Automaton-Based Task Knowledge from Generative Models

Ufuk Topcu

The University of Texas at Austin

u-t-autonomous.info

autonomous
SYSTEMS GROUP


Chat GPT: Friend or Foe?
Melissa Heikkilä, MIT Technology Review
Gary Marcus, New York University
AI UK


UK

DALL·E History Collections

Edit the detailed description

Surprise me Upload

John Oliver looking confused and saying "what are you talking about?" in a speech bubble.

Generate

I think…

Generative models are neither foes nor friends.

They are emerging tools that deserve attention.

Automaton-Based Representations of Task Knowledge from Generative Language Models

Yunhao Yang, Jean-Raphaël Gaglione, Cyrus Neary, Ufuk Topcu
The University of Texas at Austin, USA
{yunhaoyang234, jr.gaglione, cneary, utopcu}@utexas.edu

Abstract

Automaton-based representations of task knowledge play an important role in control and planning for sequential decision-making problems. However, obtaining the high-level task knowledge required to build such automata is often difficult. Meanwhile, large-scale *generative language models* (GLMs) can automatically generate relevant task knowledge. However, the textual outputs from GLMs cannot be formally verified or used for sequential decision-making. We propose a novel algorithm named `GLM2FSA`, which constructs a *finite state automaton* (FSA) encoding high-level task knowledge from a brief natural-language description of the task goal. `GLM2FSA` first sends queries to a GLM to extract task knowledge in textual form, and then it builds an FSA to represent this text-based knowledge. The proposed algorithm thus fills the gap between natural-language task descriptions and automaton-based representations, and the constructed FSAs can be formally verified against user-defined specifications. We accordingly propose a method to iteratively refine the queries to the GLM based on the outcomes, e.g., counter-examples, from verification. We demonstrate `GLM2FSA`'s ability to build and refine automaton-based representations of everyday tasks (e.g., crossing a road or making a phone call), and also of tasks that require highly-specialized knowledge (e.g., executing secure multi-party computation).

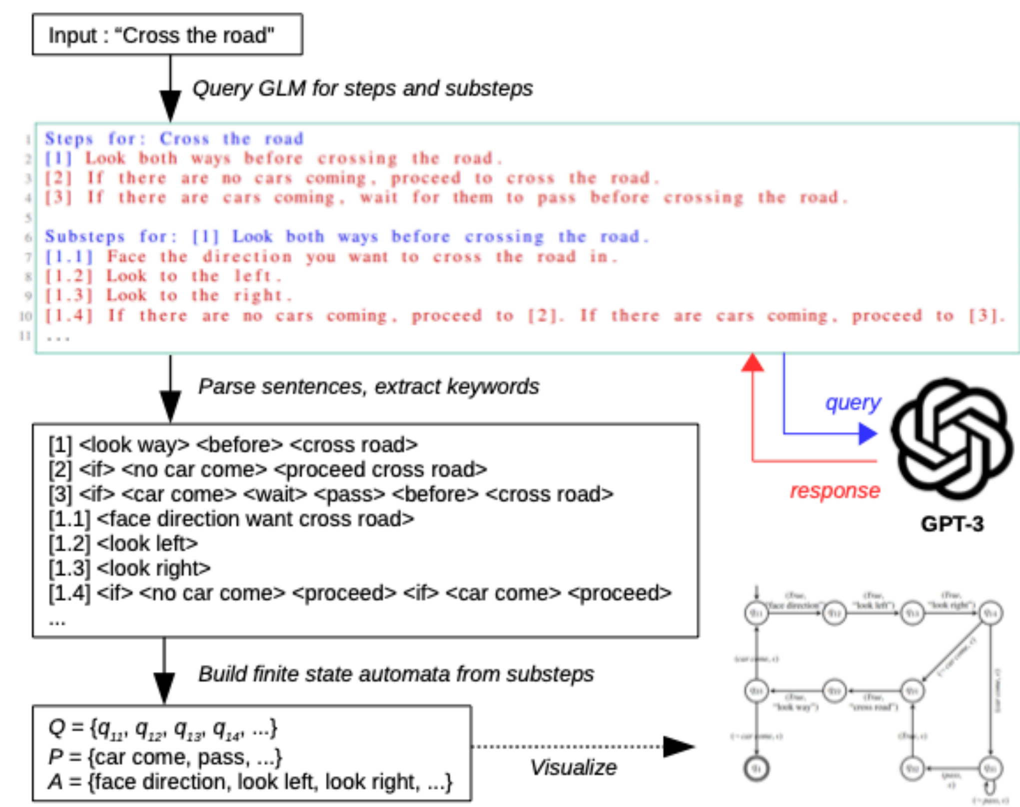


Figure 1: Demonstration of `GLM2FSA` in a real example (the output FSA is presented in Figure 5).

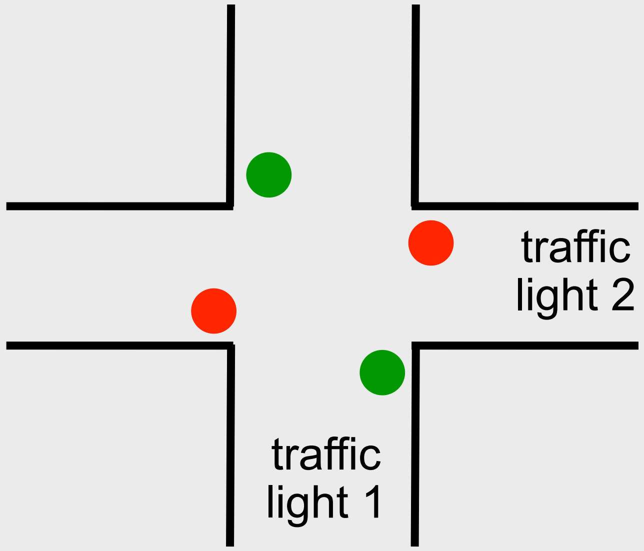
1 Introduction

be constructed in the first place. Even in cases in which an oracle exists, either the learning algorithm or the oracle requires prior information, such as the set of possible actions available to the agent and the set of environmental responses.

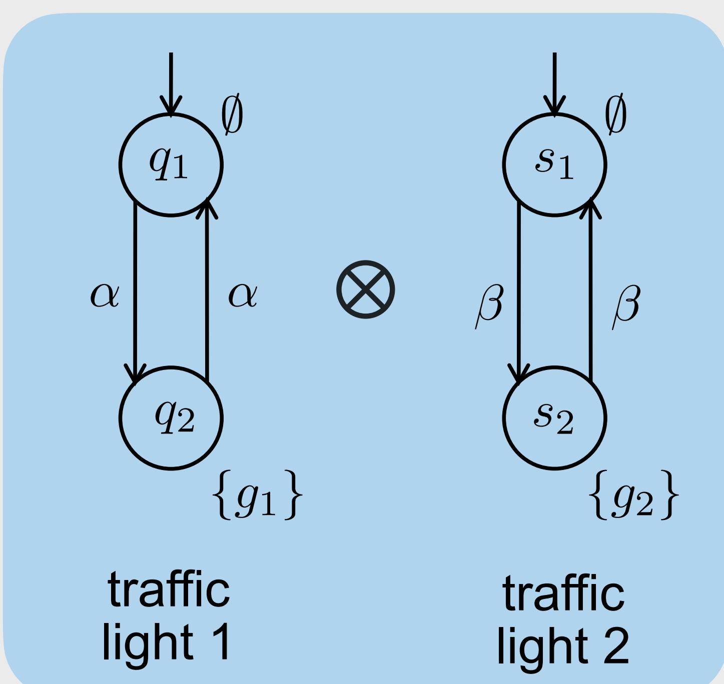
$$\text{“}\mathscr{M} \otimes \mathscr{C} \models \Phi\text{”}$$

model, environment assumptions, controller, system requirements, …

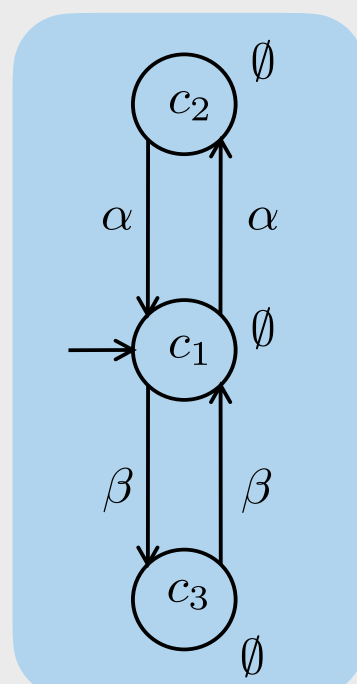
- Model checking, planning,…
- Reactive synthesis, games on graphs, …
- Probabilistic verification and synthesis
- Reinforcement learning



$$\mathscr{M} \otimes \mathscr{C} \models \Phi$$

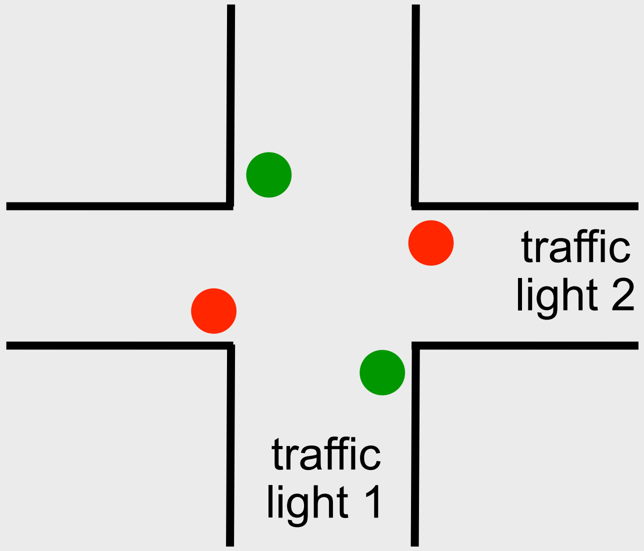


$\otimes$

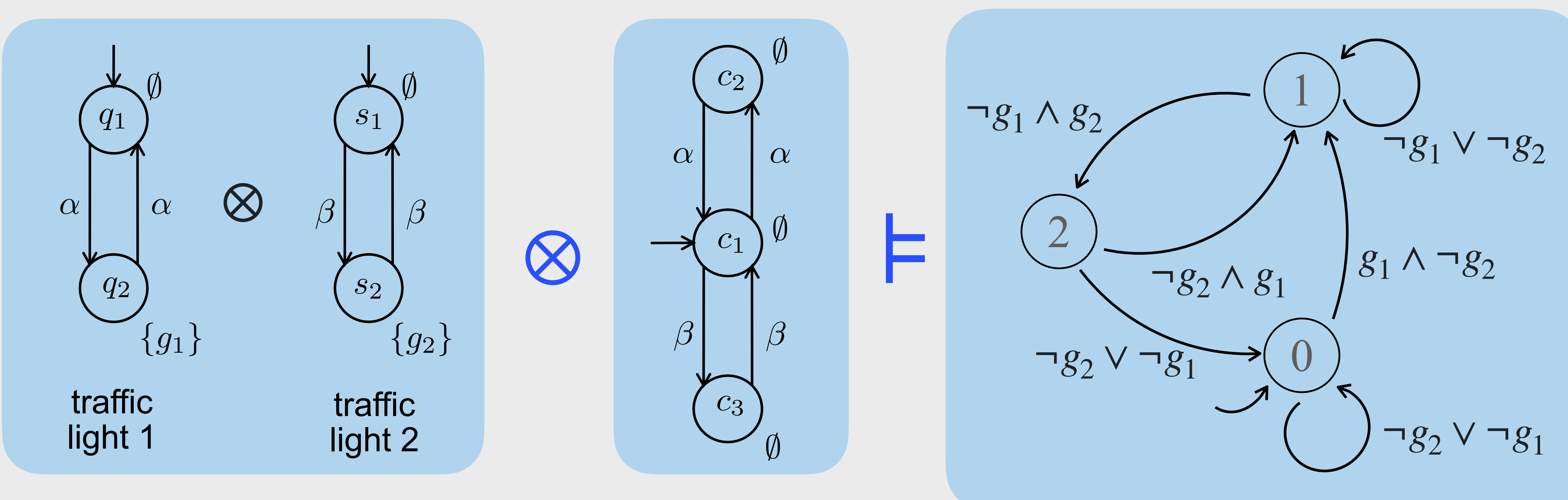


$\models$

$$\Box\Diamond g_1 \wedge \Box\Diamond g_2 \wedge \Box\left(\neg g_1 \vee \neg g_2\right)$$



$$\mathscr{M} \otimes \mathscr{C} \models \Phi$$



Where do the automata come from?

