Reassessing Graph Linearization for Sequence-to-sequence AMR Parsing: On the Advantages and Limitations of Triple-Based Encoding

Jeongwoo Kang[1] **Maximin Coavoux**[1] **Cédric Lopez**[2] **Didier Schwab**[1]
[1]Univ. Grenoble Alpes, CNRS, Grenoble INP, LIG, 38000 Grenoble, France
[2]Emvista, Immeuble Le 610, 10 Rue Louis Breguet Bâtiment D, 34830 Jacou, France
[1]{firstname}.{lastname}@univ-grenoble-alpes.fr
[2]{firstname}.{lastname}@emvista.com

Abstract

Sequence-to-sequence models are widely used to train Abstract Meaning Representation (Banarescu et al., 2013, AMR) parsers. To train such models, AMR graphs have to be linearized into a one-line text format. While Penman encoding is typically used for this purpose, we argue that it has limitations: (1) for deep graphs, some closely related nodes are located far apart in the linearized text (2) Penman's tree-based encoding necessitates inverse roles to handle node re-entrancy, doubling the number of relation types to predict. To address these issues, we propose a triple-based linearization method and compare its efficiency with Penman linearization. Although triples are well suited to represent a graph, our results suggest room for improvement in triple encoding to better compete with Penman's concise and explicit representation of a nested graph structure.

1 Introduction

Abstract Meaning Representation (AMR) captures text meaning, such as "who does what to whom," and represents it in graphs (see Figure 1). Structured information is easier for computers to process and therefore, AMR is widely used in NLP applications, e.g., machine translation (Wein and Schneider, 2024), text generation (Huang et al., 2023), or human-robot interaction systems (Bonial et al., 2019, 2023).

Sequence-to-sequence (seq2seq) approaches have recently gained popularity for AMR parsing due to strong performance and easy implementation. For prediction, the model receives an input sentence and outputs an AMR graph in text format. To train seq2seq models for AMR parsing, graph linearization to represent an AMR graph in a one-line text format is a prerequisite. Penman encoding is the most common method for AMR graph linearization, representing graphs as *tree*-like structures. It uses variables (e.g. `s`, `s2` in Figure 2)

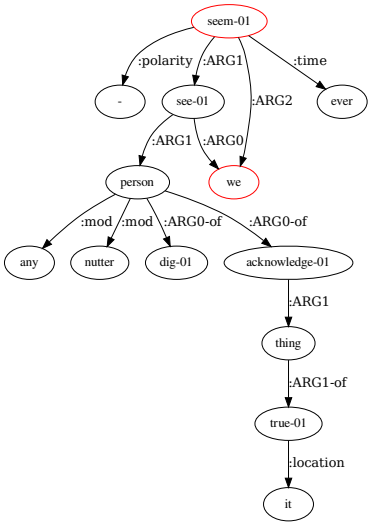

Figure 1: AMR graph for "We never seem to see any of the dug-in nutters acknowledge the truth in it." Example from the AMR 3.0 dataset (Knight et al., 2020).

```
(s / seem-01 :polarity -
        :ARG1 (s2 / see-01
            :ARG0 w
            :ARG1 (p / person
                    :mod (a / any)
                    :mod (n / nutter)
                    :ARG0-of (d / dig-01)
                    :ARG0-of (a2 / acknowledge-01
                        :ARG1 (t / thing
                                :ARG1-of (t2 / true-01
                                    :location (i / it))))))
        :ARG2 (w / we)
        :time (e / ever))
```

Figure 2: AMR in Penman encoding for Figure 2.

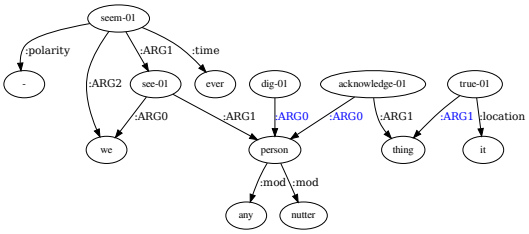

Figure 3: AMR graph without inverse roles.

