# Heuristic Guidance of Scalable Explanatory Inference

**Pat Langley** PATRICK.W.LANGLEY@GMAIL.COM
Institute for the Study of Learning and Expertise, Palo Alto, California 94306 USA
School of Computer Science, University of Auckland, Private Bag 92019, Auckland 1142 NZ

**Mohan Sridharan** M.SRIDHARAN@BHAM.AC.UK
School of Computer Science, University of Birmingham, Birmingham B15 2TT UK

## Abstract

Despite well-known limitations, human cognition exhibits remarkable abilities for scaling to factors like task complexity and knowledge base size. In this paper, we revisit a recently proposed theory of explanatory inference and its implementation in the PENUMBRA system, which we hypothesize will support similar properties. We examine the computational costs associated with the architecture's basic inference cycle, which alternates between selecting a focus belief, elaborating current explanations, and repairing violated constraints. At a higher level, we study PENUMBRA's effectiveness at searching the space of alternative explanations for a set of observations. We conclude with comments on related work and proposals for future research.

## 1. Introduction

Decades of research have established that humans are inherently limited information processors. People have imperfect memories, they can focus on only one thing at a time, and they have difficulty with long chains of reasoning. Yet despite these limitations, they are remarkably effective at many complex cognitive tasks, and their performance improves further as they gain expertise in an area. Any complete computational theory of high-level intelligence must account for these intriguing and important phenomena. Human cognition is amazingly scalable to both the complexity of tasks and the size of knowledge bases, due at least partially to its reliance on heuristics to mitigate its processing constraints.

In this paper, we address the general task of explaining observations, in which an intelligent system, human or otherwise, attempts to understand events it encounters in terms of available knowledge. Many instances of such explanation involve abductive reasoning (Peirce, 1878), in that they require one to introduce plausible assumptions. There is a substantial literature on abductive and explanatory inference that we will not review here. Some of this tradition has examined issues of efficiency, but there have been few efforts to draw links to human cognition. For example, we know that as people acquire domain expertise, their increased knowledge does not slow down their processing. They can also construct complex, highly interconnected explanations without much difficulty. Moreover, when confronted with many possible explanations (e.g., different parses for a sentence), people often find the best alternative as rapidly as when there are fewer choices.

In the sections that follow, we present a promising account of such scalable explanatory reasoning. We start by reviewing an earlier theory of associative abduction and its implementation in PENUMBRA, an architecture for plausible inference. After this, we examine the computational costs of the system's basic cognitive cycle, both formally and empirically, in terms of factors like the number of rules and their complexity. Next we consider the effort required to construct complete explanations, focusing specifically on scaling to increasing numbers of consistent alternatives. In closing, we discuss related work on the efficiency of complex cognitive tasks and proposals for future research on this topic.

## 2. A Theory of Explanatory Inference

In a recent article, Langley and Meadows (2019) presented a new theory of abductive inference for explaining observations in terms of background knowledge. They distinguished between derivational abduction, in which observed facts serve as the roots of proof graphs, and associative abduction, in which they are terminal nodes of such graphs, along with assumptions. The theory falls into the second paradigm and incorporated a number of postulates about representation and processing, which we review in this section. We also summarize PENUMBRA, an implemented architecture that reflects these theoretical ideas.

### 2.1 Representational Postulates

As with other theories of complex cognition, Langley and Meadows' account distinguishes between stable knowledge and dynamic beliefs. The framework posits two forms of generic knowledge structures. *Definitions* specify a higher-level relation in terms of other more basic ones; these are often organized hierarchically, much as in a traditional grammar, with defined terms serving as nonterminal symbols. *Constraints* comprise a set of relations that are mutually exclusive, so they indicate inconsistency when they are satisfied jointly. A dynamic working memory contains three types of ground beliefs – *observed*, *derived*, and *abduced* – that differ in their origins.

The theory also states that *justifications* – instances of applied definitions – are stored and organized into higher-level explanations. The latter structures take the form of proof graphs, with observed and abduced beliefs as terminal nodes and with derived beliefs as nonterminals. Moreover, although these beliefs are stored in a single working memory, they are associated with distinct *worlds*, which are stored in a distributed manner by annotating each belief with the worlds in which it does *not* hold. Finally, the framework organizes worlds into a *phylogenetic tree* that traces their evolution, with *closed* worlds containing known constraint violations and *active* ones, at the frontier, thought to be internally consistent.

### 2.2 Processing Postulates

This theory of explanatory reasoning also includes tenets about processing. The most basic claim is that this mental activity involves two cognitive cycles: an outer observation loop that accepts new facts from the environment and an inner inference loop that extends and revises explanations of these facts. The inference cycle alternates among selecting an existing belief on which to focus, invoking

definitional rules to elaborate explanations, and using constraints to detect and repair inconsistencies in these accounts. The chosen focus belief mediates the retrieval of relevant knowledge, in that processing only considers definitions and constraints with an antecedent that unifies with this focus. When a retrieved definition is selected for application, the inference process generates a new derived belief for its head and, if some of the antecedents are unmatched, creates new abduced beliefs for them. If the retrieval mechanism reveals that a constraint is violated – because two mutually exclusive beliefs reside in the same worlds – then the reasoning mechanism deactivates these inconsistent worlds and generates new, active children that remove the conflict. Elaboration through definitions is a monotonic process, whereas repair of violated constraints is nonmonotonic.

At a higher level, the process of explanation construction is aided by knowledge but driven by data, in that it responds to observations that may arrive incrementally. Because multiple accounts of the same observed facts may be possible, this involves a search through a space of alternative explanations that are consistent with the data. Heuristics for selecting focus beliefs, definitions, and constraints determine the order in which candidates are generated, and thus guide the search process. However, because worlds are encoded in a distributed manner, this activity also exhibits a form of implicit parallelism, in that each inference step can elaborate or repair multiple worlds that share the beliefs involved.