Capturing task knowledge in automata is not straightforward

Automata learning is an alternative but…
- May require too many queries to a human
- Building automated oracles is not easy

Complete prior information, e.g., the set of possible actions and environment responses, may not be available

Humans are heavily involved in the often-overlooked "preprocessing"

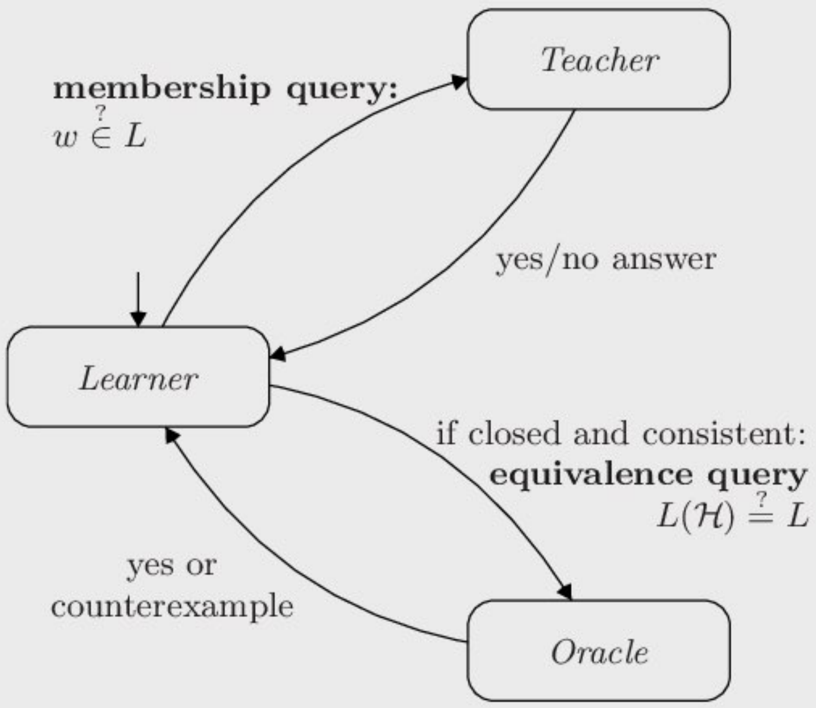


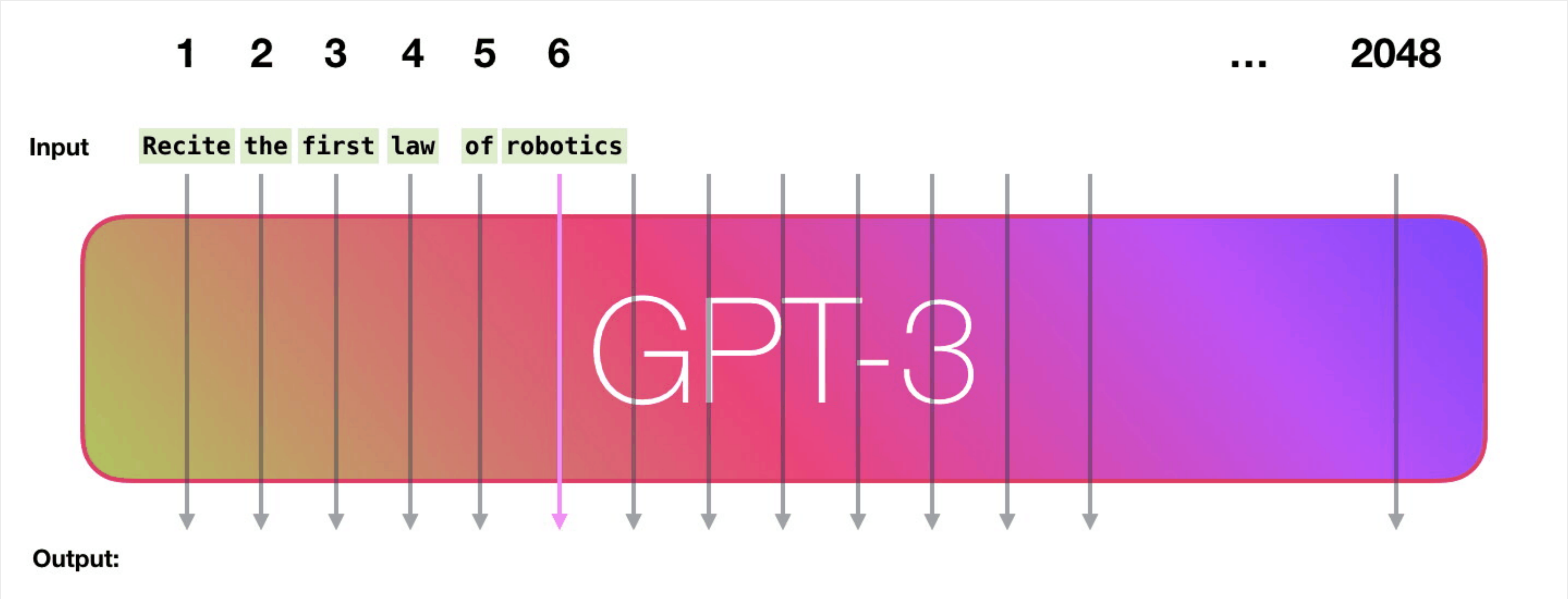
Figure from Bollig, et al., IJCAI, 2009

Hypotheses

Generative language models can help distill task knowledge into automaton-based representations.

Hypotheses

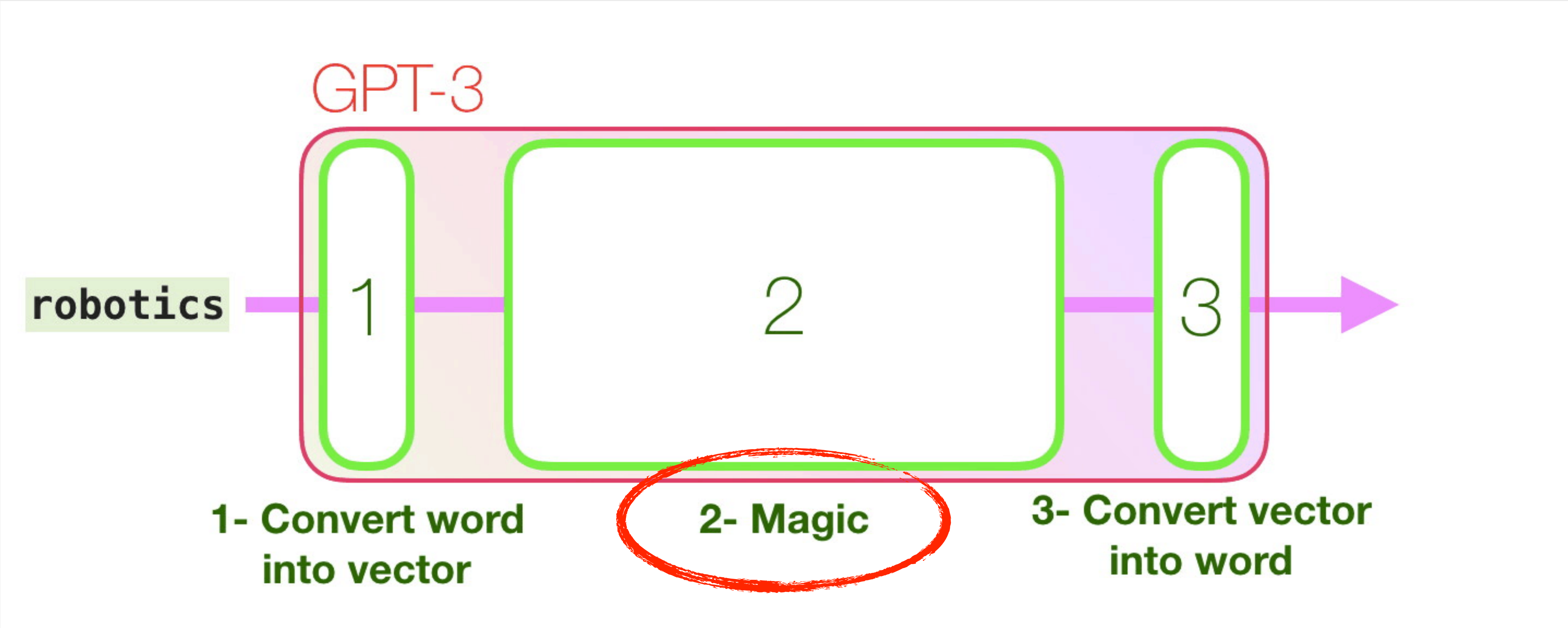
Generative language models can help distill task knowledge into automaton-based representations.



https://jalammar.github.io/how-gpt3-works-visualizations-animations/

Hypotheses

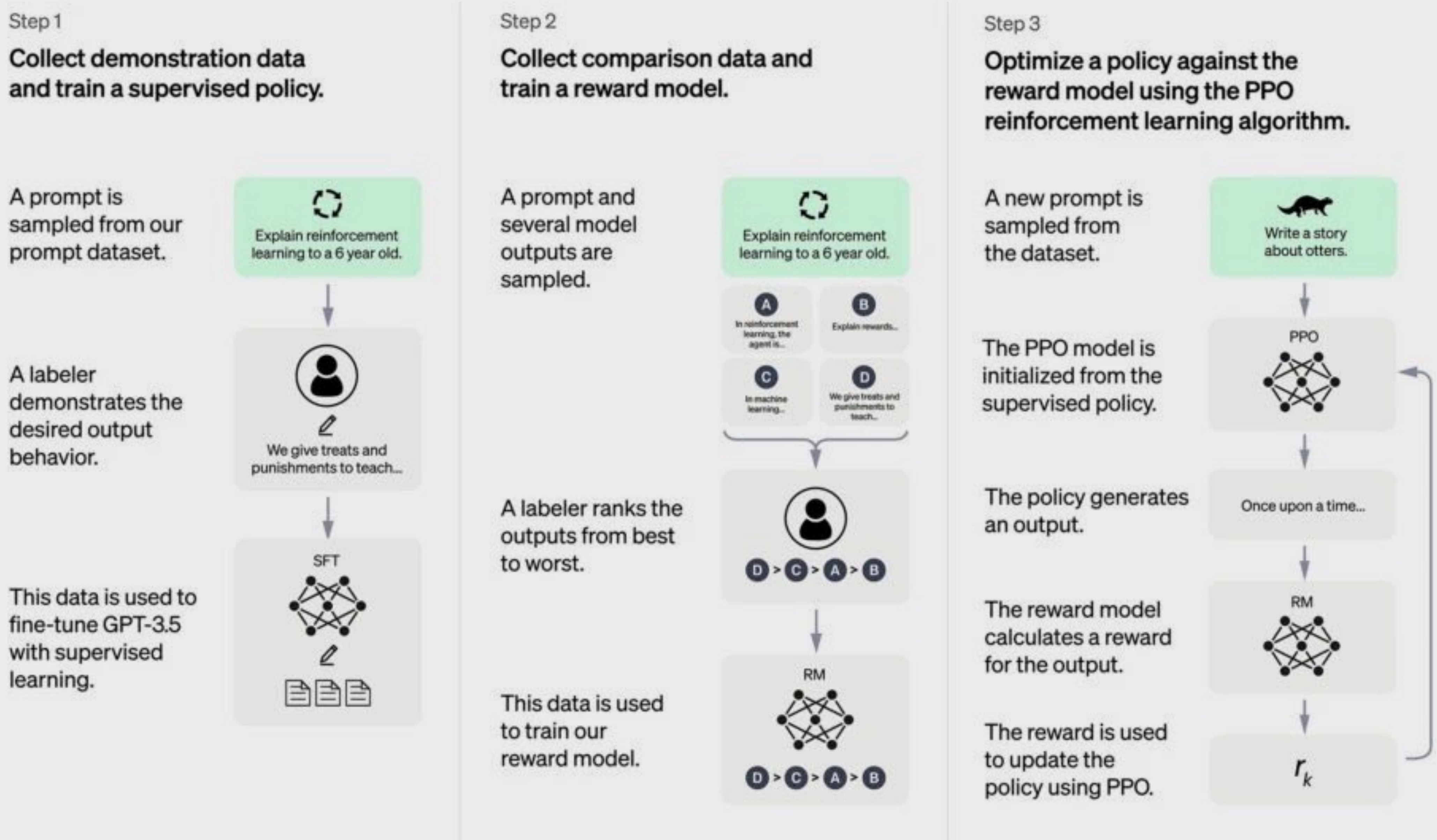
Generative language models can help distill task knowledge into automaton-based representations.



https://jalammar.github.io/how-gpt3-works-visualizations-animations/

Hypotheses

Generative language models can help distill task knowledge into automaton-based representations.



