Natural Language Deduction through Search over Statement Compositions

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

Abstract

In settings from fact-checking to question answering, we frequently want to know whether a collection of evidence (premises) entails a hypothesis. Existing methods primarily focus on the end-to-end discriminative version of this task, but less work has treated the *generative* version in which a model searches over the space of statements entailed by the premises to constructively derive the hypothesis. We propose a system for doing this kind of deductive reasoning in natural language by decomposing the task into separate steps coordinated by a search procedure, producing a tree of intermediate conclusions that faithfully reflects the system's reasoning process. Our experiments on the EntailmentBank dataset (Dalvi et al., 2021) demonstrate that the proposed system can successfully prove true statements while rejecting false ones. Moreover, it produces natural language explanations with a 17% absolute higher step validity than those produced by an end-to-end T5 model.

1 Introduction

When we read a passage from a novel, a Wikipedia entry, or any other piece of text, we gather meaning from it beyond what is written on the page. We make inferences based on the text by combining information across multiple statements and by applying our background knowledge. This ability to synthesize meaning and determine the consequences of a set of statements is a significant part of natural language understanding. Humans are able to give step-by-step explanations of the reasoning that they do as part of these processes. However, approaches that involve end-to-end discriminative fine-tuning of pre-trained language models have no such notion of step-by-step deduction; these models are black boxes and do not offer explanations for their predictions. This limitation prevents users from understanding and
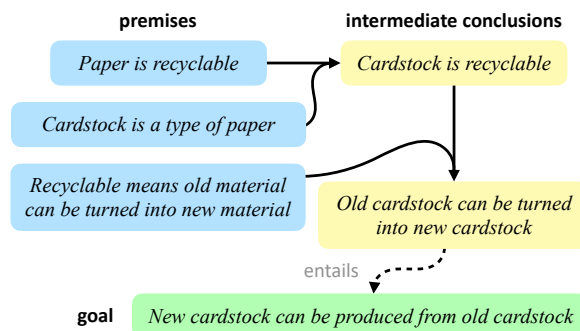

Figure 1: An example of multi-step natural language deduction performed by our system. From input premises, our model generates new statements in a heuristic-guided way to try to prove a given hypothesis.

accommodating models' affordances (Hase and Bansal, 2020; Bansal et al., 2021).

A simple way of representing this kind of reasoning process is through an entailment tree (Dalvi et al., 2021): a derivation indicating how each intermediate conclusion was composed from its premises, exemplified in Figure 1. Generative sequence-to-sequence models can be fine-tuned to carry out this task given example trees (Tafjord et al., 2021; Dalvi et al., 2021). However, we argue in this paper that this conflates the generation of individual reasoning steps with the planning of the overall reasoning process. An end-to-end model is encouraged by its training objective to generate steps that arrive at the goal, but it is not constrained to do so by following a sound structure. As we will show, the outputs of such methods may skip steps or draw unsound conclusions from unrelated premises while claiming a hypothesis is proven. Generating an explanation is not enough; we need explanations to be consistent and faithfully reflect a reasoning process to which the model commits (Jacovi and Goldberg, 2020).

This paper proposes a system that factors the reasoning process into discrete search over intermediate steps. The core of our system is a
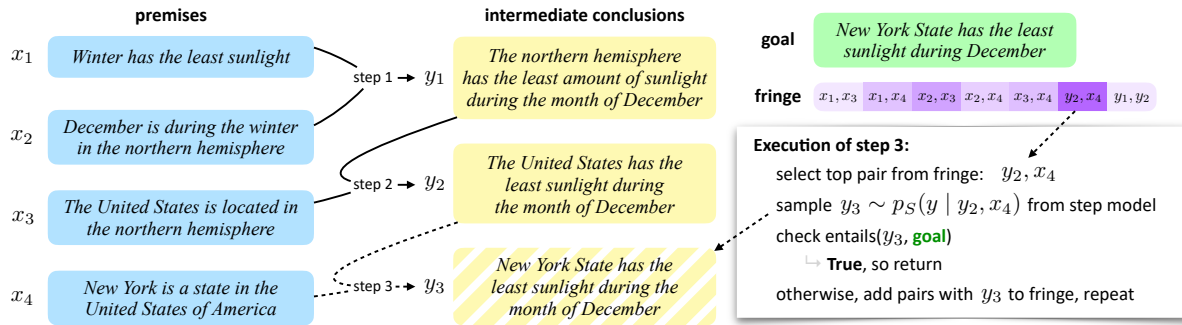
4871

Figure 2: Overview of the problem setting and our approach. The search fringe tracks which statements should be combined next, scored according to the heuristic h (indicated here by purple shading). See Algorithm 1 for a detailed account of the search procedure.

step deduction module that generates the direct consequence of composing a pair of statements. This module is used as a primitive in a search procedure over entailed statements, guided by a learned heuristic function. By decoupling the deduction itself from the search over statement combinations, our system's design ensures that each step's conclusion builds on its inputs, avoiding the pitfalls of end-to-end generation.

We evaluate our method on the EntailmentBank dataset (Dalvi et al., 2021). Thanks to our system's factored design and its ability to capitalize on additional semi-synthetic single step data (Bostrom et al., 2021), we observe that 82% of the reasoning steps produced by our approach are sound, a 17% absolute increase compared to an end-to-end model replicated from prior work.

Our contributions are: (1) A factored, interpretable system for natural language deduction, separating the concerns of generating intermediate conclusions from those of planning entailment tree structures; (2) Exploration of several search heuristics, including a learned goal-oriented heuristic; (3) Comparison to an end-to-end model from prior work (Dalvi et al., 2021) in two settings of varying difficulty.

2 Problem Description and Motivation

The general setting we consider is shown in Figure 2. We assume we are given a collection of evidence sentences $X = \{x_1 \ldots x_n\}$[1] and a goal statement g. We want to construct an **entailment tree** deriving g from X. An entailment tree is a tree of statements with the property that each statement

is directly entailed by its children. Thus, if we can produce a tree with root g and leaves X, it follows that g is transitively entailed by the premises in X.

