Learning Collections of Functions

Anthony Platanios
e.a.platanios@cs.cmu.edu

Thesis Committee:

Tom Mitchell (Chair)	[Carnegie Mellon University]
Graham Neubig	[Carnegie Mellon University]
Rich Caruana	[Microsoft]
Eric Horvitz	[Microsoft]

GOOGLE \ SCIENCE \ TECH

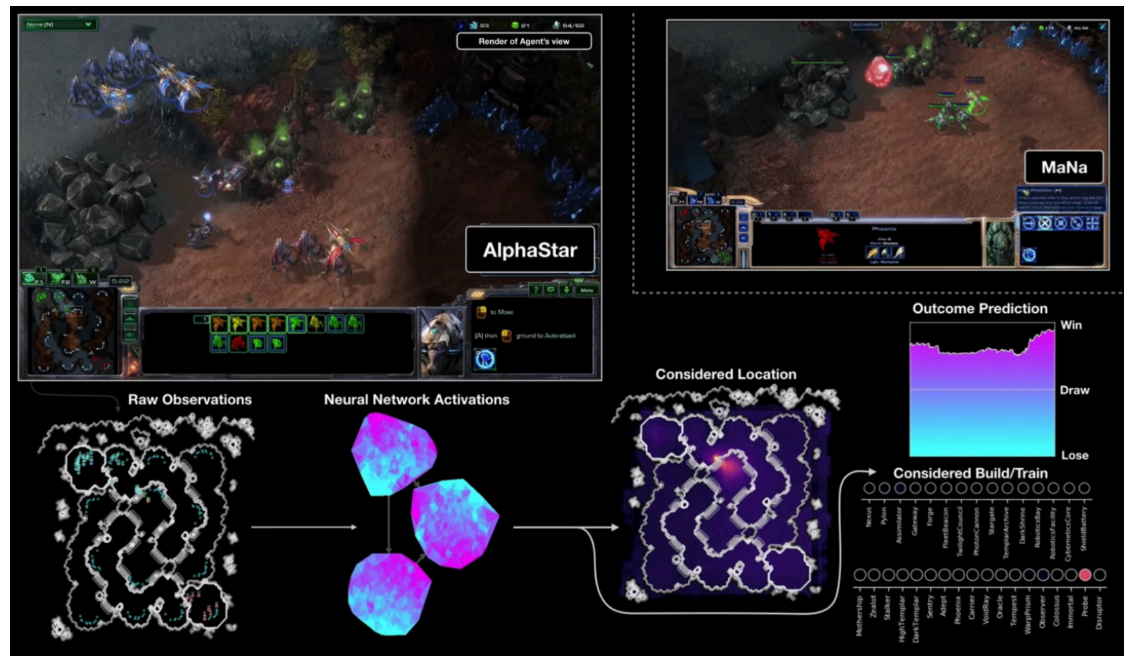
DeepMind's Go-playing AI doesn't need human help to beat us anymore

The company's latest AlphaGo AI learned superhuman skills by playing itself over and over

By James Vincent | Oct 18, 2017, 1:00pm EDT

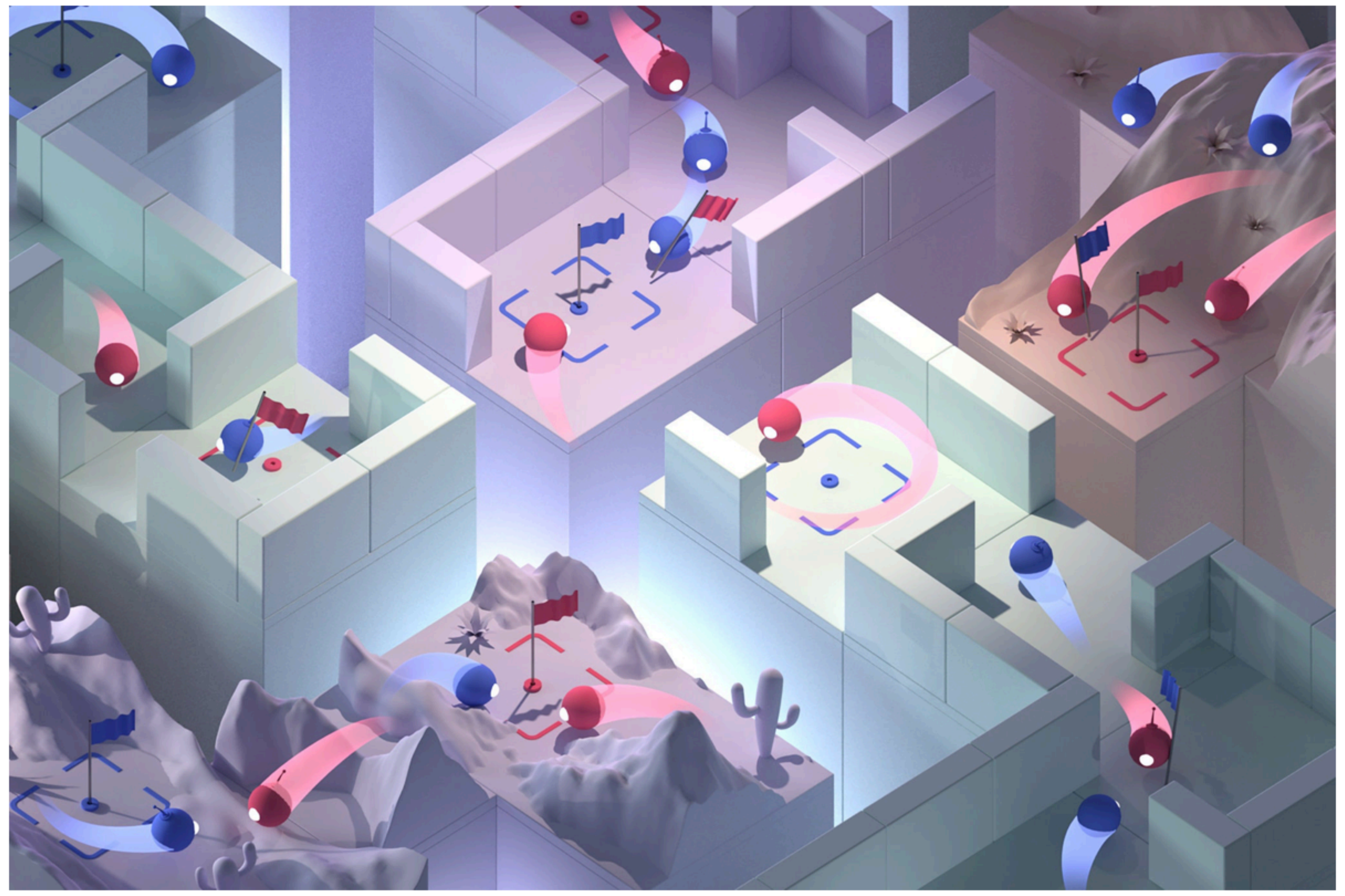
StarCraft II-playing AI AlphaStar takes out pros undefeated

Devin Coldewey @techcrunch / 5 months ago Comment



DeepMind Can Now Beat Us at Multiplayer Games, Too

Chess and Go were child's play. Now A.I. is winning at capture the flag. Will such skills translate to the real world?



DeepMind

By Cade Metz

May 30, 2019

56

When Is Technology Too Dangerous to Release to the Public?

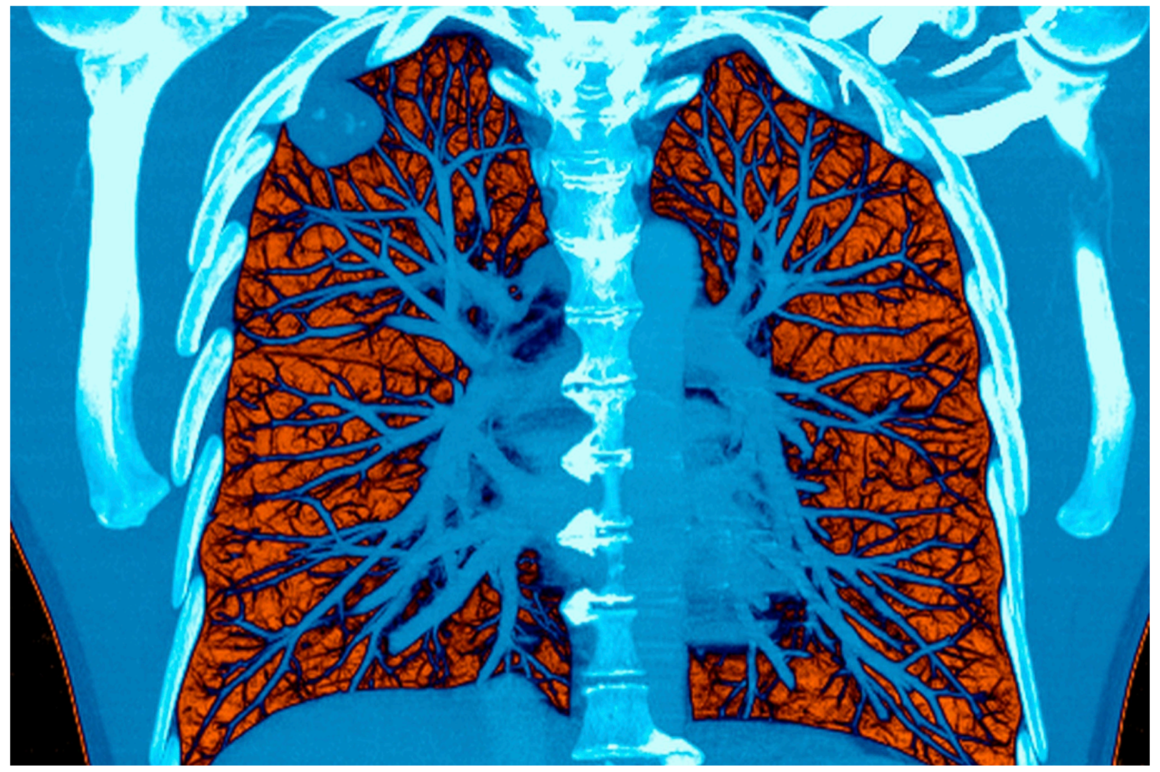
A new text-generating algorithm has reignited a long-running debate.

By AARON MAK

FEB 22, 2019 • 5:56 PM

A.I. Took a Test to Detect Lung Cancer. It Got an A.

Artificial intelligence may help doctors make more accurate readings of CT scans used to screen for lung cancer.



A colored CT scan showing a tumor in the lung. Artificial intelligence was just as good, and sometimes better, than doctors in diagnosing lung tumors in CT scans, a new study indicates. Voisin/Science Source

By Denise Grady

May 20, 2019

40

When seeing is no longer believing

Inside the Pentagon's race against deepfake videos

Advances in artificial intelligence could soon make creating convincing fake audio and video – known as "deepfakes" – relatively easy. Making a person appear to say or do something they did not has the potential to take the war of disinformation to a whole new level. Scroll down for more on deepfakes and what the US government is doing to combat them.

What is missing?

Highway

ResNet50 Classifier → **Dam (99%)**

What if a model knew that this is a road?

And, what if yet another model knew that roads cannot lead into dams?

If these models were able to interact with each other, then this mistake would be highly unlikely!

Example borrowed from Hendrycks et al., 2020.

Thesis Statement

multi-task learning

A computer system that learns to perform multiple tasks jointly and that is aware of the relationships between these tasks, will be able to learn more efficiently and effectively than a system that learns to perform each task in isolation.

Moreover, the relationships between the tasks may either be explicitly provided through supervision or implicitly learned by the system itself, and will allow the system to self-reflect and evaluate itself without any task-specific supervision.

self-reflection

Never-Ending Learning

We will never truly understand *human learning* until we build *machines* that:

1. learn many different types of knowledge from diverse experiences,
2. over many years,
3. and become *better learners* over time.

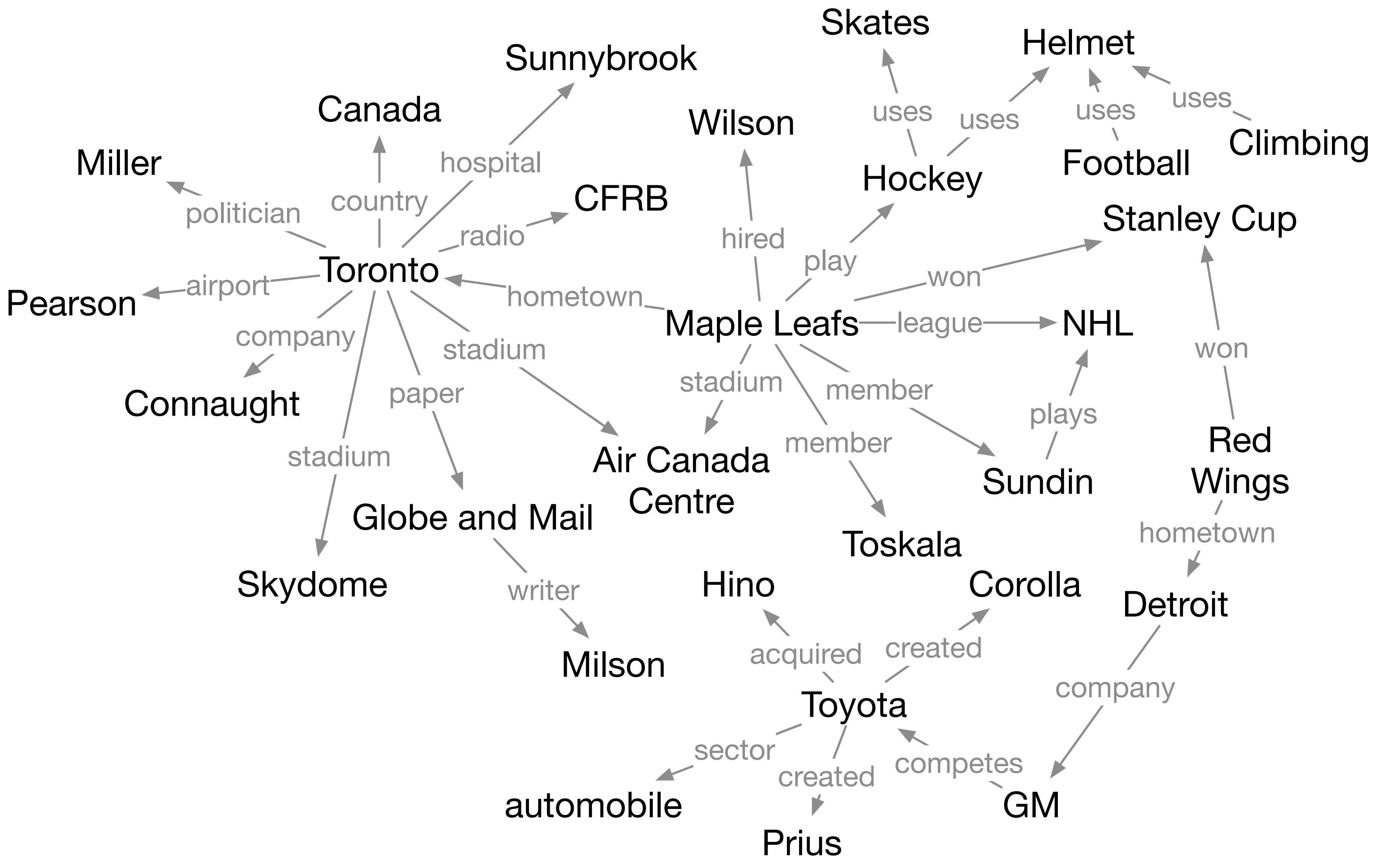
Most current machine learning systems are much more narrow, learning just a single function or data model based on statistical analysis of a single data set.

[Mitchell, ..., **Platanios**, ..., AAAI 2015 & CACM 2018]

Never-Ending Language Learning

NELL

Knowledge Base



~120 million beliefs
~4,100 distinct learning tasks

[Mitchell, …, **Platanios**, …, AAAI 2015 & CACM 2018]

Knowledge Integration

Never-Ending Language Learning

<u>Example Task:</u> Determine whether a noun phrase refers to a city or not.

Context Classifier

"lives in **Pittsburgh**"

"city" appears after "lives in"

Morphology Classifier

"**Pittsburgh**"

"city" ends with "-burgh"

noisy overlapping
sets of beliefs

Integrate Knowledge

Integrate noisy beliefs into
a confident set of *facts*.

"Pittsburgh is a city"

confident facts that will be
added to the knowledge base

Machine Learning

Data Programming

snorkel

DeepDive

weak supervision

Crowdsourcing

figure eight

amazon mechanical turk

HIVE

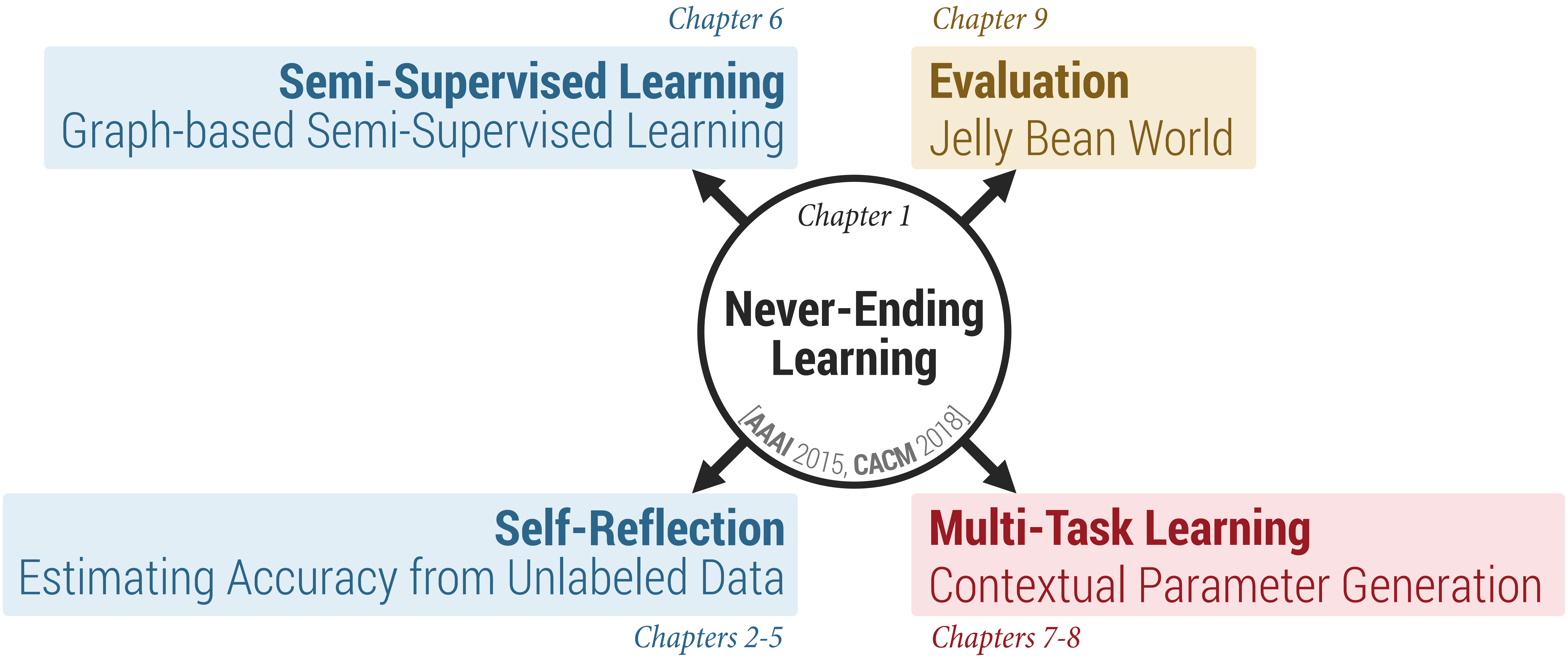
noisy overlapping
sets of labels

Aggregate Labels

Aggregate the noisy labels to
obtain a single label per example.

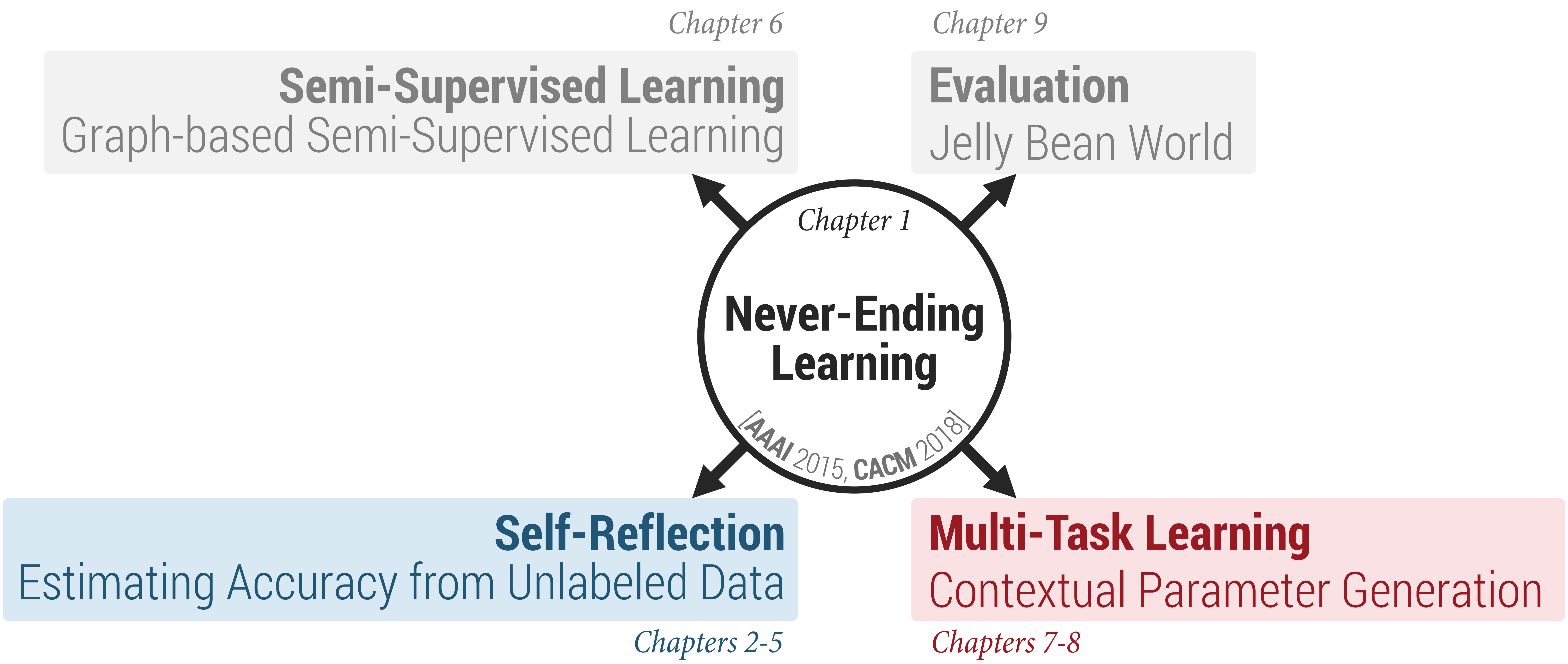
use aggregated labels to train
machine learning models

Overview



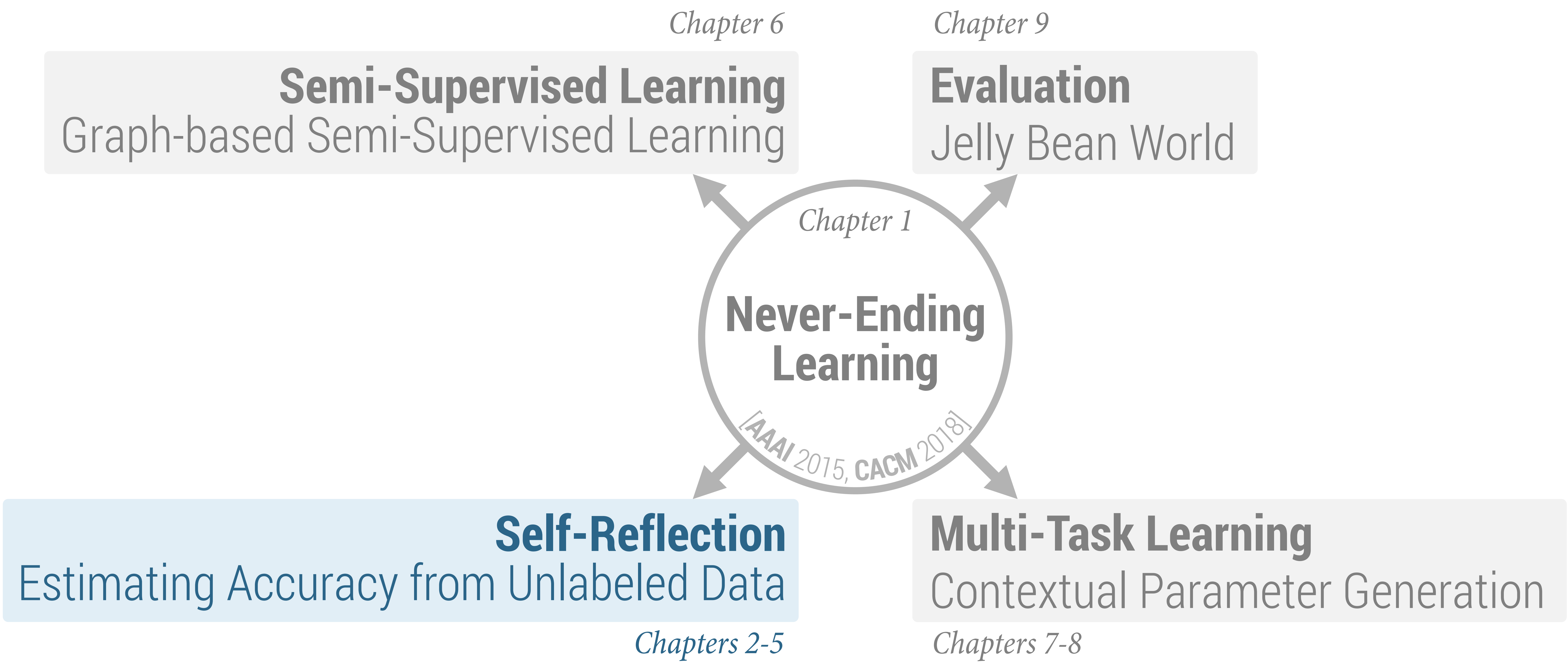
Chapter 6
Semi-Supervised Learning
Graph-based Semi-Supervised Learning
Chapter 9
Evaluation
Jelly Bean World
Chapter 1
Never-Ending Learning
[AAAI 2015, CACM 2018]
Self-Reflection
Estimating Accuracy from Unlabeled Data
Chapters 2-5
Multi-Task Learning
Contextual Parameter Generation
Chapters 7-8

Overview

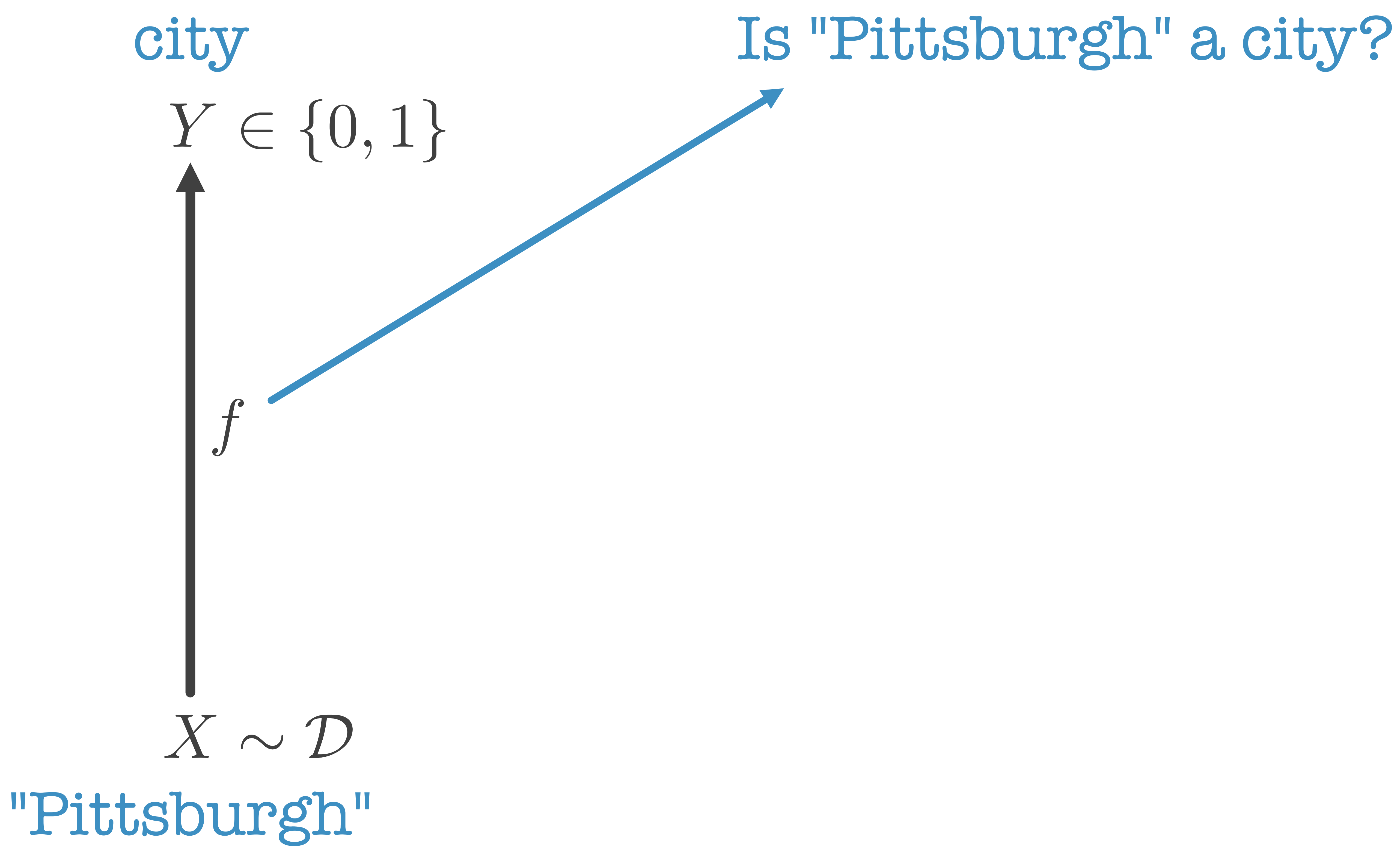


Chapter 6
Semi-Supervised Learning
Graph-based Semi-Supervised Learning
Chapter 9
Evaluation
Jelly Bean World
Chapter 1
Never-Ending Learning
[AAAI 2015, CACM 2018]
Self-Reflection
Estimating Accuracy from Unlabeled Data
Chapters 2-5
Multi-Task Learning
Contextual Parameter Generation
Chapters 7-8

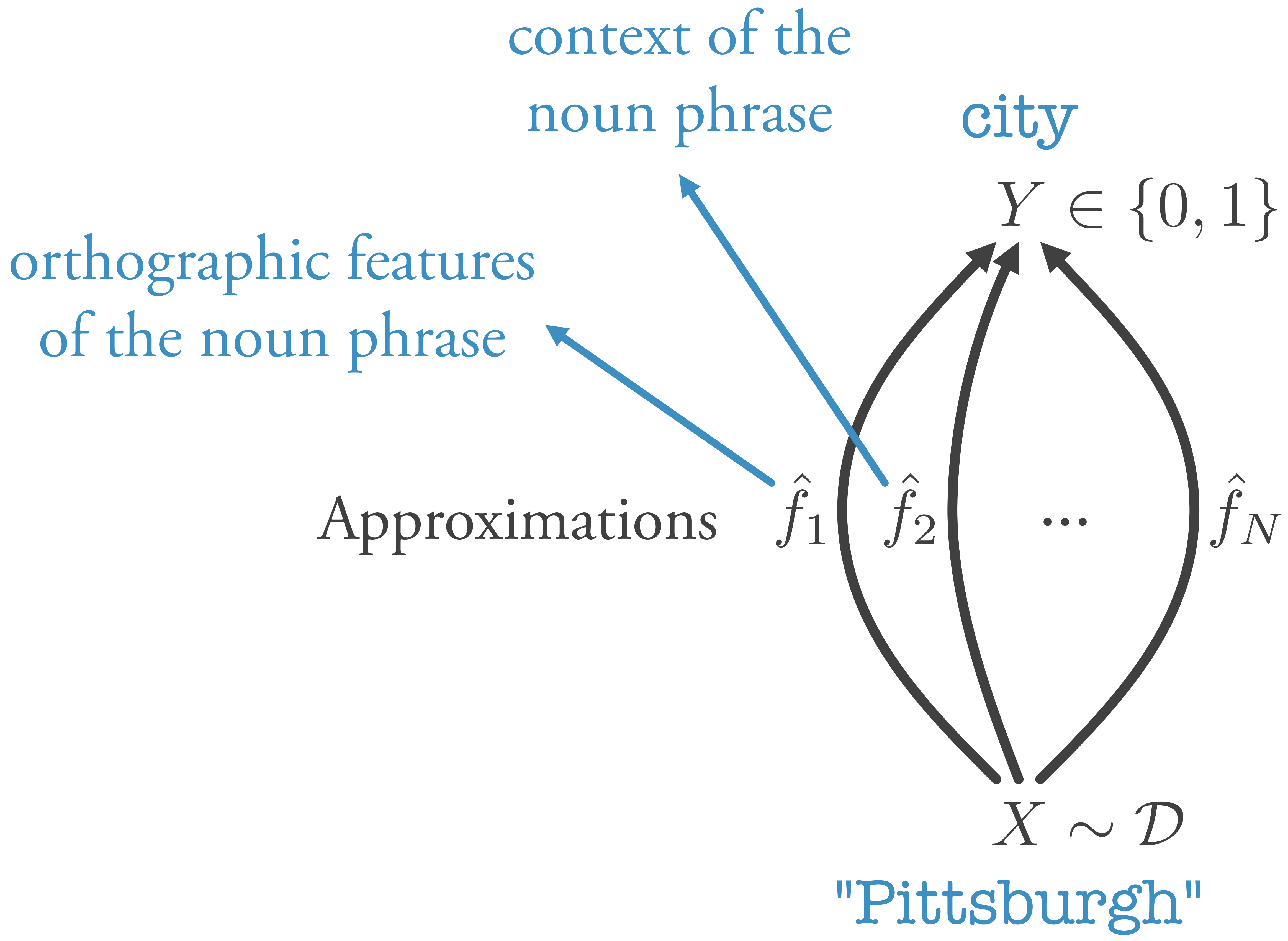
Overview



Chapter 6
Semi-Supervised Learning
Graph-based Semi-Supervised Learning
Chapter 9
Evaluation
Jelly Bean World
Chapter 1
Never-Ending Learning
[AAAI 2015, CACM 2018]
Self-Reflection
Estimating Accuracy from Unlabeled Data
Chapters 2-5
Multi-Task Learning
Contextual Parameter Generation
Chapters 7-8

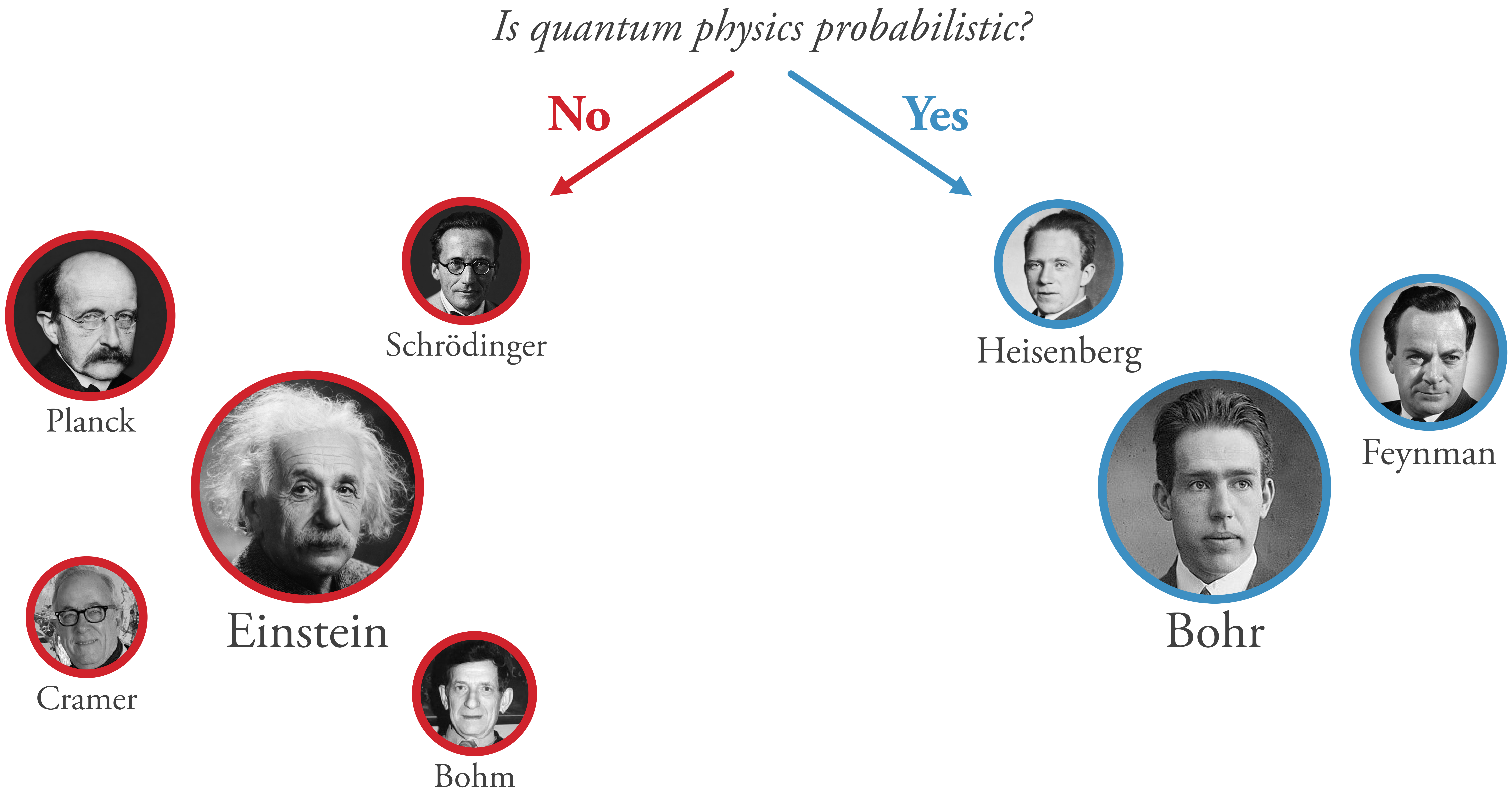

city
Is "Pittsburgh" a city?
$Y \in \{0, 1\}$
f
$X \sim \mathcal{D}$
"Pittsburgh"

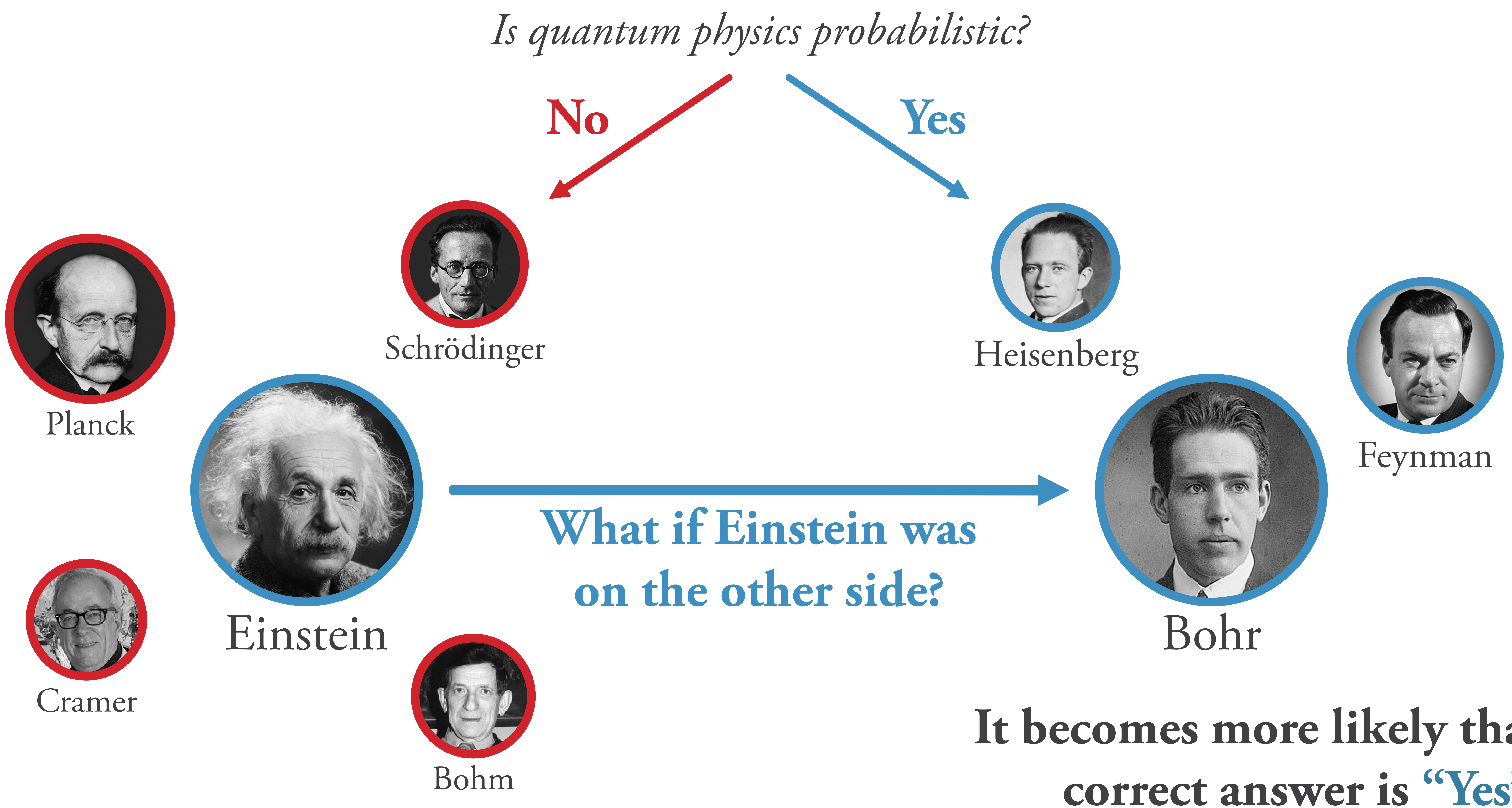
A Direct Approach



What would a human do?

Self-Reflection





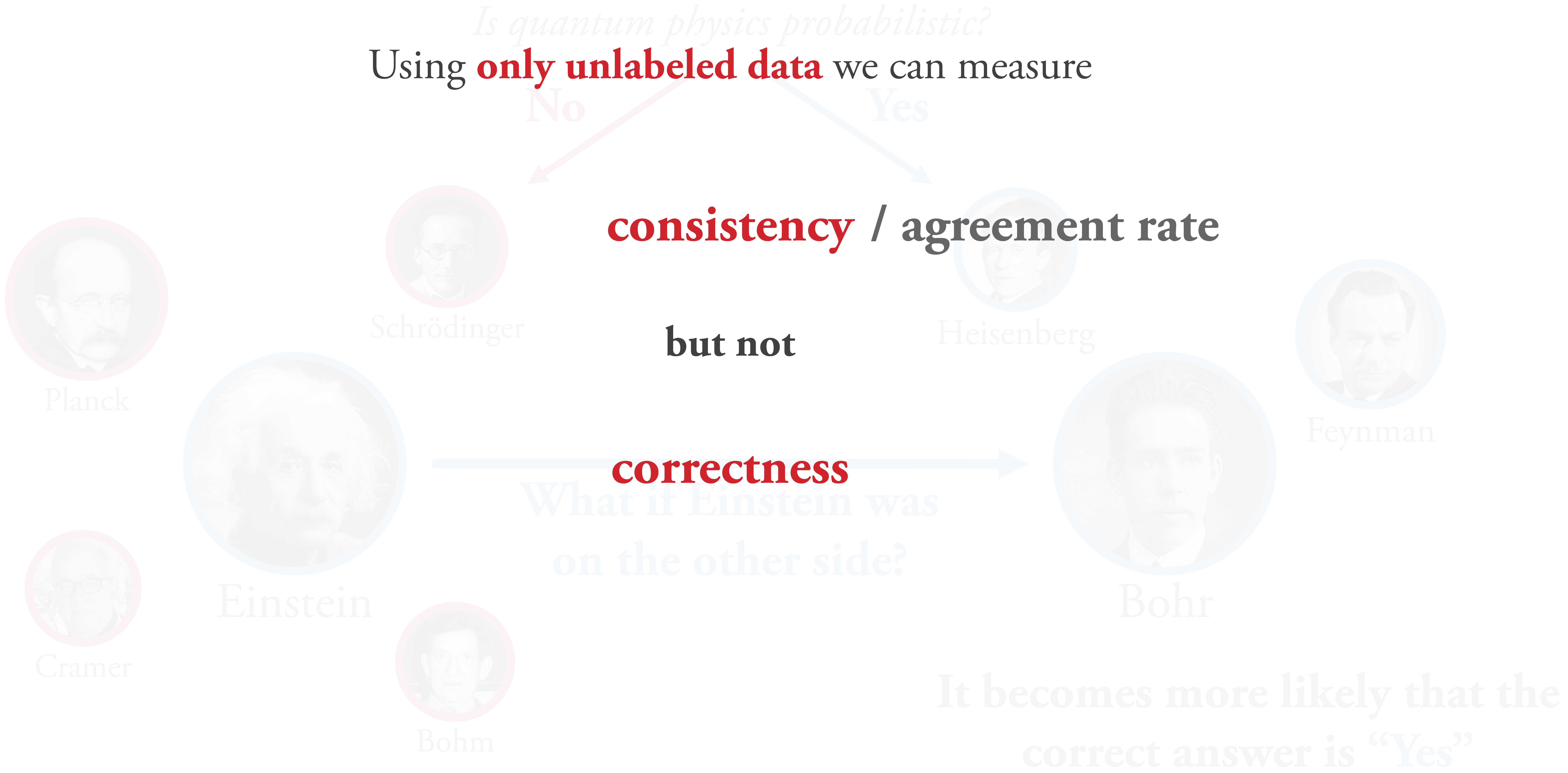
It becomes more likely that the correct answer is "Yes"

Using **only unlabeled data** we can measure

consistency / agreement rate

but not

correctness



Using **only unlabeled data** we can measure

consistency

Does this implication hold?

correctness

If yes, under what conditions?

Dependence?

consistency

Probability that predictor $\hat{f}_i$ and predictor $\hat{f}_j$ agree:

$$a_{ij} = P_{X \sim \mathcal{D}}([\hat{f}_i(X) = \hat{f}_j(X)])$$

Can be estimated using unlabeled $X_1, \ldots, X_S$:

$$\hat{a}_{ij} = \frac{1}{S} \sum_{s=1}^{S} \mathbb{1}_{[\hat{f}_i(X_s) = \hat{f}_j(X_s)]}$$

[**Platanios**, Blum, Mitchell, UAI 2014]

Self-Reflection

consistency

Probability that predictor $\hat{f}_i$ and predictor $\hat{f}_j$ agree:

$$a_{ij} = P_{X \sim \mathcal{D}}([\hat{f}_i(X) = \hat{f}_j(X)])$$

Can be estimated using unlabeled $X_1, \ldots, X_S$:

$$\hat{a}_{ij} = \frac{1}{S} \sum_{s=1}^{S} \mathbb{1}_{[\hat{f}_i(X_s) = \hat{f}_j(X_s)]}$$

correctness

Probability that predictor $\hat{f}_i$ is wrong:

$$e_i = P_{X \sim \mathcal{D}}(\underbrace{[\hat{f}_i(X) \neq Y]}_{\substack{E_i \\ \textit{error event}}})$$

[**Platanios**, Blum, Mitchell, UAI 2014]

A Direct Approach

consistency

Probability that predictor $\hat{f}_i$ and predictor $\hat{f}_j$ agree:

$$a_{ij} = P_{X \sim \mathcal{D}}([\hat{f}_i(X) = \hat{f}_j(X)])$$

$$a_{ij} = \underbrace{P_{\mathcal{D}}(E_i \cap E_j)}_{\textit{both are wrong}} + \underbrace{P_{\mathcal{D}}(\bar{E}_i \cap \bar{E}_j)}_{\textit{both are right}}$$

Can be estimated using unlabeled $X_1, \ldots, X_S$:

$$\hat{a}_{ij} = \frac{1}{S} \sum_{s=1}^{S} \mathbb{1}_{[\hat{f}_i(X_s) = \hat{f}_j(X_s)]}$$

correctness

Probability that predictor $\hat{f}_i$ is wrong:

$$e_i = P_{X \sim \mathcal{D}}(\underbrace{[\hat{f}_i(X) \neq Y]}_{\substack{E_i \\ \textit{error event}}})$$

[**Platanios**, Blum, Mitchell, UAI 2014]

Self-Reflection

consistency

Probability that predictor $\hat{f}_i$ and predictor $\hat{f}_j$ agree:

$$a_{ij} = P_{X\sim\mathcal{D}}([\hat{f}_i(X) = \hat{f}_j(X)])$$

Can be estimated using unlabeled $X_1, \ldots, X_S$:

$$\hat{a}_{ij} = \frac{1}{S}\sum_{s=1}^{S} \mathbb{1}_{[\hat{f}_i(X_s)=\hat{f}_j(X_s)]}$$

$$a_{ij} = \overbrace{P_{\mathcal{D}}(E_i \cap E_j)}^{\textit{both are wrong}} + \overbrace{P_{\mathcal{D}}(\bar{E}_i \cap \bar{E}_j)}^{\textit{both are right}}$$

inclusion-exclusion principle

$$a_{ij} = 1 - e_i - e_j + 2e_{ij}$$

$$e_{ij} = P_{X\sim\mathcal{D}}([\hat{f}_i(X) \neq Y \text{ and } \hat{f}_j(X) \neq Y])$$

correctness

Probability that predictor $\hat{f}_i$ is wrong:

$$e_i = P_{X\sim\mathcal{D}}(\underbrace{[\hat{f}_i(X) \neq Y]}_{\substack{E_i \\ \textit{error event}}})$$

[**Platanios**, Blum, Mitchell, UAI 2014]

consistency

Probability that predictor $\hat{f}_i$ and predictor $\hat{f}_j$ agree:

$$a_{ij} = P_{X \sim \mathcal{D}}([\hat{f}_i(X) = \hat{f}_j(X)])$$

Can be estimated using unlabeled $X_1, \ldots, X_S$:

$$\hat{a}_{ij} = \frac{1}{S} \sum_{s=1}^{S} \mathbb{1}_{[\hat{f}_i(X_s) = \hat{f}_j(X_s)]}$$

correctness

Probability that predictor $\hat{f}_i$ is wrong:

$$e_i = P_{X \sim \mathcal{D}}([\underbrace{\hat{f}_i(X) \neq Y}_{\substack{E_i \\ \textit{error event}}}])$$

consistency and correctness are indeed related

$$a_{ij} = 1 - e_i - e_j + 2e_{ij}$$

[**Platanios**, Blum, Mitchell, UAI 2014]

A Direct Approach

consistency

Probability that predictor $\hat{f}_i$ and predictor $\hat{f}_j$ agree:

$$a_{ij} = P_{X \sim \mathcal{D}}([\hat{f}_i(X) = \hat{f}_j(X)])$$

Can be estimated using unlabeled $X_1, \ldots, X_S$:

$$\hat{a}_{ij} = \frac{1}{S} \sum_{s=1}^{S} \mathbb{1}_{[\hat{f}_i(X_s) = \hat{f}_j(X_s)]}$$

$$a_{ij} = 1 - e_i - e_j + 2e_{ij}$$

independence

$$e_{ij} \xrightarrow{\text{independence}} e_i e_j$$

correctness

Probability that predictor $\hat{f}_i$ is wrong:

$$e_i = P_{X \sim \mathcal{D}}(\underbrace{[\hat{f}_i(X) \neq Y]}_{\substack{E_i \\ \textit{error event}}})$$

independence

Assuming we have 3 predictors:

$$\left.\begin{aligned} \hat{a}_{12} &= 1 - e_1 - e_2 + 2e_1e_2 \\ \hat{a}_{13} &= 1 - e_1 - e_3 + 2e_1e_3 \\ \hat{a}_{23} &= 1 - e_2 - e_3 + 2e_2e_3 \end{aligned}\right\} e_i = \frac{c \pm (1 - 2\hat{a}_{jk})}{\pm 2(1 - 2\hat{a}_{jk})}$$

where: $c = \sqrt{(2\hat{a}_{12} - 1)(2\hat{a}_{13} - 1)(2\hat{a}_{23} - 1)}$

[**Platanios**, Blum, Mitchell, UAI 2014]

A Direct Approach

consistency

Probability that predictor $\hat{f}_i$ and predictor $\hat{f}_j$ agree:

$$a_{ij} = P_{X\sim\mathcal{D}}([\hat{f}_i(X) = \hat{f}_j(X)])$$

Can be estimated using unlabeled $X_1, \ldots, X_S$:

$$\hat{a}_{ij} = \frac{1}{S}\sum_{s=1}^{S}\mathbb{1}_{[\hat{f}_i(X_s)=\hat{f}_j(X_s)]}$$

$$a_{ij} = 1 - e_i - e_j + 2e_{ij}$$

independence helps us solve the problem

correctness

Probability that predictor $\hat{f}_i$ is wrong:

$$e_i = P_{X\sim\mathcal{D}}(\underbrace{[\hat{f}_i(X) \neq Y]}_{E_i})$$

error event

independence

Assuming we have 3 predictors:

$$\left.\begin{aligned}\hat{a}_{12} &= 1 - e_1 - e_2 + 2e_1e_2\\ \hat{a}_{13} &= 1 - e_1 - e_3 + 2e_1e_3\\ \hat{a}_{23} &= 1 - e_2 - e_3 + 2e_2e_3\end{aligned}\right\} e_i = \frac{c \pm (1 - 2\hat{a}_{jk})}{\pm 2(1 - 2\hat{a}_{jk})}$$

where: $c = \sqrt{(2\hat{a}_{12} - 1)(2\hat{a}_{13} - 1)(2\hat{a}_{23} - 1)}$

[**Platanios**, Blum, Mitchell, UAI 2014]

Self-Reflection

consistency

Probability that predictor $\hat{f}_i$ and predictor $\hat{f}_j$ agree:

$$a_{ij} = P_{X\sim\mathcal{D}}([\hat{f}_i(X) = \hat{f}_j(X)])$$

Can be estimated using unlabeled $X_1, \ldots, X_S$:

$$\hat{a}_{ij} = \frac{1}{S}\sum_{s=1}^{S} \mathbb{1}_{[\hat{f}_i(X_s)=\hat{f}_j(X_s)]}$$

correctness

Probability that predictor $\hat{f}_i$ is wrong:

$$e_i = P_{X\sim\mathcal{D}}(\underbrace{[\hat{f}_i(X) \neq Y]}_{\substack{E_i \\ \textit{error event}}})$$

Independence is a very strong assumption!

Without it we end up with more unknowns than equations!

$$a_{ij} = 1 - e_i - e_j + 2e_{ij}$$

[**Platanios**, Blum, Mitchell, UAI 2014]

Self-Reflection

consistency

Probability that predictor $\hat{f}_i$ and predictor $\hat{f}_j$ agree:

$$a_{ij} = P_{X\sim\mathcal{D}}([\hat{f}_i(X) = \hat{f}_j(X)])$$

Can be estimated using unlabeled $X_1, \ldots, X_S$:

$$\hat{a}_{ij} = \frac{1}{S}\sum_{s=1}^{S} \mathbb{1}_{[\hat{f}_i(X_s)=\hat{f}_j(X_s)]}$$

correctness

Probability that predictor $\hat{f}_i$ is wrong:

$$e_i = P_{X\sim\mathcal{D}}([\underbrace{\hat{f}_i(X) \neq Y}_{\substack{E_i \\ \textit{error event}}}])$$

Constrained Optimization Problem

$$a_{ij} = 1 - e_i - e_j + 2e_{ij}$$

use as constraints

define an objective function / regularizer

[**Platanios**, Blum, Mitchell, UAI 2014]

A Direct Approach

consistency

Probability that predictor $\hat{f}_i$ and predictor $\hat{f}_j$ agree:

$$a_{ij} = P_{X \sim \mathcal{D}}([\hat{f}_i(X) = \hat{f}_j(X)])$$

Can be estimated using unlabeled $X_1, \ldots, X_S$:

$$\hat{a}_{ij} = \frac{1}{S} \sum_{s=1}^{S} \mathbb{1}_{[\hat{f}_i(X_s) = \hat{f}_j(X_s)]}$$

correctness

Probability that predictor $\hat{f}_i$ is wrong:

$$e_i = P_{X \sim \mathcal{D}}(\underbrace{[\hat{f}_i(X) \neq Y]}_{\substack{E_i \\ \textit{error event}}})$$

Constrained Optimization Problem

constraints

Valid probabilities:

$$e_{ij} \leq \min\{e_i, e_j\}$$
$$a_{ij} = 1 - e_i - e_j + 2e_{ij}$$
$$0 \leq e_i \leq 1, 0 \leq e_{ij} \leq 1$$

objective

Relax the independence assumption:

$$c(\boldsymbol{e}) = \sum_{ij} (e_{ij} - e_i e_j)^2$$

and keep the joint error rates.

[**Platanios**, Blum, Mitchell, UAI 2014]

A Direct Approach

consistency

Probability that predictor $\hat{f}_i$ and predictor $\hat{f}_j$ agree:

$$a_{ij} = P_{X \sim \mathcal{D}}([\hat{f}_i(X) = \hat{f}_j(X)])$$

Can be estimated using unlabeled $X_1, \ldots, X_S$:

$$\hat{a}_{ij} = \frac{1}{S} \sum_{s=1}^{S} \mathbb{1}_{[\hat{f}_i(X_s) = \hat{f}_j(X_s)]}$$

correctness

Probability that predictor $\hat{f}_i$ is wrong:

$$e_i = P_{X \sim \mathcal{D}}(\underbrace{[\hat{f}_i(X) \neq Y]}_{\substack{E_i \\ \textit{error event}}})$$

Constrained Optimization Problem

constraints

Valid probabilities:

$$e_{ij} \leq \min\{e_i, e_j\}$$
$$a_{ij} = 1 - e_i - e_j + 2e_{ij}$$
$$0 \leq e_i \leq 1, 0 \leq e_{ij} \leq 1$$

objective

Relax the independence assumption:

$$c(\boldsymbol{e}) = \sum_{ij} (e_{ij} - e_i e_j)^2$$

and keep the joint error rates.

NOTE **BRAIN** is harder because the classifiers and the regions are highly dependent!

NELL

Task: Predict whether a noun phrase belongs to a category (e.g., `city`).

4 classifiers

15 categories

~300,000 noun phrases

BRAIN

Task: Find which of two 40 second long story passages corresponds to a time series of fMRI neural activity.

11 classifiers

11 brain regions

1,000 passages

[**Platanios**, Blum, Mitchell, UAI 2014]

NOTE **BRAIN** is harder because the classifiers and the regions are highly dependent!

NELL

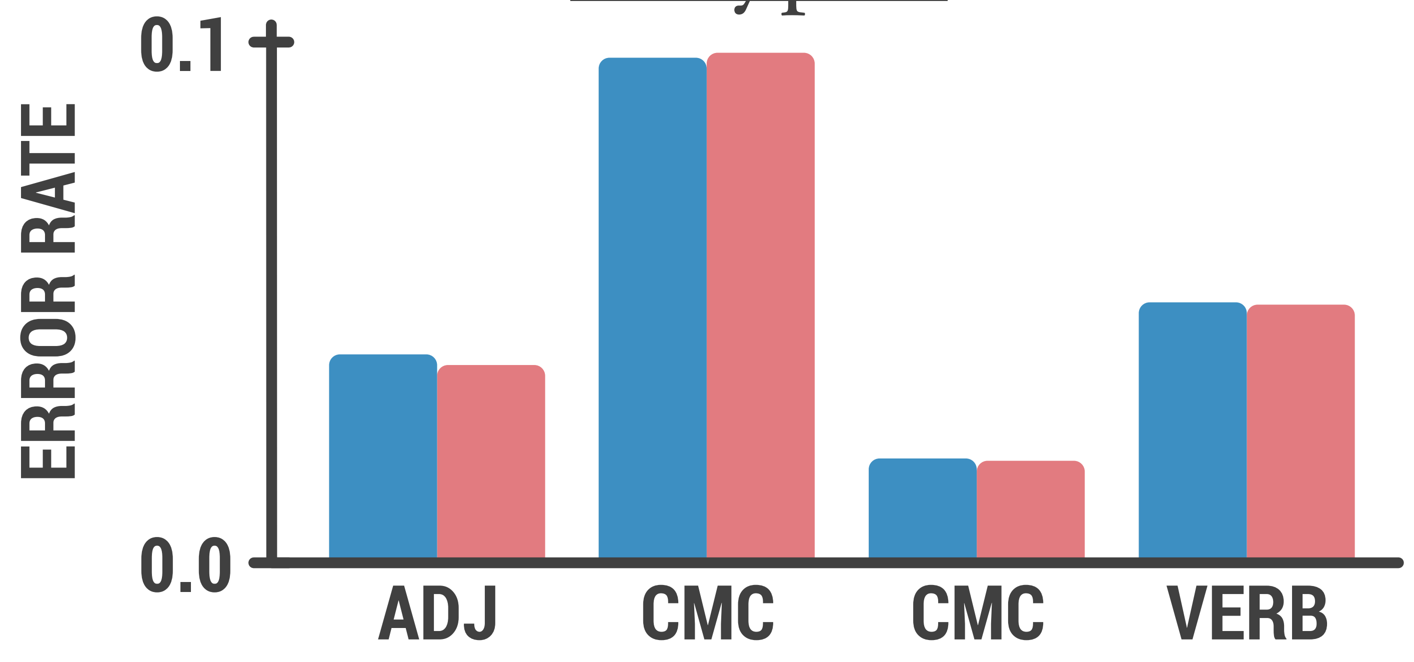
Task: Predict whether a noun phrase belongs to a category (e.g., `city`).

4 classifiers

15 categories

~300,000 noun phrases

bodypart



TRUE ERROR RATES   ESTIMATED ERROR RATES

BRAIN

Task: Find which of two 40 second long story passages corresponds to a time series of fMRI neural activity.

11 classifiers

11 brain regions

1,000 passages

[**Platanios**, Blum, Mitchell, UAI 2014]

NOTE **BRAIN** is harder because the classifiers and the regions are highly dependent!

NELL

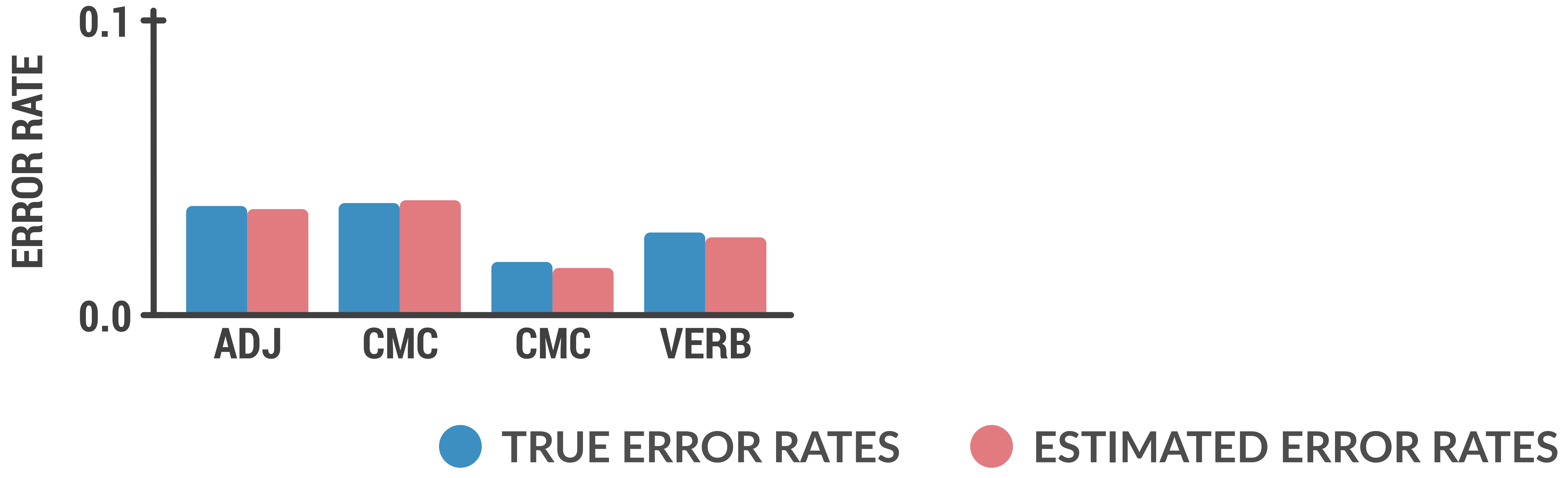
Task: Predict whether a noun phrase belongs to a category (e.g., `city`).

4 classifiers

15 categories

~300,000 noun phrases

bird



BRAIN

Task: Find which of two 40 second long story passages corresponds to a time series of fMRI neural activity.

11 classifiers

11 brain regions

1,000 passages

[**Platanios**, Blum, Mitchell, UAI 2014]

NOTE **BRAIN** is harder because the classifiers and the regions are highly dependent!

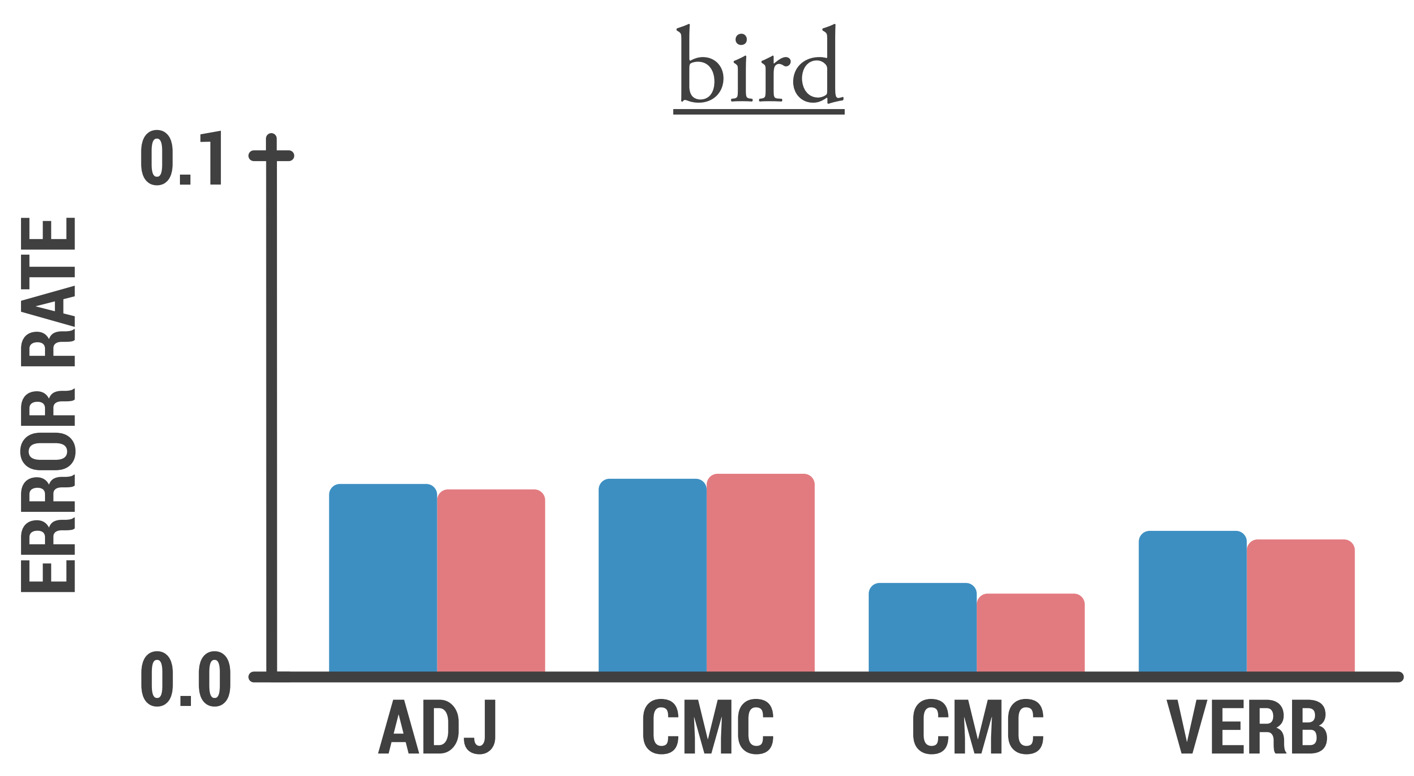
NELL

Task: Predict whether a noun phrase belongs to a category (e.g., `city`).

4 classifiers

15 categories

~300,000 noun phrases



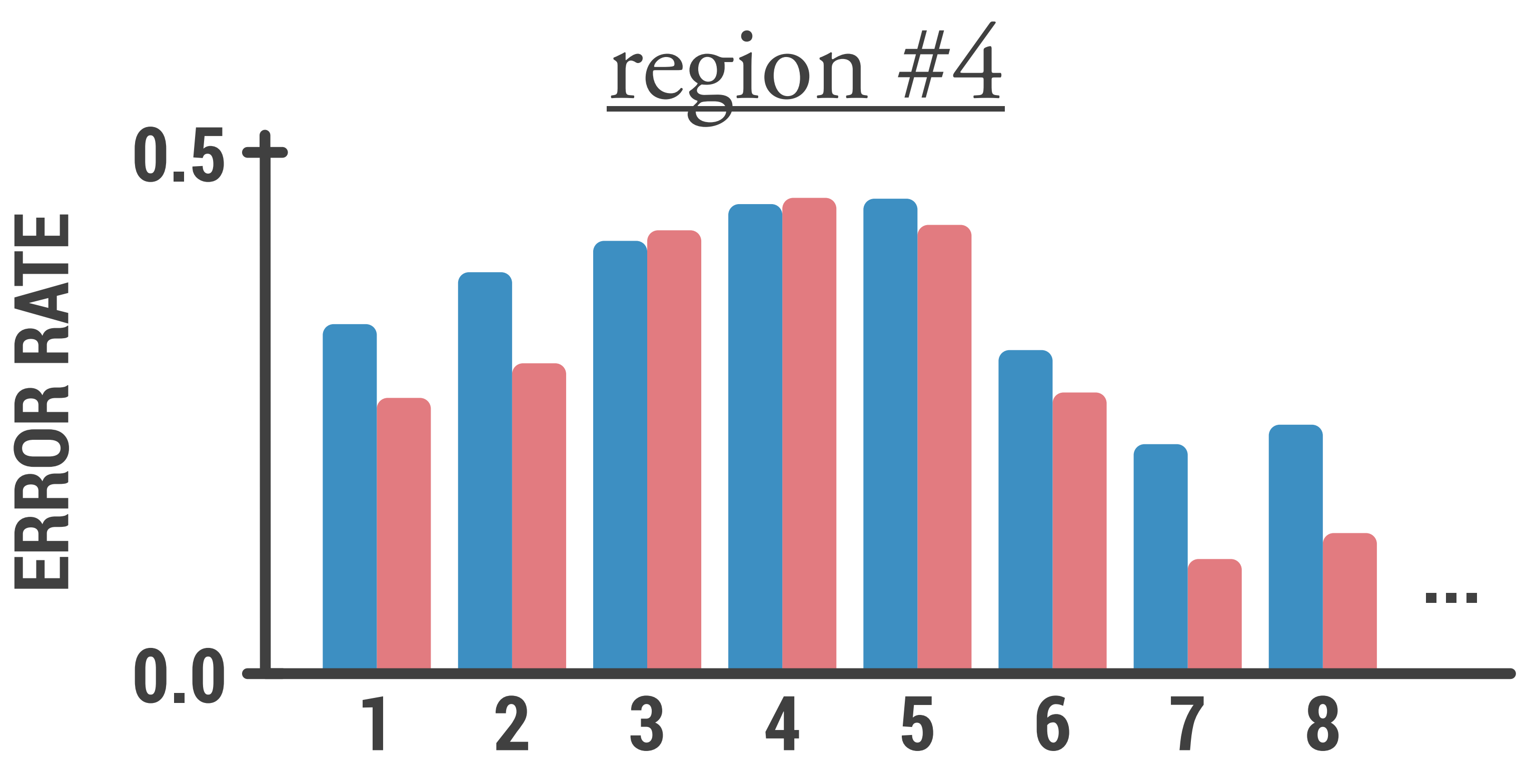
BRAIN

Task: Find which of two 40 second long story passages corresponds to a time series of fMRI neural activity.

11 classifiers

11 brain regions

1,000 passages



TRUE ERROR RATES ESTIMATED ERROR RATES

[**Platanios**, Blum, Mitchell, UAI 2014]

NOTE **BRAIN** is harder because the classifiers and the regions are highly dependent!

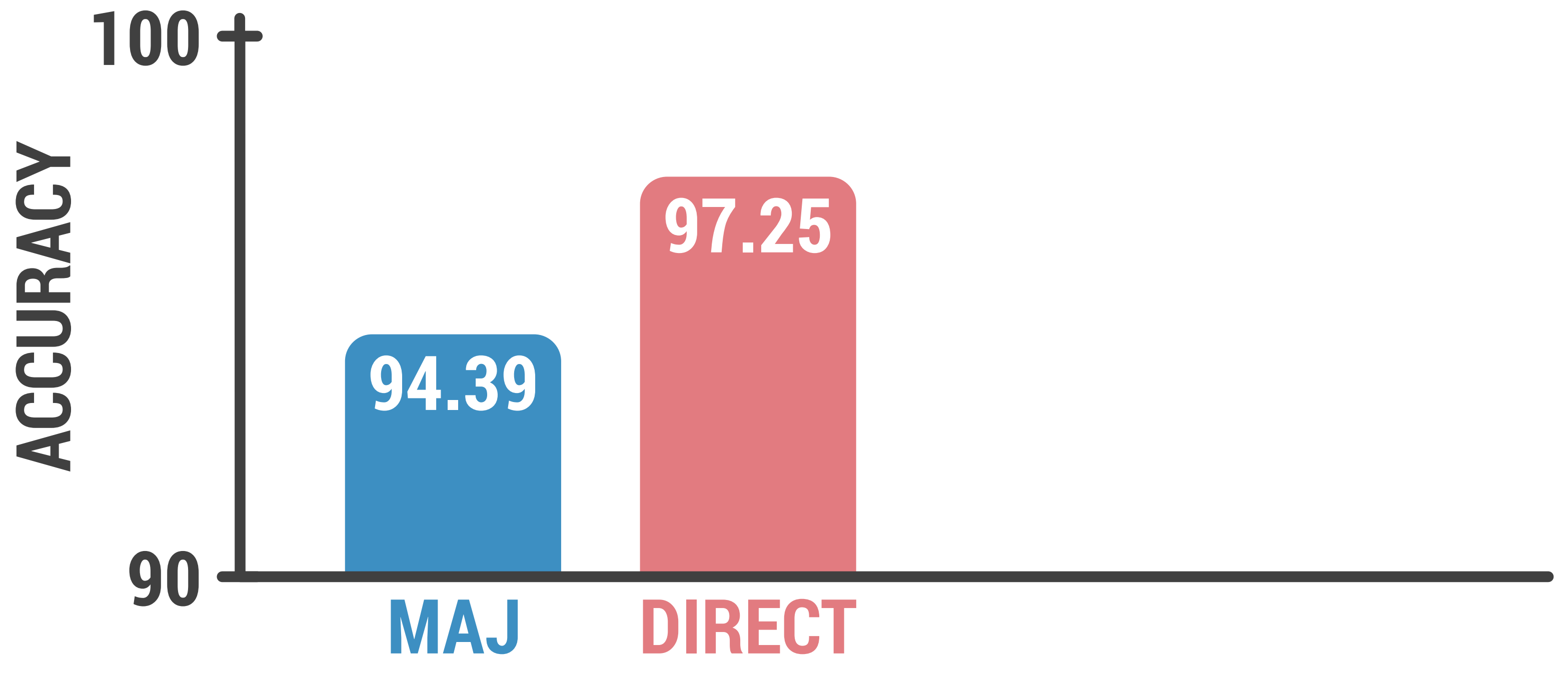
NELL

Task: Predict whether a noun phrase belongs to a category (e.g., `city`).

4 classifiers

15 categories

~300,000 noun phrases



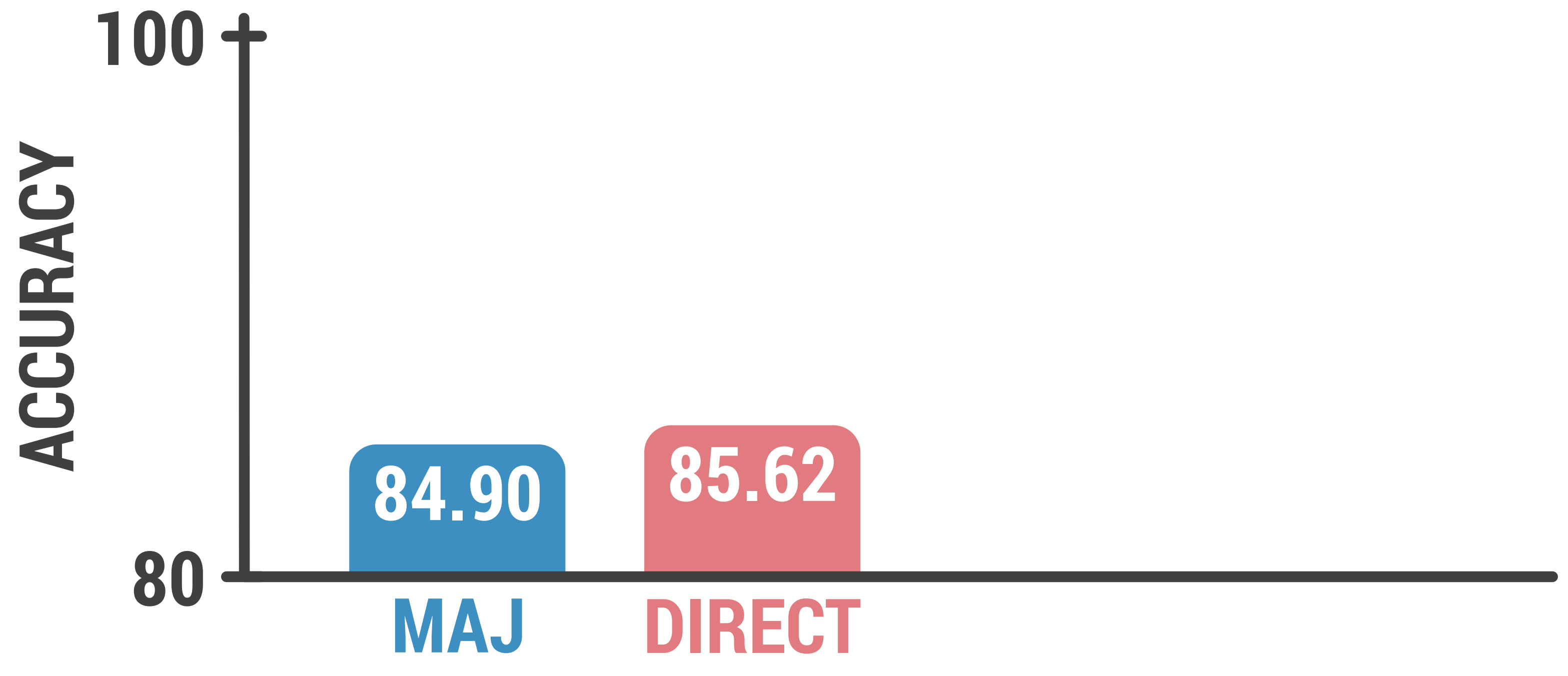
BRAIN

Task: Find which of two 40 second long story passages corresponds to a time series of fMRI neural activity.

11 classifiers

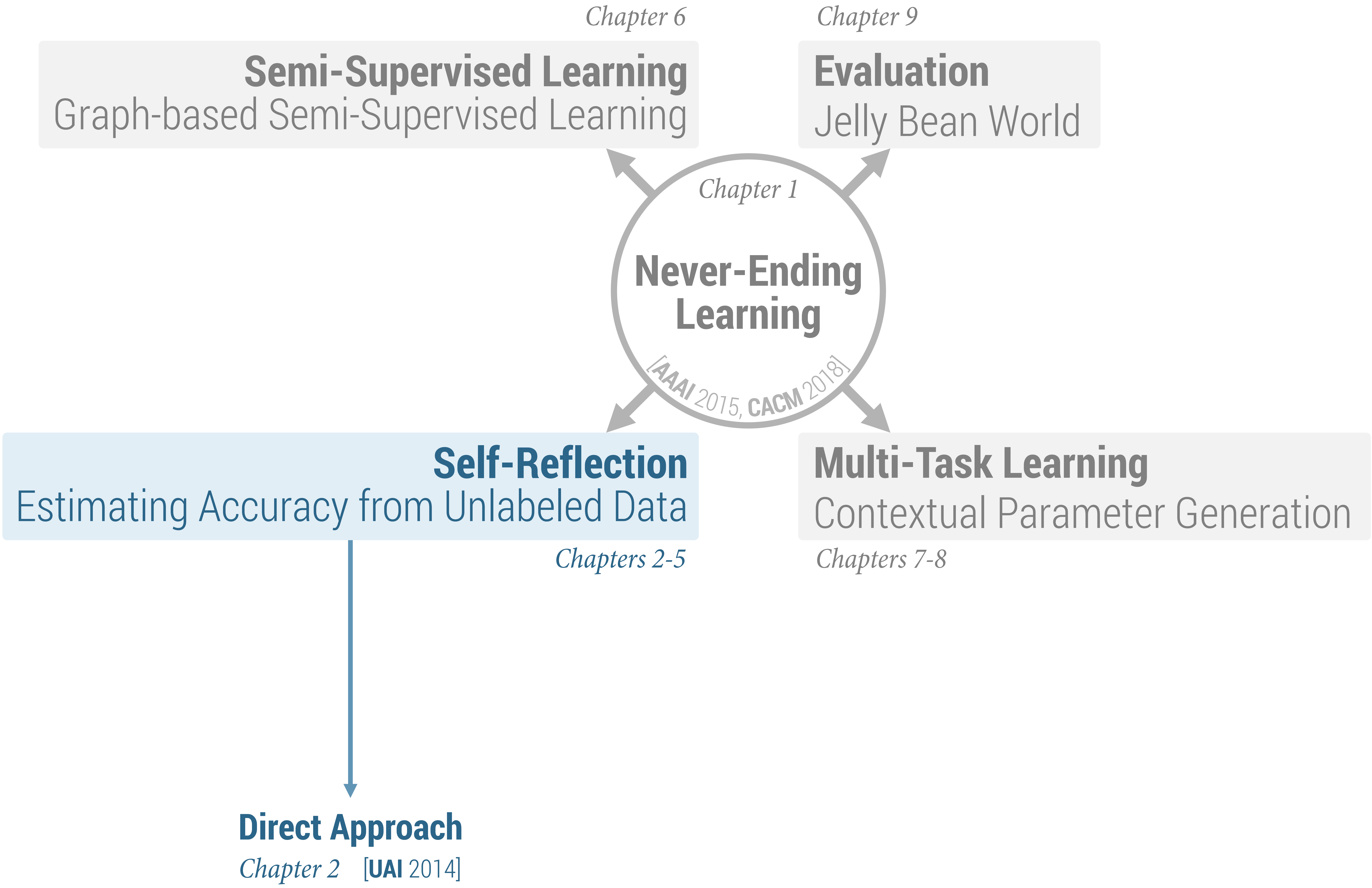
11 brain regions

1,000 passages



[**Platanios**, Blum, Mitchell, UAI 2014]

Self-Reflection



Self-Reflection

Self-Reflection
Estimating Accuracy from Unlabeled Data
Chapters 2-5

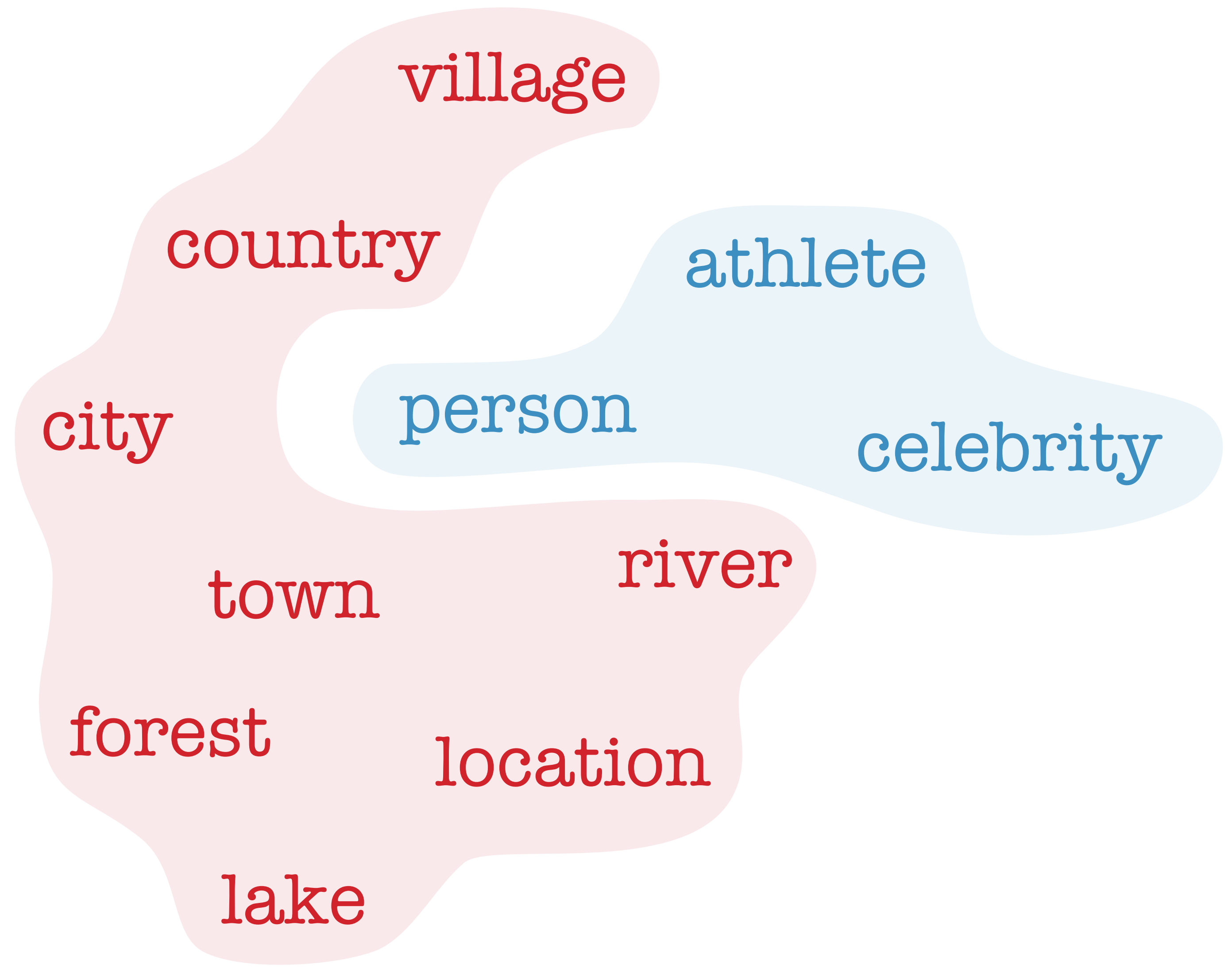
Direct Approach
Chapter 2 [**UAI** 2014]

+ dependencies

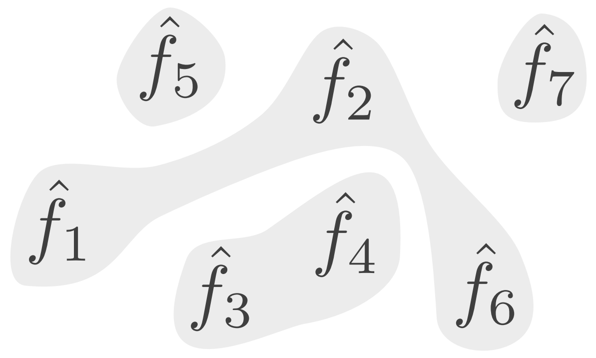
Bayesian Approach
Chapter 3 [**ICML** 2016]

Limitation #1: Dependencies

among tasks:

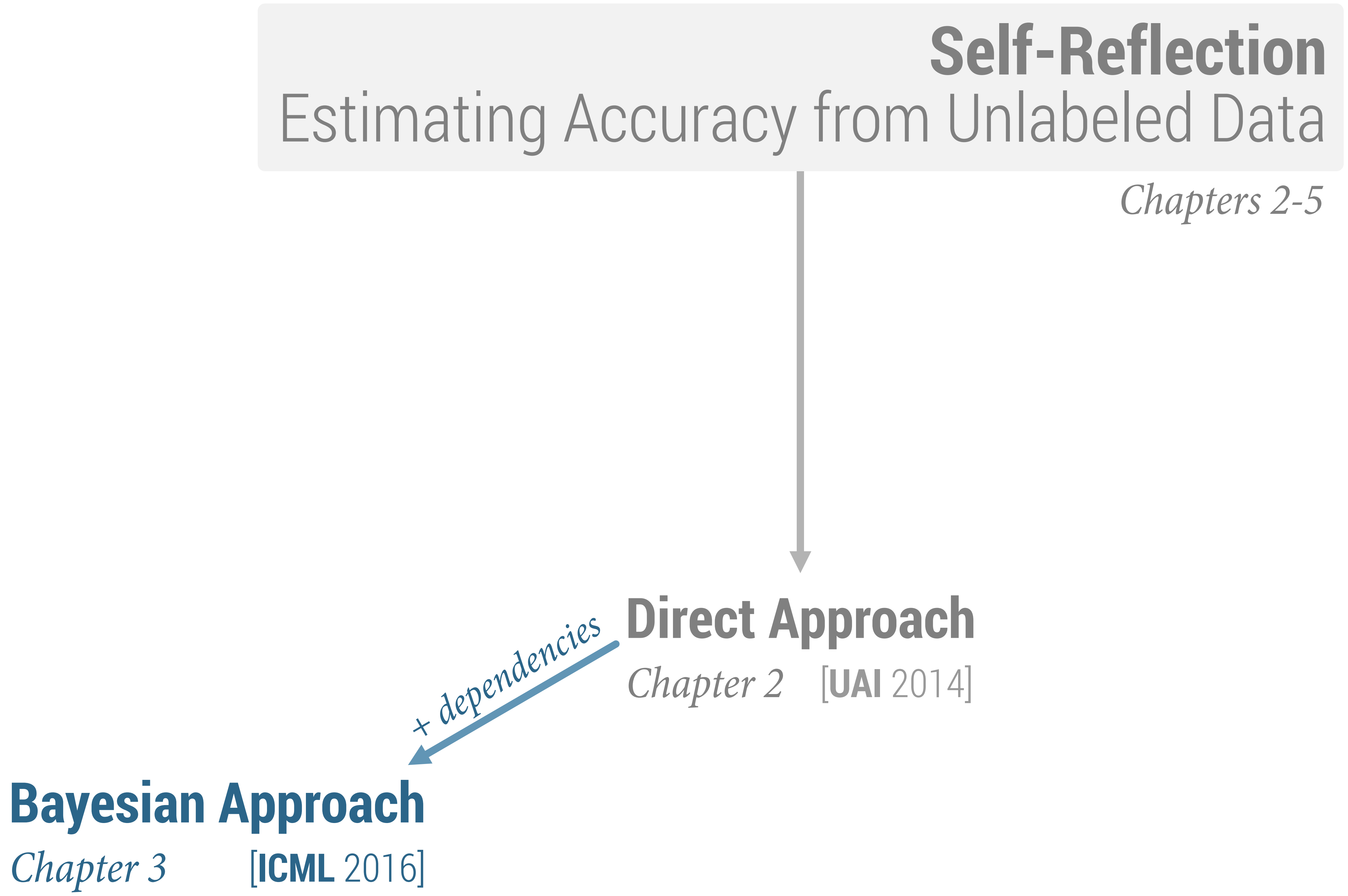


and among functions:



We can represent them using a *Bayesian model* with a *non-parametric clustering prior* that may be *hierarchical.*

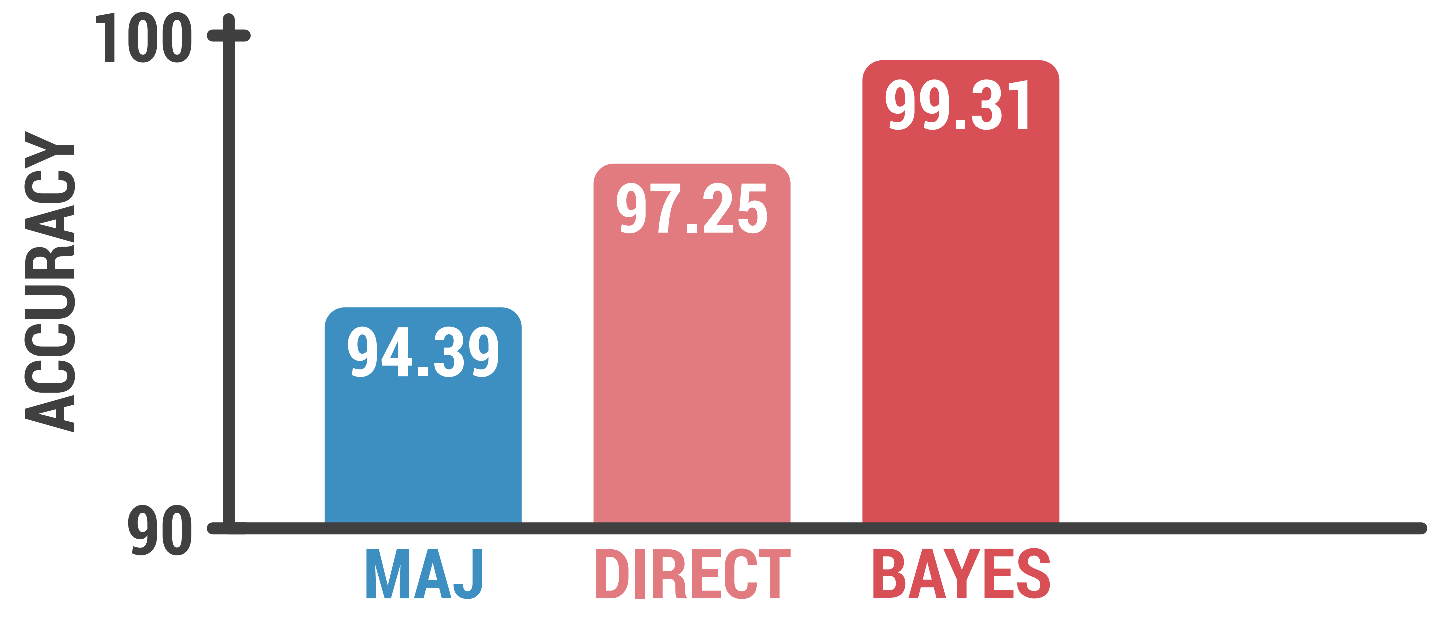
Self-Reflection



Limitation #1: Dependencies

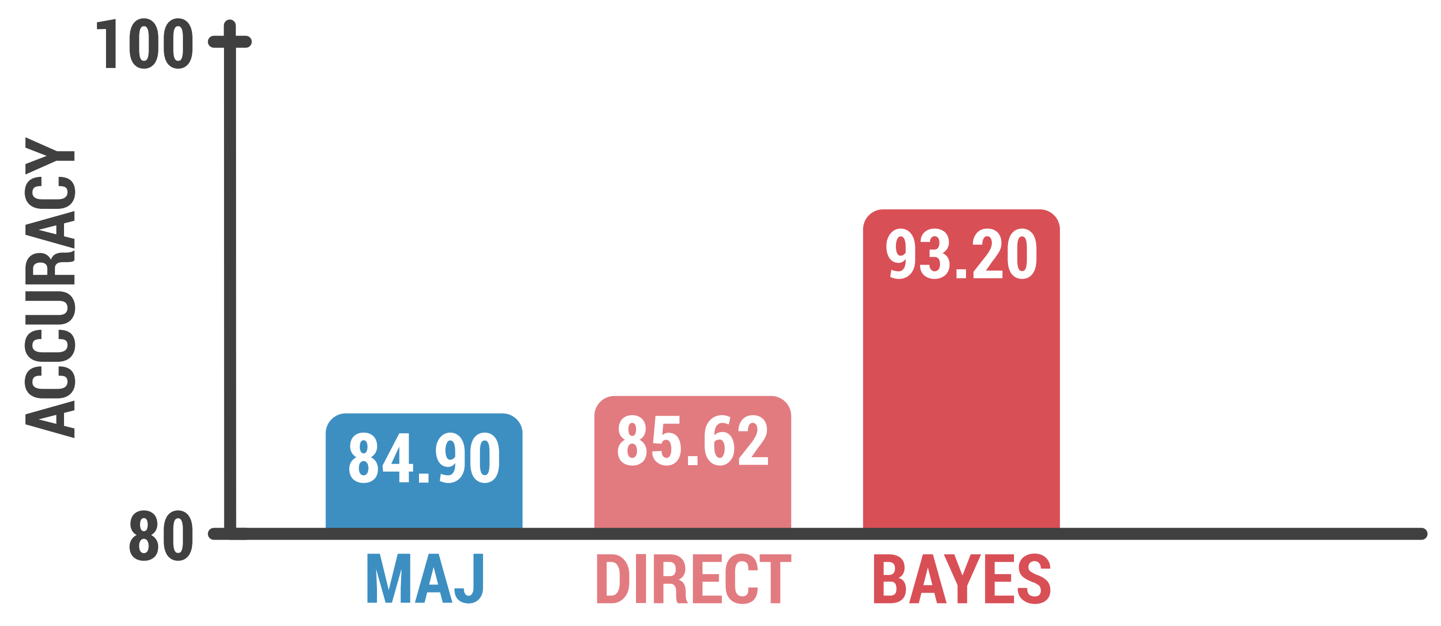
NELL

4 classifiers | 15 categories | ~300,000 noun phrases

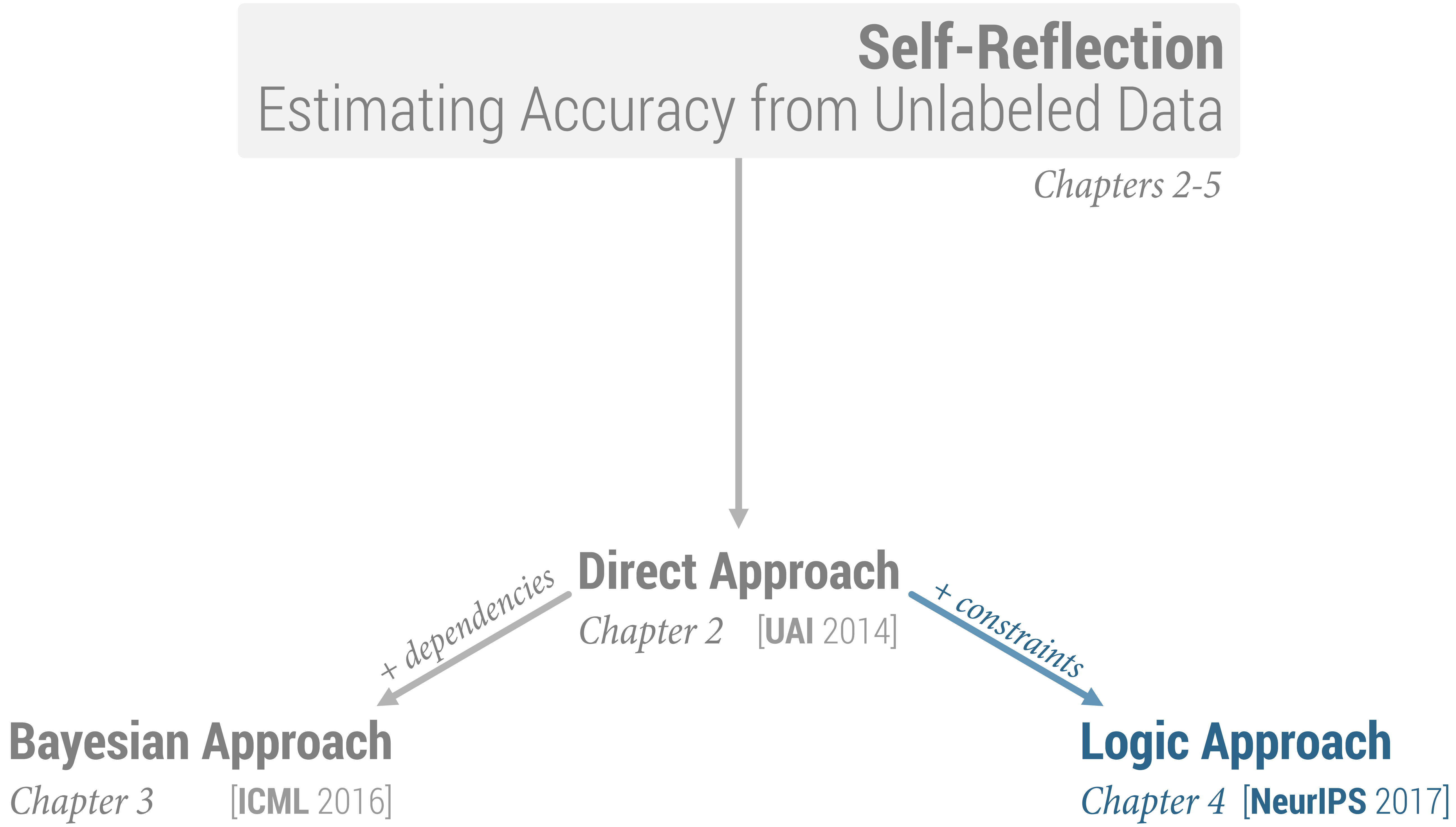


BRAIN

11 classifiers | 11 brain regions | 1,000 passages

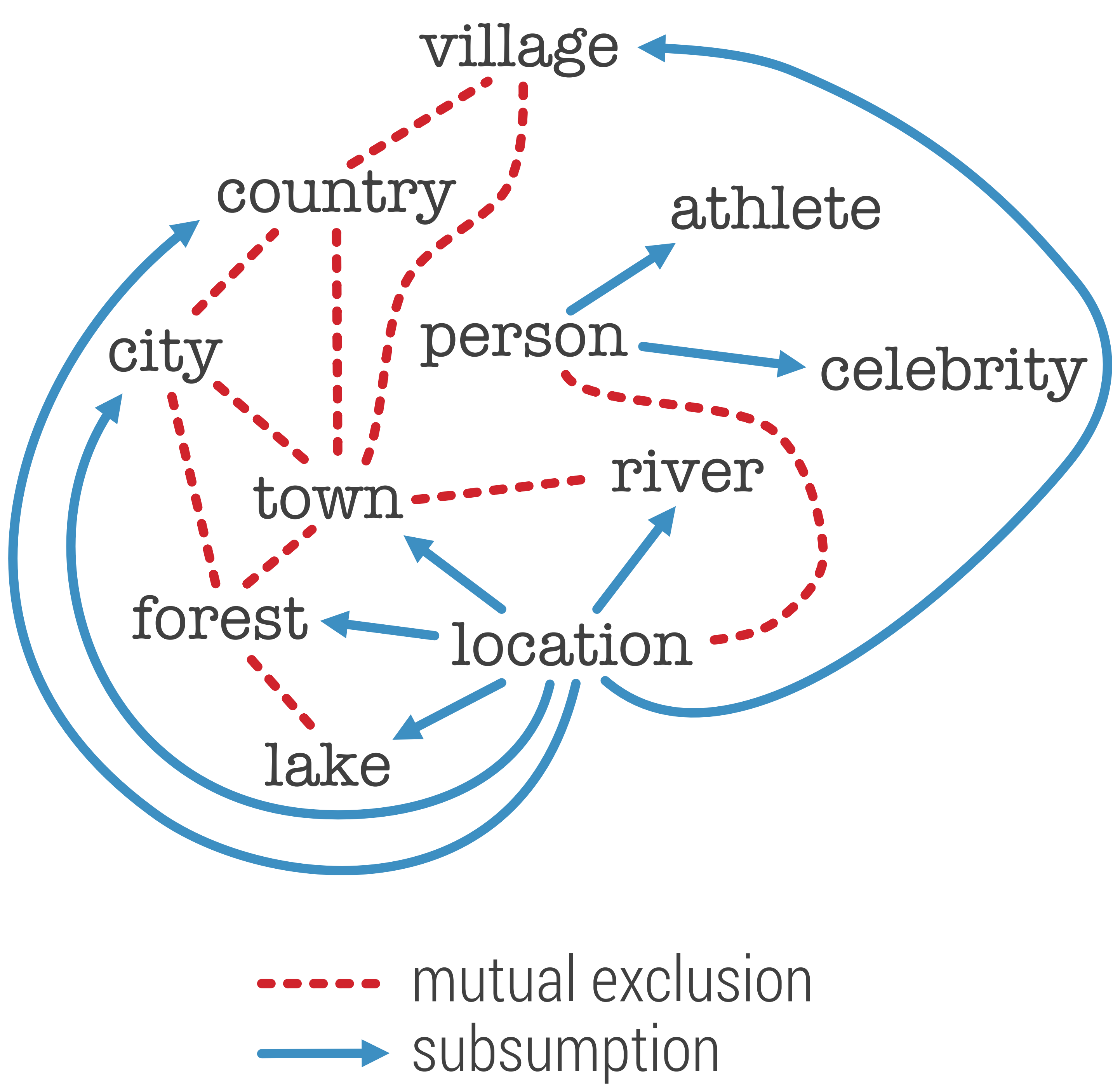


Self-Reflection



Limitation #2: Logical Constraints

between tasks:



