Computer Science Area Paper

Schematic Knowledge Representations for Story Understanding

Lane Lawley

Supervisor
Dr. Lenhart Schubert

April 2019

University of Rochester
Department of Computer Science

Abstract

The goal of mechanizing the understanding of natural language stories is long-standing, owing to the difficulty of achieving it: to fully "understand" natural stories, a reader must infer many unstated facts, entities, and motivations. Because these inferences require an immense amount of "common sense" knowledge about the world—an amount too large for researchers hand-craft *a priori*—we must look to automatic knowledge acquisition techniques. Much contemporary work on knowledge acquisition leans heavily on machine learning techniques, which require a huge amount of data, and yet generally fail to derive coherent knowledge representations or robust reasoning techniques. Human children, however—some of the world's biggest fans of stories—seem to learn highly general patterns of events not from thousands or millions of concrete examples, as many machine learning systems require, but often with only one or two.

To help explain how children accomplish this, and how we might mechanize those processes, this paper will focus primarily on *schemas*: structured, symbolic representations of stereotypical patterns of events. In this paper, I hope to demonstrate the salience and applicability of schemas in story understanding by outlining some notable schema theories, discussing several past approaches to the problem of generating schemas from text corpora, and introducing my group's own approach to that problem.

Contents

1 Introduction **4**

2 Schema Theories and Models **6**

2.1 "Strategies of Discourse Comprehension" . . . 7

2.1.1 The Gist of the Model . . . 7

2.1.2 Assumptions . . . 7

2.1.3 The Model . . . 9

2.1.4 Discussion . . . 11

2.2 Minsky's Frames . . . 11

2.2.1 Frame language implementations . . . 12

2.2.2 Matching . . . 17

2.2.3 Frame systems . . . 18

2.3 Schank's Scripts . . . 19

2.3.1 Conceptual Dependency (CD) Theory . . . 19

2.3.2 Causal chains . . . 22

2.3.3 Scripts . . . 23

2.3.4 Discussion . . . 25

2.4 FrameNet . . . 26

2.4.1 FrameNet frames . . . 27

2.4.2 Applications . . . 28

2.4.3 Discussion . . . 29

3 Past Approaches to Schema Learning **30**

3.1 Classical Schema Learning . . . 31

3.1.1 The Integrated Partial Parser (IPP) . . . 31

3.1.2 GENESIS . . . 35

3.2 Statistical Schema Learning . . . 42

3.2.1 Unsupervised Learning of Narrative Event Chains . . . 42

3.2.2 Learning statistical scripts with LSTM recurrent neural networks 45

4 Towards Scalable Schema Learning 51
4.1 What's needed for schema learning? . . . 51
4.1.1 Symbolic representations . . . 52
4.1.2 Slot relationships . . . 53
4.1.3 Branching, conditionals, and variables . . . 53
4.1.4 Time, probability, and meta-reasoning . . . 53
4.1.5 Analog representations . . . 54
4.1.6 Behavioral axioms . . . 54
4.1.7 Primitive actions . . . 54
4.1.8 Language learning . . . 54
4.2 Implementing what's needed . . . 55
4.3 Episodic Logic (EL) . . . 55
4.3.1 Why EL? . . . 55
4.3.2 What is EL? . . . 56
4.3.3 Explanation of an example . . . 57
4.3.4 Discussion . . . 59
4.4 What we talk about when we talk about "protoschemas"... . . . 60
4.4.1 Explanation of an example protoschema . . . 62
4.5 ...and why we're talking about protoschemas in the first place. . . . 66
4.6 Integrating machine learning techniques . . . 68
4.6.1 Word representations . . . 68
4.6.2 Identifying good candidates for generalization . . . 69
4.7 Conclusion . . . 69

A An Example Schankian Script 80

B Workshop paper excerpt: matching, inference, & future work 82
B.1 Matching Algorithm . . . 82
B.2 Planning and Meta-Reasoning . . . 83
B.3 Matching Example . . . 84
B.3.1 ULF Processing . . . 85
B.3.2 The Initial Match . . . 85
B.3.3 Further Matching . . . 85
B.4 Conclusion & Future Work . . . 87
B.5 Immediate Future: Generalization . . . 87
B.6 "Fuzzy Matching" . . . 88
B.7 A Note on Condition Strictness . . . 88
B.8 Scaling the Matching Algorithm . . . 88
B.8.1 Role Assignment Restriction . . . 88
B.8.2 Instance Abandonment . . . 89

B.8.3 Schema Retrieval 89

C Critical Acclaim for *Lane's Area Paper* **90**

Chapter 1

Introduction

True natural language understanding requires an understanding of content as well as linguistic form. To convincingly discuss topics from the real world, an AI system would require at least cursory knowledge of a vast breadth of real objects and kinds of events. Intelligent reasoning would require even more structure of this knowledge, to facilitate logical inference, analogy-making, and efficient storage and retrieval. Research on the psychology of learning supports the importance of structured, interconnected knowledge (Murphy and Medin, 1985; Keil, 1992) and logical inference (Schank et al., 1986) in forming conceptual categories; these theories suggest that even "simple" tasks like categorization rely on a vast array of cognitive processes and world knowledge, rather than on intrinsic lists of features.

A key difficulty in mechanizing knowledge-based cognition is the transduction of real-world knowledge into a computationally manipulable form—the "*knowledge acquisition bottleneck*". Most digitized human knowledge exists as natural language, and must be converted into a representation conducive to reasoning. But to learn by processing every written fact would be difficult, to say the least—and certainly not at all like how humans accumulate knowledge. We each seem to possess vast amounts of knowledge, much of which is not explicitly represented in the sources we've learned from. This surprising disparity has been noted since antiquity, when the *Meno* concluded—perhaps somewhat extremely—that most human knowledge is inborn (Plato, 402 BCE). Landauer and Dumais (1997) theorize that knowledge within some domains is highly interconnected, allowing inference and analogy to help us generalize and synthesize new knowledge even with limited exposure to existing knowledge. If inference and analogy assist knowledge generation, it follows that more powerful inference tools would result in more powerful knowledge generation tools. But more powerful inferences call for a richer knowledge structure—so how do we select one?

In this paper, I will focus on **schemas**: information structures that represent *generalized* patterns of commonly occurring events. Schemas are well-studied in the field of psychology. In his 1932 book *Remembering* (Bartlett, 1932), Frederic Bartlett found that, when people forgot details of stories they were told, they were likely to re-cast events like "paddling in a canoe" in terms of more familiar types of event from their own cultures, such as "sailing in a boat". Jean Piaget called schemas *"cohesive, repeatable action sequence[s] possessing component actions that are tightly interconnected and governed by a core meaning"*, and suggested that understanding new information involved either *assimilating* that information into an existing schema, or *accommodating* the information by changing an existing schema to fit it. He also suggested that children start with a small initial set of schemas to "seed" the lifelong schema generation process.

The idea of schemas quickly caught on in the fields of cognitive science and artificial intelligence, and Chapter 2 will start this paper off by describing some influential models of schemas from those fields; the literature is long and the nights are short, however, and so its focus will primarily be on models of schemas that pertain to story understanding. In Chapter 3, I will discuss existing work on automatically generating schema-based knowledge, separating the work into two major classes: *classical* approaches, which use mainly symbolic representations, and *statistical* approaches, which are based on modern machine learning techniques. Finally, in Chapter 4, I will discuss the "protoschema" approach—my own research group's approach to schema learning, based on the theories of my advisor, Dr. Lenhart Schubert—and briefly discuss some of my initial work on implementing that approach.

Chapter 2

Schema Theories and Models

Stories are purposefully structured accounts of events in the real world—which events are, in a sense, structured themselves, by the laws of the physical universe at the lowest level, or sometimes by higher-level rules about events derived therefrom (e.g. rules from kinematics, biology, and other high-level disciplines). There can be no doubt, then, that a highly structured model of information is not only *practically* necessary to "understand" a story, it is an essential prerequisite of even defining the story understanding problem. What's more, the sheer number of possible entities, actions, and qualities one must be able to understand in a story is hinted at by—and yet certainly staggeringly larger than—the number of nouns, verbs, and adjectives in any modern human language. Some "world knowledge" of these concepts must then also be a prerequisite of true story understanding.

In this paper, I'll explore a classic concept that's turned up, in some form or another, in philosophy, psychology, cognitive science, and, most recently, artificial intelligence: that of *schemas*. Though the exact nature of schemas varies from theory to theory, the word always refers to structured pieces of information that represent stereotypical objects, situations, or concepts. When schema theories focus on stereotypical, procedural event patterns, such as the steps involved in "going to a restaurant", they are often called *scripts*. When they focus on static, database-like representations of objects or abstract concepts, such as "toaster" or "train trip", they are often called *frames*. Throughout this paper, unless I specify an alternate sense, I will use the word "schemas" to refer to a class of information structures containing both scripts and frames.

To explore the historical importance of schemas in theories of story understanding, I will begin this chapter with a summary and discussion of the hugely influential theory of discourse understanding due to Van Dijk and Kintsch (1983). Then, I will examine some past approaches to schemas through the lens of that theory. In fact, it is worth noting that van Dijk and Kintsch's theory of discourse understanding came a considerable number of years *after* Minsky's frames (Section 2.2 on page 11) and Schank and Abelson's scripts (Section 2.3 on page 19), and was heavily influenced by those theories. As I'm discussing story understanding, however, I choose to introduce it first because I feel that we should discuss frames and scripts with the goal at hand also in mind.

2.1 "Strategies of Discourse Comprehension"

The book "Strategies of Discourse Comprehension" (Van Dijk and Kintsch, 1983) presents a hierarchical, structured model of meaning representation for events and stories. It has been highly influential, leading to widely cited works such as Landauer's latent semantic analysis (LSA) (Landauer et al., 1998), McNamara's experiments on textual coherence and story understanding (McNamara et al., 1996), Graesser's work on the role of inferences in text comprehension (Graesser et al., 1994), Zwaan's "event indexing" model and associated experiments (Zwaan et al., 1995), and many others. Given the intuitiveness, generality, and impressive long-term influence of the theory, it seems appropriate to use it as a lodestar for outlining the necessary features of a story understanding system.

2.1.1 The Gist of the Model

van Dijk and Kintsch describe a "strategic" model of discourse comprehension, where a listener actively constructs a mental interpretation of events while hearing about those events from a speaker—or a text. By applying a set of "strategies" to representations of the spoken story information, unspoken contextual information, and "presuppositional" knowledge they already possessed, a listener (or reader) iteratively distills the story into progressively larger and more abstract units of meaning. I will first describe some basic cognitive and discourse assumptions their model makes, then outline the model in more detail.

2.1.2 Assumptions

Throughout these descriptions of the model's major assumptions, I will refer to a hypothetical parade, observed and written about in a newspaper article by "Jessie" the journalist, and later read about by "Nellie" the newspaper reader.

- **The constructivist assumption:** Jessie and Nellie both understand the parade by constructing mental representations. Jessie constructs hers on the basis of her observations, and Nellie constructs hers based on the linguistic summary provided by Nellie. These representations form the basis for understanding.

- **The interpretative assumption:** The representations that Jessie and Nellie have constructed do not represent only the raw sensory data of the parade or the newspaper; Jessie constructs a mental representation of a parade *event*, and Nellie constructs one of a story event. These representations are higher-level than the observation data alone.

- **The on-line assumption:** the women construct their mental representations *in real time*, rather than after concluding their observations: Jessie as she observes the parade, and Nellie as she reads the article.

- **The presuppositional assumption:** each woman needs prior knowledge to make sense of the parade. Jessie needs knowledge of events involving vehicles and their movement, crowds of people, celebratory events, etc. Nellie requires similar knowledge, plus knowledge of how a story relates to the events it describes.

- **The strategic assumption:** each woman achieves understanding by combining her internal "presuppositional" knowledge, the raw "input" data she is observing, and additional knowledge of the "situation" in which the observed events are embedded (e.g. being in Times Square on a foggy day, or being at home reading a left-leaning newspaper). The combination of these three types of knowledge can happen in any "order", with a new observation altering a representation formed minutes ago, and can serve to predict unobserved information to fill in logical gaps.

- **The functionality assumption:** Discourse is represented as an event as well, and that event's properties interact with the process of constructing an interpretation of the discussed events. Nellie, aware of the author of the newspaper article and knowing that Jessie is trusted, will likely attempt to "match" her understanding with Jessie's. If Jessie were explaining the parade to Nellie in person, Nellie would also be aware that she could interrupt to request missing information from Jessie.

- **The pragmatic assumption:** A story is given by a series of "speech acts", knowledge of which is necessary to interpret the form and function of pieces of the story. Story segments can *be* speech acts with unspoken goals (e.g. *"It would certainly be a shame if someone felt the need to silence you by force."* is a threat), or can be presented through a series of speech acts with narrative goals (e.g. *"One should note, before reading on, that the bill was vetoed last June."* both informs the reader of a fact and explicitly directs their interpretative process).

- **The interactionist assumption:** The "top-level" interpretation of a discourse "embeds" the interpretation of the story itself. This is a generalization of the pragmatic assumption, allowing goal-directed speech acts to simply be events in a discourse situation, represented alongside the events of the story they tell of.

- **The situational assumption:** The interpretation of the discourse, like the interpretation of the embedded story events, requires its own situational context—a speech act may have different interpretations when spoken in a bar than when spoken in a courtroom.

2.1.3 The Model

Propositional Form

The model begins with the conversion of some observed input into a "propositional" form. Propositions, the authors say, are abstract representations of the meanings of declarative sentences. This definition is somewhat vague, and for good reason: in the logical literature, there has been significant disagreement over the definition of a proposition. Many early and simple logical formalisms conceive of propositions as expressions to be resolved into truth values: "the boy is ill", "she knows that the boy is ill", and "the sky is blue" are all simply "aliases" for either $\top$ (truth) or $\perp$ (falsity). The propositions contain *terms*, as well: "I know who the Queen of England is" and "I know who the Queen of the Bahamas is" give the same truth value, as "the Queen of England" and "the Queen of the Bahamas" refer to the same underlying entity (the woman herself) in the "domain" of entities of the logic. But these sentences clearly do not always have the same truth value: someone might not know that the Bahamas is a Commonwealth state.

van Dijk and Kintsch note that a logical language in which all terms and propositions are only "references" to truth values or entities in the domain is an insufficient model of natural language. The sentence "the Queen of England is the Queen of England" is seen as a useless tautology and not worth stating aloud, whereas the sentence "the Queen of England is the Queen of the Bahamas" is seen as potentially surprising and educational. However, these sentences are logically equivalent: the terms all resolve to the exact same entity. Without going into too much formal detail, van Dijk and Kintsch make it clear that their model a propositional form where a term or proposition cannot simply be re-written as the “extensional” value to which it refers, but rather carries its own “intensional” meaning. Such intensional logics abound in the literature, and the authors do not dictate any specific propositional form.

Propositional Schemas

After forming atomic propositions from observations, the listener (or reader) organizes those propositions into higher-level “propositional schemas”. While their propositions are atomic predications—for example, `Book(b)`, `Boy(John)`, or ¬ `Sick(mother(Mary))`—sentences can be arbitrarily complex. Upon reading “John had a book, which Jake stole and gave to Mary”, we want a structure that uses the same individual referred to by “a book” in the “have”, “steal”, and “give” propositions. We may also want to relate the times of these events (e.g. John had the book before it was stolen). Finally, an atomic proposition within a sentence may not refer to an atomic intensional predicate in the underlying logical form, but rather to another complex structure of propositions with its own associated sub-propositions and situational context. The structures used to organize the propositions in this way are called “propositional schemas”, and are the main units of knowledge in the model.

Textbase

The “textbase” is the main representation of the discourse. The listener forms the textbase by applying “strategies”—which can be thought of as production rules, that is, actions taken when some conditions are met—to propositional knowledge. The textbase is simply a set of related propositional schemas, but its core characteristic is its hierarchical organization: the propositional schemas taken from the surface form of the discourse are combined with prior knowledge and used to infer new propositions or propositional schemas, then matched or combined into higher-level propositional schemas. By combining low-level propositional structures into higher-level ones, the raw text is distilled into representations of story events (a crowd gathered, the crowd watched the parade floats move forward, the parade floats stopped), higher-level event patterns (like a crowd gathering for a celebration), and, eventually, a story “gist”, e.g. “a Memorial Day parade happened downtown today”. This entire hierarchical textbase structure is basis for understanding the text: it encodes the narrative structure, which encompasses all of the story information, in a logical form conducive to inference.

2.1.4 Discussion

van Dijk and Kintsch's model demonstrates the applicability of hierarchical, logical knowledge structures to understanding stories. In their theory, events have common representations, and representations gained by observation can be transmitted verbally and impart much of the same information—missing information can also be "filled in" if similar events have been observed in the past. As we noted at the start of this section, the ideas they presented about representing stories in this sort of logical form have formed the basis for later work on computational story understanding. Echoes of their model's emphasis on using knowledge representations that can model the beliefs and goals of others will also be heard in Chapter 4 on page 51, where I describe my own research group's schema representation.

2.2 Minsky's Frames

In Marvin Minsky's 1974 article "A Framework for Representing Knowledge" (Minsky, 1974), he introduced "frames": data structures that "package up" information about entity types (or specific entities). These entities are not limited to physical objects: they can also be abstract concepts, e.g. family structures, or situations, e.g. weddings. He posits, like van Dijk and Kintsch, that a concept must be understood not in isolation, but as it is "framed" by relevant background knowledge. The background knowledge in a frame is organized in a "slot-and-filler" structure, much like a relational database: a wedding might have slots for the individuals getting married, the location, the time, and so on.

Although Minsky's 1974 article highlights the importance of slots, default values, "sub-frames", and conditions that values must meet to fill slots, it does not define a specific frame language. To more concretely illustrate some salient features of Minsky's conception of frames, I'll present an s-expression domain language, due to James F. Allen[1], in Section 2.2.1 on the next page.

After concretely exploring some of those aspects of frames, I'll describe Minsky's ideas for a matching algorithm, and some of the applications he proposed for "frame systems" (i.e. collections of inter-related frames).

[1]The language definition is taken from a mix of personal correspondence and lecture notes; while I have no formal citation, I am happy to furnish the relevant documents upon request. The "IF-ADDED" and "IF-NEEDED" procedural attachment rule syntax is due to (Brachman and Levesque, 2004).

2.2.1 Frame language implementations

Minsky's ideas about frames were fairly loose, and he noted in the original article that his theory was not unified or coherent, and that many details had been left to future implementations of the ideas. One of the first attempts to implement frames was KRL (the Knowledge Representation Language), due to Daniel G. Bobrow and Terry Winograd (Winograd, 1973). KRL was extremely powerful—essentially a programming language—but in attempting to provide both human-like expressivity and a formal semantics, it *"just bogged down under the weight of trying to satisfy all those things at once"* (in Winograd's own words). As later frame languages were developed, many of the loosely specified powers of frames were cast aside in favor of a more tractable formal semantics. As a result, many modern implementations of frames are equivalent to hierarchical "attribute-value" systems: in such systems, frames have typed attribute slots, and may be defined in terms of other frames via an inheritance hierarchy (like those found in object-oriented programming), but the attributes of frames are not easily related to one another by complex logical formulas.

Most descendants of Minsky's frames fall into the class of what are now called "description logics", beginning with KL-ONE (Brachman and Schmolze, 1989). Other frame-like systems, most of them equivalent to description logics, are regarded as languages for what what Tim Berners-Lee called the "Semantic Web" (Berners-Lee et al., 2001), an extension of the World Wide Web in which information is given a formal representation to facilitate its understanding by *computers*, rather than just humans. The Semantic Web spawned the Web Ontology Language (OWL), and its highly formal description logic variant, OWL DL (McGuinness and Van Harmelen, 2004). OWL has been, and continues to be, used as a standard taxonomic description language, especially for biomedical- and health care-related applications (Smith et al., 2007).

In the remainder of this section, I will present an example frame language—which I've rendered as both s-expressions and equivalent first-order logic predicates—due to James F. Allen. This frame language is representative of the general capabilities of most modern description logics.

An Example Frame Language

Figure 2.1 on the following page gives some example frames which exhibit the main features of our example frame language. This section will define the frame language by defining each syntactic construct of those examples. Figure 2.2 on page 14 provides an interpretation of the `CAR` example frame as a first-order logic predicate.

```
(Define CAR as
    (a VEHICLE with
        wheels   (a-set (a WHEEL) with cardinality 4)
        body     (a CAR-BODY)
        engine   (a ENGINE)
        transmission-type (one-of AUTOMATIC, MANUAL)
    ))

(Define RACECAR inherits-from
    (a CAR with
        sponsors (a-set (a COMPANY))
        top-speed (a SPEED)
    ))

(Define TOURNAMENT-RACE as
    (a EVENT with
        city (a CITY with default TALLADEGA)
        racecars (a-set (a RACECAR))
        audience (a-set (a PERSON))
        ticket-price (a NUMBER)
        ticket-revenue
            [IF-NEEDED (|SELF:audience| * SELF:ticket-price)]
        next-race (a TOURNAMENT-RACE)
            [IF-ADDED SELF:next-race:prev-race ← SELF]
        prev-race (a TOURNAMENT-RACE)
            [IF-ADDED SELF:prev-race:next-race ← SELF]
    ))
```

Figure 2.1: Some examples of Minsky-style frames

```
$\forall x.$ CAR$(x)$

    ; Bidirectionality given by the "as" keyword
    $\iff$

    ; Based on vehicle frame
    VEHICLE$(x)$ $\wedge$

    ; wheels
    ; (set with cardinality 4; element uniqueness axioms
    ; included for rigor)
    Set(wheels$(x)$) $\wedge$
    $\exists a,b,c,d$ $(\neg(a=b)$ $\wedge\neg(a=c)$ $\wedge\neg(a=d)$ $\wedge\neg(b=c)$ $\wedge\neg(b=d)$ $\wedge\neg(c=d)$
        $\wedge$ $(\forall e$ $(e \in$wheels$(x)$ $\iff$ $(e=a \vee e=b \vee e=c \vee e=d))))$ $\wedge$

    ; body
    Body(car-body$(x)$) $\wedge$

    ; engine
    Engine(engine$(x)$) $\wedge$

    ; transmission-type
    ; (generally an XOR, but the equality propositions for unique
    ; individuals here don't require us to be explicit about that)
    (transmission-type$(x)$ $=$AUTOMATIC $\vee$ transmission-type$(x)$ $=$MANUAL)
```

Figure 2.2: A first-order logic interpretation of the `CAR` frame from Figure 2.1 on the previous page.
Symbols with only their first letters capitalized represent predicates. Multi-character lower-case symbols represent functions. Single-character lower-case symbols represent variables. $\in$ and $=$ are special infix predicates.