The Sixth Workshop on Insights from Negative Results in NLP, pages 15–23

as node IDs to manage co-references. In addition, parentheses represent nested structures of AMR graphs. However, Penman has key limitations to training a seq2seq model:

1) **Parent-Child Distance**: Parent and child nodes may appear far apart in the linearized text, despite being closely connected in the graph. For example, in Figure 2, `seem-01` and `we` are encoded distant in Penman format (highlighted in red) despite their proximity in the graph of Figure 2. This is observed when a preceding sibling node has a deep sub-graph. We hypothesize that this long distance increases the difficulty of learning strong parent-child connections, especially in deeper graphs. 2) **Inverse Roles**: Penman represents a graph in a tree-based format. To be specific, when a node has multiple parent nodes (node reentrancy), the child node is duplicated to maintain a single-rooted tree structure. To fit an AMR graph into a tree structure, Penman introduces inverse roles by rewriting `:relation` as `:relation-of` (see Figure 2 where inverse roles are highlighted in blue). This increases the number of relations the model must learn, potentially complicating training and reducing model performance. Figure 3 shows how inverse roles are unnecessary in a graph-based representation.

To address these issues, we propose an alternative triple-based format for AMR graph linearization. A triple consists of a parent node, a child node, and a relation type between them, ensuring that parent and child nodes remain adjacent in the linearized text. This format also eliminates inverse roles by replacing `(node A, relation-of, node B)` with `(node B, relation, node A)`. In the rest of the paper, we compare Penman and triple-based formats with examples, highlighting their strengths and limitations in training a seq2seq AMR parser. Our contributions to seq2seq AMR parsing are:

- A triple-based linearization method for training seq2seq AMR parsers.
- A detailed comparison with Penman linearization, focusing on performance across varying graph depths and lengths, and identifying areas for improvement.

2 Related Work

Triple encoding has been used in relation extraction (Huguet Cabot and Navigli, 2021; Ye et al., 2021; Saxena et al., 2022) and discourse representation structure (DRS) parsing (van Noord et al., 2018a,b). The closest to our approach is van Noord et al. (2018a), who convert AMR graphs into DRS triples. However, their representation differs from ours by mapping AMR relations to DRS roles and adding extra information that does not exist in AMR. In addition, they did not include training an AMR parser in their work. While triple encoding is widely used in relation extraction and DRS parsing, it has not been used for seq2seq AMR parsers. In our work, we propose using it to linearize AMR graphs, analyzing its strengths and weaknesses as an encoding method.

```
s instance seem-01          s2 instance see-01   p instance person
a instance any              n instance nutter    d instance dig-01
a2 instance acknowledge-01  t instance thing     t2 instance true-01
i instance it               w instance we        e instance ever
s polarity -                s ARG1 s2            s2 ARG0 w
s2 ARG1 p                   p mod a              p mod n
d ARG0 p                    a2 ARG0 p            a2 ARG1 t
t2 ARG1 t                   t2 location i        s ARG2 w
s time e
```

Figure 4: Triple linearization of the graph in Figure 2.

3 Methodology: Triple Linearization

Triple representation mitigates the challenges of Penman linearization (described in Section 1) by encoding a graph as a set of triples. Figure 4 illustrates that the parent-child nodes, which were distantly located in Figure 2, are now adjacent in the triple representation (highlighted in red). We hypothesize this helps the model learn direct parent-child relationships, especially in deeper graphs. Triples also eliminate inverse roles by reversing the order of two nodes, as shown in Figure 4 (highlighted in blue).

The triple format, widely used for graph representation (e.g., RDF), aligns better with AMR's graph structure than Penman's tree-based encoding. Its flexibility supports graphs with multiple roots or re-entrancies, making it potentially suitable for broader semantic frameworks. To assess its utility, we trained seq2seq AMR parsers using triple and Penman formats.