Crucially, we assume that the intermediate nodes of this tree must be generated and are not present in the input premise set X. This condition differs from prior work on question answering with multi-hop reasoning (Yang et al., 2018; Chen et al., 2019) or models that build a proof structure but do not generate new statements (Saha et al., 2020, 2021). We therefore require a generative step deduction model S to produce intermediate conclusions given their immediate children, a concept explored by Tafjord et al. (2021) and Bostrom et al. (2021).

S yields a distribution $p_S(y \mid x_1 \ldots x_m)$ over step conclusions y conditioned on inputs x_i. In our approach, we assume that each step has exactly $m = 2$ inputs. Some nodes in our evaluation dataset, EntailmentBank, have more than two children, but in preliminary investigation we found it possible to express the reasoning in these steps in a binary branching format. We do not apply this arity constraint to baseline models.

3 Methods

Our proposed system consists of a step model S, a search procedure involving a heuristic h, and an entailment model which judges whether a generated conclusion entails the goal. An overview of the responsibilities and task data required by each module is presented in Table 1.

3.1 Step Deduction Model

Our step models are instances of the T5 pre-trained sequence-to-sequence model (Raffel et al., 2020) fine-tuned on a combination of deduction step datasets from prior work (Dalvi et al., 2021;

---

[1]In the settings we consider in this paper, n does not exceed 25. However, this is not a hard limit imposed by our approach, and for very large premise sets, a retrieval system could be used to prune the premise set to a manageable size.

Module	Task data format (Datasets used)
Step model S	Single-step deductions (ParaPattern, EntailmentBank)
Heuristic h	Entailment trees* (EntailmentBank)
Goal entailment model	Single-sentence NLI (WANLI, EBEntail)

Table 1: Each module in our proposed system operates independently, and most components can leverage existing resources without the need for full tree data. *Our best-performing parametric heuristic uses full tree supervision, but nonparametric heuristics are supported.

Algorithm 1

> **procedure** SCSEARCH($X = \{x_1 \dots x_n\}$, g):
> $fringe \leftarrow \{\langle x_i, x_j \rangle \mid x_i, x_j \in X, i \neq j\}$
> $visited \leftarrow \{\}$
> $i \leftarrow 1$
> $maxSteps \in \mathbb{N}$
>   **while** $|fringe| > 0 \wedge i \leq maxSteps$ **do**
> $inputs \leftarrow \underset{\langle x_1, x_2 \rangle \in fringe}{\mathrm{argmax}} \; h(\{x_1, x_2\}, g)$
> $fringe \leftarrow fringe \setminus \{inputs\}$
>   sample y_i from $p_S(y \mid inputs)$
>   **if** $y_i \notin visited$ **then**
> $visited \leftarrow visited \cup \{y'\}$
>   **yield** $\langle inputs, y_i \rangle$
>   **if** entails(y_i, g) **then return**
> $fringe \leftarrow fringe \cup \{\langle y_i, x_j \rangle | x_j \in X\}$
> $\cup \, \{\langle y_i, y_j \rangle | 1 \leq j < i\}$
> $i \leftarrow i + 1$

Bostrom et al., 2021), which we discuss further in Section 4.4. At inference time, we decode from these models using nucleus sampling (Holtzman et al., 2020) with $p = 0.9$.

3.2 Search

We define a search procedure over sentence compositions that we call SCSEARCH, described in Algorithm 1. SCSEARCH is a best-first search procedure where the search fringe data structure is a max-heap of statement pairs ordered by the heuristic scoring function h. SCSEARCH iteratively combines statement pairs using the step deduction model, adding generated conclusions to the forest of entailed statements.

The heuristic function $h(\{x_1 \dots x_m\}, g) \to \mathbb{R}$ accepts candidate step inputs x_i and a goal hypothesis g and returns a real-valued score indicating the priority of the potential step in the expansion order.

These priority values reflect multiple factors. We want to prioritize **compatible** compositions: combining statements from which we can make a meaningful inference, as opposed to unrelated sentences. We also want to prioritize **useful** steps: among compatible compositions, we should prefer those that are likely to derive statements that help prove the hypothesis.

We consider several potential realizations of the heuristic function h:

Breadth-first Naively, the earliest fringe items are explored first; all compositions of initial premises will be explored before any composition involving an intermediate conclusion.

Overlap This heuristic scores potential steps according to the number of tokens shared between input sentences. This heuristic is focused on compatibility, as overlap indicates expressions that might be unifiable (e.g., *paper* in the first composition step of Figure 1). In the **Overlap+Goal** version, token overlap with the goal hypothesis is also incorporated into the score.

Repetition Past work on step deduction models (Bostrom et al., 2021) has identified that these models tend to "back off" to copying the input when given incompatible premises.

This heuristic aims to exploit this behavior as a measure of premise compatibility. Potential steps are scored according to $-p_S(x_1|x_1...x_n)$, i.e., the negative likelihood of repeating the first input.

Learned This heuristic uses an additional pre-trained model fine-tuned to predict whether input statements are part of a gold explanation of the hypothesis or not. We train this model on sets of step inputs drawn from a collection of valid steps augmented with negative samples produced by replacing one input with a random statement.

The **Learned+Goal** version of this heuristic is also trained with the goal hypothesis in its input, so as to be able to select useful premises and guide search towards the goal. Note that the step model S which generates statements still does not see the goal; the goal only informs which compositions are explored first during the search. See Appendix A for training details of our learned heuristic models.

