Proceedings of the Twelfth Annual Conference on Advances in Cognitive Systems ACS-2025 (387-406)

Understanding Written Procedures

Gary Borchardt BORCHARDT@CSAIL.MIT.EDU
Sue Felshin SFELSHIN@CSAIL.MIT.EDU
Boris Katz BORIS@CSAIL.MIT.EDU
Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA 02139 USA

Pat Langley PATRICK.W.LANGLEY@GMAIL.COM
Institute for the Study of Learning and Expertise, Palo Alto, California 94306 USA

Abstract

In this paper, we pose a new problem for AI researchers: to develop systems that understand written instructions for complex procedures. We state the problem, the challenges it poses, and why existing techniques cannot yet solve it. We review prior research on language processing that could be useful, including its limitations, and analyze the computational elements that solutions to the problem must incorporate. Finally, we consider the use of controlled experiments to evaluate systems that learn procedures from instructions and to gain insight into their behavior.

1. Background and Motivation

Throughout its history, artificial intelligence has been concerned with general methods and many of its most important insights have dealt with generic frameworks for representation, reasoning, and problem solving. However, there has also been general agreement that these abilities alone do not suffice and that successful AI systems must incorporate substantial expertise about their particular application domains. This led to recognition of the "knowledge acquisition bottleneck", which concerns providing systems with the expertise they need. Initial responses, associated with the expert systems movement, relied on the manual entry of domain content, but this approach was tedious, expensive, and prone to errors. Tools for interactive knowledge acquisition mitigated these issues, as did high-level formalisms like production systems and logic programming, but the process was still a difficult one. The field of machine learning was launched largely to overcome the bottleneck by acquiring expertise automatically from training data.

Supervised induction methods led to early successes (Langley & Simon, 1995) that have continued to the present, but this approach relies on manual labeling of instances, which is arduous and costly for large data sets. Also, this paradigm is well suited for classification tasks but not for procedural ones, so recent efforts have developed alternative responses. One involves learning control policies from trial-and-error runs in simulators, which has produced self-driving vehicles, game players (Clark & Storkey, 2015), and other impressive artifacts, but it depends on accurate simulators, which are often unavailable. Recent research on interactive task learning (Gluck & Laird,

This work is licensed under a Creative Commons Attribution International 4.0 License.

2018) has explored a middle path that tutors AI systems about procedures in natural modalities like language and sketches, but this process is also tedious and time consuming. We need a more effective approach to overcoming the knowledge acquisition bottleneck in procedural settings.

Fortunately, many activities are already described in manuals and other written documents, so a natural response is to acquire knowledge by reading and understanding them. There has been considerable work on learning by reading, but it has focused on extracting simple facts and concepts from texts, not on mastering procedural content. We desire intelligent systems that can read the manual for any device, extract and store the procedures it describes in a standard format, and later access this knowledge when needed. Expertise acquired in this manner would be useful for robotic automation of physical procedures, assisting human operators on familiar but complex tasks, and training novices on entirely new activities.

In the remaining pages, we report progress on the problem of learning complex procedures from written instructions. We start by specifying the task and noting challenges that require additional research. Next we review some previous approaches to language processing that offer useful ideas, as well as their limitations. After this, we present a theoretical framework for semantic analysis and synthesis that addresses our problem, along with an implemented system, SPROCKET, that embodies its ideas. In addition, we discuss experimental studies of the system's behavior on procedural descriptions from a variety of domains. We close by considering alternative approaches to the problem of learning from written instructions and promising directions for future research.

2. Understanding Instructions and Its Challenges

The research community cannot make progress on a problem without first specifying it clearly. As with other AI problems, we can define the task of instruction understanding in terms of the inputs provided and the outputs generated:

- *Given*: Generic knowledge about a class of domains, such as cooking or electrical maintenance;
- *Given*: Written instructions for a multi-step procedure, say making an omelet or replacing fuses;
- *Find*: A computer-interpretable encoding that can be used later for some performance task.

This is a *learning* problem in that, after processing the inputs, the system has expertise it lacked earlier. It differs from most work on machine learning, which focuses on statistical induction, but we regularly say that people learn from reading instructions, so the label should apply to machines as well. The performance task that uses the learned expertise is another matter we will discuss later.

What do we intend by "written instructions" in the statement above? This refers to a natural language description of some procedure that is recorded in some medium. Instructions often come in the form of written text, but they could also start as an audio recording or even a video that includes subtitles or transcribed commentary. Written recipes for food preparation are familiar examples, but Table 1 provides another complex instance: a *maintenance requirement card* that US Navy personnel use when repairing equipment. These instructions describe a procedure that anyone can carry out on specific devices in the form of written English with clearly separated steps. The learner must acquire procedural expertise from this single description; it cannot extract regularities from thousands or even tens of different examples. Finally, because the instructions are recorded, the learner has no opportunity to ask clarification questions or obtain other information not included.

Table 1. A maintenance requirement card that provides instructions for how to inspect and replace actuator controller fuses. The text is based on US Navy material but does not come from an actual requirement card.

Inspect and Replace Actuator Controller Fuses:

a. Place Service Disconnect in "OFF" position.
b. Open actuator panel door.

WARNING: Consider all electrical leads to be energized until positively proven they are de-energized.

c. Use multimeter to check voltages between each utilized connection in the upper terminal block and the bottom, neutral bus bar.
d. If any wire shows voltage greater than 0 V, then perform no more maintenance. Close panel door, leave Service Disconnect in position, and inform supervisor of findings.
e. Using fuse puller, remove each fuse and inspect for damage or discoloration.

NOTE: Fuse resistance should be checked while fuse is removed from panel.

f. Use multimeter to check resistance of each fuse. Resistance should be less than 0.1 ohm.
g. Reinstall fuses, replacing with new any that failed inspection.
h. Close panel door.
i. Restore Service Disconnect to "ON" position.
j. Report to supervisor any replaced fuses, provide the failed fuses which were replaced.

What do we mean by "complex procedures" in the problem statement? They comprise a set of *partially-ordered actions* that transform one or more initial situations into one that satisfies some *goal*. These situations are *relational* in that they involve configurations of objects or entities, some affected by the actions. Procedures are also *causal* in that they describe the effects of activities under certain conditions, and thus specify how to alter the world to achieve goals. In many cases, they are also *hierarchical* in that they decompose complex activities into simpler ones, with higher levels abstracting away the details. Finally, they are often *conditional* in that they specify ways to achieve goals in different situations. Most of these features are apparent in the sample instructions presented in Table 1.

Why does this class of problems pose a challenge for AI? From the perspective of language processing, the brevity of instructions means they often omit not only steps but also objects that play key roles. Another issue is that understanding procedural descriptions requires the construction of explanations, often causal, that relate actions to goals that may be implicit. Moreover, the knowledge supporting this inference is generic, at the level of common sense, rather than specific to the activity being mastered. Finally, the conditional character of many procedures requires representing and reasoning about hypotheticals, raising another hurdle seldom addressed in language processing.

