Learning General Event Schemas with Episodic Logic

Lane Lawley and Benjamin Kuehnert and Lenhart Schubert
University of Rochester
Department of Computer Science
{llawley@cs, bkuehner@u, schubert@cs}.rochester.edu

Abstract

We present a system for learning generalized, stereotypical patterns of events—or "schemas"—from natural language stories, and applying them to make predictions about other stories. Our schemas are represented with Episodic Logic, a logical form that closely mirrors natural language. By beginning with a "head start" set of *protoschemas*—schemas that a 1- or 2-year-old child would likely know—we can obtain useful, general world knowledge with very few story examples—often only one or two. Learned schemas can be combined into more complex, composite schemas, and used to make predictions in other stories where only partial information is available.

1 Introduction

We present a novel approach to learning rich, symbolic event schemas from natural language texts. While most modern approaches to automated script learning (e.g. (Chambers and Jurafsky, 2011; Pichotta and Mooney, 2016a; Yuan et al., 2018)) learn linear sequences of simple tuple representations of events, our schema representation allows for typed and interrelated participating entities; multiple temporally related subevents; specification of goals, preconditions, and postconditions; and nesting of subschemas as steps in another schema.

We mitigate the "brittleness" of past symbolic approaches (e.g., GENESIS (Mooney, 1990) and IPP (Lebowitz, 1980)) by parsing stories into Episodic Logical Form (ELF) (Schubert and Hwang, 2000), a logical representation that closely resembles natural English, but allows for complex event representation and powerful inference procedures. As Stratos et al. (2011) argue, Episodic Logic facilitates "Natural Logic-like inference while also providing greater generality". EL, and its underspecified variant ULF, facilitate NLog-like inferences using a combination of lexical and semantic knowledge (Schubert, 2014; Kim et al., 2019). Because most nouns and verbs are preserved as predicates in ELFs, we also utilize existing lexical resources, like WordNet's hypernym hierarchy for generalizing schema predicates (e.g. DOG.N and ELEPHANT.N to PLACENTAL_MAMMAL.N), and semantic word embeddings for retrieving relevant schema candidates for a story from a large number of known schemas.

We also bypass the need for large amounts of data by giving the system a "head start" in the form of a relatively small number of initial schemas targeting the commonsense knowledge of a very young child, from which more complex schemas can be learned and composed. These "protoschemas" describe basic action types—e.g., eating, searching, moving from place to place, transferring possession of objects—at a very general level, along with their underlying motivations and pre- and postconditions. More complex schemas—e.g., "a monkey climbs a tree, gets a coconut, and eats the coconut"—can be composed by "chaining" these simpler ones together after matching them to a story.

From a corpus of several hundred short children's stories, we have acquired dozens of schema matches, generalized them into new schemas, automatically composed some of them into more complex schemas, and used those generalized schemas to make predictions on unseen stories with only partial information.

2 Episodic Logic

Our schema representation is based on Episodic Logic (EL) (Schubert and Hwang, 2000), a formal knowledge representation with semantic types and operators common to many natural languages. EL

uses first-order quantification, but has type-shifting and reification operators to map predicate and sentence intensions to domain individuals, allowing it to represent higher-order propositions.

EL is a good fit for schemas in part because of its *characterizing* operator `**`, which relates an EL formula to a situational argument, an "episode" that it characterizes. For example, the EL formula `((I.PRO EAT.V (K STEAK.N)) ** E1)` says that `E1` is a (possibly repetitive, habitual) episode of me eating steak[1]. Episodes can have multiple formulas "true in" them, where these formulas characterize subepisodes with limited temporal bounds. This makes them ideal for representing entire schemas, which are "packages" of formulas all true together within some span of time.

2.1 Overview

Although an adequate explanation of the features and syntax of EL would not fit within these margins—please refer to (Schubert and Hwang, 2000) for more detail—we offer a brief guide to understanding some of the formulas in, e.g., Figure 2.

2.1.1 Propositions

An atomic EL proposition has a **prefix argument** (sentential subject), an infixed **predicate**, and zero or more **postfix arguments**. In exact EL syntax, if there are postfixed arguments then the monadic predicate formed by the infix predicate together with its postfixed arguments is bracketed (e.g., see Figure 1. Monadic predicates as well as complete formulas may have **modifiers** applied to them. In the formula `(I.PRO (QUICKLY.ADV-A (EAT.V (K STEAK.N))))`, the prefix argument is the individual `I.PRO`, the infix predicate is the verbal predicate `EAT.V`, the postfix argument is the kind-level individual `(K STEAK.N)`, and the modifier is the adverb `QUICKLY.ADV-A`. When there are no predicate modifiers, atomic formulas with postfix arguments can be "flattened", as in the formula `(I.PRO EAT.V (K STEAK.N))` above.

