Natural Language Deduction with Incomplete Information

Zayne Sprague **Kaj Bostrom** **Swarat Chaudhuri** **Greg Durrett**
Department of Computer Science
The University of Texas at Austin
zaynesprague@utexas.edu, {kaj,swarat,gdurrett}@cs.utexas.edu

Abstract

A growing body of work studies how to answer a question or verify a claim by generating a natural language "proof": a chain of deductive inferences yielding the answer based on a set of premises. However, these methods can only make sound deductions when they follow from evidence that is given. We propose a new system that can handle the underspecified setting where not all premises are stated at the outset; that is, additional assumptions need to be materialized to prove a claim. By using a natural language generation model to abductively infer a premise given another premise and a conclusion, we can impute missing pieces of evidence needed for the conclusion to be true. Our system searches over two fringes in a bidirectional fashion, interleaving deductive (forward-chaining) and abductive (backward-chaining) generation steps. We sample multiple possible outputs for each step to achieve coverage of the search space, at the same time ensuring correctness by filtering low-quality generations with a round-trip validation procedure. Results on a modified version of the EntailmentBank dataset and a new dataset called *Everyday Norms: Why Not?* show that abductive generation with validation can recover premises across in- and out-of-domain settings.[1]

1 Introduction

Substantial prior work in domains like question answering (Rajpurkar et al., 2016; Yang et al., 2018; Kwiatkowski et al., 2019), textual entailment (Bowman et al., 2015; Williams et al., 2018; Nie et al., 2020), and other types of reasoning (Clark et al., 2021; Dalvi et al., 2021) deals with making inferences from stated information, where we draw conclusions and answer questions based on textual context provided directly to a model. However, a growing body of research studies the problem of reasoning given incomplete information, especially for tasks labeled as commonsense reasoning (Talmor et al., 2019; Rajani et al., 2019). Current approaches in these domains often work through latent reasoning by large language models (Lourie et al., 2021), with only a few explicitly materializing the missing knowledge (Bosselut et al., 2019; Bhagavatula et al., 2020; Arabshahi et al., 2021; Liu et al., 2022; Katz et al., 2022). However, making knowledge explicit is critical to make reasoning processes *explainable*: it allows users to critique those explanations and allows systems to reuse inferred knowledge across scenarios (Dalvi et al., 2022).

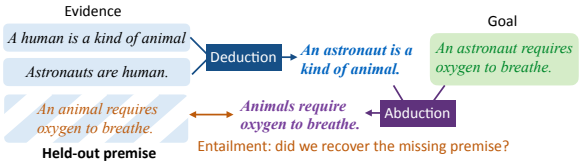

Figure 1: An example of deductive (previous work) and abductive (our work) reasoning used together to search for missing evidence needed to entail a goal in a depth 2 tree from EntailmentBank.

The materialization of new knowledge is naturally formulated as *abduction*: generating an explanation given a premise and a conclusion. Abductive reasoning as a text generation task is fundamentally challenging, as it is an underspecified task with a large search space of valid generations, hence why prior work has framed it as a multiple-choice problem (Bhagavatula et al., 2020). Nevertheless, the freeform generation setting is the one that real-world explainable reasoning systems are faced with.

In this paper, we develop an approach that combines abductive reasoning with multistep deductive reasoning. We build on recent discrete search-based approaches that construct *entailment*

[1]Code and data publicly available at https://github.com/Zayne-sprague/Natural_Language_Deduction_with_Incomplete_Information.git

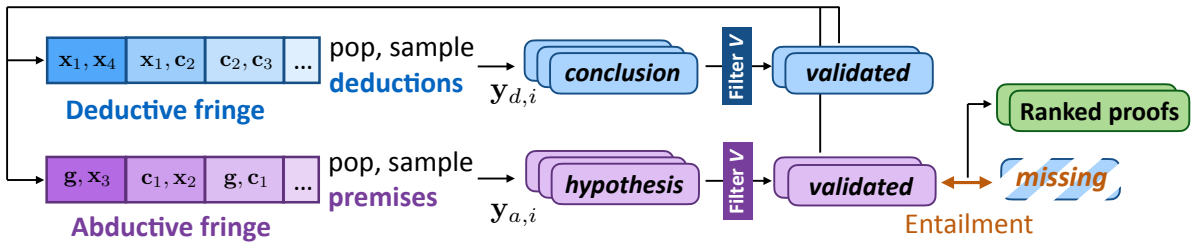

Figure 2: Overview of the ADGV system and its components. First, on the left, priority queues of possible deductive (blue) and abductive (purple) step inputs give the highest-scoring step inputs for each step type. Multiple generations are produced at each step, and each sample is validated and discarded if invalid (middle). Finally (right), the new validated samples are pushed onto the queues following the rules in Table 6 and we optionally check whether a particular missing premise has been recovered in our experiments.

trees (Dalvi et al., 2021; Bostrom et al., 2022; Yang and Deng, 2021; Hong et al., 2022) to represent deductive inferences in natural language. Although more transparent than discriminative end-to-end models, these methods have so far required all necessary premises to be explicitly provided, and cannot account for abductive reasoning.

Our input is a set of *incomplete* premise facts and a goal; our algorithm searches forward from the premises and backwards from the goal to build a proof that entails the goal *and* recovers a missing premise through a combination of deductive and abductive inferences. Figure 1 shows an example. To constrain the model's generation, we incorporate a validation criterion to test the consistency of each logical inference. We call this new system ADGV (Abduction and Deductive Generation with Validation, Figure 2). At its core, ADGV follows a similar heuristic search to Bostrom et al. (2022), iteratively generating conclusions and adding them to the search frontier, but incorporates abductive steps (analogous to backward-chaining) to make the search two-sided.

We evaluate on a new task variant that requires recovering a missing premise from a subset of textual evidence and a goal. We use two datasets: EntailmentBank (Dalvi et al., 2021) and *Everyday Norms: Why Not?*, a new dataset that we construct that requires combining information about situations with general principles. We assess both *coverage* of held-out premises on our test examples and *step validity* of the steps used to construct them, thereby establishing the ability of ADGV to recover premises as well as construct entailment trees reaching the goal of the original example. Although our approach can reconstruct premises with a high validity rate, achieving high coverage has significant headroom for future work.

Our contributions are: (1) introduction of a new task for natural language understanding, recovering a premise in an underspecified entailment tree, along with a new dataset, *Everyday Norms: Why Not?*; (2) a new abductive step model and ADGV inference method, which combines forward and backward search; (3) new validation techniques that improve step validity in these models.

2 Problem Description

We study the task of generating a natural language proof tree T that entails a goal $\mathbf{g}$ given a set of textual evidence $X = \{\mathbf{x}_1 \ldots \mathbf{x}_n\}$. Unique to our work, we remove one of the pieces of textual evidence $\mathbf{x}_m$ creating an underspecified setting where a deduction system operating over stated premises (Dalvi et al., 2021; Bostrom et al., 2022) cannot build an entailment tree capable of reaching the goal. The task is then to prove the goal $\mathbf{g}$ while also recovering $\mathbf{x}_m$, which requires searching backwards from the goal to generate missing information. An overview of our abductive reasoning system can be seen in Figure 2.

Note that there is a trivial solution to this problem, which is to immediately assume that $\mathbf{x}_m = \mathbf{g}$, leading to a vacuous proof. There is no easy way to rule out this solution, as it is hard to come up with a first-order principle for what makes an atomic premise. In existing datasets like EntailmentBank (Dalvi et al., 2021), premises can be low-level definitions ("*revolving around means orbiting*") or more complex process descriptions ("*Photosynthesis means producers / green plants convert from carbon dioxide and water and solar energy into carbohydrates and food and oxygen for themselves*"). Other past work (Dalvi et al., 2022; Weir and Van Durme, 2022) has use large language models to determine atomicity, but this also fails

to yield a consistent principle beyond preferring statements that are attested in large web corpora.

As a result, we will use our search procedure to iteratively unroll a goal into simpler statements in an attempt to recover the specific premise $\mathbf{x}_m$ with *some* tree that we find. We will evaluate according to two criteria. Our first criterion is **recall** of the missing premise at some point along the search process, using a scoring metric $E(\mathbf{x'}_i, \mathbf{x}_m) \in \mathbb{R}$ to determine if a generated premise $\mathbf{x'}_i$ is logically equivalent to $\mathbf{x}_m$. Our second criterion is **validity** of the tree that yields $\mathbf{x}_m$, judged according to human ratings.

3 Methods

Our approach is based on two generative modules called *step models*. Our deductive step model S_d defines a probability distribution $P_{S_d}(\mathbf{y} \mid \mathbf{x}_1 \ldots \mathbf{x}_n)$ over valid conclusions $\mathbf{y}$ given premises $\mathbf{x}_1 \ldots \mathbf{x}_n$, all of which are represented as natural language strings. We use the same notion of deduction as in past work (Bostrom et al., 2021), where the model should place probability mass over correct and useful conclusions that can be inferred from the premises (i.e., not simply copying a premise). Following past work, we set $n = 2$, which we find sufficient to handle both of our datasets.

New in this work, we additionally introduce an *abductive* step model $S_a = P_{S_a}(\mathbf{x'} \mid \mathbf{x}_1 \ldots \mathbf{x}_n, \mathbf{c})$. This model is meant to "reverse" the behavior of the forward model in a similar fashion as backward-chaining in Prolog (Robinson, 1965). Specifically, it takes a conclusion statement $\mathbf{c}$ as well as one premise $\mathbf{x}$ and generates a hypothesis $\mathbf{x'}$. The generated hypothesis, $\mathbf{x'}$, constitutes a new piece of information that the step model infers is necessary to make the original conclusion $\mathbf{c}$ true. This operation can then be chained repeatedly to uncover more general and abstract information. We find in our work that setting $n = 1$ (one premise and a conclusion $\mathbf{c}$) is sufficient.

Deductive inferences in the domains we consider may be lexically underspecified, but typically represent a clear logical operation. However, abduction does not. An example can be seen in Figure 3: the abductive model can produce multiple valid generations at varying levels of specificity. Determining the truth of these generations is extremely challenging as in other work that tries to generate intermediate unstated inferences (Rajani et al., 2019; Wiegreffe et al., 2022; Dalvi et al., 2022; Liu et al., 2022). To mitigate this, we introduce round-trip validators which enforce the condition that the forward and abductive models' generations must agree.

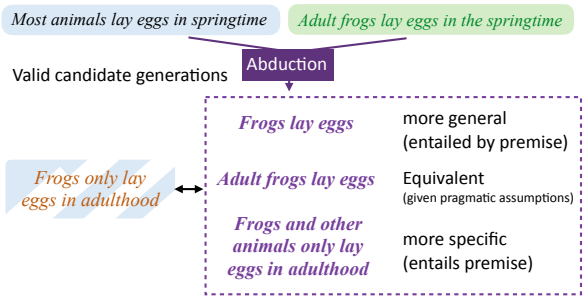

Figure 3: The abductive model can generate numerous valid inferences from a premise (blue) and goal (green) which can relate to the reference premise in a few ways: mutual entailment (middle generation) and entailment in either direction based on whether the generation is more general or more specific than the missing premise.

Models and Data Our system revolves around structuring the application of the two step models, S_d and S_a, with a search procedure. We first describe the mechanics of the step models and then the heuristic search procedure, which employs two heuristics H^d and H^a to guide which step generation to perform next.

Both models are trained on data from EntailmentBank (Dalvi et al., 2021). Following (Bostrom et al., 2022), we do not rely on complete trees from EntailmentBank to train the step models, but instead view a tree T as a collection of steps $T_i = (\mathbf{x}_{i,1}, \ldots, \mathbf{x}_{i,n} \rightarrow \mathbf{c}_i)$.

3.1 Step Models

Our **abductive step model** is an instance of a pre-trained language model. We specialize it to map from a conclusion statement and a single premise to a hypothesized missing premise, yielding the distribution $p_{S_a}(\mathbf{x'} \mid \mathbf{x}, \mathbf{c})$.