From the perspective of machine learning, understanding instructions introduces many of the constraints on human-like learning discussed by Langley (2022). These require that, to mimic humans, learning should involve the incremental, piecemeal acquisition of modular, relational elements in the presence of background knowledge. The ability to satisfy these constraints supports rapid learning, in this case the construction of complex procedures from individual text descriptions.

Statistical language processing may still play a supportive role in this process, but only to the extent that it helps an AI system acquire new cognitive structures in a nonstatistical manner. In particular, large language models, while able to offer commentary on the details and variants of common procedures, do not as of yet directly address the task of codifying procedural knowledge in usable form from single descriptions of novel procedures.

3. Previous Research and Its Limits

Natural language processing is a mature field with a long history, so the paucity of work on understanding procedural instructions is concerning. Nevertheless, the literature has explored many promising ideas, so we should review the areas that seem most relevant. These include:

- *Sentence processing*, which analyzes the syntactic structure and meaning content of sentences using knowledge about lexical items, grammar, and semantics (e.g., Katz, 1997; Mooney, 2007; McShane et al., 2018; Kamath & Das, 2019). However, research in this area has emphasized syntactic parsing and even work on semantic methods has focused on individual sentences, whereas we need larger-scale analysis.
- *Discourse processing*, which operates over extended text that it organizes into larger, coherent structures inferred using knowledge about plausible relations among segments (e.g., Grosz & Sidner, 1986; Webber, Egg, & Kordoni, 2012). This appears relevant to comprehending instructional text, but most treatments have used linguistic markers rather than causal knowledge to identify structure, and there has been little concern with procedures.
- *Story understanding*, which interprets the actions and events in a narrative in terms of knowledge about participants' beliefs and goals about the world and others (e.g., Dyer, 1983; Winston & Holmes, 2018). This paradigm also seems relevant to our problem, but it deals with specific events rather than generic procedures, often involves multiple actors with conflicting goals, and analyzes one unfolding sequence rather than hypotheticals.
- *Abductive inference*, which constructs explanations of situations and events in terms of knowledge about the physical world, human agency, and interaction (e.g., Ng & Mooney, 1990; Meadows et al., 2014; Gordon, 2018). This literature has more direct relevance to instruction understanding but, again, it has focused on specific events rather than generic activities. However, its methods for guiding search over explanations may prove useful.
- *Question answering*, which stores content in some accessible format, retrieves elements pertinent to queries, and communicates these results in appropriate ways (e.g., Lehnert, 1978; Katz et al., 2006; Khashabi et al., 2018). This subfield has also developed promising techniques, but it has emphasized answering simple factual questions and it has not typically addressed either causal or hypothetical reasoning.
- *Learning by reading*, which interprets written documents, extracts their content, and stores this information in a format that can be accessed and used later (e.g., Forbus et al., 2007; Mitchell et al., 2018; Friedman et al., 2017). Research on this topic is very similar in spirit to our own, but it has focused on the acquisition of factual content rather than generic procedures, so the application of its techniques will be limited.

There have been some efforts on extracting procedures from text using statistical analysis of corpora (e.g., Kiddon et al., 2015; Park & Nezhad, 2018), but they require thousands of examples to produce only simple structures, so we do not consider them viable responses to the challenge. Instead, we must develop cognitive systems that, as Marcus and Davis (2021) have proposed, exhibit a range of intellectual abilities that are associated with the human mind.

4. A Framework for Semantic Analysis and Synthesis

Langley, Shrobe and Katz (in press) analyze the acquisition of procedures from written instructions as a cognitive task that involves three subtasks: syntactic processing, semantic interpretation, and procedure construction. These subtasks can be accomplished sequentially or they can, to a degree, be accomplished in tandem. For instance, syntactic processing and semantic interpretation, addressed concurrently, can inform each other, and semantic interpretation can be informed by knowledge of previous procedures. Understanding of procedures can be demonstrated through performance tasks such as: procedure execution, training of human performers, execution monitoring, using procedures in planning contexts, and answering questions.

In this section, we present a framework for interpreting procedural descriptions that combines syntactic and semantic processing with support for question answering. Appendix A lists sample procedural descriptions and questions of the sort addressed by this framework. Figure 1 outlines the approach, with processes depicted as blue rectangles, knowledge sets as gray or green rectangles, and background knowledge in green. Arrows indicate flow of data between processes and knowledge sets. Procedural descriptions are submitted to the syntactic analysis process, which relies on two sets of background knowledge, a lexicon and a grammar, to produce parse structures that specify language tokens and expressions identified within the text. Semantic processing occurs next, as two activities—semantic analysis and semantic synthesis—that work together to interpret entities, events, and relationships associated with the procedural description. Three sets of background knowledge support semantic analysis and synthesis:

- A taxonomy of entity types in domains of interest, the principal organizing relation being that one entity type "can function as" another type for purposes of participation in events;
- A set of event models that describe the unfolding of particular types of events in terms of time-varying properties, relationships, and other attributes of participating entities; and
- A set of interpretation rules that suggest candidate event models and related structures as interpretations of language expressions recognized within the parsed text.

Semantic analysis uses the interpretation rules to isolate mentions of entities, events, and relationships within the text, then interprets these quantities to the extent possible based on information in the text and available from previously-formed interpretations. This process may leave some interpretations unfinished. Examples include when an event reference in the text fails to mention one or more participants, when an event reference allows for alternative versions of an event, or when it is unclear if an entity reference refers to a previously established entity in the procedure.

After this, semantic synthesis completes all partial interpretations of entities, events, and relationships by considering them in the context of the evolving scene model, which describes the

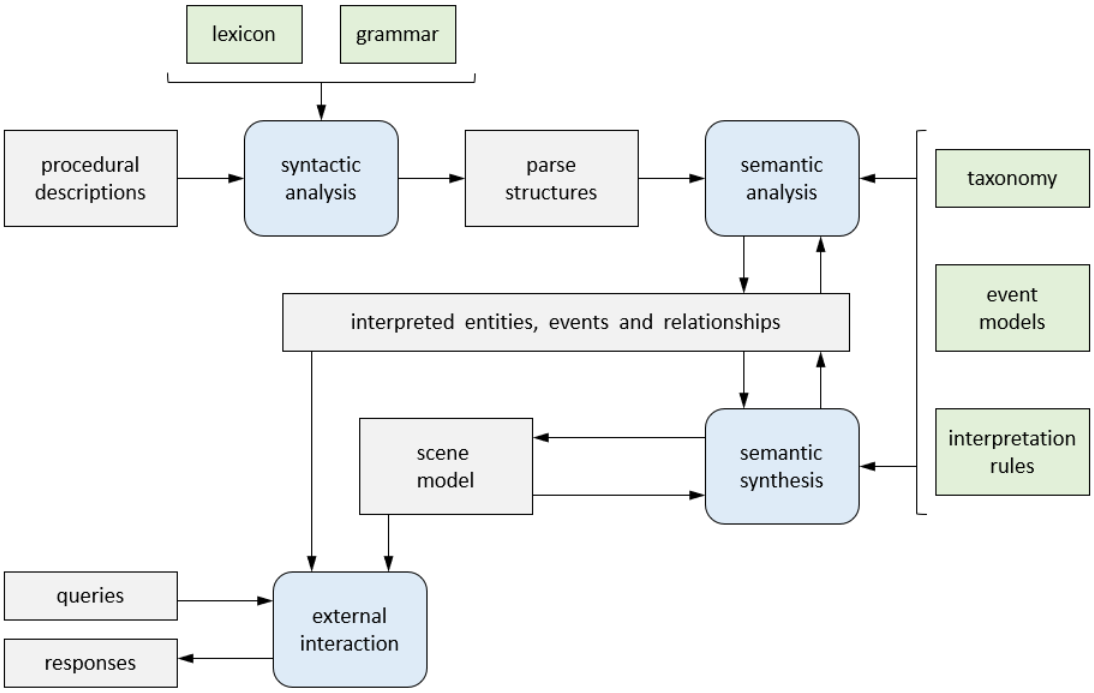

Figure 1. Processes (blue) and knowledge sets (gray or green, with background knowledge in green) involved in the approach to interpretation.