Not all EL formulas use verbal predicates: type constraint formulas, like `(?X STEAK.N)` or `?D RED.A`, are examples of formulas with nominal and adjectival predicates.

2.1.2 Quantifiers

Although explicit quantifiers are not present in the schemas we present here—a schema's variables are implicitly Skolem functions of the schema's head episode—we will note that EL supports the standard first-order quantifiers $\exists$ and $\forall$. It also has nonstandard quantifiers like `Most` and `Few`, to represent sentences like "Most students who have studied here have gone on to be successful". Nonstandard quantifiers use "restrictors" to filter the quantified variables with an arbitrary predicate.

3 Schema Representation

In this section, we will describe our schema representation. Although sequential and causally connected events play a large role in our schemas, our schema language is differentiated from causal representations such as (Luo et al., 2016) and sequential script representations such as (Pichotta and Mooney, 2016b) by the expressiveness and interconnectedness of its constituent logical formulas. The language is designed to model the schema's **Steps**, the **Roles** (types) of participating entities, and the motivating **Goals**, **Preconditions**, and **Postconditions** of the schema as a whole.

An example schema our system has learned can be seen in Figure 1. The EL formulas specifying the semantic contents of a schema organized into sections; we describe the sections below.

```
1  (EPI-SCHEMA ((?X_D EAT.379.V ?X_C)
2                    ** ?X_E)
3    :ROLES
4      !R1 (?X_D AGENT.N)
5      !R2 (?X_C FOOD.N)
6      !R3 (?X_C GRASS.N)
7      !R4 (?X_D COW.N)
8    :GOALS
9      ?G1 (?X_D (WANT.V (THAT (NOT
10         (?X_D HUNGRY.A)))))
11   :PRECONDS
12     ?I1 (?X_D HAVE.V ?X_C)
13     ?I2 (?X_D HUNGRY.A)
14   :POSTCONDS
15     ?P1 (NOT (?X_D (HAVE.V ?X_C)))
16     ?P2 (NOT (?X_D HUNGRY.A))
17   :EPISODE-RELATIONS
18     !W1 (?P1 AFTER ?X_E)
19     !W2 (?I1 BEFORE ?X_E)
20   :NECESSITIES
21     !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
22  )
23  )
```

Figure 1: A schema learned by applying the eating protoschema to the sentence "The cow ate the grass".

---

[1]Here, the `STEAK.N` predicate is reified into an abstract individual—the kind of food, steak—by the `K` operator so it can be used as an argument of the `EAT.V` predicate.

3.1 Overall Structure

A schema is represented by its *header*, seen in line 1 of Figure 2. A schema's header is an EL proposition and an episode characterized by the proposition, here `?E`. The header episode summarizes the entire schema, and can be used to embed a schema as a step inside another schema.

The rest of the schema is laid out in two types of sections: *fluent* and *nonfluent* sections. Nonfluent sections such as `Roles` and `Episode-relations` contain formulas that hold true regardless of time, such as the types or physical properties of objects. Fluent sections such as `Steps` and `Preconds` contain formulas whose truth values are time-dependent, such as an action taken by someone. We will now examine these sections, and what they're used for, in more detail.

3.2 Roles

The **Roles** section of a schema is a *nonfluent* section meant for putting "eternal" type constraints on the participating entities in the schema. In addition to type constraints, e.g. `(?X DOG.N)`, nonfluent relational constraints between entities can also be specified in this section, e.g. `(?X PERTAINS_TO.N ?Y)`.

When individuals from story formulas are bound to slot variables in the schema, these "type" constraints are evaluated to judge how well the individuals fit those slots. Some constraints may be broken—this is a key part of the generalization process—but the soft scoring metric in Section 4.3.1 helps identify good matches.

3.3 Preconditions, Postconditions, and Goals

Schemas specify preconditions, postconditions, and goals characterize the motivations of the agents involved. Fluent constraints in the precondition section are tacitly assumed to start before the schema's header episode (adjoining or overlapping it), and those in the postcondition section extend beyond the header episode (post-adjoining or overlapping it). Schema matches can be "chained together" into composite, multi-step schemas by unifying their pre- and postconditions, or their goals and postconditions. The schema in Figure 2 examplifies a learned "chained" schema.