OpenAI

Why to connect generative models to automata?

The outputs of generative language models are not compatible downstream sequential decision-making processes

Distilling task-related knowledge in automata will help integrate into verification, synthesis, or reinforcement learning

- Interrogate the outputs of generative models
- Refine the distilled knowledge
- Integrate additional knowledge available from independent sources

Speculation: Maybe help improve generative models themselves

How to connect generative models to automata?

(GLM2FAS: Generative Language Model to Finite-State Automaton)

Input : “Cross the road”

Query GLM for steps and substeps

```
Steps for: Cross the road
[1] Look both ways before crossing the road.
[2] If there are no cars coming, proceed to cross the road.
[3] If there are cars coming, wait for them to pass before crossing the road.

Substeps for: [1] Look both ways before crossing the road.
[1.1] Face the direction you want to cross the road in.
[1.2] Look to the left.
[1.3] Look to the right.
[1.4] If there are no cars coming, proceed to [2]. If there are cars coming, proceed to [3].
...
```

query

response

GPT-3

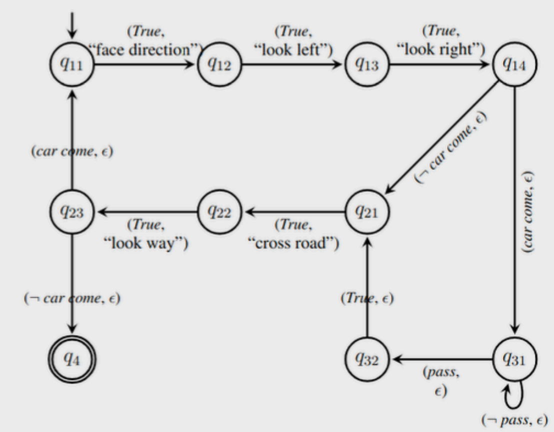
Parse sentences, extract keywords

[1] <look way> <before> <cross road>
[2] <if> <no car come> <proceed cross road>
[3] <if> <car come> <wait> <pass> <before> <cross road>
[1.1] <face direction want cross road>
[1.2] <look left>
[1.3] <look right>
[1.4] <if> <no car come> <proceed> <if> <car come> <proceed>
...

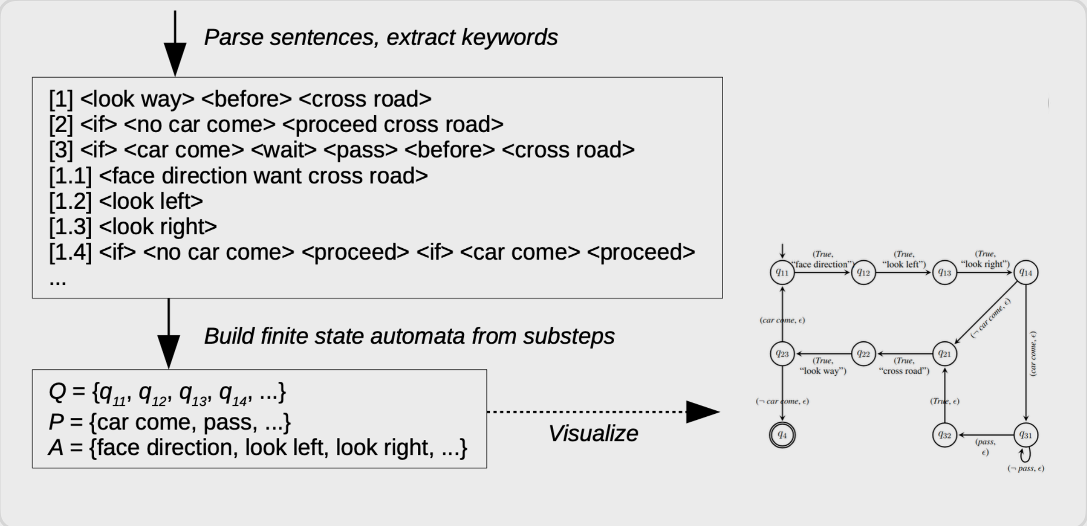
Build finite state automata from substeps

$Q = \{q_{11}, q_{12}, q_{13}, q_{14}, \ldots\}$
$P = \{$car come, pass, ...$\}$
$A = \{$face direction, look left, look right, ...$\}$

Visualize

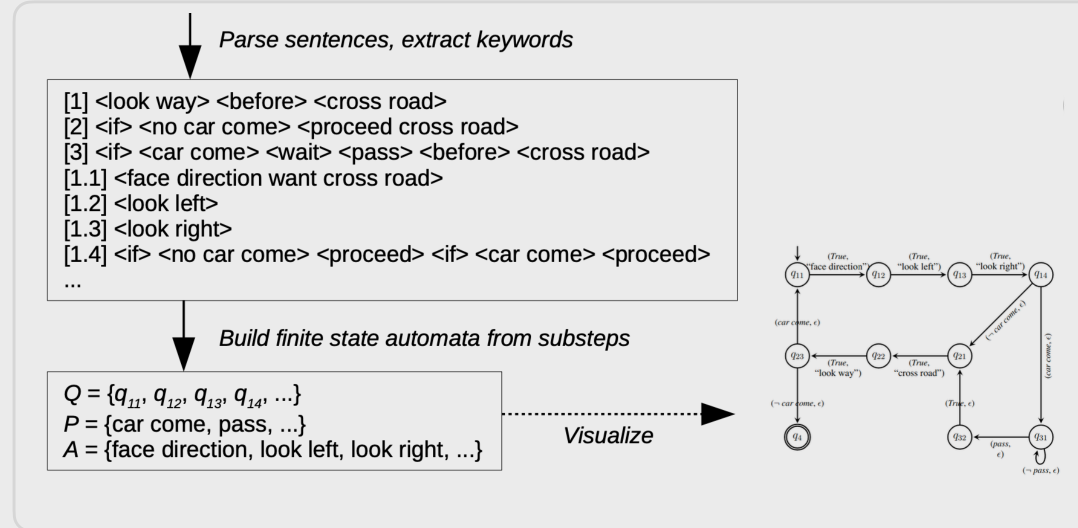


Semantic parsing and automaton construction



Category	Grammar	
Default-Transition	VP^A	<look left>
Direct-Transition	$VP^A \longleftarrow VP^A$ [j]	<proceed> <[2]>
Conditional-Transition (if)	**if** VP^C, VP^A VP^A **if** VP^C	<if> <no car come>, <cross road>
Conditional-Transition (if else)	**if** VP^C, $VP^A{}_1$. **if** $\neg$ VP^C, $VP^A{}_2$ **if** VP^C $VP^A{}_1$ **else** $VP^A{}_2$ $VP^A{}_1$ **if** VP^C, **else** $VP^A{}_2$	<if> <no car come> <proceed [2]>. <if> <car come> <proceed [3]>
Self-Transition	$VP^A \longleftarrow$ **wait** VP^C VP^A $VP^A \longleftarrow VP^A$ **after** VP^C $VP^A \longleftarrow VP^A$ **until** VP^C	<wait> <(car) pass> <cross road> <cross road> <after> <car pass> <stay> <until> <car pass>

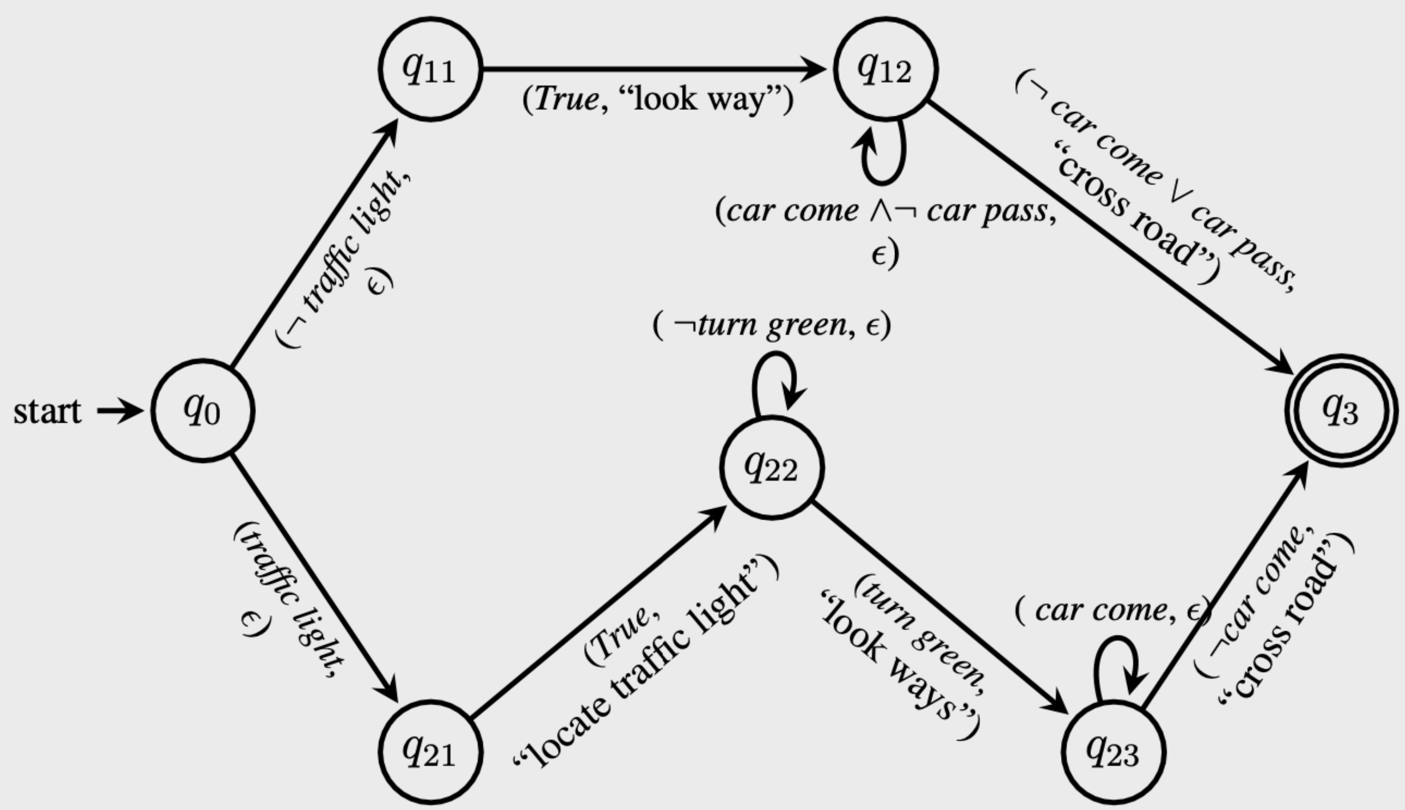
Semantic parsing and automaton construction



Category	Grammar	Transition Rule
Default-Transition	VP^A	$q_i \xrightarrow{(True,\ VP^A)} q_{i+1}$
Direct-Transition	$VP^A \longleftarrow VP^A$ [j]	$q_i \xrightarrow{(True,\ \epsilon)} q_j$
Conditional-Transition (if)	**if** VP^C, VP^A VP^A **if** VP^C	$q_i \xrightarrow{(\neg VP^C,\ \epsilon)} q_i$, $q_i \xrightarrow{(VP^C,\ VP^A)} q_j$
Conditional-Transition (if else)	**if** VP^C, $VP^A{}_1$. **if** $\neg$ VP^C, $VP^A{}_2$ **if** VP^C $VP^A{}_1$ **else** $VP^A{}_2$ $VP^A{}_1$ **if** VP^C, **else** $VP^A{}_2$	$q_k \xleftarrow{(\neg VP^C,\ VP^A{}_2)} q_i \xrightarrow{(VP^C,\ VP^A{}_1)} q_j$
Self-Transition	$VP^A \longleftarrow$ **wait** VP^C VP^A $VP^A \longleftarrow VP^A$ **after** VP^C	$q_i \xrightarrow{(\neg VP^C,\ \epsilon)} q_i$, $q_i \xrightarrow{(VP^C,\ VP^A)} q_{i+1}$
	$VP^A \longleftarrow VP^A$ **until** VP^C	$q_i \xrightarrow{(\neg VP^C,\ VP^A)} q_i$, $q_i \xrightarrow{(VP^C,\ \epsilon)} q_{i+1}$