The abductive step model is trained on the EntailmentBank dataset by converting each step $T_i = (\mathbf{x}_1, \mathbf{x}_2 \rightarrow \mathbf{c})$ into multiple abductive steps by ablating each input in turn: $\mathbf{x}_1, \mathbf{c} \rightarrow \mathbf{x}_2$ and $\mathbf{x}_2, \mathbf{c} \rightarrow \mathbf{x}_1$. We ensure the conclusion $\mathbf{c}$ is always appended at the end of the input so the model can learn asymmetric relationships between premises and the input conclusion. The model is trained with teacher forcing to generate exactly the correct premise; however, during inference we sample as many as $k = 40$ generations from the abductive

Algorithm 1 Abductive and Deductive Generation with Validation

Inputs: a collection of premises X, a goal $\mathbf{g}$, and maxSteps
procedure ADGV($X = \{\mathbf{x}_1 \dots \mathbf{x}_n\}$, $\mathbf{g}$, maxSteps):
  $fringe_d \leftarrow \{(\mathbf{x}_i, \mathbf{x}_j) | \ \mathbf{x}_i, \mathbf{x}_j \in X, i \neq j\}$
  $fringe_a \leftarrow \text{pairs}(\mathbf{g}, X)$
  $seen_d \leftarrow X$
  $seen_a \leftarrow \{\mathbf{g}\}$
  $i \leftarrow 1$
  while $|fringe_d| + |fringe_a| > 0 \wedge i \leq \text{maxSteps}$ **do**
    $i \leftarrow i + 1$
    $(\mathbf{x}_{d,1}, \mathbf{x}_{d,2}) \leftarrow \text{pop argmax}_{H_d}(fringe_d)$
    $(\mathbf{c}_a, \mathbf{x}_a) \leftarrow \text{pop argmax}_{H_a}(fringe_a)$
    Sample $\mathbf{y}_{d,1}, \dots, \mathbf{y}_{d,k'} \sim p_{S_d}(\mathbf{y} \mid \mathbf{x}_{d,1}, \mathbf{x}_{d,2})$
    Sample $\mathbf{y}_{a,1}, \dots, \mathbf{y}_{a,k} \sim p_{S_a}(\mathbf{y} \mid \mathbf{c}_a, \mathbf{x}_a)$
    for each $\mathbf{y}_{d,j}$ **do**
      if $\mathbf{y}_d \notin seen_d \wedge V(\mathbf{x}_{d,1}, \mathbf{x}_{d,2}, \mathbf{y}_d)$ **then**
        $seen_d \leftarrow seen_d \cup \{\mathbf{y}_d\}$
        $fringe_d \leftarrow fringe_d \cup \text{pairs}(\mathbf{y}_d, seen_d)$
        $fringe_a \leftarrow fringe_a \cup \text{pairs}(\mathbf{y}_d, seen_a)$
    for each $\mathbf{y}_{a,j}$ **do**
      if $\mathbf{y}_a \notin seen_a \wedge V(\mathbf{x}_a, \mathbf{y}_a, \mathbf{c}_a)$ **then**
        yield $\mathbf{y}_a$
        $seen_a \leftarrow seen_a \cup \{\mathbf{y}_a\}$
        $fringe_a \leftarrow fringe_a \cup \text{pairs}(\mathbf{y}_a, seen_d)$

model to account for underspecification.

Our **deductive step model** follows Bostrom et al. (2022) and is trained in a similar fashion as the abductive step model. We fine-tune a pretrained language model to map a set of premises to a conclusion statement, giving the distribution $p_{S_d}(\mathbf{c} \mid x_1, ..., x_n)$. This model is trained on the EntailmentBank dataset only (not using data from Bostrom et al. (2021)), using intermediate steps $T_i = (\mathbf{x}_1, \mathbf{x}_2 \rightarrow \mathbf{c})$ as training examples. During inference, we sample as many as $k' = 10$ generations from the deductive model to account for underspecification.

3.2 Search

Our search relies on several modules, first selecting steps to take, then sampling generations from the different step types, validating generations, and finally populating the fringe with new generations. The search algorithm is outlined in Algorithm 1. The search operates over two fringes, an abductive and deductive fringe, which it will process in an interleaved fashion while adding new work items to both fringes. We allow the search to iterate until a specified number of steps maxSteps is reached.

Prioritizing the Fringe: Learned heuristic models During search, we order the entries in the deductive fringe according to the Learned (Goal) heuristic model from Bostrom et al. (2022). For the abductive fringe, however, we train a custom learned heuristic.[2]

To train the abductive heuristic, we produce a pool of positive abductive steps from the gold EntailmentBank train dataset by selecting an arbitrary intermediate step and pairing each of its inputs with the step's conclusion to yield a single positive example. We also produce negative samples by pairing an intermediate conclusion $\mathbf{c}$ and an arbitrary premise or other intermediate conclusion that is not part of $\mathbf{c}$'s subtree (previous inputs). The heuristic model is an instance of DeBERTa-v3 Large finetuned on all positive and negative samples. Further details are in the appendix.

Generating and Filtering We allow for multiple generations to be sampled per step to fully explore the search space; however, this may lead to either invalid or redundant generations that need to be pruned. A combination of validators $V(inputs, \mathbf{y}_i)$ remove any generations that do not meet a set of criteria, pruning their branch in the search space. The fringe is then populated using the valid generations following the rules in Table 6.

Our core validation methods to ensure logical correctness rely on a notion of **round-trip consistency**: we want deductive generations to work in reverse when plugged into the abductive model, and vice versa. More specifically, our **Deductive Agreement** module validates abductive steps, ensuring that the abductive generation (when combined with its input premise) produces the original conclusion. For example, the abductive step $(\mathbf{c}, \mathbf{x} \rightarrow \mathbf{x}')$ is validated by taking the corresponding deductive step $(\mathbf{x}, \mathbf{x}' \rightarrow \mathbf{c}')$. The validator then checks that the scoring metric $E(\mathbf{c}, \mathbf{c}')$ is above a set threshold t_d.

The **Abductive Agreement** validator ensures that each input of a deductive step can be recovered using the output of the deductive step and the other input. For example, the deductive step $(\mathbf{x}_1, \mathbf{x}_2 \rightarrow \mathbf{c})$ is validated by taking two corresponding abductive steps $(\mathbf{x}_1, \mathbf{c} \rightarrow \mathbf{x}_2')$ and $(\mathbf{x}_2, \mathbf{c} \rightarrow \mathbf{x}_1')$. The scoring metric is then checked for the two pairs $E(\mathbf{x}_1, \mathbf{x}_1')$ and $E(\mathbf{x}_2, \mathbf{x}_2')$. Both generated inputs' scores must be above a threshold t_a for the output to be considered valid.

Other Validation Methods We also used two other validators: de-duplication and consanguinity

[2]Note that adding goal conditioning to an abductive heuristic does not make sense as the model typically *already* has knowledge of the goal in its inputs.

thresholding. De-duplication removes any non-unique outputs as well as any output that is copied directly from the inputs of the step. Consanguinity thresholding looks at the "ancestry" of a generation up to depth η and blocks generating from any pair that shares a given statement in their ancestry. We set $\eta = 1$ to prevent combination of two of the same statement; higher thresholds did not help.

3.3 Premise Recovery Scoring

When the search concludes, we score each abductive generation $\mathbf{x}'$ to test for the recovery of $\mathbf{x}_m$ through a scoring metric $E(\mathbf{x}', \mathbf{x}_m)$ which we then filter to candidates that pass a threshold t_m. To score each abduction, our system uses a harmonic mean $s = E(\mathbf{x}_m, \mathbf{x}') = \frac{2s_r s_e}{s_r + s_e}$ of $s_r = \text{ROUGE-1}(\mathbf{x}_m, \mathbf{x}')$ and an entailment scoring $s_e = entailed(\mathbf{x}', \mathbf{x}_m)$ according to an entailment model. Every $\mathbf{x}'$ that recovers $\mathbf{x}_m$ has exactly one corresponding derivation that entails the goal, so we can associate it with a deductive proof tree.

3.4 Re-Ranking Proofs

Each proof found is re-ranked using the average deductive agreement score for every step in the proof using the validator. The score is calculated on a single step $T_i = (\mathbf{x}_1, \mathbf{x}_2 \rightarrow \mathbf{c})$ by recreating $\mathbf{c}$ using the deductive step model $\mathbf{c}' = S_d(\mathbf{x}_1, \mathbf{x}_2) \rightarrow \mathbf{c}'$. We then test $\mathbf{c}'$ for entailment of the original step's conclusion $s = entail(\mathbf{c}', \mathbf{c})$ and taking the entailment probability as a score. Averaging these probabilities across all steps, $score = \frac{1}{n}\sum_{i=0}^{n} s_i$ where n are the total number of steps in the proof, favors proofs with both deductive and abductive steps that verify deductively and minimizes the expected fraction of errors in the proof.

4 Everyday Norms: Why Not?

To evaluate our method, we need data consisting of entailment trees T as shown in Figure 4. EntailmentBank (Dalvi et al., 2021) is the only existing dataset suitable for this evaluation; however, it is limited to the elementary science domain and we found that step models can often elide minor steps such as synonym replacements, making many instances easy to solve.

We collect a new English-language dataset called *Everyday Norms: Why Not?* (ENWN) describing why an action is or isn't appropriate given a set of circumstances and a set of assumed norms.[3]

[3]We note that unlike Delphi (Jiang et al., 2021), all of the assumptions here are made explicit in the inputs (*taking your neighbor's property without asking permission is stealing*, *stealing is wrong*) rather than relying on the system's priors. Our emphasis is on benchmarking the ability of systems to produce conclusions given stated premises, not trying to automate moral judgments per se (Talat et al., 2021).

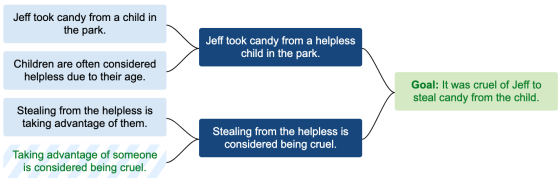

Figure 4: An entailment tree example from the ENWN dataset. Light blue blocks are premises (given as input to the step models), with the last being the gold missing premise, and dark blue blocks are gold intermediate steps (used during training but hidden during inference). The green bock is the goal statement.

ENWN consists of 100 entailment trees annotated by the first two authors of this paper. Each example considers a unique situation, providing an ethical judgement and its justification in the form of an entailment tree. Premise statements include both information about a situation as well as ethical norms. Intermediate steps are written to be similar in form to those in the EntailmentBank dataset, with the exception that all steps have two input statements.

Examples are given in Figure 4 and Appendix B. ENWN trees are slightly larger than EntailmentBank trees, with an average of 4.71 steps in comparison to EntailmentBank's 4.26. Anecdotally, we note that most steps in ENWN cannot be easily elided as they do not involve premises expressing trivial identities, such as "*Green is a color*," which occur often in EntailmentBank.

5 Experimental Setup

We evaluate our models on the premise recovery task according to the two criteria stated previously, recall of missing premises and validity. Recall of missing premises, which we refer to as coverage, is defined using our recovery scoring $E(\mathbf{x}_m, \mathbf{x}') \geq 0.7$. An example only has to produce one tree containing the missing premise to be counted towards the coverage metric. We use human evaluation to evaluate validity.

We evaluate on the English-language EntailmentBank (Dalvi et al., 2021) test set and our new *Everyday Norms: Why Not?* (ENWN) dataset.

To control for tree depth, our test examples are produced by slicing each full entailment tree into *treelets* and removing a single premise from each treelet. Slicing trees allows us to create settings of varying difficulty (deeper treelets being more difficult) and since each treelet has at least two premises we can generate many individual examples. For our evaluations in Table 1 we use a random sampling of 100 treelets for both the EntailmentBank and ENWN datasets.

We compare various models, including End-to-End (E2E), Deductive only (DG), Abductive only (AG), Abductive and Deductive (ADG), and finally our full model, Abductive and Deductive with Validation (ADGV). We now proceed to describe these models.

5.1 Baselines

Deductive Generation Only (DG) The first baseline we compare against is the deduction system of Bostrom et al. (2022). We will simply use this system as originally specified, applying it to the incomplete premises to see if the missing premise can be inferred through deduction alone.