3.3 Goal Entailment

In order to determine when the search has succeeded, we need a module to judge the entailment relationship between each generated conclusion and the goal. We use a DeBERTa
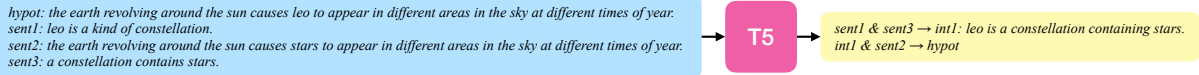
hypot: the earth revolving around the sun causes leo to appear in different areas in the sky at different times of year.
sent1: leo is a kind of constellation.
sent2: the earth revolving around the sun causes stars to appear in different areas in the sky at different times of year.
sent3: a constellation contains stars.

T5

sent1 & sent3 → int1: leo is a constellation containing stars.
int1 & sent2 → hypot

Figure 3: An example of the EntailmentWriter end-to-end system's linearized input and output format. EntailmentWriter takes the goal hypothesis as input, making it possible to hallucinate content based on it, and generation of → *hypot* may occur prematurely, before enough evidence is included to truly derive the goal.

Statement	Goal Hypothesis	Label
A fish is a kind of scaled animal that uses their scales to defend themselves.	Scales are used for protection by fish.	Entailment
Information in an organism's chromosomes causes an inherited characteristic to be passed from parent to offspring by dna.	Children usually resemble parents.	Neutral
A rheumatoid arthritis will happen first before other diseases affect the body's tissues.	The immune system becomes disordered first before rheumatoid arthritis occurs.	Neutral

Table 2: Examples of instances from the EBENTAIL dataset. The goal hypotheses are from EntailmentBank, and the statements are from the SCSEARCH algorithm applied to the original EntailmentBank premises. Although the third example contains a contradiction, the 'neutral' and 'contradiction' labels are merged in EBENTAIL, as both reflect a failure to entail the goal.

model (He et al., 2021) fine-tuned on the WANLI dataset (Liu et al., 2022) to predict the probability that each derived statement entails the goal. In order to mitigate the domain mismatch between WANLI and the scientific facts that make up EntailmentBank, we also fine-tune our goal entailment model on EBENTAIL, a set of 300 examples of generated conclusions sampled from SCSEARCH paired with corresponding goal hypotheses which we manually label for their entailment relationship.

EBENTAIL To produce a set of reference judgments for threshold selection and entailment model evaluation, we sampled 150 instances of generated conclusions from SCSEARCH inference over EntailmentBank examples with their corresponding gold goals. Three annotators labeled the entailment relationship between these generated inferences and the goal. We select a consensus annotation with a majority vote. These examples form an in-domain evaluation set which we call EBENTAIL. To simultaneously train and evaluate on these judgements, we use 3-fold cross-validation where each cross-validation fold is constructed to contain goals not seen in its respective training fold.

We extend EBENTAIL with EBENTAIL-ACTIVE, consisting of 150 additional instances with the lowest confidence (highest prediction entropy) following the initial fine-tuning, which are then manually labeled by at least one annotator.

Thresholding During inference, rather than returning the highest-scoring class, a threshold value α is applied to the predicted probability of the 'entailment' class. This threshold allows for better control over trade-off between precision and recall. For our main experiments, we use an entailment score threshold of $\alpha = 0.81$ selected via cross-validation on EBENTAIL.

4 Experimental Setup

Our experiments assess whether our SCSEARCH system, which factors the deduction process into separate step generation and search modules, can do better than end-to-end baselines on two axes: (1) proving correct (and only correct) goals, and (2) producing more consistent entailment steps in the tree.

4.1 Evaluation: Goal Discrimination

We evaluate our models in two settings, both derived from the validation and test sets of the English-language EntailmentBank dataset. Each setting consists of a 1:1 mixture of examples with valid goals (the original EntailmentBank validation examples) and negative examples with invalid goals, produced by replacing the goal of each positive example with a distinct one drawn from another example. **Each system is evaluated on whether it can prove the correct goals with valid steps and successfully reject incorrect goals.**

To construct hard negatives, candidate replace-

ment goals are ranked according to TF-IDF weighted overlap between tokens in the destination example and tokens in their original example. For each negative example, the replacement goal with the highest overlap score is selected, excluding goals from examples whose premise sets are subsets of the destination example's premises. We manually check negative examples to ensure they cannot be derived from the provided premises.

In **Task 1**, examples contain only gold premises (between 2 and 15), while in **Task 2**, each premise set is expanded to 25 premises through the addition of distractors retrieved from the original premise corpora. We set the $maxSteps$ hyperparameter of the SCSEARCH algorithm to 20. The maximum gold tree step count in EntailmentBank is 17, so our approach can theoretically recover any binarized gold tree given the right heuristic scores.

Note that we focus our evaluation on this goal discrimination task and validating that individual steps of entailment trees are correct, *not* on recovering the exact trees in EntailmentBank. Our deduction model frequently constructs correct entailment trees that do not match the reference, particularly since our approach is not trained end-to-end on this dataset.

4.2 End-to-end Baseline

We compare against an **End-to-end T5** model that we train following the EntailmentWriter paradigm (Dalvi et al., 2021). The EntailmentWriter system involves fine-tuning a sequence-to-sequence language model to generate an entire entailment tree in linearized form, conditioned on the concatenation of a set of premises and a hypothesis. The EntailmentWriter tree linearization format is shown in Figure 3. In the original work, Dalvi et al. (2021) fine-tune T5-11b (Raffel et al., 2020); we replicate their training setup using T5-3b instead for parity with our other experiments.

In order to evaluate whether an end-to-end model *intrinsically* distinguishes between valid and invalid entailment tree structures, we use the average output confidence over trees generated by the model trained without negative examples, computed as the mean token log-likelihood $\frac{1}{L}\sum_{i=1}^{L}\log p_\theta(t_i \mid t_1 \ldots t_{i-1})$. This is motivated by the hypothesis that a model trained as a density estimator for trees composed of sound steps should assign low likelihood to unsound trees. We fit a linear model to predict the goal validity $\in$

$\{0, 1\}$ based on this quantity. We refer to this discriminative setup as **End-to-end (Intrinsic)**.