“Cross the road”

```
Steps for: Cross the road
[1] Look both ways before crossing the road.
[2] If there are no cars coming, proceed to
    cross the road.
[3] If there are cars coming, wait for them to
     pass before crossing the road.
```



The set of atomic propositions:

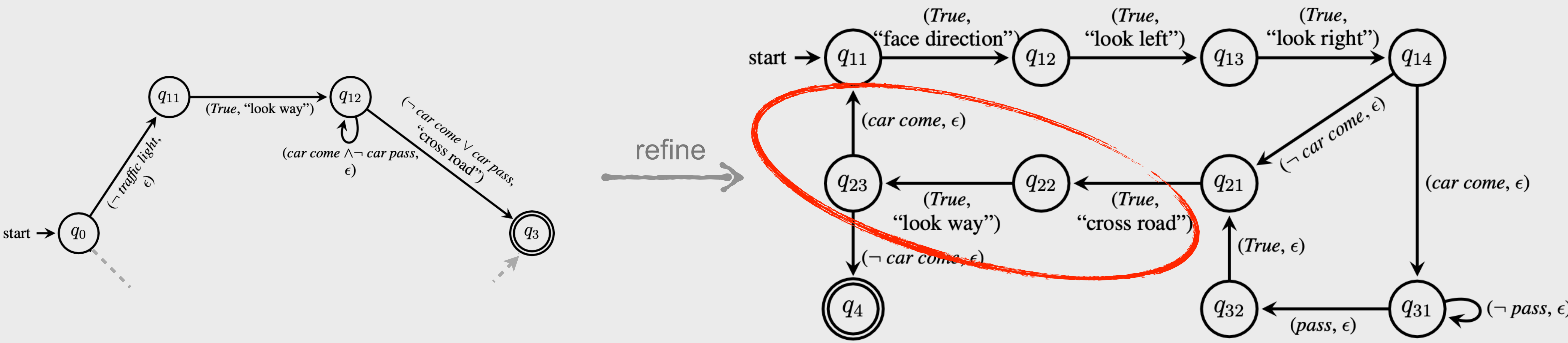
$$\{car\ come, car\ pass, \ldots \ldots turn\ green, traffic\ light\}$$

The set of output symbols:

$$\{\text{"look way"}, \text{"cross road"}, \ldots \ldots \text{"locate traffic light"}, \epsilon\}$$

```
Steps for: Cross the road at the traffic light
[1] Locate the traffic light.
[2] Wait for the traffic light to turn green.
[3] Look both ways before crossing the road.
[4] Cross the road if no cars are coming.
```

More about “Cross the road”



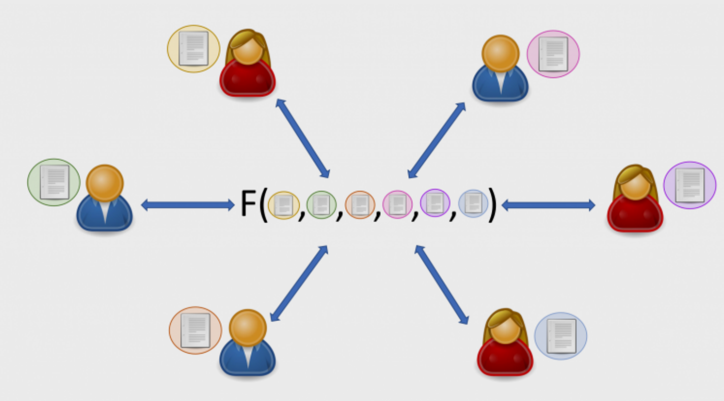
Substeps for: [1] Look both ways before crossing the road.
[1.1] Face the direction you want to cross the road in.
[1.2] Look to the left.
[1.3] Look to the right.
[1.4] If there are no cars coming, go to [2]. If there are cars coming, go to [3].

Substeps for: [2] If there are no cars coming, proceed to cross the road.
[2.1] Walk across the road.
[2.2] Once you have reached the other side, look both ways again to make sure no cars are coming.
[2.3] If there are no cars coming, proceed to [4]. If there are cars coming, back to [1].

Substeps for: [3] If there are cars coming, wait for them to pass before crossing the road.
[3.1] Wait for the cars to pass.
[3.2] Once the cars have passed, back to [2].

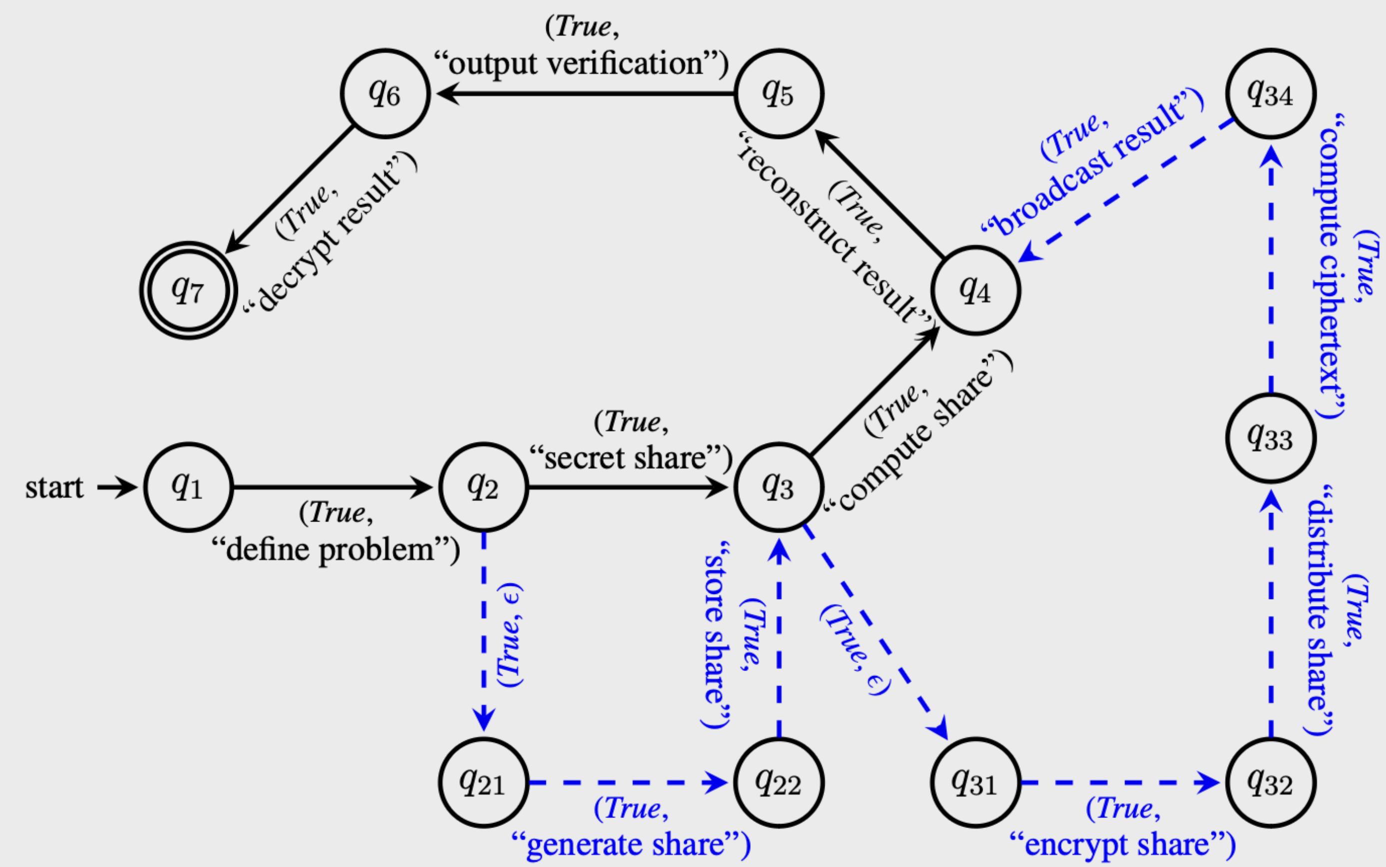
The set of output symbols: {"look way", "face direction", "cross road", "look right", "look left", ϵ}

A less obvious example: Secure multi-party computation



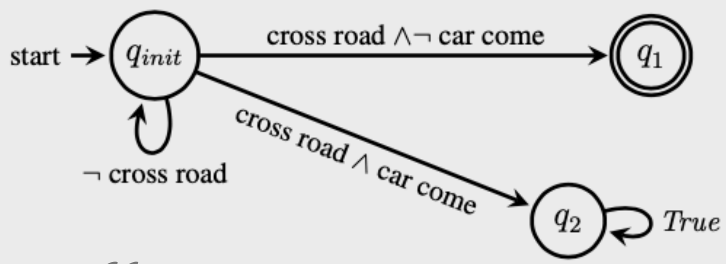
Method “for parties to jointly compute a function over their inputs while keeping those inputs private.” (Wikipedia)

```
Steps for: secure multi-party computation
[1] Define problem and inputs.
[2] Secret sharing of inputs.
[3] Compute secret shares.
[4] Reconstruct the final result.
[5] Output verification.
[6] Decrypt the final result.

Substeps for: [2] Secret sharing of inputs.
[2.1] Generate random secret shares.
[2.2] Securely store secret shares.

Substeps for: [3] Compute secret shares.
[3.1] Encrypt secret share.
[3.2] Distribute encrypted shares.
[3.3] Compute ciphertext.
[3.4] Broadcast result.
```



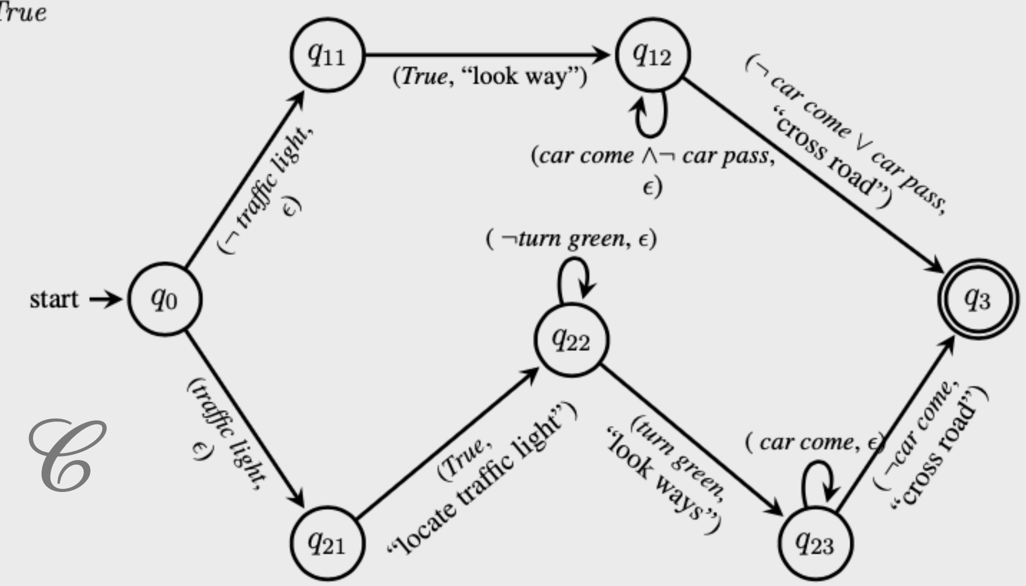
Go back to verification

$$\mathscr{M} \otimes \mathscr{C} \models \Phi \checkmark$$



$\mathscr{M}$

$\Phi = \neg$ traffic light $\rightarrow \Diamond$ goal



$\mathscr{C}$

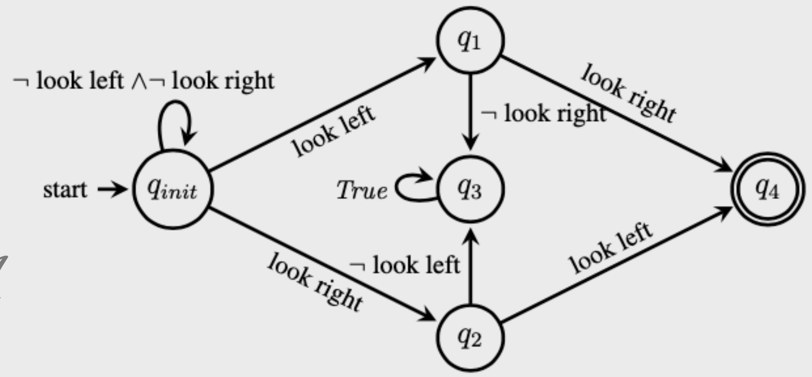
Additional information, e.g., a model, available

