Augmenting Operations Research with Auto-Formulation of Optimization Models From Problem Descriptions

Rindra Ramamonjison[1][†] **Haley Li**[*] **Timothy Yu**[†]

Shiqi He[*] **Vishnu Rengan**[*] **Amin Banitalebi-Dehkordi**[†]

Zirui Zhou[†] **Yong Zhang**[†]

Abstract

We describe an augmented intelligence system for simplifying and enhancing the modeling experience for operations research. Using this system, the user receives a suggested formulation of an optimization problem based on its description. To facilitate this process, we build an intuitive user interface system that enables the users to validate and edit the suggestions. We investigate controlled generation techniques to obtain an automatic suggestion of formulation. Then, we evaluate their effectiveness with a newly created dataset of linear programming problems drawn from various application domains. Code and data are available at https://github.com/nl4opt/nl4opt-competition

1 Introduction

Many real-world decision-making problems can be formulated and solved as mathematical optimization problems. The field of operations research (OR) has seen success in applications ranging from increasing bike-share ridership and efficiency (Beairsto et al., 2021; Ma et al., 2016), managing wastewater collection and treatment systems (Tao et al., 2020), to finding a revenue-maximizing pricing strategy (Bitran and Caldentey, 2016). In fact, optimization solvers can tackle different types of problems as they are powered by efficient algorithms such as the simplex method (Nash, 2000) or interior-point methods (Karmarkar, 1984).

However, modeling a problem into a proper formulation is a complex and time-consuming process. First, a domain expert must describe the problem and identify its variables, parameters, objective and constraints. Then, an OR expert needs to translate this problem description into a precise formulation using a modeling language, thus making the process inefficient and limits the accessibility of the solvers to non-experts (Hürlimann, 2013).

We propose an augmented intelligence system to simplify and enhance the modeling process. We partially automate the process using NLP models to suggest a formulation that the users can validate or edit the suggestions using an intuitive interface. Partial automation avoids the manual writing of the formulation by using a modeling language, thereby reducing the time and expertise to build optimization models. The intuitive interface also makes solvers more accessible to non-technical users.

From an NLP perspective, there are many challenges to parsing an optimization problem's formulation from its natural language description:

• **Limited dataset.** The strenuous nature of modeling makes the cost of creating and labeling a large dataset prohibitive. Thus, efficient few-shot semantic parsers must be trained in a low-resource setting. Therefore, the solution must be built leveraging methods that excel on a limited training dataset.

• **Document-level input.** Most semantic parsers operate primarily at the sentence level. In contrast, long paragraph inputs describe the many variables, parameters, and constraints of optimization problems. Also, the parsing task involves a high level of compositionality and ambiguity.

• **Context-free output.** The outputs of most semantic parsing tasks share some contextual information with the inputs (e.g. database table or column names in SQL queries). In contrast, our task has a context-free tabular format, which makes it difficult to align the input-output pair.

• **Domain-agnostic parsing.** Finally, OR can tackle a diverse range of applications (Williams, 2013). Hence, the semantic parser must generalize well not only to new problem instances but also new application domains.

In this paper, we describe the underlying system that addresses these challenges and that enables

---

1 rindranirina.ramamonjison@huawei.com
† Huawei Technologies Canada
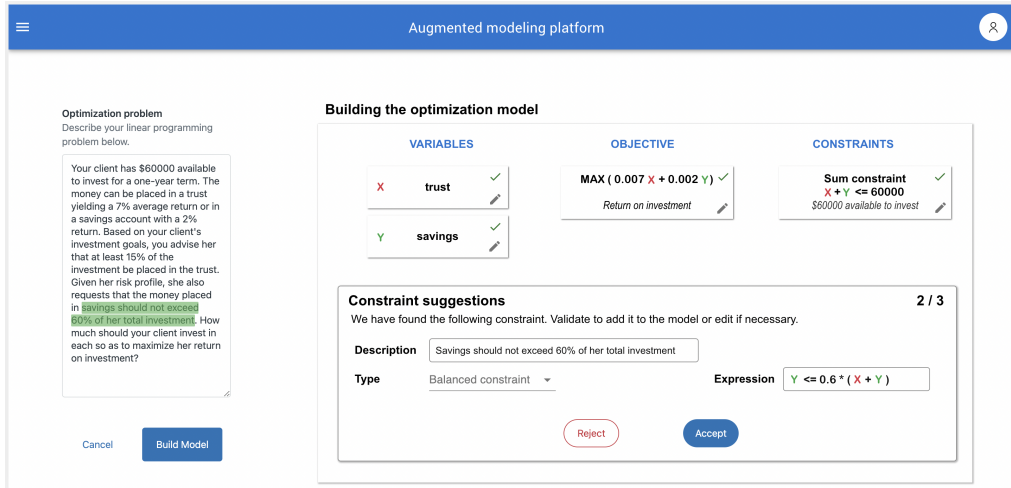* University of British Columbia, Vancouver

Figure 1: Augmented modeling platform.

the augmented modeling application. Our contributions are:

1. A novel augmented intelligence application that simplifies and enhances the modeling process.

2. New controllable generation methods for parsing the formulation of optimization problems from its natural language description.

3. The first dataset on linear programming word problems, with which we test and analyze the effectiveness of the methods for this emerging application.

2 Related Work

Augmented intelligence systems. Augmented intelligence systems use AI to assist (and not replace) the users in performing certain tasks. These systems could improve the experience or creativity of users in artistic applications such as story writing (Clark et al., 2018), music composition (Huang et al., 2020), poetry composition (Uthus et al., 2022), and sketching (Fan et al., 2019). This approach has also seen success in more task-oriented applications by helping teachers to grade homework efficiently (Malik et al., 2021), salespeople to summarize sales calls (Asi et al., 2022), and improving pneumonia diagnostic accuracy (Patel et al.). We adopt a similar approach for OR and focus on improving the modeling process.

Semantic parsing and generation. Semantic parsing maps natural language utterances into a machine-interpretable representation (Kamath and Das, 2019). This mapping has been extensively studied for output representations such as

SQL queries (Gan et al., 2020), Unix commands (Bharadwaj and Shevade, 2021), or logical forms for querying a knowledge base (Dong and Lapata, 2016). We tackle a different and challenging semantic parsing task as explained in Section 1.

Building on the success of attention models in sequence-to-sequence tasks (Sutskever et al., 2014; Luong et al., 2015), encoder-decoder architectures have been adopted for designing semantic parsers (Dong and Lapata, 2016, 2018; Wang et al., 2020). We propose using an intermediate representation (IR) that serves as a bridge between natural language and the canonical output format. Our two-stage mapping strategy is different from (Dong and Lapata, 2018), which initially generates a sketch of the query and then fills out the slots. In contrast to prior methods of constrained decoding (Hokamp and Liu, 2017; Scholak et al., 2021), our approach uses a simple beam search and leverages a prompt-guided generation and copying mechanism to guide the decoding.

Datasets on Mathematical World Problems (MWP). Recent works have studied the use of NLP models to automatically solve MWP (Wang et al., 2017; Ughade and Kumbhar, 2019) Most existing MWP datasets have focused on returning the solutions of elementary arithmetic problems (Roy and Roth, 2015; Koncel-Kedziorski et al., 2016) and algebra problems (Kushman et al., 2014; Huang et al., 2016). More challenging benchmarks have been recently proposed such as SVAMP (Patel et al., 2021), MATH (Hendrycks et al., 2021) and GSM8K (Cobbe et al., 2021). In contrast, we build the first linear programming word problems (LPWP) dataset and evaluate methods of generating
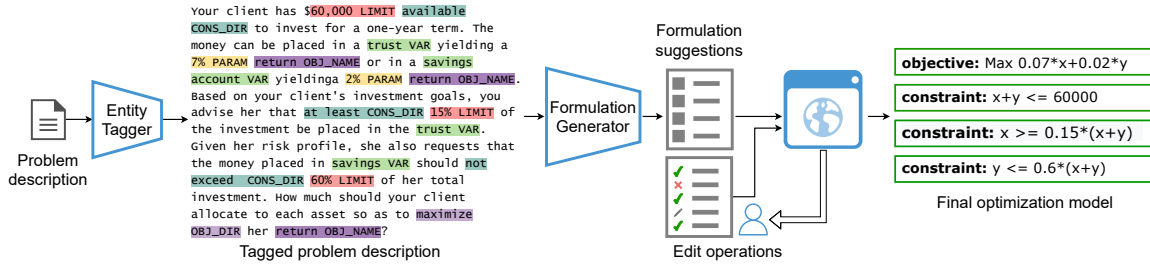
Figure 2: System diagram for augmented modeling framework.

the formulations as inputs to optimization solvers which can efficiently return an optimal solution.

3 Augmented Modeling Interface

We present an interactive system that enables users to model an optimization problem in collaboration with an AI system. To use this application, the user first describes the problem using natural language. Then, the system will suggest to the user the formulation of the optimization model including the decision variables, objective, and problem constraints. The system suggests the components of the formulation one at a time allowing the users to accept, reject, or edit the suggestions.

An example scenario of a portfolio optimization problem is shown in Figure 1. Here, the system suggested the formulation of a balance constraint given the description "*savings should not exceed 60% of her total investment*". Had the system presented an incorrect formulation (e.g. an upper bound constraint), the user can change the type using the interface and the system reformulates the constraint expression automatically. The user can also edit the description, which will be stored as metadata of the model. In the same fashion, the user can add variables or constraints manually or edit the problem description to update the model. When the user is satisfied with the formulation, the system forwards it to an optimization solver, which then returns either an optimal solution of the problem to the user or warnings for some infeasible constraints.

Figure 2 shows an overview of the underlying auto-formulation system. It consists of an entity tagger, a formulation generator, and an augmented modeling interface. Given a problem description, the entity tagger labels the keywords that indicate the components of the optimization problem. For example, "return" and "at least" are tagged as an objective name and constraint direction, respectively. Then, the formulation generator uses the text description and the corresponding tagged entities to generate the formulation suggestions, which

are then presented to the user by the augmented modeling interface. In our implementation (experimental settings in the Appendix), we used an XLM-RoBERTa pre-trained transformer and fine-tuned it for entity recognition using the dataset described in Section 5.2.

4 OptGen: Controllable Generation of Optimization Formulation

Here, we present the methods behind our model OptGen, for generating the suggested formulation.

4.1 Two-stage mapping approach

It is difficult to directly map the problem description p to the formulation f due to the characteristics of the input-output pair (p, f) as mentioned in Section 1. First, the input document p can be unstructured and ambiguous especially when it describes many constraints of different types. From the input, we must precisely extract the canonical representation f of an optimization formulation.

As shown on the right of Figure 3, this canonical representation is a context-free table in which the column header is either a variable symbol or a constraint's right-hand-side (rhs) limit. Each table row contains the parameters of the objective function or constraint. As a result, the canonical formulation f is context-free since it abstracts away the contextual information of p.

Instead, we adopt a two-stage mapping $p \mapsto r \mapsto f$ to bridge the gap between the natural language input and context-free formulation.