Frame Definitions

A frame type is defined in terms of another frame type in a single-inheritance (or equivalence) relation. A frame can inherit from the empty (i.e. slotless) type "THING". In the example, CAR is defined **as** a VEHICLE, whereas RACECAR is defined to **inherit from** a CAR. The rough semantics of these definition keywords, in first-order logic, are that $\forall x.\, CAR(x) \iff VEHICLE(x) \wedge C1(x) \wedge C2(x) \wedge C3(x)$. $C1$, $C2$, and $C3$ are the constraints introduced by the new `wheels`, `engine`, and `transmission-type` slots—we'll detail those constraints in Subsection 2.2.1. In plain English: all cars are just vehicles with four wheels, an engine, and an automatic or manual transmission, **and** all vehicles with those properties are, by definition, cars. The **inherits-from** definition, then, means $\forall x.\, RACECAR(x) \implies CAR(x) \wedge C1(x) \wedge C2(x)$, where $C1$ and $C2$ are constraints for the `sponsors` and `top-speed` slots. In plain English: all racecars are cars with sponsors and a top speed, **but** a car with sponsors and a top speed *might not be a racecar*. We call a definition with the **inherits-from** keyword a "partial definition", and one with the **as** keyword a "full definition".

Slots

What (Minsky, 1974) calls "terminals", we refer to as "slots". A slot is a key-value pair where the value is an individual, i.e. either a *literal*, like the number 24 or the string "hello", or a *reified frame*, like `(a PERSON with name "Lane")`. Here, the word `a` *reifies* that frame predicate such that the value of the slot is just some person whose name is Lane, even if we don't know the value at the time of the definition. [2] In addition to reifying frame predicates, we can also reify some "atomic" type predicates like `NUMBER`, `STRING`, and other things that would be difficult (or potentially impossible) to define as frames. One notable atomic type is the set, which can be reified without specifying its objects by using the `a-set` operator, which takes a reified predicate and a cardinality and indicates an unspecified set.

The `one-of` operator takes a list of predicates (or literals, which it transforms into equality predicates) and yields their reified exclusive disjunction, i.e., specifies a type predicate that is true of a value when and only when exactly one of the constituent predicates are true of that value. This can be used to construct explicit sets without additional syntax: `(a-set (one-of 1, 2, 3, 4) with cardinality 4)` is only true of the

[2]Though we have not yet formalized a predicate logic for the semantics of these frames, we will not need explicit reification semantics as defined in (Quine, 1985) to do so; because we will ultimately render frames and frame instances as predicates, we can simply apply a slot's un-reified frame predicate to the value of its associated slot function. So, it turns out that first-order logic is still sufficient, so long as we leave aside issues of intensionality.

set $\{1, 2, 3, 4\}$. [3]

Not all slots need to have literals assigned to them; in fact, frames are useful in large part *because* they allow prediction of certain typed properties upon observation of some other typed properties. If we heard of an event in a city that involved race cars, we might "match" those observations to slots in a `TOURNAMENT-RACE` frame, and predict that there is probably an audience and a ticket price, even if we don't know the exact values for those slots.

Note that a fact we're generally aware of—that the car's engine is located inside the car body—cannot be represented in this language. As we discussed in the introduction to this section, modern frame languages typically do not support this kind of arbitrary inter-relation of slots.

Procedural Attachment Rules

In some cases, we may wish to attach *procedures* to frame slots, rather than just values or reified frame predicates. (Brachman and Levesque, 2004) introduce procedural attachment rules, which allow for arbitrary computation when triggered. There are two types of procedural attachment rules, both introduced by (Brachman and Levesque, 2004): `IF-NEEDED` and `IF-ADDED`.

An `IF-NEEDED` rule takes the place of a literal value or a reified frame predicate in a slot. It can be thought of as a "lazy" computation: the `ticket-revenue` rule in the `TOURNAMENT-RACE` frame in Figure 2.1 on page 13 multiplies the cardinality of the set `SELF:audience` by the number `SELF:ticket-price` to calculate the final value for ticket-revenue. More advanced procedures can be used to compute these values; (Brachman and Levesque, 2004) do not define a computational framework, but we assume Turing-completeness and a type system that matches our frame language's.

An `IF-ADDED` rule attached to slot specifies a procedure that is executed when a value is assigned to that slot. The rule in the `next-race` slot in the `TOURNAMENT-RACE` frame in Figure 2.1 on page 13 ensures that, if a `TOURNAMENT-RACE` frame instance T_{n+1} is assigned to the `next-race` slot of instance T_n, the `prev-race` slot of frame T_{n+1} will receive value T_n (the reverse rule is attached to the `prev-race` slot). This maintains a kind of programmer-defined "referential consistency". However, these rules could be used for many other purposes: for example, we might want to set the value of slot A as soon as a value is assigned to slot B, conditional on some other slot values. The model of computation here is also not well-defined, but we assume that the procedure can make arbitrary modifications to any frame instances in the "frame universe".

[3]To avoid explicitly specifying the cardinality, we can define a cardinality `auto`, syntactically valid only when the set predicate is given by `one-of`, which takes its value from the cardinality of the `one-of`'s argument set. This syntax was not given in the original frame language due to Allen and Schubert; I'm suggesting it only for completeness.

Default Values

We might wish to have values pre-assigned to slots, pending an explicit assignment. Minsky calls these "default" assignments. It is fitting that they are analogous to the non-monotonic [4] "default logic" first presented in (Reiter, 1987a), in which certain default rules define the truth of a family of propositions pending more specific information about the falsity of a sub-family of those propositions. However, Minsky's motivation for default values is grounded more in cognitive modeling than in formal inference: he stipulates that characteristic default values exist in frames in the human mind. He gives an example sentence, "John kicked the ball", noting that a listener likely forms a mental picture of a "vaguely particular" ball—perhaps a soccer ball—absent any information in the story about the ball's characteristics. Default values are syntactically quite straightforward in our frame language: the construction `(a CITY with default TALLADEGA)`, found in the `city` slot of the `TOURNAMENT-RACE` frame in Figure 2.1 on page 13, can be interpreted as an English noun phrase to obtain its rough definition.

2.2.2 Matching

Of particular importance to schema research writ large is the problem of correctly mapping world observations to their schematic representation. In the case of frames, Minsky outlines a three-step matching process:

1. A weak, cheaply executable heuristic "evokes" a frame as potentially appropriate for an observed situation. This might be as simple as an evocative word like "buy" triggering a "commercial transaction" frame.

2. The candidate frame's relevance is tested more thoroughly: additional heuristics involving specific observed relations and entities are applied. In the case of a commercial transaction frame, it might confirm the presence and/or exchange of money.

3. Once the frame is deemed appropriate, a "true" matching process begins. A frame may have many terminals: not only would the commercial transaction frame could have slots for time, location, people involved, the object being exchanged, the type of currency, and more—it could also need to match all of the slots in those sub-frames! A scenario is an intricate structure with many features. Thus, the system will choose which terminals to match on the basis

[4] It should suffice, here, to loosely define a non-monotonic logic as a logical system in which a deduction may be valid in some entailment chains but not others.

of some *goals*—perhaps the frame system works for the IRS, and only needs to know the name of what was traded, by whom, and for how much of which type of currency. It needn't then concern itself with the model or color of a car being sold, with the relative heights of those making the deal, etc.

Of course, just because we reach step 3 does not mean that we have the correct frame: if there are significant inconsistencies, or observed features relevant to our goals with no corresponding slots, we may wish to find a new frame, or even create a new one to describe our observations. Minsky proposes using slot assignment errors as the basis for not only second-attempt frame retrieval heuristics, but also frame modification/creation. [5]

2.2.3 Frame systems

Minsky's original article discussed "frame systems": corpora of inter-related frames, each of which provide a different "view" of something. Due to the power and generality he ascribed to frames, the set of things frame systems could be employed to represent was noted to be quite diverse, including abstract concepts, like a kind of situation, as well as concrete entities, like a 3D representation of a room with many possible viewpoints). Given the focus of this area paper, I will remark on their application to story understanding.

Applications to story understanding

Minsky believed that nested frames are well suited for modeling the meaning of a discourse. He imagines a frame for a story, with slots representing the setting, characters, and other features of the story. Each of those slots would, in turn, be a complex frame, but Minsky notes that the first line of a "properly told" story "usually" establishes the larger-scale components, such as setting and protagonist, before such details are needed. This notion of a story's sentence-sequential structure is, of course, quite reductive: efficiently building the frame graph by reading sentences in order would rely on the story being told in a top-down way! But the general notion of representing a story as a hierarchical, decreasingly abstract tree of narrative structures down to event particulars should sound familiar: (Van Dijk and Kintsch, 1983) base their theory of discourse structure on this notion—which theory I describe in Section 2.1 on page 7. [6]

[5]The use of slot value assignment inconsistencies as the basis for refining a world model has been explored before, in both highly structured situation models, e.g. (McDermott, 1974), and emergent, connectionist models, e.g. (Botvinick and Plaut, 2004).

[6]And, indeed, van Dijk and Kintsch refer to Minsky's article on frames in *Strategies of Discourse Comprehension*.

2.3 Schank's Scripts

One of the most influential theories of script-based language understanding is due to Schank and Abelson (1977). Inspired by the "script theory" of psychologist Silvan Tomkins (Tomkins, 1978), their book "Scripts, Plans, Goals, and Knowledge Understanding" makes scripts slightly more computationally concrete by extending Conceptual Dependency (CD) theory (Schank, 1969)—a theory of *sentence* understanding based largely around breaking verbs into "conceptual primitives"—into a more general theory of *situation* understanding. To extend sentence understanding to situation understanding, the authors introduce the notion of a *script*: a stereotyped, generalized template encapsulating and relating participants and sequential events.

To illustrate the role of a script in less abstract terms: there is little mystery to the short story *"Gaurav went to the French restaurant. He ordered coq au vin, and asked to be reseated."* We have a good idea of why those actions happened, and of what happened in between: he was probably seated by a host, attended to by a waiter, asked by the waiter what food he wanted, knew the restaurant had French food, knew that only the staff could assign him to a different table, and many more obvious, but explicitly unstated, facts and events. The scripts that Schank and Abelson describe allow for this kind of situational knowledge retrieval.

In this section, I will briefly introduce CD theory, with a focus on its conceptual primitives. Then, I will describe the types, properties, and uses of scripts as defined in the book, referring throughout to one of its example scripts, represented in Appendix A on page 80. I'll close the section with a brief discussion of my thoughts on script theory

2.3.1 Conceptual Dependency (CD) Theory

Roger Schank's CD theory (Schank, 1975) was an attempt to model sentence understanding in a way conducive to computation—that is, by creating a formal representation of sentences. Schank posited one essential "axiom" of CD theory: **any pair of sentences with the same meaning should have the same representation**. Although the sentences "I sent her to the store" and "I sent our daughter to the grocery store" may mean the same thing when uttered in a particular context, to represent the first sentence, all its implicit, contextual, and inferable information must be "packaged up" into the unique meaning representation of the utterance.

Clearly, we cannot package up all inferences drawable from the sentence and the context: there are potentially infinite such inferences, and many may not be interesting. So, Schank further posits a standard set of inference rules, and a representation conducive to inference—this is, after all, all being done with computation in mind. So, Schank settles on a representational scheme with the following components:

- **Primitives**: "atomic" representations of actions in the world. For example, the verbs "buy" and "sell" should be decomposable into the same primitive actions. In fact, Schank posits that *all* verbs can be decomposed into only about 11 primitive actions.
- **Active conceptualizations**: meaning propositions about actions undertaken by an actors, of the form `Actor Action Object Direction (Instrument)`.
- **Stative conceptualizations**: meaning propositions about the states of objects in the world, of the form `Object (is in) State (with Value)`.

The primitive actions in the CD theory meaning representation are described below, in Section 2.3.1.

Active conceptualizations

- **`PTRANS`**: changing the physical location of an object. So, "Jake goes to school" becomes `Jake` **`PTRANS`** `Jake to school`.
- **`ATRANS`**: transferring possession of something to another actor. (Schank and Abelson, 1977) actually generalizes this a bit to "an abstract relationship such as possession, ownership, or control". A buying action and a selling action are both characterized by the same two **`ATRANS`** actions: one of money, and the other of an object.
- **`MTRANS`**: communicating mental information to another actor. They also define **`CP`**, the conscious processor, and **`LTM`**, long term memory, to be used in **`MTRANS`** propositions. For example, "tell" is a person-to-person **`MTRANS`**, and "remember" is an **`LTM`**-to-**`CP`** **`MTRANS`** (i.e., your memory is communicating information to your conscious mind, as if both were actors).
- **`MBUILD`**: synthesizing new information from known information. "Deduce", "imagine", and "decide" are examples of **`MBUILD`**-characterized actions.
- **`ATTEND`**: focusing a sense organ on an environmental stimulus. **`ATTEND`** `eye` is "see", **`ATTEND`** `nose` is "smell", etc. This is often used as the `Instrument` argument of active conceptualizations characterized by **`MTRANS`**: so "see" is, more accurately, **`MTRANS`** `to` **`CP`** `by instrument of` **`ATTEND`** `eye to object`.

- **GRASP**: grasping an object for manipulation. The verb "throw" can be expressed as a **GRASP** followed by the end of a **GRASP**.
- **PROPEL**: exerting physical force on something. That thing need not necessarily move as a result. "John threw the ball" can be expressed as the end of a **GRASP**, followed by a **PROPEL**.
- **MOVE**: moving one's own body part. **MOVE** is often used as an instrument for other actions: so "John threw the ball" will also be "by instrument of " John **MOVE**-ing his arm. **MOVE** is also the non-instrumental characterizing primitive for verbs like "kiss" or "scratch".
- **INGEST**: taking an object inside of one's self, e.g. "eat", "drink", and "breathe". The exact set of orifices **INGEST** supports is both unclear in the literature and probably best left unexplored.
- **EXPEL**: excretion, secretion, or otherwise ejection of an object from the body. "Sweat", "exhale", "cry", and many other colorful examples are characterized by **EXPEL**.
- **SPEAK**: producing a sound, not necessarily directly through one's own bodily apparatus, and, unlike **MTRANS**, not necessarily for the communication of a coherent piece of information. "Squeal", "play music", and "purr" all involve **SPEAK**.

The creative composition of this sort of primitive action to represent more complex actions was not a revolutionary idea to AI researchers at th time: as I detail in Section 3.1.2 on page 35, the STRIPS system's (Fikes et al., 1972) "macro-operations" were hugely influential in the planning literature—no doubt including this book—due to the simplicity and power of the situation models and planning algorithms they facilitate.

Stative conceptualizations

Primitive actions form the basis for active conceptualizations (**ACT**s). Stative conceptualizations (**STATE**s), on the other hand, can be thought of as attribute-value pairs. Some states are qualitative, e.g. "the sky is blue". However, some are quantitative, and so Schank suggested a normalized *scale*, from -10 to 10, for such attributes, such as **HEALTH** (-10: dead, diseased, under the weather, tolerable, 10: "in the pink" [7]),

[7]I actually had to look this phrase up, which may date this research.

AWARENESS (-10: dead, unconscious, asleep, awake, 10: keen) [8], and more; Schank does not provide a definitive list of "primitive quantities".

2.3.2 Causal chains

In chapter 2 of the book, the authors introduce *causal chains*, wherein a set of uniquely identifiable CD theory conceptualizations—both active and stative—are related. Put more simply: causal chains relate actions to the states they effect, the states that bring them about, and other actions.

They define the following set of causal relations:

- $\Uparrow_r$: an **ACT** *results in* a **STATE**.
- $\Uparrow_E$: a **STATE** *enables* an **ACT**.
- $\Uparrow_I$: a **STATE** or **ACT** *initiates* a mental **STATE**.
- $\Uparrow_R$: a mental **ACT** *is the reason for* a physical **ACT**.
- $\Uparrow_{dE}$: a **STATE** *disables* an **ACT**.
- $\Uparrow_{rE}$: an **ACT** leads to a **STATE**, which then enables an **ACT**. *(This is an abbreviation for a $\Uparrow_r$-to-$\Uparrow_E$ chain.)*
- $\Uparrow_{IR}$: an **ACT** or **STATE** initiates some mental state, which is then the reason for an **ACT**. *(This is an abbreviation for a $\Uparrow_I$-to-$\Uparrow_R$ chain.)*

So, "John gave Bill an orange for his cold" can be expressed as the causal chain `Bill` **`HEALTH`**`(POS` [9] `change)` $\Rightarrow_r$ `Bill` **`INGEST`** `orange` **`to`** **`INSIDE`**`(Bill)` $\Rightarrow_{rE}$ `John` **`ATRANS`** `orange` **`to`** `Bill` [10].

N.B.: The relation arrows in the causal chain above seem to point in the "wrong" direction. I have preserved their direction in this example, which was originally given in (Schank and Abelson, 1977).

Causal chains provide the first level of inter-sentential connection necessary to achieve situational understanding. But stories, and even observations in the world, frequently omit steps in causal chains. Some events also contain multiple causal chains, as they do not necessarily have one end goal: going to a restaurant for a date may have the goals of impressing a potential romantic partner *and* eating some

[8]-10 seems to be "dead" for an awful lot of the quantities Schank suggests.

[9]Here, `POS` indicates a positive change in quantity.

[10]Here, I've rotated the arrows 90° for formatting reasons.

delicious food, but we nevertheless have certain stereotyped expectations about what may occur: one person may pull a chair out from the table for the other, one person will likely pay for all of the food, etc. So, we need an event representation richer than causal chains, not only to practically understand sparse stories, but also to model complex situation types.

2.3.3 Scripts

We come now to *scripts*, the core of the book. A script describes an entire situation: sequences of events, their causal relations, and sets of involved actors, objects, locations, and other entities falling under the purview of CD theory. Befitting their name, this book's "scripts" have many similarities with play scripts, i.e. screenplays: both representations are divided into "scenes"; both "set the stage", so to speak, by introducing actors, sets, and notable objects in each scene; and both describe a sequence of actions (for screenplays, mostly speech actions). However, they differ from screenplays in an important way: scripts in the Schank and Abelson sense model generalized, "stereotyped" event sequences that fit many actual situations in the world, for example "eating at a restaurant" or "graduating college". The specific actors, objects, and locations are not fixed in this representation, but may be filled in on the basis of actual observed events. In this way, scripts are useful for explaining, and making predictions about, *arbitrary* stories and observations.

Script structure

Scripts have a kind of dual structure: they have named slots, filled with values [11], and they have temporal action sequences describing an overall situation's procession. The named slots hold values of the types supported by CD theory: they may have *actors*, which are animate objects capable of performing active conceptualizations; *props*, which are physical objects; and locations. The event sequences are divided into *scenes*: for example, their book's canonical "restaurant" script, which I've illustrated in Appendix A on page 80, has scenes for entering, ordering, eating, and exiting.

Every scene has a *MAINCON* ("main conceptualization"). Any instantiation of a scene implies that its MAINCON certainly happened. In the example script in Appendix A on page 80, for example, the MAINCON of Scene 2 ("Ordering") is `S MTRANS `I want F' to W`, that is, the customer telling the waiter that they want food.

[11]These structures are quite similar to Minsky's frames, which are described in Section 2.2 on page 11.

Scripts may have *entry conditions*: in the restaurant example, the customer must be hungry and have money for the script to be instantiated. They also may have *results*, that is, "postcondition" stative conceptualizations attributed to the entire script. Finally, scripts may express complex control flows by specifying multiple entry points, branches, and loops within and between their scenes. In the restaurant example, Scene 2 has three labeled *paths*, which alter the "execution" of the script depending on whether the menu is already at the table upon sitting down, and Scene 2 may go to a path in Scene 4 or a path in Scene 3 depending on whether the cook is able to make the requested food.

Finally, scripts may have *tracks*: conceptual specifications of the script type. They can also be thought of as "subclasses", in the object-oriented programming sense. For example, the restaurant script they present is on the "coffee shop" track, but restaurant scripts would also need tracks, they say, for pubs, diners, and all other sorts of restaurant.

Script retrieval

A true story understanding system would possess an immense number of scripts, and so we'll need a sophisticated retrieval system to find the scripts evoked by stories. We also need to be able to "match" the stories with the retrieved scripts, both to confirm that we've retrieved an appropriate script and to facilitate our ultimate goal of inference and understanding. [12]

As both the story being understood [13] and the atomic elements of scripts are ultimately composed of many conceptualizations, this script model makes use of some conceptualizations over others in retrieving scripts. We call these retrieval-oriented conceptualizations *headers*, and they come in four forms:

- **Precondition headers** are the conceptualizations in the script's "entry conditions" section. These allow us to recall scripts when their preconditions are true at some state. [14]

- **Instrumental headers** are observed conceptualizations referring to more than one context. For example, *"Gaurav took the bus to his office"* might evoke both the "taking a bus" and "working at an office" scripts. These facilitate the retrieval, instantiation, and causal relation of multiple scripts.

[12]This is also a problem for Minsky's frames, as I discuss in Section 2.2 on page 11.

[13]It is assumed here that some natural language text, or some set of observations, has already been converted into such a form.

[14]This is quite similar to how STRIPS and GENESIS identify relevant "macrops"/schemas; see Section 3.1.2 on page 36 for more information.

- **Locale headers** evoke scripts characteristic of specific places. For example, *"Mariel went to the soccer field"* might evoke the "play soccer" script.

- **Internal conceptualization headers** are conceptualizations that occur in any scene of the script. Observing many actions in a script should probably evoke that script. One of these headers would likely yield a "weaker match" on its own than one of the other three types.