Constructed using GLM2FSA

Specification

A case that does not pass the verification

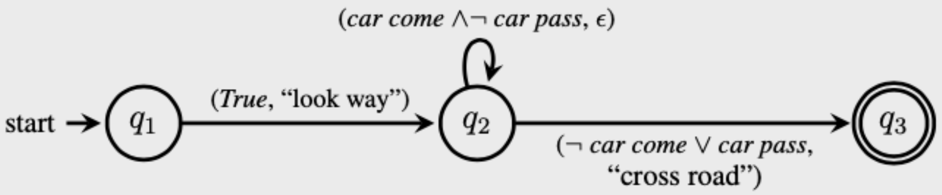
$$\mathscr{M} \otimes \mathscr{C} \models \Phi$$



$\mathscr{M}$

Specification

$\Phi = \neg$ traffic light $\rightarrow \Diamond$ goal

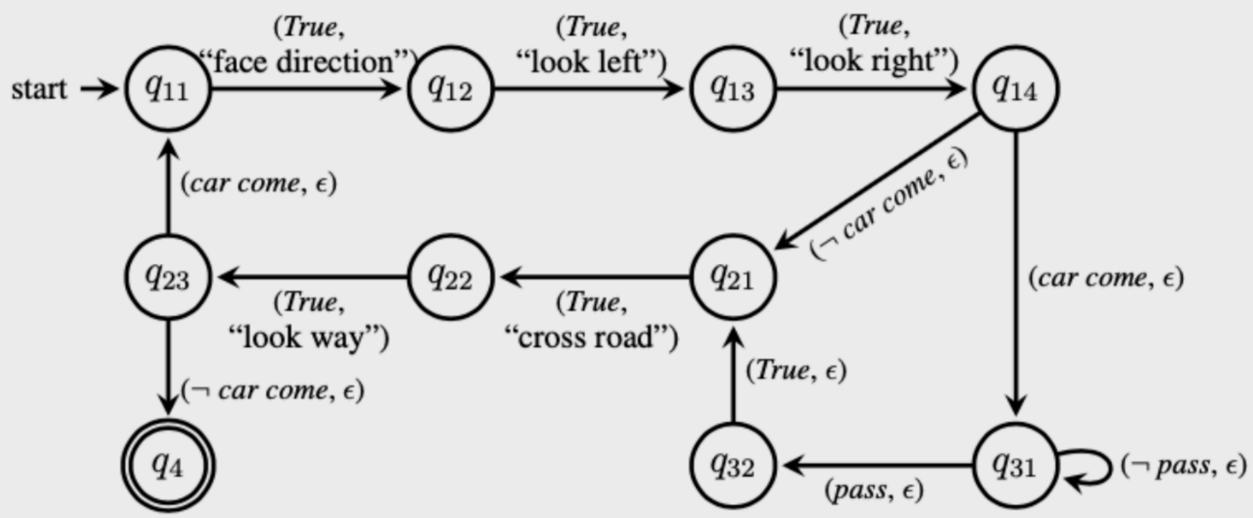


$\mathscr{C}$: Part of the "top-layer" from GLM2FSA

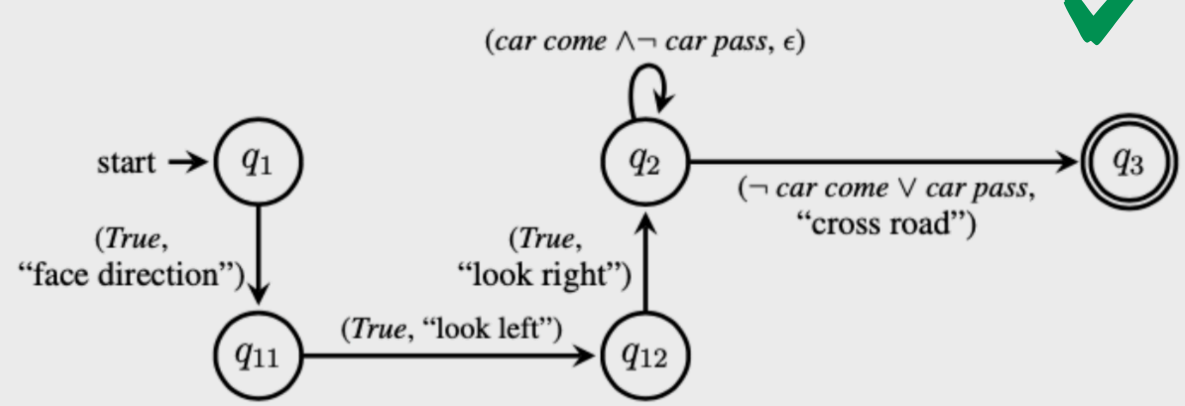
✘ The model checker finds a counter-example: $q_{init}, q_{init}, q_{init}, \ldots$

Refinement of the controller

Query for additional sub-steps



Prune "unintuitive transitions" while keeping it verified



A few next steps

Clean things up. Understand what we are actually doing.

Expand the class of finite-state objects that can be distilled. Integrate into automata learning.

Probabilistic versions

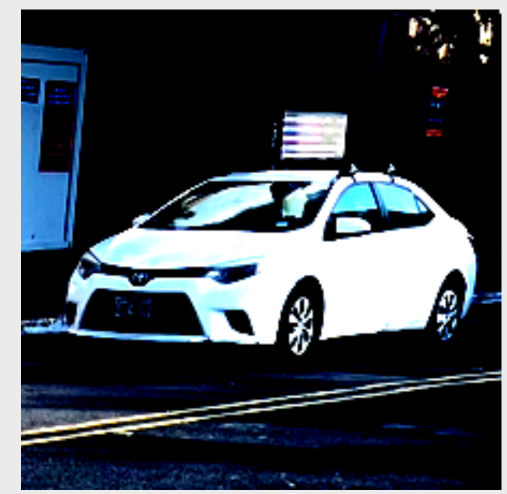
Joint planning and perception through generative models for joint language and image

Utilize for task-guided reinforcement learning

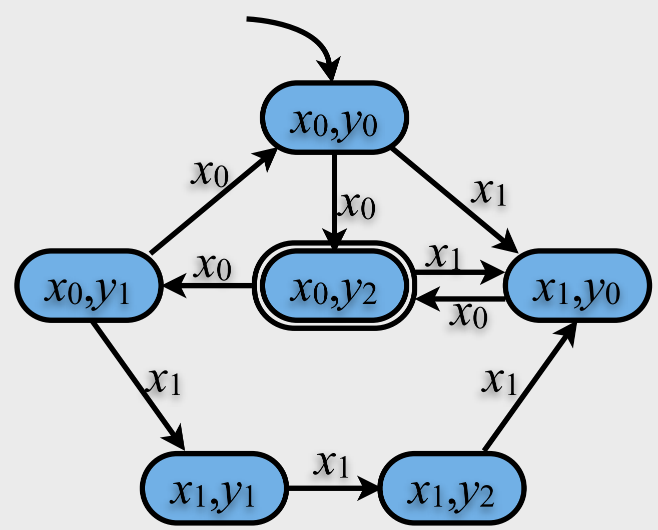
Generative models jointly for language and image. Why?

x_1: "at the cross walk"

x_0: "the traffic light is green"

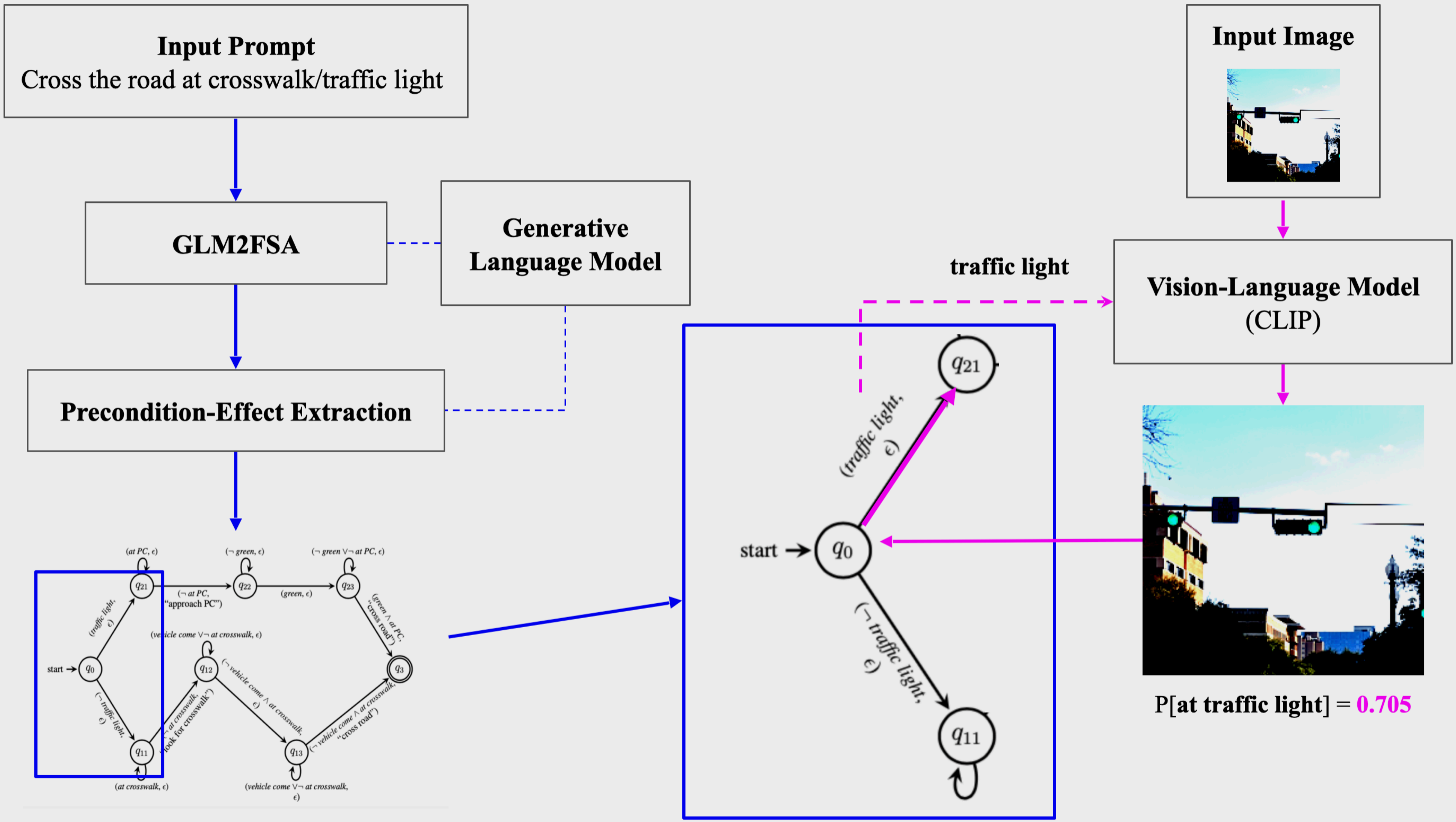


x_2: "a car is approaching"