Text-to-IR mapping. We first define an intermediate representation (IR) r of the problem to simplify the parsing. As illustrated in Figure 3, a Text-to-IR mapping model generates a set of entity-typed declarations $\{D_i\}_{i=1}^n$ defined in an extended markup format to simplify its parsing. Note that other formats can be used for the IR (e.g. a format defined by a context-free grammar). Each declaration D_i is a sequence of tokens that represents
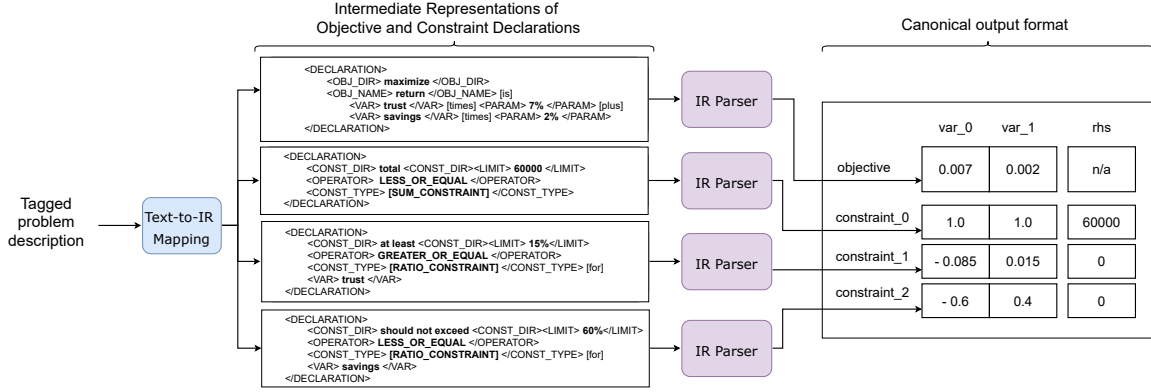
Figure 3: Overview of our formulation generation approach.

a typed and structured representation of either the optimization objective or a constraint. Each D_i is defined based on a predefined template of the different objective and constraint types. The tokens defining each declaration are wrapped in special tags. For example, `<OBJ_DIR> maximize </OBJ_DIR>` and `<OBJ_NAME> return </OBJ_NAME>` define the direction and the name of the objective. Since these tokens are derived from the input description, the IR preserves the context of the problem. We describe the grammar of the IR in the Appendix.

IR parsing. An IR parser converts each IR declaration to the canonical format. We use an XML parser and apply simple transformations to convert numerical words to decimal numbers and to follow the following conventions. First, the canonical format always "minimize" a cost function. When the objective direction is "maximize," we instead change the sign of each objective parameter. Similarly, each inequality constraint must have a `LESS_OR_EQUAL` operator and convert each inequality constraint to the form $a^\top x \leq b$.

4.2 Prompt-guided generation model

We use an autoregressive model that is built upon the BART language model (Lewis et al., 2020). A prompt-guided generation is proposed to improve the accuracy of the Text-to-IR mapping. The idea is to decode the declarations of the IR one by one by using a declaration prompt to focus the generation. A declaration prompt is a prefix of the IR declaration of an objective or a constraint. For an objective, the prompt is composed of the entity tokens of the objective's direction and name. Similarly, the prompt for a constraint is defined by the entity tokens of the constraint direction. These tokens are obtained from the output of the Entity Tagger model shown in Figure 2. The declaration

prompt is added to the input text of the encoder. The role of the prompt is to provide contextual triggers for the decoder to focus on the relevant parts of the declaration to be generated. As illustrated in Figure 4, the model is trained to generate the IR of the declaration based on the declaration prompt and the problem description.

One key requirement of the Text-to-IR mapping is the ability to extract the variable names and data parameters from the descriptions and copy these important mentions from the input description into the output IR of the decoder. To augment the capability of BART encoder-decoder model, we leverage a copy mechanism that computes the probability distribution P_{copy} over the input tokens using cross-attention scores. The copy distribution is calculated at each time step t by taking the mean of the decoder's cross-attention scores across all attention heads as follows:

$$e_{t,i} = \frac{(W_s s_t)^T W_h h_i}{\sqrt{d_k}}$$

$$\alpha_{t,i} = \text{softmax}(e_{t,i})$$

$$P_{\text{copy}} = \frac{1}{n_H} \sum_i \alpha_{t,i}$$

where W_s and W_h are the projection matrices for the encoder and decoder. Then, s_t, h_i, and n_H are the decoder hidden state at time step t, the encoder hidden state for the attention head i, and the number of heads respectively.

We add the special tokens of the IR into the BART target vocabulary and mask out any vocabulary words that are not present in the source input. Following (See et al., 2017), we use a soft switch $p_{\text{gen}} \in [0, 1]$ to choose between generating a word from the vocabulary by sampling from P_{vocab}, or copying a word from the input sequence by sampling from P_{copy}. Thus, the final probability distri-
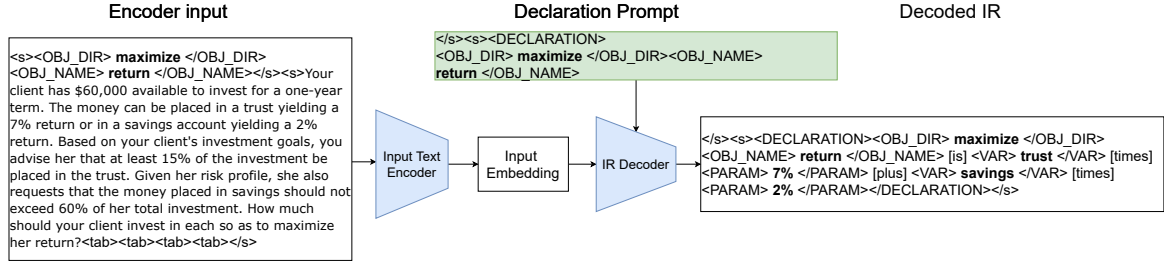
Figure 4: Description of prompt-guided generation method.

bution of a word w is given by:

$$P(w) = p_{\text{gen}} P_{\text{vocab}}(w) + (1 - p_{\text{gen}}) P_{\text{copy}}(w)$$

and is used to compute the loss for timestep t as the negative loglikelihood of the target word y_t for that timestep during training. Then, we average the loss over all time steps. As a result, the model is trained to produce tokens from either the IR vocabulary or the input problem description.

5 Experiments

5.1 Dataset

Dataset description. We curated a first-ever LPWP dataset[1] and use it to train and evaluate our methods. The dataset contains a collection of LP problems of the form:

$$\min_{\mathbf{x} \in \mathbb{R}^n} \mathbf{c}^\top \mathbf{x} \;\; \text{s.t.} \;\; \mathbf{a}_i^\top \mathbf{x} \le b_i, \;\; i = 1, \ldots, m$$

where $\mathbf{c}$ and $\mathbf{a}_i$ represent the parameters of the objective and the i-th constraint, respectively. b_i is the right-hand-side (rhs) limit, and the goal is to find $\mathbf{x}$ that minimizes the objective value. The objective and constraint functions are linear with respect to variables in the LP problems. Each example has a text description of the problem and is annotated with the IR, math representation, and canonical formulation as shown in Figure 3. Table 1 summarizes the statistics of the dataset. More details and examples of the dataset are provided in the Appendix.

The dataset contains 1101 LP problems from the source domain (advertising, investment, sales) and target domain (production, science, transportation). The train, dev, and test splits contain 713, 99, and 289 samples, respectively. The training split is comprised of solely of samples from the source domain whereas the dev and test splits contain samples from both source and target domains with a source-to-target domain ratio of 1:3.

[1]We plan to release this dataset and open-source the code to the research community for future research.

Number of Problems	1101
Number of Declarations	4216
Number of Constraint Types	6
Average Number of Variables	2.08
Average Number of Constraints	2.83

Table 1: Summary statistics of the LPWP dataset.

Dataset creation and quality control. The dataset was created from scratch and annotated internally by a team of 20 researchers and engineers with different levels of expertise in OR/NLP. The process included 3 stages: (1) problem creation, (2) NER annotation, and (3) REL annotation and declaration generation. Each stage was followed by rigorous quality checks and verification. The creation process is described in Figure 5.

Additional details of the dataset including the creation process, examples of the problem and their corresponding math formulation and IR, exclusion criteria, and inter-annotator agreement score, etc, are reported in the Appendix.

5.2 Results and Discussions

Baseline and metrics. We conducted experiments using the dataset described in Section 5.1. We use BART model (Lewis et al., 2020) as baseline for the two-stage mapping approach. In addition, we adopt the Text-to-Table (T2T) model (Wu et al., 2021) as a baseline for the direct approach that directly produces the canonical form. An example of its output is shown in Table 3 in the Appendix.

For evaluation, we measure the declaration-level mapping accuracy on the canonical formulation defined as:

$$\text{Acc} = 1 - \frac{\sum_{i=1}^{N} \min\{\text{FP}_i + \text{FN}_i, D_i\}}{\sum_{i=1}^{N} D_i},$$

where for a given problem i, D_i is the number of ground-truth declarations, false positives FP_i is the number of non-matched predicted declarations, and false negatives FN_i is the number of excess unmatched ground-truth declarations. In other words,
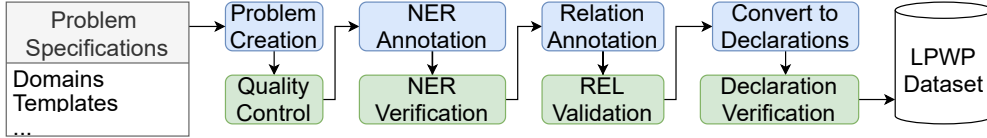
Figure 5: Overview of dataset creation.

FN_i is only non-zero when there are more ground-truth declarations than predicted declarations. The min is to prevent negative accuracy and overpenalization on single problems.

Main Results. Table 2 summarizes our results on the source and target domains. The BART baseline achieved the worst performance on all domains. In fact, BART hallucinated by producing too many constraints or many wrong parameters. Next, the direct mapping approach of T2T model achieved the highest accuracy of 88% on the Source domain but generalized poorly on out-of-domain (Target) test set. While it seemed to learn the task, T2T overfit to the Source training data. In contrast, our proposed OptGen achieved an absolute +18% accuracy improvement over direct mapping (T2T) on the Target domain. It was able to generalize better than the other models and produced the fewest errors on the problem structure and parameters. Our two-stage approach helped in this regard while the T2T must directly learn to convert to the canonical form. A detailed error analysis can be found in Figures 8 and 9 in the Appendix.

Ablation study. We also analyzed the importance of the individual methods used in our model in Table 2. It shows that the copy mechanism is important for accurately mapping the description to the equivalent formulation. Without the copy mechanism, we see significant accuracy drops of about 5% and 30% on Source and Target respectively. We show a qualitative comparison of generated IR formulations for two LP problems in Tables 12 and 13 in the Appendix. While our model could perfectly generate the correct representations, the model without copy mechanisms produced many errors. For example, it hallucinated the wrong constraint limits or detected the wrong constraint types. The prompt-guided generation method also led to slightly better performance on the Target domain. Finally, we noticed +6% improvement for T2T on the target domain when using contextual prompts.

These results show the importance of using controlled generation techniques for learning the syntax and grammar of the target IR language and for

Method	Source	Target	Sci	Prod	Trans
T2T	0.83	0.39	0.46	0.37	0.37
T2T + Prompt	**0.88**	0.45	0.49	0.46	0.40
BART	0.52	0.20	0.21	0.19	0.20
OptGen w/o copy	0.55	0.34	0.38	0.32	0.33
OptGen w/o prompt	0.58	0.61	**0.64**	0.63	0.57
OptGen	0.60	**0.63**	0.60	**0.66**	**0.64**

Table 2: Results for each model on the declaration-level mapping accuracy metric. Source consists of samples from the source domain test split. Target is a weighted mean (by number of declarations) of the science (Sci), production (Prod), and transportation (Trans) target domains test split.

accurately mapping the input description to the IR formulation.