We also train a variant of the end-to-end baseline, **End-to-end T5 (Classify)**, to explicitly predict whether a given goal is valid by including a flag token T or F at the start of the model's output. We augment the model's training data with an equal number of negative examples by randomly resampling goal hypotheses. We prepend T to the target sequence of positive examples, while the target sequence for negative examples is F. Note that this model can predict T and then output a nonsensical entailment tree, as the trees are post-hoc explanations of the decision.

4.3 Metrics

For both Task 1 and Task 2, consistent models should be able to reach gold goals while also failing to prove invalid goals. To measure the former, we report the number of valid goals reached by each system as **Goal%**. For our search systems, this metric is computed as the proportion of positive examples for which any generated conclusion had a goal entailment score higher than the threshold $\alpha = 0.81$ (see Section 4.4). Goal% scores for the end-to-end model correspond to its self-reported success rate — the proportion of positive examples for which the model emits the T token.

We report the average number of steps expanded before reaching a valid goal as **#Steps**. This metric is averaged over examples for which a system is able to reach the goal.

To measure whether systems are able to distinguish invalid goals from valid goals, we also compute precision-recall curves and report the area under the receiver operating characteristic (**AUROC**) of each system for both tasks. We produce these curves for search models by varying the goal entailment threshold α. For the **End-to-end (Intrinsic)** model, we vary a threshold on the average generated token likelihood, and for the **End-to-end (Classify)** model, we vary a threshold on the value of $p(\mathsf{T})/(p(\mathsf{T}) + p(\mathsf{F}))$, the score assigned to the "valid" flag token by the model out the two possible validity flags.

In addition to evaluating end-to-end performance through the above tasks, we would also like to understand the internal consistency of generated trees. To that end, we conduct a manual study of step validity. We sample 100 steps uniformly across valid-goal Task 2 examples for each of

	Task 1			Task 2		
System	**Goal%**	**AUROC**	**#Steps**	**Goal%**	**AUROC**	**#Steps**
Breadth-first	33.5 ± 0.9	0.88 ± 0.0	3.2 ± 2.6	5.3 ± 0.3	0.68 ± 0.01	10.6 ± 6.1
Overlap	19.4 ± 0.5	0.78 ± 0.01	1.8 ± 1.2	1.8 ± 0.2	0.61 ± 0.02	4.8 ± 5.0
Overlap (Goal)	31.7 ± 1.0	0.83 ± 0.01	2.4 ± 1.6	26.5 ± 1.2	0.74 ± 0.01	3.5 ± 2.8
Repetition	19.7 ± 1.2	0.81 ± 0.0	2.0 ± 1.4	0.9 ± 0.4	0.61 ± 0.01	2.0 ± 1.4
Learned	28.2 ± 1.7	0.83 ± 0.01	2.8 ± 2.3	7.9 ± 2.0	0.71 ± 0.01	6.8 ± 4.0
Learned (Goal)	45.0 ± 1.8	0.90 ± 0.01	2.7 ± 2.1	47.4 ± 1.8	0.85 ± 0.01	4.5 ± 4.1
SCSearch	48.8 ± 1.5	0.91 ± 0.01	3.0 ± 2.1	52.9 ± 1.6	0.85 ± 0.01	3.8 ± 3.4
End-to-end (Classify)	100.0 ± 0.0	0.97 ± 0.00	2.8 ± 1.6	100.0 ± 0.0	0.95 ± 0.00	2.2 ± 1.2
End-to-end (Intrinsic)	-	0.57 ± 0.02	-	-	0.62 ± 0.02	-

Table 3: Results from our main experiments on the EntailmentBank test sets. Mean $\pm$ standard deviation is reported for each metric, taken across 10 trials varying the random seed used for nucleus sampling. **Goal%** indicates the proportion of valid goals reached by a system's generated trees using the $\alpha = 0.81$ threshold. **AUROC** indicates the area under the receiver operating characteristic when attempting to distinguish gold goals from invalid goals. **#Steps** indicates the average number of steps expanded before reaching the goal among trees which reached valid goals; this metric's standard deviation is computed at the example level. See Section 4.3 for more details.
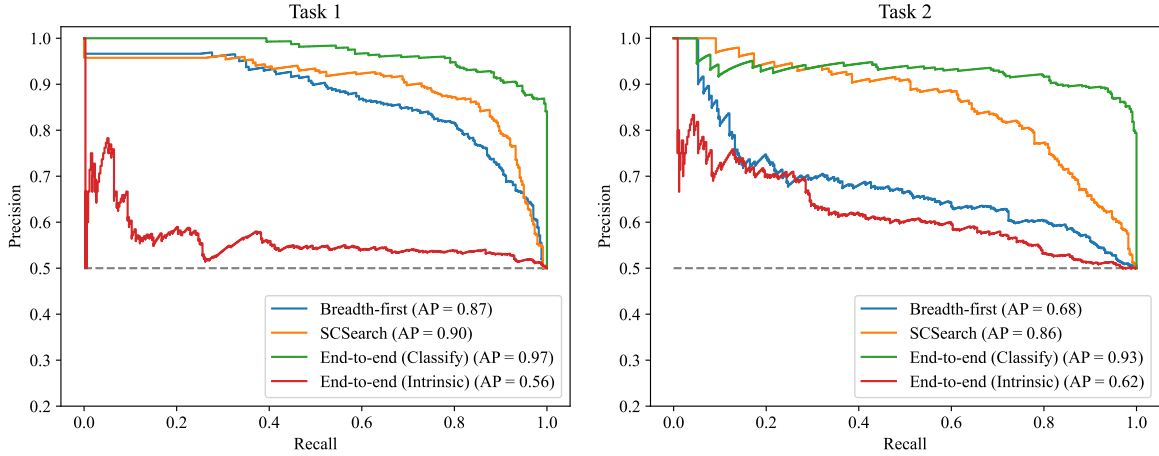

Figure 4: Precision-recall curves for breadth-first search, our full method, and the end-to-end baseline. The dashed line at precision$= 0.5$ corresponds to random chance.