Despite its advantages, triple linearization can result in verbose linearization, slowing down the learning process, and may be less effective at capturing nested structures of graphs compared to Penman. This study evaluates both linearization methods to train seq2seq AMR parsers, exploring: **(1)** whether triple representation improves AMR parsing; **(2)** which graphs benefit most from triple representation, such as those with deep structures or large size, and **(3)** if combining triple and Penman

representations enhances parsing performance.

Experiments involve training models respectively with triple, Penman, and both formats (multi-task learning). Using both formats may serve as a form of data augmentation, as it effectively doubles the training data by representing one example in two linearized formats. We train and evaluate our model with English AMR 3.0 (Knight et al., 2020) data. We evaluate our model using SMATCH (Cai and Knight, 2013) score by counting the matching triples between two graphs. We analyze results by graph depth and size to determine which types of graphs benefit from different encoding methods.

Triple linearization strategies To linearize AMR graphs in triples, we extract all triples and unfold them in depth-first search order using the PENMAN library.[1] Four linearization strategies are applied, varying in whether variables[2] or inverse roles are retained. We provide an example for each linearization type in Table 3 and Figure 7 of Appendix A. Each model is named based on linearization type as follows:

- **Triple_X_var_X_invrole**: Variables and inverse roles are removed. Variables are replaced by node names, and inverse roles are converted by reversing node order.[3] Reversing inverse roles reduces the number of relation types from 155 to 115 in our training data. Triples are separated by a pipe symbol (|).
- **Triple_X_var_O_invrole**: Variables are removed, but inverse roles are retained.
- **Triple_O_var_O_invrole**: Both variables and inverse roles are retained. Variables and their instances are represented as triples with the `instance` relation (e.g., `f instance fruit`). This approach is the most comprehensive, as no information is lost from the original graph during linearization.
- **Triple_O_var_X_invrole**: Variables are retained, but inverse roles are removed.

[1] `https://penman.readthedocs.io/en/`, version 1.3.0

[2] Removing variables is a common pre-processing strategy for seq2seq AMR parsing (Konstas et al., 2017; van Noord and Bos, 2017). This leads to information loss but effectively reduces data sparsity for training.

[3] For the models discussed in this article, variables and inverse roles are removed in this manner.

4 Experimental Setup

Models are trained using the large mBART model (Tang et al., 2021) on each linearization type.[4]

4.1 Baseline

To compare our method with existing approaches using Penman encoding, we trained a model on AMR graphs linearized using Penman encoding, which serves as our baseline. Note that maintaining inverse roles is a necessary aspect of Penman encoding and **X_invrole** types are not available for Penman encoding. For training, we employed the same mBART model and trained two models as follows (see Table 3 for examples):

- **Penman_X_var_O_invrole**: Variables are removed, but inverse roles are retained.[5]
- **Penman_O_var_O_invrole**: Both variables and inverse roles are retained.

4.2 Multi-task Learning

As mentioned in Section 1, combining triple and Penman encodings may offer complementary benefits. To test this, we trained models in a multi-task learning framework by merging two differently encoded datasets. During training, the model learns from shuffled examples with a token indicating the encoding type. For predictions, the model is instructed to use either triple or Penman encoding to use the corresponding decoding strategy to reconstruct graphs. We trained four models:

- **Multi_tri_O_var_O_invrole**: Both variables and inverse roles are retained. The main task is triple encoding, with Penman encoding as an auxiliary task. This means that the best model is selected based on the performance on the validation set using triple encoding, while Penman encoding is treated as an auxiliary task to help the triple learning.
- **Multi_penman_O_var_O_invrole**: Both variables and inverse roles are retained. The main task is Penman encoding, with triple encoding as an auxiliary.
- **Multi_tri_X_var_O_invrole**: Variables are removed but inverse roles are retained. The best model is chosen based on the model's performance in triple prediction.

[4] The model was chosen based on our goal of developing a multilingual system, which was not covered in this article.

[5] We used the script from van Noord and Bos (2017).