Limitations and future works. Our preliminary results show the potential and the importance of controllable generation methods for enabling the augmented modeling system. While the proposed OptGen generation model was shown to generalize better than baseline models, its accuracy performance should still be enhanced further by improving the decoding method or by using edit-based models (Malmi et al., 2022) to automatically correct the erroneous parts of the formulation. As future work, we will conduct human evaluation of the augmented system by measuring the efficiency improvement perceived by real users. As the current dataset only covers LP problems, we will also expand it to cover other types of problems such as mixed-integer programs, which have different types of constraints. Other directions can also be explored to build more data-efficient methods.

6 Conclusion

We introduced an augmented modeling platform, in which users enter the descriptions of optimization problems and interact with an AI system to efficiently model their formulations. To this end, we described the underlying system for this emerging application and proposed controllable generation methods to enable it. We also created a training dataset of linear programming word problems to evaluate the effectiveness of the proposed methods. Our findings showed that the design of generation models and methods can have significant impact on the accuracy of the system's suggested formula-

tions and that the system should help the users to validate and edit the suggestions.

7 Ethics Statement

This augmented intelligence system is intended to parse the formulation of optimization problems from its natural language description to aid stakeholders in their decision-making. The dataset was created taking special care to exclude samples with inappropriate language or names of real people, products or companies. The harm to users resulting from incorrect parsing is limited. However, depending on the application, the system may be used in sensitive or critical applications, such as a power grid, flights scheduling, etc. In such cases, the solver should be used with caution and the modeling process should be validated by the domain expert. Finally, operations research has historically been applied in tactical military operations. We must understand the potential negative impact of misusing this technology for society at large and the users must seriously consider the ethical concerns related to military applications.

References

Abedelkadir Asi, Song Wang, Roy Eisenstadt, Dean Geckt, Yarin Kuper, Yi Mao, and Royi Ronen. 2022. An end-to-end dialogue summarization system for sales calls. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track*, pages 45–53, Hybrid: Seattle, Washington + Online. Association for Computational Linguistics.

Jeneva Beairsto, Yufan Tian, Linyu Zheng, Qunshan Zhao, and Jinhyun Hong. 2021. Identifying locations for new bike-sharing stations in glasgow: an analysis of spatial equity and demand factors. *Annals of GIS*, 0(0):1–16.

Shikhar Bharadwaj and Shirish Shevade. 2021. Explainable natural language to bash translation using abstract syntax tree. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 258–267, Online. Association for Computational Linguistics.

Gabriel R. Bitran and René A. Caldentey. 2016. An overview of pricing models for revenue management. *IEEE Engineering Management Review*, 44:134–134.

Elizabeth Clark, Anne Spencer Ross, Chenhao Tan, Yangfeng Ji, and Noah A. Smith. 2018. Creative writing with a machine in the loop: Case studies on slogans and stories. *23rd International Conference on Intelligent User Interfaces*.

Karl Cobbe, Vineet Kosaraju, et al. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

Li Dong and Mirella Lapata. 2016. Language to logical form with neural attention. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 33–43, Berlin, Germany. Association for Computational Linguistics.

Li Dong and Mirella Lapata. 2018. Coarse-to-fine decoding for neural semantic parsing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 731–742, Melbourne, Australia. Association for Computational Linguistics.

Judith E. Fan, Monica Dinculescu, and David Ha. 2019. Collabdraw: An environment for collaborative sketching with an artificial agent. In *Proceedings of the 2019 on Creativity and Cognition*, page 556–561, New York, NY, USA. Association for Computing Machinery.

Yujian Gan, Matthew Purver, and John R. Woodward. 2020. A review of cross-domain text-to-SQL models. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop*, pages 108–115, Suzhou, China. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. 2021. Measuring mathematical problem solving with the math dataset. *NeurIPS*.

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546, Vancouver, Canada. Association for Computational Linguistics.

Cheng-Zhi Anna Huang, Hendrik Vincent Koops, Ed Newton-Rex, Monica Dinculescu, and Carrie J. Cai. 2020. Ai song contest: Human-ai co-creation in songwriting.

Danqing Huang, Shuming Shi, Chin-Yew Lin, Jian Yin, and Wei-Ying Ma. 2016. How well do computers solve math word problems? large-scale dataset construction and evaluation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 887–896, Berlin, Germany. Association for Computational Linguistics.

Tony Hürlimann. 2013. *Mathematical modeling and optimization: an essay for the design of computer-based modeling tools*, volume 31. Springer Science & Business Media.

Aishwarya Kamath and Rajarshi Das. 2019. A survey on semantic parsing.

N. Karmarkar. 1984. A new polynomial-time algorithm for linear programming. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, STOC '84, page 302–311, New York, NY, USA. Association for Computing Machinery.

Rik Koncel-Kedziorski, Subhro Roy, Aida Amini, Nate Kushman, and Hannaneh Hajishirzi. 2016. MAWPS: A math word problem repository. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1152–1157, San Diego, California. Association for Computational Linguistics.

Nate Kushman, Yoav Artzi, Luke Zettlemoyer, and Regina Barzilay. 2014. Learning to automatically solve algebra word problems. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 271–281, Baltimore, Maryland. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Yingying Ma, Xiaoran Qin, Jianmin Xu, and Xiangli Zou. 2016. Research on pricing method of public bicycle service: A case study in guangzhou. In *IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*, pages 156–161.

Ali Malik, Mike Wu, Vrinda Vasavada, Jinpeng Song, Madison Coots, John Mitchell, Noah D. Goodman, and Chris Piech. 2021. Generative grading: Near human-level accuracy for automated feedback on richly structured problems. In *Proceedings of the 14th International Conference on Educational Data Mining, EDM 2021, virtual, June 29 - July 2, 2021*. International Educational Data Mining Society.

Eric Malmi, Yue Dong, Jonathan Mallinson, Aleksandr Chuklin, Jakub Adamek, Daniil Mirylenka, Felix Stahlberg, Sebastian Krause, Shankar Kumar, and Aliaksei Severyn. 2022. Text generation with text-editing models.

John C Nash. 2000. The (dantzig) simplex method for linear programming. *Computing in Science & Engineering*, 2(1):29–31.

Arkil Patel, Satwik Bhattamishra, and Navin Goyal. 2021. Are NLP models really able to solve simple math word problems? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2080–2094, Online. Association for Computational Linguistics.

Bhavik N. Patel, Louis Rosenberg, Gregg Willcox, David Baltaxe, Mimi Lyons, Jeremy Irvin, Pranav Rajpurkar, Timothy Amrhein, Rajan Gupta, Safwan Halabi, Curtis Langlotz, Edward Lo, Joseph Mammarappallil, A. J. Mariano, Geoffrey Riley, Jayne Seekins, Luyao Shen, Evan Zucker, and Matthew Lungren. Human–machine partnership with artificial intelligence for chest radiograph diagnosis. *npj Digital Medicine*, 2(1).

Subhro Roy and Dan Roth. 2015. Solving general arithmetic word problems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1743–1752, Lisbon, Portugal. Association for Computational Linguistics.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. Picard: Parsing incrementally for constrained auto-regressive decoding from language models.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NIPS'14, page 3104–3112, Cambridge, MA, USA.

Diana Qing Tao, Martin Pleau, et al. 2020. Analytics and Optimization Reduce Sewage Overflows to Protect Community Waterways in Kentucky. *Interfaces*, 50(1):7–20.

Shounaak Ughade and Satish Kumbhar. 2019. Survey on mathematical word problem solving using natural language processing. In *2019 1st International Conference on Innovations in Information and Communication Technology (ICIICT)*, pages 1–5.

David Uthus, Maria Voitovich, and R.j. Mical. 2022. Augmenting poetry composition with Verse by Verse. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Industry Track*, pages 18–26, Hybrid: Seattle, Washington + Online. Association for Computational Linguistics.

Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7567–7578, Online. Association for Computational Linguistics.

Yan Wang, Xiaojiang Liu, and Shuming Shi. 2017. Deep neural solver for math word problems. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 845–854, Copenhagen, Denmark. Association for Computational Linguistics.

H Paul Williams. 2013. *Model building in mathematical programming*. John Wiley & Sons.

Xueqing Wu, Jiacheng Zhang, and Hang Li. 2021. Text-to-table: A new way of information extraction.

8 Appendix

This section contains supplementary materials including qualitative comparisons, examples from the dataset, as well as additional details which were omitted from the main paper due to space limitations.

A More details on dataset description

In Table 4, we showcase the different constraint types, along with their use in a problem, their math formulation, and their IR. Tables 6, 7 and 8 provide examples showing the problem description, IR and the math formulations.

B More details on dataset creation

The LPWP dataset creation process consists of three stages.

B.1 Stage 1: Problem creation

First, we invited 15 researchers and engineers to create original LP problems. They created the problems following some specifications such as problem domains and templates. The problem domains include advertising, investment, sales, agriculture, manufacturing, transportation and health sciences. During the creation step, 5 additional people were tasked with performing quality control and giving guidance to the problem creators. This included screening for the exclusion criteria (e.g. use of inappropriate language or names of real people, products or companies) and making sure the produced problem followed the specified domain and template. In fact, we used these templates to ensure the diversity of the problem structure (e.g. number of constraints, objectives and constraint types) in the dataset. If a mistake was detected in the problem, the problem creator would have been asked to fix it and the correction was verified. We made sure that each problem was verified by at least two people.

B.2 Stage 2: NER Annotation

At this stage, annotators were required to locate and classify entities mentioned in the problem. There are in total 6 types of targeted entities: variable (VAR), parameter (PARAM), limit (LIMIT), constraint direction (CONST_DIR), objective direction (OBJ_DIR) and objective name (OBJ_NAME). We used the Prodigy annotation tool to annotate the problems. A screenshot of one annotation session is provided in Figure 6. We asked an additional annotator to resolve disagreements by reading the guideline thoughtfully and asking the annotators for clarifications if needed.

To ensure the quality of the NER annotations, four OR/NLP experts annotated more than 10% of the entire dataset, with an equal split between each domain, separately to compute the inter-annotator agreement. We measured an average pairwise micro-averaged F1 score of 97.7% for the inter-annotator agreement, showing the reliability of the annotation process.

B.3 Stage 3: Relation annotation and declaration generation

In the final stage, we annotated relations between entities for representing the objective and constraint declarations. Using a custom Prodigy annotation recipe, we integrated some validation checks to detect mistakes in the relation annotation period. These checks ensured that the annotator corrected all mistakes before proceeding to the next problem. Table 5 summarizes the relations required to represent each constraint type. The validation checks verified the number, labels, directions, and the entity labels of all relations that represent one constraint or objective. To increase the efficiency of the annotation process, we also provided guidelines and trained annotators to annotate each type of constraints. Similar to stage 2, we worked with annotators to resolve complex relations. A screenshot of a relation annotation session is provided in Figure 7.

After annotating the relations, we used a Python script to automatically convert the extracted relations into optimization declarations. A team of 5 people were asked to verify the correctness of the optimization declarations for all problems.

C Examples of target domain problems

Tables 9, 10 and 11 provide additional examples from the target domain: production, transportation, and sciences respectively.

D Predicted vs gold IR for target domain examples

Table 14-20 demonstrate the predictions of our model in comparison to the gold standard (ground-truth) for several examples from the target domain. We show examples for when the model produced the gold IR formulation in Table 14 and Table 15.