activity from the beginning of the procedure up through the most recent event to be completely interpreted. Once a new event has been fully interpreted, semantic synthesis adds it to the scene model. Semantic synthesis also introduces new entities, events, and relationships where appropriate. The process introduces new entities if they are required by an event model yet are unrecognized among established entities in the scene. It also adds new events that depict ramifications of other events, along with new relationships that hold among events and entities in the growing scene model.

To complete the interpretation of an event, semantic synthesis may need to choose among possible interpretations that specify variants of the event or alternative identities of unmentioned event participants. In this case, it can apply preferences, such as favoring interpretations that involve participant entities active in immediately-preceding events or favoring interpretations with preconditions of events satisfied by persisting effects of preceding ones. After semantic analysis and synthesis have finished, the external interaction process fields and responds to questions about the results of interpretation. The scene model and the constructed interpretations of entities, events, and relationships form a basis for responses to these questions.

Together, these elements and their interactions provide a complete framework for reading written instructions, interpreting and combining their contents into stored procedures, and using them to answer questions about these activities. We have kept the theory intentionally abstract, so it makes no commitments about either internal representations or mechanisms that operate on them,

affording both symbolic and statistical approaches. Nevertheless, it provides strong constraints on possible implementations of the framework (Langley, 2018).

5. The SPROCKET Interpretation System

We have constructed an integrated system, SPROCKET, that instantiates the framework just outlined and serves as a vehicle for demonstrating and evaluating its adequacy. In this section, we describe the operation of its components and how they jointly interpret written descriptions of procedures. Appendix B details the strategies that SPROCKET uses for semantic interpretation.

5.1 Syntactic Processing

The first of SPROCKET's constituents focuses on parsing the input text. This module relies on START (Katz, 1997; Katz, Borchardt, & Felshin, 2006), a natural language system that performs syntactic analysis of text coupled with initial semantic analysis to produce a set of nested ternary expressions. This parsing process is applied to the title of a procedural description and then successively to the sentences that form the body of this description.

An example will illustrate START's operation. This involves two consecutive sentences in a description for replacing cork on a clarinet, extracted from the Web site ifixit.com:

> Cut the new cork to match the width of the tenon joint. Peel off the backside of the cork to reveal the adhesive.

The module parses these sentences individually, generating a set of ternary expressions, some of which include:

```
[cut-11, has_purpose-11, match-12]
[you, cut-11, cork-3]
[you, match-12, width-1]
[cork-3, has_property-12, new]
[width-1, related-to-13, tenon_joint-1]
[peel_off-13, has_purpose-14, reveal-14]
[you, peel_off-13, backside-1]
[you, reveal-14, adhesive-1]
[backside-1, related-to-15, cork-3]
```

These parse structures illustrate how START captures syntactic relationships within each sentence, plus some semantic relationships. For example, has_purpose specifies a relationship between events and related-to denotes a relationship between entities. The system also unifies some entity references across sentences. For instance, cork-3 is used to depict both "the new cork" in the first sentence and "the cork" in the second.

START operates using a lexicon that contains approximately 95,000 English terms, including lemmas (dictionary forms of terms), morphological and derivational inflections of those forms, and alternative parts of speech for those forms. Entries in the lexicon specify terms' parts of speech, their lexical properties, and their relationships to other terms by synonymy and hypernymy. The system's encodings of English grammar let it identify a wide range of language expressions, including ones for noun–modifier relationships, verb–argument relationships, and relationships between clauses. This version of START was able to parse nearly all sentences in the 52 procedural descriptions that we discuss later. However, in rare cases, we had to create new lexicon entries, and in others, we restructured complex sentences to enable parsing.

Figure 2. (a) An event model for "pouring off", from SPROCKET's background knowledge. (b) The scene model, including the completed interpretation of a "pouring off" event.

5.2 Semantic Analysis

SPROCKET's second module addresses semantic analysis, semantic synthesis, and external interaction. This builds on IMPACT, a system for describing and reasoning about events (Borchardt et al., 2014; Borchardt, 2015). Semantic processing relies on use of a taxonomy of entity types, a collection of event models, and a set of interpretation rules. IMPACT renders knowledge elements using an extension of the Moebius encoding of parsed language (Borchardt, 2014), and it describes the contents of event models using the Transition Space representation (Borchardt, 1994, 2014, 2015; Borchardt et al., 2014), which has roots in Waltz's (1982) Event Shape Diagram representation.

Figure 2 (a) presents an event model formulated using Transition Space and the Moebius encoding. This event model depicts an event of pouring off a liquid quantity from a solid quantity. Angle brackets distinguish variables in Transition Space descriptions (e.g., <'liquid_quantity_component 1'>). Appendix C provides additional examples and details of the representation.

Semantic processing proceeds by selecting, instantiating, transforming, matching, and combining event models. This focuses on one sentence at a time, beginning with application of interpre-

tation rules, which isolate and operate on instances of syntactic constructions in the parsed text: nouns; nouns modified by participles or adjectives; clauses; and relationships indicated through the use of connectives, prepositional phrases and quantifiers. As the interpretation rules fire, they trigger semantic analysis and semantic synthesis for the elements—entities, events, and relationships—that were isolated by the rule firings.

In the course of its interpretation of a procedural description, SPROCKET constructs a variety of temporary knowledge elements that serve as partial or complete interpretations of entities, events, and relationships. In particular, semantic analysis produces partial interpretations of events in those cases where not all participants are mentioned in the text. For example, in a procedural description for making "sausage stuffed jalapenos" from the Web site allrecipes.com, semantic analysis forms an initial interpretation for the event in the sentence

Drain grease.

by creating a partial instantiation of the event model for "pouring off" presented in Figure 2 (a), in which the variables <'human 1'> and <'liquid_quantity_component 1'> have been replaced with constants 'you' and 'grease 001', respectively, and the variables <'solid_quantity_component 1'>, <'time 1'> and <'time 2'> are retained pending continued processing.

