A Method for Learning Schemas for Story Understanding and Inference

•••

Lane Lawley's talk

7/13/2020

Outline

1. Motivation
2. Prior work
3. Problem statement
4. Proposed research
   a. The protoschema approach
   b. Walkthrough of schema language
   c. Walkthrough of learning procedure
5. Evaluation
6. Timeline

Motivation

Genuine story understanding requires commonsense world knowledge.

Motivation

Genuine story understanding requires commonsense world knowledge.

I crashed my car.

When I finally left the hospital, all I wanted to do was sleep.

I had to call a taxi.

Motivation

Genuine story understanding requires commonsense world knowledge.

I crashed my car.
    (the car broke)
    (I was hurt)
    (I went to the hospital because I was hurt)
When I finally left the hospital, all I wanted to do was sleep.
    (it was late at night)
    (I wanted to go home to sleep)
I had to call a taxi.
    (to go home and sleep)
    (because my car was broken)

Motivation

“Schemas” package up related information about a situation.

They can be compared to movie scripts:

- Actors with defined roles
- Props with defined types
- Relationships between actors and props
- A sequence of events involving the actors and props

Piaget, Bartlett, and other psychologists believe schemas are a key form of human knowledge representation.

Motivation

Minsky (1974), Schank & Abelson (1977), and van Dijk & Kintsch (1978) all provide models of narrative understanding using schemas, also called “scripts” or “frames”.

“...the process of understanding a passage consists in finding a schema which will account for it.” (Rumelhart, 1976)

Motivation

Why should we think schemas are necessary to understand story meaning?

Motivation

"You shall know a word by the company it keeps." (Firth, 1957)

The **cat** likes milk.

I adopted a **cat** at the shelter.

A dog is similar to a **cat** in that it has four legs.

Motivation

“You shall know a word by the company it keeps.” (Firth, 1957)

The likes milk.
I adopted a cat at the shelter.
A dog is similar to a in that it has four legs.

“Distributional hypothesis”:

The meaning of a word is the probability distribution of the contexts of its use.

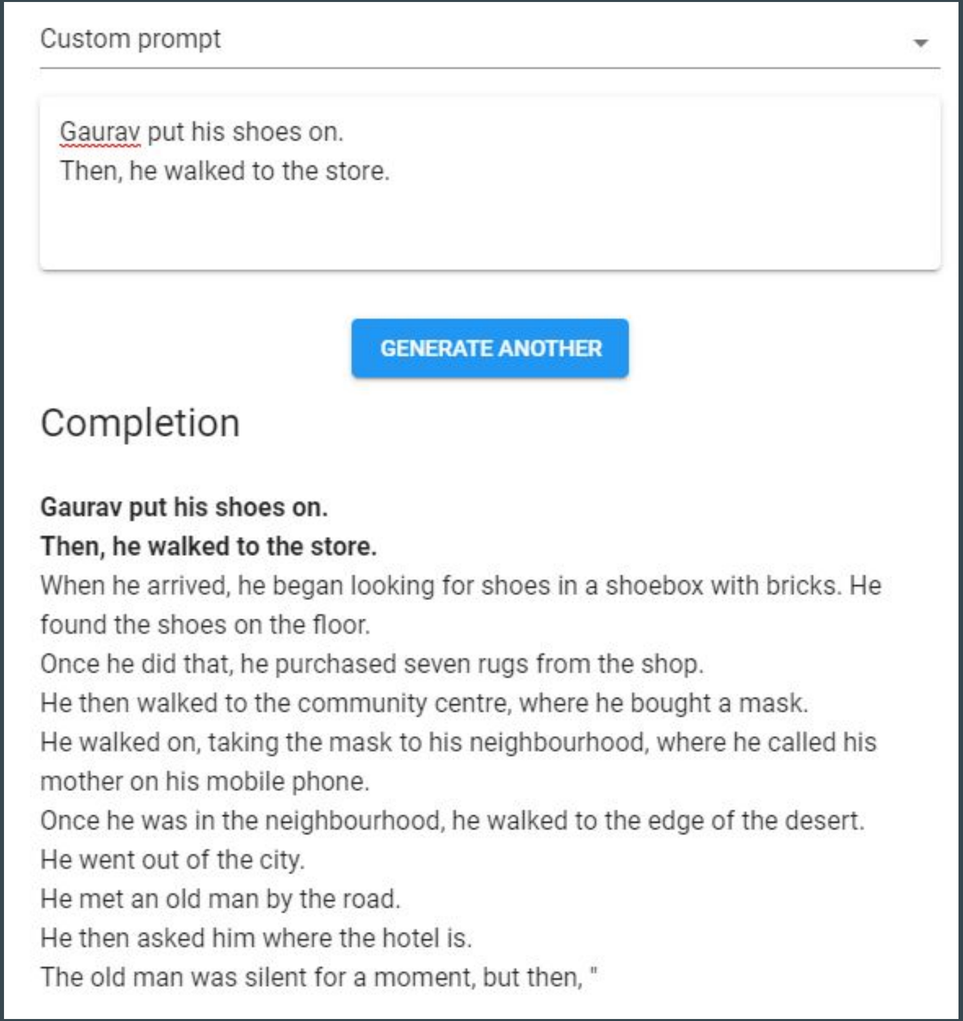
Motivation

an event?

“You shall know ~~a word~~ by the company it keeps.” ~~(Firth, 1957)~~

The likes milk.
I adopted a cat at the shelter.
A dog is similar to a in that it has four legs.

I got in my car.
I needed groceries.
I went to the store.
I got a shopping cart.
I bought a stuffed animal.



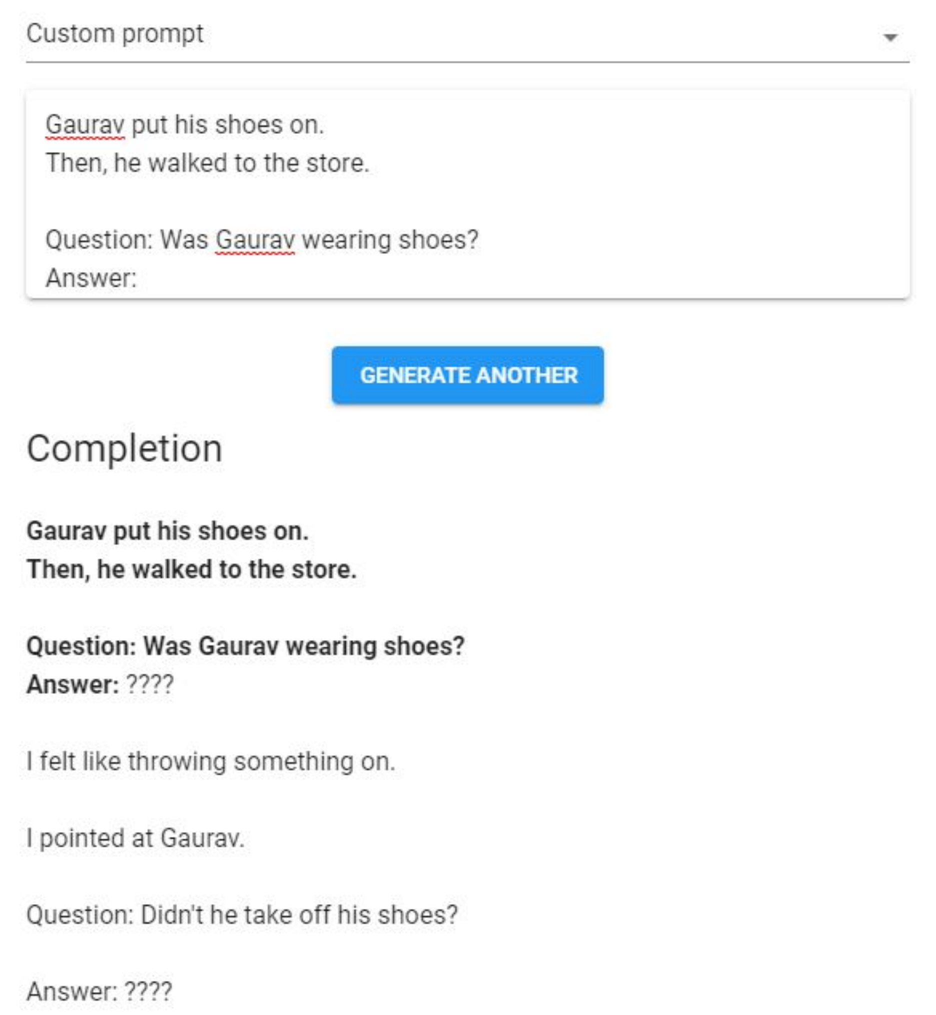
Gaurav put his shoes on.
Then, he walked to the store.

Completion

Gaurav put his shoes on.
Then, he walked to the store.
When he arrived, he began looking for shoes in a shoebox with bricks. He found the shoes on the floor.
Once he did that, he purchased seven rugs from the shop.
He then walked to the community centre, where he bought a mask.
He walked on, taking the mask to his neighbourhood, where he called his mother on his mobile phone.
Once he was in the neighbourhood, he walked to the edge of the desert.
He went out of the city.
He met an old man by the road.
He then asked him where the hotel is.
The old man was silent for a moment, but then, "

Very complex distributional text models (like GPT2, shown here) can predict story text.

But do they really understand <u>meaning</u>?



Gaurav put his shoes on.
Then, he walked to the store.

Question: Was Gaurav wearing shoes?
Answer:

Completion

Gaurav put his shoes on.
Then, he walked to the store.

Question: Was Gaurav wearing shoes?
Answer: ????

I felt like throwing something on.

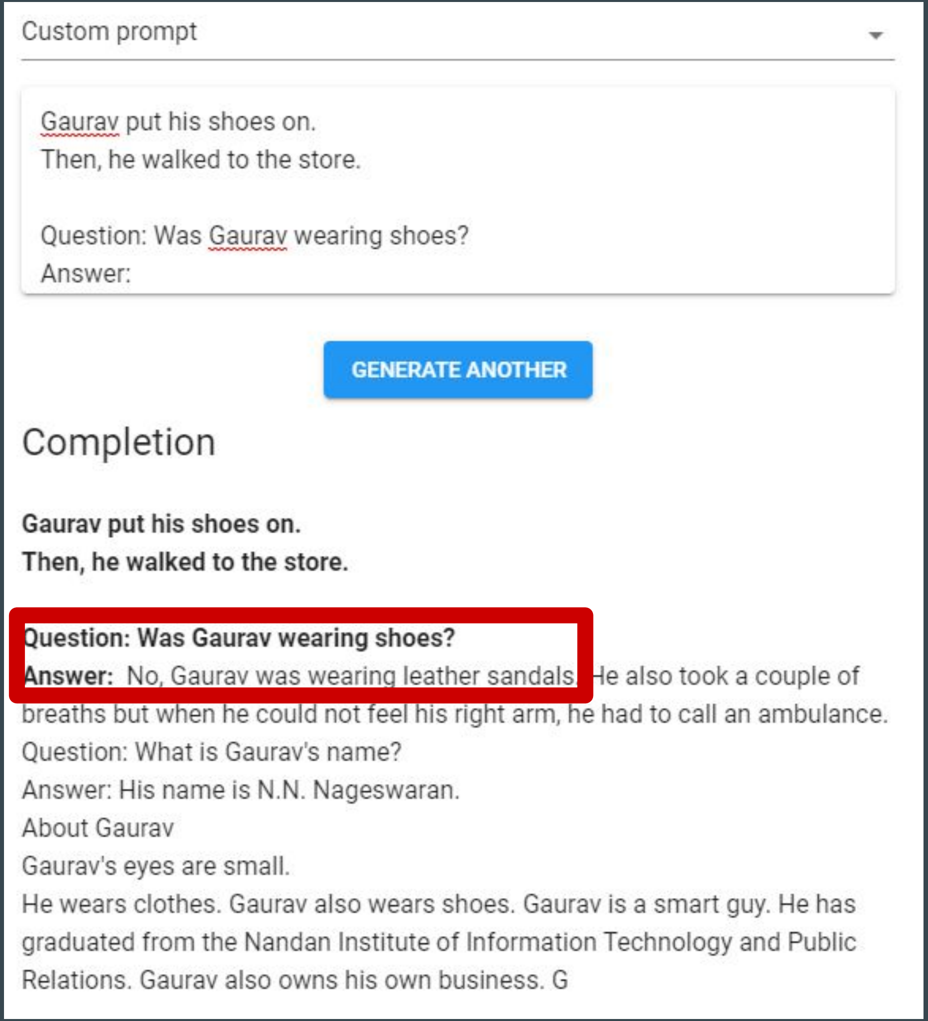
I pointed at Gaurav.

Question: Didn't he take off his shoes?

Answer: ????

These “black box” models are ultimately distributions over words.

Asking them questions, or assessing the semantics they’ve learned, is nontrivial.



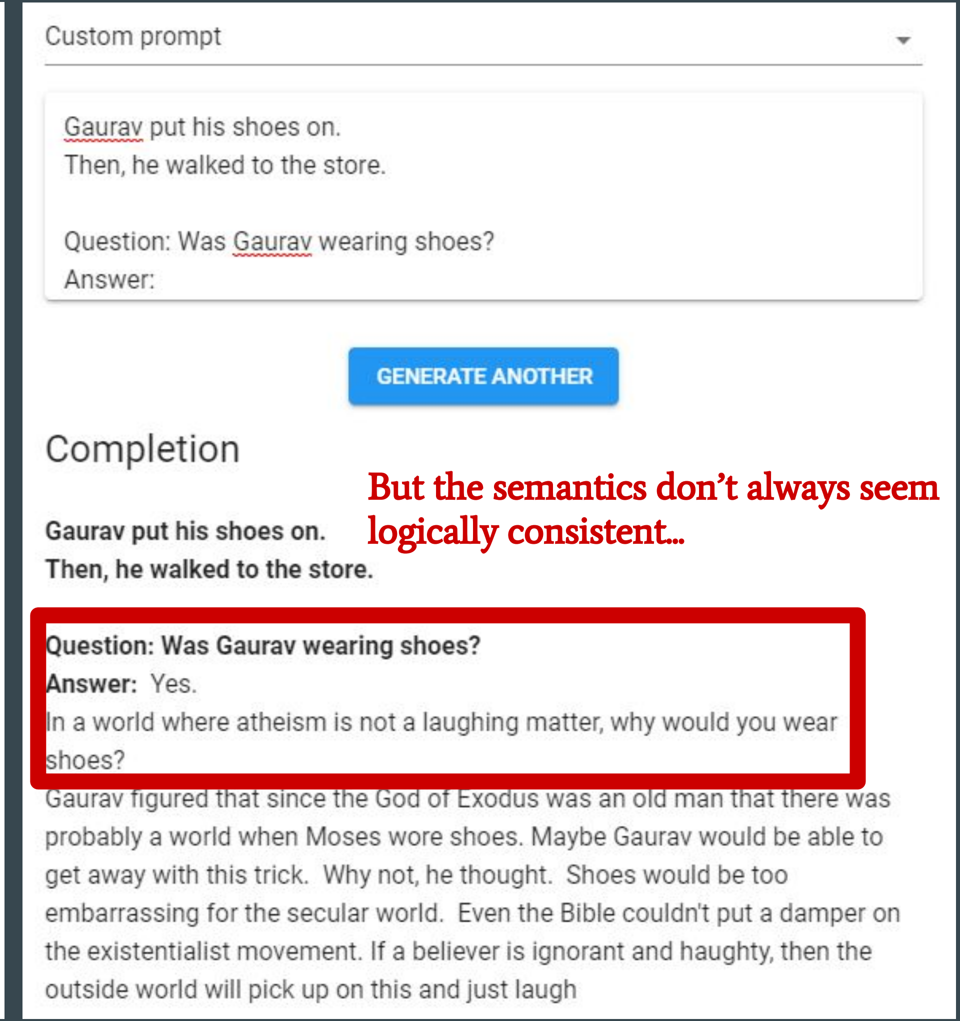
Gaurav put his shoes on.
Then, he walked to the store.

Question: Was Gaurav wearing shoes?
Answer:

Completion

Gaurav put his shoes on.
Then, he walked to the store.

Question: Was Gaurav wearing shoes?
Answer: No, Gaurav was wearing leather sandals. He also took a couple of breaths but when he could not feel his right arm, he had to call an ambulance.
Question: What is Gaurav's name?
Answer: His name is N.N. Nageswaran.
About Gaurav
Gaurav's eyes are small.
He wears clothes. Gaurav also wears shoes. Gaurav is a smart guy. He has graduated from the Nandan Institute of Information Technology and Public Relations. Gaurav also owns his own business. G



Gaurav put his shoes on.
Then, he walked to the store.

Question: Was Gaurav wearing shoes?
Answer:

Completion

Gaurav put his shoes on.
Then, he walked to the store.

Question: Was Gaurav wearing shoes?
Answer: Yes.
In a world where atheism is not a laughing matter, why would you wear shoes?
Gaurav figured that since the God of Exodus was an old man that there was probably a world when Moses wore shoes. Maybe Gaurav would be able to get away with this trick. Why not, he thought. Shoes would be too embarrassing for the secular world. Even the Bible couldn't put a damper on the existentialist movement. If a believer is ignorant and haughty, then the outside world will pick up on this and just laugh

But the semantics don't always seem logically consistent...

Motivation

General schemas let us draw inferences from incomplete information.

```
Restaurant schema:
     1. Person X is hungry
     2. X goes to restaurant Y
     3. X enters Y
     4. Host H shows X seat S
     5. X sits at seat S
     6. X asks waiter W for menu M

     etc...
```

Motivation

General schemas let us draw inferences from incomplete information.

```
Restaurant schema:
     1. Person X is hungry
     2. X goes to restaurant Y
     3. X enters Y
     4. Host H shows X seat S
     5. X sits at seat S
     6. X asks waiter W for menu M

     etc...
```

Motivation

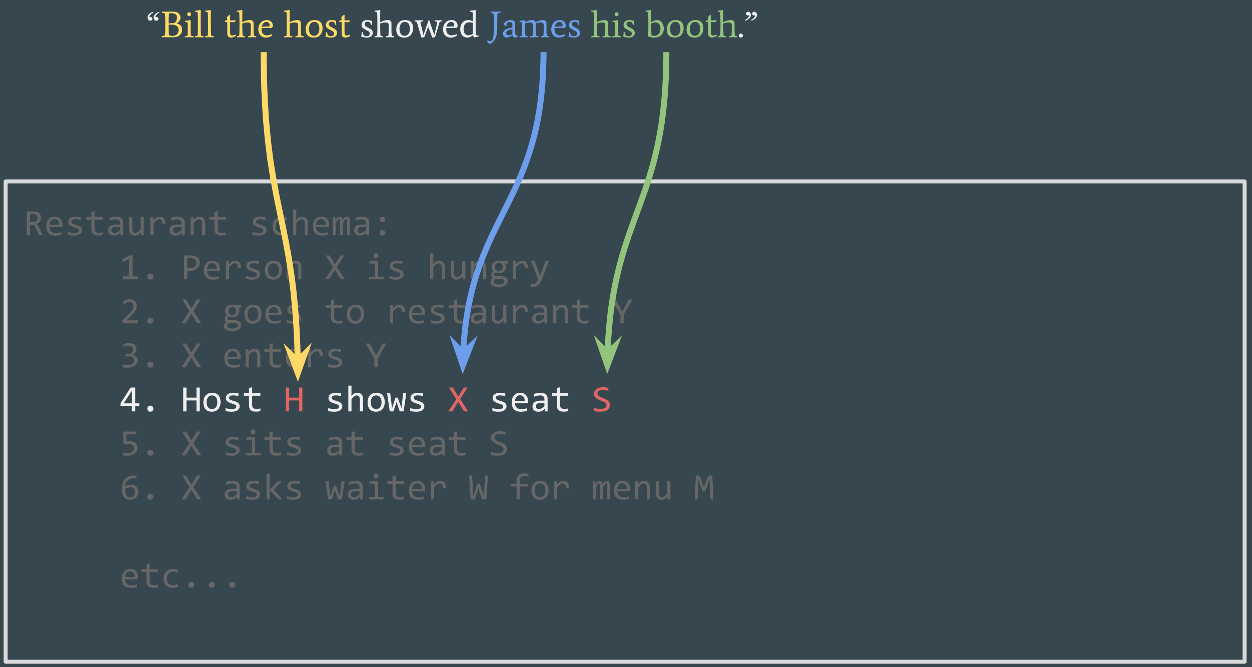
General schemas let us draw inferences from incomplete information.

“Bill the host showed James his booth.”

```
Restaurant schema:
    1. Person X is hungry
    2. X goes to restaurant Y
    3. X enters Y
    4. Host H shows X seat S
    5. X sits at seat S
    6. X asks waiter W for menu M

    etc...
```

Motivation

General schemas let us draw inferences from incomplete information.



"Bill the host showed James his booth."

```
Restaurant schema:
    1. Person X is hungry
    2. X goes to restaurant Y
    3. X enters Y
    4. Host H shows X seat S
    5. X sits at seat S
    6. X asks waiter W for menu M

    etc...
```

Motivation

General schemas let us draw inferences from incomplete information.

"Bill the host showed James his booth."

```
Restaurant schema:
    1. Person X is hungry
    2. X goes to restaurant Y
    3. X enters Y
    4. Host Bill shows James seat booth
    5. X sits at seat S
    6. X asks waiter W for menu M

    etc...
```

Motivation

General schemas let us draw inferences from incomplete information.

“Bill the host showed James his booth.”

```
Restaurant schema:
    1. Person James is hungry
    2. James goes to restaurant Y
    3. James enters Y
    4. Host Bill shows James seat booth
    5. James sits at seat booth
    6. James asks waiter W for menu M

    etc...
```

Motivation

General schemas let us draw **inferences** from incomplete information.

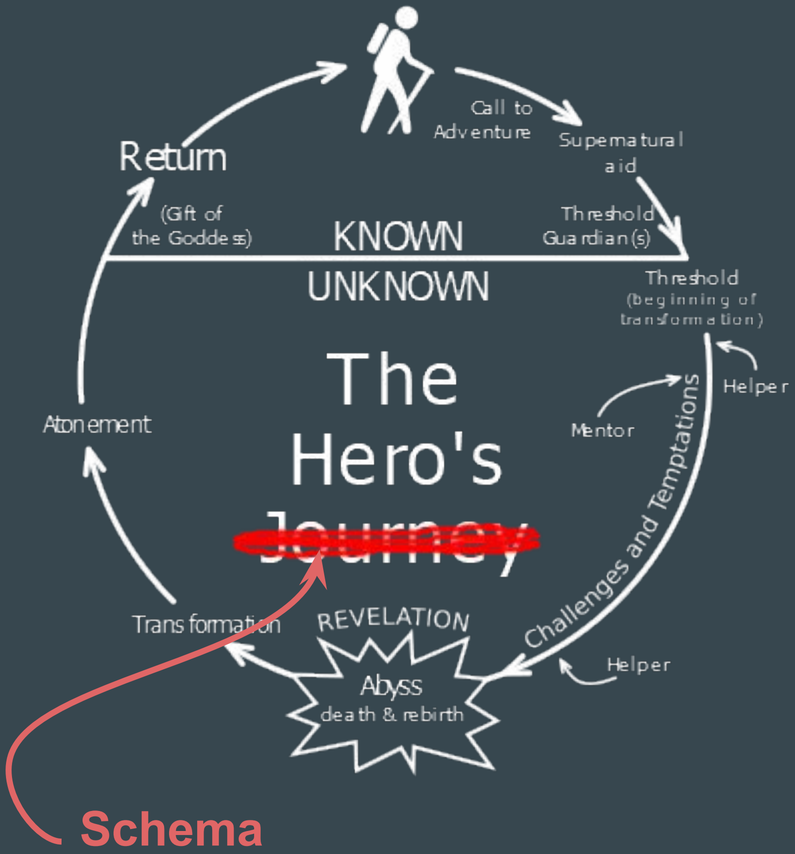
> “Bill the host showed James his booth.”

```
Restaurant schema:
     1. Person James is hungry
     2. James goes to restaurant Y
     3. James enters Y
     4. Host Bill shows James seat booth
     5. James sits at seat booth
     6. James asks waiter W for menu M

     etc...
```

Motivation

Schema steps can be entire sub-stories.

