



Towards Learning Rich Logical Schemas From Natural Language Stories

Lane Lawley

llawley@cs.rochester.edu
cs.rochester.edu/u/llawley

Gene Kim

gkim21@cs.rochester.edu
cs.rochester.edu/u/gkim21

Lenhart Schubert

schubert@cs.rochester.edu
cs.rochester.edu/u/schubert



Talk Outline

1. A little background
2. Overview of our approach
3. Schema matching
4. Schema generalization
5. Conclusion



Talk Outline

1. A little background
2. Overview of our approach
3. Schema matching
4. Schema generalization
5. Conclusion



Background

Schemas are *stereotypical patterns of events* that can be *instantiated* into specific stories

Schank & Abelson's **scripts**, Minsky's **frames**...



Background: uses

Inferences from incomplete information:

“Skid, crash, hospital”

What skidded and crashed? Who was in the hospital?

Planning:

“I’m hungry. What are some ways I know for someone to get food?”



Background: learning from natural language

GENESIS: used database of actions, preconds, effects to abstract causal chains

(Mooney, 1991)

Cluster verb/subj and verb/obj pairs w/ co-referring args, use SVM for ordering

(Chambers & Jurafsky, 2008)

LSTMs to find sequences of 5-tuples (verb, subj, obj, preposition, prep arg)

(Pichotta & Mooney, 2016)



Talk Outline

1. A little background
2. Overview of our approach
3. Schema matching
4. Schema generalization
5. Conclusion



How does our approach differ?

Statistical script learning uses simple event representations...

(...and requires a lot of data.)

We use **episodic logic** as our representation language.

EL is closer to “surface-form” English than FOL, & much more expressive than 2- or 5-tuples.



How does our approach differ?

Symbolic approaches to schema learning have (historically) had bootstrapping issues.

- GENESIS (Mooney, 1991) relied on many pre-programmed actions like “ARREST” to understand its stories, which were targeted at adults. These actions were fully specified in FOL w/ preconditions, postconditions, etc.
- IPP (Lebowitz, 1983) had a similar pre-programmed action set, & targeted news articles.

We target **children’s stories** and bootstrap with only **a set of initial schemas** that any 1- or 2-year old child would possess, covering wide sets of actions (do action for pleasure, help someone do an action...).

We can also **compare two similar stories** to find *common* components of a schema, rather than inferring *necessary* ones from fully-specified causal chains in one story.



Representation: EL/ULF

English:

Rivka insisted that she didn't eat chametz.

Episodic Logic (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)
```

E1 is an episode of Rivka insisting
that there is some E2, *and*

E2 is an episode of Rivka not eating
chametz, *and*

E2 happened at or before E1.



Representation: EL/ULF

English:

Rivka insisted that she didn't eat chametz.

Episodic Logic (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)
```

Underspecified Logical Form (ULF):

```
(|Rivka| ((past insist.v)
  (that (she.pro ((past do.aux-s)
    not (eat.v (k chametz.n)))))))
```

**Tense information not “deindexed”
into explicit episodes like E1 & E2...**


Example Schema

```
(epi-schema ((?x do_to_enable_action.v ?a1 ?a2) ** ?e)
  (:Nonfluent-conds
    !r1 (?a1 action1.n)
    !r2 (?a2 action1.n)
    !r3 (?x agent6.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)
    !w2 (?e1 cause-of ?p1)))
```



Talk Outline

1. A little background
2. Overview of our approach
3. Schema matching
4. Schema generalization
5. Conclusion



```
(epi-schema ((?x do_to_enable_action.v ?a1 ?a2) ** ?e)
  (:Nonfluent-conds
    !r1 (?a1 action1.n)
    !r2 (?a2 action1.n)
    !r3 (?x agent6.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)
    !w2 (?e1 cause-of ?p1)))
```



; The monkey can climb a tree.

```
((E1.SK AT-ABOUT.P NOW0) ^  
  (MONKEY1.SK MONKEY.N) ^  
  (TREE1.SK TREE.N) ^  
  ((MONKEY1.SK (CAN.MD (CLIMB.V TREE1.SK))) ** E1.SK))
```

; He climbs the tree and gets a cocoanut.

```
((E2.SK AT-ABOUT.P NOW1) ^  
  (TREE2.SK TREE.N) ^  
  ((HE.PRO (CLIMB.V TREE2.SK)) ** E2.SK))
```