3.4 Temporal Relations

The episodes characterized by fluent formulas within the body of a schema can all be complexly interrelated using constraints from the Allen Interval Algebra (Allen, 1983) as well as causal and quantitative temporal constraints. Pre- and post-conditions are implicitly constrained to be true at the start and end of the schema's header episode, respectively, and steps, by default, are ordered sequentially as listed in the schema, but additional constraints can be specified in the **Episode-relations** section of each schema. To evaluate these interval constraint propositions, we implemented a time graph specialist module (Gerevini and Schubert, 1993). The time graph models the temporal projection of each episode as a pair of time points, corresponding to the beginning and end of the episode. The time graph has time points as vertices, and an edge between t_1 and t_2 if $t_1 \leq t_2$. Then, querying the graph for propositions can be done with a graph transversal. The time graph also keeps track of "chains", which are long consecutive sequences of time points in the graph. This allows the module to exploit the often linear structure of stories, and it achieves high efficiecy on the subalgebra of Allen's Interval Algebra that can be expressed in terms of $\leq$ point-relations.

4 Schema Learning

In this section, we describe how our system learns new schemas from natural language stories. We describe our story parsing process, the process of matching parsed stories to schemas, how schema matches can be generalized to create new schemas, and how partial schema matches can be used to predict events in similar stories with missing details.

4.1 The Protoschema Approach

As noted, we generate new schemas from stories by starting with an initial set of *protoschemas* that we would expect a 1- or 2-year-old child to have; these encode very general knowledge about physical and communicative actions, with their preconditions and effects. Examples of protoschemas we've already written include movement of an agent from one location to another, consumption of food, and possession and transfer of possession. These protoschemas are then invoked by actions in stories—for example, the "travel" protoschema matched a "climb" action in a story to yield a "monkey climbs a tree" schema, which was eventually incorporated

as the first step of the chained schema in Figure 2.[2]

4.2 Story Parsing

We first process raw stories with the AllenNLP coreference analyzer (Gardner et al., 2017), and then use the first stage of the BLLIP parser (Charniak, 2000) for an initial syntactic parse. The syntactic parse is then converted to *Unscoped Logical Form* (ULF) (Kim and Schubert, 2019), an underspecified variant of EL, by tree transductions, and then a second transduction phase processes the ULF into full EL.

Our current parsing pipeline converts about 50 percent of (very brief, typically 2-5 sentence) stories to valid Episodic Logic formulas; our rules cannot transduce some grammatical features into ULF, including quotations and rhetorical questions. Kim (2019) is investigating direct English-to-ULF conversion using a cache transition parser, and we hope that this approach will boost our parsing accuracy.

4.3 Matching

Matching formulas in semantically parsed stories to formulas in schemas underlies both learning and prediction. The formulas comprising a schema are intended to be relatively simple—with complex conjunctions split into separate formulas—and unifiable with formulas parsed from real stories. Unification of a story formula with a schema formula binds individual constants from the former to variables in the latter. These bindings are then substituted in the rest of the schema instance, thereby "filling in" some of the missing information. This information is likely to be correct if the story events and participant types matched to the schema can be assumed to provide good evidence for an occurrence of the stereotyped pattern of events the schema captures. We refer to any schema instance with one or more bound variables as a *match*.

Using EL formula unification as a primitive, we implement schema matching by iterating through the formulas in an EL parse of a story, matching each formula to any schema formula retrieved as a candidate, and applying the bindings to the schema. When the story has been fully iterated through, or all schema variables have been bound, the match is complete.

We randomly permute story formulas and unify them, in the randomized order, with schema formulas. We try multiple permutations to explore the space of possible matches, and cache low-level unification results to speed up the process.

4.3.1 Partial Matches and Scoring

When a schema is matched to a story, some constraints may be broken; this is a natural part of the learning process. A schema for a cow eating grass matched to a story about a dog eating grass violates the cow constraint on a participating entity, but is a valuable source of knowledge if properly generalized. On the other hand, too many broken constraints are indicative of a poor match between a schema candidate and a story.

Schema matches are heuristically scored by counting satisfied constraints, weighted by constraint type. Confirmed Role constraints are worth half as many points as confirmed events in the Steps section. Confirming the schema's header formula is worth twice the points of any other event.

For inexact matches—e.g., (?X COW.N) and (ROVER.NAME DOG.N)—the score of the binding is further weighted by the approximate semantic similarity of the two words. If one subsumes the other in a hypernym hierarchy, the strength is scaled by the distance of the two in that hierarchy. If neither subsumes the other, but they share a common ancestor hypernym, the strength is half their average distance to that ancestor.