Schemas can represent everything from recipes to narratives.



Overview

We propose:

1. A schema language with features necessary for natural language, such as:
   a. Typed and inter-related entities
   b. Complex temporal relations
   c. Recursive nesting of schemas within other schemas
   d. Preconditions, postconditions, and agent motivations

2. A method for learning these schemas from stories based on the idea of "protoschemas"---an initial set of very abstract, very general behavioral schemas covering the knowledge of a very young child.

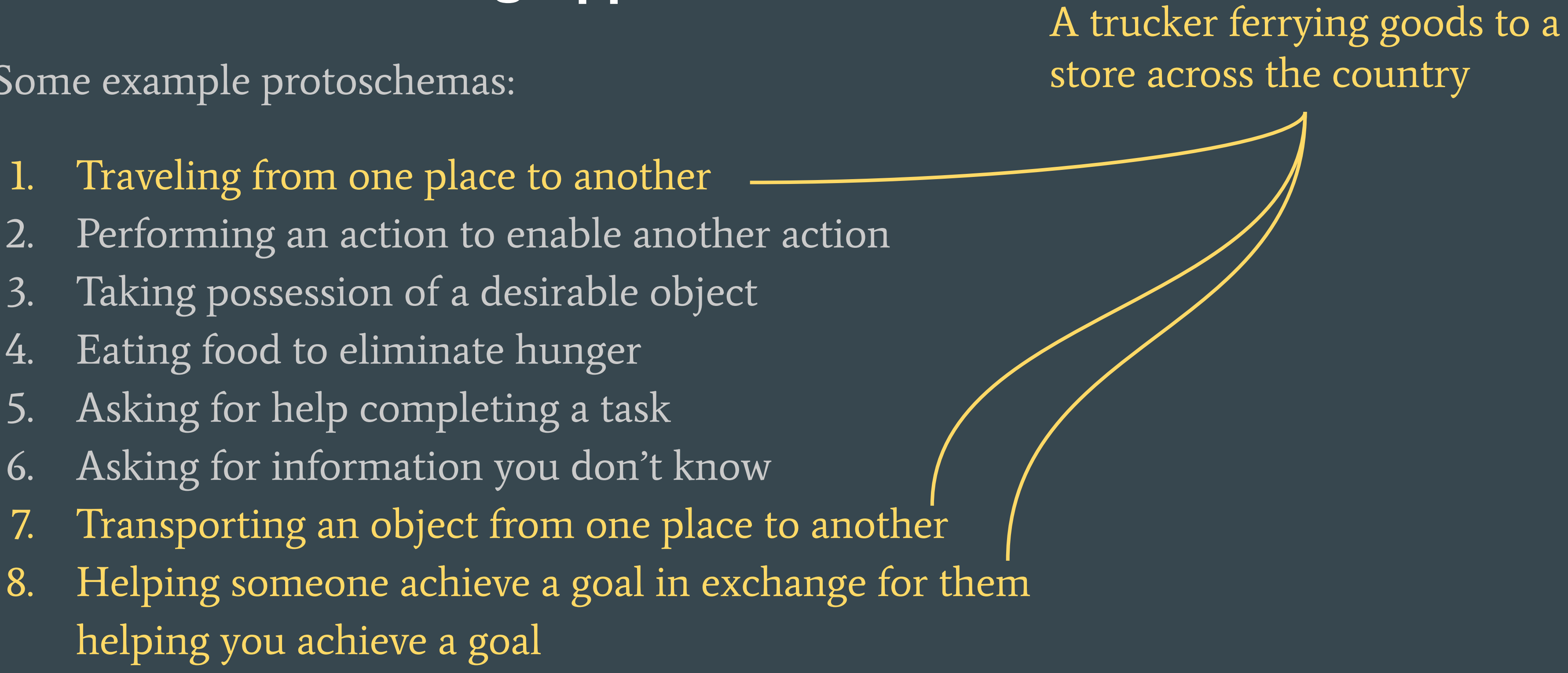
Overview of Learning Approach

Some example protoschemas:

1. Traveling from one place to another
2. Performing an action to enable another action
3. Taking possession of a desirable object
4. Eating food to eliminate hunger
5. Asking for help completing a task
6. Asking for information you don’t know
7. Transporting an object from one place to another
8. Helping someone achieve a goal in exchange for them helping you achieve a goal

Overview of Learning Approach

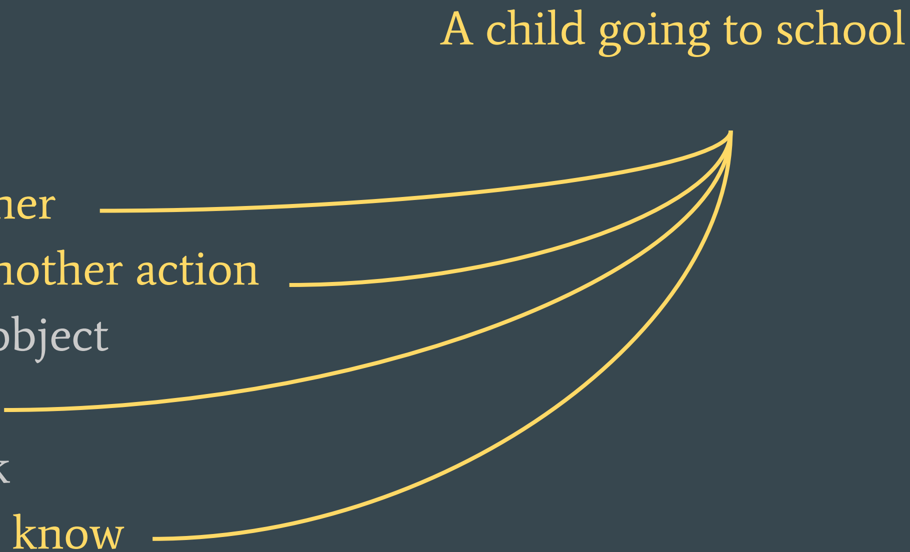
Some example protoschemas:



A trucker ferrying goods to a store across the country

1. Traveling from one place to another
2. Performing an action to enable another action
3. Taking possession of a desirable object
4. Eating food to eliminate hunger
5. Asking for help completing a task
6. Asking for information you don’t know
7. Transporting an object from one place to another
8. Helping someone achieve a goal in exchange for them helping you achieve a goal

Overview of Learning Approach


A child going to school

Some example protoschemas:

1. Traveling from one place to another
2. Performing an action to enable another action
3. Taking possession of a desirable object
4. Eating food to eliminate hunger
5. Asking for help completing a task
6. Asking for information you don’t know
7. Transporting an object from one place to another
8. Helping someone achieve a goal in exchange for them helping you achieve a goal

```
(epi-schema ((?x ((adv-a (from.p ?l1)) ((adv-a (to.p ?l2)) travel.v)) ?l2) ** ?e)
    :Roles
        !r1 (?x agent.n)
        !r2 (?l1 location.n)
        !r3 (?l2 location.n)
        !r4 (not (?l1 = ?l2))
    :Necessities
        !n1 (!r1 necessary-to-degree 1.0)
        !n2 (!r4 necessary-to-degree 1.0)
    :Goals
        ?g1 (?x (want.v (to ((adv-a (at.p ?l2)) be.v))))
    :Preconds
        ?i1 (?x (at.p ?l1))
        ?i2 (not (?x (at.p ?l2)))
    :Postconds
        ?p1 (not (?x (at.p ?l1)))
        ?p2 (?x (at.p ?l2))
)
```

Let’s look at our schema language, with this example.

```
(epi-schema ((?x ((adv-a (from.p ?l1)) ((adv-a (to.p ?l2)) travel.v)) ?l2) ** ?e)
	:Roles
		!r1 (?x agent.n)
		!r2 (?l1 location.n)
		!r3 (?l2 location.n)
		!r4 (not (?l1 = ?l2))
	:Necessities
		!n1 (!r1 necessary-to-degree 1.0)
		!n2 (!r4 necessary-to-degree 1.0)
	:Goals
		?g1 (?x (want.v (to ((adv-a (at.p ?l2)) be.v))))
	:Preconds
		?i1 (?x (at.p ?l1))
		?i2 (not (?x (at.p ?l2)))
	:Postconds
		?p1 (not (?x (at.p ?l1)))
		?p2 (?x (at.p ?l2))
)
```

“?e is an episode of ?x **traveling** from ?l1 to ?l2”

```
(epi-schema ((?x ((adv-a (from.p ?l1)) ((adv-a (to.p ?l2)) travel.v)) ?l2) ** ?e)
    :Roles
        !r1 (?x agent.n)
        !r2 (?l1 location.n)
        !r3 (?l2 location.n)
        !r4 (not (?l1 = ?l2))
    :Necessities
        !n1 (!r1 necessary-to-degree 1.0)
        !n2 (!r4 necessary-to-degree 1.0)
    :Goals
        ?g1 (?x (want.v (to ((adv-a (at.p ?l2)) be.v))))
    :Preconds
        ?i1 (?x (at.p ?l1))
        ?i2 (not (?x (at.p ?l2)))
    :Postconds
        ?p1 (not (?x (at.p ?l1)))
        ?p2 (?x (at.p ?l2))
)
```

“?e is an episode of ?x traveling from ?l1 to ?l2”

```
(epi-schema ((?x ((adv-a (from.p ?l1)) ((adv-a (to.p ?l2)) travel.v)) ?l2) ** ?e)
        :Roles
                !r1 (?x agent.n)
                !r2 (?l1 location.n)
                !r3 (?l2 location.n)
                !r4 (not (?l1 = ?l2))
        :Necessities
                !n1 (!r1 necessary-to-degree 1.0)
                !n2 (!r4 necessary-to-degree 1.0)
        :Goals
                ?g1 (?x (want.v (to ((adv-a (at.p ?l2)) be.v))))
        :Preconds
                ?i1 (?x (at.p ?l1))
                ?i2 (not (?x (at.p ?l2)))
        :Postconds
                ?p1 (not (?x (at.p ?l1)))
                ?p2 (?x (at.p ?l2))
)
```

“?x is an agent”
“?l1 and ?l2 are locations”
“?l1 and ?l2 are distinct”

```
(epi-schema ((?x ((adv-a (from.p ?l1)) ((adv-a (to.p ?l2)) travel.v)) ?l2) ** ?e)
	:Roles
		!r1 (?x agent.n)
		!r2 (?l1 location.n)
		!r3 (?l2 location.n)
		!r4 (not (?l1 = ?l2))
	:Necessities
		!n1 (!r1 necessary-to-degree 1.0)
		!n2 (!r4 necessary-to-degree 1.0)
	:Goals
		?g1 (?x (want.v (to ((adv-a (at.p ?l2)) be.v))))
	:Preconds
		?i1 (?x (at.p ?l1))
		?i2 (not (?x (at.p ?l2)))
	:Postconds
		?p1 (not (?x (at.p ?l1)))
		?p2 (?x (at.p ?l2))
)
```

“?x is definitely an agent”
“?l1 and ?l2 are definitely distinct”

```
(epi-schema ((?x ((adv-a (from.p ?l1)) ((adv-a (to.p ?l2)) travel.v)) ?l2) ** ?e)
        :Roles
                !r1 (?x agent.n)
                !r2 (?l1 location.n)
                !r3 (?l2 location.n)
                !r4 (not (?l1 = ?l2))
        :Necessities
                !n1 (!r1 necessary-to-degree 1.0)
                !n2 (!r4 necessary-to-degree 1.0)
        :Goals
                ?g1 (?x (want.v (to ((adv-a (at.p ?l2)) be.v))))
        :Preconds
                ?i1 (?x (at.p ?l1))
                ?i2 (not (?x (at.p ?l2)))
        :Postconds
                ?p1 (not (?x (at.p ?l1)))
                ?p2 (?x (at.p ?l2))
)
```

“?x wants to be at ?l2”

```
(epi-schema ((?x ((adv-a (from.p ?l1)) ((adv-a (to.p ?l2)) travel.v)) ?l2) ** ?e)
          :Roles
                    !r1 (?x agent.n)
                    !r2 (?l1 location.n)
                    !r3 (?l2 location.n)
                    !r4 (not (?l1 = ?l2))
          :Necessities
                    !n1 (!r1 necessary-to-degree 1.0)
                    !n2 (!r4 necessary-to-degree 1.0)
          :Goals
                    ?g1 (?x (want.v (to ((adv-a (at.p ?l2)) be.v))))
          :Preconds
                    ?i1 (?x (at.p ?l1))
                    ?i2 (not (?x (at.p ?l2)))
          :Postconds
                    ?p1 (not (?x (at.p ?l1)))
                    ?p2 (?x (at.p ?l2))
)
```

“?x is at ?l1”
“?x is not at ?l2”

```
(epi-schema ((?x ((adv-a (from.p ?l1)) ((adv-a (to.p ?l2)) travel.v)) ?l2) ** ?e)
        :Roles
                !r1 (?x agent.n)
                !r2 (?l1 location.n)
                !r3 (?l2 location.n)
                !r4 (not (?l1 = ?l2))
        :Necessities
                !n1 (!r1 necessary-to-degree 1.0)
                !n2 (!r4 necessary-to-degree 1.0)
        :Goals
                ?g1 (?x (want.v (to ((adv-a (at.p ?l2)) be.v))))
        :Preconds
                ?i1 (?x (at.p ?l1))
                ?i2 (not (?x (at.p ?l2)))
        :Postconds
                ?p1 (not (?x (at.p ?l1)))
                ?p2 (?x (at.p ?l2))
)
```

"?x is not at ?l1"
"?x is at ?l2"

```
(epi-schema ((?x ((adv-a (from.p ?l1)) ((adv-a (to.p ?l2)) travel.v)) ?l2) ** ?e)
        :Roles
                !r1 (?x agent.n)
                !r2 (?l1 location.n)
                !r3 (?l2 location.n)
                !r4 (not (?l1 = ?l2))
        :Necessities
                !n1 (!r1 necessary-to-degree 1.0)
                !n2 (!r4 necessary-to-degree 1.0)
        :Goals
                ?g1 (?x (want.v (to ((adv-a (at.p ?l2)) be.v))))
        :Preconds
                ?i1 (?x (at.p ?l1))
                ?i2 (not (?x (at.p ?l2)))
        :Postconds
                ?p1 (not (?x (at.p ?l1)))
                ?p2 (?x (at.p ?l2))
)
```

```
(epi-schema ((?x ((adv-a (from.p ?l1)) ((adv-a (to.p ?l2)) travel.v)) ?l2) ** ?e)
        :Roles
                !r1 (?x agent.n)
                !r2 (?l1 location.n)
                !r3 (?l2 location.n)
                !r4 (not (?l1 = ?l2))
        :Necessities
                !n1 (!r1 necessary-to-degree 1.0)
                !n2 (!r4 necessary-to-degree 1.0)
        :Goals
                ?g1 (?x (want.v (to ((adv-a (at.p ?l2)) be.v))))
        :Preconds
                ?i1 (?x (at.p ?l1))
                ?i2 (not (?x (at.p ?l2)))
        :Postconds
                ?p1 (not (?x (at.p ?l1)))
                ?p2 (?x (at.p ?l2))
)
```

Let's match our schema to this story.

The monkey can climb a tree.

He climbs the tree and gets a cocoanut.

He drops the cocoanut to the ground.

He comes down and eats it.

```
(epi-schema ((?x ((adv-a (from.p ?l1)) ((adv-a (to.p ?l2)) travel.v)) ?l2) ** ?e)
        :Roles
                !r1 (?x agent.n)
                !r2 (?l1 location.n)
                !r3 (?l2 location.n)
                !r4 (not (?l1 = ?l2))
        :Necessities
                !n1 (!r1 necessary-to-degree 1.0)
                !n2 (!r4 necessary-to-degree 1.0)
        :Goals
                ?g1 (?x (want.v (to ((adv-a (at.p ?l2)) be.v))))
        :Preconds
                ?i1 (?x (at.p ?l1))
                ?i2 (not (?x (at.p ?l2)))
        :Postconds
                ?p1 (not (?x (at.p ?l1)))
                ?p2 (?x (at.p ?l2))
)
```

The English is parsed to Episodic Logic.

```
(TREE28.SK TREE.N)

(MONKEY27.SK MONKEY.N)

(COCOANUT32.SK COCOANUT.N)

((MONKEY27.SK ((CAN.MD CLIMB.V)
TREE28.SK)) ** E26.SK)

((MONKEY27.SK (CLIMB.V
TREE28.SK)) ** E34.SK)

((MONKEY27.SK (GET.V
COCOANUT32.SK)) ** E33.SK)

((MONKEY27.SK (EAT.V
COCOANUT32.SK)) ** E35.SK)
```

```
(epi-schema ((?x ((adv-a (from.p ?l1)) ((adv-a (to.p ?l2)) travel.v)) ?l2) ** ?e)
        :Roles
                !r1 (?x agent.n)
                !r2 (?l1 location.n)
                !r3 (?l2 location.n)
                !r4 (not (?l1 = ?l2))
        :Necessities
                !n1 (!r1 necessary-to-degree 1.0)
                !n2 (!r4 necessary-to-degree 1.0)
        :Goals
                ?g1 (?x (want.v (to ((adv-a (at.p ?l2)) be.v))))
        :Preconds
                ?i1 (?x (at.p ?l1))
                ?i2 (not (?x (at.p ?l2)))
        :Postconds
                ?p1 (not (?x (at.p ?l1)))
                ?p2 (?x (at.p ?l2))
)
```

The English is parsed to Episodic Logic.

```
(TREE28.SK TREE.N)

(MONKEY27.SK MONKEY.N)

(COCOANUT32.SK COCOANUT.N)

((MONKEY27.SK ((CAN.MD CLIMB.V)
TREE28.SK)) ** E26.SK)

((MONKEY27.SK (CLIMB.V
TREE28.SK)) ** E34.SK)

((MONKEY27.SK (GET.V
COCOANUT32.SK)) ** E33.SK)

((MONKEY27.SK (EAT.V
COCOANUT32.SK)) ** E35.SK)
```

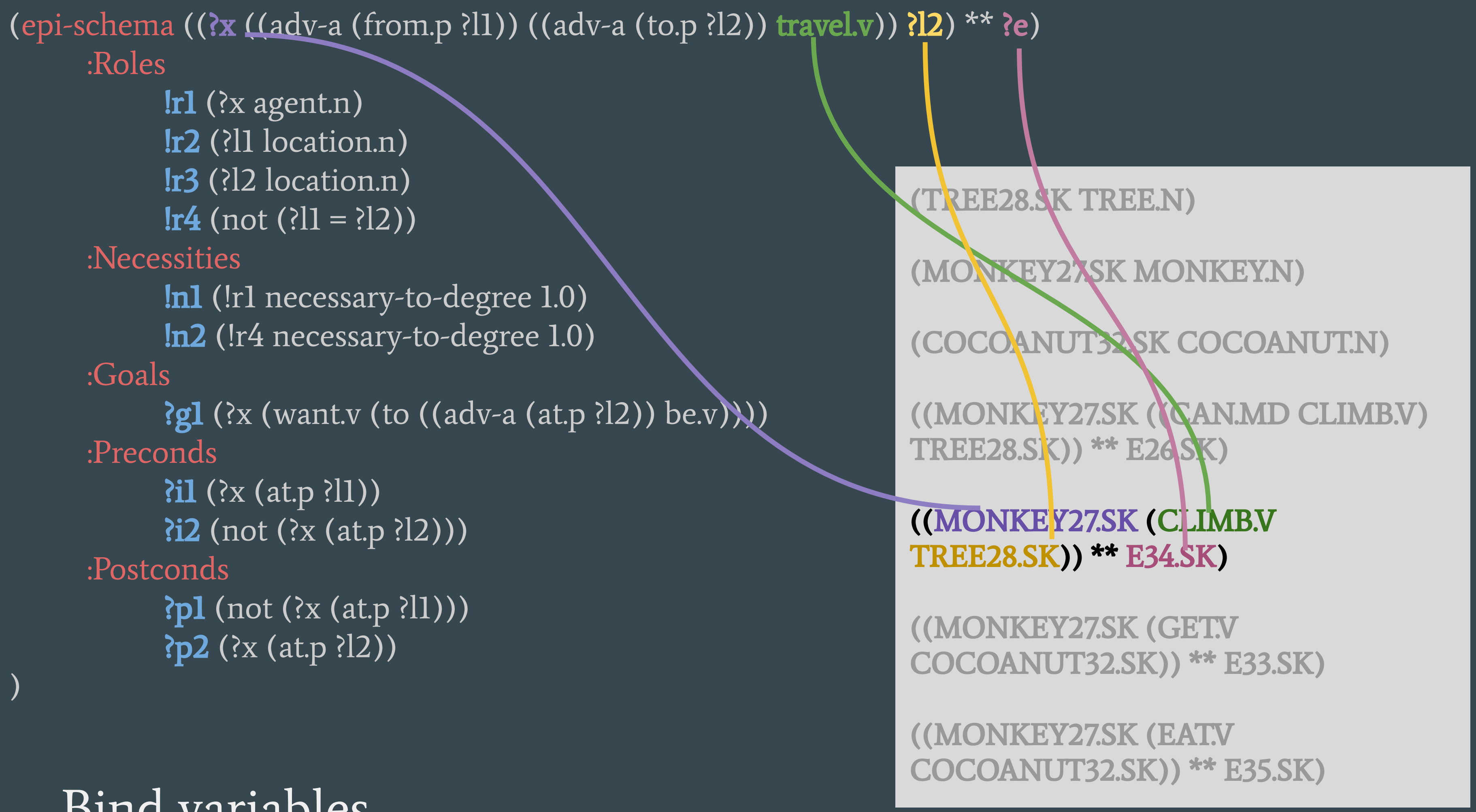
```
(epi-schema ((?x ((adv-a (from.p ?l1)) ((adv-a (to.p ?l2)) travel.v)) ?l2) ** ?e)
        :Roles
                !r1 (?x agent.n)
                !r2 (?l1 location.n)
                !r3 (?l2 location.n)
                !r4 (not (?l1 = ?l2))
        :Necessities
                !n1 (!r1 necessary-to-degree 1.0)
                !n2 (!r4 necessary-to-degree 1.0)
        :Goals
                ?g1 (?x (want.v (to ((adv-a (at.p ?l2)) be.v))))
        :Preconds
                ?i1 (?x (at.p ?l1))
                ?i2 (not (?x (at.p ?l2)))
        :Postconds
                ?p1 (not (?x (at.p ?l1)))
                ?p2 (?x (at.p ?l2))
)
```

```
(TREE28.SK TREE.N)

(MONKEY27.SK MONKEY.N)

(COCOANUT32.SK COCOANUT.N)

((MONKEY27.SK ((CAN.MD CLIMB.V)
TREE28.SK)) ** E26.SK)

((MONKEY27.SK (CLIMB.V
TREE28.SK)) ** E34.SK)

((MONKEY27.SK (GET.V
COCOANUT32.SK)) ** E33.SK)

((MONKEY27.SK (EAT.V
COCOANUT32.SK)) ** E35.SK)
```

```
(epi-schema ((?x ((adv-a (from.p ?l1)) ((adv-a (to.p ?l2)) travel.v)) ?l2) ** ?e)
        :Roles
                !r1 (?x agent.n)
                !r2 (?l1 location.n)
                !r3 (?l2 location.n)
                !r4 (not (?l1 = ?l2))
        :Necessities
                !n1 (!r1 necessary-to-degree 1.0)
                !n2 (!r4 necessary-to-degree 1.0)
        :Goals
                ?g1 (?x (want.v (to ((adv-a (at.p ?l2)) be.v))))
        :Preconds
                ?i1 (?x (at.p ?l1))
                ?i2 (not (?x (at.p ?l2)))
        :Postconds
                ?p1 (not (?x (at.p ?l1)))
                ?p2 (?x (at.p ?l2))
)
```