BEST SCHEMA MATCH:

(7 points; 2 from # of consistent temporal constraints, 5 from # of bound variables)

```
(epi-schema ((MONKEY1.SK DO_TO_ENABLE_ACTION.V (KA (CLIMB.V TREE2.SK))
             (KA (GET.V COCONUT1.SK)))
            ** ?E)
 (:NONFLUENT-CONDS
  !R1 ((KA (CLIMB.V TREE2.SK)) ACTION1.N)
  !R2 ((KA (GET.V COCONUT1.SK)) ACTION1.N)
  !R3 (MONKEY1.SK AGENT6.N)
)
 (:GOALS
  ?G1 (MONKEY1.SK WANT1.V
       (THAT (MONKEY1.SK CAN.MD (DO2.V (KA (GET.V COCONUT1.SK))))))
)
 (:INIT-CONDS
  ?I1 (NOT (MONKEY1.SK (CAN.MD (DO2.V (KA (GET.V COCONUT1.SK))))))
)
 (:STEPS
  E2.SK (MONKEY1.SK DO2.V (KA (CLIMB.V TREE2.SK)))
)
 (:POST-CONDS
  E3.SK (MONKEY1.SK (CAN.MD (DO2.V (KA (GET.V COCONUT1.SK))))))
)
 (:EPISODE-RELATIONS
  !W1 (E2.SK CONSEC E3.SK)
  !W2 (E2.SK CAUSE-OF E3.SK)
  !W3 (E2.SK CAUSE-OF E3.SK)
))
```

```
(epi-schema ((?x do_to_enable_action.v ?a1 ?a2) ** ?e)
 (:Nonfluent-conds
  !r1 (?a1 action1.n)
  !r2 (?a2 action1.n)
  !r3 (?x agent6.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)
  !w2 (?e1 cause-of ?p1)))
```

BINDINGS:

```
?X: MONKEY1.SK
?A1: (KA (CLIMB.V
TREE2.SK))
?E1: E2.SK
?A2: (KA (GET.V
COCONUT1.SK))
?E2: E3.SK
```




Talk Outline

1. A little background
2. Overview of our approach
3. Schema matching
4. Schema generalization
5. Conclusion



Generalizing protoschema matches

Starting protoschema: *“agent does action to enable action”*

New schema: *“monkey climbs tree to get coconut”*

We could generalize a bit...

“mammal climbs tree to get fruit”



Generalizing protoschema matches

Starting protoschema: *“agent does action to enable action”*

New schema: *“monkey climbs tree to get coconut”*

We could generalize a bit...

“mammal climbs tree to get fruit”

But...

“vertebrate climbs plant to get ingredient”



Two issues (of many...)

1. Combinatorial explosions of possible generalizations
2. Not all schemas are “refinements” of protoschemas



Lazy composite generalization

Lazy: We could hold off on generalizing until we see a second, similar story, to narrow the combinatorial generalization space

Composite: Incorporate some “incidental” information into a new schema, even when it doesn’t match an existing schema/protoschema



Lazy composite generalization example

Simeon and Pedro like to fish. Sometimes they sit on the bridge. Sometimes they sit on the bank of the river. They have poles, long lines, and little iron hooks. This morning Simeon caught a large fish. Pedro caught many small ones. The boys caught some crabs, too. They use a net to catch the crabs.

There are fish in their pond. They are very nice fish. We will come and catch them. We will take the long rod, and the hook and line. We must have a bag, too. It must be strong, to keep the fish safe.



Roles

Simeon and Pedro like to fish. Sometimes they sit on the bridge. Sometimes they sit on the bank of the river. They have poles, long lines, and little iron hooks. This morning Simeon caught a large fish. Pedro caught many small ones. The boys caught some crabs, too. They use a net to catch the crabs.

There are fish in their pond. They are very nice fish. We will come and catch them. We will take the long rod, and the hook and line. We must have a bag, too. It must be strong, to keep the fish safe.



Shared roles

Simeon and Pedro like to **fish**. Sometimes they sit on the bridge. Sometimes they sit on the bank of the river. They have poles, long **lines**, and little iron **hooks**. This morning Simeon caught a large fish. Pedro caught many small ones. The boys caught some crabs, too. They use a net to catch the crabs.

There are **fish** in their pond. They are very nice fish. We will come and catch them. We will take the long rod, and the **hook** and **line**. We must have a bag, too. It must be strong, to keep the fish safe.