### 2.3 The Penumbra System

Langley and Meadows (2019) also reported PENUMBRA, an implemented system that embodies these theoretical ideas. This comes with a programming language for specifying definitions, constraints, and beliefs that follow the representational postulates mentioned earlier. PENUMBRA's syntax is similar to Prolog, with definitions having a head and antecedents, each comprising a predicate and arguments, with the latter being variables that may be shared across relations. Constraints include a head and a set of antecedents, each having a predicate and variable arguments, that are mutually exclusive and thus should never appear in the same explanation. Observed, derived, and abduced beliefs are analogous to ground facts in Prolog, with arguments of predicates being constant terms or, in some cases, Skolems. The language does not include notation for specifying worlds, as processing always starts with a root world and new ones are generated automatically.

The architecture also includes an interpreter for running programs stated in this syntax. This also follows the theory by incorporating an observation cycle that accepts environmental input and an inner inference cycle that constructs explanations. On each inference cycle, PENUMBRA selects a focus belief, checks for violations of constraints linked to this focus, and, if it detects any, deactivates the inconsistent worlds and creates active children that eliminate the problem. If the system detects no violations, it selects a definition with an antecedent that unifies with the focus, then applies the rule to elaborate worlds in which the matched conditions hold. Application of definitions produces a new derived belief based on the rule's head and abduced beliefs based on its unmatched antecedents. Both cycles continue until no more observations are available and no focus belief is selected. PENUMBRA includes eight parameters that determine how the interpreter selects focus beliefs, constraints, and definitions, with different settings producing different search for explanations.

*Table 1.* Characteristics of PENUMBRA structures that influence processing time during the inference cycle. The cost term $T_R$ should be constant throughout a run because the knowledge base is fixed, whereas $T_F$ should increase during a run as the number of beliefs grows.

| |
|---|
| $N_R$ is the number of distinct relational predicates |
| $N_C$ is the number of constraints |
| $N_D$ is the number of definitions |
| $N_O$ is the number of observed beliefs |
| $N_B$ is the total number of beliefs |
| $N_E$ is the number of consistent worlds / explanations |
| $A_C$ is the average number of alternatives per constraint |
| $A_D$ is the average number of antecedents per definition |
| $C_P$ is the average number of constraints per predicate |
| $D_P$ is the average number of definitions per predicate |
| $B_P$ is the average number of beliefs per predicate |
| $B_E$ is the average number of beliefs per world / explanation |
| $T_F$ is the processing time per focus cycle |
| $T_R$ is the processing time per rule cycle |

## 3. Scalability of the Inference Cycle

Like most other reasoning systems, PENUMBRA operates in cognitive cycles, with the main work occurring during an inference loop. If we want to ensure that its construction of explanations is efficient, then we must first show that its cycle-level processing scales well to complicating factors. This was the motivation for early research on efficient matching in production system architectures (Klahr, Langley, & Neches, 1987), which led to techniques like Rete (Forgy, 1982) and TREAT (Miranker, 1987). In this section, we examine analogous issues for our architecture for explanatory inference, which we can partition into the cost of selecting definitional rules and the expense of selecting focus beliefs.[1] We start with a formal analysis and then report empirical studies for the more expensive activity, selecting definitions.

We can identify a variety of factors that could influence PENUMBRA's processing time, which we have listed in Table 1. Some of these concern knowledge stored in the system's long-term memory, whereas others deal with the content of working memory. The first set includes the number of definition rules, $N_D$, the average number of antecedents per definition, $A_D$, the number of distinct relational predicates, $N_R$, the average number of definitions per predicate, $D_P$, the average number of alternatives per constraint, $A_C$, and the number of constraints, $N_C$. We will not examine the last two constraint-related factors here, but the others play a role in our analysis. Factors related to working memory include the number of observed beliefs, $N_O$, the total number of beliefs, $N_B$, and

1. We will not discuss the cost of checking constraints, which is negligible because they are indexed by predicates that appear in them and they do not involve combinatorial matching.

the number of consistent worlds or explanations, $N_E$. Because PENUMBRA adopts a distributed representation, the number of beliefs per world is also relevant, but we will not consider it here.

### 3.1 Processing Costs of the Inference Cycle

We can analyze separately the three stages of PENUMBRA's inference cycle – choosing a focu belief, checking for constraint violations, and selecting a definition to apply. The computational cost of the first stage is most straighforward. The time for focus selection should be affected only by the number of beliefs $N_B$ in working memory. For a naive strategy, this gives

$$T_F = j * N_B \ ,$$

but we must add a few caveats. One is that $N_B$ will grow over time, as the system applies definitions and elaborates its explanations. However, if alternative worlds share many of their beliefs, as we have observed in practice, this growth will be sublinear in the number of consistent accounts $N_E$. Also, we can limit the number of beliefs considered, say by shelving all but the $k$ most recently added elements. Older beliefs would still be available for matching, but they would not compete for attention unless refreshed.

The computational expense of constraint checking is somewhat more complicated. Because PENUMBRA indexes constraints by predicates that appear in them, and because it checks them only after it has selected a focus, it must consider only $C_P$ candidate rules. The total number of constraints, $N_C$, should not affect processing time. Each of the retrieved constraints has $A_C$ mutually exclusive antecedents, one of which has already matched the focus belief, so the system must consider the remaining $A_C - 1$ of them. Each of these, in turn, can potentially unify with $B_P$ beliefs that share their predicate, which gives the expression

$$T_C = i \cdot C_P \cdot (A_C - 1) \cdot B_P \ .$$

Because $A_C$, the number of mutually exclusive relations per constraint, will typically be small, the important factors are $C_P$ and $B_P$. Note that, on a given cycle, PENUMBRA may well overlook some related constraints because they do not involve the focus belief, but it may detect them later, when its attention shifts. We have not discussed the cost of repairing violated constraints once detected because the system stores the rule instances that produced each belief, so it can remove the sources of an inconsistency by a simple retrieval process.