5.3 Semantic Synthesis

The next stage, semantic synthesis, considers alternative, complete instantiations of this event model, each of which replaces remaining variables with constants that refer to typed entities available in the scene model and introduces time points that position the event at the model's current end. SPROCKET then compares each generated interpretation of the event to the scene model and applies heuristics to select a preferred interpretation. For example, the selected candidate might replace <'solid_quantity_component 1'> with an entity recently active in the scene, 'sausage 001', in addition to specifying the start and end times of the event as 'time 006' and 'time 007'. The module adds this interpretation to the scene model, creating the version depicted in Figure 2 (b).

We can view semantic synthesis as carrying out greedy heuristic search through a space of possible event interpretations and resulting scene interpretations. Given multiple candidate interpretations, the system considers five factors, in order, until one candidate emerges as the best choice:

- Fewer new entities are introduced into the scene at the start of the event;
- Fewer existing but recently inactive entities are involved in the event;
- Fewer new entities are created in the course of the event's unfolding;
- Entities in the event were involved in more recent activities; and
- More preconditions of the event are fulfilled by a preceding activity.

In practice, these heuristics have worked well for SPROCKET, without need for backtracking. However, the system could benefit from additional heuristics or more sophisticated search strategies.

SPROCKET also maintains a number of event models that specify expected context-dependent ramifications of other events. For example, when an entity changes position, it may no longer be "at", "on", or "in" another entity like a pan. Similarly, when an entity comes to be "in" a container, the module may predict it will combine with other objects already in it. Following each addition of

Table 2. Queries supported by the SPROCKET system.

What is the specification of this procedure? What events are involved in this procedure? What entities are involved in this procedure?	What happens to all entities during this procedure? What happens during this event? What happens to this thing during the procedure? What is asserted in the scene at this time point?
Which things are involved in this event? How are other things related to this event? What is the statement of this event? Which procedure sentence is this event associated with?	Which events is this procedure sentence associated with? Which things are mentioned in this expression? Which terms is this procedure sentence associated with?
Which events involve this thing? How are other things related to this thing? Which procedure terms is this thing associated with?	Which thing is this procedure term associated with? Which procedure sentences is this term associated with?

a fully interpreted event to the scene model, the system considers all matches of these ramification event models to the current ending portion of the scene model and, on the basis of these matches, conditionally includes the additional assertions.

5.4 Question Answering

After SPROCKET has interpreted all the sentences in a procedural description, the human user can inspect and assess its interpretation through a graphical/menu-based query facility. The system constructs responses by referring to content generated during earlier processing: the scene model; interpretations of entities, events and relationships; the input text; and the parse structures produced by START. Table 2 lists the different queries that this final module supports.

Using this facility, a SPROCKET user can assess its interpretations with respect to broader questions such as those listed in Appendix A: "Which entities are distinct from one another?", "Which entities participate in which events?", "Which versions of particular events occur?", "What relationships exist between pairs of events and entities?", "What events occur as ramifications of other events?", and "What changes do the various entities undergo?". For example, after processing the sentence "Drain grease" above, asking about associated events will reveal the unmentioned participant to be the sausage involved in an earlier step.

6. Experimental Results

To test the viability of our approach, we evaluated SPROCKET's performance in interpreting procedural descriptions and the usefulness of its knowledge elements. We applied the system to 52 procedural descriptions extracted from the Web sites allrecipes.com, eatingwell.com, and ifixit.com, as well as documents supplied by the U.S. Navy. These descriptions concerned a variety of topics like meal preparation, mechanical assembly and disassembly, cleaning, adhesives, fabrics, electrical systems, lubrication, fluids, and dyes. Each new description required addition of some knowledge, producing a total of roughly 600 taxonomy entries, 200 event models, and 600 interpretation rules.

To evaluate SPROCKET's performance, we first formed expectations for the interpretation of specific elements of interpretation—entities, events, and relationships—associated with each proce-

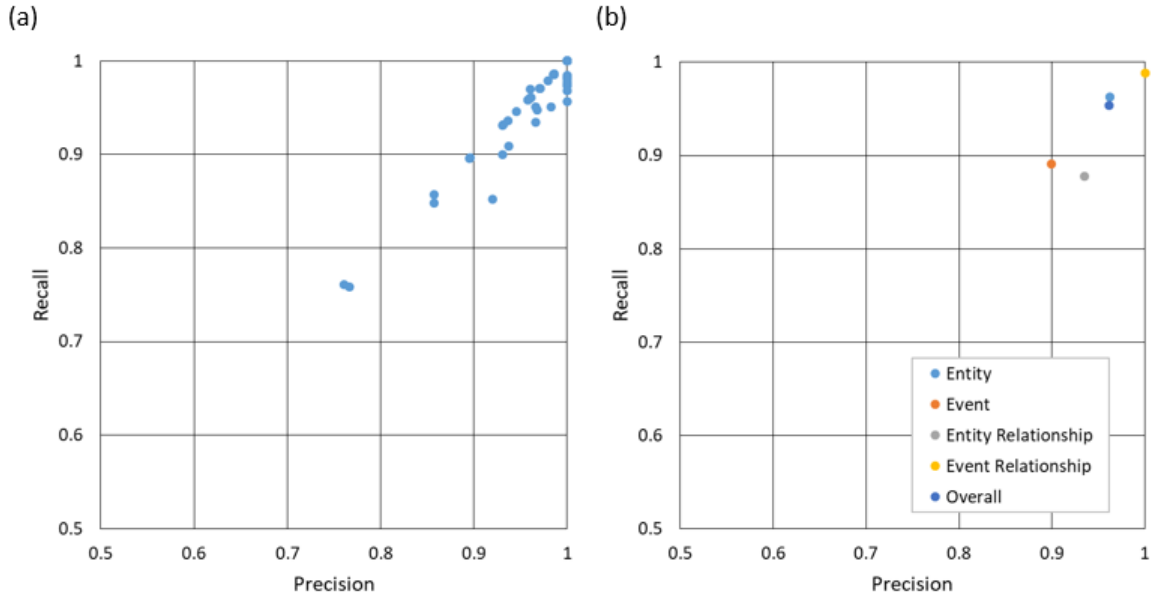

Figure 3. Relative precision and recall for (a) interpretation of 52 individual procedural descriptions, and (b) collectively, by type of interpretation element.

dural description. The number of expectations varied considerably among the 52 tested procedural descriptions, from 24 expected elements of interpretation for one description, to 172 for another. We then compared our expectations with the interpretations that SPROCKET generated. We judged each element of interpretation produced by the system as being acceptable or unacceptable. In addition, we identified any omitted—or in rare cases added—elements of interpretation relative to our expectations. These assessments let us calculate a variant of precision and recall that gauged the system's interpretations relative to these expectations.

Figure 3 (a) presents the precision and recall scores relative to expectations for individual procedural descriptions, whereas Figure 3 (b) shows the scores for specific types of interpretive elements. In Figure 3 (a), all procedural descriptions attain relative precision and recall scores of at least 0.75, and typically 0.85 or better, suggesting that SPROCKET produces acceptable interpretations of the 52 procedural descriptions when given appropriate knowledge elements.