We can represent them using a *probabilistic logic framework*. We use *probabilistic soft logic (PSL)* to obtain a scalable method.

previously unable to run on GPU server
now runs in ~1 hour on a MacBook Pro

Self-Reflection

Self-Reflection
Estimating Accuracy from Unlabeled Data
Chapters 2-5

Direct Approach
Chapter 2 [**UAI** 2014]

+ dependencies

Bayesian Approach
Chapter 3 [**ICML** 2016]

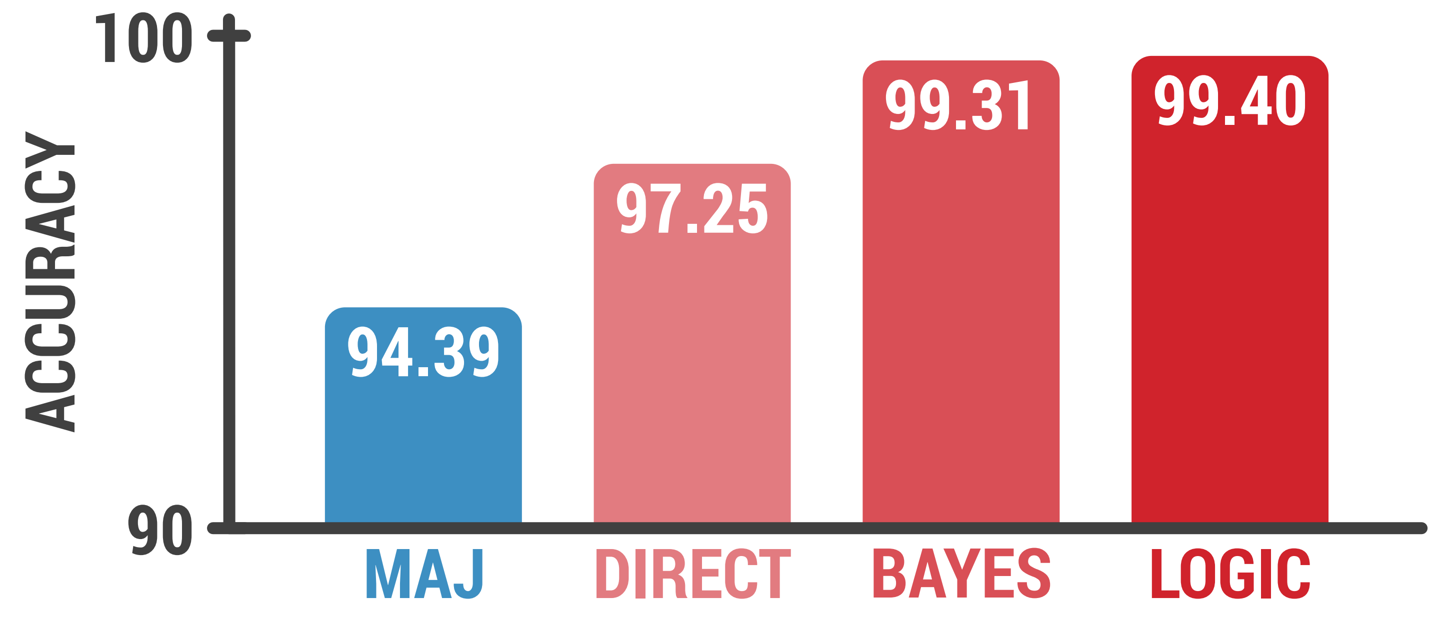
+ constraints

Logic Approach
Chapter 4 [**NeurIPS** 2017]

Limitation #2: Logical Constraints

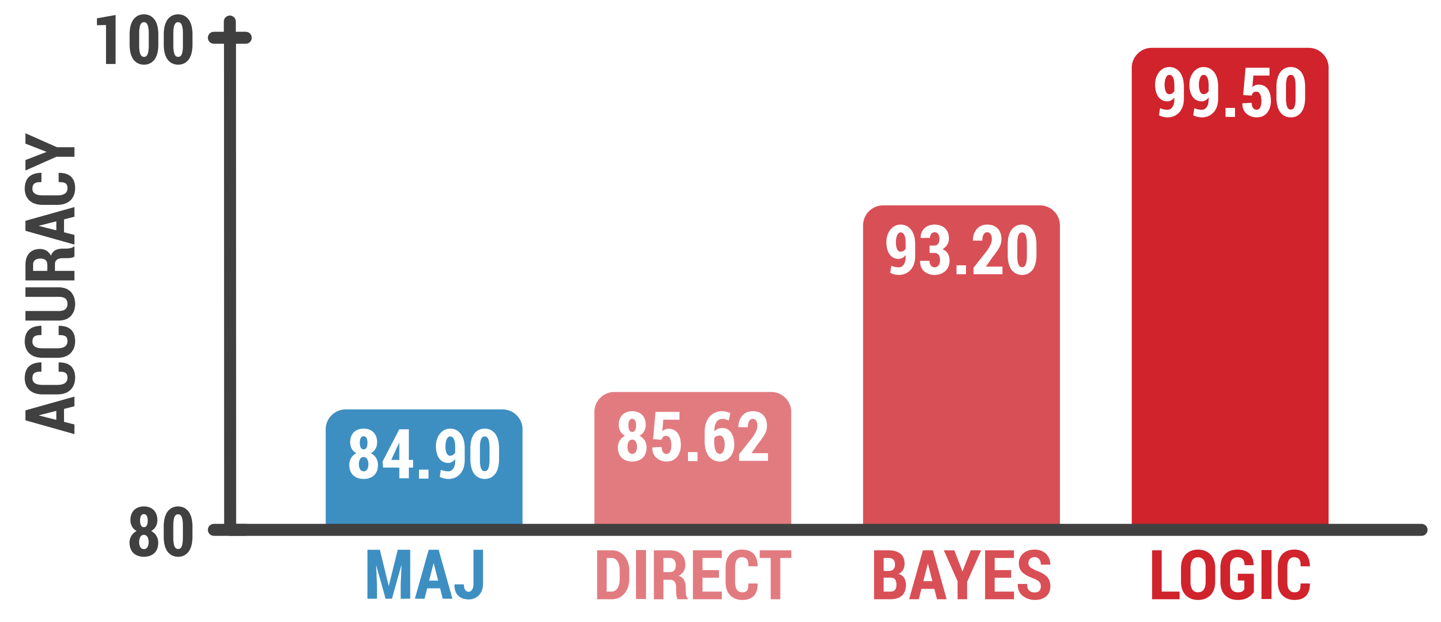
NELL

4 classifiers | 15 categories | ~300,000 noun phrases



BRAIN

11 classifiers | 11 brain regions | 1,000 passages



Self-Reflection

Self-Reflection
Estimating Accuracy from Unlabeled Data
Chapters 2-5

Direct Approach
Chapter 2 [**UAI** 2014]

+ dependencies

Bayesian Approach
Chapter 3 [**ICML** 2016]

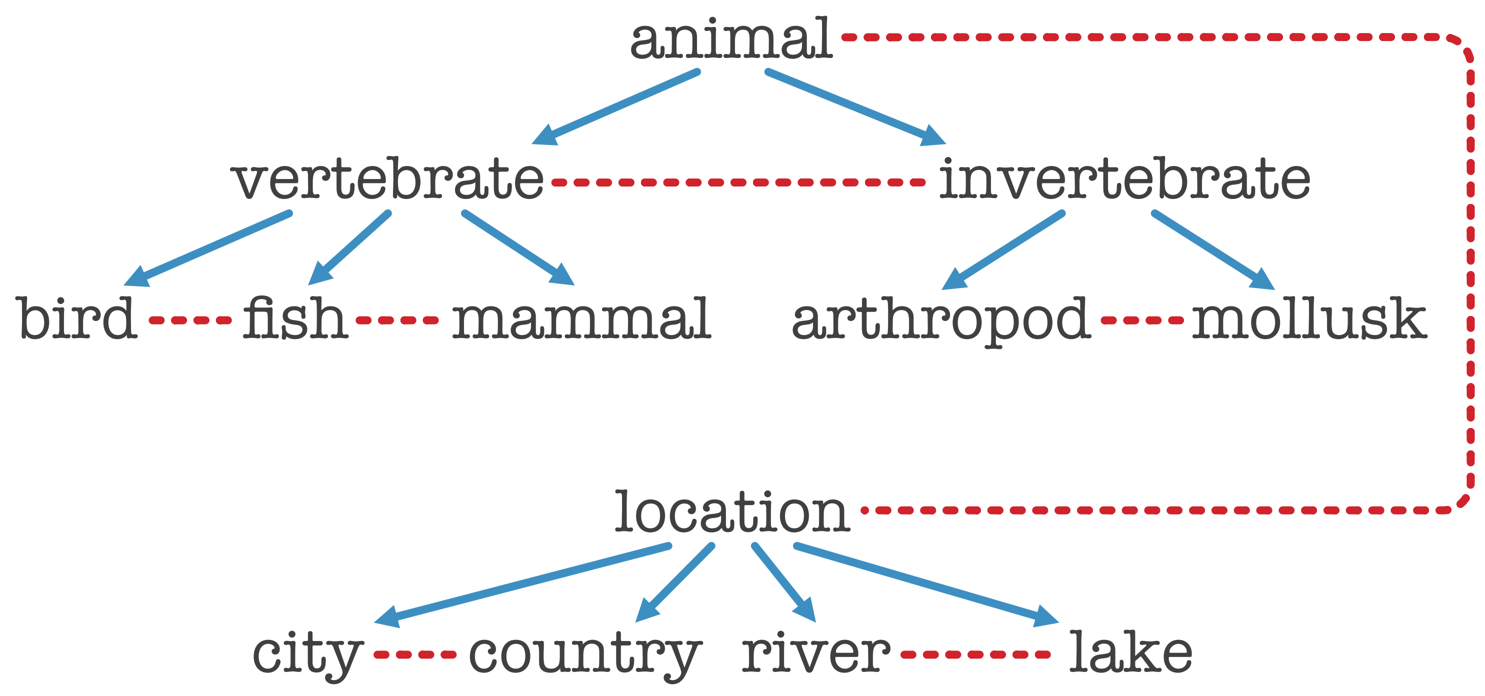
+ constraints

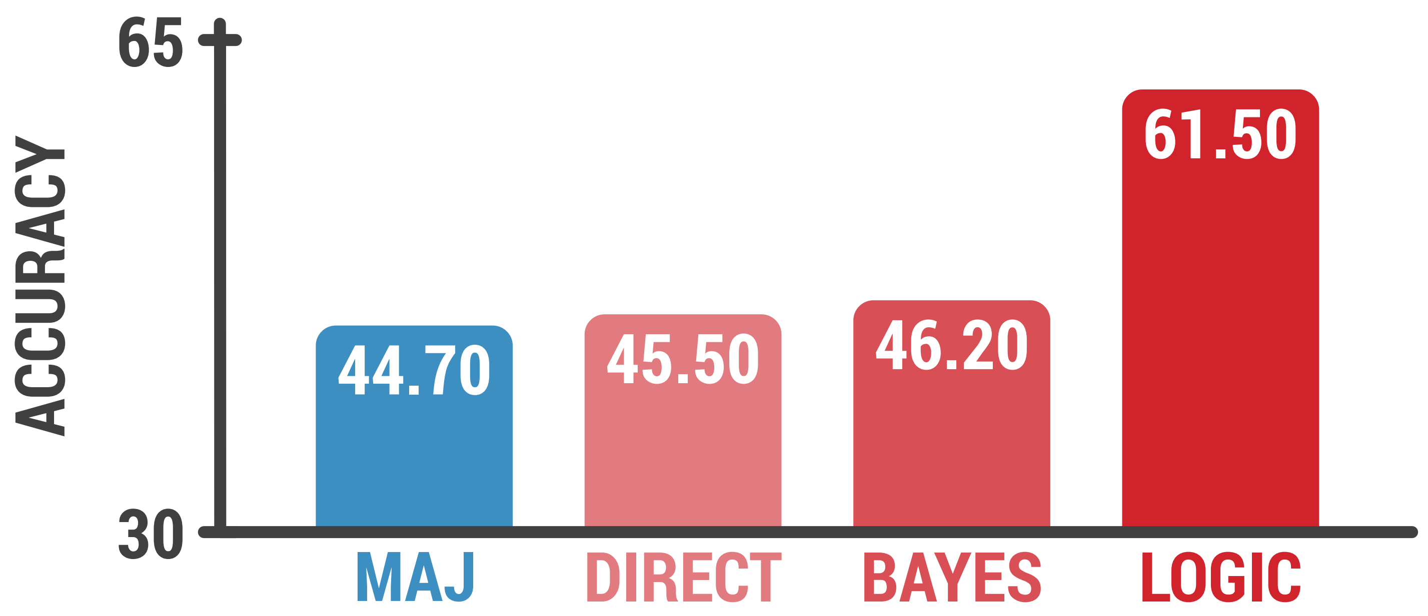
Logic Approach
Chapter 4 [**NeurIPS** 2017]

Limitation #2: Logical Constraints

NELL

6 classifiers | 15 categories | ~550,000 noun phrases





Self-Reflection

Self-Reflection
Estimating Accuracy from Unlabeled Data
Chapters 2-5

Active Learning
Appendix B

+ dependencies

Direct Approach
Chapter 2 [**UAI** 2014]

+ constraints

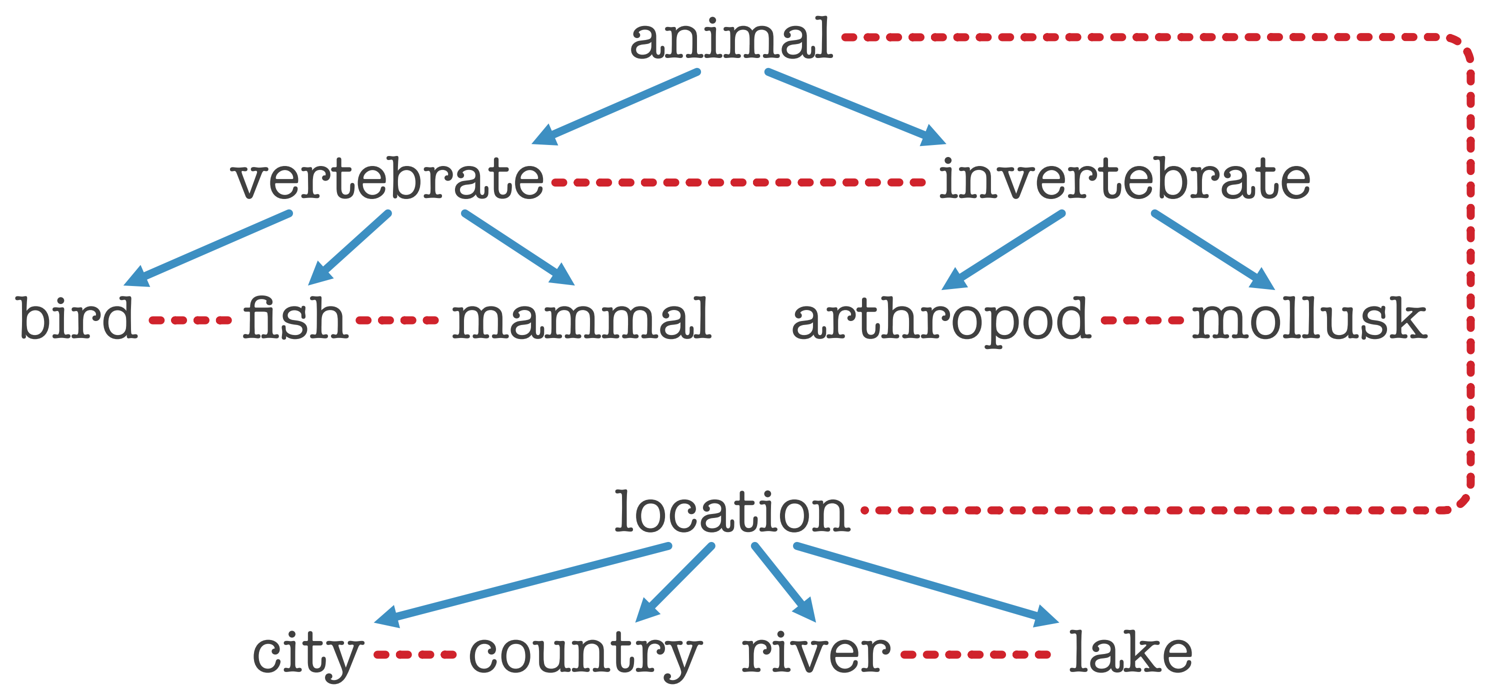
Bayesian Approach
Chapter 3 [**ICML** 2016]

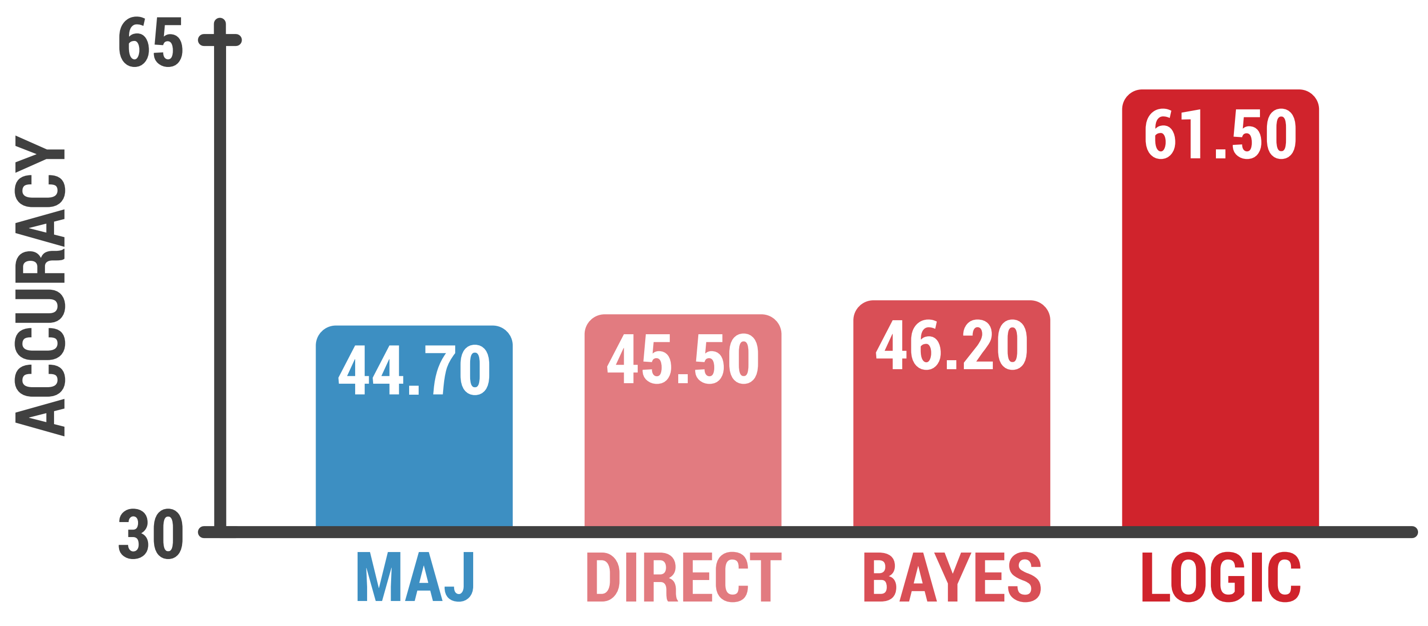
Logic Approach
Chapter 4 [**NeurIPS** 2017]

Limitation #2: Logical Constraints

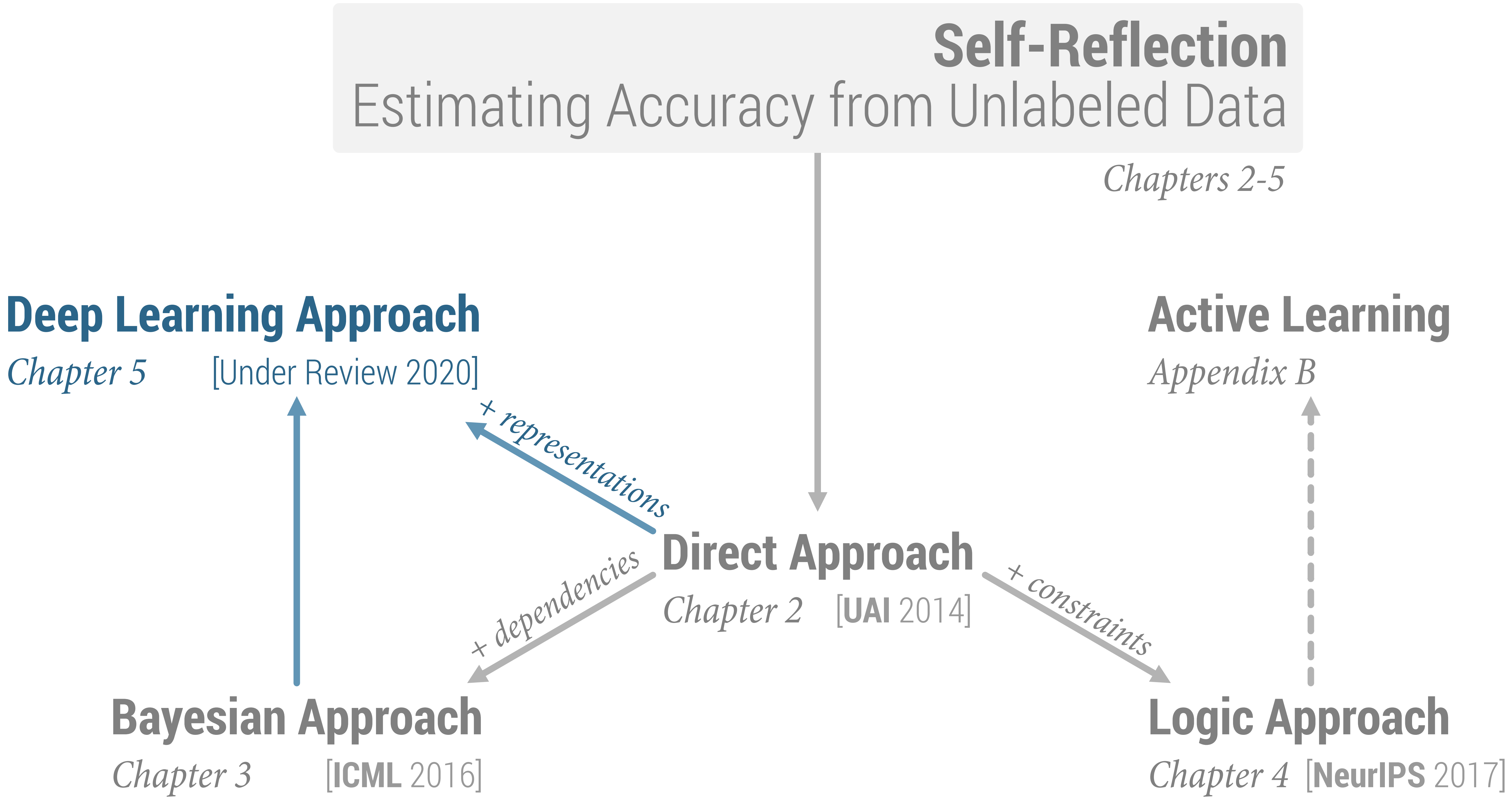
NELL

6 classifiers | 15 categories | ~550,000 noun phrases



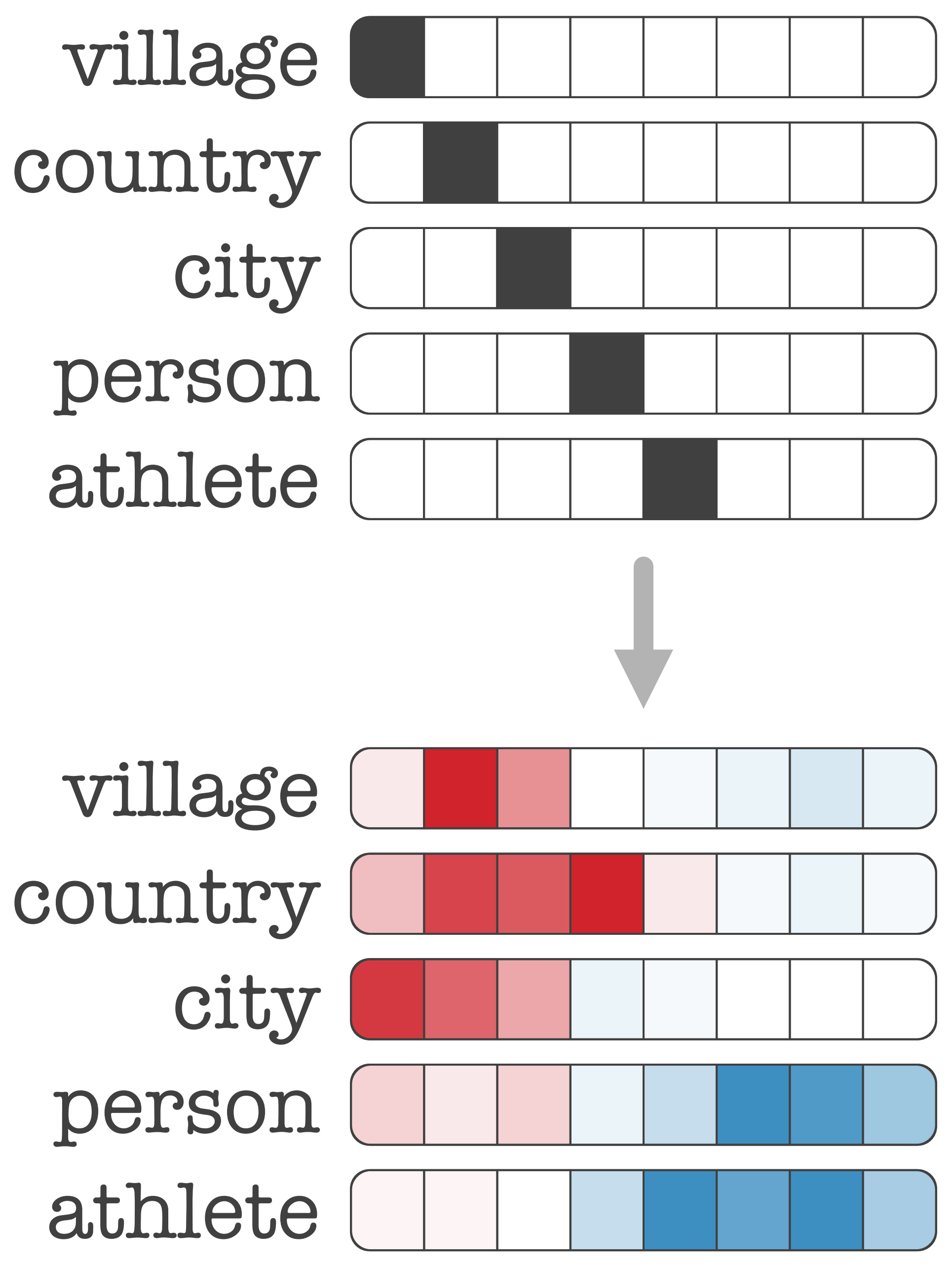


Self-Reflection



Limitation #3: Representations

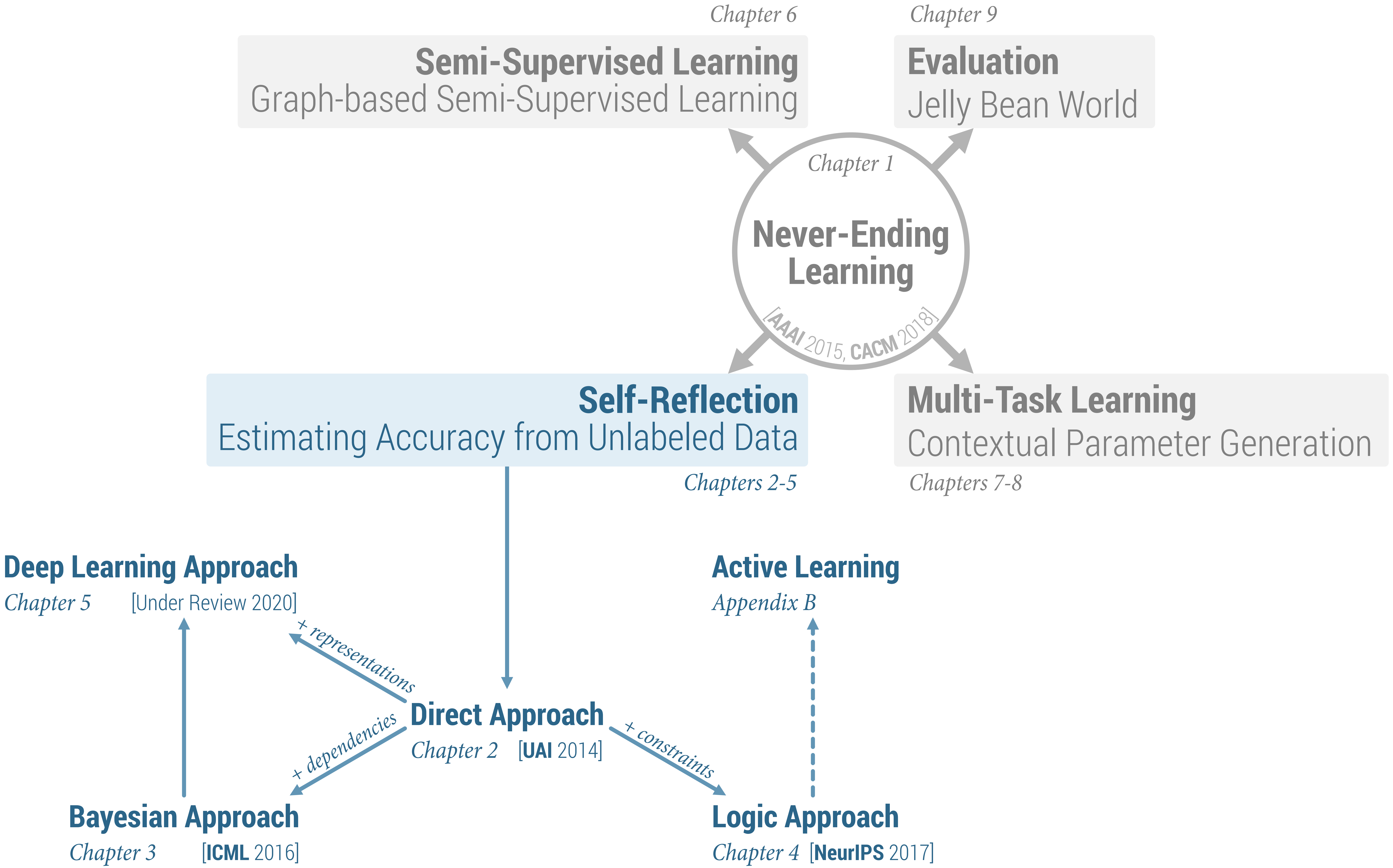
of tasks:



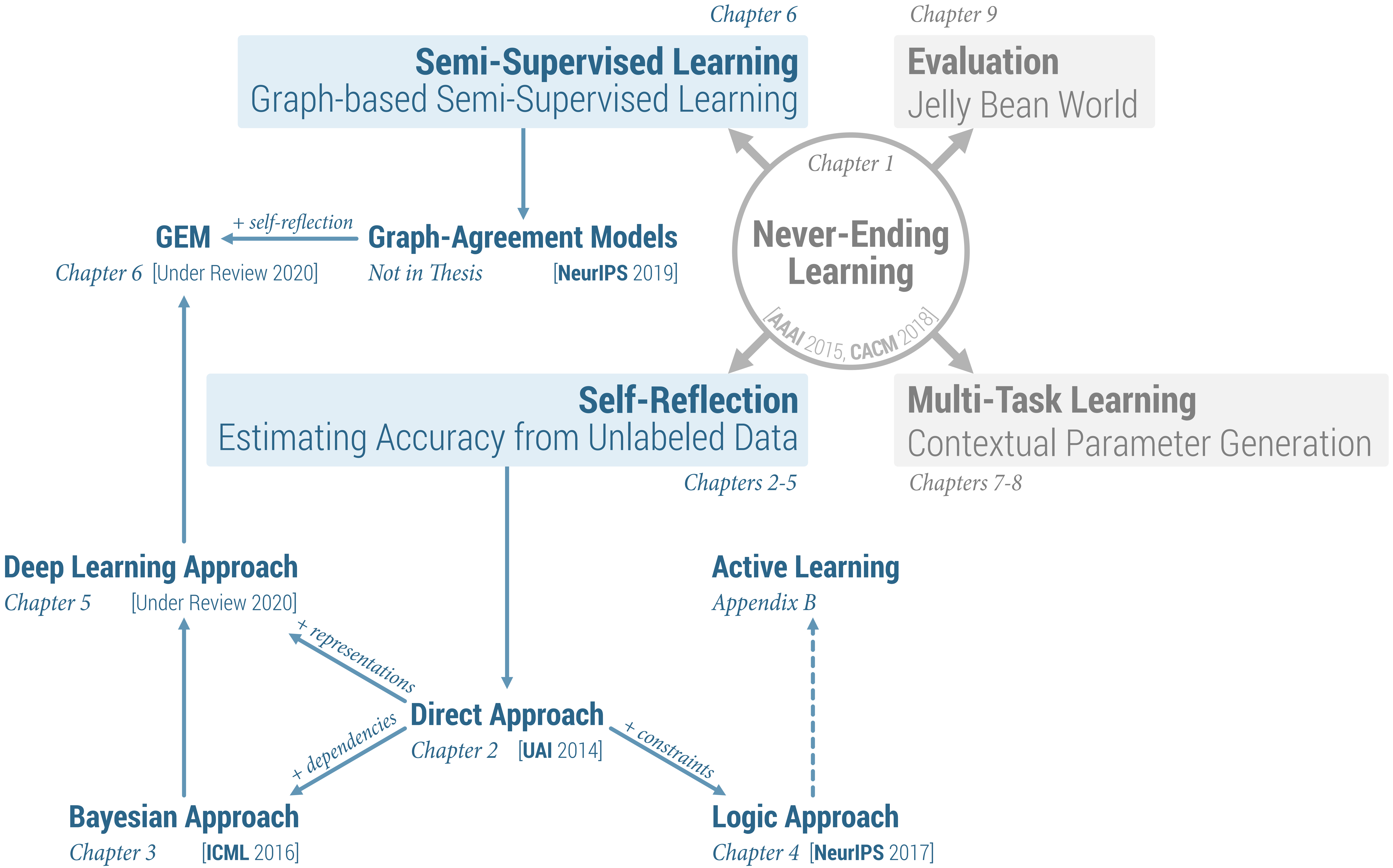
and of instances and functions.

We can use *deep learning*.

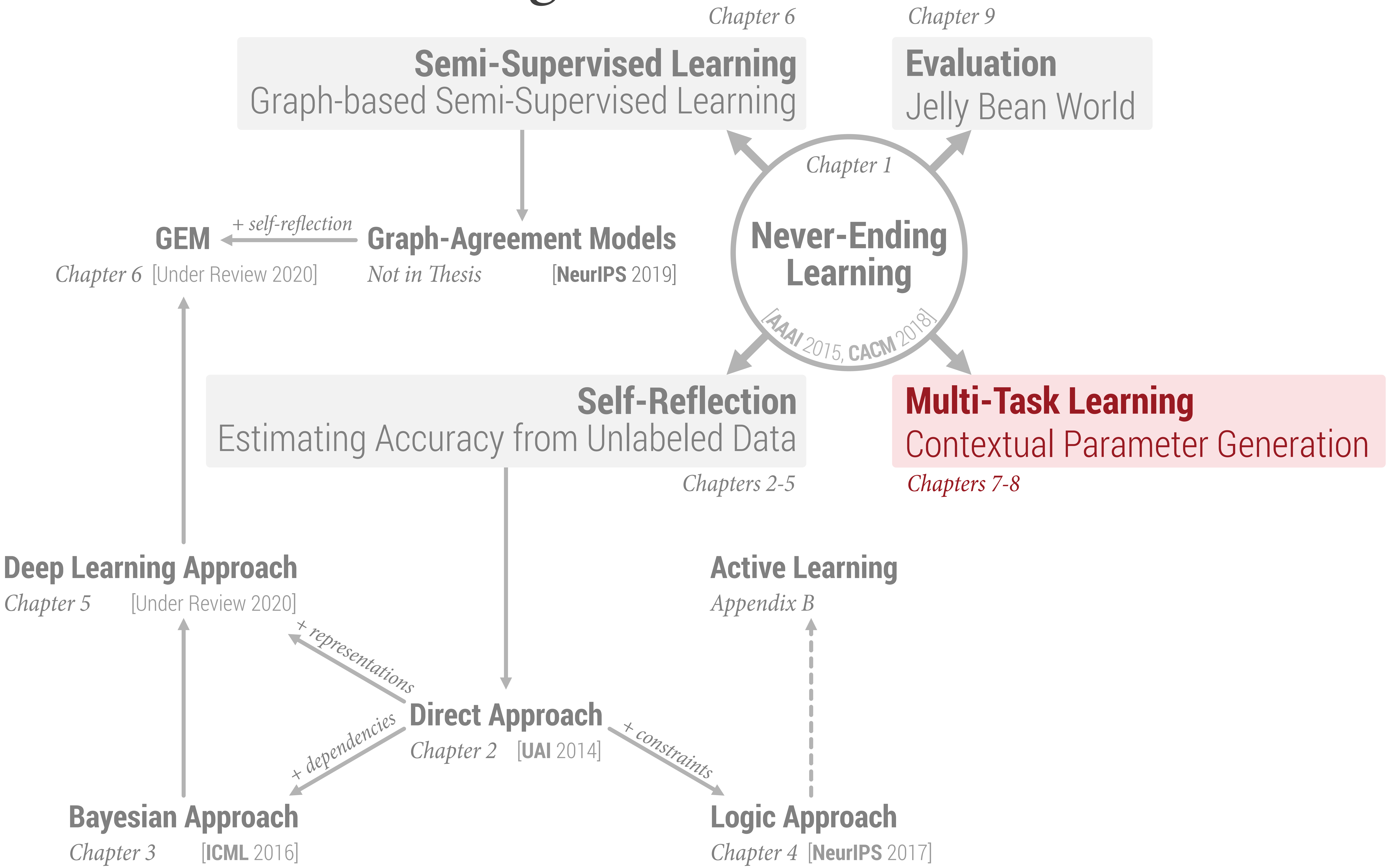
Self-Reflection



Self-Reflection



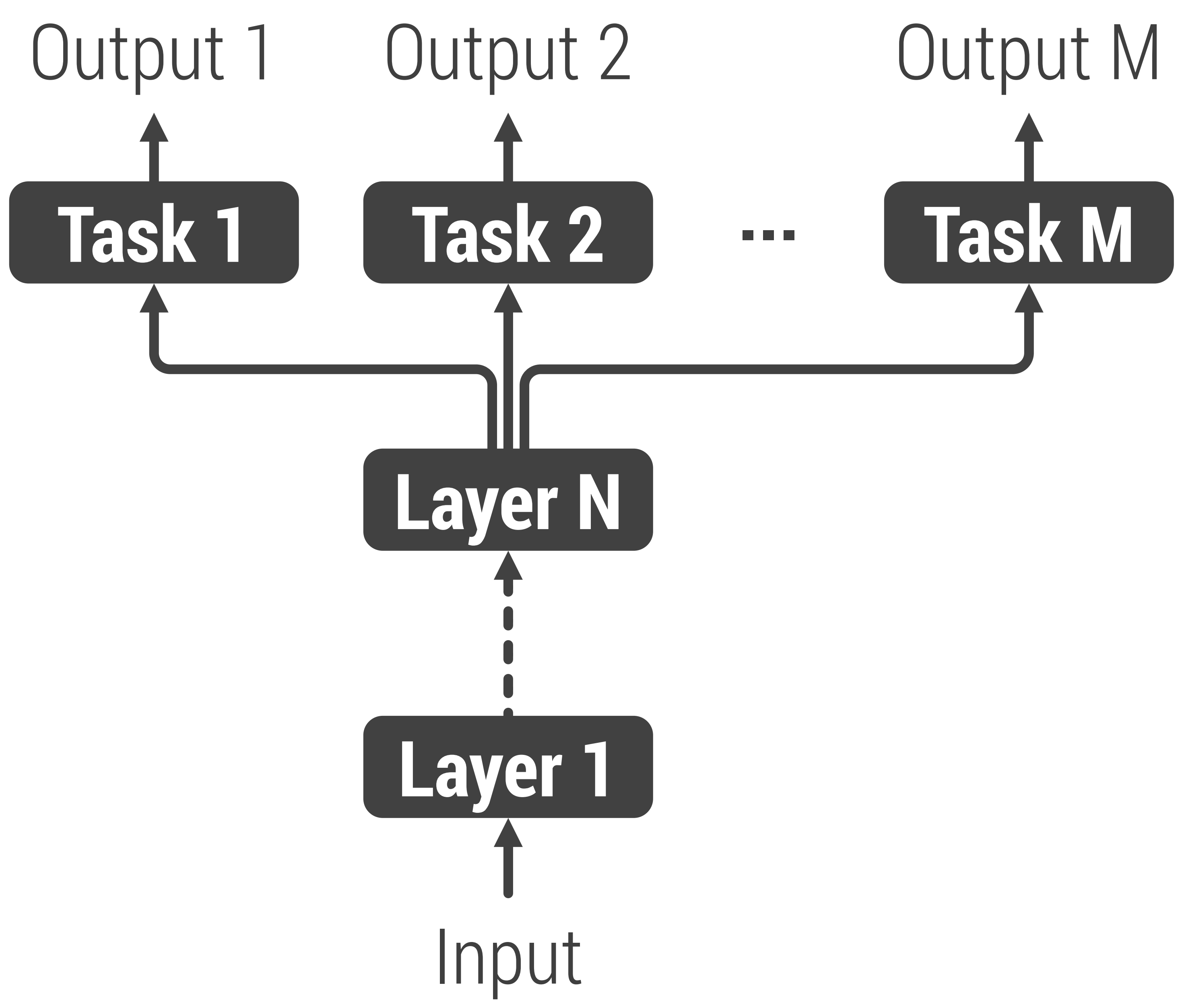
Multi-Task Learning



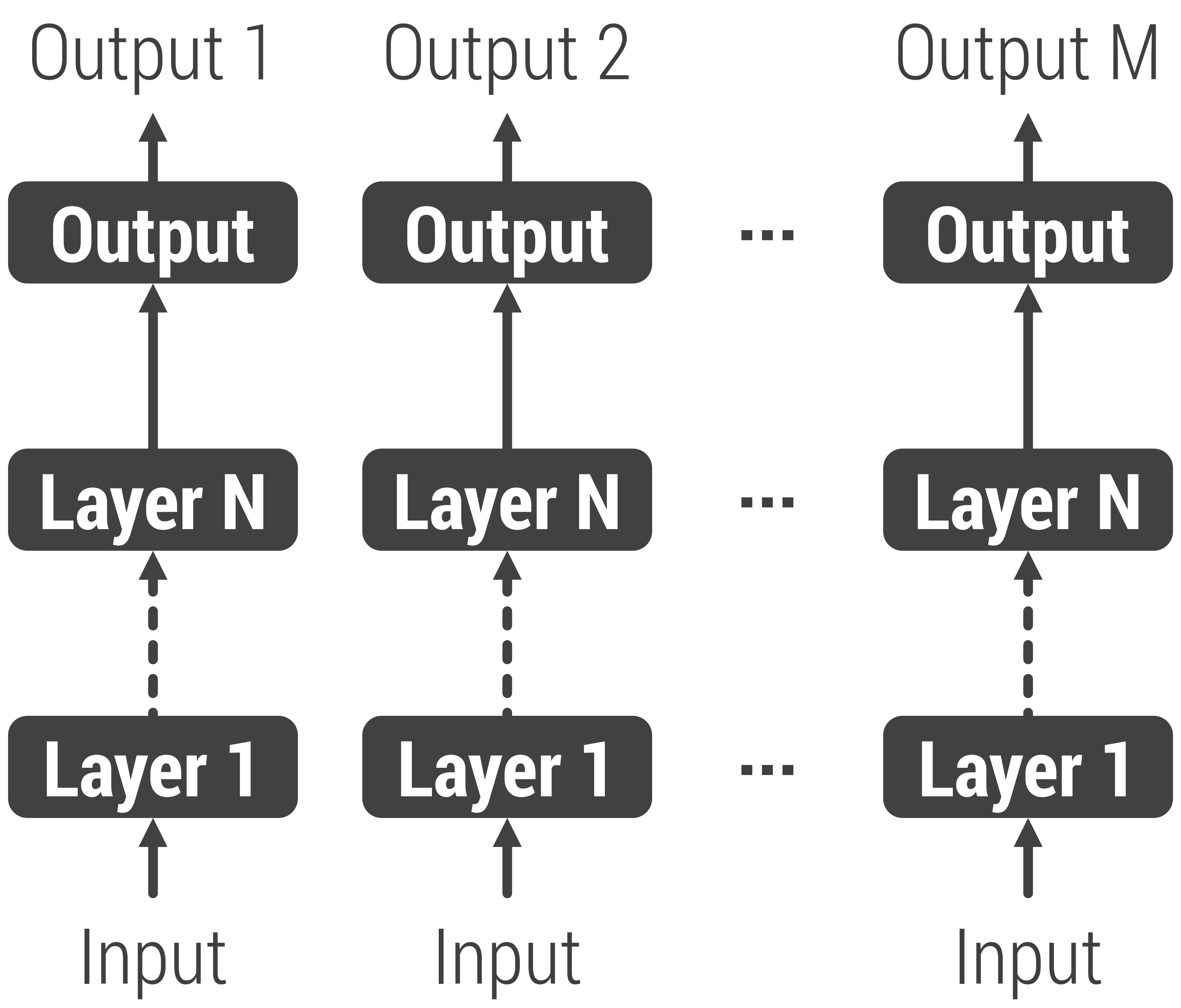
Multi-Task Learning

Multi-task learning is currently performed in one of two ways:

Hard Parameter Sharing



Soft Parameter Sharing



Parameters have similar values

Multi-Task Learning

Multi-task learning is currently performed in one of two ways:

Hard Parameter Sharing	**Soft Parameter Sharing**

A never-ending learning system must support learning tasks that keep changing.

Hard Parameter Sharing	Soft Parameter Sharing
- Adding new tasks requires *re-training*.	- Adding new tasks requires *copying/re-training*.
- Highly prone to *negative transfer*.	- Hard to have tasks interact in a constructive manner — tasks typically only interact through *pre-training*.

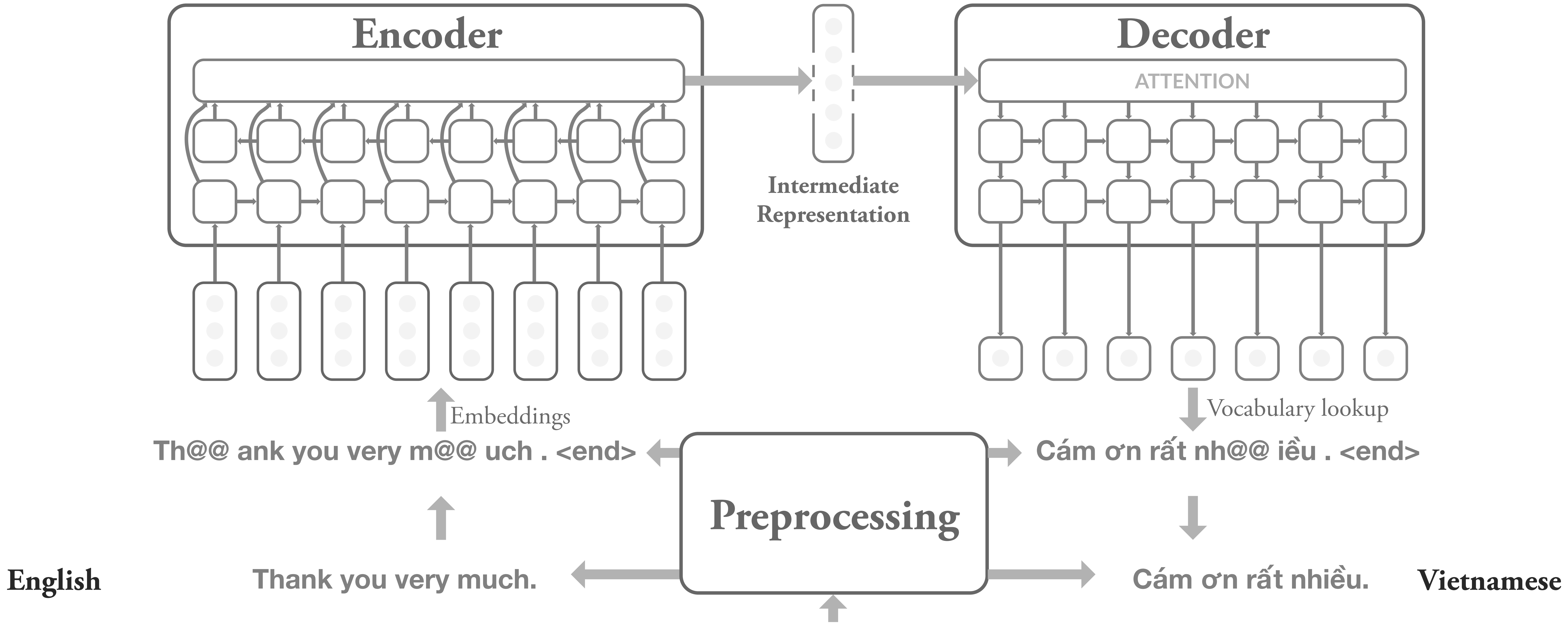
Do not allow for **task dependencies** at inference time (e.g., task composition) or for **zero-shot learning**!

Multi-Task Learning

	Hard Sharing	Soft Sharing
Avoids copying/re-training	✗	✗
Avoids negative transfer	✗	✓
Enables positive transfer	✓	✗
Enables task dependencies	✗	✗
Enables zero-shot learning	✗	✗
Enables fast adaptation	✗	✗

What if we learn *task representations* and feed them as *inputs*?

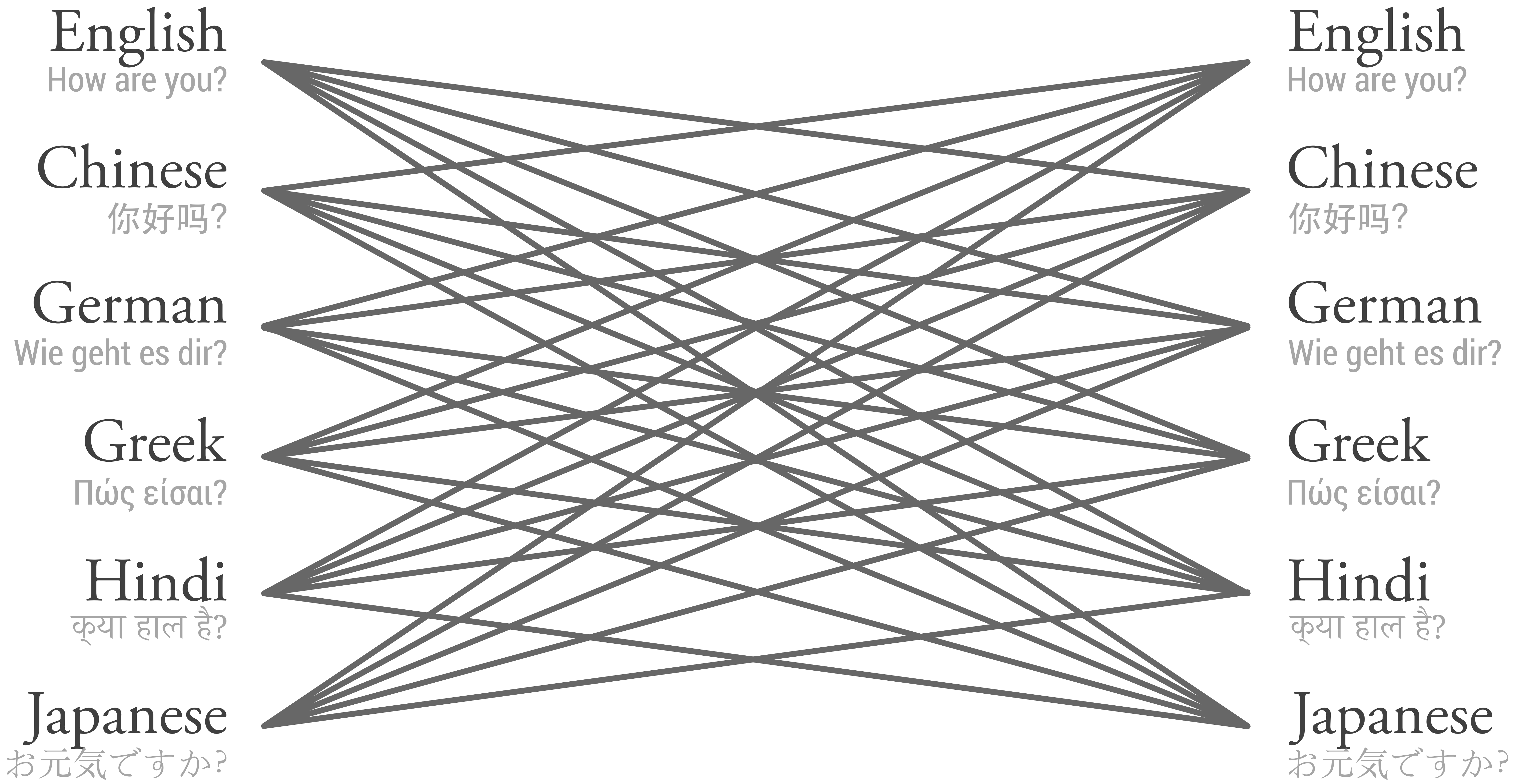
There have been some attempts. For example, in **machine translation (MT)**.



Multi-Task Learning

Contextual Parameter Generation

What about *multilingual translation?*

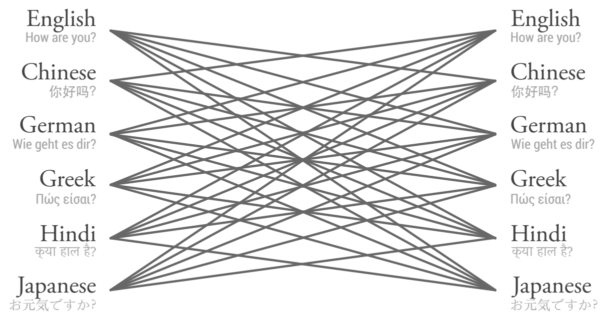


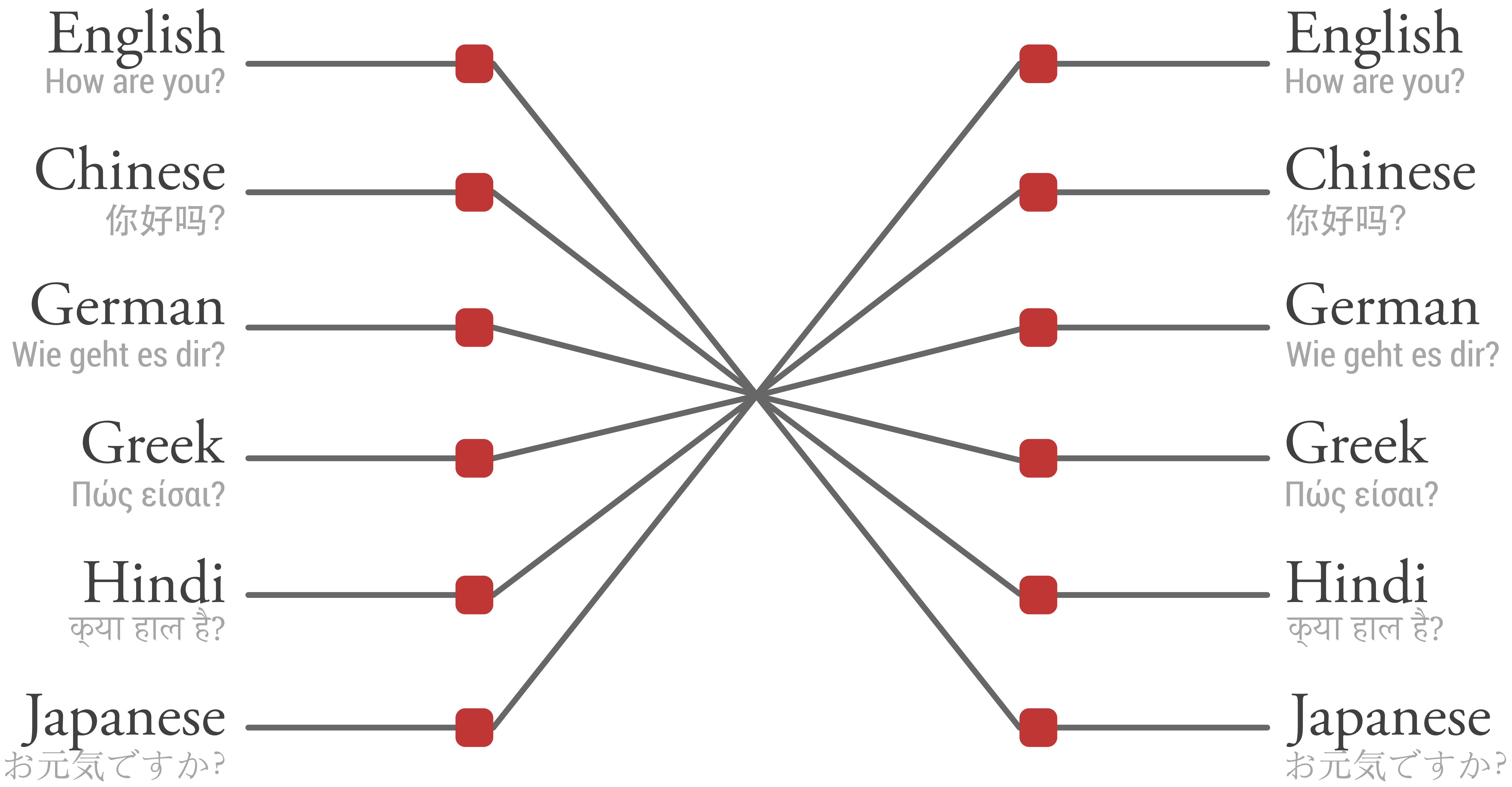
No Parameter Sharing

Multi-Task Learning

What about *multilingual translation?*

Pairwise





Hard Parameter Sharing

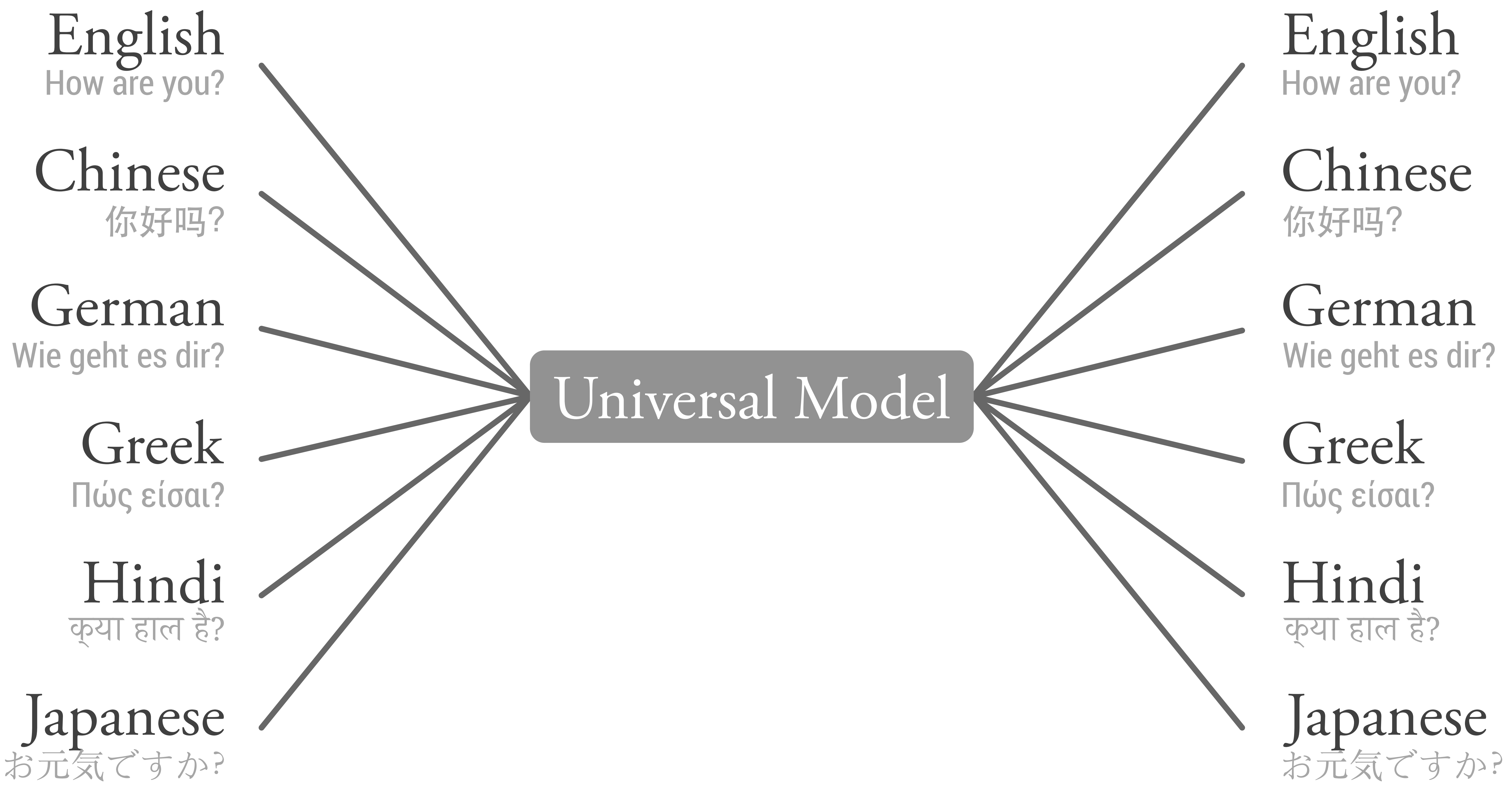
[Firat, Cho, Bengio, NAACL 2016 & Luong, Le, Sutskever, Vinyals, Kaizer, ICLR 2016]

Multi-Task Learning

What about *multilingual translation?*

Pairwise

Per-Language Encoder/Decoder

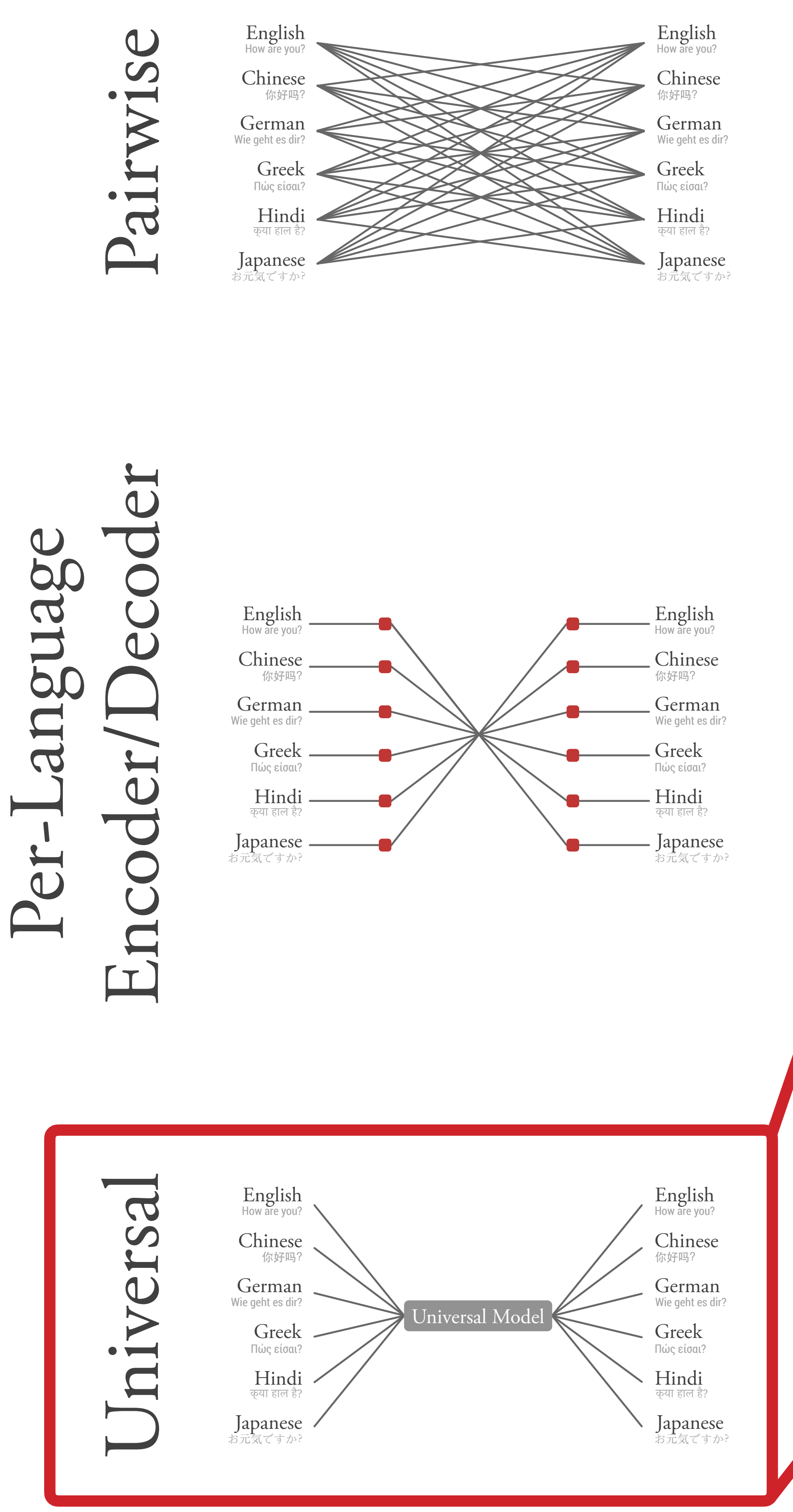


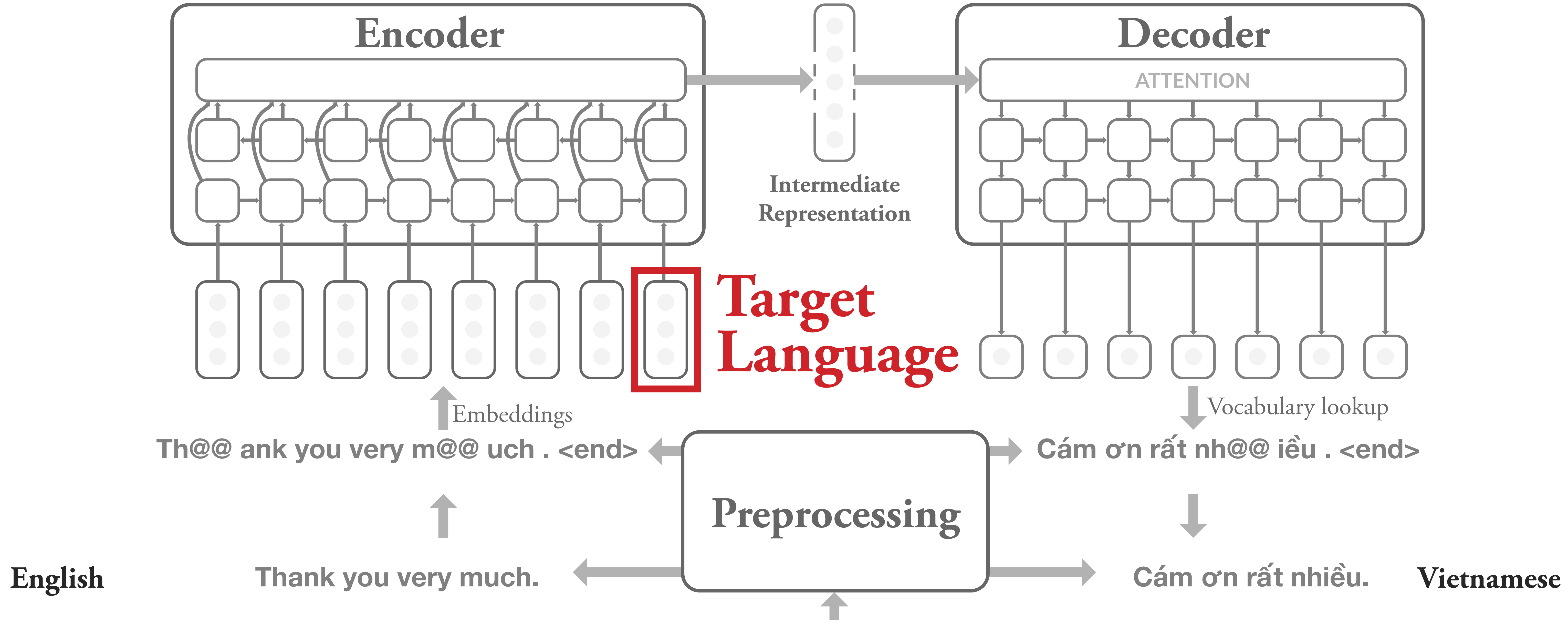
[Ha et al., IWSLT 2016 & Johnson et al., ACL 2017]

Multi-Task Learning

Contextual Parameter Generation

What about *multilingual translation?*





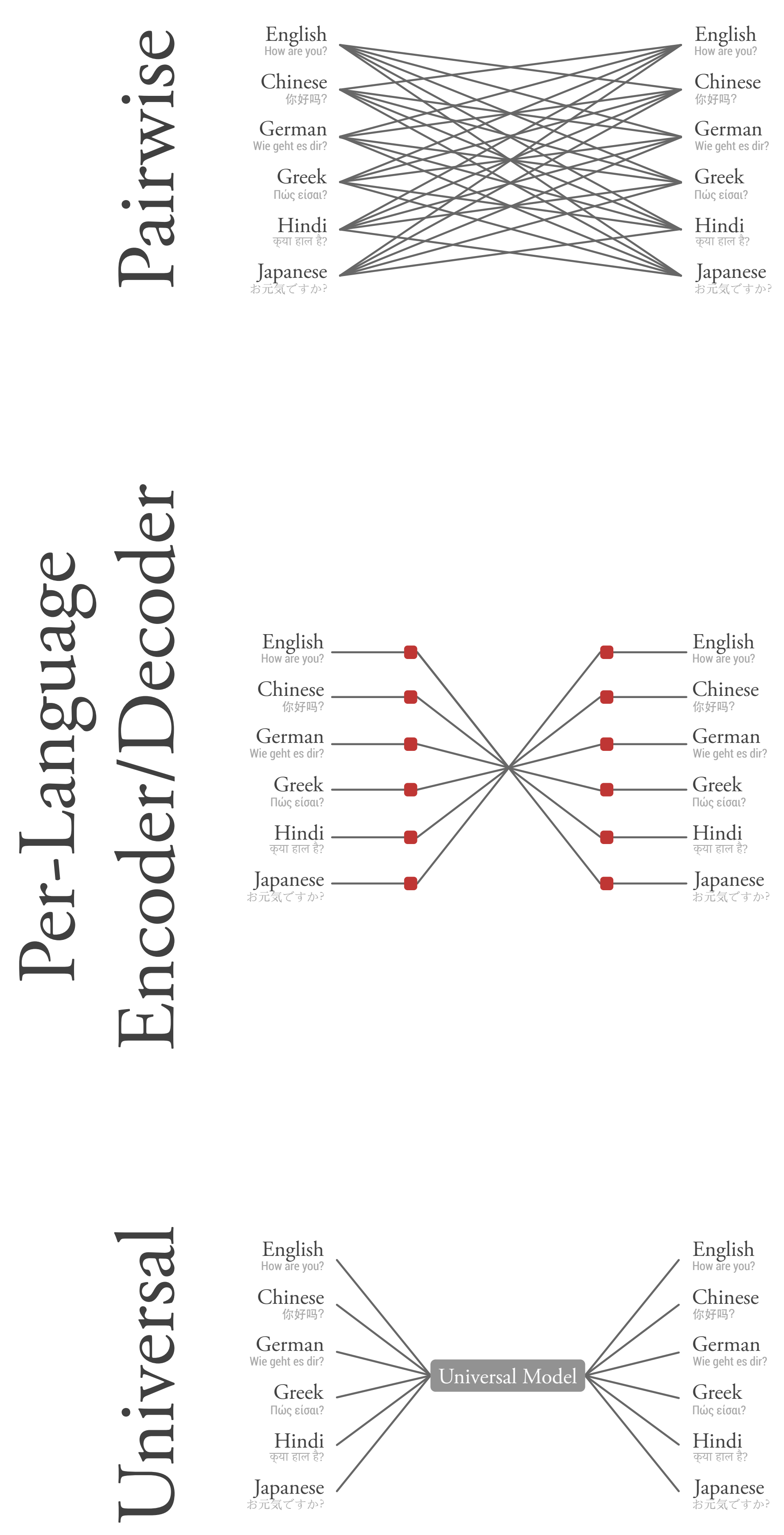
- Lacks language-specific parameterization.
- Cannot generalize to the other settings.
- It does not make sense to treat the target language in the same way as the input sentence words.

[Ha et al., IWSLT 2016 & Johnson et al., ACL 2017]

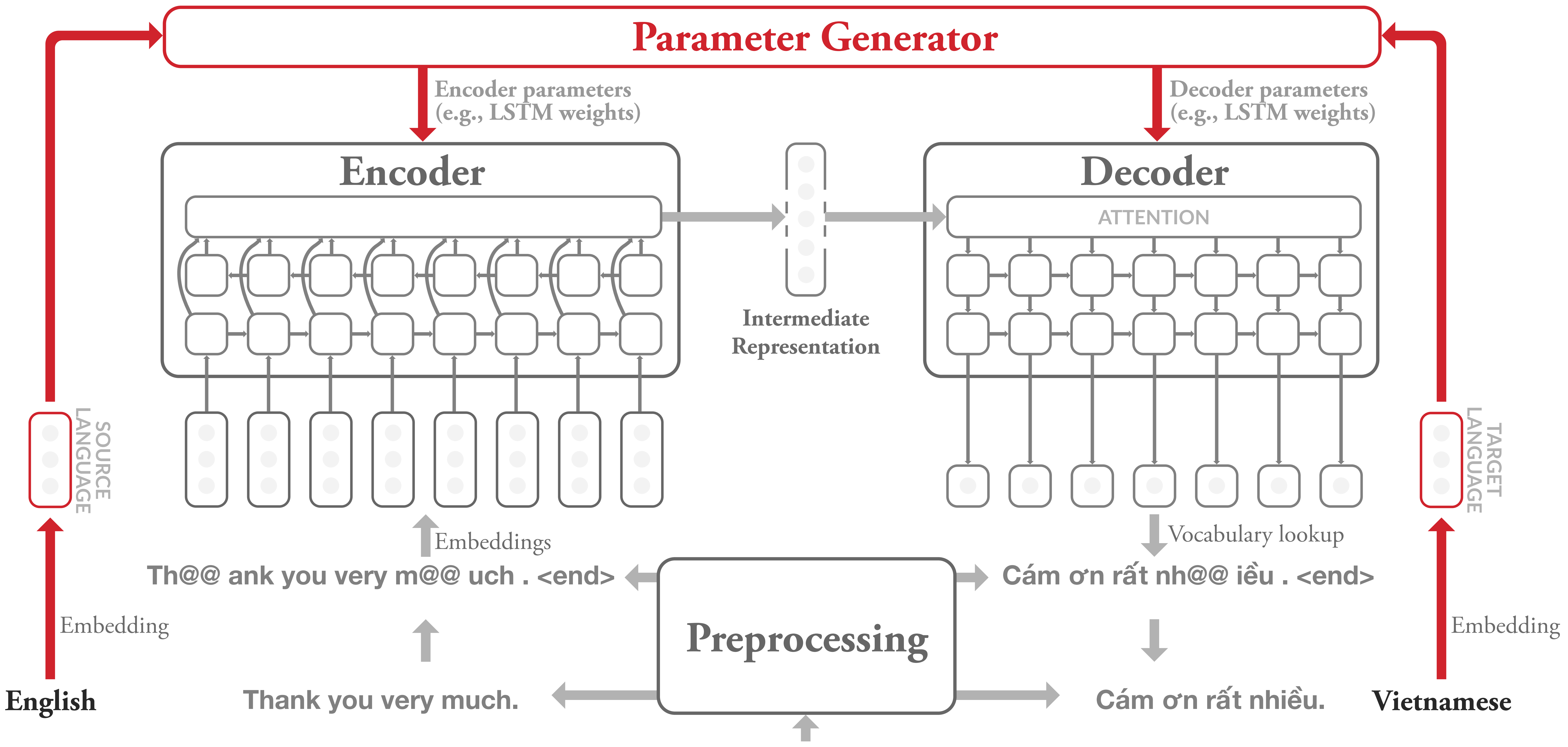
Multi-Task Learning

Contextual Parameter Generation

What about *multilingual translation?*



We proposed to explicitly allow conditioning on the languages:



[**Platanios**, Sachan, Neubig, Mitchell, EMNLP 2018]

Contextual Parameter Generation

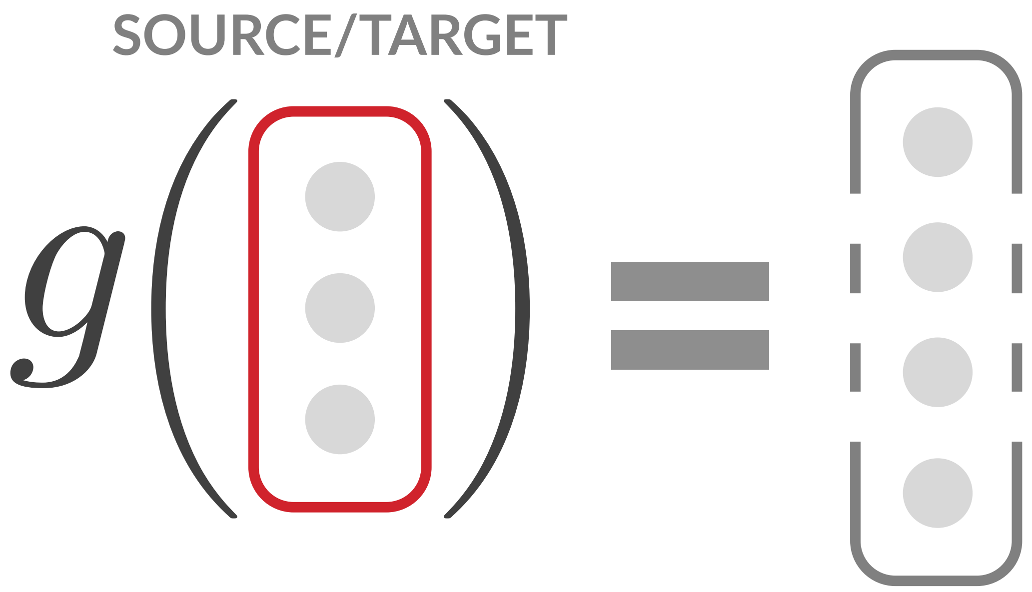
How is the *parameter generator* defined?

Let l_s refer to the source language and l_t refer to the target language. Then:

DECOUPLED

$$\theta^{(enc)} = g^{(enc)}(l_s)$$

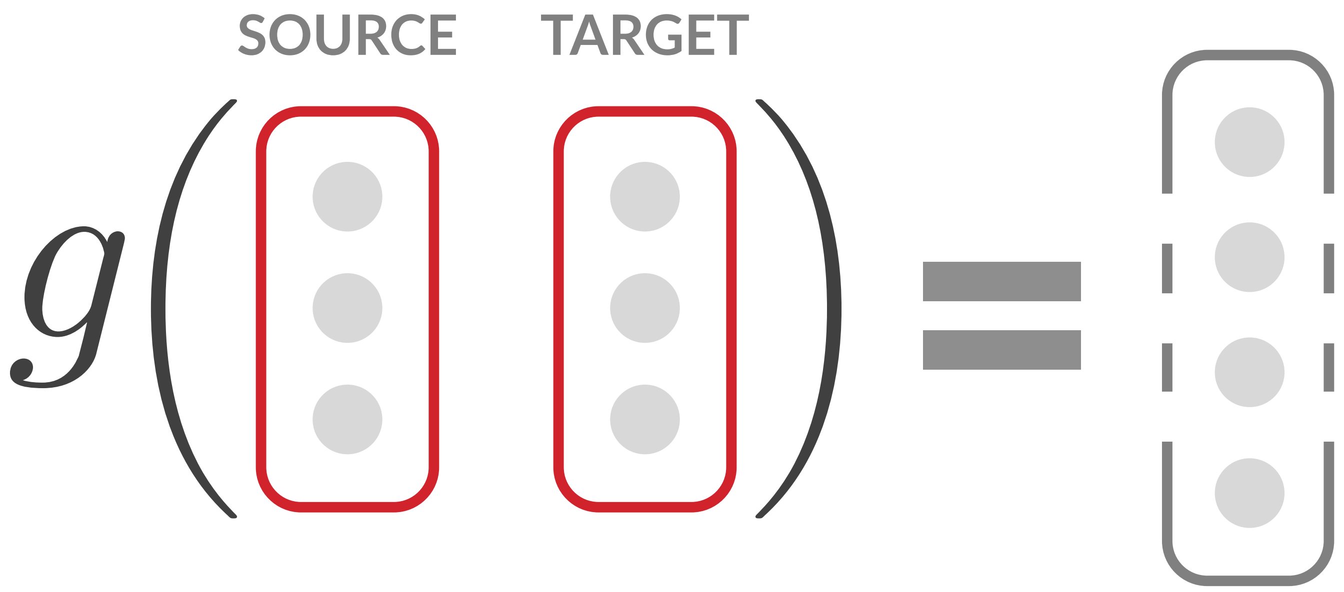
$$\theta^{(dec)} = g^{(dec)}(l_t)$$



COUPLED

$$\theta^{(enc)} = g^{(enc)}(l_s, l_t)$$

$$\theta^{(dec)} = g^{(dec)}(l_s, l_t)$$



[**Platanios**, Sachan, Neubig, Mitchell, EMNLP 2018]

Multi-Task Learning

How is the *parameter generator* defined?

Our goal is to provide a *simple form* for the parameter generator networks, that works and for which we can reason about. For this reason, we use simple *linear transforms:*

$$g^{(enc)}(l_s) = W^{(enc)} l_s$$

$$g^{(dec)}(l_t) = W^{(dec)} l_t$$

We also performed experiments with other forms that enabled more **controlled parameter sharing**.

For each language, the parameters of the encoder/decoder are defined as a *linear combination of the M columns of the corresponding weight matrix*, where M is the language embedding size.

[**Platanios**, Sachan, Neubig, Mitchell, EMNLP 2018]

Multi-Task Learning

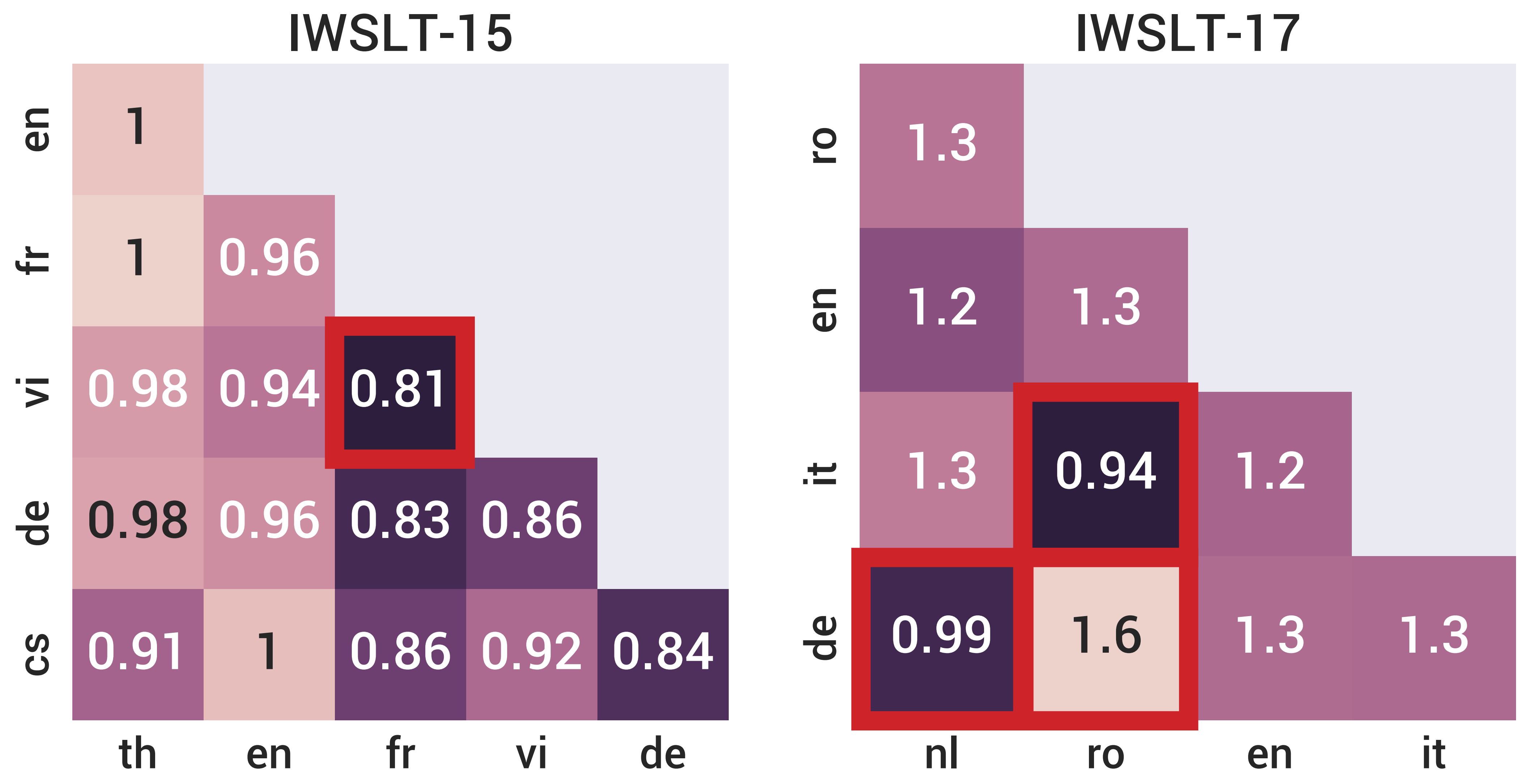
Contextual Parameter Generation

Language acts as the *context* in which translation is performed.

Using contextual parameter generation resulted in:

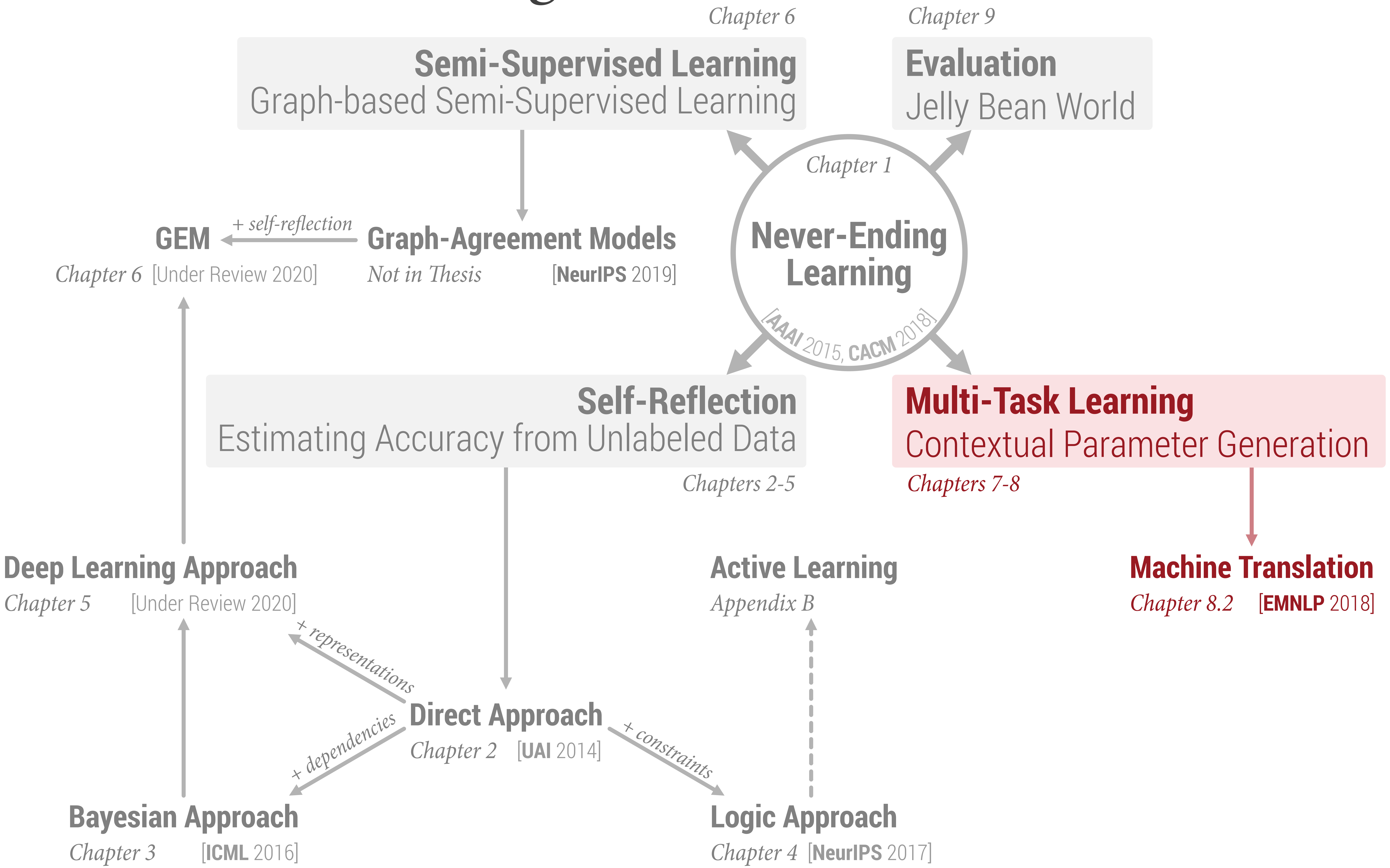
- Significant *performance gains* (+3 BLEU) and *reduced training time*.
- Ability to perform *zero-shot learning*.
- *Interpretable* task embeddings.

<u>Cosine Distances:</u>



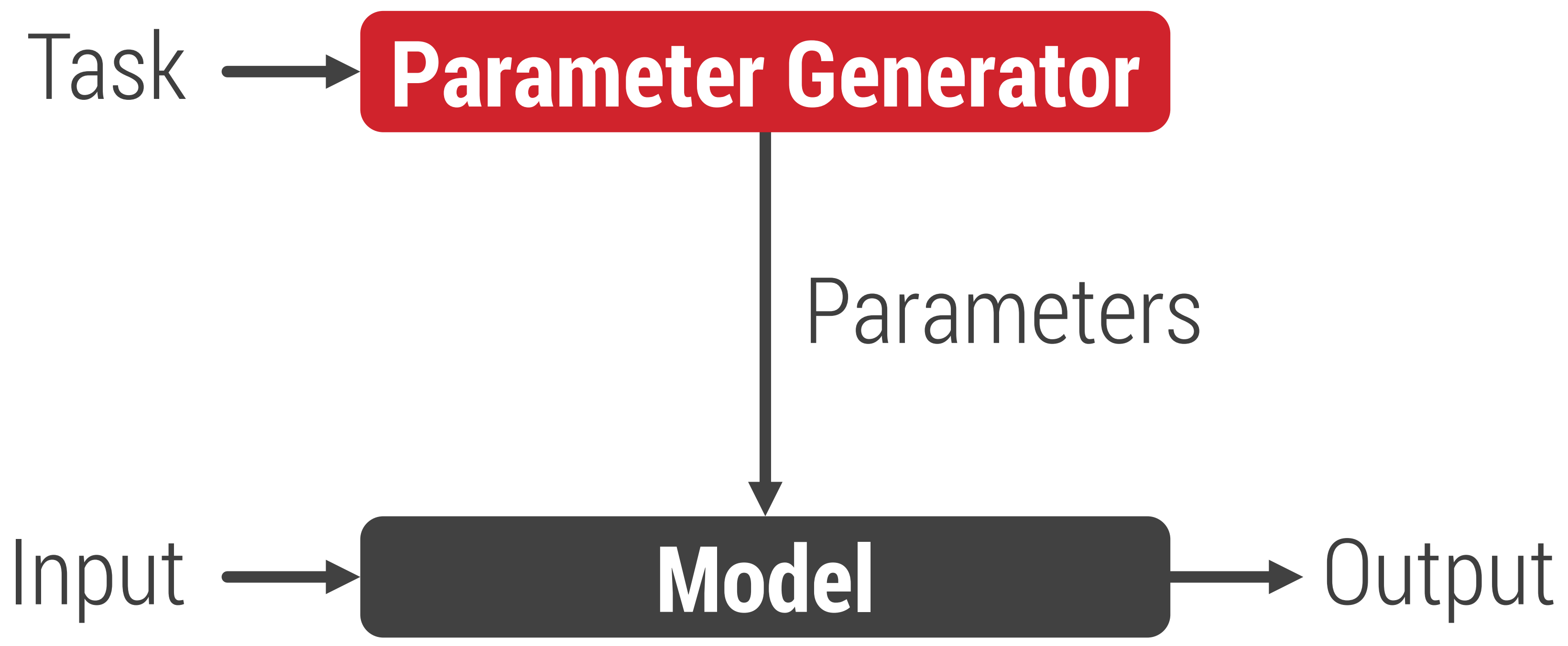
[**Platanios**, Sachan, Neubig, Mitchell, EMNLP 2018]

Multi-Task Learning

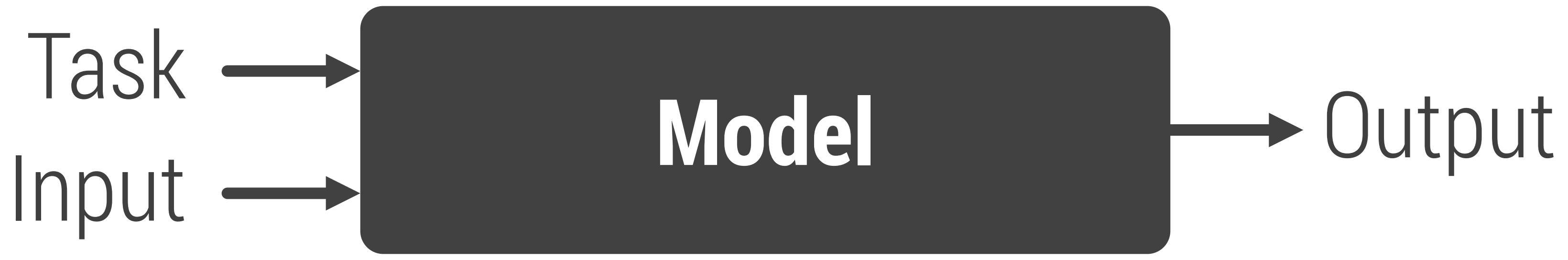


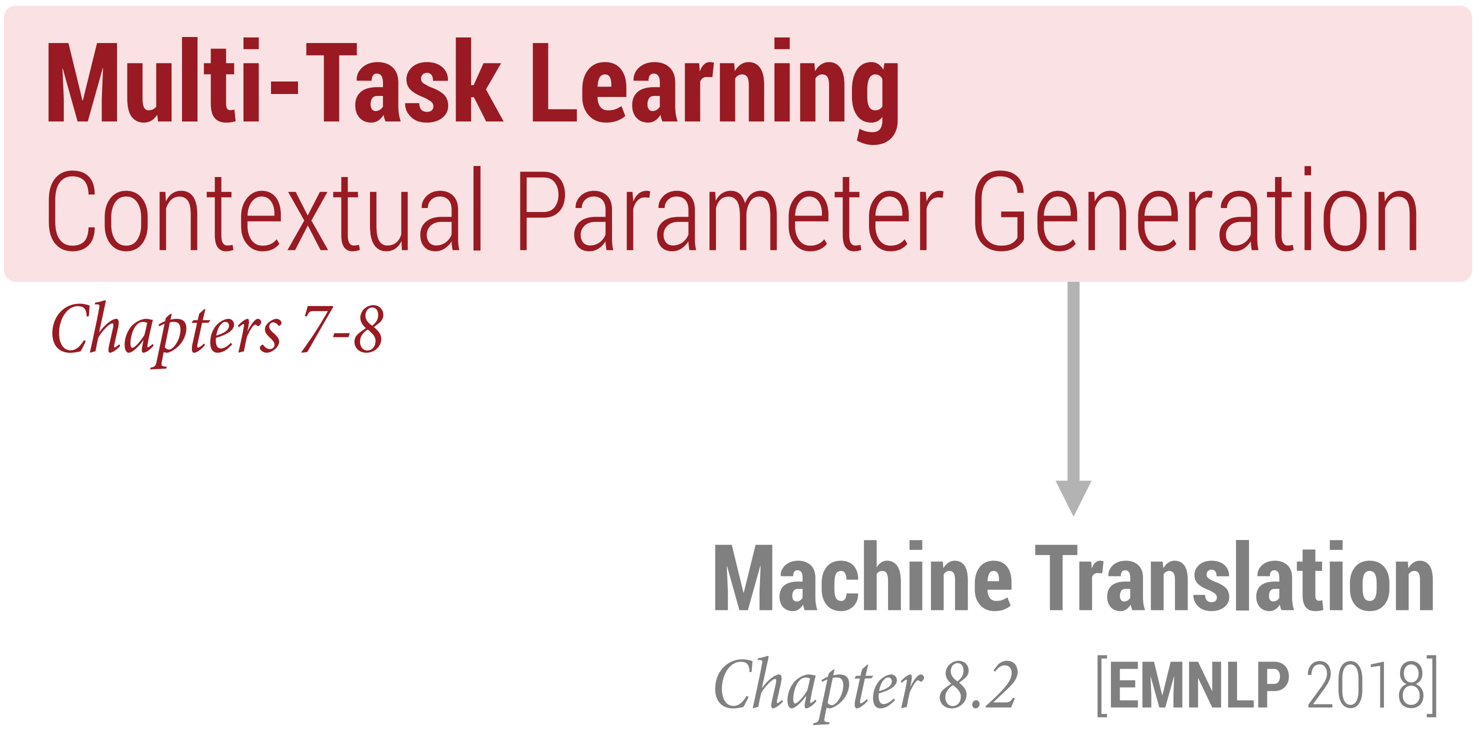
Multi-Task Learning

Contextual Parameter Generation



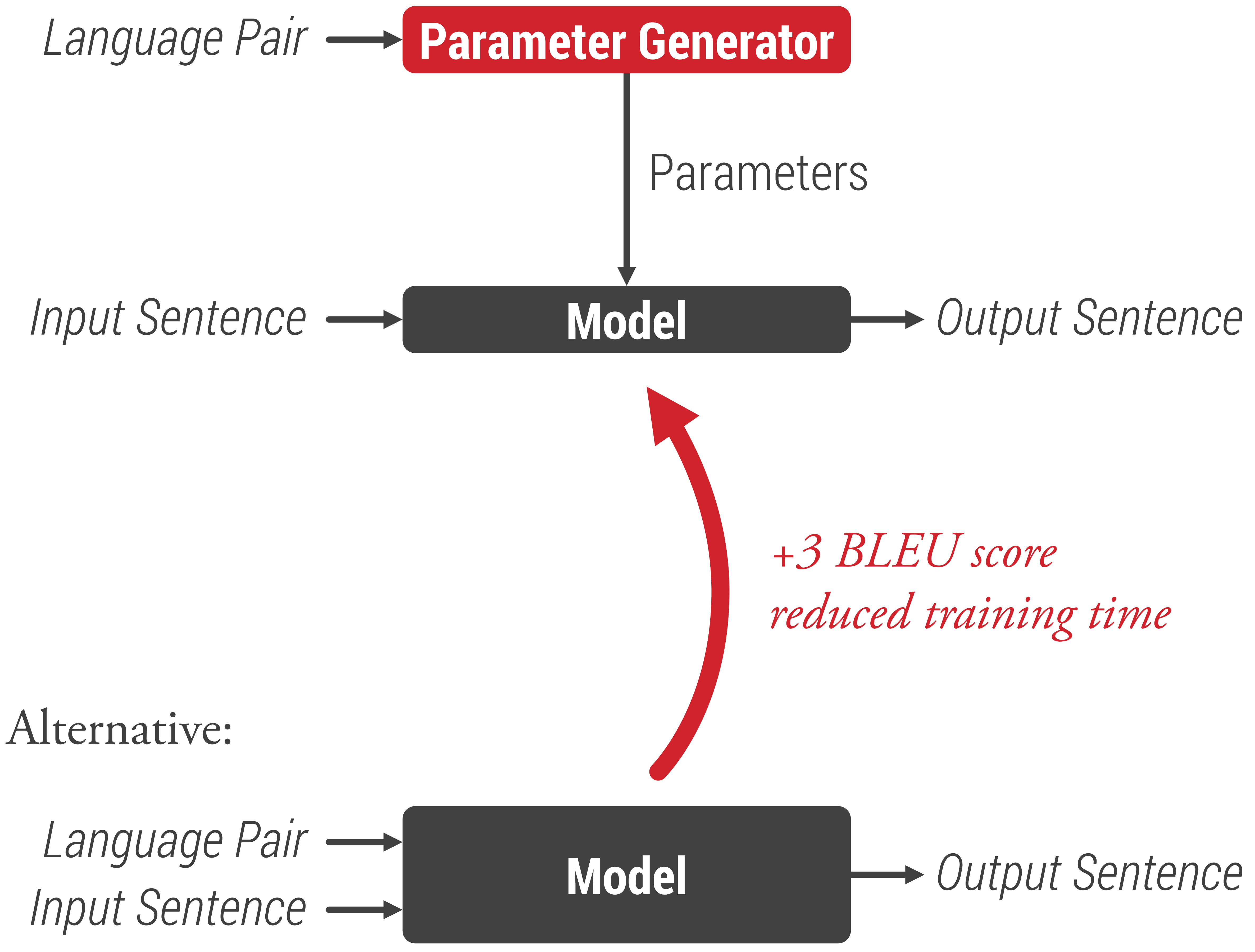
Alternative:





Multi-Task Learning

Contextual Parameter Generation for Machine Translation



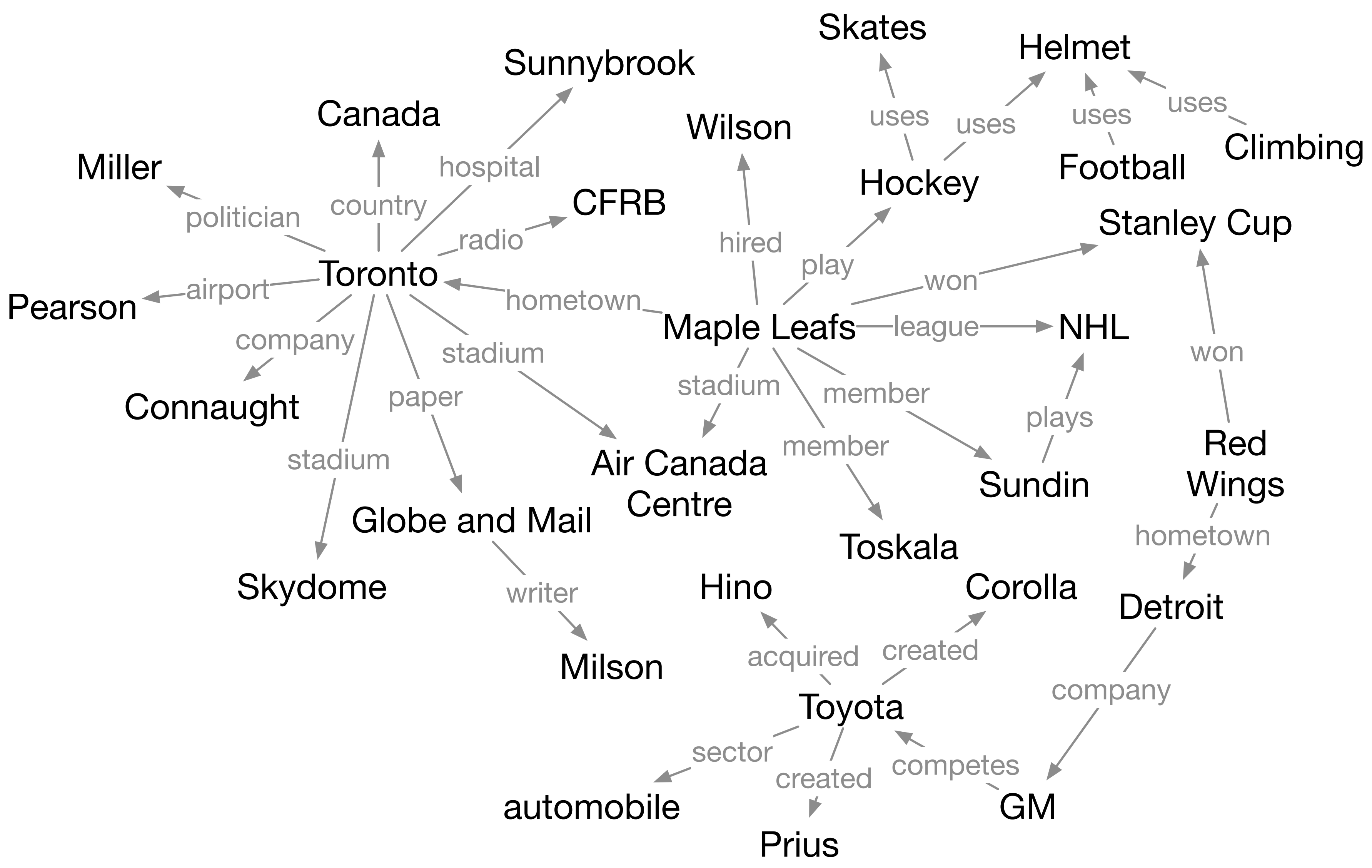
Multi-Task Learning
Contextual Parameter Generation
Chapters 7-8

Machine Translation
Chapter 8.2 [**EMNLP** 2018]

Multi-Task Learning

Contextual Parameter Generation for Link Prediction

Given a knowledge graph that contains triples of the form (source entity, relation, target entity), we want to answer questions of the form (source entity, relation, ?).