Finally, we can analyze PENUMBRA's retrieval and matching mechanisms to calculate the cost of selecting a definitional rule instance on a given cycle. Again, we can assume the system has already selected a focus belief that involves some predicate. Because definitions are indexed by predicates that occur in their antecedents, it will only retrieve the $D_P$ rules in which they appear; the total number of definitions, $N_D$, will not affect processing time. The cost of matching each retrieved rule is influenced by the number of beliefs, $B_P$, that contain the focus predicate, and the number of antecedents $A_D$ in the definition. Because PENUMBRA allows partial matching in support of abductive reasoning, it considers all ways that antecedents (other than the one that unifies with the focus) can match or fail to match.

For each antecedent, it retrieves $B_P$ beliefs with the same predicate. The system considers all combinations of these belief-antedecent pairs, as well as the possibility that each antecedent has no match, giving $(B_P + 1)^{(A_D-1)}$ candidate partial matches. It will reject many candidates because their variables do not bind consistently, but it must still consider them. Multiplying this expression by the number of retrieved definitions gives the equation

$$T_R = k \cdot D_P \cdot (B_P + 1)^{(A_D-1)}$$

as the computation time required to find all partial matches of definitions that unify with a given focus belief. The constant $k$ includes the time needed per cognitive cycle to evaluate each of these candidates and select the best-scoring one. This analysis assumes that the factors $D_P$, $B_P$, and $A_D$ remain constant for a particular run, when they will typically vary, but the equation has direct implications for PENUMBRA's scaling behavior.

### 3.2 Experimental Studies of Rule Selection

The analyses above suggest a number of hypotheses that are subject to empirical test. We decided to focus on one subprocess, selecting a definition to use in elaboration, because it is substantially more expensive than choosing a focus or checking constraints. The associated analytic equation includes four factors, so we designed and carried out experiments that varied each of them separately. We devised a number of synthetic knowledge bases and working memories that would let us vary these factors systematically, and thus test the analysis' predictions empirically.

Table 2 presents abstract versions of the rules for nine different values of $D_P$ and $A_D$. In fact, each antedecent take two arguments, at least one of which is shared with another condition, to ensure relevance to the relational character of many explanatory tasks. For instance, the first rule for the $A_D = 3, D_P = 2$ condition is *(d1 ?x ?y ?z ?w) <= (p1 ?x ?y) (p2 ?y ?z) (p3 ?z w)*, which matches against beliefs like *(p1 a b)*, *(p2 b c)*, and *(p3 c d)*. The table does not show beliefs, which we can vary independently of the definitions by creating multiple copies of element sets like those above that have disjoint arguments. Note that the rules are not organized in a hierarchy, such as that for a context-free grammar; they specify a flat set of connected relations. However, this structure should not affect the results of experiments, which should depend only on the factors from the analysis.

We designed and ran four experiments using these synthetic knowledge bases. For each experimental condition, we ran PENUMBRA 50 times in nonincremental mode, in that all observed beliefs were available at the outset, letting it continue until the system made no further inferences. The number of inference steps in these ranged from 9 to 34. The knowledge bases did not include any constraints, which meant that the only processes in operation were focus and definition selection. Also, we specified that PENUMBRA should only apply rules when all their antecedents were satisfied, so it never introduced any default assumptions, although the mechanism still considered partial matches. For each run, we measured $T_R$, the time to select a definition to apply, in CPU seconds. We averaged the observed values for $T_R$ across cycles within each run and across runs for each condition, which lets us calculate both means and standard errors.

The first and most important conjecture deals with the apparent independence of the time needed for selecting rules from the number of definitions:

*Table 2.* Synthetic knowledge structures that vary two factors – the number of antecedents per definition, $A_D$ and the number of definitions per predicate, $D_P$, that should influence the cost of rule selection. Each antecedent takes two arguments, at least one of which it shares with another condition.

| | $A_D = 2$ | $A_D = 3$ | $A_D = 4$ |
|---|---|---|---|
| $D_P = 2$ | D1 <= P1 P2<br>D2 <= P2 P3<br>D3 <= P3 P4<br>D4 <= P4 P5<br>D5 <= P5 P6<br>D6 <= P6 P1 | D1 <= P1 P2 P3<br>D2 <= P3 P4 P5<br>D3 <= P5 P6 P7<br>D4 <= P7 P8 P9<br>D5 <= P2 P4 P8<br>D6 <= P1 P6 P9 | D1 <= P1 P2 P3 P4<br>D2 <= P3 P4 P5 P6<br>D3 <= P5 P6 P7 P8<br>D4 <= P7 P8 P1 P2 |
| $D_P = 3$ | D1 <= P1 P2<br>D2 <= P1 P3<br>D3 <= P1 P4<br>D4 <= P2 P3<br>D5 <= P2 P4<br>D6 <= P3 P4 | D1 <= P1 P2 P3<br>D2 <= P2 P3 P4<br>D3 <= P3 P4 P5<br>D4 <= P4 P5 P6<br>D5 <= P5 P6 P1<br>D6 <= P6 P1 P2 | D1 <= P1 P2 P3 P4<br>D2 <= P1 P3 P5 P7<br>D3 <= P2 P4 P6 P7<br>D4 <= P5 P6 P7 P8<br>D5 <= P1 P2 P5 P6<br>D6 <= P3 P4 P7 P8 |
| $D_P = 4$ | D1 <= P1 P2<br>D2 <= P1 P3<br>D3 <= P1 P4<br>D4 <= P1 P5<br>D5 <= P2 P3<br>D6 <= P2 P4<br>D7 <= P2 P5<br>D8 <= P3 P4<br>D9 <= P3 P5<br>D10 <= P4 P5 | D1 <= P1 P3 P4<br>D2 <= P1 P3 P5<br>D3 <= P1 P4 P6<br>D4 <= P1 P5 P6<br>D5 <= P2 P3 P4<br>D6 <= P2 P3 P5<br>D7 <= P2 P4 P6<br>D8 <= P2 P5 P6 | D1 <= P1 P2 P3 P4<br>D2 <= P2 P3 P4 P5<br>D3 <= P3 P4 P5 P6<br>D4 <= P4 P5 P6 P1<br>D5 <= P5 P6 P1 P2<br>D6 <= P6 P1 P2 P3 |