In both diagrams, precision and recall scores fall mainly along and below the diagonal. This is because errors of commission negatively affect both precision and recall, whereas errors of omission negatively affect only recall. Acceptable interpretations produced by the system include the examples described in Appendix A. For example, when interpreting a procedure for replacing oil in a compressor, from the expression "no more oil comes out", the system correctly chooses an event version and unmentioned participant such that "no more oil comes out of the reservoir". On inspection, the two least-well interpreted procedural descriptions in Figure 3 (a) suffered from errors in the attachment of prepositional phrases and references to an entity involved in many events. In Figure 3 (b), the relatively lower recall score for entity relationship interpretations is due to diffi-

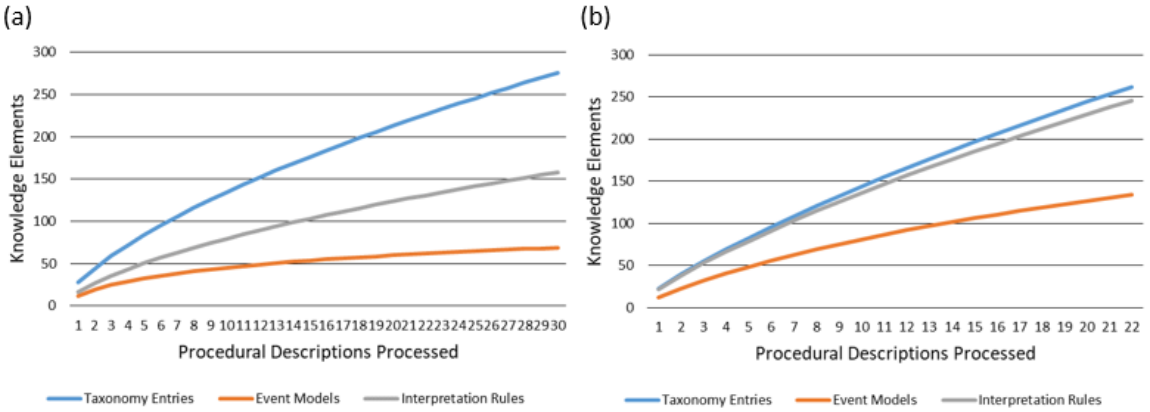

Figure 4. Number of knowledge elements SPROCKET required to correctly interpret (a) 30 procedural descriptions for cooking and (b) 22 descriptions for maintenance and repair, with separate curves for taxonomy elements, event models, and interpretation rules. Each curve reflects results for 10,000 presentation orders.

culties in interpreting unassociated abstract quantities, as when mention of "the two edges" appears without its related quantity (e.g., "the two edges of the spring").

We also tested the benefits of knowledge provided to SPROCKET. Our hypothesis is that as the system is exposed to more and more procedural descriptions, and corresponding knowledge is added, the system's requirements for further additions of knowledge will decrease relative to its requirements for descriptions processed earlier. To test this, we began by identifying the specific knowledge elements—taxonomy entries, event models, and interpretation rules—needed to interpret each of the 52 procedural descriptions processed by SPROCKET. We then used this information to calculate, for a particular ordering of procedural descriptions submitted to the system, when each needed knowledge element would have to be supplied to the system—perhaps earlier if used during the interpretation of a procedural description early in the ordering, or perhaps much later if that particular knowledge element was not used during the interpretation of any procedural description before some later description in the ordering.

For the 30 cooking-related descriptions that SPROCKET processed, and separately for the 22 maintenance and repair descriptions it handled, we considered 10,000 randomly ordered presentations of each set's descriptions, recording in each case the specific progression in knowledge that would be required to process the descriptions in that that order of presentation. Figure 4 (a) shows the growing knowledge needs of the system, averaged over the runs, as it encounters the 30 cooking-oriented procedural descriptions, whereas Figure 4 (b) depicts the analogous numbers for the 22 maintenance and repair descriptions.

For both sets of procedural descriptions and across all three knowledge types—taxonomy entries, event models, and interpretation rules—the slope of each curve decreases monotonically, indicating that the need for further additions of knowledge indeed diminishes as SPROCKET processes successive descriptions and accumulates knowledge. The most pronounced reduction in need for new knowledge was associated with event models for the 30 cooking recipes. This makes sense, as there are arguably fewer types of common events that arise in cooking contexts than in general

maintenance and repair. The number of interpretation rules grows more rapidly than event models, which also makes sense, because multiple language expressions can describe the same event type. Finally, entries in the taxonomy rise the fastest, presumably due to the many language terms needed. Mitigating this is the fact that augmenting the taxonomy is a relatively easy task.

A separate question related to usefulness of knowledge elements is whether their addition can interfere with the system's performance by undermining the system's ability to interpret previously encountered procedural descriptions. In the course of SPROCKET's processing of 52 descriptions, each accompanied by appropriate knowledge elements, we have not observed interference of this sort. Interpretations supported previously by the system have not been adversely affected by either the addition of new knowledge elements or by the occasional refinement of ones entered previously.

7. Related Research

Our research on learning procedures from written instructions has drawn on ideas from the earlier work discussed in Section 3, but we should also examine more recent efforts that address similar tasks. For instance, research in computational linguistics and language understanding has produced strategies for coreference resolution, ellipsis processing, and word sense disambiguation in text. These techniques draw on various forms of knowledge. In particular, McShane and Nirenburg (2021) describe the use of an ontology in the early stages of semantic interpretation and domain-specific knowledge in later ones. Similarly, our event models describe commonly occurring events that can be stored in an ontology applied in the early stages of semantic interpretation.

Our approach introduced this content manually, but other recent work has explored its automated extraction. For example, Miglani and Yorke-Smith (2020) report a system that acquires knowledge like that in our event models, whereas Bosselut et al. (2018) and Dalvi et al. (2019) examine its use in forming interpretations of entities, events, and inter-event relationships. Our approach complements their work by exploring the broad contributions of knowledge about events in service of semantically interpreting procedural descriptions.

Recently, Cohen and Mooney (2023) have leveraged large language models to identify unmentioned entities and actions in textual descriptions of familiar procedures. More generally, we might expect that large language models could perform other component tasks within the interpretation process, particularly in the semantic analysis stage of our framework—tasks such as suggesting temporal ordering of events from information in an input text, recognizing explicit relationships specified between pairs of entities and events, and suggesting paraphrases of events in service of advancing possible interpretations of these events. Separately, our own investigations suggest that large language models can serve as a source of background knowledge about states and changes that occur during common events—knowledge useful for constructing a library of event models. In addition, we might expect that large language models can serve as a source of background knowledge for building out a taxonomy and set of interpretation rules as used within our framework.

These initial results herald an opportunity for developing hybrid interpretation systems that employ both statistical and symbolic mechanisms, as explored by McShane, Nirenburg, and English (2024). Hybrid systems for interpreting procedural descriptions may benefit from both the wide coverage of large language models and the enhanced precision of symbolic techniques like those described here.