Script interaction

Scripts are not self-contained: many scripts can be instantiated at the same time, and may co-refer to actors, props, locations, and even the same actions. Scripts may also interfere with one another: the authors give the example story *"John was eating in a dining car. The train stopped short. The soup spilled."*. Imagine that the first sentence is part of a restaurant script, and the second is part of a train script. Even if we infer that the stopping-short event from the train script caused the soup spilling event, we still need to "tell" the restaurant that it cannot proceed normally. To identify which scripts need to be informed of which events, we can make use of existing knowledge (soup is a food, and can thus evoke our instance of the restaurant script), co-reference (we might know of an existing soup in the restaurant script instance), and future inputs (if John then exclaims something to the waiter about soup, it facilitates a co-reference interaction). The authors explore many types and specific examples of script interaction, all of which boil down to the essential issues of determining when (or whether) certain script instances begin or end—this is especially an issue when scripts can be interrupted—and ambiguous matching of conceptualizations into one of multiple active script instances. Complete solutions to these fundamental issues are not engineered in the book; it would be largely up to an implementer of this system to solve them.

2.3.4 Discussion

This book presents one of the first serious attempts to lay out a framework for situation understanding based on generalized scripts. The scripts' underlying causal chains relate them naturally to plans, and the simplicity of the CD theory action/state representation lends itself to computational approaches. However, there are a few issues with the approach.

The CD theory conceptualizations, especially the 11 primitive actions, are a cumbersome representation for many complex verbs, and the task of breaking natural vocabularies down into these primitives is likely intractable. What's more, the small set of specific primitives they propose is probably insufficient: Lytinen (1992) notes, twenty years after conceptual dependency theory was introduced (Schank, 1969), that "no one seems to take seriously the notion that such a small set of representational elements covers the conceptual base that underlies all natural languages."

Identifying the full set of "tracks" of scripts would also be quite difficult; even characterizing which entities—like location and type of food sold for a "coffee shop" vs. "diner" track—should be generalized to form new tracks is a difficult problem, and probably one not worth solving, as the notion of "track" should probably be generalized into a more flexible inheritance model.

All in all, this book presents some very compelling ideas about the role of scripts in story understanding, but these fundamental aspects of the model seem to be intractable, even in the modern day, and must be addressed in future automated systems that learn structured scripts like these—which systems may well also reveal other intractable aspects of this framework.

2.4 FrameNet

The Berkeley FrameNet project (Baker et al., 1998), started in 1997 and continuously ongoing, aims to provide a comprehensive corpus of semantic frames that describe events, objects, and abstract relationships in the real world. Their frames bear more resemblance to Minsky's frames (Section 2.2 on page 11) than to Schank and Abelson's scripts (Section 2.3 on page 19), in that their representation does not center around temporal ordering of atomic actions, instead taking a hierarchical slot-and-filler structure to represent concepts at various levels of abstraction.

This structural disparity can be seen as an artifact of the "looseness" of the FrameNet model: the semantics of FrameNet are dictated only by some relatively simple inheritance and composition relationships, with the actual, real-world "meaning" of frames and slots encoded only as natural language text descriptions. Whereas Schank and Abelson base the low-level semantics of their scripts in CD theory conceptualizations, and Minsky "punts" the semantic definition of his frames to actual implementers, the FrameNet project leaves the task to natural language understanding researchers, choosing to focus on *demarcating*, rather than *formally defining*, the core semantic roles of various stereotyped concepts. In this regard, FrameNet is something of a "compromise"; by eschewing a formal, structured semantic representation of the natures of its roles, FrameNet sacrifices inferential and predictive capability for a simpler frame generation task, allowing for its immediate application to tasks such as semantic role labeling (see Section 2.4.2 on page 28).

Today, the FrameNet corpus comprises 1224 frames [15]—all of them hand-engineered—

[15] Frame count information taken from https://framenet.icsi.berkeley.edu/fndrupal/

and they cover many topics: some example frame names are “invasion scenario”, “losing track of”, “key”, “hair configuration”, and “change of leadership”.

2.4.1 FrameNet frames

Frames in FrameNet depend heavily on natural language to represent concepts. Every frame includes a description, in ordinary language, of the concept it represents. However, they do also provide some semantic structure: they demarcate, name, and informally define (i.e., using natural language) the actors, objects, and other conceptual entities necessary to understand the frame, in the form of **Frame Elements** (**FEs**). It also specifies a list of words that evoke the frame, which it calls **Lexical Units** (**LUs**). Finally, FrameNet has a set of **Relations** for composing and relating frames. These constructs are described below.

Frame Elements (FEs)

FEs are the “slots” of the frame: they represent the items, agents, and other frames that participate in the frame. For example, the `Firefighting` frame includes the FEs `Fire` and `Firefighters`. FEs may be constrained by **semantic types**: in the `Firefighting` frame, `Fire` has no semantic type specified, while `Firefighters` has the semantic type `Sentient`.

Frames have two types of FEs: **Core FEs**, which essentially define the frame’s meaning, and non-core FEs, which “add information” to the frame, but do not define its meaning. The Core FEs of the `Cooking_creation` frame are `Cook` and `Produced_food`, while its non-core FEs include `Container`, `Ingredients`, `Manner`, `Place`, and `Time`. (These latter three non-core FEs are extremely common in FrameNet frames; they apply to most events.)

Lexical Units (LUs)

LUs are words—specifically, lemmas with part-of-speech tags—that evoke frames. LUs are also sense-unambiguous: words with many possible senses, like “run” or “play”, are split into multiple LUs (one for each sense). Each frame a list of the lexical units that evoke it. LUs for the `Cooking_creation` frame include `bake.v`, `cooking.n`, and `roast.v`. LUs are only connected to the frame: FEs do not have associated LUs.

`current_status` at the time of writing.

Relations

Frames may be related to one another in several important ways:

- Relations like **Causative_of** and **Inchoative_of** provide the means for causal connection of frames.

- The **Subframe** relation allows composition of frames into more complex scenarios. For example, the `Visiting_scenario` frame has the subframes `Visiting_scenario_arri` `Visiting_scenario_stay`, and `Visiting_scenario_departing`. Subframes may be related with the **Precedes** relation to establish their chronology in larger scenarios.

- Inheritance relations allow more specific frames to build atop the FEs of more abstract frames, whose FEs are mapped into those of the more specific frame. For example, the `Vehicle_landing` frame inherits from the `Arriving` frame.

- The **Perspectivized_in** relation relates neutral-perspective frames, like `Employment_scenario`, with frames that represent the same scenario from a different perspective, like `Employer_scenario` and `Employee_scenario`.

- The **Using** relation is like the inheritance relation, but "less strictly defined" [16]. For example, the `Cooking_creation` frame *uses* the `Apply_heat` frame, but does not inherit from it, as cooking something is not a "kind of" applying heat.

- The **See_also** relation relates similar frames that are not quite semantically identical. For example, the `Awareness` frame notes to *see also* the `Certainty` and `Opinion` frames.

2.4.2 Applications

FrameNet has seen extensive use since its inception, most notably for the task of semantic role labeling (SRL) in sentences, for which its Frame Elements are suited almost by definition. In fact, the very first attempt at automatic SRL, by Gildea and Jurafsky (2002), used FrameNet as the source of its semantic roles. Following their discriminative model (of role-arguments-given-the-frame), work on identifying, and populating the roles of, frames evoked by text has continued, aided by modern learning technologies: Thompson et al. (2003) extended the model to a generative model of both the evoked frame *and* its arguments, Giuglea and Moschitti (2006) did

[16]Source: https://framenet.icsi.berkeley.edu/fndrupal/glossary

SRL with a support-vector machine model, and later systems like Open-SESAME (Swayamdipta et al., 2017) have used recurrent neural networks.

FrameNet has been used for many "downstream" tasks, as well. Rahman and Ng (2011) applied knowledge extracted FrameNet, and the subject-predicate-object fact database YAGO (Suchanek et al., 2007), to the task of co-reference resolution. Shen and Lapata (2007) used FrameNet for question answering by computing question-candidate answer similarity scores using both common semantic roles and mutual frame evocation as heuristics. Petruck and Ellsworth (2018) used the FrameNet model to represent spatial relations, harkening back to (Minsky, 1974), in which frames were first posited as a potential spatial scene representation.

2.4.3 Discussion

FrameNet is a practical system intended to support downstream natural language tasks. Though it encodes minimal formal structure, instead "falling back" to natural language descriptions of frames and roles, its real value has come from the generality of its frames, which cover many situations, and the usefulness of its semantic role definitions in the form of Frame Elements. However, FrameNet's frames are all manually generated, and its heavy reliance on natural language descriptions make automatic frame generation extremely difficult. So while FrameNet is certainly a valuable resource, and a monumental effort of manual schema engineering, it is unlikely to underlie any future system capable of full story understanding; its inherent scalability issues are simply insufficient to reliably model and relate the huge breadth of concepts needed to understand arbitrary stories.

Chapter 3

Past Approaches to Schema Learning

There are more things in heaven and earth, Horatio,
Than are dreamt of in your philosophy.
—William Shakespeare, *Hamlet*

Assembling a corpus of schemas necessary to understand a wide array of world situations is a monumental task that, to date, has no clear solution. The paper *What kinds of knowledge are needed for genuine understanding?* (Schubert, 2015) explores just that question, and refers to the KNEXT system (Schubert, 2002) in doing so. The KNEXT system extracted *hundreds of millions* [1] of generalized "factoids"—simple propositions like "a person may write book" or "a computer may crash", rendered as Episodic Logic statements—from parsed natural language texts. 80% of factoids presented to a crowd were deemed usable. However, these factoids were atomic; as schemas have extremely rich, hierarchical structure, their extraction procedures are likely to be commensurately more difficult. If the scale of knowledge is so huge, we know one thing for sure: hand-writing a corpus of schemas suitable for world understanding is unlikely to be a good time. We must turn to automation.

There has been work on both classical, symbolic approaches to schema learning, and statistical, connectionist approaches (though far more of the latter, in contemporary literature). In this chapter, I'll detail some select approaches from each of these families, setting up the concluding discussion, in Section 4 on page 51, of the potential unification of statistical and non-statistical methods as a "way forward" for schema learning.

[1] Well, nearly 200 million.

3.1 Classical Schema Learning

3.1.1 The Integrated Partial Parser (IPP)

One of the first computational applications of the theory of scripts in (Schank and Abelson, 1977)—as well as the theory of memory Schank presented in (Schank, 1982)—is the **Integrated Partial Parser** (**IPP**), first presented by Michael Lebowitz, a student of Roger Schank's, in his dissertation (Lebowitz, 1980). [2] IPP's main focus is on the integration of *episodic memory*—a term coined by Endel Tulving to describe the comprehensive memory of an event, as opposed to the explicit knowledge of declarative facts, which he termed *semantic memory* (Tulving, 1972)—into conceptual understanding and generalization processes. Schank argued, in (Schank and Abelson, 1977) as well as many future works, that episodic memory must be organized around "stereotyped" episodes—i.e., scripts—for the purposes of both efficient lookup *and* reasoning about new situations that seem to "follow" the scripts. IPP provides a concrete first computer implementation of this sort of general episodic memory, and demonstrates its potential for generalization.

Outline of the Approach

IPP organizes its episodic memory in terms of two fundamental types of unit. The first is called the **Simple Memory Organization Packet** (**S-MOP**), and is inspired by the Memory Organization Packets defined in (Schank, 1980). S-MOPs correspond to the general notion of schemas: they are packets of information representing generalized, stereotypical events (i.e., scripts). The other type of unit is the **Action Unit** (**AU**), which represents a concrete event, though not necessarily an instance of an S-MOP. AUs resemble frames in the Minsky sense: they are attribute-value structures.

N.B.: in all of the literature on IPP, the term "Action Unit" is used to describe both the specification of the AU's attributes and *any concrete instantiation of an AU specification. For the remainder of this section, I will use the terms* "AU template" *and* "AU instance" .

An example of two AU instances is shown in Figure 3.1 on the next page.

S-MOPs comprise three core sections, each of which is hierarchical in that it may be composed of other S-MOPs, or of AUs. The AUs are "atomic", and thus somewhat of an analog to the active conceptualizations in Schank's theory (see Section 2.3.1 on page 20), although they may represent much more complex actions or situations.

[2] A contemporary application of (Schank, 1982) is the **CYRUS** fact retrieval system developed by Janet Kolodner (Kolodner, 1983), which I don't describe in this paper.

```
$HIJACK
    ACTOR       = *gunman*
    PLANE       = *Portuguese 727*
    PASSENGERS  = *83 passengers*

GS-GET-RANSOM
    ACTOR       = *gunman*
    AMOUNT      = *$10 million*
```

Figure 3.1: Two example instances of IPP's action units (AUs)

```
ACTOR        [terrorist]
PASSENGERS   [group of people]
VEHICLE      [airplane]
TO           [city or country] [prep "to"]
FROM         [city or country] [prop "from"]
```

Figure 3.2: The rules for filling the roles in the `$HIJACK` AU

The three core sections of an S-MOP are:

1. **Methods**: this section represents "the way the S-MOP is carried out", and is analogous to the *track* in Schankian scripts. For example, the S-MOP `S-EXTORT` might have the AU `$HIJACK` as its method. An S-MOP only has one method.

2. **Results**: this section represents the effects made true when the event described by an S-MOP is complete. Example results for `S-EXTORT` are the AUs `GS-RELEASE-HOSTAGES` and `GS-GET-RANSOM`.

3. **Scenes**: this section represents events that "often occur" in instances of the S-MOP; it is analogous to the scenes in Schankian scripts. Scenes are an optional component of an S-MOP: they are not required to understand the nature and effects of the S-MOP, but can provide additional information on it. Example scenes for `S-EXTORT` are the AUs `G$-NEGOTIATE` and `G$-SIEGE`.

Story Parsing

To understand stories, IPP must convert them into its memory representation. They analyze the story with a syntactic parser, which identifies AUs by keywords, and then makes use of more keywords, as well as syntactic features, to fill their roles. An example of a rule specification for filling the roles of the `$HIJACK` AU is shown in Figure 3.2 on the previous page. When processing the story `A JORDANIAN GUNMAN HIJACKED A KUWAITI JETLINER WITH 112 PASSENGERS ABOARD THURSDAY AND FORCED IT TO FLY TO BAHRAIN`, the word **`HIJACKED`** triggers an instantiation of the `$HIJACK` AU. The subject of the sentence is assigned to the `ACTOR` slot. "Jetliner" is recognized as an airplane, and is assigned to the "VEHICLE" slot. The means by which "Jetliner" is known to be an airplane are unclear, but alluded to in various sections of the dissertation: words can store pointers to other words as synonyms, and things like common names can be recognized to satisfy a "person predicate"; in any case, this lexical knowledge is hardcoded into the system.

S-MOPs may be instantiated by lexical "triggers" in similar ways, and their instantiation triggers the attempted instantiation of their components, which are either more S-MOPs or AUs. There are several complex heuristics for attaching the AUs to S-MOPs. I won't detail the specifics of these rules here; for our purposes, it suffices to note that these rules exploit the hierarchical nature of composed S-MOPs, and the shared role fillers between them and their children, to identify the children in a "top-down" manner, or to attach previous "orphan" instances to their relevant S-MOPs.

Generalization

Instantiated S-MOPs are generalized into something called **spec-MOPs** (specialized S-MOPs), which represent the general class shared by several concrete instances of an S-MOP. For example, one of their results was the `ITALY-KID-GEN` spec-MOP, derived from several `S-EXTORT` instances of stories about kidnappings in Italy. The spec-MOP generalization algorithm works on the basis of the storage system used to index S-MOPs, and their instances, by similarity.

S-MOP instances are stored in a structure called a *discrimination net* (Charniak et al., 1980), a branching graph of predicates that classifies items by successively applying the predicates to move them along the path to their class. It can be thought of as a generalization, to directed acyclic graphs and n-ary branch orders, of a decision tree. S-MOP instances are indexed by adding them to this structure as terminal leaves, with the predicate branches given by tests for their features. The logarithmic depth of a balanced discrimination net enables efficient retrieval of S-MOP instances, and thus of the S-MOPs they instantiate: few features need to be tested for retrieval.

To keep the number of leaves small and ensure efficient balancing, leaf S-MOP instances are generalized based on a similarity heuristic. The heuristic used in the original system was "share at least 4 features" when generalizing S-MOP instances into a spec-MOP, or "share at least 2 features" when generalizing spec-MOP instances into another, even more specific spec-MOP. However, the author suggests that better heuristics are admitted and encouraged. The new spec-MOP template is created by removing the shared features, constraining them to their shared value, as can be seen in the generalization of `S-EXTORT` instances to `ITALY-KID-GEN` based on, for example, the shared location of Italy.

Implications for the Natural Emergence of Schemas

It is extremely cool[3] that IPP's generalization procedure can be seen as a "by-product" of a well-known procedure for building balanced decision trees. This illustrates how the capacity for story generalization—and, later, even analogical reasoning—can *emerge* from the somewhat more mundane, logistical-sounding task of optimizing the recall of specific episodes from a database. This property of IPP provides compelling computational evidence for this statement from (Schank and Abelson, 1977): *"As an economy measure in the storage of episodes, when enough of them are alike they are remembered in terms of a standard generalized episode which we will call a script."*.

IPP's storage-derived generalization procedure is not the only support for that statement: in a study of recollections made by former White House Counsel John Dean, Ulric Neisser compared Dean's statements with factual recordings of the events Dean claimed to be describing, and found that Dean's recollections shared many properties of the actual events, but were not specifically identical to any of them (Neisser, 1981). Neisser called this "repisodic" memory, and offhandedly referred to the case study as possible evidence for "theoretical ideas about 'story grammars' and 'schemata'".

Construing Neisser's study of John Dean as evidence for schemas-as-memory in the human brain also supports an interpretation of IPP's generalization procedure as evidence for the natural emergence of schemas from efficient episodic retrieval techniques. Evidence has been presented suggesting that potential neural circuitry for episodic memory evolved slowly, in pieces, across a long timeline of ancestors and species (Allen and Fortin, 2013). This suggests that each "step along the way" to our own episodic memory was locally beneficial to the survival of a species, and thus that schema-like representations in the human brain could be the result of compounded

[3]I had originally written "particularly interesting" here, but I just can't write something that restrained to describe the reaction I had when I saw this connection.

“micro-improvements” to our memory systems—much like IPP’s generalized S-MOPs are a consequence of its efficient episode indexing system. Of course, I’m not an evolutionary psychologist, so I’ll put this line of speculation to rest here.

Discussion

My excessive fawning *supra* over possible implications of IPP’s generalization scheme is not to say that it is necessarily a scalable system, as-is, for the practical task of schema learning; in fact, I believe it has several practical shortcomings, mostly related to the amount of hand-engineered information needed to “bootstrap” it. Note that IPP can only generate *spec-MOPs*: *specifications* of known S-MOPs. This means that the initial set of S-MOPs must, in principle, conceptually “cover” all schemas one wishes to generate.

This scalability concern also holds for the Action Unit definitions—including the rules used to match tokens in a syntactic parse to AU roles. Converting a syntactic parse tree into a cognitive action representation requires a magnitude of semantic reasoning and knowledge about objects, and the ontologies in which those objects are embedded, that is intractable to embed by hand in the IPP’s format.

Many of these bootstrapping issues have enjoyed the attention of reasonably successful contemporary research: **(1)** “fuzzy” semantic word representations have seen rapid improvement as distributional word embeddings, e.g. (Mikolov et al., 2013), as well as in more comprehensive *language models* (distributions of possible sentences), e.g. ELMo (Peters et al., 2018); **(2)** the problem of textual semantic role labeling given a corpus of known frames needs no longer rely on syntactic heuristics alone, as it does in IPP (see Section 2.4 on page 26 for examples of automatic SRL systems); and **(3)** “world knowledge” about objects, of the kinds that IPP requires for role filling, is beginning to accumulate, in both hyponymy ontologies (for determining “is-a” relationships), e.g. WordNet’s hypernym graph (Miller, 1995), and in corpora of what Pustejovsky (1991) calls “telic” knowledge (for determining “what something is *good for*”) (Yamada and Baldwin, 2004).

But IPP assumes one thing a mass schema generator cannot assume: the existence of a large number of initial schemas. We’ll now look at later approaches to schema generation, and examine whether they are better suited to unsupervised schema learning.

3.1.2 GENESIS

Most approaches to schema acquisition have used statistical extraction models to cope with the vast amount of natural language text needed for automatic generalization. However, the GENESIS system, first presented in Ray Mooney’s dissertation (Mooney, 1988), attacks the schema learning problem symbolically, generalizing causal, structured schemas without the need for a huge number of examples.

GENESIS generalizes schemas [4] from single narratives using *explanation-based learning* (Mitchell et al., 1986), a symbolic learning method which uses a set of symbolic rules to "explain" observed actions, thereby offering clues for what is generalizable without compromising the underlying teleology of the narrative. [5]

Outline of the Approach

At its core, GENESIS is a cyclic combination of classical planning and pre-existing schema matching techniques. A schema can only be used to interpret a teleological sequence of actions (i.e. a *plan*) if that schema is already known. Absent any relevant schemas, GENESIS resorts to classical situation planning techniques, manipulating an internal logical representation of a story to generate a plan, composed of known operations, that aligns with the character's actions in the story. If such a story-explaining plan is found, it is then analyzed, and novel action sequences necessary to achieve important goals are generalized and stored as schemas.

The technique of storing plans to enable "sub-sequence" analysis is not new: the STRIPS problem solver system stored generalized plans in just such a way using *triangle tables*: data structures that allowed efficient determination of the next applicable step of a multi-step plan given an arbitrary world state (Fikes et al., 1972). In fact, this similarity is the result of direct inspiration: GENESIS's internal logical representation and planning algorithm are modified versions of those used in STRIPS.

Knowledge Representation

GENESIS first converts stories into an internal knowledge representation, whose ontology comprises *object*, *attribute*, *state*, and *action* classes. *Objects* are atomic, symbolic representations of entities in the planning domain, e.g. *sunrise* or *Jacob*. Note that these are not necessarily *physical* entities, but rather, the entire "domain" of the logical system in which the planning inferences occur. *Attributes* are "rules" about objects that are unaffected by actions. *States* are conjunctions of temporal propositions representing the world at a given time. *Actions* are temporal predicates endowed with axioms whereby associated *preconditions* are implied by the action

[4] I am hesitant, unlike these authors, to use the Greek pluralization rule for the word *schema*.