- *Processing time per rule selection $T_R$ is constant in the size of the number of definitions $N_D$.*

To test this claim we varied the number of definitions while holding other variables constant ($A_D$ = 3, $D_P$ = 3, $B_P$ = 3). We started from the six definitions in the center of Table 2 ($N_D = 6$) and introduced new rules with the same structure but different predicates for other conditions ($N_D = 12, 18, 24$). Figure 1 (left) shows the results of this experiment, which differ from those predicted. Rather than being constant, the CPU time required to select a rule appears to grow as a linear function of the number of definitions in memory. The time per inference cycle is small and the slope is small, but the curve suggests that PENUMBRA may slow down substantially when provided with thousands of rules. We should revisit both our analysis and the software to determine the source of the discrepency. If our analysis is sound, then the growth is due to an inefficient implementation.

A second hypothesis relates the time needed to select a rule to the 'branching factor' of definitions from predicates that appear in their antecedents:

- *Processing time per rule selection $T_R$ is a linear function of the average number of definitions per predicate $D_P$.*

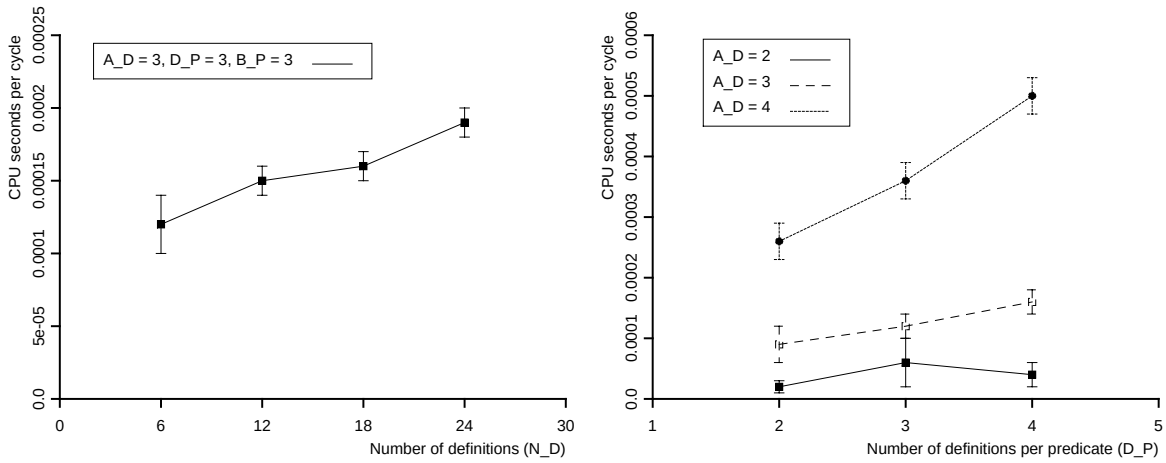


*Figure 1.* Processing time per inference cycle in CPU as a function of (left) the total number of definitions ($N_D$) and (right) the number of definitions per predicate ($D_P$). For the first plot, we set $A_D = 3$, $D_P = 3$, and $B_P = 3$; for the second, we set $B_P = 3$.

To check this prediction we varied the number of definitions per predicate ($D_P$ = 2, 3, 4) while holding the beliefs per predicate constant at $B_P$ = 3. We repeated this at three settings for the number of antecedents per definition. Figure 2 (right) presents the results with separate curves for $A_D$ = 2, 3, and 4, each corresponding to one of the columns in Table 2. These generally support the hypothesis, as the two upper curves are approximately linear in $D_P$. The lowest curve does not increase with this factor, but this could be a floor effect.

A third implication of the analysis is that the time needed to select a definition grows exponentially with rule complexity:

- *Processing time per rule selection $T_R$ is an exponential function of the average number of antecedents per definition $A_D$.*

We tested this claim by varying the number of antecedents per definition ($A_D$ = 2, 3, 4) while holding the beliefs per predicate constant at $B_P$ = 3. We examined PENUMBRA's behavior at three settings for the number of definitions per predicate. Figure 2 (left) displays the CPU time per inference cycle, giving separate curves for $D_P$ = 2, 3, and 4, each of which corresponds to one of the rows in Table 2. These differ in the number of definitions, which we could hold constant in this study. All three curves are nonlinear, which is consistent with the prediction of exponential growth in $A_D$, antecedents per definition. This is not especially concerning, as it seems unlikely that PENUMBRA's definitions will often have more than a few antecedents.