Model	*The Little Prince*	AMR 3.0
Triple_O_var_O_invrole	76.2 ± 0.3	80.0 ± 0.2
Triple_O_var_X_invrole	76.1 ± 0.5	80.3 ± 0.1
Triple_X_var_O_invrole	76.5 ± 0.2	78.7 ± 0.1
Triple_X_var_X_invrole	76.2 ± 0.2	78.9 ± 0.8
Penman_O_var_O_invrole	**77.0** ± 0.4	**80.9** ± 0.2
Penman_X_var_O_invrole	76.7 ± 0.1	80.2 ± 0.1
Multi_tri_O_var_O_invrole	<u>76.9</u> ± 0.2	80.3 ± 0.1
Multi_tri_X_var_O_invrole	76.3 ± 0.2	78.9 ± 0.1
Multi_penman_O_O_invrole	76.7 ± 0.2	<u>80.6</u> ± 0.3
Multi_penman_X_var_O_invrole	76.1 ± 0.1	79.8 ± 0.1

Table 1: SMATCH scores for evaluation (with the highest scores in **bold** and the second-highest scores <u>underlined</u>).

- **Multi_penman_X_var_O_invrole**: Variables are removed but inverse roles are retained. The best model is selected based on performance in Penman prediction.

5 Results and Insights

Global results. Table 1 presents results on two test sets: *The Little Prince*[6] and AMR 3.0. The Penman single-task model with variables (**Penman_O_var_O_invrole**) performs best on both test sets (77.0 and 80.9, respectively). Among single-task triple models, **Tri_O_var_O_invrole** achieves the best results, with a marginal gap from the best model ($\leq$1 SMATCH).

Preserving variables consistently improves performance, contradicting the assumption that removing them aids learning by reducing data sparsity. This suggests variable removal leads to critical information loss. In addition, learning from Penman encoding while performing an auxiliary triple task reduces performance, whereas the reverse improves it. This indicates Penman encoding provides structural information beneficial to triple encoding but not vice versa.

Within triple linearization, removing inverse roles improves performance on AMR 3.0 but not *The Little Prince*. Given that AMR 3.0 includes longer sentences, removing inverse roles may benefit longer sentences while harming shorter ones. However, the inconsistency across test sets could also suggest inverse roles have only a marginal effect.

Triple linearization does not improve learning for deeper graphs. We hypothesized that triple linearization could enhance training by positioning child and parent nodes closer, especially when a preceding sibling has a deep subgraph. In Penman encoding, such cases place these nodes farther apart. Assuming this issue is more common in deeper graphs, we analyzed results by reference graph depth (distance from the root to the furthest node), focusing on AMR 3.0, which has greater depth variety than *The Little Prince*.

Our results (Figure 5, Appendix) show SMATCH scores by depth align with overall scores, indicating no learning benefit for deeper graphs with triple encoding. The best model, **Penman_O_var_O_inverserole**, consistently performed best across depths. This contradicts our hypothesis that triple encoding benefits seq2seq AMR learning by bringing parent-child nodes closer together. Instead, the results emphasize the benefit of Penman's concise graph representation and its ability to explicitly encode nested structures, which play a more critical role in model performance.

Triple linearization does not improve learning for longer graphs. We also analyzed performance by graph length (i.e., token count in the linearized reference graph), assuming verbose triple encoding would degrade performance on longer graphs. Since graph depth and length are not always correlated, results may differ from the depth analysis. Figure 6 in the Appendix shows the results per graph length with token counts of reference graphs grouped into buckets of 50 for clarity. For shorter graphs, the gap between models is smaller, but as the graphs become longer, the performance of triple models decreases more noticeably, resulting in a wider gap. This supports our earlier hypothesis regarding the limitations of triple encoding: its verbose representation likely contributes to the observed performance degradation on lengthy graphs.