More importantly, we illustrate and analyze different errors when the model was used to parse examples from the target domain. For each example, detailed analysis of the errors are given in the captions of Table 16-20.

E   Qualitative comparison of test predictions

In Table 12 and 13, we qualitatively compare the generated IR of two different optimization problems. Our model perfectly matched the gold IR for both problems. When the copy mechanism is not used, the model made few errors when generating the constraint declarations. In the first example (Table 12), the model hallucinates the wrong constraint limits in the first two constraints. Furthermore, the constraint type is invalid for the second constraint. Similar errors happen in the problem of Table 13.

F   Experimental settings

NER experiments. We trained the XLM-RoBERTa transformer as the baseline model. The training was performed as a two-step approach with a training followed by a fine-tuning step. The training step utilized the Hugging-Face `get_linear_schedule_with_warmup` function that uses a learning rate decreasing linearly to 0 from the initial learning rate (1E-4) after a warmup period. This step was run for a maximum of 25 epochs on a batch of 64 samples with the early stopping callback function set to stop training monitoring the loss of the development split with a patience of 5 epochs and minimum change of 0.001. A model checkpoint was also used to save a checkpoint of the model that performed the best on the development set. The fine-tuning step also utilized a learning rate of 0.0001 for a maximum of 30 epochs with the same callback functions used in the training step. To set the learning rate, we used a grid search using a development set.

Generation experiments. We trained the BART baseline model and our model for a total of 200 epochs, using a learning rate of 1E-06, and with a batch size of 32. The corresponding performance on the dev set for the best model is summarized in Table 21. We trained the Text-to-Table for a total of 4000 updates, using a learning rate of 1E-05, and with max-tokens of 4096. To set the learning rate, we used a grid search using a development set.
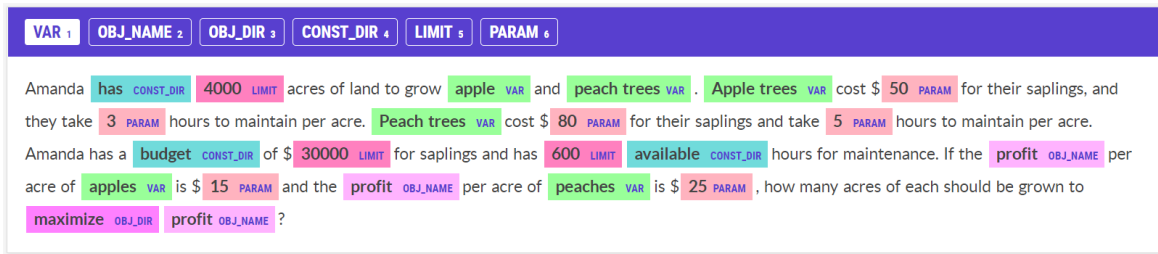
Figure 6: Screenshot of NER annotation using Prodigy.
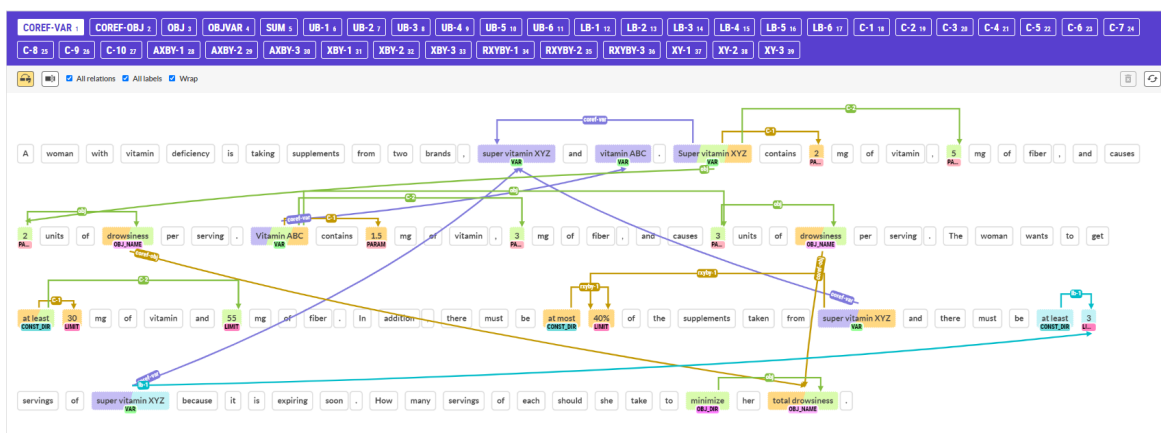
Figure 7: Screenshot of REL annotation using Prodigy.

Problem Description	A hotel employs cleaners and receptionists. Cleaners earn $500 per week and receptionists earn $350 per week. The hotel requires a minimum of 100 workers of whom at least 20 must be receptionists. To keep the hotel clean and running smoothly, the number of receptionists should be at least a third of the number of cleaners. The hotel wants to keep the weekly wage bill below $30000. Formulate a LP to minimize the wage bill.
Text-to-Table + Prompt Form	<table><tr><td></td><td>cleaners</td><td>receptionists</td><td>rhs</td></tr><tr><td>objective</td><td>500.0</td><td>350.0</td><td></td></tr><tr><td>minimum</td><td>-1.0</td><td>-1.0</td><td>-100.0</td></tr><tr><td>at least</td><td>0.0</td><td>-1.0</td><td>-20.0</td></tr><tr><td>at least</td><td>0.3333</td><td>-1.0</td><td>0.0</td></tr><tr><td>below</td><td>500.0</td><td>350.0</td><td>30000.0</td></tr></table>
Canonical Form	<table><tr><td></td><td>var_0</td><td>var_1</td><td>rhs</td></tr><tr><td>objective</td><td>500.0</td><td>350.0</td><td></td></tr><tr><td>constraint_0</td><td>-1.0</td><td>-1.0</td><td>-100.0</td></tr><tr><td>constraint_1</td><td>0.0</td><td>-1.0</td><td>-20.0</td></tr><tr><td>constraint_2</td><td>0.3333</td><td>-1.0</td><td>0.0</td></tr><tr><td>constraint_3</td><td>500.0</td><td>350.0</td><td>30000.0</td></tr></table>

Table 3: Text-to-Table example: Problem description, and expected output. Newline tokens have been replaced with newline characters, indentation was added, and column dividers were omitted for readability.

Constraint Type	Problem Description	Math Formulation + IR Representation
Sum Constraint	A bike shop sells two models of a bike: a mountain bike and a road bike. (...) The bike shop owner knows that the monthly demand will be **at most 150 bikes**. (...) How many bikes of each type should be stocked in order to maximize profit?	x + y <= 150 <DECLARATION><CONST_DIR> at most </CONST_DIR><LIMIT> 150 </LIMIT> <OPERATOR> LESS_OR_EQUAL </OPERATOR> <CONST_TYPE> [SUM_CONSTRAINT] </CONST_TYPE></DECLARATION>
Upper Bound	An ice cream bar sells vanilla and chocolate ice cream cones. (...) The ice cream bar must make at lest 20 cones of vanilla ice cream but **cannot make more than 50 cones**. (...) How many cones of each flavor should they make to maximize profit?	x <= 50 <DECLARATION><CONST_DIR> cannot make more than </CONST_DIR><LIMIT> 50 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [UPPER_BOUND] </CONST_TYPE> [for] <VAR> vanilla ice cream </VAR> </DECLARATION>
Lower Bound	There is only 5000 grams of a rare flower extract needed to make both youth and adult doses. (...) **A minimum of 10 adult doses** need to be made. (...) How many of each dose should be prepared to maximize profit?	y >= 10 <DECLARATION><CONST_DIR> minimum </CONST_DIR><LIMIT> 10 </LIMIT> <OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LOWER_BOUND] </CONST_TYPE> [for] <VAR> adult doses </VAR> </DECLARATION>
Linear Constraint (e.g. minimum requirement or capacity constraint)	A smoothie store sells two types of smoothies (...). Each **small smoothie requires 2 units of ice cream** and 1 unit of peanut butter. Each **large smoothie requires 3 units of ice cream** and 2 units of peanut butter. The company **only has a total of 20 units of ice cream** and 18 units of peanut butter. (...) how many of each should the store sell to maximize profit?	2x + 3y <= 20 <DECLARATION><CONST_DIR> total </CONST_DIR><LIMIT> 20 </LIMIT> <OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> small smoothie </VAR> [TIMES] <PARAM> 2 </PARAM><VAR> large smoothie </VAR> [TIMES] <PARAM> 3 </PARAM></DECLARATION>
Ratio Control Constraint	A furniture store only stocks and sells dining tables and chairs. (...) Because chairs sell in larger quantities, **at least 70% of all furniture in the store must be chairs**. (...) Formulate an LP to maximize profit.	x >= 70/100 * (x+y) <DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 70% </LIMIT> <OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [RATIO_CONSTRAINT] </CONST_TYPE> [for] <VAR> chairs </VAR> </DECLARATION>
Balance Constraint Type-1 : X <= B*Y	There is only 5000 grams of a rare flower extract needed to make both youth and adult doses. (...) Demand is such that **at least three times as many youth doses are needed than the adult doses**. (...) . How many of each dose should be prepared to maximize profit?	x >= 3y <DECLARATION><CONST_DIR> at least </CONST_DIR><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [XBY_CONSTRAINT] </CONST_TYPE><VAR> adult doses </VAR> [TIMES] <PARAM> three </PARAM> [is] <VAR> youth doses </VAR></DECLARATION>
Balance Constraint Type-2 : X <= Y	A peanut farmer has to send his product to the city. (...) He wants to spend at most $3000 and the **number of train trips must not exceed the number of truck trips**. Formulate a LP to maximize the number of peanut packages that can be transported.	x <= y <DECLARATION><CONST_DIR> must not exceed </CONST_DIR><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [XY_CONSTRAINT] </CONST_TYPE><VAR> truck trips </VAR> [is] <VAR> train trips </VAR> </DECLARATION>