(WordNet hypernym; a schema match is triggered)

```
(TREE28.SK TREE.N)

(MONKEY27.SK MONKEY.N)

(COCOANUT32.SK COCOANUT.N)

((MONKEY27.SK ((CAN.MD CLIMB.V)
TREE28.SK)) ** E26.SK)

((MONKEY27.SK (CLIMB.V
TREE28.SK)) ** E34.SK)

((MONKEY27.SK (GET.V
COCOANUT32.SK)) ** E33.SK)

((MONKEY27.SK (EAT.V
COCOANUT32.SK)) ** E35.SK)
```



```
(epi-schema ((?x ((adv-a (from.p ?l1)) ((adv-a (to.p ?l2)) travel.v)) ?l2) ** ?e)
        :Roles
                !r1 (?x agent.n)
                !r2 (?l1 location.n)
                !r3 (?l2 location.n)
                !r4 (not (?l1 = ?l2))
        :Necessities
                !n1 (!r1 necessary-to-degree 1.0)
                !n2 (!r4 necessary-to-degree 1.0)
        :Goals
                ?g1 (?x (want.v (to ((adv-a (at.p ?l2)) be.v))))
        :Preconds
                ?i1 (?x (at.p ?l1))
                ?i2 (not (?x (at.p ?l2)))
        :Postconds
                ?p1 (not (?x (at.p ?l1)))
                ?p2 (?x (at.p ?l2))
)
```

(TREE28.SK TREE.N)

(MONKEY27.SK MONKEY.N)

(COCOANUT32.SK COCOANUT.N)

((MONKEY27.SK ((CAN.MD CLIMB.V) TREE28.SK)) ** E26.SK)

((MONKEY27.SK (CLIMB.V TREE28.SK)) ** E34.SK)

((MONKEY27.SK (GET.V COCOANUT32.SK)) ** E33.SK)

((MONKEY27.SK (EAT.V COCOANUT32.SK)) ** E35.SK)

Bind variables

```
(epi-schema ((MONKEY27.SK ((adv-a (from.p ?l1)) ((adv-a (to.p TREE28.SK))
CLIMB.V)) TREE28.SK) ** E34.SK)
    :Roles
        !r1 (MONKEY27.SK agent.n)
        !r2 (?l1 location.n)
        !r3 (TREE28.SK location.n)
        !r4 (not (?l1 = TREE28.SK))
    :Necessities
        !n1 (!r1 necessary-to-degree 1.0)
        !n2 (!r4 necessary-to-degree 1.0)
    :Goals
        ?g1 (MONKEY27.SK (want.v (to ((adv-a (at.p TREE28.SK)) be.v))))
    :Preconds
        ?i1 (MONKEY27.SK (at.p ?l1))
        ?i2 (not (MONKEY27.SK (at.p TREE28.SK)))
    :Postconds
        ?p1 (not (MONKEY27.SK (at.p ?l1)))
        ?p2 (MONKEY27.SK (at.p TREE28.SK))
)
```

```
(TREE28.SK TREE.N)

(MONKEY27.SK MONKEY.N)

(COCOANUT32.SK COCOANUT.N)

((MONKEY27.SK ((CAN.MD CLIMB.V)
TREE28.SK)) ** E26.SK)

((MONKEY27.SK (CLIMB.V TREE28.SK)) ** E34.SK)

((MONKEY27.SK (GET.V COCOANUT32.SK)) **
E33.SK)
```

Replace bound variables in rest of schema

```
(epi-schema ((MONKEY27.SK ((adv-a (from.p ?l1)) ((adv-a (to.p TREE28.SK))
CLIMB.V)) TREE28.SK) ** E34.SK)
    :Roles
            !r1 (MONKEY27.SK agent.n)
            !r2 (?l1 location.n)
            !r3 (TREE28.SK location.n)
            !r4 (not (?l1 = TREE28.SK))
    :Necessities
            !n1 (!r1 necessary-to-degree 1.0)
            !n2 (!r4 necessary-to-degree 1.0)
    :Goals
            ?g1 (MONKEY27.SK (want.v (to ((adv-a (at.p TREE28.SK)) be.v))))
    :Preconds
            ?i1 (MONKEY27.SK (at.p ?l1))
            ?i2 (not (MONKEY27.SK (at.p TREE28.SK)))
    :Postconds
            ?p1 (not (MONKEY27.SK (at.p ?l1)))
            ?p2 (MONKEY27.SK (at.p TREE28.SK))
)
```

```
(TREE28.SK TREE.N)

(MONKEY27.SK MONKEY.N)

(COCOANUT32.SK COCOANUT.N)

((MONKEY27.SK ((CAN.MD CLIMB.V)
TREE28.SK)) ** E26.SK)

((MONKEY27.SK (CLIMB.V TREE28.SK)) ** E34.SK)

((MONKEY27.SK (GET.V COCOANUT32.SK)) **
E33.SK)
```

Match score: check schema’s constraints on the matched values against the “knowledge base” of the story

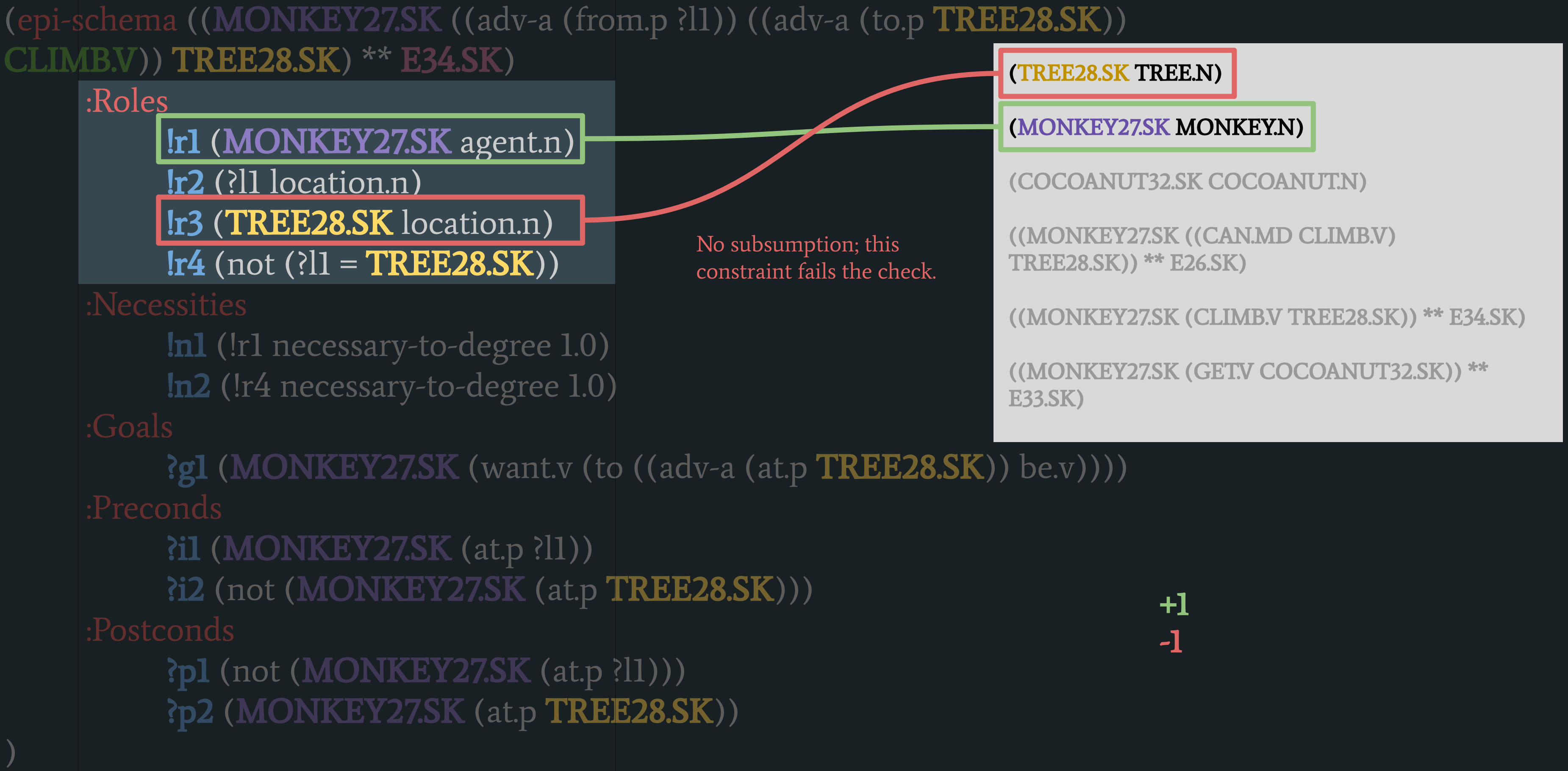
```
(epi-schema ((MONKEY27.SK ((adv-a (from.p ?l1)) ((adv-a (to.p TREE28.SK))
CLIMB.V)) TREE28.SK) ** E34.SK)
    :Roles
        !r1 (MONKEY27.SK agent.n)
        !r2 (?l1 location.n)
        !r3 (TREE28.SK location.n)
        !r4 (not (?l1 = TREE28.SK))
    :Necessities
        !n1 (!r1 necessary-to-degree 1.0)
        !n2 (!r4 necessary-to-degree 1.0)
    :Goals
        ?g1 (MONKEY27.SK (want.v (to ((adv-a (at.p TREE28.SK)) be.v))))
    :Preconds
        ?i1 (MONKEY27.SK (at.p ?l1))
        ?i2 (not (MONKEY27.SK (at.p TREE28.SK)))
    :Postconds
        ?p1 (not (MONKEY27.SK (at.p ?l1)))
        ?p2 (MONKEY27.SK (at.p TREE28.SK))
)
```

“AGENT.N” subsumes “ANIMAL.N”

“ANIMAL.N” subsumes “MONKEY.N” in wordnet

```
(TREE28.SK TREE.N)

(MONKEY27.SK MONKEY.N)

(COCOANUT32.SK COCOANUT.N)

((MONKEY27.SK ((CAN.MD CLIMB.V)
TREE28.SK)) ** E26.SK)

((MONKEY27.SK (CLIMB.V TREE28.SK)) ** E34.SK)

((MONKEY27.SK (GET.V COCOANUT32.SK)) **
E33.SK)
```

+1



```
(epi-schema ((MONKEY27.SK ((adv-a (from.p ?l1)) ((adv-a (to.p TREE28.SK))
CLIMB.V)) TREE28.SK) ** E34.SK)
    :Roles
        !r1 (MONKEY27.SK agent.n)
        !r2 (?l1 location.n)
        !r3 (TREE28.SK location.n)
        !r4 (not (?l1 = TREE28.SK))
    :Necessities
        !n1 (!r1 necessary-to-degree 1.0)
        !n2 (!r4 necessary-to-degree 1.0)
    :Goals
        ?g1 (MONKEY27.SK (want.v (to ((adv-a (at.p TREE28.SK)) be.v))))
    :Preconds
        ?i1 (MONKEY27.SK (at.p ?l1))
        ?i2 (not (MONKEY27.SK (at.p TREE28.SK)))
    :Postconds
        ?p1 (not (MONKEY27.SK (at.p ?l1)))
        ?p2 (MONKEY27.SK (at.p TREE28.SK))
)
```

(TREE28.SK TREE.N)

(MONKEY27.SK MONKEY.N)

(COCOANUT32.SK COCOANUT.N)

((MONKEY27.SK ((CAN.MD CLIMB.V) TREE28.SK)) ** E26.SK)

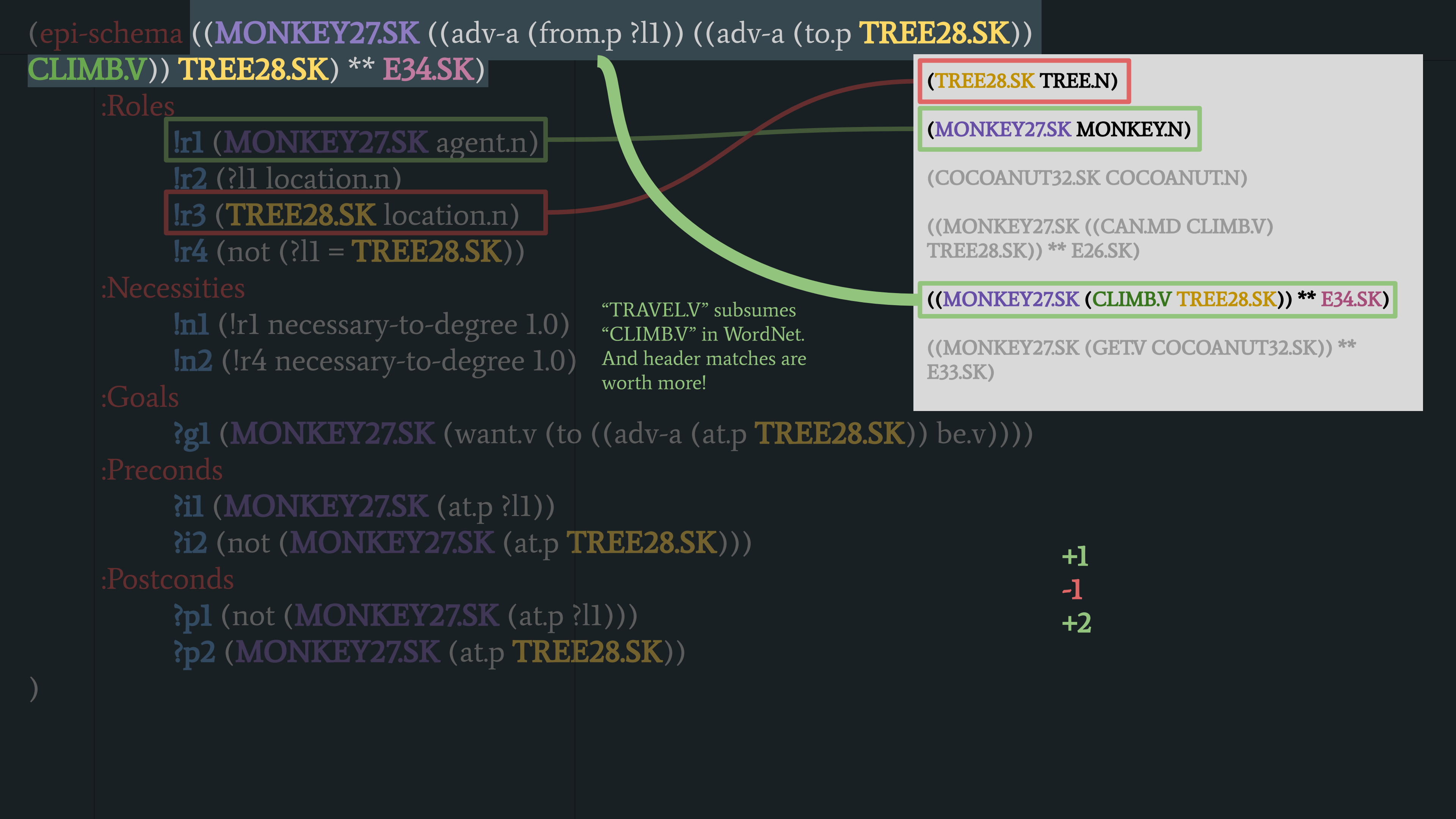
((MONKEY27.SK (CLIMB.V TREE28.SK)) ** E34.SK)

((MONKEY27.SK (GET.V COCOANUT32.SK)) ** E33.SK)

No subsumption; this constraint fails the check.

+1

-1



```
(epi-schema ((MONKEY27.SK ((adv-a (from.p ?l1)) ((adv-a (to.p TREE28.SK))
CLIMB.V)) TREE28.SK) ** E34.SK)
    :Roles
        !r1 (MONKEY27.SK agent.n)
        !r2 (?l1 location.n)
        !r3 (TREE28.SK location.n)
        !r4 (not (?l1 = TREE28.SK))
    :Necessities
        !n1 (!r1 necessary-to-degree 1.0)
        !n2 (!r4 necessary-to-degree 1.0)
    :Goals
        ?g1 (MONKEY27.SK (want.v (to ((adv-a (at.p TREE28.SK)) be.v))))
    :Preconds
        ?i1 (MONKEY27.SK (at.p ?l1))
        ?i2 (not (MONKEY27.SK (at.p TREE28.SK)))
    :Postconds
        ?p1 (not (MONKEY27.SK (at.p ?l1)))
        ?p2 (MONKEY27.SK (at.p TREE28.SK))
)
```

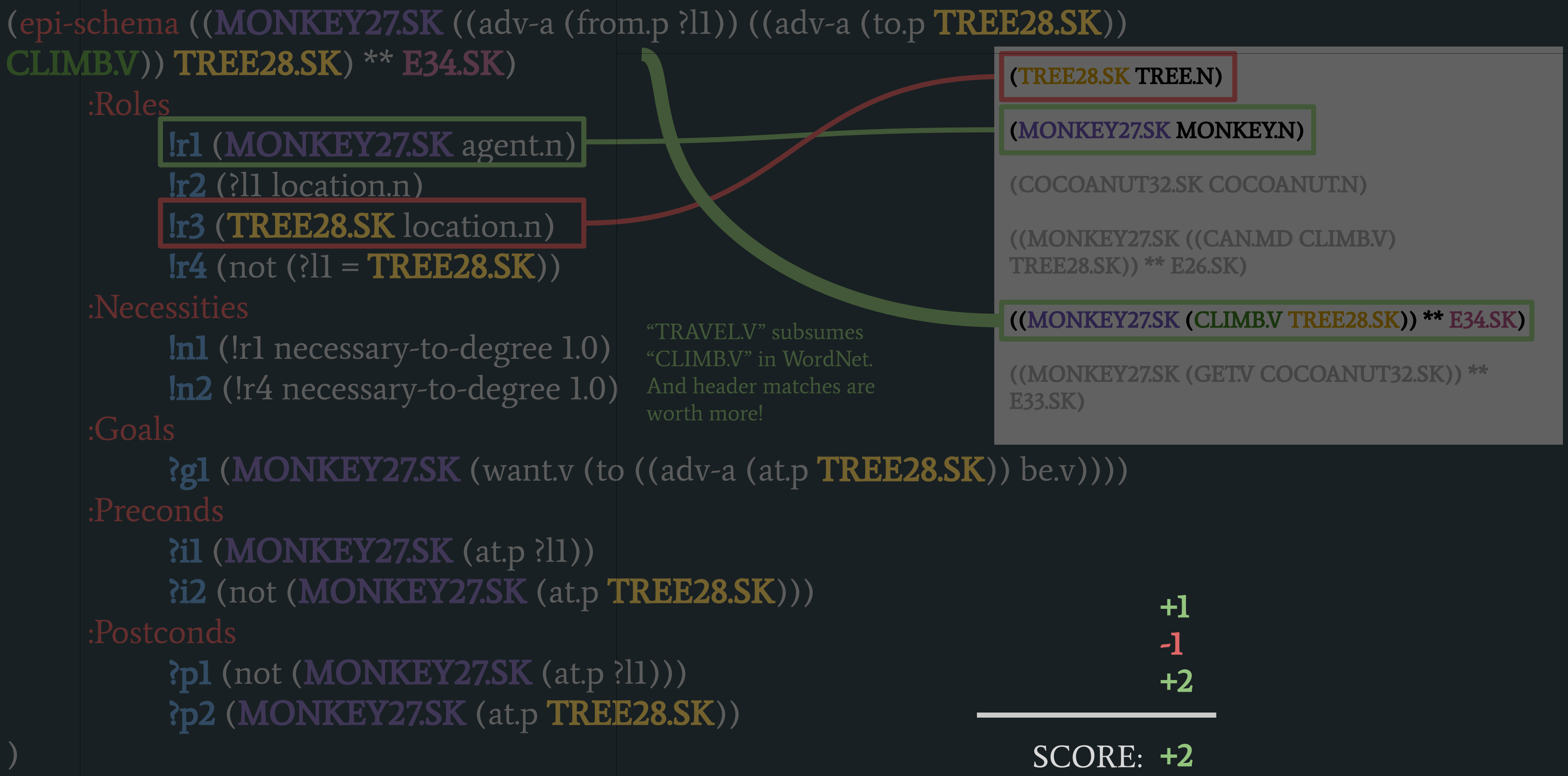
```
(TREE28.SK TREE.N)

(MONKEY27.SK MONKEY.N)

(COCOANUT32.SK COCOANUT.N)

((MONKEY27.SK ((CAN.MD CLIMB.V)
TREE28.SK)) ** E26.SK)

((MONKEY27.SK (CLIMB.V TREE28.SK)) ** E34.SK)

((MONKEY27.SK (GET.V COCOANUT32.SK)) **
E33.SK)
```

"TRAVEL.V" subsumes "CLIMB.V" in WordNet. And header matches are worth more!

+1

-1

+2



```
(epi-schema ((MONKEY27.SK ((adv-a (from.p ?l1)) ((adv-a (to.p TREE28.SK))
CLIMB.V)) TREE28.SK) ** E34.SK)
        :Roles
                !r1 (MONKEY27.SK agent.n)
                !r2 (?l1 location.n)
                !r3 (TREE28.SK location.n)
                !r4 (not (?l1 = TREE28.SK))
        :Necessities
                !n1 (!r1 necessary-to-degree 1.0)
                !n2 (!r4 necessary-to-degree 1.0)
        :Goals
                ?g1 (MONKEY27.SK (want.v (to ((adv-a (at.p TREE28.SK)) be.v))))
        :Preconds
                ?i1 (MONKEY27.SK (at.p ?l1))
                ?i2 (not (MONKEY27.SK (at.p TREE28.SK)))
        :Postconds
                ?p1 (not (MONKEY27.SK (at.p ?l1)))
                ?p2 (MONKEY27.SK (at.p TREE28.SK))
)
```

(TREE28.SK TREE.N)

(MONKEY27.SK MONKEY.N)

(COCOANUT32.SK COCOANUT.N)

((MONKEY27.SK ((CAN.MD CLIMB.V)
TREE28.SK)) ** E26.SK)

((MONKEY27.SK (CLIMB.V TREE28.SK)) ** E34.SK)

((MONKEY27.SK (GET.V COCOANUT32.SK)) **
E33.SK)

"TRAVEL.V" subsumes "CLIMB.V" in WordNet. And header matches are worth more!