For example:

Task

(Toronto, capitalOf, ?)

Input

Multi-Task Learning

Contextual Parameter Generation

Chapters 7-8

Machine Translation

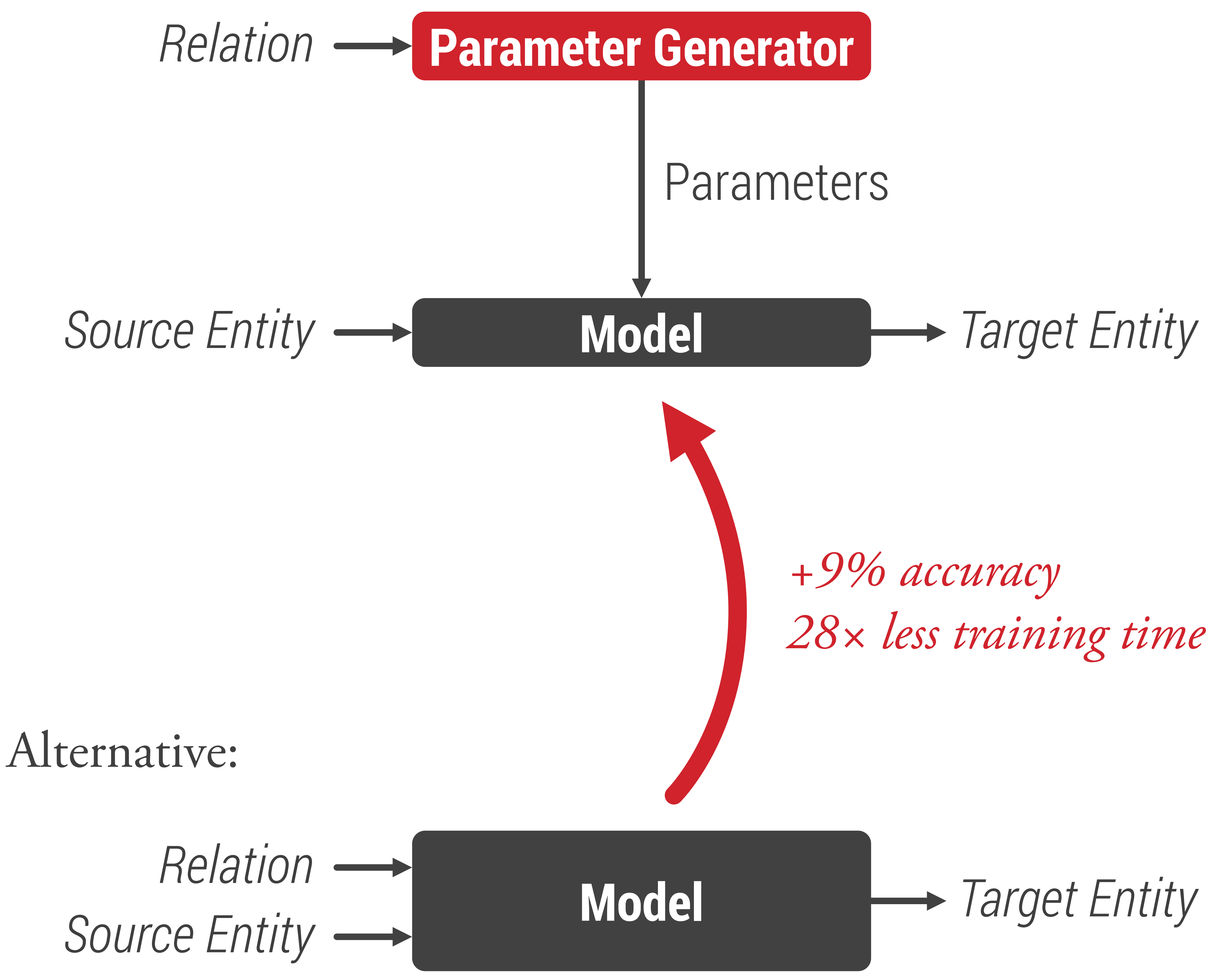
Chapter 8.2 [**EMNLP** 2018]

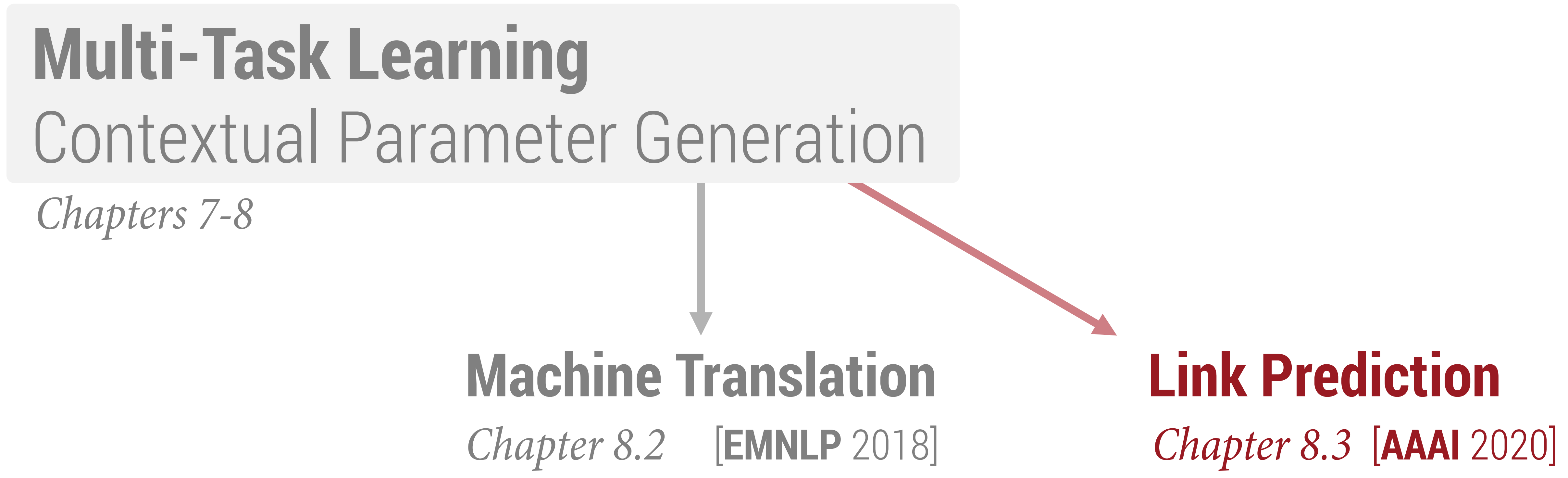
Link Prediction

Chapter 8.3 [**AAAI** 2020]

Multi-Task Learning

Contextual Parameter Generation for Link Prediction

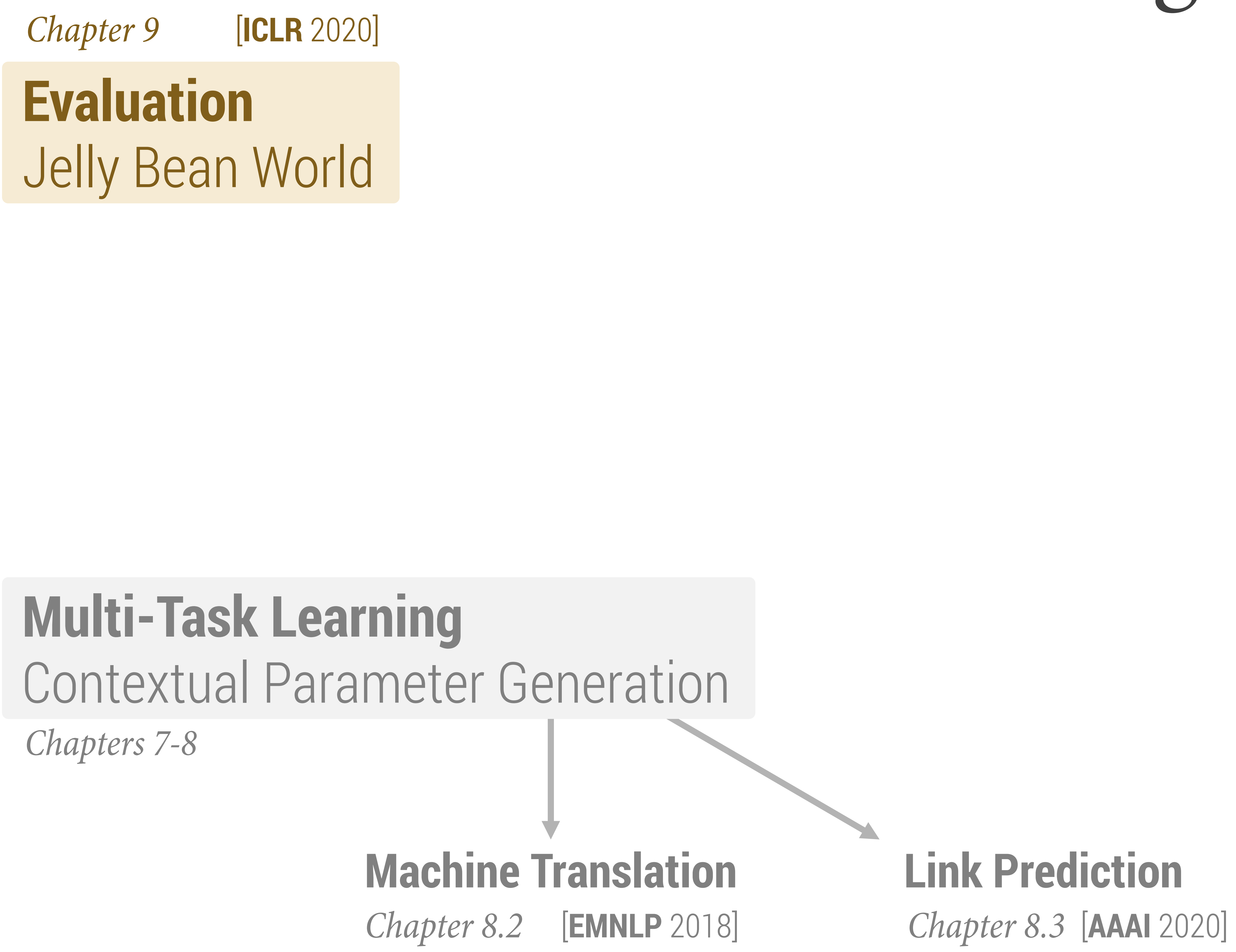




Contextual Parameter Generation for Task Compositions

Let us consider the following grid world:

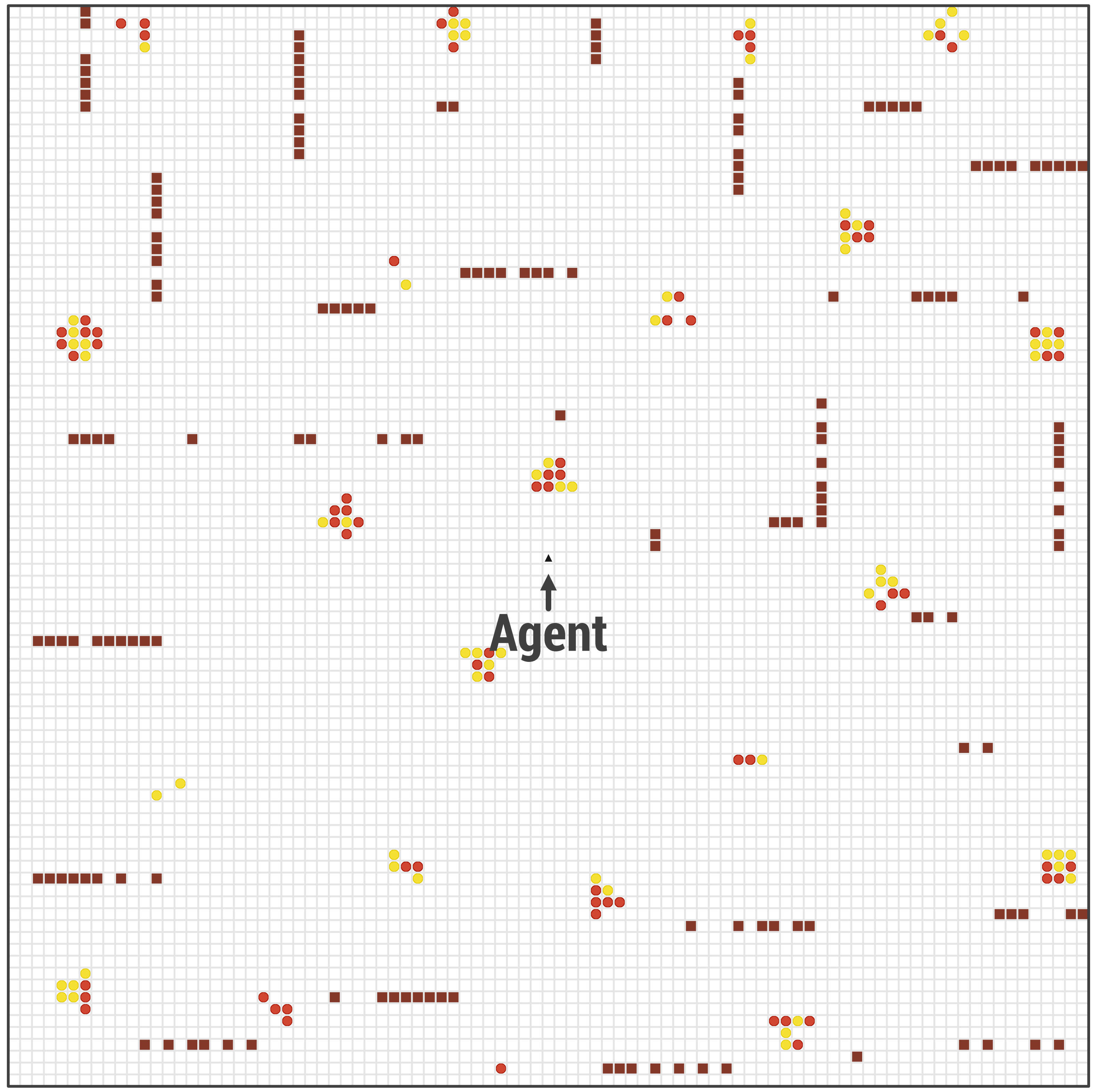
Infinite two-dimensional grid.



Multi-Task Learning

Contextual Parameter Generation for Task Compositions

Let us consider the following grid world:



Contains items of various types.

Chapter 9 [**ICLR** 2020]

Evaluation
Jelly Bean World

Multi-Task Learning
Contextual Parameter Generation
Chapters 7-8

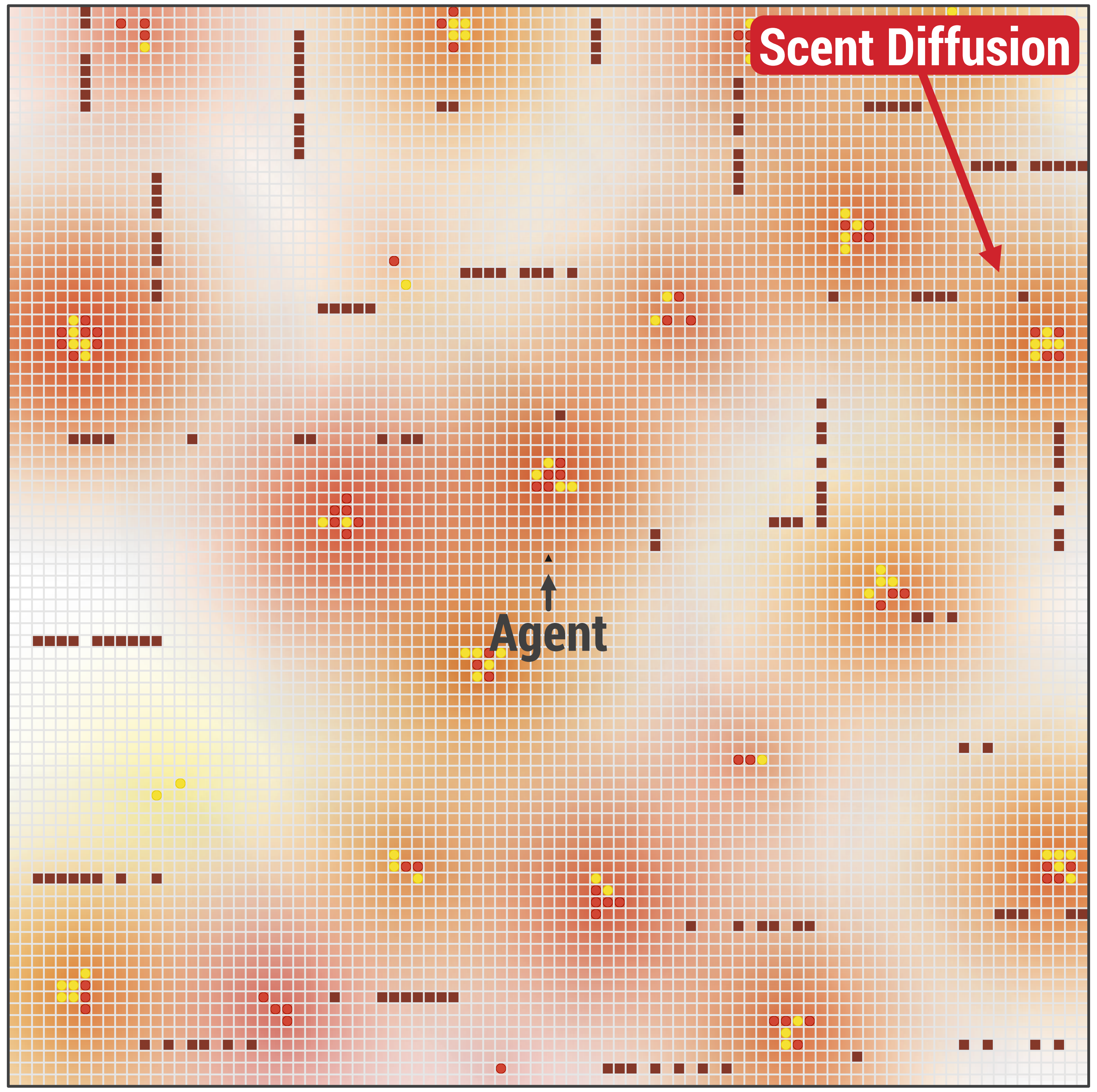
Machine Translation
Chapter 8.2 [**EMNLP** 2018]

Link Prediction
Chapter 8.3 [**AAAI** 2020]

Multi-Task Learning

Contextual Parameter Generation for Task Compositions

Let us consider the following grid world:



Each item has a *color* and a *scent*.

Chapter 9 [**ICLR** 2020]

Evaluation
Jelly Bean World

Multi-Task Learning
Contextual Parameter Generation
Chapters 7-8

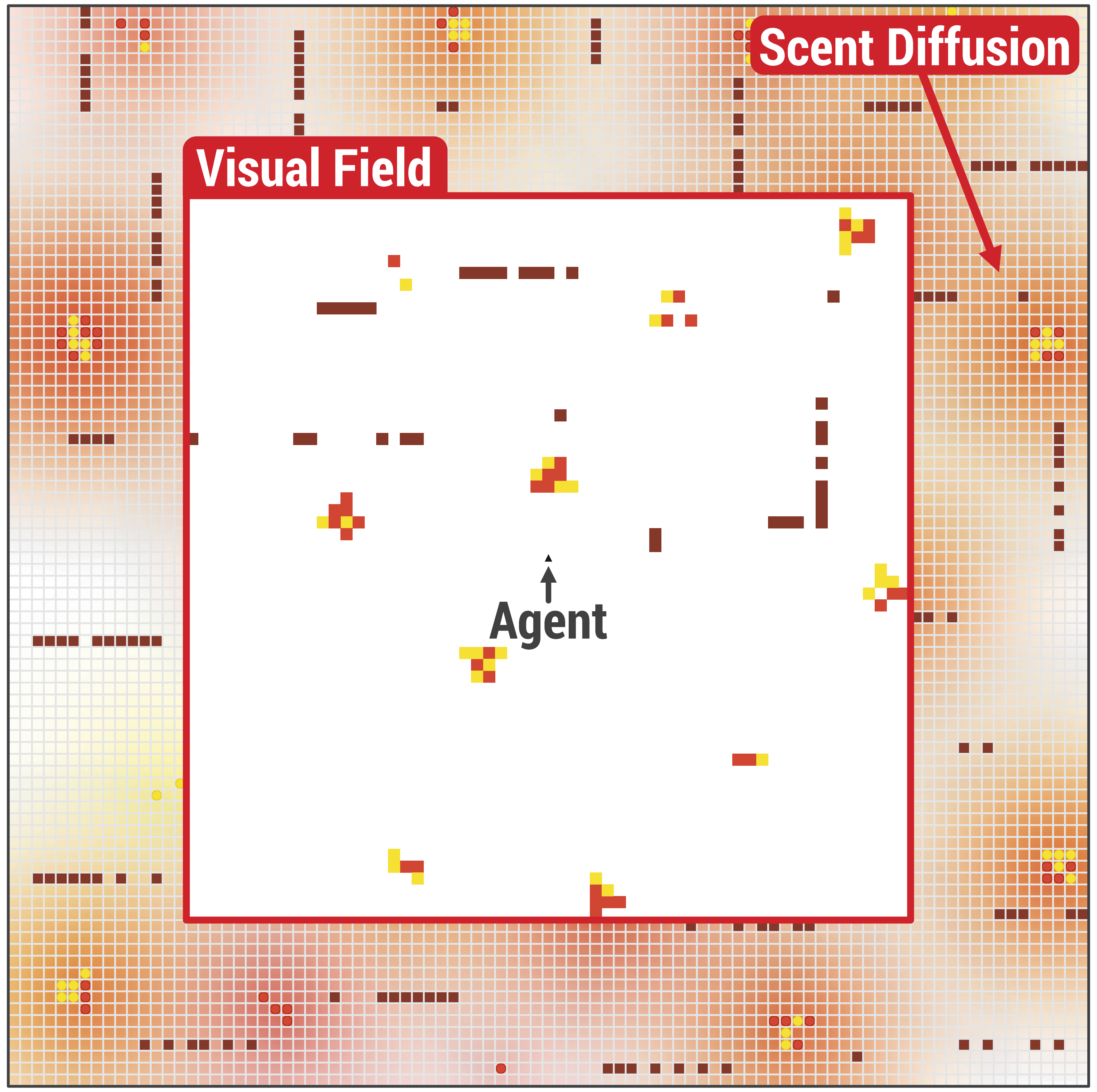
Machine Translation
Chapter 8.2 [**EMNLP** 2018]

Link Prediction
Chapter 8.3 [**AAAI** 2020]

Multi-Task Learning

Contextual Parameter Generation for Task Compositions

Let us consider the following grid world:



Each item has a *color* and a *scent*.

Chapter 9 [**ICLR** 2020]

Evaluation
Jelly Bean World

Multi-Task Learning
Contextual Parameter Generation
Chapters 7-8

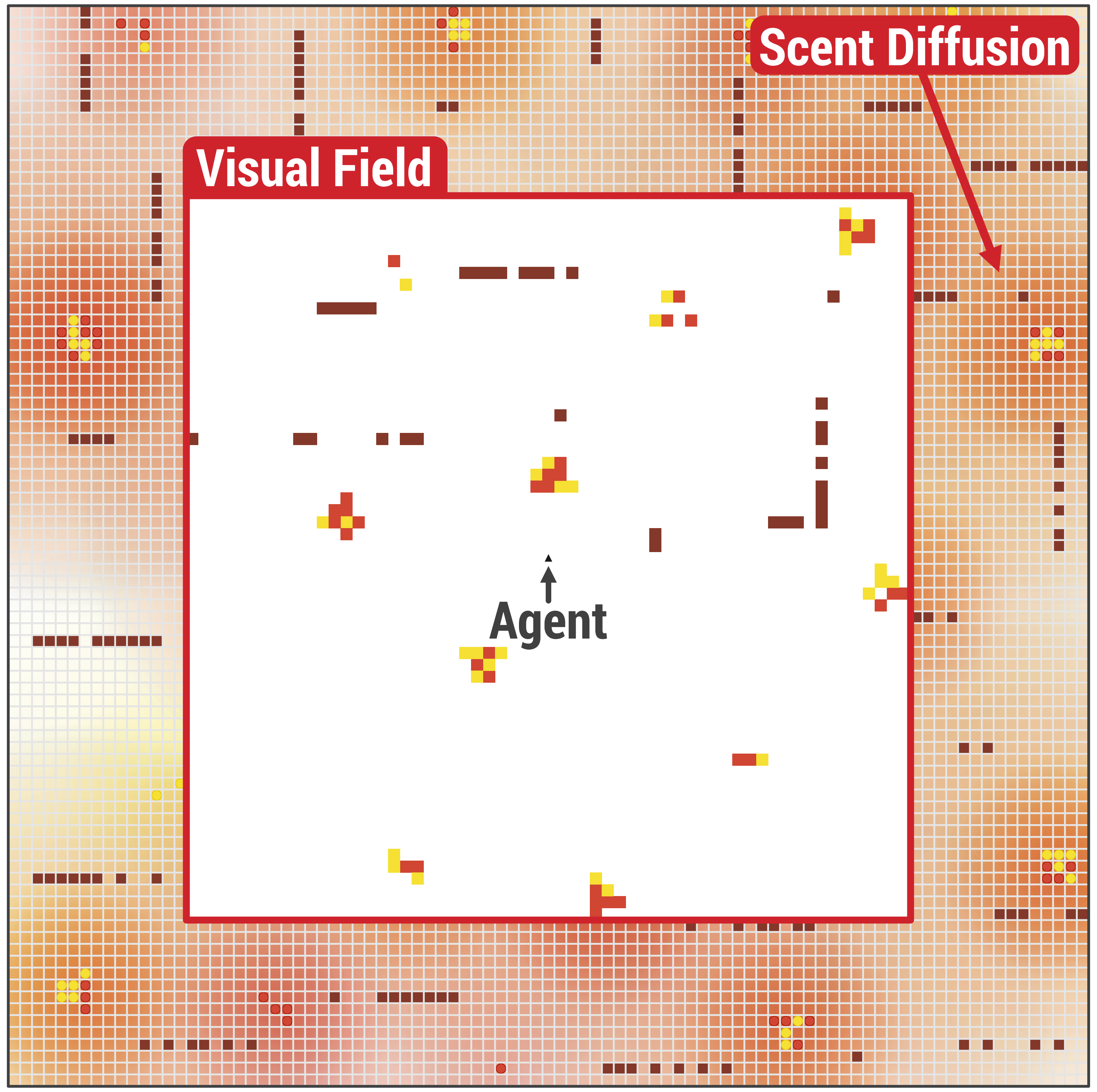
Machine Translation
Chapter 8.2 [**EMNLP** 2018]

Link Prediction
Chapter 8.3 [**AAAI** 2020]

Multi-Task Learning

Contextual Parameter Generation for Task Compositions

Let us consider the following grid world:



We can define arbitrary reward functions in terms of items the agent must *collect* or *avoid*.

Chapter 9 [**ICLR** 2020]

Evaluation
Jelly Bean World

Multi-Task Learning
Contextual Parameter Generation
Chapters 7-8

Machine Translation
Chapter 8.2 [**EMNLP** 2018]

Link Prediction
Chapter 8.3 [**AAAI** 2020]

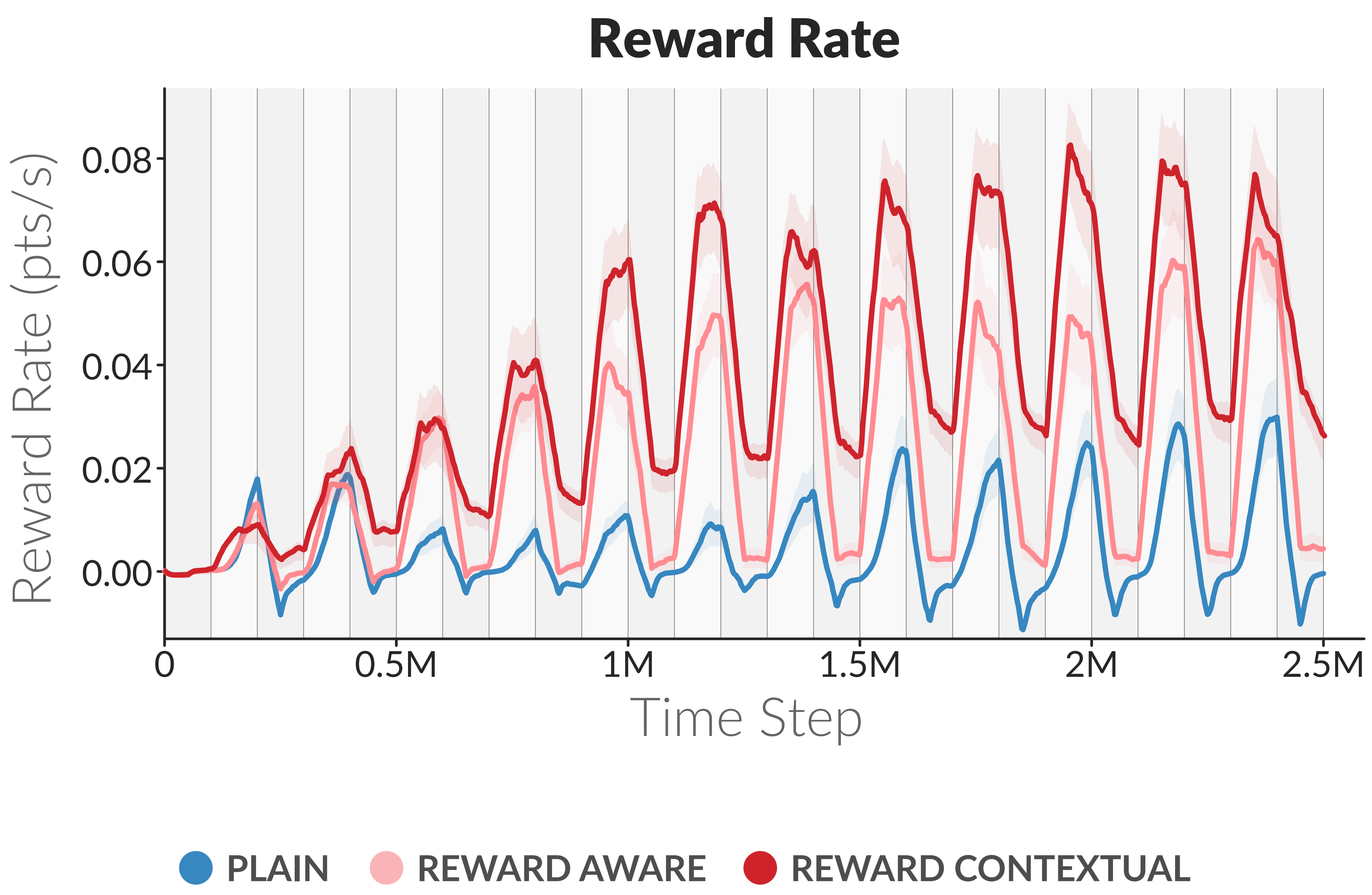
Contextual Parameter Generation for Task Compositions

Let us consider the following example:

Collect[**JellyBean**] ∧ Avoid[**Onion**]

Switch every 100,000 steps

Avoid[**JellyBean**] ∧ Collect[**Onion**]



Multi-Task Learning

Chapter 9 [**ICLR** 2020]

Evaluation
Jelly Bean World

Jelly Bean World
Chapter 8.4

Multi-Task Learning
Contextual Parameter Generation
Chapters 7-8

Machine Translation
Chapter 8.2 [**EMNLP** 2018]

Link Prediction
Chapter 8.3 [**AAAI** 2020]

Multi-Task Learning

	Hard Sharing	Soft Sharing	Task as Input	Task as Context
Avoids copying/re-training	✗	✗	✓	✓
Avoids negative transfer	✗	✓	✗	✓
Enables positive transfer	✓	✗	✓	✓
Enables task dependencies	✗	✗	✓	✓
Enables zero-shot learning	✗	✗	✓	✓
Enables fast adaptation	✗	✗	✗	✓

Caveats: The **number of parameters increases** and the CPG-enhanced networks have higher expressive power and thus **higher risk of overfitting**.

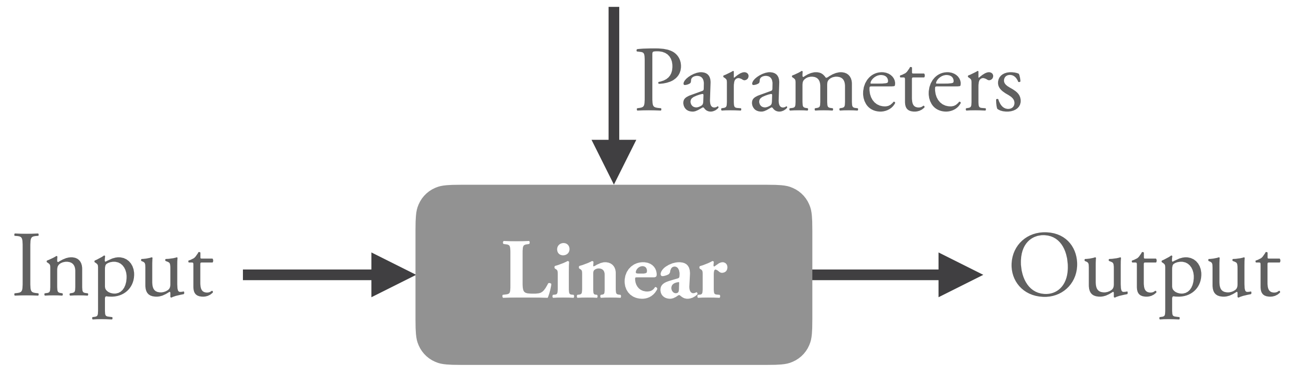
Why does contextual parameter generation work so well?

- Is it related to hierarchical modeling in probabilistic models?
- How does it increase the expressive power of neural networks?

Multi-Task Learning

Why does contextual parameter generation work so well?

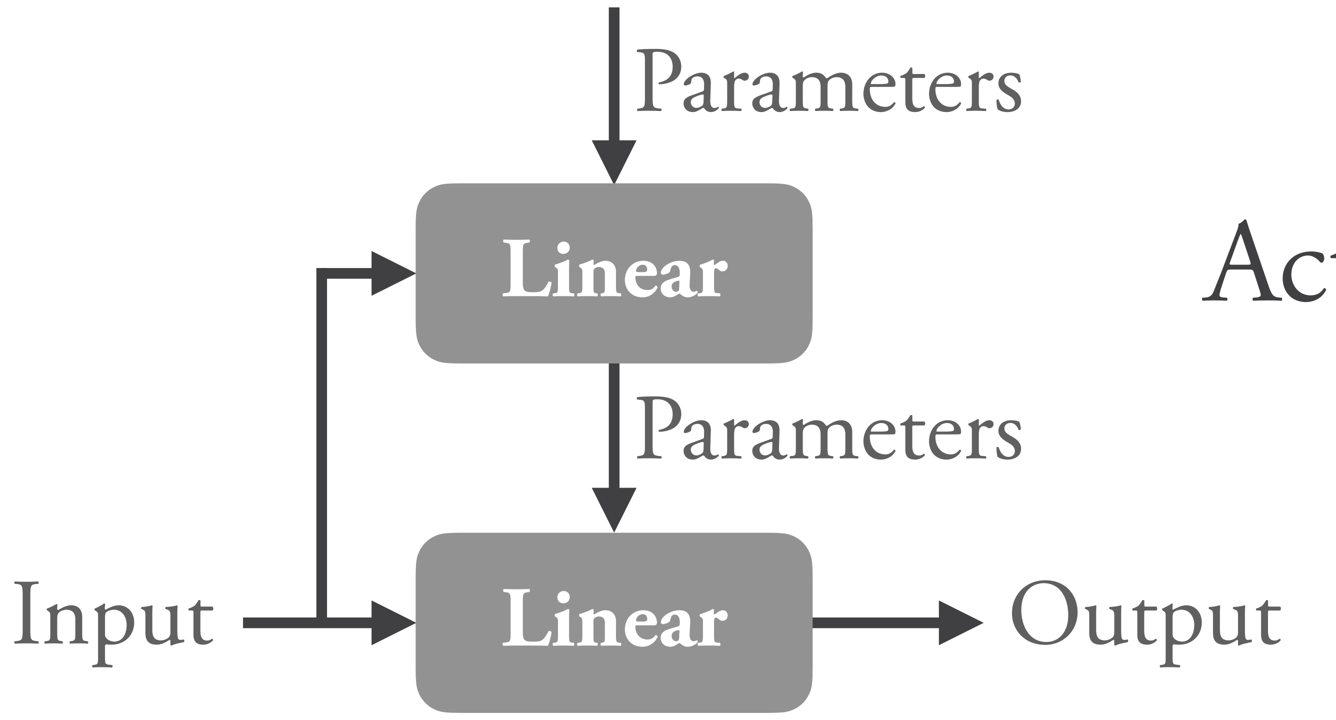
- Is it related to hierarchical modeling in probabilistic models?
- How does it increase the expressive power of neural networks?



Cannot represent the XOR function!

Why does contextual parameter generation work so well?

- Is it related to hierarchical modeling in probabilistic models?
- How does it increase the expressive power of neural networks?

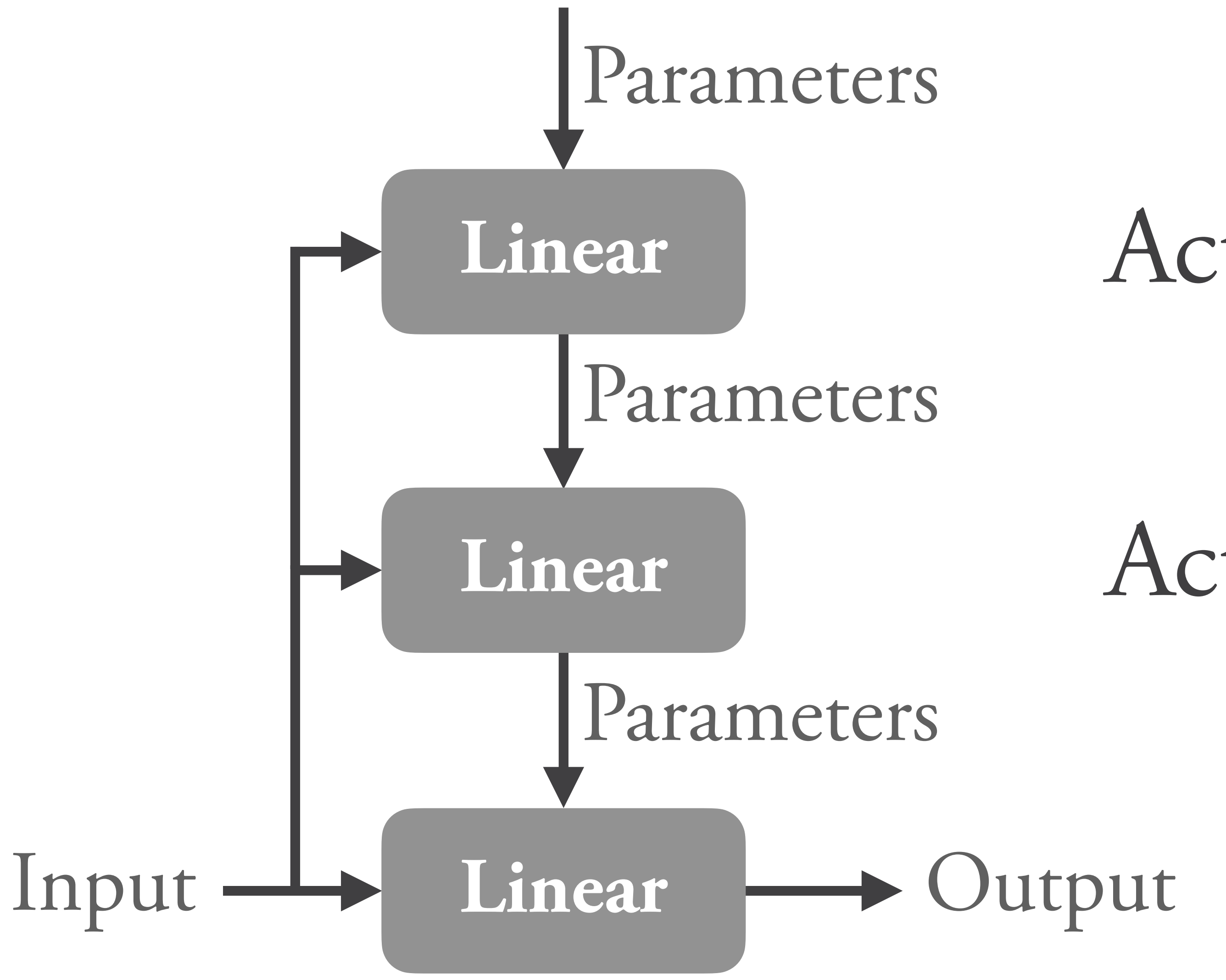


Acts a **quadratic kernel!**

Can represent the XOR function!

Why does contextual parameter generation work so well?

- Is it related to hierarchical modeling in probabilistic models?
- How does it increase the expressive power of neural networks?



Acts a **cubic kernel!**

Acts a **quadratic kernel!**

Can represent the XOR function!

Thesis Overview

Thesis Overview

Thesis Overview

Thesis Overview

Thesis Statement

multi-task learning

A computer system that learns to perform multiple tasks jointly and that is aware of the relationships between these tasks, will be able to learn more efficiently and effectively than a system that learns to perform each task in isolation.

Self-Reflection
Estimating Accuracy from Unlabeled Data
Chapters 2-5

Multi-Task Learning
Contextual Parameter Generation
Chapters 7-8

Moreover, the relationships between the tasks may either be explicitly provided through supervision or implicitly learned by the system itself, and will allow the system to self-reflect and evaluate itself without any task-specific supervision.

self-reflection

Lessons Learned & Open Questions

multi-task learning

Contextual parameter generation is a highly effective method for multi-task learning. **!**

Contextual parameter generation increases a model's representational capacity. **!**

Can we obtain guarantees for contextual parameter generation? **?**

! Consistency is related to correctness!

! Dependencies among the predictors control this relationship.

! Crossing boundaries between paradigms can yield significant gains.

? Can we obtain guarantees for the underlying truth?

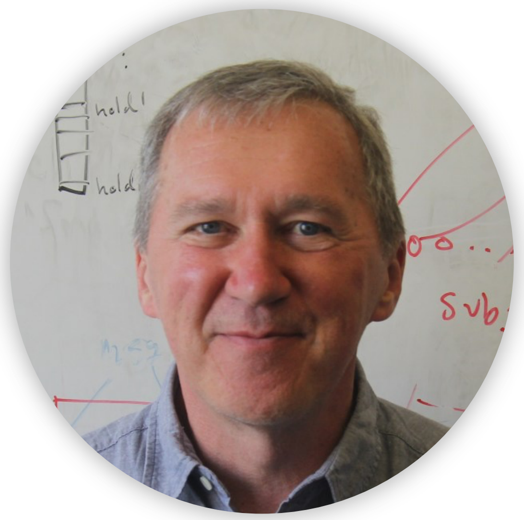
? Can we train models with the sole objective of becoming consistent?

? What does this tell us about human learning?

self-reflection

Neural Cognitive Architectures!

Thanks to my collaborators and colleagues that made this work possible!


Tom Mitchell


Eric Horvitz


Rich Caruana


Graham Neubig


Otilia Stretcu


Maruan Al-Shedivat


Avinava Dubey


Mrinmaya Sachan


Abulhair Saparov


George Stoica


Avrim Blum


Ashish Kapoor


Hoifung Poon


Alex Smola

Publications

(1) Estimating Accuracy from Unlabeled Data.
Platanios, Blum, and Mitchell. In **UAI** 2014.

(2) Estimating Accuracy from Unlabeled Data: A Bayesian Approach.
Platanios, Dubey, and Mitchell. In **ICML** 2016.

(3) Estimating Accuracy from Unlabeled Data: A Probabilistic Logic Approach.
Platanios, Poon, Mitchell, and Horvitz. In **NeurIPS** 2017.

(4) Learning from Imperfect Annotations.
Platanios, Al-Shedivat, Xing, and Mitchell. Under review 2020.

(5) Learning When To Trust Your Neighbors: Robust Graph-Based Semi-Supervised Learning.
Platanios, Stretcu, Saparov, and Mitchell. Under review 2020.

self-reflection

(6) Contextual Parameter Generation for Universal Neural Machine Translation.
Platanios, Sachan, Neubig, and Mitchell. In **EMNLP** 2018.

(7) Contextual Parameter Generation for Knowledge Graph Link Prediction.
Platanios*, Stretcu*, Stoica*, Póczos, and Mitchell. In **AAAI** 2020.

contextual parameter generation

(8) Competence-based Curriculum Learning.
Platanios, Stretcu, Neubig, Póczos, and Mitchell. In **NAACL** 2019.

(9) Learn to Walk Before You Run: Coarse-to-Fine Curriculum Learning for Classification.
Stretcu, **Platanios**, Mitchell, and Póczos. Under review 2020.

curriculum learning

(10) Never-Ending Learning.
Mitchell, ..., **Platanios**, ..., and Welling. In **AAAI** 2015.

(11) Never-Ending Learning.
Mitchell, ..., **Platanios**, ..., and Welling. In **CACM** 2019.

(12) Jelly Bean World: A Testbed for Never-Ending Learning.
Platanios*, Saparov*, and Mitchell. In **ICLR** 2020.

never-ending learning

(13) Gaussian Process-Mixture Conditional Heteroscedasticity.
Platanios and Chatzis. In **TPAMI** 2014.

(14) Active Learning amidst Logical Knowledge.
Platanios, Kapoor, and Horvitz. In **arXiv** 2017.

(15) Agreement-based Learning.
Platanios. In **arXiv** 2018.

(16) Deep Graphs.
Platanios and Smola. In **arXiv** 2018.

(17) Graph Agreement Models for Semi-Supervised Learning.
Stretcu, Viswanathan, Movshovitz-Attias, **Platanios**, Ravi and Tomkins. In **NeurIPS** 2019.

other

Open-Source Projects

deep learning

TensorFlow Scala

type-safe linear algebra, tensors, and neural networks

Watch 68 | Star 754 | Fork 80

github.com/eaplatanios/tensorflow_scala

machine translation

SymphonyMT | Star 28

github.com/eaplatanios/symphony-mt

Swift for TensorFlow
differentiable programming

Watch 268 | Star 5.2k | Fork 487

github.com/tensorflow/swift

other

makina | Star 20

github.com/eaplatanios/makina

reinforcement learning

swift-rl | Star 60

github.com/eaplatanios/swift-rl

swift-ale | Star 21

github.com/eaplatanios/swift-ale

Workshops Organized

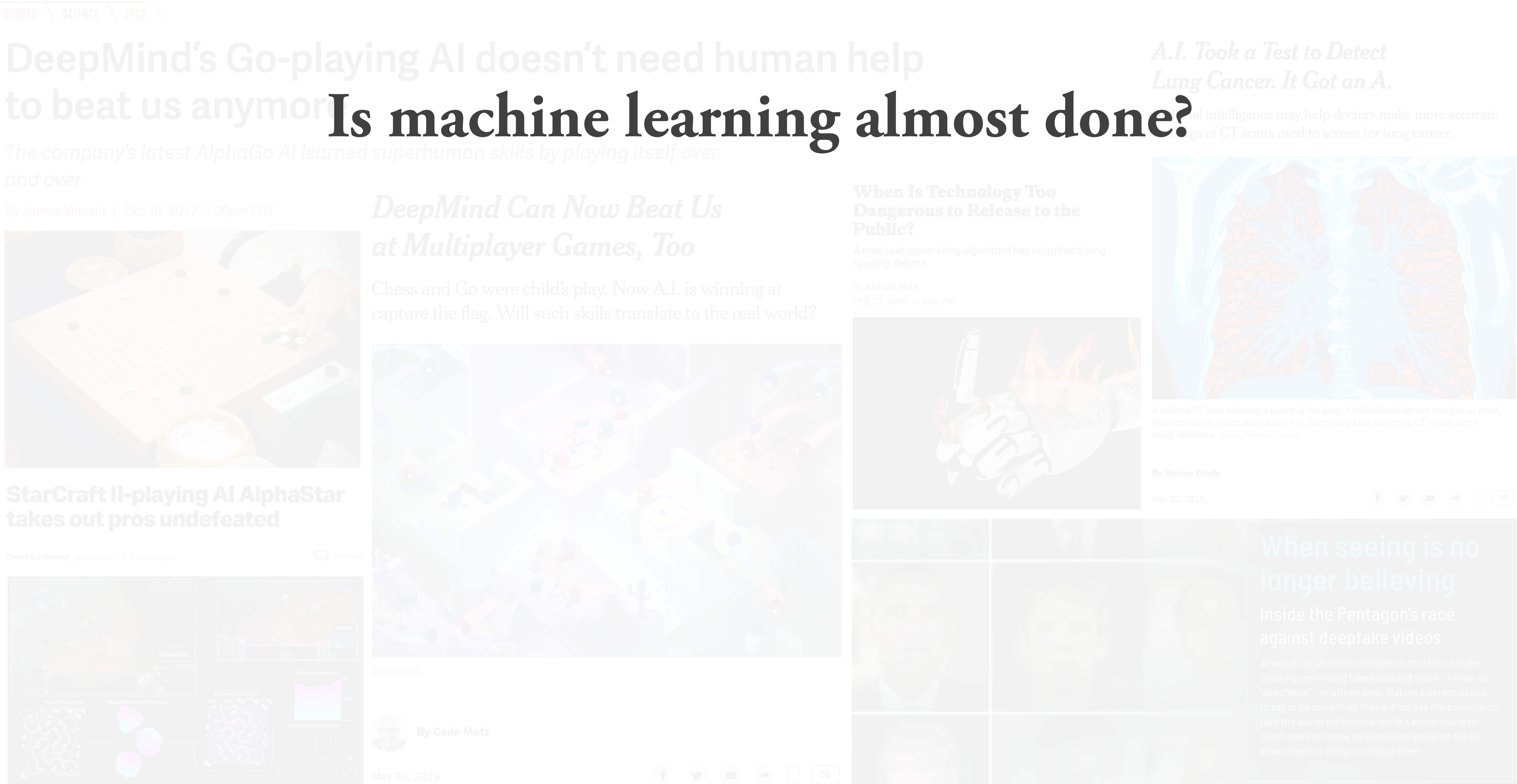
LEARNING WITH LIMITED LABELED DATA
NIPS 2017

LEARNING WITH LIMITED LABELED DATA
ICLR 2019

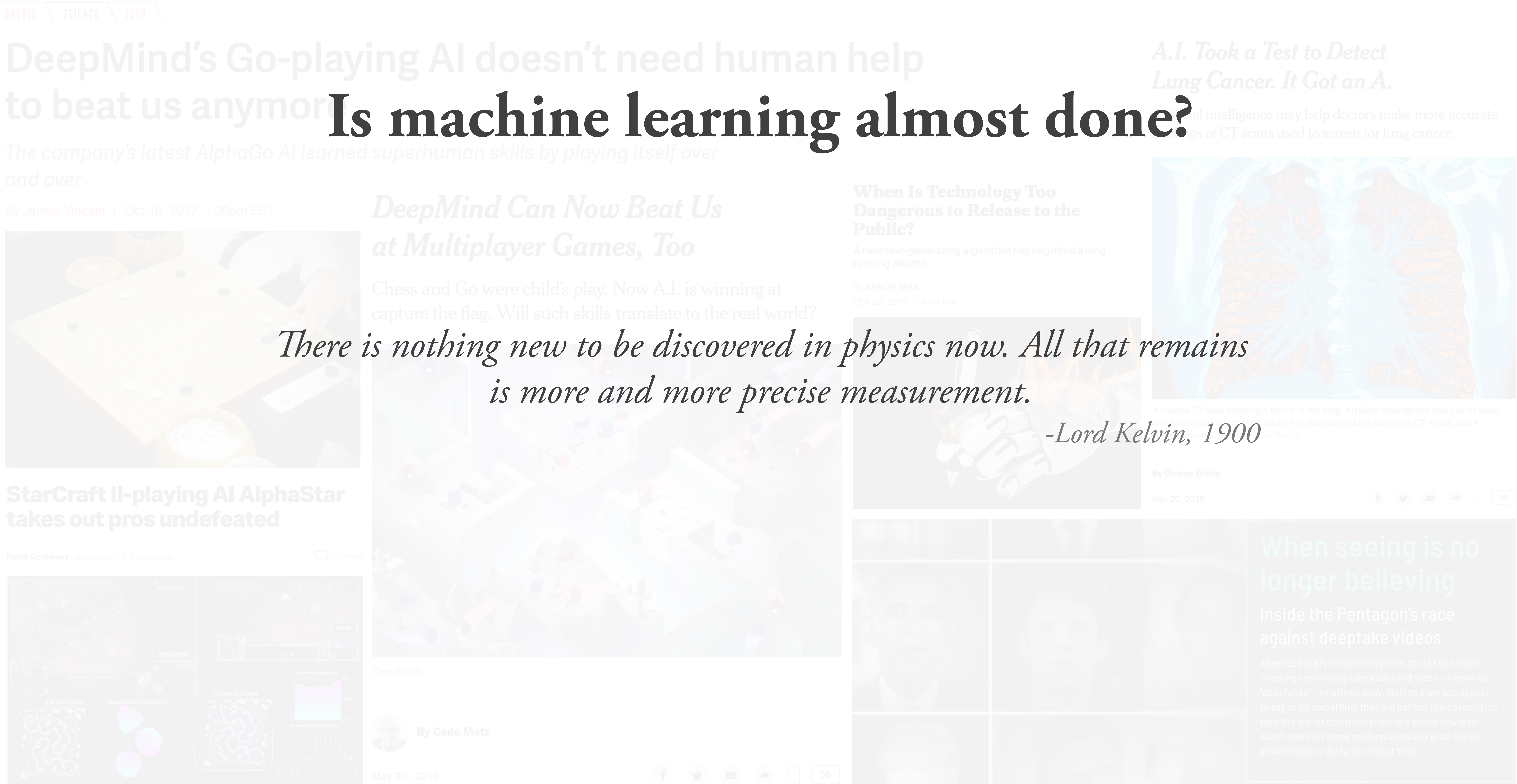
Workshop on
Adaptive & Multitask Learning: Algorithms and Systems
ICML 2019

e.a.platanios@cs.cmu.edu

Is machine learning almost done?



Is machine learning almost done?



There is nothing new to be discovered in physics now. All that remains is more and more precise measurement.

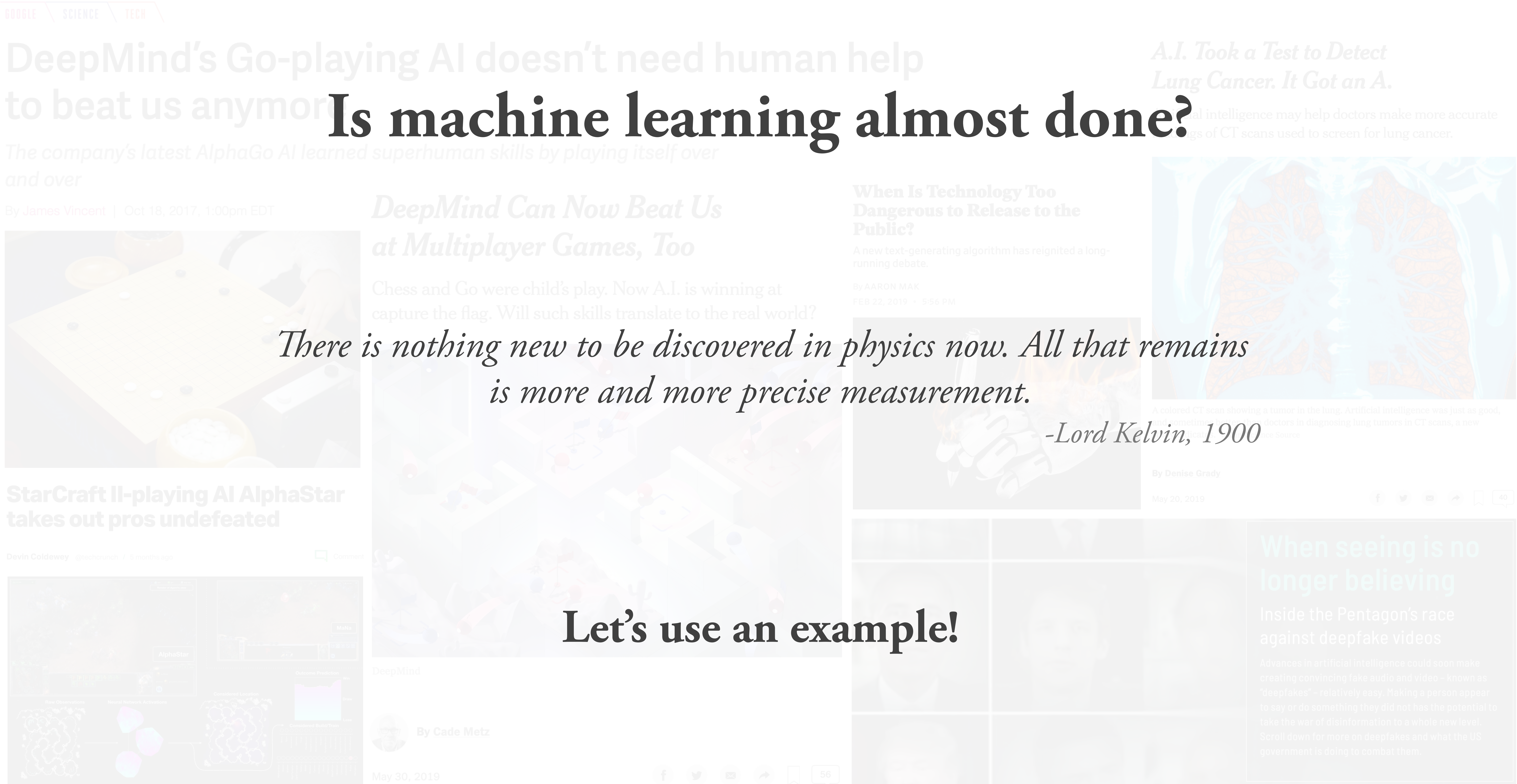
-Lord Kelvin, 1900

Is machine learning almost done?

There is nothing new to be discovered in physics now. All that remains is more and more precise measurement.

-Lord Kelvin, 1900

Let's use an example!



What is missing?

Highway

Example borrowed from Hendrycks et al., 2020.

What is missing?

Highway

ResNet50 Classifier → **Dam (99%)**

Example borrowed from Hendrycks et al., 2020.

What is missing?

Highway

ResNet50 Classifier → **Dam (99%)**

What if a model knew that this is a road?

Example borrowed from Hendrycks et al., 2020.

What is missing?

Highway

ResNet50 Classifier → **Dam (99%)**

What if another model knew that this is a bridge?

What if a model knew that this is a road?

Example borrowed from Hendrycks et al., 2020.

What is missing?

Highway

ResNet50 Classifier → **Dam (99%)**

What if another model knew that this is a bridge?

What if a model knew that this is a road?

What if yet another model knew that roads cannot lead into dams?

Example borrowed from Hendrycks et al., 2020.

What is missing?

Highway

ResNet50 Classifier → **Dam (99%)**

What if another model knew that this is a bridge?

What if a model knew that this is a road?

What if yet another model knew that roads cannot lead into dams?

If these models were able to interact with each other, then this mistake would be highly unlikely!

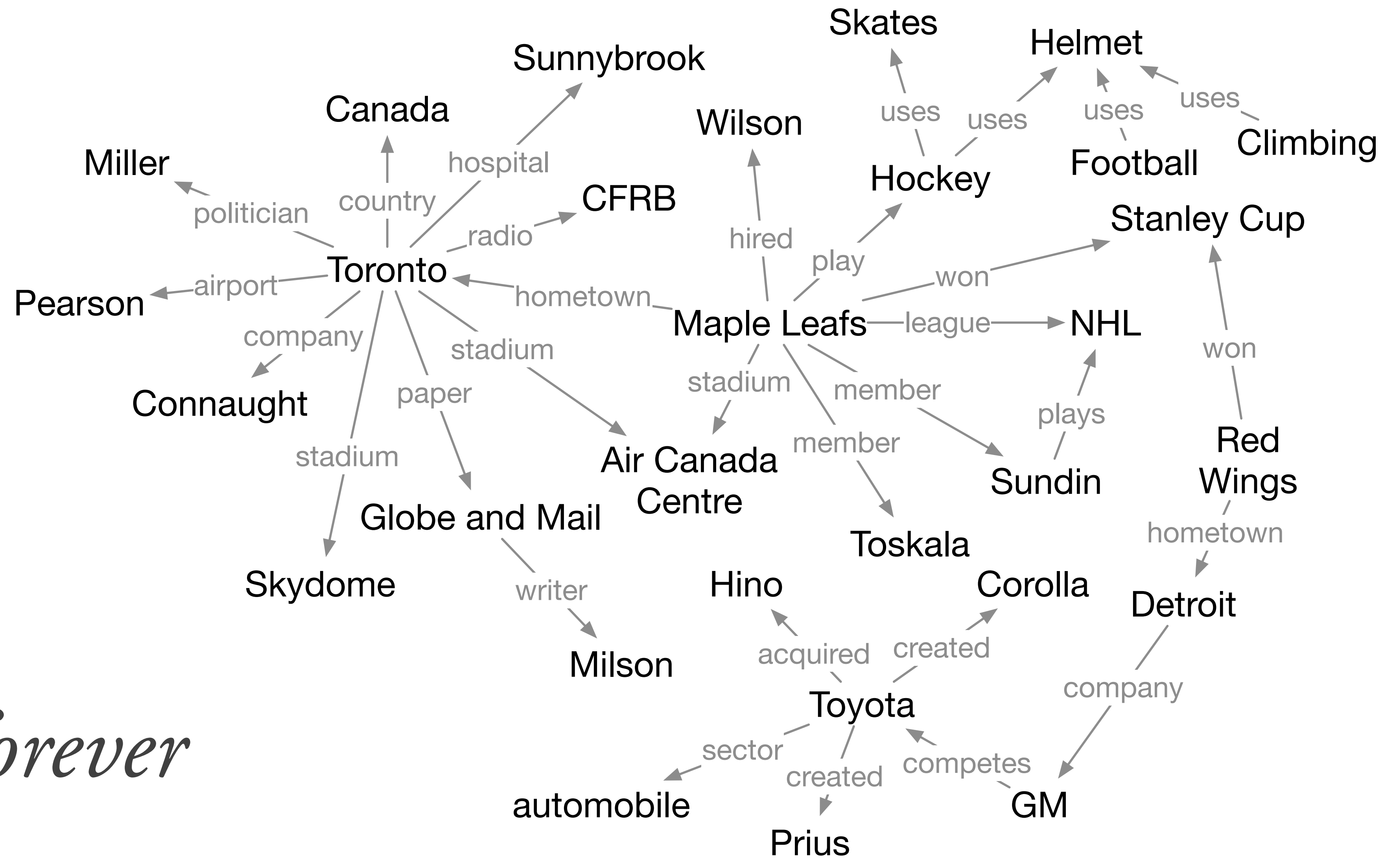
Example borrowed from Hendrycks et al., 2020.

Never-Ending Language Learning

NELL

- Reads websites
- Gets better with time
- Keeps getting better *forever*

Knowledge Base

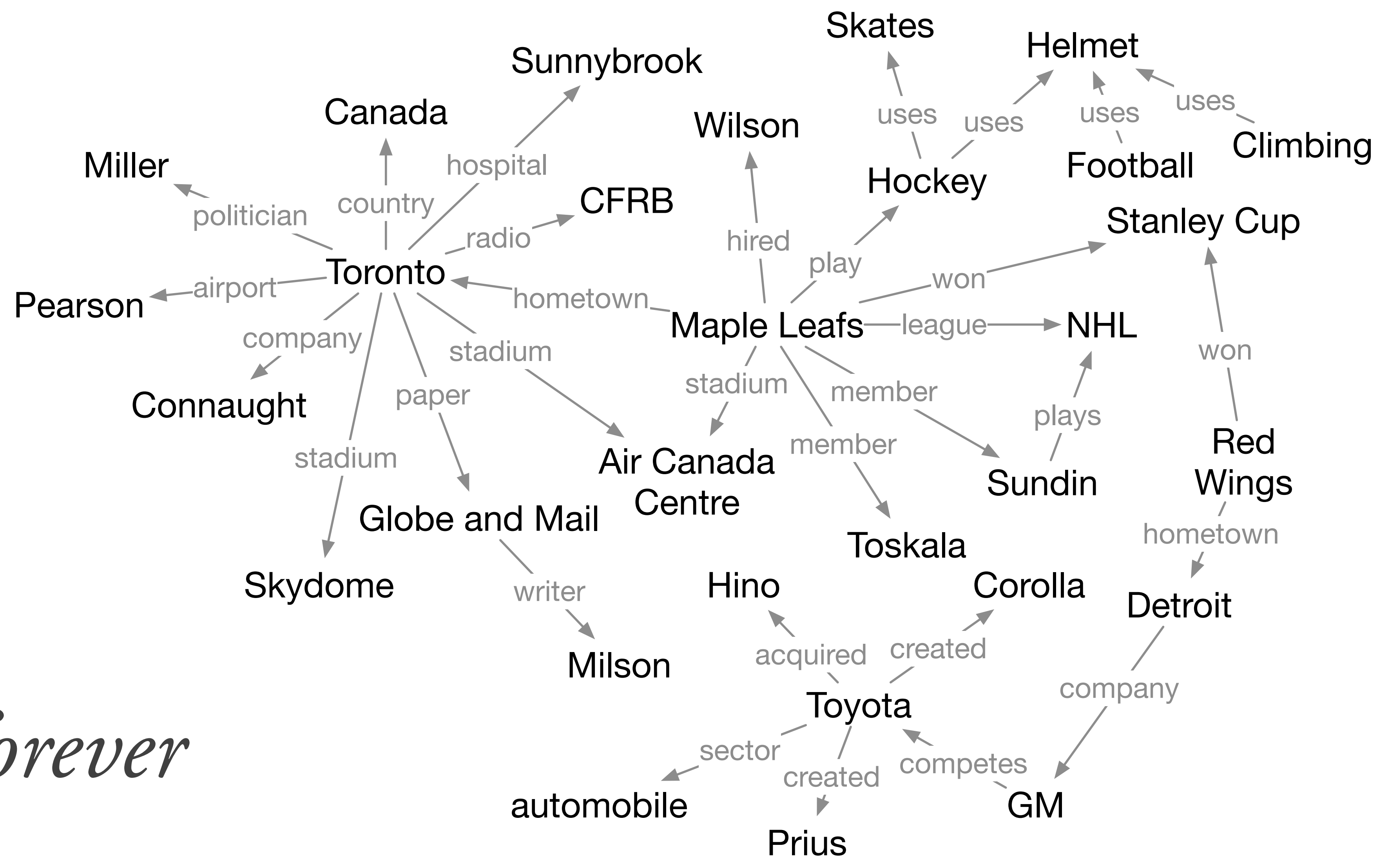


Never-Ending Language Learning

NELL

- Reads websites
- Gets better with time
- Keeps getting better *forever*

Knowledge Base



1.233 billion web pages

~120 million beliefs
~4,100 distinct learning tasks

[Mitchell, ..., **Platanios**, ..., AAAI 2015 & CACM 2018]

Never-Ending Language Learning

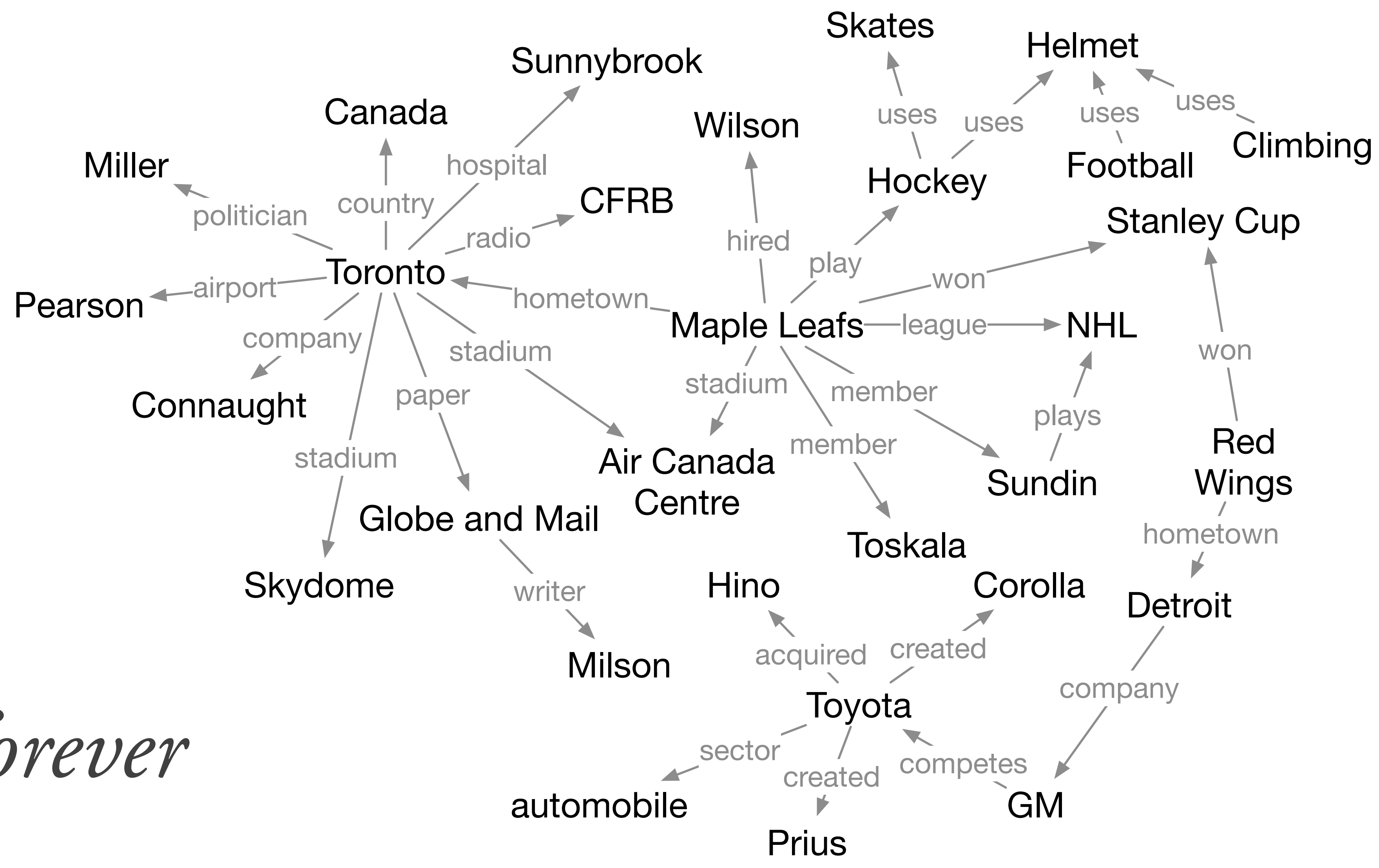
Microsoft

Google amazon

NELL

- Reads websites
- Gets better with time
- Keeps getting better *forever*

Knowledge Base



1.233 billion web pages

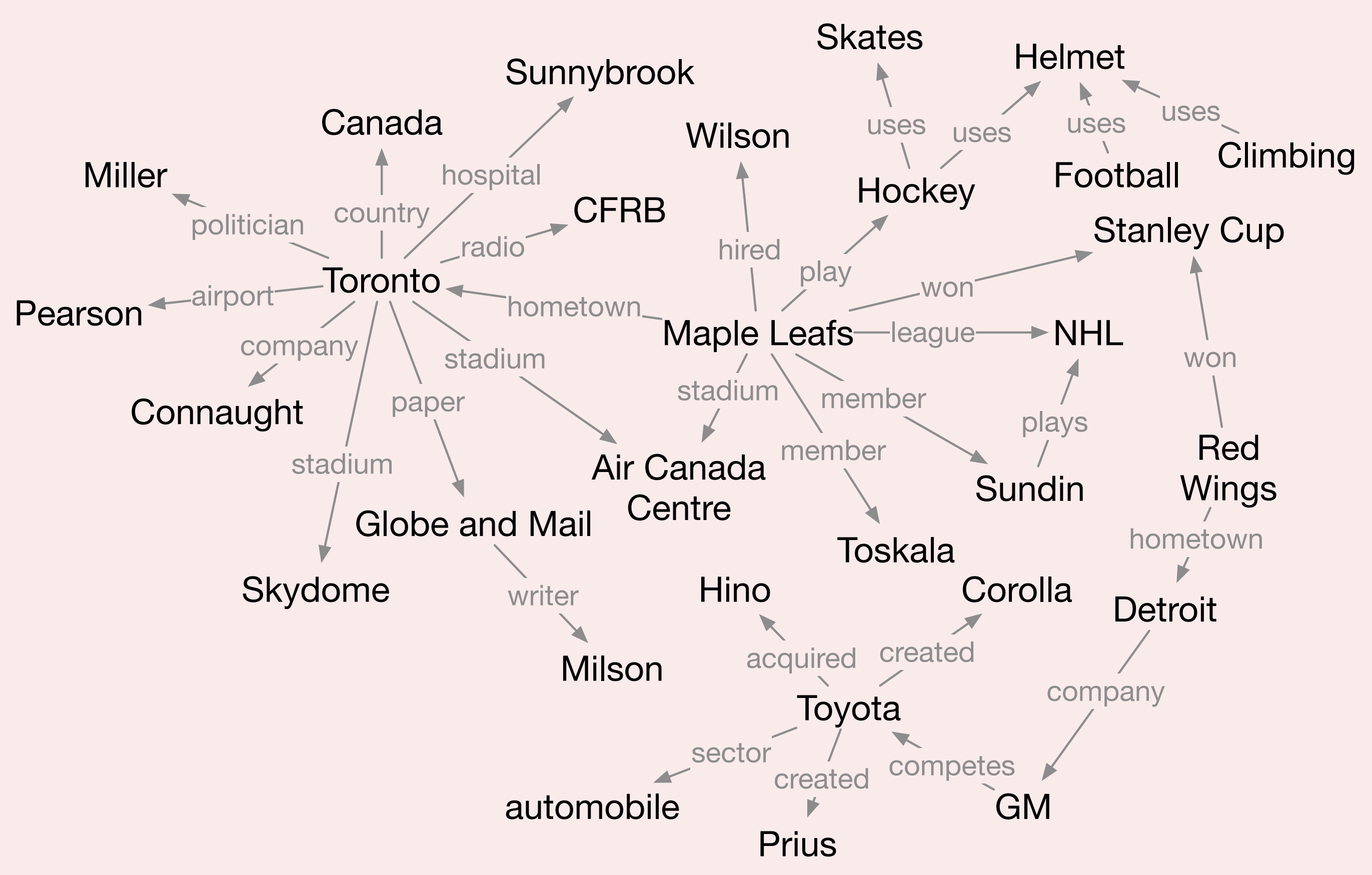
~120 million beliefs
~4,100 distinct learning tasks

Never-Ending Language Learning

Downstream Applications

Conversational Agents

Language Understanding

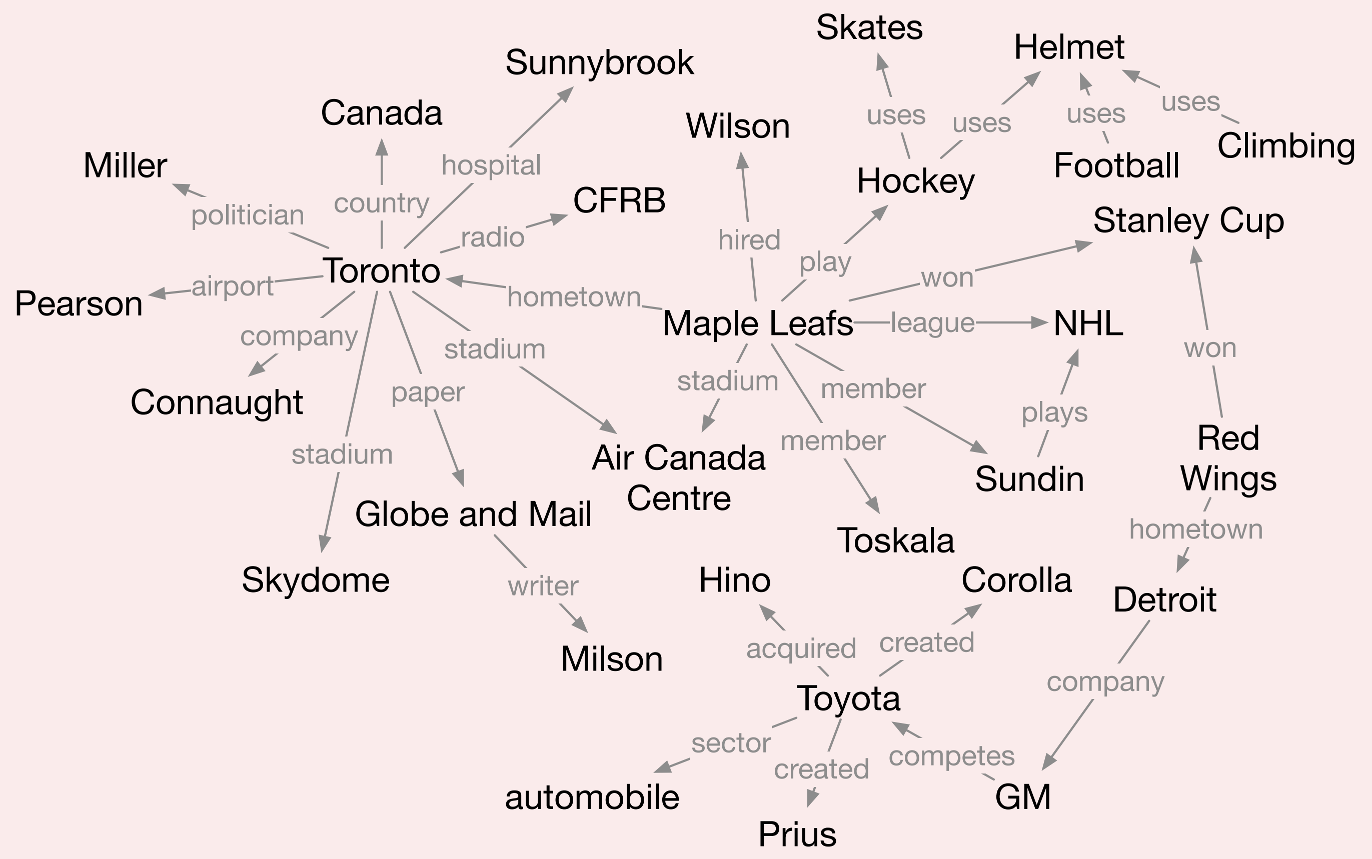


Never-Ending Language Learning

Downstream Applications

Conversational Agents

Language Understanding



Google

amazon

Microsoft

Understanding of Human Learning

Continual / never-ending learning

Multiple diverse types of knowledge and tasks

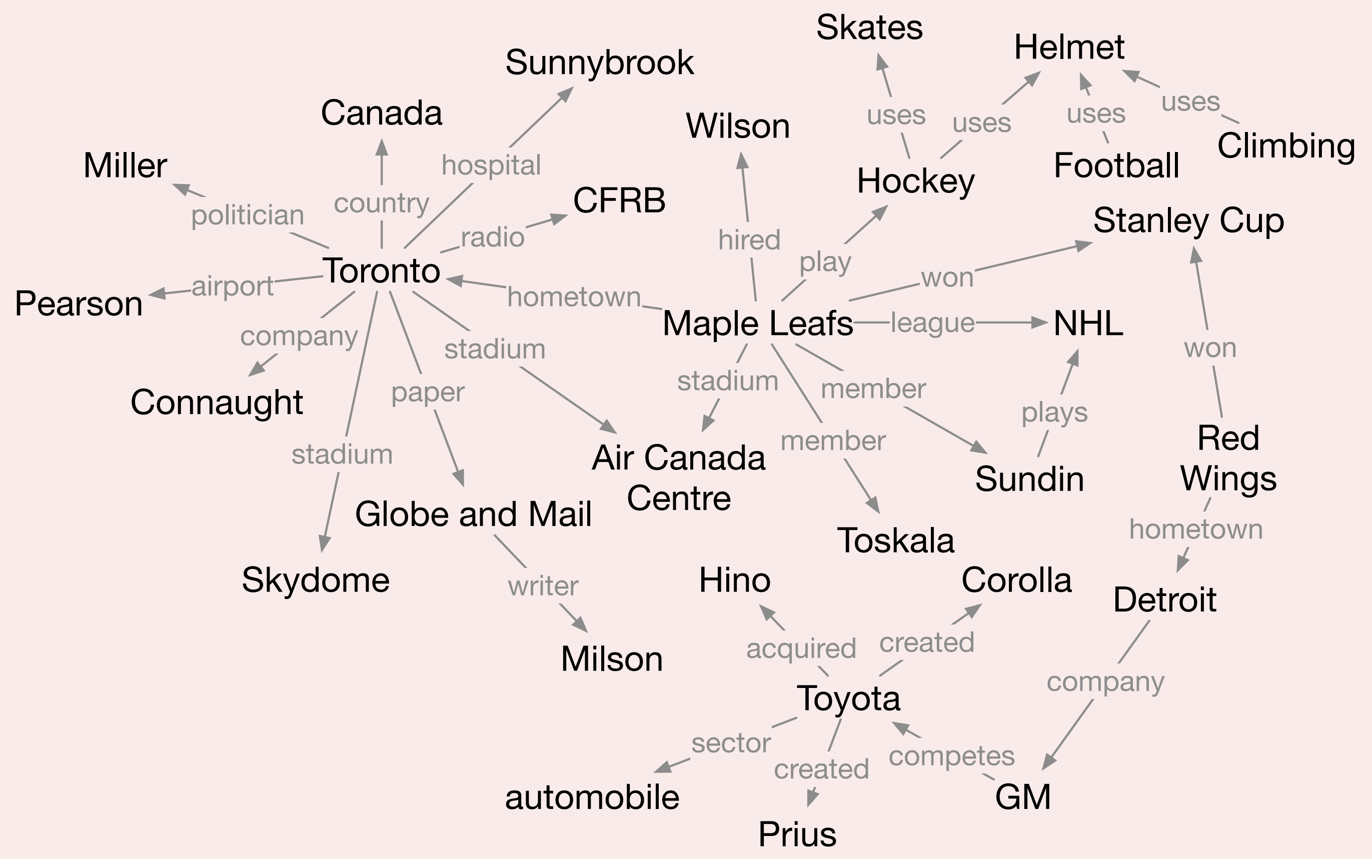
Mostly *self-supervised learning*

…

Downstream Applications

Conversational Agents

Language Understanding



Google amazon Microsoft

Never-Ending Language Learning

Understanding of Human Learning

Continual / never-ending learning

Multiple diverse types of knowledge and tasks

Mostly *self-supervised learning*

…

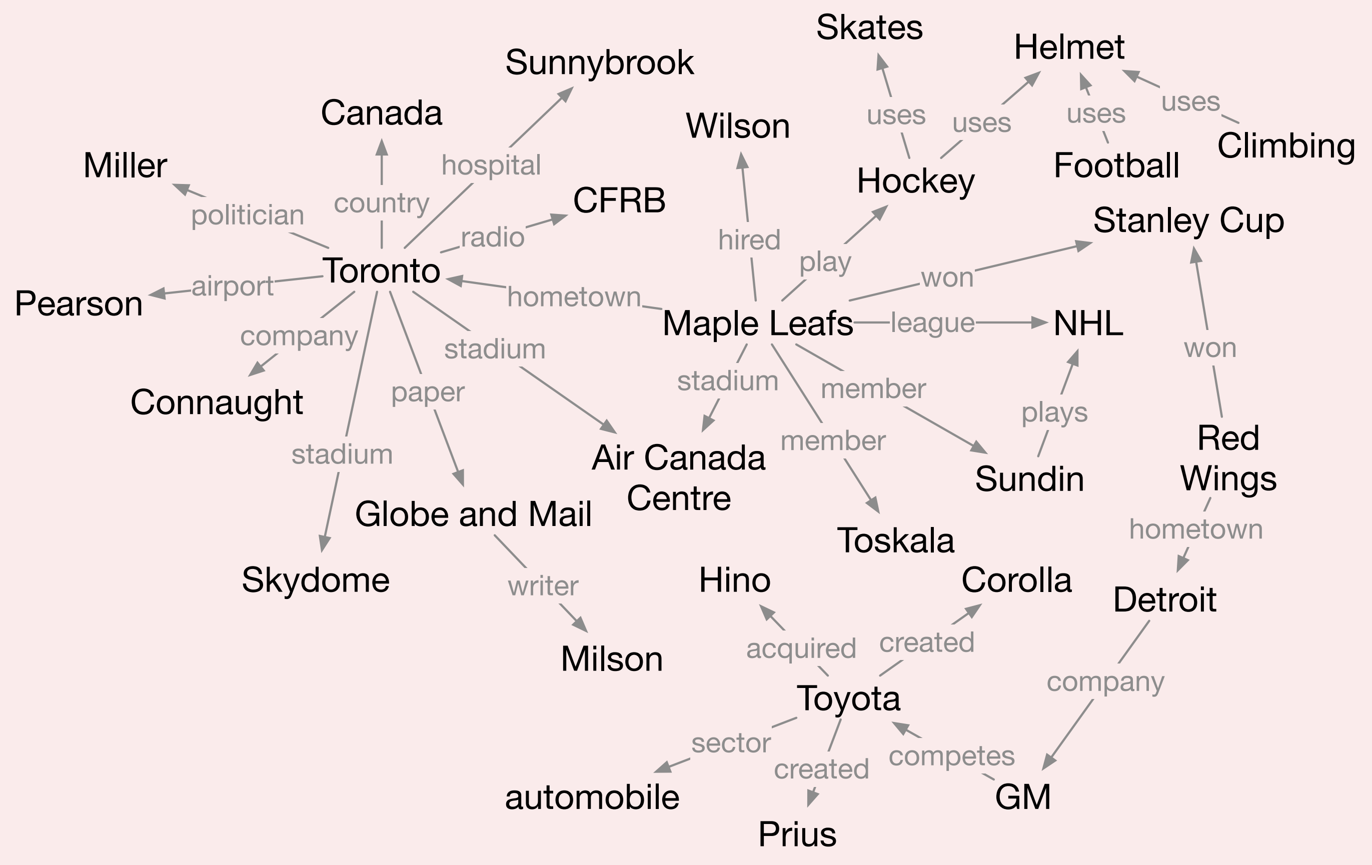
NELL is facing multiple difficult challenges!

Let us first see how it works…

Downstream Applications

Conversational Agents

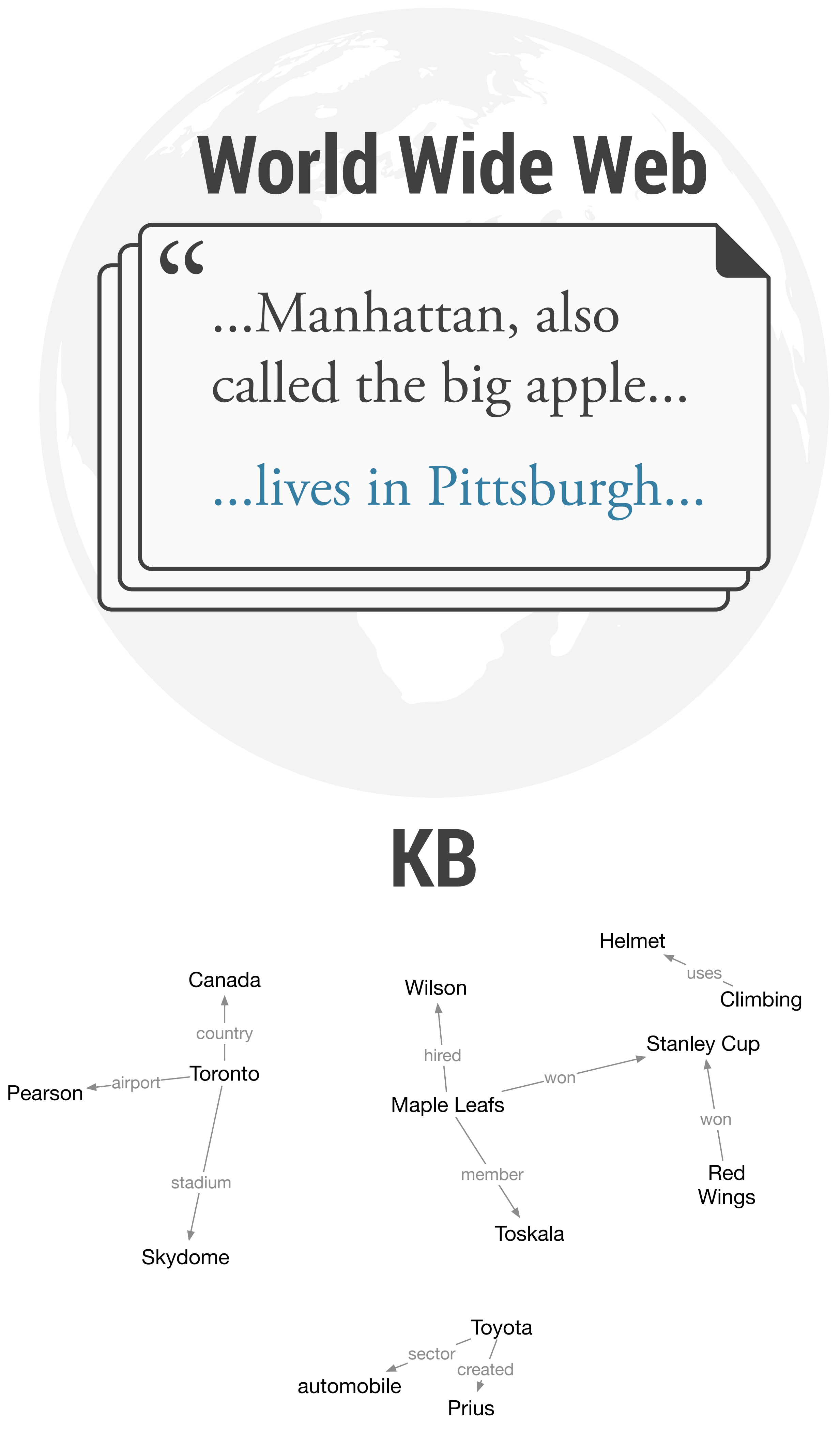
Language Understanding



Never-Ending Language Learning

NELL always has access to the world-wide web.

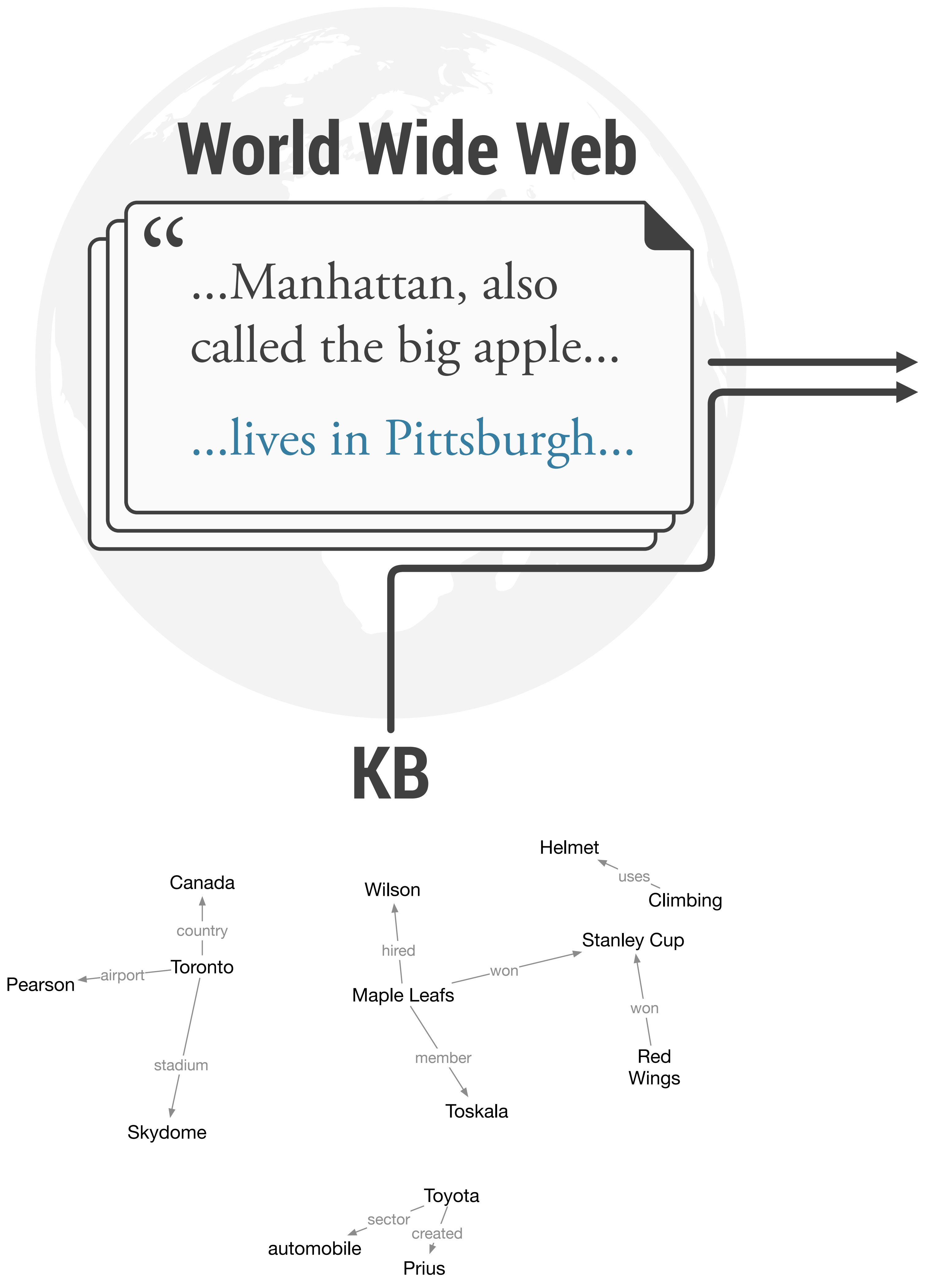
Never-Ending Language Learning



It also starts with a *very small number* of externally provided facts.