6 Conclusion

We introduced triple linearization as an alternative to Penman linearization, hypothesizing that it could improve training for several reasons: (1) parent and child nodes are always located together in a triple, (2) the elimination of inverse roles may simplify training by reducing the number of relations, (3) and triples more closely resemble the underlying graph structure, while Penman encoding represents a graph in a more tree-like format. Contrary to our hypothesis, Penman has proven to be a more effective linearization method to train a seq2seq parsing.

[6] https://github.com/flipz357/AMR-World

However, the gap between the best Penman model and certain triple-based models is marginal. Our results show a potential to train a seq2seq AMR parser that predicts a graph directly (not a tree-based representation) while maintaining equivalent performance. Notably, the model's output in triples more naturally aligns with AMR's graph structure than Penman. Our code to train and evaluate the model is available on https://github.com/Emvista/Triple_AMR_Parser.git.

Acknowledgments

We thank reviewers for their helpful comments and Cécile Macaire for proof-reading this article. Jeongwoo Kang and Maximin Coavoux gratefully acknowledge the support of the French National Research Agency (grant ANR-23-CE23-0017-01). This work was granted access to the HPC resources of IDRIS under the allocation 2024-AD011012853R2 made by GENCI.

References

Laura Banarescu, Claire Bonial, Shu Cai, Madalina Georgescu, Kira Griffitt, Ulf Hermjakob, Kevin Knight, Philipp Koehn, Martha Palmer, and Nathan Schneider. 2013. Abstract Meaning Representation for sembanking. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 178–186, Sofia, Bulgaria. Association for Computational Linguistics.

Claire Bonial, Julie Foresta, Nicholas C. Fung, Cory J. Hayes, Philip Osteen, Jacob Arkin, Benned Hedegaard, and Thomas Howard. 2023. Abstract Meaning Representation for grounded human-robot communication. In *Proceedings of the Fourth International Workshop on Designing Meaning Representations*, pages 34–44, Nancy, France. Association for Computational Linguistics.

Claire N. Bonial, Lucia Donatelli, Jessica Ervin, and Clare R. Voss. 2019. Abstract Meaning Representation for human-robot dialogue. In *Proceedings of the Society for Computation in Linguistics (SCiL) 2019*, pages 236–246.

Shu Cai and Kevin Knight. 2013. Smatch: an evaluation metric for semantic feature structures. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 748–752, Sofia, Bulgaria. Association for Computational Linguistics.

Kuan-Hao Huang, Varun Iyer, I-Hung Hsu, Anoop Kumar, Kai-Wei Chang, and Aram Galstyan. 2023. ParaAMR: A large-scale syntactically diverse paraphrase dataset by AMR back-translation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8047–8061, Toronto, Canada. Association for Computational Linguistics.

Pere-Lluís Huguet Cabot and Roberto Navigli. 2021. REBEL: Relation extraction by end-to-end language generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2370–2381, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Kevin Knight, Bianca Badarau, Laura Baranescu, Claire Bonial, Madalina Bardocz, Kira Griffitt, Ulf Hermjakob, Daniel Marcu, Martha Palmer, Tim O'Gorman, and Nathan Schneider. 2020. Abstract meaning representation (amr) annotation release 3.0 ldc2020t02. Philadelphia: Linguistic Data Consortium.

Ioannis Konstas, Srinivasan Iyer, Mark Yatskar, Yejin Choi, and Luke Zettlemoyer. 2017. Neural AMR: Sequence-to-sequence models for parsing and generation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 146–157, Vancouver, Canada. Association for Computational Linguistics.

Apoorv Saxena, Adrian Kochsiek, and Rainer Gemulla. 2022. Sequence-to-sequence knowledge graph completion and question answering. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2814–2828, Dublin, Ireland. Association for Computational Linguistics.

Yuqing Tang, Chau Tran, Xian Li, Peng-Jen Chen, Naman Goyal, Vishrav Chaudhary, Jiatao Gu, and Angela Fan. 2021. Multilingual translation from denoising pre-training. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3450–3466, Online. Association for Computational Linguistics.