+1
-1
+2

SCORE: +2

```
(epi-schema ((MONKEY27.SK ((adv-a (from.p ?l1)) ((adv-a (to.p TREE28.SK))
CLIMB.V)) TREE28.SK) ** E34.SK)
    :Roles
        !r1 (MONKEY27.SK agent.n)
        !r2 (?l1 location.n)
        !r3 (TREE28.SK location.n)
        !r4 (not (?l1 = TREE28.SK))
    :Necessities
        !n1 (!r1 necessary-to-degree 1.0)
        !n2 (!r4 necessary-to-degree 1.0)
    :Goals
        ?g1 (MONKEY27.SK (want.v (to ((adv-a (at.p TREE28.SK)) be.v))))
    :Preconds
        ?i1 (MONKEY27.SK (at.p ?l1))
        ?i2 (not (MONKEY27.SK (at.p TREE28.SK)))
    :Postconds
        ?p1 (not (MONKEY27.SK (at.p ?l1)))
        ?p2 (MONKEY27.SK (at.p TREE28.SK))
)
```

(TREE28.SK TREE.N)

(MONKEY27.SK MONKEY.N)

(COCOANUT32.SK COCOANUT.N)

((MONKEY27.SK ((CAN.MD CLIMB.V) TREE28.SK)) ** E26.SK)

((MONKEY27.SK (CLIMB.V TREE28.SK)) ** E34.SK)

((MONKEY27.SK (GET.V COCOANUT32.SK)) ** E33.SK)

Incorporate story’s constraints on bound values

```
(epi-schema ((MONKEY27.SK ((adv-a (from.p ?l1)) ((adv-a (to.p TREE28.SK))
CLIMB.V)) TREE28.SK) ** E34.SK)
        :Roles
                !r1 (MONKEY27.SK agent.n)
                !r2 (?l1 location.n)
                !r3 (TREE28.SK location.n)
                !r4 (not (?l1 = TREE28.SK))
                !r5 (TREE28.SK TREE.N)
                !r6 (MONKEY27.SK MONKEY.N)
        :Necessities
                !n1 (!r1 necessary-to-degree 1.0)
                !n2 (!r4 necessary-to-degree 1.0)
        :Goals
                ?g1 (MONKEY27.SK (want.v (to ((adv-a (at.p TREE28.SK)) be.v))))
        :Preconds
                ?i1 (MONKEY27.SK (at.p ?l1))
                ?i2 (not (MONKEY27.SK (at.p TREE28.SK)))
        :Postconds
                ?p1 (not (MONKEY27.SK (at.p ?l1)))
                ?p2 (MONKEY27.SK (at.p TREE28.SK))
)
```

```
(TREE28.SK TREE.N)

(MONKEY27.SK MONKEY.N)

(COCOANUT32.SK COCOANUT.N)

((MONKEY27.SK ((CAN.MD CLIMB.V)
TREE28.SK)) ** E26.SK)

((MONKEY27.SK (CLIMB.V TREE28.SK)) ** E34.SK)

((MONKEY27.SK (GET.V COCOANUT32.SK)) **
E33.SK)
```

Incorporate story’s constraints on bound values

```
(epi-schema ((MONKEY27.SK ((adv-a (from.p ?l1)) ((adv-a (to.p TREE28.SK))
CLIMB.V)) TREE28.SK) ** E34.SK)
        :Roles
                !r1 (MONKEY27.SK agent.n)
                !r2 (?l1 location.n)
                !r3 (TREE28.SK location.n)
                !r4 (not (?l1 = TREE28.SK))
                !r5 (TREE28.SK TREE.N)
                !r6 (MONKEY27.SK MONKEY.N)
        :Necessities
                !n1 (!r1 necessary-to-degree 1.0)
                !n2 (!r4 necessary-to-degree 1.0)
        :Goals
                ?g1 (MONKEY27.SK (want.v (to ((adv-a (at.p TREE28.SK)) be.v))))
        :Preconds
                ?i1 (MONKEY27.SK (at.p ?l1))
                ?i2 (not (MONKEY27.SK (at.p TREE28.SK)))
        :Postconds
                ?p1 (not (MONKEY27.SK (at.p ?l1)))
                ?p2 (MONKEY27.SK (at.p TREE28.SK))
)
```

```
(TREE28.SK TREE.N)

(MONKEY27.SK MONKEY.N)

(COCOANUT32.SK COCOANUT.N)

((MONKEY27.SK ((CAN.MD CLIMB.V)
TREE28.SK)) ** E26.SK)

((MONKEY27.SK (CLIMB.V TREE28.SK)) ** E34.SK)

((MONKEY27.SK (GET.V COCOANUT32.SK)) **
E33.SK)
```

```
(epi-schema ((MONKEY27.SK ((adv-a (from.p ?l1)) ((adv-a (to.p TREE28.SK))
CLIMB.V)) TREE28.SK) ** E34.SK)
        :Roles
                !r1 (MONKEY27.SK agent.n)
                !r2 (?l1 location.n)
                !r3 (TREE28.SK location.n)
                !r4 (not (?l1 = TREE28.SK))
                !r5 (TREE28.SK TREE.N)
                !r6 (MONKEY27.SK MONKEY.N)
        :Necessities
                !n1 (!r1 necessary-to-degree 1.0)
                !n2 (!r4 necessary-to-degree 1.0)
        :Goals
                ?g1 (MONKEY27.SK (want.v (to ((adv-a (at.p TREE28.SK)) be.v))))
        :Preconds
                ?i1 (MONKEY27.SK (at.p ?l1))
                ?i2 (not (MONKEY27.SK (at.p TREE28.SK)))
        :Postconds
                ?p1 (not (MONKEY27.SK (at.p ?l1)))
                ?p2 (MONKEY27.SK (at.p TREE28.SK))
)
```

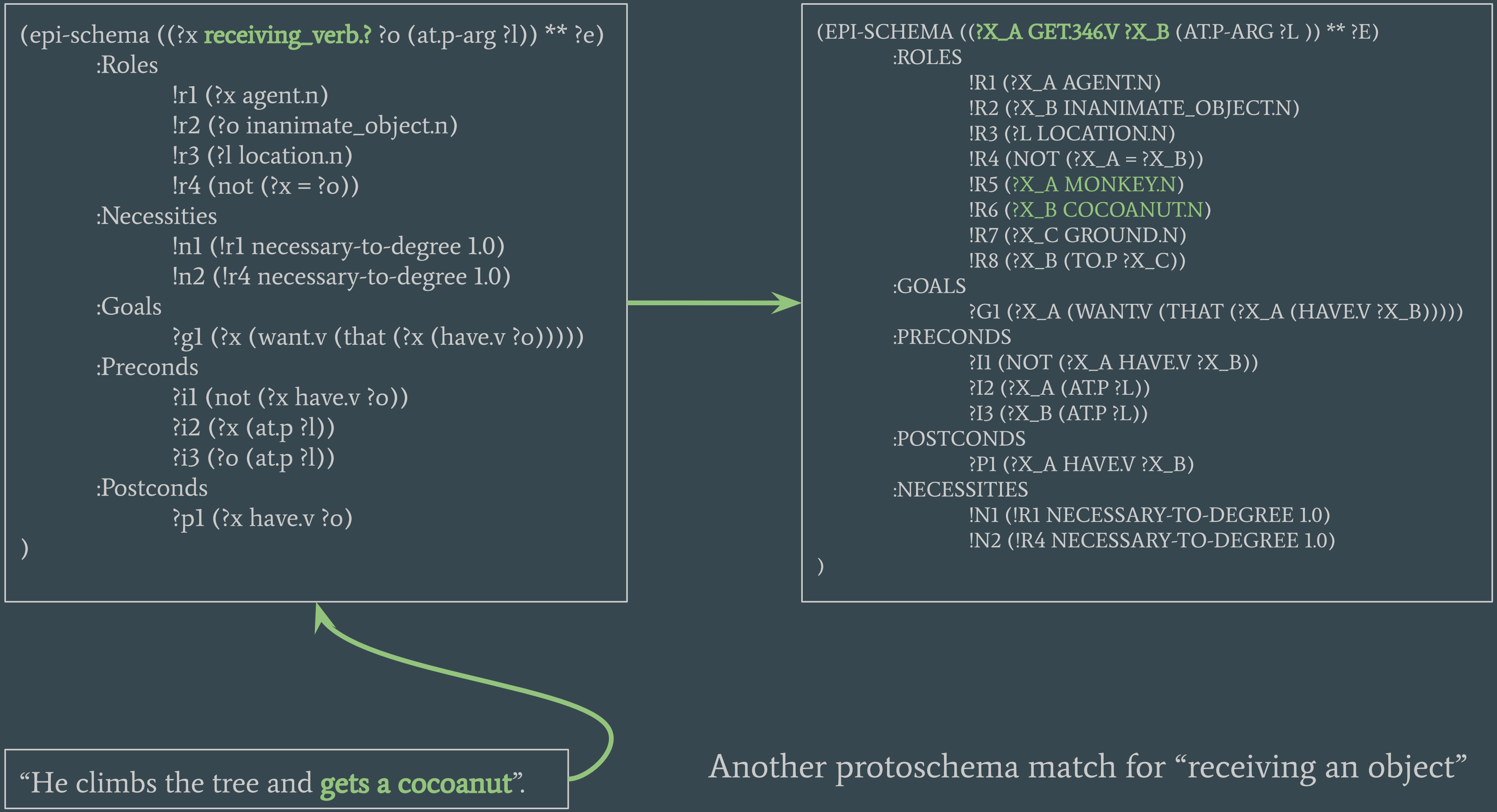
After incorporating the extra constraints, we generalize constants back to variables...

```
(epi-schema ((?x ((adv-a (from.p ?l1)) ((adv-a (to.p ?l2)) CLIMB.V)) ?l2) ** ?e)
        :Roles
                !r1 (?x agent.n)
                !r2 (?l1 location.n)
                !r3 (?l2 location.n)
                !r4 (not (?l1 = ?l2))
                !r5 (?l2 TREE.N)
                !r6 (?x MONKEY.N)
        :Necessities
                !n1 (!r1 necessary-to-degree 1.0)
                !n2 (!r4 necessary-to-degree 1.0)
        :Goals
                ?g1 (?x (want.v (to ((adv-a (at.p ?l2)) be.v))))
        :Preconds
                ?i1 (?x (at.p ?l1))
                ?i2 (not (?x (at.p ?l2)))
        :Postconds
                ?p1 (not (?x (at.p ?l1)))
                ?p2 (?x (at.p ?l2))
)
```

After incorporating the extra constraints, we generalize constants back to variables...

```
(epi-schema ((?x ((adv-a (from.p ?l1)) ((adv-a (to.p ?l2)) CLIMB.V)) ?l2) ** ?e)
        :Roles
                !r1 (?x agent.n)
                !r2 (?l1 location.n)
                !r3 (?l2 location.n)
                !r4 (not (?l1 = ?l2))
                !r5 (?l2 TREE.N)
                !r6 (?x MONKEY.N)
        :Necessities
                !n1 (!r1 necessary-to-degree 1.0)
                !n2 (!r4 necessary-to-degree 1.0)
        :Goals
                ?g1 (?x (want.v (to ((adv-a (at.p ?l2)) be.v))))
        :Preconds
                ?i1 (?x (at.p ?l1))
                ?i2 (not (?x (at.p ?l2)))
        :Postconds
                ?p1 (not (?x (at.p ?l1)))
                ?p2 (?x (at.p ?l2))
)
```

Our “an agent travels” protoschema has now generated an “a monkey climbs a tree” schema!



```
(epi-schema ((?x receiving_verb.? ?o (at.p-arg ?l)) ** ?e)
        :Roles
                !r1 (?x agent.n)
                !r2 (?o inanimate_object.n)
                !r3 (?l location.n)
                !r4 (not (?x = ?o))
        :Necessities
                !n1 (!r1 necessary-to-degree 1.0)
                !n2 (!r4 necessary-to-degree 1.0)
        :Goals
                ?g1 (?x (want.v (that (?x (have.v ?o)))))
        :Preconds
                ?i1 (not (?x have.v ?o))
                ?i2 (?x (at.p ?l))
                ?i3 (?o (at.p ?l))
        :Postconds
                ?p1 (?x have.v ?o)
)
```

```
(EPI-SCHEMA ((?X_A GET.346.V ?X_B (AT.P-ARG ?L )) ** ?E)
        :ROLES
                !R1 (?X_A AGENT.N)
                !R2 (?X_B INANIMATE_OBJECT.N)
                !R3 (?L LOCATION.N)
                !R4 (NOT (?X_A = ?X_B))
                !R5 (?X_A MONKEY.N)
                !R6 (?X_B COCOANUT.N)
                !R7 (?X_C GROUND.N)
                !R8 (?X_B (TO.P ?X_C))
        :GOALS
                ?G1 (?X_A (WANT.V (THAT (?X_A (HAVE.V ?X_B)))))
        :PRECONDS
                ?I1 (NOT (?X_A HAVE.V ?X_B))
                ?I2 (?X_A (AT.P ?L))
                ?I3 (?X_B (AT.P ?L))
        :POSTCONDS
                ?P1 (?X_A HAVE.V ?X_B)
        :NECESSITIES
                !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
                !N2 (!R4 NECESSARY-TO-DEGREE 1.0)
)
```

“He climbs the tree and **gets a cocoanut**”.

Another protoschema match for “receiving an object”

```
(EPI-SCHEMA ((?X_A GET.346.V ?X_B (AT.P-ARG ?L )) ** ?E)
    :ROLES
        !R1 (?X_A AGENT.N)
        !R2 (?X_B INANIMATE_OBJECT.N)
        !R3 (?L LOCATION.N)
        !R4 (NOT (?X_A = ?X_B))
        !R5 (?X_A MONKEY.N)
        !R6 (?X_B COCOANUT.N)
        !R7 (?X_C GROUND.N)
        !R8 (?X_B (TO.P ?X_C))
    :GOALS
        ?G1 (?X_A (WANT.V (THAT (?X_A (HAVE.V ?X_B)))))
    :PRECONDS
        ?I1 (NOT (?X_A HAVE.V ?X_B))
        ?I2 (?X_A (AT.P ?L))
        ?I3 (?X_B (AT.P ?L))
    :POSTCONDS
        ?P1 (?X_A HAVE.V ?X_B)
    :NECESSITIES
        !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
        !N2 (!R4 NECESSARY-TO-DEGREE 1.0)
)
```

```
(epi-schema ((?x ((adv-a (from.p ?l1)) ((adv-a (to.p ?l2))
   CLIMB.347.V)) ?l2) ** ?e)
    :Roles
        !r1 (?x agent.n)
        !r2 (?l1 location.n)
        !r3 (?l2 location.n)
        !r4 (not (?l1 = ?l2))
        !r5 (?l2 TREE.N)
        !r6 (?x MONKEY.N)
    :Necessities
        !n1 (!r1 necessary-to-degree 1.0)
        !n2 (!r4 necessary-to-degree 1.0)
    :Goals
        ?g1 (?x (want.v (to ((adv-a (at.p ?l2)) be.v))))
    :Preconds
        ?i1 (?x (at.p ?l1))
        ?i2 (not (?x (at.p ?l2)))
    :Postconds
        ?p1 (not (?x (at.p ?l1)))
        ?p2 (?x (at.p ?l2))
)
```

```
(EPI-SCHEMA ((?X_A GET.346.V ?X_B (AT.P-ARG ?L )) ** ?E)
        :ROLES
                !R1 (?X_A AGENT.N)
                !R2 (?X_B INANIMATE_OBJECT.N)
                !R3 (?L LOCATION.N)
                !R4 (NOT (?X_A = ?X_B))
                !R5 (?X_A MONKEY.N)
                !R6 (?X_B COCOANUT.N)
                !R7 (?X_C GROUND.N)
                !R8 (?X_B (TO.P ?X_C))
        :GOALS
                ?G1 (?X_A (WANT.V (THAT (?X_A (HAVE.V ?X_B)))))
        :PRECONDS
                ?I1 (NOT (?X_A HAVE.V ?X_B))
                ?I2 (?X_A (AT.P ?L))
                ?I3 (?X_B (AT.P ?L))
        :POSTCONDS
                ?P1 (?X_A HAVE.V ?X_B)
        :NECESSITIES
                !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
                !N2 (!R4 NECESSARY-TO-DEGREE 1.0)
)
```

```
(epi-schema ((?x ((adv-a (from.p ?l1)) ((adv-a (to.p ?l2))
     CLIMB.347.V)) ?l2) ** ?e)
        :Roles
                !r1 (?x agent.n)
                !r2 (?l1 location.n)
                !r3 (?l2 location.n)
                !r4 (not (?l1 = ?l2))
                !r5 (?l2 TREE.N)
                !r6 (?x MONKEY.N)
        :Necessities
                !n1 (!r1 necessary-to-degree 1.0)
                !n2 (!r4 necessary-to-degree 1.0)
        :Goals
                ?g1 (?x (want.v (to ((adv-a (at.p ?l2)) be.v))))
        :Preconds
                ?i1 (?x (at.p ?l1))
                ?i2 (not (?x (at.p ?l2)))
        :Postconds
                ?p1 (not (?x (at.p ?l1)))
                ?p2 (?x (at.p ?l2))
)
```

We can link pre- and post-conditions to hypothesize an intentional multi-step schema. (Like GENESIS.)

```
(EPI-SCHEMA ((?X_A GET.346.V ?X_B (AT.P-ARG ?L )) ** ?E)
    :ROLES
        !R1 (?X_A AGENT.N)
        !R2 (?X_B INANIMATE_OBJECT.N)
        !R3 (?L LOCATION.N)
        !R4 (NOT (?X_A = ?X_B))
        !R5 (?X_A MONKEY.N)
        !R6 (?X_B COCOANUT.N)
        !R7 (?X_C GROUND.N)
        !R8 (?X_B (TO.P ?X_C))
    :GOALS
        ?G1 (?X_A (WANT.V (THAT (?X_A (HAVE.V ?X_B)))))
    :PRECONDS
        ?I1 (NOT (?X_A HAVE.V ?X_B))
        ?I2 (?X_A (AT.P ?L))
        ?I3 (?X_B (AT.P ?L))
    :POSTCONDS
        ?P1 (?X_A HAVE.V ?X_B)
    :NECESSITIES
        !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
        !N2 (!R4 NECESSARY-TO-DEGREE 1.0)
)
```

?X_A ← ?x

?L ← ?l2

```
(epi-schema ((?x ((adv-a (from.p ?l1)) ((adv-a (to.p ?l2))
      CLIMB.347.V)) ?l2) ** ?e)
    :Roles
        !r1 (?x agent.n)
        !r2 (?l1 location.n)
        !r3 (?l2 location.n)
        !r4 (not (?l1 = ?l2))
        !r5 (?l2 TREE.N)
        !r6 (?x MONKEY.N)
    :Necessities
        !n1 (!r1 necessary-to-degree 1.0)
        !n2 (!r4 necessary-to-degree 1.0)
    :Goals
        ?g1 (?x (want.v (to ((adv-a (at.p ?l2)) be.v))))
    :Preconds
        ?i1 (?x (at.p ?l1))
        ?i2 (not (?x (at.p ?l2)))
    :Postconds
        ?p1 (not (?x (at.p ?l1)))
        ?p2 (?x (at.p ?l2))
)
```

```
(EPI-SCHEMA ((?x GET.346.V ?X_B (AT.P-ARG ?l2 )) ** ?E)
        :ROLES
                !R1 (?x AGENT.N)
                !R2 (?X_B INANIMATE_OBJECT.N)
                !R3 (?l2  LOCATION.N)
                !R4 (NOT (?x = ?X_B))
                !R5 (?x MONKEY.N)
                !R6 (?X_B COCOANUT.N)
                !R7 (?X_C GROUND.N)
                !R8 (?X_B (TO.P ?X_C))
        :GOALS
                ?G1 (?x (WANT.V (THAT (?x (HAVE.V ?X_B)))))
        :PRECONDS
                ?I1 (NOT (?x HAVE.V ?X_B))
                ?I2 (?x (AT.P ?l2 ))
                ?I3 (?X_B (AT.P ?l2 ))
        :POSTCONDS
                ?P1 (?x HAVE.V ?X_B)
        :NECESSITIES
                !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
                !N2 (!R4 NECESSARY-TO-DEGREE 1.0)
)
```

```
(epi-schema ((?x ((adv-a (from.p ?l1)) ((adv-a (to.p ?l2))
     CLIMB.347.V)) ?l2) ** ?e)
        :Roles
                !r1 (?x agent.n)
                !r2 (?l1 location.n)
                !r3 (?l2 location.n)
                !r4 (not (?l1 = ?l2))
                !r5 (?l2 TREE.N)
                !r6 (?x MONKEY.N)
        :Necessities
                !n1 (!r1 necessary-to-degree 1.0)
                !n2 (!r4 necessary-to-degree 1.0)
        :Goals
                ?g1 (?x (want.v (to ((adv-a (at.p ?l2)) be.v))))
        :Preconds
                ?i1 (?x (at.p ?l1))
                ?i2 (not (?x (at.p ?l2)))
        :Postconds
                ?p1 (not (?x (at.p ?l1)))
                ?p2 (?x (at.p ?l2))
)
```

```
(EPI-SCHEMA ((?x GET.346.V ?X_B (AT.P-ARG ?l2 )) ** ?E)
        :ROLES
                !R1 (?x AGENT.N)
                !R2 (?X_B INANIMATE_OBJECT.N)
                !R3 (?l2  LOCATION.N)
                !R4 (NOT (?x = ?X_B))
                !R5 (?x MONKEY.N)
                !R6 (?X_B COCOANUT.N)
                !R7 (?X_C GROUND.N)
                !R8 (?X_B (TO.P ?X_C))
        :GOALS
                ?G1 (?x (WANT.V (THAT (?x (HAVE.V ?X_B)))))
        :PRECONDS
                ?I1 (NOT (?x HAVE.V ?X_B))
                ?I2 (?x (AT.P ?l2 ))
                ?I3 (?X_B (AT.P ?l2 ))
        :POSTCONDS
                ?P1 (?x HAVE.V ?X_B)
        :NECESSITIES
                !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
                !N2 (!R4 NECESSARY-TO-DEGREE 1.0)
)
```

```
(epi-schema ((?x ((adv-a (from.p ?l1)) ((adv-a (to.p ?l2))
     CLIMB.347.V)) ?l2) ** ?e)
        :Roles
                !r1 (?x agent.n)
                !r2 (?l1 location.n)
                !r3 (?l2 location.n)
                !r4 (not (?l1 = ?l2))
                !r5 (?l2 TREE.N)
                !r6 (?x MONKEY.N)
        :Necessities
                !n1 (!r1 necessary-to-degree 1.0)
                !n2 (!r4 necessary-to-degree 1.0)
        :Goals
                ?g1 (?x (want.v (to ((adv-a (at.p ?l2)) be.v))))
        :Preconds
                ?i1 (?x (at.p ?l1))
                ?i2 (not (?x (at.p ?l2)))
        :Postconds
                ?p1 (not (?x (at.p ?l1)))
                ?p2 (?x (at.p ?l2))
)
```

```
(EPI-SCHEMA ((?X_D EAT.323.V ?X_E) ** E443.SK)
        :ROLES
                !R1 (?X_D AGENT.N)
                !R2 (?X_E FOOD.N)
                !R3 (?X_D MONKEY.N)
                !R4 (?X_E COCOANUT.N)
                !R5 (?X_F GROUND.N)
                !R6 (?X_E (TO.P ?X_F))
        :GOALS
                ?G1 (?X_D (WANT.V (THAT (NOT (?X_D
                        HUNGRY.A)))))
        :PRECONDS
                ?I1 (?X_D HAVE.V ?X_E)
                ?I2 (?X_D HUNGRY.A)
        :POSTCONDS
                ?P1 (NOT (?X_D (HAVE.V ?X_E)))
                ?P2 (NOT (?X_D HUNGRY.A))
        :NECESSITIES
                !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
)
```

```
(EPI-SCHEMA ((?x GET.346.V ?X_B (AT.P-ARG ?l2 )) ** ?E)
        :ROLES
                !R1 (?x AGENT.N)
                !R2 (?X_B INANIMATE_OBJECT.N)
                !R3 (?l2  LOCATION.N)
                !R4 (NOT (?x = ?X_B))
                !R5 (?x MONKEY.N)
                !R6 (?X_B COCOANUT.N)
                !R7 (?X_C GROUND.N)
                !R8 (?X_B (TO.P ?X_C))
        :GOALS
                ?G1 (?x (WANT.V (THAT (?x (HAVE.V ?X_B)))))
        :PRECONDS
                ?I1 (NOT (?x HAVE.V ?X_B))
                ?I2 (?x (AT.P ?l2 ))
                ?I3 (?X_B (AT.P ?l2 ))
        :POSTCONDS
                ?P1 (?x HAVE.V ?X_B)
        :NECESSITIES
                !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
                !N2 (!R4 NECESSARY-TO-DEGREE 1.0)
)
```