[Mitchell, ..., **Platanios**, ..., AAAI 2015 & CACM 2018]

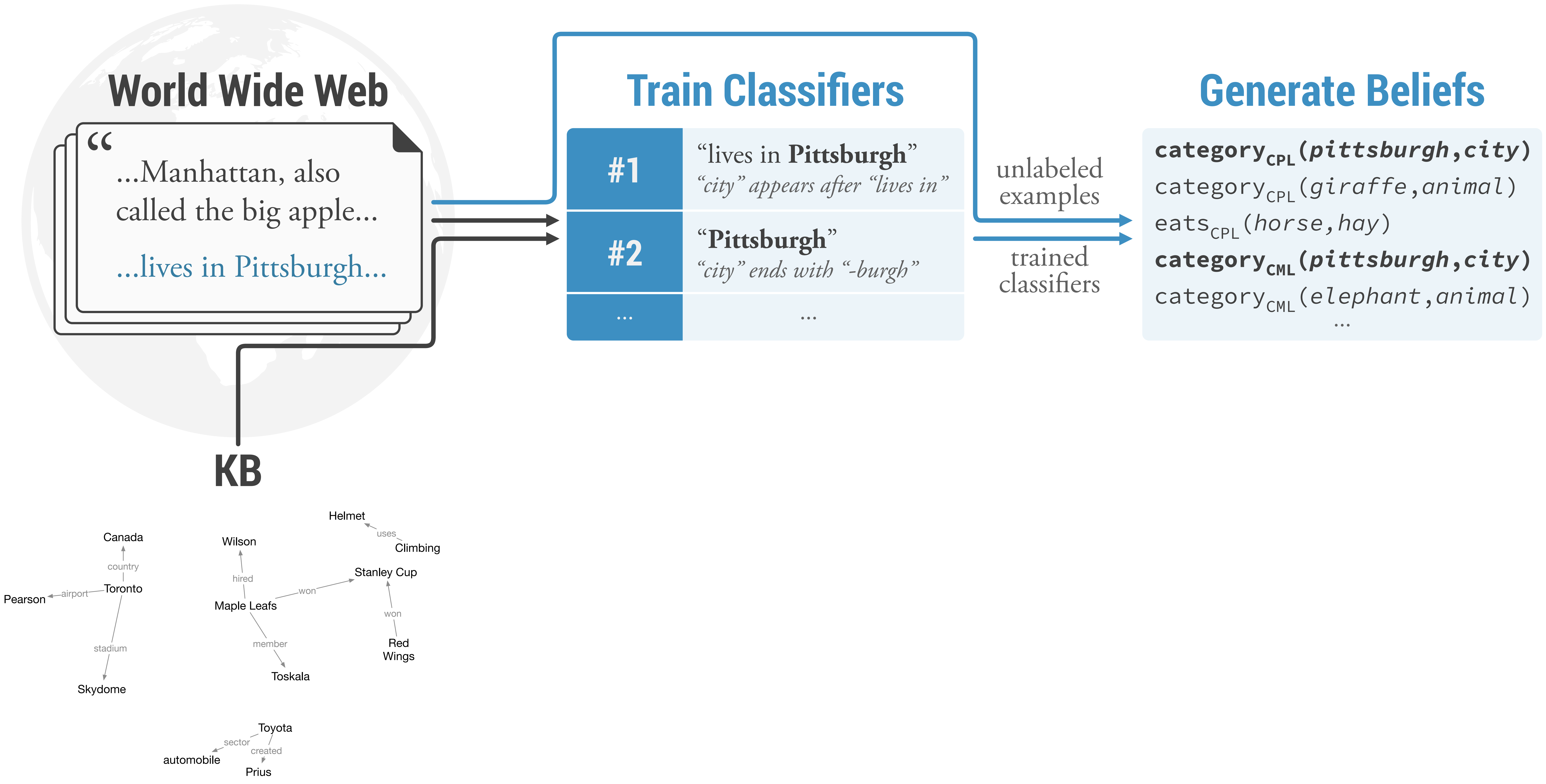
Never-Ending Language Learning



Train Classifiers

#1	“lives in **Pittsburgh**” *“city” appears after “lives in”*	Classifier using context of noun phrases
#2	“**Pittsburgh**” *“city” ends with “-burgh”*	Classifier using morphology of noun phrases
…	…	

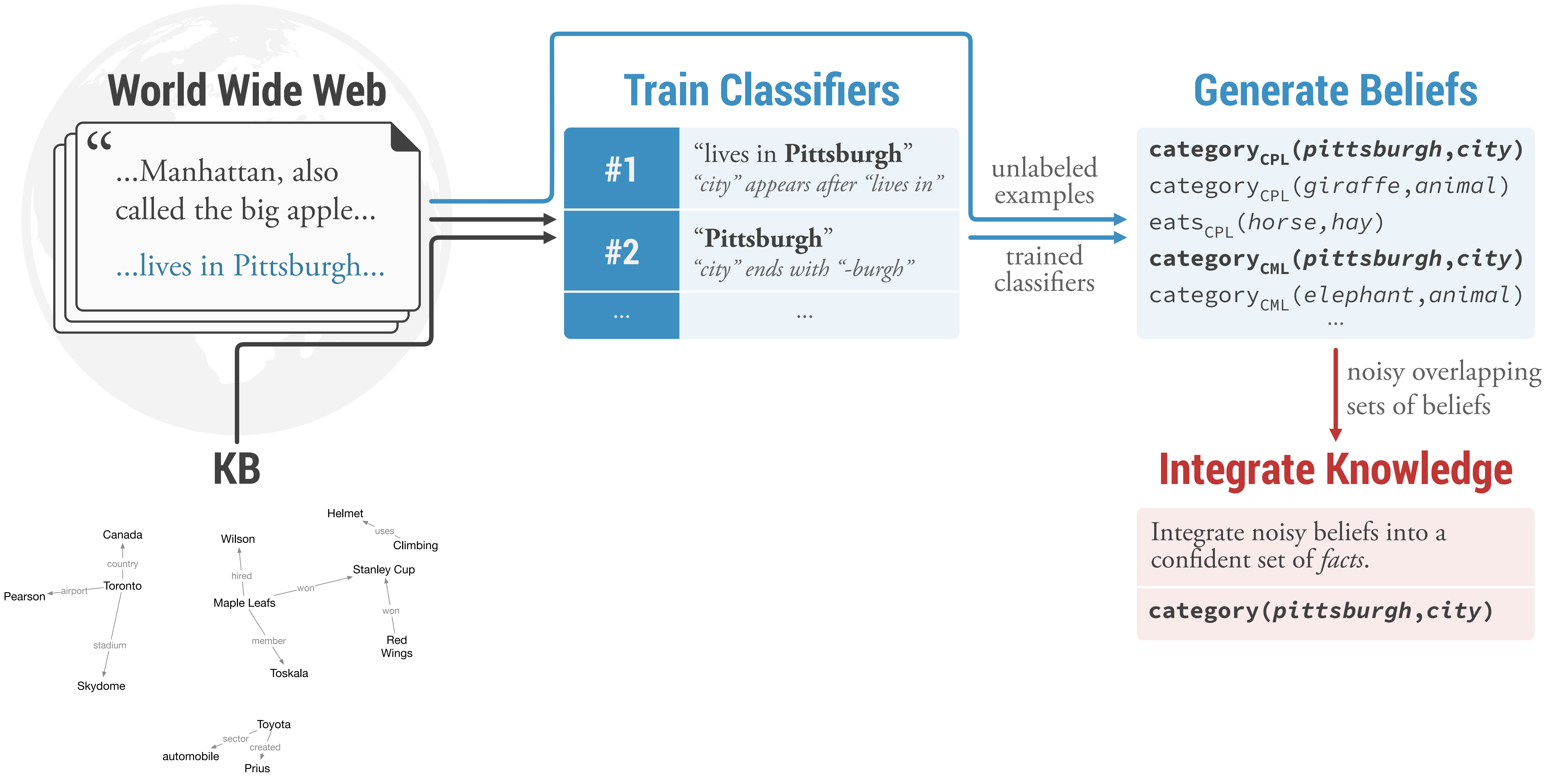
Never-Ending Language Learning



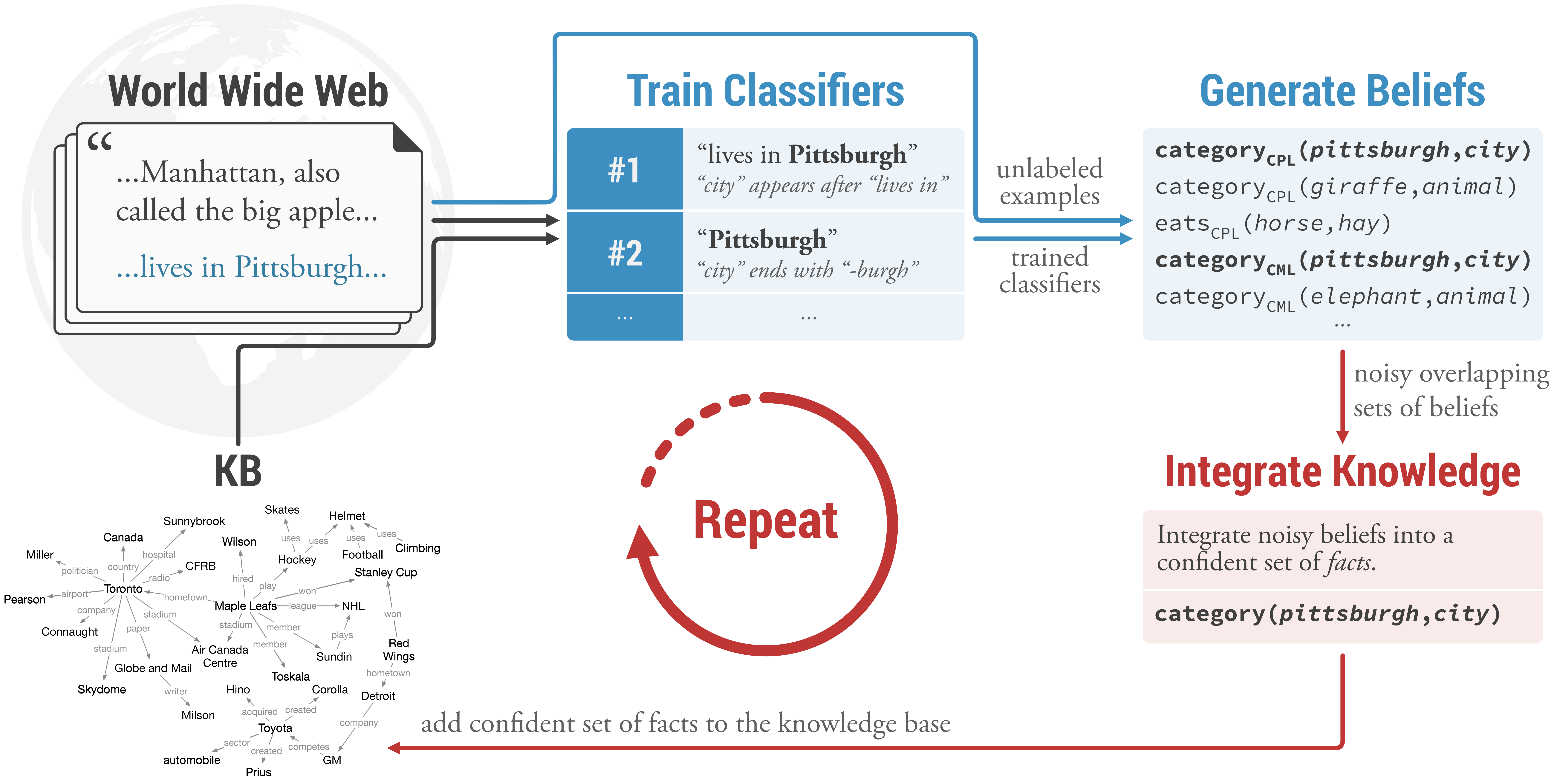
[Mitchell, …, **Platanios**, …, AAAI 2015 & CACM 2018]

Never-Ending Language Learning

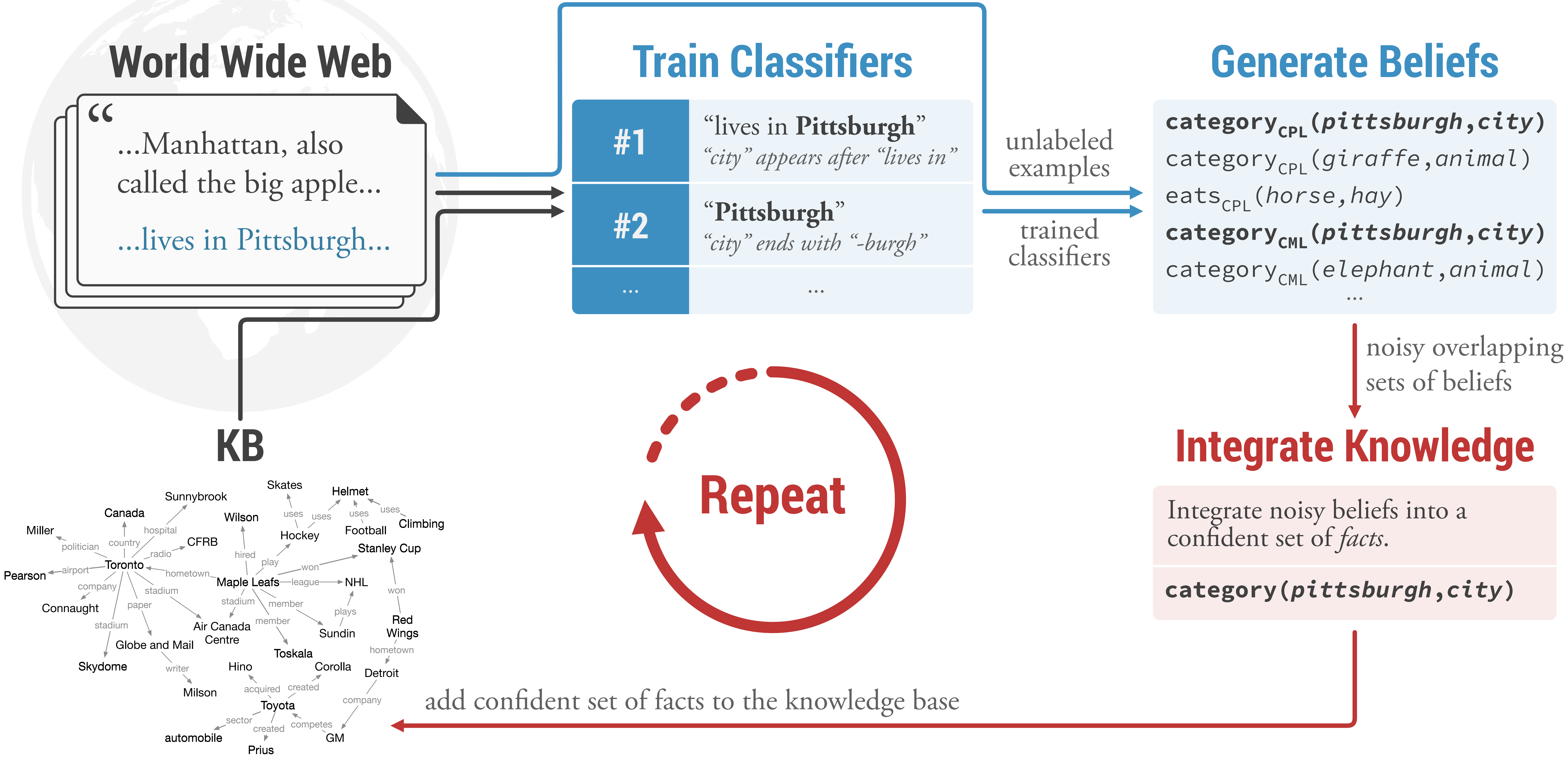
Architecture



Never-Ending Language Learning



Never-Ending Language Learning



[Blum and Mitchell, ACM 1998]
[Mitchell, ..., **Platanios**, ..., AAAI 2015 & CACM 2018]

category$_{CPL}$(*pittsburgh*,*city*)
category$_{CPL}$(*giraffe*,*animal*)
eats$_{CPL}$(*horse*,*hay*)
category$_{CML}$(*pittsburgh*,*city*)
category$_{CML}$(*elephant*,*animal*)
...

noisy overlapping sets of beliefs

This step is crucial!
Errors can accumulate and confident mistakes will ruin long-term performance!

Integrate Knowledge

Integrate noisy beliefs into a confident set of *facts*.

category(*pittsburgh*,*city*)

add confident set of facts to the knowledge base

Never-Ending Language Learning

NELL

Generate Beliefs

$\mathtt{category}_{CPL}(\mathit{pittsburgh},\mathit{city})$

$\mathtt{category}_{CPL}(\mathit{giraffe},\mathit{animal})$

$\mathtt{eats}_{CPL}(\mathit{horse},\mathit{hay})$

$\mathtt{category}_{CML}(\mathit{pittsburgh},\mathit{city})$

$\mathtt{category}_{CML}(\mathit{elephant},\mathit{animal})$

…

noisy overlapping sets of beliefs

Integrate Knowledge

Integrate noisy beliefs into a confident set of *facts*.

$\mathtt{category}(\mathit{pittsburgh},\mathit{city})$

confident facts that will be added to the knowledge base

Never-Ending Language Learning

Machine Learning

Data Programming

snorkel

DeepDive

weak supervision

noisy overlapping sets of labels

Crowdsourcing

amazon mechanical turk

figure eight

HIVE

…

noisy overlapping sets of labels

Aggregate Labels

Aggregate the noisy labels to obtain a single label per example.

use aggregated labels to train machine learning models

NELL

Generate Beliefs

$\mathbf{category}_{\mathbf{CPL}}(\boldsymbol{pittsburgh},\boldsymbol{city})$

$\mathtt{category}_{\mathtt{CPL}}(\mathit{giraffe},\mathit{animal})$

$\mathtt{eats}_{\mathtt{CPL}}(\mathit{horse},\mathit{hay})$

$\mathbf{category}_{\mathbf{CML}}(\boldsymbol{pittsburgh},\boldsymbol{city})$

$\mathtt{category}_{\mathtt{CML}}(\mathit{elephant},\mathit{animal})$

…

noisy overlapping sets of beliefs

Integrate Knowledge

Integrate noisy beliefs into a confident set of *facts*.

category(*pittsburgh*,*city*)

confident facts that will be added to the knowledge base

Never-Ending Language Learning

Machine Learning

Data Programming

snorkel

DeepDive

weak supervision

Crowdsourcing

amazon mechanical turk

figure eight

HIVE

...

noisy overlapping sets of labels

noisy overlapping sets of labels

Aggregate Labels

Aggregate the noisy labels to obtain a single label per example.

use aggregated labels to train machine learning models

NELL

Generate Beliefs

category$_{\text{CPL}}$(*pittsburgh*,*city*)
category$_{\text{CPL}}$(*giraffe*,*animal*)
eats$_{\text{CPL}}$(*horse*,*hay*)
category$_{\text{CML}}$(*pittsburgh*,*city*)
category$_{\text{CML}}$(*elephant*,*animal*)
...

noisy overlapping sets of beliefs

Integrate Knowledge

Integrate noisy beliefs into a confident set of *facts*.

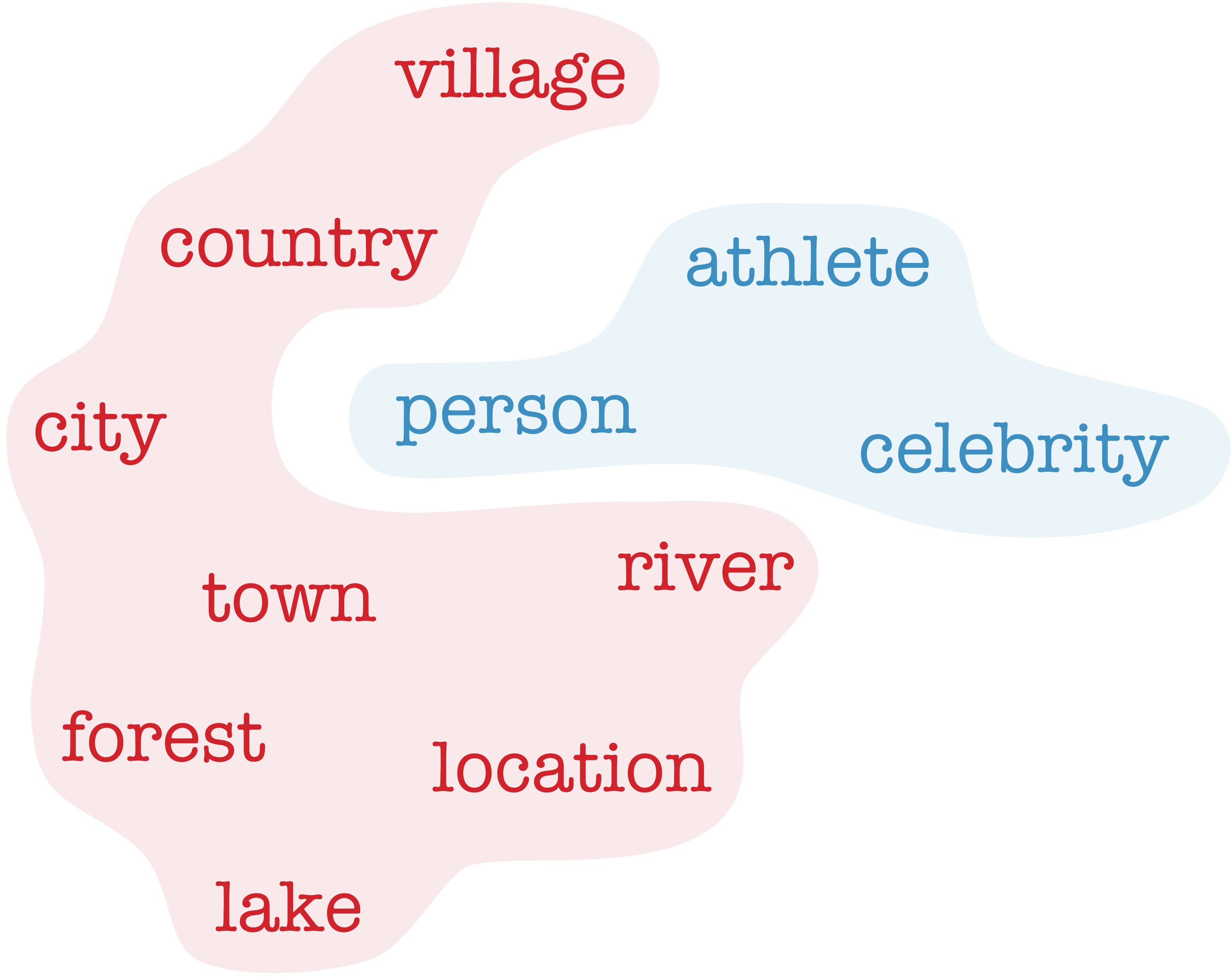
category(*pittsburgh*,*city*)

confident facts that will be added to the knowledge base

This problem is important for machine learning as a whole!

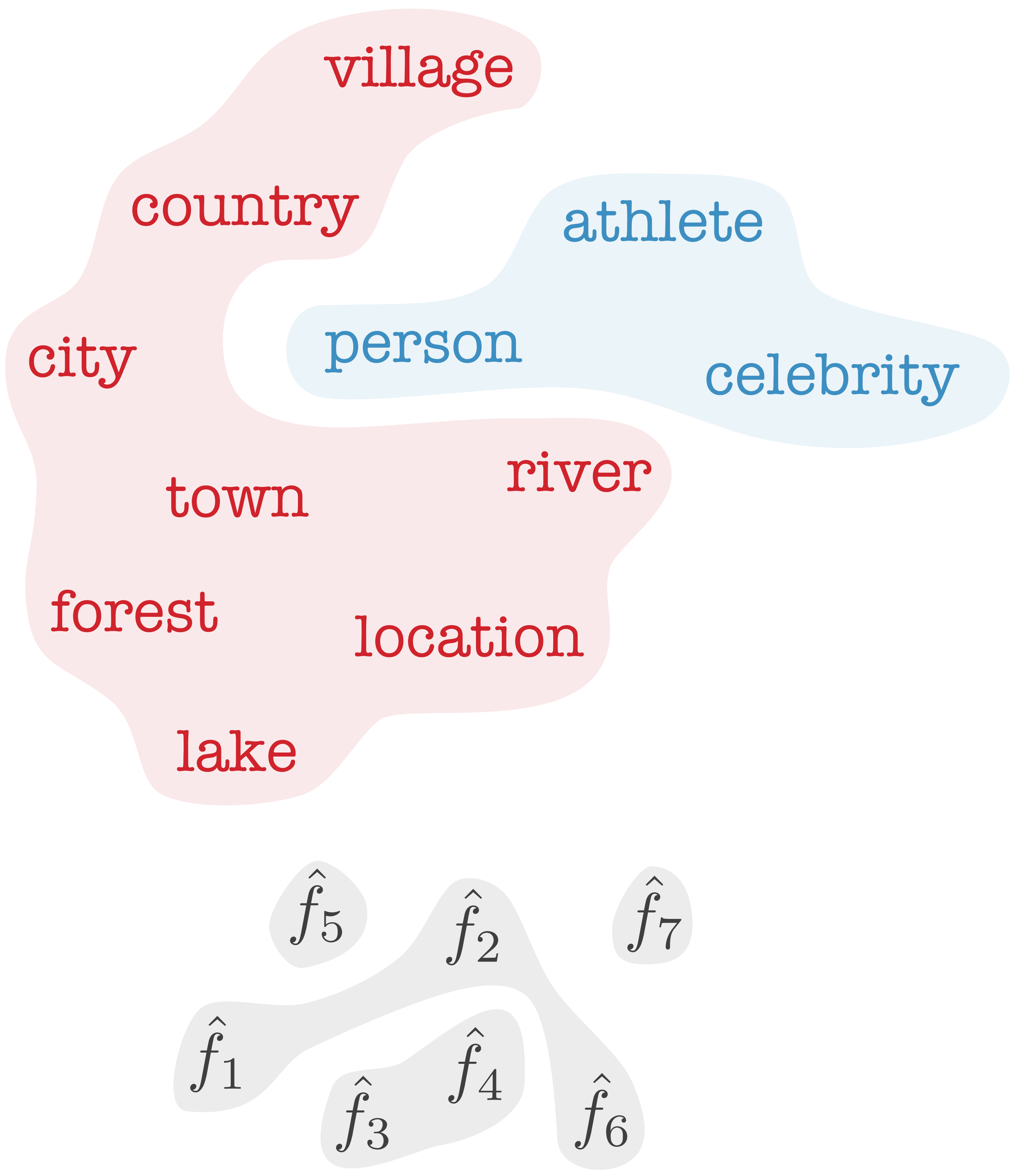
Limitation #1

Dependencies



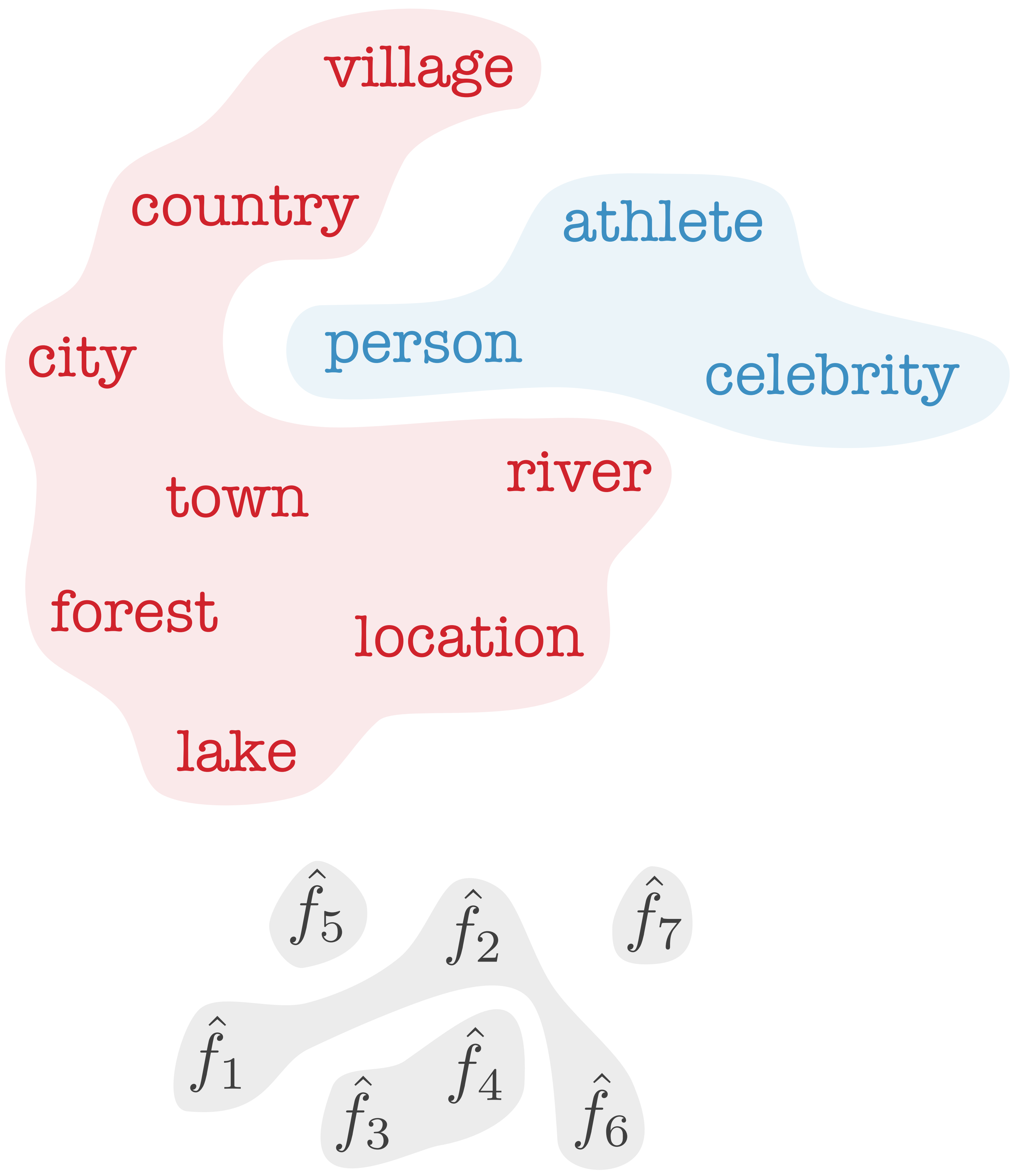
Limitation #1

Dependencies

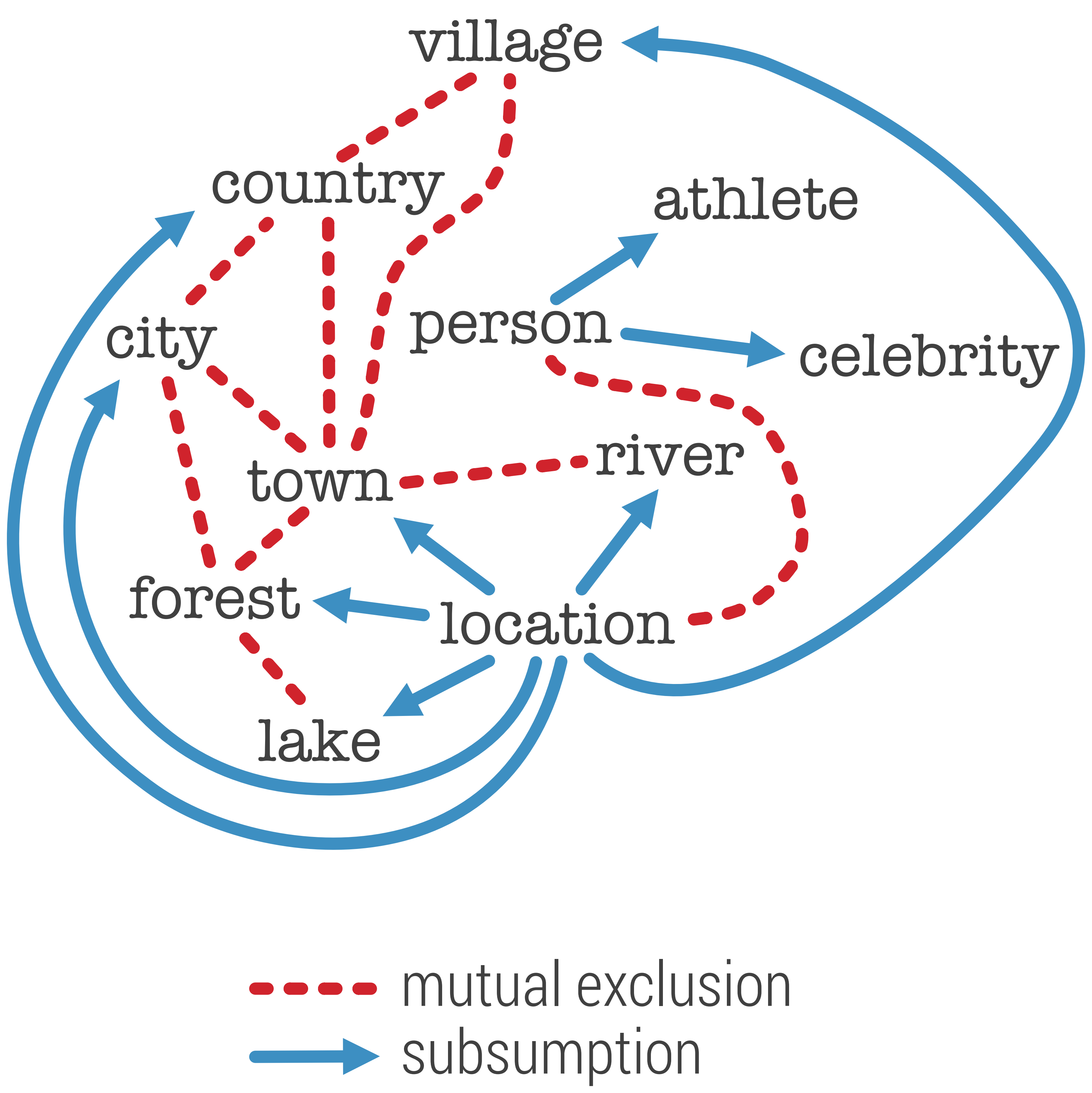


Self-Reflection

Limitation #1
Dependencies



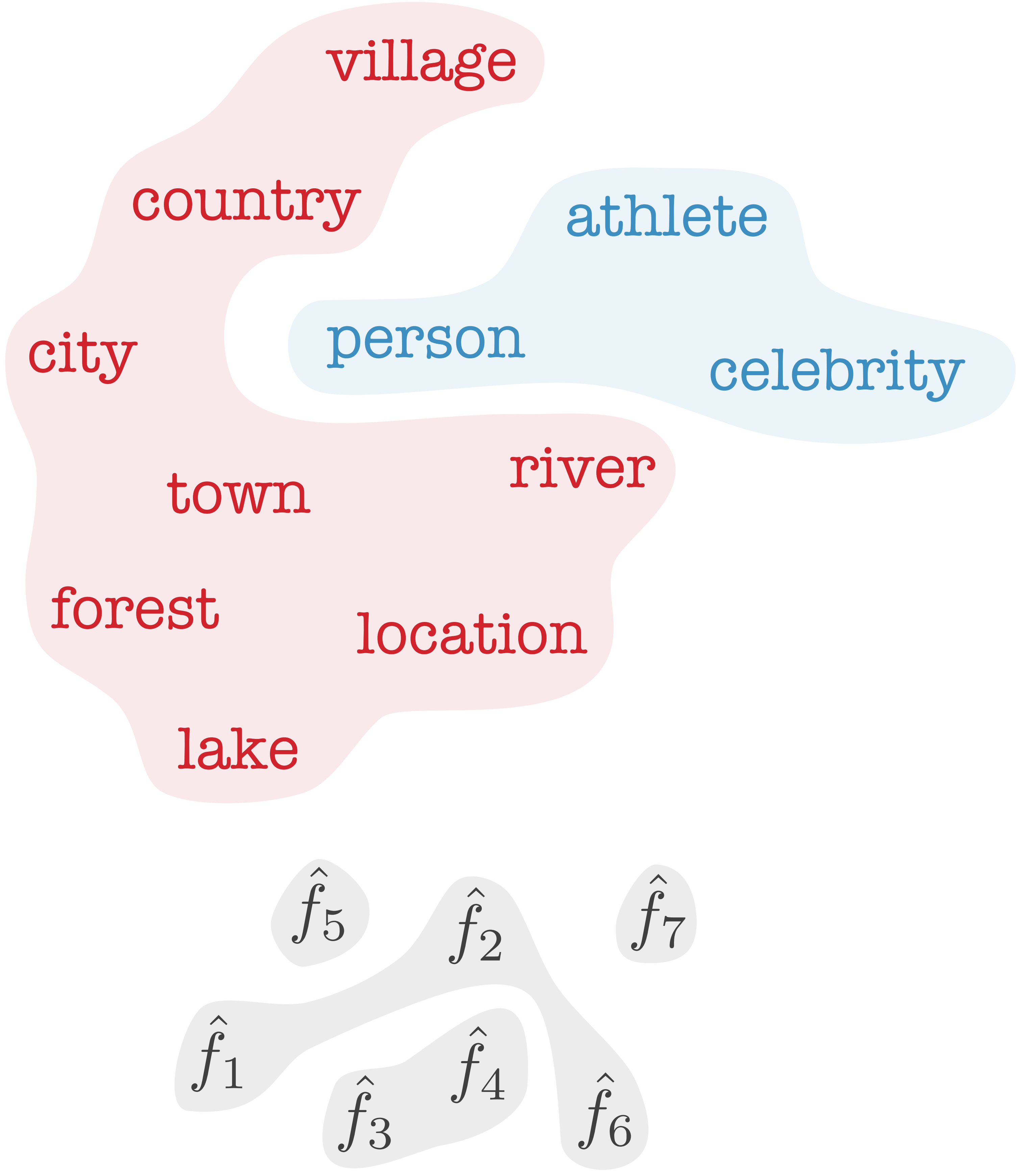
Limitation #2
Logical Constraints



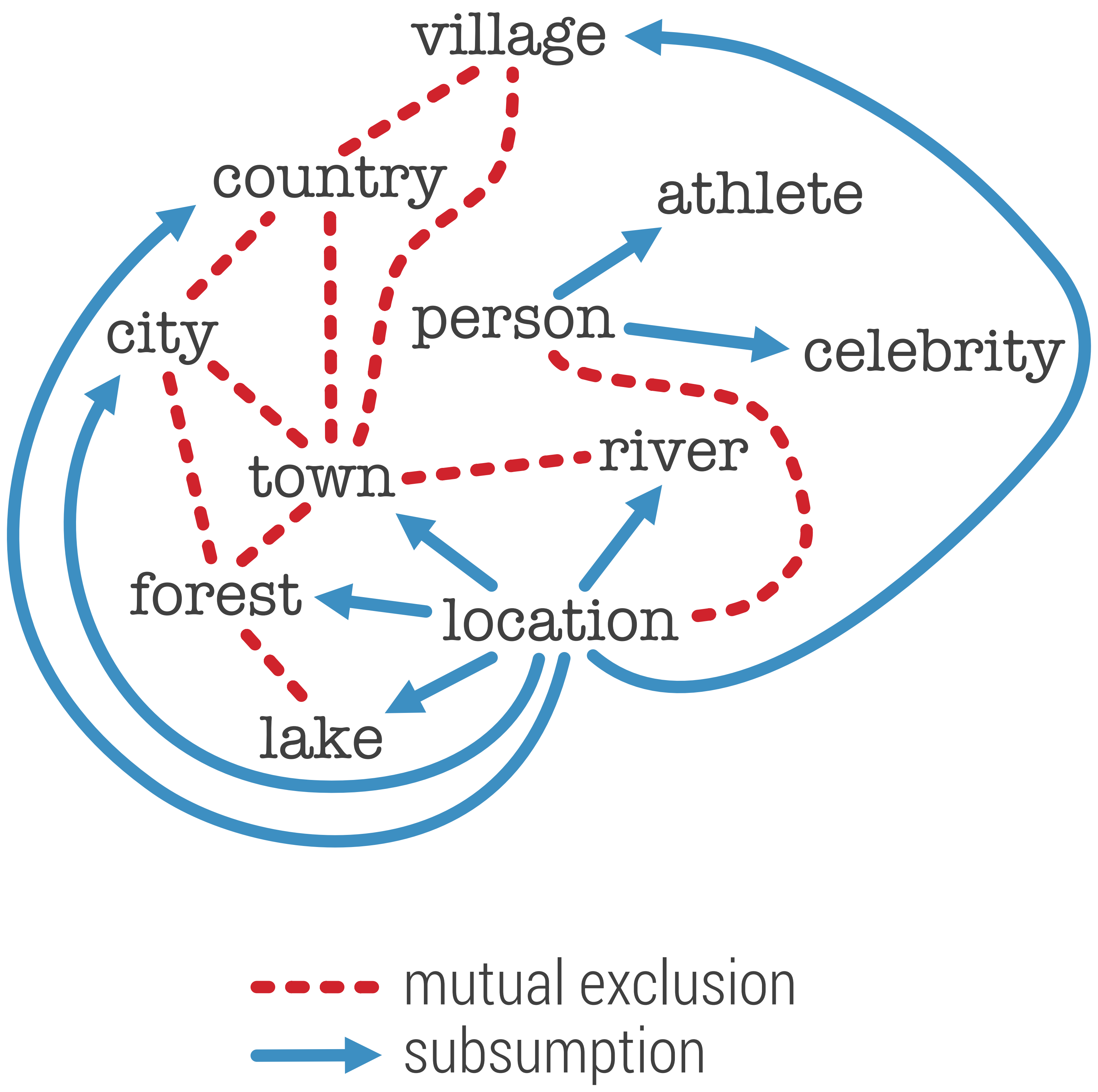
Self-Reflection

A Direct Approach

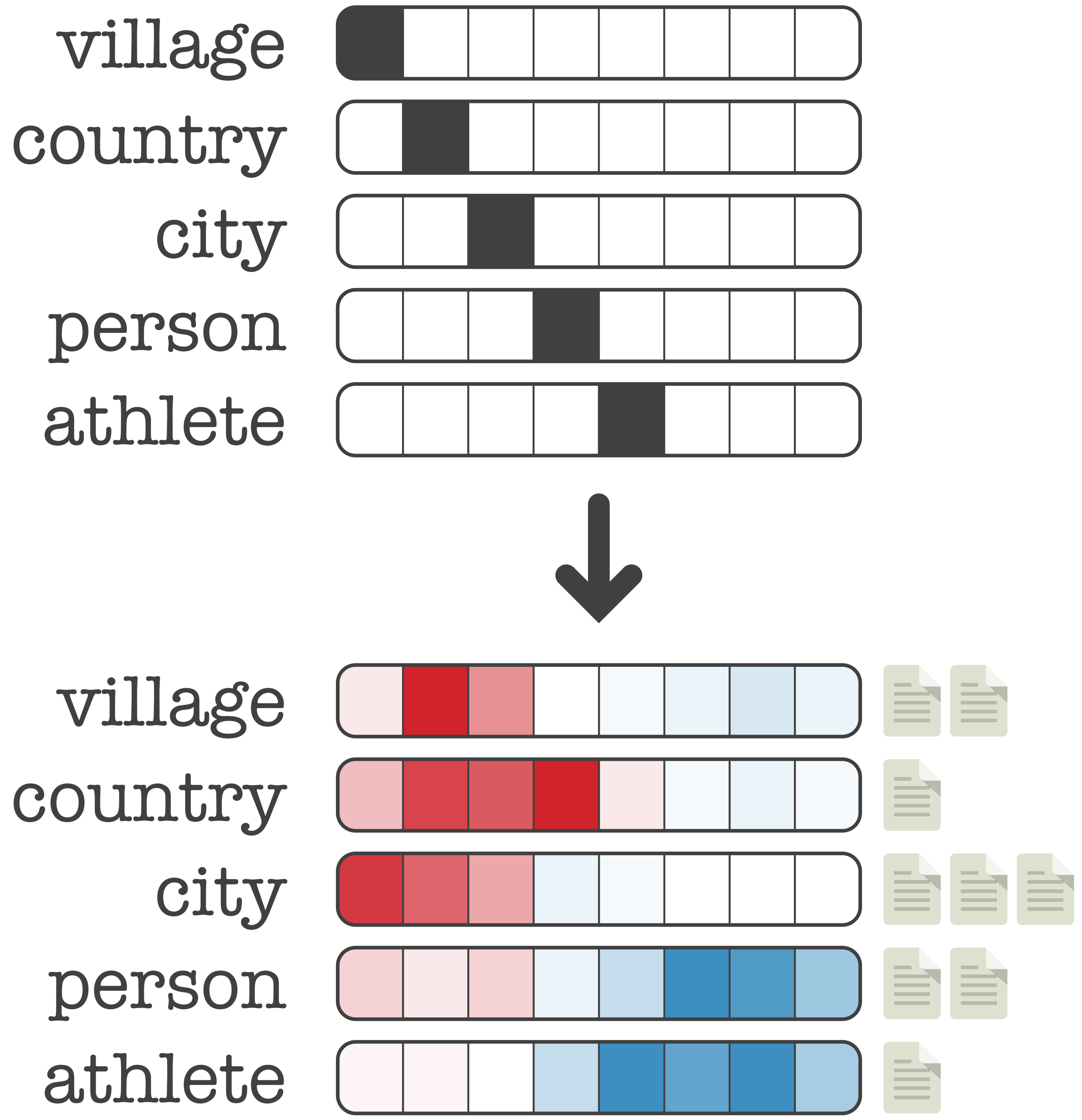
Limitation #1
Dependencies



Limitation #2
Logical Constraints

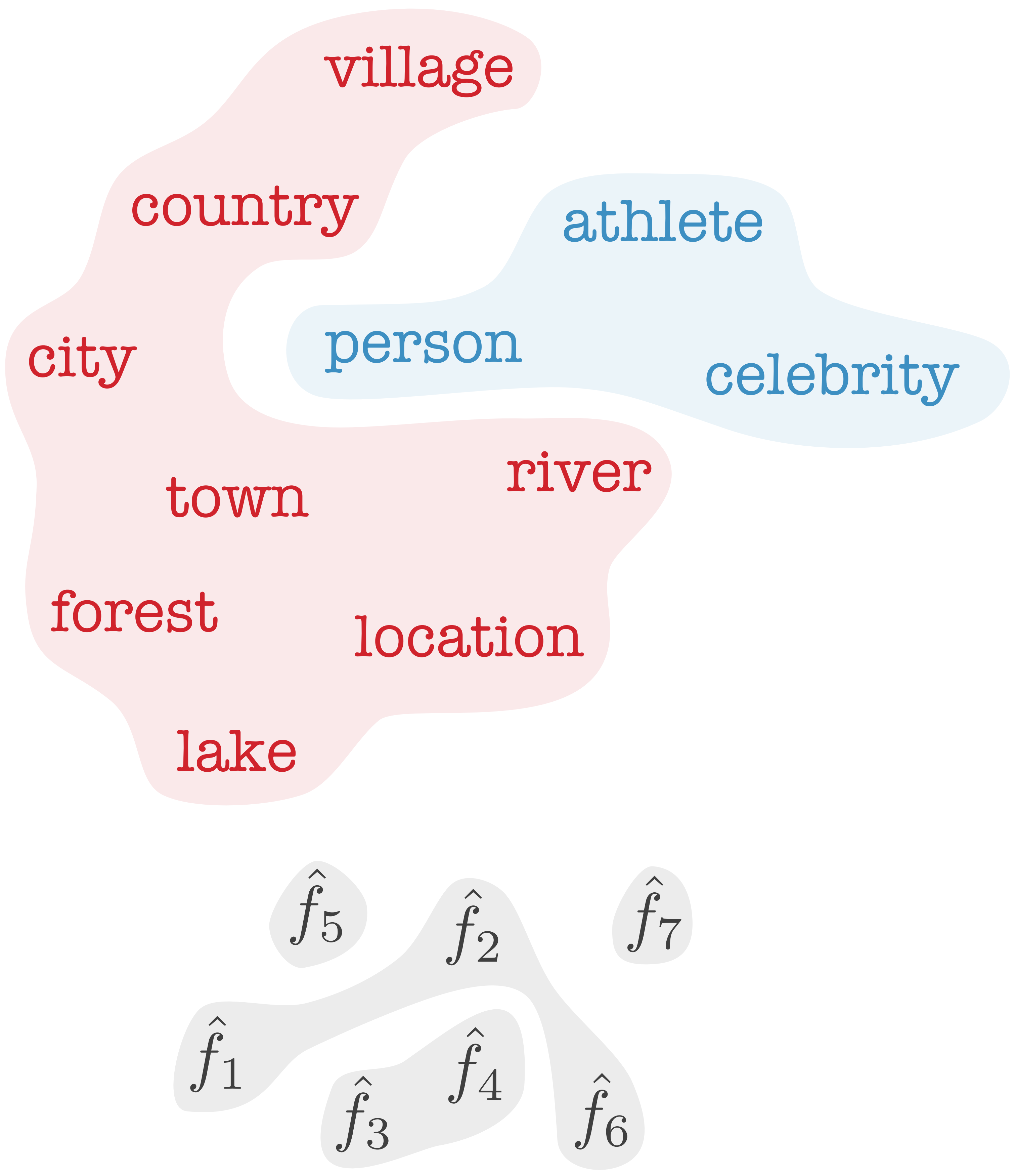


Limitation #3
Representations

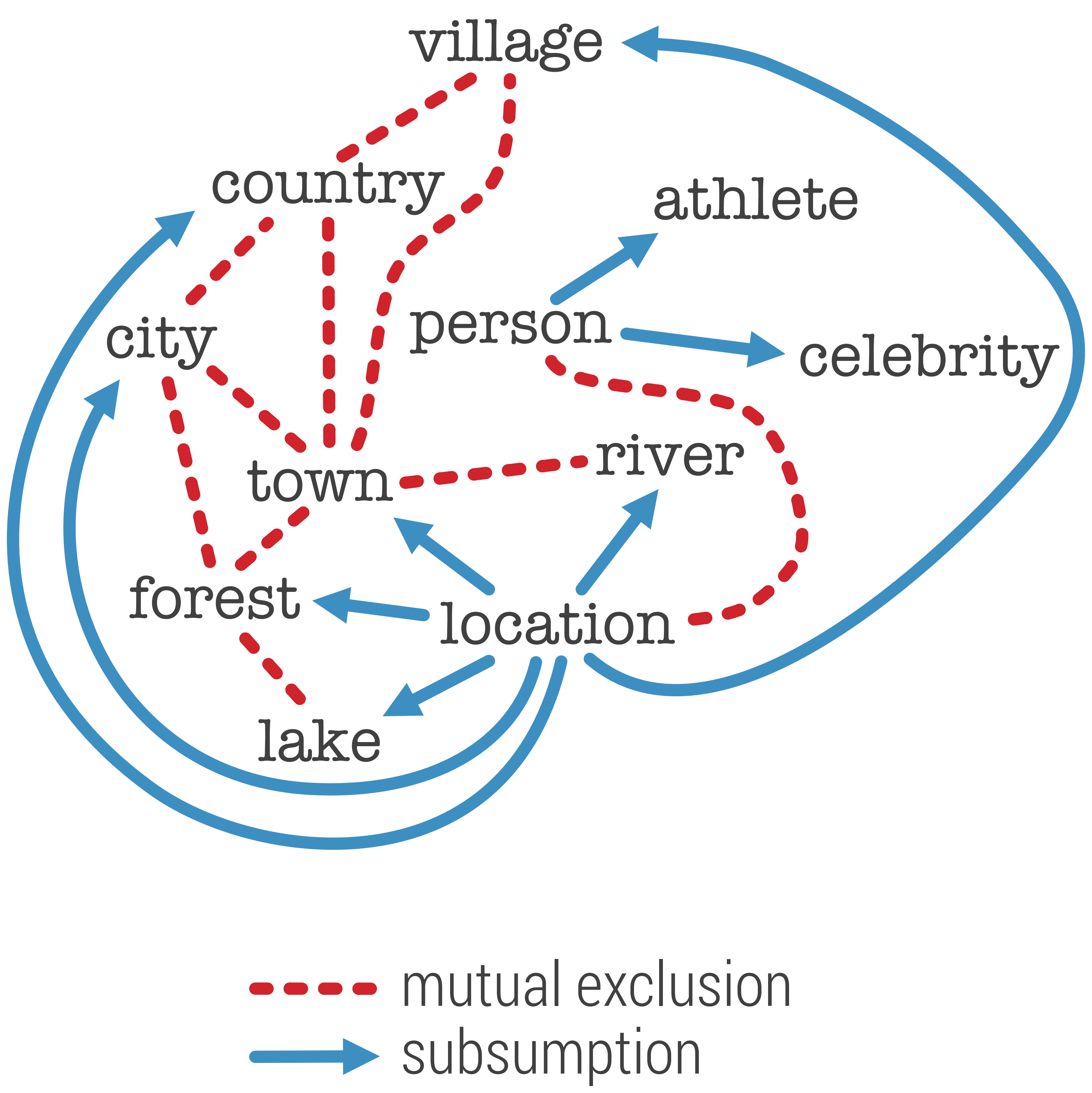


Self-Reflection

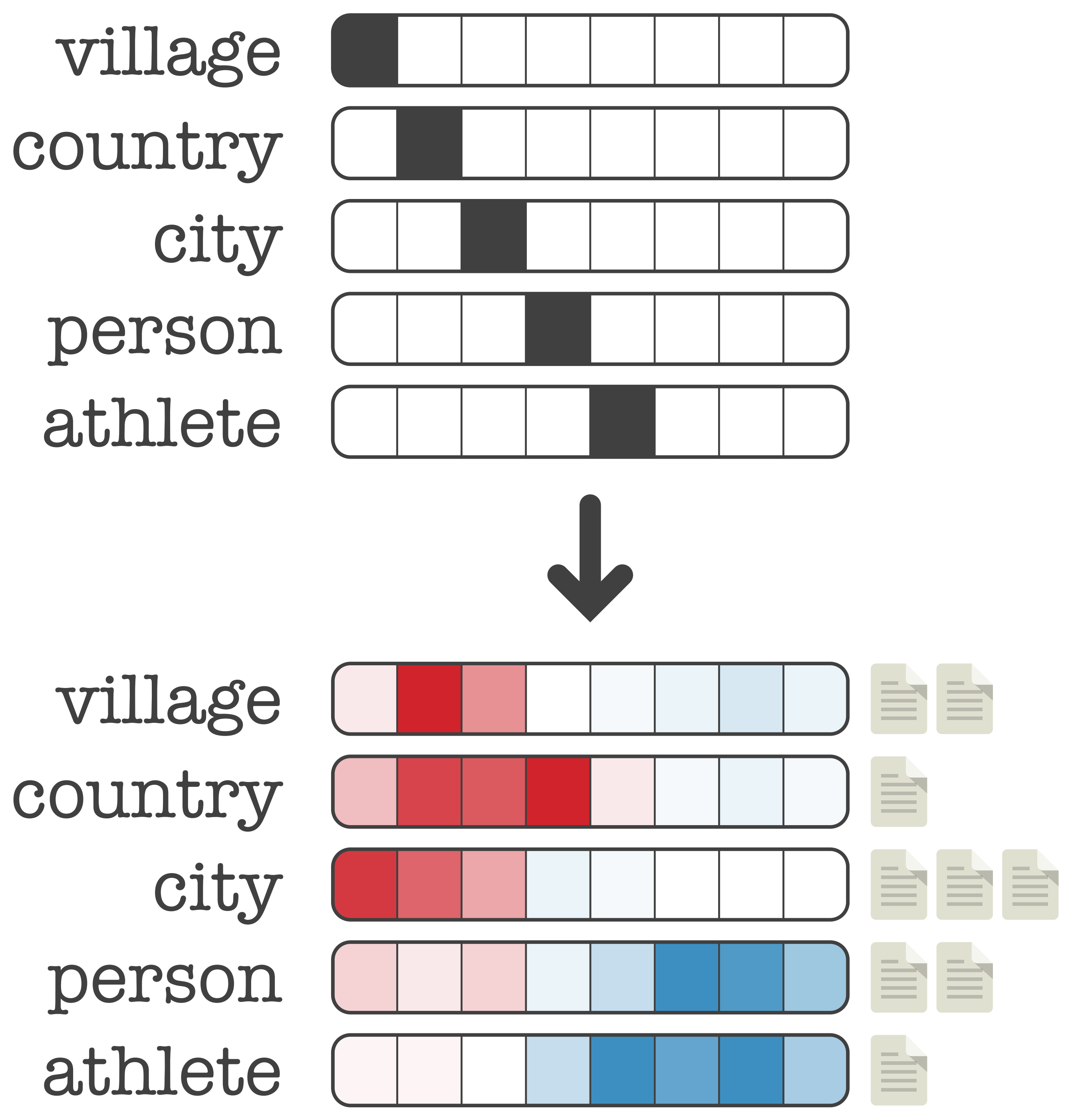
Limitation #1
Dependencies



Limitation #2
Logical Constraints



Limitation #3
Representations

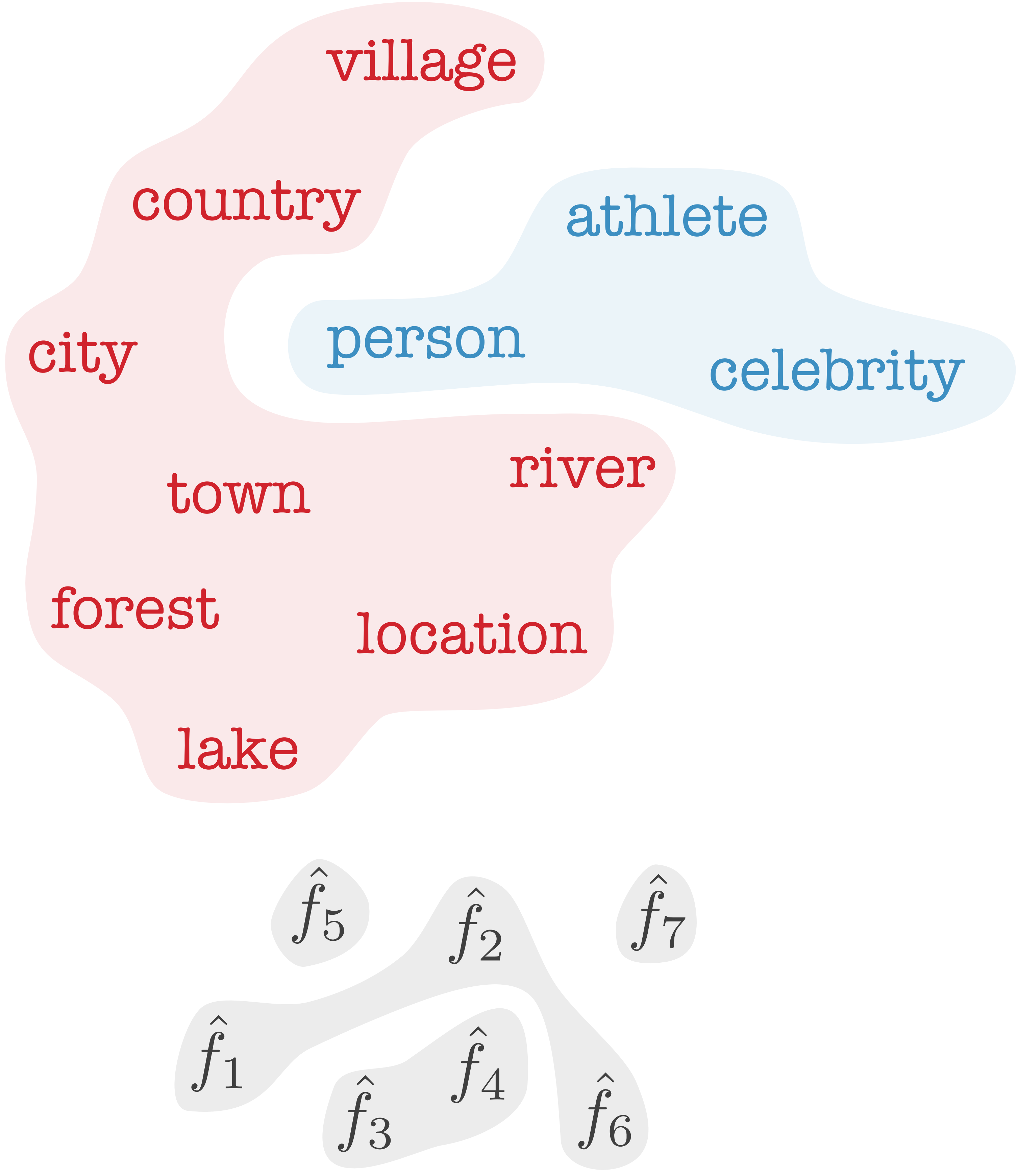


similarly for the predictors

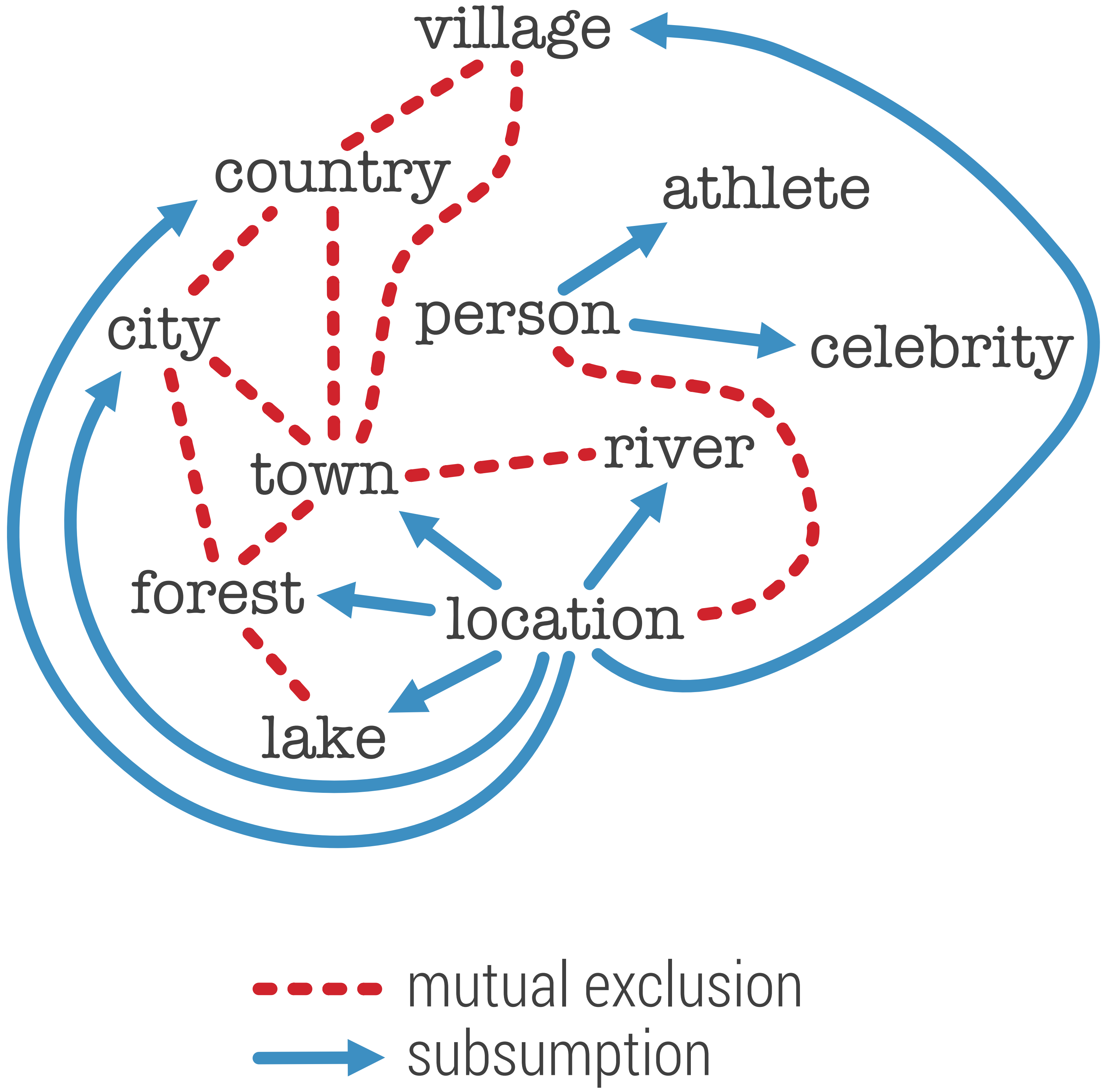
Self-Reflection

What if we have some labeled data?

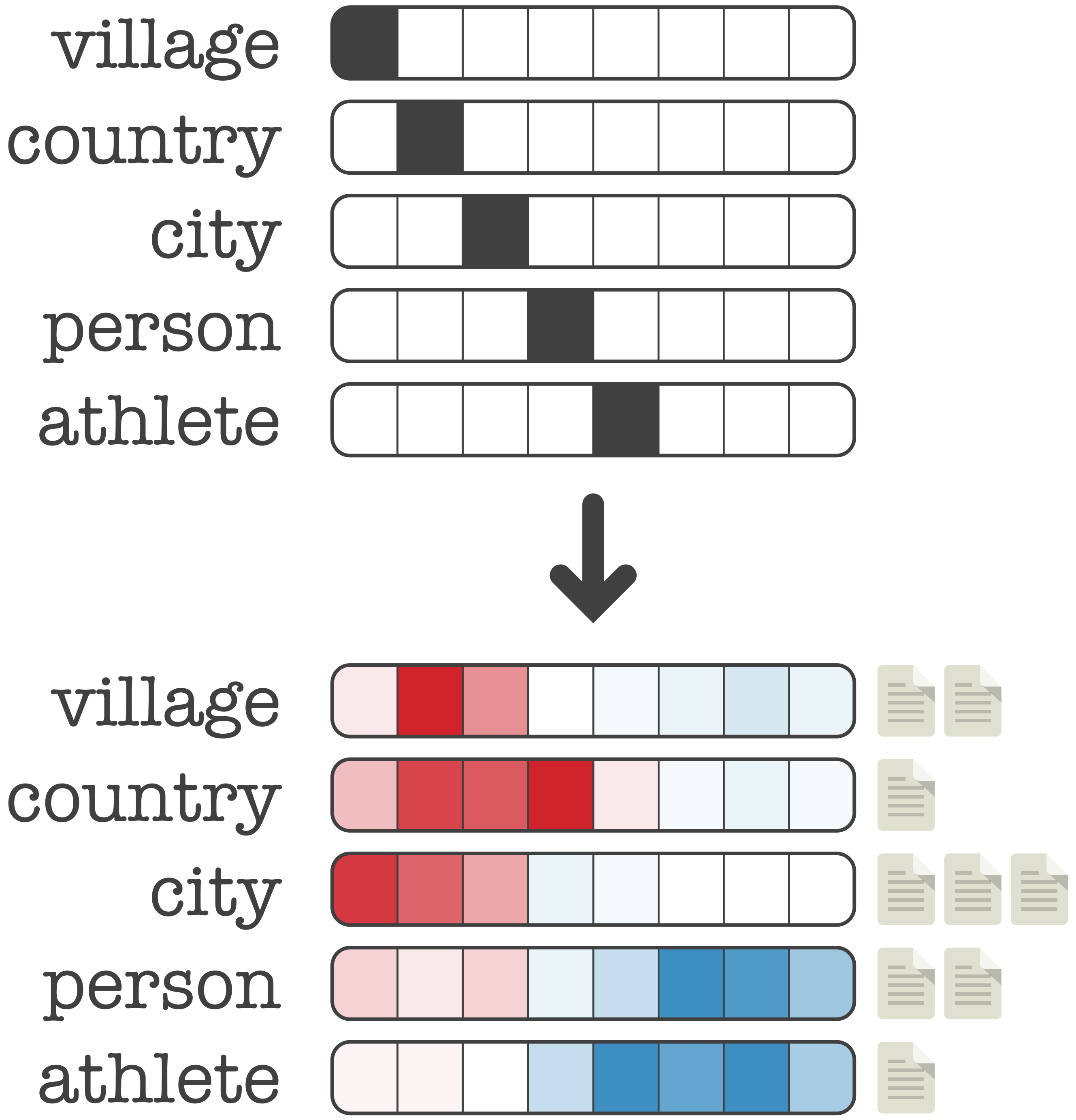
Limitation #1
Dependencies



Limitation #2
Logical Constraints

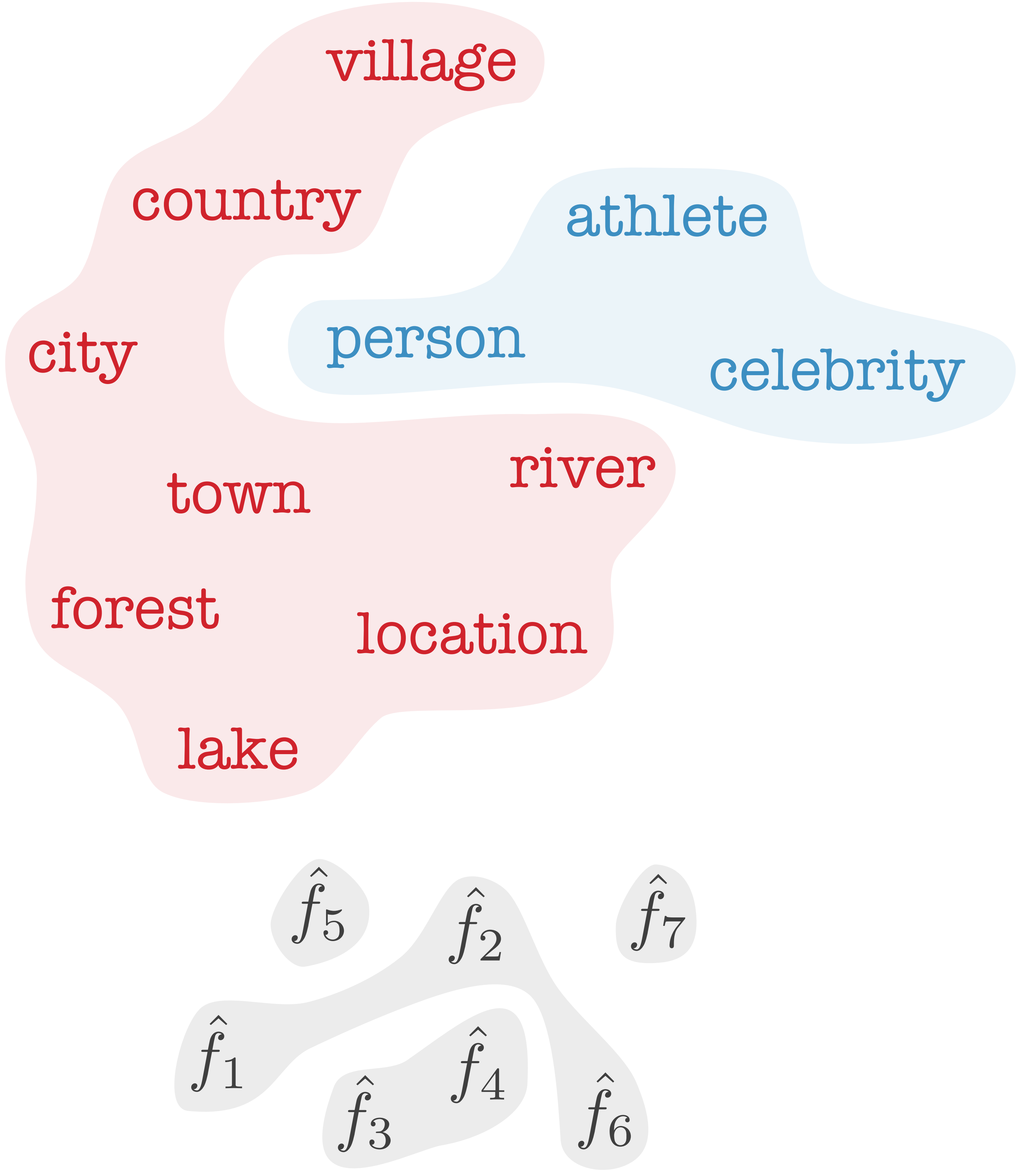


Limitation #3
Representations



similarly for the predictors

Limitation #1
Dependencies



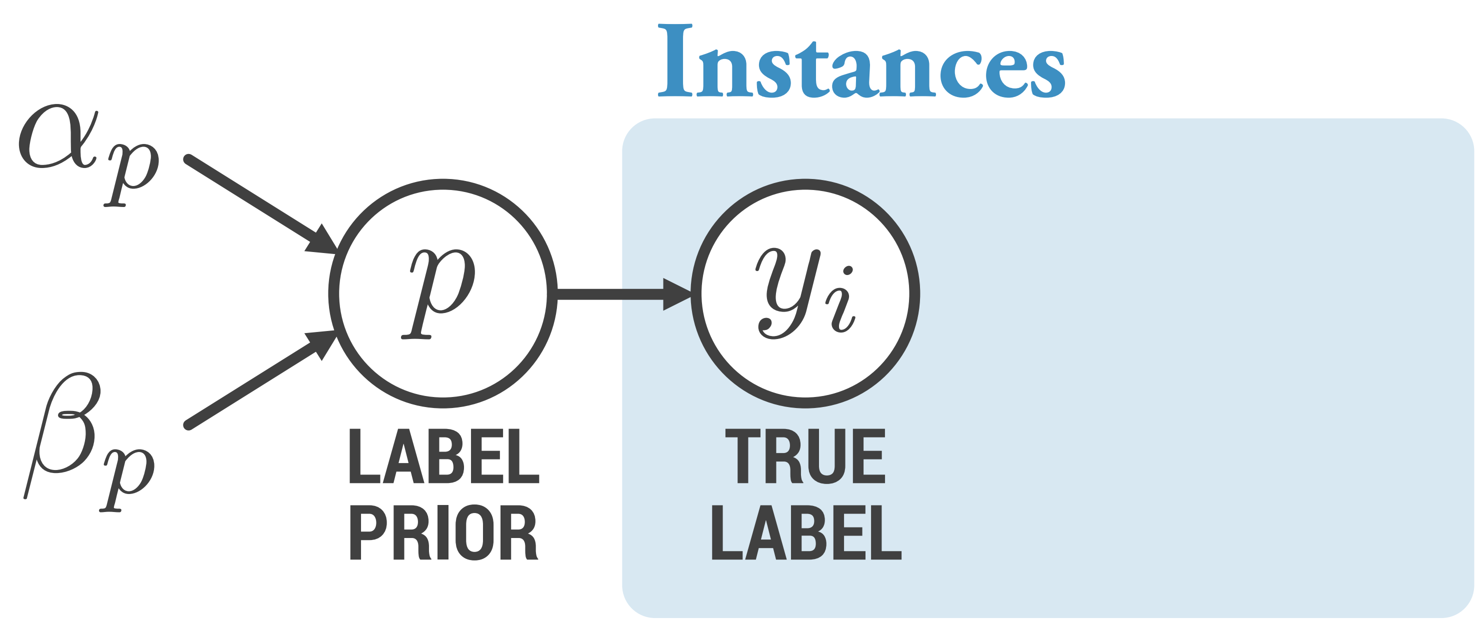
Limitation #2
Logical Constraints

Limitation #3
Representations

similarly for the predictors

A Bayesian Approach

We can start by describing how the predictions were generated:



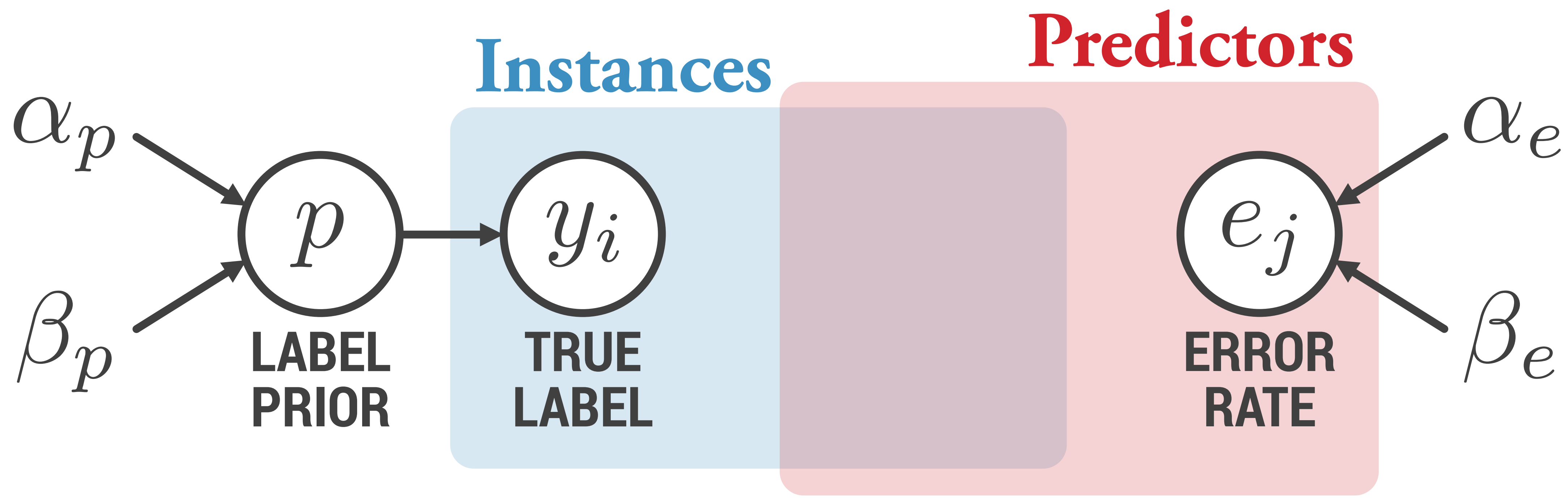
$$i = 1, \ldots, S$$

$$p \sim \text{Beta}(\alpha_p, \beta_p)$$
$$y_i \sim \text{Bernoulli}(p)$$

[**Platanios**, Dubey, Mitchell, ICML 2016]

A Bayesian Approach

We can start by describing how the predictions were generated:



$i = 1, \ldots, S$

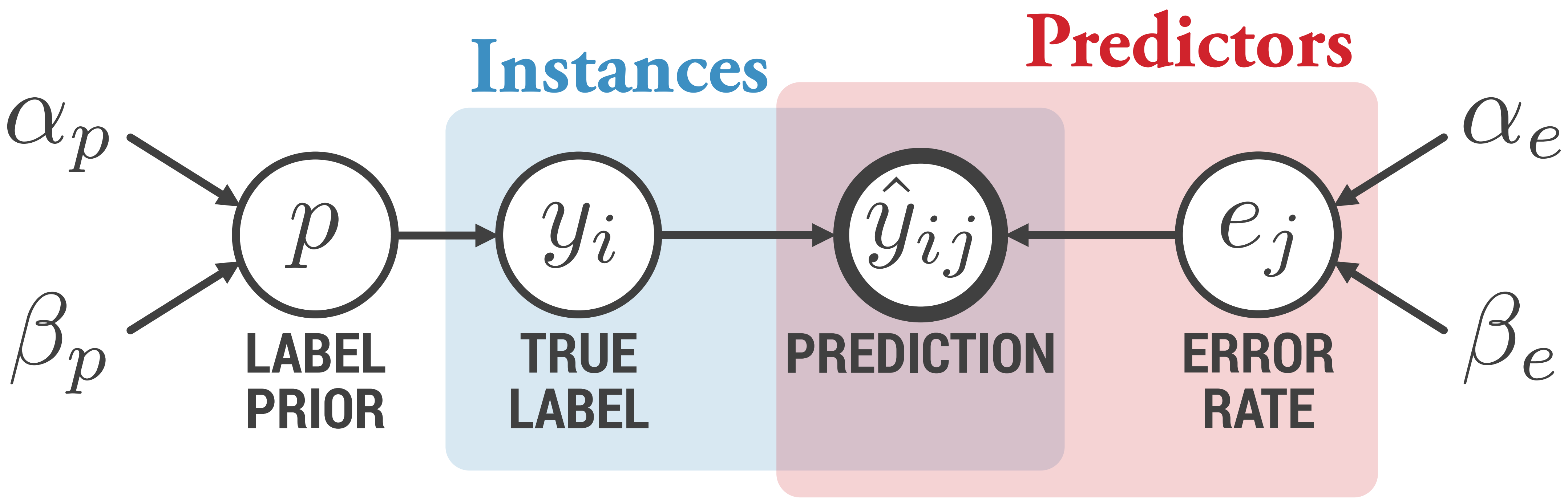
$j = 1, \ldots, N$

$$
\begin{aligned}
p &\sim \mathrm{Beta}(\alpha_p, \beta_p) \\
y_i &\sim \mathrm{Bernoulli}(p) \\
e_j &\sim \mathrm{Beta}(\alpha_e, \beta_e)
\end{aligned}
$$

[**Platanios**, Dubey, Mitchell, ICML 2016]

A Bayesian Approach

We can start by describing how the predictions were generated:



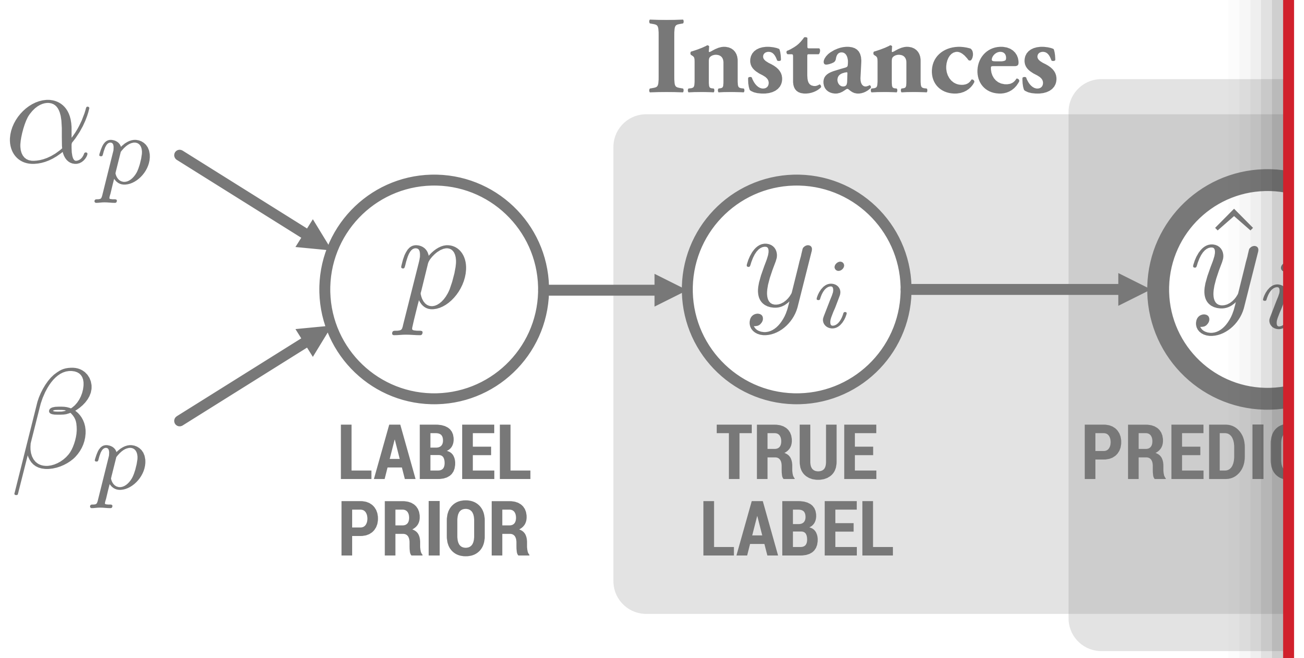
$$i = 1, \ldots, S$$
$$j = 1, \ldots, N$$

$$\begin{aligned} p &\sim \mathrm{Beta}(\alpha_p, \beta_p) \\ y_i &\sim \mathrm{Bernoulli}(p) \\ e_j &\sim \mathrm{Beta}(\alpha_e, \beta_e) \end{aligned}$$

$$\hat{y}_{ij} = \begin{cases} y_i & \text{with probability } 1 - e_j, \\ 1 - y_i & \text{otherwise.} \end{cases}$$

[**Platanios**, Dubey, Mitchell, ICML 2016]

A Bayesian Approach



$$
\begin{aligned}
p &\sim \mathrm{Beta}(\alpha_p, \beta_p) \\
y_i &\sim \mathrm{Bernoulli}(p) \\
e_j &\sim \mathrm{Beta}(\alpha_e, \beta_e)
\end{aligned}
$$

$\hat{y}_{ij} =$

inference

We use Gibbs sampling:

$$
\begin{aligned}
P(p \mid \cdot) &= \mathrm{Beta}(\alpha_p + \sigma_{\boldsymbol{\ell}}, \beta_p + S - \sigma_{\boldsymbol{\ell}}), \\
P(y_i \mid \cdot) &\propto p^{y_i}(1-p)^{1-y_i}\pi_i, \\
P(e_j \mid \cdot) &= \mathrm{Beta}(\alpha_e + \sigma_j, \beta_e + S - \sigma_j),
\end{aligned}
$$

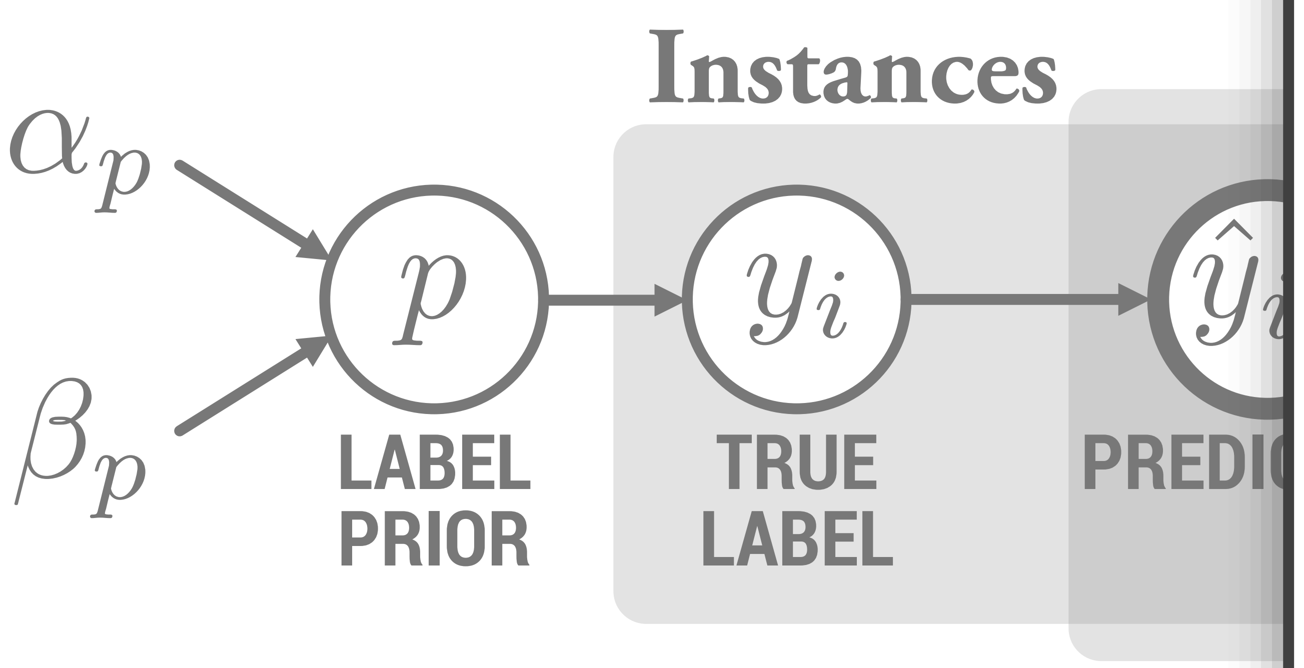
where:

$$
\sigma_{\boldsymbol{y}} = \sum_{i=1}^{S} y_i, \quad \sigma_j = \sum_{i=1}^{S} \mathbb{1}_{\{\hat{y}_{ij} \neq y_i\}},
$$

$$
\pi_i = \prod_{j=1}^{N} e_j^{\mathbb{1}_{\{\hat{y}_{ij} \neq y_i\}}} (1 - e_j)^{\mathbb{1}_{\{\hat{y}_{ij} - y_i\}}}
$$

[**Platanios**, Dubey, Mitchell, ICML 2016]

A Bayesian Approach



$$p \sim \text{Beta}(\alpha_p,$$
$$y_i \sim \text{Bernoulli}(p)$$
$$e_j \sim \text{Beta}(\alpha_e, \beta_e)$$
$$\hat{y}_{ij} -$$

inference

We use Gibbs sampling:

$$P(p \mid \cdot) = \text{Beta}(\alpha_p + \sigma_{\boldsymbol{\ell}}, \beta_p + S - \sigma_{\boldsymbol{\ell}}),$$
$$P(y_i \mid \cdot) \propto p^{y_i}(1-p)^{1-y_i}\pi_i,$$
$$P(e_j \mid \cdot) = \text{Beta}(\alpha_e + \sigma_j, \beta_e + S - \sigma_j),$$

where:

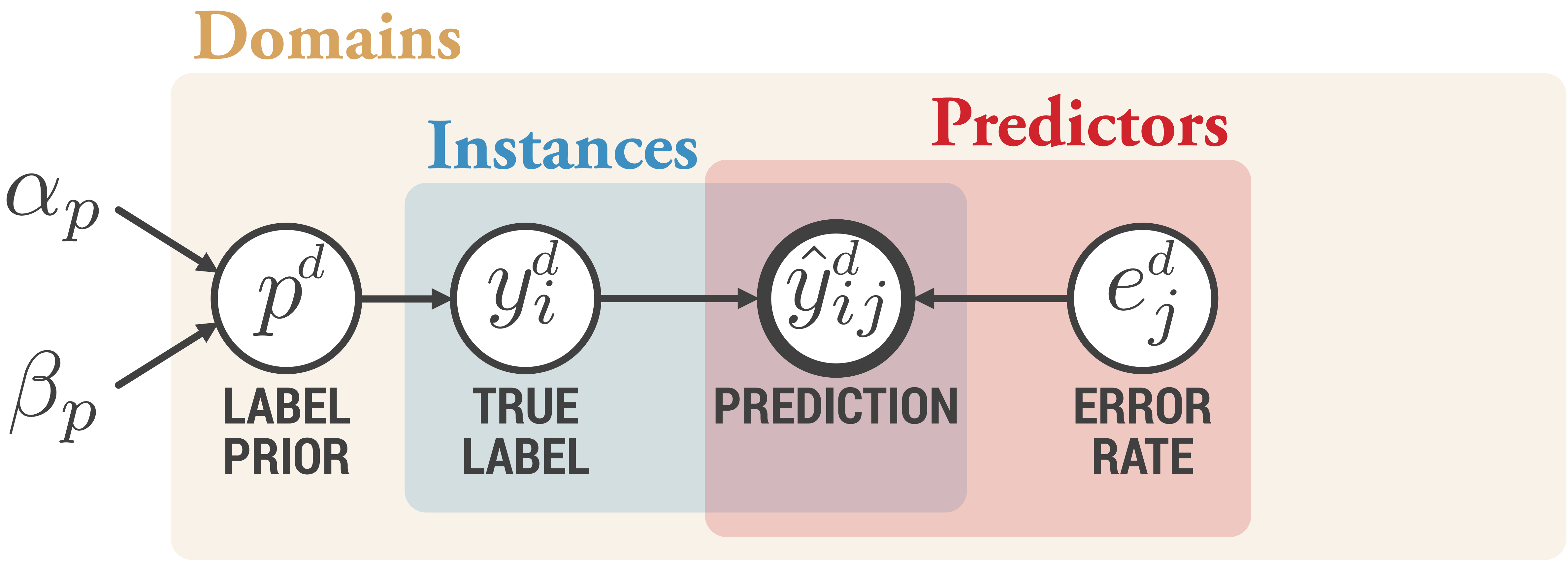
$$\sigma_{\boldsymbol{y}} = \sum_{i=1}^{S} y_i, \quad \sigma_j = \sum_{i=1}^{S} \mathbb{1}_{\{\hat{y}_{ij} \neq y_i\}},$$

$$\pi_i = \prod_{j=1}^{N} e_j^{\mathbb{1}_{\{\hat{y}_{ij} \neq y_i\}}} (1-e_j)^{\mathbb{1}_{\{\hat{y}_{ij} = y_i\}}}$$

implicit use of agreement rates

[**Platanios**, Dubey, Mitchell, ICML 2016]

A Bayesian Approach



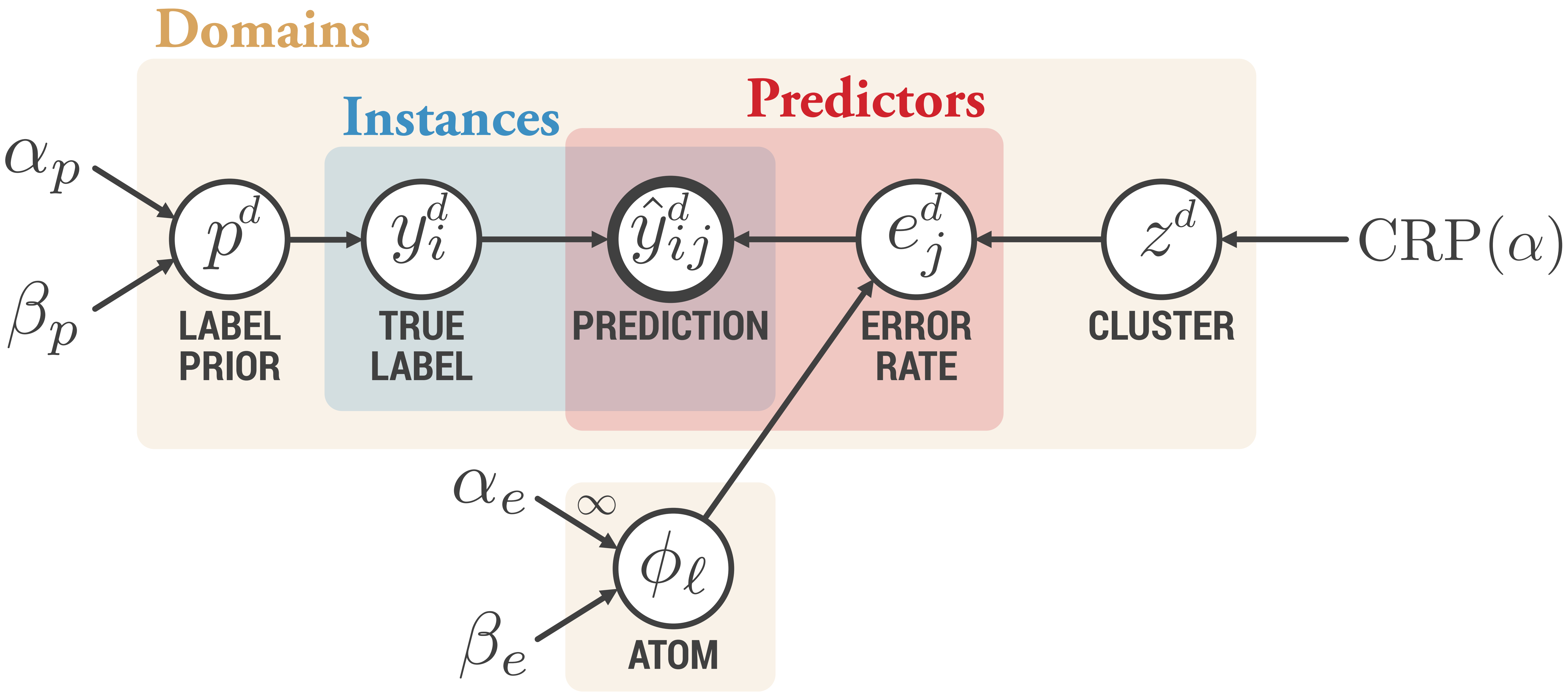
$$i = 1, \ldots, S^d$$
$$j = 1, \ldots, N$$
$$d = 1, \ldots, D$$

$$p^d \sim \text{Beta}(\alpha_p, \beta_p)$$
$$y_i^d \sim \text{Bernoulli}(p^d)$$
$$e_j^d \sim \text{Beta}(\alpha_e, \beta_e)$$

$$\hat{y}_{ij}^d = \begin{cases} y_i^d & \text{with probability } 1 - e_j^d, \\ 1 - y_i^d & \text{otherwise.} \end{cases}$$

[**Platanios**, Dubey, Mitchell, ICML 2016]

A Bayesian Approach



$$i = 1, \ldots, S^d$$
$$j = 1, \ldots, N$$
$$d = 1, \ldots, D$$
$$l = 1, \ldots, \infty$$

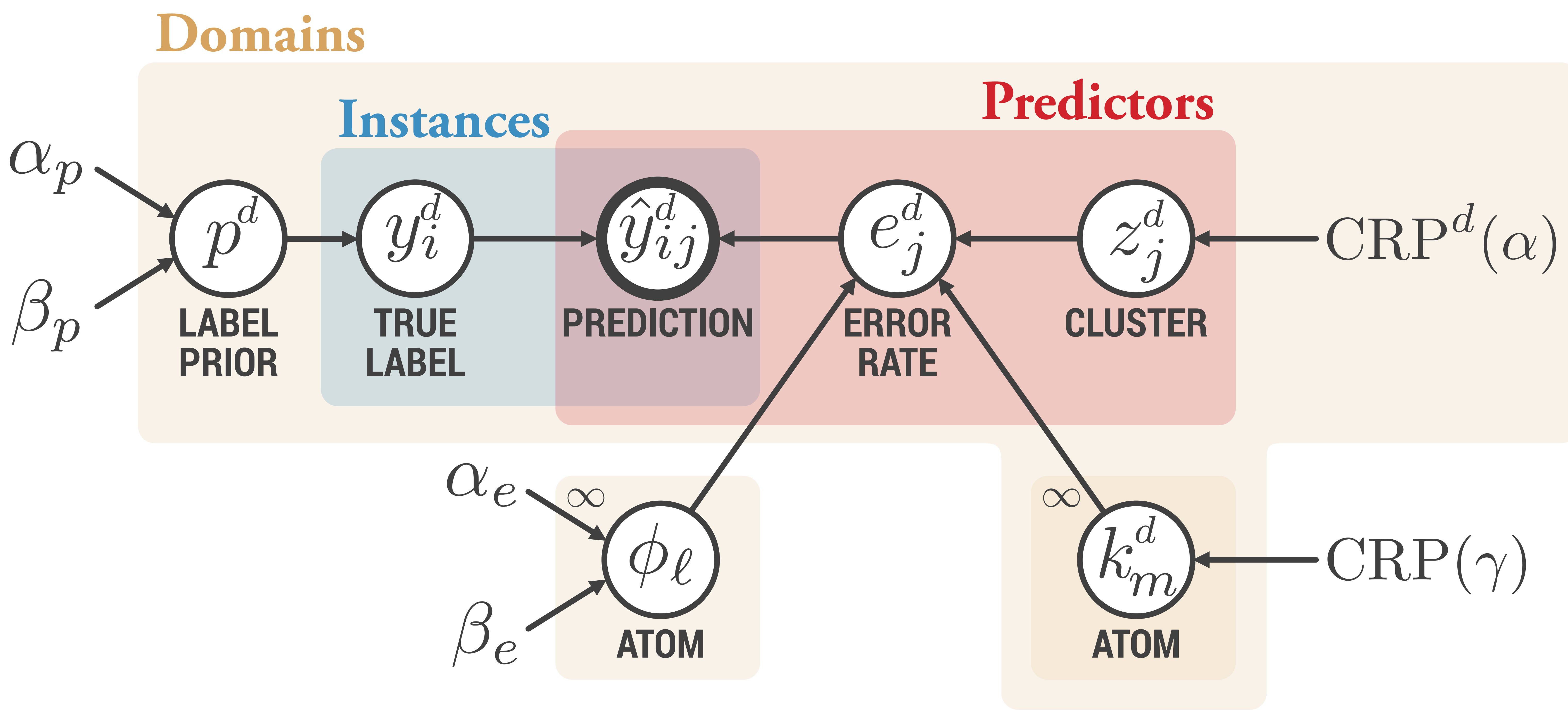
$$p^d \sim \text{Beta}(\alpha_p, \beta_p)$$
$$y_i^d \sim \text{Bernoulli}(p^d)$$
$$e_j^d = [\phi_{z^d}]_j$$

$$\hat{y}_{ij}^d = \begin{cases} y_i^d & \text{with probability } 1 - e_j^d, \\ 1 - y_i^d & \text{otherwise.} \end{cases}$$

$$\phi_l \sim \text{Beta}(\alpha_e, \beta_e)$$
$$z^d \sim \text{CRP}(\alpha)$$

[**Platanios**, Dubey, Mitchell, ICML 2016]

A Bayesian Approach



$$i = 1, \ldots, S^d$$
$$j = 1, \ldots, N$$
$$d = 1, \ldots, D$$
$$l = 1, \ldots, \infty$$
$$m = 1, \ldots, \infty$$

$$p^d \sim \text{Beta}(\alpha_p, \beta_p)$$
$$y_i^d \sim \text{Bernoulli}(p^d)$$
$$e_j^d = \phi_{k^d_{z_j^d}}$$

$$\hat{y}_{ij}^d = \begin{cases} y_i^d & \text{with probability } 1 - e_j^d, \\ 1 - y_i^d & \text{otherwise.} \end{cases}$$

$$\phi_l \sim \text{Beta}(\alpha_e, \beta_e)$$
$$z_j^d \sim \text{CRP}^d(\alpha)$$
$$k_m^d \sim \text{CRP}(\gamma)$$

[**Platanios**, Dubey, Mitchell, ICML 2016]

A Bayesian Approach

NOTE **BRAIN** is harder because the classifiers and the regions are highly dependent!

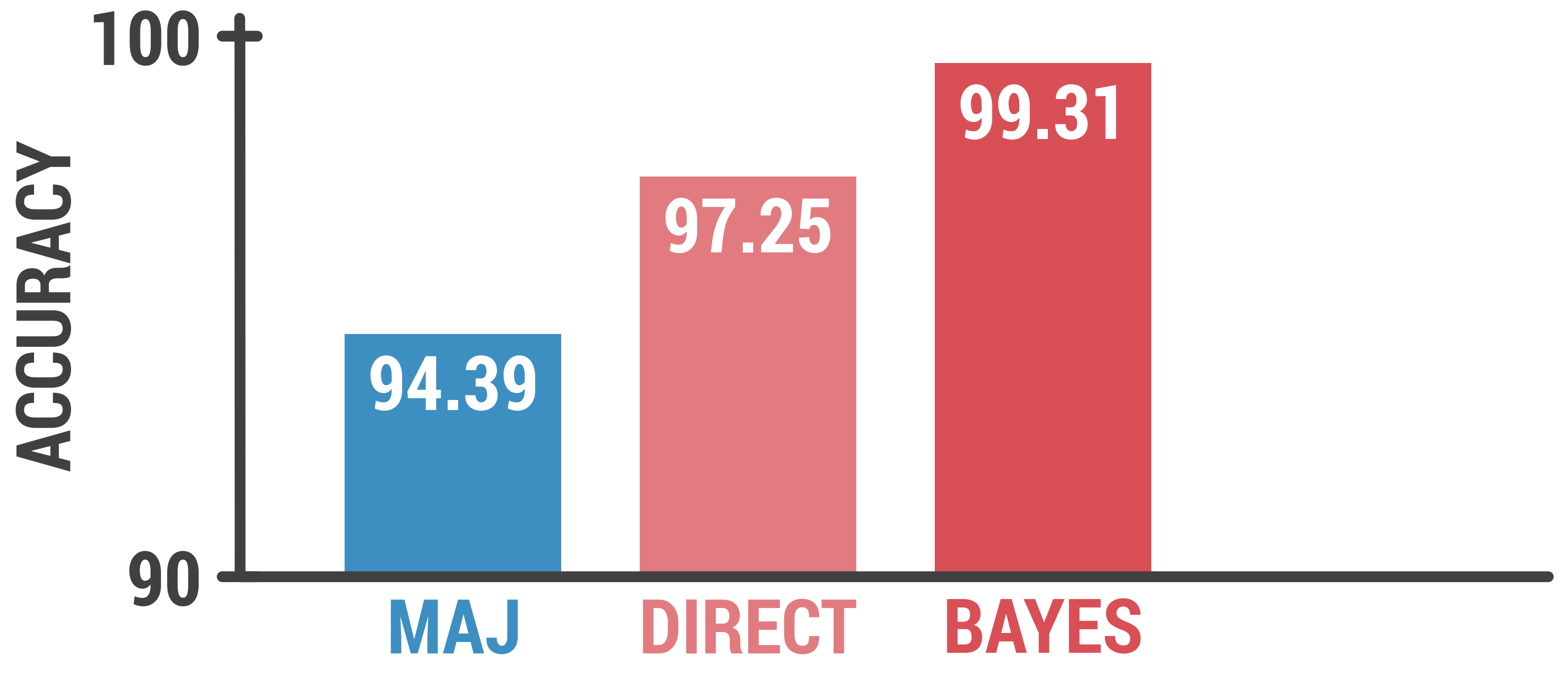
NELL

Task: Predict whether a noun phrase belongs to a category (e.g., `city`).