A final hypothesis states that the computation time needed for rule selection is polynomial in how many beliefs involve the same predicate as the focus belief:

- *Processing time per rule cycle $T_R$ is a polynomial function of the average number of beliefs per predicate $B_P$.*

To evaluate this conjecture, we varied the number of beliefs per predicate ($B_P$ = 1, 2, 3, 4, 5) while holding the other factors constant ($A_D$ = 6, $D_P$ = 3, $N_D$ = 6). As Figure 2 (right) reveals, the

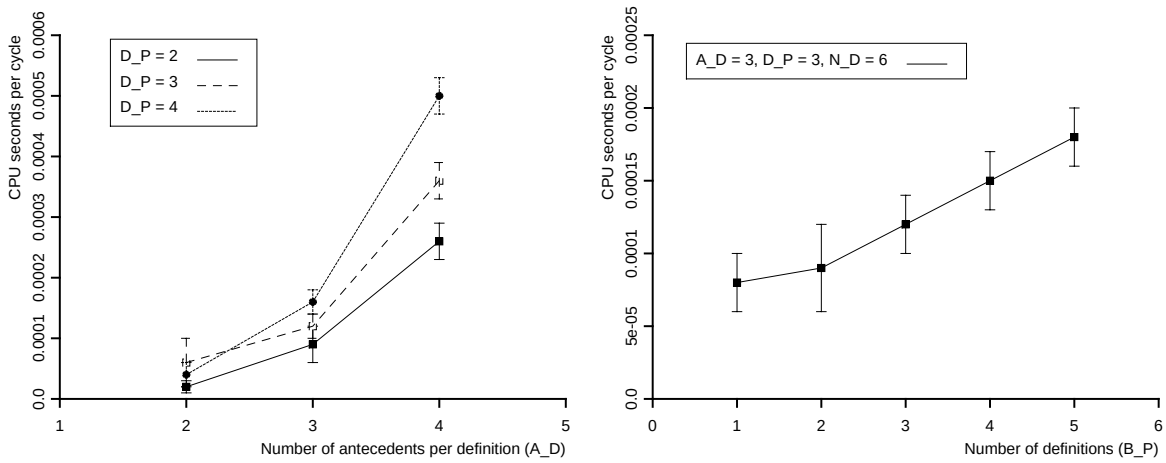


*Figure 2.* Processing time per inference cycle in CPU as a function of the number of (left) antecedents per definition and (right) beliefs per predicate. For the first plot, we set $B_P = 3$; for the second, we set $N_D = 6$, $A_D = 3$, and $D_P = 3$.

increase in CPU time per inference cycle with $B_P$ was more rapid than linear, but appears less than exponential. This is consistent with the prediction of polynomial growth in this factor, although we should extend the curve with higher settings to increase confidence in this conclusion. This relationship is a greater concern than exponential growth in $A_D$, since the number of beliefs per predicate may increase over an extended run.

## 4. Scalability of Explanation Construction

Although efficient behavior at the level of the inference cycle is important, it is not enough on its own. The process of generating full explanations must also operate in a scalable manner. For example, we would like the time needed to construct a good account to increase slowly with the number of observations and with the complexity of the resulting structures. Informal analysis suggests that the number of inference cycles should grow as a linear function of both factors, but we will not address them here. Instead, we will focus on an even more important matter – scalability to the number of alternative explanations – that arises because the task involves combinatorial search.

Consider a phenomenon well known in human language processing. Some sentences have many possible parses, yet we take little if any more time to understand them than ones with fewer parses (*cite*). Given what we know about human cognition, the natural conclusion is that this occurs because people draw on effective heuristics to guide their choices. Such heuristics can occasional mislead processing, as demonstrated by the existence of garden path sentences (e.g., *The old man the boats*), but these are exceptions that prove the rule of thumb. PENUMBRA relies on heuristics at two key choice points: selection of focus beliefs and selection of definitions. This suggests a hypothesis about the system's behavior:

- *Given effective heuristics, the time needed to find the best explanation is independent of the number of consistent worlds.*

In other words, during its search through the space of explanations, PENUMBRA should find the best candidate first and should not be distracted by competing accounts that have lower quality. However, before we can test this claim, we must first make it operational. This means that we must specify what we mean by the term 'best' and which heuristics are likely to be effective.

### 4.1 Explanation Quality and Heuristics

The notion that one explanation is 'best' implies some criterion for determining their quality.[2] A classic criterion is *simplicity*, which favors accounts that involve fewer reasoning steps or that rely on fewer assumptions (e.g., Peng & Reggia, 1986). However, Ng and Mooney (1990) have argued instead for using *coherence*, which they define in terms of the number of ways that beliefs are linked to each other through available knowledge. Another approach associates numbers with individual rules, possible assumptions, or both. For instance, Hobbs et al.'s (1993) abduction system assigned weights to each possible assumption and ranked alternative explanations by the summed weights of their assumptions, with lower totals being better, whle Appelt and Pollack (1992) take a similar approach. Parsers that rely on probabilistic context-free grammars instead assign probabilities to each rule, then compute a posterior probability for candidate parse trees by multiplying the scores for each rule involved in them (e.g., Charniak & Shimony, 1990).

We will not take a position here about which of these criteria are more desirable, but, for the sake of illustration, we will adopt a variant of the latter scheme in testing our hypothesis. Suppose that each PENUMBRA definition comes with a score between zero and one, and that the quality of an explanation is the average score of its component rule instances. Naturally, the heuristic for selecting rules should bear some relation to their role in overall quality. If the criterion prefers accounts with higher-scoring rule instances, then a promising heuristic is to favor candidates with higher individual scores. This is similar in spirit to conflict resolution in production system architectures that prefer instances of rules with higher strenghts (e.g., Anderson & Lebiere, 1998). Given that PENUMBRA can apply rules with unsatisfied antecedents, we might modulate this score, say by multiplying it by the fraction of unmatched conditions. This will not be relevant to the experiments reported later, which involve purely deductive reasoning, but it may play a role in future work.

However, an effective heuristic for rule selection is not enough for PENUMBRA to generate high-quality accounts it considers low-quality ones. For this ability, the system must also carry out depth-first search through the space of explanations by elaborating on the most promising world before turning to others. This strategy will sometimes lead it to extend worlds that were not the best choice in hindsight, as with garden path sentences, but that is the nature of heuristics. Nevertheless, they often recommend the right option, in this context leading the elaboration process to construct the highest-quality explanation before it turns to others.