What about fish/crab, pond/river?

Simeon and Pedro like to fish. Sometimes they sit on the bridge. Sometimes they sit on the bank of the river. They have poles, long lines, and little iron hooks. This morning Simeon caught a large fish. Pedro caught many small ones. The boys caught some crabs, too. They use a net to catch the crabs.

There are fish in their pond. They are very nice fish. We will come and catch them. We will take the long rod, and the hook and line. We must have a bag, too. It must be strong, to keep the fish safe.



What about fish/crab, pond/river?

LCH(FISH, CRAB) = **ANIMAL**

LCH(POND, RIVER) = **BODY_OF_WATER**

LCH(SIMEON, PEDRO, WE, BOYS, THEY) = **PERSON**



Lazy composite generalization

Sentences with generalized terms:

1. PERSONs like to fish
2. Sometimes PERSONs sit on bridge
3. Sometimes PERSONs sit on bank of BODY_OF_WATER
4. PERSONs have poles, long LINEs, little iron HOOKs
5. PERSON caught large FISH
6. PERSONs use net to catch ANIMAL
7. FISH in BODY_OF_WATER
8. FISH are very nice
9. PERSONs will catch FISH
10. PERSONs will take long rod, HOOK, and LINE
11. PERSONs must have bag
12. Bag must be strong to keep FISH safe



Lazy composite generalization

Sentences with generalized terms:

1. PERSONs like to fish
2. Sometimes PERSONs sit on bridge
3. Sometimes PERSONs sit on bank of BODY_OF_WATER
4. PERSONs have poles, long LINEs, little iron HOOKs
- 5. PERSON caught large FISH**
6. PERSONs use net to catch ANIMAL
7. FISH in BODY_OF_WATER
8. FISH are very nice
- 9. PERSONs will catch FISH**
10. PERSONs will take long rod, HOOK, and LINE
11. PERSONs must have bag
12. Bag must be strong to keep FISH safe



Lazy composite generalization + protoschemas

Sentences with generalized terms:

1. PERSONs like to fish
2. Sometimes PERSONs sit on bridge
3. Sometimes PERSONs sit on bank of BODY_OF_WATER
4. **PERSONs have** poles, long **LINEs**, little iron **HOOKs**
5. **PERSON** caught large **FISH**
6. PERSONs use net to catch ANIMAL
7. FISH in BODY_OF_WATER
8. FISH are very nice
9. **PERSONs will catch FISH**
10. **PERSONs will take** long rod, **HOOK**, and **LINE**
11. PERSONs must have bag
12. Bag must be strong to keep FISH safe

```
(EPI-SCHEMA ((?X TAKE_TO_POSSESS.V ?O) ** ?E)
```

```
...
```

```
)
```



Current working schema...

ROLES

?x PERSON.N

?1 LINE.N

?h HOOK.N

?f FISH.N

?b BODY_OF_WATER.N

EVENTS

(?x TAKE_TO_POSSESS.V ?h)

(?x TAKE_TO_POSSESS.V ?1)

(?x CATCH.V ?f)



Transitive co-presence for “body of water”?

Simeon and Pedro like to fish.
Sometimes they sit on the bridge.
**Sometimes they sit on the bank
of the river. They have poles,
long lines, and little iron hooks.**
This morning Simeon caught a
large fish. Pedro caught many small
ones. The boys caught some crabs,
too. They use a net to catch the
crabs.

**There are fish in their pond.
They are very nice fish. We
will come and catch them.**
We will take the long rod, and
the hook and line. We must
have a bag, too. It must be
strong, to keep the fish safe.



Another fishing story

Simeon and Pedro like to fish. Sometimes they sit on the bridge. Sometimes they sit on the bank of the river. They have poles, long lines, and little iron hooks. This morning Simeon caught a large fish. Pedro caught many small ones. The boys caught some crabs, too. They use a net to catch the crabs.

There are fish in their pond. They are very nice fish. We will come and catch them. We will take the long rod, and the hook and line. We must have a bag, too. It must be strong, to keep the fish safe.

Boys like to catch fish. It is a good sport. Here is Tom with his rod and line. A hook is on the end of the line. He has a bag, too. The bag is to put the fish in. Here is the fish for Tom to catch. It swims with its tail. It can swim very fast.