Rik van Noord, Lasha Abzianidze, Hessel Haagsma, and Johan Bos. 2018a. Evaluating scoped meaning representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*, Miyazaki, Japan. European Language Resources Association (ELRA).

Rik van Noord, Lasha Abzianidze, Antonio Toral, and Johan Bos. 2018b. Exploring neural methods for parsing discourse representation structures. *Transactions of the Association for Computational Linguistics*, 6:619–633.

Rik van Noord and Johan Bos. 2017. Neural semantic parsing by character-based translation: Experiments with abstract meaning representations. *Computational Linguistics in the Netherlands Journal*, 7:93–108.

Shira Wein and Nathan Schneider. 2024. Lost in translationese? reducing translation effect using Abstract Meaning Representation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long*

Papers), pages 753–765, St. Julian's, Malta. Association for Computational Linguistics.

Hongbin Ye, Ningyu Zhang, Shumin Deng, Mosha Chen, Chuanqi Tan, Fei Huang, and Huajun Chen. 2021. Contrastive triple extraction with generative transformer. In *Proceedings of the AAAI conference on artificial intelligence*, volume 35, pages 14257–14265.

A Appendix

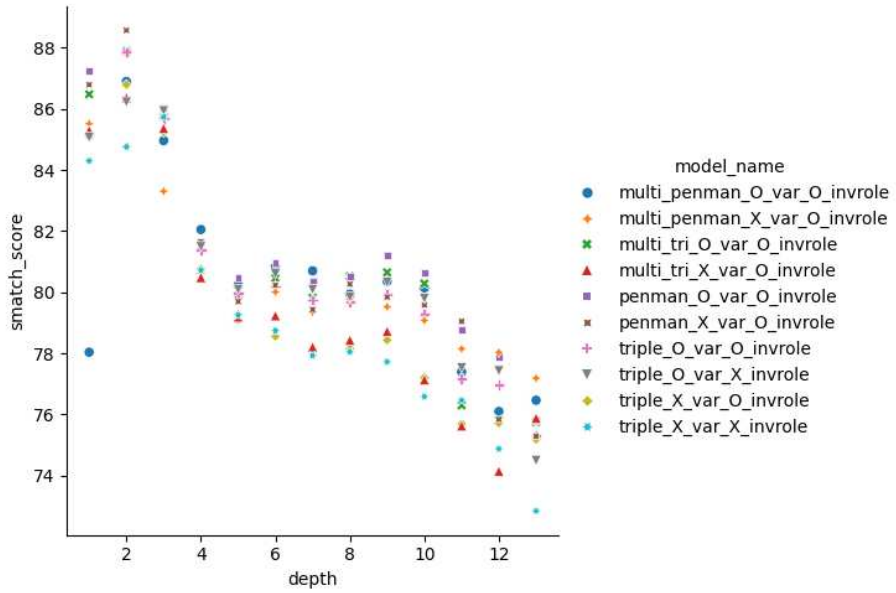

Figure 5: SMATCH score per graph depth.