PENUMBRA's criterion for selecting focus beliefs is the mediator that determines whether it behaves in a depth-first manner. If the system selects a focus at random, then it will tend to switch between elaborating one world and extending others, approximating breadth-first search. In contrast, if it prefers to focus attention on more recently generated beliefs, then it will elaborate on the

2. Some authors (e.g., Eckroth & Josephson, 2014) have defined abduction as 'inference to the best explanation', but not all explanations are abductive and finding the best should not be part of its definition.

*Table 3.* A simple subset of English syntax stated as a context-free grammar, each with an associated score, and five parses of the sentence *A dog chased the cat a stick on the roof* with this grammar.

| *Grammatical Rules* | *Score* | *Allowed Parses* | *Score* |
|---|---|---|---|
| *S -> NP VP* | *1.0* | *((A dog) (chased (the cat (with (a stick))) (on (the roof))))* | *0.726* |
| *VP -> V NP* | *0.4* | *((A dog) (chased (the cat) (with (a stick)) (on (the roof))))* | *0.716* |
| *VP -> V NP PP* | *0.9* | *((A dog) (chased (the cat (with (a stick)) (on (the roof)))))* | *0.721* |
| *VP -> V NP PP PP* | *0.5* | *((A dog) (chased (the cat) (with (a stick (on (the roof))))))* | *0.721* |
| *NP -> ART N* | *0.5* | *((A dog) (chased (the cat (with (a stick (on (the roof))))))* | *0.679* |
| *NP -> ART N PP* | *0.2* | | |
| *NP -> ART N PP PP* | *0.8* | | |
| *PP -> P NP* | *0.9* | | |

current world before it extends others.[3] Such a bias has been a common method for conflict resolution in production system architectures, and Young (1982) has noted that it offers a natural way to model depth-first strategies. To keep this scheme from producing infinite loops, it is often combined with a 'refraction' technique that forbids reapplying the same rule instance until its matched elements are refreshed. PENUMBRA can mimic this mechanism by decreasing a belief's score once it has served as the focus, redirecting attention to unused elements instead.

Recall that applying a definitional rule, even in the purely deductive case, produces at least one new belief: its instantiated head. If the system focuses on more recent elements, then it will first consider rules that match it and extend the current explanation. An example should clarify this effect. Table 3 presents a context-free grammar for a subset of English that includes prepositional phrases, which are an important source of ambiguity. PENUMBRA encodes this syntactic knowledge differently in that it includes explicit constraints (e.g., the same prepositional phrase cannot be attached to both a noun and a verb), but the traditional notation will simplify our discussion. Also, each definition has an associated weight that indicates its preference relative to others. Suppose we provide PENUMBRA with this knowledge and with the sentence *A dog chased the cat a stick on the roof*. Table 3 shows five parses supported by the grammar, along with the average score of rules used to construct them and the resulting scores for each interpretation.

For the sake of argument, assume that we present PENUMBRA with the entire sentence at once, rather than incrementally. On each inference cycle, the system focuses on the most recently created belief that has not led to a rule application. Combined with a heuristic for selecting the retrieved (through connection with the focus) and matched definition with the highest score, this bias generates the first parse tree in the table before it considers the others. PENUMBRA first produces beliefs for the four noun phrases and then embeds the latter two in prepositional phrases. Next, the system decides to include *(with (a stick))* in *(the car (with (a stick)))* because the ..., after which includes this noun phrase and the prepositional phrase *(on (the roof))* into the verb phase *(chased (the cat (with (a stick))) (on (the roof))*. Finally, it combines this constituent with the noun phrase *(a dog)* to produce the first parse tree in the table.

3. Because PENUMBRA represents worlds in a distributed fashion, each inference can extend more than one explanation.

Of course, PENUMBRA does not encode this as a list structure, but rather as as a set of beliefs, including nonterminal predicates, that are connected through rule instances. Moreover, the system can reuse many of these elements (especially those at lower levels) in other parses it considers later, since they share many constituents. Each alternative parse is associated with a distinct world, but these are stored as markers on working memory elements, so that common beliefs needed only be inferred once. However, we are concerned here with how PENUMBRA can arrive at the best-scoring explanation before even considering these other interpretations.

### 4.2 Experiments on Explanation Scalability

We can build on this informal analysis to to test our hypothesis about the scalability of explanation contruction in PENUMBRA. This predicts that, as we increase the number of consistent accounts, the time taken to find the best candidate will be unaffected. Whether the two heuristics we discussed – recency for selection of focus beliefs and definition score for rule selection – ar effective enough to produce this result is an empirical question, but the parsing example give cause for hope. Because we have already examined scaling of costs per inference cycle, we will use number of cycles as the dependent variable here rather than CPU time.

Again, the main independent factor is the number of consistent explanations for a given set of observations that are supported by PENUMBRA's knowledge base. However, we also want to hold constant other possible influences on processing time. We have already addressed factors like the number of rules, their number of antecedents, and the number of beliefs per predicate in our study of cycle time. The two remaining influences are the number of observations to be explained and the complexity of explanations. The parsing task is a convenient setting to test the hypothesis because it is familiar, we can easily control the number of observations (sentence length), and different parse trees require approximately the same number of inferences.

However, it would seem difficult to vary the number of explanations without also altering these two factors. Fortunately, we can use a simple strategy to vary the number of alternative parses while holding them constant. Briefly, we can start with a general grammar like the one in Table 3 and restrict it by removing rules or by splitting nonterminal symbols. For example, deleting the third *VP* rule will eliminate one of the five parses. Similarly, replacing *PP* with *PP1* in some cases and *PP2* in others, along with replacing *PP => P NP* by *PP1 => P1 NP* and *PP2 => P2 NP*, allows only two parses. This change also requires modifying rules that associate parts of speech with particular words like *with* and *on*, which we omitted from the table.[4] We used this approach create three variations on the initial grammar that accepted different subsets of the original five parses of the sentence *A dog chased the cat a stick on the roof*, which we held constant across the study.