System	Step Validity
Learned (Goal)	74.0%
SCSearch	**82.3%**
End-to-end	65.0%

Table 4: Results from manual annotation of step validity. Steps are sampled from inference on Task 2. Macro-average inter-annotator agreement (Cohen's κ) is 0.72.

three systems: our full system, our system without mid-training, and the end-to-end system. The resulting set of 300 steps is shuffled and triply labeled by three annotators without knowledge of which examples came from which system. For each example, the annotators assess whether a single step's conclusion can be inferred from the input premises with minimal additional world knowledge.

We discuss these results in Section 5.2.

4.4 Implementation and Training

All sequence-to-sequence language models we experiment with are instances of T5-3b (Raffel et al., 2020) with 3 billion parameters. Bridging entailment models and learned heuristic models are derived from DeBERTa Large (He et al., 2021) with 350M parameters. Fine-tuning is performed with the Hugging Face `transformers` library (Wolf et al., 2020). Further details including fine-tuning hyperparameters are included in the appendix.

Step deduction model We train our step deduction model using gold steps sampled from trees in the EntailmentBank (**EB**) training split, totaling 2,762 examples. Our full system, which we refer to as **SCSearch**, is also "mid-trained" on ParaPattern substitution data (Bostrom et al., 2021).

The ParaPattern data is derived semi-synthetically from English Wikipedia, totaling ~120k examples. In the mid-training configuration, an instance of T5 is fine-tuned for one epoch on the ParaPattern data and then for one epoch on the EntailmentBank data, after which optimal validation loss is reached.

Learned heuristic models Data for our learned heuristic models is constructed by taking step inputs from the EntailmentBank training set's gold trees. For each positive step example, a corresponding negative example is produced by replacing one input statement with a sentence drawn at random from all statements in the training set that do not appear in the original step's subtree. Examples used to train the Learned (Goal) heuristic additionally contain the original step's gold goal concatenated to their input. Our full SCSearch system uses the Learned (Goal) heuristic.

5 Results

The results of our experiments on Task 1 and Task 2 are shown in Table 3. Table 4 shows the results of our manual step validity evaluation.

5.1 Complete Deductions

The end-to-end approach is not a sound model of entailment tree structure. End-to-end T5 (Classify) is nominally able to "prove" 100% of gold goals by finishing a proof with `->` `hypot`. However, as shown in Figure 4, using the generation confidence of End-to-end T5 to discriminate between valid goals and invalid goals (the Intrinsic method) is not much better than random chance, since the model has similar confidence when "proving" invalid goals as it does when generating trees for valid goals. This means that the model's output distribution does not penalize the generation of invalid steps.

Our approach is able to prove most goals while exhibiting much better internal consistency than the end-to-end approach. Our full SCSearch system nearly matches the performance of the End-to-end (Classify) system on Task 1 and Task 2, losing chiefly in high-recall settings. Critically, Section 5.2 will show that it achieves a much higher rate of step validity in the process.

Mid-training the step deduction model also increases the proportion of reachable valid goals by 3-5% (compare Learned (Goal) to SCSearch in Table 3). It is worth noting that the chosen threshold for our goal entailment module sacrifices recall in favor of avoiding false positives, as shown in Table 5, meaning that our reported Goal% rate is an underestimate. In Figure 4, we see that our SCSearch method can achieve 80% recall at roughly 80% precision with a slightly lower threshold.

A goal-oriented heuristic is critical. If we compare Breadth-first, Repetition, our two Overlap methods, and our two Learned heuristics, both Table 3 and Figure 4 show that the incorporating goal information into the planning process is essential for Task 2, as only the Overlap (Goal) and Learned (Goal) heuristics are able to reach a reasonable number of valid goals in the presence of distractor premises. We can see in Figure 4 how much breadth-first degrades from Task 1 to Task 2, largely due to timing out.

5.2 Individual Step Validity

The most crucial divergence between our method and the end-to-end method arises in the evaluation of individual steps. The End-to-end (Classify) method can produce correct decisions, and as shown in Figure 3 it can always claim to have produced the goal statement, but is its reasoning sound?

Table 4 shows that our best SCSearch model produces valid inferences 82% of the time, improving by 17% absolute over the end-to-end model as well as improving over the version of our system without mid-training on ParaPattern. This result confirms our hypothesis about the end-to-end method: despite often predicting the right label and predicting an associated entailment tree, these entailment trees are not as likely to be valid due to their post-hoc nature and the conflation of search and generation.

We can use our step validity rate to approximate the expected number of fully-valid trees based on the observed depth distribution. Under the conservative assumption that observed errors are distributed uniformly w.r.t. depth, the expected number of fully valid trees in a dataset $D = \{T_1 \ldots T_{|D|}\}$ for validity rate v can be computed as $\frac{1}{|D|} \sum_{i=1}^{|D|} v^{|T_i|}$. At a step validity rate of 82% we should expect ~58% of trees generated by our system to be error-free. Under the same assumption, according to the end-to-end model's step validity rate we should expect only ~35% of its trees to be fully valid.

In Section 5.4 we examine observed error patterns in invalid steps. Crucially, even when our system produces a tree involving an invalid step, it is easy to audit the tree and determine exactly where the reasoning error occurred, since each step is conditioned only on its immediate premises. In contrast, the end-to-end model attends to all premises and the hypothesis at every step, meaning that when an inconsistent step is generated, it is difficult to diagnose the cause.

5.3 Goal Entailment

Model	α	F1	Precision	Recall
DeBERTa WANLI	0.92	71.8	65.1	**80.0**
+EBENTAIL	0.83	72.4	91.3	60.0
+EBENTAIL-ACTIVE	0.81	**77.9**	**95.8**	65.7

Table 5: Evaluation of goal-bridging entailment models on EBENTAIL. Both of the models fine-tuned on EBENTAIL are evaluated using 3-fold cross validation. α indicates the best entailment score threshold.