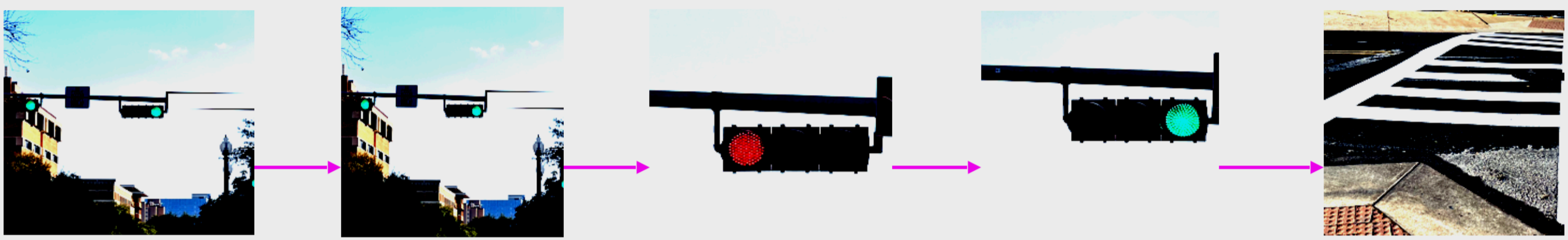
A reactive control logic determining the system choices y in reaction to the changes in the environment variables x

A tentative workflow



A glimpse of (potential) results on “cross the road”

$q_0 \rightarrow q_{21}$ | $q_{21} \rightarrow q_{22}$ | $q_{22} \rightarrow q_{22}$ | $q_{22} \rightarrow q_{23}$ | $q_{22} \rightarrow q_{23}$



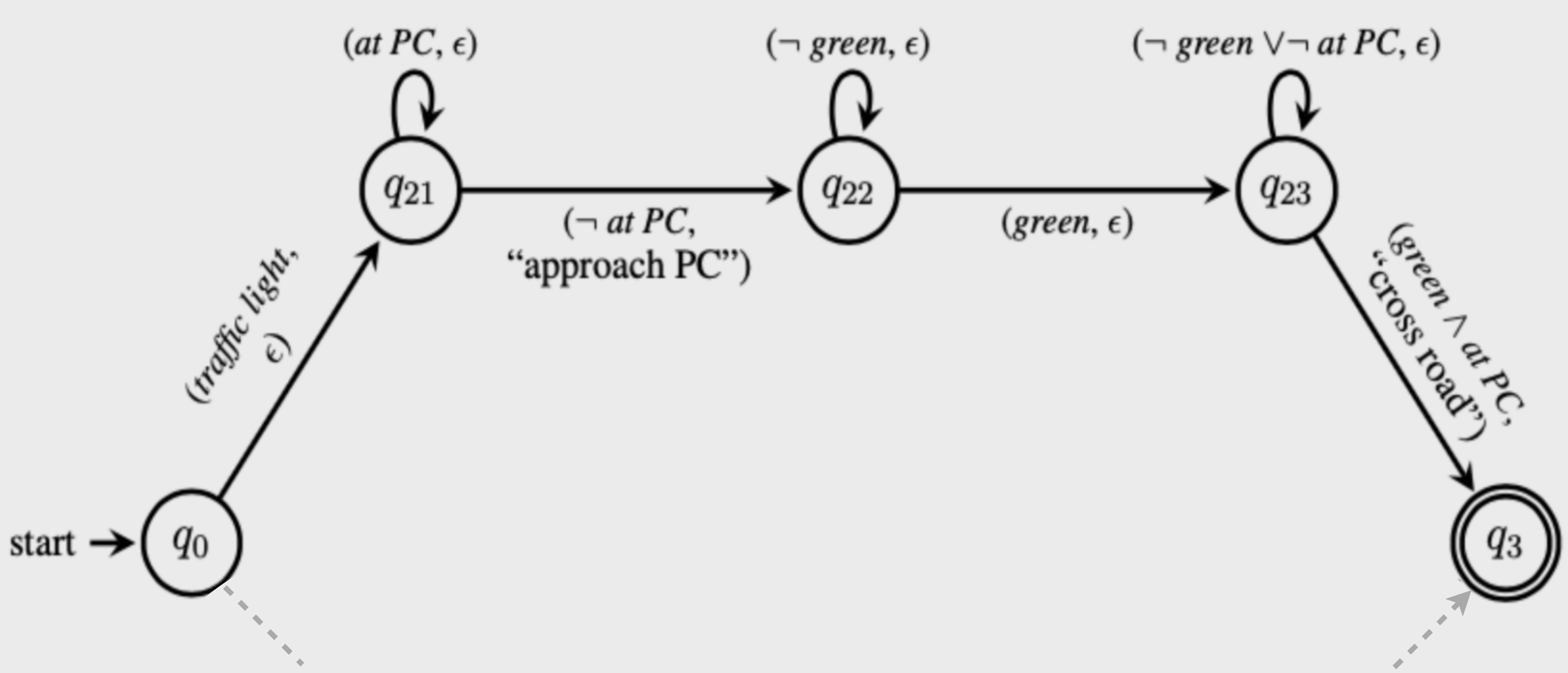
P[at traffic light] = 0.705

P[at pedestrian crossing] = 0.152

P[traffic light is green] = 0.366

P[traffic light is green] = 0.577

P[at pedestrian crossing] = 0.998



A few next steps

Clean things up. Understand what we are actually doing.

Expand the class of finite-state objects that can be distilled. Integrate into automata learning.

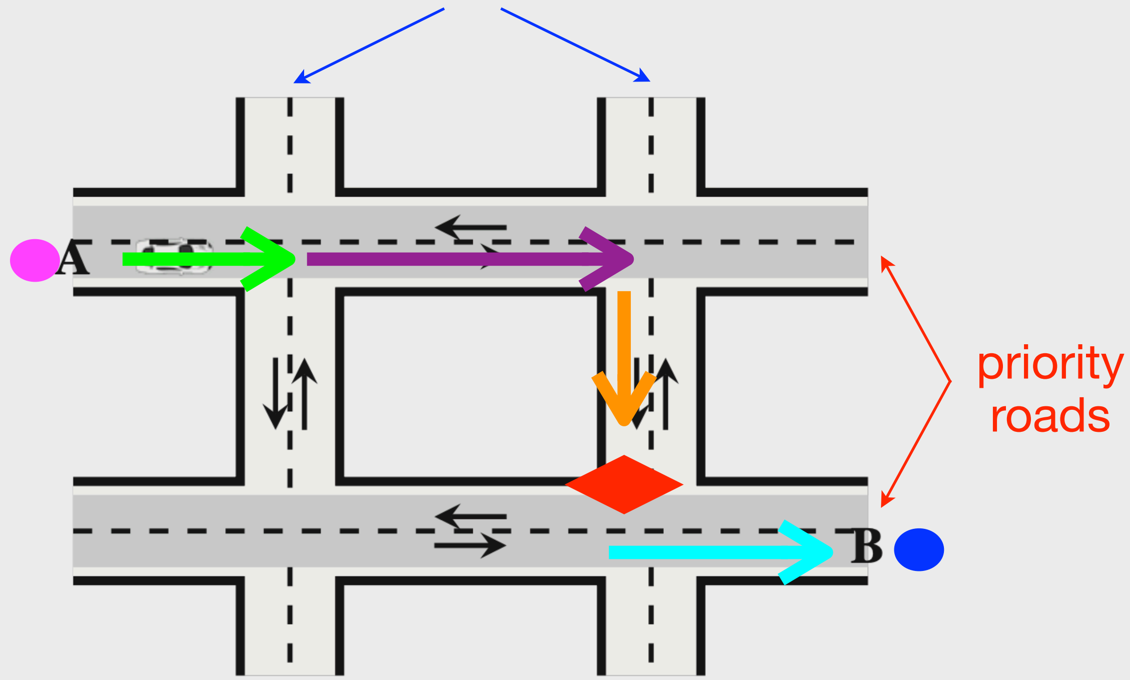
Probabilistic versions

“Grounding” through generative models for joint language and image.

Utilize for task-guided reinforcement learning

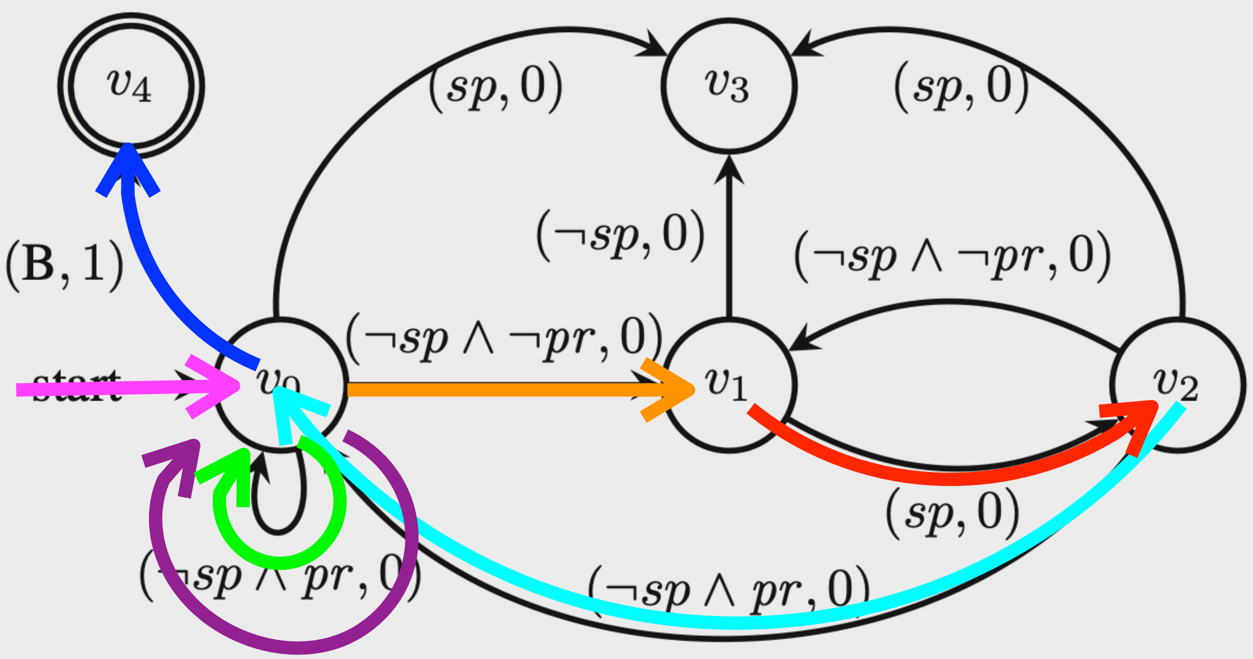
How to represent contextual information?

(using **reward machines**)



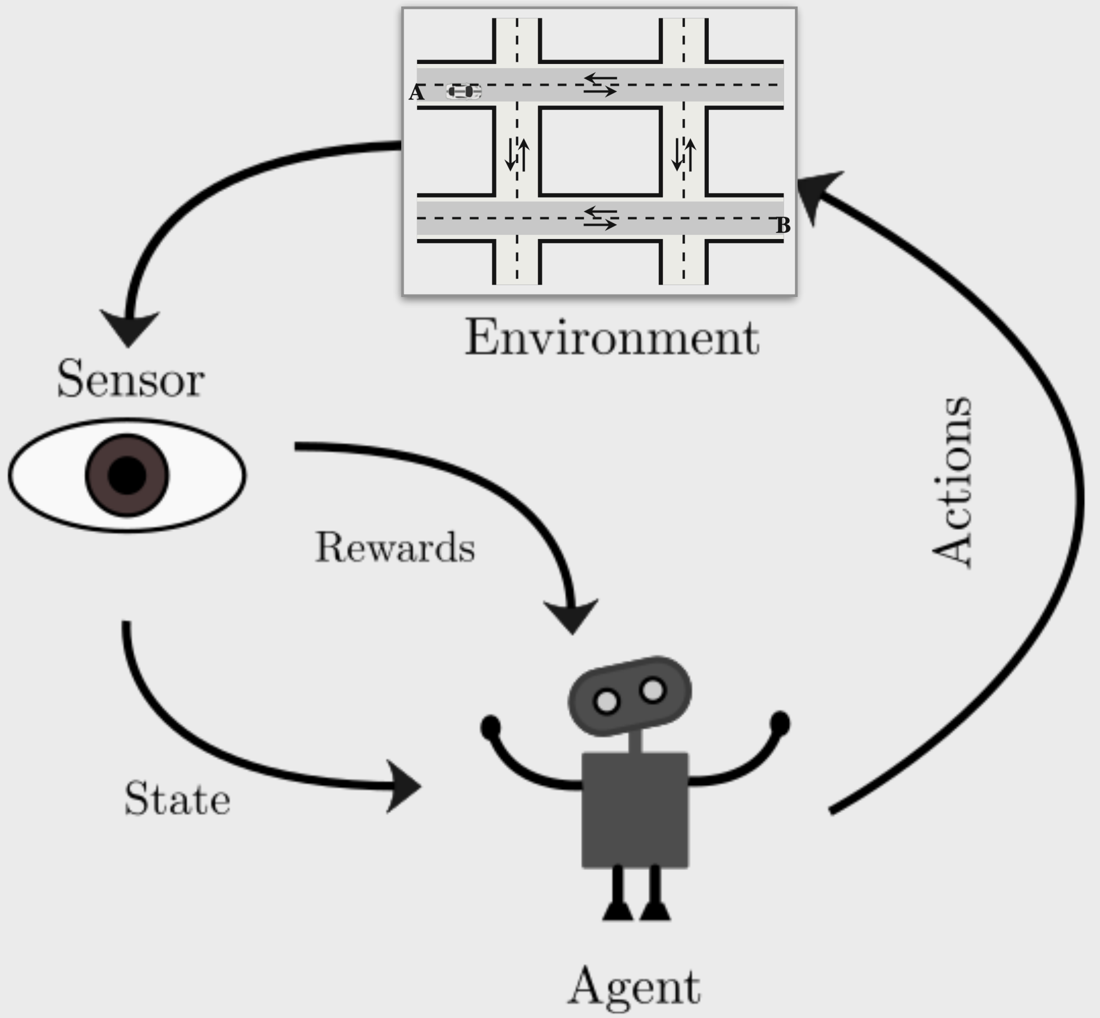
A car to drive from A to B while obeying the "traffic rules":

- Traveling on an ordinary road, stop at intersection for one time step.
- Traveling on a priority road, do not stop at intersection.