We should also note that our core hypothesis comes with a corollary. Because the ability to find the best explanation before others depends on use of effective heuristics, this should not occur when PENUMBRA instead selects focus beliefs or definitions at random. The earlier informal analysis suggests that both are prerequisites for finding the best account earlier than its competitors. Thus, our experiment should vary two factors: the number of consistent explanations (in this case parses) and whether the system relies on plausible heuristics described earlier to select focus beliefs and

4. The approach does require altering the number of definitional rules, and possibly the number of constraints, but we are not concerned with time per cycle here, only the number of inferences need to find the best explanation.

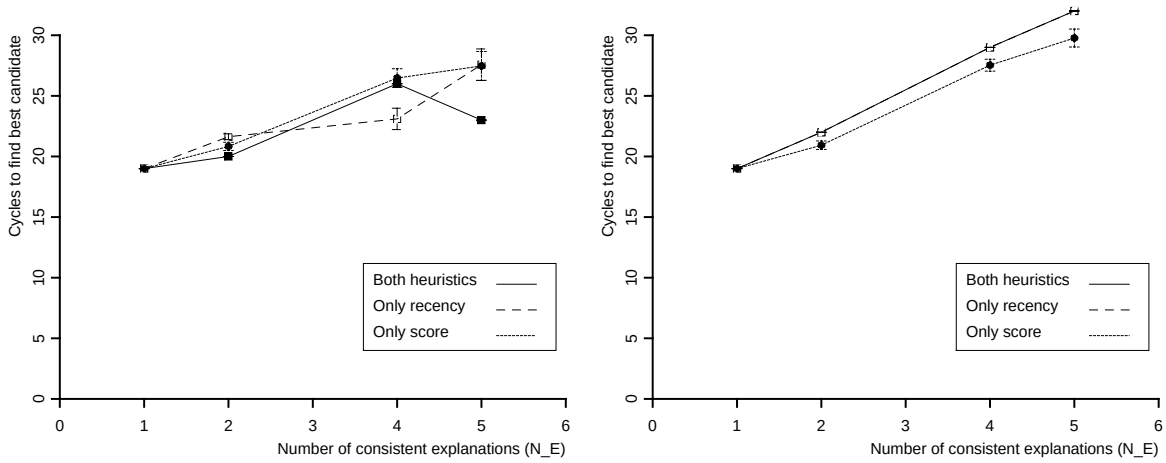


*Figure 3.* Number of cycles required (left) to find the best parse with variants of the grammar in Table 3 and (right) to find all parses using both belief recency and rule score as heuristics, using only rule score, and using only recency.

rule instances. We should note that other quality measures and rule heuristics, say ones that revolve around coherence, are definitely possible, but we will delay their examination until future research.

Because PENUMBRA stores beliefs in a distributed manner, by attaching worlds in which they do not hold, it is not obvious at first glance how to determine when it has found the best explanation. However, the system stores with each world the inference cycle on which it was modified most recently. If we run it until completion, that is, until it makes no more inferences and thus has found all consistent worlds, then we can find both the best-scoring world and the cycle on which it was last elaborated. We can use this number as the dependent measure of when the system found the best explanation. Of course, running PENUMBRA until it exhaustively finds all consistent explanations goes against the purpose of heuristic guidance, but we are relying on it only to test our hypothesis. If that claim holds, then in future studies it can halt after finding the first few explanations, since we will know they are high quality.

We followed this strategy using four variants of the grammar in Table 3, running the system 30 times and averaging the results. Figure 3 (left) presents the findings for this experiment. The *x* axis denotes the number of parses that are possible for the observed word sequence and the *y* axis encodes the inference cycle on which PENUMBRA found the best-scoring parse. The graph shows three separate curves. The solid line represents the system's behavior when it uses a recency heuristic to select focus beliefs and rule scores to select which rule instance to apply. Contrary to predictions, the number of inference cycles needed to find the best parse grows slowly with the number of alternatives, although there is an encouraging dip on the final point. Inspection of system traces revealed that it encountered constraint violations midway through the runs, indicating that it was not pursuing the full depth-first search as intended. The dotted line shows PENUMBRA's behavior when it instead selected a focus belief at random and the dashed line does the same when it chose rule instances at random. In both conditions, the number of inferences needed to find the best parse increases at about the same rate as version that used both criteria. This suggests that the

two heuristics are doing no better than random, so development of more effective ones should be a high priority for future work.

We also recorded the number of inference steps needed to find all parses for the sentence. The heuristics should play no role here, as the system must generate all possible accounts, regardless of their order. Figure 3 (right) shows that these curves fall only slightly above those for finding the best interpretation. The reason is that, when multiple parses were possible, they shared most of their constituents, which meant that only a few inferences remained to generate alternative parses after PENUMBRA found the first one. Thus, although the two heuristics provided less assistance than predicted for finding the best explanation before other candidates, this ability may be less important than expected in many settings.

## 5. Discussion

The research we have reported here draws on ideas from a variety of traditions in artificial intelligence. We have already mentioned some prior work on abductive inference, but we have been influenced the most by PENUMBRA's direct predecessors – AbRA (Bridewell & Langley, 2011) and UMBRA (Meadows et al., 2014) – that share its alternation between selecting a focus belief and deciding which rule to apply. As discussed elsewhere (Langley & Meadows, 2019), the main differences revolve around the use of constraints to handle inconsistencies and related support for multiple worlds. Our concern with scalability has its origins in work on efficient matching in production systems, which led to algorithms like Rete (Forgy, 1982) and TREAT (Miranker, 1987). However, these assumed all-or-none matching, which does not suffice for abduction scenarios that involve partial satisfaction of rules' antecedents. PENUMBRA's use of focus beliefs is a direct response to this issue, and our analysis of inference cycle costs took it into account, even though our experiments addressed purely deductive tasks.