The results in Table 3 depend on an accurate assessment of when we have successfully deduced the hypothesis. To that end, we evaluate our goal entailment model against labeled test data. Table 5 shows the results of our evaluation on the EBENTAIL dataset described in Section 3.3. We view precision as more important than recall, as a stringent criteria for determining whether a tree has reached the goal increases confidence in our evaluation results.

Our best F1 score, using EBENTAIL-ACTIVE, is only slightly lower than the lowest F1 score of the annotators when evaluated against the majority vote. Annotator agreement is moderate; macro-averaged inter-annotator F1 is 0.83 and Cohen's κ is 0.54. This indicates that the problem of determining when a statement straightforwardly entails the goal is subjective; the boundary between 'trivial' entailment and a case which needs an additional reasoning step is somewhat fuzzy.

5.4 Error Analysis: Step Model

Although our step model cannot hallucinate based on the hypothesis, it can still fail to produce valid intermediate steps due to other challenges.

One error type we see is **indiscriminate unification**: when given incompatible premises, the step model will sometimes still attempt to combine them, resulting in improper conclusions. For example, given the premises *"Earthquake can*

change the earth's surface. In a short amount of time is similar to rapidly." one conclusion generated by the model is *"Earthquake changes the earth's surface rapidly."* This could be avoided through the use of more selective heuristics, or by explicitly supervising step models with negative examples in order to encourage conservative conclusions in these cases.

We also observe **compounding errors** in conclusions generated from erroneous premises. For example, given the premises *"Offspring will inherit a scar on both knees except not both knees. Offspring will inherit a scar on the knee from parents."* the model generates *"Parents will inherit a scar on both knees."* This kind of relation assignment mistake is uncommon outside of instances involving bad premises. These errors could potentially be mitigated by training heuristics to avoid corrupted premises.

5.5 Error Analysis: Goal Entailment

An additional source of error arises from cases where the goal entailment model is unable to predict the correct entailment relationship between an output from the step model and the goal hypothesis.

One reason this arises is **definitional knowledge**. When given the premise *"Sugar is soluble in water,"* the model does not predict entailment of the goal hypothesis *"Sugar cubes will dissolve in water when they are combined."* English speakers who know the definition of 'soluble' and recognize that sugar cubes are made of sugar could reasonably understand this as entailed. However, the degree of definitional knowledge that should be expected of the entailment model is subjective and often a source of annotator disagreement.

Another cause of errors are **ill-formed statements**. For example, the model predicts that *"Lichens and soil are similar to being produced by breaking down rocks."* entails *"Lichens breaking down rocks can form soil."* However, it is unclear what is "similar" in the generated statement due to poor syntax. Labels for examples like this often vary depending on how the annotator understood the step model's output. Improving the step model to reduce compounding generation errors will mitigate this issue.

Finally, the entailment may sometimes be predicated on **context**. The model predicts that *"A new moon will occur on june 30 when the moon*

orbits the earth." entails *"The next new moon will occur on june 30."* In this case, the model is assuming that 'a new moon' occurring is equivalent to 'the next new moon' occurring. Depending on annotator assumptions, cases like this can also be somewhat subjective.

Future work could expand the training data of our NLI model to account for the subjectivity of NLI judgments (Pavlick and Kwiatkowski, 2019; Chen et al., 2020; Nie et al., 2020), particularly by modifying our data collection procedure (Zhang et al., 2021).

6   Related work

Our work reflects an outgrowth of several lines of work in reading comprehension and textual reasoning. Multi-hop question answering models (Chen et al. 2019; Min et al. 2019; Nishida et al. 2019, inter alia) also build derivations linking multiple statements to support a conclusion. However, these models organize selected premises into a chain or leave them unstructured as opposed to composing them into an explicit tree.

The NLProlog system (Weber et al., 2019) frames multi-hop reading comprehension explicitly as a proof process, performing proof search using soft rule unification over vector representations of predicates and arguments. Similar backward search ideas were used in Arabshahi et al. (2021). PRover (Saha et al., 2020) and ProofWriter (Tafjord et al., 2021) also frame natural language deduction as proof search, although both systems are evaluated in a synthetic domain of limited complexity. Betz and Richardson (2021) also use synthetic data to improve reasoning models through mid-training, although the improvements they observe are limited to premise selection performance.

Hu et al. (2020) and Chen et al. (2021) propose systems which perform single-sentence natural language inference through proof search in the natural logic space. Our work also relates to earlier efforts on natural logic (MacCartney and Manning, 2009; Angeli et al., 2016) but is able to cover far more phenomena by relaxing the strict constraints of this framework. Finally, the Leap of Thought system (Talmor et al., 2020) tackles some related ideas in a discriminative reasoning framework.

The recent chain-of-thought (Wei et al., 2022) and Scratchpads (Nye et al., 2021) methods also generate intermediate text as part of answer prediction. However, like the end-to-end baseline we consider, these techniques are free to generate unsound derivations. Published results with these techniques are strongest for tasks involving mathematical reasoning or programmatic execution, whereas on textual reasoning datasets like StrategyQA (Geva et al., 2021) they only mildly outperform a few-shot baseline.

7   Discussion and Conclusion

In this work, we propose a system that performs natural language reasoning through generative deduction and heuristic-guided search. We demonstrate that our system produces entailment trees that are more internally consistent than those of an end-to-end model, and that its factored design allows it to successfully prove valid goals while being unable to hallucinate trees for invalid goals. We believe that this modular deduction framework can be readily extended to empower future reasoning systems.

8   Limitations

The baseline approach we consider in this work, end-to-end modeling of entailment tree generation, enjoys the convenience of simple inference and quadratic complexity. However, the computational overhead of sequence-to-sequence models places a hard limit on the tree size and premise count that can be handled in the end-to-end setting; moreover, recent results call into question how well end-to-end Transformers can generalize this type of reasoning (Zhang et al., 2022). Our structured approach allows arbitrarily large premise sets and step counts. However, by discretizing the reasoning in the SCSearch procedure, we do face a runtime theoretically exponential in proof size to do exhaustive search. In practice, we limit our search to a finite horizon and find that this suffices to provide a practical wall clock runtime, never exceeding 5 seconds for any single example. Future work on higher tree depths may have to reckon with the theoretical limitations of this procedure, possibly through the use of better heuristics.

