the Magic-Set Transformation [2] to improve the efficiency of the query execution. This optimization cuts down the execution time from hours to less than 10 seconds.

4 Evaluation

Dataset. We evaluate our approach on date-related questions in the CUAD dataset: agreement date, effective date, expiration date, notice period before the expiration date, and the renewal term. Each task contains a legal document, a question, and a ground truth answer with its explanation. We ensure that no invalid answers, such as "[]/[]/[]" or empty labels exist in our dataset. In total, we have collected 1,342 annotations over 511 legal documents for the 5 kinds of questions.

Metrics. We propose a different metric for fine-grained date comparison. The original CUAD paper adopts the Jaccard similarity coefficient to compare the predicted string and the explanation string. However, in date calculation questions, an explanation may not directly reflect the real answer. Therefore, we use an off-the-shelf date parser to extract the numerical form of the ground truth gt_n and the prediction $\hat{y}_n$. If gt_n is of type `Date`, the accuracy is 1 if gt_n and y_n are exact match; if gt_n is of type `Period`, the accuracy is 1 if gt_n and y_n are within ϵ error bound.

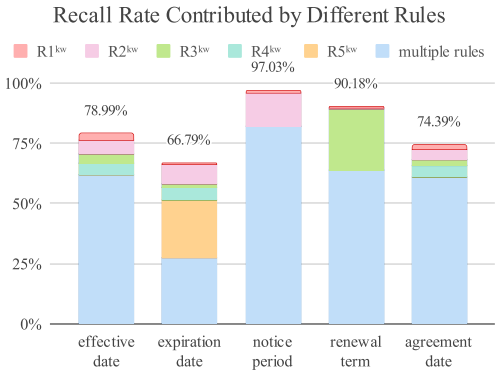

Figure 3: Recall on the CUAD dataset. R_i^{kw} represents the ith rule of the question kw. The "multiple rules" category covers tasks that multiple rules can solve, while the other categories represent tasks that can only be solved by the specific rule. The full list of R_i^{kw} is provided in the appendix.

Algorithm setup. Our algorithm heavily relies on pre-trained neural networks. We select the best performing neural networks from spaCy and allenNLP libraries. Specifically, we use the "en_core_web_lg" network in spaCy [11] to perform tokenization and similarity checks; state-of-the-art coreference resolution network [12]; ELMO-NER for named entity recognition [17]; and SRL-BERT for semantic role labeling [19]. We set the ϵ comparing two periods to be 0.01. We construct the 18 rules by analyzing 100 tasks in the dataset.

Results. Figure 3 shows the evaluation results on the CUAD dataset. Overall, our approach achieves a 77.7% recall rate on the dataset without fine-tuning. For instance, the rule "expiration date = effective date + term length" succeeds in calculating 64 unique solutions for the question "expiration date." This means more than 24% of expiration date's answers do not directly occur in the passage. Instead, date calculation is required to solve these problems.

5 Conclusion and Future Work

We introduced a neural symbolic approach to perform numerical reasoning over legal contracts. The natural language processing networks convert the legal documents into a database, while a Datalog engine performs high-level reasoning atop the database. Our approach can capture the semantics of legal documents, perform a precise calculation over numerical constructs, and maintain the explainability of the reasoning process. In the future, we plan to fine-tune the pre-trained neural networks through a differentiable probabilistic Datalog engine. Another interesting direction we intend to explore is to automatically generate logic rules from natural language questions.

References

[1] ANONYMOUS., A. Scallop: From probabilistic deductive databases to scalable differentiable reasoning.

[2] BALBIN, I., PORT, G. S., RAMAMOHANARAO, K., AND MEENAKSHI, K. Efficient bottom-up computation of queries on stratified databases. *J. Log. Program. 11*, 3–4 (Aug. 1991), 295–344.

[3] BROWN, T. B., MANN, B., RYDER, N., SUBBIAH, M., KAPLAN, J., DHARIWAL, P., NEELAKANTAN, A., SHYAM, P., SASTRY, G., ASKELL, A., AGARWAL, S., HERBERT-VOSS, A., KRUEGER, G., HENIGHAN, T., CHILD, R., RAMESH, A., ZIEGLER, D. M., WU, J., WINTER, C., HESSE, C., CHEN, M., SIGLER, E., LITWIN, M., GRAY, S., CHESS, B., CLARK, J., BERNER, C., MCCANDLISH, S., RADFORD, A., SUTSKEVER, I., AND AMODEI, D. Language models are few-shot learners. *CoRR abs/2005.14165* (2020).