[5] As (Mooney, 1990) notes, explanation-based learning can be viewed in a dichotomy with *similarity-based learning*, which depends on a large number of examples and a similarity metric to learn generalized structures—neural networks and other statistical schema learning approaches, to be detailed in further subsections of this chapter, are examples of this latter end of the dichotomy.

Arrest(?a,?b,?d(?b))			
Constraints	**Preconds**	**Motivations**	**Postconds**

Isa(?a,Character)
Isa(?b,Character)
¬Equal(?a,?b)
Illegal(?d(?b))
	At(?b,?l)
	At(?a,?l)
		Believe(?a,?d(?b))
		Occupation(?a,cop)
			Arrested(?b,?d(?b),?a)

Figure 3.3: A table from (Mooney, 1990) illustrating an *Arrest* example action in the GENESIS representation. Constraints are atemporal and checked against state-independent attributes, whereas preconditions are temporal and state-dependent.

predicate's truth at some time, associated *postconditions* are implied at some time after the action, and associated *motivations* for the action are implied of relevant entities by the action's truth at some time.

The "frame problem" (McCarthy and Hayes, 1969) of axiomatizing the temporal persistence of these predicates is handled by the "STRIPS assumption" (Fikes et al., 1972): stated simply, propositions may be added to or deleted from the working knowledge base at some timestep [6] if and only if an action occurred with those postconditions at that timestep.

The Planning Process

Because GENESIS is a schema learning system, it deals primarily with *action*-type knowledge; an example action, taken from (Mooney, 1990), is shown in Figure 3.3. If GENESIS encounters an action in a story that does not match an existing (or instantiable) schema, it will attempt to deduce a teleology, via planning, to explain the action with past actions. The unknown action's preconditions are taken as goals, and a backward-chaining inference process begins on the Horn clauses in the knowledge base at the action's time: the system must prove that the preconditions and motivations of the unknown action were made true by the postconditions of past actions.

The authors note that motivations are seldom made explicit, and so they manually encode prior *thematic goals* [7] into GENESIS. An example of a thematic goal, taken from their paper, is a police officer's desire to arrest lawbreakers: `Arrested(?b,?d,?a)` $\wedge$ `Occupation(?a, cop)` $\implies$ `ThemeGoalMet(?a, Arrested(?b, ?d, ?a))`.

[6] Under the *closed-world assumption* (Reiter, 1987b), we assume that any predicates not present in the knowledge base are false; the knowledge base is parameterized by a discrete time variable.

[7] (defined by (Schank and Abelson, 1977) as atomic goals, derived from atomic needs and wants, which require no explanation)

Schema Generalization

Once the process of achieving a goal has been explained, GENESIS uses three criteria to determine whether to generalize the explanation into a schema, in order to keep the massive number of possible generalizations to a useful minimum. These criteria are designed to encourage the generation of novel “end-to-end” schemas that one character can undertake as composite actions in order to achieve an inherently useful and interesting goal. They are:

1. The goal achieved should be a thematic goal, rather than a composite goal.
2. The explanation should not just be an instantiation of an already learned schema.
3. The top-level actions of the explanation should be undertaken by the character whose goal we’ve explained.

If an explanation meets these criteria, GENESIS “prunes” the explanation, removing irrelevant and/or overly restrictive facts. For example, a top-level explanation of an arrest might have four steps: a police officer goes to a known drug producer’s house, the police officer tricks the drug producer into letting them inside, the police officer observes the drug lab, and, finally, the police officer makes an arrest, satisfying a thematic goal. However, these actions are composite: perhaps the police officer tricks the drug producer into letting them inside by pretending to be their uncle. Pretending to have a false identity may be essential to satisfying a constraint that A only lets B in their house if A thinks B is trusted. However, the fact that uncles are trusted is not necessary in this causal graph, and may thus be pruned for generalization.

Finally, GENESIS generalizes the schema by removing unnecessary restrictions using a generalization algorithm (EGGS) given by (Mooney, 1988) [8]. Mooney gives the following example plans to help illustrate what restrictions are “necessary’:

- `GoThru(Door1,Room1,Room2)`
- `PushThru(Box1,Door1,Room2,Room1)`

With the constant-to-variable substitutions `Door1` $\leftarrow$ `?d`, `Room1` $\leftarrow$ `?r1`, `Room2` $\leftarrow$ `?r2`, and `Door1` $\leftarrow$ `?b`, the resulting plan is *too specific*: the preconditions of the operators, as defined earlier in that paper, do not require that the robot push the box into the same room it started in. However, consider this plan:

[8](which is itself derived from the STRIPS generalization algorithm given by (Fikes et al., 1972))

- `GoThru(Door1,Room1,Room2)`
- `PushFromRoom2ToRoom1(Box1)`

The conditions for `PushFromRoom2ToRoom1` do, in fact, require that the robot start in room 2 and end in room 1. So, simply replacing the constants with variables is *too general*: the rooms *must* be Room1 and Room2. So, the algorithm must replace constants by variables whenever possible, subject to the constraint that the pre- and post-conditions of the relevant operators are respected.

It turns out that the algorithm for doing this involves unifying all possible pairs well-formed formulas (WFFs) in the logical representation using the unification algorithm given by (Robinson, 1965). The resulting unifications (which can be thought of as bindings, or substitution maps) are then applied to the entire explanation structure. Once this is done, GENESIS has produced a generalized schema for doing "the kind of thing" a character did in the story to achieve their goal.

Example Generalization

The authors present a "case study" of GENESIS wherein a "learning narrative", which described a schema instance in great detail, was presented to the system, followed by a "test narrative", which described another instance of the same schema but left out many important actions. The system was able to apply the generalized schema it obtained from the learning narrative to the test narrative, constructing an explanation from very sparse information.

To show that the test narrative was insufficient for question-answering purposes, the authors published the following interaction with GENESIS on the basis of their input (the test narrative). This is shown in Figure 3.4 on the following page. After presenting it with the learning narrative (Figure 3.5 on page 41), it generalized a schema, and applied that schema to the test narrative to fill in the story details (Figure 3.6 on page 42).

Discussion

The generalized schema shown in figure 3.5 on page 41 seems quite comprehensive: in addition to "type" predicates on its entities, e.g. `?b9 is a person`, it also relates those entities via actions, e.g. `?s9 goes from ?s4 to ?l4`. Furthermore, the actions are not simply subject-verb-object triples: they specify pre- and post-conditions, as

[9] [sic]

[10] Also [sic], and somewhat suspicious—why would this inconsistency with "suggests" on the previous line occur in what is ostensibly the literal output?

```
Input: Alice was at a corner wearing tight blue jeans.
   Stan told Alice if she had sex with him then he would give her $75.
   Stan went to jail.

Ready for questions:

> Paraphrase.
Stan solicited Alice's sexual favors for $75. Stan was put in a jail.

> Why did Stan tell Alice if she had sex with him then he would give
   her money?
Because Stan believed that Alice was a prostitute and because Stan
   needed to have sex.

> Who arrested Stan?
Answer unknown.
```

Figure 3.4: An unproductive Q&A session with GENESIS after it had only seen the sparse test narrative.

well as motivations, all of which contribute to the complexity and goal-orientedness of the final schema. However, I do want to note some important concerns related to the scalability of its schema learning.

The stories that GENESIS learned from were artificial, containing very little extraneous detail and really instantiating only one substantial schema. Real stories often have many simultaneously active schemas, as well as details and stylistic artifacts that complicate schema recognition. GENESIS's causal explanation-based generalization does seemingly deal with some of these issues, but its efficacy on real, "dirty" stories has not been demonstrated at scale.

The causal explanations that GENESIS relies on for schema construction also rely on a significant corpus of hand-generated actions, which specify preconditions, postconditions, and goals. The STRIPS-style, first-order predicate logic these actions are written in also lacks the expressive power of natural language. This all limits GENESIS rather severely: without a complete understanding of the world-meaning of every action verb it encounters, GENESIS is unable to generalize schemas from natural language. This sort of complete understanding would be very difficult to give a schema learning system *a priori*.

GENESIS (subsection 3.1.2 on page 35) was a rather compelling attack on the task of schema learning, but I do think that these issues ultimately lead to poor scalability.

```
Input: Jane is a policewoman.
   She dressed in a short red skirt and went to a corner.
   Bob approached the corner and told her if she had sex with
      him then he would give her $50.
   Jane arrested Bob for soliciting.
   Bob is Mary's husband and he told her that Jane entrapped him.

Thematic goal achieved: Jane is happy that Bob is under arrest for
soliciting[9] Jane's sexual favors for the $50.
Explanation suitable for generalization.
Pruning...
Generalizing...
Packaging...
Creating New Schema: (DressSolicitArrest ?b9 ?c2 ?s2 ?s4 ?I4 ?a9 ?c3)
?b9 is not equal to ?a9.
?b9 is a person. ?c2 is an apparel. ?b9 has ?c2. ?b9 puts on ?c2.
?s2 is a location. ?I4 is a corner. ?b9 is at ?s2.
?b9 goes from ?s2 to ?14. ?a9 is a person. ?s4 is a location.
?a9 is at ?s4. ?s9 goes from ?s4 to ?14. ?c3 is money.
?a9 needs to have sex. ?c2 is sexy.
?a9 solicits ?'b9s sexual favors for ?c3. ?b9 is a police officer.
?b9 arrests ?a9 for soliciting ?'b9s sexual favors for ?c3.
Having: ?b9 put on ?c2. suggests DressSolicitArrest
Having: ?a9 solicited ?b9's sexual favors for ?c3. suggest[10]
   DressSolicitArrest
Having: ?b9 arrested ?a9 for soliciting ?'b9s sexual favors for ?c3,
   suggest
DressSolicitArrest
Unknown word '?b9 entrap ?'a9 refers to DressSolicitArrest
Ready for questions:
> Paraphrase.
Jane had a skirt. Jane put on a skirt. Jane was at some place.
Jane went to a corner. Bob was at some place. Bob went to the corner.
Bob needed to have sex. The skirt was sexy.
Bob solicited Jane's sexual favors for $50. Jane was a police officer.
Jane arrested Bob for soliciting 'Janes sexual favors for the $50.
> Why did Bob go to the corner?
So Bob could solicit 'Janes sexual favors for the $50.
> Why did Bob solicit 'Janes sexual favors?
Because Bob believed that Jane was a prostitute and because Bob
needed to have sex. .
> Why did Jane arrest Bob?
Because Jane believed that Bob solicited 'Janes sexual favors for the
$50 and because Jane was a police officer.
```

Figure 3.5: A schema derivation and Q&A session with GENESIS after it had seen the learning narrative.

```
Input: Alice was at a corner wearing tight blue jeans.
Stan told Alice if she had sex with him then he would give her $75.
Stan went to jail.
Thematic goal achieved: Alice is happy that Stan is under arrest for
   soliciting 'Alices sexual favors for the $75.
Ready for questions:
> Paraphrase.
Alice had a pair of jeans. Alice put on the pair of jeans.
Alice was at some place. Alice went to a corner.
Stan was at some place. Stan went to the corner.
Stan needed to have sex. The pair of jeans were sexy.
Stan solicited Alice's sexual favors for $75.
Alice was a police officer.
Alice arrested Stan for soliciting 'Alices sexual favors for the $75.
> Who arrested Stan?
Alice arrested Stan for soliciting 'Alices sexual favors for the $75.
> Why did Alice arrest Stan?
Because Alice was a police officer and because Alice believed that
Stan solicited 'Alices sexual favors for the $75.
```

Figure 3.6: A much more productive Q&A session regarding the test narrative with GENESIS after it had formed a schema from the learning narrative.

3.2 Statistical Schema Learning

Shortly after GENESIS, there was a cultural shift in approaches schema learning: probabilistic models of event sequences dominated, and classical explanation-based learning fell out of vogue. The reasons for this seem to be twofold: (1) "connectionist" machine learning techniques like neural networks were becoming viable again due to the introduction of the backpropagation training algorithm by (Rumelhart et al., 1986); and, (2) as Karl Pichotta points out in his Ph.D. thesis (Pichotta, 2017), natural language is often ambiguous, and non-probabilistic, symbolic models of events do not easily support the kind of probabilistic inferences we make when interpreting ambiguous texts.

This section will explore some examples of the statistical kind of schema learning that has come to dominate the literature since the mid-80s, then briefly discuss the advantages and disadvantages of this sort of approach.

3.2.1 Unsupervised Learning of Narrative Event Chains

One of the earlier influential papers on statistical script learning is *Unsupervised Learning of Narrative Event Chains* (Chambers and Jurafsky, 2008). This paper defined, and sought to learn, *narrative event chains*: a set of events with a mutual "protagonist" and a partial temporal order. The events in the chains were modeled as 2-tuples of verbs and entities.

Event Model

The authors defined events as 2-tuples of of the form *(event, dependency)*, where *event* is a verb, and *dependency* is an entity—the protagonist—as either the *subject* or *object* of the verb. This latter element of the tuple is called a *typed dependency.*

The Learning Method

The authors use a three-stage approach to learning event chains:

1. First, they perform **narrative event induction** on the syntactically parsed text corpus: events are extracted from the parsed clauses. Then, they calculate, using approximate pointwise mutual information (PMI), a pairwise relationship metric[11] for the set of events, based on the co-occurrence of shared grammatical arguments—the hypothesis here is that shared arguments imply co-occurrence of the events in a narrative chain. The pairwise event co-occurrence scores can then be combined to find the next most likely event in a chain given all the events in the chain so far by maximizing $\max_{j:0<j<m} \sum_{i=0}^{n} pmi(e_i, f_j)$[12], where e_i is the i-th event in a chain of n events, candidate event f_j is the j-th event in the training corpus of m events, and $pmi(e_i, f_j)$ is an approximation of the pointwise mutual information of e_i and f_j.[13]

   The approximate pointwise mutual information of the two events is defined by

$$pmi(e_i, f_j) = \log \frac{P(e_i, f_j)}{P(e_i)P(f_j)}$$

   The marginal event probabilities in the denominator of the PMI approximation are given simply by the observed frequencies of those events in the training

[11]More specifically: the pointwise mutual information of the outcomes of two random variables—here, event representations drawn from a distribution of observed ones—is a quantification of the effect of statistical dependence in the co-incidence of both outcomes.

[12]N.B.: maximizing this quantity is equivalent to picking the event f_j from the training corpus with the most occurrences of shared co-referring entity arguments with e_i, as can be inferred from the definitions to follow.

[13]N.B.: they only need to calculate pointwise mutual information here, rather than the full distributional mutual information, because of their "Markov assumption" that an event's likelihood is predicted only by the last event.

corpus. The approximated pairwise joint event distribution in the numerator is defined by

$$P(e_i, f_j) = \frac{C(e_i, f_j)}{\sum_{e1} \sum_{e2} C(e1, e2)}$$

where, for any pair of events $e1$ and $e2$ having respective words $w1$ and $w2$ and respective dependencies $d1$ and $d2$, $C(e1, e2)$ is the number of times the dependencies $d1$ and $d2$ were filled by co-referring entities. Put more simply: this PMI approximation assumes that any statistical dependence in the joint distribution is due to shared argument entities, and thus quantifies the effect of entity co-reference in event co-occurrence. Put *even more* simply: an event is more likely to come next if it shares arguments with the last event.[14]

2. After the narrative event chain distribution is induced, they compute **partial temporal ordering** by using support vector machines (SVMs) to predict the presence of a *before* relation of two events using syntactic information, including verb tense and grammatical aspect. The training data for the SVM was taken from the Timebank corpus of event relations (Pustejovsky et al., 2003), whose relations were simplified down into the *before* relation used by this paper, and an *other* relation.

3. Finally, they use an agglomerative clustering algorithm—in which events begin with their own clusters, which are then merged based on a similarity metric—to **identify the event chains**. The clustering algorithm uses the pointwise mutual information from the first step as a similarity score. Once the chains are constructed, their temporal ordering is induced by the SVM model in the second step.

Experimental Setup

The authors used the Gigaword corpus (Parker et al., 2011) as a document source for their experiments. The protagonist for each document is the entity with the highest number of mentions—they used the OpenNLP co-reference engine (Baldridge, 2005) for entity recognition. Evaluation was performed using a metric the authors introduced in this paper: the **narrative cloze** task, an extension to narratives of the cloze test defined by (Taylor, 1953), in which humans were tasked with filling in missing words in texts. In the narrative cloze test, the system fills missing event tuples into narrative chains.

[14]Put *comedically* simply: “yeah, that makes sense, I recognize that guy from before”.

Results

They defined the baseline for their evaluation as a version of their model that did not use the protagonist, or its relation to each event as a subject or object, in making predictions. As the baseline could then only produce verb event chains, rather than 2-tuple event chains, they then modified their experimental model to use the protagonist co-reference information during training, but to omit it in the output chains. It is unclear to me, from reading the paper, where the "golden" event data for each document they used to compute accuracy came from. They reported a nearly **37%** accuracy improvement relative to the baseline, demonstrating the efficacy of the typed protagonist information. Removing the OpenNLP co-reference resolution step caused accuracy to drop by **5.7%**.

Discussion

This was a hugely influential paper in statistical script learning, and while the authors themselves note that the event sequences are a far cry from the richness of Schank & Abelson's scripts, as defined in (Schank and Abelson, 1977), they did pave the way for more expressive event models in future approaches, such the 5-tuples in (Pichotta and Mooney, 2016), which I describe in Section 3.2.2.

3.2.2 Learning statistical scripts with LSTM recurrent neural networks

Karl Pichotta and Raymond Mooney's paper *Learning statistical scripts with LSTM recurrent neural networks* (Pichotta and Mooney, 2016) was the first application of long short-term memory (LSTM) neural networks to the task of script learning. LSTM units, introduced by (Hochreiter and Schmidhuber, 1997), can be thought of as memory cells that learn when to store, and when to forget, input items. Recurrent neural networks—that is, neural networks whose neuron connection graphs contain a cycle–enhanced with LSTM units are extremely powerful sequence learners, and have been applied to such diverse tasks as handwriting recognition (Graves and Schmidhuber, 2009), machine translation (Sutskever et al., 2014), and music generation (Eck and Schmidhuber, 2002). Hoping that the memory cells would help learn long-range narrative structure in event sequences, Pichotta and Mooney applied them here to script learning.

Event Model

Pichotta and Mooney extend the 2-tuple event representation from (Chambers and Jurafsky, 2008) into a 5-tuple representation of the form (v, e_s, e_o, e_p, p), where v is a lemmatized verb (i.e. the naked dictionary-form verb); three nouns e_s, e_o, and e_p are related to the event as the subject, object, and prepositional attachment of v, respectively; and p is the preposition that attaches e_p to v. Each element of the tuple is known as an *event component*, and, excepting v, may be *null* (i.e. unspecified). (Note that e_p is unspecified if and only if p is unspecified; they are coupled event components.)

The authors define two types of script model from these event tuples. **Noun models** predict the verb, noun, and preposition lemmas of events in a sequence, as a generative model. **Entity models** predict almost the same sequences, but substitute *entity IDs* for nouns. Entity IDs are unique integers representing entities with multiple mentions in the text, and are derived from a co-reference analysis. They note that prior script learning systems, e.g. (Chambers and Jurafsky, 2008), had all been entity models.

Given these script model definitions, they define four separate architectures, each trained independently:

1. **LSTM-noun-noun**: predicts argument nouns given argument nouns.
2. **LSTM-ent-ent**: predicts entity IDs given entity IDs.
3. **LSTM-both-noun**: predicts argument nouns given both argument nouns and entity IDs.
4. **LSTM-both-ent**: predicts entity IDs given both argument nouns and entity IDs.

The Learning Method

Because LSTMs make their sequential predictions "one step at a time", each tuple must take five timesteps to predict. Each timestep's input comprises several one-hot vectors (i.e. vectors of all 0s, except for one 1), which are assigned continuous distributional representations, a.k.a. *embeddings* (similarly to the one-hot vocabulary vectors in (Mikolov et al., 2013)). The first one-hot vector gives the current tuple index, ranging from 0 to 5; this modular sequence is easy for the LSTM to learn, and ensures that the timesteps output valid groups of 5 tuple elements. The second one-hot vector represents the vocabulary word at that index. Finally, the third one-hot vector—present in all architectures except for **LSTM-noun-noun**—represents the entity ID at the current timestep.

The embeddings of these vectors are given as input to the LSTM network, which produces an output vector of the same length as the vocabulary. The vector is normalized into a probability distribution by a softmax function. The network is trained by back-propagating an error value calculated with the *cross-entropy method* (Rubinstein and Kroese, 2013), in which the cross-entropy—roughly, the average number of bits needed to encode an outcome from some distribution Q, weighted by the probability of that outcome in another distribution P—between the output distribution and the target distribution is iteratively minimized for each training sample. Entity models also produce a probability distribution over the next entity ID in the same way. Likely tuples must be constructed from the most likely sequences of samples from five consecutive word and entity distributions; finding the most likely such sequence is intractable, as there are $|V|$ entries in each probability distribution (where V is the vocabulary set), for a total search time bounded by $O(5^{|V|})$. So, for a more tractable approximation, likely sequences of 5 observations are generated using a five-step beam search.

Narrative Cloze Experiment

The first evaluation was performed on the narrative cloze model introduced by (Chambers and Jurafsky, 2008) (and described here in Section 3.2.1 on page 44). Because they defined a new event representation, the authors had to define their baseline against which to evaluate their architectures. They actually defined four baselines:

1. The **unigram model** ignores the document and samples events by their frequency in the training set. They note that (Pichotta and Mooney, 2014) showed this baseline to be "surprisingly competitive". They created a noun unigram model and an entity unigram model to cover all of their architectures.

2. The **all-bigram model** constructs statistics of event "skip" bigrams, where up to two events were allowed to "intervene" in between the events in the bigram. This is called a 2-skip bigram model, and is an instance of the more general skip n-gram model developed for script learning systems by (Jans et al., 2012). The 2-skip bigram model models the probability of an event in the sequence given the previous event in the sequence, and can be thought of as a Markov approximation of the full joint distribution. To generate an inference for some position t in the sequence, the authors maximize the objective function

$$S(a) = \sum_{i=0}^{t-1} \log P(a|s_i)$$

where s_i represents the i-th event of the sequence, and $P(b|a)$ is the probability of event b given the previous event a, given by the 2-skip bigram model.