Abductive Generation Only (AG) This model only uses abductive generation. Though this can be effective for certain tree structures and small trees, it cannot generate any intermediate steps requiring forward inference as in Figure 1.

End to End (E2E) Finally, we compare against an end-to-end model that generates a premise conditioned on a set of premises and a goal. We use T5 3B fine-tuned on an adapted EntailmentBank dataset with appropriately constructed training examples; more details are in Appendix A. Note that this model **does not** generate a proof and *only* infers the premise, which we will see can lead to reasoning shortcuts.

5.2 Implementation Details

We run our search for $\mathrm{maxSteps}$ timesteps. Each system is given the same number of backward steps to control for the steps that can actually generate the missing premise (2, 4, 8, 16, 25 for depths 1, 2, 3, 4, and all respectively). A forward step budget is added on top of this (2, 4, 8, 16, and 25 for depths 1, 2, 3, 4, and all respectively), which does increase wall-clock time for two-fringe models (ADG and ADGV) in relation to single fringe models (AG and DG). All models are allowed to sample multiple generations; for abductive steps we sample 40 generations and for deductive steps we sample 10 generations.

Runtimes lengthen as the total number of steps increase and the total number of generations sampled increase. For the largest model (ADGV) with 50 total steps, 40 abductive generation samples per step and 10 deductive generation samples per step, examples are completed in 1 to 2 minutes on average.

Our sequence-to-sequence models are instances of T5 3B (Raffel et al., 2020). Our entailment models and learned heuristic models use DeBERTa Large (He et al., 2021) with 350M parameters. All models are implemented with the Hugging Face `transformers` library (Wolf et al., 2020). Further details including fine-tuning hyperparameters are included in the appendix.

Premise Recovery Scores All of the $entailed(\mathbf{x}, \mathbf{y})$ calls performed during the search use the same EBEntail-Active model as in Bostrom et al. (2022). We define the *rightward entailment* score in our work as $score = entailed(\mathbf{x}'_m, \mathbf{x}_m)$. This entailment can be read as the generated missing premise entailing the actual missing premise. We empirically found this to agree best with our annotations, as discussed in Section 6.3, and used rightward for our results.

6 Results

As our chief goal is to infer missing premises, we begin with premise recovery (coverage) results, shown for all baselines and our best models across both datasets in Table 1. We then discuss human evaluation of both step validity (Table 4) and coverage (Table 5).

6.1 Coverage Results

Abductive generations are required for recovering premises. DG cannot recover any of the premises at any level of depth,[4] illustrating that these premises truly are unstated assumptions not derivable through forward inference.

Using deductive steps generally improves coverage (and validity). AG is capable of producing the missing premise nearly as often (and sometimes more so) than ADG. However, because the re-ranking algorithm in Section 3.4 favors steps

[4]Because the Forward step model requires at least 2 premise statements to perform a step, the model was not run in the D1 setting because those trees only have 1 premise.

System	Entailment Bank D1	D2	D3	D4	Full	Everyday Norms: Why Not? D1	D2	D3	D4	Full
DG	N/A	0%	0%	0%	0%	N/A	0%	0%	0%	0%
AG	**76%**	43%	27%	22%	37%	**57%**	**16%**	**13%**	13%	**12%**
ADG	**76%**	**46%**	**28%**	**25%**	**39%**	**58%**	**16%**	**11%**	**14%**	13%
ADGV	53%	20%	10%	8%	14%	30%	3%	4%	2%	2%
E2E	73%	56%	53%	46%	56%	41%	28%	19%	21%	21%

Table 1: The percentage of premises recovered across both datasets stratified by the depth of trees. Each Dk setting is restricted to trees of only that depth, with full containing full trees that represent all depths (but not a union of all other settings). The E2E baseline is separated out as it does not produce proofs along with its generations.

System	Entailment Bank Count	Len	Score	P Recall	Everyday Norms: Why Not? Count	Len	Score	P Recall
AG	10.30	3.99	43%	59%	6.67	3,.51	24%	32%
ADG	9.97	7.74	50%	66%	3.46	6.38	29%	48%
ADGV	7.43	4.11	80%	82%	2.50	8.12	83%	79%

Table 2: We compare ADGV with two baselines, ADG and AG, on 4 metrics across both the Entailment Bank and ENWN datasets. *Count* is the number of proofs solved on average per tree. *Len* is the length of the proofs on average (how many deductive and abductive steps). *Score* is the average of the entailment probabilities from the deductive agreement score, see section 3.4. Finally, *P Recall* is premise recall: the percentage of the original premises used in the proof. On average the ADGV algorithm produces fewer proofs, but higher scoring proofs that use more of the original premises in its proofs than the baseline methods.

Model	Recovered
E2E	56%
E2E w/o Goal	32%

Table 3: We compare the End-to-End (E2E) model with a variant of the E2E model where no goal is given. Logically, without the goal, it should be impossible to derive the correct missing premise as the space of all generations is too large. Despite this large space, the E2E w/o Goal model is capable recovering the premise 32% of the time, illustrating the existence of shortcuts the model can exploit.

with high deductive agreement, ADG produces slightly higher quality proofs in general, shown in Table 2's *Score* column.

Validators vastly improve quality at the cost of recall Although using validators produces far fewer proofs in Table 1, the quality of proof trees is vastly improved in the ADGV setting. We study the statistics of these generated trees in Table 2. Because there are not actually many valid ways to recover a missing premise, lower proof counts typically indicate more reliable proofs. Shorter proofs also tend to be more consistent with those in the gold entailment trees. Score is the deductive agreement score used to rank the proofs, with higher scores indicating better validity. Finally, Premise Recall (P Recall) is the percentage of the original premises used in the proof. High Premise Recall indicates that more of the input was used to derive the missing statement which indirectly leads to better quality and indicates less hallucination.

Appendix E shows examples of successful and unsuccessful proofs from this method. These illustrate the difficulty of our dataset instances, highlighting how we need to not only chain together the correct inferences and produce the correct statement but also do so within the search budget. Exhaustive search over the space of natural language statements leads to an exponentially large fringe; however, overly heavy filtering may remove a precisely-worded intermediate conclusion needed to recover the missing premise exactly. Finding a balance is a key challenge with stronger methods.

While E2E can recover many premises, it does not construct proofs and uses shortcuts In nearly every depth setting, the E2E model recovers a higher number of premises than our methods. However, the mechanisms that produce these generations can be unsound. Often, when abduction is performed, the level of specificity to abstract or retain is underspecified (as mentioned in Section 3). The E2E model is able to learn these levels of specificity and perform a "premise algebra" from priors in the training data that the step generation baselines cannot exploit (see

Model	Top Proof Valid	VND	Invalid	All Valid	VND	Invalid
ADG	52.8%	11.1%	36.1%	40.4%	6.4%	53.2%
ADGV	**87.0%**	**4.3%**	**8.7%**	**72.5%**	**0%**	**27.5%**

Table 4: Results of manually annotating a total of 200 reasoning steps for validity sampled across two models, ADG and ADGV, and two settings, Top Proof and All, (50 samples per pair) showing that ADGV yields significantly higher quality.

examples in Appendix D). That is, the model can identify keywords that are systematically missing from examples and infer that the missing premise must use them.

Table 3 shows an experiment in which the E2E model is given a set of incomplete premises without the goal and is asked to produce the missing premises. We find that this E2E without Goal model is capable of solving 32% of the examples showing that more than half of the examples solved by the E2E model in Table 1 could have been solved using premise algebra shortcuts. In contrast, our model cannot exploit these shortcuts.

ENWN is a challenging dataset for future work Even the "premise hacking" E2E model only achieves around 20% recovery of missing premises on the full setting. Producing a valid tree that recovers the correct premise is out of range of our current models given our computation budget. We expect scaling the sizes of our models and using improved filtering during search to prioritize the right branches may lead to improvements.

6.2 Human Step Validity Evaluation

Beyond coverage, we want to ensure that our models are taking sound abduction steps, which can also help evaluate whether the model is able to make valid inferences even if the missing premise is not recovered.

We collected steps in two settings: steps from the top ranking proof in cases where the missing premise was recovered (*Top Proof*) and steps in the search state explored at any time from any example (*All*). We then labeled these steps for validity. Soundness is defined as whether the abductive inference yields (1) a true new premise (2) that validates in the forward deductive inference.

The label set includes Y (valid), N (invalid), VND ("valid but not deductive": a true premise that doesn't result in a valid forward deduction). Only examples labeled Y are considered a *valid* step. Ties between valid and invalid annotations favors invalid. Agreement across the multiple labels (Cohen's κ) was 0.48.

As shown in Table 4, on average, using validators produces nearly twice as many valid steps while searching for a proof. Because the proofs are re-ranked once found, the gap between ADG and ADGV in the Top Proof setting is not as dramatic, but still shows a major improvement in creating sound proofs. Having valid steps in complete proofs is important for soundness, but having more valid steps anywhere in the search state demonstrates that the ADGV search explores valid branches of reasoning more often than not.

6.3 Human Coverage Evaluation

Our coverage numbers in Table 1 are an automatic estimate. We undertook additional human validation to ensure that these numbers are representative of actual premise recovery rates.

We sampled 100 steps that were identified as having recovered a premise. Three of the authors then annotated each step as truly recovering the missing premise based on either exhibiting mutual entailment ($\mathbf{x}'_m \leftrightarrow \mathbf{x}_m$) or more specific premises ($\mathbf{x}'_m \rightarrow \mathbf{x}_m$), see Figure 3 for an example. Statements that were more general but did not entail the missing premise were relatively rare and were not considered correct (although they can be valid abductive inferences in some cases), along with other unrelated or bad cases. Ties between the annotators favored the negative (the premise was not recovered); however, annotator agreement was reasonably high with Cohen's κ at 0.74.

Table 5 measures the premise recovery agreement (*coverage*) of the ADG and ADGV systems with manual annotators. We note that the majority of premises marked as recovered by the system are valid missing premises, supporting the validity of our results in Table 1. However, we see that the validated results in ADGV tend to align better with human judgments by 14%; this casts the recovery results of Table 1 in a more favorable light for the ADGV system.

6.4 Error Analysis: ADGV

Underspecification Although validation can help avoid abductive underspecification, the validation models can fail to filter invalid steps. For example, $\mathbf{x}_a$ = "*A reptile does not have fur.*" and $\mathbf{g}$ = "*Animals in the taxonomic groups bird and reptile do not have fur.*" combine together produce

Model	Recovered
ADG	68%
ADGV	**82%**

Table 5: Fraction of automatically-identified recovered premises that our human labeling identified as correct from two of our systems.

the abductive generation "*Birds do not have fur.*" Although this inference passes validation, if we attempt to recreate the goal through forward deduction, we would fail as information about taxonomic groups of animals is not specified. The validator thresholds could be changed to filter this, but this is a challenging case anyway as it is not obvious how to phrase an abductive generation to yield the correct result here.

Cascading errors There is no way for ADGV to test for fallacious generations or false premises. For example, if $\mathbf{x}_a$ = "*A plant is a kind of living thing.*" and $\mathbf{c}$ = "*Grass and a cow are both a kind of plant.*", $\mathbf{c}$ is a false statement, but the abductive step model can still produce a valid generation "*Grass and a cow are both living things.*". However, any proof generated that includes this step would be unsound because $\mathbf{c}$ is false.

Premises that subsume their conclusions If a premise statement $\mathbf{x}_a$ includes a conclusion $\mathbf{c}$, there is nothing to infer from the resulting abductive step that would be meaningful. However, the abductive heuristic can still select these steps and generate abductive inferences that bypass validation. For example, if $\mathbf{x_a}$ = "*A substance is highly reflective, able to conduct electricity, and have high melting points.*" and $\mathbf{c}$ = "*The substance has high melting point.*", $\mathbf{x_a}$ entails c on its own (as well as additional information), leading the step model to generate $\mathbf{x_b}$ = "*The substance is highly reflective and able to conduct electricity.*" Although $\mathbf{x_b}$ may be true, it is not strictly an abduction, and as an independent statement will tend to pollute the search on the next step. Preventing premises from combining with conclusions they already entail could reduce search state complexity and increase step validity, but this is left for future work.