[4] CHALKIDIS, I., FERGADIOTIS, M., MALAKASIOTIS, P., ALETRAS, N., AND ANDROUTSOPOULOS, I. LEGAL-BERT: the muppets straight out of law school. *CoRR abs/2010.02559* (2020).

[5] CHEN, K., XU, W., CHENG, X., XIAOCHUAN, Z., ZHANG, Y., SONG, L., WANG, T., QI, Y., AND CHU, W. Question directed graph attention network for numerical reasoning over text. *CoRR abs/2009.07448* (2020).

[6] DE RAEDT, L., KIMMIG, A., AND TOIVONEN, H. Problog: A probabilistic prolog and its application in link discovery. In *Proceedings of the 20th International Joint Conference on Artifical Intelligence* (San Francisco, CA, USA, 2007), IJCAI'07, Morgan Kaufmann Publishers Inc., p. 2468–2473.

[7] DENG, X., AWADALLAH, A. H., MEEK, C., POLOZOV, O., SUN, H., AND RICHARDSON, M. Structure-grounded pretraining for text-to-sql. *CoRR abs/2010.12773* (2020).

[8] DEVLIN, J., CHANG, M., LEE, K., AND TOUTANOVA, K. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR abs/1810.04805* (2018).

[9] DUA, D., WANG, Y., DASIGI, P., STANOVSKY, G., SINGH, S., AND GARDNER, M. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. In *Proc. of NAACL* (2019).

[10] HENDRYCKS, D., BURNS, C., CHEN, A., AND BALL, S. CUAD: an expert-annotated NLP dataset for legal contract review. *CoRR abs/2103.06268* (2021).

[11] HONNIBAL, M., MONTANI, I., VAN LANDEGHEM, S., AND BOYD, A. spaCy: Industrial-strength Natural Language Processing in Python, 2020.

[12] LEE, K., HE, L., LEWIS, M., AND ZETTLEMOYER, L. End-to-end neural coreference resolution. In *EMNLP* (2017).

[13] MANHAEVE, R., DUMANCIC, S., KIMMIG, A., DEMEESTER, T., AND RAEDT, L. D. Deepproblog: Neural probabilistic logic programming. *CoRR abs/1805.10872* (2018).

[14] MINERVINI, P., BOSNJAK, M., ROCKTÄSCHEL, T., AND RIEDEL, S. Towards neural theorem proving at scale. *CoRR abs/1807.08204* (2018).

[15] MINERVINI, P., BOSNJAK, M., ROCKTÄSCHEL, T., RIEDEL, S., AND GREFENSTETTE, E. Differentiable reasoning on large knowledge bases and natural language. *CoRR abs/1912.10824* (2019).

[16] MINERVINI, P., RIEDEL, S., STENETORP, P., GREFENSTETTE, E., AND ROCKTÄSCHEL, T. Learning reasoning strategies in end-to-end differentiable proving. *CoRR abs/2007.06477* (2020).

[17] PETERS, M. E., AMMAR, W., BHAGAVATULA, C., AND POWER, R. Semi-supervised sequence tagging with bidirectional language models. *CoRR abs/1705.00108* (2017).

[18] RAEDT, L. D., DUMANCIC, S., MANHAEVE, R., AND MARRA, G. From statistical relational to neuro-symbolic artificial intelligence. *CoRR abs/2003.08316* (2020).

[19] SHI, P., AND LIN, J. Simple BERT models for relation extraction and semantic role labeling. *CoRR abs/1904.05255* (2019).

[20] SI, X., RAGHOTHAMAN, M., HEO, K., AND NAIK, M. Synthesizing datalog programs using numerical relaxation. *CoRR abs/1906.00163* (2019).

[21] SUN, Y., WANG, S., FENG, S., DING, S., PANG, C., SHANG, J., LIU, J., CHEN, X., ZHAO, Y., LU, Y., LIU, W., WU, Z., GONG, W., LIANG, J., SHANG, Z., SUN, P., LIU, W., OUYANG, X., YU, D., TIAN, H., WU, H., AND WANG, H. ERNIE 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation. *CoRR abs/2107.02137* (2021).

[22] THORNE, J., YAZDANI, M., SAEIDI, M., SILVESTRI, F., RIEDEL, S., AND HALEVY, A. Y. Neural databases. *CoRR abs/2010.06973* (2020).