Table 4: Different types of constraints with examples from the LPWP dataset.
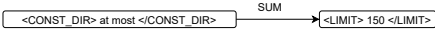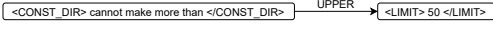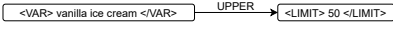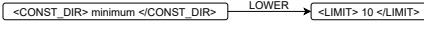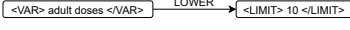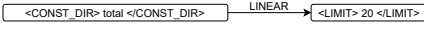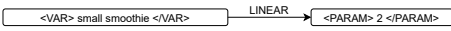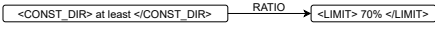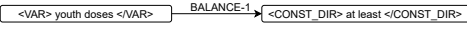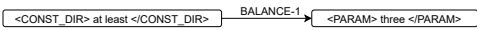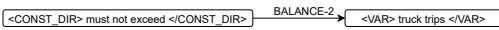
Constraint Type	Problem Description	Relation Annotation Examples
Sum Constraint	A bike shop sells two models of a bike: a mountain bike and a road bike. (...) The bike shop owner knows that the monthly demand will be **at most 150 bikes**. (...) How many bikes of each type should be stocked in order to maximize profit?	• CONST_DIR -> LIMIT <CONST_DIR> at most </CONST_DIR> —SUM→ <LIMIT> 150 </LIMIT>
Upper Bound	An ice cream bar sells vanilla and chocolate ice cream cones. (...) The ice cream bar must make at lest 20 cones of vanilla ice cream but **cannot make more than 50 cones**. (...) How many cones of each flavor should they make to maximize profit?	• CONST_DIR -> LIMIT <CONST_DIR> cannot make more than </CONST_DIR> —UPPER→ <LIMIT> 50 </LIMIT> • VAR -> LIMIT <VAR> vanilla ice cream </VAR> —UPPER→ <LIMIT> 50 </LIMIT>
Lower Bound	There is only 5000 grams of a rare flower extract needed to make both youth and adult doses. (...) A **minimum of 10 adult doses** need to be made. (...) How many of each dose should be prepared to maximize profit?	• CONST_DIR -> LIMIT <CONST_DIR> minimum </CONST_DIR> —LOWER→ <LIMIT> 10 </LIMIT> • VAR -> LIMIT <VAR> adult doses </VAR> —LOWER→ <LIMIT> 10 </LIMIT>
Linear Constraint (e.g. minimum requirement or capacity constraint)	A smoothie store sells two types of smoothies (...). Each **small smoothie requires 2 units of ice cream** and 1 unit of peanut butter. Each **large smoothie requires 3 units of ice cream** and 2 units of peanut butter. The company **only has a total of 20 units of ice cream** and 18 units of peanut butter. (...) how many of each should the store sell to maximize profit?	• CONST_DIR -> LIMIT <CONST_DIR> total </CONST_DIR> —LINEAR→ <LIMIT> 20 </LIMIT> • VAR (X) -> PARAM <VAR> small smoothie </VAR> —LINEAR→ <PARAM> 2 </PARAM> • VAR (Y) -> PARAM <VAR> large smoothie </VAR> —LINEAR→ <PARAM> 3 </PARAM>
Ratio Control Constraint	A furniture store only stocks and sells dining tables and chairs. (...) Because chairs sell in larger quantities, **at least 70% of all furniture in the store must be chairs**. (...) Formulate an LP to maximize profit.	• CONST_DIR -> LIMIT <CONST_DIR> at least </CONST_DIR> —RATIO→ <LIMIT> 70% </LIMIT> • VAR -> LIMIT <VAR> chairs </VAR> —RATIO→ <LIMIT> 70% </LIMIT>
Balance Constraint Type-1 : X <= B*Y	There is only 5000 grams of a rare flower extract needed to make both youth and adult doses. (...) Demand is such that **at least three times as many youth doses are needed than the adult doses**. (...) . How many of each dose should be prepared to maximize profit?	• VAR (X) -> CONST_DIR <VAR> youth doses </VAR> —BALANCE-1→ <CONST_DIR> at least </CONST_DIR> • CONST_DIR-> PARAM <CONST_DIR> at least </CONST_DIR> —BALANCE-1→ <PARAM> three </PARAM> • PARAM -> VAR (Y) <PARAM> three </PARAM> —BALANCE-1→ <VAR> adult doses </VAR>
Balance Constraint Type-2 : X <= Y	A peanut farmer has to send his product to the city. (...) He wants to spend at most $3000 and the **number of train trips must not exceed the number of truck trips**. Formulate a LP to maximize the number of peanut packages that can be transported.	• VAR (X) -> CONST_DIR <VAR> train trips </VAR> —BALANCE-2→ <CONST_DIR> must not exceed </CONST_DIR> • CONST_DIR-> VAR (Y) <CONST_DIR> must not exceed </CONST_DIR> —BALANCE-2→ <VAR> truck trips </VAR>

Table 5: REL annotation examples for different constraint types.

Problem Description	There is only 5000 grams of a rare flower extract needed to make both youth and adult doses. Youth doses contain 20 grams of extract and adult doses contain 35 grams. Demand is such that at least three times as many youth doses are needed than the adult doses. A minimum of 10 adult doses need to be made. Youth doses are sold for a profit of $5 while adult doses are sold at a profit of $3. How many of each dose should be prepared to maximize profit?
Intermediate Representation	<pre>\<DECLARATION\>\n \<OBJ_DIR\> maximize \</OBJ_DIR\>\n \<OBJ_NAME\> profit \</OBJ_NAME\> [is]\n \<VAR\> Youth doses \</VAR\> [TIMES] \<PARAM\> 5 \</PARAM\>\n \<VAR\> adult doses \</VAR\> [TIMES] \<PARAM\> 3 \</PARAM\>\n\</DECLARATION\>\n\n\<DECLARATION\>\n \<CONST_DIR\> only \</CONST_DIR\>\<LIMIT\> 5000 \</LIMIT\>\n \<OPERATOR\> LESS_OR_EQUAL \</OPERATOR\>\n \<CONST_TYPE\> [LINEAR_CONSTRAINT] \</CONST_TYPE\> [is]\n \<VAR\> Youth doses \</VAR\> [TIMES] \<PARAM\> 20 \</PARAM\>\n \<VAR\> adult doses \</VAR\> [TIMES] \<PARAM\> 35 \</PARAM\>\n\</DECLARATION\>\n\nDECLARATION\>\n \<CONST_DIR\> at least \</CONST_DIR\>\n \<OPERATOR\> GREATER_OR_EQUAL \</OPERATOR\>\n \<CONST_TYPE\> [XBY_CONSTRAINT] \</CONST_TYPE\>\n \<VAR\> adult doses \</VAR\> [TIMES]\n \<PARAM\> three \</PARAM\> [is]\n \<VAR\> youth doses \</VAR\>\n\</DECLARATION\>\n\n\<DECLARATION\>\n \<CONST_DIR\> minimum \</CONST_DIR\>\<LIMIT\> 10 \</LIMIT\>\n \<OPERATOR\> GREATER_OR_EQUAL \</OPERATOR\>\n \<CONST_TYPE\> [LOWER_BOUND] \</CONST_TYPE\> [for]\n \<VAR\> adult doses \</VAR\>\n\</DECLARATION\></pre>
Canonical Form	<table><tr><td></td><td>var_0</td><td>var_1</td><td>rhs</td></tr><tr><td>objective</td><td>5</td><td>3</td><td></td></tr><tr><td>constraint_0</td><td>20</td><td>35</td><td>5000</td></tr><tr><td>constraint_1</td><td>-1</td><td>3</td><td>0</td></tr><tr><td>constraint_2</td><td>0</td><td>-1.0</td><td>-10</td></tr></table>
Math Formulation	<pre>max 5x + 3y\nsubject to\n 20x + 35y <= 5000\n x >= 3y\n y >= 10</pre>

Table 6: Original dataset - Resource allocation example: problem description, intermediate representation, canonical form, and math formulation.

Problem Description	Your client has $60,000 available to invest for a 1 year term. The money can be placed in a trust yielding a 2% return or in a savings account yielding a 3% return. Based on your experience, you advise your client that at least 15% of the investment be placed in the trust and that at most 80% of the investment be placed in the savings account. How much should your client invest in each so as to maximize his return on investment?
| Intermediate Representation | ```
<DECLARATION>
    <OBJ_DIR> maximize </OBJ_DIR>
    <OBJ_NAME> return </OBJ_NAME> [is]
    <VAR> trust </VAR> [TIMES] <PARAM> 2% </PARAM>
    <VAR> savings account </VAR> [TIMES] <PARAM> 3%
        </PARAM>
</DECLARATION>

<DECLARATION>
    <CONST_DIR> available </CONST_DIR><LIMIT> 60,000
        </LIMIT>
    <OPERATOR> LESS_OR_EQUAL </OPERATOR>
    <CONST_TYPE> [SUM_CONSTRAINT] </CONST_TYPE>
</DECLARATION>

<DECLARATION>
    <CONST_DIR> at least </CONST_DIR><LIMIT> 15%
        </LIMIT>
    <OPERATOR> GREATER_OR_EQUAL </OPERATOR>
    <CONST_TYPE> [RATIO_CONSTRAINT] </CONST_TYPE> [for]
    <VAR> trust </VAR>
</DECLARATION>

<DECLARATION>
    <CONST_DIR> at most </CONST_DIR><LIMIT> 80% </LIMIT>
    <OPERATOR> LESS_OR_EQUAL </OPERATOR>
    <CONST_TYPE> [RATIO_CONSTRAINT] </CONST_TYPE> [for]
    <VAR> savings account </VAR>
</DECLARATION>
``` |
| Canonical Form | |

	var_0	var_1	rhs
objective	0.02	0.03	
constraint_0	1	1	60000
constraint_1	-0.85	0.15	0
constraint_2	-0.8	0.2	0

| Math Formulation | ```
max (2/100)*x + (3/100)*y
subject to
   x + y <= 60000
   x >= (15/100)*(x+y)
   y <= (80/100)*(x+y)
``` |

Table 7: Original dataset - Investment allocation example: problem description, intermediate representation, canonical form, and math formulation.

Problem Description	A farmer has 500 acres of land to grow turnips and pumpkins. Turnips require 50 minutes of watering and $80 worth of pesticide per acre. Pumpkins require 90 minutes of watering and $50 worth of pesticide per acre. The farmer has 40000 minutes available for watering and $34000 available to spend on pesticide. If the revenue per acre of turnips is $300 and the revenue per acre of pumpkins is $450, how many acres of each should he grow to maximize his revenue.
Intermediate Representation	``` <DECLARATION> <OBJ_DIR> maximize </OBJ_DIR> <OBJ_NAME> revenue </OBJ_NAME> [is] <VAR> turnips </VAR> [TIMES] <PARAM> 300 </PARAM> <VAR> pumpkins </VAR> [TIMES] <PARAM> 450 </PARAM> </DECLARATION> <DECLARATION> <CONST_DIR> has </CONST_DIR><LIMIT> 500 </LIMIT> <OPERATOR> LESS_OR_EQUAL </OPERATOR> <CONST_TYPE> [SUM_CONSTRAINT] </CONST_TYPE> </DECLARATION> <DECLARATION> <CONST_DIR> available </CONST_DIR><LIMIT> 40000 </LIMIT> <OPERATOR> LESS_OR_EQUAL </OPERATOR> <CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> Turnips </VAR> [TIMES] <PARAM> 50 </PARAM> <VAR> Pumpkins </VAR> [TIMES] <PARAM> 90 </PARAM> </DECLARATION> <DECLARATION> <CONST_DIR> available </CONST_DIR><LIMIT> 34000 </LIMIT> <OPERATOR> LESS_OR_EQUAL </OPERATOR> <CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> Turnips </VAR> [TIMES] <PARAM> 80 </PARAM> <VAR> Pumpkins </VAR> [TIMES] <PARAM> 50 </PARAM> </DECLARATION> ```
Canonical Form	<table><tr><td></td><td>var_0</td><td>var_1</td><td>rhs</td></tr><tr><td>objective</td><td>300</td><td>450</td><td></td></tr><tr><td>constraint_0</td><td>1</td><td>1</td><td>500</td></tr><tr><td>constraint_1</td><td>50</td><td>90</td><td>40000</td></tr><tr><td>constraint_2</td><td>80</td><td>50</td><td>34000</td></tr></table>
Math Formulation	``` max 300x + 450y subject to x + y <= 500 50x + 90y <= 40000 80x + 50y <= 34000 ```

Table 8: Original Dataset - Farming example: problem description, intermediate representation, canonical form, and math formulation.