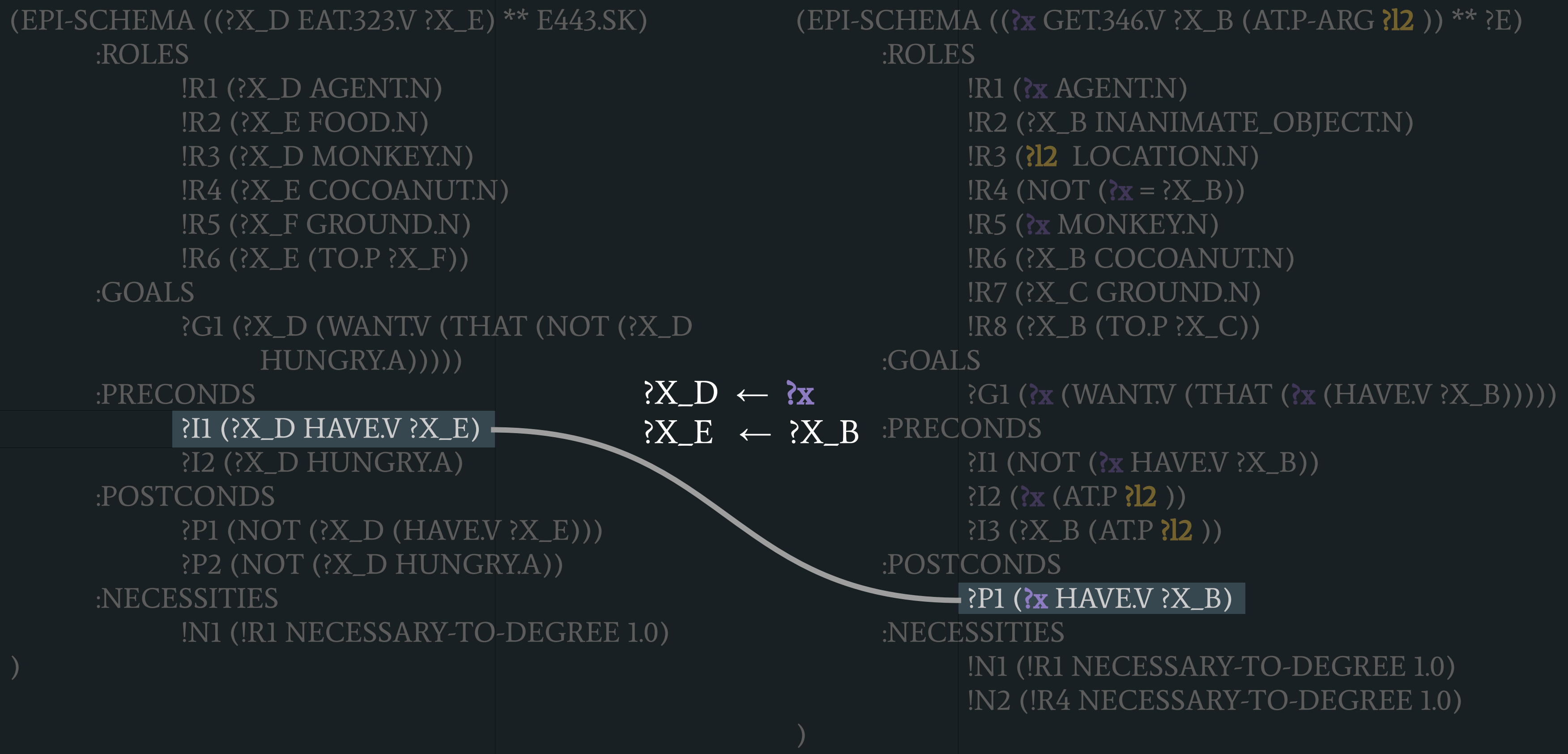
```
(EPI-SCHEMA ((?X_D EAT.323.V ?X_E) ** E443.SK)
        :ROLES
                !R1 (?X_D AGENT.N)
                !R2 (?X_E FOOD.N)
                !R3 (?X_D MONKEY.N)
                !R4 (?X_E COCOANUT.N)
                !R5 (?X_F GROUND.N)
                !R6 (?X_E (TO.P ?X_F))
        :GOALS
                ?G1 (?X_D (WANT.V (THAT (NOT (?X_D
                        HUNGRY.A)))))
        :PRECONDS
                ?I1 (?X_D HAVE.V ?X_E)
                ?I2 (?X_D HUNGRY.A)
        :POSTCONDS
                ?P1 (NOT (?X_D (HAVE.V ?X_E)))
                ?P2 (NOT (?X_D HUNGRY.A))
        :NECESSITIES
                !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
)
```

```
(EPI-SCHEMA ((?x GET.346.V ?X_B (AT.P-ARG ?l2 )) ** ?E)
        :ROLES
                !R1 (?x AGENT.N)
                !R2 (?X_B INANIMATE_OBJECT.N)
                !R3 (?l2  LOCATION.N)
                !R4 (NOT (?x = ?X_B))
                !R5 (?x MONKEY.N)
                !R6 (?X_B COCOANUT.N)
                !R7 (?X_C GROUND.N)
                !R8 (?X_B (TO.P ?X_C))
        :GOALS
                ?G1 (?x (WANT.V (THAT (?x (HAVE.V ?X_B)))))
        :PRECONDS
                ?I1 (NOT (?x HAVE.V ?X_B))
                ?I2 (?x (AT.P ?l2 ))
                ?I3 (?X_B (AT.P ?l2 ))
        :POSTCONDS
                ?P1 (?x HAVE.V ?X_B)
        :NECESSITIES
                !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
                !N2 (!R4 NECESSARY-TO-DEGREE 1.0)
)
```

We got this “eat” schema, too, from the same story. We can link this with the others as well.

```
(EPI-SCHEMA ((?X_D EAT.323.V ?X_E) ** E443.SK)
        :ROLES
                !R1 (?X_D AGENT.N)
                !R2 (?X_E FOOD.N)
                !R3 (?X_D MONKEY.N)
                !R4 (?X_E COCOANUT.N)
                !R5 (?X_F GROUND.N)
                !R6 (?X_E (TO.P ?X_F))
        :GOALS
                ?G1 (?X_D (WANT.V (THAT (NOT (?X_D
                        HUNGRY.A)))))
        :PRECONDS
                ?I1 (?X_D HAVE.V ?X_E)
                ?I2 (?X_D HUNGRY.A)
        :POSTCONDS
                ?P1 (NOT (?X_D (HAVE.V ?X_E)))
                ?P2 (NOT (?X_D HUNGRY.A))
        :NECESSITIES
                !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
)
```

```
(EPI-SCHEMA ((?x GET.346.V ?X_B (AT.P-ARG ?l2 )) ** ?E)
        :ROLES
                !R1 (?x AGENT.N)
                !R2 (?X_B INANIMATE_OBJECT.N)
                !R3 (?l2  LOCATION.N)
                !R4 (NOT (?x = ?X_B))
                !R5 (?x MONKEY.N)
                !R6 (?X_B COCOANUT.N)
                !R7 (?X_C GROUND.N)
                !R8 (?X_B (TO.P ?X_C))
        :GOALS
                ?G1 (?x (WANT.V (THAT (?x (HAVE.V ?X_B)))))
        :PRECONDS
                ?I1 (NOT (?x HAVE.V ?X_B))
                ?I2 (?x (AT.P ?l2 ))
                ?I3 (?X_B (AT.P ?l2 ))
        :POSTCONDS
                ?P1 (?x HAVE.V ?X_B)
        :NECESSITIES
                !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
                !N2 (!R4 NECESSARY-TO-DEGREE 1.0)
)
```



```
(EPI-SCHEMA ((?X_D EAT.323.V ?X_E) ** E443.SK)
        :ROLES
                !R1 (?X_D AGENT.N)
                !R2 (?X_E FOOD.N)
                !R3 (?X_D MONKEY.N)
                !R4 (?X_E COCOANUT.N)
                !R5 (?X_F GROUND.N)
                !R6 (?X_E (TO.P ?X_F))
        :GOALS
                ?G1 (?X_D (WANT.V (THAT (NOT (?X_D
                        HUNGRY.A)))))
        :PRECONDS
                ?I1 (?X_D HAVE.V ?X_E)
                ?I2 (?X_D HUNGRY.A)
        :POSTCONDS
                ?P1 (NOT (?X_D (HAVE.V ?X_E)))
                ?P2 (NOT (?X_D HUNGRY.A))
        :NECESSITIES
                !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
)
```

?X_D ← ?x

?X_E ← ?X_B

```
(EPI-SCHEMA ((?x GET.346.V ?X_B (AT.P-ARG ?l2 )) ** ?E)
        :ROLES
                !R1 (?x AGENT.N)
                !R2 (?X_B INANIMATE_OBJECT.N)
                !R3 (?l2  LOCATION.N)
                !R4 (NOT (?x = ?X_B))
                !R5 (?x MONKEY.N)
                !R6 (?X_B COCOANUT.N)
                !R7 (?X_C GROUND.N)
                !R8 (?X_B (TO.P ?X_C))
        :GOALS
                ?G1 (?x (WANT.V (THAT (?x (HAVE.V ?X_B)))))
        :PRECONDS
                ?I1 (NOT (?x HAVE.V ?X_B))
                ?I2 (?x (AT.P ?l2 ))
                ?I3 (?X_B (AT.P ?l2 ))
        :POSTCONDS
                ?P1 (?x HAVE.V ?X_B)
        :NECESSITIES
                !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
                !N2 (!R4 NECESSARY-TO-DEGREE 1.0)

)
```

```
(EPI-SCHEMA ((?x EAT.323.V ?X_B) ** E443.SK)
        :ROLES
                !R1 (?x AGENT.N)
                !R2 (?X_B FOOD.N)
                !R3 (?x MONKEY.N)
                !R4 (?X_B COCOANUT.N)
                !R5 (?X_F GROUND.N)
                !R6 (?X_B (TO.P ?X_F))
        :GOALS
                ?G1 (?x (WANT.V (THAT (NOT (?x
                        HUNGRY.A)))))
        :PRECONDS
                ?I1 (?x HAVE.V ?X_B)
                ?I2 (?x HUNGRY.A)
        :POSTCONDS
                ?P1 (NOT (?x (HAVE.V ?X_B)))
                ?P2 (NOT (?x HUNGRY.A))
        :NECESSITIES
                !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
)
```

```
(EPI-SCHEMA ((?x GET.346.V ?X_B (AT.P-ARG ?l2 )) ** ?E)
        :ROLES
                !R1 (?x AGENT.N)
                !R2 (?X_B INANIMATE_OBJECT.N)
                !R3 (?l2  LOCATION.N)
                !R4 (NOT (?x = ?X_B))
                !R5 (?x MONKEY.N)
                !R6 (?X_B COCOANUT.N)
                !R7 (?X_C GROUND.N)
                !R8 (?X_B (TO.P ?X_C))
        :GOALS
                ?G1 (?x (WANT.V (THAT (?x (HAVE.V ?X_B)))))
        :PRECONDS
                ?I1 (NOT (?x HAVE.V ?X_B))
                ?I2 (?x (AT.P ?l2 ))
                ?I3 (?X_B (AT.P ?l2 ))
        :POSTCONDS
                ?P1 (?x HAVE.V ?X_B)
        :NECESSITIES
                !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
                !N2 (!R4 NECESSARY-TO-DEGREE 1.0)
)
```

So, all together….

```
(epi-schema ((?x ((adv-a (from.p ?l1)) ((adv-a (to.p ?l2))
     CLIMB.347.V)) ?l2) ** ?e)
          :Roles
                    !r1 (?x agent.n)
                    !r2 (?l1 location.n)
                    !r3 (?l2 location.n)
                    !r4 (not (?l1 = ?l2))
                    !r5 (?l2 TREE.N)
                    !r6 (?x MONKEY.N)
          :Necessities
                    !n1 (!r1 necessary-to-degree 1.0)
                    !n2 (!r4 necessary-to-degree 1.0)
          :Goals
                    ?g1 (?x (want.v (to ((adv-a (at.p ?l2)) be.v))))
          :Preconds
                    ?i1 (?x (at.p ?l1))
                    ?i2 (not (?x (at.p ?l2)))
          :Postconds
                    ?p1 (not (?x (at.p ?l1)))
                    ?p2 (?x (at.p ?l2))
)
```

```
(EPI-SCHEMA ((?x GET.346.V ?X_B (AT.P-ARG ?l2 )) ** ?E)
          :ROLES
                    !R1 (?x AGENT.N)
                    !R2 (?X_B INANIMATE_OBJECT.N)
                    !R3 (?l2  LOCATION.N)
                    !R4 (NOT (?x = ?X_B))
                    !R5 (?x MONKEY.N)
                    !R6 (?X_B COCOANUT.N)
                    !R7 (?X_C GROUND.N)
                    !R8 (?X_B (TO.P ?X_C))
          :GOALS
                    ?G1 (?x (WANT.V (THAT (?x (HAVE.V
?X_B)))))
          :PRECONDS
                    ?I1 (NOT (?x HAVE.V ?X_B))
                    ?I2 (?x (AT.P ?l2 ))
                    ?I3 (?X_B (AT.P ?l2 ))
          :POSTCONDS
                    ?P1 (?x HAVE.V ?X_B)
          :NECESSITIES
                    !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
                    !N2 (!R4 NECESSARY-TO-DEGREE 1.0)
)
```

```
(EPI-SCHEMA ((?x EAT.323.V ?X_B) ** E443.SK)
          :ROLES
                    !R1 (?x AGENT.N)
                    !R2 (?X_B FOOD.N)
                    !R3 (?x MONKEY.N)
                    !R4 (?X_B COCOANUT.N)
                    !R5 (?X_F GROUND.N)
                    !R6 (?X_B (TO.P ?X_F))
          :GOALS
                    ?G1 (?x (WANT.V (THAT (NOT (?x
                    HUNGRY.A)))))
          :PRECONDS
                    ?I1 (?x HAVE.V ?X_B)
                    ?I2 (?x HUNGRY.A)
          :POSTCONDS
                    ?P1 (NOT (?x (HAVE.V ?X_B)))
                    ?P2 (NOT (?x HUNGRY.A))
          :NECESSITIES
                    !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
)
```

So, all together....

```
(epi-schema ((?x ((adv-a (from.p ?l1)) ((adv-a (to.p ?l2))
      CLIMB.347.V)) ?l2) ** ?e)
        :Roles
                !r1 (?x agent.n)
                !r2 (?l1 location.n)
                !r3 (?l2 location.n)
                !r4 (not (?l1 = ?l2))
                !r5 (?l2 TREE.N)
                !r6 (?x MONKEY.N)
        :Necessities
                !n1 (!r1 necessary-to-degree 1.0)
                !n2 (!r4 necessary-to-degree 1.0)
        :Goals
                ?g1 (?x (want.v (to ((adv-a (at.p ?l2)) be.v))))
        :Preconds
                ?i1 (?x (at.p ?l1))
                ?i2 (not (?x (at.p ?l2)))
        :Postconds
                ?p1 (not (?x (at.p ?l1)))
                ?p2 (?x (at.p ?l2))
)
```

```
(EPI-SCHEMA ((?x GET.346.V ?X_B (AT.P-ARG ?l2 )) ** ?E)
        :ROLES
                !R1 (?x AGENT.N)
                !R2 (?X_B INANIMATE_OBJECT.N)
                !R3 (?l2  LOCATION.N)
                !R4 (NOT (?x = ?X_B))
                !R5 (?x MONKEY.N)
                !R6 (?X_B COCOANUT.N)
                !R7 (?X_C GROUND.N)
                !R8 (?X_B (TO.P ?X_C))
        :GOALS
                ?G1 (?x (WANT.V (THAT (?x (HAVE.V
?X_B)))))
        :PRECONDS
                ?I1 (NOT (?x HAVE.V ?X_B))
                ?I2 (?x (AT.P ?l2 ))
                ?I3 (?X_B (AT.P ?l2 ))
        :POSTCONDS
                ?P1 (?x HAVE.V ?X_B)
        :NECESSITIES
                !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
                !N2 (!R4 NECESSARY-TO-DEGREE 1.0)
)
```

```
(EPI-SCHEMA ((?x EAT.323.V ?X_B) ** E443.SK)
        :ROLES
                !R1 (?x AGENT.N)
                !R2 (?X_B FOOD.N)
                !R3 (?x MONKEY.N)
                !R4 (?X_B COCOANUT.N)
                !R5 (?X_F GROUND.N)
                !R6 (?X_B (TO.P ?X_F))
        :GOALS
                ?G1 (?x (WANT.V (THAT (NOT (?x
                HUNGRY.A)))))
        :PRECONDS
                ?I1 (?x HAVE.V ?X_B)
                ?I2 (?x HUNGRY.A)
        :POSTCONDS
                ?P1 (NOT (?x (HAVE.V ?X_B)))
                ?P2 (NOT (?x HUNGRY.A))
        :NECESSITIES
                !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
)
```

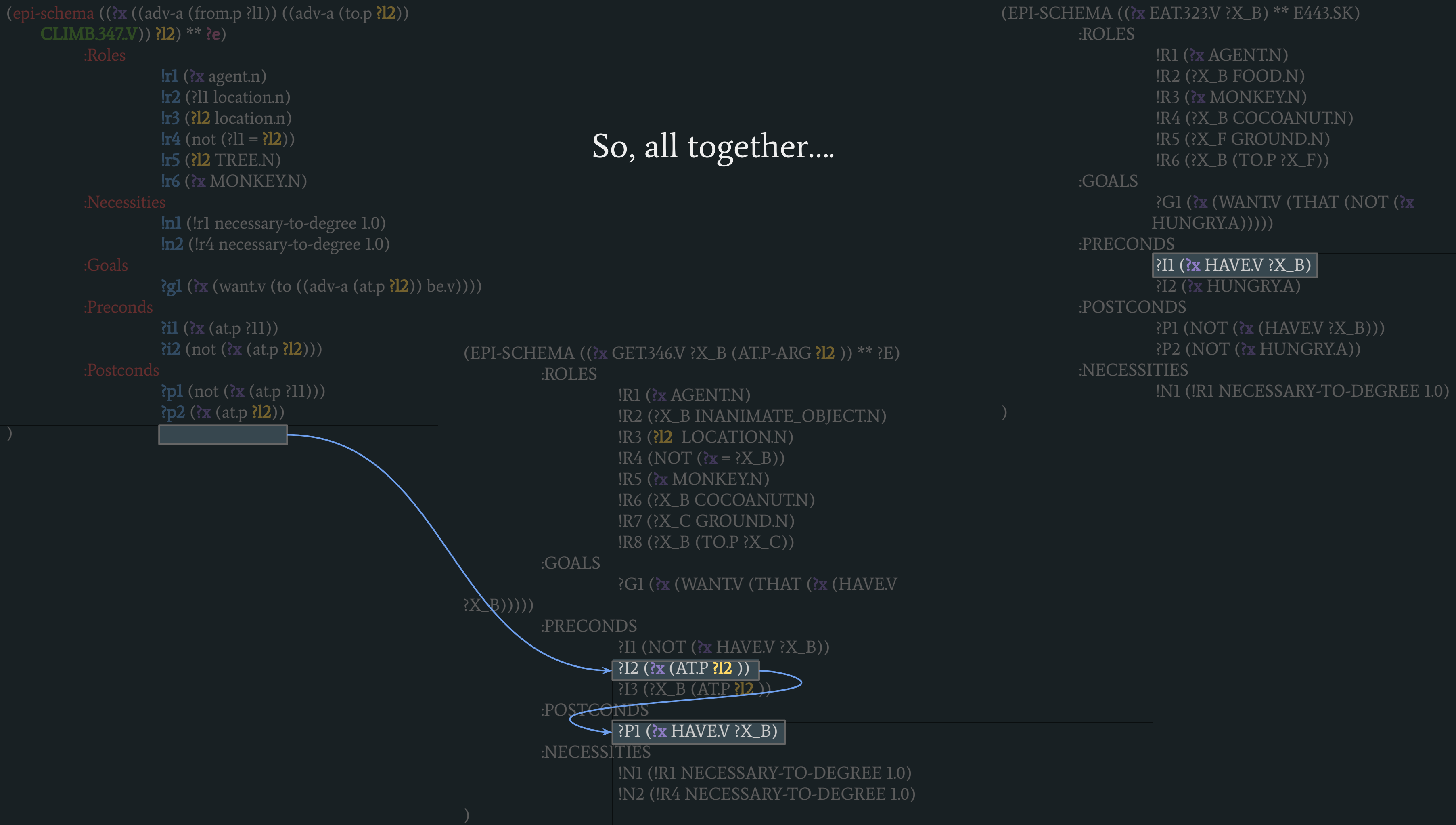
So, all together....

```
(epi-schema ((?x ((adv-a (from.p ?l1)) ((adv-a (to.p ?l2))
     CLIMB.347.V)) ?l2) ** ?e)
          :Roles
                    !r1 (?x agent.n)
                    !r2 (?l1 location.n)
                    !r3 (?l2 location.n)
                    !r4 (not (?l1 = ?l2))
                    !r5 (?l2 TREE.N)
                    !r6 (?x MONKEY.N)
          :Necessities
                    !n1 (!r1 necessary-to-degree 1.0)
                    !n2 (!r4 necessary-to-degree 1.0)
          :Goals
                    ?g1 (?x (want.v (to ((adv-a (at.p ?l2)) be.v))))
          :Preconds
                    ?i1 (?x (at.p ?l1))
                    ?i2 (not (?x (at.p ?l2)))
          :Postconds
                    ?p1 (not (?x (at.p ?l1)))
                    ?p2 (?x (at.p ?l2))
)
```

```
(EPI-SCHEMA ((?x GET.346.V ?X_B (AT.P-ARG ?l2 )) ** ?E)
          :ROLES
                    !R1 (?x AGENT.N)
                    !R2 (?X_B INANIMATE_OBJECT.N)
                    !R3 (?l2  LOCATION.N)
                    !R4 (NOT (?x = ?X_B))
                    !R5 (?x MONKEY.N)
                    !R6 (?X_B COCOANUT.N)
                    !R7 (?X_C GROUND.N)
                    !R8 (?X_B (TO.P ?X_C))
          :GOALS
                    ?G1 (?x (WANT.V (THAT (?x (HAVE.V
?X_B)))))
          :PRECONDS
                    ?I1 (NOT (?x HAVE.V ?X_B))
                    ?I2 (?x (AT.P ?l2 ))
                    ?I3 (?X_B (AT.P ?l2 ))
          :POSTCONDS
                    ?P1 (?x HAVE.V ?X_B)
          :NECESSITIES
                    !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
                    !N2 (!R4 NECESSARY-TO-DEGREE 1.0)
)
```

```
(EPI-SCHEMA ((?x EAT.323.V ?X_B) ** E443.SK)
          :ROLES
                    !R1 (?x AGENT.N)
                    !R2 (?X_B FOOD.N)
                    !R3 (?x MONKEY.N)
                    !R4 (?X_B COCOANUT.N)
                    !R5 (?X_F GROUND.N)
                    !R6 (?X_B (TO.P ?X_F))
          :GOALS
                    ?G1 (?x (WANT.V (THAT (NOT (?x
                   HUNGRY.A)))))
          :PRECONDS
                    ?I1 (?x HAVE.V ?X_B)
                    ?I2 (?x HUNGRY.A)
          :POSTCONDS
                    ?P1 (NOT (?x (HAVE.V ?X_B)))
                    ?P2 (NOT (?x HUNGRY.A))
          :NECESSITIES
                    !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
)
```

So, all together....