[23] WANG, B. Mesh-Transformer-JAX: Model-Parallel Implementation of Transformer Language Model with JAX. https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

[24] YU, T., ZHANG, R., YANG, K., YASUNAGA, M., WANG, D., LI, Z., MA, J., LI, I., YAO, Q., ROMAN, S., ZHANG, Z., AND RADEV, D. R. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *CoRR abs/1809.08887* (2018).

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We have future work to improve our method.
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [Yes]
   (b) Did you include complete proofs of all theoretical results? [Yes] In appendix **??**
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] In supplemental material
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] In supplemental material
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] At the beginning of evaluation section.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [N/A]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] In supplemental material.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We generate synthetic dataset.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] We generate synthetic dataset.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Rules

We manually write down the rules by analyze 100 question and answer pairs. Here is the set of rules that leads to the final results.

Rule Description	The date that is equivalent to the keyword "effective date".
Datalog Query	`effective_date_id(NID) :- effective_date_noun(NID), num(NID, _, "DATE").`

Table 3: Effective Date & Agreement Date, Rule 1

Rule Description	The date that is associated with equivalents of "effective date".
Datalog Query	`effective_date_id(NID) :- effective_date_verb(VID), num_id(NID, _, "DATE"), argm(VID, NID, "argm_tmp").`

Table 4: Effective Date & Agreement Date, Rule 2

Rule Description	The date overlaps with the equivalents of the keyword "effective date".
Datalog Query	`effective_date_id(NID) :- effective_date_noun(NID1), equiv_date(NID1, NID).`

Table 5: Effective Date & Agreement Date, Rule 3

Rule Description	The date overlaps with the phrase associated with equivalence of "effective date".
Datalog Query	`effective_date_id(NID) :- effective_date_verb(VID), equiv_date(NID1, NID), argm(VID, NID1, "argm_tmp").`

Table 6: Effective Date & Agreement Date, Rule 4

Rule Description	The date that is equivalent to the keyword "expiration date".
Datalog Query	`expiration_date_id(NID) :- expiration_noun(NID), num(NID, _, "DATE").`

Table 7: Expiration Date, Rule 1

Rule Description	The date that is associated with equivalents of "expiration date".
Datalog Query	`expiration_date_id(NID) :- expiration_verb(VID), num(NID, _, "DATE"), argm(VID, NID, "argm_tmp").`

Table 8: Expiration Date, Rule 2

Rule Description	The date overlaps with the equivalents of the keyword "expiration date".
Datalog Query	`expiration_date_id(NID) :- expiration_noun(NID1), equiv_date(NID1, NID).`

Table 9: Expiration Date, Rule 3

Rule Description	The date overlaps with the phrase associated with equivalence of "expiration date".
Datalog Query	`expiration_date_id(NID) :- expiration_verb(VID), equiv_date(NID1, NID), argm(VID, NID1, "argm_tmp").`

Table 10: Expiration Date, Rule 4

Rule Description	Expiration date = effective date + expiration date.
Datalog Query	`expiration_date_id(M) :- effective_date_id(NID), date_id(NID, D), period_id(PID, P), term_id(PID), M = D + P.`

Table 11: Expiration Date, Rule 4

Rule Description	The period that is associated with equivalents of "term".
Datalog Query	`renewal_term_id(NID) :- term_verb(VID), num(NID, _, "PERIOD"), argm(VID, NID, "argm_tmp").`

Table 12: Renewal Term, Rule 1

Rule Description	The date that is associated with equivalents of "term".
Datalog Query	`renewal_term_id(NID) :- term_verb(VID), num_id(NID, _, "DATE"), argm(VID, NID, "argm_tmp").`

Table 13: Renewal Term, Rule 2

Rule Description	The period overlaps with the phrase associated with equivalents of "term".
Datalog Query	`renewal_term_id(NID) :- term_verb(VID), equiv_period(NID1, NID), argm(VID, NID1, "argm_tmp").`

Table 14: Renewal Term, Rule 3

Rule Description	The date overlaps with the phrase associated with equivalents of "term".
Datalog Query	`renewal_term_id(NID) :- term_verb(VID), equiv_date(NID1, NID), argm(VID, NID1, "argm_tmp").`