4 classifiers

15 categories

~300,000 noun phrases



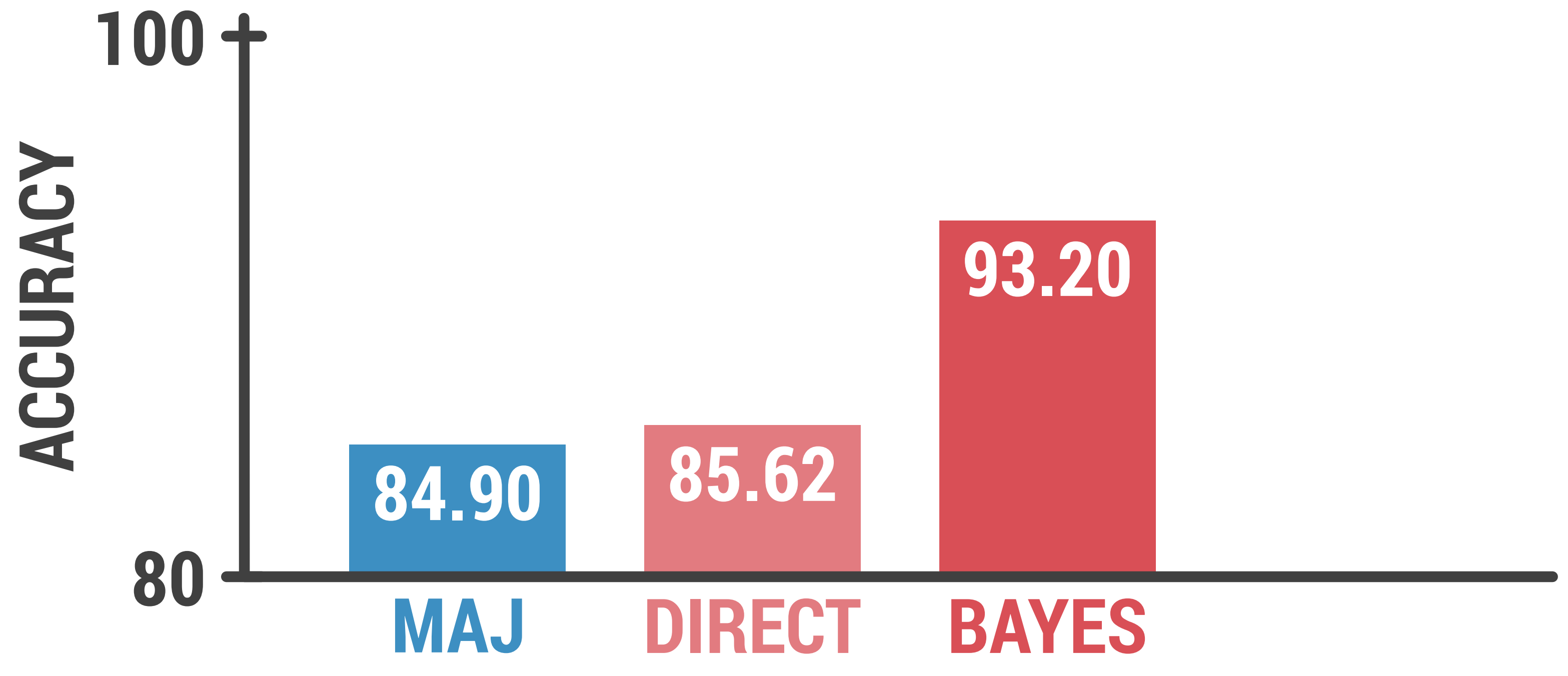
BRAIN

Task: Find which of two 40 second long story passages corresponds to a time series of fMRI neural activity.

11 classifiers

11 brain regions

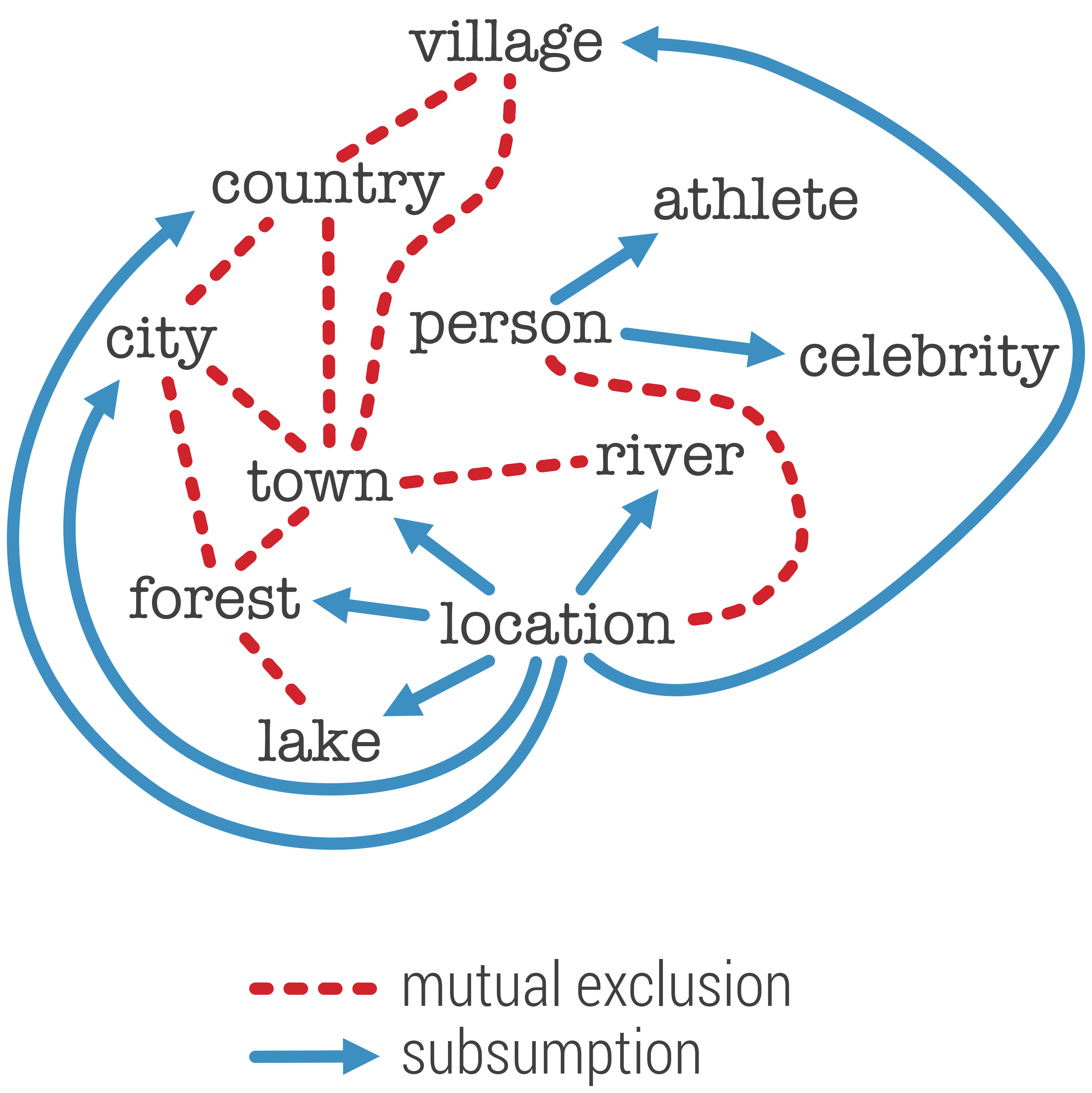
1,000 passages



[**Platanios**, Dubey, Mitchell, ICML 2016]

Limitation #1
Dependencies

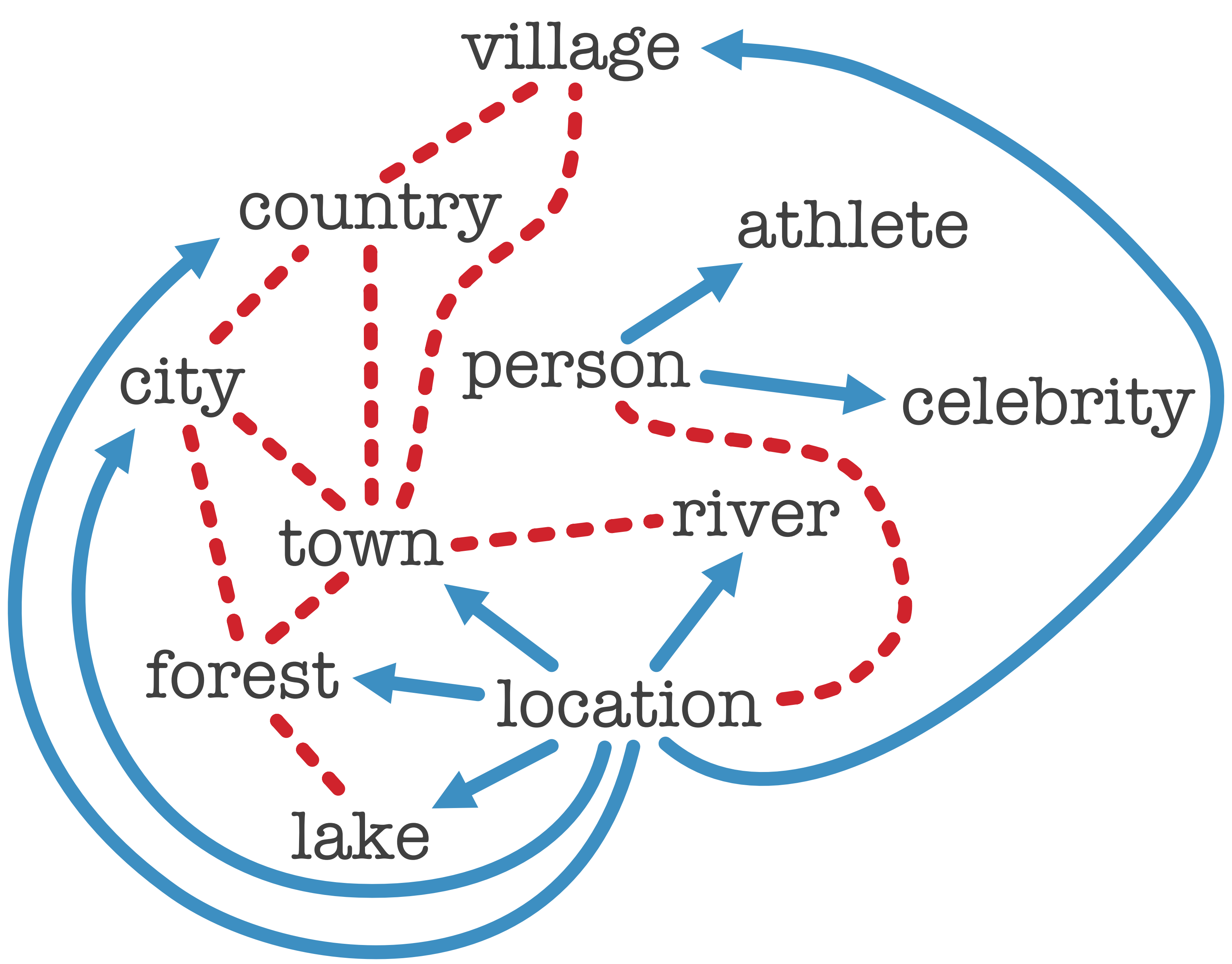
Limitation #2
Logical Constraints



Limitation #3
Representations

similarly for the predictors

A Logic-based Approach



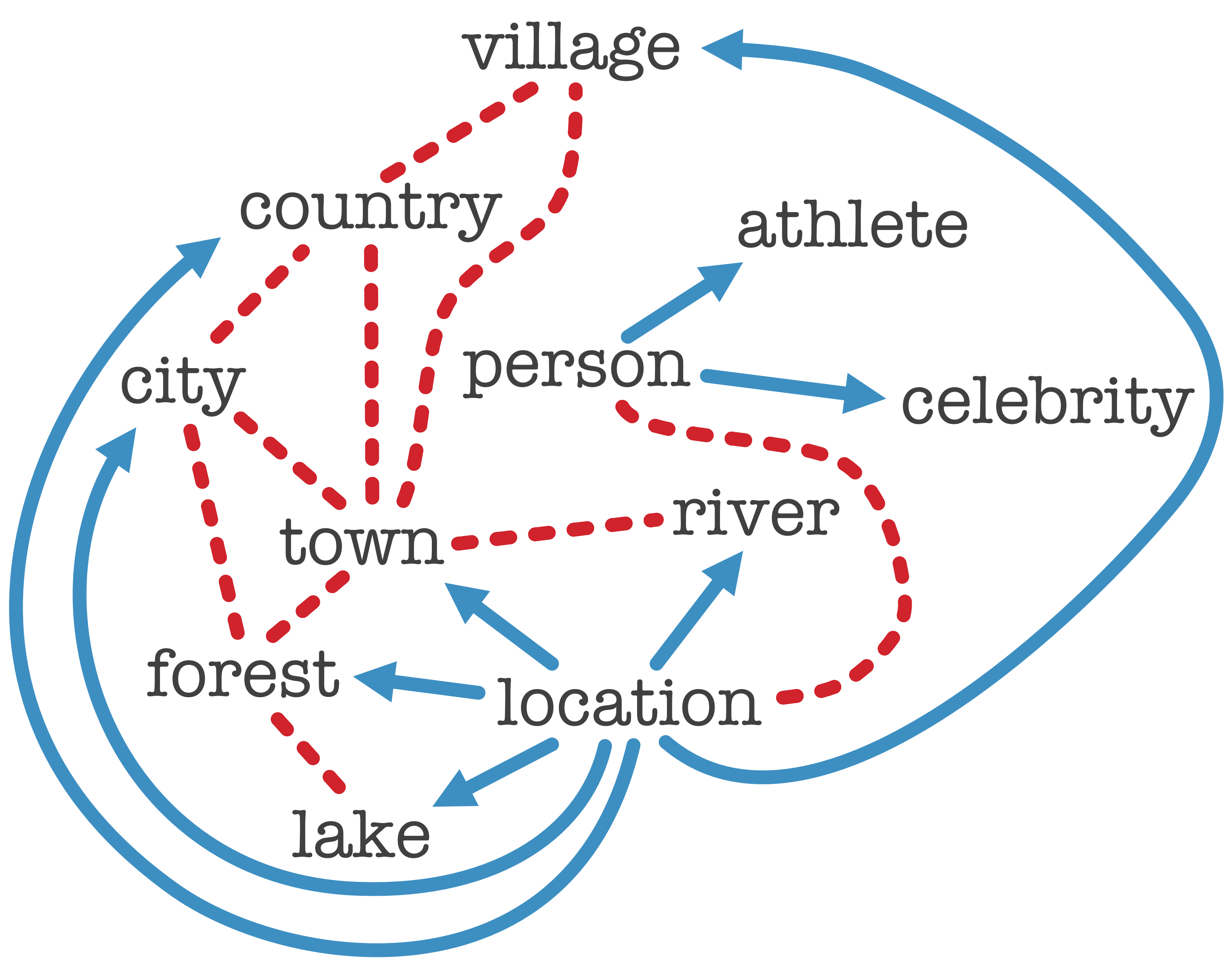
if something is an athlete it cannot also be a country
if something is an athlete it must also be a person

mutual exclusion
subsumption

[**Platanios**, Poon, Horvitz, Mitchell, NeurIPS 2017]

A Logic-based Approach

Let us try to define some logical rules:



if something is an athlete it cannot also be a country
if something is an athlete it must also be a person

mutual exclusion
subsumption

[**Platanios**, Poon, Horvitz, Mitchell, NeurIPS 2017]

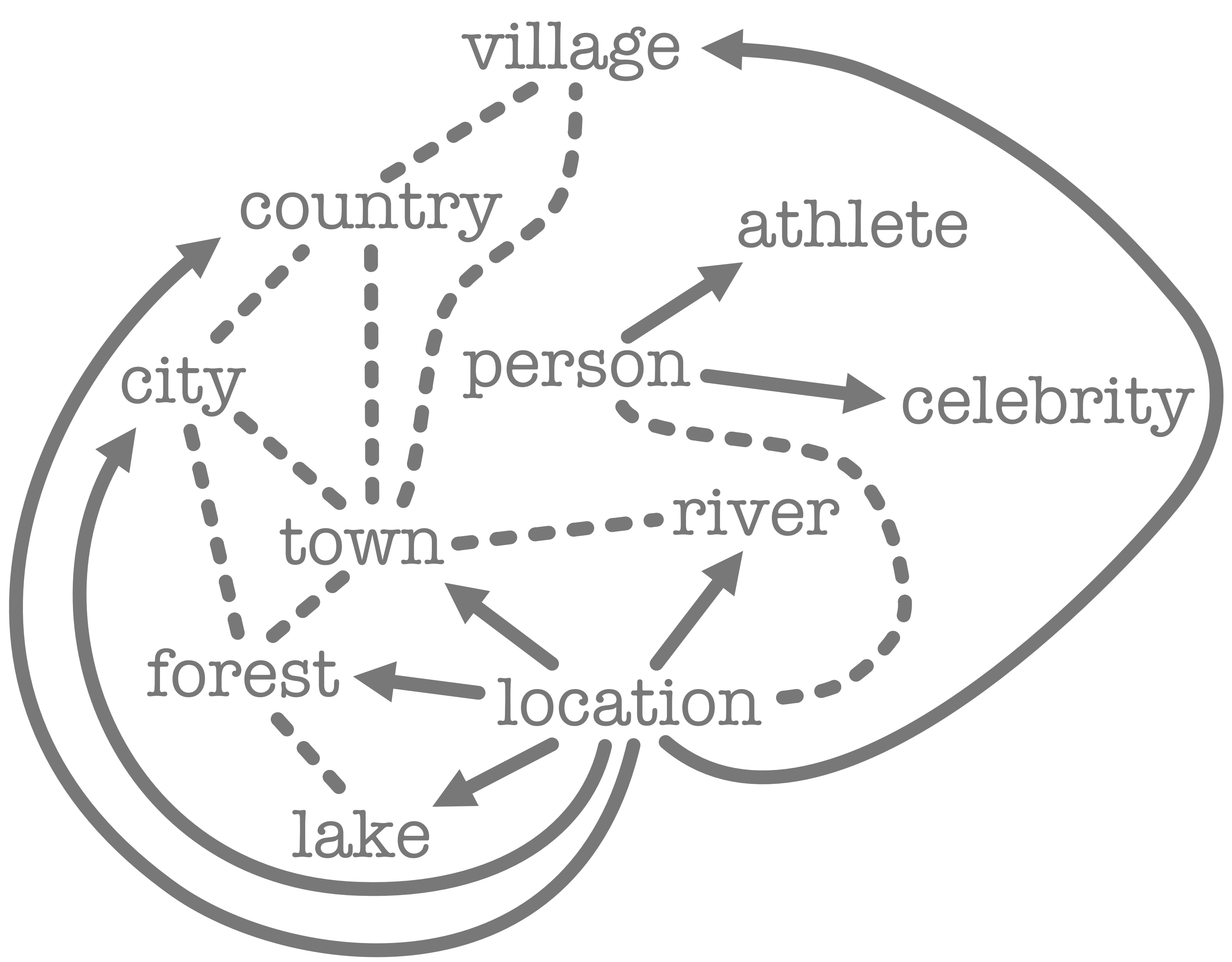
A Logic-based Approach

Let us try to define some logical rules:

mutual exclusion

$$\mathrm{ME}(d_1, d_2) \wedge \hat{y}_{ij}^{d_1} \wedge y_i^{d_2} \rightarrow e_j^{d_1}$$

$$\mathrm{ME}(\texttt{athlete}, \texttt{country}) \wedge \hat{y}_{USA,\ \text{Classifier \#1}}^{\texttt{athlete}} \wedge y_{USA}^{\texttt{country}} \rightarrow e_{\text{Classifier \#1}}^{\texttt{athlete}}$$



if something is an athlete it cannot also be a country — mutual exclusion
if something is an athlete it must also be a person — subsumption

[**Platanios**, Poon, Horvitz, Mitchell, NeurIPS 2017]

A Logic-based Approach

Let us try to define some logical rules:

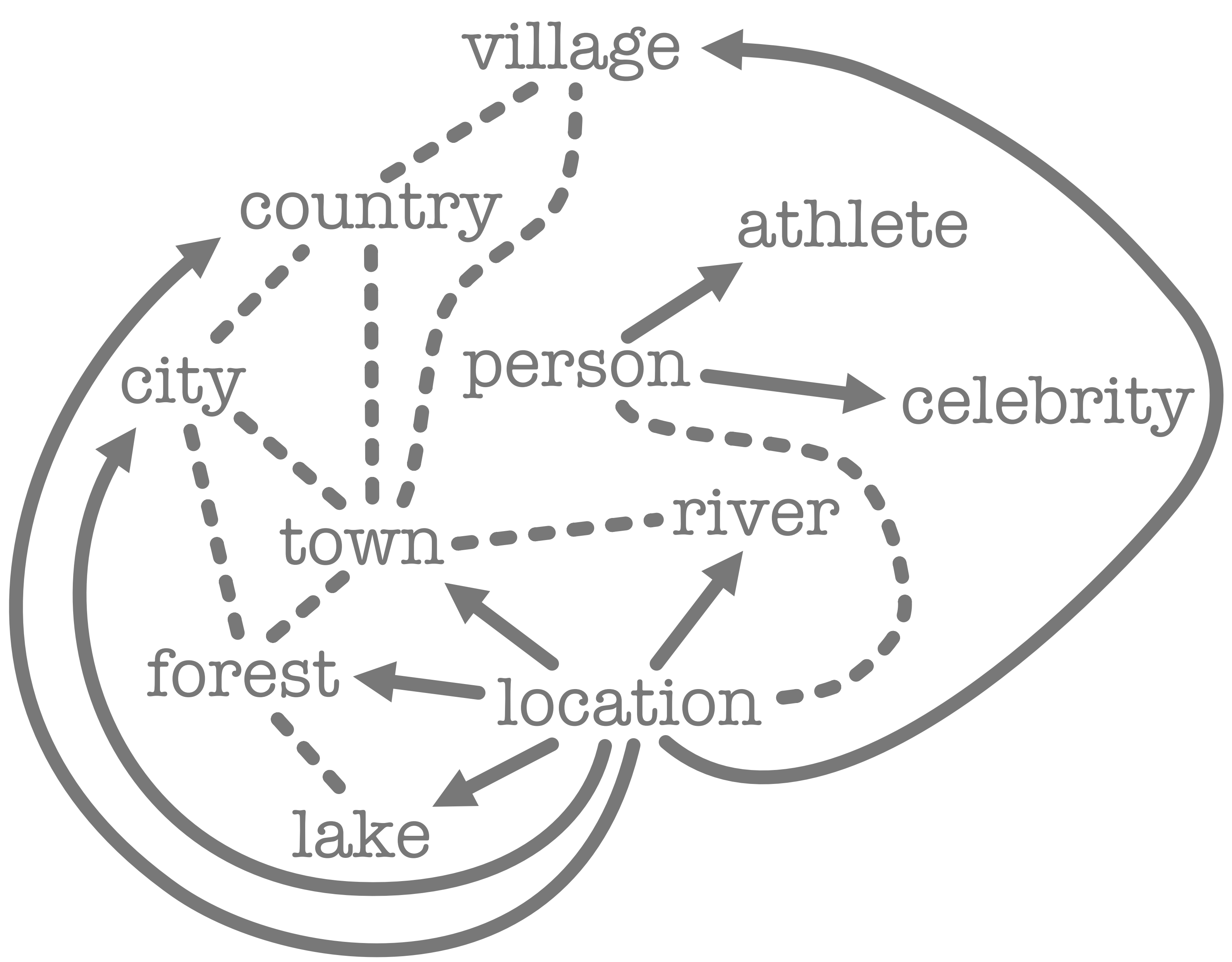
mutual exclusion

$$\mathrm{ME}(d_1, d_2) \wedge \hat{y}_{ij}^{d_1} \wedge y_i^{d_2} \rightarrow e_j^{d_1}$$

subsumption

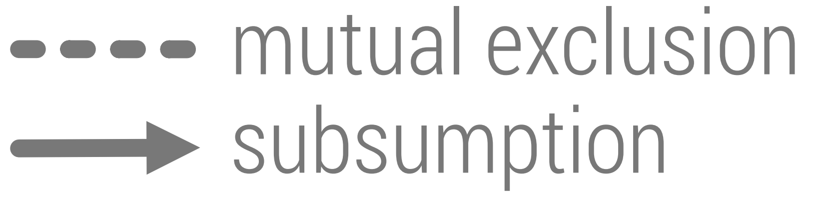
$$\mathrm{SUB}(d_1, d_2) \wedge \neg\hat{y}_{ij}^{d_1} \wedge y_i^{d_2} \rightarrow e_j^{d_1}$$

$$\mathrm{SUB}(\text{person}, \text{athlete}) \wedge \neg\hat{y}_{\textit{Bolt}, \text{ Classifier \#1}}^{\text{person}} \wedge y_{\textit{Bolt}}^{\text{athlete}} \rightarrow e_{\text{Classifier \#1}}^{\text{person}}$$



if something is an athlete it cannot also be a country

if something is an athlete it must also be a person



[**Platanios**, Poon, Horvitz, Mitchell, NeurIPS 2017]

A Logic-based Approach

Latent Variables
Observed Variables

Let us try to define some logical rules:

mutual exclusion

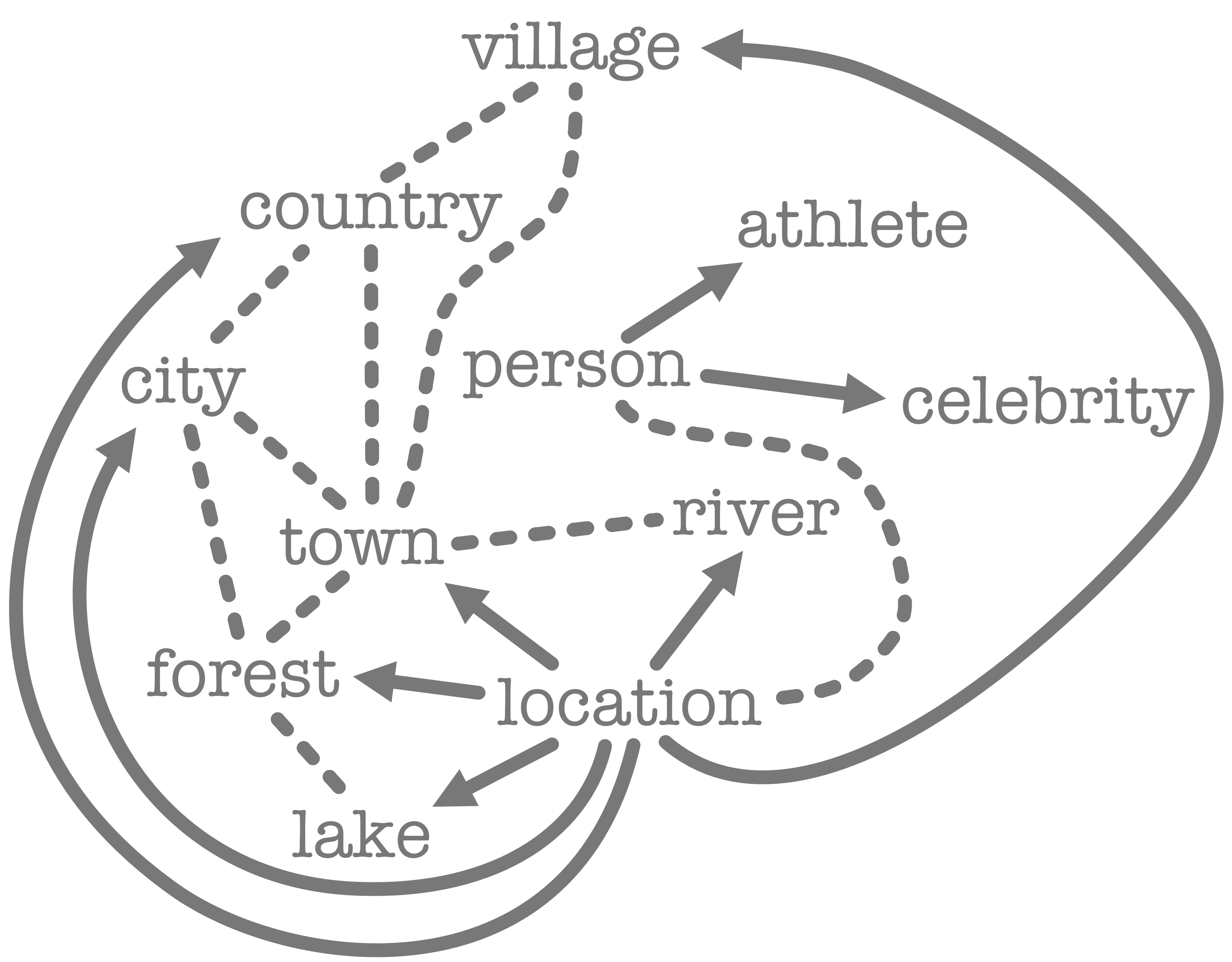
$$\mathrm{ME}(d_1, d_2) \wedge \hat{y}_{ij}^{d_1} \wedge y_i^{d_2} \rightarrow e_j^{d_1}$$

subsumption

$$\mathrm{SUB}(d_1, d_2) \wedge \neg\hat{y}_{ij}^{d_1} \wedge y_i^{d_2} \rightarrow e_j^{d_1}$$

ensemble

$$\hat{y}_{ij}^{d} \wedge \neg e_j^{d} \rightarrow y_i^{d}$$
$$\neg\hat{y}_{ij}^{d} \wedge \neg e_j^{d} \rightarrow \neg y_i^{d}$$
$$\hat{y}_{ij}^{d} \wedge \neg e_j^{d} \rightarrow \neg y_i^{d}$$
$$\neg\hat{y}_{ij}^{d} \wedge e_j^{d} \rightarrow y_i^{d}$$



if something is an athlete it cannot also be a country — mutual exclusion
if something is an athlete it must also be a person — subsumption

[**Platanios**, Poon, Horvitz, Mitchell, NeurIPS 2017]

A Logic-based Approach

Latent Variables
Observed Variables

Let us try to define some logical rules:

mutual exclusion

$$\mathrm{ME}(d_1, d_2) \wedge \hat{y}_{ij}^{d_1} \wedge y_i^{d_2} \rightarrow e_j^{d_1}$$

subsumption

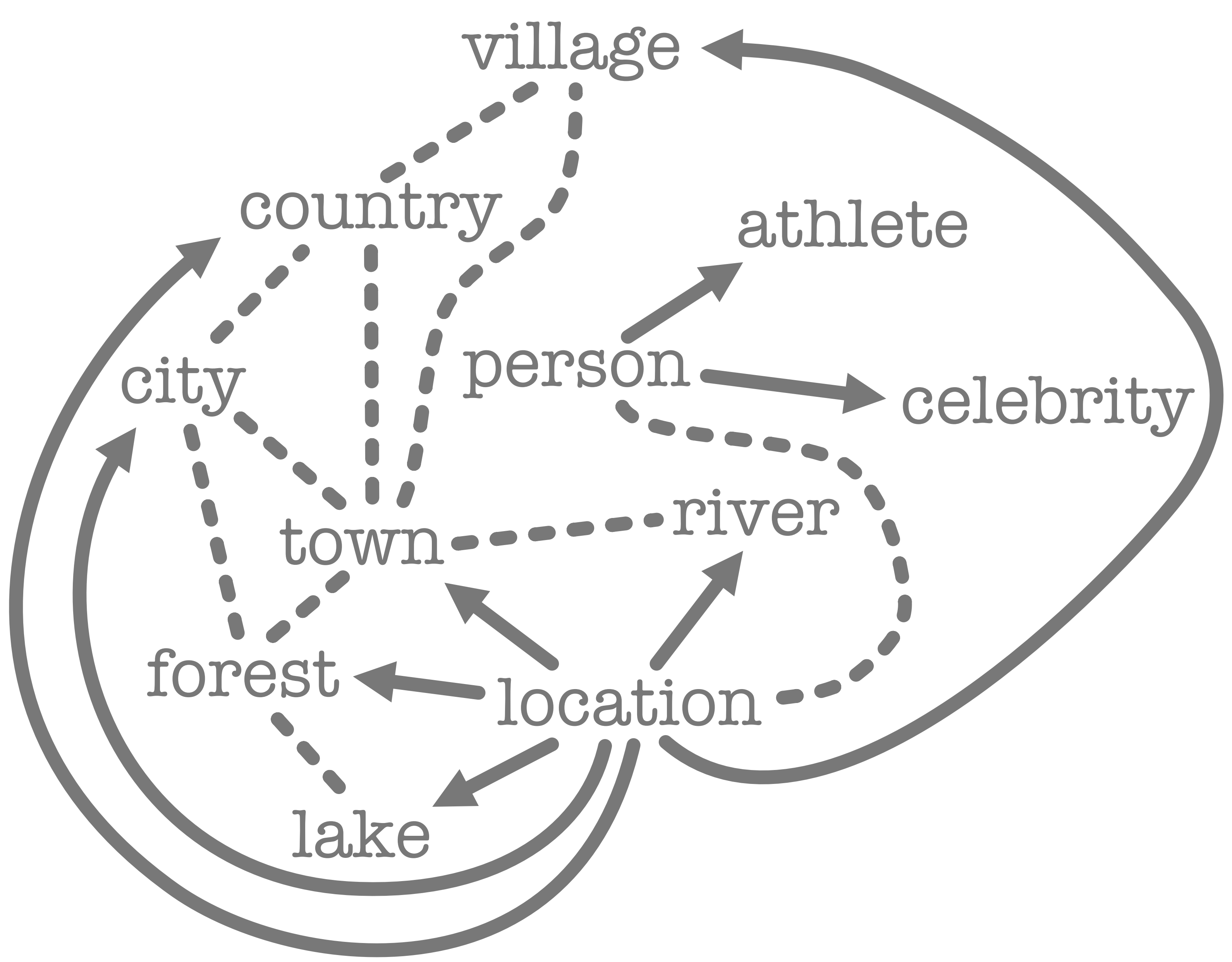
$$\mathrm{SUB}(d_1, d_2) \wedge \neg\hat{y}_{ij}^{d_1} \wedge y_i^{d_2} \rightarrow e_j^{d_1}$$

ensemble

$$\hat{y}_{ij}^{d} \wedge \neg e_j^{d} \rightarrow y_i^{d}$$
$$\neg\hat{y}_{ij}^{d} \wedge \neg e_j^{d} \rightarrow \neg y_i^{d}$$
$$\hat{y}_{ij}^{d} \wedge \neg e_j^{d} \rightarrow \neg y_i^{d}$$
$$\neg\hat{y}_{ij}^{d} \wedge e_j^{d} \rightarrow y_i^{d}$$

identifiability

$$\hat{y}_{ij}^{d} \rightarrow y_i^{d}$$
$$\neg\hat{y}_{ij}^{d} \rightarrow \neg y_i^{d}$$



if something is an athlete it cannot also be a country — mutual exclusion
if something is an athlete it must also be a person — subsumption

[**Platanios**, Poon, Horvitz, Mitchell, NeurIPS 2017]

A Logic-based Approach

Let us try to define some logical rules:

mutual exclusion

$$\mathrm{ME}(d_1, d_2) \wedge \hat{y}_{ij}^{d_1} \wedge y_i^{d_2} \rightarrow e_j^{d_1}$$

subsumption

$$\mathrm{SUB}(d_1, d_2) \wedge \neg\hat{y}_{ij}^{d_1} \wedge y_i^{d_2} \rightarrow e_j^{d_1}$$

ensemble

$$\hat{y}_{ij}^{d} \wedge \neg e_j^{d} \rightarrow y_i^{d}$$
$$\neg\hat{y}_{ij}^{d} \wedge \neg e_j^{d} \rightarrow \neg y_i^{d}$$
$$\hat{y}_{ij}^{d} \wedge \neg e_j^{d} \rightarrow \neg y_i^{d}$$
$$\neg\hat{y}_{ij}^{d} \wedge e_j^{d} \rightarrow y_i^{d}$$

identifiab

$$\hat{y}_{ij}^{d} \rightarrow y$$
$$\neg\hat{y}_{ij}^{d} \rightarrow \neg$$

learning

Any **probabilistic logic framework** can be used, in theory (e.g., *Markov Logic Networks*).

[**Platanios**, Poon, Horvitz, Mitchell, NeurIPS 2017]

A Logic-based Approach

Latent Variables
Observed Variables

Let us try to define some logical rules:

mutual exclusion

$$\mathrm{ME}(d_1, d_2) \wedge \hat{y}_{ij}^{d_1} \wedge y_i^{d_2} \rightarrow e_j^{d_1}$$

subsumption

$$\mathrm{SUB}(d_1, d_2) \wedge \neg\hat{y}_{ij}^{d_1} \wedge y_i^{d_2} \rightarrow e_j^{d_1}$$

ensemble

$$\hat{y}_{ij}^{d} \wedge \neg e_j^{d} \rightarrow y_i^{d}$$
$$\neg\hat{y}_{ij}^{d} \wedge \neg e_j^{d} \rightarrow \neg y_i^{d}$$
$$\hat{y}_{ij}^{d} \wedge \neg e_j^{d} \rightarrow \neg y_i^{d}$$
$$\neg\hat{y}_{ij}^{d} \wedge e_j^{d} \rightarrow y_i^{d}$$

identifiab

$$\hat{y}_{ij}^{d} \rightarrow y$$
$$\neg\hat{y}_{ij}^{d} \rightarrow \neg$$

learning

~~Any **probabilistic logic framework** can be used, in theory (e.g., *Markov Logic Networks*).~~ **Scalability is a big issue!**

[**Platanios**, Poon, Horvitz, Mitchell, NeurIPS 2017]

A Logic-based Approach

Latent Variables
Observed Variables

Let us try to define some logical rules:

mutual exclusion

$$\mathrm{ME}(d_1, d_2) \wedge \hat{y}_{ij}^{d_1} \wedge y_i^{d_2} \rightarrow e_j^{d_1}$$

subsumption

$$\mathrm{SUB}(d_1, d_2) \wedge \neg\hat{y}_{ij}^{d_1} \wedge y_i^{d_2} \rightarrow e_j^{d_1}$$

ensemble

$$\hat{y}_{ij}^{d} \wedge \neg e_j^{d} \rightarrow y_i^{d}$$
$$\neg\hat{y}_{ij}^{d} \wedge \neg e_j^{d} \rightarrow \neg y_i^{d}$$
$$\hat{y}_{ij}^{d} \wedge \neg e_j^{d} \rightarrow \neg y_i^{d}$$
$$\neg\hat{y}_{ij}^{d} \wedge e_j^{d} \rightarrow y_i^{d}$$

identifiab

$$\hat{y}_{ij}^{d} \rightarrow$$
$$\neg\hat{y}_{ij}^{d} \rightarrow$$

learning

~~Any **probabilistic logic framework** can be used, in theory (e.g., *Markov Logic Networks*).~~ **Scalability is a big issue!**

GROUNDING

$$\mathrm{ME}(\text{athlete}, \text{country}) \wedge \hat{y}_{00}^{\text{athlete}} \wedge y_0^{\text{country}} \rightarrow e_0^{\text{athlete}}$$
$$\mathrm{ME}(\text{athlete}, \text{country}) \wedge \hat{y}_{01}^{\text{athlete}} \wedge y_0^{\text{country}} \rightarrow e_1^{\text{athlete}}$$
$$\mathrm{ME}(\text{athlete}, \text{country}) \wedge \hat{y}_{10}^{\text{athlete}} \wedge y_1^{\text{country}} \rightarrow e_0^{\text{athlete}}$$
$$\cdots$$

[**Platanios**, Poon, Horvitz, Mitchell, NeurIPS 2017]

A Logic-based Approach

Latent Variables
Observed Variables

Let us try to define some logical rules:

mutual exclusion

$$\mathrm{ME}(d_1, d_2) \wedge \hat{y}_{ij}^{d_1} \wedge y_i^{d_2} \rightarrow e_j^{d_1}$$

subsumption

$$\mathrm{SUB}(d_1, d_2) \wedge \neg\hat{y}_{ij}^{d_1} \wedge y_i^{d_2} \rightarrow e_j^{d_1}$$

ensemble

$$\hat{y}_{ij}^{d} \wedge \neg e_j^{d} \rightarrow y_i^{d}$$
$$\neg\hat{y}_{ij}^{d} \wedge \neg e_j^{d} \rightarrow \neg y_i^{d}$$
$$\hat{y}_{ij}^{d} \wedge \neg e_j^{d} \rightarrow \neg y_i^{d}$$
$$\neg\hat{y}_{ij}^{d} \wedge e_j^{d} \rightarrow y_i^{d}$$

identifiab

$$\hat{y}_{ij}^{d} \rightarrow$$
$$\neg\hat{y}_{ij}^{d} \rightarrow$$

learning

~~Any **probabilistic logic framework** can be used, in theory (e.g., *Markov Logic Networks*).~~ **Scalability is a big issue!**

GROUNDING

$$\mathrm{ME}(\text{athlete}, \text{country}) \wedge \hat{y}_{00}^{\text{athlete}} \wedge y_0^{\text{country}} \rightarrow e_0^{\text{athlete}}$$
$$\mathrm{ME}(\text{athlete}, \text{country}) \wedge \hat{y}_{01}^{\text{athlete}} \wedge y_0^{\text{country}} \rightarrow e_1^{\text{athlete}}$$
$$\mathrm{ME}(\text{athlete}, \text{country}) \wedge \hat{y}_{10}^{\text{athlete}} \wedge y_1^{\text{country}} \rightarrow e_0^{\text{athlete}}$$
$$\cdots$$

Too expensive! → **custom algorithm**

[**Platanios**, Poon, Horvitz, Mitchell, NeurIPS 2017]

A Logic-based Approach

Latent Variables
Observed Variables

Let us try to define some logical rules:

mutual exclusion

$$\mathrm{ME}(d_1, d_2) \wedge \hat{y}_{ij}^{d_1} \wedge y_i^{d_2} \rightarrow e_j^{d_1}$$

subsumption

$$\mathrm{SUB}(d_1, d_2) \wedge \neg\hat{y}_{ij}^{d_1} \wedge y_i^{d_2} \rightarrow e_j^{d_1}$$

ensemble

$$\hat{y}_{ij}^{d} \wedge \neg e_j^{d} \rightarrow y_i^{d}$$
$$\neg\hat{y}_{ij}^{d} \wedge \neg e_j^{d} \rightarrow \neg y_i^{d}$$
$$\hat{y}_{ij}^{d} \wedge \neg e_j^{d} \rightarrow \neg y_i^{d}$$
$$\neg\hat{y}_{ij}^{d} \wedge e_j^{d} \rightarrow y_i^{d}$$

identifiab

$$\hat{y}_{ij}^{d} \rightarrow y$$
$$\neg\hat{y}_{ij}^{d} \rightarrow$$

learning

~~Any **probabilistic logic framework** can be used, in theory (e.g., *Markov Logic Networks*).~~ **Scalability is a big issue!**

GROUNDING

$$\mathrm{ME}(\text{athlete}, \text{country}) \wedge \hat{y}_{00}^{\text{athlete}} \wedge y_0^{\text{country}} \rightarrow e_0^{\text{athlete}}$$
$$\mathrm{ME}(\text{athlete}, \text{country}) \wedge \hat{y}_{01}^{\text{athlete}} \wedge y_0^{\text{country}} \rightarrow e_1^{\text{athlete}}$$
$$\mathrm{ME}(\text{athlete}, \text{country}) \wedge \hat{y}_{10}^{\text{athlete}} \wedge y_1^{\text{country}} \rightarrow e_0^{\text{athlete}}$$
$$\ldots$$

Too expensive! → **custom algorithm**

INFERENCE

We use **Probabilistic Soft Logic (PSL)** with a customized **stochastic consensus ADMM** algorithm to parallelize inference.

[**Platanios**, Poon, Horvitz, Mitchell, NeurIPS 2017]

A Logic-based Approach

Latent Variables
Observed Variables

Let us try to define some logical rules:

mutual exclusion

$$\mathrm{ME}(d_1, d_2) \wedge \hat{y}_{ij}^{d_1} \wedge y_i^{d_2} \rightarrow e_j^{d_1}$$

subsumption

$$\mathrm{SUB}(d_1, d_2) \wedge \neg\hat{y}_{ij}^{d_1} \wedge y_i^{d_2} \rightarrow e_j^{d_1}$$

ensemble

$$\hat{y}_{ij}^{d} \wedge \neg e_j^{d} \rightarrow y_i^{d}$$
$$\neg\hat{y}_{ij}^{d} \wedge \neg e_j^{d} \rightarrow \neg y_i^{d}$$
$$\hat{y}_{ij}^{d} \wedge \neg e_j^{d} \rightarrow \neg y_i^{d}$$
$$\neg\hat{y}_{ij}^{d} \wedge e_j^{d} \rightarrow y_i^{d}$$

identifiab

$$\hat{y}_{ij}^{d} \rightarrow$$
$$\neg\hat{y}_{ij}^{d} \rightarrow$$

learning

~~Any **probabilistic logic framework** can be used, in theory (e.g., *Markov Logic Networks*).~~ **Scalability is a big issue!**

GROUNDING

$$\mathrm{ME}(\texttt{athlete}, \texttt{country}) \wedge \hat{y}_{00}^{\texttt{athlete}} \wedge y_0^{\texttt{country}} \rightarrow e_0^{\texttt{athlete}}$$
$$\mathrm{ME}(\texttt{athlete}, \texttt{country}) \wedge \hat{y}_{01}^{\texttt{athlete}} \wedge y_0^{\texttt{country}} \rightarrow e_1^{\texttt{athlete}}$$
$$\mathrm{ME}(\texttt{athlete}, \texttt{country}) \wedge \hat{y}_{10}^{\texttt{athlete}} \wedge y_1^{\texttt{country}} \rightarrow e_0^{\texttt{athlete}}$$
$$\ldots$$

Too expensive! → **custom algorithm**

INFERENCE

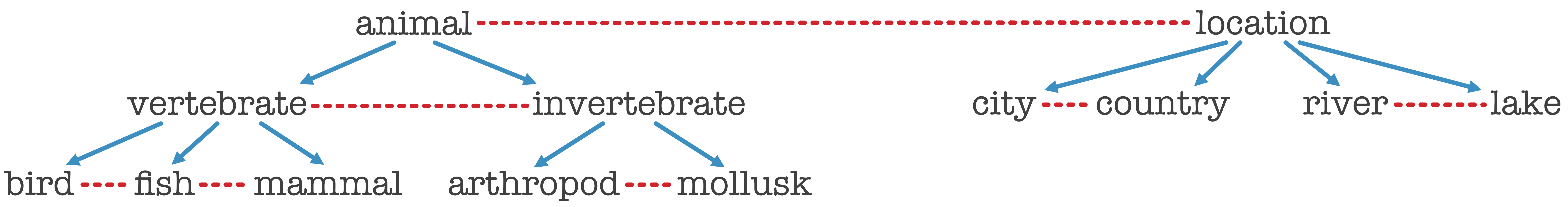
We use **Probabilistic Soft Logic (PSL)** with a customized **stochastic consensus ADMM** algorithm to parallelize inference.

previously unable to run on GPU server
now runs in ~1 hour on a MacBook Pro

[**Platanios**, Poon, Horvitz, Mitchell, NeurIPS 2017]

NELL

6 classifiers | 15 categories | ~550,000 noun phrases

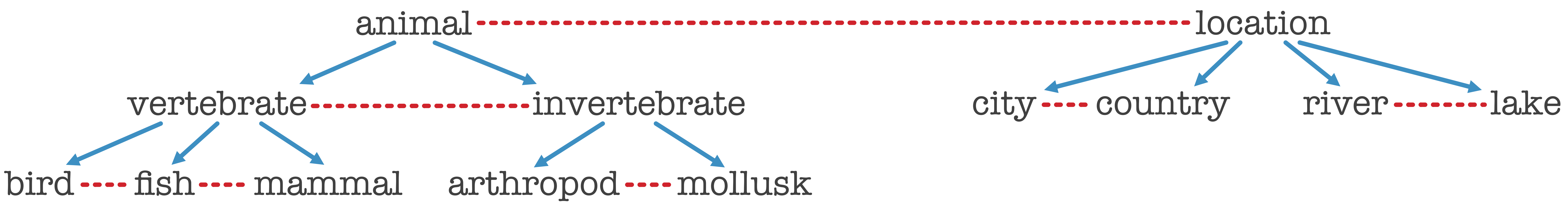


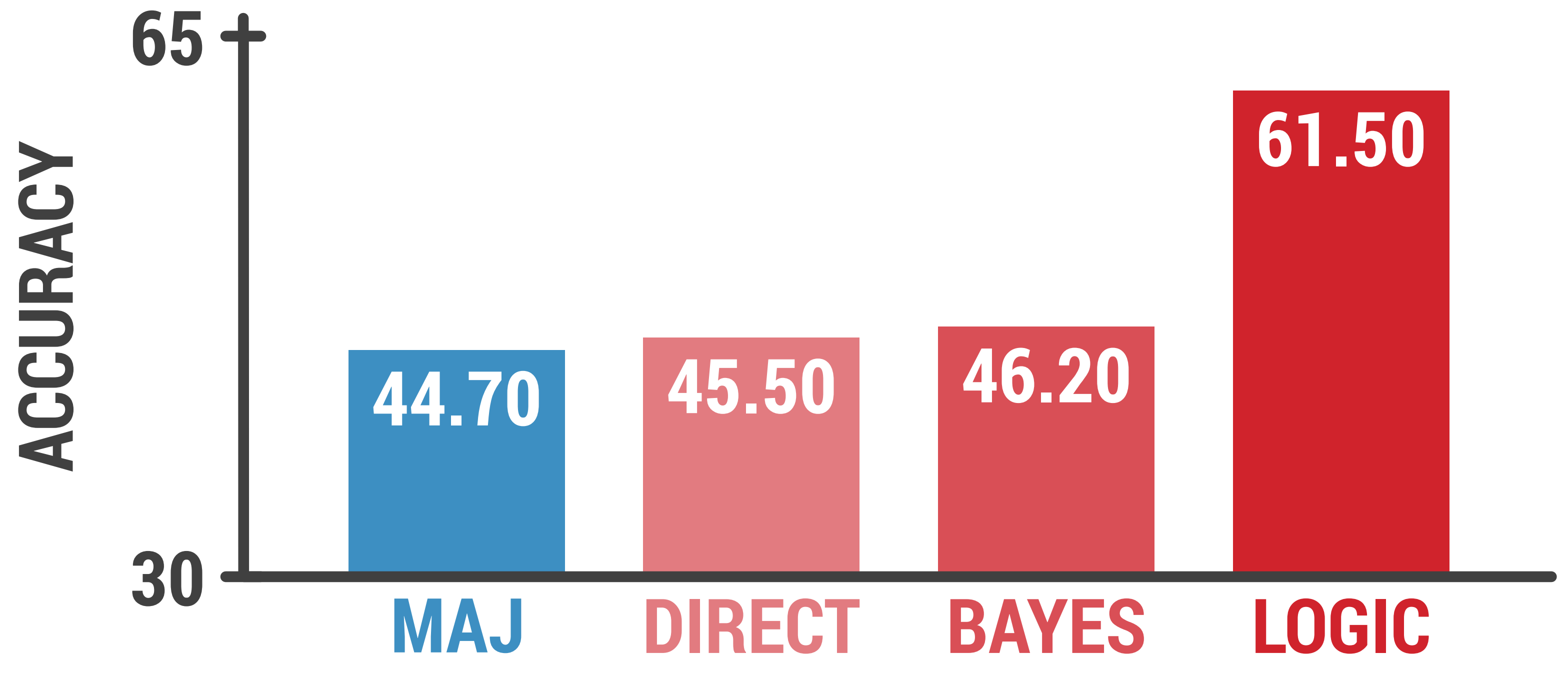
[**Platanios**, Poon, Horvitz, Mitchell, NeurIPS 2017]

A Logic-based Approach

NELL

6 classifiers | 15 categories | ~550,000 noun phrases





[**Platanios**, Poon, Horvitz, Mitchell, NeurIPS 2017]

A Logic-based Approach

NOTE **BRAIN** is harder because the classifiers and the regions are highly dependent!

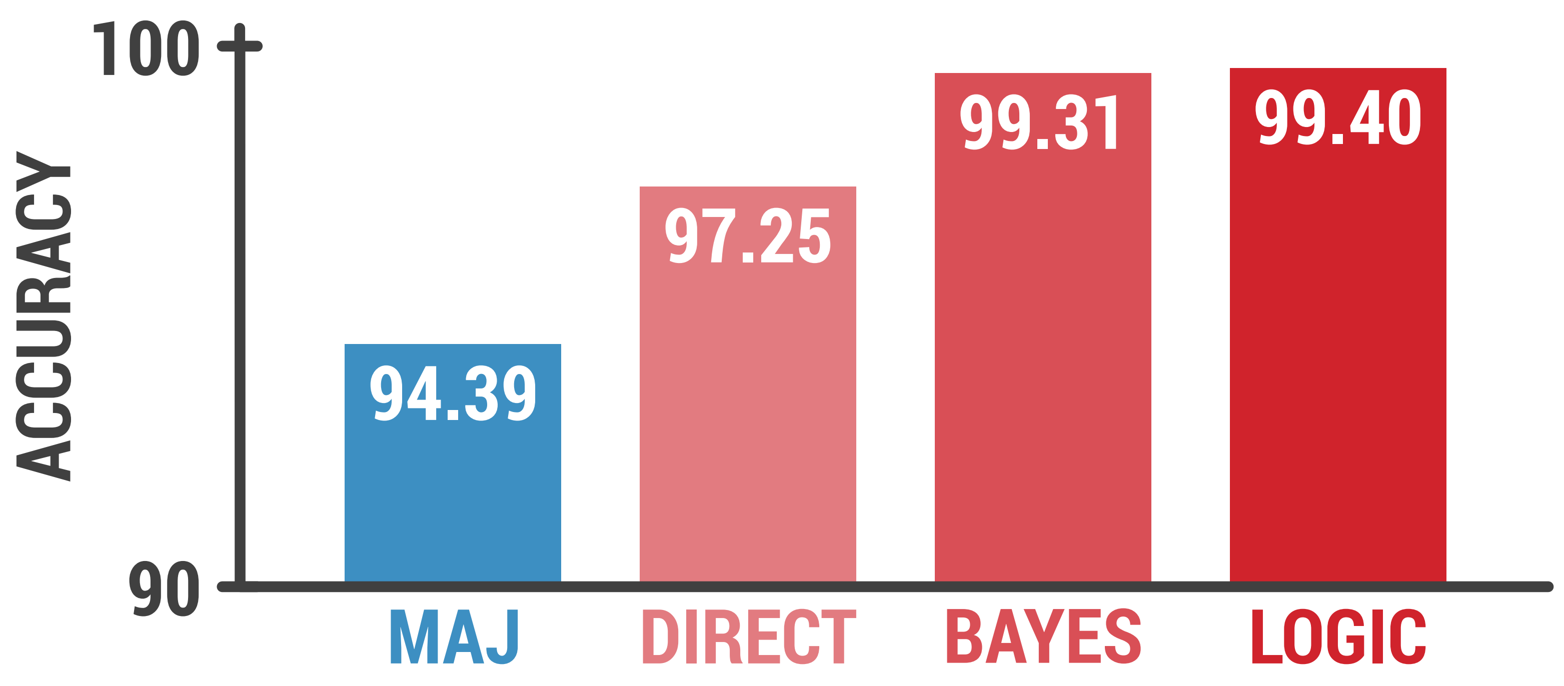
NELL

Task: Predict whether a noun phrase belongs to a category (e.g., `city`).

4 classifiers

15 categories

~300,000 noun phrases



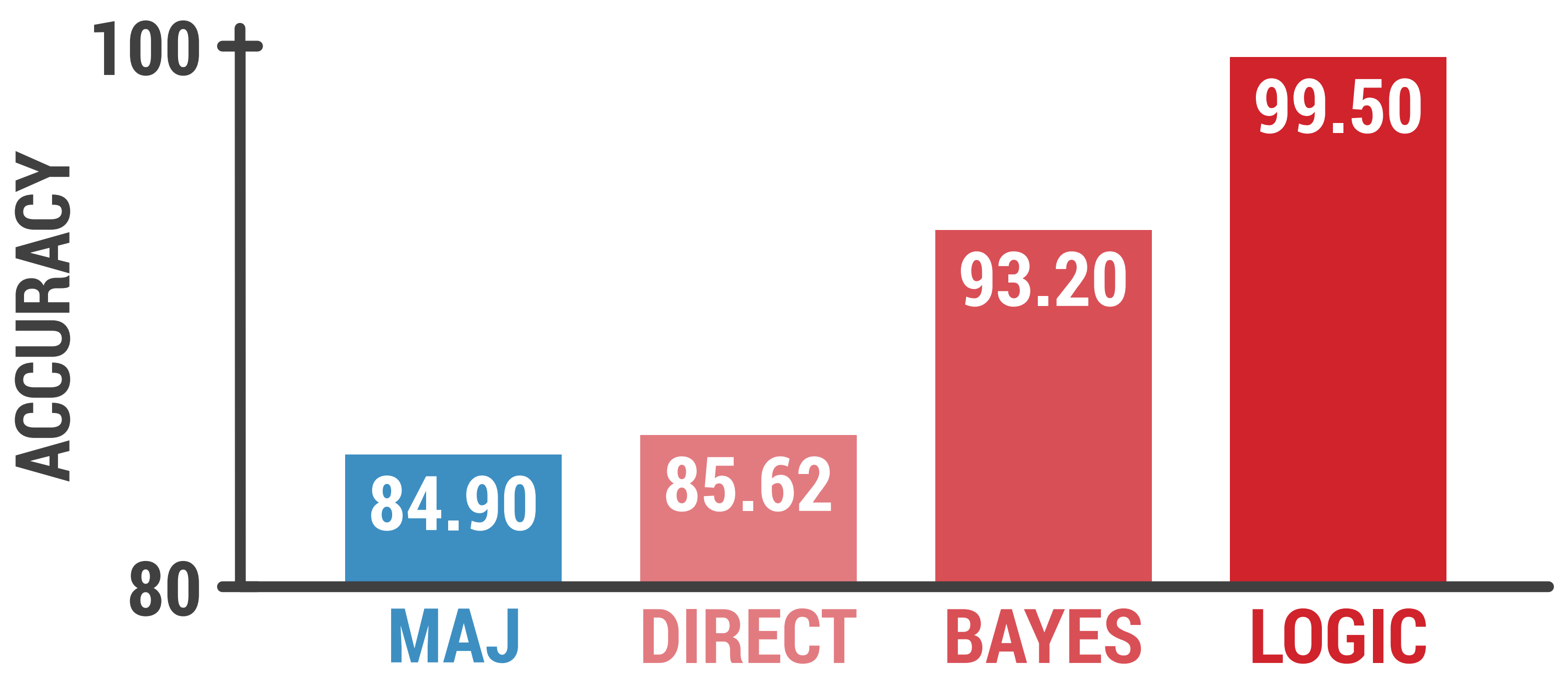
BRAIN

Task: Find which of two 40 second long story passages corresponds to a time series of fMRI neural activity.

11 classifiers

11 brain regions

1,000 passages

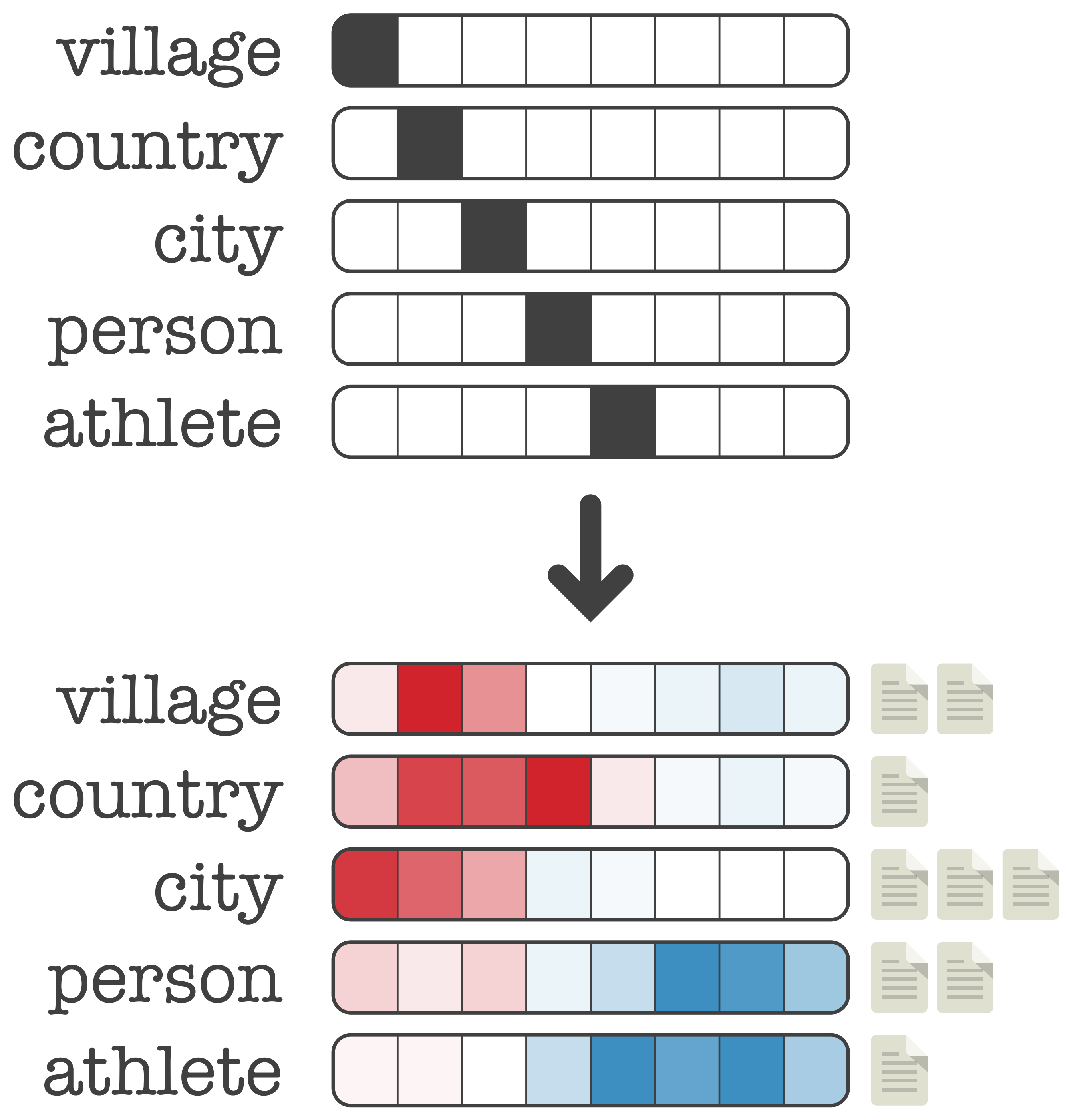


[**Platanios**, Poon, Horvitz, Mitchell, NeurIPS 2017]

Limitation #1
Dependencies

Limitation #2
Logical Constraints

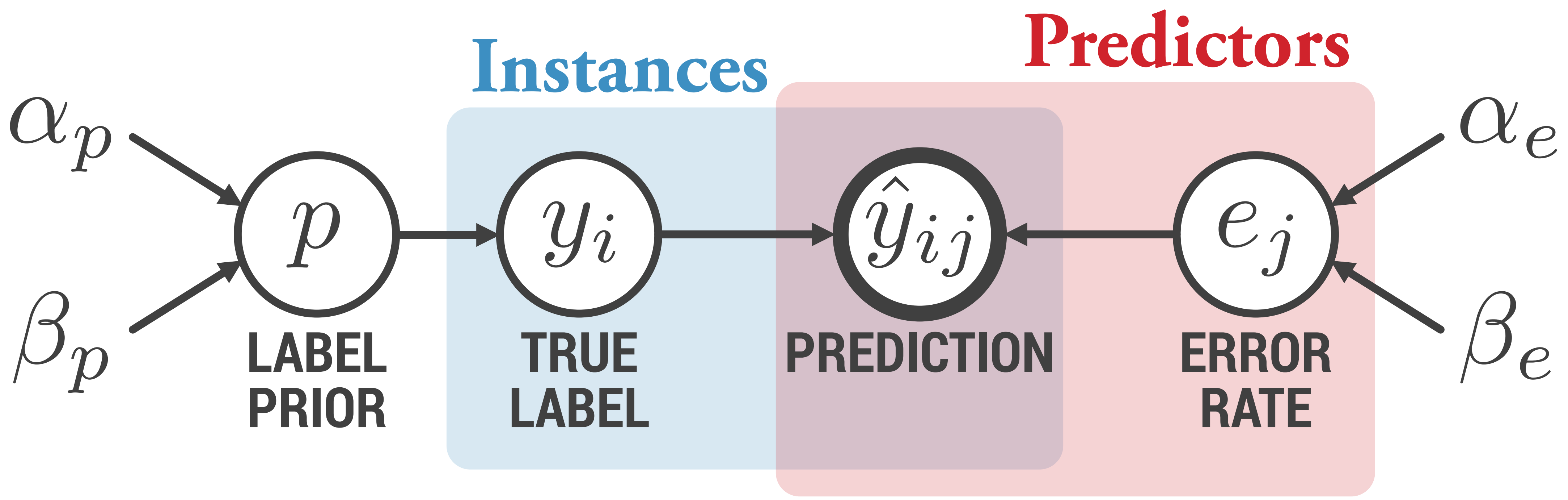
Limitation #3
Representations



similarly for the predictors

A Deep Approach

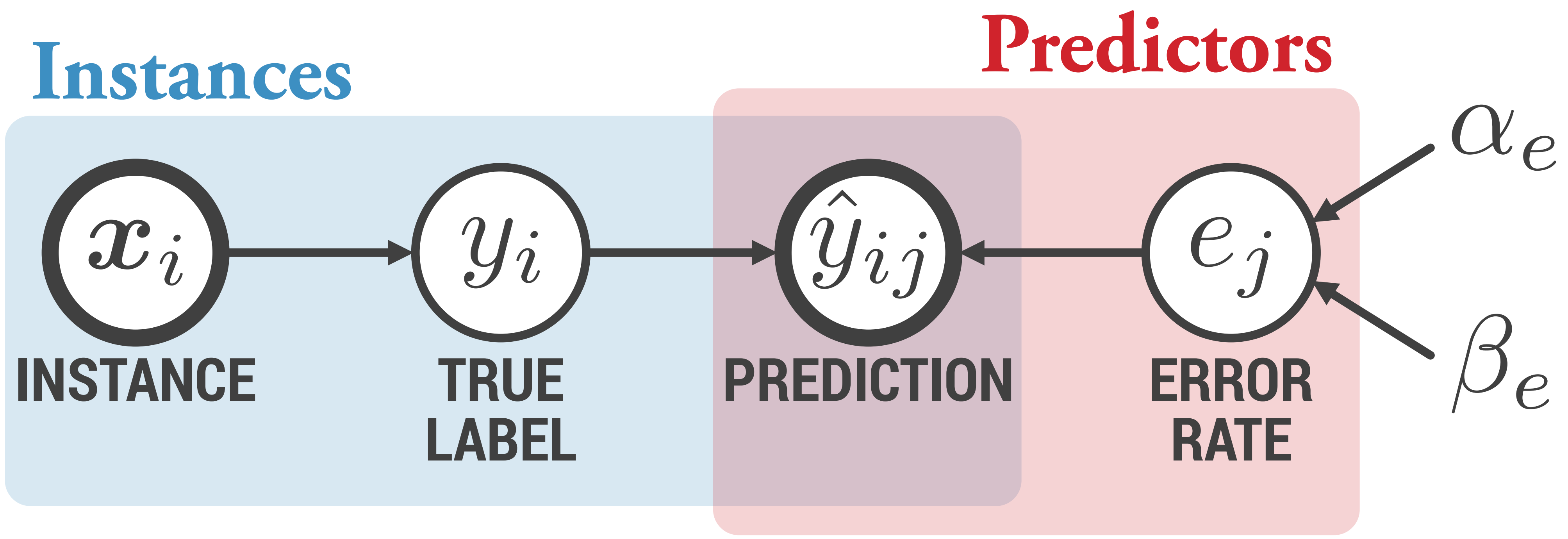
We can start by thinking about how to modify our Bayesian model:



[**Platanios**, Al-Shedivat, Xing, Mitchell, *under review* 2020]

A Deep Approach

We can start by thinking about how to modify our Bayesian model:

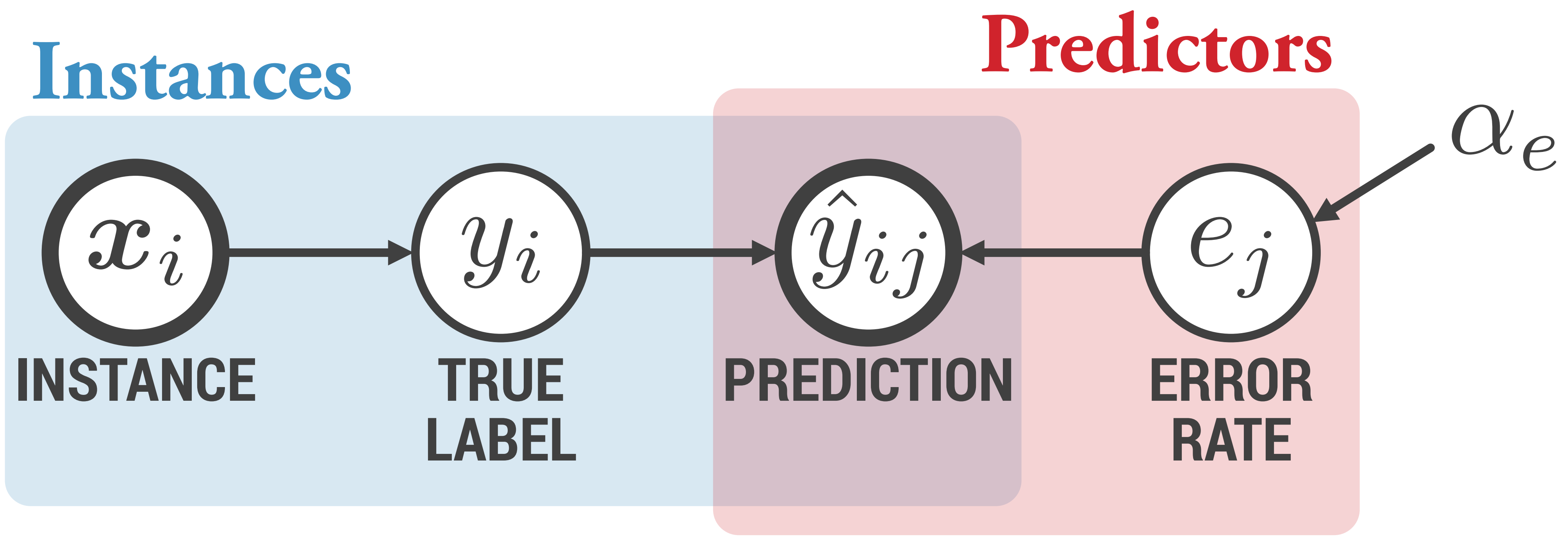


$$y_i \sim \text{Bernoulli}(h_\theta(\boldsymbol{x}_i))$$

[**Platanios**, Al-Shedivat, Xing, Mitchell, *under review* 2020]

A Deep Approach

We can start by thinking about how to modify our Bayesian model:

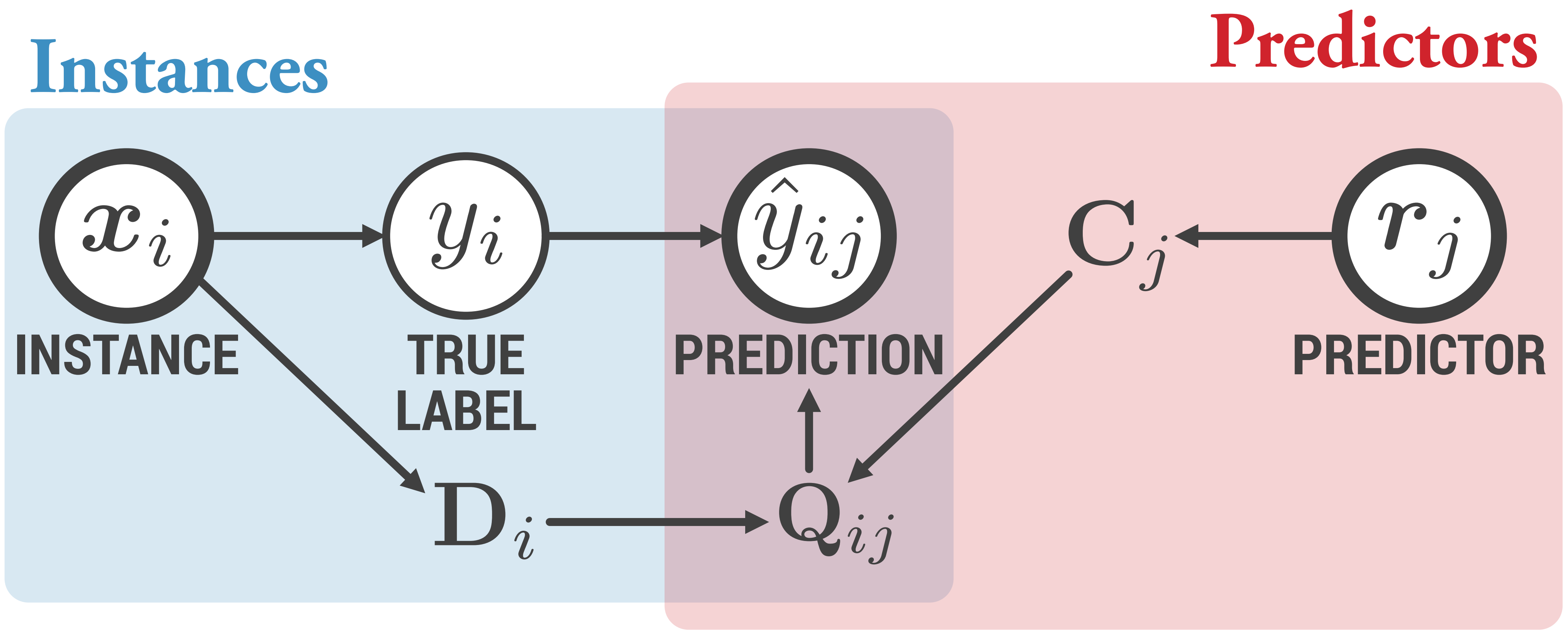


$$y_i \sim \text{Categorical}(h_\theta(\boldsymbol{x}_i))$$
$$\hat{y}_{ij} \sim \text{Categorical}([\,e_j\,]_{y_i\cdot})$$

[**Platanios**, Al-Shedivat, Xing, Mitchell, *under review* 2020]

A Deep Approach

We can start by thinking about how to modify our Bayesian model:



$$\left.\begin{aligned} \mathbf{D}_i &= d_\phi(\boldsymbol{x}_i) \quad \textbf{DIFFICULTY} \\ \mathbf{C}_j &= c_\psi(\boldsymbol{r}_j) \quad \textbf{COMPETENCE} \end{aligned}\right\} \underbrace{\mathbf{Q}_{ij} = \mathbf{D}_i \bullet_3 \mathbf{C}_j}_{\textbf{QUALITY}}$$

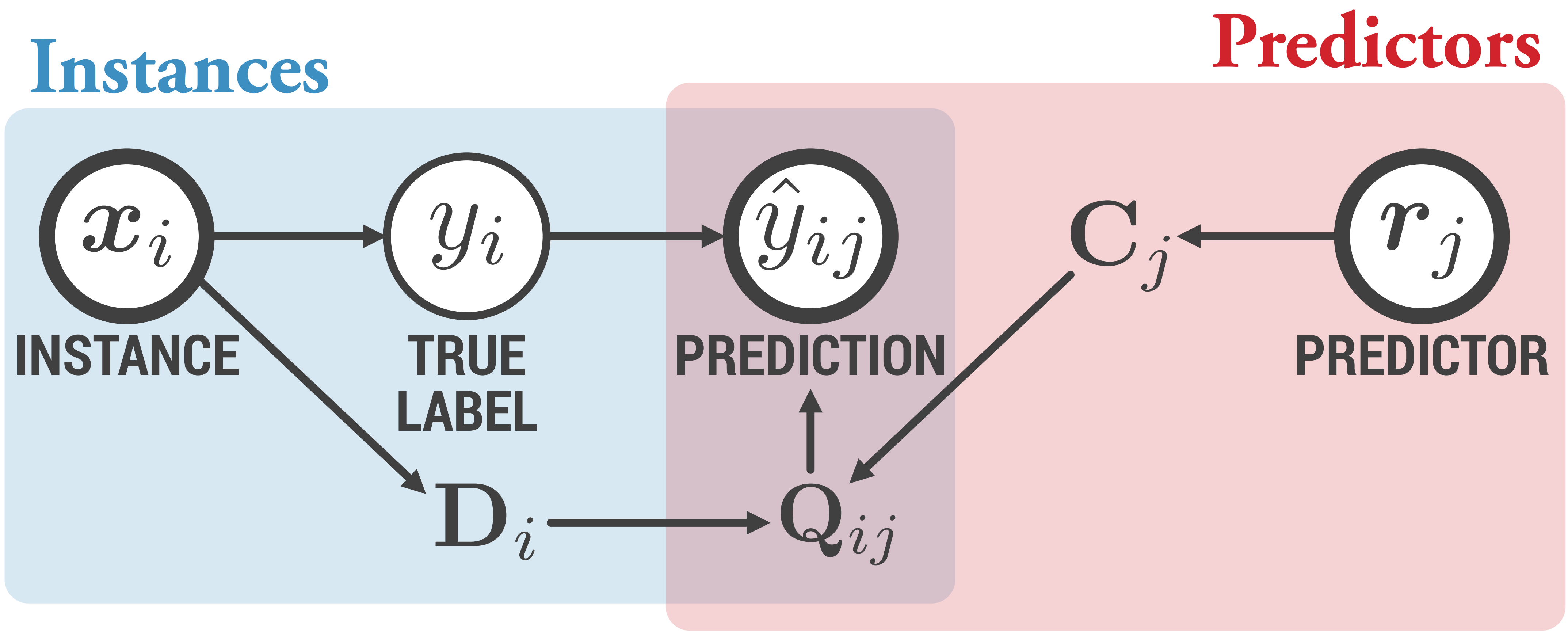
$$y_i \sim \text{Categorical}(h_\theta(\boldsymbol{x}_i))$$

$$\hat{y}_{ij} \sim \text{Categorical}([\mathbf{Q}_{ij}]_{y_i \cdot})$$

[**Platanios**, Al-Shedivat, Xing, Mitchell, *under review* 2020]

A Deep Approach

We can start by thinking about how to modify our Bayes



$$\left.\begin{aligned} \mathbf{D}_i &= d_\phi(\boldsymbol{x}_i) \quad \textbf{DIFFICULTY} \\ \mathbf{C}_j &= c_\psi(\boldsymbol{r}_j) \quad \textbf{COMPETENCE} \end{aligned}\right\} \underbrace{\mathbf{Q}_{ij} = \mathbf{D}_i \bullet_3 \mathbf{C}_j}_{\textbf{QUALITY}}$$

$$y_i \sim \text{Categorical}(h_\theta(\boldsymbol{x}_i))$$

$$\hat{y}_{ij} \sim \text{Categorical}([\mathbf{Q}_{ij}]_{y_i \cdot})$$

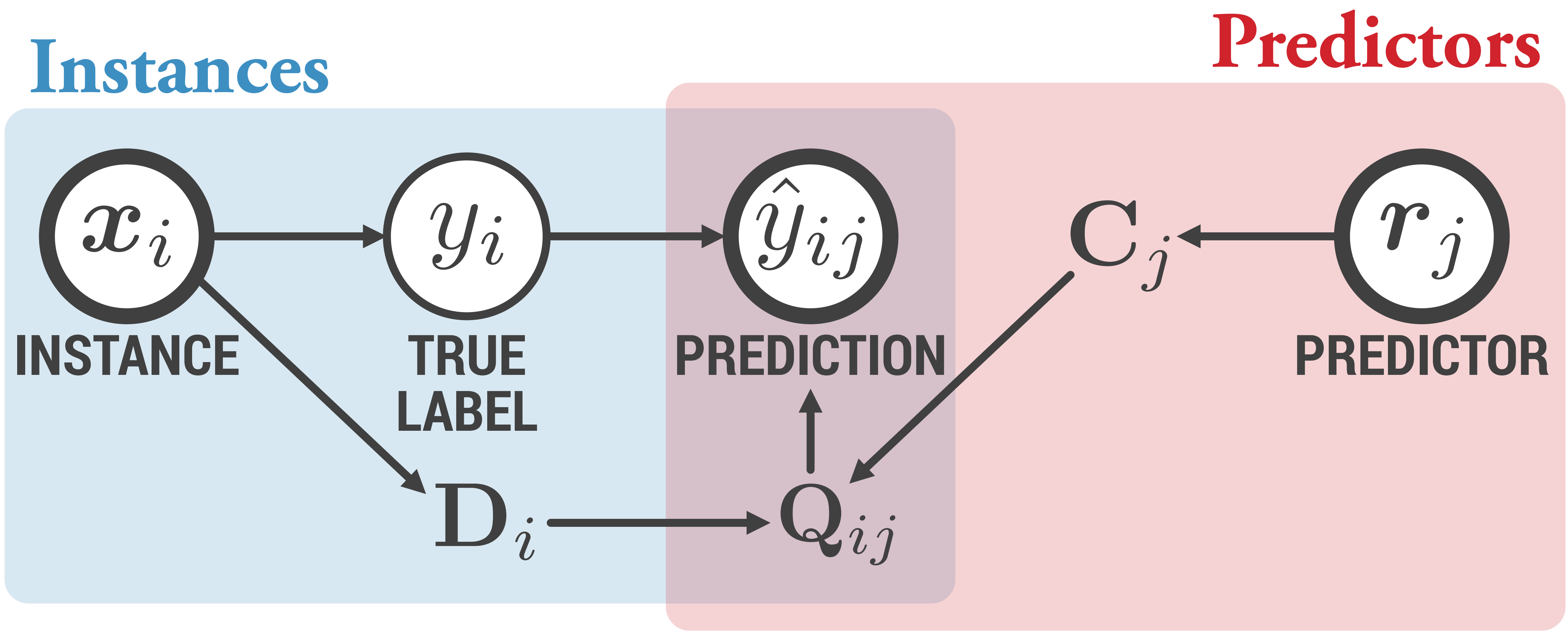
[**Platanios**, Al-Shedivat, Xing, Mitchell, *under review* 2020]

inference

We use the Expectation-Maximization (EM) algorithm.

A Deep Approach

We can start by thinking about how to modify our Bayes



$$\left.\begin{aligned} \mathbf{D}_i &= d_\phi(\boldsymbol{x}_i) \quad \textbf{DIFFICULTY} \\ \mathbf{C}_j &= c_\psi(\boldsymbol{r}_j) \quad \textbf{COMPETENCE} \end{aligned}\right\} \underbrace{\mathbf{Q}_{ij} = \mathbf{D}_i \bullet_3 \mathbf{C}_j}_{\textbf{QUALITY}}$$

$$y_i \sim \text{Categorical}(h_\theta(\boldsymbol{x}_i))$$

$$\hat{y}_{ij} \sim \text{Categorical}([\mathbf{Q}_{ij}]_{y_i \cdot})$$

inference

We use the Expectation-Maximization (EM) algorithm.

E-STEP

Compute the expectation of the latent true labels:

$$\mathbb{E}_{\boldsymbol{y}|\mathcal{D}}\left\{\mathbb{1}_{[y_i=k]}\right\} = \frac{\lambda_i^k}{\sum_{l=1}^C \lambda_i^l}$$

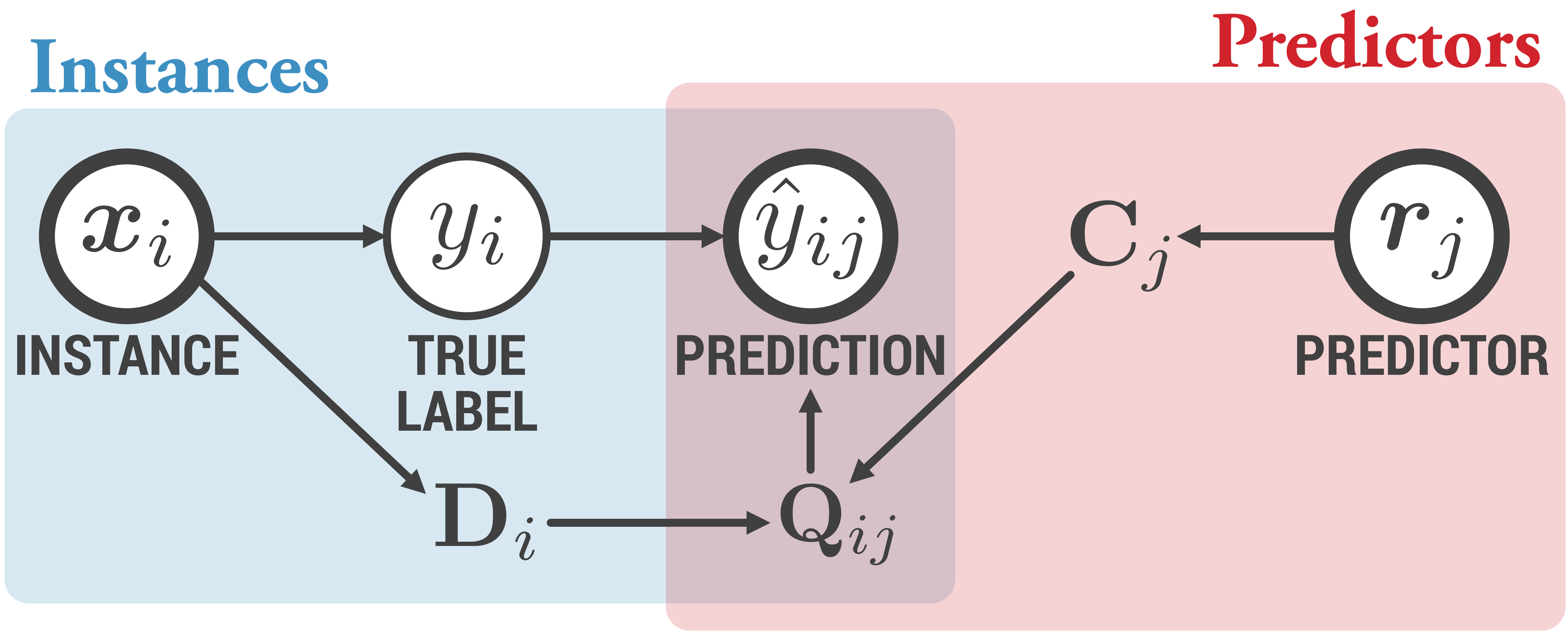
where:

$$\lambda_i^k = [h_\theta(\boldsymbol{x}_i)]_k \prod_{j \in \mathcal{M}_i} \frac{[\mathbf{Q}_{ij}]_{k\hat{y}_{ij}}}{\sum_{l=1}^C [\mathbf{Q}_{ij}]_{l\hat{y}_{ij}}}$$

[**Platanios**, Al-Shedivat, Xing, Mitchell, *under review* 2020]

A Deep Approach

We can start by thinking about how to modify our Bayes



$$\left.\begin{aligned} \mathbf{D}_i &= d_\phi(\boldsymbol{x}_i) \quad \textbf{DIFFICULTY} \\ \mathbf{C}_j &= c_\psi(\boldsymbol{r}_j) \quad \textbf{COMPETENCE} \end{aligned}\right\} \underbrace{\mathbf{Q}_{ij} = \mathbf{D}_i \bullet_3 \mathbf{C}_j}_{\textbf{QUALITY}}$$

$$y_i \sim \text{Categorical}(h_\theta(\boldsymbol{x}_i))$$

$$\hat{y}_{ij} \sim \text{Categorical}([\mathbf{Q}_{ij}]_{y_i\cdot})$$

inference

We use the Expectation-Maximization (EM) algorithm.

M-STEP

Maximize the data likelihood:

$$\log \mathcal{L} = \sum_{i=1}^{N} \sum_{k=1}^{C} \tilde{y}_i^k \log[h_\theta(\boldsymbol{x}_i)]_k$$

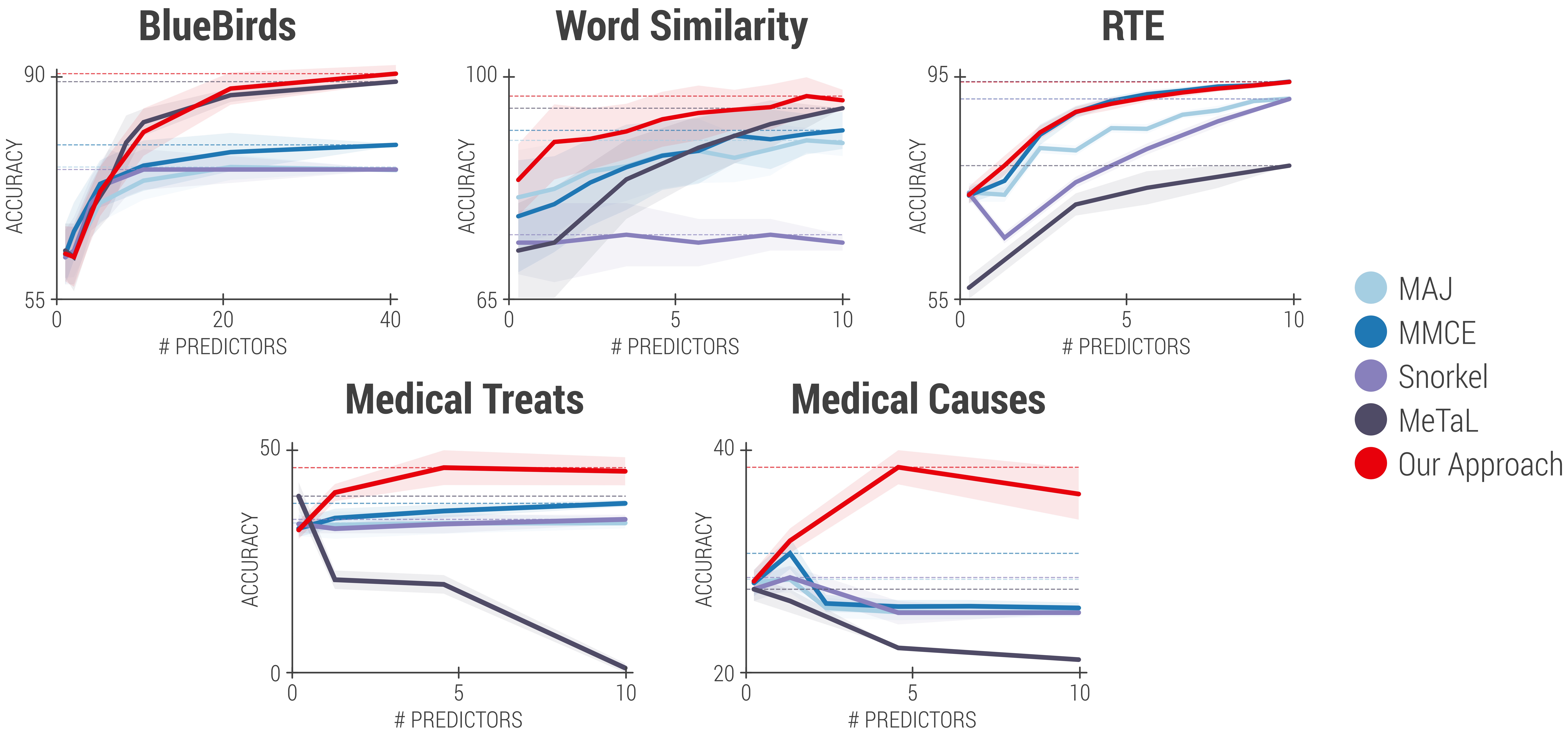
$$+ \sum_{i=1}^{N} \sum_{j \in \mathcal{M}_i} \sum_{k=1}^{C} \tilde{y}_i^k \frac{[\mathbf{Q}_{ij}]_{k\hat{y}_{ij}}}{\sum_{l=1}^{C} [\mathbf{Q}_{ij}]_{l\hat{y}_{ij}}}$$

[**Platanios**, Al-Shedivat, Xing, Mitchell, *under review* 2020]

A Deep Approach

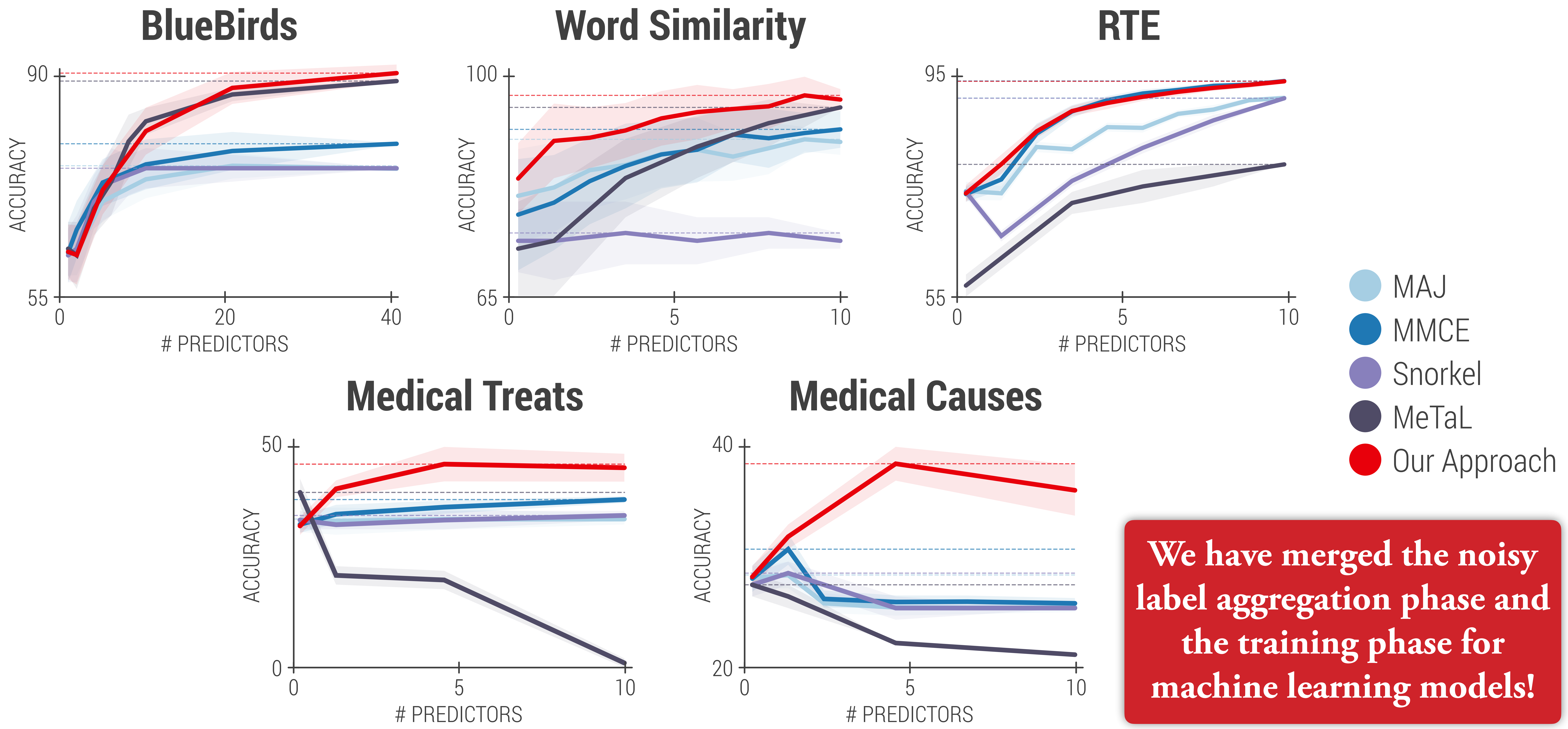
[**Platanios**, Al-Shedivat, Xing, Mitchell, *under review* 2020]

A Deep Approach



[**Platanios**, Al-Shedivat, Xing, Mitchell, *under review* 2020]

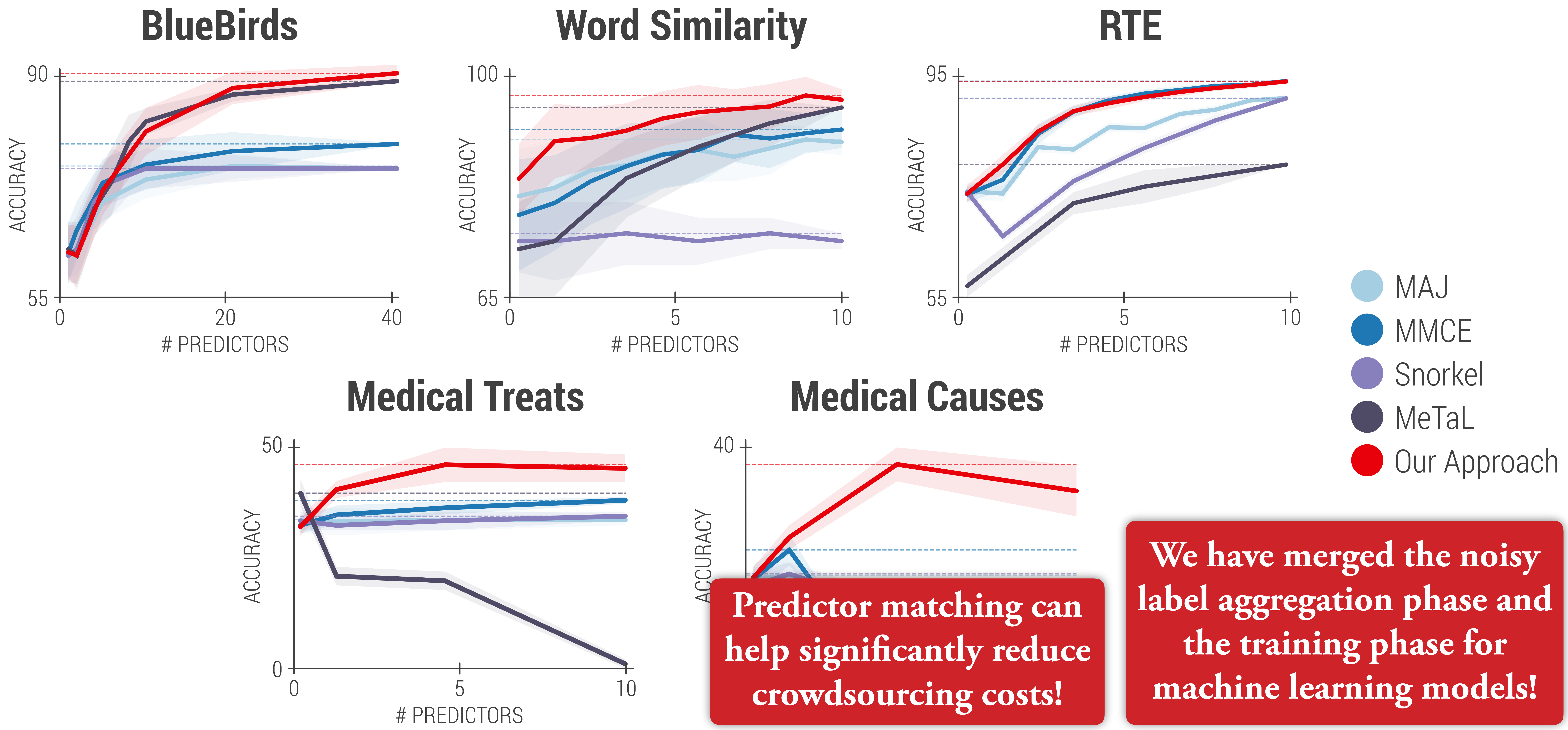
A Deep Approach



We have merged the noisy label aggregation phase and the training phase for machine learning models!

[**Platanios**, Al-Shedivat, Xing, Mitchell, *under review* 2020]

A Deep Approach

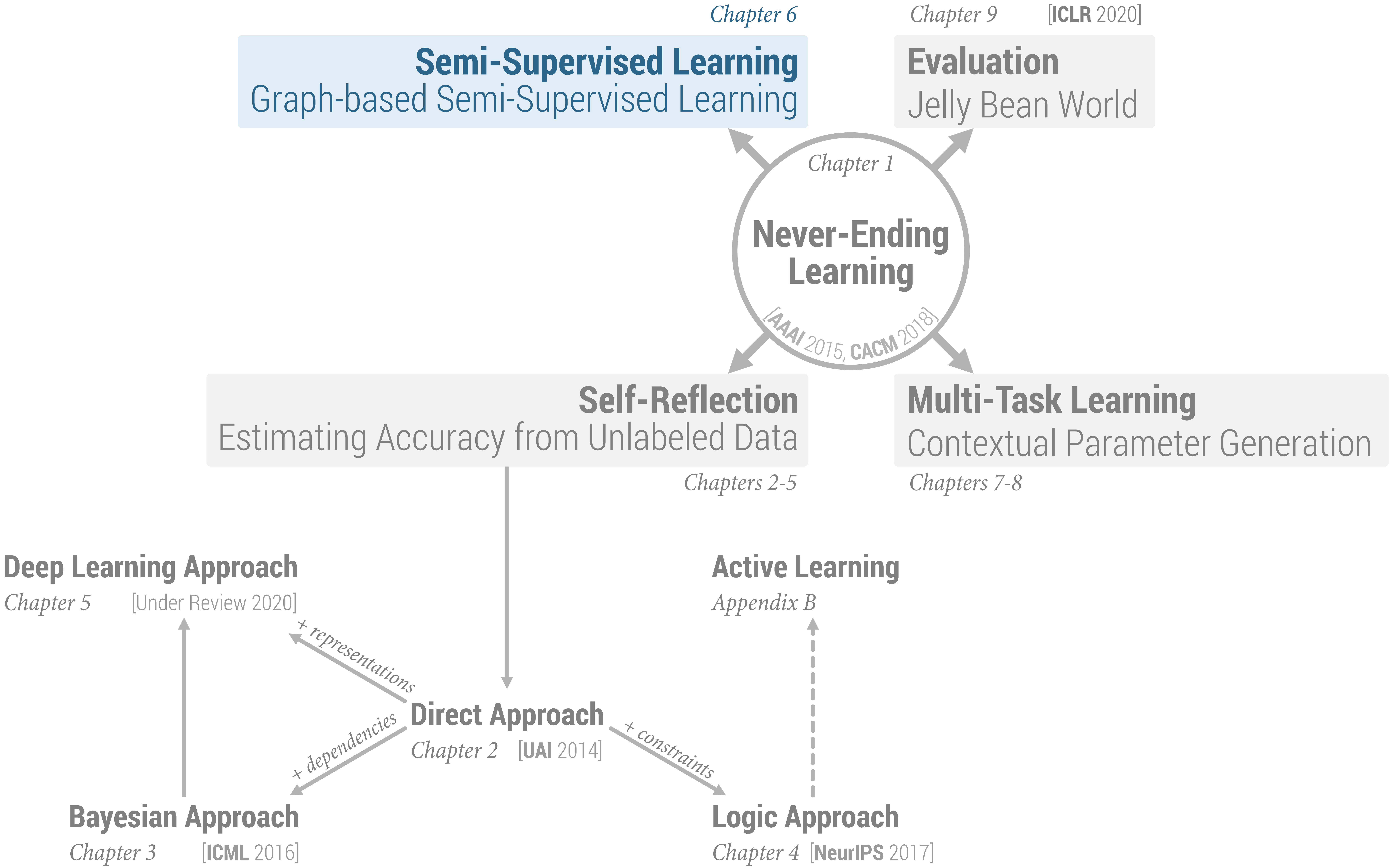


Predictor matching can help significantly reduce crowdsourcing costs!