3. The **rewritten all-bigram model** is a version of the all-bigram model with different behavior for co-occurring events, that is, events with co-occurring entities. For all co-occurring events a and b in the training process, the training data is augmented with all possible event pairs equivalent to a and b except for some subset of the shared entity IDs being re-written as other entity IDs. The intuition here, the authors say, is that co-occurrence relationships between events are likely to be *generalizable* to other entities. Inferences are generated by maximizing $S(a)$, as they were in the all-bigram model.

4. The **2D rewritten all-bigram model** is like the rewritten all-bigram model, except it generates inferences by maximizing the objective function

$$S_2(a) = \sum_{i=0}^{t-1} \log P(a|s_i) + \sum_{j=t+1}^{l} \log P(s_j|a)$$

   where l is the sequence length. (Note that the first term of $S_2(a)$ is the definition of $S(a)$ from the all-bigram model.) The idea here is to incorporate not just the probability of an event following some event s_i, as in $S(a)$, but also the probability of that event *preceding* some event s_j.

The authors found **LSTM-both-noun** to be the best noun model, and **LSTM-both-end** to be the best entity model, indicating that receiving noun and entity information together gives the strongest performance. The full table of their results is not reproduced here, as it is not particularly useful for our discussion; it can be found in (Pichotta and Mooney, 2016).

Experimental Setup for Mechanical Turk

Citing the inherent difficulty of the narrative cloze task, the authors also evaluated the likelihoods of their event inferences using annotators on Amazon Mechanical Turk. They presented the annotators with English story snippets and five examples of event inferences, four of which were generated by the model being evaluated, and one of which was always generated by a randomly chosen event from the set of 10,000 most frequent events. The annotators rated the likelihoods of each of the five inferences on a scale from 0 ("Nonsense") to 5 ("Very Likely"). They collected three annotator ratings for each event inference. Inferences were generated for each of four systems: **LSTM-both-noun**, **All-bigram-noun**, **LSTM-both-ent**, and **All-bigram-ent**. They calculated a "filtered" overall score for each model by discarding ratings from annotators whose average score for the randomly chosen event was greater than 1 ("Very Unlikely/Irrelevant").

The authors found that both LSTM models outperformed both bigram models, with filtered average ratings of 3.67 for **LSTM-both-noun** and 3.08 for **LSTM-both-ent**. The average filtered scores for the baseline models were 2.21 for **All-bigram-noun** and 2.87 for **All-bigram-ent**. The filtered score for the random events was 0.87.

Discussion

Pichotta and Mooney made two notable contributions here: (1) they presented a richer model than (Chambers and Jurafsky, 2008) by both extending the event representation and applying script learning techniques suggested by (Jans et al., 2012); and, (2) they demonstrated the effectiveness of LSTMs in learning script sequences. I've represented some examples of learned sequences in Figure 3.7 on the next page.

Though the script sequences in that figure appear to tell stories, it's important to note that important semantic information has been introduced by the authors into the grammar of the interpretations. For example, the event tuple bigram `(capture, squad, villager, ., .) (give, inhabitant, group, ., .)` was interpreted as a continuous sentence, but "give" is a multi-argument verb: an equally valid interpretation of the latter event tuple could be "The inhabitants gave a group", with "group" filling in for the object-given argument rather than the recipient argument. Of course, one could imagine solving such ambiguities using semantic role disambiguation tools, making use of semantic corpora such as FrameNet (Section 2.4 on page 26), but the fundamental issue of the restrictive event representation would remain.

However, despite the limited event representation, the authors have shown that LSTMs can learn interesting event structures from large text corpora, and one can imagine several uses and extensions. Perhaps using a conceptual hierarchy to generalize entities at the word-level could help create more general scripts from lone story examples? Or maybe the generated event sequences could be used as suggestions for an explanation-based learning system with a richer event representation to elaborate on. This system invites many interesting extensions.

Generated event tuples	**Authors' interpretations**
`(pass, route, creek, north, in)`	The route passes the creek in the North
`(traverse, it, river, south, to)`	It traverses the river to the South
`(issue, ., recommendation, government, from)`	A recommendation was issued from the government
`(guarantee, ., regulation, ., .)`	Regulations were guaranteed
`(administer, agency, program, ., .)`	The Agency administered the program
`(post, ., correction, website, through)`	A correction was posted through a website
`(ensure, standard, ., ., .)`	Standards were ensured
`(assess, ., transparency, ., .)`	Transparency was assessed
`((establish, ., ., citizen, by)`	Established by citizens, ...
`(end, ., liberation, ., .)`	... the liberation was ended
`(kill, ., man, ., .)`	A man was killed
`(rebuild, ., camp, initiative, on)`	The camp was rebuilt on an initiative
`(capture, squad, villager, ., .)`	A squad captured a villager ...
`(give, inhabitant, group, ., .)`	... [which] the inhabitants had given the group

Figure 3.7: Examples of probabilistically generated scripts from the LSTM model given by (Pichotta and Mooney, 2016), also showing those authors' English interpretations of the event tuples.

Chapter 4

Towards Scalable Schema Learning

So far, this paper has examined several major approaches to schema representations and schema learning systems, noting in each case some limitations. The largest modern frame corpora, FrameNet (Section 2.4 on page 26) and the similar frame corpus PropBank (Palmer et al., 2005), rely heavily on natural language to define semantics of their roles, and as such are difficult to generate automatically. Initial approaches to schema learning (i.e. IPP in Section 3.1.1 on page 31 and GENESIS in Section 3.1.2 on page 35) suffered from a “bootstrapping problem”: they required a large amount of initial information about the real world to begin large-scale generalization of schemas from text. And modern approaches to script learning have relied on extremely limited expressivity—often representing events as subject-predicate-object triples, or slight extensions thereof—and fail to describe the often complex relationships between the “slots” of a schema.

So where do we go from here? How can we learn, at scale, a large number of rich, expressive descriptions of temporally related events and complexly related participants in those events? This section will primarily detail my group’s ideas about scalable schema learning, and with my current progress on implementing some of those ideas.

4.1 What’s needed for schema learning?

Two components are necessary for schema learning by definition: a model of schemas to learn, and an algorithm to learn them. These cannot be selected independently: as we saw in IPP (Section 3.1.1 on page 31), the knowledge representation, and the knowledge stored in it, informs even the lowest levels of the learning algorithm, i.e. the syntactic parsing process. A third component that is almost certainly necessary for practical learning is *a priori* common-sense world knowledge: unless one can implement Turing's "child machine" (Turing, 1950)—a robot that is a complete ambulatory and sensory facsimile of a human child—schemas must be learned from more indirect sources, like natural language texts, and so some corpus of knowledge is necessary to "bridge" purely lexical information with the sorts of observations children would get by virtue of their eyes and ears (e.g. vacuum cleaners are loud, loudness is unpleasant). In other words, we must "level the playing field" to give our schema learning systems similar capabilities as the most powerful learning systems we know of: human children.

As my advisor says frequently: *"Babies do not start intellectually as tabulae rasae."* Here, I'll introduce a few capabilities that our group considers to be a necessary "head start" for a scalable schema learning system—and some evidence for the presence of these capabilities in actual humans.

4.1.1 Symbolic representations

Although the debate between *connectionist* (i.e. neural, emergent) models of human cognition vs. *classical* (i.e. symbolic, logical) ones rages on, there is evidence that symbolic representations, and symbolic reasoning, exist in the human brain—regardless of the nature of its "implementation". In an fMRI study conducted at Purdue (Siskind, 2015), Jeffrey Mark Siskind's group created a simple space of sentences with 4 possible actors, 3 possible verbs (*carry, fold, leave*), 3 possible objects (*chair, shirt, tortilla*), and 2 directions (*on the left, on the right*), for a total of 72 possible sentences. They recorded videos representing each of these sentences, and took fMRI brain image sequences of human subjects watching these videos.

Using SVM classifiers, they attempted to decode brain image sequences to determine each of the video components independently (actor, verb, object, and direction), as well as the pairs, triples, and full sentences. They were able to determine all combinations of components with accuracy well above chance. Additionally, they trained fMRI classifiers to determine the same set of actions from brain activity when viewing text input *and*, separately, when viewing video input. They were able to get comparable, above-chance accuracy when decoding verbs from images of one modality *using the classifier trained on the other modality*.

Their findings strongly suggest that the brain not only uses compositional action representations (i.e., with "slots"), but that the areas in the brain that store these representations are common to both text *and* video input—in other words, the brain develops a common symbolic representation of its observations. These results strongly suggest that the brain uses symbols to store information about the world, and these symbols can be generated from many kinds of observation.

4.1.2 Slot relationships

The "slot-and-filler" structure of Minsky's frames (see Section 2.2 on page 11) has informed much work on frame-like ontologies. Modern implementations of these frame-like ontologies, such as the description logic **OWL (Web Ontology Language)** (McGuinness and Van Harmelen, 2004), have an analyzable, formal attribute-value semantics, and support slot type constraints, inheritance, composition, and many other useful forms of taxonomic reasoning. However, description logics like **OWL** do not support the expression of formal, complex *relationships* between their slots: in a frame for a children's school, for example, with slots for a principal and a vice principal, modern frames generally could not formally impose inter-slot constraints such as "the principal is the boss of the vice principal". FrameNet (Section 2.4 on page 26) does relate its slots to one another, but only informally, with natural language. Any schema representation we choose to *learn* mechanically should support these sorts of slot relationships with a semantics that can be *reasoned about* mechanically.

4.1.3 Branching, conditionals, and variables

When someone is working to cook a dish, they don't only follow a series of steps. They may have to *branch* (e.g. *if the dough looks wet, lay it on the table and wait five minutes*), execute *loops* (e.g. *check the dough every five minutes, removing it when it appears golden brown*), and manipulate named variables (e.g. *note the weight of the dough in grams; when it's done baking, allow it to cool for half as many minutes as its weight*). In this sense, the schemas that describe our behavior are quite comparable to computer programs, and their representation should facilitate these operations.

4.1.4 Time, probability, and meta-reasoning

Schemas control human behavior in the real world, and so some important aspects of that world should be easily expressible in the language as well. The real world is inherently *temporal*, and so we should be able to express relative times (e.g. *it happened before that*) as well as specified magnitudes (e.g. *it happened two weeks ago*). We also need to express *probabilities and certainties* (e.g. *he has a 50% chance of winning*, or *it is possible that it will rain tomorrow*). Finally, humans operate in a world with other self-motivated agents, including other humans, and as such must reason about the knowledge and beliefs of others, as well as knowledge transfer mechanisms such as speech. So, whatever form we choose for our schemas should facilitate *attitudinal* inferences such as *John believes that the dough is done.* As we'll see later, certain logical forms have less trouble than others expressing statements of these kinds.

4.1.5 Analog representations

Although Minsky discusses applying symbolic frames to 3D scene understanding, human beings seem to have especially strong, unconscious intuitions about certain physical and spatial properties. 3D object representations are suspected to be analog in humans, that is, in some sort of spatial correspondence with the objects themselves: the time humans take to mentally rotate 3D images is proportional to the degree of rotation (Cooper and Shepard, 1973). Additionally, object representation is suspected to occur largely in a dedicated brain area (Rosenberg et al., 2013). Spatial reasoning is an extremely important aspect of story understanding, as stories take place in a physical world. And, owing to the computational difficulty of spatial reasoning, we should try to separate these "specialist" systems from the symbolic machinery, and optimize them independently, to whatever extent we can.

4.1.6 Behavioral axioms

Even human newborns cry for food. This is notable: being hungry is unpleasant, and even in the earliest days of their lives, human beings are "pre-programmed" with desires to avoid certain displeasures, to seek certain rewards, and to pursue actions—even highly primitive, instinctive ones like crying—that help realize those desires. This kind of ingrained tendency toward goal-driven action is central to understanding one's own actions, as well as the actions of others.

4.1.7 Primitive actions

OK, (Schank, 1975) may have something of a point. There do seem to be certain primitive actions—crying, laughing, grasping, holding, eating—that infants start out with, experiment with, and compose to achieve basic goals. However, the key distinction from the primitive actions of Schank's CD theory is that these actions form a *"starter pack"*, rather than an *atomic representation*: one cannot compose all actions from what babies know how to do, and, even in the cases where one can, one still does not conceptualize complex actions like "fishing" in terms of thousands of separate grasping actions. However, some basic actions do seem to give a good head start to infants.

4.1.8 Language learning

Young humans acquire language rapidly, and language can then be used to learn nearly arbitrary information. Siskind's work, described above, suggests that our linguistic representations are split into symbols that are then used to represent entities and actions across verbal and visual modalities. Language is important for learning—especially when your goal is to learn from texts, which are more plentiful, and arguably easier to extract features from, than richer modalities such as videos.

4.2 Implementing what's needed

Imbuing a computer with any of the capabilities listed above is no small task, but learning them from scratch in a neural model seems especially daunting: how could we acquire the data needed to learn these things at the scale needed to cover the vast number of possible situations that exist? Our research group, led by Lenhart Schubert at the University of Rochester, is pursuing alternative approaches to constructing this "head start", such that schema learning can proceed from a small number of examples (relative to the data needed to train neural networks). Georgiy Platonov is pursuing computational understanding of spatial descriptions (Platonov and Schubert, 2018); Gene Kim is developing a semantic parser for mapping English to an intermediate semantic representation that directly enables certain discourse inferences while also providing a starting point for transduction into a nonindexical[1] logical language (Kim and Schubert, 2019); and together with his students, Lenhart Schubert has spent years developing that formal logical language, known as **Episodic Logic (EL)** (Hwang and Schubert, 1993), hypothesizing that the semantic types available in this language are the ones innate in human "mentalese".

As I join my advisor and colleagues to pursue the eventual goal of scalable schema acquisition, I will—and, in fact, have already begun to—make heavy use of Episodic Logic to define a schema representation and enable powerful inferences on, and generalizations of, schemas in that representation. So, before I detail our approach to the schema learning problem, and the work I've done toward that approach so far, I will briefly describe EL.

4.3 Episodic Logic (EL)

4.3.1 Why EL?

Before we define Episodic Logic, we should motivate it: why do we use EL instead of more "battle-hardened" existing logical formalisms? First-order logic (FOL)—used

[1]Details of what this means are available in (Kim and Schubert, 2019); here, I'll just say that "deindexing" includes tasks like converting the speaker-relative tense information in sentences to explicit, atemporal statements about individual episodes.

for planning and knowledge representation by the STRIPS planning system (Fikes et al., 1972), GENESIS (Mooney, 1990), and many other classical AI systems—would be a natural first choice: its syntax and semantics are simple and known by academics and practitioners alike. However, FOL is an *extensional* logic by nature, and as such does not allow for modal statements such as "I know that Gaurav loves Mariel"—the proposition "Gaurav loves Mariel" only denotes a truth value, and such a sentence would always be equivalent to "I know that `TRUE`" or "I know that `FALSE`", depending on whether Gaurav loves Mariel or not in that world.[2]

Additionally, very important to the notion of schemas is the notion of an *episode*, a.k.a. a situation, a.k.a. an event. Schemas are stereotyped patterns of situations, and so a representation of situations is necessary. FOL has traditionally tackled this with Davidsonian *events* (Davidson, 1967), where the event is a first-class domain entity passed as an extra argument to situational predicates. As we'll see below, this approach is actually quite similar to how Episodic Logic handles events, except that Davidsonian event variables can only be passed to positive, atomic predication literals; Episodic Logic's semantics allow arbitrarily complex formulas to hold in certain events.

4.3.2 What is EL?

EL is a predicate logic whose semantics are intended to model some important aspects of natural language, and of the human thought patterns that give rise to natural language. As its name implies, one of EL's most distinctive features is its use of *episode* variables to denote time-bounded "temporal" propositions, such as "Palutena is sleepy" or "I'll go running for an hour"—as well as "atemporal" propositions, which necessarily hold true eternally, like "hydrogen is an element" or $2 + 3 = 5$.

EL naturally supports other aspects of natural language as well. It is an *intensional* logic: a sentence like "Gaurav loves Mariel" can represented as not just a truth value, but as a function from possible situations to truth values called a *sentence intension*. Sentence intensions may be transmuted into individuals using *reification* operator `That`, and predicates may be similarly reified into individuals with the kind-forming operator `K`[3]; this allows abstract entities, such as *"the process of writing an area paper"*, as well as concrete entities, such as *"Lane's area paper"*, to act as predicate arguments without defining special semantics for higher-order predicates (that is, predicates with predicate arguments).

EL's episodes are associated with EL formulas primarily via the *characterizing* operator `**`. Intuitively, the EL form of sentence like "I am writing my area paper"

[2]I like to think that he does.

[3]Or `Ke` for forming kinds of events, or `Ka` for forming kinds of actions.

characterizes an episode `e`, then `e` is an "episode *of*" me writing my area paper. This is contrasted with the operator `*`; while `[`ϕ` ** e]` means that `e` is an episode *of* ϕ, `[`ϕ` * e]` only asserts that ϕ is *true in* `e`. While it is always true that `[`ϕ` ** e]` $\implies$ `[`ϕ` * e]`, the converse is not necessarily true: the sun setting could be true in an episode *of* me writing my area paper, but that does not also make it an episode *of* the sun setting.

4.3.3 Explanation of an example

As EL is intended as a representation for natural language, we often make use of primarily lexical predicates, which represent verbs, nouns, adjectives, and other words. The nature of these lexical predicates, and the rough capabilities of EL's semantics to relate them, may be best presented—for our purposes—by the explanation of some illustrative examples. Let's first look at an English sentence and its EL form, and then note its meaning and various features—as well as some features of EL that may not be covered in the example.

English: *Rivka insisted that she didn't eat chametz.*

EL: `[E1.sk before Now0], [[|Rivka| insist.v (that (some e2 [e2 at-or-before E1.sk] [(not [|Rivka| eat.v (K chametz.n)]) ** e2]))] ** E1.sk)`

Restricted quantification

The existential quantifications of `e1` and `e2`[4] illustrate the "restrictor-matrix" structure of EL quantification: if `x` is a variable, `Q` is a quantifier, and ϕ and ψ are arbitrary EL formulas, the quantification formula `[Q x :` ϕ ψ`]` predicates the truth of the "restrictor" formula ϕ whenever the "matrix" formula ψ holds true. When `Q` is $\exists$, this is equivalent to the unrestricted quantification `[`$\exists$` x (`$\phi \wedge \psi$`)]`; similarly, when `Q` is $\forall$, it is equivalent to `[`$\forall$` x (`$\phi \rightarrow \psi$`)]`. The restrictor-matrix form is only necessary for some of EL's nonstandard quantifiers, e.g. `Many` and `Few`, which cannot be reduced to an unrestricted form.

Temporal relations

The temporal relation `before` is used to relate the time of `E1.sk` to the special time constant `Now0`—which represents the time that the speaker is uttering this—and the time of `e2` to `E1.sk`, and thus also `e2` to `Now`. Such temporal relations—as well as the tacit temporal relations in their transitive closures—place the episode times within

[4]E1.sk is a Skolem constant—it has been assigned a name to remove the top-level existential quantification.

the ontological domain $\mathcal{R}_4$, representing spacetime, and connect their trajectories (see Figure 4.1 on page 61). More temporal relations exist, special functions `start-of(e)` and `end-of(e)` allow temporal predications about the instantaneous endpoints of episodes, and disjoint intervals may even form an episode; for more information on the temporal semantics of EL, please refer to (Hwang and Schubert, 1993).

Representing sentences

"Sentences" in EL are often verb predications: the heart of this one is the 2-place predicate `insist.v`, with `|Rivka|` as the first argument, and a `That`-reified sentence as the second. This syntax is just a "pretty" version that reflects the subject-verb-object nature of the English language; semantically, it would be more honest to write the predication out with the arguments all postfixed, but nobody wants to see that.

Reification and *kind*-forming

The kind-forming operator `K` is used here because `chametz.n` is a one-place predicate that determines whether its argument is is chametz[5]. So, because we don't know the specific chametz individual, we choose to make a statement about her eating a kind of chametz, which we accomplish by *reifying* the `chametz.n` predicate, rather than *Skolemizing* a variable (i.e. assigning it a concrete name, to bypass existential quantification) and attaching the predicate to it (i.e. assigning it a name). [6] In order to use a predicate as an argument to another predicate, we must first reify it as an individual.

We can reify other sorts of things, too. The reified verb phrase `(Ka (drink.v (K water.n)))` [7] is the *kind of action* of drinking water, and belongs to the class $\mathcal{K}_A$ in the EL ontology (see Figure 4.1 on page 61).

The reified sentence `(Ke (|Mary| drink.v (K water.n)))` is the *kind of event* where Mary drinks water, and belongs to the ontological class $\mathcal{K}_E$. This is semantically distinct from the other reification of this sentence, which uses the `That` operator, `(That (|Mary| drink.v (K water.n)))`, which is a *proposition*, and thus belongs to the ontological class $\mathcal{P}$. An intuition for their semantic distinction is quite easy

[5]Food containing leaven; prohibited during Pesach, a.k.a. Passover.

[6]Well, it actually provides a function from possible situations to whether its argument is food in each of those situations, but I've omitted this detail for clarity *supra*, preferring instead this footnote: an intentional extensional intension extension.

[7]What's being reified is actually not a well-formed EL predication, assuming `drink.v` is a 2-place predicate; the actor argument is missing! What's *actually* being reified here, implicitly using the `K` operator, is something like the *lambda abstraction* `(λa (a drink.v (K water.n)))`; however, even that's not quite right, as there are additional complications related to event characterization. For a full discussion of this, see page 7 of (Schubert and Hwang, 2000).

to obtain from the names of their operators: "the kind of event of Mary drinking water" is obviously different from "that Mary is drinking water".

The ** and * operators

** and * are *modal* operators: statements like `[`ϕ` ** e]` and `[`ψ` * e]` may not, in general, have their formula arguments, ϕ and ψ, replaced with other formulas that seem to have the same truth value. So, the EL formula `((|Andre| sleep.v) * e)` does *not* entail `(((|Andre| sleep.v) ∧ ( (|Fido| bark.v) ∨ ¬ (|Fido| bark.v) )) * e)`, even though it would seem to have the same truth value; in fact, the entity `|Fido|` might not have any role in the episode, and so even tautological statements involving it may have no binary truth value in `e`.