Table 15: Renewal Term, Rule 4

Rule Description	The period associated with both "termination" and "notice".
Datalog Query	`notice_period_id(NID) :- notice_verb(VID), termination_verb(VID), num(NID, _, "PERIOD"), argm(VID, NID, "argm_tmp").`

Table 16: Notice Period, Rule 1

Rule Description	The date associated with both "termination" and "notice".
Datalog Query	`notice_period_id(NID) :- notice_verb(VID), termination_verb(VID), num(NID, _, "DATE"), argm(VID, NID, "argm_tmp").`

Table 17: Notice Period, Rule 2

Rule Description	The period overlapped with the phrase that associates with both "termination" and "notice".
Datalog Query	`notice_period_id(NID) :- notice_verb(VID), termination_verb(VID), equiv_period(NID1, NID), argm(VID, NID1, "argm_tmp").`

Table 18: Notice Period, Rule 3

Rule Description	The date overlapped with the phrase that associates with both "termination" and "notice".
Datalog Query	`notice_period_id(NID) :- notice_verb(VID), termination_verb(VID), equiv_date(NID1, NID), argm(VID, NID1, "argm_tmp").`

Table 19: Notice Period, Rule 4

Here are the helper functions.

```
// Known rules for query construction
decl equiv(Int, Int).
equiv(A, B) :- coref(A, B).
equiv(A, B) :- synonym(A, B).
equiv(A, B) :- equiv(B, A).
equiv(A, B) :- equiv(A, C), equiv(C, B).

decl equiv_date(Int, Int).
decl equiv_period(Int, Int).
equiv_date(A, B) :- overlap(A, B), num_id(B, _, "DATE").
equiv_period(A, B) :- overlap(A, B), num_id(B, _, "PERIOD").

// helpers

decl vid(Int, String).
vid(VID, V) :- argm(VID, _, _), noun_id(VID, V, _).
vid(VID, V) :- argm(VID, _, _), num_id(VID, V, _).

decl eq_start_verb(Int).
eq_start_verb(VID) :- verb_id(VID, "start").
eq_start_verb(VID1) :- verb_id(VID, "start"), equiv(VID, VID1).

decl eq_terminate_verb(Int).
eq_terminate_verb(VID) :- verb_id(VID, "terminate").
eq_terminate_verb(VID1) :- verb_id(VID, "terminate"), equiv(VID, VID1).

decl eq_effective_date_noun(Int).
eq_effective_date_noun(NID) :- noun_id(NID, "effective date", _).
eq_effective_date_noun(NID1) :- noun_id(NID, "effective date", _), equiv(NID, NID1).

decl eq_term_noun(Int).
eq_term_noun(NID) :- noun_id(NID, "term", _).
eq_term_noun(NID1) :- noun_id(NID, "term", _), equiv(NID, NID1).

decl eq_termination_noun(Int).
eq_termination_noun(NID) :- noun_id(NID, "terminate date", _).
eq_termination_noun(NID1) :- noun_id(NID, "terminate date", _), equiv(NID, NID1).

decl eq_notice_noun(Int).
eq_notice_noun(NID) :- noun_id(NID, "notice", _).
eq_notice_noun(NID) :- noun_id(NID, "notice period", _).
eq_notice_noun(NID1) :- noun_id(NID, "notice", _), equiv(NID, NID1).
eq_notice_noun(NID1) :- noun_id(NID, "notice period", _), equiv(NID, NID1).

decl termination_verb(Int).
termination_verb(VID) :- verb_id(VID, V), eq_termination_noun(NID), argm(VID, NID, _).
termination_verb(VID) :- verb_id(VID, V), eq_termination_noun(VID).
termination_verb(VID) :- eq_terminate_verb(VID).

decl effective_date_verb(Int).
effective_date_verb(VID) :- verb_id(VID, V), eq_effective_date_noun(NID),
                            argm(VID, NID, _).
effective_date_verb(VID) :- verb_id(VID, V), eq_effective_date_noun(VID).
effective_date_verb(VID) :- eq_start_verb(VID).

decl term_verb(Int).
term_verb(VID) :- verb_id(VID, V), eq_term_noun(NID), argm(VID, NID, _).
```

```
term_verb(VID) :- verb_id(VID, "remain").
term_verb(NID) :- verb_id(VID, "remain"), equiv(VID, NID).

decl notice_verb(Int).
notice_verb(VID) :- verb_id(VID, V), eq_notice_noun(NID), argm(VID, NID, _).
```