7 Related work

Our work stems from well established models in the question answering domain (Rajpurkar et al., 2016; Yang et al., 2018; Kwiatkowski et al., 2019). Specifically, models have often looked at either generating the correct answer or selecting statements from a set to derive an answer in a "multi-hop" manner (Chen et al., 2019; Min et al., 2019; Nishida et al., 2019). Although discriminative models select evidence for their answers, there is little reasoning being exposed making it hard to detect affordances taken by the end-to-end approaches (Hase and Bansal, 2020; Bansal et al., 2021).

Recently, step-by-step models have been used to create entailment trees that expose a model's reasoning down to individual deductive operations (Bostrom et al., 2022; Dalvi et al., 2021; Ribeiro et al., 2022). Some with the ability to perform backward inferences have also been introduced (Hong et al., 2022; Qu et al., 2022). However, these methods focus on entailing a goal rather than recovering missing evidence. Other work has explored validating step model generations (Yang et al., 2022), but to our knowledge none have used abductive and deductive step models to mutually validate each other.

Chain-of-thought prompting techniques have been used to conduct step-by-step reasoning by eliciting intermediate steps from large language models (Wei et al., 2022; Creswell et al., 2022), but these have been applied to other problems and some preliminary experiments indicate that they do not immediately work for our setting. A related method has been proposed which decomposes statements into inferred premises via backward inference (Jung et al., 2022), although this approach does not simultaneously connect forward inferences from provided premises as our proposed method does.

8 Conclusion

In this work, we tackle the generation of missing premise statements in textual reasoning through the use of abduction. We introduce a new system capable of abductive and deductive step generation, which yields inferred missing premises while building a proof showing its reasoning. Furthermore, we propose a novel validation method that reduces hallucination and other common failure modes in end-to-end and stepwise searches. Future work can improve our system by scaling up the models used, plus using additional notions of validation as discussed in the error analysis. We believe our overall framework can be a promising foundation for future reasoning systems.

9 Limitations

End-to-end models are able to produce a single generation per example reducing the time complexity for sufficiently small sets of premises. Step-by-step models like our search procedure in this work are capable of handling sets of any size of premises for the search, but do increase the execution time per example, especially when using validators that require doing generation themselves. Nevertheless, validators do reduce the total time required for running a set of examples due to their ability of pruning the search space and thus removing numerous heuristic and generation calls. With better heuristics and validators it may be possible to reduce the time complexity further, but that is left for future work.

Both the EntailmentBank and ENWN dataset were written in English and capture relatively limited domains of textual reasoning. Different languages might introduce easier lexical patterns for abstraction though and could be a promising path forward. We believe ADGV and its variants should work on non-English languages, but testing this was left to future work.

ENWN draws on everyday ethical scenarios because this was a domain we found fruitful to exhibit the kind of reasoning our system can do. However, we do not follow in the steps of Delphi (Jiang et al., 2021) in making *any* claims about its ability to make systems ethical or say anything about "values" encoded in pre-trained models. We do not support its use as part of any user-facing system at this time.

Acknowledgments

This work was supported by NSF CAREER Award IIS-2145280, the NSF Institute for Foundations of Machine Learning, ARL award W911NF-21-1-0009, a grant from Open Philanthropy, and gifts from Salesforce and Adobe. This material is also based on research that is in part supported by the Air Force Research Laboratory (AFRL), DARPA, for the KAIROS program under agreement number FA8750-19-2-1003. The views and conclusions contained herein are those of the authors and do not represent the views of DARPA or the U.S. Government. Thanks to the anonymous reviewers for their helpful feedback.

References

Forough Arabshahi, Jennifer Lee, Mikayla Gawarecki, Kathryn Mazaitis, Amos Azaria, and Tom Mitchell. 2021. Conversational neuro-symbolic commonsense reasoning. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(6):4902–4911.

Gagan Bansal, Tongshuang Wu, Joyce Zhou, Raymond Fok, Besmira Nushi, Ece Kamar, Marco Tulio Ribeiro, and Daniel Weld. 2021. Does the Whole Exceed Its Parts? The Effect of AI Explanations on Complementary Team Performance. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, New York, NY, USA. Association for Computing Machinery.

Chandra Bhagavatula, Ronan Le Bras, Chaitanya Malaviya, Keisuke Sakaguchi, Ari Holtzman, Hannah Rashkin, Doug Downey, Wen tau Yih, and Yejin Choi. 2020. Abductive commonsense reasoning. In *International Conference on Learning Representations*.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. COMET: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779, Florence, Italy. Association for Computational Linguistics.

Kaj Bostrom, Zayne Sprague, Swarat Chaudhuri, and Greg Durrett. 2022. Natural language deduction through search over statement compositions. In *Findings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-Findings)*.

Kaj Bostrom, Xinyu Zhao, Swarat Chaudhuri, and Greg Durrett. 2021. Flexible generation of natural language deductions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6266–6278, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Jifan Chen, Shih-ting Lin, and Greg Durrett. 2019. Multi-hop question answering via reasoning chains. *arXiv*, abs/1910.02610.

Peter Clark, Oyvind Tafjord, and Kyle Richardson. 2021. Transformers as soft reasoners over language. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI'20.

Antonia Creswell, Murray Shanahan, and Irina Higgins. 2022. Selection-inference: Exploiting large language

models for interpretable logical reasoning. *arXiv preprint arXiv:2205.09712*.

Bhavana Dalvi, Peter Jansen, Oyvind Tafjord, Zhengnan Xie, Hannah Smith, Leighanna Pipatanangkura, and Peter Clark. 2021. Explaining answers with entailment trees. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7358–7370, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Bhavana Dalvi, Oyvind Tafjord, and Peter Clark. 2022. Towards teachable reasoning systems. *ArXiv*, abs/2204.13074.

Peter Hase and Mohit Bansal. 2020. Evaluating explainable AI: Which algorithmic explanations help users predict model behavior? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5540–5552, Online. Association for Computational Linguistics.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2021. DeBERTa: Decoding-enhanced BERT with disentangled attention. In *International Conference on Learning Representations*.

Ruixin Hong, Hongming Zhang, Xintong Yu, and Changshui Zhang. 2022. METGEN: A module-based entailment tree generation framework for answer explanation. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 1887–1905, Seattle, United States. Association for Computational Linguistics.

Liwei Jiang, Jena D. Hwang, Chandra Bhagavatula, Ronan Le Bras, Maxwell Forbes, Jon Borchardt, Jenny Liang, Oren Etzioni, Maarten Sap, and Yejin Choi. 2021. Delphi: Towards Machine Ethics and Norms. *arXiv*.

Jaehun Jung, Lianhui Qin, Sean Welleck, Faeze Brahman, Chandra Bhagavatula, Ronan Le Bras, and Yejin Choi. 2022. Maieutic prompting: Logically consistent reasoning with recursive explanations. *arXiv preprint arXiv:2205.11822*.

Uri Katz, Mor Geva, and Jonathan Berant. 2022. Inferring Implicit Relations in Complex Questions with Language Models. In *Findings of the Conference on Empirical Methods in Natural Language Processing (EMNLP-Findings)*.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Jiacheng Liu, Alisa Liu, Ximing Lu, Sean Welleck, Peter West, Ronan Le Bras, Yejin Choi, and Hannaneh Hajishirzi. 2022. Generated knowledge prompting for commonsense reasoning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3154–3169, Dublin, Ireland. Association for Computational Linguistics.

Nicholas Lourie, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2021. UNICORN on RAINBOW: A Universal Commonsense Reasoning Model on a New Multitask Benchmark. In *Proceedings of the Conference of the Association for the Advancement of Artificial Intelligence (AAAI)*.

Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. 2019. Multi-hop reading comprehension through question decomposition and rescoring. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6097–6109, Florence, Italy. Association for Computational Linguistics.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial NLI: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901, Online. Association for Computational Linguistics.

Kosuke Nishida, Kyosuke Nishida, Masaaki Nagata, Atsushi Otsuka, Itsumi Saito, Hisako Asano, and Junji Tomita. 2019. Answering while summarizing: Multi-task learning for multi-hop QA with evidence extraction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2335–2345, Florence, Italy. Association for Computational Linguistics.

Hanhao Qu, Yu Cao, Jun Gao, Liang Ding, and Ruifeng Xu. 2022. Interpretable proof generation via iterative backward reasoning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Online and Seattle, USA. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Nazneen Fatema Rajani, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Explain yourself! leveraging language models for commonsense reasoning. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4932–4942, Florence, Italy. Association for Computational Linguistics.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ questions

for machine comprehension of text. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 2383–2392, Austin, Texas. Association for Computational Linguistics.

Danilo Neves Ribeiro, Shen Wang, Xiaofei Ma, Henghui Zhu, Rui Dong, Xinchi Chen, Zhu Peng, Zhiheng Huang, Andrew Arnold, and Dan Roth. 2022. Entailment tree explanations via iterative retrieval-generation reasoner. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Online and Seattle, USA. Association for Computational Linguistics.

J. A. Robinson. 1965. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41.

Zeerak Talat, Hagen Blix, Josef Valvoda, Maya Indira Ganesh, Ryan Cotterell, and Adina Williams. 2021. A Word on Machine Ethics: A Response to Jiang et al. (2021). *arXiv*.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*.

Nathaniel Weir and Benjamin Van Durme. 2022. Dynamic generation of interpretable inference rules in a neuro-symbolic expert system. *arXiv preprint arXiv:2209.07662*.

Sarah Wiegreffe, Jack Hessel, Swabha Swayamdipta, Mark Riedl, and Yejin Choi. 2022. Reframing Human-AI Collaboration for Generating Free-Text Explanations. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, Online and Seattle, USA. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Kaiyu Yang and Jia Deng. 2021. Learning symbolic rules for reasoning in quasi-natural language. *arXiv preprint arXiv:2111.12038*.

Kaiyu Yang, Jia Deng, and Danqi Chen. 2022. Generating natural language proofs with verifier-guided search. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. HotpotQA: A dataset for diverse, explainable multi-hop question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2369–2380, Brussels, Belgium. Association for Computational Linguistics.

Step Signature	Step Type
$\mathbf{x}, \mathbf{x} \rightarrow \mathbf{y}_d$	Deductive
$\mathbf{x}, \mathbf{y}_d \rightarrow \mathbf{y}_d$	Deductive
$\mathbf{g}, \mathbf{x} \rightarrow \mathbf{y}_a$	Abductive
$\mathbf{g}, \mathbf{y}_d \rightarrow \mathbf{y}_a$	Abductive
$\mathbf{y}_a, \mathbf{x} \rightarrow \mathbf{y}_a$	Abductive
$\mathbf{y}_a, \mathbf{y}_d \rightarrow \mathbf{y}_a$	Abductive

Table 6: A list of possible input statement types each step model can take. $\mathbf{x}$ refers to a premise, $\mathbf{y}_d$ refers to an intermediate deductive conclusion, $\mathbf{g}$ refers to the goal, and $\mathbf{y}_a$ refers to an abductive hypothesis. Note that the deductive model can accept inputs in any order but the abductive model cannot, as the abduction operation is not commutative. Also note that deductive outputs can be used as inputs to abductive steps, but not the other way around; allowing deductive steps to accept abductive generations could result in vacuous proofs.

A Implementation Details

All experiments were conducted using Hugging Face `transformers` version 4.20.0.

For all experiments in this paper a set of 3 Quadro 8000 GPUs with 48GB of RAM were used.

Model weights from Bostrom et al. (2022) were used for the Deductive step model, Learned (Goal)+PPM heuristic model and the entailment model.

Default hyperparameters from HuggingFace are used if not otherwise specified for all Step models and the End-to-End model. No hyperparameters sweeps were conducted on these:

Hyperparameter	Value
Base model	T5 3B
Total batch size	8
Initial LR	5e-5
Epoch count	3 (early stopping on val. loss)

Table 7: Abductive Step Model `transformers` default if unspecified)

Hyperparameter	Value
Base model	DeBERTa-v3 Large
Total batch size	32
Initial LR	2e-5
Epoch count	2 (early stopping on val. loss)

Table 8: Abductive learned heuristic model fine-tuning

B Everyday Norms: Why Not? Examples

See Figure 5.