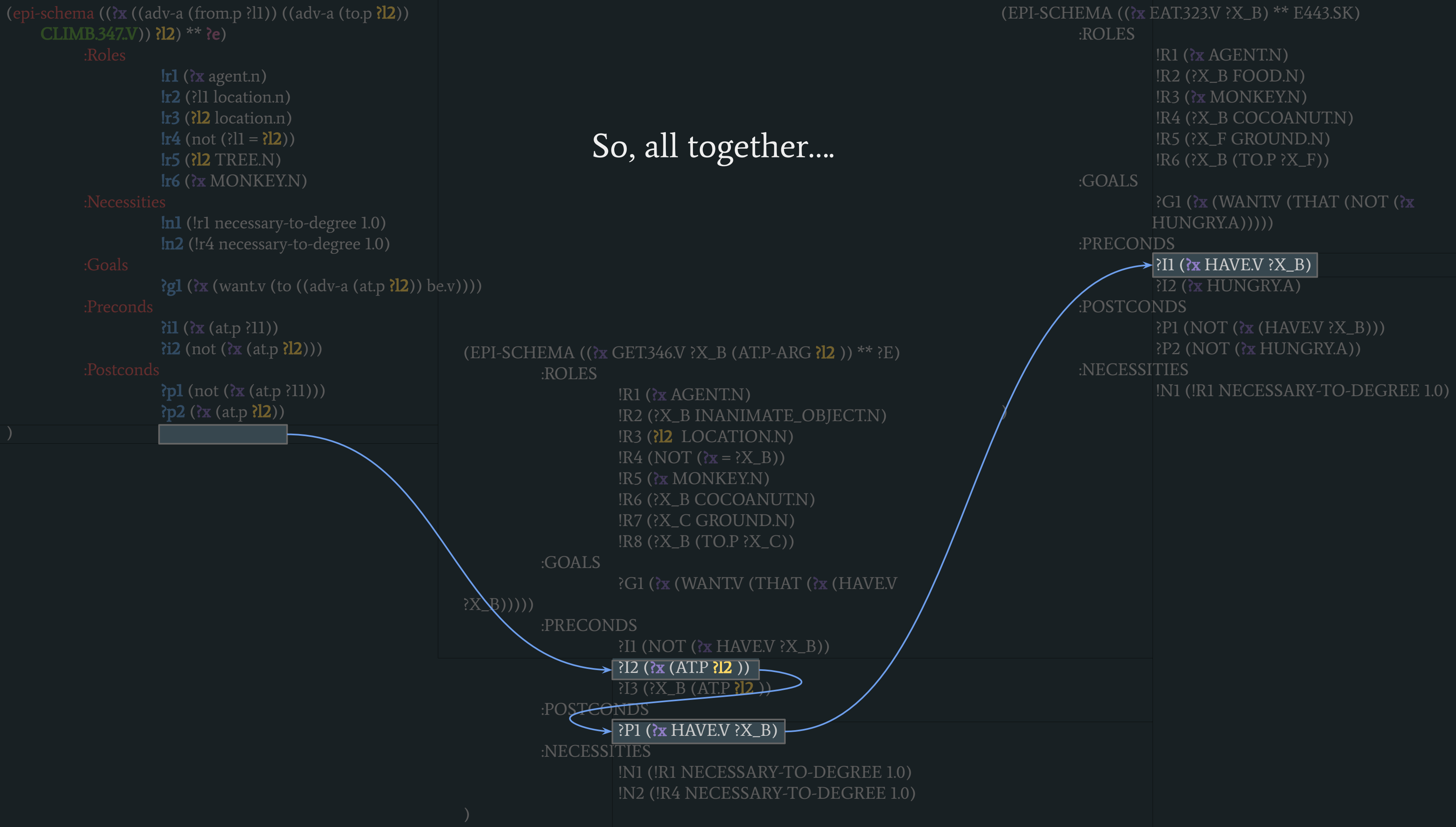


```
(epi-schema ((?x ((adv-a (from.p ?l1)) ((adv-a (to.p ?l2))
    CLIMB.347.V)) ?l2) ** ?e)
        :Roles
                !r1 (?x agent.n)
                !r2 (?l1 location.n)
                !r3 (?l2 location.n)
                !r4 (not (?l1 = ?l2))
                !r5 (?l2 TREE.N)
                !r6 (?x MONKEY.N)
        :Necessities
                !n1 (!r1 necessary-to-degree 1.0)
                !n2 (!r4 necessary-to-degree 1.0)
        :Goals
                ?g1 (?x (want.v (to ((adv-a (at.p ?l2)) be.v))))
        :Preconds
                ?i1 (?x (at.p ?l1))
                ?i2 (not (?x (at.p ?l2)))
        :Postconds
                ?p1 (not (?x (at.p ?l1)))
                ?p2 (?x (at.p ?l2))
)
```

```
(EPI-SCHEMA ((?x GET.346.V ?X_B (AT.P-ARG ?l2 )) ** ?E)
        :ROLES
                !R1 (?x AGENT.N)
                !R2 (?X_B INANIMATE_OBJECT.N)
                !R3 (?l2  LOCATION.N)
                !R4 (NOT (?x = ?X_B))
                !R5 (?x MONKEY.N)
                !R6 (?X_B COCOANUT.N)
                !R7 (?X_C GROUND.N)
                !R8 (?X_B (TO.P ?X_C))
        :GOALS
                ?G1 (?x (WANT.V (THAT (?x (HAVE.V
?X_B)))))
        :PRECONDS
                ?I1 (NOT (?x HAVE.V ?X_B))
                ?I2 (?x (AT.P ?l2 ))
                ?I3 (?X_B (AT.P ?l2 ))
        :POSTCONDS
                ?P1 (?x HAVE.V ?X_B)
        :NECESSITIES
                !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
                !N2 (!R4 NECESSARY-TO-DEGREE 1.0)
)
```

```
(EPI-SCHEMA ((?x EAT.323.V ?X_B) ** E443.SK)
        :ROLES
                !R1 (?x AGENT.N)
                !R2 (?X_B FOOD.N)
                !R3 (?x MONKEY.N)
                !R4 (?X_B COCOANUT.N)
                !R5 (?X_F GROUND.N)
                !R6 (?X_B (TO.P ?X_F))
        :GOALS
                ?G1 (?x (WANT.V (THAT (NOT (?x
                HUNGRY.A)))))
        :PRECONDS
                ?I1 (?x HAVE.V ?X_B)
                ?I2 (?x HUNGRY.A)
        :POSTCONDS
                ?P1 (NOT (?x (HAVE.V ?X_B)))
                ?P2 (NOT (?x HUNGRY.A))
        :NECESSITIES
                !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
)
```

So, all together....

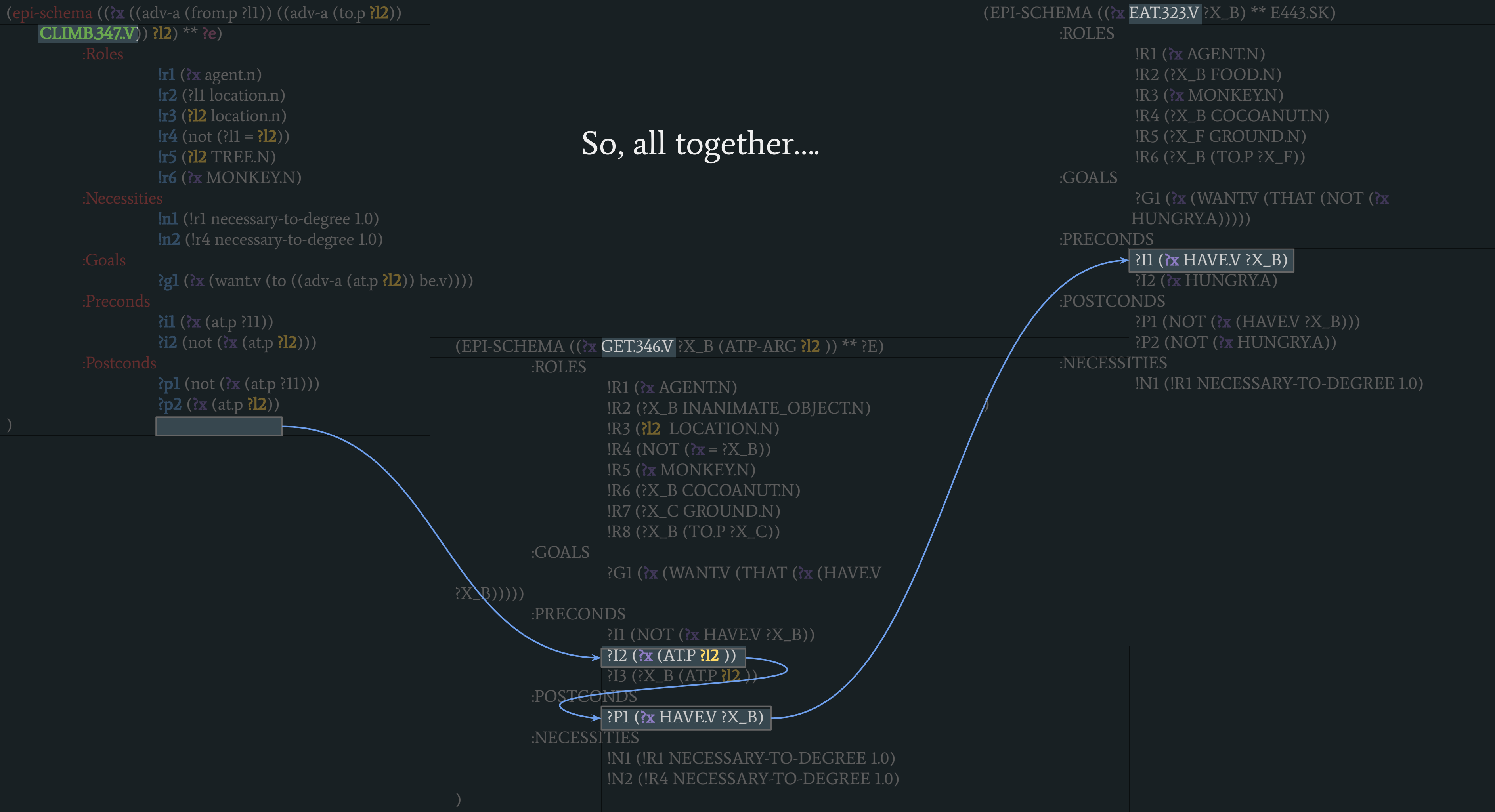


```
(epi-schema ((?x ((adv-a (from.p ?l1)) ((adv-a (to.p ?l2))
     CLIMB.347.V)) ?l2) ** ?e)
        :Roles
                !r1 (?x agent.n)
                !r2 (?l1 location.n)
                !r3 (?l2 location.n)
                !r4 (not (?l1 = ?l2))
                !r5 (?l2 TREE.N)
                !r6 (?x MONKEY.N)
        :Necessities
                !n1 (!r1 necessary-to-degree 1.0)
                !n2 (!r4 necessary-to-degree 1.0)
        :Goals
                ?g1 (?x (want.v (to ((adv-a (at.p ?l2)) be.v))))
        :Preconds
                ?i1 (?x (at.p ?l1))
                ?i2 (not (?x (at.p ?l2)))
        :Postconds
                ?p1 (not (?x (at.p ?l1)))
                ?p2 (?x (at.p ?l2))
)
```

```
(EPI-SCHEMA ((?x GET.346.V ?X_B (AT.P-ARG ?l2 )) ** ?E)
        :ROLES
                !R1 (?x AGENT.N)
                !R2 (?X_B INANIMATE_OBJECT.N)
                !R3 (?l2  LOCATION.N)
                !R4 (NOT (?x = ?X_B))
                !R5 (?x MONKEY.N)
                !R6 (?X_B COCOANUT.N)
                !R7 (?X_C GROUND.N)
                !R8 (?X_B (TO.P ?X_C))
        :GOALS
                ?G1 (?x (WANT.V (THAT (?x (HAVE.V
?X_B)))))
        :PRECONDS
                ?I1 (NOT (?x HAVE.V ?X_B))
                ?I2 (?x (AT.P ?l2 ))
                ?I3 (?X_B (AT.P ?l2 ))
        :POSTCONDS
                ?P1 (?x HAVE.V ?X_B)
        :NECESSITIES
                !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
                !N2 (!R4 NECESSARY-TO-DEGREE 1.0)
)
```

```
(EPI-SCHEMA ((?x EAT.323.V ?X_B) ** E443.SK)
        :ROLES
                !R1 (?x AGENT.N)
                !R2 (?X_B FOOD.N)
                !R3 (?x MONKEY.N)
                !R4 (?X_B COCOANUT.N)
                !R5 (?X_F GROUND.N)
                !R6 (?X_B (TO.P ?X_F))
        :GOALS
                ?G1 (?x (WANT.V (THAT (NOT (?x
                HUNGRY.A)))))
        :PRECONDS
                ?I1 (?x HAVE.V ?X_B)
                ?I2 (?x HUNGRY.A)
        :POSTCONDS
                ?P1 (NOT (?x (HAVE.V ?X_B)))
                ?P2 (NOT (?x HUNGRY.A))
        :NECESSITIES
                !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
)
```

So, all together….



```
(epi-schema ((?x ((adv-a (from.p ?l1)) ((adv-a (to.p ?l2))
    CLIMB.347.V)) ?l2) ** ?e)
        :Roles
                !r1 (?x agent.n)
                !r2 (?l1 location.n)
                !r3 (?l2 location.n)
                !r4 (not (?l1 = ?l2))
                !r5 (?l2 TREE.N)
                !r6 (?x MONKEY.N)
        :Necessities
                !n1 (!r1 necessary-to-degree 1.0)
                !n2 (!r4 necessary-to-degree 1.0)
        :Goals
                ?g1 (?x (want.v (to ((adv-a (at.p ?l2)) be.v))))
        :Preconds
                ?i1 (?x (at.p ?l1))
                ?i2 (not (?x (at.p ?l2)))
        :Postconds
                ?p1 (not (?x (at.p ?l1)))
                ?p2 (?x (at.p ?l2))
)
```

```
(EPI-SCHEMA ((?x GET.346.V ?X_B (AT.P-ARG ?l2 )) ** ?E)
        :ROLES
                !R1 (?x AGENT.N)
                !R2 (?X_B INANIMATE_OBJECT.N)
                !R3 (?l2  LOCATION.N)
                !R4 (NOT (?x = ?X_B))
                !R5 (?x MONKEY.N)
                !R6 (?X_B COCOANUT.N)
                !R7 (?X_C GROUND.N)
                !R8 (?X_B (TO.P ?X_C))
        :GOALS
                ?G1 (?x (WANT.V (THAT (?x (HAVE.V
?X_B)))))
        :PRECONDS
                ?I1 (NOT (?x HAVE.V ?X_B))
                ?I2 (?x (AT.P ?l2 ))
                ?I3 (?X_B (AT.P ?l2 ))
        :POSTCONDS
                ?P1 (?x HAVE.V ?X_B)
        :NECESSITIES
                !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
                !N2 (!R4 NECESSARY-TO-DEGREE 1.0)
)
```

```
(EPI-SCHEMA ((?x EAT.323.V ?X_B) ** E443.SK)
        :ROLES
                !R1 (?x AGENT.N)
                !R2 (?X_B FOOD.N)
                !R3 (?x MONKEY.N)
                !R4 (?X_B COCOANUT.N)
                !R5 (?X_F GROUND.N)
                !R6 (?X_B (TO.P ?X_F))
        :GOALS
                ?G1 (?x (WANT.V (THAT (NOT (?x
                HUNGRY.A)))))
        :PRECONDS
                ?I1 (?x HAVE.V ?X_B)
                ?I2 (?x HUNGRY.A)
        :POSTCONDS
                ?P1 (NOT (?x (HAVE.V ?X_B)))
                ?P2 (NOT (?x HUNGRY.A))
        :NECESSITIES
                !N1 (!R1 NECESSARY-TO-DEGREE 1.0)
)
```

```
(EPI-SCHEMA ((?X_H CLIMB_GET_EAT.349.PR ?X_G ?X_I) ** ?E)
    :ROLES
        !R1 (?X_G TREE.N)
        !R2 (?X_I INANIMATE_OBJECT.N)
        !R3 (NOT (?X_H = ?X_I))
        !R4 (?X_G LOCATION.N)
        !R5 (?X_I FOOD.N)
        !R6 (?X_I COCOANUT.N)
        !R7 (?X_H MONKEY.N)
    :PRECONDS
        ?I1 (?X_H (AT.P ?L1_2))
        ?I2 (NOT (?X_H (AT.P ?X_G)))
    :STEPS
        ?E1 (?X_H ((ADV-A (FROM.P ?L1)) ((ADV-A (TO.P ?X_G)) CLIMB.347.V)) ?X_G)
        ?E2 (?X_H GET.346.V ?X_I (AT.P-ARG ?X_G))
        ?E3 (?X_H EAT.323.V ?X_I)
    :POSTCONDS
        ?P1 (NOT (?X_H (HAVE.V ?X_I)))
        ?P2 (NOT (?X_H HUNGRY.A))
    :EPISODE-RELATIONS
        !W1 (?E1 BEFORE ?E2)
        !W2 (?E2 BEFORE ?E3)
        !W3 (?E1 DURING ?E)
        !W4 (?E2 DURING ?E)
)
```

```
(EPI-SCHEMA ((?X_H CLIMB_GET_EAT.349.PR ?X_G ?X_I) ** ?E)
        :ROLES
                !R1 (?X_G TREE.N)
                !R2 (?X_I INANIMATE_OBJECT.N)
                !R3 (NOT (?X_H = ?X_I))
                !R4 (?X_G LOCATION.N)
                !R5 (?X_I FOOD.N)
                !R6 (?X_I COCOANUT.N)
                !R7 (?X_H MONKEY.N)
        :PRECONDS
                ?I1 (?X_H (AT.P ?L1_2))
                ?I2 (NOT (?X_H (AT.P ?X_G)))
        :STEPS
                ?E1 (?X_H ((ADV-A (FROM.P ?L1)) ((ADV-A (TO.P ?X_G)) CLIMB.347.V)) ?X_G)
                ?E2 (?X_H GET.346.V ?X_I (AT.P-ARG ?X_G))
                ?E3 (?X_H EAT.323.V ?X_I)
        :POSTCONDS
                ?P1 (NOT (?X_H (HAVE.V ?X_I)))
                ?P2 (NOT (?X_H HUNGRY.A))
        :EPISODE-RELATIONS
                !W1 (?E1 BEFORE ?E2)
                !W2 (?E2 BEFORE ?E3)
                !W3 (?E1 DURING ?E)
                !W4 (?E2 DURING ?E)
)
```

By linking pre- and post-conditions, we’ve formed a multi-step schema:

“Monkey climbs tree to get cocoanut to eat it”

```
(EPI-SCHEMA ((?X_H CLIMB_GET_EAT.349.PR ?X_G ?X_I) ** ?E)
        :ROLES
                !R1 (?X_G TREE.N)
                !R2 (?X_I INANIMATE_OBJECT.N)
                !R3 (NOT (?X_H = ?X_I))
                !R4 (?X_G LOCATION.N)
                !R5 (?X_I FOOD.N)
                !R6 (?X_I COCOANUT.N)
                !R7 (?X_H MONKEY.N)
        :PRECONDS
                ?I1 (?X_H (AT.P ?L1_2))
                ?I2 (NOT (?X_H (AT.P ?X_G)))
        :STEPS
                ?E1 (?X_H ((ADV-A (FROM.P ?L1)) ((ADV-A (TO.P ?X_G)) CLIMB.347.V)) ?X_G)
                ?E2 (?X_H GET.346.V ?X_I (AT.P-ARG ?X_G))
                ?E3 (?X_H EAT.323.V ?X_I)
        :POSTCONDS
                ?P1 (NOT (?X_H (HAVE.V ?X_I)))
                ?P2 (NOT (?X_H HUNGRY.A))
        :EPISODE-RELATIONS
                !W1 (?E1 BEFORE ?E2)
                !W2 (?E2 BEFORE ?E3)
                !W3 (?E1 DURING ?E)
                !W4 (?E2 DURING ?E)
)
```

By linking pre- and post-conditions, we’ve formed a multi-step schema:

“Monkey climbs tree to get cocoanut to eat it”

```
(EPI-SCHEMA ((?X_H CLIMB_GET_EAT.349.PR ?X_G ?X_I) ** ?E)
        :ROLES
                !R1 (?X_G TREE.N)
                !R2 (?X_I INANIMATE_OBJECT.N)
                !R3 (NOT (?X_H = ?X_I))
                !R4 (?X_G LOCATION.N)
                !R5 (?X_I FOOD.N)
                !R6 (?X_I COCOANUT.N)
                !R7 (?X_H MONKEY.N)
        :PRECONDS
                ?I1 (?X_H (AT.P ?L1_2))
                ?I2 (NOT (?X_H (AT.P ?X_G)))
        :STEPS
                ?E1 (?X_H ((ADV-A (FROM.P ?L1)) ((ADV-A (TO.P ?X_G)) CLIMB.347.V)) ?X_G)
                ?E2 (?X_H GET.346.V ?X_I (AT.P-ARG ?X_G))
                ?E3 (?X_H EAT.323.V ?X_I)
        :POSTCONDS
                ?P1 (NOT (?X_H (HAVE.V ?X_I)))
                ?P2 (NOT (?X_H HUNGRY.A))
        :EPISODE-RELATIONS
                !W1 (?E1 BEFORE ?E2)
                !W2 (?E2 BEFORE ?E3)
                !W3 (?E1 DURING ?E)
                !W4 (?E2 DURING ?E)
)
```

New temporal constraints on episodes impose the order

```
(EPI-SCHEMA ((?X_H CLIMB_GET_EAT.349.PR ?X_G ?X_I) ** ?E)
        :ROLES
                !R1 (?X_G TREE.N)
                !R2 (?X_I INANIMATE_OBJECT.N)
                !R3 (NOT (?X_H = ?X_I))
                !R4 (?X_G LOCATION.N)
                !R5 (?X_I FOOD.N)
                !R6 (?X_I COCOANUT.N)
                !R7 (?X_H MONKEY.N)
        :PRECONDS
                ?I1 (?X_H (AT.P ?L1_2))
                ?I2 (NOT (?X_H (AT.P ?X_G)))
        :STEPS
                ?E1 (?X_H ((ADV-A (FROM.P ?L1)) ((ADV-A (TO.P ?X_G)) CLIMB.347.V)) ?X_G)
                ?E2 (?X_H GET.346.V ?X_I (AT.P-ARG ?X_G))
                ?E3 (?X_H EAT.323.V ?X_I)
        :POSTCONDS
                ?P1 (NOT (?X_H (HAVE.V ?X_I)))
                ?P2 (NOT (?X_H HUNGRY.A))
        :EPISODE-RELATIONS
                !W1 (?E1 BEFORE ?E2)
                !W2 (?E2 BEFORE ?E3)
                !W3 (?E1 DURING ?E)
                !W4 (?E2 DURING ?E)
)
```

Same idea, different story…

Let’s demonstrate *predictions*.

```
(EPI-SCHEMA ((?X_C GO_FIND_EAT.566.PR ?X_A ?X_B ?X_D) ** ?E)
        :ROLES
                !R1 (?X_A FIELD.N)
                !R2 (?X_B (OF.P (K GRASS.N)))
                !R3 (?X_B SPOT.N)
                !R4 (?X_C COW.N)
                !R5 (?X_D FOOD.N)
                !R6 (?X_D GRASS.N)
        :PRECONDS
                ?I1 (?X_C (AT.P ?L1_2))
                ?I2 (NOT (?X_C (AT.P ?X_A)))
        :STEPS
                ?E1 (?X_C (OUT.ADV ((ADV-A (TO.P ?X_A))
                        ((ADV-A (FROM.P ?L1)) GO.563.V))) ?X_A)
                ?E2 (?X_C FIND.562.V ?X_B)
                ?E3 (?X_C EAT.564.V ?X_D)
        :POSTCONDS
                ?P1 (NOT (?X_C (HAVE.V ?X_D)))
                ?P2 (NOT (?X_C HUNGRY.A))
        :EPISODE-RELATIONS
                !W1 (?E1 BEFORE ?E2)
                !W2 (?E2 BEFORE ?E3)
                !W3 (?E1 DURING ?E)
                !W4 (?E2 DURING ?E)
)
```

The cow left the barn.

It went out to **the field**.

The other cows were out in **the field**.

The cow found **a spot of grass**.

The cow ate **the grass**.

```
(EPI-SCHEMA ((?X_C GO_FIND_EAT.566.PR ?X_A ?X_B ?X_D) ** ?E)
        :ROLES
                !R1 (?X_A FIELD.N)
                !R2 (?X_B (OF.P (K GRASS.N)))
                !R3 (?X_B SPOT.N)
                !R4 (?X_C COW.N)
                !R5 (?X_D FOOD.N)
                !R6 (?X_D GRASS.N)
        :PRECONDS
                ?I1 (?X_C (AT.P ?L1_2))
                ?I2 (NOT (?X_C (AT.P ?X_A)))
        :STEPS
                ?E1 (?X_C (OUT.ADV ((ADV-A (TO.P ?X_A))
                        ((ADV-A (FROM.P ?L1)) GO.563.V))) ?X_A)
                ?E2 (?X_C FIND.562.V ?X_B)
                ?E3 (?X_C EAT.564.V ?X_D)
        :POSTCONDS
                ?P1 (NOT (?X_C (HAVE.V ?X_D)))
                ?P2 (NOT (?X_C HUNGRY.A))
        :EPISODE-RELATIONS
                !W1 (?E1 BEFORE ?E2)
                !W2 (?E2 BEFORE ?E3)
                !W3 (?E1 DURING ?E)
                !W4 (?E2 DURING ?E)
)
```

The cow left the barn.

It went out to **the field**.

The other cows were out in **the field**.

The cow found **a spot of grass**.

The cow ate **the grass**.