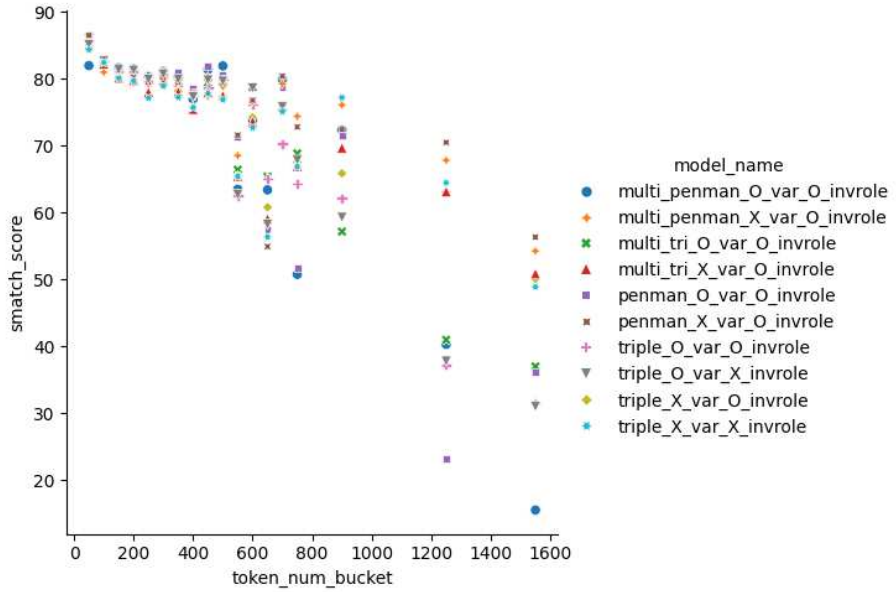

Figure 6: SMATCH score per graph length. The length is measured by the number of tokens in a linearized graph and token counts are grouped into buckets of 50 for clarity.

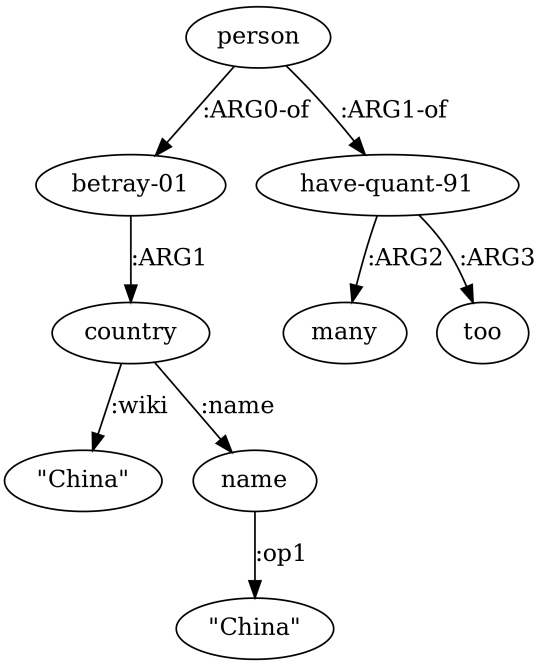

Figure 7: AMR graph for “There are too many traitors of China!”. This example is drawn from AMR 3.0 dataset (Knight et al., 2020).

Triple_X_var_X_invrole	Triple_X_var_O_invrole	Triple_O_var_O_invrole	Triple_O_var_X_invrole
person ARG0-of betray-01 | betray-01 ARG1 country | country name " China " | person ARG1-of have-quant-91 | have-quant-91 ARG2 many | have-quant-91 ARG3 too	betray-01 ARG0 person | betray-01 ARG1 country | country name " China " | have-quant-91 ARG1 person | have-quant-91 ARG2 many | have-quant-91 ARG3 too	p instance person | b instance betray-01 | c instance country | h instance have-quant-91 | m instance many | t instance too | p ARG0-of b | b ARG1 c | c name " China " | p ARG1-of h | h ARG2 m | h ARG3 t	p instance person | b instance betray-01 | c instance country | h instance have-quant-91 | m instance many | t instance too | b ARG0 p | b ARG1 c | c name " China " | h ARG1 p | h ARG2 m | h ARG3 t

Table 2: Triple encoding examples of Figure 7.

Penman_X_var_O_invrole	Penman_O_var_O_invrole
(person :ARG0-of (betray-01 :ARG1 (country :name " China ")) :ARG1-of (have-quant-91 :ARG2 (many) :ARG3 (too)))	b / person :ARG0-of (b / betray-01 :ARG1 (c / country :name " China " Salut Maximin, j'ai soumis la version actuelle (changement du titre, ajouter des commentaires de Cédroc dans la conclusion) au workshop)) :ARG1-of (h / have-quant-91 :ARG2 (m / many) :ARG3 (t / too)))

Table 3: Penman encoding examples of Figure 7.