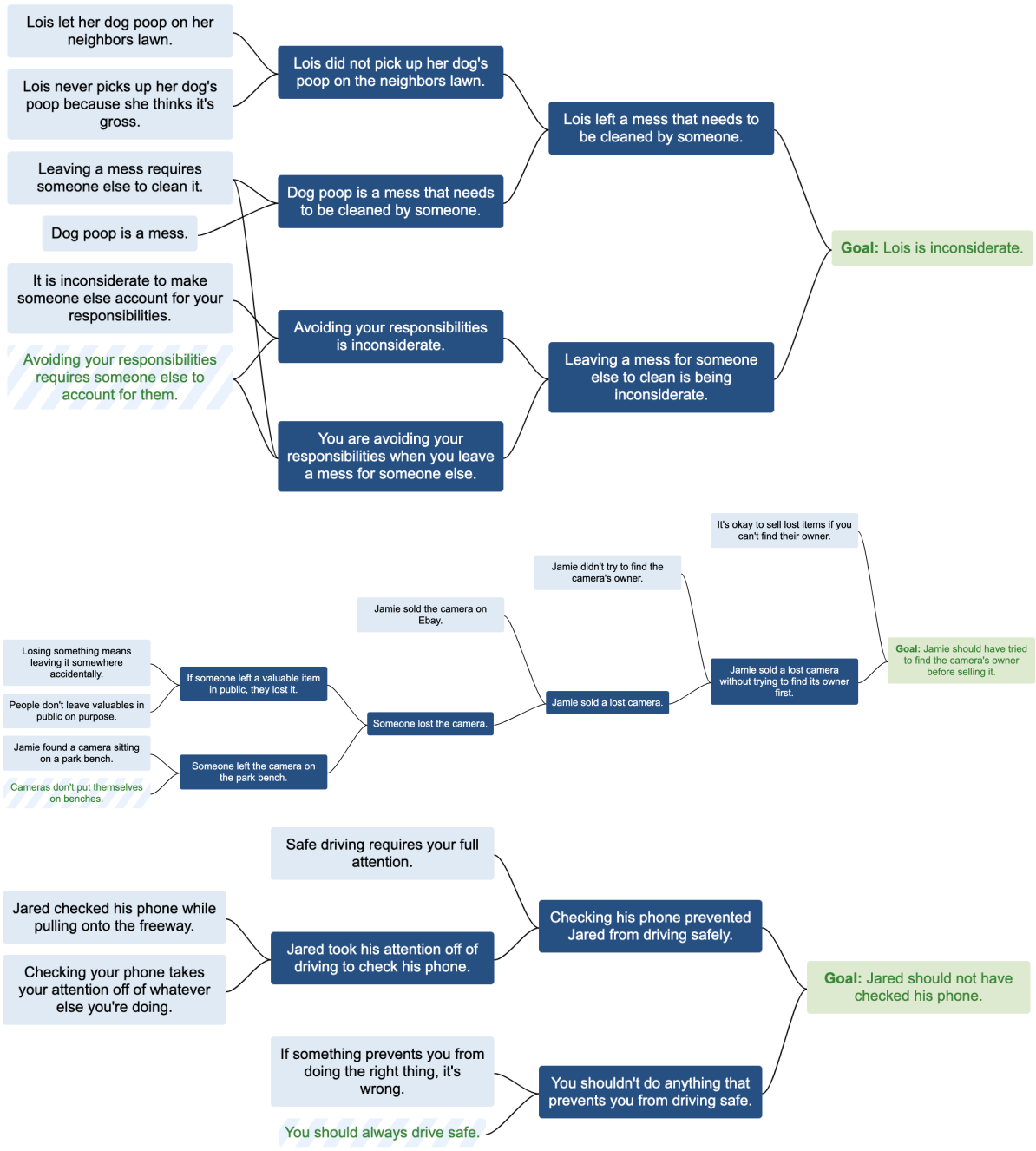

Figure 5: Three example entailment trees from the *Everyday Norms: Why Not?* dataset. Light blue boxes with white text are given premises, dark blue boxes are intermediate deductive steps, green boxes are the goal statements of the examples and striped blue boxes with green text are gold missing premises. The arity for any intemerdiate step in *Everyday Norms: Why Not?* is always two.

C Premise Recovery Generation Examples

See Table 9.

D E2E no goal premise recovery

In Table 10 we show three examples of the ablated model E2E w/o goal correctly generating the missing premise despite being given insufficient information to do so logically. This behavior is problematic as correctly identifying which premise to generate is a vast search space without the goal to direct the model — clearly indicating that the E2E model has learned shortcuts in the data set and is taking advantage of them. Ideally, models would make sound inferences without using spurious patterns from the training dataset to create generations, which is exactly what our step models are designed to do.

Depth	Gold	AG		ADG		ADGV	
1	Some birds eat nectar.	some birds are animals that eat nectar.	0.79	birds can eat nectars.	0.81	birds that eat nectar eat nectar.	0.72
1	As the number of pathways increases, the traffic congestion in that area usually decreases.	as the number of pathways in an area increase, the traffic congestion in that area usually decreases.	0.93	as the number of pathways in an area increases, the traffic congestion in that area usually decreases.	0.93	as the number of pathways increase, the traffic congestion in that area usually decreases.	0.99
1	If fossils of an aquatic animal or plant are found in a place then that place used to be covered by water in the past.	so if fossils of aquatic animals is found in a place then that place used to be covered by water in the past.	0.86	. if fossils of aquatic animals are found in a place then that place used to be covered by water in the past.	0.88	if fossils of aquatic animals are found in a place then that place used to cover by water in the past.	0.87
2	Losing electrons causes the electrical charge of an object to be unbalanced.	as electrons go out of an object, the electrical charge in the object becomes unbalanced.	0.71	when an object loses electrons, the electrical charge of the object becomes unbalanced.	0.81	when objects lose electrons, the electrical charge of that object changes from balanced to unbalanced.	0.76
2	Acid rain causes water pollution.	acid rain is a source of water pollution.	0.72	acid rain lowers water quality.	0.73	acid rain is a pollutant.	0.70
2	Cold fronts cause thunderstorms as they pass by.	a cold front causes storms as it passes by.	0.80	cold fronts cause precipitation as they pass by.	0.83	cold fronts cause storms as they pass by.	0.89
3	Water absorbs solar energy in the water cycle.	water absorbs solar energy.	0.74	water absorbs solar energy.	0.74	water absorbs solar energy.	0.74
3	A fox is a kind of animal.	a fox is a kind of animal species.	0.80	fox is a kind of animal.	0.95	fox is a kind of animal.	0.95
3	Plants perform photosynthesis.	plants perform photosynthesis.	0.99	plants perform photosynthesis.	0.99	plants perform photosynthesis.	0.99
4	Cell division produces cells.	cell division produces cells.	0.99	cell division generates cells.	0.84	cell division produces cells.	0.99
4	Large birds are a kind of organism.	a large bird is an organism.	0.73	large birds are a kind of organism.	0.99	large birds are a kind of organism.	0.99

Table 9: A random sample of 11 abductive steps on the varying depth experiments from Table 1. The depth column corresponds with the depth on Table 1. The gold column shows the original missing premise, then each following column represents one of the models showing it's best generation for that missing premise along with the $E(\mathbf{x}', \mathbf{x})$ score.

Hidden Goal	Missing Premise	E2E Output	Input
The difference between the earth and the moon is that the moon revolves around a planet.	The moon orbits the earth.	The moon orbits the earth.	The sun is a kind of star. Revolving around means orbiting. The earth revolves around the sun. Earth is a kind of planet.
Earth is a celestial object located in the milky way galaxy.	Earth is a kind of planet.	Earth is a kind of planet.	A planet is a kind of celestial object / celestial body. Earth is located in the milky way galaxy.
Dogs will inherit the color of fur from their parents.	Inheriting is when an inherited characteristic is copied / is passed from parent to offspring by genetics / dna.	Inheriting is when an inherited characteristic is passed from parent to offspring by genetics / dna.	A dog is a kind of animal. Fur is often part of an animal. The color of / coloration of fur is an inherited characteristic.

Table 10: Three examples of an End-to-End (E2E) model given only a subset of premises (no goal) generating a missing premise. In two out of the three example above the E2E model is capable of computing the missing premise word for word.

E Example Proofs and Failed Searches

We include multiple example proofs generated by our models from Table 1 at depth = all. Each figure visualizes proofs from a specific model (ADGV on Figure 6 and ADG on Figure 7) and shows two examples from the ENWN dataset on the top and one example from the EntailmentBank dataset on the bottom. Furthermore, we show 8 examples of where the ADGV model failed to produce a proof with a caption explaining where the errors occurred. The first 7 failed search examples are from ENWN and the last is from EntailmentBank.

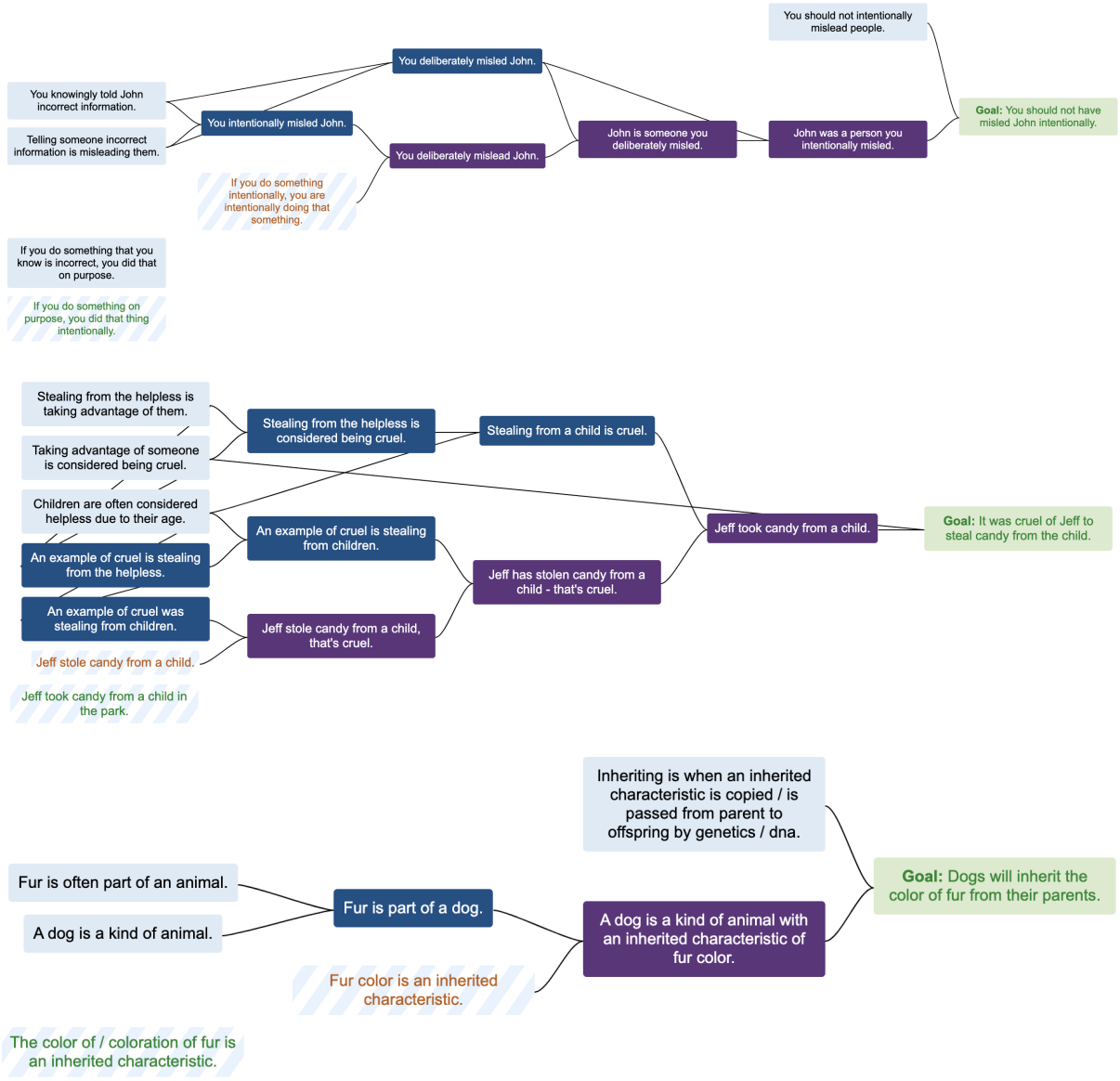

Figure 6: Example **successful proofs** using ADG from the Depth = all experiment. Boxes with blue stripes and orange text $\mathbf{x}'$ are generated premises from the abductive model where blue stripes with green text are the gold missing premise $\mathbf{x}_m$. Light blue boxes are premises, dark blue are intermediate, purple are abductions, and green is the goal of the entailment tree. Note that the gold missing premise is never incorporated in the proof because we are trying to regenerate it through our step models.