The hypernym score accounts for half of the overall weight of an inexact match; the other half is provided by their semantic similarity according to a pre-trained word embedding model. [3]

4.4 Generalizing Matches

To generalize a match into a new, "learned" schema, we need to incorporate incidental information about the matched value. For example, the variables of the travel.v protoschema can be bound by the constants in the formula ((MONKEY27.SK (CLIMB.V TREE28.SK)) ** E34.SK) in a story about a monkey climbing a tree, but regeneralizing the constants MONKEY27.SK and TREE28.SK into unconstrained variables would remove all the information we learned. However, if we incorporate formulas about the types of those objects into our new schema—such as the formulas (MONKEY27.SK MONKEY.N) and

---

[2]"travel" was invoked by "climb" by way of the WordNet hypernym hierarchy.

[3]*GoogleNews-vectors-negative300.bin*, Mikolov et al. (2013)

(`TREE28.SK TREE.N`)—we can then general-
ize the constants but maintain knowledge of their
types.

4.4.1 Re-Matching Learned Schemas

Once a protoschema has been matched to a story
and generalized into a learned schema, it may con-
tain extraneous details or overly specific constraints.
To filter out such details or constraints, we search
for at least one more match of the learned schema to
another story, downgrading details and constraints
that were not matched again. To learn (potentially)
more abstract versions of learned schemas, we re-
tain both basic types and generalized types in the
abstract versions, with certainties reflecting their
match frequencies.

4.5 Prediction

Prediction is relatively straightforward: Given a
story, we try to identify a similar schema, such as
the learned schema in Figure 2, and match as many
formulas as we can to it. We find similar schemas
by average pairwise distance between story words
and schema word predicates in the pre-trained word
vector space. After we've substituted story entities
for variables, we may fill in other formulas in the
schema. Schema formulas whose variables have
all been filled in, but are not present in the story,
are predictions: in effect, we guess that the schema
underlies the observed events, and infer further
aspects of the situation from its explicitly provided
aspects.

5 Results

Using 511 simple stories taken from a children's
first reader (McGuffey, 1901) and the ROCstories
corpus (Mostafazadeh et al., 2017), and 13 proto-
schemas[4], we obtained 665 schemas, with a mean
score of -0.899, a median score of 0.292, a mini-
mum score of -19.304, and a maximum score of 4.5
according to the scoring metric in Section 4.3.1. Af-
ter filtering out the 314 negative-scoring schemas,
we obtained 314 "specified" schemas, including
six multi-step schemas, examples of which can be
found in Figure 1 and Figure 2.

---

[4]These 13 protoschemas, including traveling from place
to place, eating food, taking possession of an object, and
searching for something, were selected to cover a large number
of sentences in a "development set" of 50 held-out stories from
our corpus of 561 stories; 511 were used in the test set. We
intend to eventually construct dozens to hundreds of initial
protoschemas.

```
1  (EPI-SCHEMA ((?X_B CLIMB_GET_EAT.PR
2                ?X_A ?X_C) ** ?E)
3    :ROLES
4      !R1 (?X_A TREE.N)
5      !R2 (?X_C INANIMATE_OBJECT.N)
6      !R3 (?X_B MONKEY.N)
7      !R4 (?X_C FOOD.N)
8      !R5 (?X_C COCOANUT.N)
9    :STEPS
10     ?E1 (?X_B CLIMB.481.V
11             (FROM.P-ARG ?L1) ?X_A)
12     ?E2 (?X_B GET.511.V ?X_C
13             (AT.P-ARG ?X_A))
14     ?E3 (?X_B EAT.541.V ?X_C)
15   :EPISODE-RELATIONS
16     !W1 (?E1 BEFORE ?E2)
17     !W2 (?E2 BEFORE ?E3)
18     !W3 (?E1 DURING ?E)
19     !W4 (?E2 DURING ?E)
20     !W5 (?E3 DURING ?E)
21 )
```

Figure 2: An example of a multi-step schema learned
by our system from protoschema matches to a story
about a monkey climbing a tree to get and eat a co-
coanut.

The schema in Figure 2 inferred, given the sen-
tences "Simeon can climb the tree" and "He gets
the cocoanuts for his mother", that Simeon was a
monkey, that he got the cocoanuts in the tree, and
that he later ate the cocoanuts. The schema in Fig-
ure 1 inferred, given the sentences "The bees like
it", "They find sweet nectar in the clover flowers",
and "It grows in the fields", that the bees went to
the fields to find the nectar. These predictions about
unseen stories are reasonable and fill in details ab-
sent in the stories themselves.