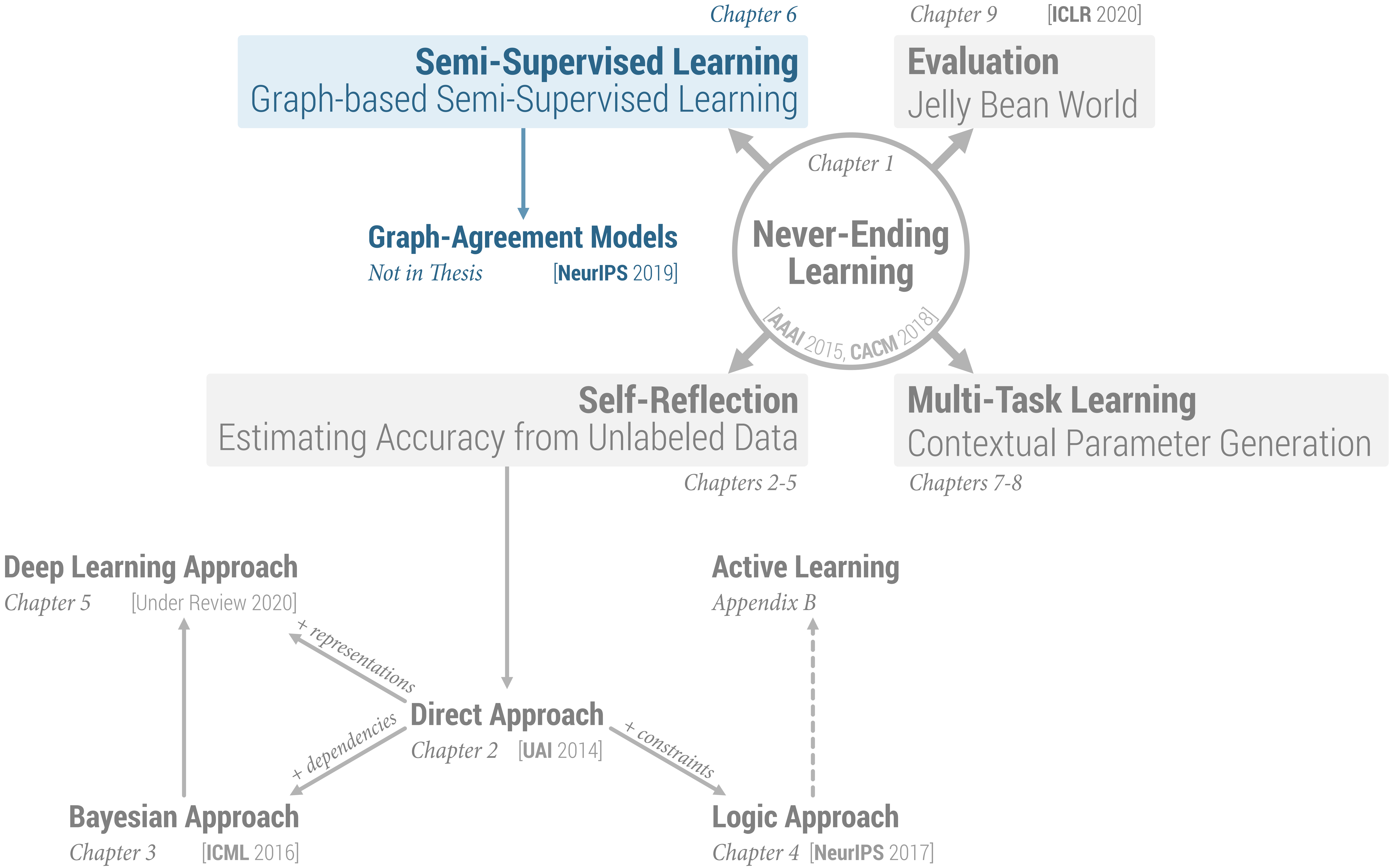
We have merged the noisy label aggregation phase and the training phase for machine learning models!

[**Platanios**, Al-Shedivat, Xing, Mitchell, *under review* 2020]

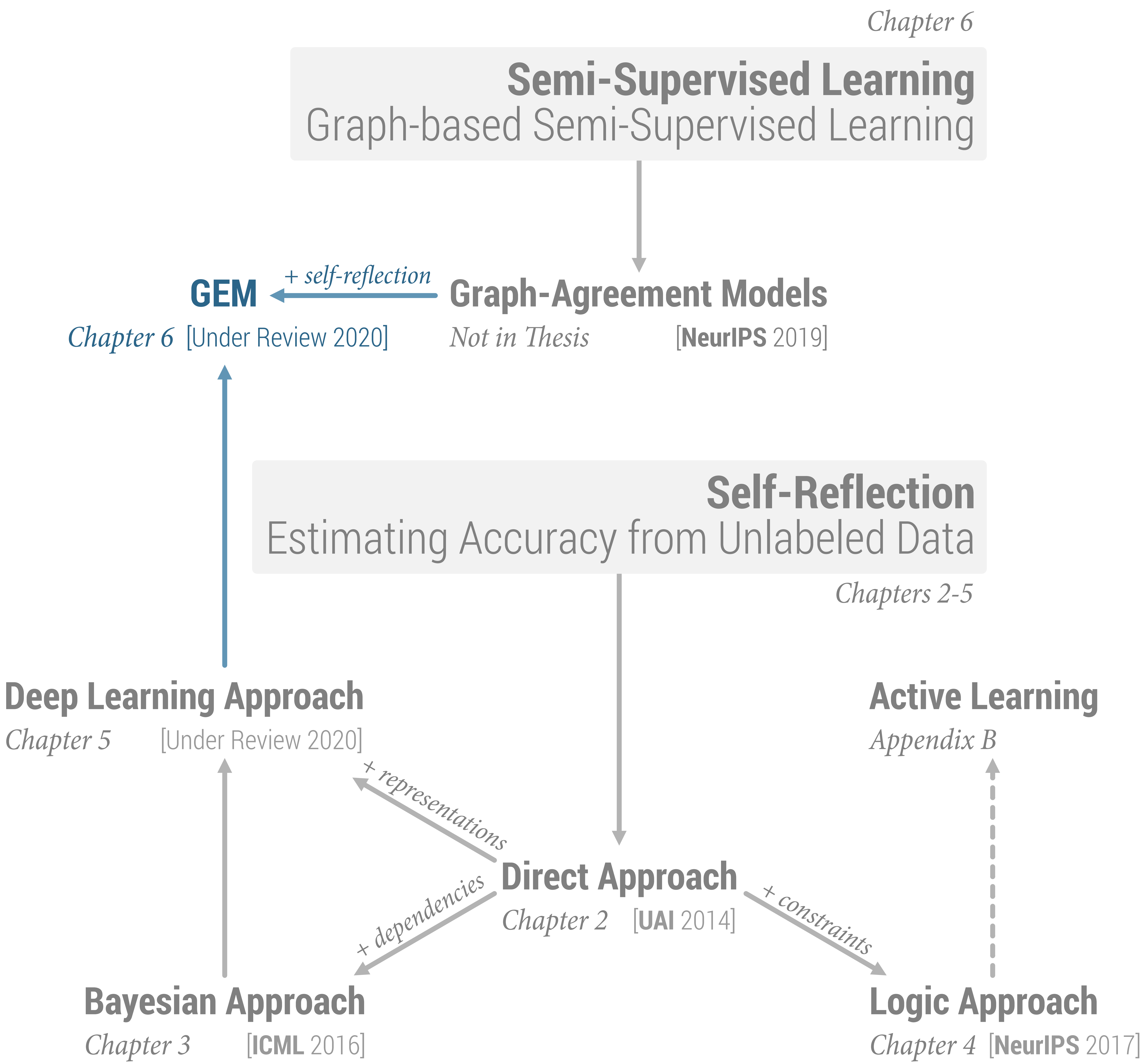
Self-Reflection



Self-Reflection

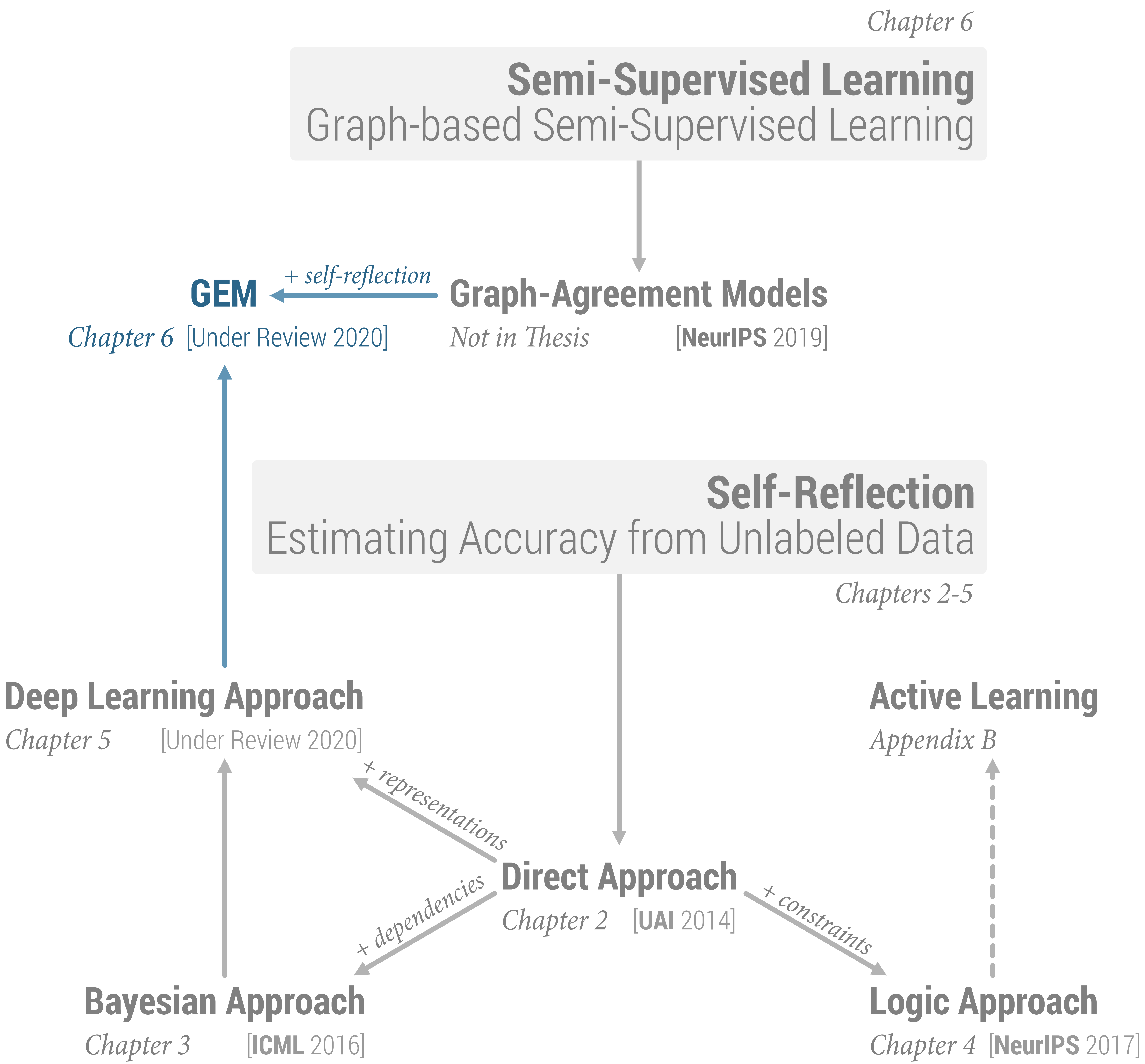


Self-Reflection

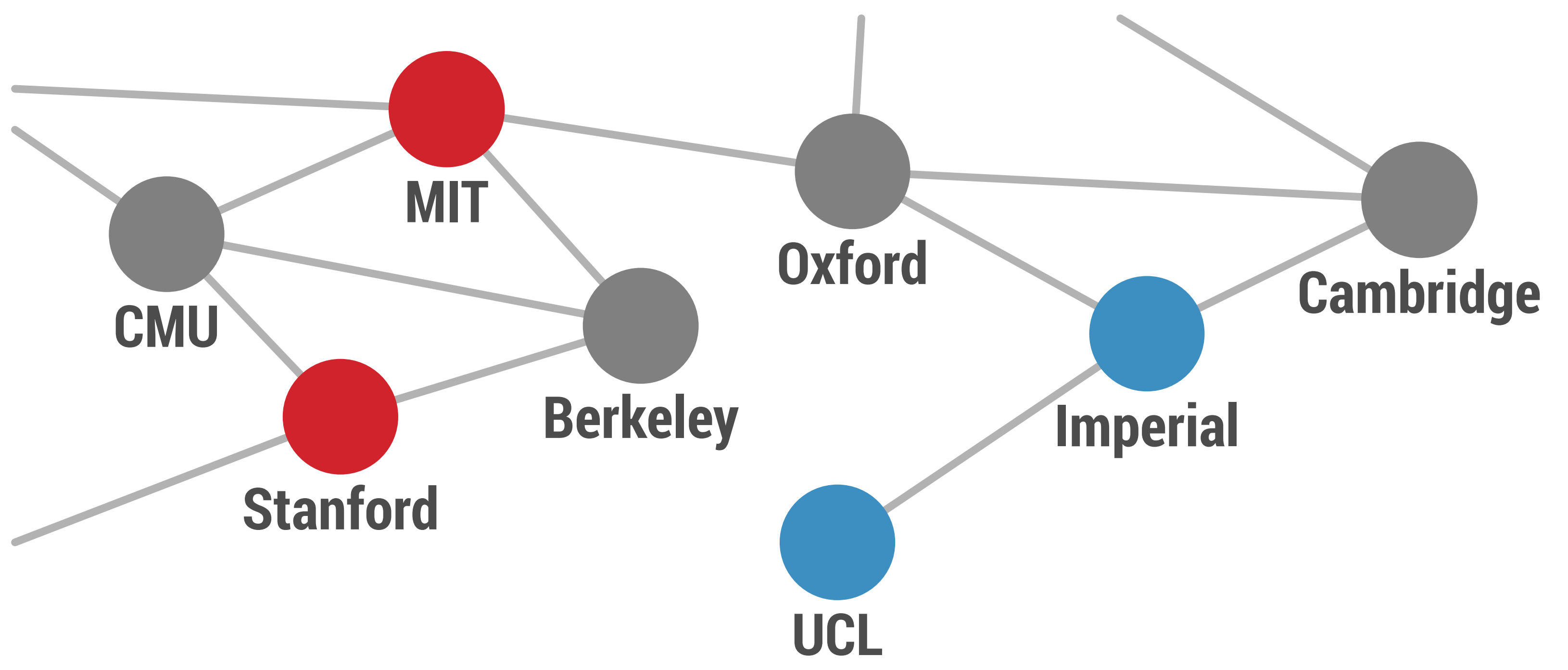


Self-Reflection over Graphs

Self-Reflection



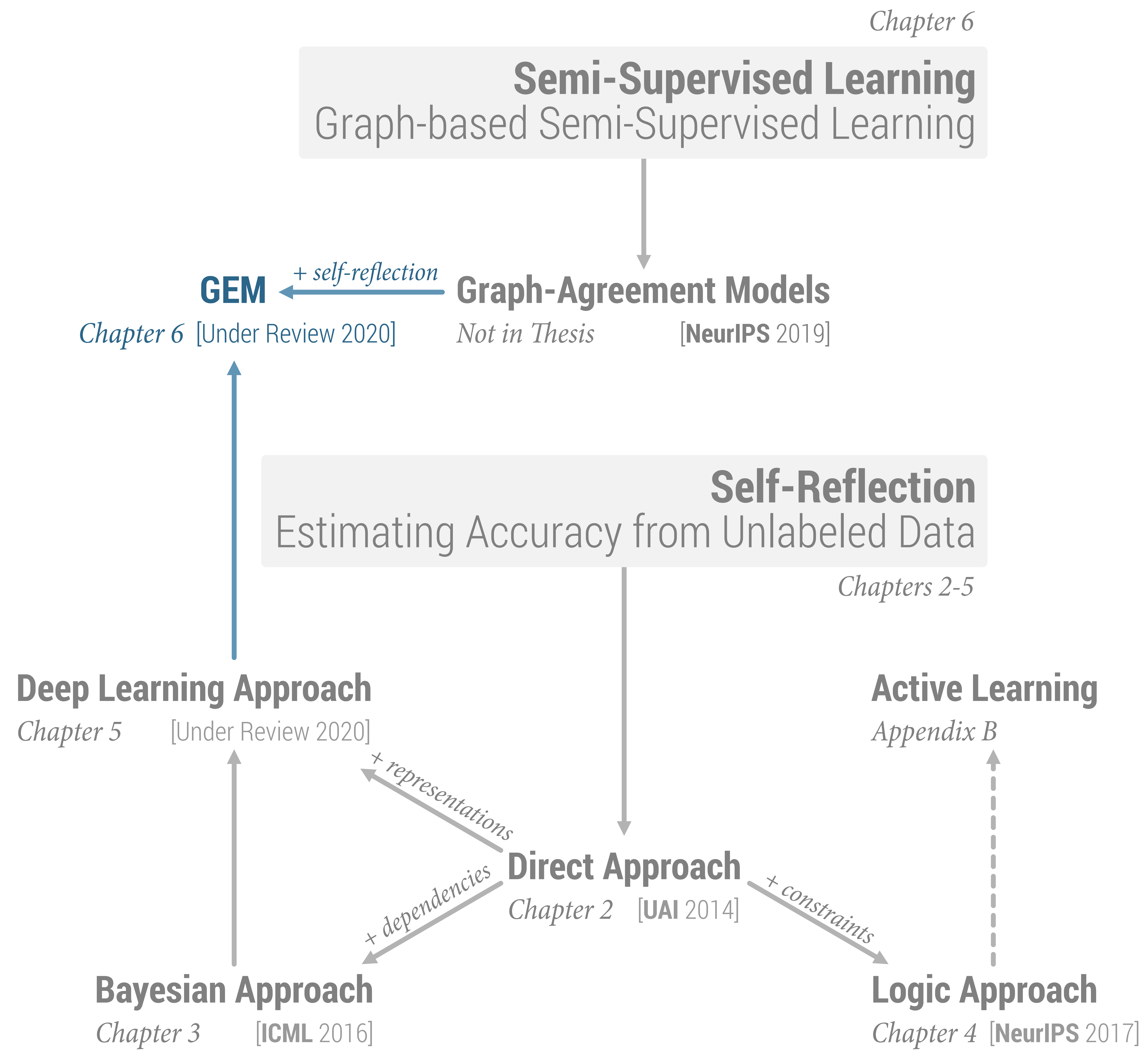
Self-Reflection over Graphs



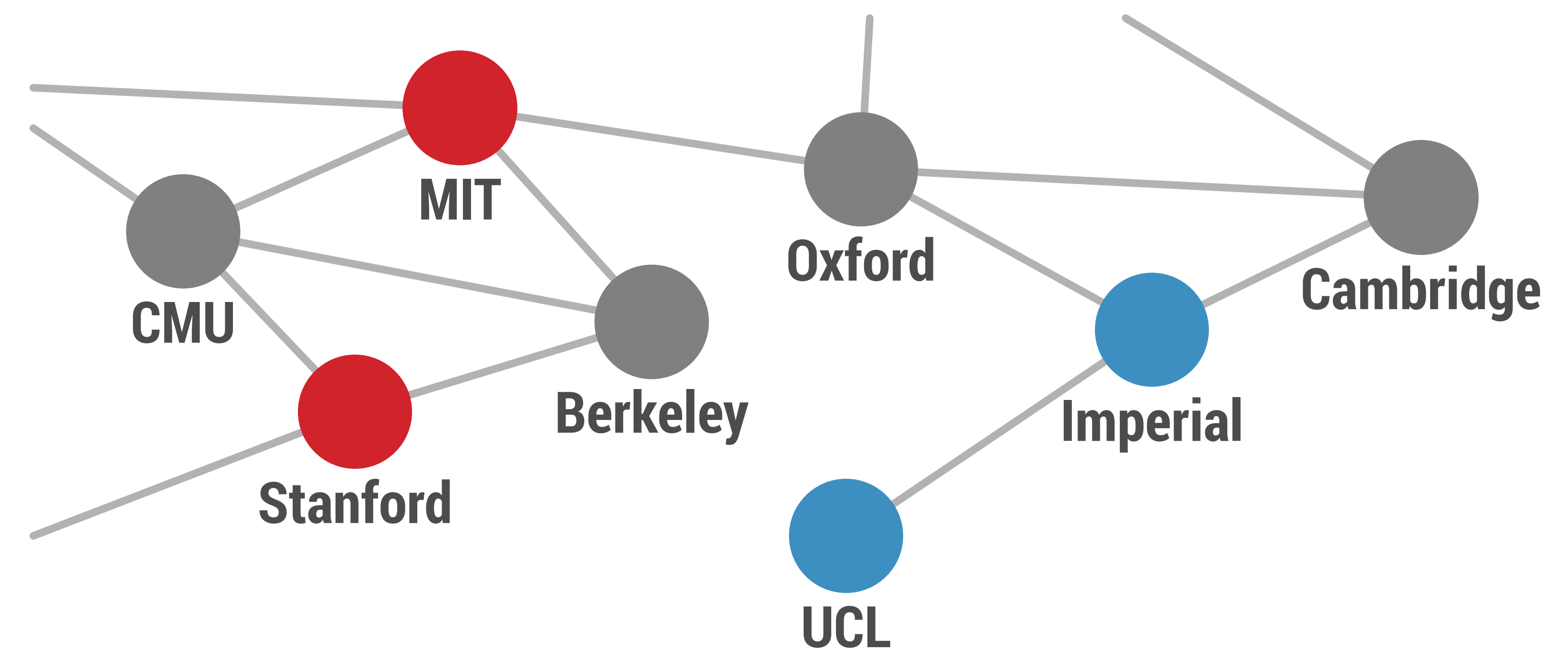
LEGEND

- Unlabeled
- Located in the USA
- Located in the UK
- Research Collaboration

Self-Reflection



Self-Reflection over Graphs



LEGEND

- Unlabeled
- Located in the USA
- Located in the UK
- Research Collaboration

We can *treat the labels of a node's neighbors as noisy predictors* of the node's label and apply our self-reflection methods!

Self-Reflection

Chapter 6

Semi-Supervised Learning
Graph-based Semi-Supervised Learning

GEM ← *+ self-reflection* — **Graph-Agreement Models**

Chapter 6 [Under Review 2020]

Not in Thesis [**NeurIPS** 2019]

Self-Reflection
Estimating Accuracy from Unlabeled Data

Chapters 2-5

Deep Learning Approach
Chapter 5 [Under Review 2020]

Active Learning
Appendix B

+ representations

Direct Approach
Chapter 2 [**UAI** 2014]

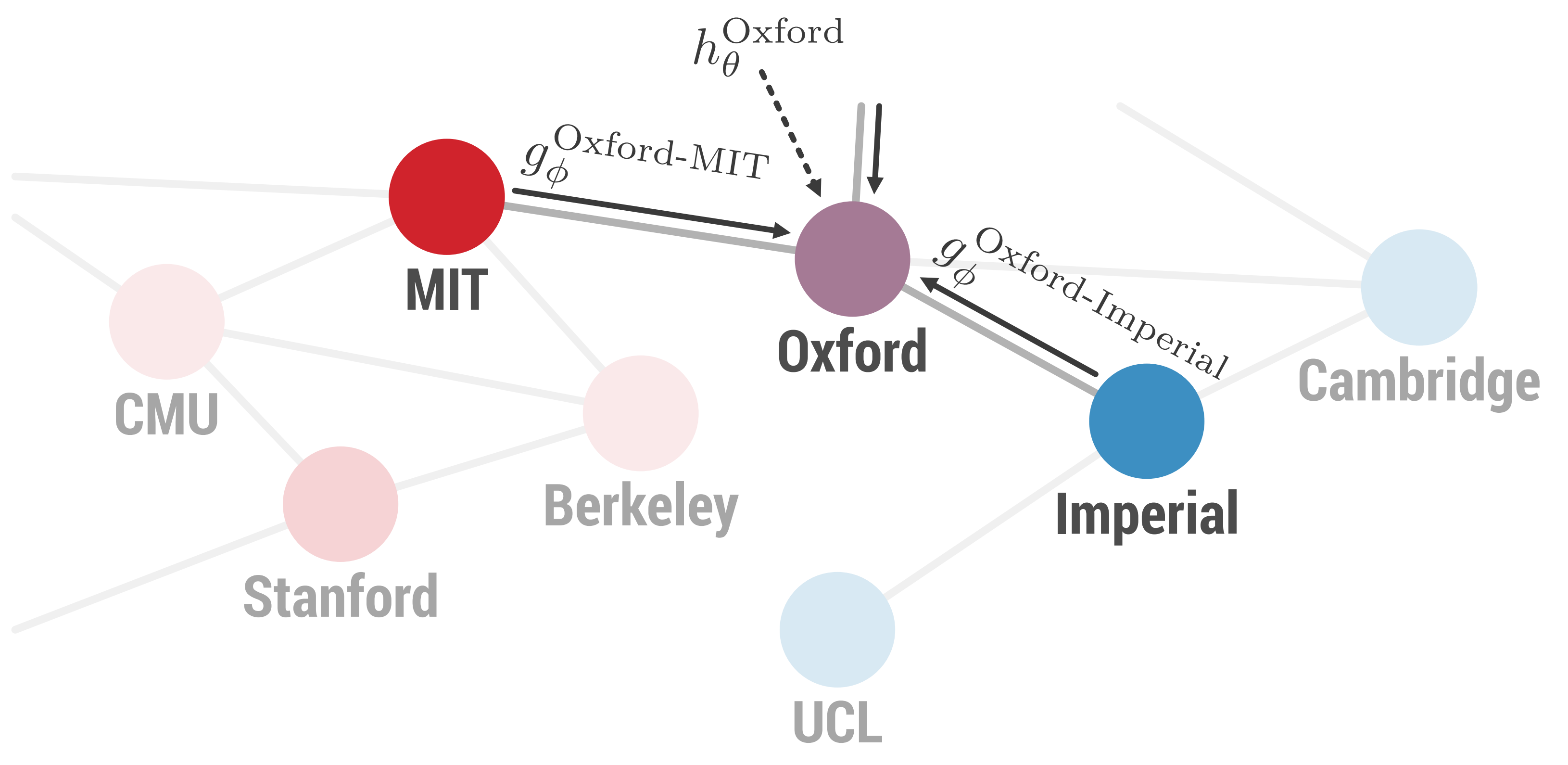
+ dependencies

+ constraints

Bayesian Approach
Chapter 3 [**ICML** 2016]

Logic Approach
Chapter 4 [**NeurIPS** 2017]

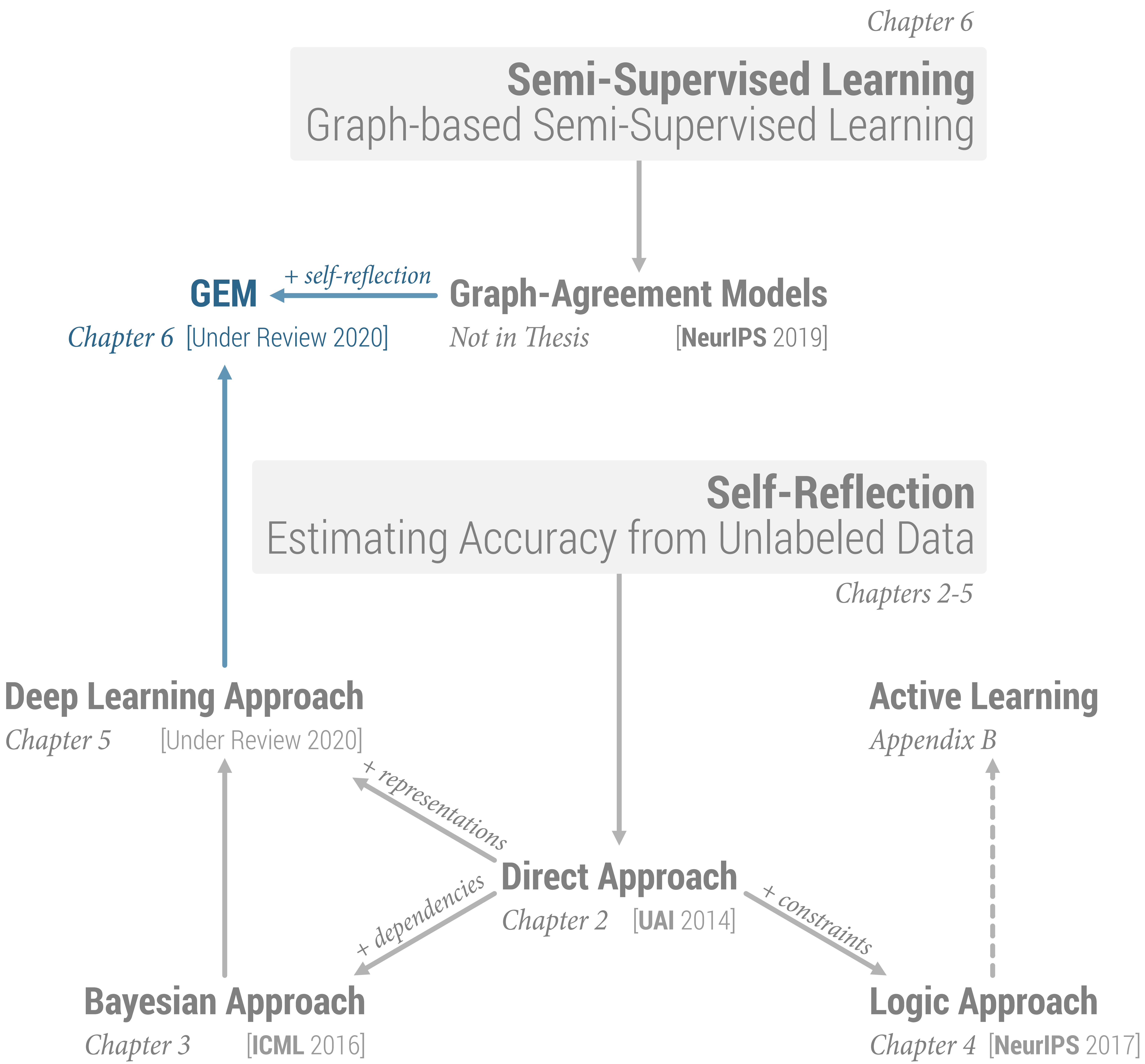
Self-Reflection over Graphs



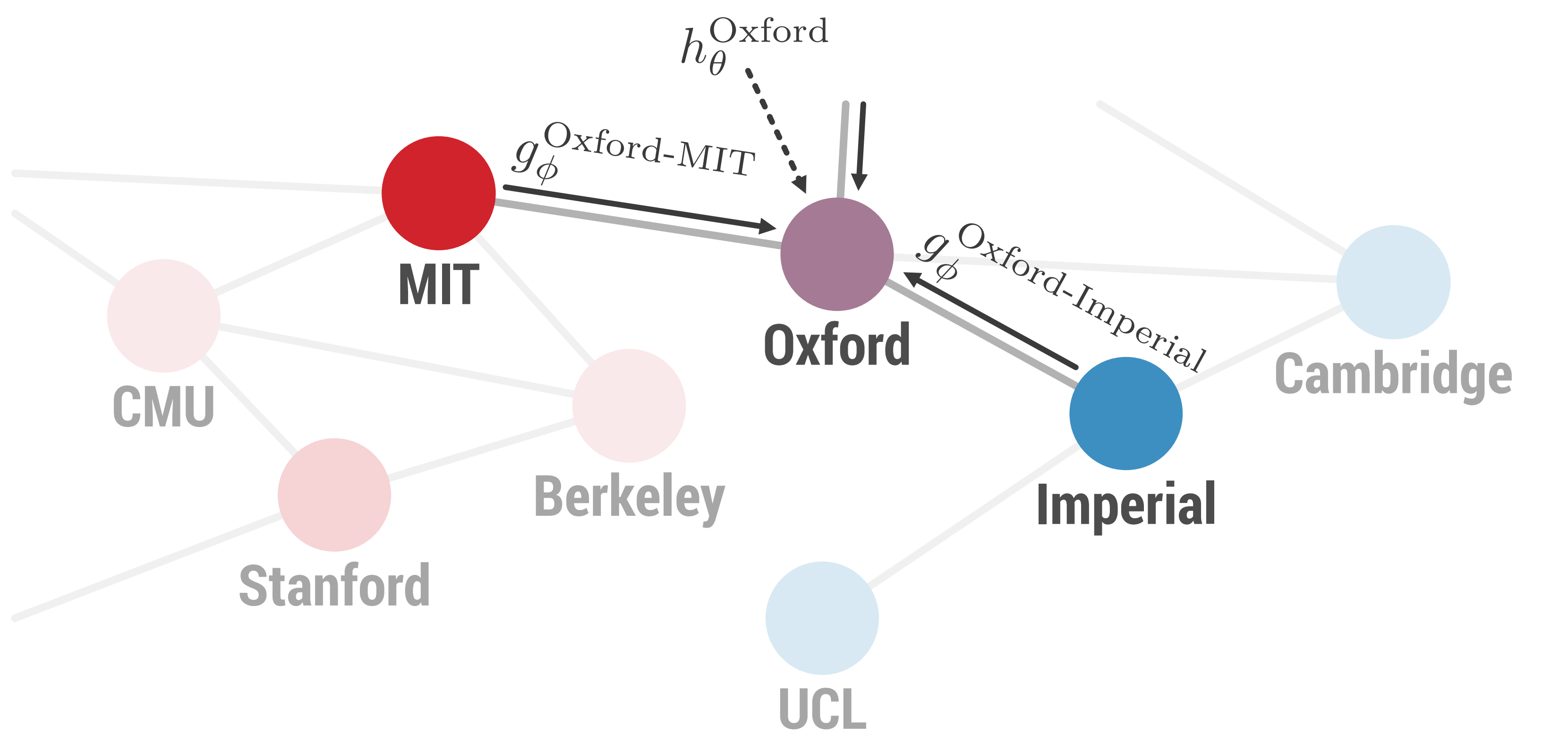
UPDATE LABELS

Compute the expected labels of the unlabeled nodes using the current model parameters.

Self-Reflection

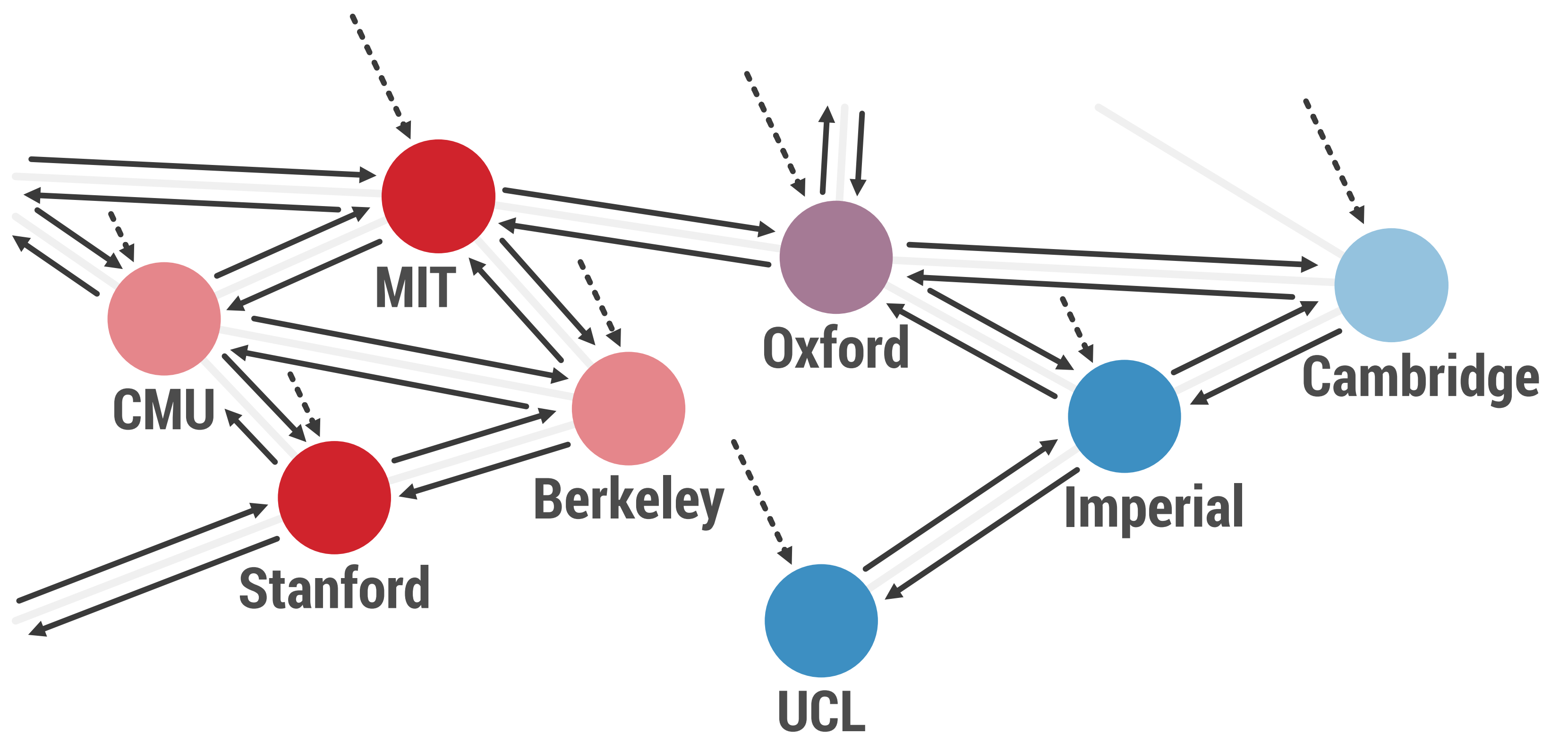


Self-Reflection over Graphs



UPDATE LABELS

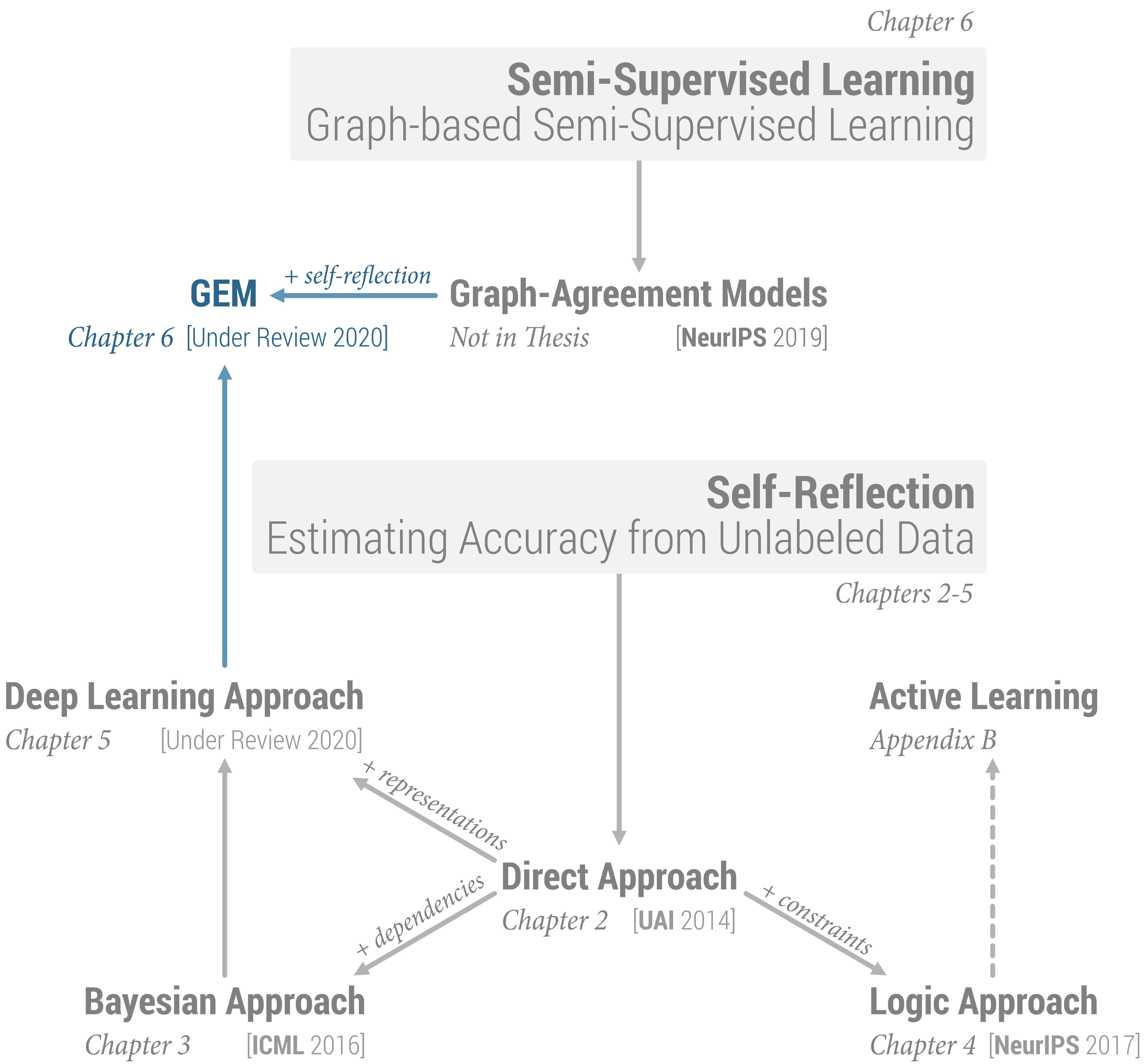
Compute the expected labels of the unlabeled nodes using the current model parameters.



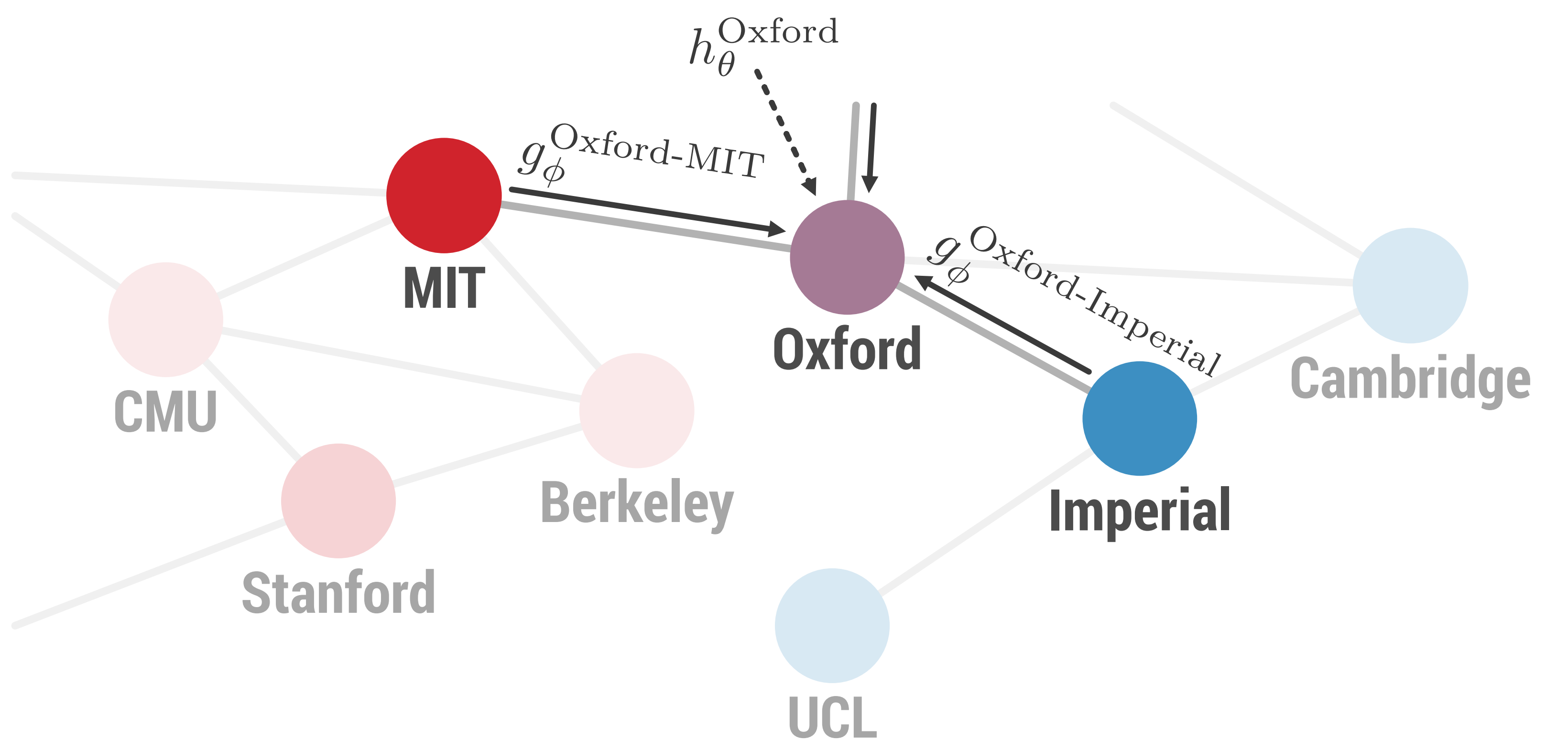
UPDATE MODELS

Update the model parameters using the current expected labels of the unlabeled nodes.

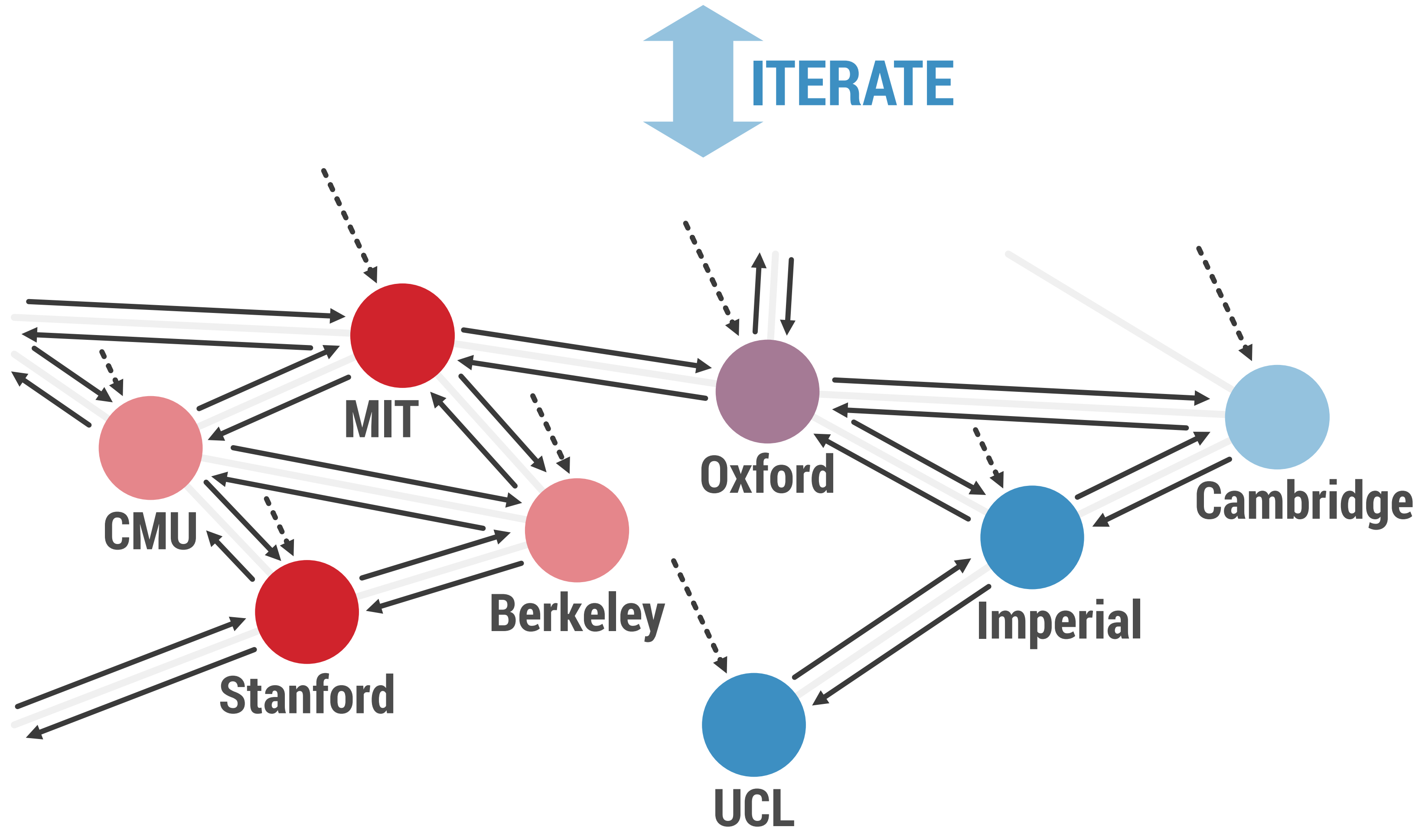
Self-Reflection



Self-Reflection over Graphs



UPDATE LABELS
Compute the expected labels of the unlabeled nodes using the current model parameters.



UPDATE MODELS
Update the model parameters using the current expected labels of the unlabeled nodes.

CPG: Controlled Parameter Sharing

Carnegie Mellon University

- The encoder/decoder parameters often have some ***"natural grouping"*** (e.g., the weight matrix of the first LSTM layer forms a group).
- The language embeddings need to represent all language-specific information and thus may need to be large.
- Only a small part of that information may be relevant for each "*group*".

CPG: Controlled Parameter Sharing

Carnegie Mellon University

- The encoder/decoder parameters often have some ***"natural grouping"*** (e.g., the weight matrix of the first LSTM layer forms a group).
- The language embeddings need to represent all language-specific information and thus may need to be large.
- Only a small part of that information may be relevant for each *"group"*.

We can use these observations to **control the amount of information sharing across languages.**

CPG: Controlled Parameter Sharing

Carnegie Mellon University

Let $\theta^{(enc)} = \{\theta_j^{(enc)}\}_{j=1}^{G}$ and $\theta_j^{(enc)} \in \mathbb{R}^{P_j^{(enc)}}$, where G is the number of groups.

CPG: Controlled Parameter Sharing

Carnegie Mellon University

Let $\theta^{(enc)} = \{\theta_j^{(enc)}\}_{j=1}^{G}$ and $\theta_j^{(enc)} \in \mathbb{R}^{P_j^{(enc)}}$, where G is the number of groups.

Then, we can define:

$$\theta_j^{(enc)} \triangleq \mathbf{W_j^{(enc)}} \mathbf{P_j^{(enc)}} \mathbf{l_s}$$

where:

$$\mathbf{W_j^{(enc)}} \in \mathbb{R}^{P_j^{(enc)} \times M'}$$

$$\mathbf{P_j^{(enc)}} \in \mathbb{R}^{M' \times M}$$

$$M' < M$$

and similarly for the decoder.

CPG: Controlled Parameter Sharing

Carnegie Mellon University

Let $\theta^{(enc)} = \{\theta_j^{(enc)}\}_{j=1}^{G}$ and $\theta_j^{(enc)} \in \mathbb{R}^{P_j^{(enc)}}$, where G is the number of groups.

Then, we can define:

$$\theta_j^{(enc)} \triangleq \mathbf{W_j^{(enc)}} \mathbf{P_j^{(enc)}} \mathbf{l_s}$$

where:

$$\mathbf{W_j^{(enc)}} \in \mathbb{R}^{P_j^{(enc)} \times M'}$$

$$\mathbf{P_j^{(enc)}} \in \mathbb{R}^{M' \times M}$$

$$M' < M$$

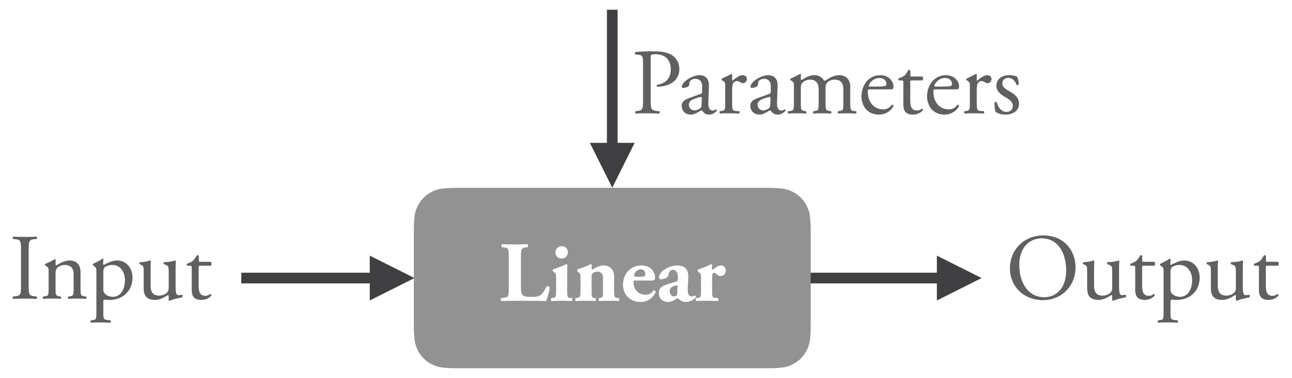
and similarly for the decoder.

If we want to increase the number of per-language parameters, we can increase **M**, while keeping **M'** fixed, and vice-versa.

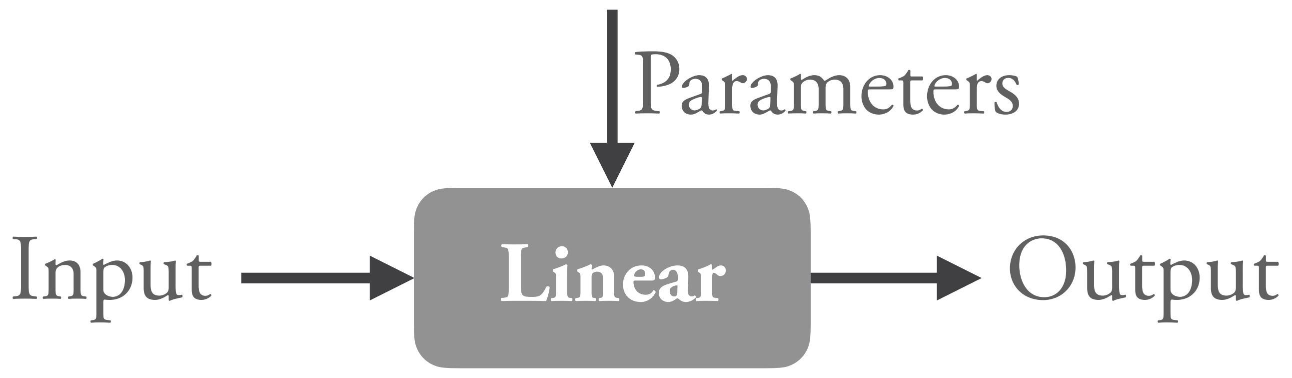
Multi-Task Learning

Why does contextual parameter generation work?

Why does contextual parameter generation work?

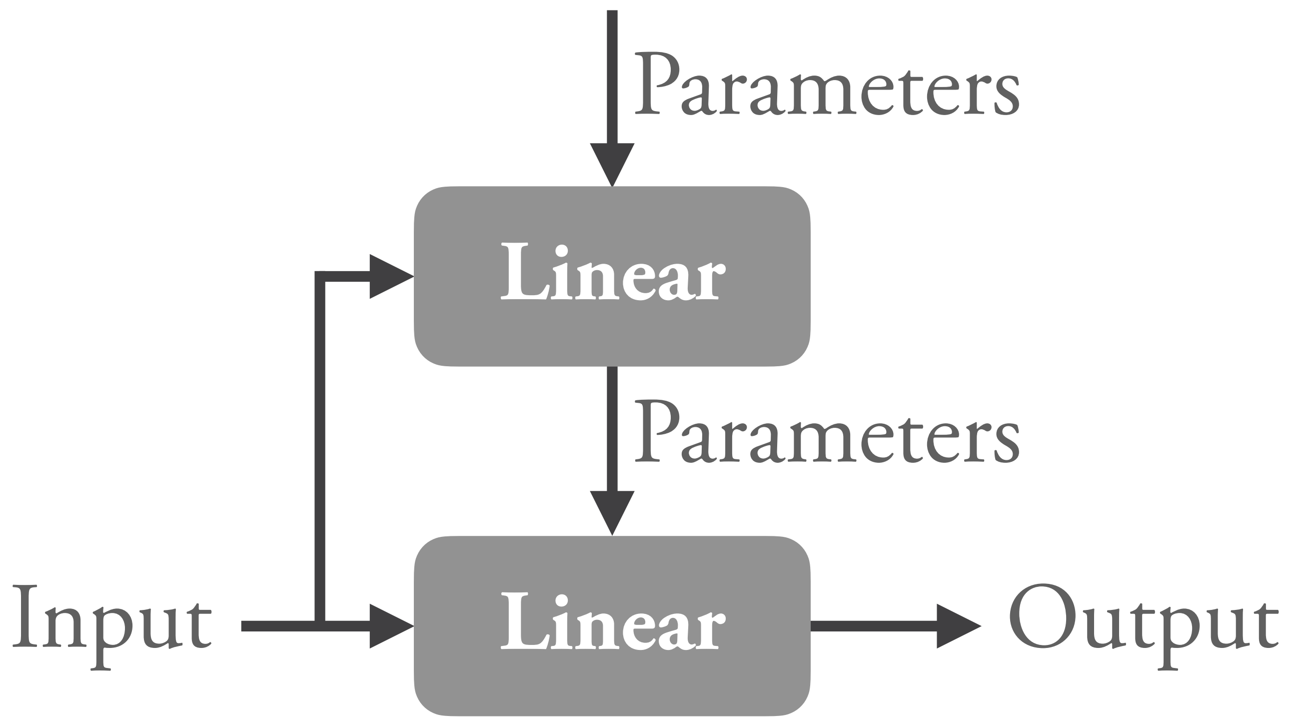


Why does contextual parameter generation work?



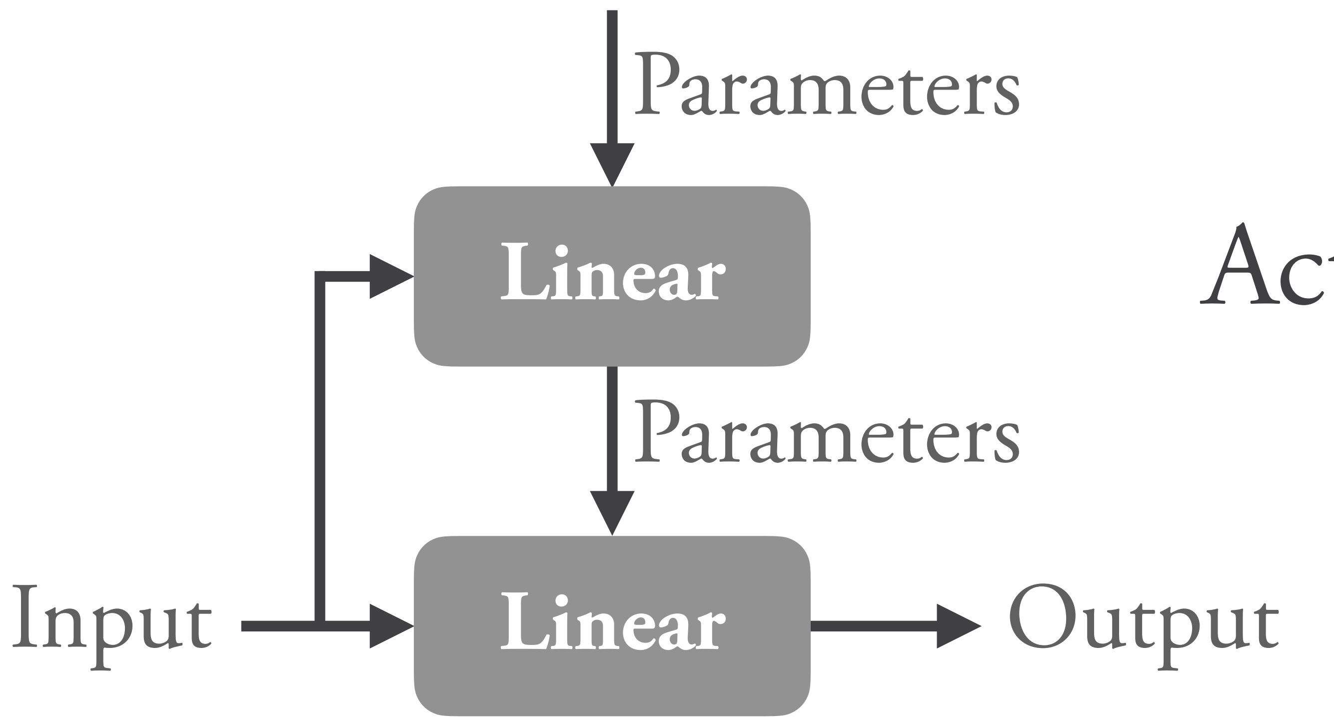
Cannot represent the XOR function!

Why does contextual parameter generation work?



Can represent the XOR function!

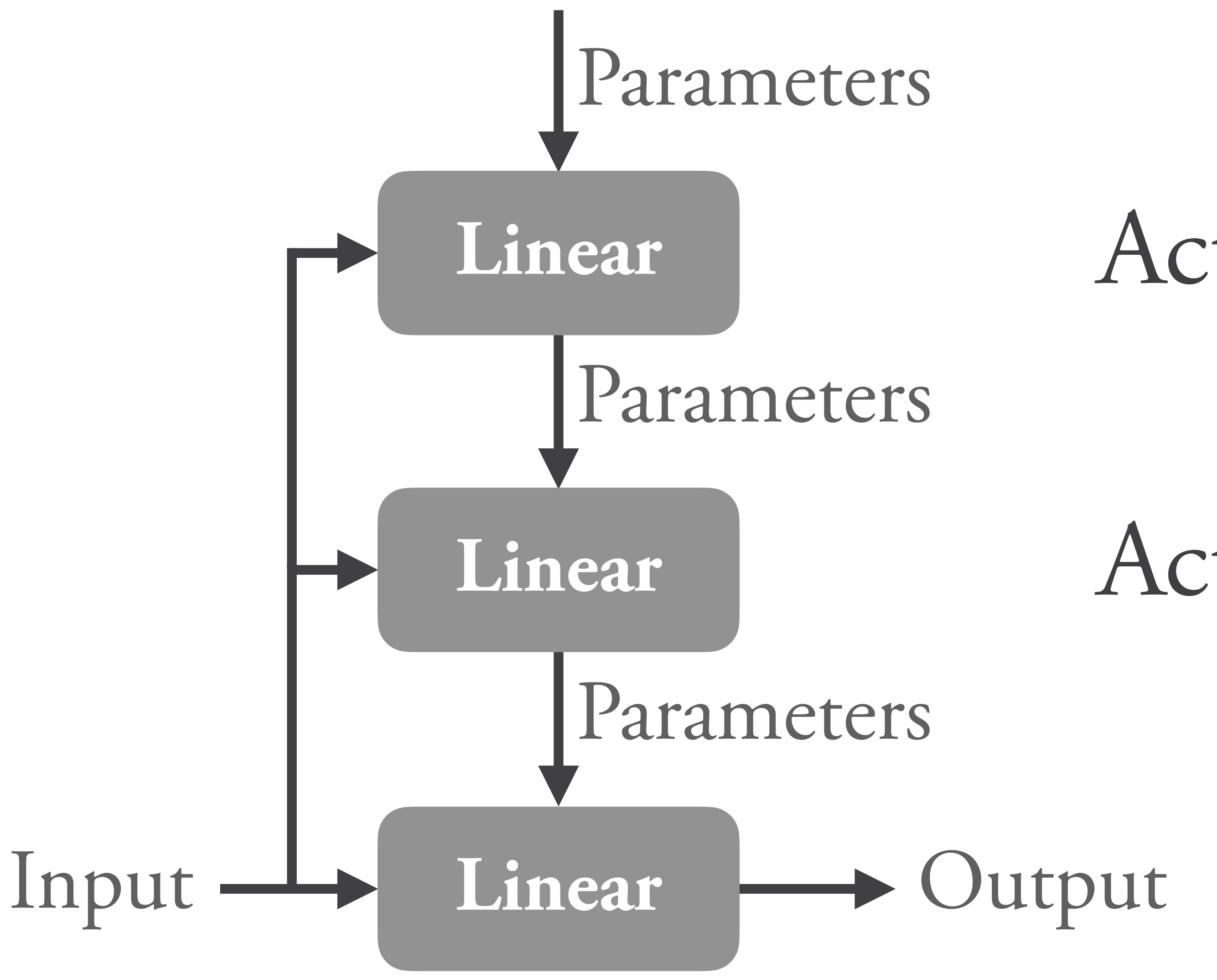
Why does contextual parameter generation work?



Acts a **quadratic kernel!**

Can represent the XOR function!

Why does contextual parameter generation work?



Acts a **cubic kernel!**

Acts a **quadratic kernel!**

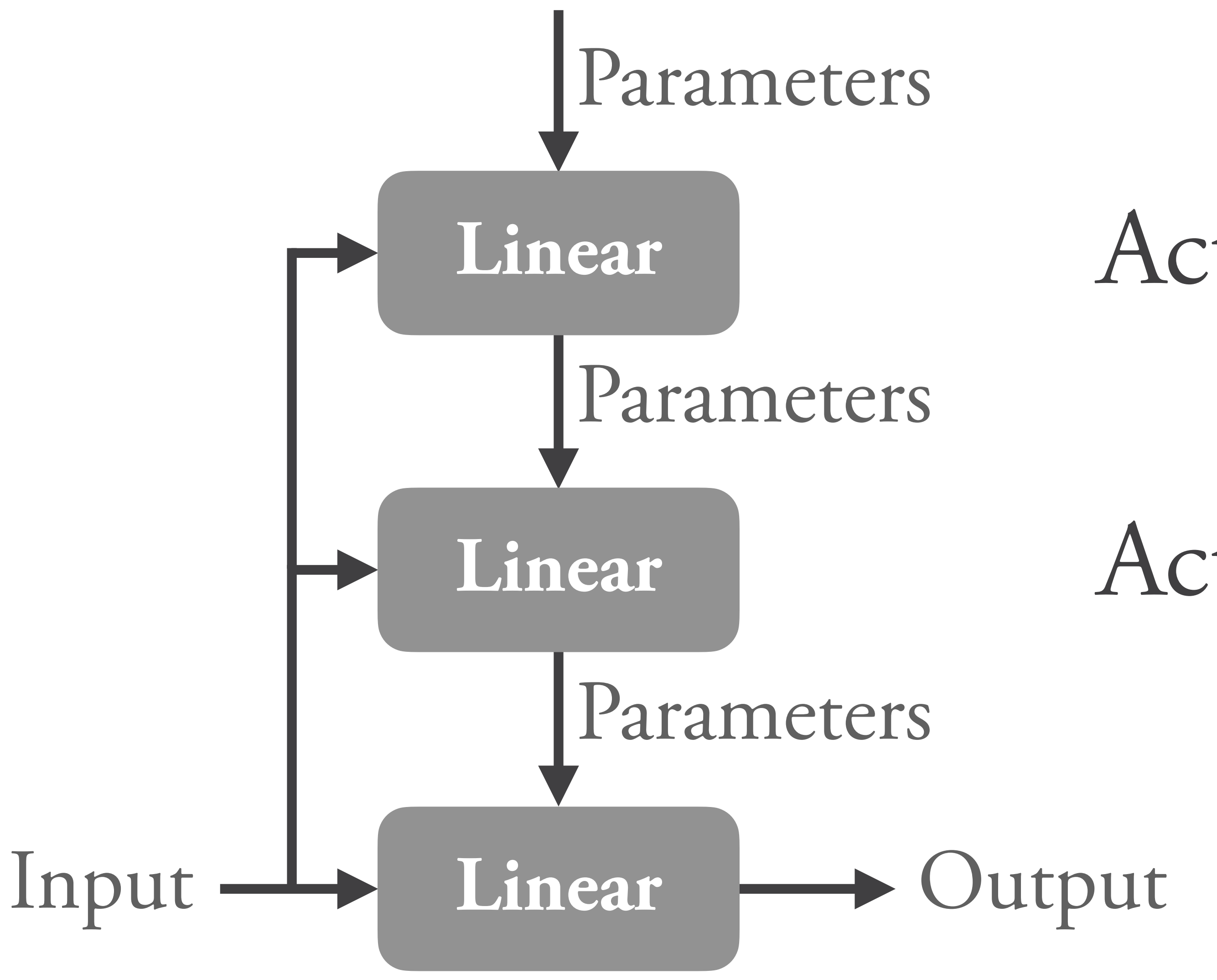
Can represent the XOR function!

Why does contextual parameter generation work?

$\mathcal{O}(p^{n+1})$

$\downarrow$

$\mathcal{O}(k^n p)$



Acts a **cubic kernel!**

Acts a **quadratic kernel!**

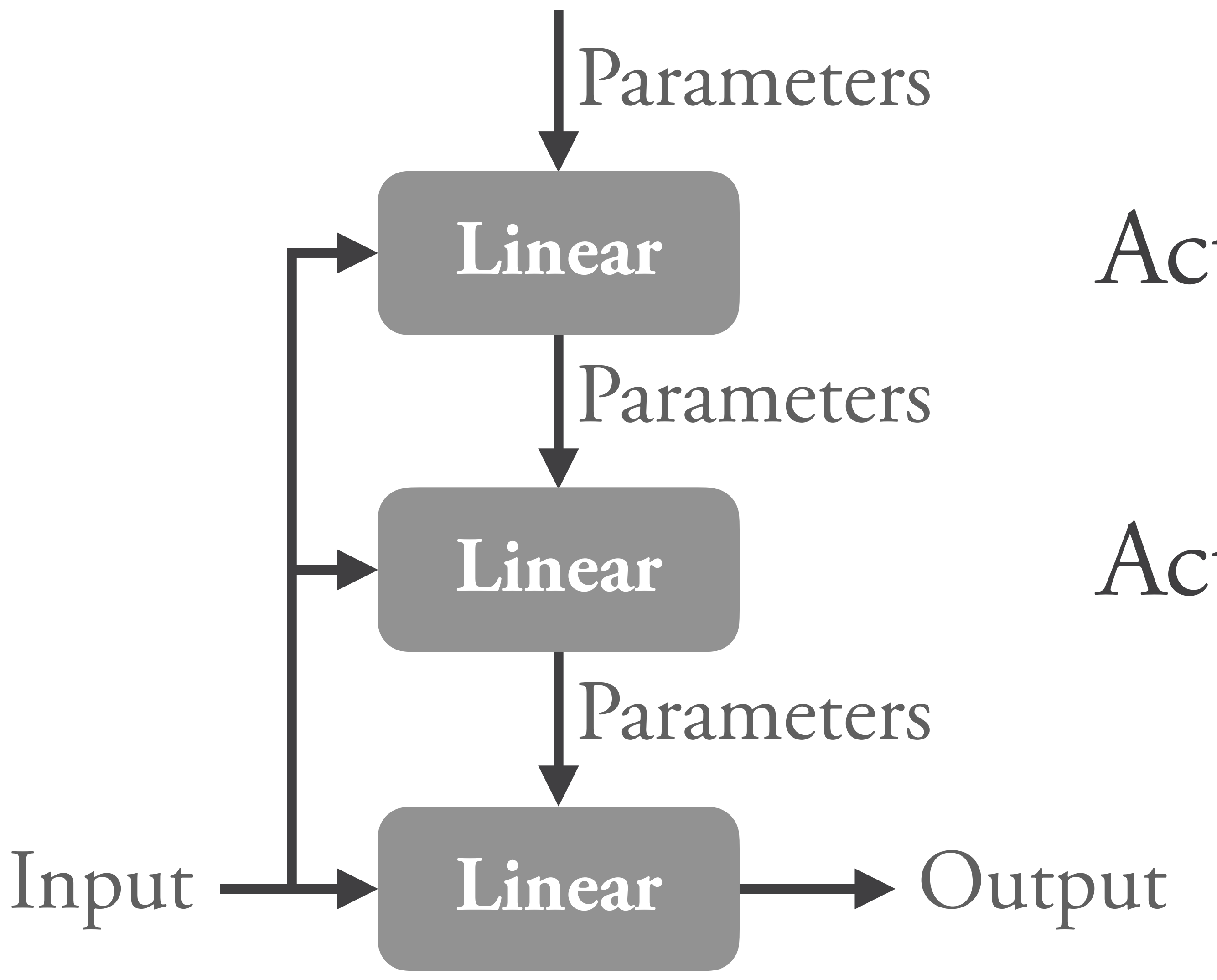
Can represent the XOR function!

Multi-Task Learning — Contextual Parameter Generation

Why does contextual parameter generation work?

$p = 1{,}024$
$n = 3$
$k = 8$

$$\mathcal{O}(p^{n+1}) \;\longrightarrow\; \mathcal{O}(k^n p)$$



Acts a **cubic kernel!**

Acts a **quadratic kernel!**

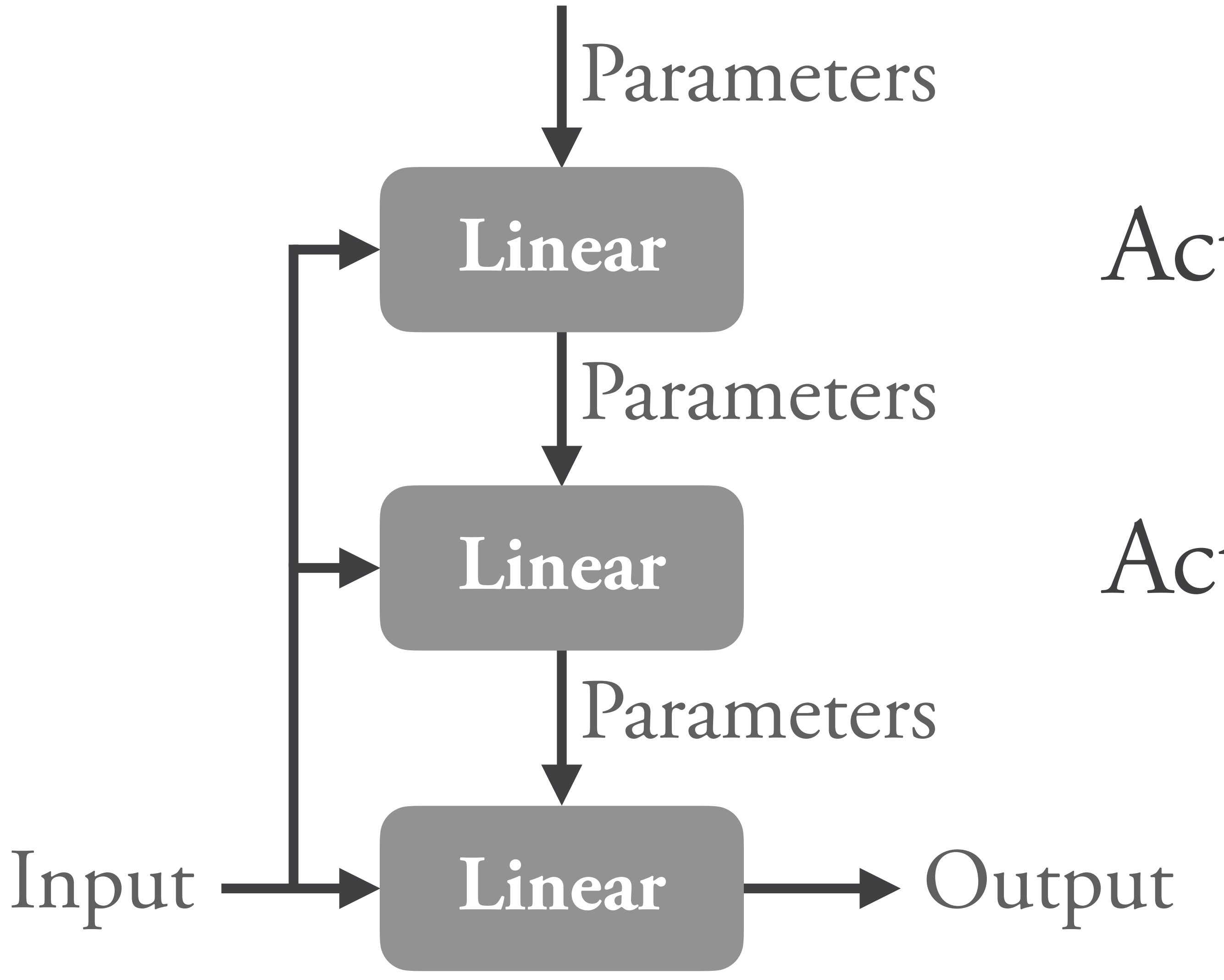
Can represent the XOR function!

Multi-Task Learning — Contextual Parameter Generation

Why does contextual parameter generation work?

$p = 1{,}024$
$n = 3$
$k = 8$

$\mathcal{O}(p^{n+1})$	1,000,000,000,000
↓	↓
$\mathcal{O}(k^n p)$	500,000



Acts a **cubic kernel!**

Acts a **quadratic kernel!**

Can represent the XOR function!

Contextual Parameter Generation

Why does contextual parameter generation work?

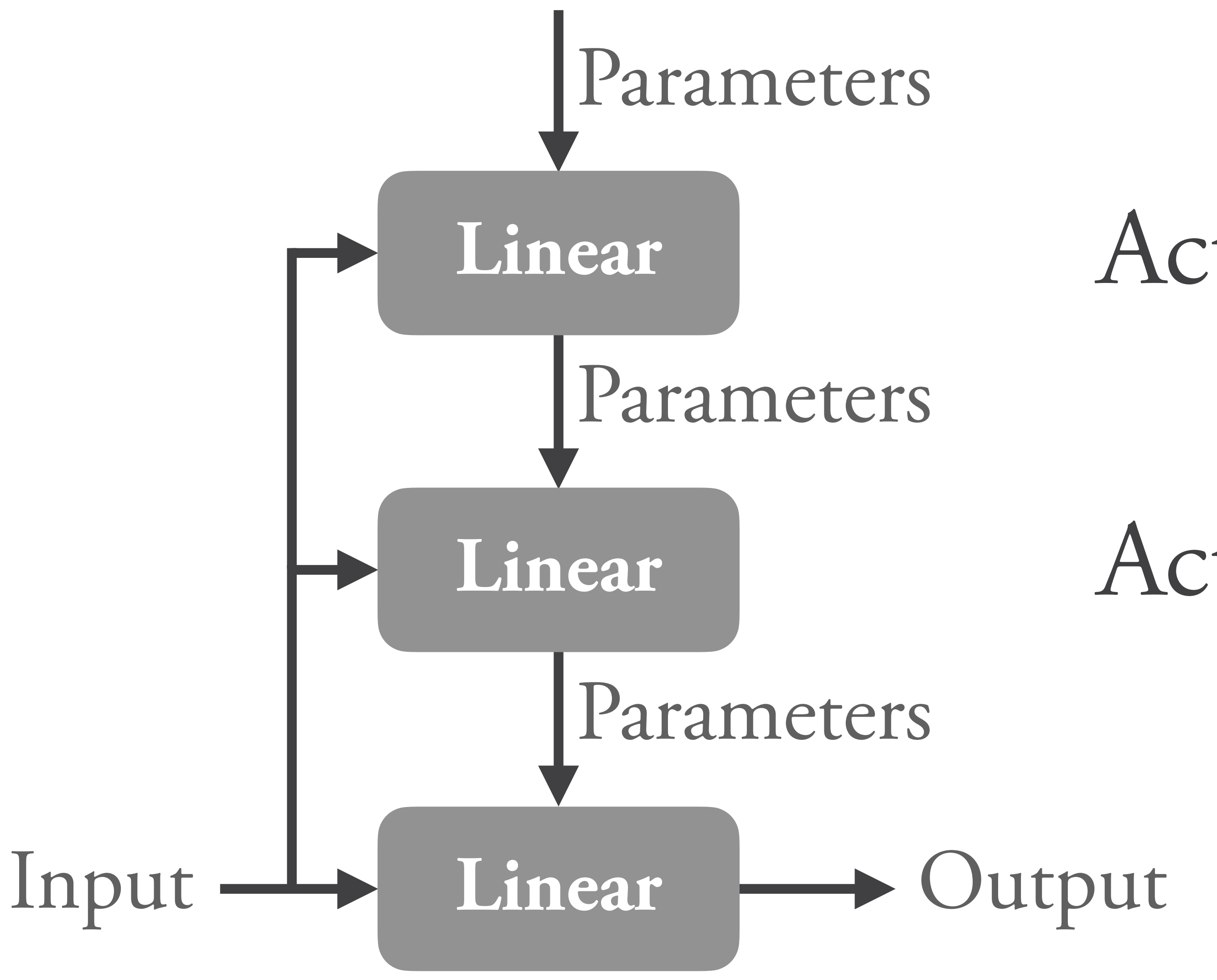
- Is it related to probabilistic graphical models?

$p = 1{,}024$
$n = 3$
$k = 8$

$\mathcal{O}(p^{n+1})$ 1,000,000,000,000

$\downarrow$ $\downarrow$

$\mathcal{O}(k^n p)$ 500,000



Acts a **cubic kernel!**

Acts a **quadratic kernel!**

Can represent the XOR function!

Multi-Task Learning

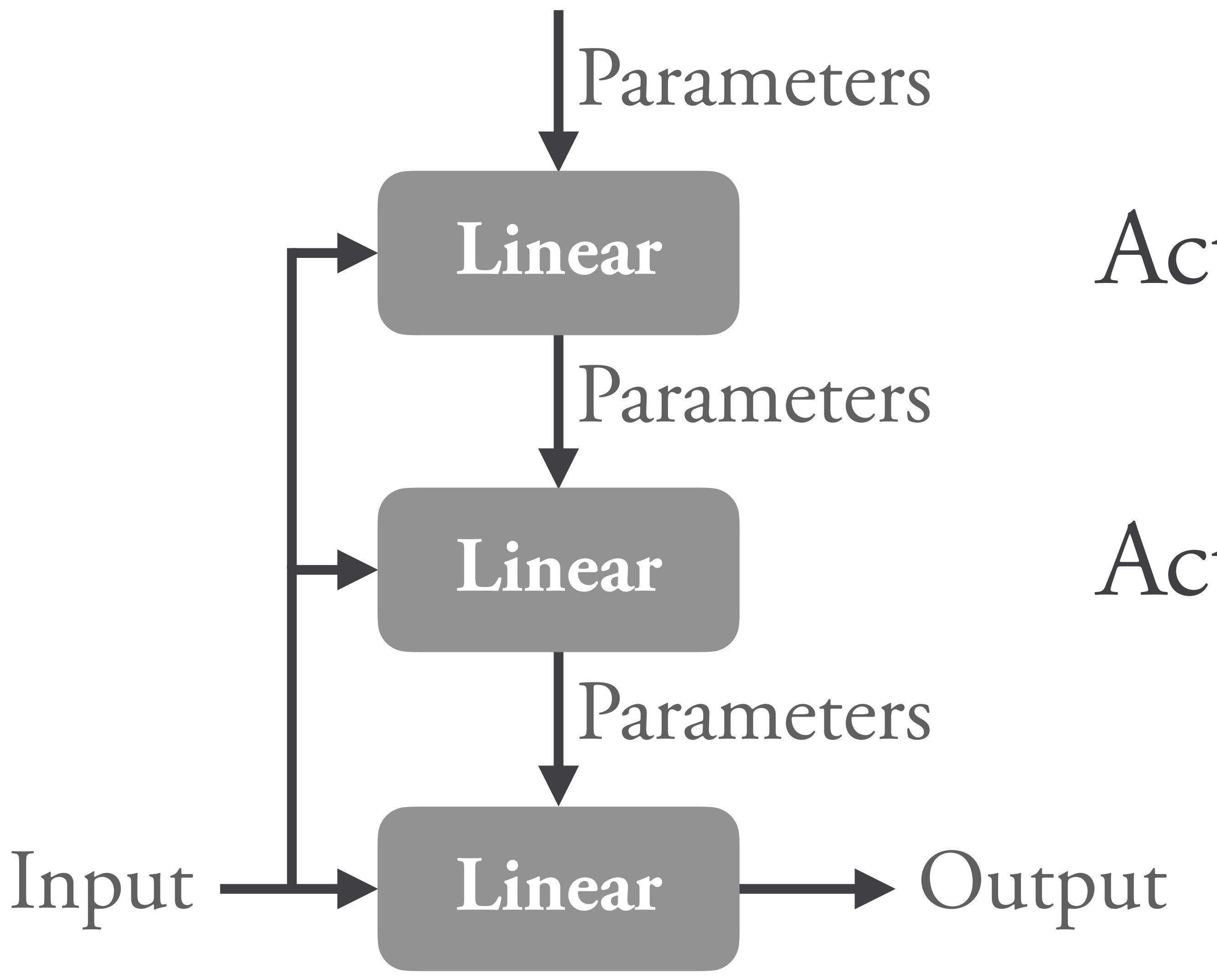
Contextual Parameter Generation

Why does contextual parameter generation work?

- Is it related to probabilistic graphical models?
- How does it increase the expressive power of neural networks?

$$p = 1{,}024$$
$$n = 3$$
$$k = 8$$

$\mathcal{O}(p^{n+1})$	1,000,000,000,000
$\downarrow$	$\downarrow$
$\mathcal{O}(k^n p)$	500,000



Acts a **cubic kernel!**

Acts a **quadratic kernel!**

Can represent the XOR function!

A Testbed for Never-Ending Learning

Jelly Bean World

A Testbed for Never-Ending Learning

non-episodic

no environment resets

A Testbed for Never-Ending Learning

non-stationary

reward function can change over time

non-episodic

no environment resets

A Testbed for Never-Ending Learning

non-stationary

reward function can change over time

multi-task

support multiple learning tasks

non-episodic

no environment resets

A Testbed for Never-Ending Learning

non-stationary

reward function can
change over time

multi-task

support multiple
learning tasks

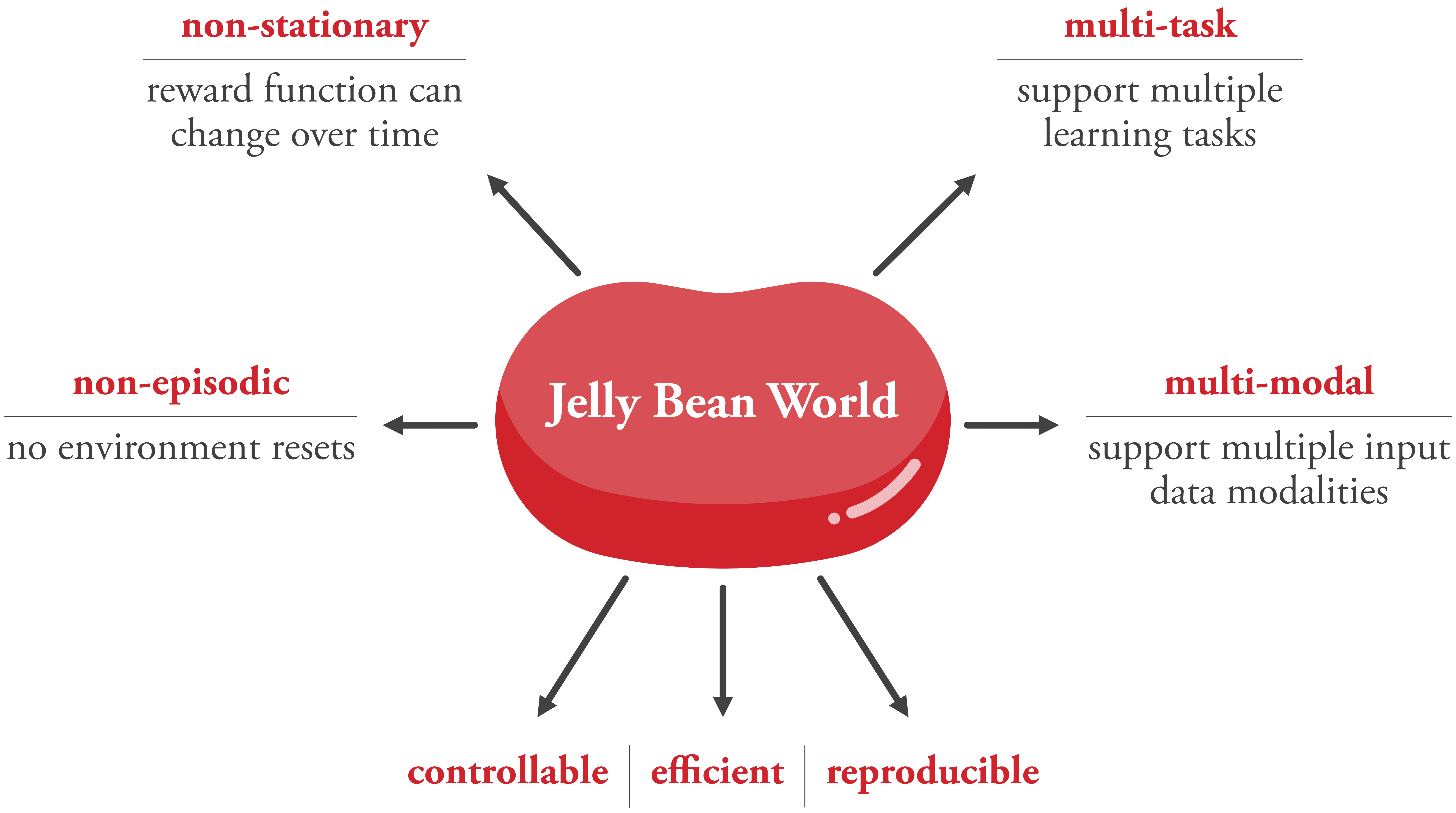
non-episodic

no environment resets

multi-modal

support multiple input
data modalities

A Testbed for Never-Ending Learning



A Testbed for Never-Ending Learning

non-stationary

reward function can change over time

multi-task

support multiple learning tasks

non-episodic

no environment resets

multi-modal

support multiple input data modalities

controllable | **efficient** | **reproducible**

[Bellemare et al., 2013]
[Pfau et al., 2018]

[Duan et al., 2016]
[Todorov et al., 2012]

[Chevalier-Boisvert et al., 2018]

[Silver et al., 2017]
[Vinyals et al., 2019]

[Johnson et al., 2016]
[Guss et al., 2019]

A Testbed for Never-Ending Learning

non-stationary

reward function can change over time

multi-task

support multiple learning tasks

non-episodic

no environment resets

multi-modal

support multiple input data modalities

controllable | **efficient** | **reproducible**

[Bellemare et al., 2013]
[Pfau et al., 2018]
[Duan et al., 2016]
[Todorov et al., 2012]
[Chevalier-Boisvert et al., 2018]
[Silver et al., 2017]
[Vinyals et al., 2019]
[Johnson et al., 2016]
[Guss et al., 2019]

Simulator

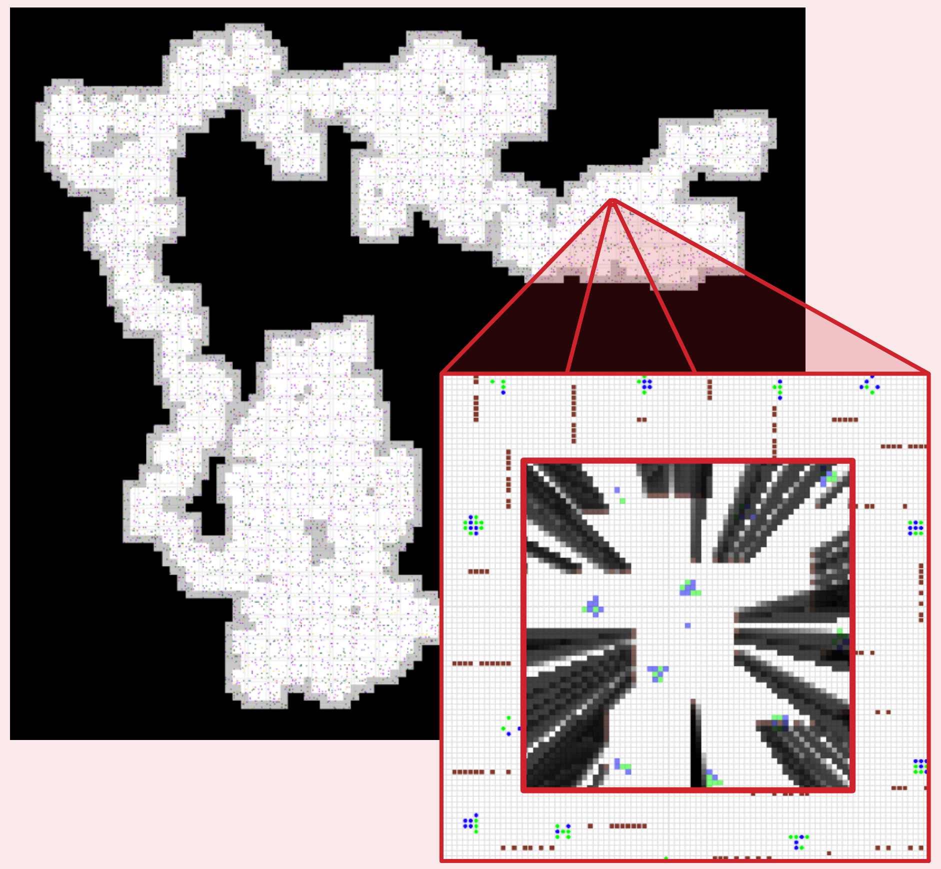
Advances time after all agents have acted, invoking modules as needed.

Simulator

Advances time after all agents have acted, invoking modules as needed.

MAP

Manages an infinite world map.

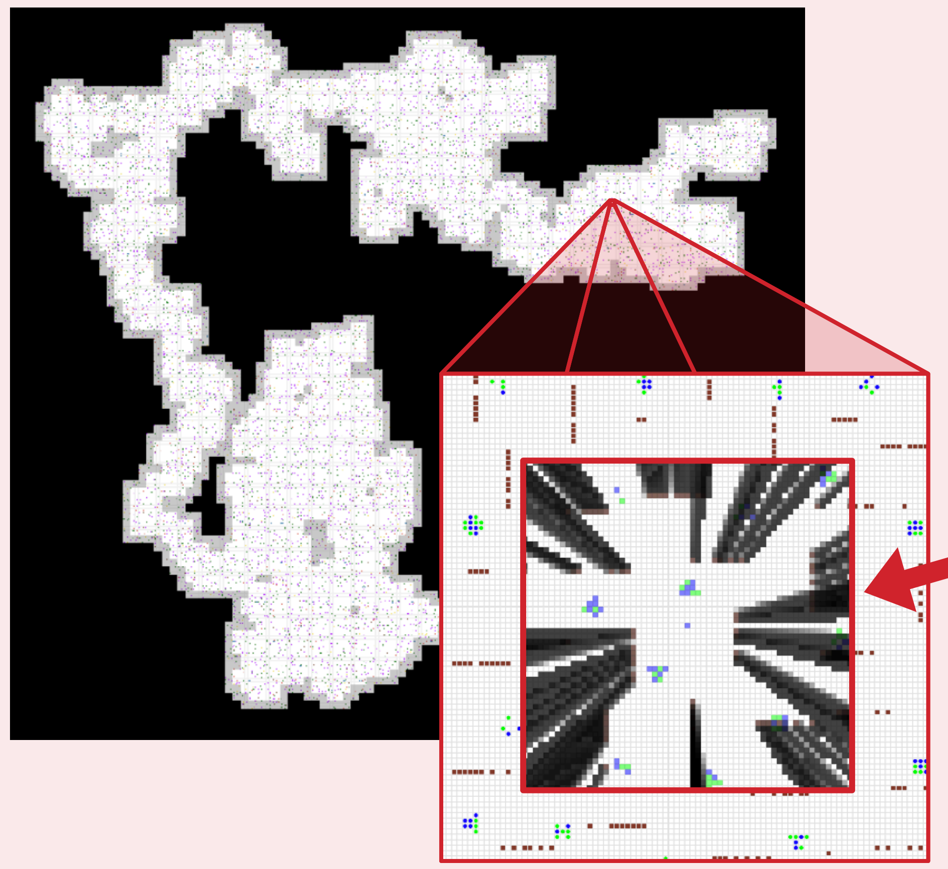


Simulator

Advances time after all agents have acted, invoking modules as needed.

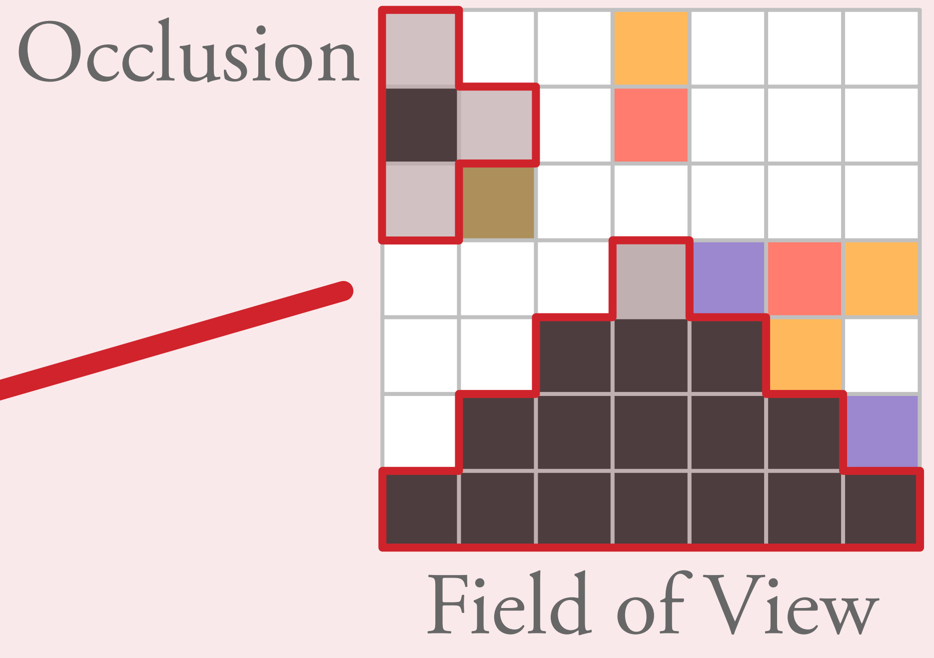
MAP

Manages an infinite world map.



VISION

Simulates the visual field of all managed agents.



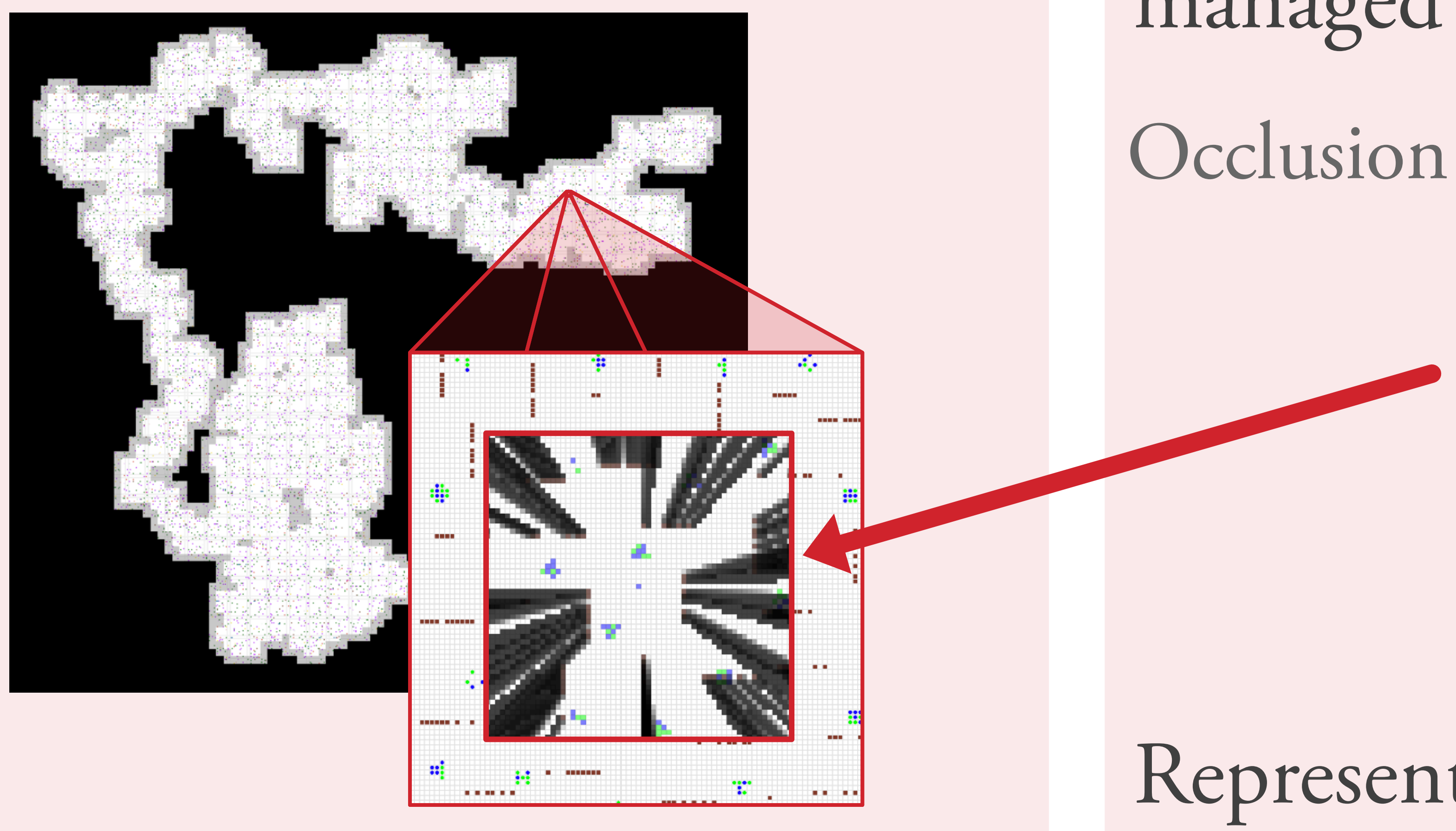
Represented as a 3D tensor.

Simulator

Advances time after all agents have acted, invoking modules as needed.