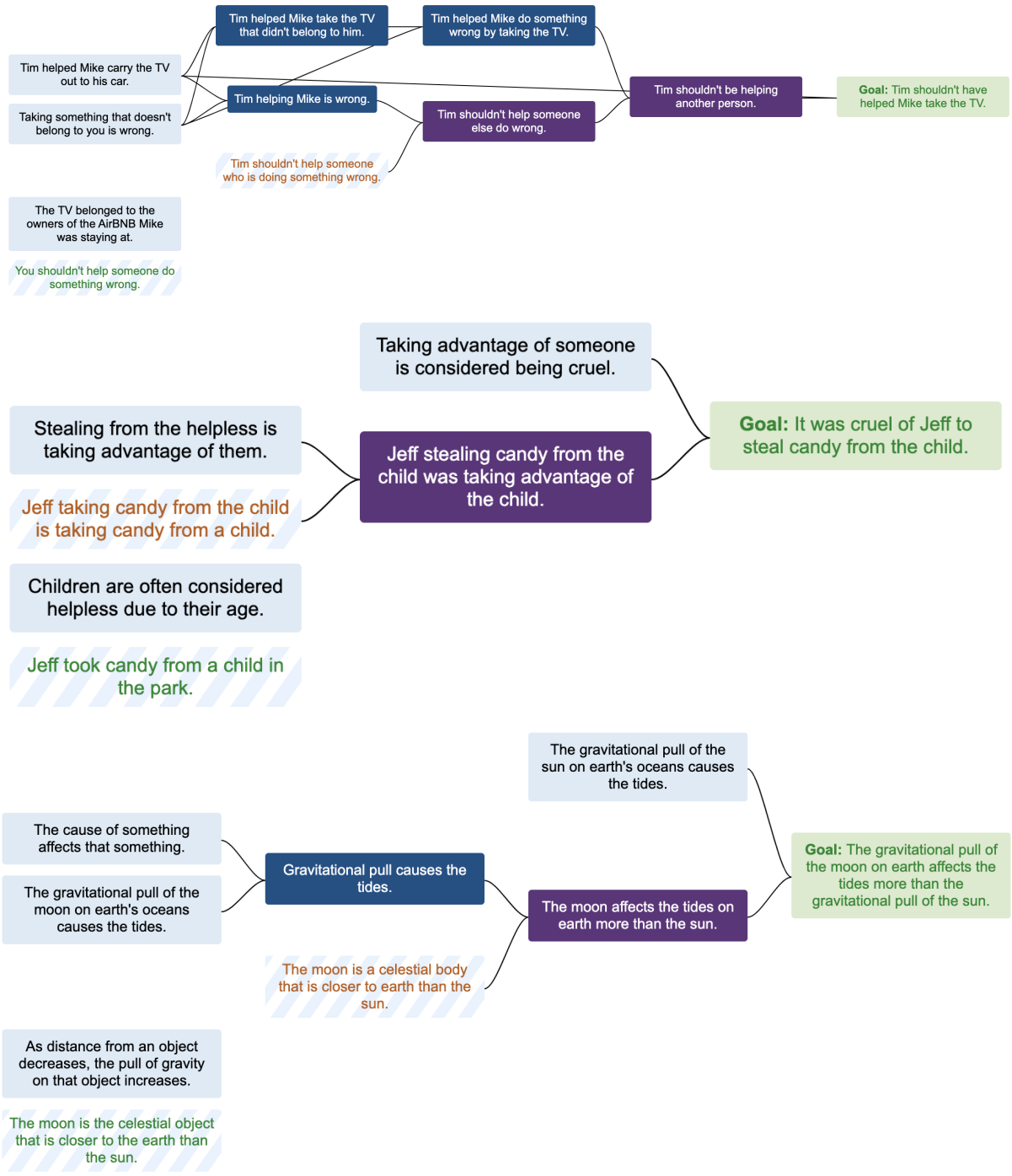

Figure 7: Example **successful proofs** using ADG from the Depth = all experiment. Boxes with blue stripes and orange text $\mathbf{x}'$ are generated premises from the abductive model where blue stripes with green text are the gold missing premise $\mathbf{x}_m$. Light blue boxes are premises, dark blue are intermediate, purple are abductions, and green is the goal of the entailment tree. Note that the gold missing premise is never incorporated in the proof because we are trying to regenerate it through our step models.

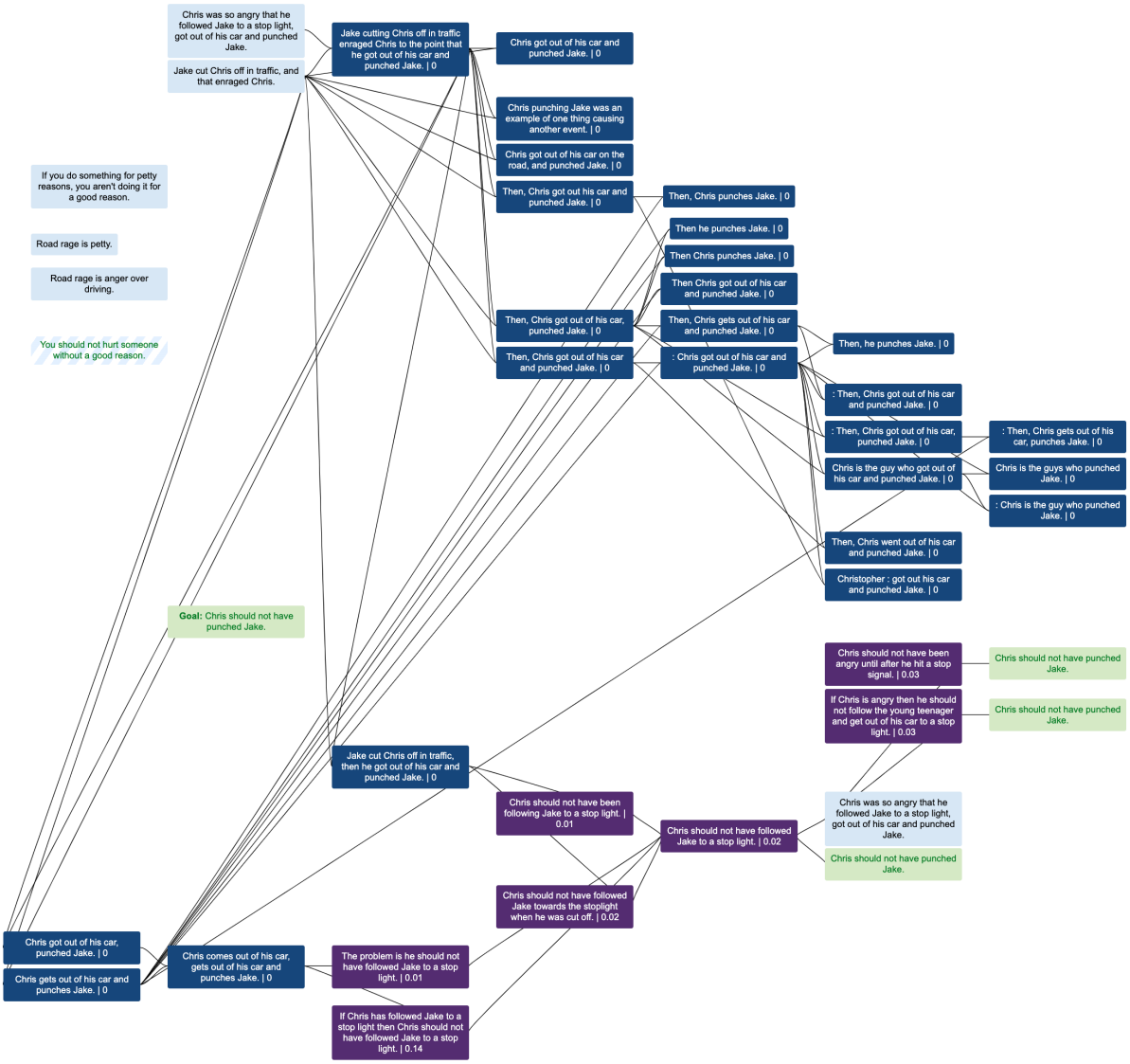

Figure 8: Example of a **failed search** using ADGV on the depth = all experiment for an example of the ENWN dataset. Here the ADGV model fails to make use of all the premises given and continuously combines generations from a subset of the premises and their generations keeping the proof at a specific level of depth that's incapable of recovering the missing premise.

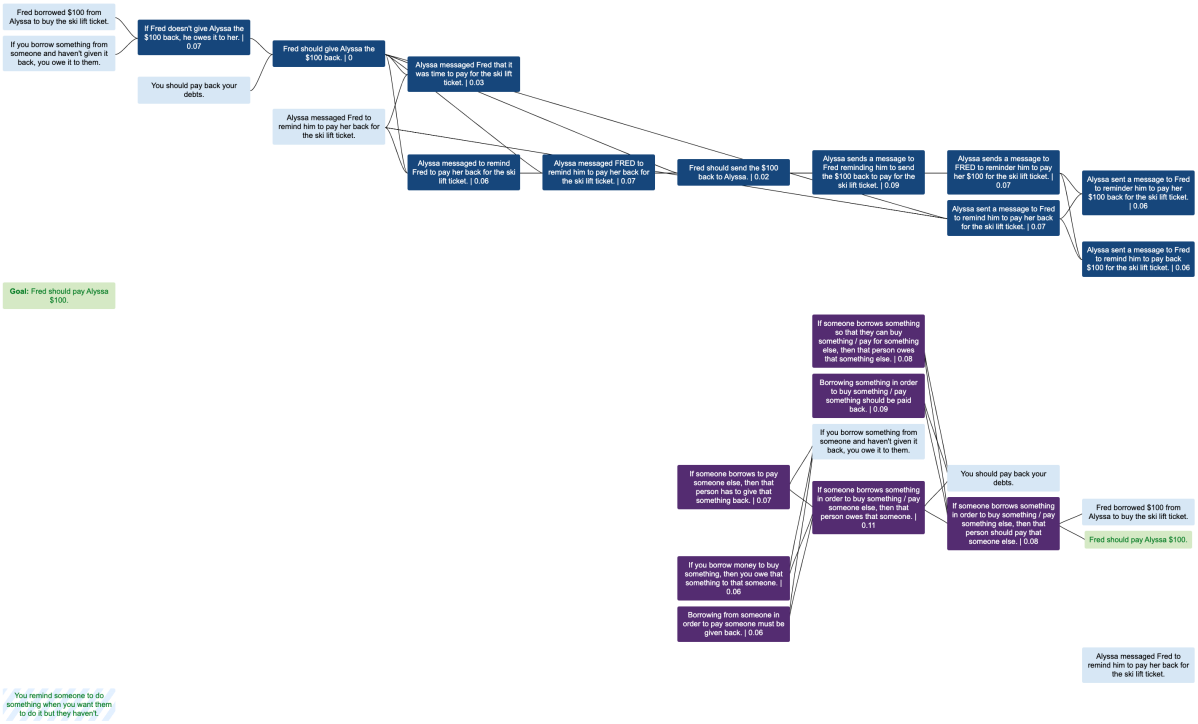

Figure 9: Example of a **failed search** using ADGV on the depth = all experiment for an example of the ENWN dataset. Here the two fringes, f_d and f_a, are kept separate despite there being information beneficial for generating the missing premise $\mathbf{x}_m$ in the forward subtree. We rank all abductive step combinations through the heuristic function H_a, which can fail to recognize useful combinations of forward deductions and open hypotheses. Ideally one of the deductions about Alyssa messaging Fred combined with an abductive generation about borrowing may have pushed the search further towards the missing premise, *You remind someone to do something when you want them to do it but they haven't.*