Characterization by a negative

Unlike Davidsonian events, EL episodes can be characterized by arbitrarily complex EL formulas, including negatives. Here, `e2` is an episode of Rivka *not* eating.

4.3.4 Discussion

I'll leave aside any further description of EL's features or formal semantics, as I think the analysis above has demonstrated what I intended it to: that Episodic Logic is an expressive language with rich, event-oriented semantics. In the sections to come, it will become clearer how these semantics naturally support schema-based reasoning. However, it is worth mentioning that EL is not only expressive, but that EL *inference* is *tractable*; it might seem that "the richer the model, the longer it stalls", but EPILOG, the main implementation of EL inference, has comparable performance to FOL inference systems on some standard theorem proving tasks (Morbini and Schubert, 2009). Although EPILOG is somewhat slower than state-of-the-art FOL engines on theorem proving tasks, it also provides more powerful inference rules, and can compute certain sorts of inferences whose FOL equivalents would take time to be translated before inference could even begin.

Additionally, EL is much closer to the "surface form" of natural English text than FOL, owing to its intensional predicates, flexible quantifiers, and other features. This surface similarity to natural language is quite beneficial for the task of parsing natural language text into EL. EL also has an "underspecified" form, which lacks certain features of EL—e.g. quantifier scoping, and the conversion of grammatical information, like tense, into temporal predications about episodes—but still represents the semantic type structure of a sentence. This form is called "ULF" (underspecified logical form) (Kim and Schubert, 2019), and it is a practical, intermediate target form for the mass-parsing of text corpora that is still useful for many applications—such as simple schemas. In fact, using ULF for schemas may actually help convert ULF to full EL, by incorporating the knowledge represented by the schemas into the parsing pipeline. We explore the use of ULF, and EL, for schemas in later sections.

4.4 What we talk about when we talk about "protoschemas"...

Our group feels that approaches to schema learning have frequently come from "the wrong direction"; although humans learn progressively more complex schemas on the basis of whatever cognitive features they start out with, many approaches to schema learning set out trying to abstract schemas from relatively "adult" stories. GENESIS began with a "head start" comprising action definitions such as `Arrest` and basic goal-oriented facts such as *Police officers want to arrest lawbreakers*, and set out to learn a schema for arresting would-be solicitors of prostitution by wearing sexy clothing. It is unlikely that human children possess this sort of knowledge, and as such could not hope to gain much from those stories. To our minds, if a schema learning system is meant to model the acquisition of schemas by human children, then the sorts of stories it is first given should, by and large, be understandable with minimal initial world knowledge.

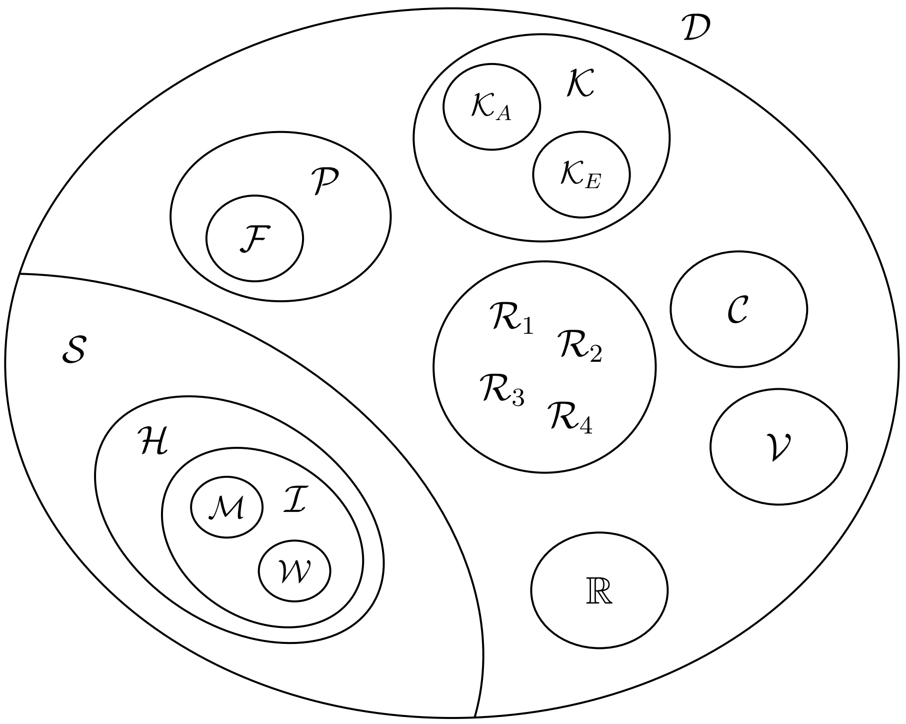
Inspired by the sorts of initial cognitive features discussed in Section 4.1 on page 51, we propose a schema learning architecture with four main components:

1. **An English-to-EL semantic parser** for obtaining canonical, semantic representations of stories.

2. **An initial set of factual knowledge** about everyday objects, their properties, their taxonomies, their associated lexical items (i.e. what they're called), their associated 3D models, and "telic" information about them (i.e. what they're good for).

3. An initial set of **protoschemas**: simple schemas about the sorts of innate behaviors and actions discussed in Section 4.1.6 on page 54 and Section 4.1.7 on page 54. Protoschemas would be goal-oriented, but rather than describing e.g. that police officers want to arrest criminals, they would describe more basic, abstract behavioral patterns such as *do an action to receive a reward* and *grasp an object to enable actions on that object.*

Episodic Logic, which we discussed in Section 4.3 on page 55, suits schemas quite well—as we'll see in the rest of this section—enabling their expressive, powerful rep-



$\mathcal{D}$	The full *domain* of EL individuals; the union of all following sets
$\mathcal{S}$	Possible situations, a.k.a. *episodes*
$\mathcal{H}$	Informationally-maximal episodes, a.k.a. *exhaustive situations*
$\mathcal{I}$	Spatially-maximal exhaustive situations, a.k.a. *possible times*, a.k.a intervals
$\mathcal{W}$	Spatiotemporally-maximal intervals, a.k.a. *possible worlds*
$\mathcal{M}$	Spatially-maximal, temporally-minimal intervals, a.k.a. *moments of time*
$\mathcal{P}$	*Propositions*
$\mathcal{F}$	Consistent propositions, a.k.a. *facts*
$\mathcal{K}$	*Kinds of individuals*
$\mathcal{K}_A$	*Kinds of actions*
$\mathcal{K}_E$	*Kinds of episodes*
$\mathcal{C}$	*Collections*
$\mathcal{V}$	*n-tuples, $n \geq 2$*
$\mathbb{R}$	*The real numbers*
$\mathcal{R}_{1\leq n\leq 4}$	*n-dimensional regions*, containing subsets of $\mathbb{R}_n$ Note that $\mathcal{R}_4$, which represents space-time regions, contains some unconnected space-time trajectories; these trajectories describe the time and place of episodes.

Figure 4.1: This represents the EL ontology. I include only some brief definitions of the notable subsets; for more formal detail, please refer to (Schubert and Hwang, 2000). (I'd like to thank mathcha.io for making the construction of this TikZ figure so painless with their graphical editor.)

resentation, and making inferences about relationships and constraints quite easy. Regarding component **(2)** *supra*, our group has done some work on extraction of simple world knowledge: Gene Kim has worked on extracting semantic verb definitions from WordNet by retrieving their glosses—simple, definitional phrasings of verbs, like *strike violently* for the verb *slam*—and parsing them into EL. Gene's ULF project (Kim and Schubert, 2019) addresses some of the difficulties of EL parsing to make this task easier in the future. This work suggests similar methods for pursuing object information. One could imagine using structured knowledge bases to make the EL parsing task easier, as well, such as MIT's ConceptNet: a graph database of entity-labeled vertices connected by relationship-labeled edges, containing crowd-sourced facts such as `[guitar]` $\xrightarrow[\texttt{is a type of}]{}$ `[instrument]` and `[book]` $\xrightarrow[\texttt{is used for}]{}$ `[learning]` (Liu and Singh, 2004).

In the remainder of this section, I'll focus on component **(3)**—protoschemas—and illustrate our current schema representation by walking through an example of a protoschema. Then, I'll explain why we see a protoschema-based system as a scalable approach to schema learning. I'll close out with a description of what I've implemented so far, what I'm working on now, and what I plan to work on in the future.

A note: large portions of this section were adapted from sections I wrote in a recent paper (Lawley et al., 2019, to appear), which was very recently accepted to a workshop at IWCS 2019. (Woohoo!)

4.4.1 Explanation of an example protoschema

Here, I'll explain the form and meaning of schemas by walking through an example schema—one of the system's built-in *protoschemas*—for the general situation of an agent doing an action to enable another action. The reason why I use the words "form" and "meaning" instead of "syntax" and "semantics", respectively, is that those latter two words might imply a formal, well-defined schema grammar, and a complete set-theoretic semantics in the logical sense (also respectively). We are treating schema learning as a very *practical* endeavor. While Episodic Logic valiantly, and with reasonable success, defines a formal semantics to represent many aspects of human language, formal models of human behavior, learning, and meta-reasoning are perhaps, for the moment, best explored by theorists in other disciplines. We are currently not prioritizing such rigor for our schema representation. However, this does not mean our system does not have "semantics" in a less formal, less edge-case-sensitive sense: there is quite a bit of structure here, as I will demonstrate throughout this section as I explain the features apparent in the `do_to_enable_action.v` protoschema represented in Figure 4.2 on the next page.

```
(epi-schema ((?x do_to_enable_action.v ?a1 ?a2) ** ?e)
    ("Nonfluent-conds"
        !r1 (?x agent6.n)
        !r2 ( or (?a1 (kind1-of.n action1.n)
                      (?a1 (kind1-of.n activity1.n)) )
        !r3 ( or (?a2 (kind1-of.n action1.n)
                      (?a2 (kind1-of.n activity1.n)) )
    )

    ("Goals"
        ?g1 (?x want1.v (that (?x can.md (do2.v ?a2))))
    )

    ("Init-conds"
        ?i1 (not (?x (can.md (do2.v ?a2))))
    )

    ("Steps"
        ?e1 (?x do2.v ?a1)
    )

    ("Post-conds"
        ?p1 (?x (can.md (do2.v ?a2)))
    )

    ("Episode-relations"
        !w1 (?e1 same-time ?e)
        !w2 (?e1 consec ?p1)
        !w3 (?e1 cause-of ?p1)
    )
  )
```

Figure 4.2: A protoschema for situations where an agent does an action to enable another action. This protoschema is explained in Section 4.4.1 on the preceding page.

Overall structure

A schema comprises a schema type and header, shown here in the first line of the schema, followed by a list of sections. Sections are lists of logical formulas, indexed by a unique (across all sections) identifier, and fall into one of two types: *episode sections*, in which all identifiers begin with a question mark (*?*), and *nonfluent sections*, in which all identifiers begin with an exclamation mark (*!*). Each *?*-identifier in an episode section introduces an episode characterized by the formula that follows. The meaning of the expressions within these sections will be explained below.

Header

The example schema here is an *epi-schema*, that is, an *episodic* schema. It describes an episode in the Episodic Logic sense: a time-bounded entity characterized by some number of formulas, and in which some superset of the characterizing propositions is true. The episode it describes, here `?e`, is called the "head episode" of the schema. The header—in this schema, `((?x do_to_enable_action.v ?a1 ?a2) ** ?e)`—is a characterizing formula of the head episode. Derivation of the header characterization from a story sentence, here `give_obj_for_poss.v`, certainly implies the rest of the schema: if a story gives the EL formula `(|Gaurav| do_to_enable_action.v (Ka (buy.v (K tea.n))) (Ka (drink.v (K tea.n))))`, we know essentially *by definition* that Gaurav bought some tea to enable the action of drinking that tea, because the name of the verb predicate in the header is unique to this schema.

However, it is extremely unlikely that `do_to_enable_action.v` will be parsed from any lexeme in a story; the point is not to "confirm" a schema on the basis of its artificial header name, but by observation of some of its pieces, which prompt us to predict the rest of the schema. An epi-schema can be viewed as a definition of an episode type, where all *?*-variables in the header and body of a schema, except the head episode (here, `?e`), are Skolem functions (equivalently, role functions) of that episode. As such, they can be equated to externally supplied constants. For example, the location variable `?x` in our example schema is interpreted as a Skolem function of `?e`, so that if `?e` receives a particular value, say `|EP1|`, then `(?x |EP1|)` in effect denotes the sole agent participating in this schema; this might become equated to an external constant such as `|Gaurav|`.

"Episode" vs. "nonfluent" sections

Fluent conditions, or episodic conditions, are "susceptible to change" over time. Nonfluent conditions hold true regardless of time. We specify nonfluent conditions in sections whose identifiers begin with (*!*). Within nonfluent sections, such as

`Nonfluent-conds` or `Episode-relations`, we interpret any nonfluent condition identifier `!nf`$_N$ as an EL *metavariable*—a.k.a. an *alias*—which can be freely substituted for its associated formula Φ_N. Within episode sections, such as `Steps` or `Init-conds`, we interpret any episode condition identifier `?e`$_N$ as an episode variable that is characterized by its associated formula Ψ_N.

Initial conditions and postconditions

Initial conditions are a form of fluent predication—they must be true at the outset of the schema, but might not be true for the entire schema episode. They might be true before the start of the schema, or they might become true at the exact start time of the schema. This temporal relationship is encoded, for each initial condition episode `?i`$_N$, by the tacit assumption of the nonfluent episode-relational predication `((start-of.f ?e) during ?i`$_N$`)`. Postconditions must be true at least at the end point of the schema, or immediately afterward, so for each postcondition episode `?p`$_N$, we tacitly assume the nonfluent predication `((end-of.f ?e) during ?p`$_N$`)`. Note that all initial conditions and postconditions *may* be true at *any* time, so long as they fit these constraints. Also note that these constraints may all be overridden by the `Episode-relations` section.

Steps and Goals

The steps section enumerates the episodes that occur in, and constitute, the head episode, and their order of enumeration here sets the default event ordering. (The `Episode-relations` section can override these and further specify the temporal and casual relations.) Goals are also episodes within the schema, but underlining their teleological contribution is necessary for action understanding, planning, and meta-reasoning—people do things for reasons.

Episode relations

The episode relations section specifies temporal and causal constraints on the subepisodes belonging to the schema's head episode (here, `?e`). The constraints it specifies may override the default relations for `Init-conds`, `Steps`, and `Postconditions`. In this example, episode `?e1` is *characterized*—as if by the `**` operator—by some agent `?x` doing some kind of action or activity `?a1` to enable some other kind of action or activity `?a2`.

The postcondition episode, `?p1`, is characterized by `?x` being able to do `?a2`. The episode relation section says that `?e1` and `?p1` are *consecutive*, and further that `?e1` is the direct cause of `?p1`. Temporal constraints in this section—which can relate the start times and end times of episodes, or provide uncertainty about start or end times—can induce a directed acyclic graph (DAG) of start and end times, a full interval graph, or a causality graph for action planning. Even the simplest induced graph, the DAG of start times, helps us to rule out schemas whose events are temporally inconsistent with a story. In the example schema, the constraint `!w1` says that step episode `?e1` shares a start and end time (a.k.a. `same-time`) with the schema's head episode `?e`.

The full temporal expressivity of the temporal relations that can be used here ends up being equivalent to Allen's Interval Algebra (AIA) (Allen, 1983), and, indeed, both the EPILOG system for EL inference and our schema system use AIA constraint solvers to reason about the chronology of episodes.

4.5 ...and why we're talking about protoschemas in the first place.

It is evident in the example schema in Figure 4.2 on page 63 that our representation allows for features absent in other schema systems. Like Minsky's frames, we can specify typed slots—e.g. `!r1 (?x agent6.n)`—but unlike most descendants of Minsky's frames, we can *relate* slots with arbitrary logical formulas, e.g. `(not (?x (can.md (do2.v ?a2))))`. Like Schank and Abelson's scripts, we can specify temporal event sequences, and organize them hierarchically—but unlike Schank and Abelson's scripts, our verb predicates do not rely on an atomic set of primitive actions; their semantics may be specified freely, in terms of other schemas, information extracted from dictionary definitions, or even relevant physical simulations, if desired. They may also remain largely unspecified, or specified only in terms of their relationship to other words; not every word needs to be understood in all its aspects in order to learn useful schemas.

But why *protoschemas*? Why is a protoschema-driven approach inherently more scalable? Why do we not simply use our schema representation to augment older, explanation-based systems like GENESIS or IPP, or attempt to learn sequences of them with LSTMs? Well, let's first recall those approaches to see what sets our system apart. GENESIS generalized schemas from stories and attempted to explain their goals and causal chains using automated proof techniques, but both the definitions of possible actions and the causal chains' ultimate motivations—e.g. "police officers want to arrest criminals"—were explicitly hand-encoded. IPP learned new, specialized schemas by noticing and combining common slot-filling values, and could not generate new teleological explanations, relying instead on the action units it already knew. And statistical schema learners are, by and large, simply pattern recognition engines without a consistent model of goals, states, or motivations.

What sets our approach to schema learning apart from past approaches is its focus on learning schemas the way a human child does. Human children learn simple schemas like “sharing toys” at young ages, but rapidly extend the underlying themes to describe new stereotyped concepts like “time-shared condominiums” as they age, by both generalizing out specific fillers like “toys”, and specifying concepts “sharing” to “time-sharing”. But each schema they learn is a modification to a schema whose overall structure already makes sense to them: the underlying theme of “sharing” is really not so different, and might even be inferred from the similarity of the word, or by noticing the similar situation of two people using the same thing. Many of these schemas from an initial set of behaviors and tendencies—the “head start” I outlined in Section 4.1 on page 51. These initial schemas do not all need to be biologically “built in”: concepts like sharing are likely learned, but are nevertheless learned quite young. We suggest that the initial set of protoschemas be roughly equivalent to what the average two-year-old child would know, with some examples being:

1. Doing something for pleasure
2. Doing something to avoid displeasure
3. Doing something to enable doing something else
4. Gaining new information
5. Doing something with someone you like for pleasure
6. Asking for someone’s help to enable doing something
7. Some basic Schankian “primitives”: basic knowledge of eating, grasping, sleeping, etc.

We do not yet have a fully formulated set of protoschemas, but those examples give an idea of the level of generality we’re aiming for: the underlying themes and processes described by a general schema like “asking for someone’s help to enable doing something” are easily recognizable in many more specific instances, such as “asking for someone’s help to enable lifting something”, or “asking your parents for help to enable doing something”. Those “general instances” of the protoschema can be learned and generalized from single examples, as schemas were in GENESIS, but they “inherit” their teleological structure from, and can likely be recognized on the basis of many shared or similar EL formulas with, their original protoschema.

4.6 Integrating machine learning techniques

As I described in Chapter 3, schema learning has shifted into a nearly complete focus on statistical machine learning (ML) techniques, which techniques often have problems with scalability and generating coherent knowledge representations. But that doesn't mean I'm dismissing these approaches entirely; on the contrary, it seems that the careful application of modern ML techniques to sub-problems—rather than the entire schema learning problem "end-to-end"—will be *necessary* for scalability. In this section, I'll discuss some potential applications of ML techniques to our protoschema approach.

4.6.1 Word representations

Choosing a representation for words is necessary to begin to understand them. Exact word matching is cheap and easy to create indexing systems for, but a sentence like "she teaches at the university" should probably evoke a schema for teaching at a "school". To enable this kind of "fuzzy matching", we'll need representations for words that are rich enough to encode this kind of similarity, but that are also simple enough to facilitate efficient schema storage and retrieval techniques.

Word embeddings provide such a representation. Yoshua Bengio described a "distributed representation for words" in which words are "embedded" in a high-dimensional vector space (Bengio et al., 2006). Often, a distance metric is defined for the vector space to represent the "semantic similarity" of two words: for example, the vector representations of "university" and "professor" should be closer together than those of "banana" and "transistor". How do we define, and implement, a notion of "semantic distance" between words? At the risk of sounding cliché, I'll refer to the classic line from (Firth, 1957): *"you shall know a word by the company it keeps"*. The frequent co-occurrence of words in small segments of a large corpus of text is often a good indicator of the relationship between the words, and has seemed to correspond well to intuitive notions of semantic similarity. ML techniques are uniquely good at processing large text corpora according to explicit heuristics like "words occurring close together", and as such have made word embeddings practical in the last decade. Mikolov et al. (2013)'s **word2vec** embedding model was particularly influential, and is still used today, as are more recent models like ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018).

We are currently using word2vec to enable "fuzzy matching" of semantically similar words when finding schemas for stories, but our representations may change as we experiment with larger and larger knowledge bases of schemas.

4.6.2 Identifying good candidates for generalization

While the event sequences learned in statistical systems, such as the systems I discussed in Section 3.2 on page 42, lack a rich, logical structure, the frequently occurring chains of events they tend to uncover often still hint at schemas. It is likely that we could use such sequences as "suggestions" to our protoschema system of some story segments to look at, and what aspects are likely to be generalizable from those stories. The protoschema system could then proceed with its normal matching and generalization process, augmented with this information. This approach bears some resemblance to the hierarchical heuristics for schema candidate identification that Minsky described for frames (see Section 2.2.2 on page 17), as well as to a common tendency in humans to consciously investigate subconscious "hunches".

4.7 Conclusion

In this final chapter of my area paper, I've tried to motivate and describe the research area I'm undertaking in pursuit of my Ph.D., often making references to the discussions of past approaches to schema learning I presented Chapter 3. I hope that I've helped whet your appetite for the protoschema revolution, but this research is still in the foggy, nascent era that I think all really exciting research must start out in. There are still many questions to answer: how do we efficiently store a large number of schemas *and* their generalizations? How do we identify candidate schemas to retrieve when processing a story, fill in the roles of those candidate schemas, and decide whether to "abandon" the instance or "confirm" it? How do we generate enough world knowledge to infer some of the basic, unstated facts about stories we'll need to match schemas? How do we keep from applying general schemas to inappropriate event sequences that still happen to match, and inferring nonsense from the matched schemas? What's the best way to use world knowledge to decide which role fillers of a schema to generalize, and how much to generalize them?