More shared roles

Simeon and Pedro like to fish. Sometimes they sit on the bridge. Sometimes they sit on the bank of the river. They have poles, **long lines**, and little iron hooks. This morning Simeon caught a large fish. Pedro caught many small ones. The boys caught some crabs, too. They use a net to catch the crabs.

There are fish in their pond. They are very nice fish. We will come and catch them. We will take the **long rod**, and the hook and **line**. **We must have a bag, too**. It must be strong, to keep the fish safe.

Boys like to catch fish. It is a good sport. **Here is Tom with his rod and line**. A hook is on the end of the line. **He has a bag, too. The bag is to put the fish in**. Here is the fish for Tom to catch. It swims with its tail. It can swim very fast.

```
(EPI-SCHEMA ((?X DO_FOR_PLEASURE.V ?A) ** ?E)
...
)
```

And a protoschema refinement...?

Simeon and Pedro like to fish. Sometimes they sit on the bridge. Sometimes they sit on the bank of the river. They have poles, long lines, and little iron hooks. This morning Simeon caught a large fish. Pedro caught many small ones. The boys caught some crabs, too. They use a net to catch the crabs.

There are fish in their pond. They are very nice fish. We will come and catch them. We will take the long rod, and the hook and line. We must have a bag, too. It must be strong, to keep the fish safe.

Boys like to catch fish. It is a good sport. Here is Tom with his rod and line. A hook is on the end of the line. He has a bag, too. The bag is to put the fish in. Here is the fish for Tom to catch. It swims with its tail. It can swim very fast.



Talk Outline

1. A little background
2. Overview of our approach
3. Schema matching
4. Schema generalization
5. Conclusion



Conclusions

- Using the *full* semantics of text is important for learning; there's a lot of meaning there!
- A relatively small initial set of schemas helps you understand a lot of behavior.
- You don't need hardcoded action specifications to learn patterns of actions, & causation can be inferred from protoschemas/correlation instead of hardcoded.



Questions?



Questions?



(Lane, stop hitting “next”, the appendices are next)



Appendix: term unification



Term unification (matching)

IN STORY:

(**BUDDY1.SK** buddy.n)

IN SCHEMA:

(**?y** friend.n)

Safe to match **BUDDY1.SK** to **?y**

“buddy” and “friend” are synonyms



Term unification (matching)

IN STORY:

(**DOG1.SK** dog.n)

IN SCHEMA:

(**?y** animal.n)

Safe to match **DOG1.SK** to **?y**

“dog” is a hyponym of “animal”



Term unification (matching)

IN STORY:

(**DOG1.SK** dog.n)

IN SCHEMA:

(**?y** border_collie.n)

NOT safe to match **DOG1.SK** to **?y**

“dog” is a hypernym of “border collie”



Term unification (matching)

IN STORY:

```
(HOOK1.SK (little.a (iron.n hook.n)))
```

IN SCHEMA:

```
(?y hook.n)
```

Safe to match; intersective modifiers
can be stripped!



Term unification (matching)

IN STORY:

(**FLAMINGO1.SK** (fake.a flamingo.n))

IN SCHEMA:

(**?y** flamingo.n)

NOT safe to match; not all modifiers
are intersective!



Term unification (generalization)

IN STORY 1:

`(?x (little.a (iron.n hook.n)))`

IN STORY 2:

`(?y hook.n)`

What's a safe generalization constraint for `?z`?

Should pick most general (`hook.n`), but “iron” could be useful info that was just *left out* of one story....



More term unification cases

`(plur hook.n)` and `hook.n`

`(plur (long.a line.n))` and `(plur line.n)`

`(trout.n bait.n)` and `(salmon.n bait.n)`



More term unification cases

(`plur hook.n`) and `hook.n`

`hook.n`

(`plur (long.a line.n)`) and `line.n`

`line.n`

(`trout.n bait.n`) and (`salmon.n bait.n`)

(`fish.n bait.n`)



More term unification cases: uncertainty?

(`plur hook.n`) and `hook.n`

`hook.n` 100% OR (`plur hook.n`) 50%

(`plur (long.a line.n)`) and `line.n`

(`trout.n bait.n`) and (`salmon.n bait.n`)

(`fish.n bait.n`) 100%



More term unification cases: uncertainty?

`(plur (long.a line.n))` and `line.n`

<code>line.n</code>	100%
<code>(plur line.n)</code>	50%
<code>(long.a line.n)</code>	50%
<code>(plur (long.a line.n))</code>	25%