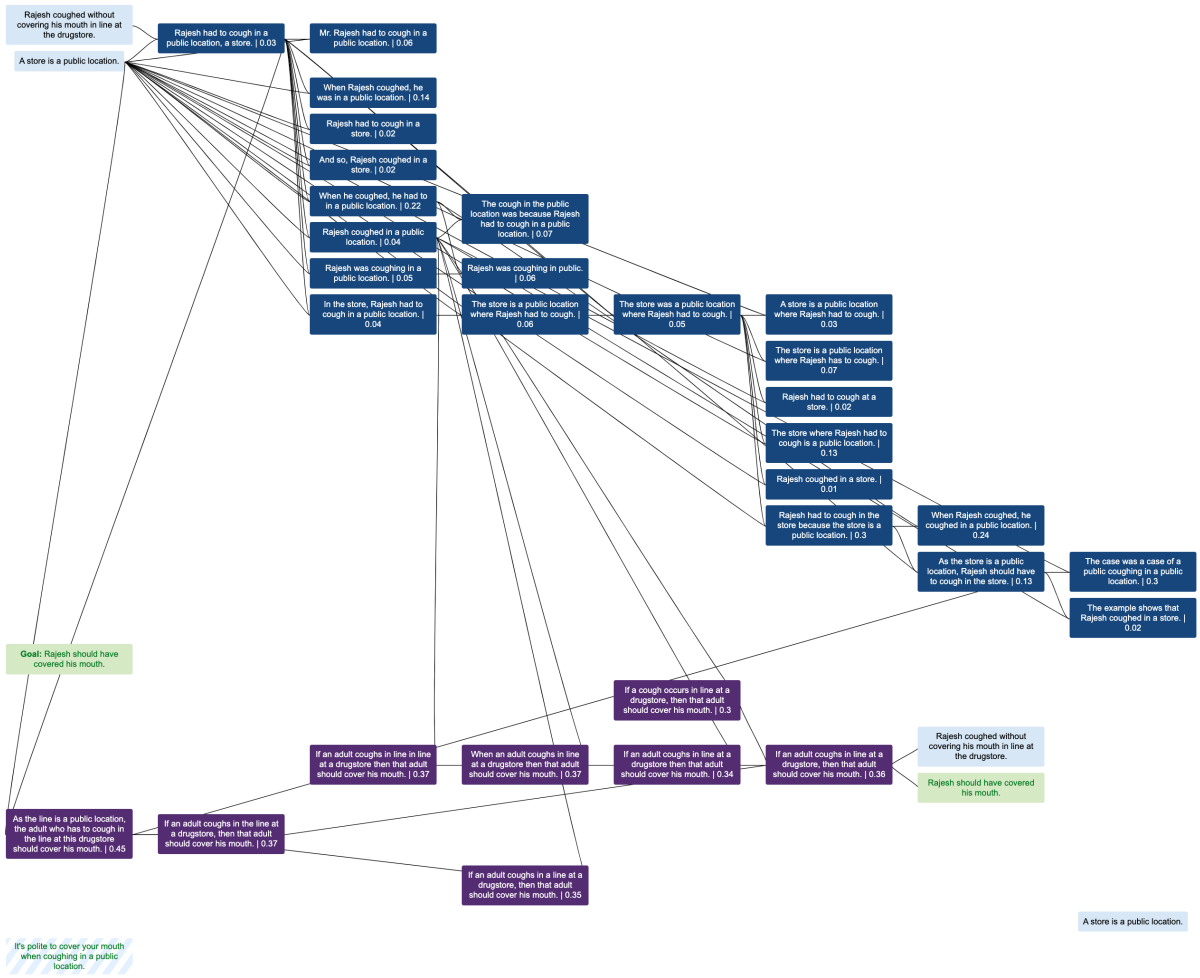

Figure 10: Example of a **failed search** using ADGV on the depth = all experiment for an example of the ENWN dataset. This is an example of a difficult entailment tree to solve for step models. The goal is to generate *It's polite to cover your mouth when coughing in a public location.*, but because there is no premise that states *It's impolite to do things you shouldn't do in public* (or something similar) the abductive model restrains itself in hallucinating such generations. This leads to a close generation *As the line is in a public location, the adult who has to cough in the line at this drugstore should cover his mouth*. Although this isn't as general as the original missing premise, this generation is fairly close to it. Depending on how lenient the system is allowed to be this could be considered a false negative and is an example of the scoring metric being too restrictive.

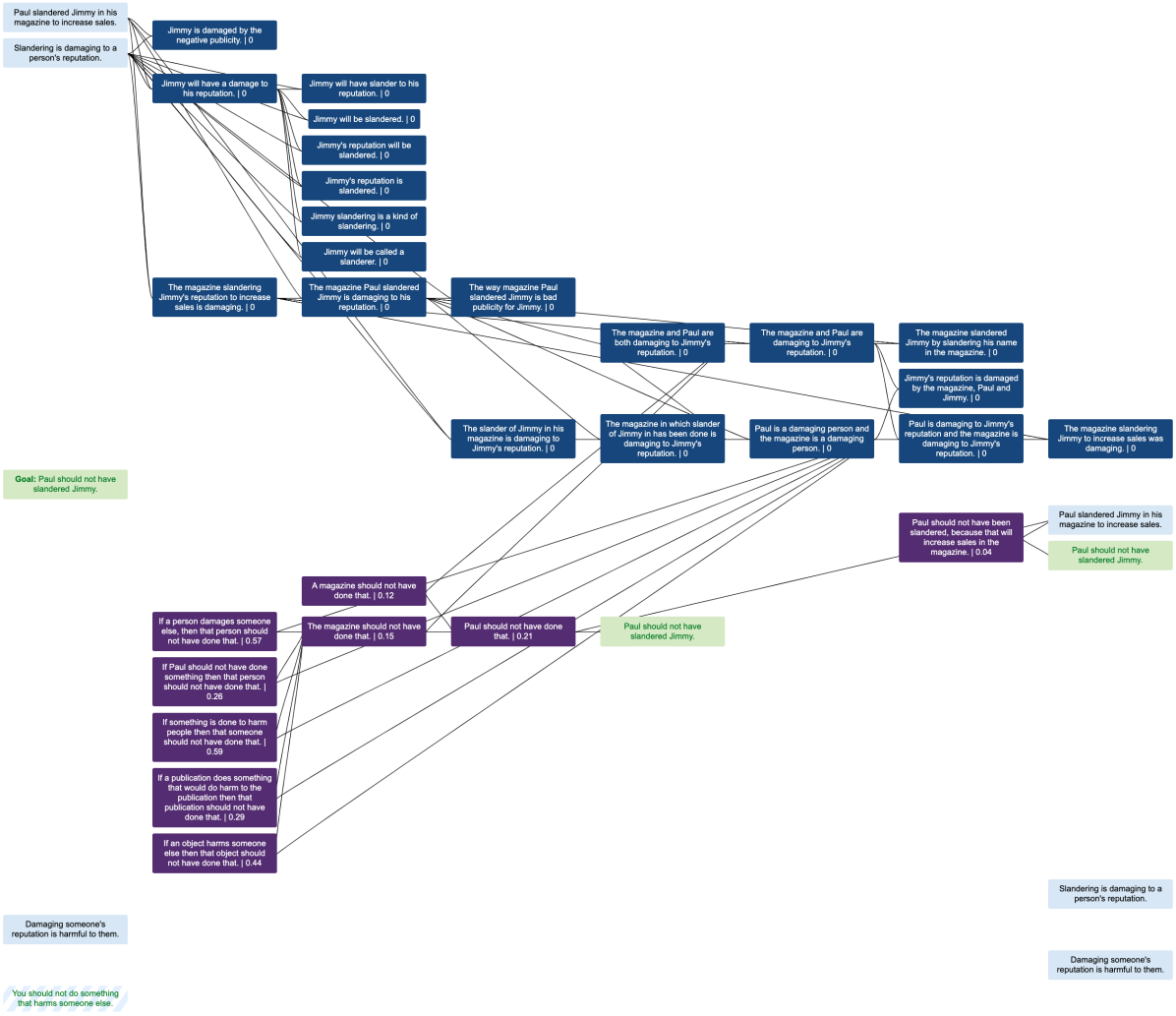

Figure 11: Example of a **failed search** using ADGV on the depth = all experiment for an example of the ENWN dataset. This is an example of a potential false negative. In this example, the abductive model generates *If a person damages someone else, then that person should not have done that.* which is extremely close (and semantically similar) to the gold missing premise *You should not do something that harms someone else.* However, the entailment model scored the generation as 0.57 which is below our threshold for entailment.

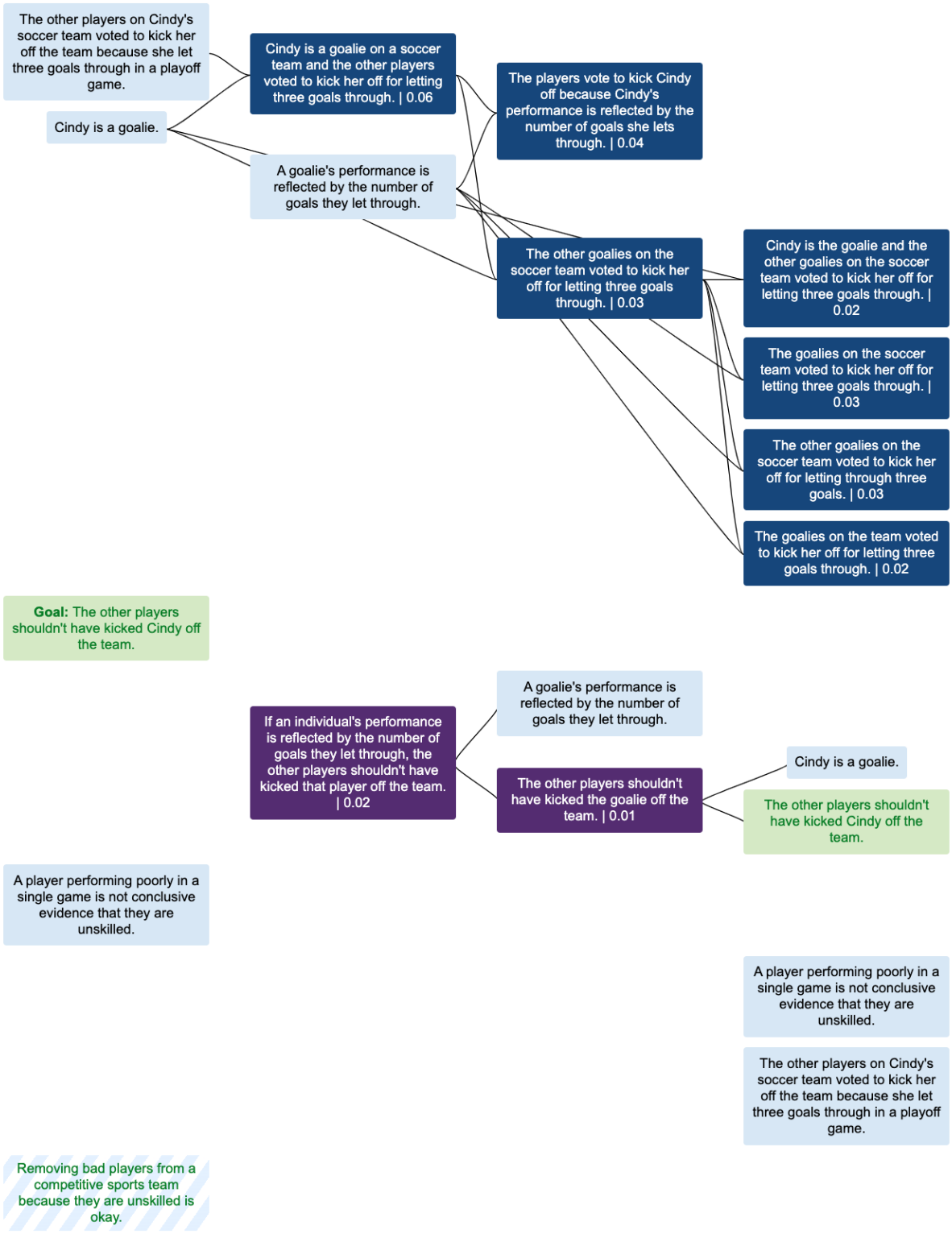

Figure 12: Example of a **failed search** using ADGV on the depth = all experiment for an example of the ENWN dataset. Most of this search's generations were filtered out by one of the validators. The second abductive generation, *If an individual's...*, is an example of the abductive model following a similar pattern seen in EntailmentBank (the dataset used to train the model). The abductive generation, although technically valid, is incorrect given the goal of the proof. Instead, we would have wanted the abductive model to take the premise *The other players on Cindy's soccer team voted to kick her off the team because she let three goals through in a playoff game.* and combine it with the first abductive generation *The other players shouldn't have kicked the goalie off the team.* This is either a failure of the heuristic H_a or a failure of the validators V for removing the generations y_a that came from that step.