8. Concluding Remarks

The research reported in this paper offers four main contributions. First, it specifies the problem of understanding procedural descriptions in which a system inputs textual descriptions and produces expertise that supports question answering and other performance tasks. Second, it casts semantic interpretation of procedural descriptions as a two-stage process. For each sentence, analysis first isolates units of meaning and forms partial interpretations of entities, events, and relationships, after which synthesis extends and selects among these interpretations as it constructs a model of the unfolding scene. Semantic analysis and synthesis operate on three collections of general knowledge: a taxonomy of entity types, a set of event models, and a set of interpretation rules. Third, the research provides SPROCKET, an implemented system that embodies this approach to understanding procedural descriptions. Finally, it evaluates SPROCKET's performance mechanisms and its knowledge structures, showing that, together, the approach to procedural understanding works as intended.

Our ongoing research focuses on extensions and refinements to the approach and its implementation. We are creating an explanation facility to provide language-formulated justifications for system interpretations based on the knowledge elements used and the operations performed on them. We are also investigating strategies for augmenting the background knowledge that supports semantic interpretation—taxonomy entries, event models and interpretation rules—as well as additional channels of descriptive information about individual procedures to be learned—for example, specifications of initial conditions; descriptions of available parts, materials and tools; language summaries of observed demonstrations; and associated diagrams, images or video.

Given multiple channels of information regarding a procedure to be learned, an extended interpretation system will frequently need to reconcile competing accounts of an activity. We expect that our techniques for semantic synthesis will help reconcile these accounts, along with related techniques we have developed for recognizing events from observations (Borchardt et al., 2014; Borchardt, 2015). More generally, we predict that an initial understanding of a procedure, formed by interpreting a written description, can serve as the basis for assessing and integrating additional information into a unified, expanded comprehension of the activity, making our work to date a springboard for more advanced abilities.

Acknowledgements

This analysis was supported by Grant N00014-20-1-2643 from the Office of Naval Research, which is not responsible for its contents. We thank Dylan Holmes, Howard Shrobe, and Patrick Winston for discussions that led to many of the ideas presented here.

References

Borchardt, G. C. (1994). *Thinking between the lines: Computers and the comprehension of causal descriptions*. Cambridge, MA: MIT Press.

Borchardt, G. C. (2014). *Moebius language reference, Version 1.2* (Technical Report MIT-CSAIL-TR-2014-005). Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

Borchardt, G. C. (2015). *A suite of techniques for describing activity in terms of events* (Technical Report MIT-CSAIL-TR-2015-009). Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

Borchardt, G., Katz, B., Nguyen, H.-L., Felshin, S., Senne, K., & Wang, A. (2014). *An analyst's assistant for the interpretation of vehicle track data* (Technical Report MIT-CSAIL-TR-2014-022). Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

Bosselut, A., Levy, O., Holtzman, A., Ennis, C., Fox, D., & Choi, Y. (2018). Simulating action dynamics with neural process networks. *Proceedings of the Sixth International Conference for Learning Representations*. Vancouver, BC.

Clark, C., & Storkey, A. (2015). Training deep convolutional neural networks to play Go. *Proceedings of the Thirty-Second International Conference on Machine Learning* (pp. 1766–1774). Lille, France.

Cohen, V., & Mooney, R. (2023). Using planning to improve semantic parsing of instructional texts. *Proceedings of the First Workshop on Natural Language Reasoning and Structured Explanations* (pp. 47–58). Toronto, ON: Association for Computational Linguistics.

Dalvi, B., Tandon, N., Bosselut, A., Yih, W., & Clark, P. (2019). Everything happens for a reason: Discovering the purpose of actions in procedural text. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing* (pp. 4496–4505). Hong Kong: Association for Computational Linguistics.

Dyer, M. G. (1983). *In-depth understanding: A computer model of integrated processing for narrative comprehension*. Cambridge, MA: MIT Press.

Forbus, K., Riesbeck, C., Birnbaum, L., Livingston, K., et al. (2007). Integrating natural language, knowledge representation and reasoning, and analogical processing to learn by reading. *Proceedings of the Twenty-Second Conference on Artificial Intelligence* (pp. 1542–1547). Vancouver, BC: AAAI Press.

Friedman, S., Burstein, M., McDonald, D., Plotnick, A., Bobrow, R., Cochran, B., & Pustejovsky, J. (2017). Learning by reading: Extending and localizing against a model. *Advances in Cognitive Systems*, *5*, 77–96.

Gluck, K. A., & Laird, J. E. (Eds.) (2018). *Interactive task learning*. Cambridge, MA: MIT Press.

Gordon, A. (2018). Interpretation of the Heider-Simmel film using incremental Etcetera Abduction. *Advances in Cognitive Systems*, *7*, 23–38.

Green, A. (2025). What is Planning Domain Definition Language (PDDL)? In Planning.wiki - The AI Planning & PDDL Wiki. Retrieved Sep 27, 2025, from http://planning.wiki/guide/whatis/pddl

Grosz, B. J., & Sidner, C. L. (1986). Attention, intentions, and the structure of discourse. *Computational Linguistics*, *12*, 175–204.

Kamath, A., & Das, R. (2019). A survey on semantic parsing. *Proceedings of the First Conference on Automated Knowledge Base Construction*. Amherst, MA.

Katz, B. (1997). Annotating the World Wide Web using natural language. *Proceedings of the Fifth RIAO Conference on Computer Assisted Information Searching on the Internet* (pp. 136–155). Montreal, QC.

Katz, B., Borchardt, G., & Felshin, S. (2006). Natural language annotations for question answering. *Proceedings of the Nineteenth International FLAIRS Conference* (pp. 303–306). Melbourne Beach, FL.

Khashabi, D., Khot, T., Sabharwal, A., & Roth, D. (2018). Question answering as global reasoning over semantic abstractions. *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence* (pp. 1905–1914). New Orleans: AAAI Press.

Kiddon, C., Ponnuraj, G. T., Zettlemoyer, L., & Choi, Y. (2015) Mise en Place: Unsupervised interpretation of instructional recipes. *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing* (pp. 982–992). Lisbon, Portugal: ACL.

Langley, P. (2018). Theories and models in cognitive systems research. *Advances in Cognitive Systems*, *5*, 3–16.

Langley, P. (2022). The computational gauntlet of human-like learning. *Proceedings of the Thirty-Sixth AAAI Conference on Artificial Intelligence* (pp. 12268–12273). Vancouver, BC: AAAI Press.

Langley, P., Shrobe, H. E., & Katz, B. (in press). A cognitive task analysis of rapid procedure acquisition from instructional documents. *Advances in Cognitive Systems*.

Langley, P., & Simon, H. A. (November, 1995). Applications of machine learning and rule induction. *Communications of the ACM*, *38*, 55–64.

Lehnert, W. G. (1978). *The process of question answering*. Hillsdale, NJ: Lawrence Erlbaum.

Marcus, G., & Davis, E. (2021). Insights for AI from the human mind. *Communications of the ACM*, *64*, 38–41.

McShane, M., Nirenburg, S., & English, J. (2018). Multi-stage language understanding and actionability. *Advances in Cognitive Systems*, *6*, 119–138.

McShane, M., & Nirenburg, S. (2021). *Linguistics for the age of AI*. Cambridge, MA: MIT Press.

McShane, M., Nirenburg, S., & English, J. (2024). *Agents in the long game of AI: Computational cognitive modeling for trustworthy, hybrid AI*. Cambridge, MA: MIT Press.