Our experiments are conducted exclusively on English datasets. While we hypothesize that our approach would work equally well for another language given a pretrained sequence-to-sequence model for that language with equivalent capacity, such models are not available universally across languages, representing an obstacle for transferring our results to languages beyond English.

Furthermore, the EntailmentBank dataset on which we train and evaluate targets the elementary science domain, raising a question of domain specificity. In future work, we plan to evaluate deduction models on additional datasets with different style, conceptual content, and types of reasoning in order to verify that the factored approach is equally applicable across diverse settings.

Acknowledgments

References

Gabor Angeli, Neha Nayak, and Christopher D. Manning. 2016. Combining natural logic and shallow reasoning for question answering. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 442–452, Berlin, Germany. Association for Computational Linguistics.

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.

Gregor Betz and Kyle Richardson. 2021. DeepA2: A modular framework for deep argument analysis with pretrained neural text2text language models. *arXiv*, abs/2110.01509.

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.

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

Tongfei Chen, Zhengping Jiang, Adam Poliak, Keisuke Sakaguchi, and Benjamin Van Durme. 2020. Uncertain natural language inference. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8772–8779, Online. Association for Computational Linguistics.

Zeming Chen, Qiyue Gao, and Lawrence S. Moss. 2021. NeuralLog: Natural language inference with joint neural and logical reasoning. In *Proceedings of *SEM 2021: The Tenth Joint Conference on Lexical and Computational Semantics*, pages 78–88, Online. Association for Computational Linguistics.

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.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? a question answering benchmark with implicit reasoning strategies. *Transactions of the Association for Computational Linguistics*, 9:346–361.

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*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Hai Hu, Qi Chen, Kyle Richardson, Atreyee Mukherjee, Lawrence S. Moss, and Sandra Kuebler. 2020. MonaLog: a lightweight system for natural language inference based on monotonicity. In *Proceedings of the Society for Computation in Linguistics 2020*,

pages 334–344, New York, New York. Association for Computational Linguistics.

Alon Jacovi and Yoav Goldberg. 2020. Towards faithfully interpretable NLP systems: How should we define and evaluate faithfulness? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4198–4205, Online. Association for Computational Linguistics.

Alisa Liu, Swabha Swayamdipta, Noah A. Smith, and Yejin Choi. 2022. WANLI: Worker and ai collaboration for natural language inference dataset creation. *arXiv*, abs/2201.05955.

Bill MacCartney and Christopher D. Manning. 2009. An extended model of natural logic. In *Proceedings of the Eight International Conference on Computational Semantics*, pages 140–156, Tilburg, The Netherlands. Association for Computational Linguistics.

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, Xiang Zhou, and Mohit Bansal. 2020. What can we learn from collective human opinions on natural language inference data? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9131–9143, 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.

Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, Charles Sutton, and Augustus Odena. 2021. Show Your Work: Scratchpads for Intermediate Computation with Language Models. *arXiv*.

Ellie Pavlick and Tom Kwiatkowski. 2019. Inherent disagreements in human textual inferences. *Transactions of the Association for Computational Linguistics*, 7:677–694.

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.

Swarnadeep Saha, Sayan Ghosh, Shashank Srivastava, and Mohit Bansal. 2020. PRover: Proof generation for interpretable reasoning over rules. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 122–136, Online. Association for Computational Linguistics.

Swarnadeep Saha, Prateek Yadav, and Mohit Bansal. 2021. multiPRover: Generating multiple proofs for improved interpretability in rule reasoning. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3662–3677, Online. Association for Computational Linguistics.

Oyvind Tafjord, Bhavana Dalvi, and Peter Clark. 2021. ProofWriter: Generating implications, proofs, and abductive statements over natural language. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3621–3634, Online. Association for Computational Linguistics.

Alon Talmor, Oyvind Tafjord, Peter Clark, Yoav Goldberg, and Jonathan Berant. 2020. Leap-of-thought: Teaching pre-trained models to systematically reason over implicit knowledge. In *Advances in Neural Information Processing Systems*, volume 33, pages 20227–20237. Curran Associates, Inc.

Leon Weber, Pasquale Minervini, Jannes Münchmeyer, Ulf Leser, and Tim Rocktäschel. 2019. NLProlog: Reasoning with weak unification for question answering in natural language. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6151–6161, Florence, Italy. 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*.

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.

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.

Honghua Zhang, Liunian Harold Li, Tao Meng, Kai-Wei Chang, and Guy Van den Broeck. 2022. On the paradox of learning to reason from data. *arXiv*, abs/2205.11502.

Shujian Zhang, Chengyue Gong, and Eunsol Choi. 2021. Learning with different amounts of annotation: From zero to many labels. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7620–7632, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

A   Implementation Details

Hugging Face `transformers` version: 4.19.1
Two NVidia Quadro 8000 GPUs were used for all experiments in this paper.

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

Table 6: End-to-end model fine-tuning configuration (hyperparameters left at `transformers` default if unspecified)

Hyperparameter	Value
Base model	T5-3b
Total batch size	12
Initial LR	5e-5
Epoch count	2 (early stopping on val. loss)

Table 7: EntailmentBank-only step deduction model fine-tuning

Hyperparameter	Value
Base model	T5-3b
Total batch size	12
Initial LR	5e-5
Epoch count	1

Table 8:   Step deduction model mid-tuning on ParaPattern-substitution

Hyperparameter	Value
Base model	DeBERTa Large MNLI
Total batch size	32
Initial LR	1e-5
Epoch count	1

Table 9:   Goal entailment model fine-tuning on EBENTAIL+EBENTAIL-ACTIVE

Hyperparameter	Value
Base model	DeBERTa-v3 Large
Total batch size	32
Initial LR	2e-5
Epoch count	2 (no goals) 7 (w/goals) (early stopping on val. loss)

Table 10: Learned heuristic model fine-tuning

B Examples

An ocean plant is a kind of water plant.