The literature on abductive reasoning has dealt extensively with search through the space of explanations. Some approaches, including answer set programming (e.g., Baral, 2003) and probabilistic parsing (e.g., Charniak & Shimony, 1990), rely on exhaustive generation of alternatives, ranking them only afterwards. Other techniques combine numeric heuristics with more selective methods, from beam search (e.g., Ng & Mooney, 1990) to branch and bound (Hobbs et al., 1993). However, these lack PENUMBRA's alternation between focus and rule selection, which offers greater opportunities for guidance of the search process, but also more chances to be led astray. As noted earlier, the system's distributed representation of competing worlds also supports a form of implicit parallelism that reduces reliance on error-free heuristics. This idea bears a close resemblance to the approach used in assumption-based truth maintenance systems (de Kleer, 1986, 1994), which adopt a similar encoding to consider multiple interpretations of observations during an abduction-like process.

Although the empirical studies to date support some of our hypotheses about PENUMBRA's scalability, they have called others into question. Future research should analyze in greater detail the reasons for the latter, and we should attempt to replicate the positive findings with more complex reasoning tasks. Our next experiments should include abduction problems that require the introduction of plausible assumptions and they should examine scaling issues for selecting focus beliefs and constraint processing to complement our initial results on choosing definitions. Moreover, we

should test the system's abilities on large knowledge bases, such as the Monroe corpus (Blaylock & Allen, 2005), that other researchers have developed.

In addition, we should explore other criteria for explanation quality and associated heuristics. One approach would calculate probabilities for alternative accounts and use them to guide search through the space of candidates. Another would adopt Ng and Mooney's (1990) notion of explanatory coherence, which reflects people's preference for narratives that are tightly interconnected. We should also examine other heuristics for selecting the focus beliefs that serve as mediators for rule application. One promising idea involves favoring candidates that hold in approximately half of the active worlds, as rules that chain off them may provide more information. The PENUMBRA architecture, with its user-modifiable parameters, offers a promising framework for exploration of strategies for scalable explanatory inference.

## Acknowledgements

This research was supported by Grant N00014-17-1-2434 from the Office of Naval Research, which is not responsible for its contents. We thank Paul Bello, Will Bridewell, Stephanie Sage, and Mohan Sridharan for useful discussions about approaches to abductive explanation.

## References

Anderson, J. R., & Lebiere, C. (1998). *The atomic components of thought*. Mahwah, NJ: Lawrence Erlbaum.

Appelt, D. E., & Pollack, M. E. (1992). Weighted abduction for plan ascription. *User Modeling and User-Adapted Interaction*, *2*, 1–25.

Baral, C. (2003). *Knowledge representation, reasoning and declarative problem solving*. Cambridge, UK: Cambridge University Press.

Blaylock, B., & Allen, J. (2005). Generating artificial corpora for plan recognition. *Proceedings of the Tenth International Conference on User Modeling* (pp. 179–188). Edinburgh: Springer.

Bridewell, W., & Langley, P. (2011). A computational account of everyday abductive inference. *Proceedings of the Thirty-Third Annual Meeting of the Cognitive Science Society*. Boston.

Charniak, E., & Shimony, S. E. (1990). Probabilistic semantics for cost based abduction. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 446–451). Cambridge, MA: AAAI Press.

de Kleer, J. (1986). An assumption-based TMS. *Artificial Intelligence*, *28*, 127–162.

de Kleer, J. (1994). *A hybrid-truth maintenance system* (Technical Report). Xerox Palo Alto Research Center, Palo Alto, CA.

Eckroth, J., & Josephson, J. R. (2014). Anomaly-driven belief revision and noise detection by abductive metareasoning. *Advances in Cognitive Systems*, *3*, 123–142.

Forgy, C. (1982). Rete: A fast algorithm for the many pattern/many object pattern match problem. *Artificial Intelligence*, *19*, 17–37.

Hobbs, J. R., Stickel, M. E., Appelt, D. E., & Martin, P. (1993). Interpretation as abduction. *Artificial Intelligence*, *63*, 69–142.

Klahr, D., Langley, P., & Neches, R. (Eds.) (1987). *Production system models of learning and development*. Cambridge, MA: MIT Press.

Langley, P., & Meadows, B. (2019). Heuristic construction of explanations through associative abduction. *Advances in Cognitive Systems*, *8*, 93–112.

Meadows, B., Langley, P., & Emery, M. (2014). An abductive approach to understanding social interactions. *Advances in Cognitive Systems*, *3*, 87–106.

Miranker, D. P. (1987). TREAT: A better match algorithm for AI production systems. *Proceedings of the Sixth National Conference on Artificial Intelligence* (pp. 42–47). Seattle, WA: Morgan Kaufmann.

Ng, H. T. & Mooney, R. J. (1990). On the role of coherence in abductive explanation. *Proceedings of the Eighth National Conference on Artificial Intelligence* (pp. 337–342). Cambridge, MA: AAAI Press.

Peng, Y., & Reggia, J. A. (1986). Plausibility of diagnostic hypotheses: The nature of simplicity. *Proceedings of the Fifth National Conference on Artificial Intelligence* (pp. 140–147). Philadelphia, PA: Morgan Kaufmann.

Peirce, C. S. (1878). Deduction, induction, and hypothesis. *Popular Science Monthly*, *13*, 470–482.

Young, R. M. (1982). Architecture-directed processing. *Proceedings of the Fourth Annual Conference of the Cognitive Science Society* (pp. 164–166). Ann Arbor, MI: Lawrence Erlbaum.