	Problem Description	A mining company has available a total of 100 square miles of mining sites and considering the use of two mining techniques: heap leaching and vat leaching. For each square mile of land, heap leaching technique can have a daily production of 3 tons of rare earth oxide per square miles but it also creates 8 tons of polluted wastewater and requires 10 extraction machines. On the other hand, vat leaching technique produces 5 tons of rare earth oxide per square miles per day while creating 17 tons of polluted wastewater and requiring 20 extraction machines. There are 100 machines available and due to environmental regulations, the amount of polluted wastewater must be at most 90 tons daily. Find the proportion of lands that use each mining technique in order to maximize the daily production of rare earth oxide.

Intermediate Representation

```
<DECLARATION>
    <OBJ_DIR> maximize </OBJ_DIR>
    <OBJ_NAME> rare earth oxide </OBJ_NAME> [is]
    <VAR> heap leaching </VAR> [TIMES] <PARAM> 3
        </PARAM>
    <VAR> vat leaching </VAR> [TIMES] <PARAM> 5 </PARAM>
</DECLARATION>

<DECLARATION>
    <CONST_DIR> available </CONST_DIR><LIMIT> 100
        </LIMIT>
    <OPERATOR> LESS_OR_EQUAL </OPERATOR>
    <CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is]
    <VAR> heap leaching </VAR> [TIMES] <PARAM> 10
        </PARAM>
    <VAR> vat leaching </VAR> [TIMES] <PARAM> 20
        </PARAM>
</DECLARATION>

<DECLARATION>
    <CONST_DIR> at most </CONST_DIR><LIMIT> 90 </LIMIT>
    <OPERATOR> LESS_OR_EQUAL </OPERATOR>
    <CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is]
    <VAR> heap leaching </VAR> [TIMES] <PARAM> 8
        </PARAM>
    <VAR> vat leaching </VAR> [TIMES] <PARAM> 17
        </PARAM>
</DECLARATION>

<DECLARATION>
    <CONST_DIR> available </CONST_DIR><LIMIT> 100
        </LIMIT>
    <OPERATOR> LESS_OR_EQUAL </OPERATOR>
    <CONST_TYPE> [SUM_CONSTRAINT] </CONST_TYPE>
</DECLARATION>
```

Canonical Form

	var_0	var_1	rhs
objective	3	5	
constraint_0	10	20	100
constraint_1	8	17	90
constraint_2	1	1	100