6 Future Work

The schemas learned and predictions generated by
the system with only 13 protoschemas are encour-
aging; we've obtained many simple schemas, like
"person sits in a chair" or "dogs run around out-
side", as well as complex, multi-step schemas used
for predictions like the ones in Section 5. Because
complex schemas are made by stringing together
protoschema matches, we plan to develop more
protoschemas—possibly dozens to hundreds—to
more fully cover the general knowledge of a two-
year-old child. With those protoschemas as a base,
we expect to generate many more useful, multi-step
schemas, use them to generate predictions about
stories, and have human judges evaluate those pre-
dictions.

Acknowledgments

References

James F Allen. 1983. Maintaining knowledge about temporal intervals. *Communications of the ACM*, 26(11):832–843.

Nathanael Chambers and Dan Jurafsky. 2011. Template-based information extraction without the templates. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 976–986. Association for Computational Linguistics.

Eugene Charniak. 2000. A maximum-entropy-inspired parser. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 132–139. Association for Computational Linguistics.

Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke S. Zettlemoyer. 2017. Allennlp: A deep semantic natural language processing platform.

Alfonso Gerevini and Lenhart Schubert. 1993. Efficient temporal reasoning through timegraphs. In *IJCAI*, pages 648–654.

Gene Kim, Benjamin Kane, Viet Duong, Muskaan Mendiratta, Graeme McGuire, Sophie Sackstein, Georgiy Platonov, and Lenhart Schubert. 2019. Generating discourse inferences from unscoped episodic logical formulas. In *Proceedings of the First International Workshop on Designing Meaning Representations*, pages 56–65.

Gene Kim and Lenhart Schubert. 2019. A type-coherent, expressive representation as an initial step to language understanding. In *Proceedings of the 13th International Conference on Computational Semantics*, Gothenburg, Sweden. Association for Computational Linguistics.

Gene Louis Kim. 2019. 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*.

Michael Lebowitz. 1980. *Generalization and Memory in an Integrated Understanding System*. Ph.D. thesis, New Haven, CT, USA. AAI8109800.

Zhiyi Luo, Yuchen Sha, Kenny Q Zhu, Seung-won Hwang, and Zhongyuan Wang. 2016. Commonsense causal reasoning between short texts. In *Fifteenth International Conference on the Principles of Knowledge Representation and Reasoning*.

William Holmes McGuffey. 1901. *The New McGuffey First Reader*. American Book Company.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, pages 3111–3119. Curran Associates, Inc.

Raymond J Mooney. 1990. *A general explanation-based learning mechanism and its application to narrative understanding*. Morgan Kaufmann.

Nasrin Mostafazadeh, Chris Brockett, Bill Dolan, Michel Galley, Jianfeng Gao, Georgios Spithourakis, and Lucy Vanderwende. 2017. Image-grounded conversations: Multimodal context for natural question and response generation. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 462–472, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Karl Pichotta and Raymond Mooney. 2016a. Statistical script learning with recurrent neural networks. In *Proceedings of the Workshop on Uphill Battles in Language Processing: Scaling Early Achievements to Robust Methods*, pages 11–16. Association for Computational Linguistics.

Karl Pichotta and Raymond J. Mooney. 2016b. Using sentence-level LSTM language models for script inference. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 279–289, Berlin, Germany. Association for Computational Linguistics.

Lenhart Schubert. 2014. Nlog-like inference and commonsense reasoning. *LiLT (Linguistic Issues in Language Technology)*, 9.

Lenhart K. Schubert and Chung Hee Hwang. 2000. Episodic Logic meets Little Red Riding Hood: A comprehensive natural representation for language understanding. In Lucja M. Iwańska and Stuart C. Shapiro, editors, *Natural Language Processing and Knowledge Representation*, pages 111–174. MIT Press, Cambridge, MA, USA.

Karl Stratos, Lenhart Schubert, and Jonathan Gordon. 2011. Episodic logic: Natural logic + reasoning. *KEOD 2011 - Proceedings of the International Conference on Knowledge Engineering and Ontology Development*.

Quan Yuan, Xiang Ren, Wenqi He, Chao Zhang, Xinhe Geng, Lifu Huang, Heng Ji, Chin-Yew Lin, and Jiawei Han. 2018. Open-schema event profiling for massive news corpora. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, pages 587–596, New York, NY, USA. ACM.