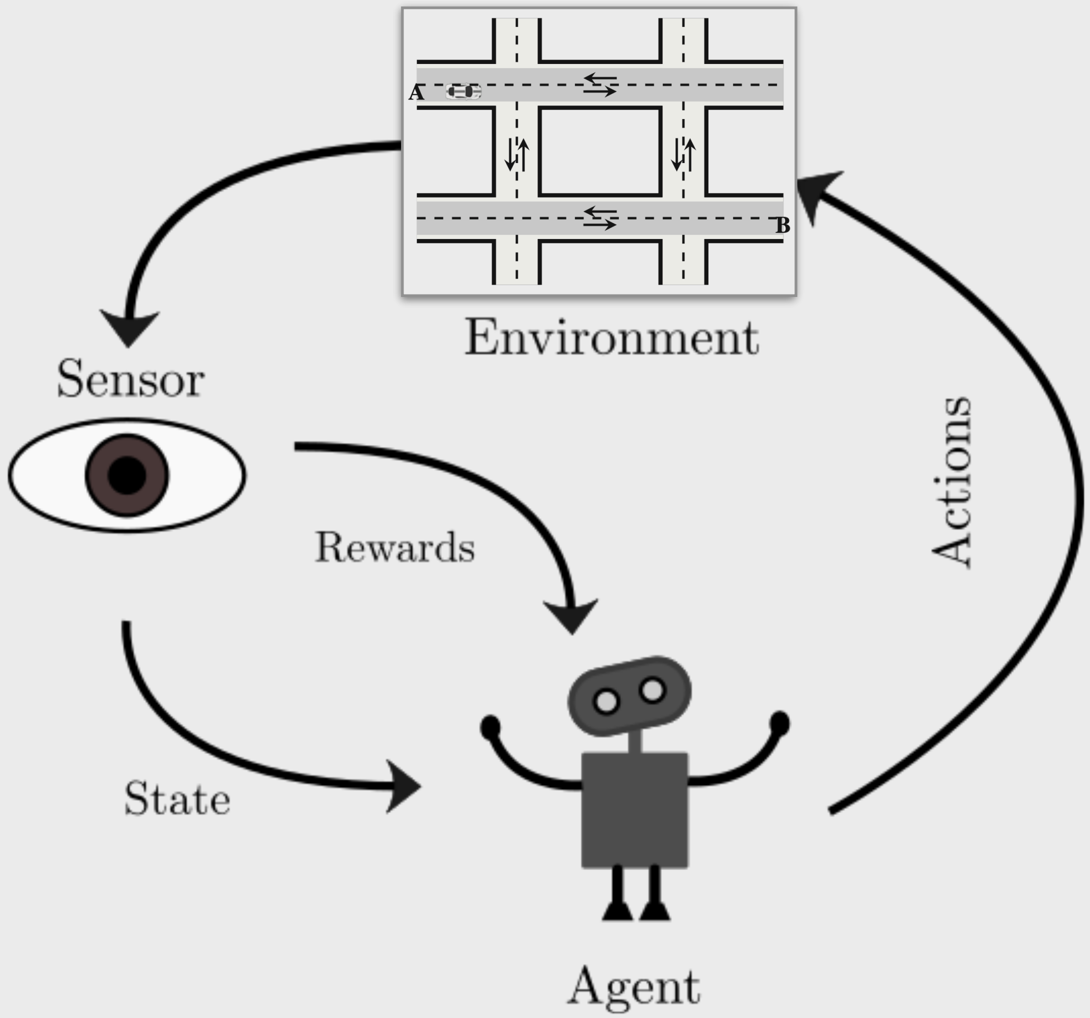
a reward machine

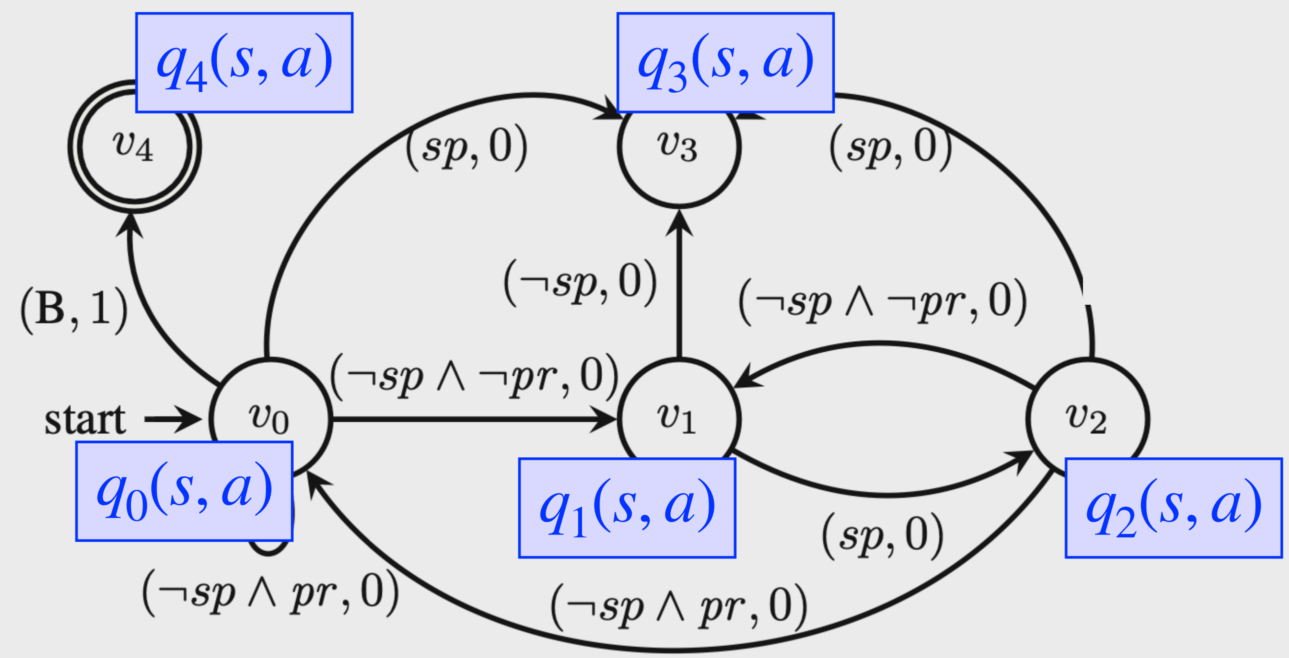
Reinforcement learning with reward machines



$$\pi^* = \arg\max_{\pi} \; q(s, a)$$

Reinforcement learning with reward machines



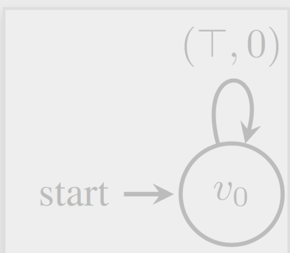


(s, a)— state and action over the environment

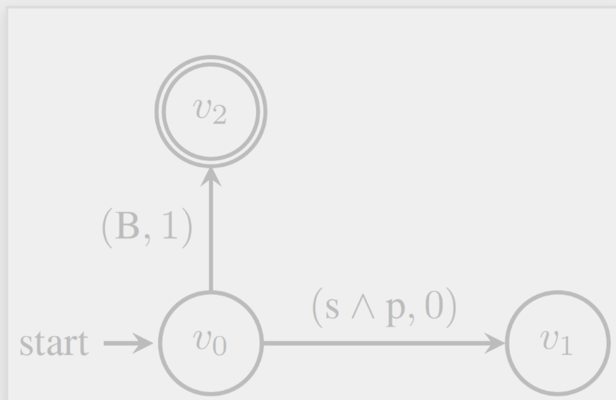
Joint task inference and reinforcement learning