| Math Formulation | ```
max 3x + 5y
subject to
   10x + 20y <= 100
   8x + 17y <= 90
   x + y <= 100
``` |

Table 9: Out-of-domain dataset - Production example: problem description, intermediate representation, canonical form, and math formulation.

Problem Description	A shipping company need to transport packages by either truck or car. A truck can transport 50 packages per trip while a car can transport 30 packages per trip. In addition, a truck uses 20 liters of gas per trip while a car uses 15 liters of gas per trip. There can be at most 5 truck trips made and at least 30% of all the trips must be made by car. The company needs to transport at least 500 packages. How many of each transportation should they use to minimize the total amount of gas consumed?
| Intermediate Representation | ```
<DECLARATION>
    <OBJ_DIR> minimize </OBJ_DIR>
    <OBJ_NAME> amount of gas </OBJ_NAME> [is]
    <VAR> truck </VAR> [TIMES] <PARAM> 20 </PARAM>
    <VAR> car </VAR> [TIMES] <PARAM> 15 </PARAM>
</DECLARATION>

<DECLARATION>
    <CONST_DIR> at most </CONST_DIR><LIMIT> 5 </LIMIT>
    <OPERATOR> LESS_OR_EQUAL </OPERATOR>
    <CONST_TYPE> [UPPER_BOUND] </CONST_TYPE> [for]
    <VAR> truck </VAR>
</DECLARATION>

<DECLARATION>
    <CONST_DIR> at least </CONST_DIR><LIMIT> 30%
        </LIMIT>
    <OPERATOR> GREATER_OR_EQUAL </OPERATOR>
    <CONST_TYPE> [RATIO_CONSTRAINT] </CONST_TYPE> [for]
    <VAR> car </VAR>
</DECLARATION>

<DECLARATION>
    <CONST_DIR> at least </CONST_DIR><LIMIT> 500
        </LIMIT>
    <OPERATOR> GREATER_OR_EQUAL </OPERATOR>
    <CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is]
    <VAR> truck </VAR> [TIMES] <PARAM> 50 </PARAM>
    <VAR> car </VAR> [TIMES] <PARAM> 30 </PARAM>
</DECLARATION>
``` |
| Canonical Form | <table><tr><td></td><td>var_0</td><td>var_1</td><td>rhs</td></tr><tr><td>objective</td><td>20</td><td>15</td><td></td></tr><tr><td>constraint_0</td><td>1</td><td>0</td><td>5</td></tr><tr><td>constraint_1</td><td>0.3</td><td>-0.7</td><td>0</td></tr><tr><td>constraint_2</td><td>-50</td><td>-30</td><td>-500</td></tr></table> |
| Math Formulation | ```
min 20x + 15y
subject to
   x <= 5
   y >= (30/100)*(x+y)
   50x + 30y >= 500
``` |

Table 10: Out-of-domain dataset - Transportation example: problem description, intermediate representation, canonical form, and math formulation.

Problem Description	A patient is undergoing radiation treatment involving two beams, Beam 1 and Beam 2. Beam 1 delivers a dose of 0.3 units of medicine per minute to the benign area of the pancreas and 0.2 units of medicine per minute to the benign area of the skin. Beam 2 delivers 0.2 units of medicine per minute to the benign area of the pancreas and 0.1 units of medicine per minute to the benign area of the skin. In addition, beam 1 delivers 0.6 units of medicine per minute to the tumor and beam 2 delivers 0.4 units of medicine per minute to the tumor. At most 4 units of medicine should be received by the skin and at least 3 units of medicine should be delivered to the tumor. How many minutes of each beam should be used to minimize the total radiation received by the pancreas?

Intermediate Representation	<pre><DECLARATION>
	    <OBJ_DIR> minimize </OBJ_DIR>
	    <OBJ_NAME> total radiation received by the pancreas
	        </OBJ_NAME> [is]
	    <VAR> Beam 1 </VAR> [TIMES] <PARAM> 0.3 </PARAM>
	    <VAR> Beam 2 </VAR> [TIMES] <PARAM> 0.2 </PARAM>
	</DECLARATION>
	
	<DECLARATION>
	    <CONST_DIR> At most </CONST_DIR><LIMIT> 4 </LIMIT>
	    <OPERATOR> LESS_OR_EQUAL </OPERATOR>
	    <CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is]
	    <VAR> Beam 1 </VAR> [TIMES] <PARAM> 0.2 </PARAM>
	    <VAR> Beam 2 </VAR> [TIMES] <PARAM> 0.1 </PARAM>
	</DECLARATION>
	
	<DECLARATION>
	    <CONST_DIR> at least </CONST_DIR><LIMIT> 3 </LIMIT>
	    <OPERATOR> GREATER_OR_EQUAL </OPERATOR>
	    <CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is]
	    <VAR> beam 1 </VAR> [TIMES] <PARAM> 0.6 </PARAM>
	    <VAR> beam 2 </VAR> [TIMES] <PARAM> 0.4 </PARAM>
	</DECLARATION></pre>

Canonical Form	<table><tr><td></td><td>var_0</td><td>var_1</td><td>rhs</td></tr><tr><td>objective</td><td>0.3</td><td>0.2</td><td></td></tr><tr><td>constraint_0</td><td>0.2</td><td>0.1</td><td>4</td></tr><tr><td>constraint_1</td><td>-0.6</td><td>-0.4</td><td>3</td></tr></table>

Math Formulation	<pre>min 0.3x + 0.2y
	subject to
	    0.2x + 0.1y <= 4
	    0.6x + 0.4y >= 3</pre>

Table 11: Out-of-domain dataset - Health Science example: problem description, intermediate representation, canonical form, and math formulation.

A furniture store only stocks and sells dining tables and chairs. The profit per dining table is $350 and the profit per chair is $75. There is 500 sq ft of space available and a dining table requires 8 sq ft of floor space while a chair requires 2 sq ft. Because chairs sell in larger quantities, at least 70% of all furniture in the store must be chairs. In terms of capital, a dining table ties up $1000 in capital and a chair ties up $150 in capital. The company wants a maximum of $20000 worth of capital tied up at any time. Formulate an LP to maximize profit.

OptGen w/o copy	OptGen
`<DECLARATION><OBJ_DIR> maximize </OBJ_DIR><OBJ_NAME> profit </OBJ_NAME> [is] <VAR> dining table </VAR> [TIMES] <PARAM> 350 </PARAM><VAR> chair </VAR> [TIMES] <PARAM> 75 </PARAM></DECLARATION>`	`<DECLARATION><OBJ_DIR> maximize </OBJ_DIR><OBJ_NAME profit </OBJ_NAME> [is] <VAR> dining table </VAR> [TIMES] <PARAM> 350 </PARAM><VAR> chair </VAR> [TIMES] <PARAM> 75 </PARAM></DECLARATION>`
`<DECLARATION><CONST_DIR> available </CONST_DIR><LIMIT> 20000 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> dining table </VAR> [TIMES] <PARAM> 8 </PARAM><VAR> chair </VAR> [TIMES] <PARAM> 2 </PARAM></DECLARATION>`	`<DECLARATION><CONST_DIR> available </CONST_DIR><LIMIT> 500 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> dining table </VAR> [TIMES] <PARAM> 8 </PARAM><VAR> chair </VAR> [TIMES] <PARAM> 2 </PARAM></DECLARATION>`
`<DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 200000 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> dining table </VAR> [TIMES] <PARAM> 1000 </PARAM><VAR> chair </VAR> [TIMES] <PARAM> 150 </PARAM> </DECLARATION>`	`<DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 70% </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [RATIO_CONSTRAINT] </CONST_TYPE> [for] <VAR> chairs </VAR> </DECLARATION>`
`<DECLARATION><CONST_DIR> maximum </CONST_DIR><LIMIT> 20000 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> dining table </VAR> [TIMES] <PARAM> 1000 </PARAM><VAR> chair </VAR> [TIMES] <PARAM> 150 </PARAM></DECLARATION>`	`<DECLARATION><CONST_DIR> maximum </CONST_DIR><LIMIT> 20000 </LIMIT> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> dining table </VAR> [TIMES] <PARAM> 1000 </PARAM><VAR> chair </VAR> [TIMES] <PARAM> 150 </PARAM></DECLARATION>`

Table 12: Qualitative comparison of model predictions with and without copy mechanism. In this example, the model without copy mechanism produced erroneous IR declarations (highlighted in red) whereas our model produces perfect matches with the gold declarations.

There is only 5000 grams of a rare flower extract needed to make both youth and adult doses. Youth doses contain 20 grams of extract and adult doses contain 35 grams. Demand is such that at least three times as many youth doses are needed than the adult doses. A minimum of 10 adult doses need to be made. Youth doses are sold for a profit of $5 while adult doses are sold at a profit of $3. How many of each dose should be prepared to maximize profit?

OptGen w/o copy	OptGen

<pre><DECLARATION><OBJ_DIR> maximize </OBJ_DIR><OBJ_NAME profit </OBJ_NAME> [is] <VAR> Youth doses </VAR> [TIMES] <PARAM> 5 </PARAM><VAR> adult doses </VAR> [TIMES] <PARAM> 3 </PARAM></DECLARATION>
 	
<DECLARATION><CONST_DIR> only </CONST_DIR><LIMIT> 200000 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> Youth doses </VAR> [TIMES] <PARAM> 20 </PARAM><VAR> adult doses </VAR> [TIMES] <PARAM> 35 </PARAM></DECLARATION>
 	
<DECLARATION><CONST_DIR> at least </CONST_DIR><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [XBY_CONSTRAINT] </CONST_TYPE><VAR> youth doses </VAR> [TIMES] <PARAM> three </PARAM> [is] <VAR> adult doses </VAR></DECLARATION>
 	
<DECLARATION><CONST_DIR> minimum </CONST_DIR><LIMIT> 200000 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> Youth doses </VAR> [TIMES] <PARAM> 20 </PARAM><VAR> adult doses </VAR> [TIMES] <PARAM> 35 </PARAM></DECLARATION></pre>
	<pre><DECLARATION><OBJ_DIR> maximize </OBJ_DIR><OBJ_NAME profit </OBJ_NAME> [is] <VAR> Youth doses </VAR> [TIMES] <PARAM> 5 </PARAM><VAR> adult doses </VAR> [TIMES] <PARAM> 3 </PARAM></DECLARATION>
	
	<DECLARATION><CONST_DIR> only </CONST_DIR><LIMIT> 5000 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> Youth doses </VAR> [TIMES] <PARAM> 20 </PARAM><VAR> adult doses </VAR> [TIMES] <PARAM> 35 </PARAM></DECLARATION>
	
	<DECLARATION><CONST_DIR> at least </CONST_DIR><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [XBY_CONSTRAINT] </CONST_TYPE><VAR> adult doses </VAR> [TIMES] <PARAM> three </PARAM> [is] <VAR> youth doses </VAR></DECLARATION>
	
	<DECLARATION><CONST_DIR> minimum </CONST_DIR><LIMIT> 10 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LOWER_BOUND] </CONST_TYPE> [for] <VAR> adult doses </VAR></DECLARATION></pre>

Table 13: Qualitative comparison of model predictions with and without copy mechanism. In this example, the model without copy mechanism produced erroneous IR declarations (highlighted in red) whereas our model produces perfect matches with the gold declarations.

(a) Problem level errors on the Source domain

(b) Problem level errors on the Target domain

Figure 8: Classification of problem level errors for each model. These errors are not mutually exclusive. Problem level syntax errors will result in a completely incorrect problem as they cannot be parsed. On models predicting the IR of a problem, a problem level syntax error is defined as one that cannot be parsed through an XML parser. On the T2T models, syntax errors are defined as a mismatch between the number of columns in each row.

(a) Declaration level errors on the Source domain.

(b) Declaration level errors on the Target domain.

Figure 9: Classification of declaration level errors for each model. Note that some of these error types are not made by the T2T models, as they do not predict constraint types, and their syntax errors are all at the problem level. The parser will skip parsing poorly formatted declarations, which are later counted as incorrect. We define a declaration level parameter error as a declaration that contains any parameter mismatch between the prediction and the ground truth.

54

A patient in the hospital can take two pills, Pill 1 and Pill 2. Per pill, pill 1 provides 0.2 units of pain medication and 0.3 units of anxiety medication. Per pill, pill 2 provides 0.6 units of pain medication and 0.2 units of anxiety medication. In addition, pill 1 causes 0.3 units of discharge while pill 2 causes 0.1 units of discharge. At most 6 units of pain medication can be provided and at least 3 units of anxiety medication must be provided. How many pills of each should the patient be given to minimize the total amount of discharge?

Gold	Pred
`<DECLARATION><OBJ_DIR> minimize </OBJ_DIR><OBJ_NAME> amount of discharge </OBJ_NAME> [is] <VAR> pill 1 </VAR> [TIMES] <PARAM> 0.3 </PARAM><VAR> pill 2 </VAR> [TIMES] <PARAM> 0.1 </PARAM></DECLARATION>`	`<DECLARATION><OBJ_DIR> minimize </OBJ_DIR><OBJ_NAME> `total` amount of discharge </OBJ_NAME> [is] <VAR> Pill 1 </VAR> [TIMES] <PARAM> 0.3 </PARAM><VAR> pill 2 </VAR> [TIMES] <PARAM> 0.1 </PARAM></DECLARATION>`
`<DECLARATION><CONST_DIR> At most </CONST_DIR><LIMIT> 6 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> pill 1 </VAR> [TIMES] <PARAM> 0.2 </PARAM><VAR> pill 2 </VAR> [TIMES] <PARAM> 0.6 </PARAM></DECLARATION>`	`<DECLARATION><CONST_DIR> At most </CONST_DIR><LIMIT> 6 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> pill 1 </VAR> [TIMES] <PARAM> 0.2 </PARAM><VAR> pill 2 </VAR> [TIMES] <PARAM> 0.6 </PARAM></DECLARATION>`
`<DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 3 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> pill 1 </VAR> [TIMES] <PARAM> 0.3 </PARAM><VAR> pill 2 </VAR> [TIMES] <PARAM> 0.2 </PARAM></DECLARATION>`	`<DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 3 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> pill 1 </VAR> [TIMES] <PARAM> 0.3 </PARAM><VAR> pill 2 </VAR> [TIMES] <PARAM> 0.2 </PARAM></DECLARATION>`

Table 14: Comparison of predicted vs. gold IR for out-of-domain example (Health Science). In this example, the predicted IR is almost equal to the gold IR except for the extra token ("total") in the objective name declaration.

An international goods exporter uses ships and planes to transport goods. A ship can take 40 containers worth of goods and uses 500 liters of fuel per trip. A plane can take 20 containers worth of goods and uses 300 liters of fuel per trip. The company needs to transport at least 500 containers worth of goods. In addition, there can be at most 10 plane trips made and a minimum of 50% of the trips made must be by ship. How many of each trip should be made to minimize the total amount of fuel consumed?

Gold	Pred

`<DECLARATION><OBJ_DIR> minimize </OBJ_DIR><OBJ_NAME> total amount of fuel </OBJ_NAME> [is] <VAR> ship </VAR> [TIMES] <PARAM> 500 </PARAM><VAR> plane </VAR> [TIMES] <PARAM> 300 </PARAM></DECLARATION>`
 	
`<DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 500 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> ship </VAR> [TIMES] <PARAM> 40 </PARAM><VAR> plane </VAR> [TIMES] <PARAM> 20 </PARAM></DECLARATION>`
 	
`<DECLARATION><CONST_DIR> at most </CONST_DIR><OPERATOR> LESS_OR_EQUAL </OPERATOR><LIMIT> 10 </LIMIT><CONST_TYPE> [UPPER_BOUND] </CONST_TYPE> [for] <VAR> plane </VAR></DECLARATION>`
 	
`<DECLARATION><CONST_DIR> minimum </CONST_DIR><OPERATOR> GREATER_OR_EQUAL </OPERATOR><LIMIT> 50% </LIMIT><CONST_TYPE> [RATIO_CONSTRAINT] </CONST_TYPE> [for] <VAR> ship </VAR></DECLARATION>`
	`<DECLARATION><OBJ_DIR> minimize </OBJ_DIR><OBJ_NAME> total amount of fuel </OBJ_NAME> [is] <VAR> ship </VAR> [TIMES] <PARAM> 500 </PARAM><VAR> plane </VAR> [TIMES] <PARAM> 300 </PARAM></DECLARATION>`
	