Schema:

“A cow goes to a field, finds a spot of grass, and eats grass”

```
(EPI-SCHEMA ((?X_C GO_FIND_EAT.566.PR ?X_A ?X_B ?X_D) ** ?E)
        :ROLES
                !R1 (?X_A FIELD.N)
                !R2 (?X_B (OF.P (K GRASS.N)))
                !R3 (?X_B SPOT.N)
                !R4 (?X_C COW.N)
                !R5 (?X_D FOOD.N)
                !R6 (?X_D GRASS.N)
        :PRECONDS
                ?I1 (?X_C (AT.P ?L1_2))
                ?I2 (NOT (?X_C (AT.P ?X_A)))
        :STEPS
                ?E1 (?X_C (OUT.ADV ((ADV-A (TO.P ?X_A))
                         ((ADV-A (FROM.P ?L1)) GO.563.V))) ?X_A)
                ?E2 (?X_C FIND.562.V ?X_B)
                ?E3 (?X_C EAT.564.V ?X_D)
        :POSTCONDS
                ?P1 (NOT (?X_C (HAVE.V ?X_D)))
                ?P2 (NOT (?X_C HUNGRY.A))
        :EPISODE-RELATIONS
                !W1 (?E1 BEFORE ?E2)
                !W2 (?E2 BEFORE ?E3)
                !W3 (?E1 DURING ?E)
                !W4 (?E2 DURING ?E)
)
```

The cow left the barn.

It went out to **the field**.

The other cows were out in **the field**.

The cow found **a spot of grass**.

The cow ate **the grass**.

Once learned, this schema unexpectedly matched to another story...

```
(EPI-SCHEMA ((?X_C GO_FIND_EAT.566.PR ?X_A ?X_B ?X_D) ** ?E)
        :ROLES
                !R1 (?X_A FIELD.N)
                !R2 (?X_B (OF.P (K GRASS.N)))
                !R3 (?X_B SPOT.N)
                !R4 (?X_C COW.N)
                !R5 (?X_D FOOD.N)
                !R6 (?X_D GRASS.N)
        :PRECONDS
                ?I1 (?X_C (AT.P ?L1_2))
                ?I2 (NOT (?X_C (AT.P ?X_A)))
        :STEPS
                ?E1 (?X_C (OUT.ADV ((ADV-A (TO.P ?X_A))
                        ((ADV-A (FROM.P ?L1)) GO.563.V))) ?X_A)
                ?E2 (?X_C FIND.562.V ?X_B)
                ?E3 (?X_C EAT.564.V ?X_D)
        :POSTCONDS
                ?P1 (NOT (?X_C (HAVE.V ?X_D)))
                ?P2 (NOT (?X_C HUNGRY.A))
        :EPISODE-RELATIONS
                !W1 (?E1 BEFORE ?E2)
                !W2 (?E2 BEFORE ?E3)
                !W3 (?E1 DURING ?E)
                !W4 (?E2 DURING ?E)
)
```

This is red clover.
The bees like it.
They find sweet nectar in the clover flowers.
They take the nectar home to make honey.
Here is white clover.
It is sweet.
It has nectar, and bees like it, too.
It grows in the fields with red clover and yellow buttercups.
Horses and cows eat clover.

```
(EPI-SCHEMA ((?X_C GO_FIND_EAT.566.PR ?X_A ?X_B ?X_D) ** ?E)
    :ROLES
        !R1 (?X_A FIELD.N)
        !R2 (?X_B (OF.P (K GRASS.N)))
        !R3 (?X_B SPOT.N)
        !R4 (?X_C COW.N)
        !R5 (?X_D FOOD.N)
        !R6 (?X_D GRASS.N)
    :PRECONDS
        ?I1 (?X_C (AT.P ?L1_2))
        ?I2 (NOT (?X_C (AT.P ?X_A)))
    :STEPS
        ?E1 (?X_C (OUT.ADV ((ADV-A (TO.P ?X_A))
            ((ADV-A (FROM.P ?L1)) GO.563.V))) ?X_A)
        ?E2 (?X_C FIND.562.V ?X_B)
        ?E3 (?X_C EAT.564.V ?X_D)
    :POSTCONDS
        ?P1 (NOT (?X_C (HAVE.V ?X_D)))
        ?P2 (NOT (?X_C HUNGRY.A))
    :EPISODE-RELATIONS
        !W1 (?E1 BEFORE ?E2)
        !W2 (?E2 BEFORE ?E3)
        !W3 (?E1 DURING ?E)
        !W4 (?E2 DURING ?E)
)
```

The schema was invoked because of some shared words...

This is red clover.
The bees like it.
They **find** sweet nectar in the clover flowers.
They take the nectar home to make honey.
Here is white clover.
It is sweet.
It has nectar, and bees like it, too.
It grows in the **fields** with red clover and yellow buttercups.
Horses and **cows eat** clover.

```
(EPI-SCHEMA ((?X_C GO_FIND_EAT.566.PR ?X_A ?X_B ?X_D) ** ?E)
      :ROLES
            !R1 (?X_A FIELD.N)
            !R2 (?X_B (OF.P (K GRASS.N)))
            !R3 (?X_B SPOT.N)
            !R4 (?X_C COW.N)
            !R5 (?X_D FOOD.N)
            !R6 (?X_D GRASS.N)
      :PRECONDS
            ?I1 (?X_C (AT.P ?L1_2))
            ?I2 (NOT (?X_C (AT.P ?X_A)))
      :STEPS
            ?E1 (?X_C (OUT.ADV ((ADV-A (TO.P ?X_A))
                  ((ADV-A (FROM.P ?L1)) GO.563.V))) ?X_A)
            ?E2 (?X_C FIND.562.V ?X_B)
            ?E3 (?X_C EAT.564.V ?X_D)
      :POSTCONDS
            ?P1 (NOT (?X_C (HAVE.V ?X_D)))
            ?P2 (NOT (?X_C HUNGRY.A))
      :EPISODE-RELATIONS
            !W1 (?E1 BEFORE ?E2)
            !W2 (?E2 BEFORE ?E3)
            !W3 (?E1 DURING ?E)
            !W4 (?E2 DURING ?E)
)
```

The schema was invoked because of some shared words...and kept because an episode in its EL form matched a step.

```
(BEES715.SK (PLUR BEE.N))

(FLOWERS721.SK ((NN CLOVER.N) (PLUR
     FLOWER.N)))

(NECTAR1.SK SWEET.A)

(NECTAR1.SK NECTAR.N)

(BEES715.SK (PLUR BEE.N))

(FIELD738.SK (PLUR FIELD.N))

((BEES715.SK ((ADV-A (IN.P FLOWERS721.SK))
     FIND.562.V) NECTAR1.SK) ** E722.SK)
```

```
(EPI-SCHEMA ((?X_C GO_FIND_EAT.566.PR ?X_A ?X_B ?X_D) ** ?E)
    :ROLES
        !R1 (?X_A FIELD.N)
        !R2 (?X_B (OF.P (K GRASS.N)))
        !R3 (?X_B SPOT.N)
        !R4 (?X_C COW.N)
        !R5 (?X_D FOOD.N)
        !R6 (?X_D GRASS.N)
    :PRECONDS
        ?I1 (?X_C (AT.P ?L1_2))
        ?I2 (NOT (?X_C (AT.P ?X_A)))
    :STEPS
        ?E1 (?X_C (OUT.ADV ((ADV-A (TO.P ?X_A))
                ((ADV-A (FROM.P ?L1)) GO.563.V))) ?X_A)
        ?E2 (?X_C FIND.562.V ?X_B)
        ?E3 (?X_C EAT.564.V ?X_D)
    :POSTCONDS
        ?P1 (NOT (?X_C (HAVE.V ?X_D)))
        ?P2 (NOT (?X_C HUNGRY.A))
    :EPISODE-RELATIONS
        !W1 (?E1 BEFORE ?E2)
        !W2 (?E2 BEFORE ?E3)
        !W3 (?E1 DURING ?E)
        !W4 (?E2 DURING ?E)
)
```

The schema was invoked because of some shared words…and kept because an episode in its EL form matched a step.

```
(BEES715.SK (PLUR BEE.N))

(FLOWERS721.SK ((NN CLOVER.N) (PLUR
     FLOWER.N)))

(NECTAR1.SK SWEET.A)

(NECTAR1.SK NECTAR.N)

(BEES715.SK (PLUR BEE.N))

(FIELD738.SK (PLUR FIELD.N))

((BEES715.SK ((ADV-A (IN.P FLOWERS721.SK))
     FIND.562.V) NECTAR1.SK) ** E722.SK)
```

```
(EPI-SCHEMA ((?X_C GO_FIND_EAT.566.PR ?X_A ?X_B ?X_D) ** ?E)
        :ROLES
                !R1 (?X_A FIELD.N)
                !R2 (?X_B (OF.P (K GRASS.N)))
                !R3 (?X_B SPOT.N)
                !R4 (?X_C COW.N)
                !R5 (?X_D FOOD.N)
                !R6 (?X_D GRASS.N)
        :PRECONDS
                ?I1 (?X_C (AT.P ?L1_2))
                ?I2 (NOT (?X_C (AT.P ?X_A)))
        :STEPS
                ?E1 (?X_C (OUT.ADV ((ADV-A (TO.P ?X_A))
                        ((ADV-A (FROM.P ?L1)) GO.563.V))) ?X_A)
                ?E2 (?X_C FIND.562.V ?X_B)
                ?E3 (?X_C EAT.564.V ?X_D)
        :POSTCONDS
                ?P1 (NOT (?X_C (HAVE.V ?X_D)))
                ?P2 (NOT (?X_C HUNGRY.A))
        :EPISODE-RELATIONS
                !W1 (?E1 BEFORE ?E2)
                !W2 (?E2 BEFORE ?E3)
                !W3 (?E1 DURING ?E)
                !W4 (?E2 DURING ?E)
)
```

```
(BEES715.SK (PLUR BEE.N))

(FLOWERS721.SK ((NN CLOVER.N) (PLUR
     FLOWER.N)))

(NECTAR1.SK SWEET.A)

(NECTAR1.SK NECTAR.N)

(BEES715.SK (PLUR BEE.N))

(FIELD738.SK (PLUR FIELD.N))

((BEES715.SK ((ADV-A (IN.P FLOWERS721.SK))
     FIND.562.V) NECTAR1.SK) ** E722.SK)
```

```
(EPI-SCHEMA ((?X_C GO_FIND_EAT.566.PR ?X_A ?X_B ?X_D) ** ?E)
        :ROLES
                !R1 (?X_A FIELD.N)
                !R2 (?X_B (OF.P (K GRASS.N)))
                !R3 (?X_B SPOT.N)
                !R4 (?X_C COW.N)
                !R5 (?X_D FOOD.N)
                !R6 (?X_D GRASS.N)
        :PRECONDS
                ?I1 (?X_C (AT.P ?L1_2))
                ?I2 (NOT (?X_C (AT.P ?X_A)))
        :STEPS
                ?E1 (?X_C (OUT.ADV ((ADV-A (TO.P ?X_A))
                         ((ADV-A (FROM.P ?L1)) GO.563.V))) ?X_A)
                ?E2 (?X_C FIND.562.V ?X_B)
                ?E3 (?X_C EAT.564.V ?X_D)
        :POSTCONDS
                ?P1 (NOT (?X_C (HAVE.V ?X_D)))
                ?P2 (NOT (?X_C HUNGRY.A))
        :EPISODE-RELATIONS
                !W1 (?E1 BEFORE ?E2)
                !W2 (?E2 BEFORE ?E3)
                !W3 (?E1 DURING ?E)
                !W4 (?E2 DURING ?E)
)
```

```
?X_C ← BEES715.SK
?X_B ← NECTAR1.SK
?E2  ← E722.SK
```

```
(BEES715.SK (PLUR BEE.N))

(FLOWERS721.SK ((NN CLOVER.N) (PLUR
     FLOWER.N)))

(NECTAR1.SK SWEET.A)

(NECTAR1.SK NECTAR.N)

(BEES715.SK (PLUR BEE.N))

(FIELD738.SK (PLUR FIELD.N))

((BEES715.SK ((ADV-A (IN.P FLOWERS721.SK))
     FIND.562.V) NECTAR1.SK) ** E722.SK)
```

```
(EPI-SCHEMA ((?X_C GO_FIND_EAT.566.PR ?X_A ?X_B ?X_D) ** ?E)
    :ROLES
        !R1 (?X_A FIELD.N)
        !R2 (?X_B (OF.P (K GRASS.N)))
        !R3 (?X_B SPOT.N)
        !R4 (?X_C COW.N)
        !R5 (?X_D FOOD.N)
        !R6 (?X_D GRASS.N)
    :PRECONDS
        ?I1 (?X_C (AT.P ?L1_2))
        ?I2 (NOT (?X_C (AT.P ?X_A)))
    :STEPS
        ?E1 (?X_C (OUT.ADV ((ADV-A (TO.P ?X_A))
                ((ADV-A (FROM.P ?L1)) GO.563.V))) ?X_A)
        ?E2 (?X_C FIND.562.V ?X_B)
        ?E3 (?X_C EAT.564.V ?X_D)
    :POSTCONDS
        ?P1 (NOT (?X_C (HAVE.V ?X_D)))
        ?P2 (NOT (?X_C HUNGRY.A))
    :EPISODE-RELATIONS
        !W1 (?E1 BEFORE ?E2)
        !W2 (?E2 BEFORE ?E3)
        !W3 (?E1 DURING ?E)
        !W4 (?E2 DURING ?E)
)
```

```
?X_C ← BEES715.SK
?X_B ← NECTAR1.SK
?E2  ← E722.SK
?X_A ← FIELD738.SK
```

```
(BEES715.SK (PLUR BEE.N))

(FLOWERS721.SK ((NN CLOVER.N) (PLUR
     FLOWER.N)))

(NECTAR1.SK SWEET.A)

(NECTAR1.SK NECTAR.N)

(BEES715.SK (PLUR BEE.N))

(FIELD738.SK (PLUR FIELD.N))

((BEES715.SK ((ADV-A (IN.P FLOWERS721.SK))
     FIND.562.V) NECTAR1.SK) ** E722.SK)
```

This formula matched, too.
Not all matched pieces are steps.
Just "seeing a field" can remind us of this cow schema!

```
(EPI-SCHEMA ((BEES715.SK GO_FIND_EAT.861.PR FIELD738.SK NECTAR1.SK ?X_D) ** ?E)
        :ROLES
                !R1    (FIELD738.SK FIELD.N)
                !R2_1 (NECTAR1.SK NECTAR.N)
                !R2_2 (NECTAR1.SK SWEET.A)
                !R2_3 (NECTAR1.SK (OF.P (K GRASS.N)))
                !R2_4 (NECTAR1.SK SPOT.N)
                !R4    (BEES715.SK COW.N)
                !R5    (?X_D FOOD.N)
                !R6    (?X_D GRASS.N)
                !R7    (BEES715.SK (PLUR BEE.N))
                !R8    (FLOWERS721.SK ((NN CLOVER.N) (PLUR FLOWER.N)))
                !R9    (BEES715.SK ANIMAL.N)
        :PRECONDS
                ?I1 (BEES715.SK (AT.P ?L1_2))
                ?I2 (NOT (BEES715.SK (AT.P FIELD738.SK)))
        :STEPS
                ?E1       (BEES715.SK (OUT.ADV ((ADV-A (TO.P FIELD738.SK)) ((ADV-A (FROM.P HOME724.SK)) GO.860.V))) FIELD738.SK)
                E722.SK (BEES715.SK ((ADV-A (IN.P FLOWERS721.SK)) FIND.562.V) NECTAR1.SK)
                ?E3       (BEES715.SK EAT.564.V ?X_D)
        :POSTCONDS
                ?P1  (NOT (BEES715.SK (HAVE.V ?X_D)))
                ?P2 (NOT (BEES715.SK HUNGRY.A))
        :EPISODE-RELATIONS
                !W1  (?E1 BEFORE E722.SK)
                !W2 (E722.SK BEFORE ?E3)
                !W3 (E722.SK (AT-ABOUT NOW178))
        :CERTAINTIES
                !C1 (!R4 CERTAIN-TO-DEGREE (/ 1 2))
                !C2 (!R7 CERTAIN-TO-DEGREE (/ 1 2))
                !C3 (!R9 CERTAIN-TO-DEGREE (/ 2 2))
)
```

```
(EPI-SCHEMA ((BEES715.SK GO_FIND_EAT.861.PR FIELD738.SK NECTAR1.SK ?X_D) ** ?E)
        :ROLES
                !R1     (FIELD738.SK FIELD.N)
                !R2_1 (NECTAR1.SK NECTAR.N)
                !R2_2 (NECTAR1.SK SWEET.A)
                !R2_3 (NECTAR1.SK (OF.P (K GRASS.N)))
                !R2_4 (NECTAR1.SK SPOT.N)
                !R4     (BEES715.SK COW.N)
                !R5     (?X_D FOOD.N)
                !R6     (?X_D GRASS.N)
                !R7     (BEES715.SK (PLUR BEE.N))
                !R8     (FLOWERS721.SK ((NN CLOVER.N) (PLUR FLOWER.N)))
                !R9     (BEES715.SK ANIMAL.N)
        :PRECONDS
                ?I1 (BEES715.SK (AT.P ?L1_2))
                ?I2 (NOT (BEES715.SK (AT.P FIELD738.SK)))
        :STEPS
                ?E1        (BEES715.SK (OUT.ADV ((ADV-A (TO.P FIELD738.SK)) ((ADV-A (FROM.P HOME724.SK)) GO.860.V))) FIELD738.SK)
                E722.SK (BEES715.SK ((ADV-A (IN.P FLOWERS721.SK)) FIND.562.V) NECTAR1.SK)
                ?E3        (BEES715.SK EAT.564.V ?X_D)
        :POSTCONDS
                ?P1  (NOT (BEES715.SK (HAVE.V ?X_D)))
                ?P2 (NOT (BEES715.SK HUNGRY.A))
        :EPISODE-RELATIONS
                !W1  (?E1 BEFORE E722.SK)
                !W2 (E722.SK BEFORE ?E3)
                !W3 (E722.SK (AT-ABOUT NOW178))
        :CERTAINTIES
                !C1 (!R4 CERTAIN-TO-DEGREE (/ 1 2))
                !C2 (!R7 CERTAIN-TO-DEGREE (/ 1 2))
                !C3 (!R9 CERTAIN-TO-DEGREE (/ 2 2))
)
```

```
(EPI-SCHEMA ((BEES715.SK GO_FIND_EAT.861.PR FIELD738.SK NECTAR1.SK ?X_D) ** ?E)
        :ROLES
                !R1     (FIELD738.SK FIELD.N)
                !R2_1 (NECTAR1.SK NECTAR.N)
                !R2_2 (NECTAR1.SK SWEET.A)
                !R2_3 (NECTAR1.SK (OF.P (K GRASS.N)))
                !R2_4 (NECTAR1.SK SPOT.N)
                !R4     (BEES715.SK COW.N)
                !R5     (?X_D FOOD.N)
                !R6     (?X_D GRASS.N)
                !R7     (BEES715.SK (PLUR BEE.N))
                !R8     (FLOWERS721.SK ((NN CLOVER.N) (PLUR FLOWER.N)))
                !R9     (BEES715.SK ANIMAL.N)
        :PRECONDS
                ?I1 (BEES715.SK (AT.P ?L1_2))
                ?I2 (NOT (BEES715.SK (AT.P FIELD738.SK)))
        :STEPS
                ?E1         (BEES715.SK (OUT.ADV ((ADV-A (TO.P FIELD738.SK)) ((ADV-A (FROM.P HOME724.SK)) GO.860.V))) FIELD738.SK)
                E722.SK (BEES715.SK ((ADV-A (IN.P FLOWERS721.SK)) FIND.562.V) NECTAR1.SK)
                ?E3         (BEES715.SK EAT.564.V ?X_D)
        :POSTCONDS
                ?P1  (NOT (BEES715.SK (HAVE.V ?X_D)))
                ?P2 (NOT (BEES715.SK HUNGRY.A))
        :EPISODE-RELATIONS
                !W1  (?E1 BEFORE E722.SK)
                !W2 (E722.SK BEFORE ?E3)
                !W3 (E722.SK (AT-ABOUT NOW178))
        :CERTAINTIES
                !C1 (!R4 CERTAIN-TO-DEGREE (/ 1 2))
                !C2 (!R7 CERTAIN-TO-DEGREE (/ 1 2))
                !C3 (!R9 CERTAIN-TO-DEGREE (/ 2 2))
)
```

WordNet lowest common hypernym of COW.N and BEE.N is ANIMAL.N

New schema is only 50% sure agent is bee or cow, but 100% sure it's an animal.

```
(EPI-SCHEMA ((BEES715.SK GO_FIND_EAT.861.PR FIELD738.SK NECTAR1.SK ?X_D) ** ?E)
        :ROLES
                !R1     (FIELD738.SK FIELD.N)
                !R2_1 (NECTAR1.SK NECTAR.N)
                !R2_2 (NECTAR1.SK SWEET.A)
                !R2_3 (NECTAR1.SK (OF.P (K GRASS.N)))
                !R2_4 (NECTAR1.SK SPOT.N)
                !R4     (BEES715.SK COW.N)
                !R5     (?X_D FOOD.N)
                !R6     (?X_D GRASS.N)
                !R7     (BEES715.SK (PLUR BEE.N))
                !R8     (FLOWERS721.SK ((NN CLOVER.N) (PLUR FLOWER.N)))
                !R9     (BEES715.SK ANIMAL.N)
        :PRECONDS
                ?I1 (BEES715.SK (AT.P ?L1_2))
                ?I2 (NOT (BEES715.SK (AT.P FIELD738.SK)))
        :STEPS
                ?E1        (BEES715.SK (OUT.ADV ((ADV-A (TO.P FIELD738.SK)) ((ADV-A (FROM.P HOME724.SK)) GO.860.V))) FIELD738.SK)
                E722.SK (BEES715.SK ((ADV-A (IN.P FLOWERS721.SK)) FIND.562.V) NECTAR1.SK)
                ?E3        (BEES715.SK EAT.564.V ?X_D)
        :POSTCONDS
                ?P1  (NOT (BEES715.SK (HAVE.V ?X_D)))
                ?P2 (NOT (BEES715.SK HUNGRY.A))
        :EPISODE-RELATIONS
                !W1  (?E1 BEFORE E722.SK)
                !W2 (E722.SK BEFORE ?E3)
                !W3 (E722.SK (AT-ABOUT NOW178))
        :CERTAINTIES
                !C1 (!R4 CERTAIN-TO-DEGREE (/ 1 2))
                !C2 (!R7 CERTAIN-TO-DEGREE (/ 1 2))
                !C3 (!R9 CERTAIN-TO-DEGREE (/ 2 2))
)
```

```
(EPI-SCHEMA ((BEES715.SK GO_FIND_EAT.861.PR FIELD738.SK NECTAR1.SK ?X_D) ** ?E)
        :ROLES
                !R1     (FIELD738.SK FIELD.N)
                !R2_1 (NECTAR1.SK NECTAR.N)
                !R2_2 (NECTAR1.SK SWEET.A)
                !R2_3 (NECTAR1.SK (OF.P (K GRASS.N)))
                !R2_4 (NECTAR1.SK SPOT.N)
                !R4     (BEES715.SK COW.N)
                !R5     (?X_D FOOD.N)
                !R6     (?X_D GRASS.N)
                !R7     (BEES715.SK (PLUR BEE.N))
                !R8     (FLOWERS721.SK ((NN CLOVER.N) (PLUR FLOWER.N)))
                !R9     (BEES715.SK ANIMAL.N)
        :PRECONDS
                ?I1 (BEES715.SK (AT.P ?L1_2))
                ?I2 (NOT (BEES715.SK (AT.P FIELD738.SK)))
        :STEPS
                ?E1       (BEES715.SK (OUT.ADV ((ADV-A (TO.P FIELD738.SK)) ((ADV-A (FROM.P HOME724.SK)) GO.860.V))) FIELD738.SK)
                E722.SK (BEES715.SK ((ADV-A (IN.P FLOWERS721.SK)) FIND.562.V) NECTAR1.SK)
                ?E3       (BEES715.SK EAT.564.V ?X_D)
        :POSTCONDS
                ?P1  (NOT (BEES715.SK (HAVE.V ?X_D)))
                ?P2 (NOT (BEES715.SK HUNGRY.A))
        :EPISODE-RELATIONS
                !W1  (?E1 BEFORE E722.SK)
                !W2 (E722.SK BEFORE ?E3)
                !W3 (E722.SK (AT-ABOUT NOW178))
        :CERTAINTIES
                !C1 (!R4 CERTAIN-TO-DEGREE (/ 1 2))
                !C2 (!R7 CERTAIN-TO-DEGREE (/ 1 2))
                !C3 (!R9 CERTAIN-TO-DEGREE (/ 2 2))
)
```

```
(EPI-SCHEMA ((BEES715.SK GO_FIND_EAT.861.PR FIELD738.SK NECTAR1.SK ?X_D) ** ?E)
        :ROLES
                !R1     (FIELD738.SK FIELD.N)
                !R2_1 (NECTAR1.SK NECTAR.N)
                !R2_2 (NECTAR1.SK SWEET.A)
                !R2_3 (NECTAR1.SK (OF.P (K GRASS.N)))
                !R2_4 (NECTAR1.SK SPOT.N)
                !R4     (BEES715.SK COW.N)
                !R5     (?X_D FOOD.N)
                !R6     (?X_D GRASS.N)
                !R7     (BEES715.SK (PLUR BEE.N))
                !R8     (FLOWERS721.SK ((NN CLOVER.N) (PLUR FLOWER.N)))
                !R9     (BEES715.SK ANIMAL.N)
        :PRECONDS
                ?I1 (BEES715.SK (AT.P ?L1_2))
                ?I2 (NOT (BEES715.SK (AT.P FIELD738.SK)))
        :STEPS
                ?E1        (BEES715.SK (OUT.ADV ((ADV-A (TO.P FIELD738.SK)) ((ADV-A (FROM.P HOME724.SK)) GO.860.V))) FIELD738.SK)
                E722.SK (BEES715.SK ((ADV-A (IN.P FLOWERS721.SK)) FIND.562.V) NECTAR1.SK)
                ?E3        (BEES715.SK EAT.564.V ?X_D)
        :POSTCONDS
                ?P1  (NOT (BEES715.SK (HAVE.V ?X_D)))
                ?P2 (NOT (BEES715.SK HUNGRY.A))
        :EPISODE-RELATIONS
                !W1  (?E1 BEFORE E722.SK)
                !W2 (E722.SK BEFORE ?E3)
                !W3 (E722.SK (AT-ABOUT NOW178))
        :CERTAINTIES
                !C1 (!R4 CERTAIN-TO-DEGREE (/ 1 2))
                !C2 (!R7 CERTAIN-TO-DEGREE (/ 1 2))
                !C3 (!R9 CERTAIN-TO-DEGREE (/ 2 2))
)
```