These are all good questions that currently have no definite answers, but there is no shortage of ideas. This year, I've implemented basic algorithms for schema matching and inference, which have generated some "common-sense" inferences from partial matches of some basic protoschemas about possession to a real children's story taken from a first reader book. The results of that experiment, and some more detailed consideration of the future work we're planning to realize a protoschema-based learning system, have been written up in a paper by me, Gene Kim, and our advisor Lenhart Schubert. This paper has been accepted to the Sixth Workshop on Natural Language and Computer Science at IWCS 2019. If you're interested in the specifics in that paper, I've attached its relevant sections—those on matching experiments and future work—as Appendix B on page 82.

Bibliography

[1] James F. Allen. “Maintaining Knowledge About Temporal Intervals.” In: *Commun. ACM* 26.11 (Nov. 1983), pp. 832–843. ISSN: 0001-0782. DOI: 10.1145/182.358434. URL: http://doi.acm.org/10.1145/182.358434.

[2] Timothy A Allen and Norbert J Fortin. “The evolution of episodic memory.” In: *Proceedings of the National Academy of Sciences* 110.Supplement 2 (2013), pp. 10379–10386.

[3] Collin F. Baker, Charles J. Fillmore, and John B. Lowe. “The Berkeley FrameNet Project.” In: *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics - Volume 1*. ACL ’98/COLING ’98. Montreal, Quebec, Canada: Association for Computational Linguistics, 1998, pp. 86–90. DOI: 10.3115/980845.980860. URL: https://doi.org/10.3115/980845.980860.

[4] Jason Baldridge. “The opennlp project.” In: *URL: http://opennlp. apache. org/index. html,(accessed 2 February 2012)* (2005), p. 1.

[5] Frederic C Bartlett. *Remembering: A study in experimental and social psychology*. Vol. 14. Cambridge University Press, 1932.

[6] Yoshua Bengio et al. “Neural Probabilistic Language Models.” In: *Innovations in Machine Learning: Theory and Applications*. Ed. by Dawn E. Holmes and Lakhmi C. Jain. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006, pp. 137–186. DOI: 10.1007/3-540-33486-6_6. URL: https://doi.org/10.1007/3-540-33486-6_6.

[7] Tim Berners-Lee, James Hendler, Ora Lassila, et al. “The semantic web.” In: *Scientific american* 284.5 (2001), pp. 28–37.

[8] Matthew Botvinick and David C Plaut. “Doing without schema hierarchies: a recurrent connectionist approach to normal and impaired routine sequential action.” In: *Psychological review* 111.2 (2004), p. 395.

[9] Ronald Brachman and Hector Levesque. *Knowledge Representation and Reasoning*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 2004. ISBN: 1558609326.

[10] Ronald J Brachman and James G Schmolze. "An overview of the KL-ONE knowledge representation system." In: *Readings in artificial intelligence and databases*. Elsevier, 1989, pp. 207–230.

[11] Nathanael Chambers and Dan Jurafsky. "Unsupervised learning of narrative event chains." In: *Proceedings of ACL-08: HLT* (2008), pp. 789–797.

[12] Eugene Charniak et al., eds. *Artificial Intelligence Programming*. Hillsdale, NJ, USA: L. Erlbaum Associates Inc., 1980. ISBN: 0898596092.

[13] Lynn A Cooper and Roger N Shepard. "Chronometric studies of the rotation of mental images." In: *Visual information processing*. Elsevier, 1973, pp. 75–176.

[14] Donald Davidson. "The Logical Form of Action Sentences." In: *The Logic of Decision and Action*. Ed. by Nicholas Rescher. University of Pittsburgh Press, 1967.

[15] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding." In: *arXiv preprint arXiv:1810.04805* (2018).

[16] Douglas Eck and Juergen Schmidhuber. "Finding temporal structure in music: Blues improvisation with LSTM recurrent networks." In: *Proceedings of the 12th IEEE workshop on neural networks for signal processing*. IEEE. 2002, pp. 747–756.

[17] Richard E Fikes, Peter E Hart, and Nils J Nilsson. "Learning and executing generalized robot plans." In: *Artificial intelligence* 3 (1972), pp. 251–288.

[18] J. R. Firth. "A synopsis of linguistic theory 1930-55." In: 1952-59 (1957), pp. 1–32.

[19] Daniel Gildea and Daniel Jurafsky. "Automatic labeling of semantic roles." In: *Computational linguistics* 28.3 (2002), pp. 245–288.

[20] Ana-Maria Giuglea and Alessandro Moschitti. "Semantic role labeling via framenet, verbnet and propbank." In: *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics. 2006, pp. 929–936.

[21] Arthur C Graesser, Murray Singer, and Tom Trabasso. "Constructing inferences during narrative text comprehension." In: *Psychological review* 101.3 (1994), p. 371.

[22] Alex Graves and Jürgen Schmidhuber. “Offline handwriting recognition with multidimensional recurrent neural networks.” In: *Advances in neural information processing systems*. 2009, pp. 545–552.

[23] Sepp Hochreiter and JÃŒrgen Schmidhuber. “Long Short-Term Memory.” In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735. eprint: https://doi.org/10.1162/neco.1997.9.8.1735. URL: https://doi.org/10.1162/neco.1997.9.8.1735.

[24] Chung Hee Hwang and Lenhart K Schubert. “Episodic logic: A situational logic for natural language processing.” In: *Situation Theory and its Applications* 3 (1993), pp. 303–338.

[25] Chung Hee Hwang and Lenhart K. Schubert. “Tense Trees As the ”Fine Structure” of Discourse.” In: *Proceedings of the 30th Annual Meeting on Association for Computational Linguistics*. ACL ’92. Newark, Delaware: Association for Computational Linguistics, 1992, pp. 232–240. DOI: 10.3115/981967.981997. URL: https://doi.org/10.3115/981967.981997.

[26] Bram Jans et al. “Skip n-grams and ranking functions for predicting script events.” In: *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics. 2012, pp. 336–344.

[27] Frank C Keil. *Concepts, kinds, and cognitive development*. mit Press, 1992.

[28] Gene Louis Kim. “Towards Parsing Unscoped Episodic Logical Forms with a Cache Transition Parser.” In: *the Poster Abstracts of the Proceedings of the 32nd International Conference of the Florida Artificial Intelligence Research Society*. Sarasota, Florida, USA, 2019.

[29] Gene Louis Kim and Lenhart Schubert. “A Type-coherent, Expressive Representation as an Initial Step to Language Understanding.” In: *arXiv preprint arXiv:1903.09333* (2019).

[30] Janet L Kolodner. “Maintaining organization in a dynamic long-term memory.” In: *Cognitive science* 7.4 (1983), pp. 243–280.

[31] Thomas K Landauer and Susan T Dumais. “A solution to Plato’s problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge.” In: *Psychological review* 104.2 (1997), p. 211.

[32] Thomas K Landauer, Peter W Foltz, and Darrell Laham. “An introduction to latent semantic analysis.” In: *Discourse processes* 25.2-3 (1998), pp. 259–284.

[33] Lane Lawley, Gene L. Kim, and Lenhart Schubert. "Towards Natural Language Story Understanding with Rich Logical Schemas." In: *Proceedings of the IWCS workshop on Natural Language and Computer Science*. 2019, to appear. URL: https://cs.rochester.edu/~llawley/papers/nlcs_schema_paper.pdf.

[34] Michael Lebowitz. "Generalization and Memory in an Integrated Understanding System." AAI8109800. PhD thesis. New Haven, CT, USA, 1980.

[35] Hugo Liu and Push Singh. "ConceptNet—a practical commonsense reasoning tool-kit." In: *BT technology journal* 22.4 (2004), pp. 211–226.

[36] Steven L Lytinen. "Conceptual dependency and its descendants." In: *Computers & Mathematics with Applications* 23.2-5 (1992), pp. 51–73.

[37] John McCarthy and Patrick J. Hayes. "Some Philosophical Problems from the Standpoint of Artificial Intelligence." In: *Machine Intelligence 4*. Ed. by B. Meltzer and D. Michie. reprinted in McC90. Edinburgh University Press, 1969, pp. 463–502.

[38] Drew V. McDermott. *Assimilation of New Information by a Natural Language Understanding System*. Tech. rep. Cambridge, MA, USA, 1974.

[39] William Holmes McGuffey. *The New McGuffey First Reader*. American Book Company, 1901.

[40] Deborah L McGuinness, Frank Van Harmelen, et al. "OWL web ontology language overview." In: *W3C recommendation* 10.10 (2004), p. 2004.

[41] Danielle S McNamara et al. "Are good texts always better? Interactions of text coherence, background knowledge, and levels of understanding in learning from text." In: *Cognition and instruction* 14.1 (1996), pp. 1–43.

[42] Tomas Mikolov et al. "Distributed Representations of Words and Phrases and Their Compositionality." In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'13. Lake Tahoe, Nevada: Curran Associates Inc., 2013, pp. 3111–3119. URL: http://dl.acm.org/citation.cfm?id=2999792.2999959.

[43] George A Miller. "WordNet: a lexical database for English." In: *Communications of the ACM* 38.11 (1995), pp. 39–41.

[44] Marvin Minsky. *A Framework for Representing Knowledge*. Tech. rep. Cambridge, MA, USA, 1974.

[45] Tom M. Mitchell, Richard M. Keller, and Smadar T. Kedar-Cabelli. "Explanation-Based Generalization: A Unifying View." In: *Mach. Learn.* 1.1 (Mar. 1986), pp. 47–80. ISSN: 0885-6125. DOI: 10.1023/A:1022691120807. URL: http://dx.doi.org/10.1023/A:1022691120807.

[46] Raymond J. Mooney. "A General Explanation-Based Learning Mechanism and its Application to Narrative Understanding." PhD thesis. Department of Computer Science, University of Illinois at Urbana-Champaign, 1988. URL: http://www.cs.utexas.edu/users/ai-lab/?mooney:phd88.

[47] Raymond J. Mooney. "Learning Plan Schemata From Observation: Explanation-Based Learning for Plan Recognition." In: *Cognitive Science* 14.4 (1990), pp. 483–509. URL: http://www.cs.utexas.edu/users/ai-lab/?mooney:cogsci90.

[48] Fabrizio Morbini and Lenhart K Schubert. "Evaluation of EPILOG: a reasoner for Episodic Logic." In: *Commonsense* 9 (2009), pp. 1–3.

[49] Gregory L Murphy and Douglas L Medin. "The role of theories in conceptual coherence." In: *Psychological review* 92.3 (1985). Early work highlighting the importance of structured conceptual theories, rather than similarity-based metrics, in knowledge representation (specifically categorization), p. 289.

[50] Ulric Neisser. "John Dean's memory: A case study." In: *Cognition* 9 (Mar. 1981), pp. 1–22. DOI: 10.1016/0010-0277(81)90011-1.

[51] Martha Palmer, Daniel Gildea, and Paul Kingsbury. "The proposition bank: An annotated corpus of semantic roles." In: *Computational linguistics* 31.1 (2005), pp. 71–106.

[52] Robert Parker et al. "English gigaword fifth edition, june." In: *Linguistic Data Consortium, LDC2011T07* 12 (2011).

[53] Matthew E Peters et al. "Deep contextualized word representations." In: *arXiv preprint arXiv:1802.05365* (2018).

[54] Miriam R. L. Petruck and Michael J. Ellsworth. "Representing Spatial Relations in FrameNet." In: *Proceedings of the First International Workshop on Spatial Language Understanding*. New Orleans: Association for Computational Linguistics, June 2018, pp. 41–45. DOI: 10.18653/v1/W18-1405. URL: https://www.aclweb.org/anthology/W18-1405.

[55] Karl Pichotta. "Advances in Statistical Script Learning." PhD thesis. Department of Computer Science, The University of Texas at Austin, 2017. URL: http://www.cs.utexas.edu/users/ai-lab/pub-view.php?PubID=127649.

[56] Karl Pichotta and Raymond Mooney. "Statistical Script Learning with Multi-Argument Events." In: *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics*. Gothenburg, Sweden: Association for Computational Linguistics, Apr. 2014, pp. 220–229. DOI: 10.3115/v1/E14-1024. URL: https://www.aclweb.org/anthology/E14-1024.

[57] Karl Pichotta and Raymond J Mooney. "Learning statistical scripts with LSTM recurrent neural networks." In: *Thirtieth AAAI Conference on Artificial Intelligence*. 2016.

[58] Plato. *Meno*. Platonic dialogues. 402 BCE.

[59] Georgiy Platonov and Lenhart Schubert. "Computational Models for Spatial Prepositions." In: *Proceedings of the First International Workshop on Spatial Language Understanding*. 2018, pp. 21–30.

[60] James Pustejovsky. "The generative lexicon." In: *Computational linguistics* 17.4 (1991), pp. 409–441.

[61] James Pustejovsky et al. "The timebank corpus." In: *Corpus linguistics*. Vol. 2003. Lancaster, UK. 2003, p. 40.

[62] Willard Van Orman Quine. "Events and reification." In: *Actions and events: Perspectives on the philosophy of Donald Davidson* (1985), pp. 162–171.

[63] Altaf Rahman and Vincent Ng. "Coreference Resolution with World Knowledge." In: *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies - Volume 1*. HLT '11. Portland, Oregon: Association for Computational Linguistics, 2011, pp. 814–824. ISBN: 978-1-932432-87-9. URL: http://dl.acm.org/citation.cfm?id=2002472.2002575.

[64] R. Reiter. "Readings in Nonmonotonic Reasoning." In: ed. by Matthew L. Ginsberg. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1987. Chap. A Logic for Default Reasoning, pp. 68–93. ISBN: 0-934613-45-1. URL: http://dl.acm.org/citation.cfm?id=42641.42646.

[65] R. Reiter. "Readings in Nonmonotonic Reasoning." In: ed. by Matthew L. Ginsberg. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1987. Chap. On Closed World Data Bases, pp. 300–310. ISBN: 0-934613-45-1. URL: http://dl.acm.org/citation.cfm?id=42641.42663.

[66] John Alan Robinson et al. "A machine-oriented logic based on the resolution principle." In: *Journal of the ACM* 12.1 (1965), pp. 23–41.

[67] Ari Rosenberg, Noah J Cowan, and Dora E Angelaki. "The visual representation of 3D object orientation in parietal cortex." In: *Journal of Neuroscience* 33.49 (2013), pp. 19352–19361.

[68] Reuven Y Rubinstein and Dirk P Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer Science & Business Media, 2013.

[69] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. “Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1.” In: ed. by David E. Rumelhart, James L. McClelland, and CORPORATE PDP Research Group. Cambridge, MA, USA: MIT Press, 1986. Chap. Learning Internal Representations by Error Propagation, pp. 318–362. ISBN: 0-262-68053-X. URL: http://dl.acm.org/citation.cfm?id=104279.104293.

[70] R.C. Schank and R.P. Abelson. *Scripts, Plans, Goals, and Understanding: An Inquiry Into Human Knowledge Structures*. The Artificial Intelligence Series. Lawrence Erlbaum Associates, 1977. ISBN: 9780470990339. URL: https://books.google.com/books?id=YZ99AAAAMAAJ.

[71] Roger C Schank. “A conceptual dependency representation for a computer-oriented semantics.” PhD thesis. University of Texas at Austin, 1969.

[72] Roger C Schank. *Dynamic memory: A theory of reminding and learning in computers and people*. Vol. 240. Cambridge University Press Cambridge, 1982.

[73] Roger C Schank. “Language and memory.” In: *Cognitive science* 4.3 (1980), pp. 243–284.

[74] Roger C. Schank. “The Primitive Acts of Conceptual Dependency.” In: *Proceedings of the 1975 Workshop on Theoretical Issues in Natural Language Processing*. TINLAP ’75. Cambridge, Massachusetts: Association for Computational Linguistics, 1975, pp. 34–37. DOI: 10.3115/980190.980205. URL: https://doi.org/10.3115/980190.980205.

[75] Roger C Schank, Gregg C Collins, and Lawrence E Hunter. “Transcending inductive category formation in learning.” In: *Behavioral and Brain Sciences* 9.4 (1986), pp. 639–651.

[76] Lenhart Schubert. “Can We Derive General World Knowledge from Texts?” In: *Proceedings of the Second International Conference on Human Language Technology Research*. HLT ’02. San Diego, California: Morgan Kaufmann Publishers Inc., 2002, pp. 94–97. URL: http://dl.acm.org/citation.cfm?id=1289189.1289263.

[77] Lenhart K Schubert. “What kinds of knowledge are needed for genuine understanding.” In: *IJCAI 2015 Workshop on Cognitive Knowledge Acquisition and Applications (Cognitum 2015)*. 2015.

[78] Lenhart K Schubert and Chung Hee Hwang. “Episodic Logic meets Little Red Riding Hood: A comprehensive, natural representation for language understanding.” In: *Natural language processing and knowledge representation: Language for Knowledge and Knowledge for Language* (2000), pp. 111–174.

[79] Dan Shen and Mirella Lapata. "Using semantic roles to improve question answering." In: *Proceedings of the 2007 joint conference on empirical methods in natural language processing and computational natural language learning (EMNLP-CoNLL)*. 2007.

[80] Jeffrey Mark Siskind. "Conducting Neuroscience to Guide the Development of AI." In: *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*. AAAI'15. Austin, Texas: AAAI Press, 2015, pp. 4067–4072. ISBN: 0-262-51129-0. URL: http://dl.acm.org/citation.cfm?id=2888116.2888285.

[81] Barry Smith et al. "The OBO Foundry: coordinated evolution of ontologies to support biomedical data integration." In: *Nature biotechnology* 25.11 (2007), p. 1251.

[82] Fabian M Suchanek, Gjergji Kasneci, and Gerhard Weikum. "Yago: a core of semantic knowledge." In: *Proceedings of the 16th international conference on World Wide Web*. ACM. 2007, pp. 697–706.

[83] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. "Sequence to sequence learning with neural networks." In: *Advances in neural information processing systems*. 2014, pp. 3104–3112.

[84] Swabha Swayamdipta et al. "Frame-Semantic Parsing with Softmax-Margin Segmental RNNs and a Syntactic Scaffold." In: *CoRR* abs/1706.09528 (2017). arXiv: 1706.09528. URL: http://arxiv.org/abs/1706.09528.

[85] Wilson L Taylor. ""Cloze procedure": A new tool for measuring readability." In: *Journalism Bulletin* 30.4 (1953), pp. 415–433.

[86] Cynthia A Thompson, Roger Levy, and Christopher D Manning. "A generative model for semantic role labeling." In: *European Conference on Machine Learning*. Springer. 2003, pp. 397–408.

[87] Silvan S Tomkins. "Script theory: Differential magnification of affects." In: *Nebraska symposium on motivation*. University of Nebraska Press. 1978.

[88] Endel Tulving et al. "Episodic and semantic memory." In: *Organization of memory* 1 (1972), pp. 381–403.

[89] A. M. Turing. "Computers & Thought." In: ed. by Edward A. Feigenbaum and Julian Feldman. Cambridge, MA, USA: MIT Press, 1950. Chap. Computing Machinery and Intelligence, pp. 11–35. ISBN: 0-262-56092-5. URL: http://dl.acm.org/citation.cfm?id=216408.216410.

[90] Teun Adrianus Van Dijk and Walter Kintsch. *Strategies of discourse comprehension*. Academic press New York, 1983.

[91] Terry Winograd. “A procedural model of language understanding.” In: *Computer Models of Thought and Language*. Ed. by R Schank and K Colby. New York, NY: WH Freeman, 1973, pp. 152–186.

[92] Ichiro Yamada and Timothy Baldwin. “Automatic discovery of telic and agentive roles from corpus data.” In: *Proceedings of the 18th Pacific Asia Conference on Language, Information and Computation*. 2004, pp. 115–126.

[93] Rolf A Zwaan, Mark C Langston, and Arthur C Graesser. “The construction of situation models in narrative comprehension: An event-indexing model.” In: *Psychological science* 6.5 (1995), pp. 292–297.