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

Miglani, S., & Yorke-Smith, N. (2020). NLtoPDDL: One-shot learning of PDDL models from natural language process manuals. *Proceedings of the ICAPS Workshop on Knowledge Engineering for Planning and Scheduling*.

Mitchell, T., Cohen, W., Hruschka, E., Talukdar, P., et al. (2018). Never-ending learning. *Communications of the ACM*, *61*, 103–115.

Mooney, R. J. (2007). Learning for semantic parsing. *Proceedings of the Eighth International Conference on Intelligent Text Processing and Computational Linguistics* (pp. 311–324). Mexico City: Springer.

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.

Park, H., & Nezhad, H. R. M. (2018). Learning procedures from text: Codifying how-to procedures in deep neural networks. *Proceedings of the 2018 World Wide Web Conference* (pp. 351–358). Lyon, France.

Waltz, D. L. (1982). Event shape diagrams. *Proceedings of the Second National Conference on Artificial Intelligence* (pp. 84–87). Pittsburgh, PA: AAAI Press.

Webber, B., Egg, M., Kordoni, V. (2012). Discourse structure and language technology. *Natural Language Engineering*, *18*, 437–490.

Winston, P. H. & Holmes, D. (2018). *The Genesis enterprise: Taking artificial intelligence to another level via a computational account of human story understanding* (CMHI Report No. 1). Computer Science and Artificial Intelligence Laboratory, Massachusetts Institute of Technology, Cambridge, MA.

Appendix A: Sample Descriptions and Questions

Table 3 lists two written procedural descriptions extracted from public Web sites, both modified slighly for ease of software processing, but retaining their original content and character. These examples illustrate six ways in which procedural descriptions can require their reader to fill in details, plus six corresponding lines of questioning that can be posed to an understanding system:

- *Which entities are distinct from one another?* An entity may be referenced in multiple ways within a procedural description. In the quiche example in Table 3, "pie plate" and "pie pan" refer to the same entity. In a separate description, "cold water" is later referred to as "hot water", following changes to the entity. Also, a description may reference a previously unmentioned entity, as when several quantities are combined, then referred to as "rice salad".
- *Which entities participate in which events?* Some entities will be explicitly mentioned as participants within events, while others may not. For instance, in the compressor example in Table 3, the filler cap and drainage plug are to be unscrewed *from the compressor*, and no more oil is to come out *from the reservoir*. In the quiche example, the instruction "Drain, crumble and set aside." describes actions to be performed on the bacon.
- *Which versions of particular events occur?* In some cases, separated from the context, descriptions of actions can suggest multiple possible variants of events. For example, in the compressor example, no more oil may be coming out of a container or no more oil may be leaving a region of space. Often, as in this case, the need to determine which version of an event is occurring arises from omitted mention of one or more participants in the event.
- *What relationships exist between pairs of events and entities?* Some relationships may be explicitly stated in the text, using connectives (e.g., "until", "if" and "while"), quantifiers (e.g., "each" and "any"), and prepositions (e.g., "of"). Other relationships may be implicit; for instance, in the compressor example, unscrewing the drainage plug enables—sets up preconditions for—the later event of screwing on the drainage plug.
- *What events occur as ramifications of other events?* Immediate ramifications may go unmentioned, as in the quiche example, when the bottom of the pie plate is lined with cheese and bacon, which then combine. Later, when the liquid mixture is poured into the pie pan, this combines with the cheese and bacon. Also, when the quiche is served, the quiche and each of its ingredients come out of the pie pan.

Table 3. Example procedural descriptions from (a) ifixit.com and (b) allrecipes.com, slightly modified.

(a)	(b)
Replace oil in a compressor. Unscrew the filler cap to allow air into the reservoir. Using the hex key, unscrew the drainage plug. With the plug removed, do this. Let the reservoir drain until no more oil comes out. Screw the drainage plug on. Add the new oil through the filler hole. Make sure that the oil level on the indicator is between the lines.	Make a quick quiche. Place bacon in a large, deep skillet. Cook over medium-high heat until the bacon is evenly brown. Drain, crumble and set aside. Preheat oven to 350 degrees F. Lightly grease a large pie pan. Line the bottom of the pie plate with cheese and the crumbled bacon. Combine eggs, the butter, onion, salt, flour and milk. Whisk together until the combination is smooth. Pour into pie pan. Bake in preheated oven for 35 minutes, until the mixture is set. Serve immediately or cool and serve.

- *What changes do the various entities undergo?* Given even a very general understanding of the events that occur during a procedure, the reader should be able to identify changes that apply to particular participating entities. For instance, in the quiche example, the bacon comes to be in the skillet, then it becomes hot, crumbled, not in the skillet, in the pie pan, part of a combination with the cheese, not in the pie pan, and so forth.

Appendix B: SPROCKET's Semantic Interpretation Strategies

The upper portion of Figure 5 shows strategies SPROCKET uses for semantic analysis. For each sentence processed, the system first conducts semantic analysis for entities, then relationships, then events, which reflects dependencies between these types of element. For instance, when processing a food recipe, it might identify "pie pan" as the same as a previous reference to "large pie pan". After this, it might use information about temporal ordering to focus consideration on a particular, mentioned event (e.g., an instance of "filling") and assemble details about this event from information provided in the textual description. These steps produce event models that serve as candidate clause interpretations, possibly with some participants left unspecified. The lower portion of Figure 5 itemizes SPROCKET's strategies for semantic synthesis. As with semantic analysis, this module first focuses on entities, then relationships, and then events in order to comply with dependencies that exist between these types of element. Three examples will illustrate some of these strategies:

- In one case, SPROCKET resolves a reference "hot water" to an entity associated with an earlier reference "cold water" involved in an event that subsequently made it "hot" in the scene model.
- In another, the system proposes identities for an unmentioned participant using known entities and selects an interpretation based on how well its elaborated specifications fit known activities.
- In a third scenario, it associates an abstract quantity that was referenced simply as "an amount" with an entity, "drywall compound", that was recently active in the scene model.

Other strategies employed within SPROCKET operate in a similar manner. Semantic analysis acts first, but in some cases will produce only partial interpretations of elements in the input description.