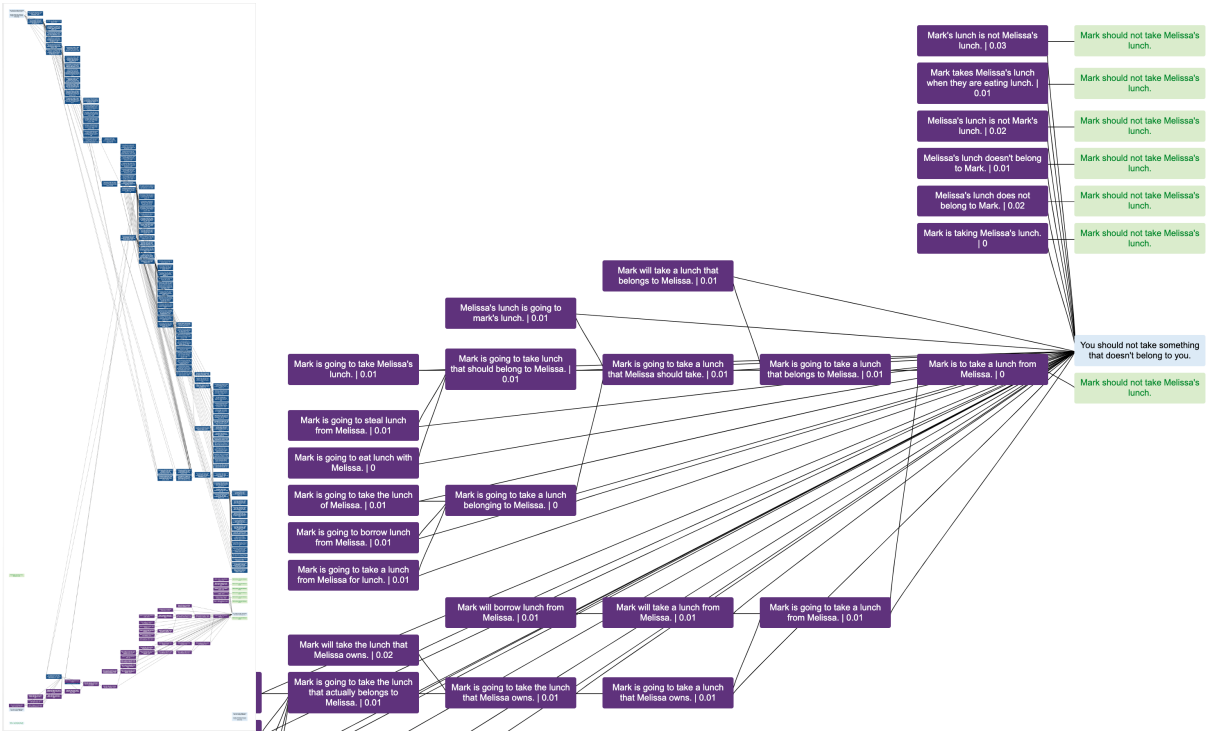

Figure 13: Example of a **failed search** using ADGV on the depth = all experiment for an example of the ENWN dataset. Although in the other failure cases we've shown the tree is somewhat small, most failure cases have an extremely large tree similar to the one in this figure on the right. One of the more common failure modes is the recombination of premises and generations to create deeper proofs that restate the same information slightly differently (left part of this figure). We tried to address this with the Consanguinity filter. Empirically we found that these slight tweaks in generations can lead to improved step recall, however, due to the scoring of both the entailment model and the heuristic functions being more favorable to specific phrasings.

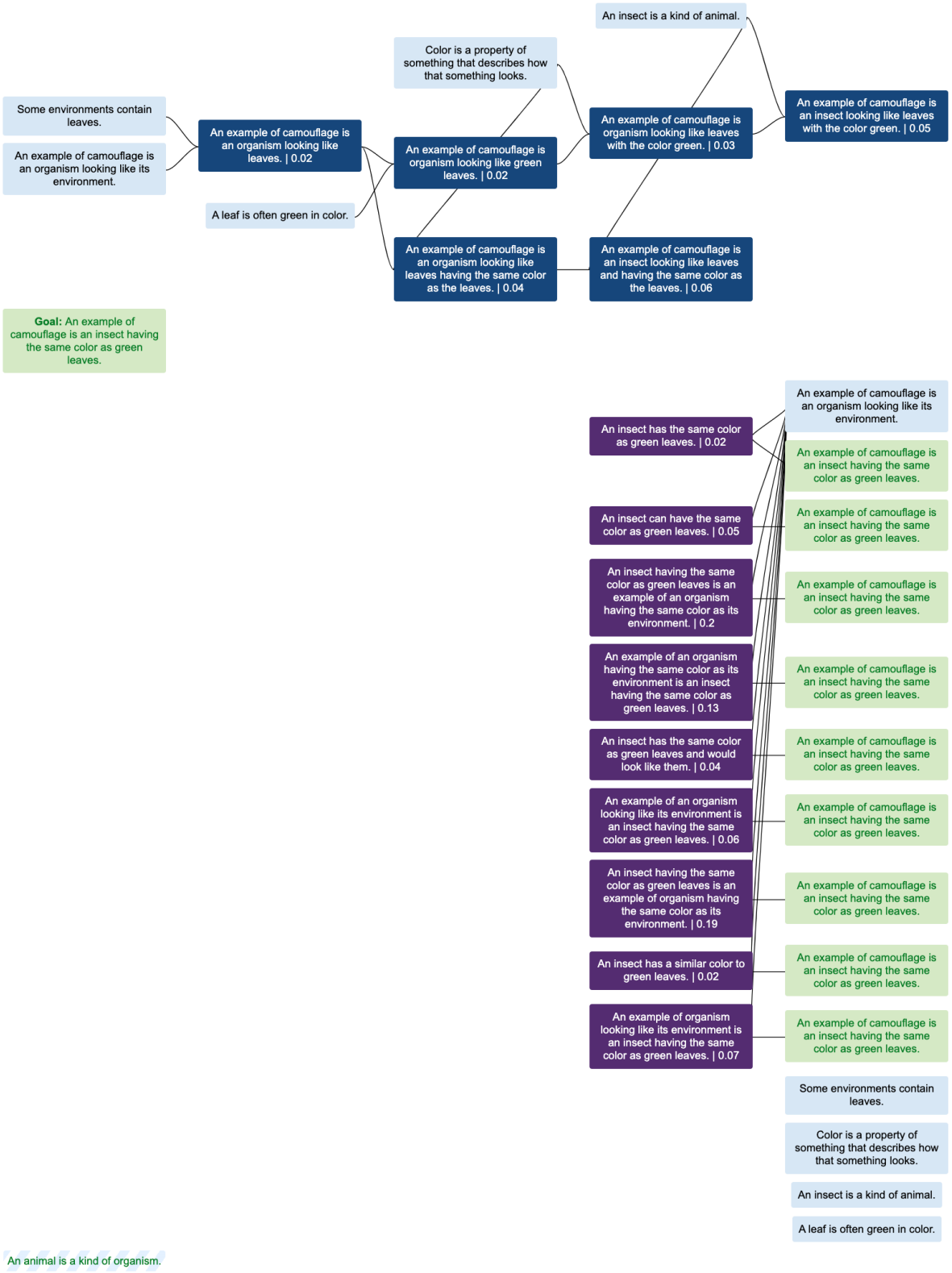

Figure 14: Examples of **failed searches** using ADGV on the depth = all experiment for an example of the EntailmentBank dataset. In this example the deductive fringe f_d makes use of all the given premises but the abductive model does not. The two fringes do not combine either.

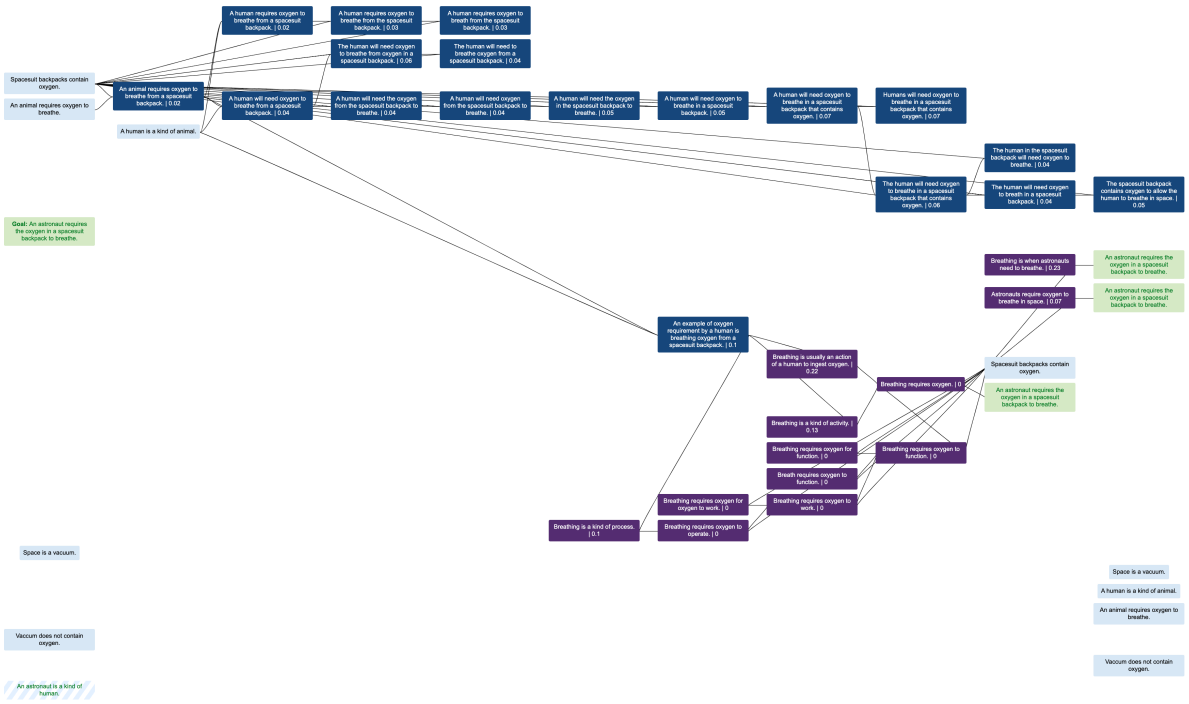

Figure 15: Example of a **failed search** using ADGV on the depth = all experiment for an example of the EntailmentBank dataset. Another example of the abductive fringe, f_a, not using all of its premises where *An animal requires oxygen to breathe.* is paramount to recovering the missing premise.