A fossil of an ocean plant is found in the desert.

A fossil of a water plant is found in the desert.
→ goal: 0.02

If fossils of a water animal or plant are found in a place then that place used to be covered by water in the past.

Goal: The desert used to be covered by water in the past.
→ goal: 0.96

Geothermal energy is a renewable resource.

Producing renewable electric energy requires renewable resources.

Producing renewable electric energy requires geothermal energy.
→ goal: 0.58

If a process requires something that that something may be able to perform that process.

Geothermal energy can be used for producing renewable electric energy.
→ goal: 0.75

If something may be able to do something else then that something has the potential to do that something else.

Goal: Geothermal energy has the potential to produce renewable electric energy.
→ goal: 0.94

Changing the ph of water can cause animals live in the water not be able to survive.

A tadpole lives in water.

A tadpole living in water will not survive if the ph of water changes.
→ goal: 0.01

A tadpole is a kind of animal.

Changing the ph of water can cause a tadpole to not be able to survive.
→ goal: 0

Acid rain changes the ph of water.

Acid rain can cause a tadpole to not be able to survive.
→ goal: 0.09

Acid rain and changing the ph of water can cause a tadpole to not survive.
→ goal: 0.85

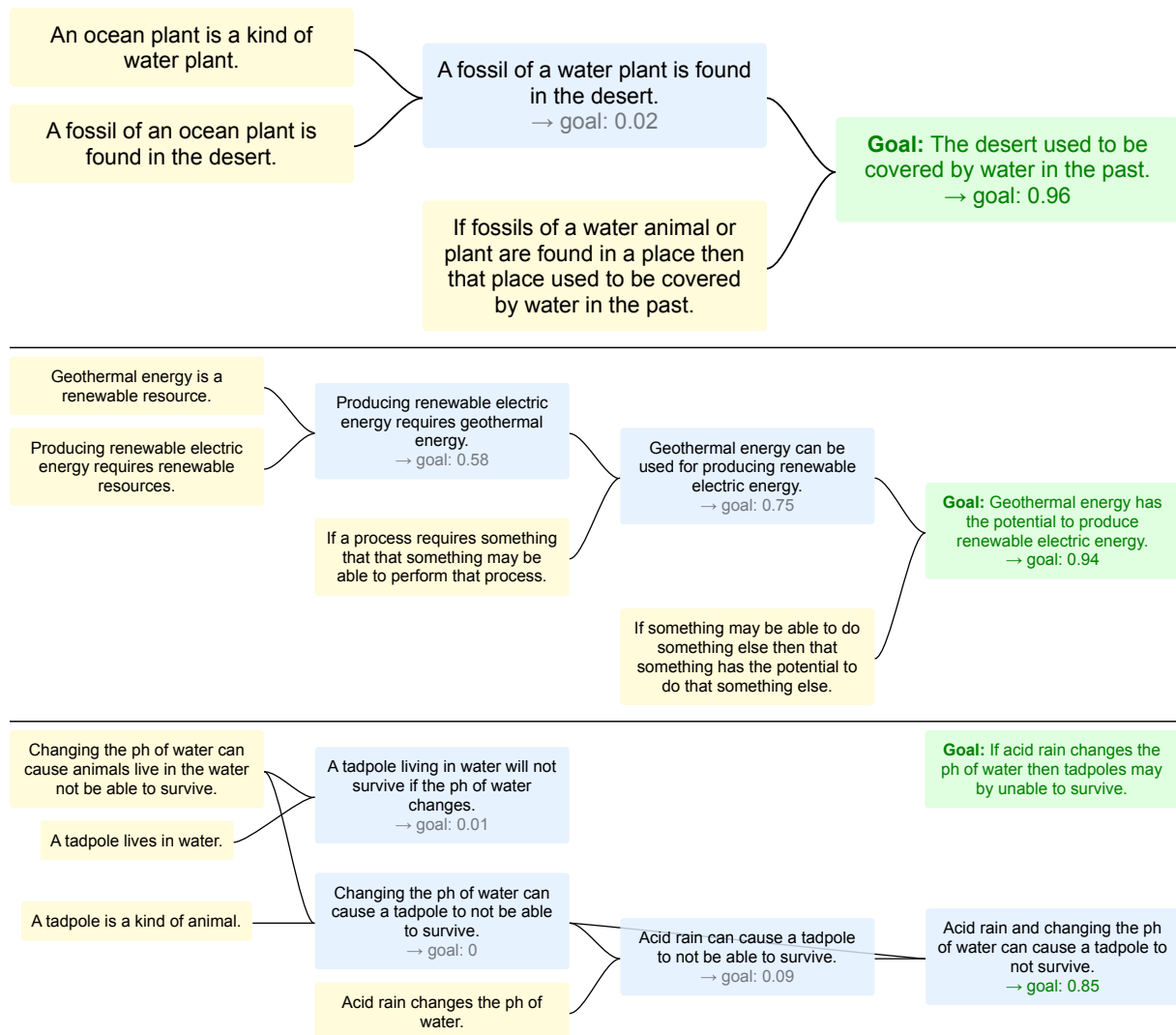**Goal:** If acid rain changes the ph of water then tadpoles may by unable to survive.

Figure 5: Examples of entailment trees generated by our SCSearch system. In the second example, the first intermediate conclusion exhibits an overzealous substitution, unifying 'renewable resources' in '[...] requires renewable resources' with 'geothermal energy' despite geothermal energy being just one example of a renewable resource. This is one form of the indiscriminate unification issue noted in Section 5.4. The last example includes an additional step sampled outside of the successful entailment tree; this step technically entails the goal but goes undetected by the goal entailment module.