How it works…

initial (trivial) hypothesis

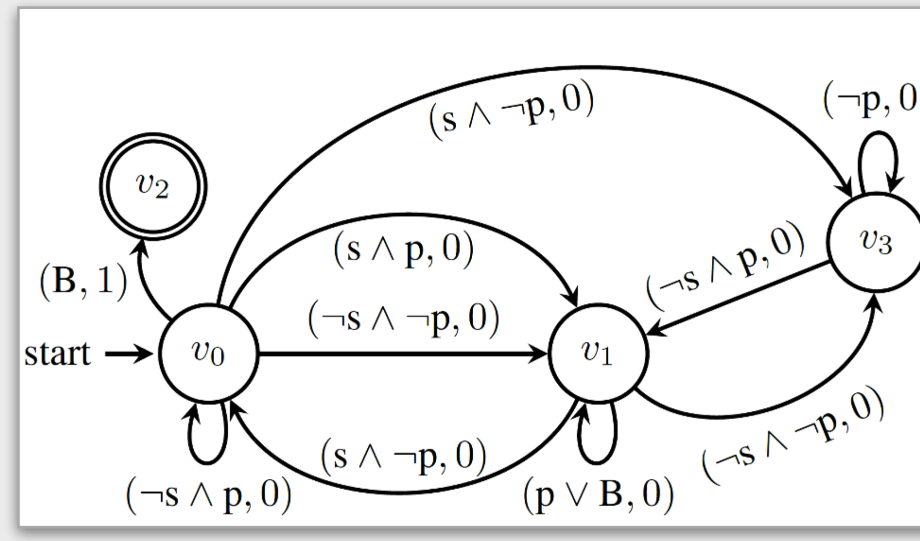


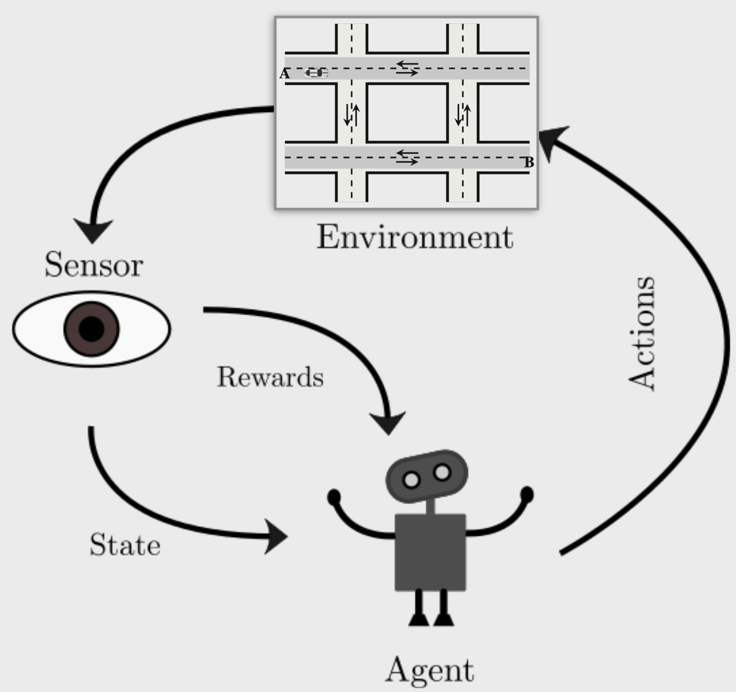
iteration 1



…

iteration n





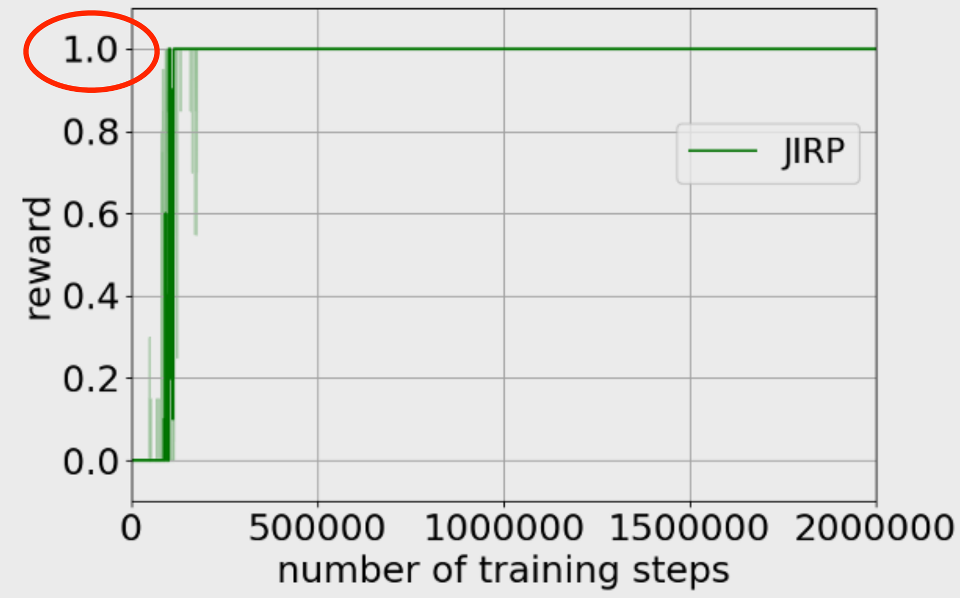
Joint task inference and reinforcement learning

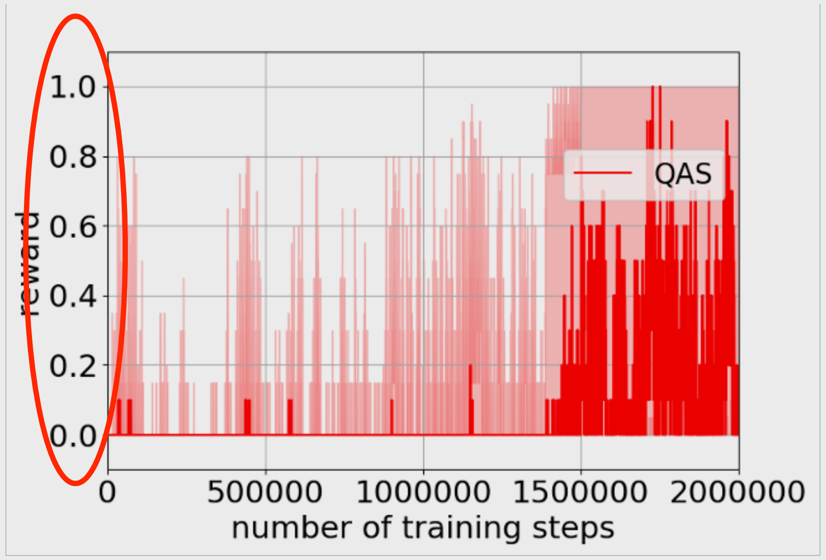
Empirically…

Two-orders-of-magnitude improvement in data efficiency.

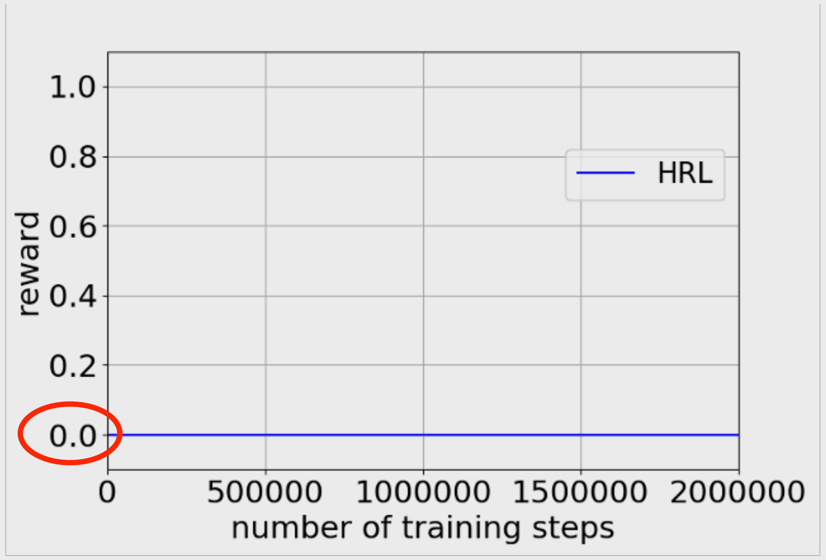
Reliable convergence (with no “additional” parameter tuning).

Consistent results across a range of benchmarks.

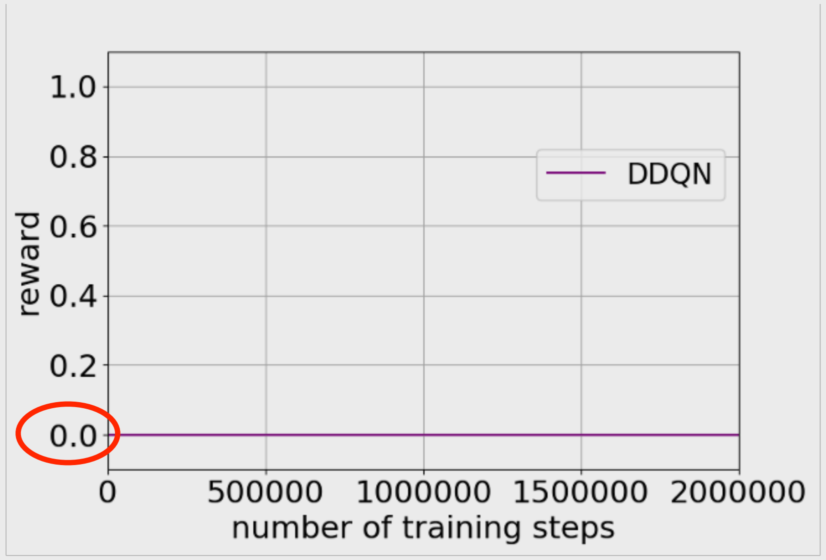




q-learning with augmented state space



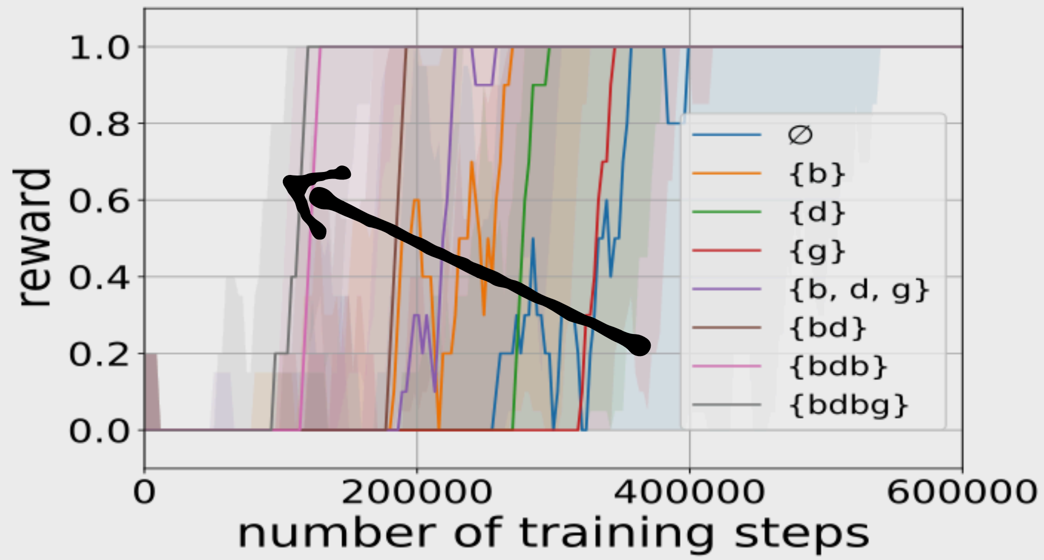
hierarchical reinforcement learning



deep reinforcement learning with double q-learning

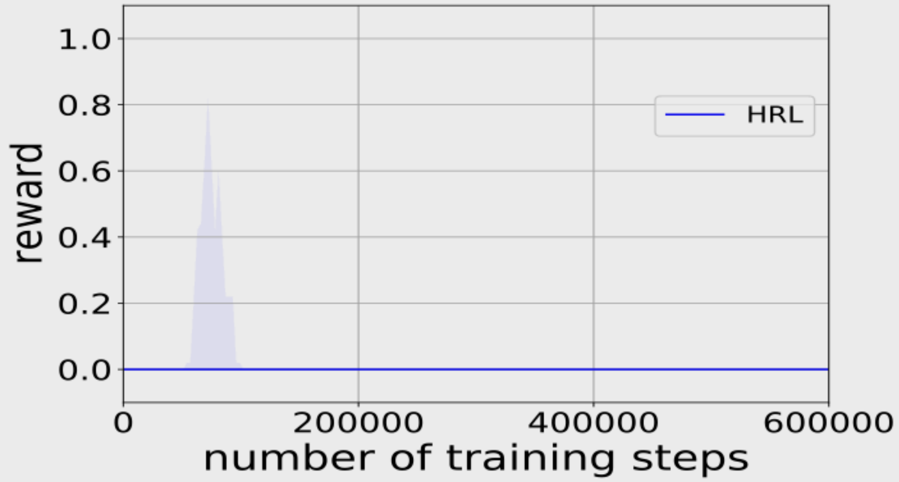
“Advice-guided” reinforcement learning

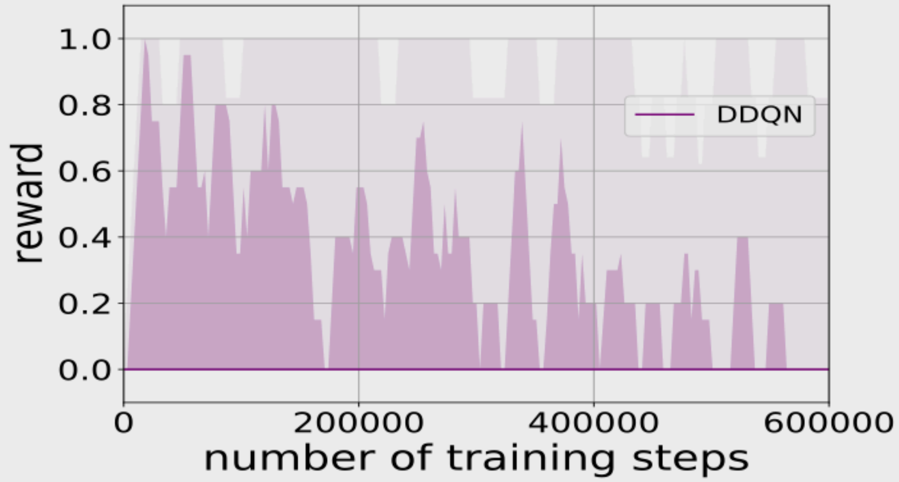
Can we warm-start? Can we recover from bad advice?



The more “informative” the initial advice, the lower the amount of data necessary.

hierarchical reinforcement learning





deep reinforcement learning with double q-learning

Summary

A (hopefully) useful interpretation:
Generative models for language and image are emerging tools that deserve attention.

They may complement the existing design flows in…
- Model checking, planning,…
- Reactive synthesis, games on graphs, …
- Probabilistic verification and synthesis
- Reinforcement learning

But, care is definitely necessary.

We may decide not to use them but that decision needs to informed rather than hype-based.