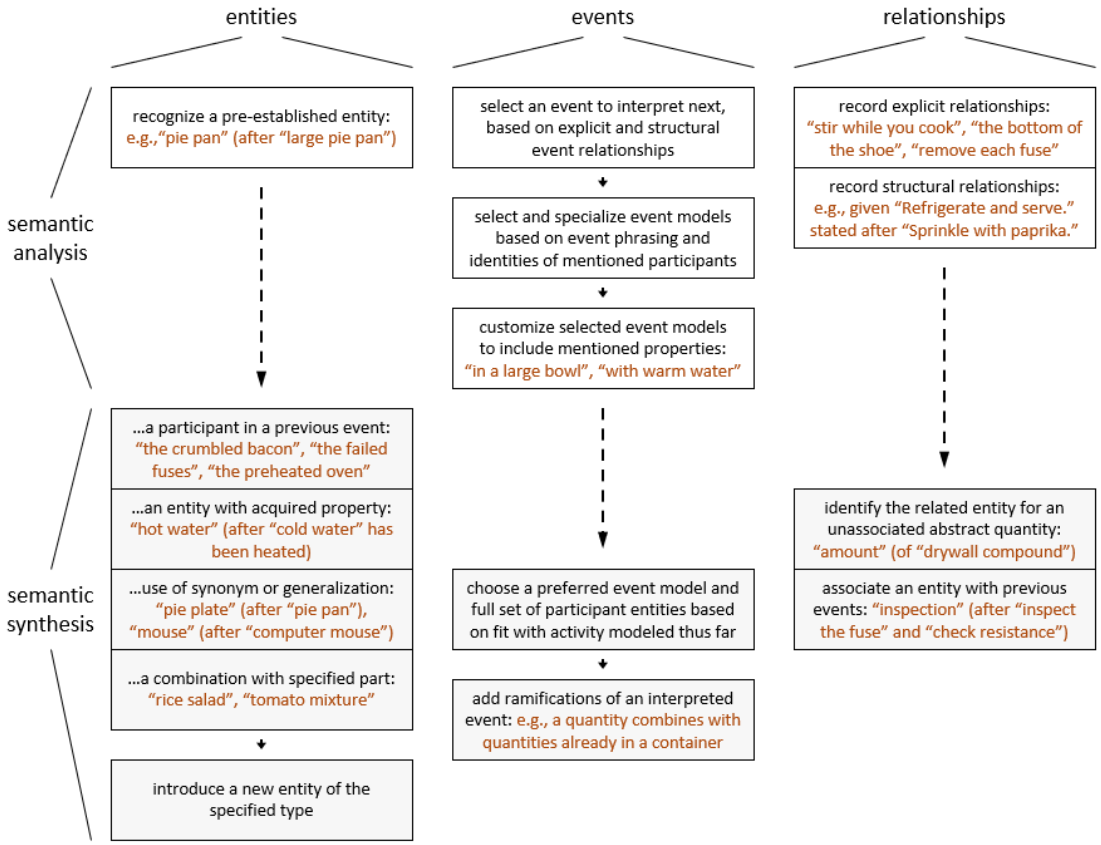

Figure 5. Strategies for semantic analysis and synthesis used by the SPROCKET interpretation system.

Semantic synthesis steps in when necessary to produce complete interpretations of elements by building on relations already established in the evolving scene model.

Appendix C: Knowledge Elements Utilized in Semantic Interpretation

Figure 6 presents three examples of knowledge elements used within SPROCKET in support of semantic analysis and synthesis. These knowledge elements are formulated using an extension of the Moebius encoding (Borchardt, 2014). Moebius specifies variables in enclosing angle brackets and language expressions using parenthesized constructions that optionally tag their elements with specifications of syntactic relations and semantic categories.

Figure 6 (a) depicts a taxonomy entry for the type bottle. Taxonomy entries form a hierarchy such that each type "can function as" each of its supertypes for purposes of participation in events. The entry for bottle lists tool and container as supertypes. Figure 6 (b) depicts an event model concerning placement of an entity in a container. SPROCKET's event models specify a statement of the event, plus a set of assertions that describe states and changes in language-motivated attributes of entities participating in the event; these assertions conform to the design of the Transition Space

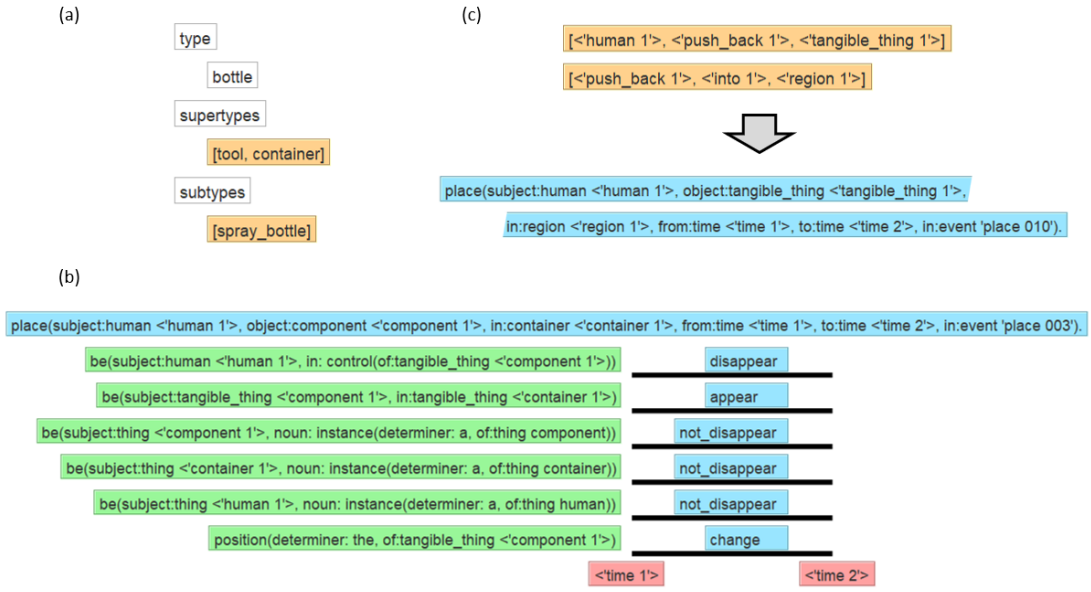

Figure 6. Examples of (a) a taxonomy entry, (b) an event model, and (c) an interpretation rule (each slightly simplified), as used within SPROCKET for semantic analysis and synthesis.

representation (Borchardt, 1994, 2014, 2015; Borchardt et al., 2014), summarized below. Figure 6 (c) depicts an interpretation rule for language expressions that involve the construction "push back". Interpretation rules match patterns of parsed language—for SPROCKET, patterns of ternary expressions produced by the START system parser—and on the basis of these matches propose candidate event models to serve as possible interpretations of the matched language expressions.

The Transition Space representation encodes event descriptions in various stages of processing during semantic interpretation, from background event models that specify their event participants and time points as variables, to partially- and fully-instantiated event models produced during semantic analysis and synthesis, to the scene model formed by combining fully-instantiated event models produced during semantic synthesis. Each event model specifies activity in terms of states and changes in boolean, qualitative and quantitative attributes of one or two entities, and attributes are encoded as simple language expressions. Concurrent states and changes combine to form transitions—specifications of activity over single intervals of time—and transitions combine to form event descriptions, which can be viewed as "paths" in a space of all possible transitions.

Transition Space is similar in expressiveness to PDDL and its successors (e.g., as described in Green, 2025); however, its usage is primarily for matching. In SPROCKET, Transition Space portrays typical occurrences of events, which may or may not include all and only their necessary conditions and effects. Also, the representation is constrained in ways that enhance its support for matching: attributes are chosen from the relatively small set of properties, functions and relationships between entities that can be expressed simply in language, and states and changes in attributes are such that they can be decomposed into a small set of primitive, pairwise comparisons, where one quantity is asserted to "equal", "not equal", "exceed" or "not exceed" another quantity.