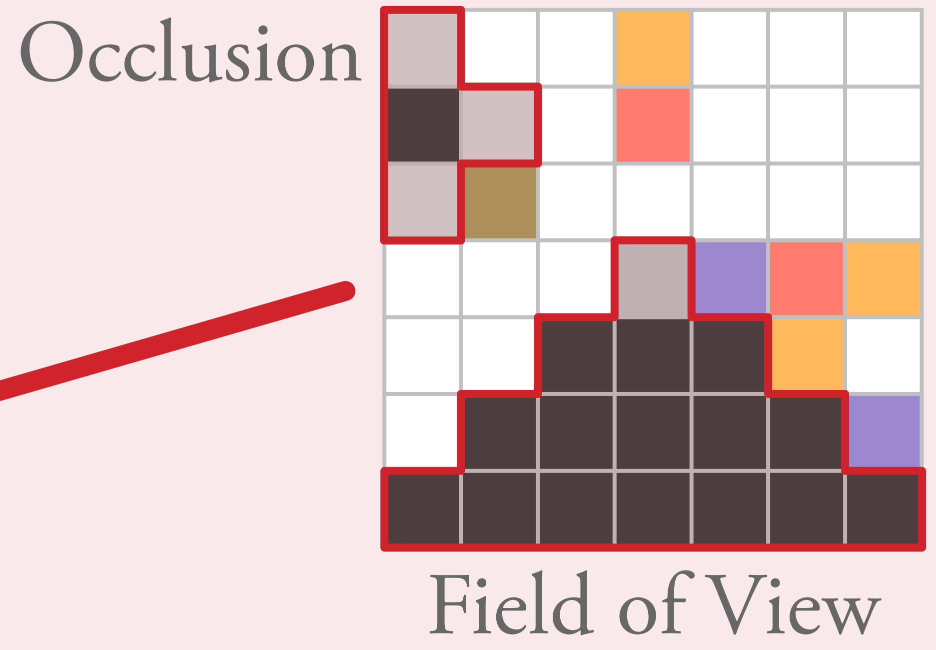
MAP

Manages an infinite world map.



VISION

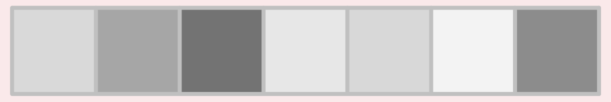
Simulates the visual field of all managed agents.



Represented as a 3D tensor.

SCENT

Simulates the diffusion of scent.



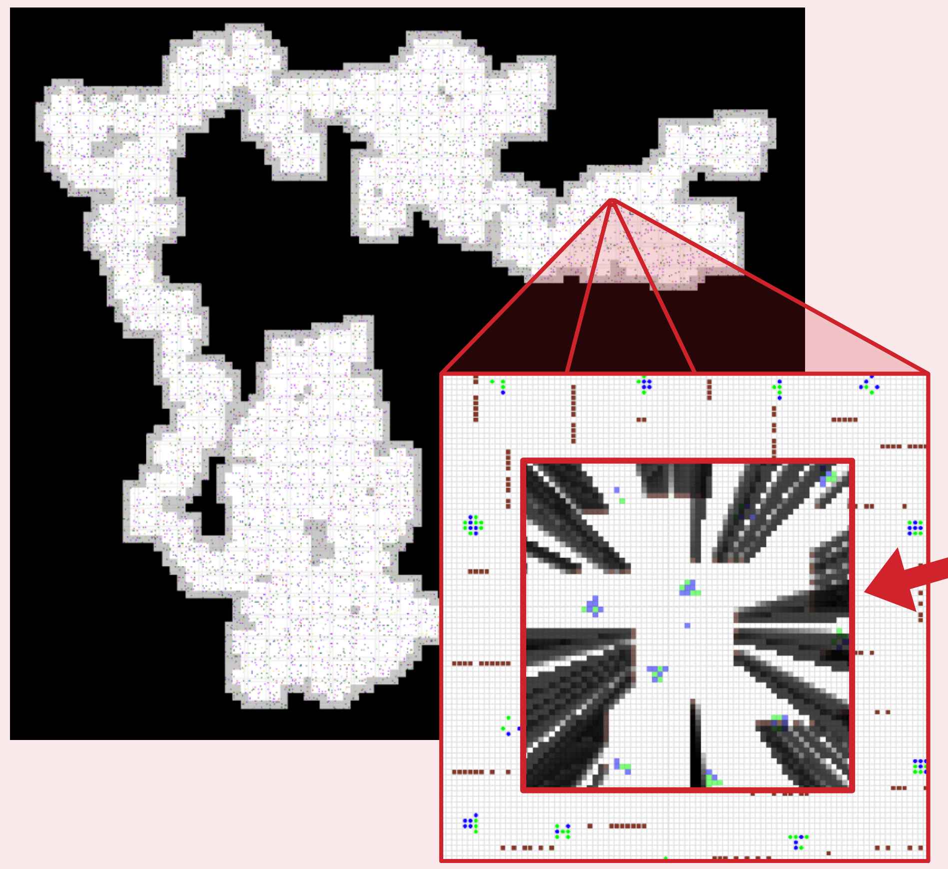
Represented as a vector.

Simulator

Advances time after all agents have acted, invoking modules as needed.

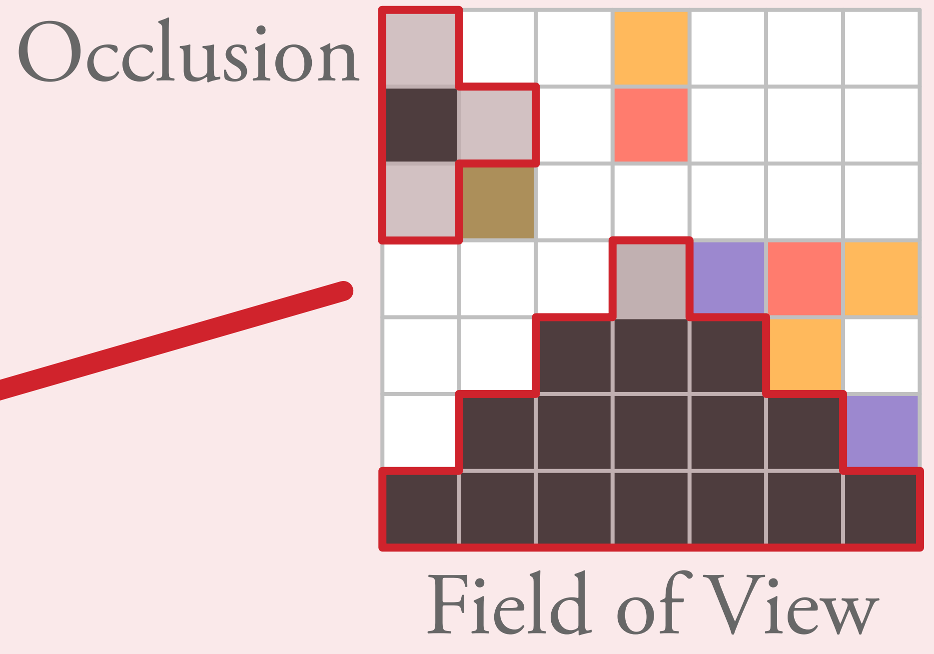
MAP

Manages an infinite world map.



VISION

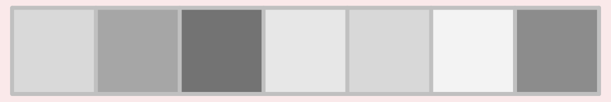
Simulates the visual field of all managed agents.



Represented as a 3D tensor.

SCENT

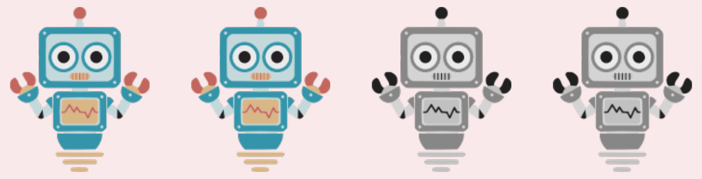
Simulates the diffusion of scent.



Represented as a vector.

AGENTS

Manages agents and handles their interaction with the map.

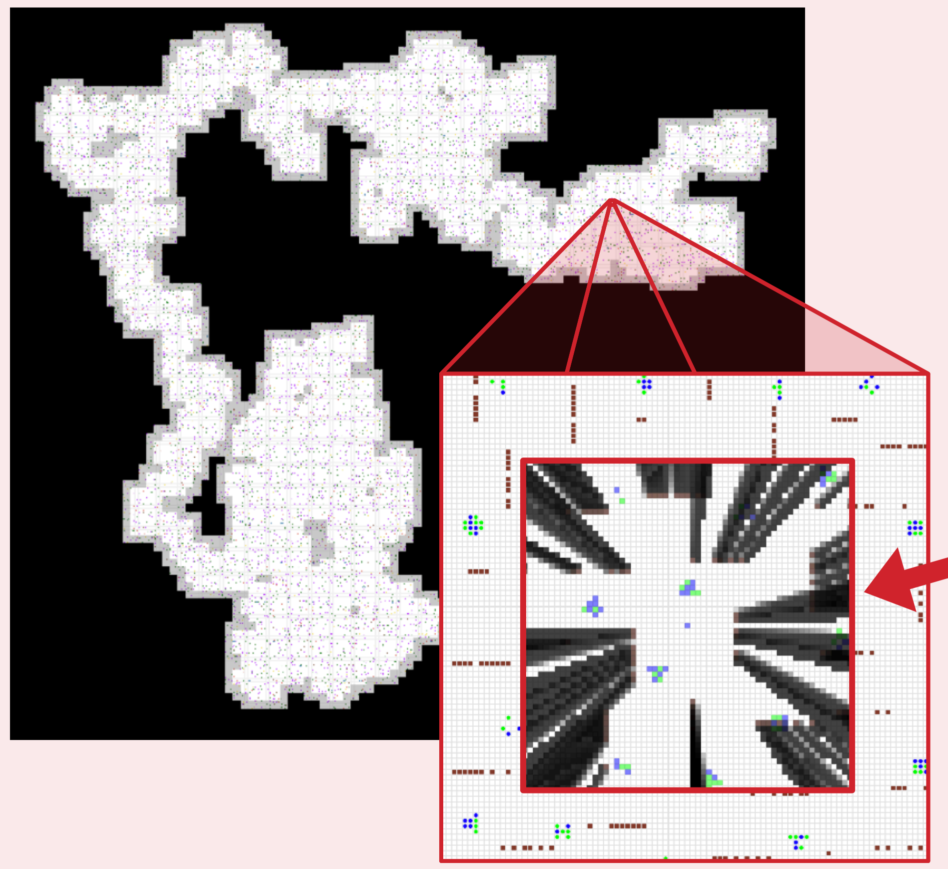


Simulator

Advances time after all agents have acted, invoking modules as needed.

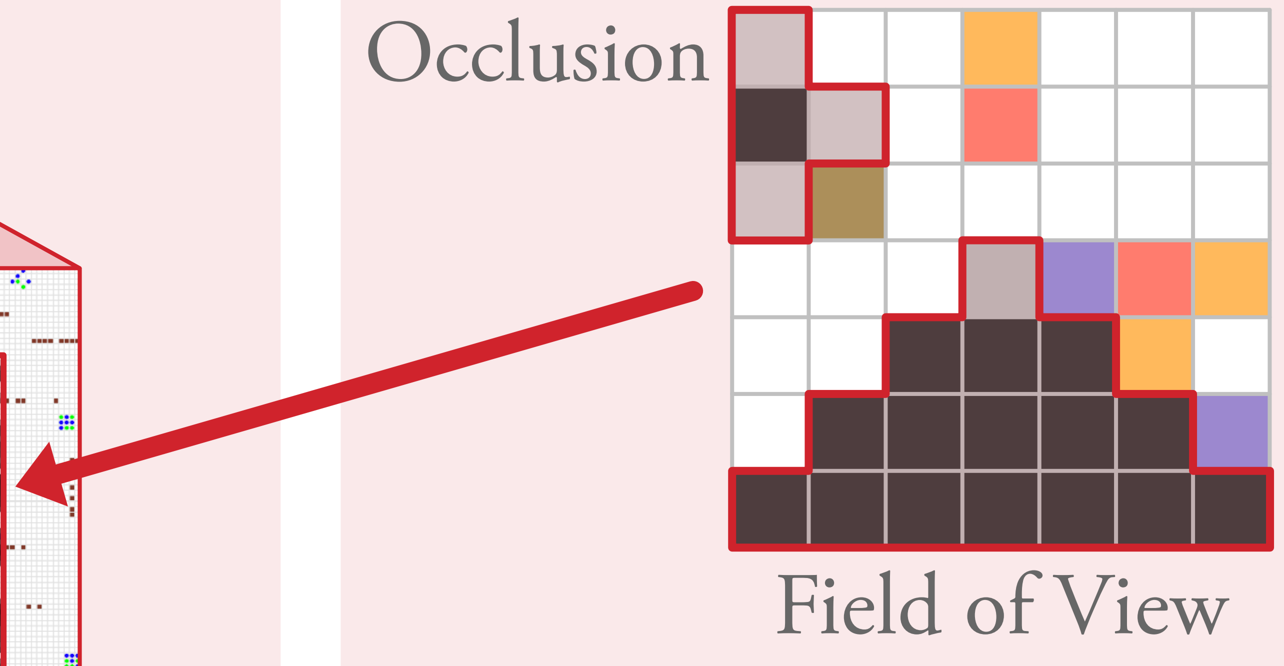
MAP

Manages an infinite world map.



VISION

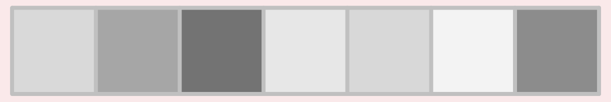
Simulates the visual field of all managed agents.



Represented as a 3D tensor.

SCENT

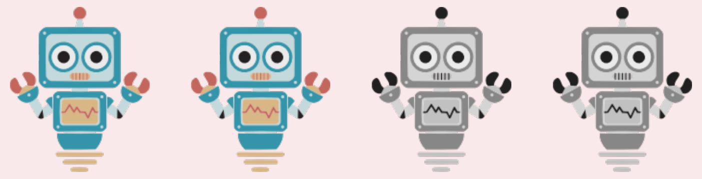
Simulates the diffusion of scent.



Represented as a vector.

AGENTS

Manages agents and handles their interaction with the map.



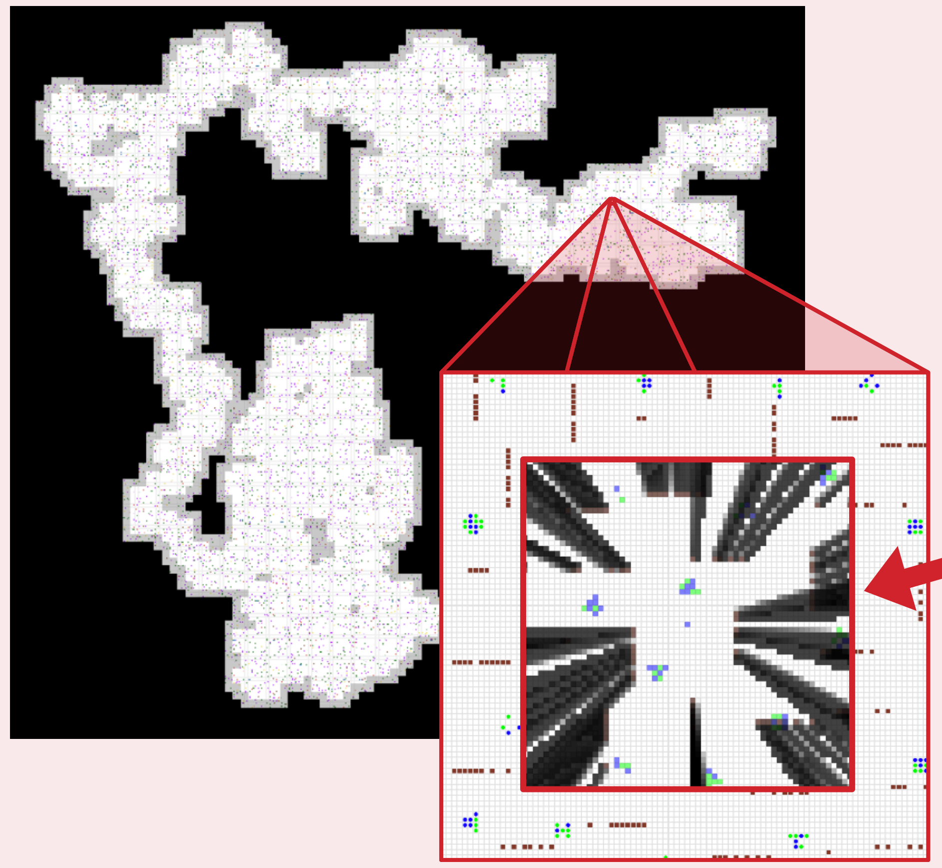
Distributed simulations are also supported using MPI.

Simulator

Advances time after all agents have acted, invoking modules as needed.

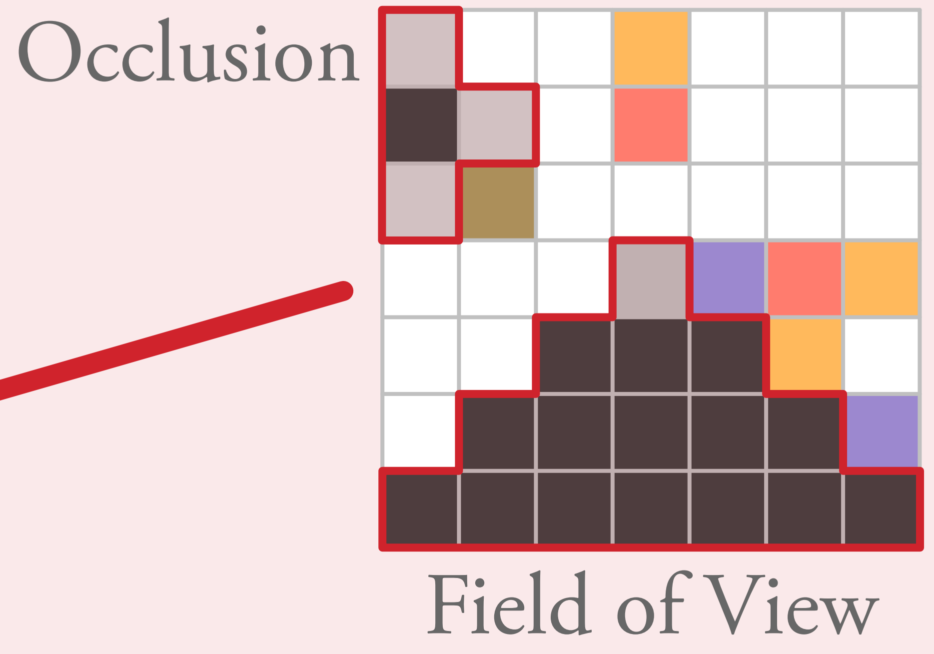
MAP

Manages an infinite world map.



VISION

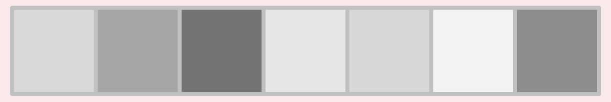
Simulates the visual field of all managed agents.



Represented as a 3D tensor.

SCENT

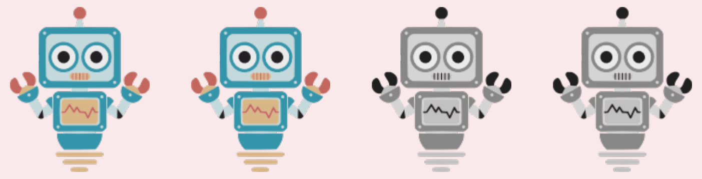
Simulates the diffusion of scent.



Represented as a vector.

AGENTS

Manages agents and handles their interaction with the map.



Distributed simulations are also supported using MPI.

Environment

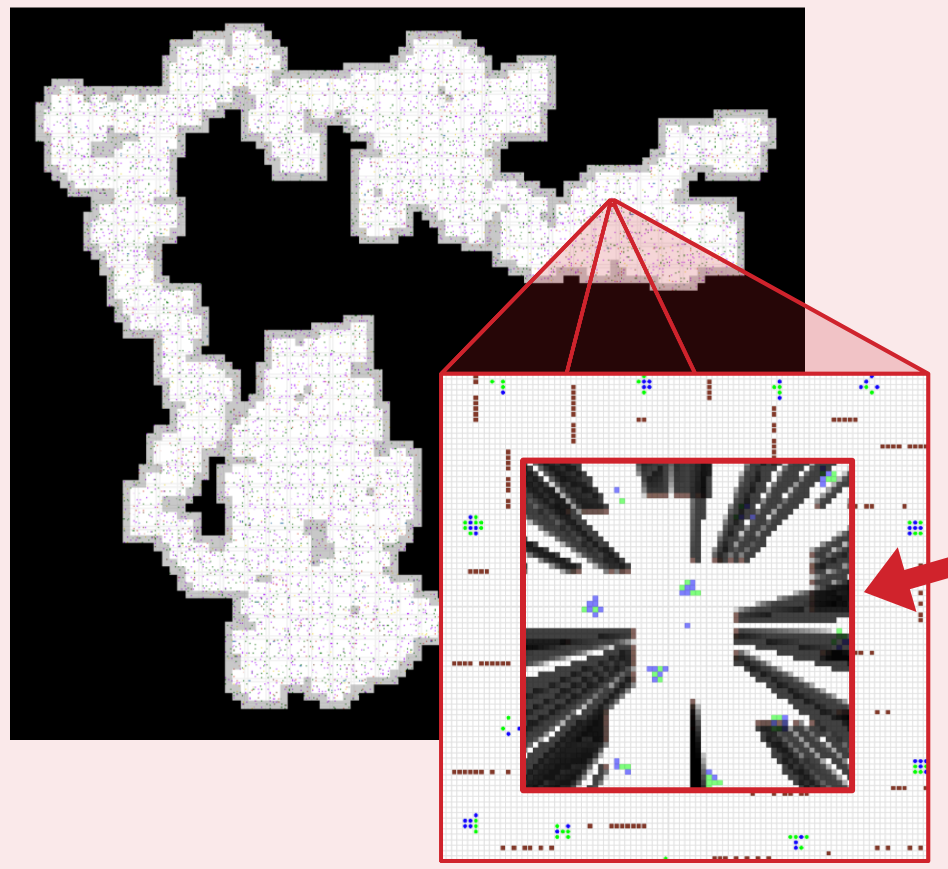
Provides an interface for reinforcement learning.

Simulator

Advances time after all agents have acted, invoking modules as needed.

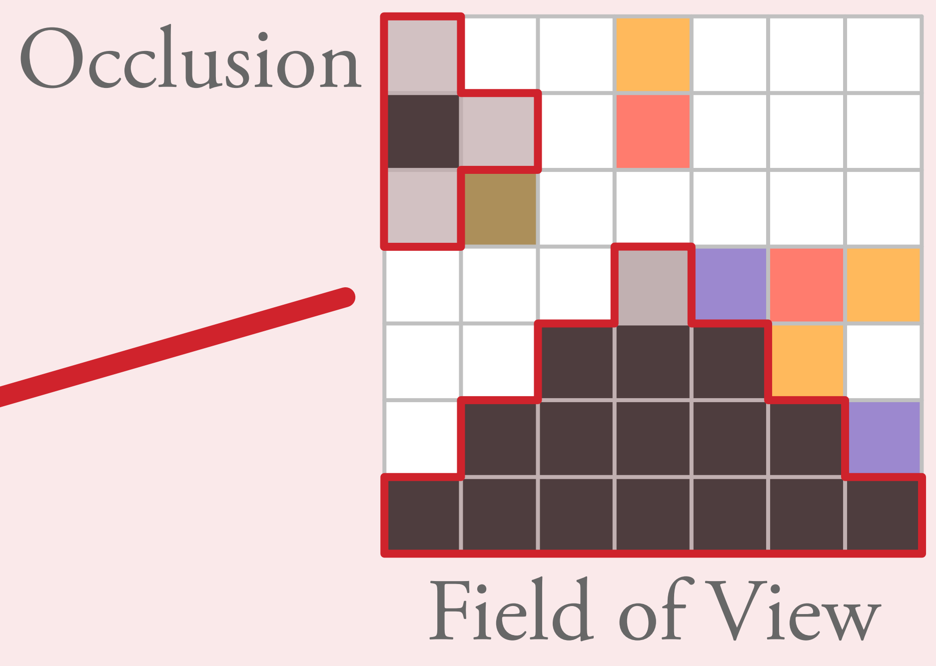
MAP

Manages an infinite world map.



VISION

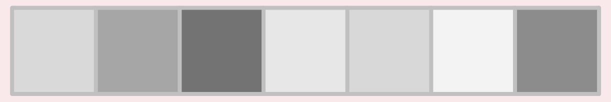
Simulates the visual field of all managed agents.



Represented as a 3D tensor.

SCENT

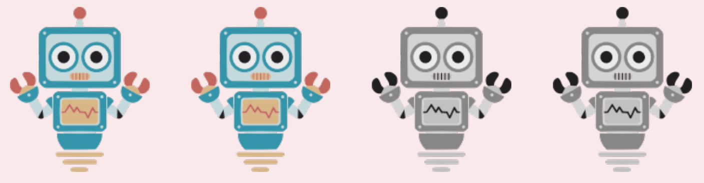
Simulates the diffusion of scent.



Represented as a vector.

AGENTS

Manages agents and handles their interaction with the map.



Distributed simulations are also supported using MPI.

Environment

Provides an interface for reinforcement learning.

REWARD FUNCTION

Specifies the reward given to the agent for each possible state transition.

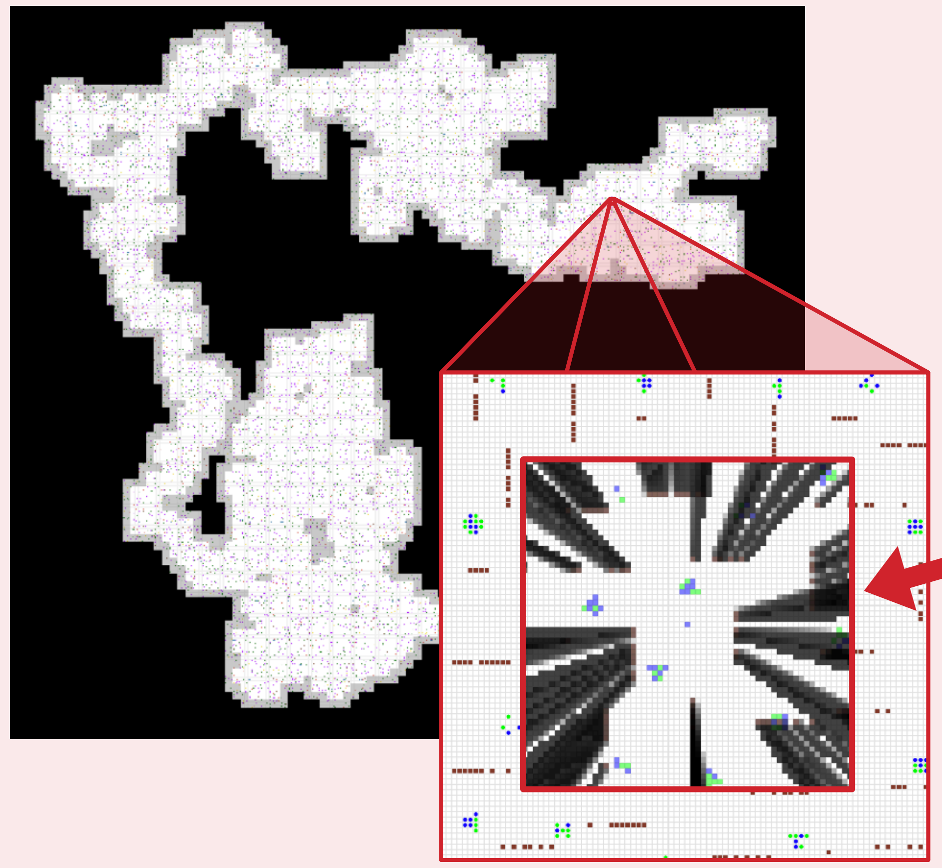
$$\texttt{Collect}[\textbf{JellyBean}] \wedge \texttt{Avoid}[\textbf{Onion}]$$

Simulator

Advances time after all agents have acted, invoking modules as needed.

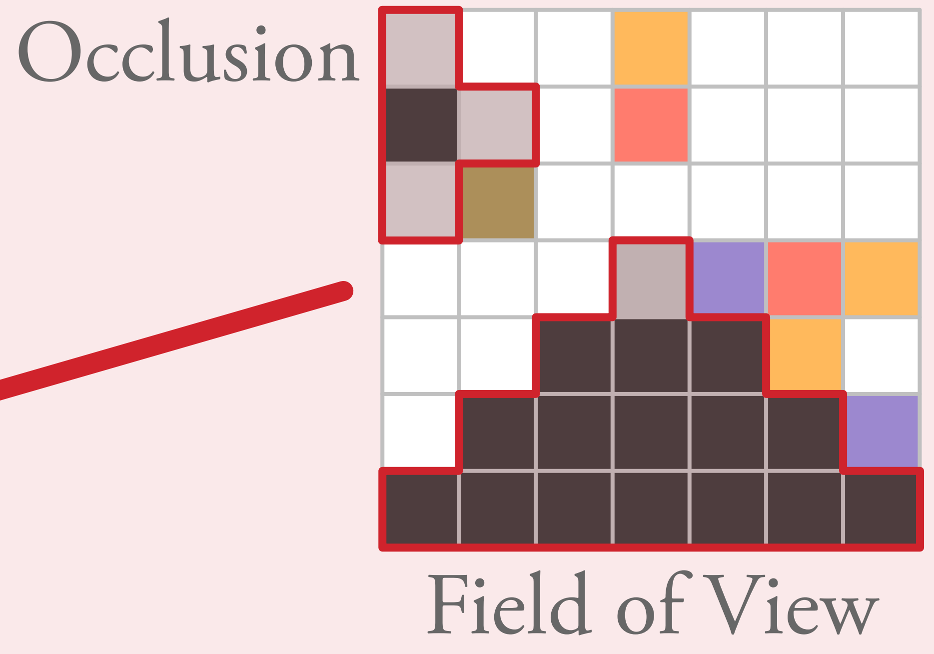
MAP

Manages an infinite world map.



VISION

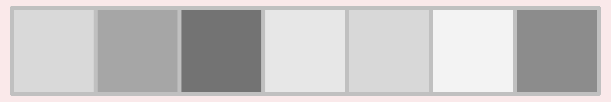
Simulates the visual field of all managed agents.



Represented as a 3D tensor.

SCENT

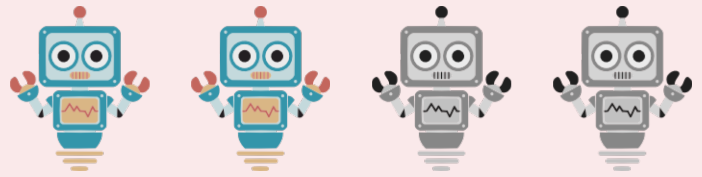
Simulates the diffusion of scent.



Represented as a vector.

AGENTS

Manages agents and handles their interaction with the map.



Distributed simulations are also supported using MPI.

Environment

Provides an interface for reinforcement learning.

REWARD FUNCTION

Specifies the reward given to the agent for each possible state transition.

Collect[**JellyBean**] ∧ Avoid[**Onion**]

REWARD SCHEDULE

Specifies the reward function for each time step.

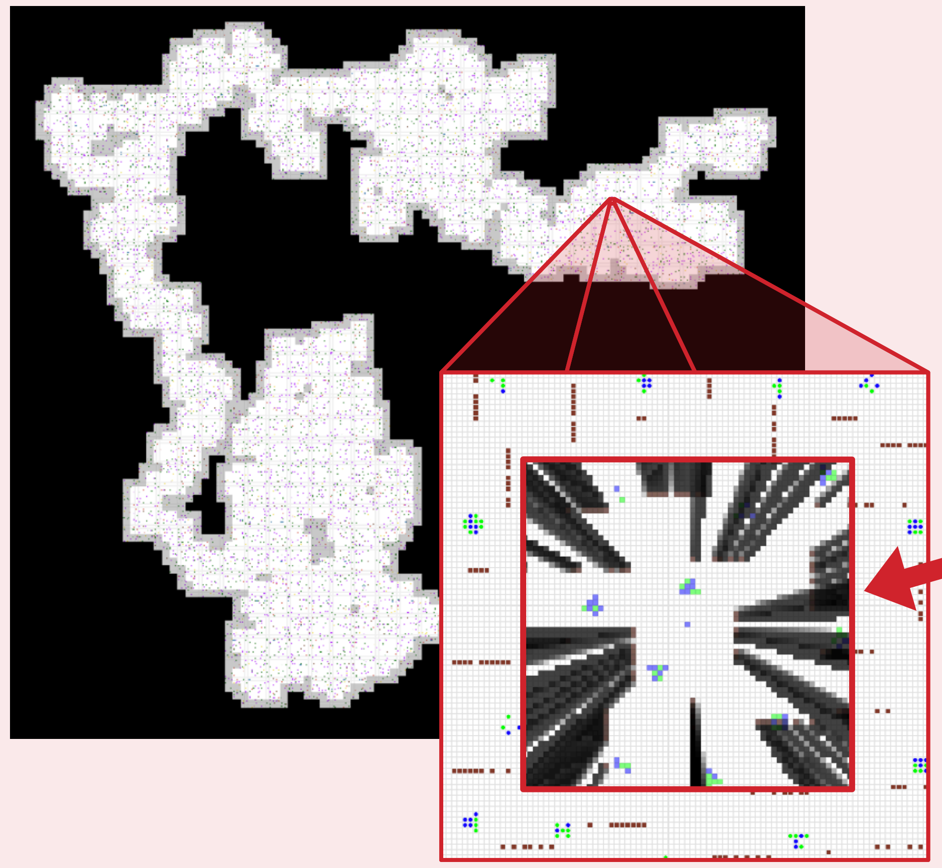
Fixed / Periodic / Random

Simulator

Advances time after all agents have acted, invoking modules as needed.

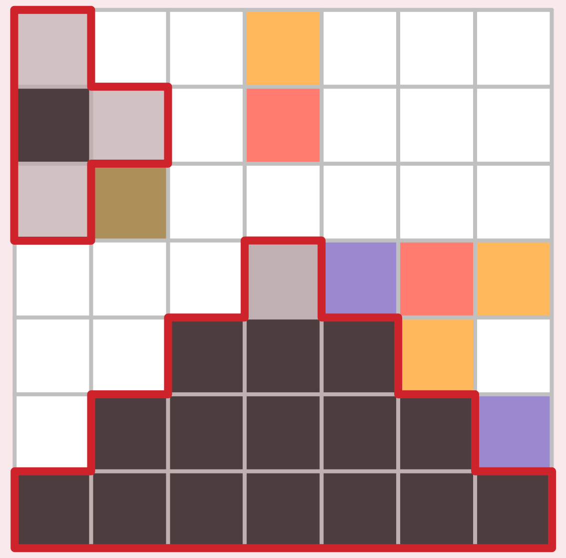
MAP

Manages an infinite world map.



VISION

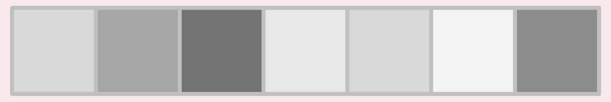
Simulates the visual field of all managed agents.



Represented as a 3D tensor.

SCENT

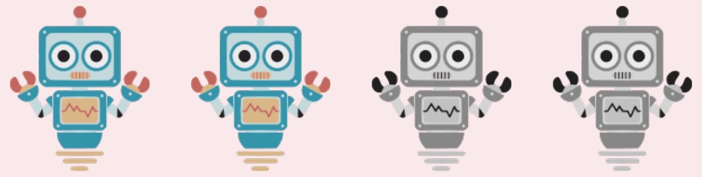
Simulates the diffusion of scent.



Represented as a vector.

AGENTS

Manages agents and handles their interaction with the map.



Distributed simulations are also supported using MPI.

Environment

Provides an interface for reinforcement learning.

REWARD FUNCTION

Specifies the reward given to the agent for each possible state transition.

Collect[**JellyBean**] $\wedge$ Avoid[**Onion**]

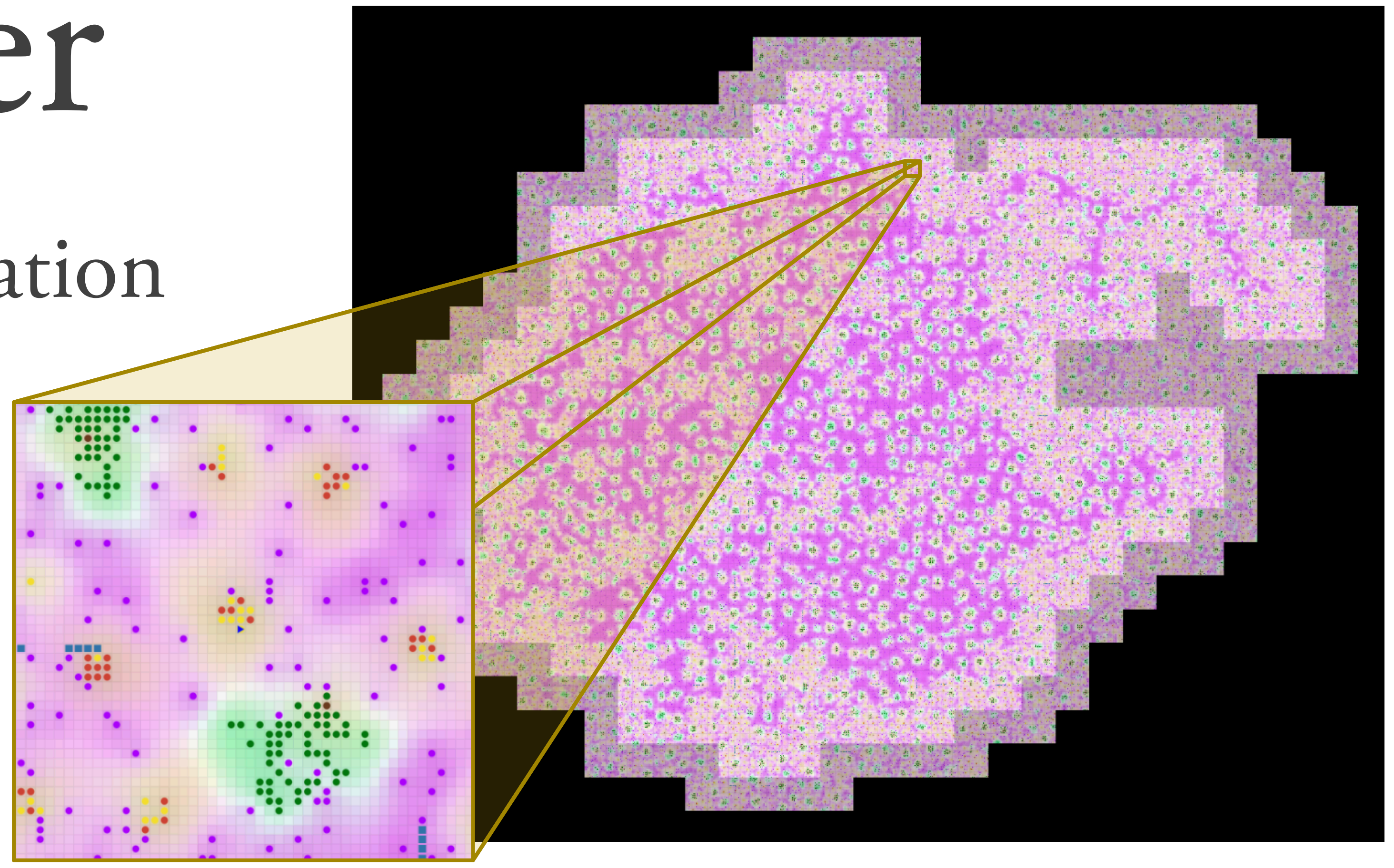
REWARD SCHEDULE

Specifies the reward function for each time step.

Fixed / Periodic / Random

Visualizer

Asynchronous simulation visualizer.



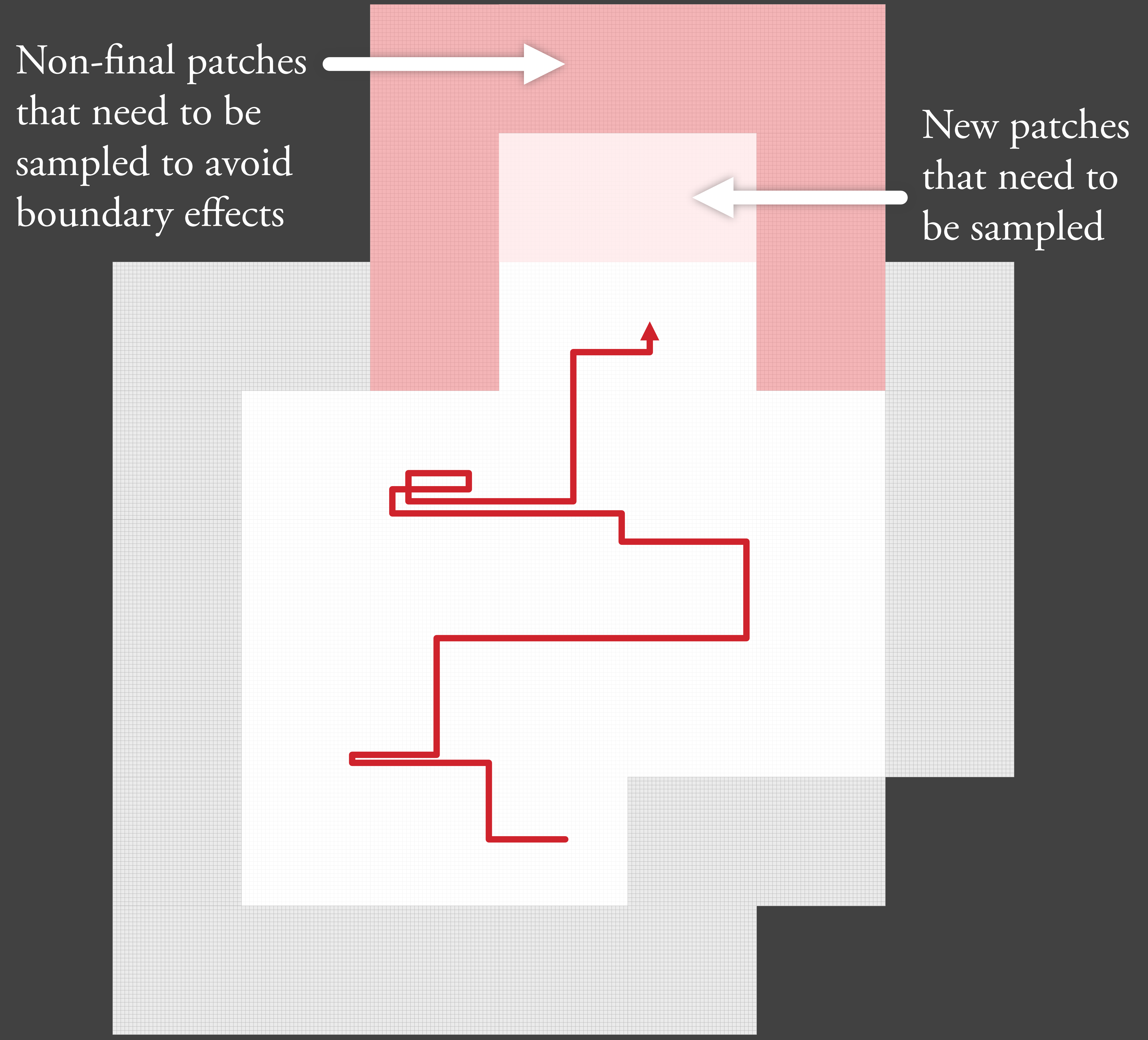
Simulator

Infinite two-dimensional grid.

Simulator

Infinite two-dimensional grid.

Procedurally generated:

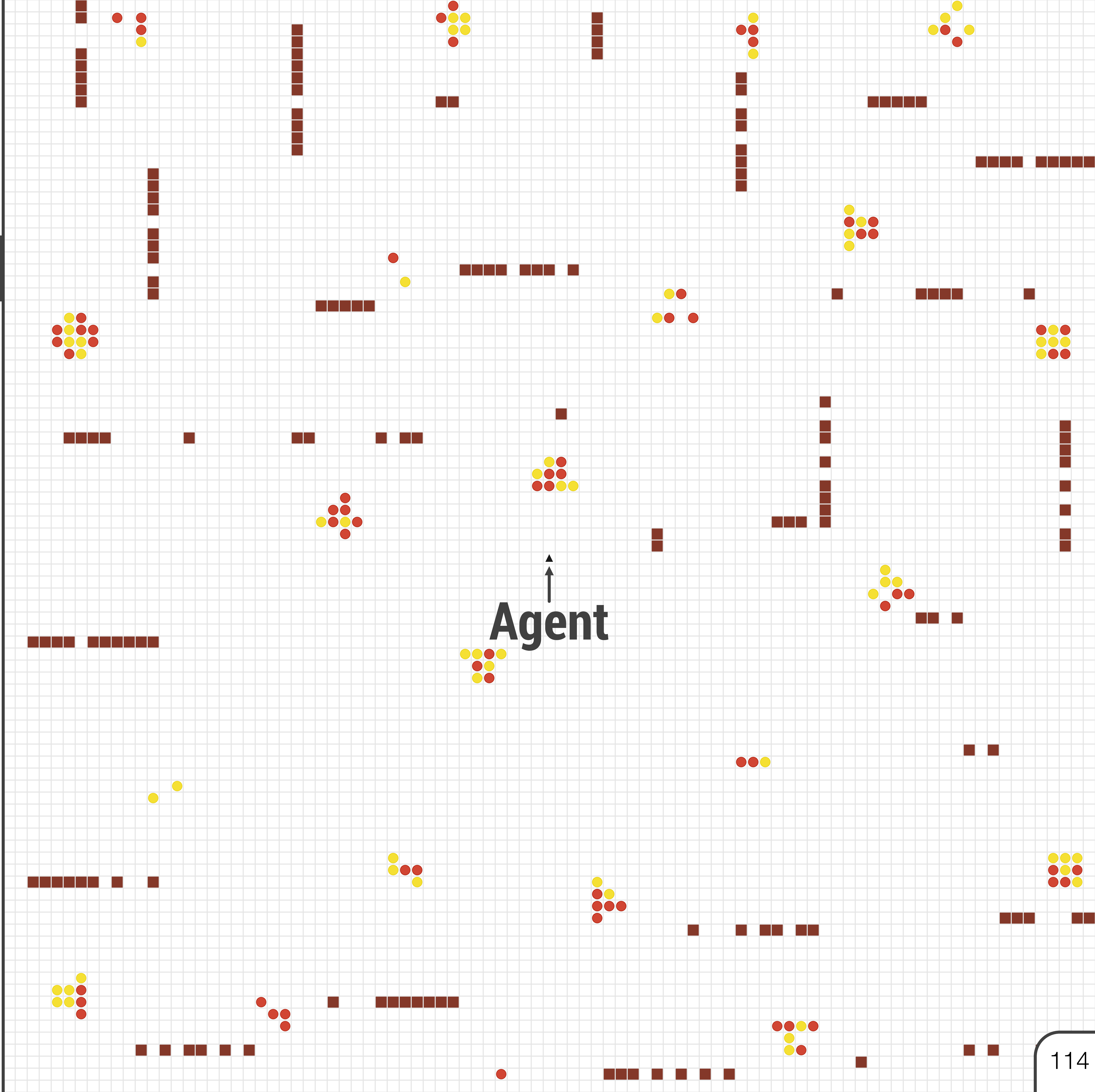


Simulator

Infinite two-dimensional grid.

Contains items of various types.

$$\mathrm{I} \triangleq \{\mathrm{I}_0, \ldots, \mathrm{I}_m\}$$



Simulator

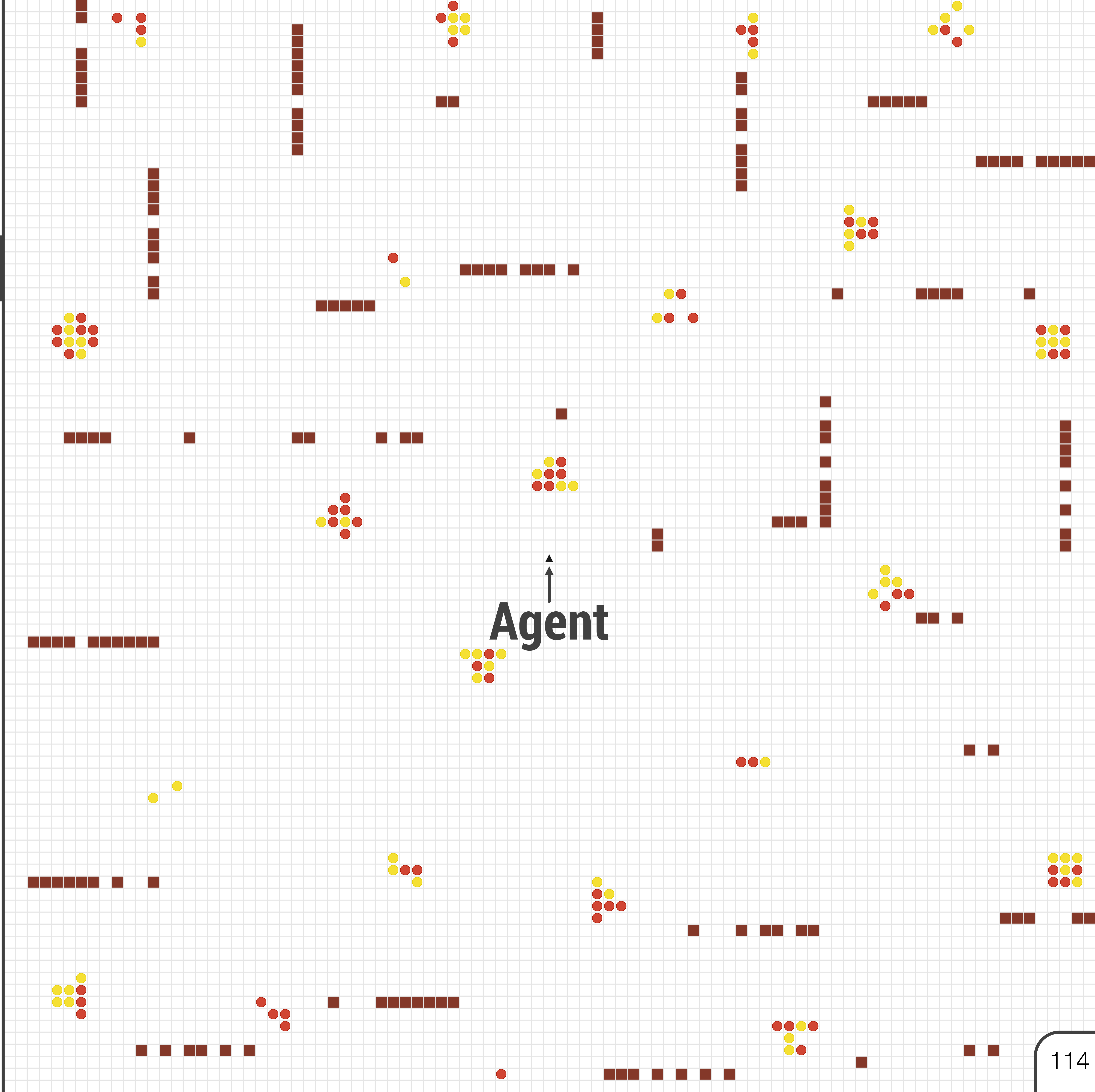
Infinite two-dimensional grid.

Contains items of various types.

$$\mathbf{I} \triangleq \{I_0, \ldots, I_m\}$$

Distributed according to a *pairwise-interaction point process*:

$$p(\mathbf{I}) \propto \exp\left\{ \sum_{i=0}^{m} \underbrace{f(I_i)}_{\textit{intensity}} + \sum_{j=0}^{m} \underbrace{g(I_i, I_j)}_{\textit{interaction}} \right\}$$

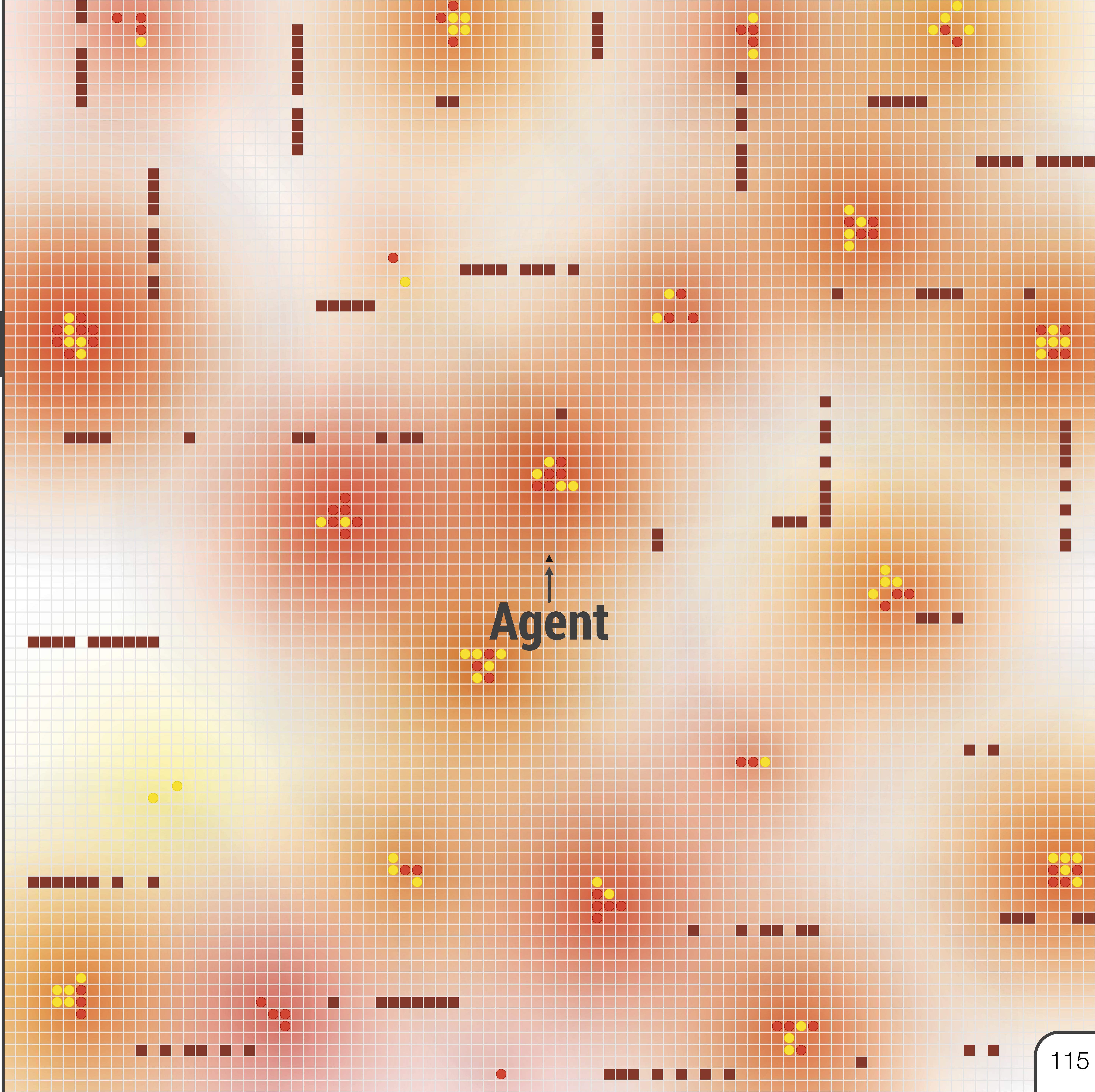


Simulator

Infinite two-dimensional grid.

Contains items of various types.

Each item has a *color* and a *scent*.



Simulator

Infinite two-dimensional grid.

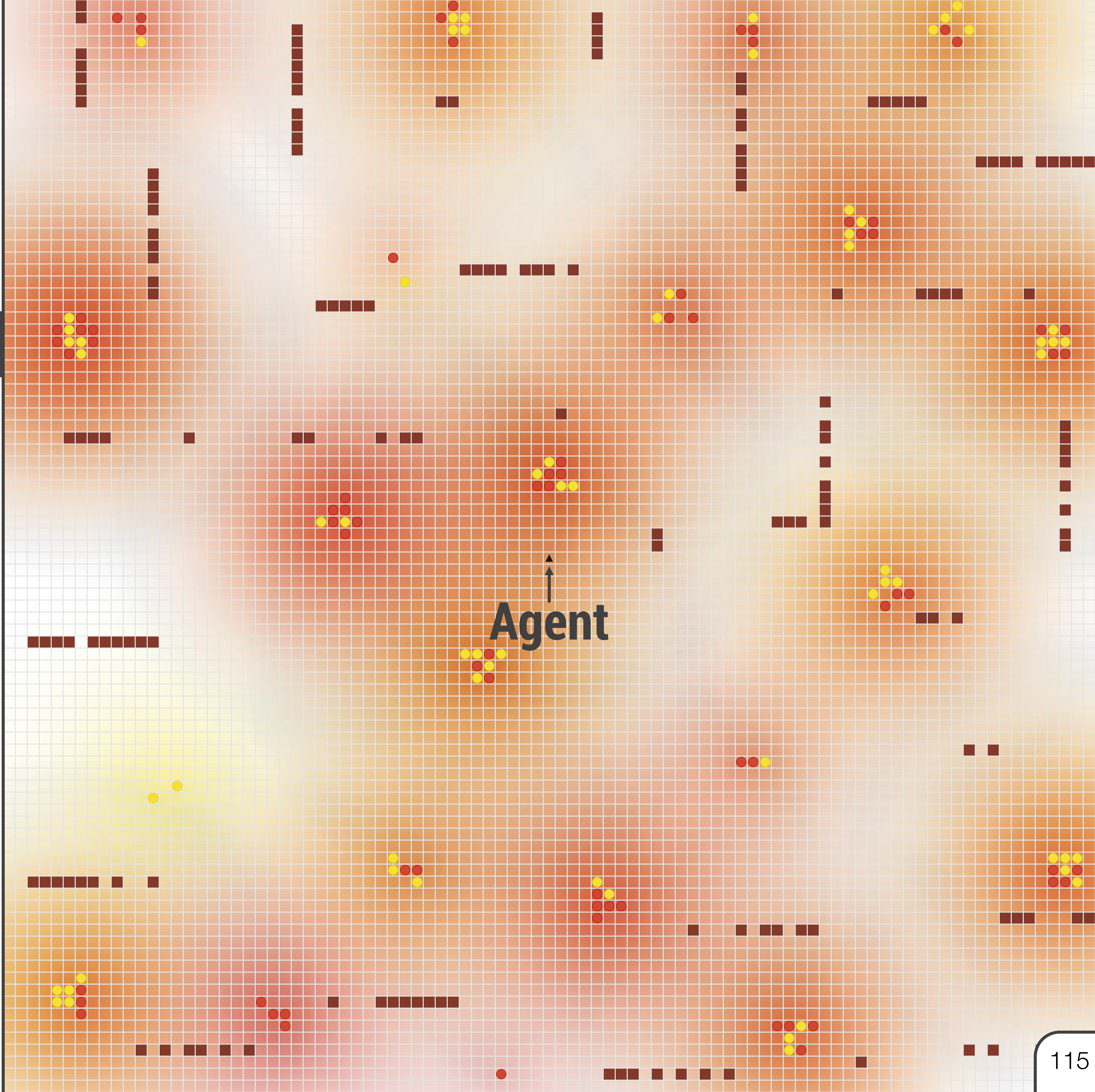
Contains items of various types.

Each item has a *color* and a *scent*.

Scent diffuses over space and time:

$$S_{x,y}^{t} = \overbrace{C_{x,y}^{t}}^{\textit{current cell scent}} + \lambda \overbrace{S_{x,y}^{t-1}}^{\textit{previous scent}}$$

$$+ \alpha \underbrace{\left(S_{x-1,y}^{t-1} + S_{x+1,y}^{t-1} + S_{x,y-1}^{t-1} + S_{x,y+1}^{t-1} \right)}_{\textit{neighboring cells diffused scent}}$$



Simulator

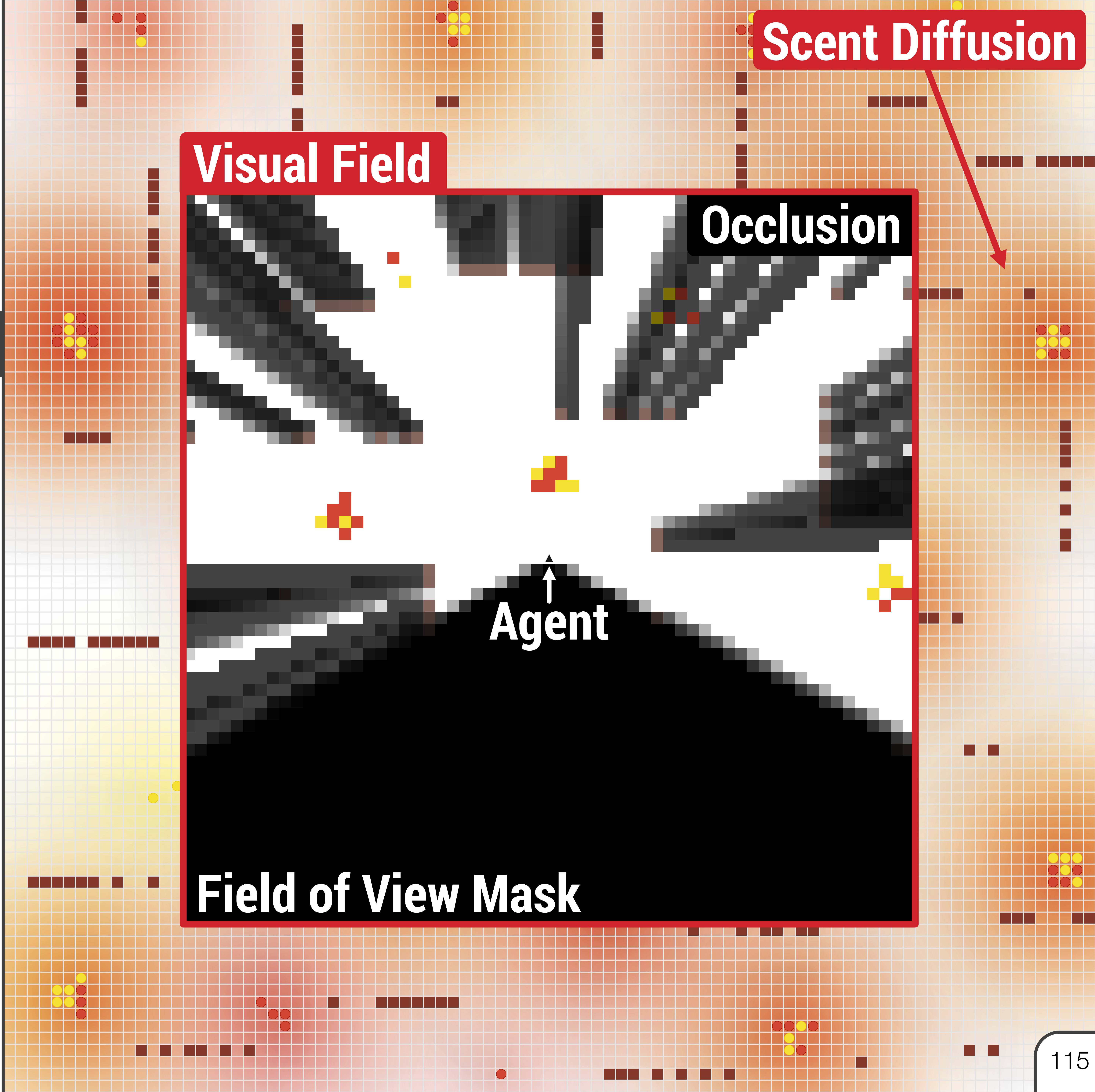
Infinite two-dimensional grid.

Contains items of various types.

Each item has a *color* and a *scent*.

Scent diffuses over space and time:

$$S^{t}_{x,y} = \overbrace{C^{t}_{x,y}}^{\textit{current cell scent}} + \lambda \overbrace{S^{t-1}_{x,y}}^{\textit{previous scent}} + \alpha \underbrace{\left(S^{t-1}_{x-1,y} + S^{t-1}_{x+1,y} + S^{t-1}_{x,y-1} + S^{t-1}_{x,y+1}\right)}_{\textit{neighboring cells diffused scent}}$$



Simulator

Infinite two-dimensional grid.

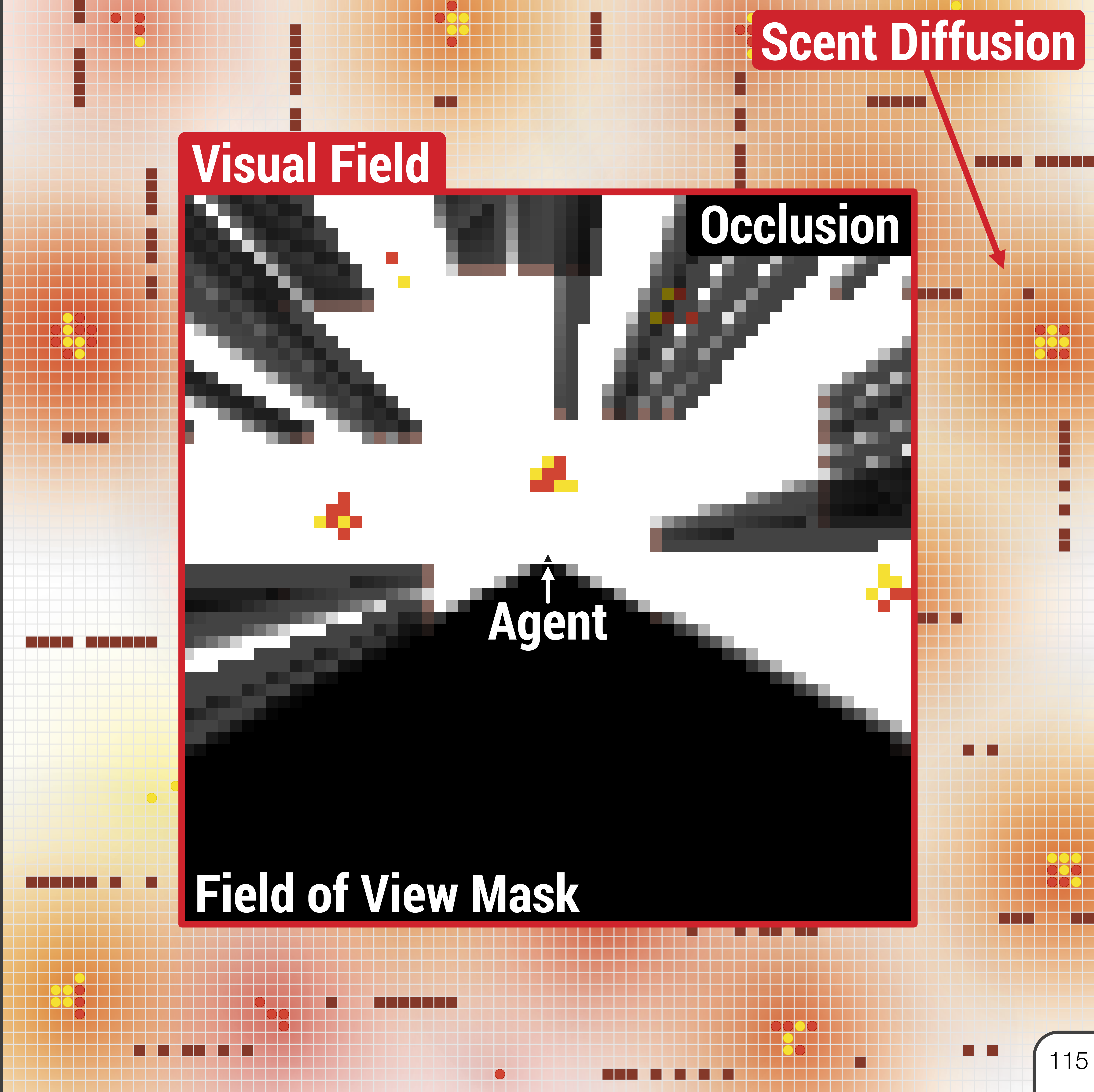
Contains items of various types.

Each item has a *color* and a *scent*.

Scent diffuses over space and time:

$$S^t_{x,y} = \overbrace{C^t_{x,y}}^{\textit{current cell scent}} + \lambda \overbrace{S^{t-1}_{x,y}}^{\textit{previous scent}} + \alpha \underbrace{\left(S^{t-1}_{x-1,y} + S^{t-1}_{x+1,y} + S^{t-1}_{x,y-1} + S^{t-1}_{x,y+1}\right)}_{\textit{neighboring cells diffused scent}}$$

Vision and scent are *complementary*.



Simulator

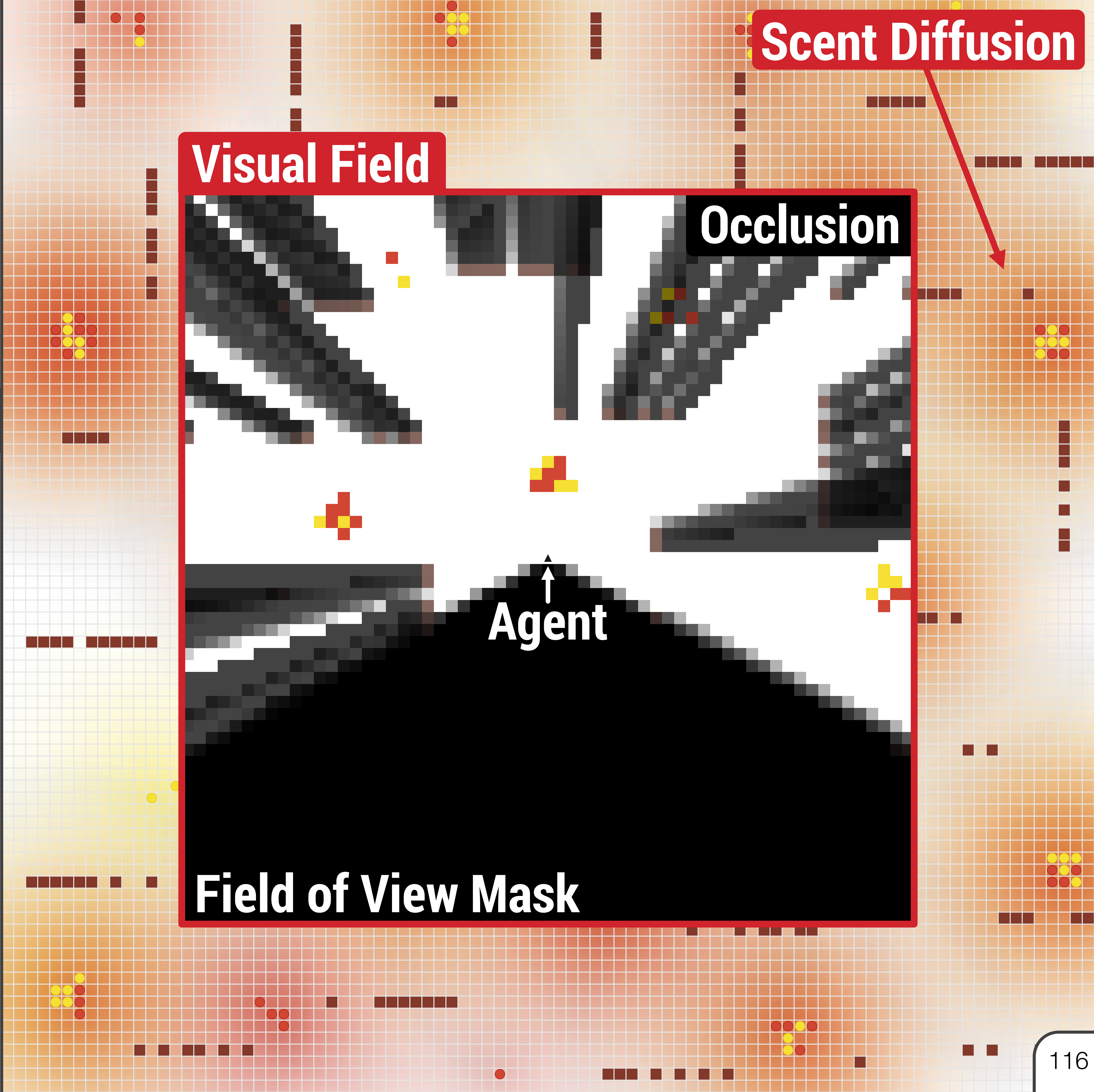
Infinite two-dimensional grid.

Contains items of various types.

Each item has a *color* and a *scent*.

Various constraints are also supported.

- Agent Collision Policies
- Item Movement Blocking
- Item Collection Requirements
- Item Collection Costs



Environment

Learning tasks can be defined in terms of *reward functions* and *reward schedules*.

Reward Functions

`Action[v]`	Give reward v to agents when take an action (i.e., not a no-op).

Environment

Learning tasks can be defined in terms of *reward functions* and *reward schedules*.

Reward Functions	
Action[v]	Give reward v to agents when take an action (i.e., not a no-op).

Environment

Learning tasks can be defined in terms of *reward functions* and *reward schedules*.

Reward Functions

`Action[v]`	Give reward v to agents when take an action (i.e., not a no-op).
`Collect[i, v]`	Give reward v to agents for each item of type i that they collect.
`Avoid[i, v]`	Give reward -v to agents for each item of type i that they collect.

Learning Tasks

Learning tasks can be defined in terms of *reward functions* and *reward schedules*.

Reward Functions

Action[v]	Give reward v to agents when take an action (i.e., not a no-op).
Collect[i, v]	Give reward v to agents for each item of type i that they collect.
Avoid[i, v]	Give reward -v to agents for each item of type i that they collect.
Explore[v]	Give reward v to agents each time they move further away from their starting position in the world map.

Learning tasks can be defined in terms of *reward functions* and *reward schedules*.

Reward Functions

Action[v]	Give reward v to agents when take an action (i.e., not a no-op).
Collect[i, v]	Give reward v to agents for each item of type i that they collect.
Avoid[i, v]	Give reward -v to agents for each item of type i that they collect.
Explore[v]	Give reward v to agents each time they move further away from their starting position in the world map.
r1 ∧ r2	Applies both r1 and r2 and returns the sum of their rewards.

Environment

Learning tasks can be defined in terms of *reward functions* and *reward schedules*.

Reward Functions

Action[v]	Give reward v to agents when take an action (i.e., not a no-op).
Collect[i, v]	Give reward v to agents for each item of type i that they collect.
Avoid[i, v]	Give reward -v to agents for each item of type i that they collect.
Explore[v]	Give reward v to agents each time they move further away from their starting position in the world map.
r1 ∧ r2	Applies both r1 and r2 and returns the sum of their rewards.

Reward Schedules

Fixed[r]	The reward function is always fixed to r, and is thus stationary.

Learning Tasks

Learning tasks can be defined in terms of *reward functions* and *reward schedules*.

Reward Functions	
Action[v]	Give reward v to agents when take an action (i.e., not a no-op).
Collect[i, v]	Give reward v to agents for each item of type i that they collect.
Avoid[i, v]	Give reward -v to agents for each item of type i that they collect.
Explore[v]	Give reward v to agents each time they move further away from their starting position in the world map.
$r1 \wedge r2$	Applies both r1 and r2 and returns the sum of their rewards.

Reward Schedules	
Fixed[r]	The reward function is always fixed to r, and is thus stationary.
Curriculum[$\{r_i, t_i\}_{i=1}^{R}$]	Use reward function r_1 for the first t_1 steps, then r_2 for t_2 steps, ..., and keep using r_R after the list of reward functions is exhausted.

Learning Tasks

Learning tasks can be defined in terms of *reward functions* and *reward schedules*.

Reward Functions	
`Action[v]`	Give reward `v` to agents when take an action (i.e., not a no-op).
`Collect[i, v]`	Give reward `v` to agents for each item of type `i` that they collect.
`Avoid[i, v]`	Give reward `-v` to agents for each item of type `i` that they collect.
`Explore[v]`	Give reward `v` to agents each time they move further away from their starting position in the world map.
$r1 \wedge r2$	Applies both `r1` and `r2` and returns the sum of their rewards.

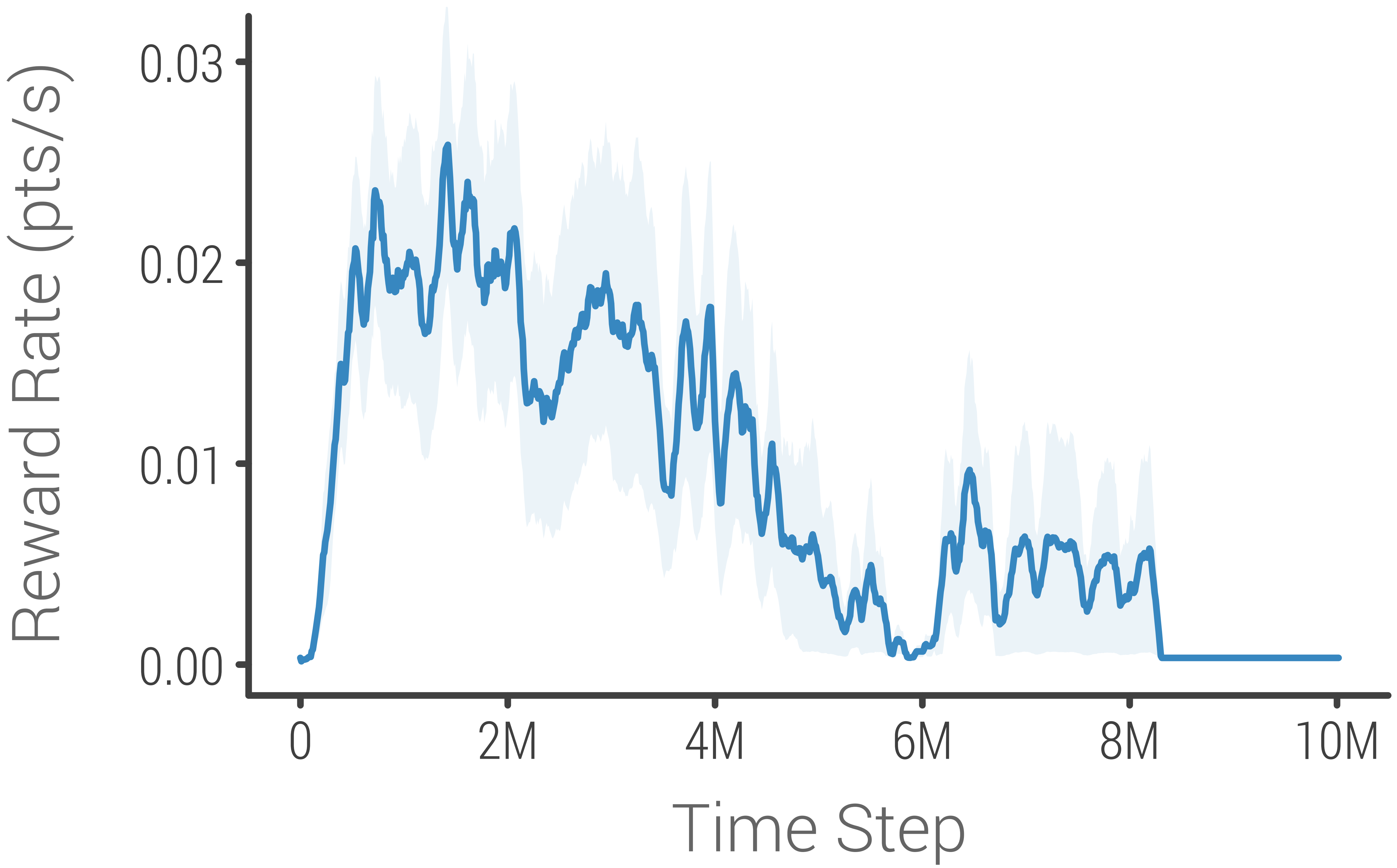
Reward Schedules	
`Fixed[r]`	The reward function is always fixed to `r`, and is thus stationary.
$\texttt{Curriculum}[\{r_i, t_i\}_{i=1}^{R}]$	Use reward function r_1 for the first t_1 steps, then r_2 for t_2 steps, ..., and keep using r_R after the list of reward functions is exhausted.
$\texttt{Cyclical}[\{r_i, t_i\}_{i=1}^{R}]$	Use reward function r_1 for the first t_1 steps, then r_2 for t_2 steps, ..., and then repeat after the list of reward functions is exhausted.

Case Studies

Reward function: Collect[**JellyBean**] $\wedge$ Avoid[**Onion**]

Case Studies

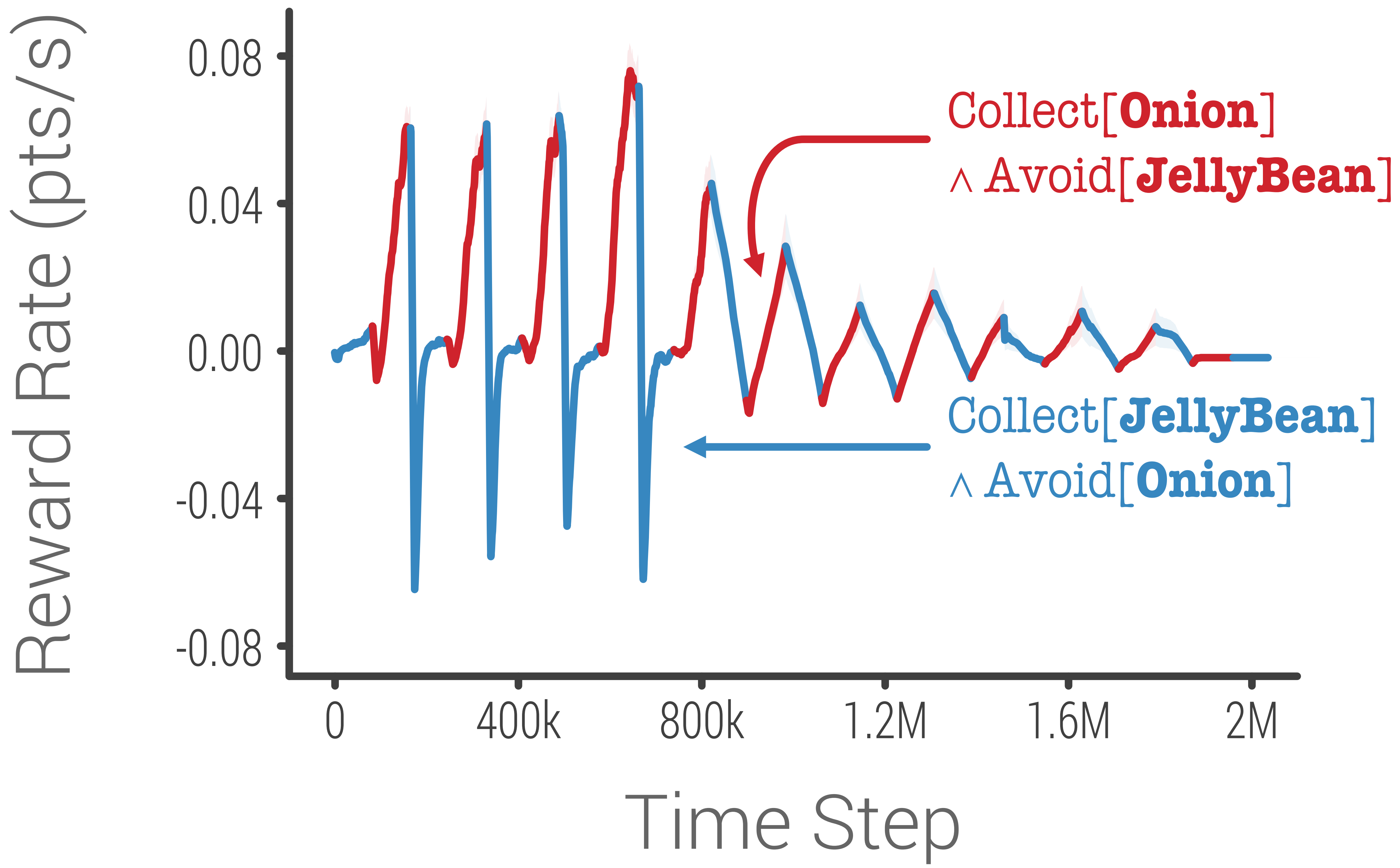
Reward function: Collect[**JellyBean**] $\wedge$ Avoid[**Onion**]



Reward rate is computed using a 100,000-step moving window and averaged over 20 runs.

Case Studies

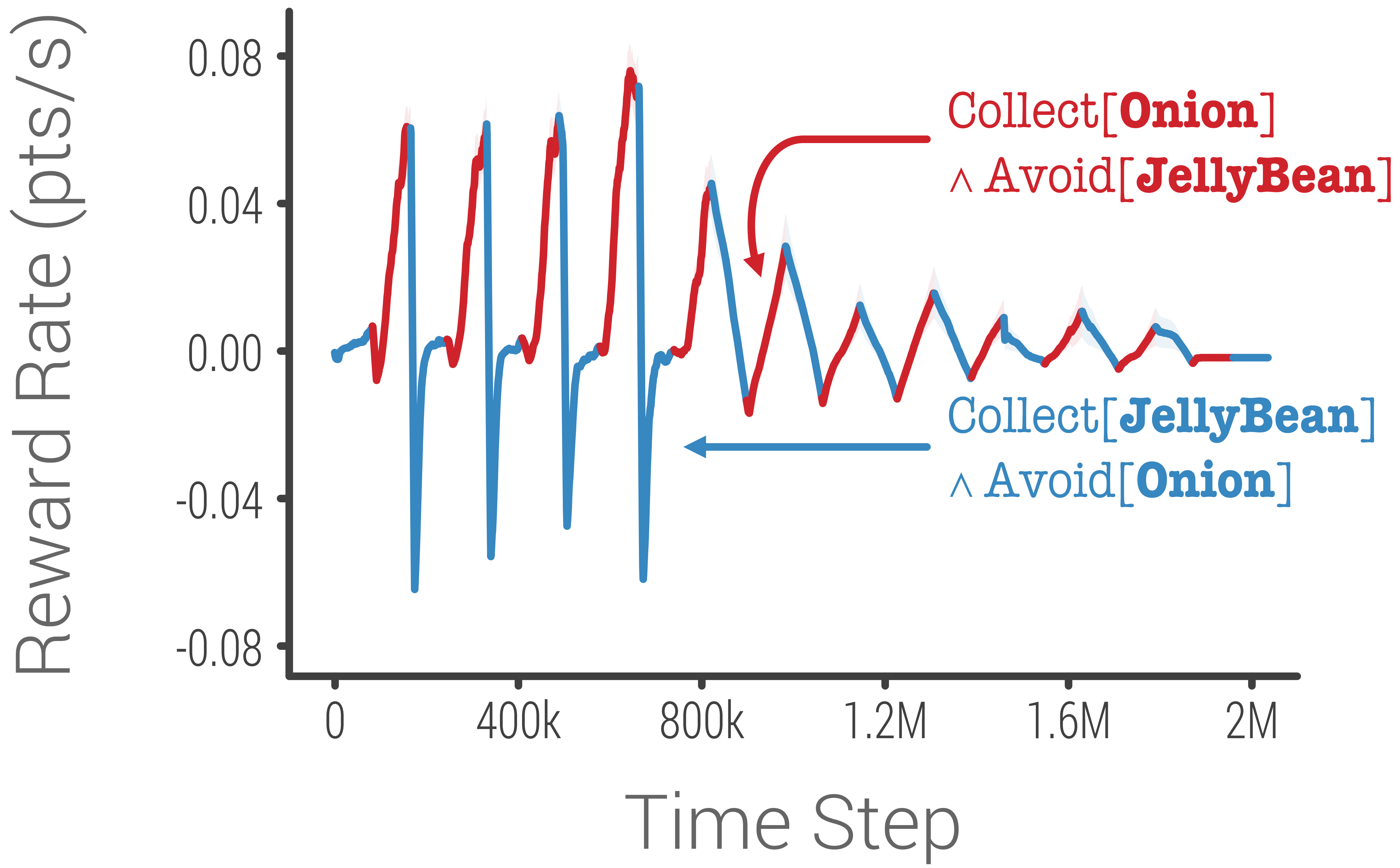
Cyclical Schedule



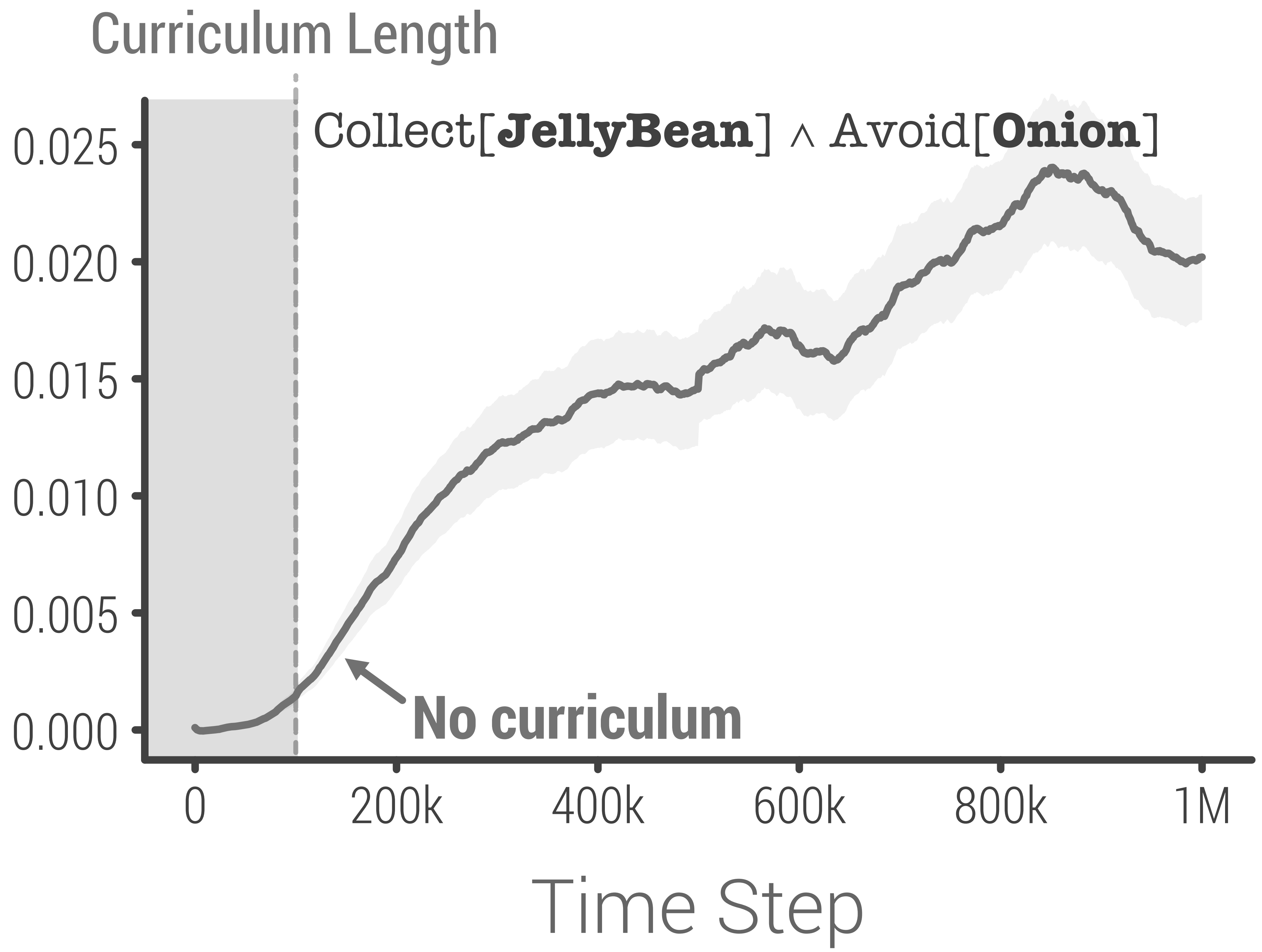
Reward rate is computed using a 100,000-step moving window and averaged over 20 runs.

Case Studies

Cyclical Schedule



Curriculum Schedule

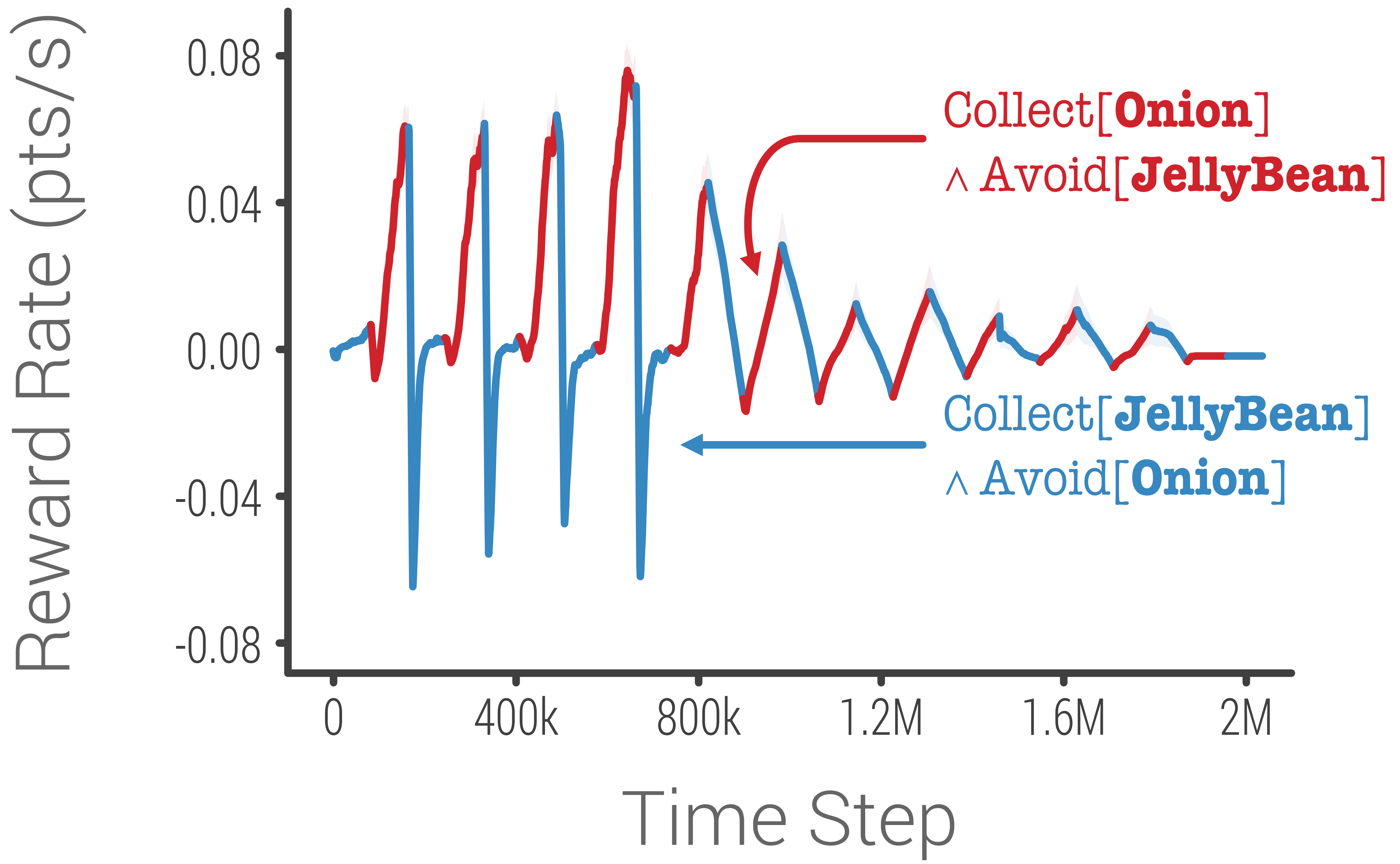


Reward rate is computed using a 100,000-step moving window and averaged over 20 runs.

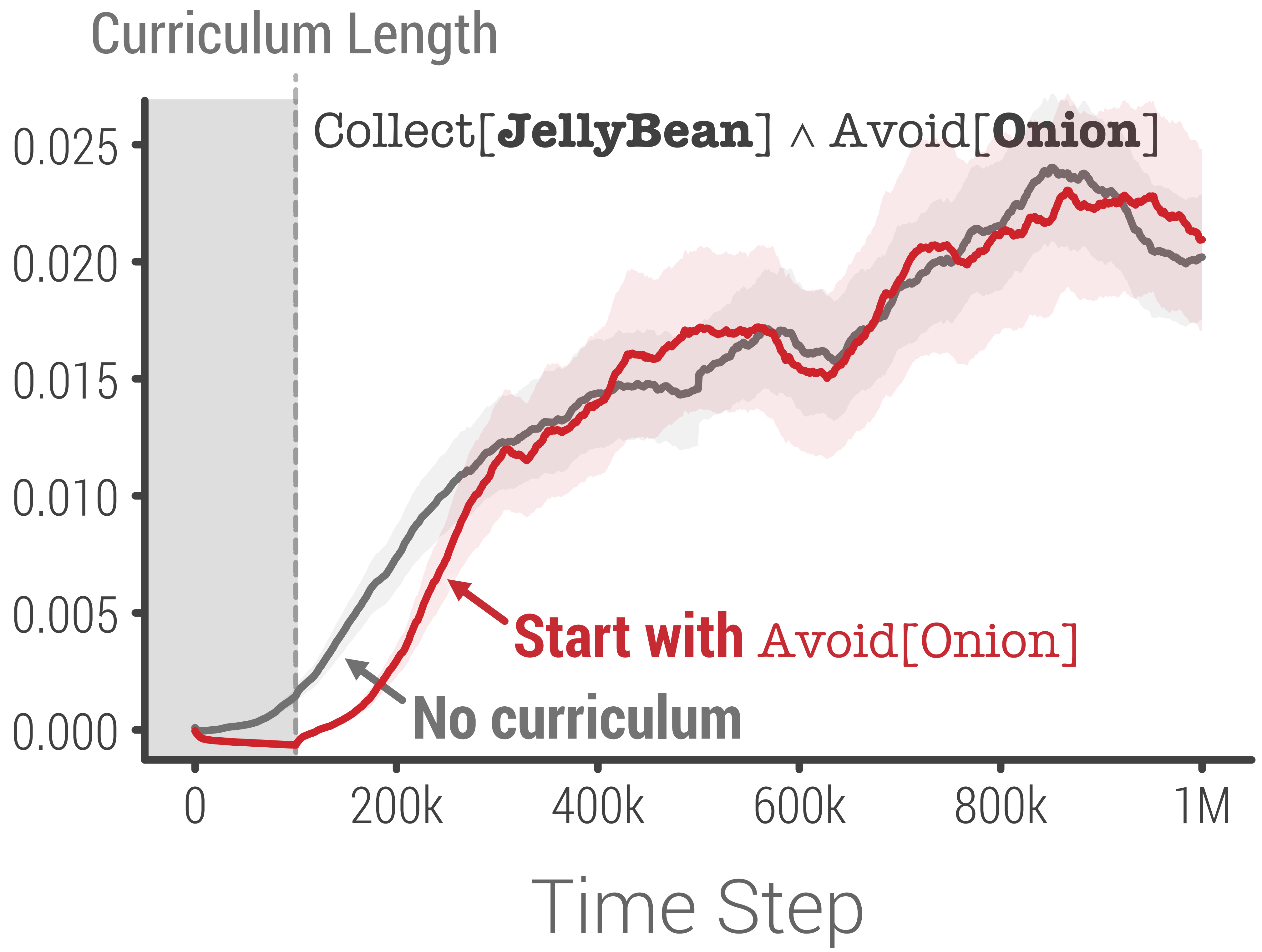
Case Studies

Non-Stationary

Cyclical Schedule



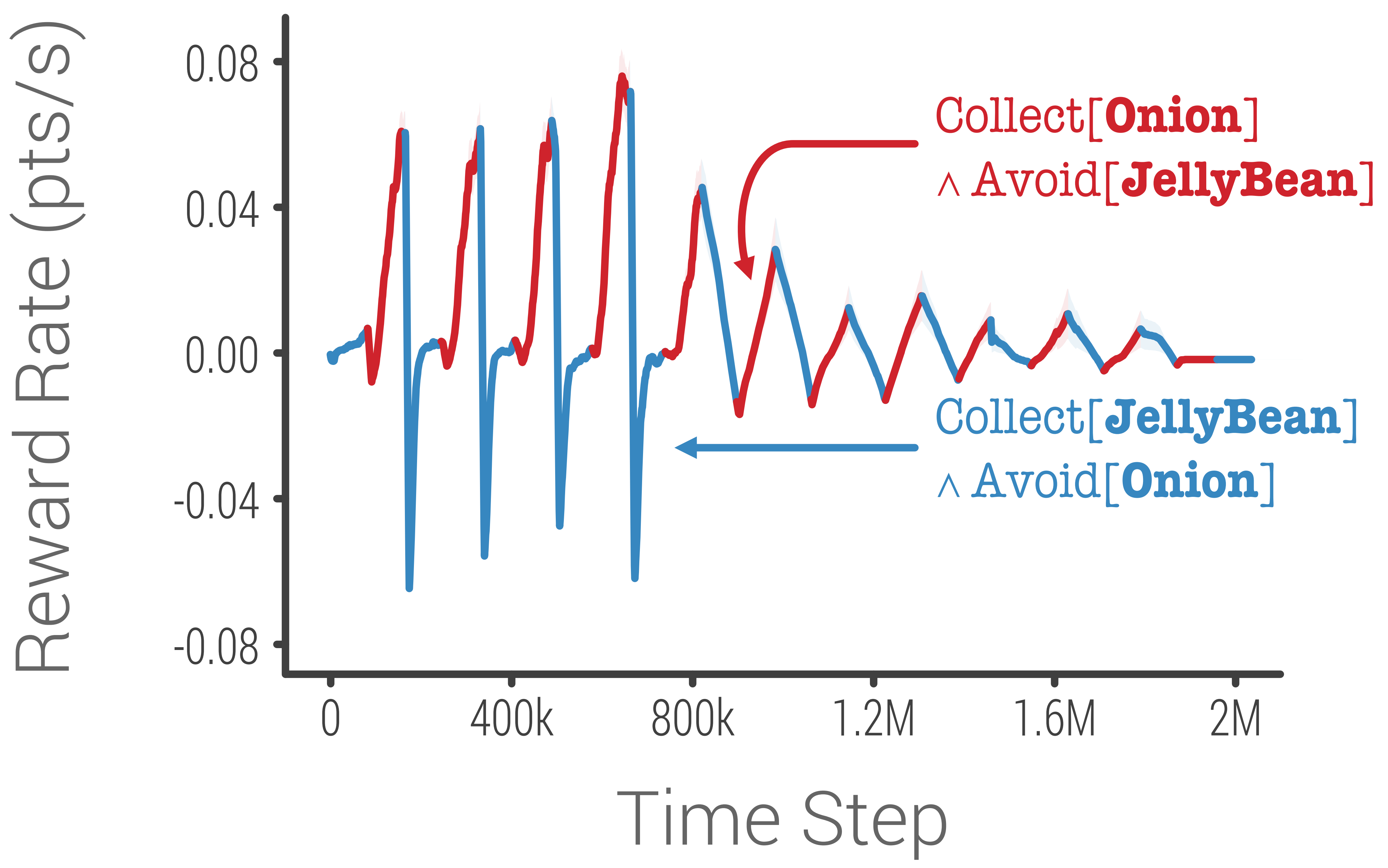
Curriculum Schedule



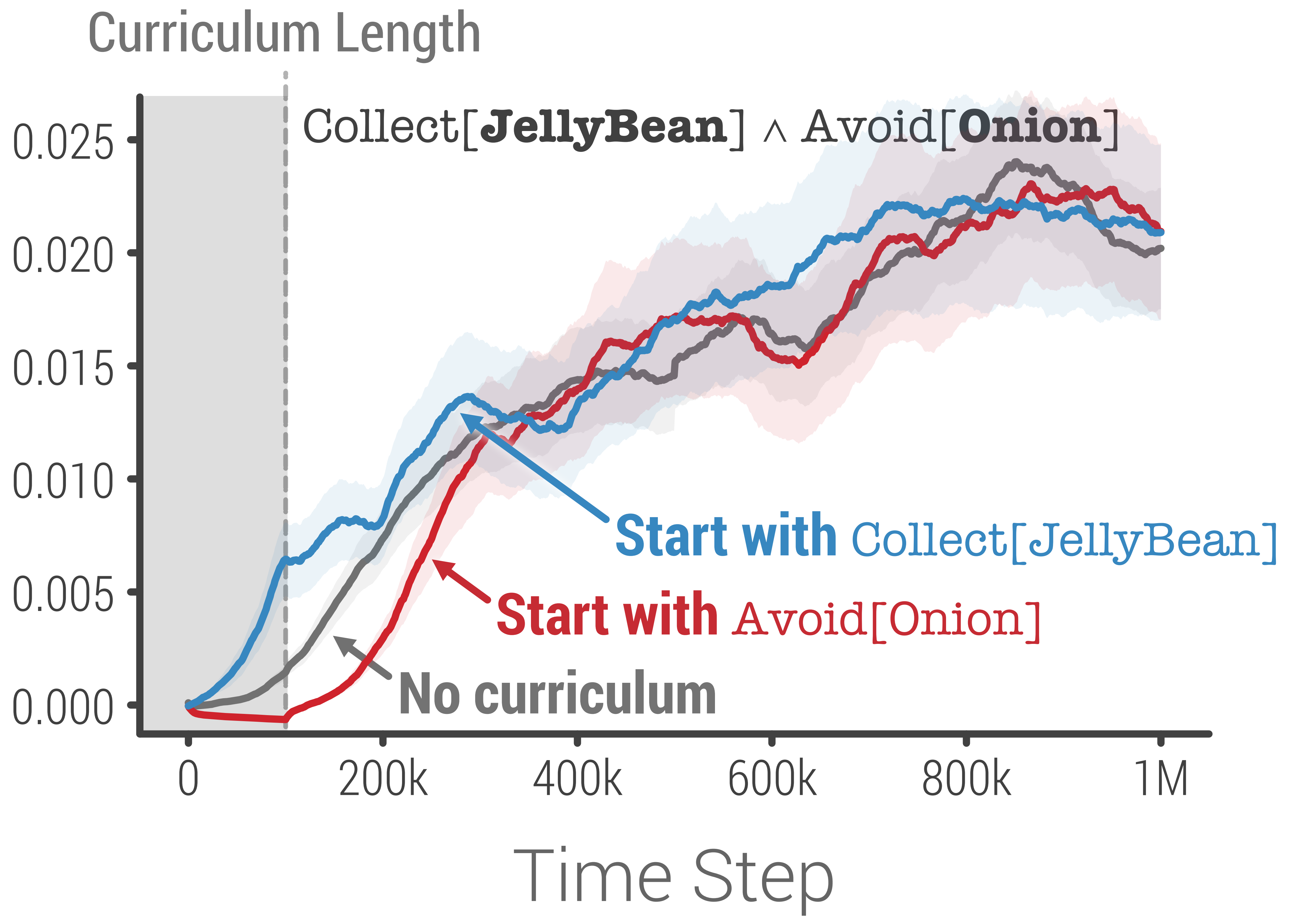
Reward rate is computed using a 100,000-step moving window and averaged over 20 runs.