	`<DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 500 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> ship </VAR> [TIMES] <PARAM> 40 </PARAM><VAR> plane </VAR> [TIMES] <PARAM> 20 </PARAM></DECLARATION>`
	
	`<DECLARATION><CONST_DIR> at most </CONST_DIR><OPERATOR> LESS_OR_EQUAL </OPERATOR><LIMIT> 10 </LIMIT><CONST_TYPE> [UPPER_BOUND] </CONST_TYPE> [for] <VAR> plane </VAR></DECLARATION>`
	
	`<DECLARATION><CONST_DIR> minimum </CONST_DIR><OPERATOR> GREATER_OR_EQUAL </OPERATOR><LIMIT> 50% </LIMIT><CONST_TYPE> [RATIO_CONSTRAINT] </CONST_TYPE> [for] <VAR> ship </VAR></DECLARATION>`

Table 15: Comparison of predicted vs. gold IR for out-of-domain example (Transportation example). In this example, the model perfectly matched the gold IR.

A pharmacy has 3000 mg of morphine to make painkillers and sleeping pills. Each painkiller pill requires 10 mg of morphine and 3 units of digestive medicine. Each sleeping pill requires 6 mg of morphine and 5 units of digestive medicine. The pharmacy needs to make at least 50 painkiller pills. Since sleeping pills are more popular, at least 70% of the pills should be sleeping pills. How many of each should the pharmacy make to minimize the total amount of digestive medicine needed?

Gold	Pred
`<DECLARATION><OBJ_DIR> minimize </OBJ_DIR><OBJ_NAME> amount of digestive medicine </OBJ_NAME> [is] <VAR> painkiller pill </VAR> [TIMES] <PARAM> 3 </PARAM><VAR> sleeping pill </VAR> [TIMES] <PARAM> 5 </PARAM></DECLARATION>`	`<DECLARATION><OBJ_DIR> minimize </OBJ_DIR><OBJ_NAME> amount of digestive medicine </OBJ_NAME> [is] <VAR> painkiller pill </VAR> [TIMES] <PARAM> 3 </PARAM><VAR> sleeping pill </VAR> [TIMES] <PARAM> 5 </PARAM></DECLARATION>`
`<DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 50 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LOWER_BOUND] </CONST_TYPE> [for] <VAR> painkiller pills </VAR></DECLARATION>`	`<DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 50 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LOWER_BOUND] </CONST_TYPE> [for] <VAR> painkiller pills </VAR></DECLARATION>`
`<DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 70% </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [RATIO_CONSTRAINT] </CONST_TYPE> [for] <VAR> sleeping pills </VAR></DECLARATION>`	`<DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 50 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LOWER_BOUND] </CONST_TYPE> [for] <VAR> painkiller pills </VAR></DECLARATION>`
`<DECLARATION><CONST_DIR> has </CONST_DIR><LIMIT> 3000 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> painkiller pill </VAR> [TIMES] <PARAM> 10 </PARAM><VAR> sleeping pill </VAR> [TIMES] <PARAM> 6 </PARAM></DECLARATION>`	`<DECLARATION><CONST_DIR> has </CONST_DIR><LIMIT> 3000 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> painkiller pill </VAR> [TIMES] <PARAM> 10 </PARAM><VAR> sleeping pill </VAR> [TIMES] <PARAM> 6 </PARAM></DECLARATION>`

Table 16: Comparison of predicted vs. gold IR for out-of-domain example (Health Science). In this example, our model predicted the wrong constraint type as lower bound instead of a ratio constraint. This shows that it is sometimes hard for the model to distinguish between similar constraint types. The rest of the declaration that follows the constraint type is also invalid.

A parent feeds their baby two flavors of baby food, apple and carrot, in order to meet the babies fat and folate requirements. Each serving of apple flavored baby food contains 2 units of fat and 5 units of folate. Each serving of carrot flavored baby food contains 4 units of fat and 3 units of folate. The baby does not like the carrot flavor, and therefore he must eat three times as many apple flavored baby food as carrot flavored baby food. However, he must eat at least 2 servings of carrot flavored baby food. If the baby can consume at most 100 units of folate, how many servings of each should he eat to maximize his fat intake?

Gold	Pred
`<DECLARATION><OBJ_DIR> maximize </OBJ_DIR><OBJ_NAME> fat intake </OBJ_NAME> [is] <VAR> apple flavored baby </VAR> [TIMES] <PARAM> 2 </PARAM> <VAR> carrot flavored baby </VAR> [TIMES] <PARAM> 4 </PARAM></DECLARATION>`	`<DECLARATION><OBJ_DIR> maximize </OBJ_DIR><OBJ_NAME> fat intake </OBJ_NAME> [is] <VAR> apple flavored baby food </VAR> [TIMES] <PARAM> 5 </PARAM></DECLARATION>`
`<DECLARATION><CONST_DIR> must eat </CONST_DIR><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [XBY_CONSTRAINT] </CONST_TYPE><VAR> carrot flavored baby </VAR> [TIMES] <PARAM> three </PARAM> [is] <VAR> apple flavored baby food </VAR></DECLARATION>`	`<DECLARATION><CONST_DIR> must eat </CONST_DIR><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [XBY_CONSTRAINT] </CONST_TYPE><VAR> carrot flavored baby food </VAR> [TIMES] <PARAM> three </PARAM> [is] <VAR> apple flavored baby food </VAR></DECLARATION>`
`<DECLARATION><CONST_DIR> at least </CONST_DIR><OPERATOR> GREATER_OR_EQUAL </OPERATOR><LIMIT> 2 </LIMIT><CONST_TYPE> [LOWER_BOUND] </CONST_TYPE> [for] <VAR> carrot flavored baby </VAR></DECLARATION>`	`<DECLARATION><CONST_DIR> at least </CONST_DIR><OPERATOR> GREATER_OR_EQUAL </OPERATOR><LIMIT> 2 </LIMIT><CONST_TYPE> [LOWER_BOUND] </CONST_TYPE> [for] <VAR> carrot flavored baby food </VAR></DECLARATION>`
`<DECLARATION><CONST_DIR> at most </CONST_DIR><LIMIT> 100 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> apple flavored baby </VAR> [TIMES] <PARAM> 5 </PARAM><VAR> carrot flavored baby </VAR> [TIMES] <PARAM> 3 </PARAM></DECLARATION>`	`<DECLARATION><CONST_DIR> at most </CONST_DIR><LIMIT> 100 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> apple flavored baby food </VAR> [TIMES] <PARAM> 2 </PARAM><VAR> carrot flavored baby food </VAR> [TIMES] <PARAM> 4 </PARAM></DECLARATION>`

Table 17: Comparison of predicted vs. gold IR for out-of-domain example (Health Science). In this example, the model produced an erroneous declaration in the objective and the last constraint. The wrong data parameters, which should describe the objective function, are instead copied into the last constraint. This example illustrates the difficulty of parsing an input document that is ambiguous.

A chocolate company can transport their boxes of chocolate either using their own vans or by renting trucks. Their vans can transport 50 boxes per trip while a truck can transport 80 boxes per trip. Since they own their vans, the cost per van trip is $30 while the cost per truck trip is $50. The company needs to transport at least 1500 boxes of chocolate and they have a budget of $1000. Since the vans also provide advertising, the number of trips by van must be larger than the number of trips by trucks. How many of trip by each should be done to minimize the total number of trips?

Gold	Pred
`<DECLARATION><OBJ_DIR> minimize </OBJ_DIR><OBJ_NAME> number of trips </OBJ_NAME> [is] <VAR> vans </VAR> [TIMES] <PARAM> ONE </PARAM><VAR> trucks </VAR> [TIMES] <PARAM> ONE </PARAM></DECLARATION>`	`<DECLARATION><OBJ_DIR> minimize </OBJ_DIR><OBJ_NAME> total number of trips </OBJ_NAME> [is] <VAR> van </VAR> [TIMES] <PARAM> ONE </PARAM><VAR> truck </VAR> [TIMES] <PARAM> ONE </PARAM></DECLARATION>`
`<DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 1500 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> vans </VAR> [TIMES] <PARAM> 50 </PARAM><VAR> truck </VAR> [TIMES] <PARAM> 80 </PARAM></DECLARATION>`	`<DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 1500 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> vans </VAR> [TIMES] <PARAM> 50 </PARAM><VAR> trucks </VAR> [TIMES] <PARAM> 80 </PARAM></DECLARATION>`
`<DECLARATION><CONST_DIR> budget </CONST_DIR><LIMIT> 1000 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> van </VAR> [TIMES] <PARAM> 30 </PARAM><VAR> truck </VAR> [TIMES] <PARAM> 50 </PARAM></DECLARATION>`	`<DECLARATION><CONST_DIR> budget </CONST_DIR><LIMIT> 1000 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> van </VAR> [TIMES] <PARAM> 30 </PARAM><VAR> truck </VAR> [TIMES] <PARAM> 50 </PARAM></DECLARATION>`
`<DECLARATION><CONST_DIR> must be larger than </CONST_DIR><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [XY_CONSTRAINT] </CONST_TYPE> <VAR> trucks </VAR> [is] <VAR> van </VAR> *@)</DECLARATION>`	`<DECLARATION><CONST_DIR> must be larger than </CONST_DIR><LIMIT> 500 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> vans </VAR> [TIMES] <PARAM> 50 </PARAM><VAR> trucks </VAR> [TIMES] <PARAM> 80 </PARAM></DECLARATION>`

Table 18: Comparison of predicted vs. gold IR for out-of-domain example (Transportation example). In this example, our model detects a linear constraint instead of a balance constraint. In fact, the balance constraints are less frequent in the training dataset whereas the linear constraints are the majority ones. This can explain this type of error as the constraint types are imbalanced in the training dataset.

A toy store decides to deliver gifts using two shipping companies, a new one and an old one. The new company can deliver 50 gifts per trip while the old company can deliver 70 gifts per trip. The new company uses 30 liters of diesel per trip while the old company uses 40 liters of diesel per trip. The toy store needs to deliver at least 1000 gifts. There can be at most 15 trips made by the new company. In order to make sure that the old company does not go out of business, at least 40% of all trips must be made by the old company. How many trips should each company make to minimize the total amount of diesel used?

Gold	Pred
`<DECLARATION><OBJ_DIR> minimize </OBJ_DIR><OBJ_NAME> total amount of diesel </OBJ_NAME> [is] <VAR> new company </VAR> [TIMES] <PARAM> 30 </PARAM><VAR> old company </VAR> [TIMES] <PARAM> 40 </PARAM></DECLARATION>`	`<DECLARATION><OBJ_DIR> minimize </OBJ_DIR><OBJ_NAME> total amount of diesel </OBJ_NAME> [is] <VAR> new company </VAR> [TIMES] <PARAM> 30 </PARAM><VAR> old company </VAR> [TIMES] <PARAM> 40 </PARAM></DECLARATION>`
`<DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 1000 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> new company </VAR> [TIMES] <PARAM> 50 </PARAM><VAR> old company </VAR> [TIMES] <PARAM> 70 </PARAM></DECLARATION>`	`<DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 1000 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> new company </VAR> [TIMES] <PARAM> 50 </PARAM><VAR> old company </VAR> [TIMES] <PARAM> 70 </PARAM></DECLARATION>`
`<DECLARATION><CONST_DIR> at most </CONST_DIR><LIMIT> 15 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [UPPER_BOUND] </CONST_TYPE> [for] <VAR> new company </VAR></DECLARATION>`	`<DECLARATION><CONST_DIR> at most </CONST_DIR><LIMIT> 15 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> new company </VAR> [TIMES] <PARAM> 50 </PARAM><VAR> old company </VAR> [TIMES] <PARAM> 70 </PARAM></DECLARATION>`
`<DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 40% </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [RATIO_CONSTRAINT] </CONST_TYPE> [for] <VAR> old company </VAR></DECLARATION>`	`<DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 1000 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> new company </VAR> [TIMES] <PARAM> 50 </PARAM><VAR> old company </VAR> [TIMES] <PARAM> 70 </PARAM></DECLARATION>`

Table 19: Comparison of predicted vs. gold IR for out-of-domain example (Transportation example). In this example, the model incorrectly generated the expressions for the last two constraints. In fact, it detects the wrong constraint types and produced the same invalid algebraic expression. This suggests that the generation could be made more precise by adding additional constrain context into the declaration prompt to distinguish between different constraints.

| A zoo needs to transport their monkeys to the vet either by bus or by car. A bus can transport 20 monkeys per trip and takes 30 minutes. A car can transport 6 monkeys per trip and takes 15 minutes. There can be at most 10 bus trips. In addition, since the monkeys get aggressive when there are too many in one place at least 60% of the trips should be by car. If the zoo needs to transport 300 monkeys, how many trips of each should be done to minimize the total time required to transport the monkeys? | |

Gold	Pred

`<DECLARATION><OBJ_DIR> minimize </OBJ_DIR><OBJ_NAME> total time </OBJ_NAME> [is] <VAR> bus </VAR> [TIMES] <PARAM> 30 </PARAM><VAR> car </VAR> [TIMES] <PARAM> 15 </PARAM></DECLARATION>`
 	
`<DECLARATION><CONST_DIR> at most </CONST_DIR><LIMIT> 10 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [UPPER_BOUND] </CONST_TYPE> [for] <VAR> bus </VAR></DECLARATION>`
 	
`<DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 60% </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [RATIO_CONSTRAINT] </CONST_TYPE> [for] <VAR> car </VAR></DECLARATION>`
 	
`<DECLARATION><CONST_DIR> needs </CONST_DIR><LIMIT> 300 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> bus </VAR> [TIMES] <PARAM> 20 </PARAM><VAR> car </VAR> [TIMES] <PARAM> 6 </PARAM></DECLARATION>`
	`<DECLARATION><OBJ_DIR> minimize </OBJ_DIR><OBJ_NAME> total time </OBJ_NAME> [is] <VAR> monkeys </VAR> [TIMES] <PARAM> 30 </PARAM><VAR> cars </VAR> [TIMES] <PARAM> 15 </PARAM></DECLARATION>`
	
	`<DECLARATION><CONST_DIR> at most </CONST_DIR><LIMIT> 10 </LIMIT><OPERATOR> LESS_OR_EQUAL </OPERATOR><CONST_TYPE> [UPPER_BOUND] </CONST_TYPE> [for] <VAR> bus </VAR></DECLARATION>`
	
	`<DECLARATION><CONST_DIR> at least </CONST_DIR><LIMIT> 60% </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [RATIO_CONSTRAINT] </CONST_TYPE> [for] <VAR> car </VAR></DECLARATION>`
	
	`<DECLARATION><CONST_DIR> needs </CONST_DIR><LIMIT> 300 </LIMIT><OPERATOR> GREATER_OR_EQUAL </OPERATOR><CONST_TYPE> [LINEAR_CONSTRAINT] </CONST_TYPE> [is] <VAR> bus </VAR> [TIMES] <PARAM> 20 </PARAM><VAR> car </VAR> [TIMES] <PARAM> 6 </PARAM></DECLARATION>`

Table 20: Comparison of predicted vs. gold IR for out-of-domain example (Transportation example). In this example, the model detects an invalid decision variable "monkeys" in the predicted objective declaration.

Method	Accuracy	Rouge-1			Rouge-2			Rouge-L		
		Recall	Precision	F1	Recall	Precision	F1	Recall	Precision	F1
OptGen	0.61	0.89	0.89	0.89	0.80	0.80	0.79	0.86	0.86	0.86

Table 21: Performance of our model on development set. Rouge scores included declaration tags to illustrate how well the model is able to reproduce syntax.