Only one step is “confirmed” from the story…

```
(EPI-SCHEMA ((BEES715.SK GO_FIND_EAT.861.PR FIELD738.SK NECTAR1.SK ?X_D) ** ?E)
        :ROLES
                !R1     (FIELD738.SK FIELD.N)
                !R2_1 (NECTAR1.SK NECTAR.N)
                !R2_2 (NECTAR1.SK SWEET.A)
                !R2_3 (NECTAR1.SK (OF.P (K GRASS.N)))
                !R2_4 (NECTAR1.SK SPOT.N)
                !R4     (BEES715.SK COW.N)
                !R5     (?X_D FOOD.N)
                !R6     (?X_D GRASS.N)
                !R7     (BEES715.SK (PLUR BEE.N))
                !R8     (FLOWERS721.SK ((NN CLOVER.N) (PLUR FLOWER.N)))
                !R9     (BEES715.SK ANIMAL.N)
        :PRECONDS
                ?I1 (BEES715.SK (AT.P ?L1_2))
                ?I2 (NOT (BEES715.SK (AT.P FIELD738.SK)))
        :STEPS
                ?E1        (BEES715.SK (OUT.ADV ((ADV-A (TO.P FIELD738.SK)) ((ADV-A (FROM.P HOME724.SK)) GO.860.V))) FIELD738.SK)
                E722.SK (BEES715.SK ((ADV-A (IN.P FLOWERS721.SK)) FIND.562.V) NECTAR1.SK)
                ?E3        (BEES715.SK EAT.564.V ?X_D)
        :POSTCONDS
                ?P1  (NOT (BEES715.SK (HAVE.V ?X_D)))
                ?P2 (NOT (BEES715.SK HUNGRY.A))
        :EPISODE-RELATIONS
                !W1  (?E1 BEFORE E722.SK)
                !W2 (E722.SK BEFORE ?E3)
                !W3 (E722.SK (AT-ABOUT NOW178))
        :CERTAINTIES
                !C1 (!R4 CERTAIN-TO-DEGREE (/ 1 2))
                !C2 (!R7 CERTAIN-TO-DEGREE (/ 1 2))
                !C3 (!R9 CERTAIN-TO-DEGREE (/ 2 2))
)
```

But instead of “unconfirmed steps”,
couldn’t we call these “predictions”?

```
(EPI-SCHEMA ((BEES715.SK GO_FIND_EAT.861.PR FIELD738.SK NECTAR1.SK ?X_D) ** ?E)
        :ROLES
                !R1    (FIELD738.SK FIELD.N)
                !R2_1 (NECTAR1.SK NECTAR.N)
                !R2_2 (NECTAR1.SK SWEET.A)
                !R2_3 (NECTAR1.SK (OF.P (K GRASS.N)))
                !R2_4 (NECTAR1.SK SPOT.N)
                !R4    (BEES715.SK COW.N)
                !R5    (?X_D FOOD.N)
                !R6    (?X_D GRASS.N)
                !R7    (BEES715.SK (PLUR BEE.N))
                !R8    (FLOWERS721.SK ((NN CLOVER.N) (PLUR FLOWER.N)))
                !R9    (BEES715.SK ANIMAL.N)
        :PRECONDS
                ?I1 (BEES715.SK (AT.P ?L1_2))
                ?I2 (NOT (BEES715.SK (AT.P FIELD738.SK)))
        :STEPS
                ?E1       (BEES715.SK (OUT.ADV ((ADV-A (TO.P FIELD738.SK)) ((ADV-A (FROM.P HOME724.SK)) GO.860.V))) FIELD738.SK)
                E722.SK (BEES715.SK ((ADV-A (IN.P FLOWERS721.SK)) FIND.562.V) NECTAR1.SK)
                ?E3       (BEES715.SK EAT.564.V ?X_D)
        :POSTCONDS
                ?P1  (NOT (BEES715.SK (HAVE.V ?X_D)))
                ?P2 (NOT (BEES715.SK HUNGRY.A))
        :EPISODE-RELATIONS
                !W1  (?E1 BEFORE E722.SK)
                !W2 (E722.SK BEFORE ?E3)
                !W3 (E722.SK (AT-ABOUT NOW178))
        :CERTAINTIES
                !C1 (!R4 CERTAIN-TO-DEGREE (/ 1 2))
                !C2 (!R7 CERTAIN-TO-DEGREE (/ 1 2))
                !C3 (!R9 CERTAIN-TO-DEGREE (/ 2 2))
)
```

"I observe that some bees found some sweet nectar in some clover flowers..."

```
(EPI-SCHEMA ((BEES715.SK GO_FIND_EAT.861.PR FIELD738.SK NECTAR1.SK ?X_D) ** ?E)
        :ROLES
                !R1     (FIELD738.SK FIELD.N)
                !R2_1 (NECTAR1.SK NECTAR.N)
                !R2_2 (NECTAR1.SK SWEET.A)
                !R2_3 (NECTAR1.SK (OF.P (K GRASS.N)))
                !R2_4 (NECTAR1.SK SPOT.N)
                !R4     (BEES715.SK COW.N)
                !R5     (?X_D FOOD.N)
                !R6     (?X_D GRASS.N)
                !R7     (BEES715.SK (PLUR BEE.N))
                !R8     (FLOWERS721.SK ((NN CLOVER.N) (PLUR FLOWER.N)))
                !R9     (BEES715.SK ANIMAL.N)
        :PRECONDS
                ?I1 (BEES715.SK (AT.P ?L1_2))
                ?I2 (NOT (BEES715.SK (AT.P FIELD738.SK)))
        :STEPS
                ?E1        (BEES715.SK (OUT.ADV ((ADV-A (TO.P FIELD738.SK)) ((ADV-A (FROM.P HOME724.SK)) GO.860.V))) FIELD738.SK)
                E722.SK (BEES715.SK ((ADV-A (IN.P FLOWERS721.SK)) FIND.562.V) NECTAR1.SK)
                ?E3        (BEES715.SK EAT.564.V ?X_D)
        :POSTCONDS
                ?P1  (NOT (BEES715.SK (HAVE.V ?X_D)))
                ?P2 (NOT (BEES715.SK HUNGRY.A))
        :EPISODE-RELATIONS
                !W1  (?E1 BEFORE E722.SK)
                !W2 (E722.SK BEFORE ?E3)
                !W3 (E722.SK (AT-ABOUT NOW178))
        :CERTAINTIES
                !C1 (!R4 CERTAIN-TO-DEGREE (/ 1 2))
                !C2 (!R7 CERTAIN-TO-DEGREE (/ 1 2))
                !C3 (!R9 CERTAIN-TO-DEGREE (/ 2 2))
)
```

“Maybe they left their home to go to a field first?”

```
(EPI-SCHEMA ((BEES715.SK GO_FIND_EAT.861.PR FIELD738.SK NECTAR1.SK ?X_D) ** ?E)
        :ROLES
                !R1     (FIELD738.SK FIELD.N)
                !R2_1 (NECTAR1.SK NECTAR.N)
                !R2_2 (NECTAR1.SK SWEET.A)
                !R2_3 (NECTAR1.SK (OF.P (K GRASS.N)))
                !R2_4 (NECTAR1.SK SPOT.N)
                !R4     (BEES715.SK COW.N)
                !R5     (?X_D FOOD.N)
                !R6     (?X_D GRASS.N)
                !R7     (BEES715.SK (PLUR BEE.N))
                !R8     (FLOWERS721.SK ((NN CLOVER.N) (PLUR FLOWER.N)))
                !R9     (BEES715.SK ANIMAL.N)
        :PRECONDS
                ?I1 (BEES715.SK (AT.P ?L1_2))
                ?I2 (NOT (BEES715.SK (AT.P FIELD738.SK)))
        :STEPS
                ?E1        (BEES715.SK (OUT.ADV ((ADV-A (TO.P FIELD738.SK)) ((ADV-A (FROM.P HOME724.SK)) GO.860.V))) FIELD738.SK)
                E722.SK (BEES715.SK ((ADV-A (IN.P FLOWERS721.SK)) FIND.562.V) NECTAR1.SK)
                ?E3        (BEES715.SK EAT.564.V ?X_D)
        :POSTCONDS
                ?P1  (NOT (BEES715.SK (HAVE.V ?X_D)))
                ?P2 (NOT (BEES715.SK HUNGRY.A))
        :EPISODE-RELATIONS
                !W1  (?E1 BEFORE E722.SK)
                !W2 (E722.SK BEFORE ?E3)
                !W3 (E722.SK (AT-ABOUT NOW178))
        :CERTAINTIES
                !C1 (!R4 CERTAIN-TO-DEGREE (/ 1 2))
                !C2 (!R7 CERTAIN-TO-DEGREE (/ 1 2))
                !C3 (!R9 CERTAIN-TO-DEGREE (/ 2 2))
)
```

“And maybe they ate some grass afterward?”

```
(EPI-SCHEMA ((BEES715.SK GO_FIND_EAT.861.PR FIELD738.SK NECTAR1.SK ?X_D) ** ?E)
        :ROLES
                !R1     (FIELD738.SK FIELD.N)
                !R2_1 (NECTAR1.SK NECTAR.N)
                !R2_2 (NECTAR1.SK SWEET.A)
                !R2_3 (NECTAR1.SK (OF.P (K GRASS.N)))
                !R2_4 (NECTAR1.SK SPOT.N)
                !R4     (BEES715.SK COW.N)
                !R5     (?X_D FOOD.N)
                !R6     (?X_D GRASS.N)
                !R7     (BEES715.SK (PLUR BEE.N))
                !R8     (FLOWERS721.SK ((NN CLOVER.N) (PLUR FLOWER.N)))
                !R9     (BEES715.SK ANIMAL.N)
        :PRECONDS
                ?I1 (BEES715.SK (AT.P ?L1_2))
                ?I2 (NOT (BEES715.SK (AT.P FIELD738.SK)))
        :STEPS
                ?E1        (BEES715.SK (OUT.ADV ((ADV-A (TO.P FIELD738.SK)) ((ADV-A (FROM.P HOME724.SK)) GO.860.V))) FIELD738.SK)
                E722.SK (BEES715.SK ((ADV-A (IN.P FLOWERS721.SK)) FIND.562.V) NECTAR1.SK)
                ?E3        (BEES715.SK EAT.564.V ?X_D)
        :POSTCONDS
                ?P1  (NOT (BEES715.SK (HAVE.V ?X_D)))
                ?P2 (NOT (BEES715.SK HUNGRY.A))
        :EPISODE-RELATIONS
                !W1  (?E1 BEFORE E722.SK)
                !W2 (E722.SK BEFORE ?E3)
                !W3 (E722.SK (AT-ABOUT NOW178))
        :CERTAINTIES
                !C1 (!R4 CERTAIN-TO-DEGREE (/ 1 2))
                !C2 (!R7 CERTAIN-TO-DEGREE (/ 1 2))
                !C3 (!R9 CERTAIN-TO-DEGREE (/ 2 2))
)
```

(Note: that prediction is weaker, because it still has an unmatched variable inside it.)

Evaluation

We have a corpus of 561 simple children’s stories taken from ROCstories (Mostafazadeh et al., 2016) and a children’s “first reader” collection.

We will split these stories (and any more we acquire) into development and test sets. (The former to develop new protoschemas & our matching code.)

The test set will be used to learn as many schemas as possible.

Evaluation

Once the schemas are learned, we will evaluate their usefulness for story understanding by rating the appropriateness of their predictions.

We will use two rating methods: human judges and an automated narrative cloze task.

Human Evaluation

Our schemas will generate three kinds of prediction:

1. Temporal predictions (What happened before that? What happened after that?)
2. Attribute predictions (“He ate a cocoanut.” => “He was a monkey.”)
3. Motivational predictions (“He climbed the cocoanut tree because he was hungry.”)

Human Evaluation

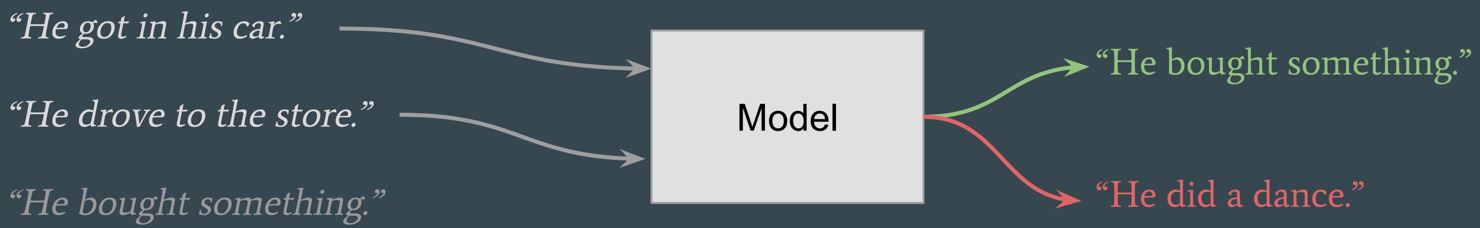
We’ll render the Episodic Logic predictions to English.

Untrained human judges (e.g. turkers) will rate the likelihood and novelty of the predictions.

Automated Narrative Cloze Evaluation

Chambers & Jurafsky (2008) introduced the “narrative cloze” test for evaluating their scripts.

It rated a model’s ability to predict “held-out” events from a sequence.


“He got in his car.”

“He drove to the store.”

“He bought something.”

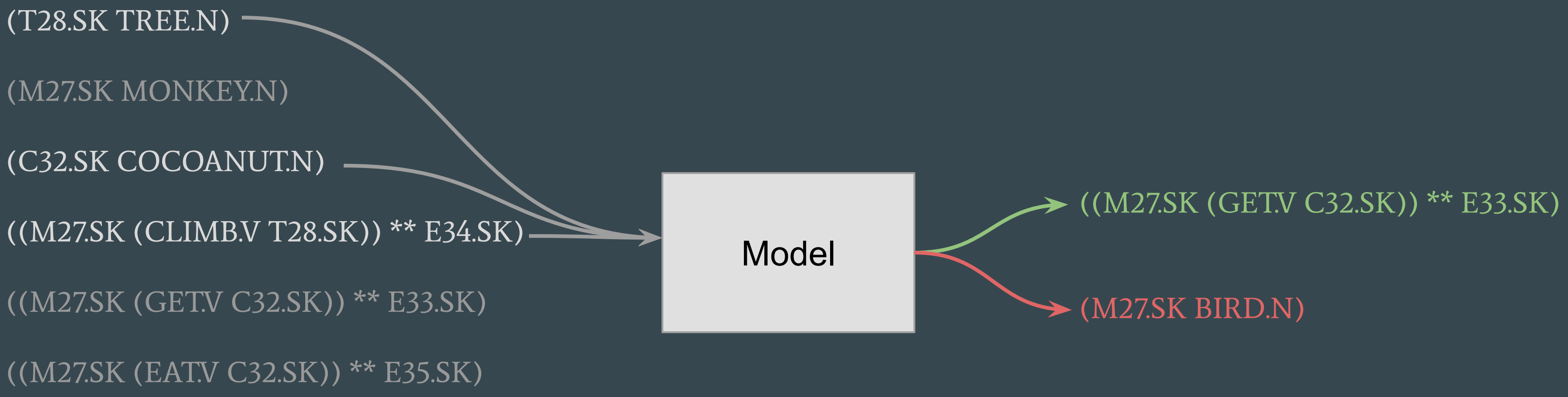
Model

“He bought something.”

“He did a dance.”

Automated Narrative Cloze Evaluation

We can modify this to allow the schemas to predict held-out EL formulas.



(T28.SK TREE.N)

(M27.SK MONKEY.N)

(C32.SK COCOANUT.N)

((M27.SK (CLIMB.V T28.SK)) ** E34.SK)

((M27.SK (GET.V C32.SK)) ** E33.SK)

((M27.SK (EAT.V C32.SK)) ** E35.SK)

Model

((M27.SK (GET.V C32.SK)) ** E33.SK)

(M27.SK BIRD.N)

Thank you!

Thank you!

Thank you!

References

Marvin Minsky. A Framework for Representing Knowledge. Tech. rep. Cambridge, MA, USA, 1974.

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.

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

David E. Rumelhart and Donald A. Norman.. Accretion, tuning and restructuring: Three modes of learning.. Tech. rep. CALIFORNIA UNIV SAN DIEGO LA JOLLA CENTER FOR HUMAN INFORMATION PROCESSING, 1976.

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

Michael Lebowitz. “Generalization and Memory in an Integrated Understanding System.” AAI8109800. PhD thesis. New Haven, CT, USA, 1980

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.

Nathanael Chambers and Dan Jurafsky. “Unsupervised learning of narrative event chains.” In: Proceedings of ACL-08: HLT (2008), pp. 789–797.

References

Karl Pichotta and Raymond J Mooney. “Learning statistical scripts with LSTM recurrent neural networks.” In: Thirtieth AAAI Conference on Artificial Intelligence. 2016.

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. Iwanska and Stuart C. ´ Shapiro, editors, Natural Language Processing and Knowledge Representation, pages 111–174. MIT Press, Cambridge, MA, USA.

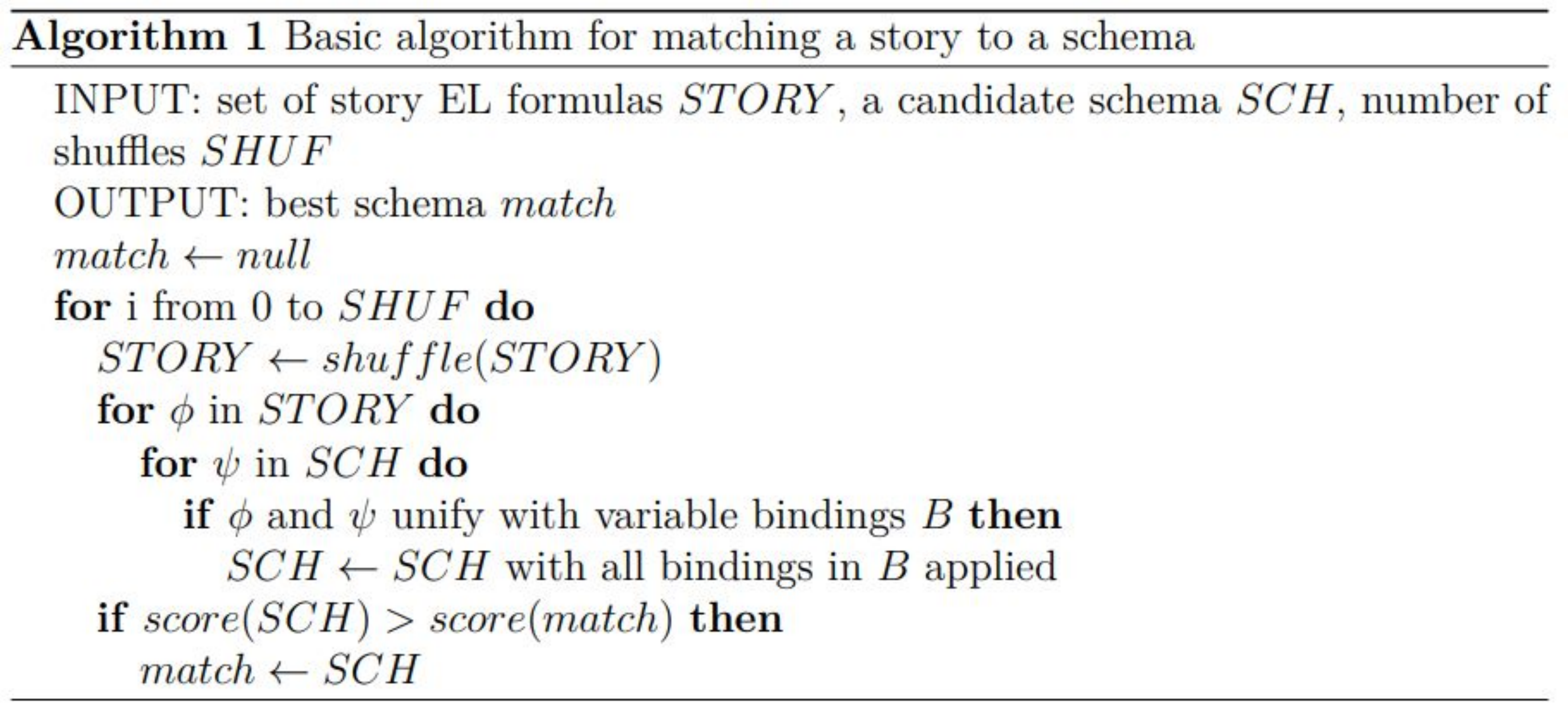
Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, Advances in Neural Information Processing Systems 26, pages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

Nasrin Mostafazadeh et al. “A corpus and evaluation framework for deeper understanding of commonsense stories.” In: arXiv preprint arXiv:1604.01696 (2016).

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

Gene Kim et al. “Generating Discourse Inferences from Unscoped Episodic Logical Formulas.” In: Proceedings of the First International Workshop on Designing Meaning Representations. 2019, pp. 56–65.

Matching Algorithm



Algorithm 1 Basic algorithm for matching a story to a schema

```
INPUT: set of story EL formulas $STORY$, a candidate schema $SCH$, number of shuffles $SHUF$
OUTPUT: best schema $match$
$match \leftarrow null$
for i from 0 to $SHUF$ do
    $STORY \leftarrow shuffle(STORY)$
    for $\phi$ in $STORY$ do
        for $\psi$ in $SCH$ do
            if $\phi$ and $\psi$ unify with variable bindings $B$ then
                $SCH \leftarrow SCH$ with all bindings in $B$ applied
    if $score(SCH) > score(match)$ then
        $match \leftarrow SCH$
```