Appendix A

An Example Schankian Script

```
  Script: RESTAURANT
  Track:  Coffee Shop                       Roles: S-Customer
  Props:  Tables                                   W-Waiter
          Menu                                     C-Cook
          F-Food                                   M-Cashier
          Check                                    O-Owner
          Money
```

```
                                            Results: S has less money
  Entry conditions: S is hungry.                       O has more money
                    S has money.                       S is not hungry
                                                       S is pleased (optional)
```

Scene 1: Entering

```
S PTRANS S into restaurant
S ATTEND eyes to tables
S MBUILD where to sit
S PTRANS S to table
S MOVE S to sitting position
```

Scene 2: Ordering (see next page for diagram)

Scene 3: Eating

```
C ATRANS F to W
W ATRANS F to S
S INGEST F
```

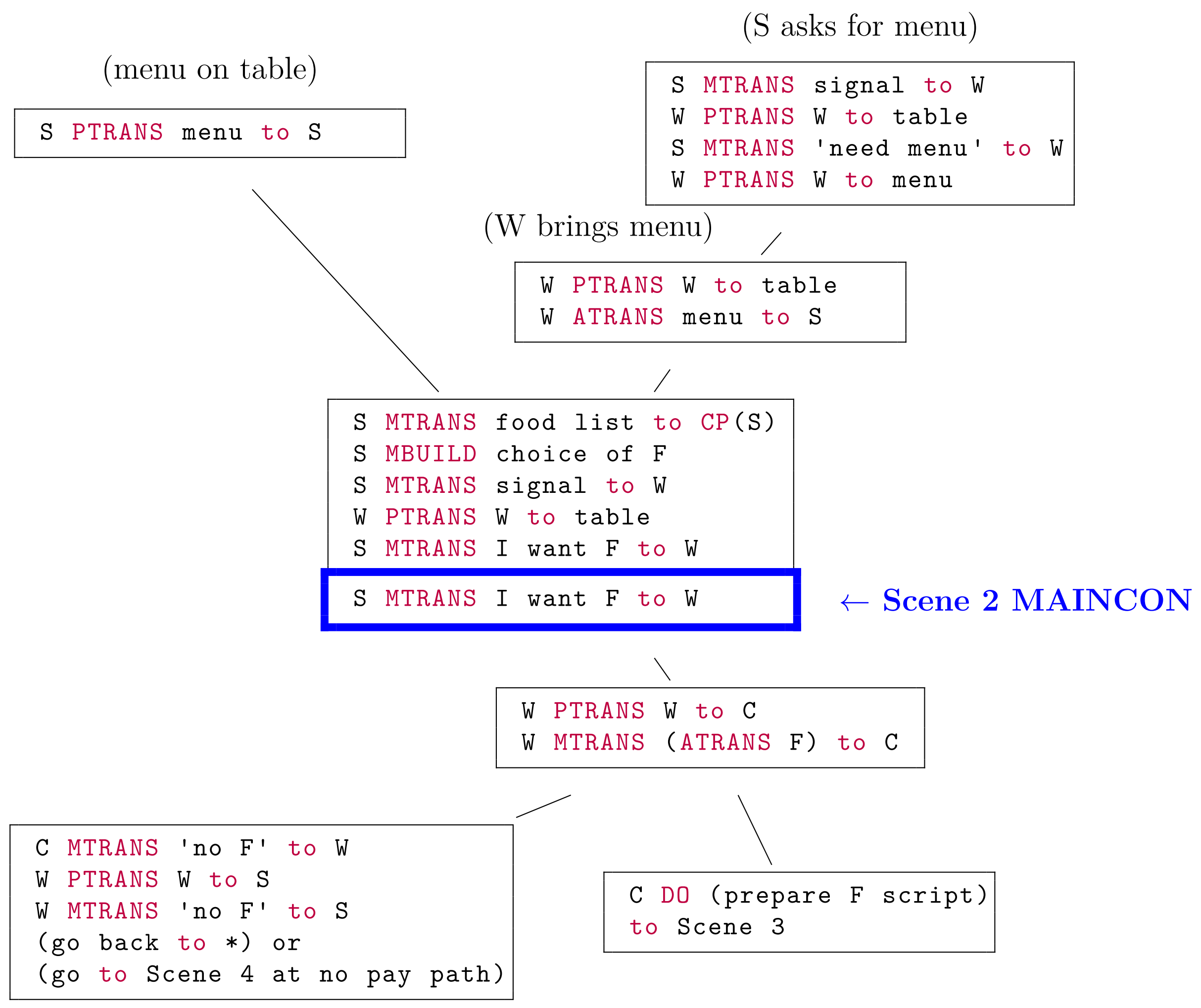
(Optionally return to Scene 2 to order more; otherwise go to Scene 4)

Scene 4: Exiting

```
                              (check not at table path):
                              S MTRANS to W (W ATRANS check to S)
                              (go to Scene 4 at check at table path)

               (check at table path):
               W MOVE (write check)
               W PTRANS W to S
               W ATRANS check to S
               S ATRANS tip to W
               S PTRANS S to M
               S ATRANS money to M
(no pay path): S PTRANS S to out of restaurant
```

(relocated from previous page) Scene 2: Ordering



Appendix B

Workshop paper excerpt: matching, inference, & future work

Using schemas to make sense of a story requires casting the story in the terms schemas: We must find "matches" between sentences and individuals in the story, and formulas and roles in the schema. We have designed, implemented, and experimented with a basic algorithm which takes a ULF parse of a story, performs some preprocessing to get a basic EL form, matches the formulas in the story to new schemas as well as schema instances stored in "working memory", generates inferences from those schemas after filling in their variables, and uses those inferences to fill in yet more schemas. In this section, we detail the algorithm, and walk through an example from one of our experiments on a children's first reader story from *The New McGuffey First Reader* (McGuffey, 1901).

```
(epi-schema
  ((?x give_obj_for_poss.v ?y ?o) ** ?e)
  (:Nonfluent-conds
    !r1 (?x agent.n)
    !r2 (?y agent.n)
    !r3 (?o object.n)
    !r4 (?l location.n) )

  (:Init-conds
    ?i1 (?x (can.aux-v (give_to.v ?y ?o)))
    ?i2 (?x (be.v (adv-e (at.p ?l))))
    ?i3 (?y (be.v (adv-e (at.p ?l)))) )

  (:Goals
    ?g1 (?x want.v
      (that (?y (have.v ?o)))) )

  (:Steps
    ?e1 (?x (give_to.v ?y ?o)) )

  (:Post-conds
    ?p1 (?y (possess.v ?o)) )

  (:Episode-relations
    !w1 (?e1 same-time ?e)
    !w2 (?e1 consec ?p1)
    !w3 (?e1 cause-of.n ?p1) ) )
```

Figure B.1: An example schema.

B.1 Matching Algorithm

The matching algorithm takes a (potentially partially-filled) schema instance SCH, which is a schema and a map of its bound variable names to their values, and a story ST, which is a list of episodes, with their characterizing formulas, along with type predications, etc. Each step to match a story WFF is based on a global "knowledge base" of episodes and formulas we know to be true. The knowledge base is initialized with the story WFF, and the schema instantiation process then begins. We can create

Algorithm 1 Algorithm for matching story formulas to schemas

```
$KB \leftarrow \emptyset$
**for** story episode $E$ in $ST$ **do**
  **for** formula $F$ in $E$ **do**
    $KB \leftarrow KB \cup \{F\}$
  **while** $\exists$ unprocessed formula $F_{ST} \in KB$ **do**
    **for** episode $E_{SCH}$ in $SCH$ episodes (linearized timeline) **do**
      **for** formula $F_{SCH} \in E_{SCH}$ **do**
        $MGU \leftarrow$ most general unifier of $F_{SCH}$ and $F_{ST}$
        **if** $MGU$ does not exist **then**
          continue
        **for** variable binding $B = V_{SCH} \rightarrow T_{ST}$ **do**
          **if** $V_{SCH}$ is not already bound to something else **then**
            substitute $T_{ST}$ for all occurrences of $V_{SCH}$ in the schema instance
        **if** schema instance is confirmed **then**
          **for** fully-bound formula $F_B$ in the instance **do**
            $KB \leftarrow KB \cup \{F_B\}$
    mark $F_{ST}$ as processed
```

a new schema instance, or update an existing one, when a WFF in the knowledge base "matches" a WFF in the schema, i.e., is successfully unified. We obtain the "most general unifier" (MGU) using the algorithm by Robinson, 1965. The MGU acts as a variable-to-term mapping, which we then use to replace variables throughout the schema instance. If a schema instance is "confirmed", we "infer" formulas within the schema whose variables have all been made concrete, and we add them to the knowledge base. As our initial schemas are quite simple, our current confirmation heuristic is simply whether all episodes in the `:Steps` section have been matched. As we continue to develop with more complex stories, and as schemas grow more complex, a more nuanced approach—perhaps involving certainties and numbers of variables mapped—is called for. Once the knowledge base has been updated, we return back to the schema matching step; we stop when we can no longer instantiate new schemas, or update existing instances, and move on to the next story WFF.

B.2 Planning and Meta-Reasoning

As the reasoning procedures and initial schemas are hypothesized to be quite general, we can make inferences about the beliefs and desires of agents within the story by "simulating" their reasoning with our own algorithms. A simple example of such an

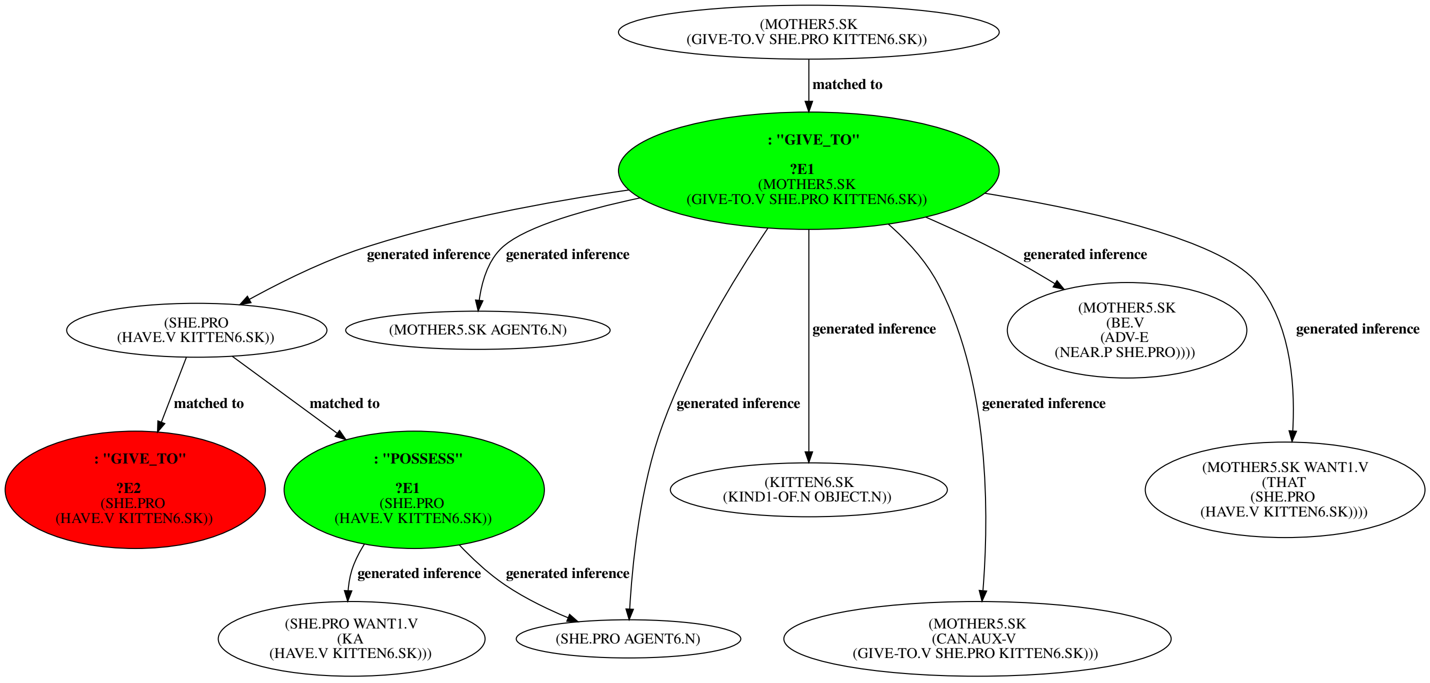

(MOTHER5.SK (GIVE-TO.V SHE.PRO KITTEN6.SK))
matched to
: "GIVE_TO" ?E1 (MOTHER5.SK (GIVE-TO.V SHE.PRO KITTEN6.SK))
generated inference
(SHE.PRO (HAVE.V KITTEN6.SK))
(MOTHER5.SK AGENT6.N)
(MOTHER5.SK (BE.V (ADV-E (NEAR.P SHE.PRO))))
(KITTEN6.SK (KIND1-OF.N OBJECT.N))
(MOTHER5.SK WANT1.V (THAT (SHE.PRO (HAVE.V KITTEN6.SK))))
: "GIVE_TO" ?E2 (SHE.PRO (HAVE.V KITTEN6.SK))
: "POSSESS" ?E1 (SHE.PRO (HAVE.V KITTEN6.SK))
(SHE.PRO WANT1.V (KA (HAVE.V KITTEN6.SK)))
(SHE.PRO AGENT6.N)
(MOTHER5.SK (CAN.AUX-V (GIVE-TO.V SHE.PRO KITTEN6.SK)))

Figure B.2: A graph of inferences made while processing a single story sentence. White vertices are inferred and story WFFs, green bubbles are "confirmed" schemas (see Section B.1 for definition), and red bubbles are unconfirmed schemas. Here, "GIVE_OBJECT" is shorthand for "give_obj_for_poss.v".

inference is that, if a mother wants her daughter to have a cat, and we know, via our general schemas, that owning a cat is pleasurable, we infer that the mother knew that her daughter having a cat would cause her daughter to experience pleasure, and we infer that the mother wanted her daughter to experience pleasure. We can draw many parallels to the planning domain: As agents have desires and beliefs, and schemas have goals, preconditions, and side effects, one could use existing schemas to solve planning problems, and, in turn, use planning algorithms to hypothesize new schemas for accomplishing certain goals.

B.3 Matching Example

Starting with the story sentence "Her mother gave the kitten to her" parsed into its EL form `(MOTHER5.SK (GIVE_TO.V SHE.PRO KITTEN6.SK))`, we'll walk through the matching algorithm, whose generated inferences are shown in Figure B.2.

B.3.1 ULF Processing

We first process the ULF verb predication `((past give.v) (the.d kitten.n) (to.p-arg her.pro))` in Figure B.3 to attach the argument marking preposition to the verb and float its argument to the front of the list. We then Skolemize the `the.d` determiners, demoting the relational noun predicate `(mother-of.n she.pro)` to the bare noun predicate `mother.n` to derive the Skolem name `MOTHER5.SK` (the 5 is a cumulative counter for Skolem constants; it is arbitrary, and an artifact of our Skolemization process), and using the noun predicate `kitten.n` to derive the Skolem name `KITTEN6.SK`. We finally obtain the EL sentence `(MOTHER5.SK (GIVE_TO.V SHE.PRO KITTEN6.SK))`, which is ready for matching.

B.3.2 The Initial Match

`GIVE_TO.V` immediately matches to the schema shown in Figure B.1, as that schema has the same name. We unify the story formula `(MOTHER5.SK (GIVE_TO.V SHE.PRO KITTEN6.SK))` with the schema header, producing the unifier
`(?x ← MOTHER5.SK, ?y ← SHE.PRO, ?o ← KITTEN6.SK)`, and we then make that substitution throughout the entire schema. Notably, every role in the schema is now bound except for the location `?l`. Note that, although the question of whether a schema is "confirmed" to have happened at any point in the matching process is difficult to answer in general, it is easy here: The actual verb in the schema's single step has been observed, so in the absence of type conflicts it is quite likely that the rest of the schema happened. All of the filled-in WFFs in the `Events`, `Goals`, `Init-conds`, and `Nonfluent-conds` sections, except those including the still-unbound variable `?l`, are added to our knowledge base as inferences from a confirmed schema.

B.3.3 Further Matching

While the story sentence has now been matched, it has generated inferences, and those inferences might match other schemas. So, we are not done; we must attempt to match each of those before moving on to the next story sentence. Indeed, in this case, the inference `(SHE.PRO (HAVE.V KITTEN6.SK))` matches to two schemas: `GIVE_TO.v` and `POSSESS.V`; the former is unconfirmed (but, upon close inspection, would appear to be an incomplete copy of the instance that generated its matched WFF), and the latter was triggered (and confirmed) by the `HAVE.V` verb predicate in the inference. The `POSSESS.V` schema generates two more inferences, one of which was also generated by the original `GIVE_TO.V` schema, and the matching process concludes for this sentence, as all generated inferences have been matched to all possible schemas.

```
; Here is May with her kitten.
(|May| ((pres be.v) here.a
        (adv-a (with.p (the.d
            (λx ((x kitten.n) and.cc (x (poss-by her.pro)))))))))
; Her mother gave the kitten to her.
((the.d (mother-of.n her.pro))
    ((past give.v) (the.d kitten.n) (to.p-arg she.pro)))
; She is kind to the pretty kitten.
(she.pro ((pres be.v) kind.a
          (adv-a (to.p (the.d (pretty.a kitten.n))))))
; She likes to see it jump and play.
(she.pro ((pres like.v) (ka (see it.v (jump.v and.cc play.v)))))
; See it run with May's ball!
(({you}.pro ((pres see.v) it.pro
    (run.v (adv-a (with.d (the.d
        (λ x ((x ball.n) and.cc (x (poss-by |May|)))))))))) !)
; It does not run far with it.
(it.pro ((pres do.aux-s) not
         (run.v far.adv-a (adv-a (with.p it.pro)))))
; If May can get the ball she will not take it.
((if.ps (|May| ((pres can.aux-v) (get.v (the.d ball.n)))))
 (she.pro ((pres will.aux-s) not (take.v it.pro))))
; She will give it to the kitten to play with.
(she.pro ((pres will.aux-s)
          (give.v (to.p-arg (the.d kitten.n)) it.pro
           (adv-a ({for}.p (ka (play-with.v {it}.pro)))))))
```

Figure B.3: A children's story, in partially post-processed ULF form. These were manually annotated but there are promising preliminary results on parsing ULF (Kim, 2019)

B.4 Conclusion & Future Work

Our approach to schemas comprises a rich logical language, initial schemas providing "low-level" inferences about the effects and motivations behind simple, general actions, and a way of doing "fuzzy matching" of inexact, but similar, formulas (see Section B.6). All of these components work together to provide inferences that enable a comprehensive, structured, and human-oriented "fleshing out" of events in stories. The inferences we generate can then be combined, generalized, and reasoned about, thus creating new schemas from relatively few examples. There is much work remaining to further develop our parsing, matching, and especially our generalization processes. We proceed here to outline some of that future work—our initial results from matching a small handful of initial schemas to a short children's story are promising.

B.5 Immediate Future: Generalization

Currently, we are focusing our efforts on the task of schema generalization: Given two (or more) sequences of predications, which may themselves contain nested schemas, can we create a schema that describes both stories with a more general pattern? We are currently experimenting on four short children's stories about fishing, and each story includes similar words used in similar contexts: "These men fish in the sea", "We will take the long rod, and the hook and line", "Here is Tom with his rod and line", "Sometimes they sit on the bank of the river", and many more thematically parallel sentences in the stories strongly suggest a schema where people are near water, have a rod, and catch fish with a rod or a net. After extracting a set of recurring events from the stories—like going to water, having a rod, catching fish, and putting fish in a basket—we plan to experiment with using narrative models like tense trees (Hwang and Schubert, 1992) to find subsequences of the events that could be interpreted as steps in a schema.

After extracting subsequences of similar events that occur in two or more stories, we can use knowledge of basic motivations—e.g. people often want to possess things—to infer a teleology of those events. If some unknown action "catch `?x`" always occurs before "possess `?x`", we might hypothesize that the catching something has an effect of possessing that thing. If that sequence is only ever seen with fish and crabs as `?x`, our confidence in the "catch to possess" schema might be lower when the object is not a marine animal, implying a "catch marine animal to possess" schema. Teleological inferences will likely prove to be very useful in sifting out reasonable new schema generalizations from a large collection of partly nonsensical ones.

B.6 “Fuzzy Matching”

So far, we have mostly discussed exact matching of WFFs using the MGU algorithm. However, in many cases, we will want to match something like a “Segway” to a schema predication about a “vehicle”, just as readily as we would match “car” to “vehicle”. Additionally, synonyms and intuitively similar words, like “run” and “jog”, should be able to match as well, perhaps with some certainty score. Hypernym hierarchies, semantic word embeddings, and logical world knowledge of object properties all affect whether we want to make inexact matches. We are actively researching how best to gather and use this information in the matching process—as well as how to index schemas for quick retrieval, even when matches are inexact.

B.7 A Note on Condition Strictness

It is currently unclear exactly how and when condition violations are acceptable. Certainly, to generalize new schemas, we must allow some “slack” in condition violations: a schema that fits, say, a “fishing” schema perfectly, except the variable constrained to be a fish is actually a crab, should cause us to infer that “catching seafood” might be a good generalization to store. However, different conditions intuitively seem violable to different degrees, and in different contexts, depending on what other conditions are met or violated. Many factors seem to affect whether we match or abandon a schema, or create a new schema, when trying to make sense of a story. These factors will be the subject of many future experiments.

B.8 Scaling the Matching Algorithm

Our algorithm is in an early stage, and several scaling problems must be solved to bring it to bear on an unsupervised learning task. We are currently experimenting on an assortment of children’s first reader stories to determine rules and heuristics to guide us as we develop the matching algorithm to cope with large numbers of possible schema matches.

B.8.1 Role Assignment Restriction

As the number of schemas scales, it will be combinatorially infeasible to maintain every possible schema instance a story WFF might prompt; if there are three humans in a schema, and ten humans mentioned in a story, there are $\binom{10}{3}$ ways to instantiate the human roles alone in that schema. In many cases, it may suffice to “let the actions speak for the agents", that is, prefer to use verb predications to identify individuals for schema roles, rather than considering role assignments directly. The schema relates arguments to its verb predications to their role definitions and "type" predications, so if we can infer that `?x` is `|Mary|` from an action that she did, we can then simply confirm or refute the additional constraint `(?x human.n)`. However, there could be examples where there is still suitable ambiguity, or where "action-first" role assignment isn't the most efficient; we hope to discover and address these examples in our ongoing story experiments.

B.8.2 Instance Abandonment

What might seem like a "promising" schema match at first could diverge suitably from the story and become useless; we would want to abandon it to free up memory, and so we don't continue fruitless comparisons of story WFFs to it. However, it is difficult to know when a schema instance is unlikely to continue to be matched to WFFs; sentences read, or inferences generated, much later on in a story could provide the missing piece to a schema instance matched many sentences ago. We plan to use our ongoing story experiments to formulate abandonment heuristics as well, or even identify potential applications of machine learning to the problem.

B.8.3 Schema Retrieval

As the number of schemas grows, and as we allow for inexact matching, whether of hypernyms, synonyms, or "functionally equivalent" terms (based on world knowledge), the task of identifying reasonable schemas as match candidates grows more difficult in turn. As a first step, indexing schemas by a handful of specific predications—specific verbs and nouns, chosen to maximally prune the search space—could allow for quick identification of relevant schemas. From there, indexing by sequences, or subsequences, of events should help us quickly identify highly specific sequential schemas within a large corpus; the human brain seems to be able to recall schemas very quickly with very terse sequences, as Winograd, 1973 demonstrates with the sequence "skid, crash, hospital". Finally, any indexing we've done will need to be augmented to perform well even with inexact matching as described above. Schema retrieval optimization will be heavily informed by our experiments with the matching space on real stories, and especially as our number of schemas grows.

Appendix C

Critical Acclaim for *Lane's Area Paper*

...messy and largely illogical.
–Roger Schank

People have managed to avert their eyes and hope for the best.
–David Chalmers

It's just that by and large, the deductive steps are so mundane, so uninteresting, that they go unnoticed.
–Lenhart Schubert

[The author is only] pretending to have a unified, coherent theory. The paper raises more questions than it answers, and I have tried to note the deficiencies of the theory.
–Marvin Minsky

There has to be a common sense cutoff for craziness ... the criteria for publication should get far, far more stringent.
–Douglas Hofstadter