Case Studies

Cyclical Schedule

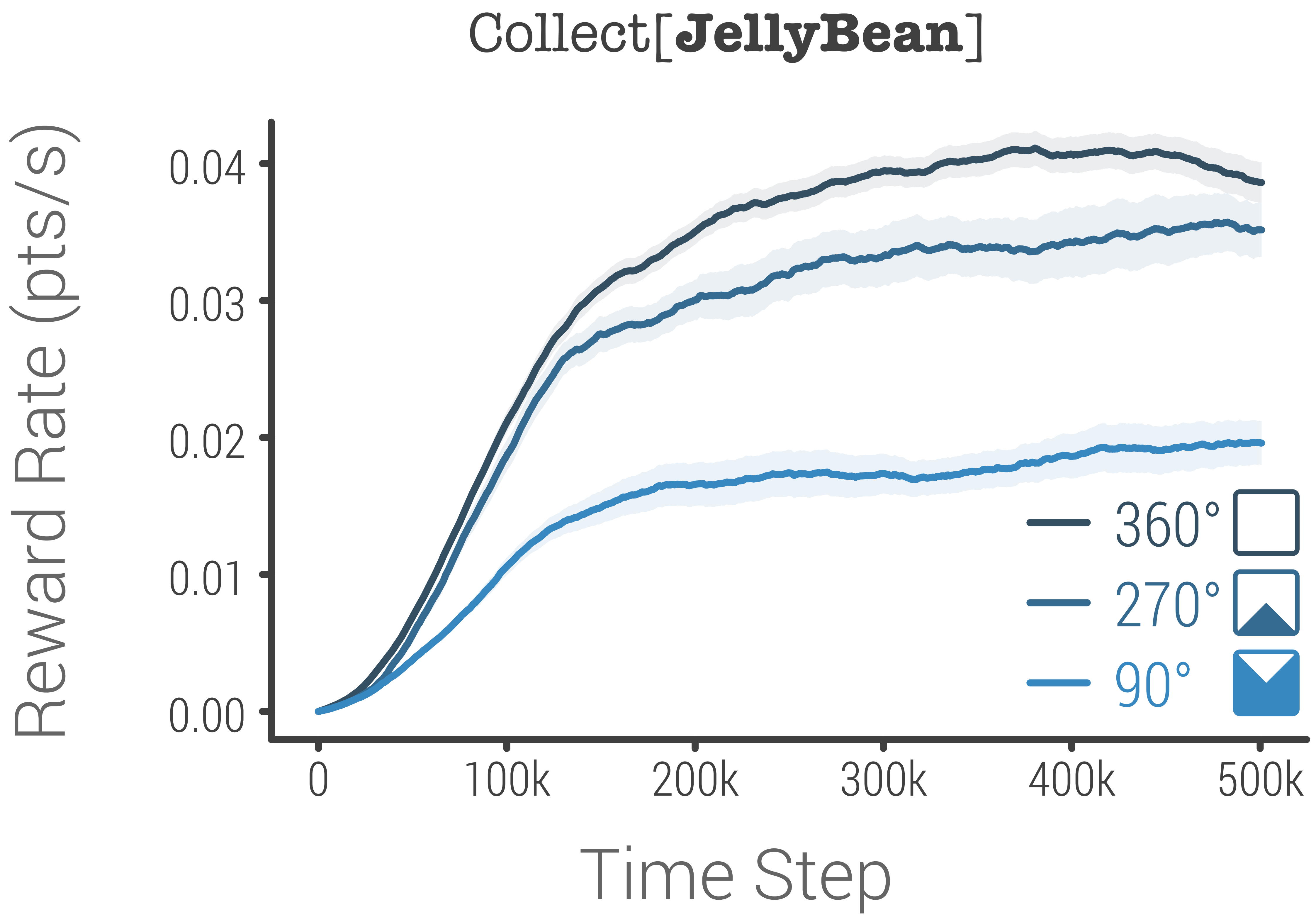


Curriculum Schedule



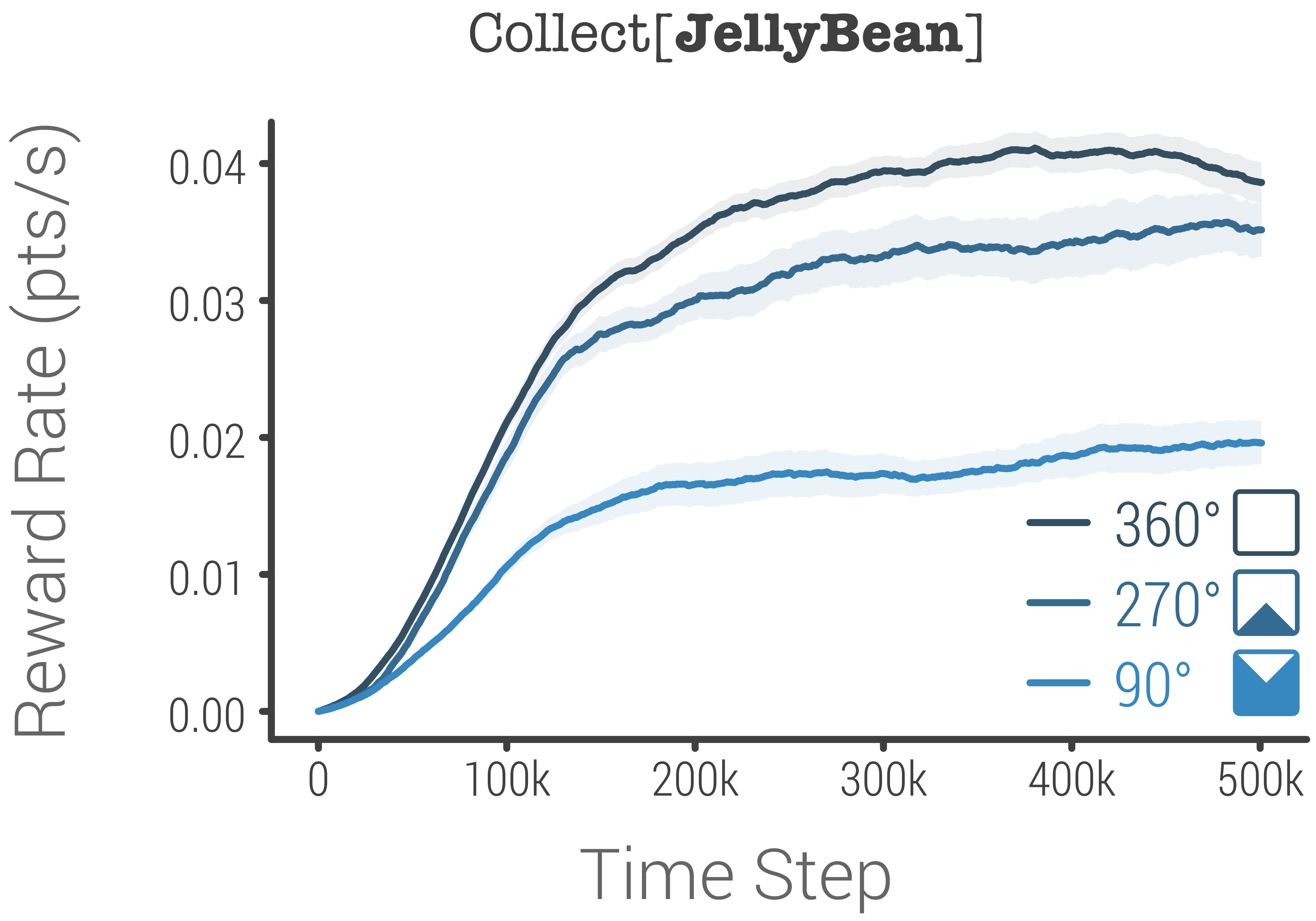
Reward rate is computed using a 100,000-step moving window and averaged over 20 runs.

Field-of-View

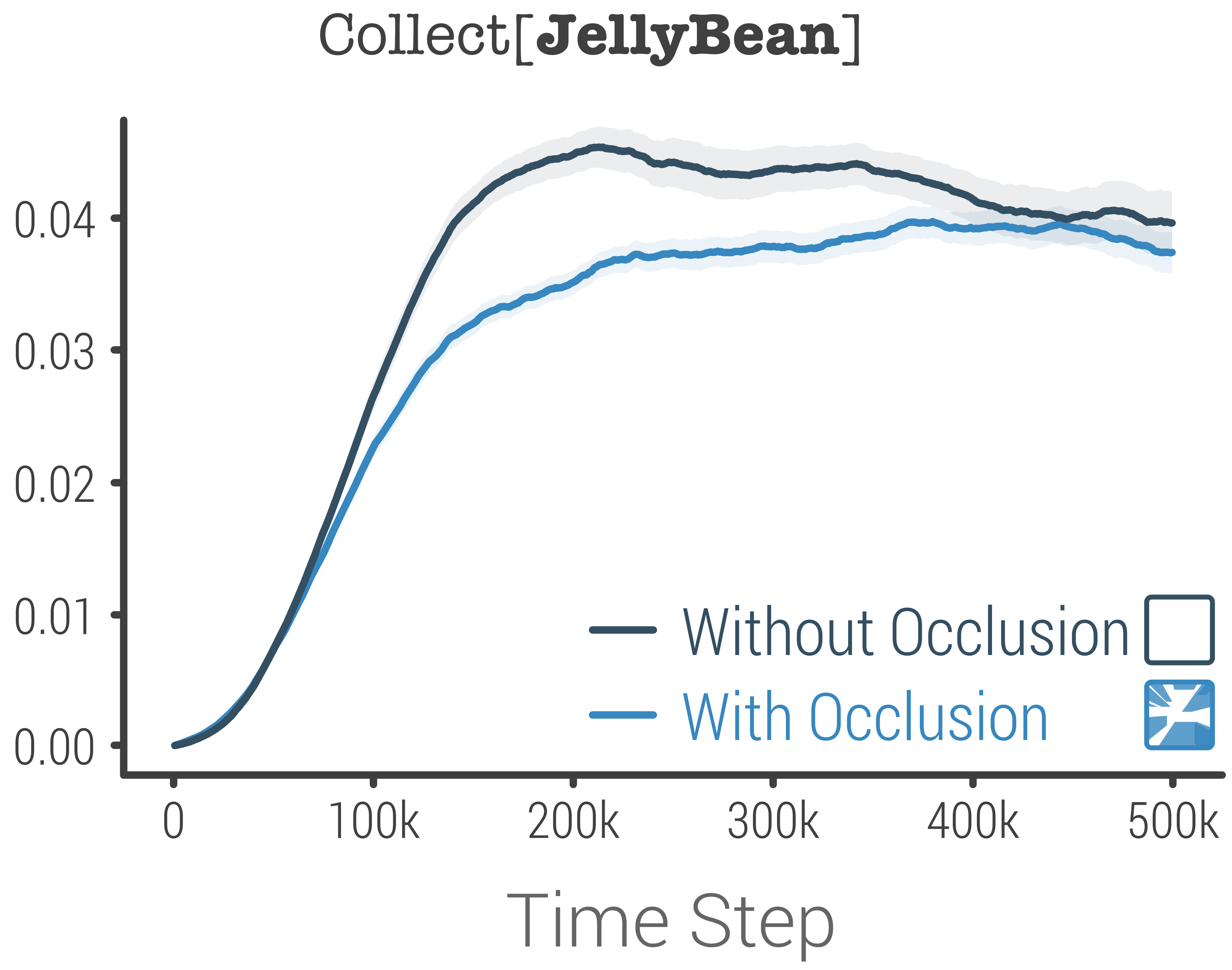


Reward rate is computed using a 100,000-step moving window and averaged over 20 runs.

Field-of-View

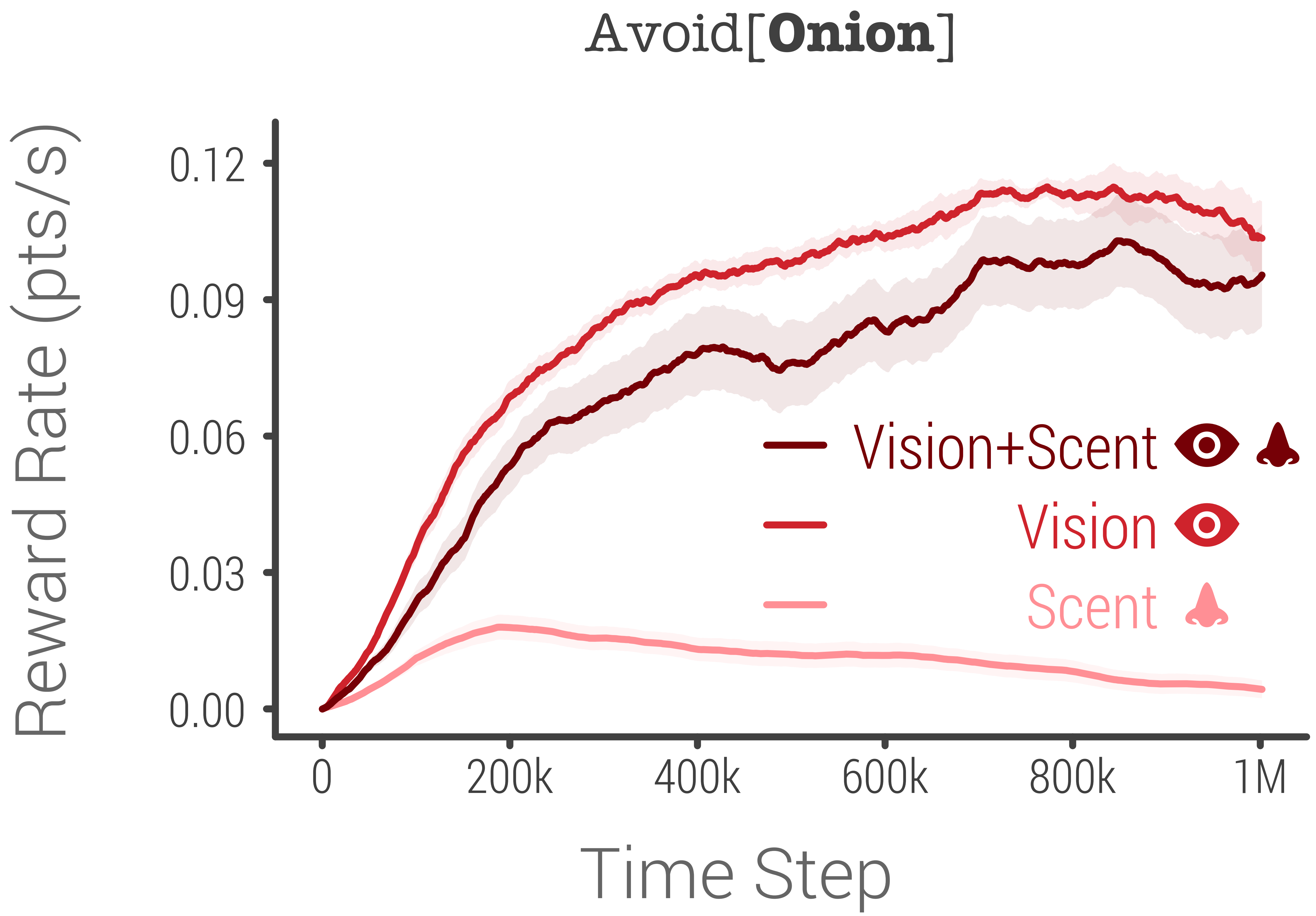


Visual Occlusion



Reward rate is computed using a 100,000-step moving window and averaged over 20 runs.

Vision/Scent Complementarity



Reward rate is computed using a 100,000-step moving window and averaged over 20 runs.

Contextual Parameter Generation for Task Compositions

Let us consider the following example:

Collect[**JellyBean**] ∧ Avoid[**Onion**]

Switch every 100,000 steps

Avoid[**JellyBean**] ∧ Collect[**Onion**]

Chapter 9 [**ICLR** 2020]

Evaluation
Jelly Bean World

Multi-Task Learning
Contextual Parameter Generation
Chapters 7-8

Machine Translation
Chapter 8.2 [**EMNLP** 2018]

Link Prediction
Chapter 8.3 [**AAAI** 2020]

Multi-Task Learning

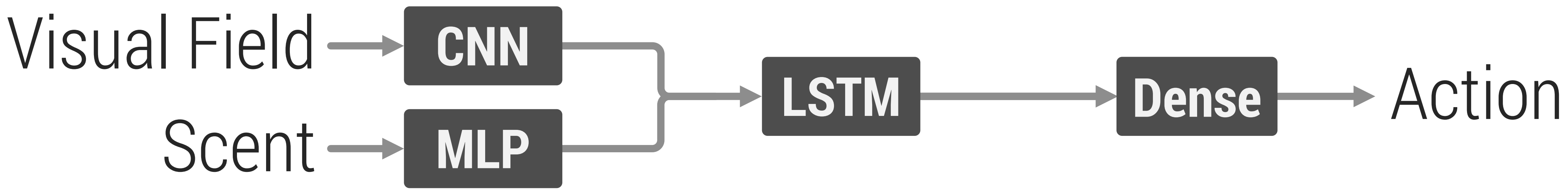
Contextual Parameter Generation for Task Compositions

Let us consider the following example:

Collect[**JellyBean**] ∧ Avoid[**Onion**]

Switch every 100,000 steps

Avoid[**JellyBean**] ∧ Collect[**Onion**]



Chapter 9 [**ICLR** 2020]

Evaluation
Jelly Bean World

Multi-Task Learning
Contextual Parameter Generation
Chapters 7-8

Machine Translation
Chapter 8.2 [**EMNLP** 2018]

Link Prediction
Chapter 8.3 [**AAAI** 2020]

Multi-Task Learning

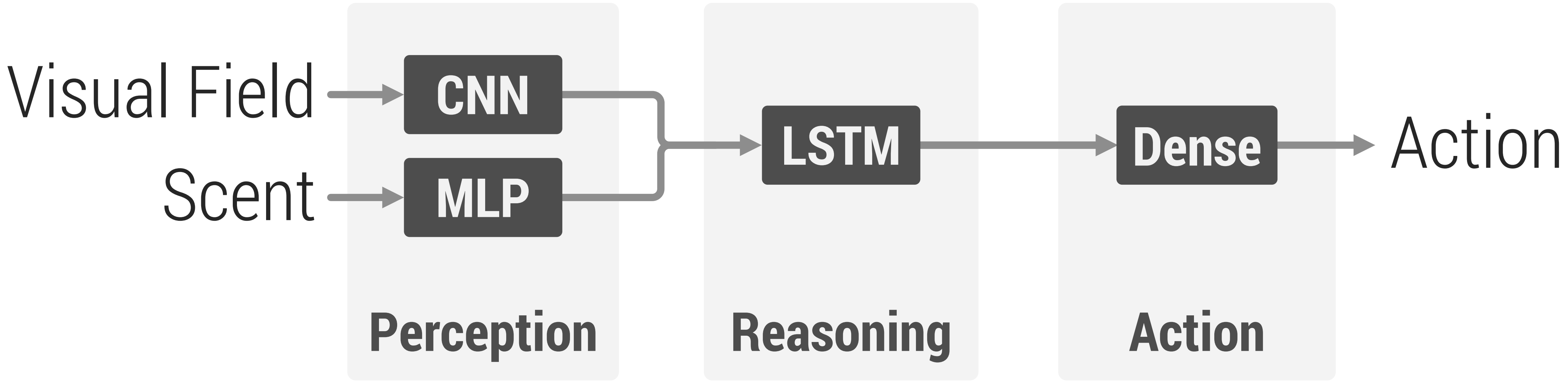
Contextual Parameter Generation for Task Compositions

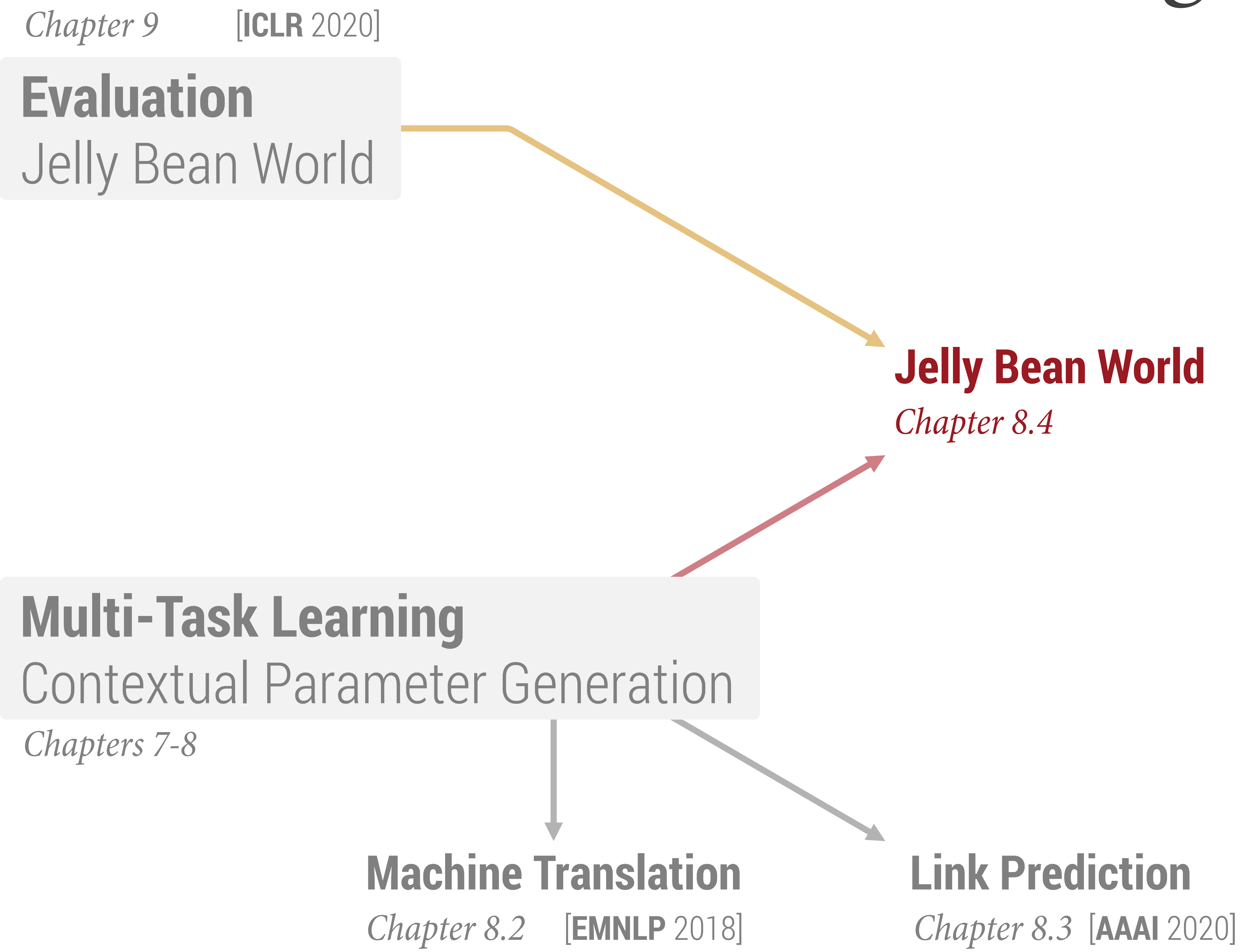
Let us consider the following example:

$$\text{Collect}[\textbf{JellyBean}] \wedge \text{Avoid}[\textbf{Onion}]$$

Switch every 100,000 steps

$$\text{Avoid}[\textbf{JellyBean}] \wedge \text{Collect}[\textbf{Onion}]$$





Multi-Task Learning

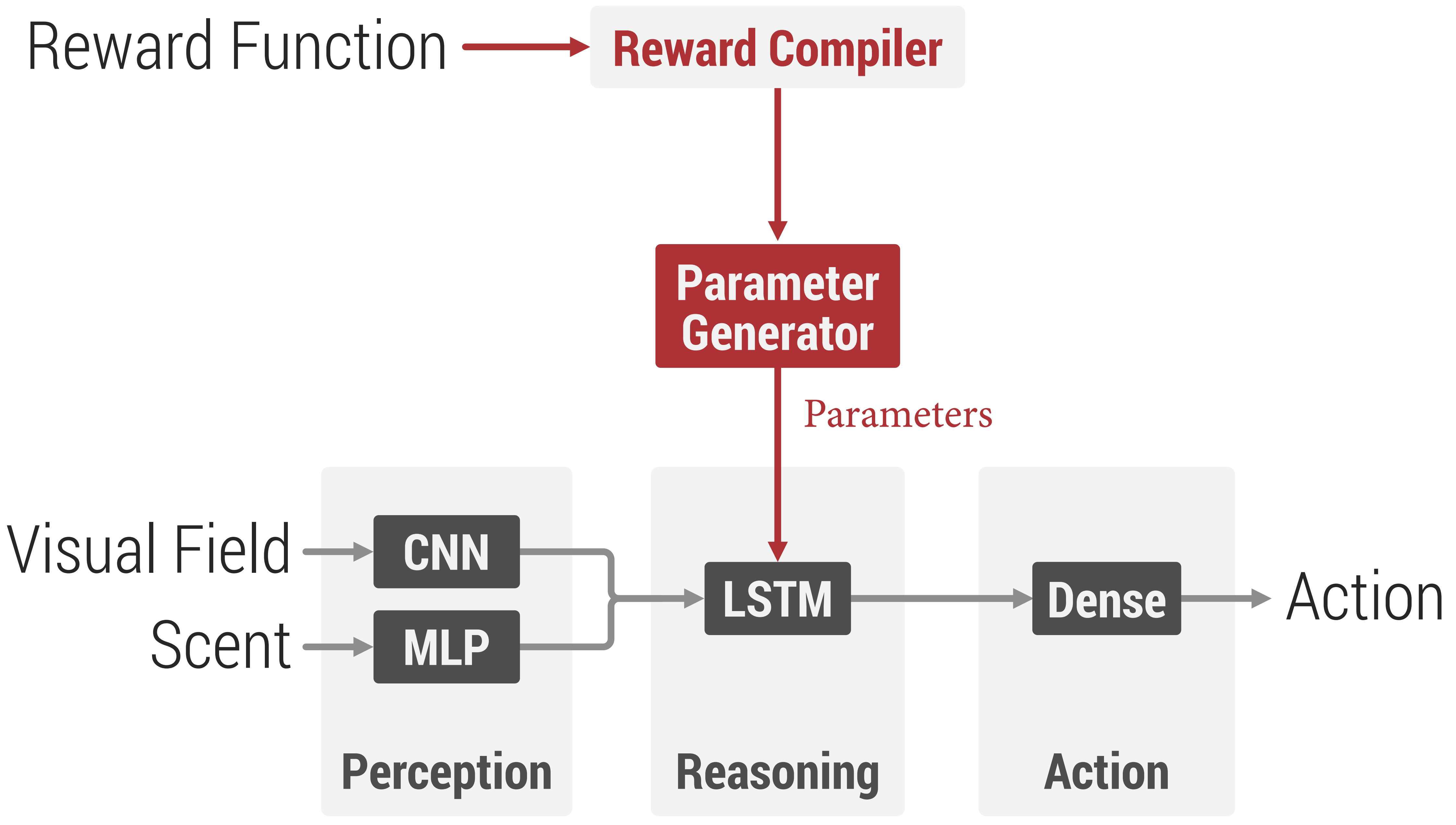
Contextual Parameter Generation for Task Compositions

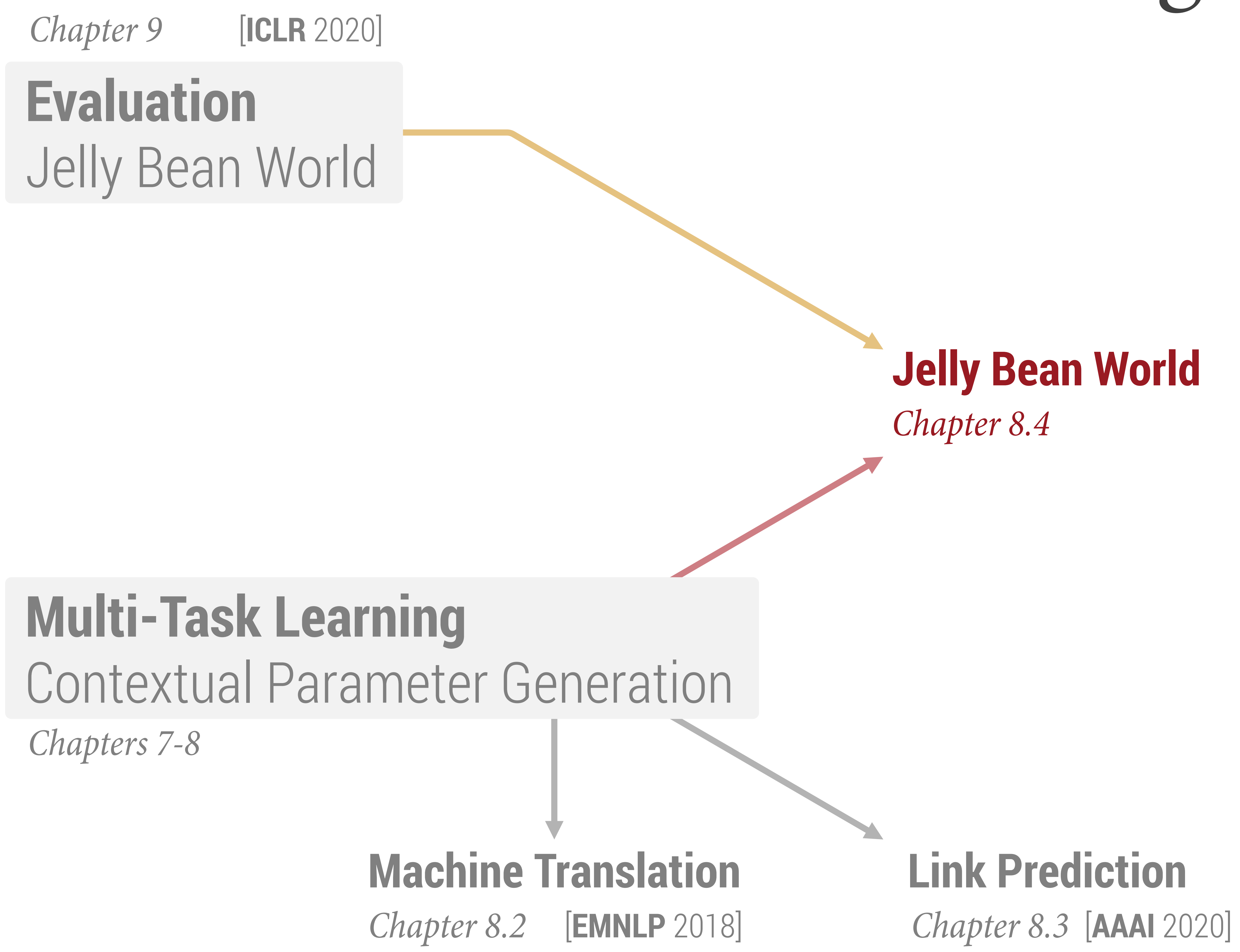
Let us consider the following example:

$$\text{Collect}[\textbf{JellyBean}] \wedge \text{Avoid}[\textbf{Onion}]$$

Switch every 100,000 steps

$$\text{Avoid}[\textbf{JellyBean}] \wedge \text{Collect}[\textbf{Onion}]$$





Multi-Task Learning

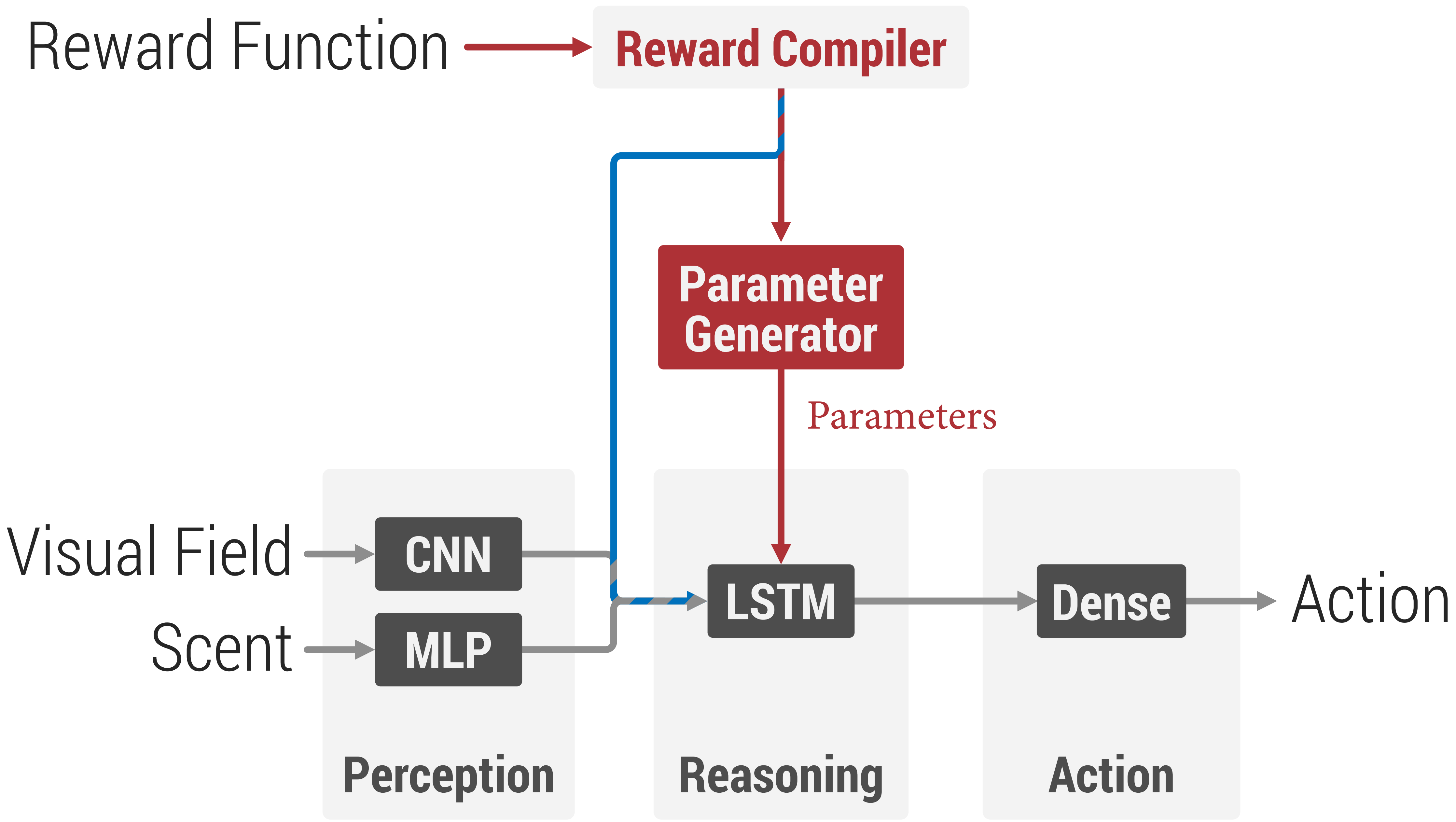
Contextual Parameter Generation for Task Compositions

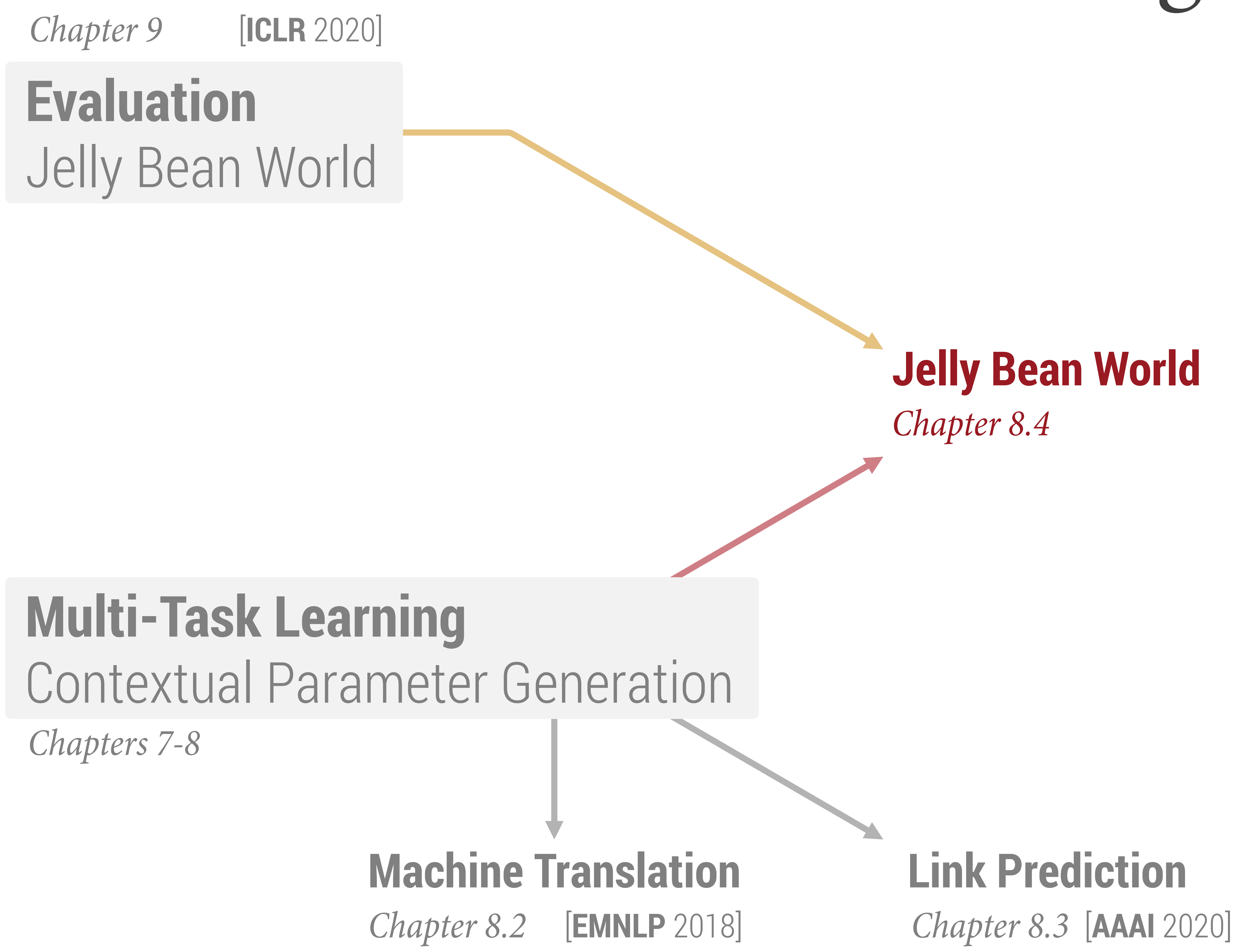
Let us consider the following example:

$$\texttt{Collect[}\textbf{JellyBean}\texttt{]} \wedge \texttt{Avoid[}\textbf{Onion}\texttt{]}$$

Switch every 100,000 steps

$$\texttt{Avoid[}\textbf{JellyBean}\texttt{]} \wedge \texttt{Collect[}\textbf{Onion}\texttt{]}$$





Multi-Task Learning

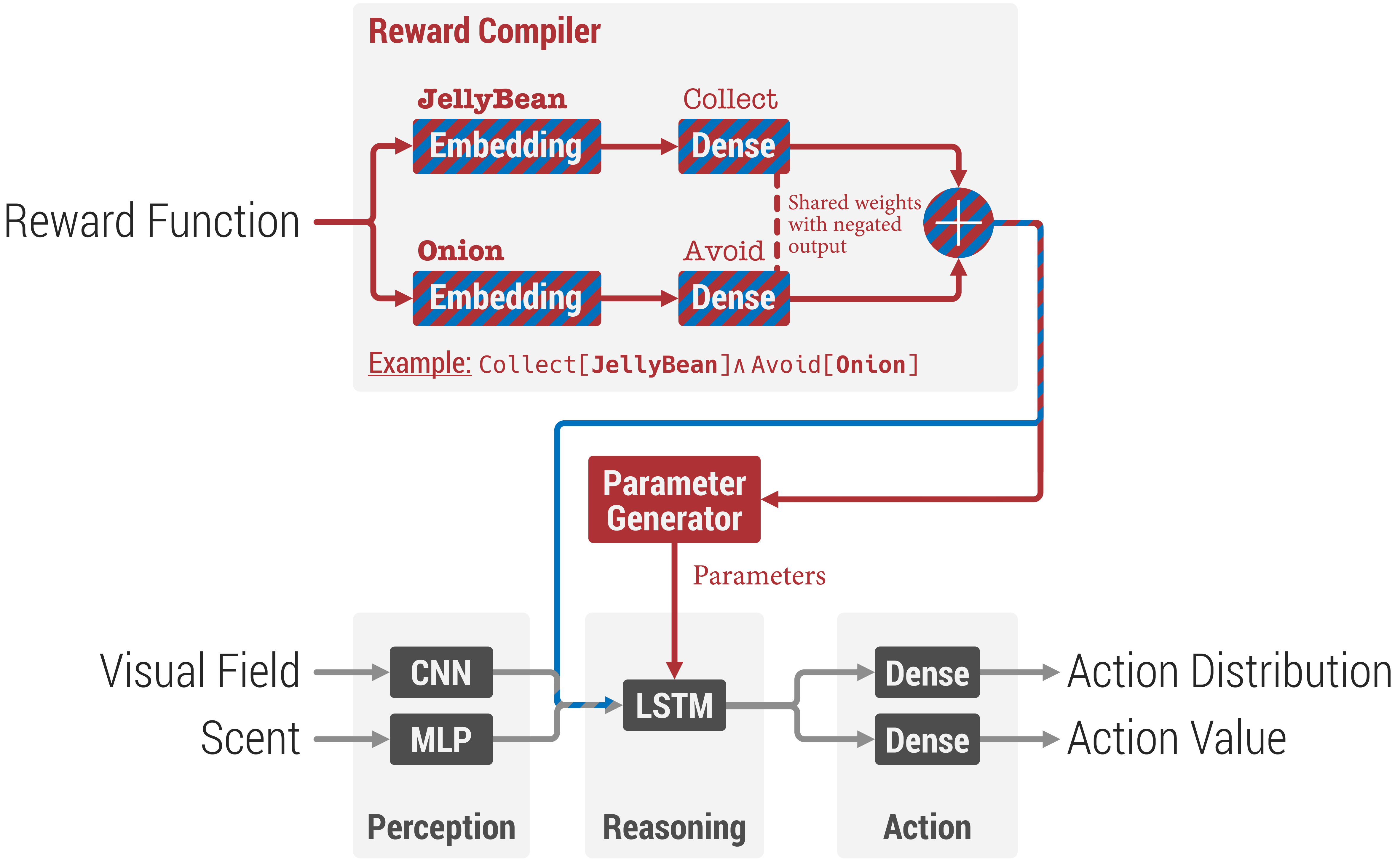
Contextual Parameter Generation for Task Compositions

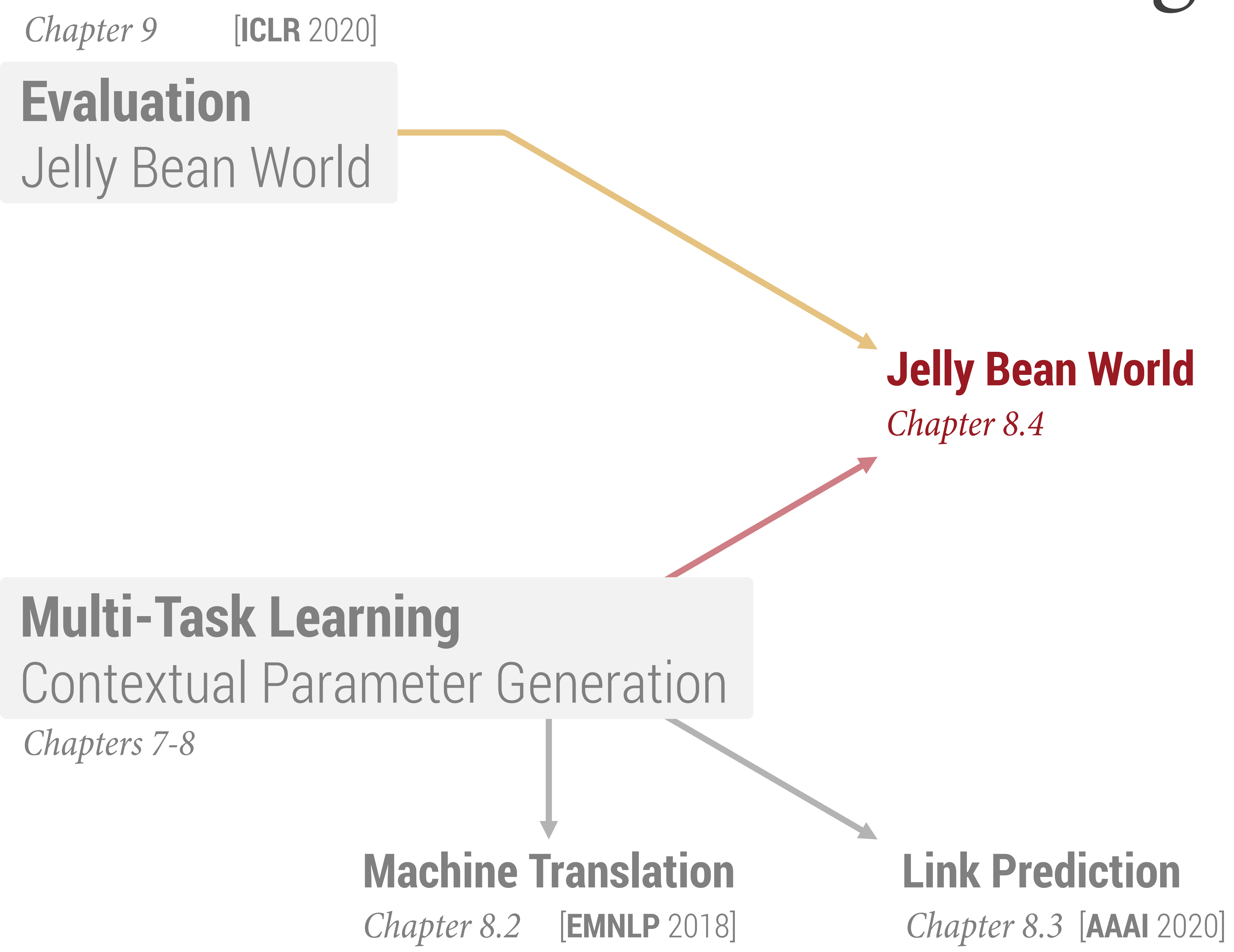
Let us consider the following example:

Collect[**JellyBean**] ∧ Avoid[**Onion**]

Switch every 100,000 steps

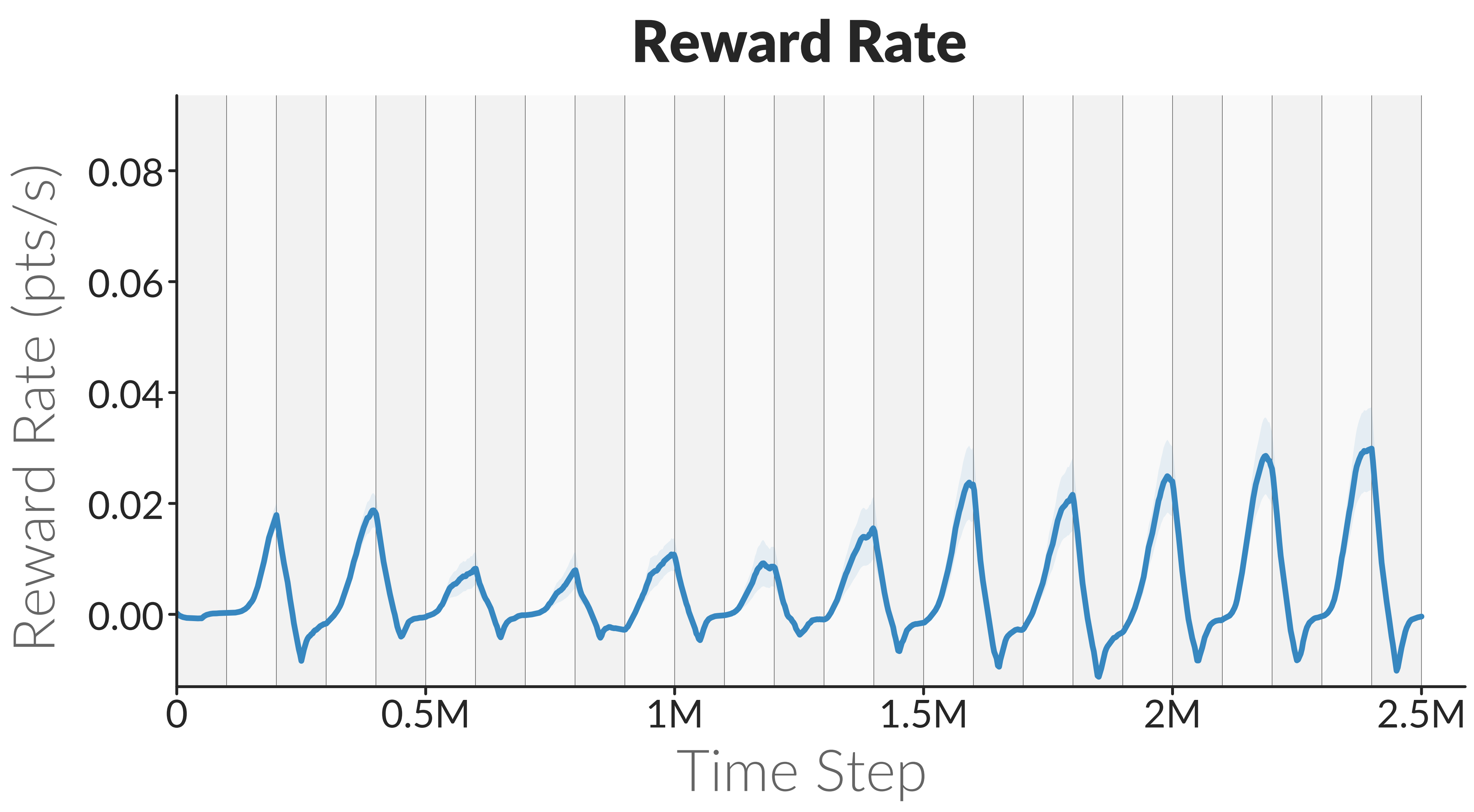
Avoid[**JellyBean**] ∧ Collect[**Onion**]

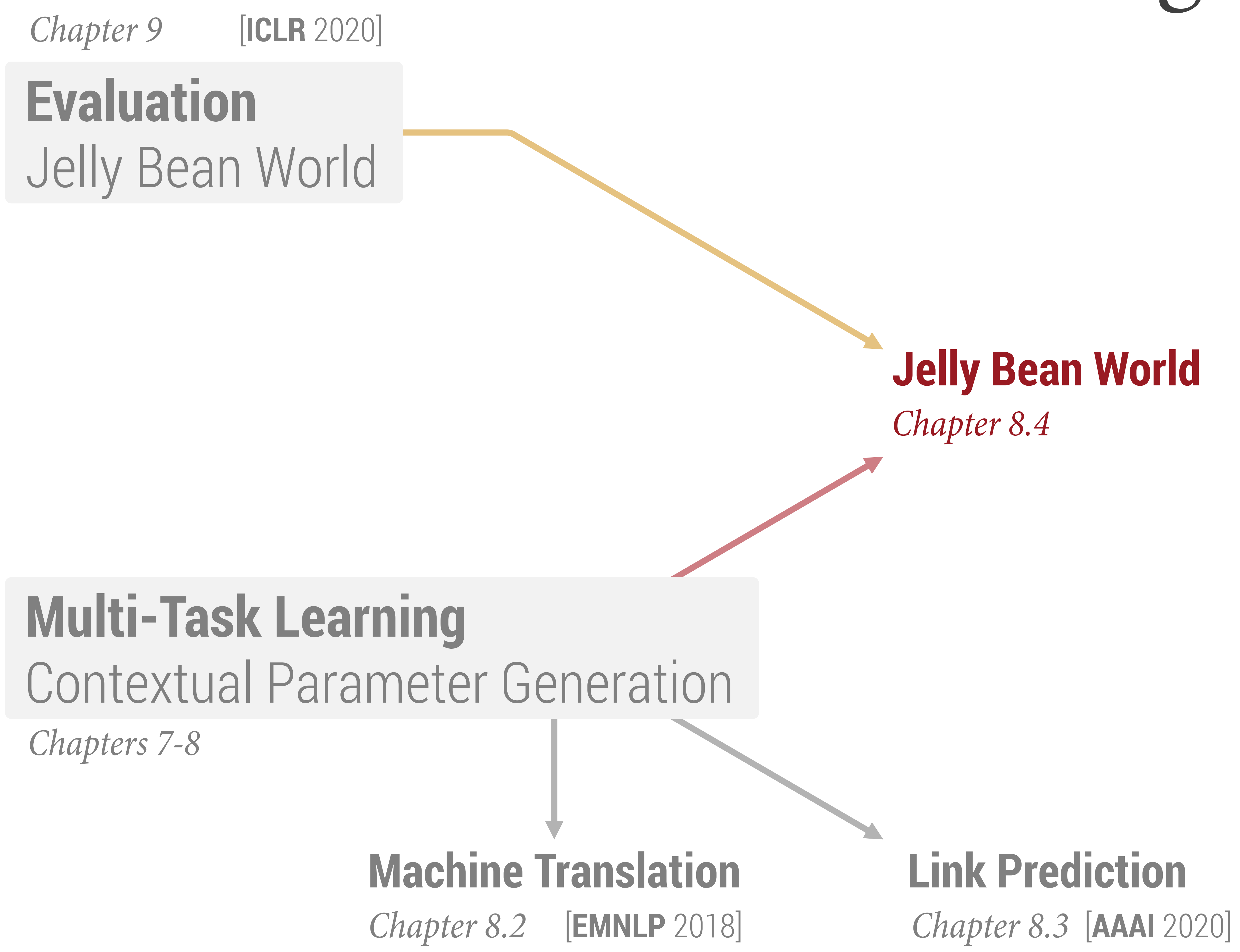




Multi-Task Learning

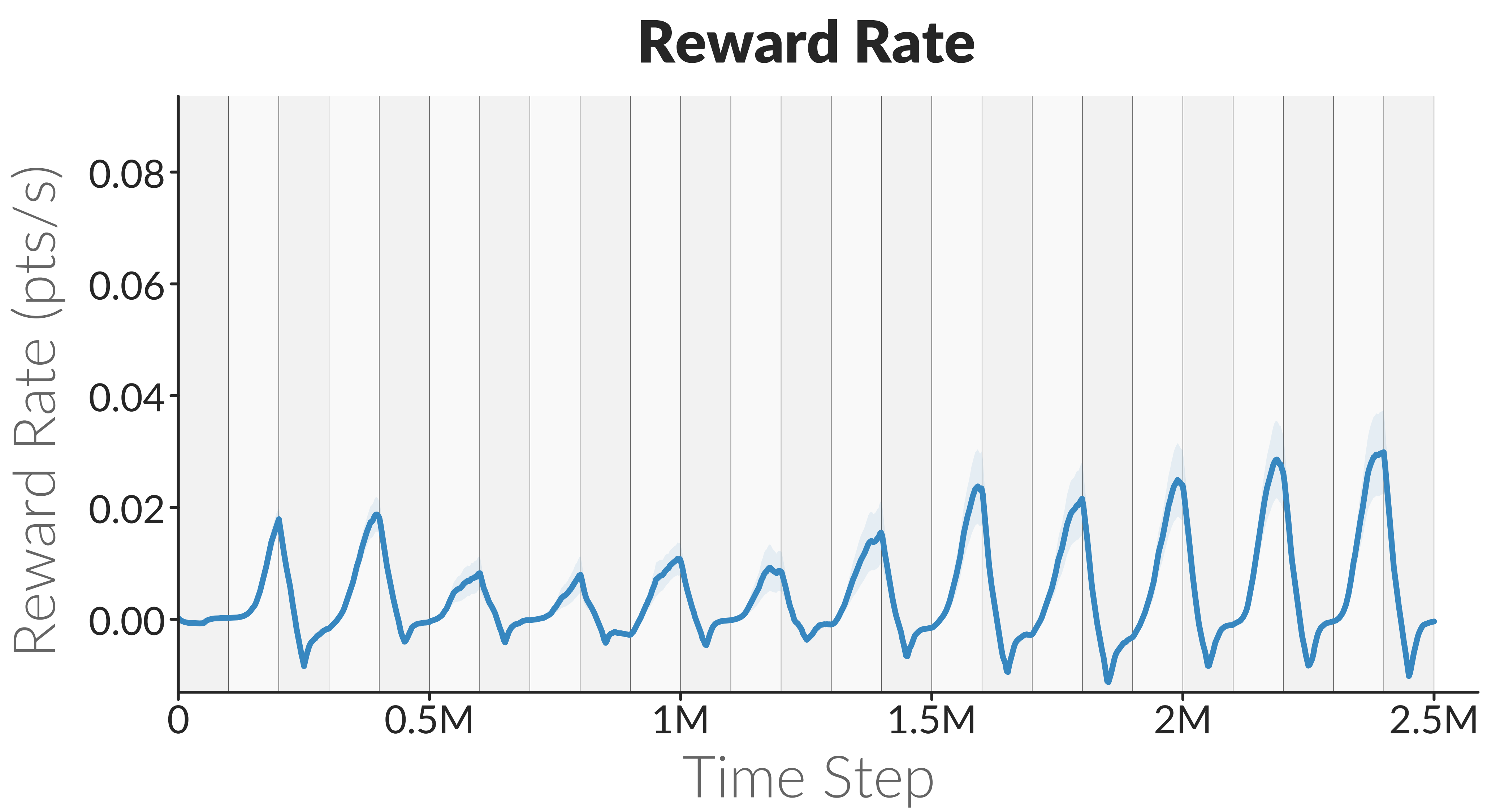
Contextual Parameter Generation for Task Compositions

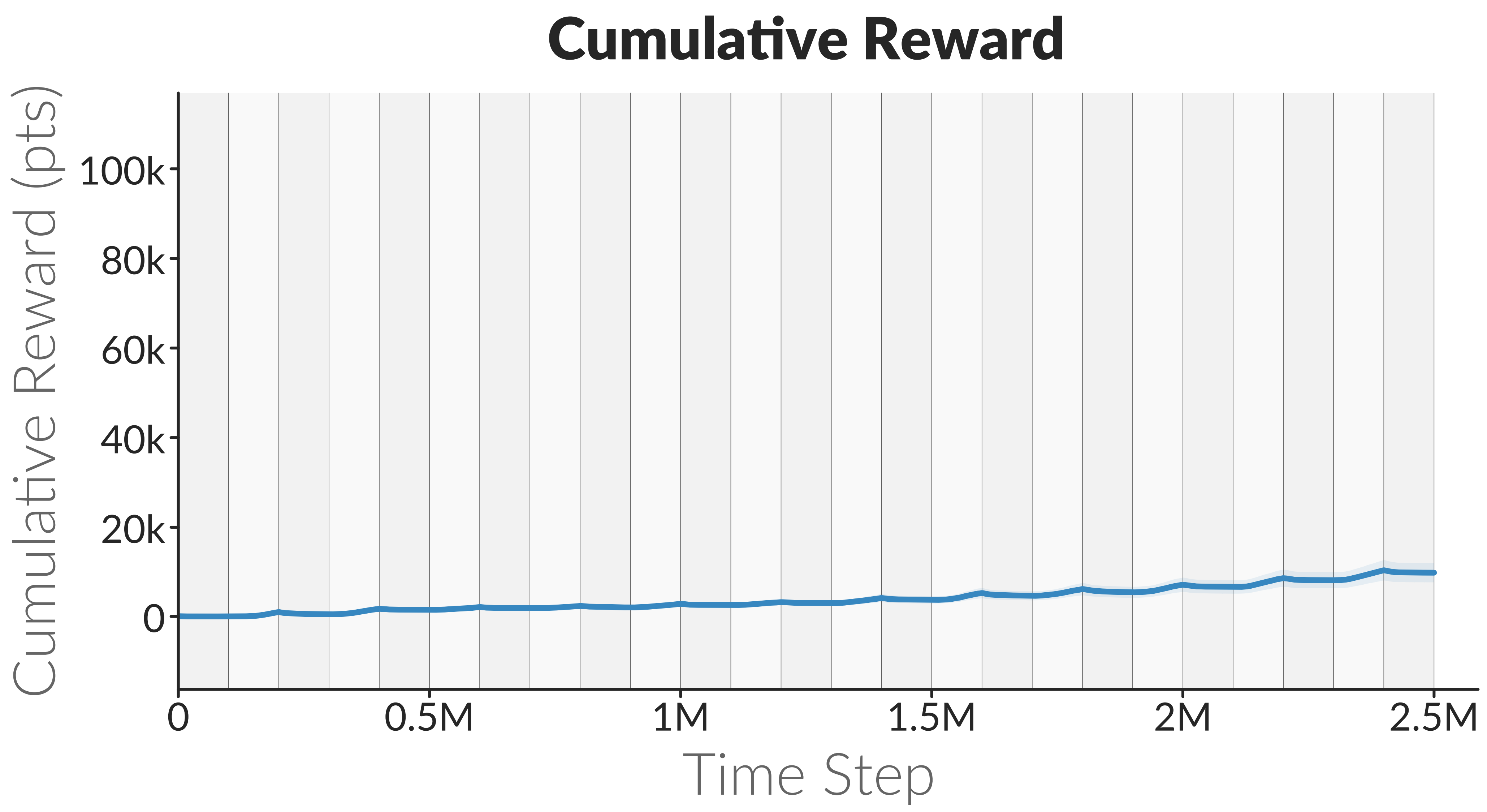


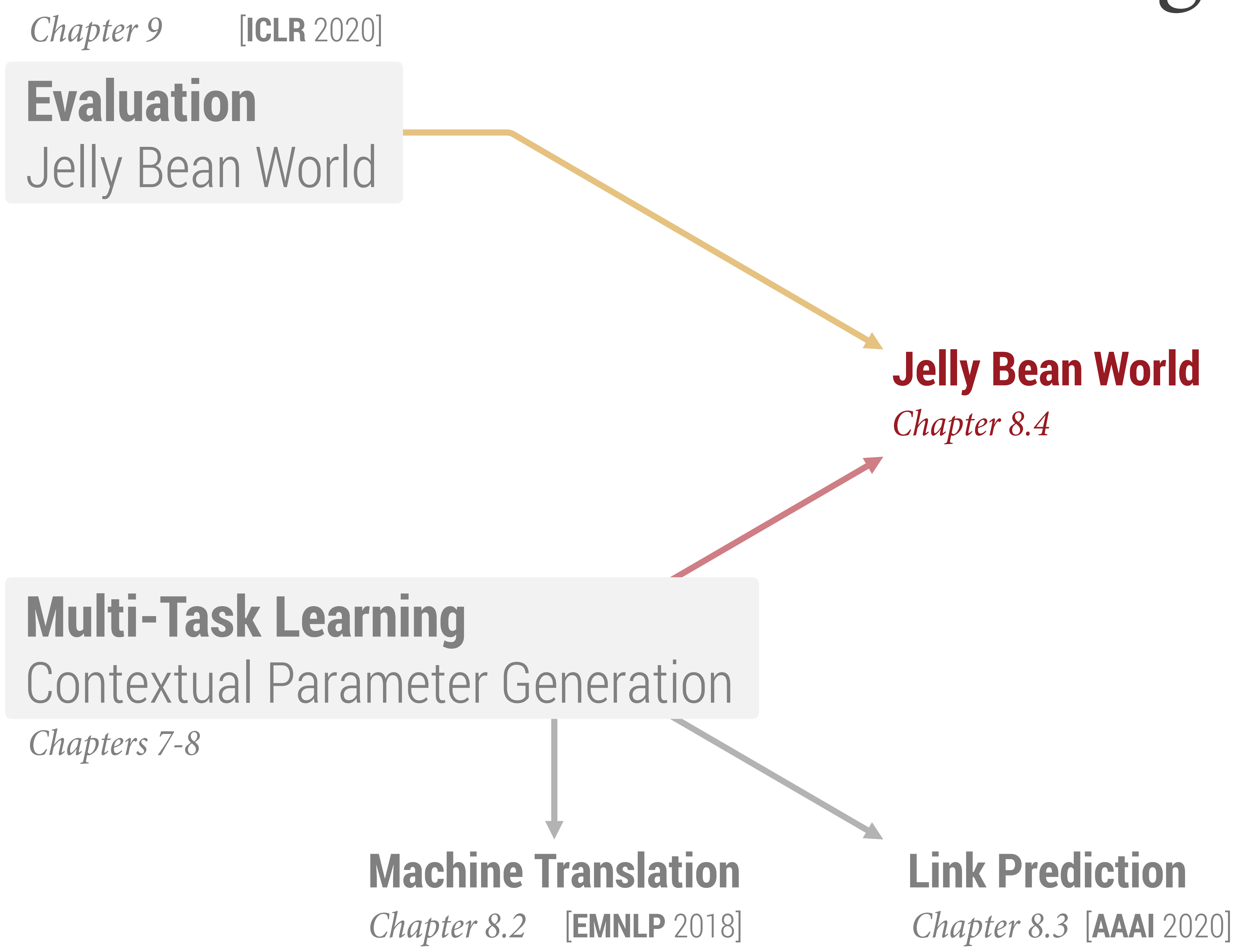


Multi-Task Learning

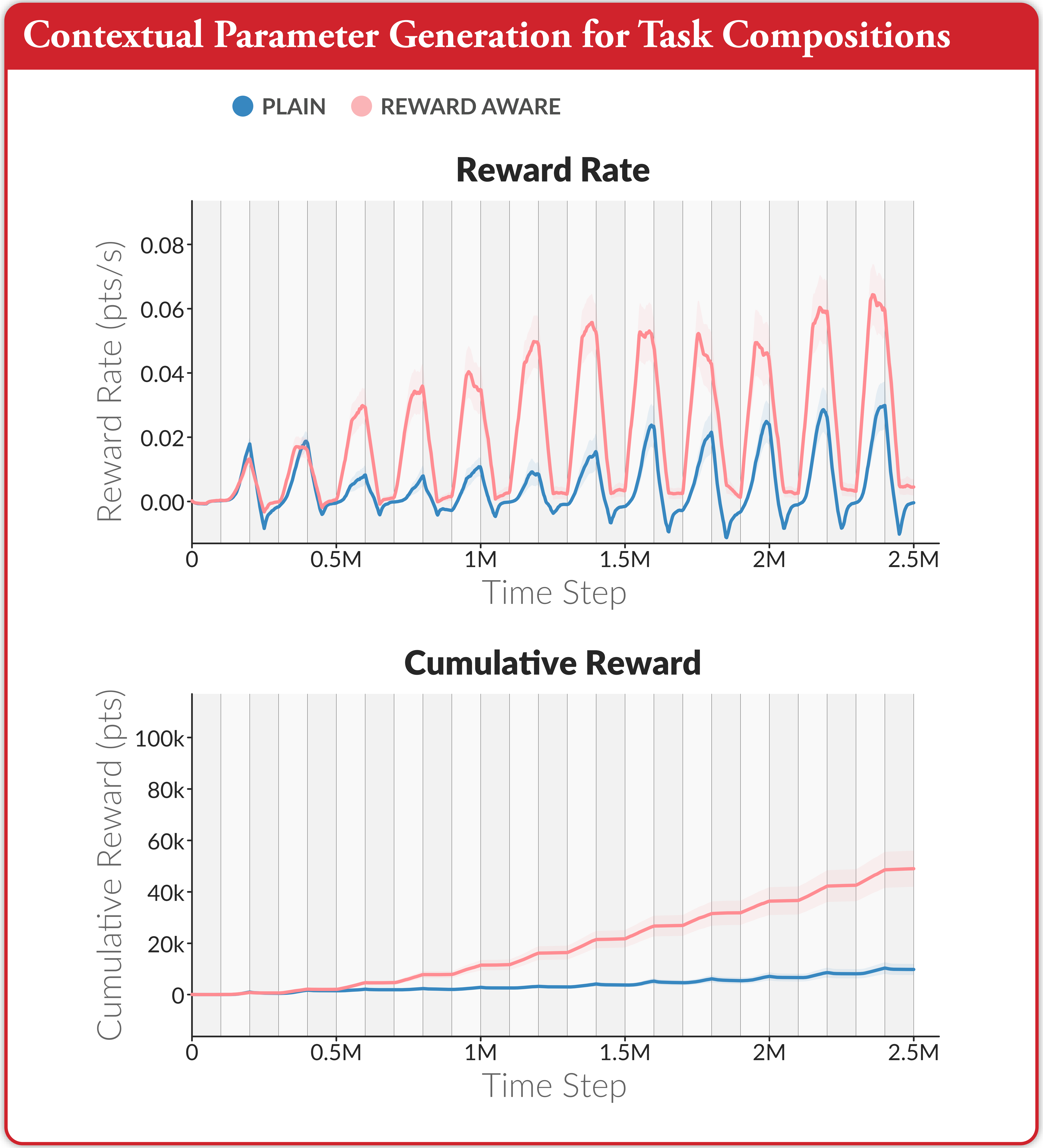
Contextual Parameter Generation for Task Compositions

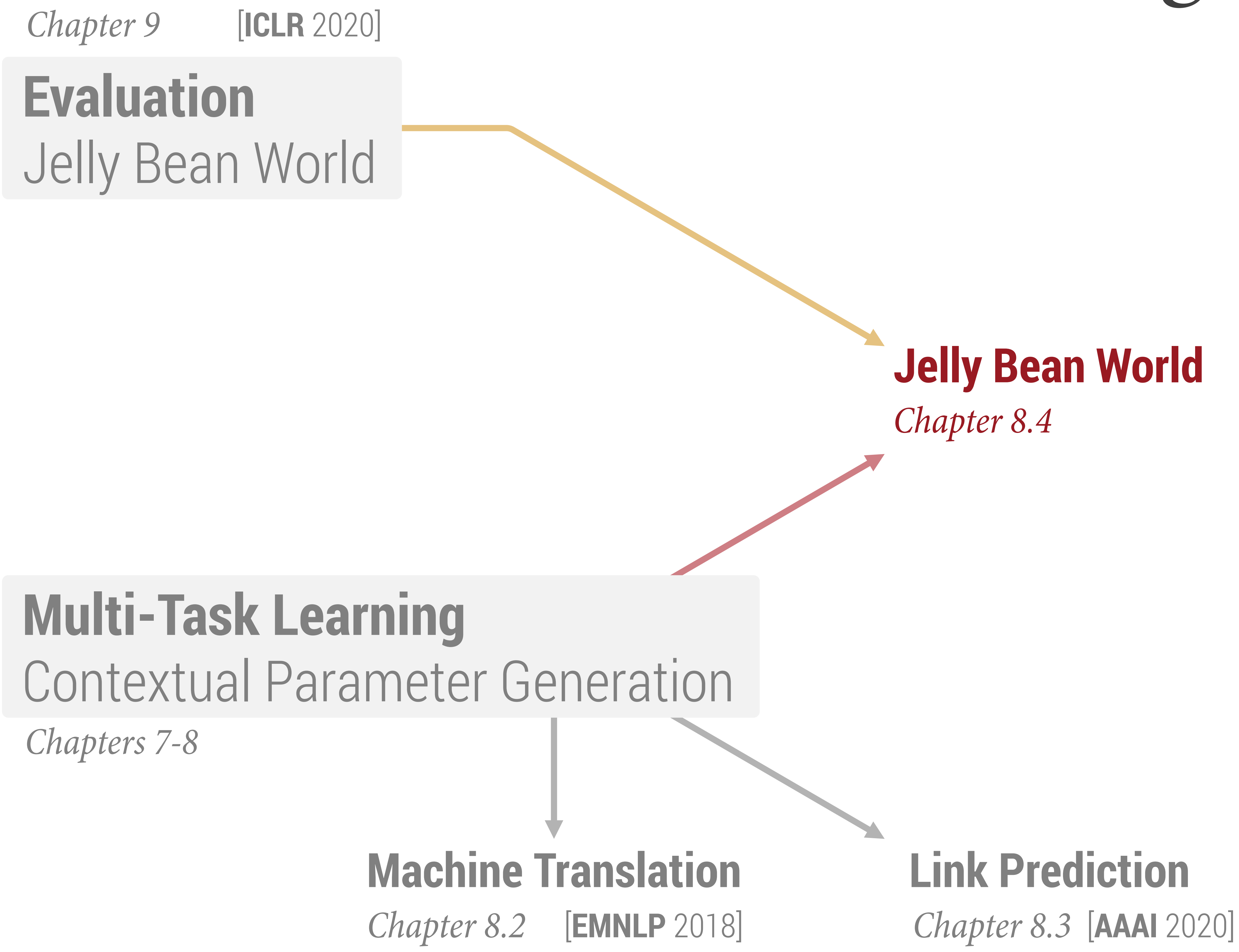






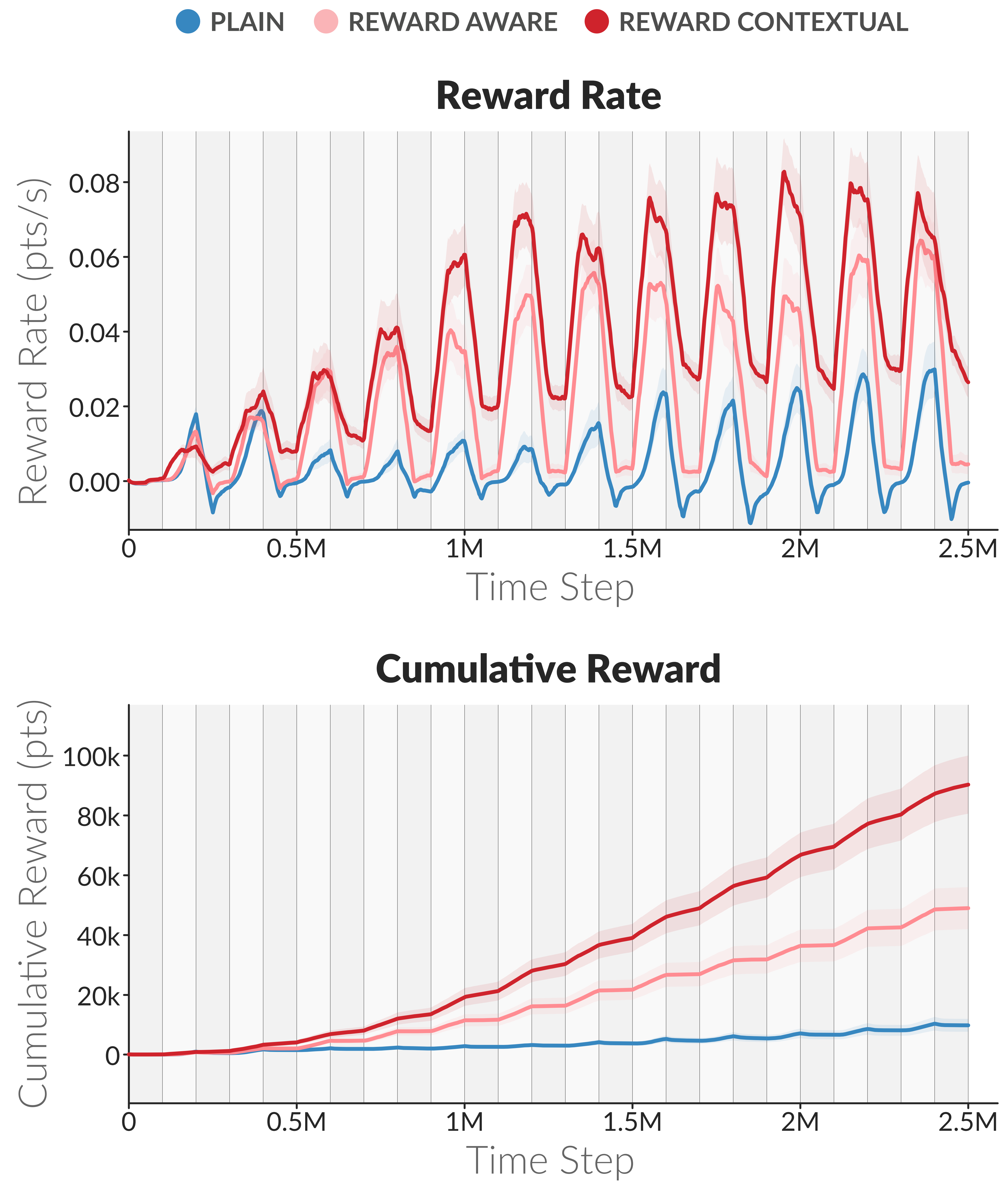
Multi-Task Learning





Multi-Task Learning

Contextual Parameter Generation for Task Compositions



Chapter 9 [**ICLR** 2020]

Evaluation
Jelly Bean World

Jelly Bean World
Chapter 8.4

Multi-Task Learning
Contextual Parameter Generation
Chapters 7-8

Machine Translation
Chapter 8.2 [**EMNLP** 2018]

Link Prediction
Chapter 8.3 [**AAAI** 2020]

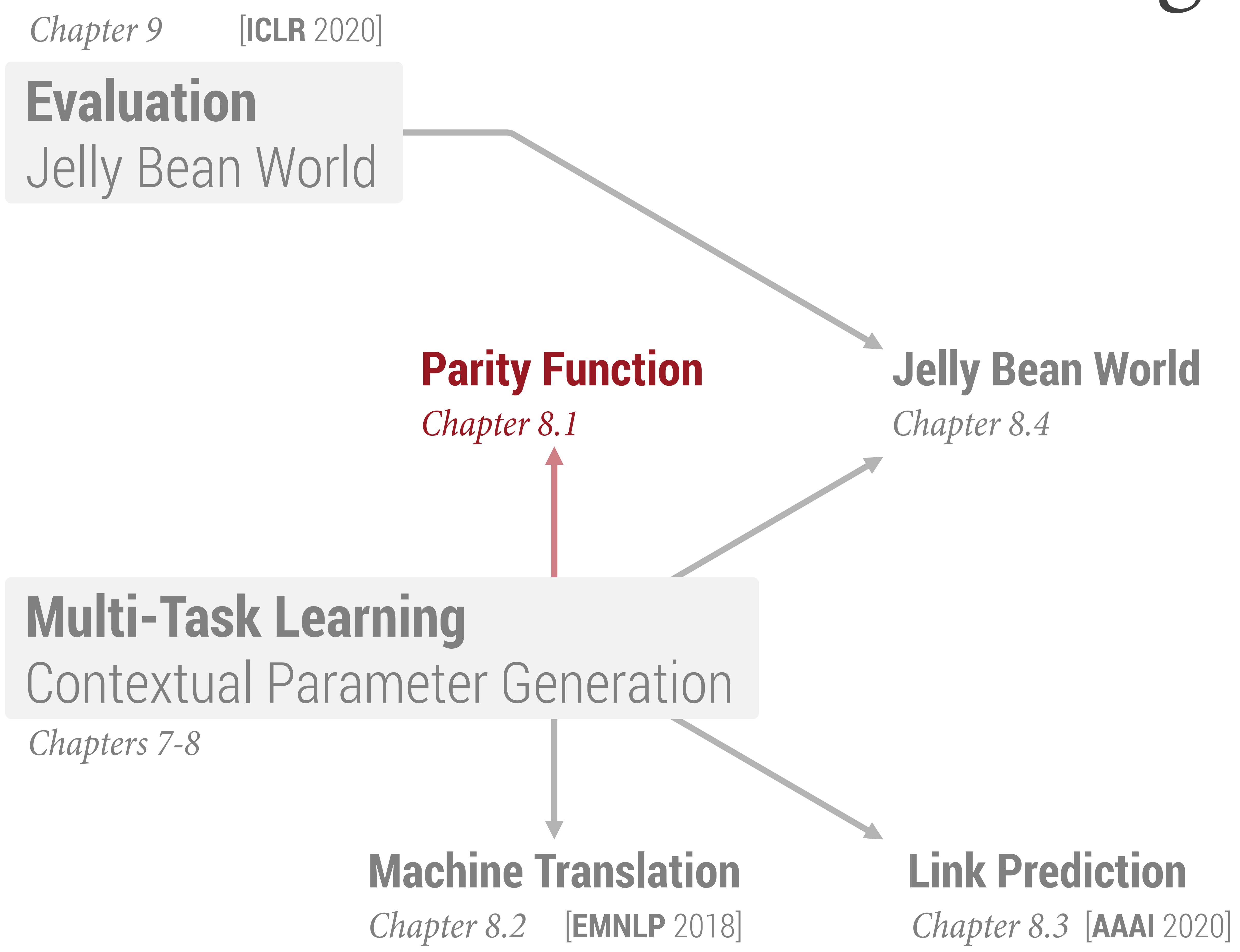
Multi-Task Learning

The Parity Function

Let us consider the following example:

$$p^n(x) = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

XOR operator



Multi-Task Learning

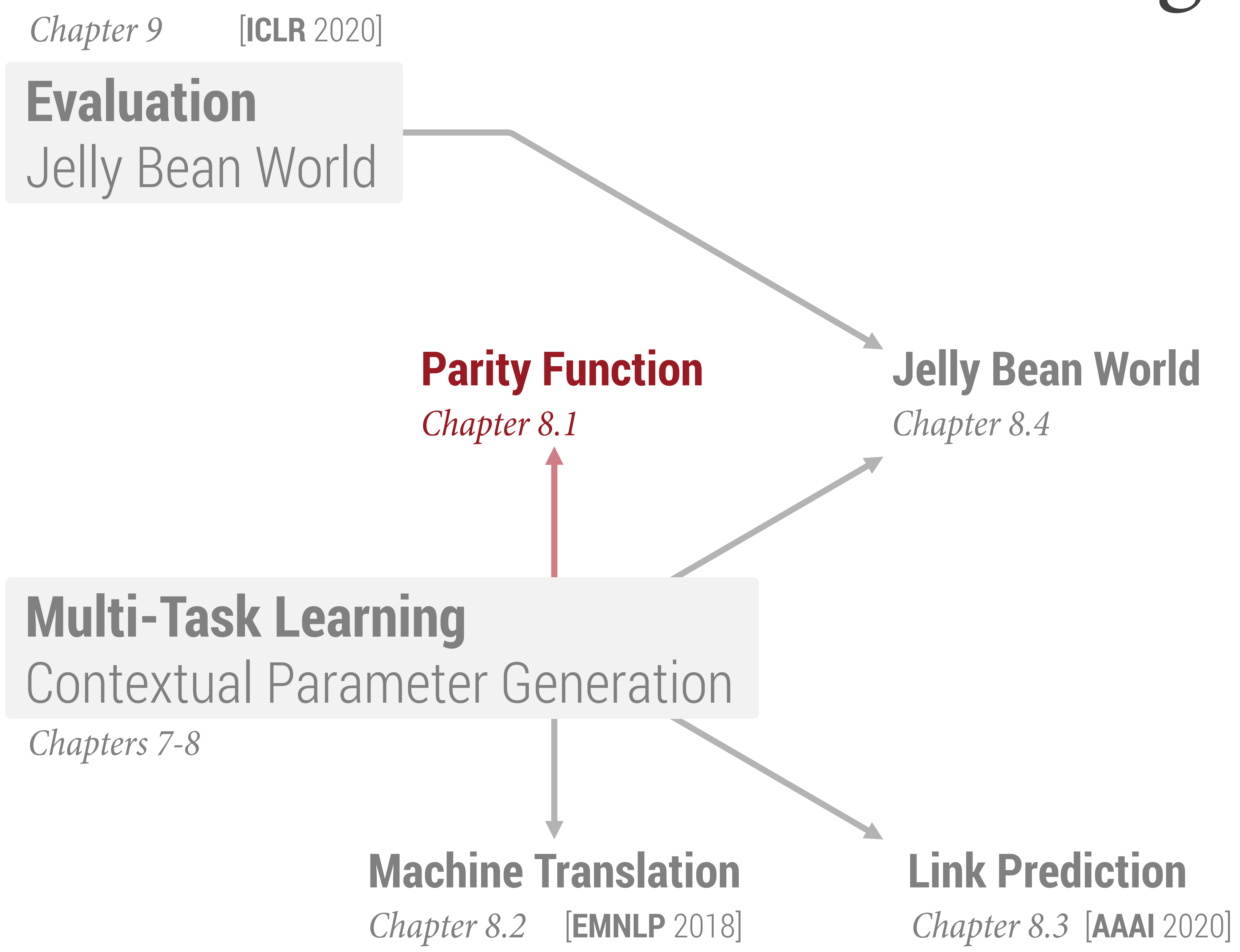
The Parity Function

Let us consider the following example:

$$p^n(x) = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

XOR operator

Train over sequences of *length 1 to 3* and evaluate accuracy over sequences of *length 1 to 100.*



Multi-Task Learning

The Parity Function

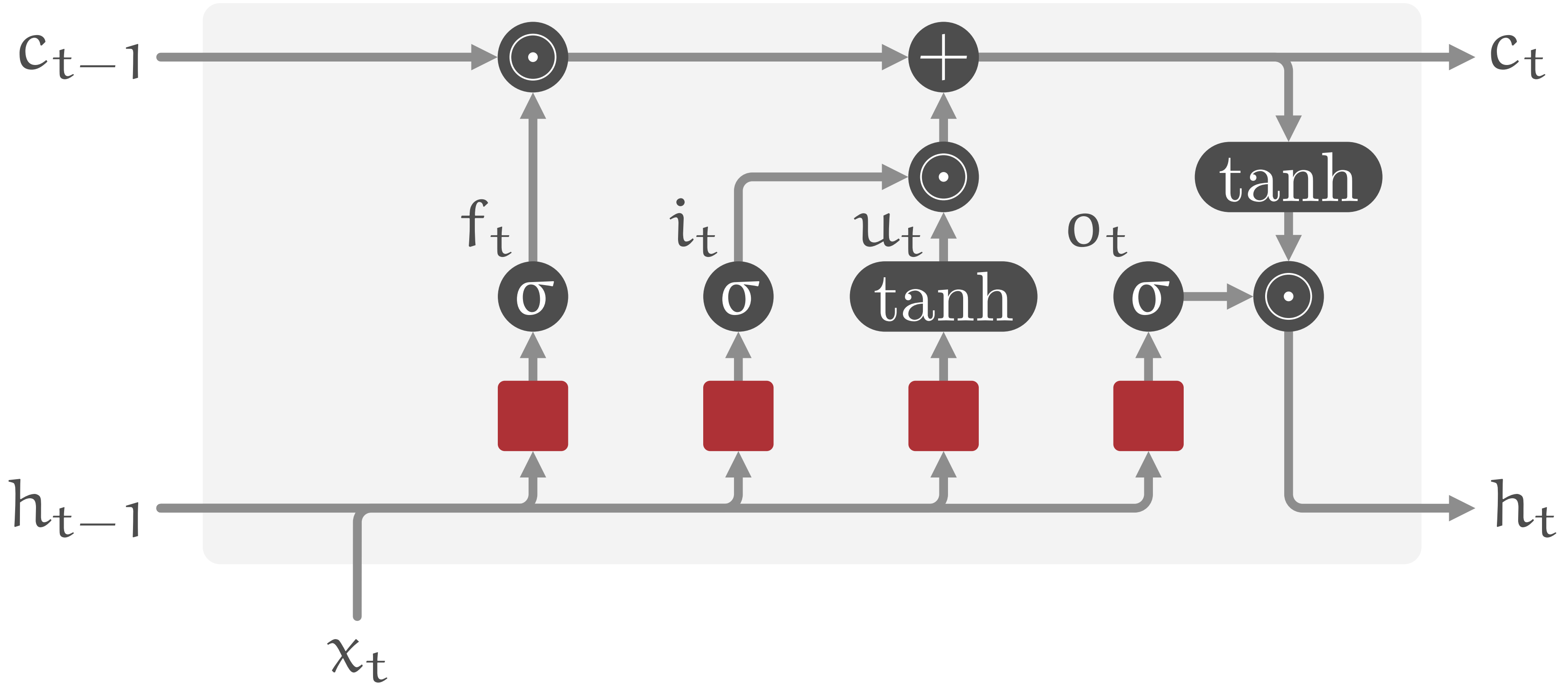
Let us consider the following example:

$$p^n(x) = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

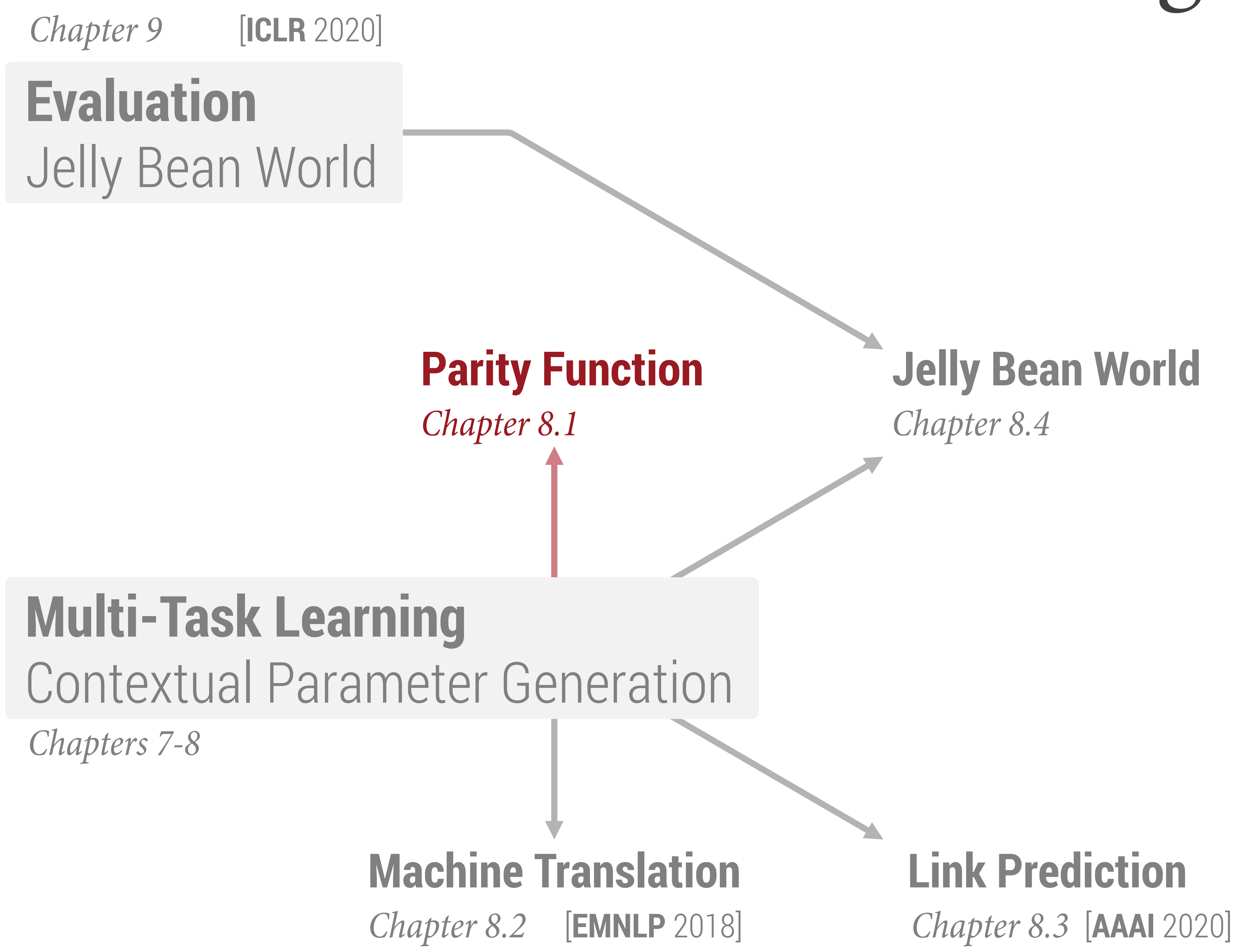
XOR operator

Train over sequences of *length 1 to 3* and evaluate accuracy over sequences of *length 1 to 100.*

Long Short-Term Memory (LSTM) network:



Linear learnable function



Multi-Task Learning

The Parity Function

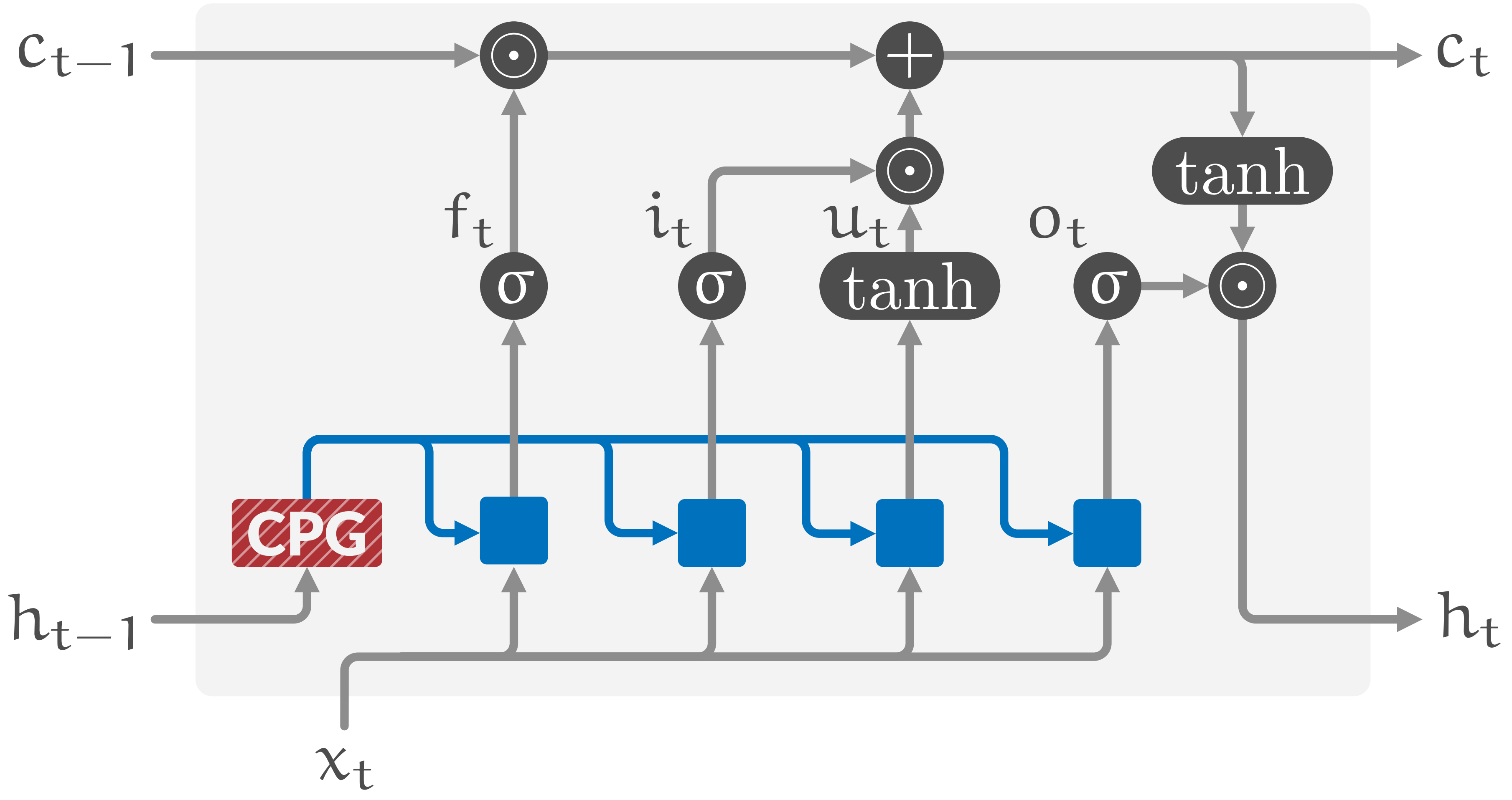
Let us consider the following example:

$$p^n(x) = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

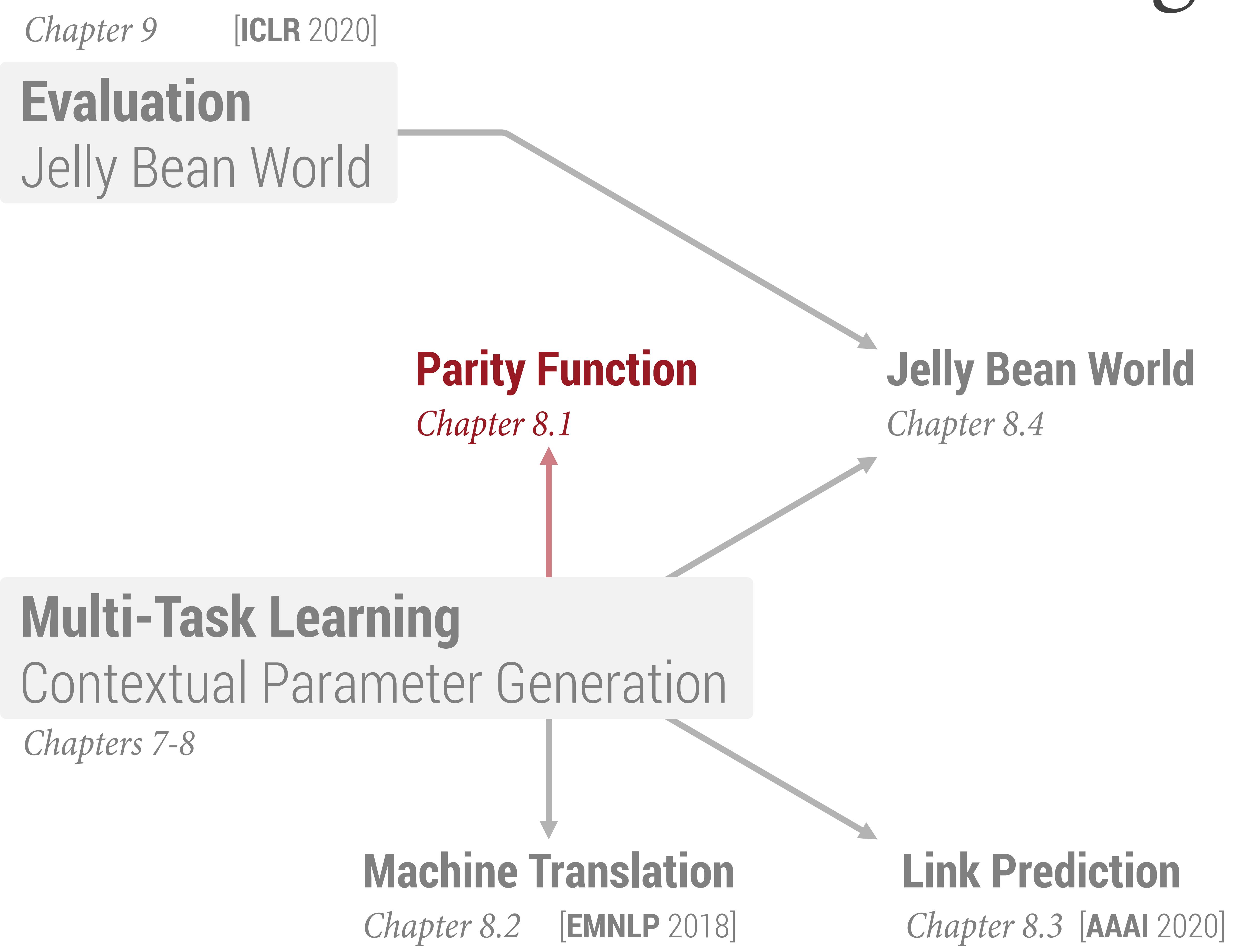
XOR operator

Train over sequences of *length 1 to 3* and evaluate accuracy over sequences of *length 1 to 100.*

Contextual Long Short-Term Memory (LSTM) network:



Linear function whose parameters are generated
Learnable parameter generator



Multi-Task Learning

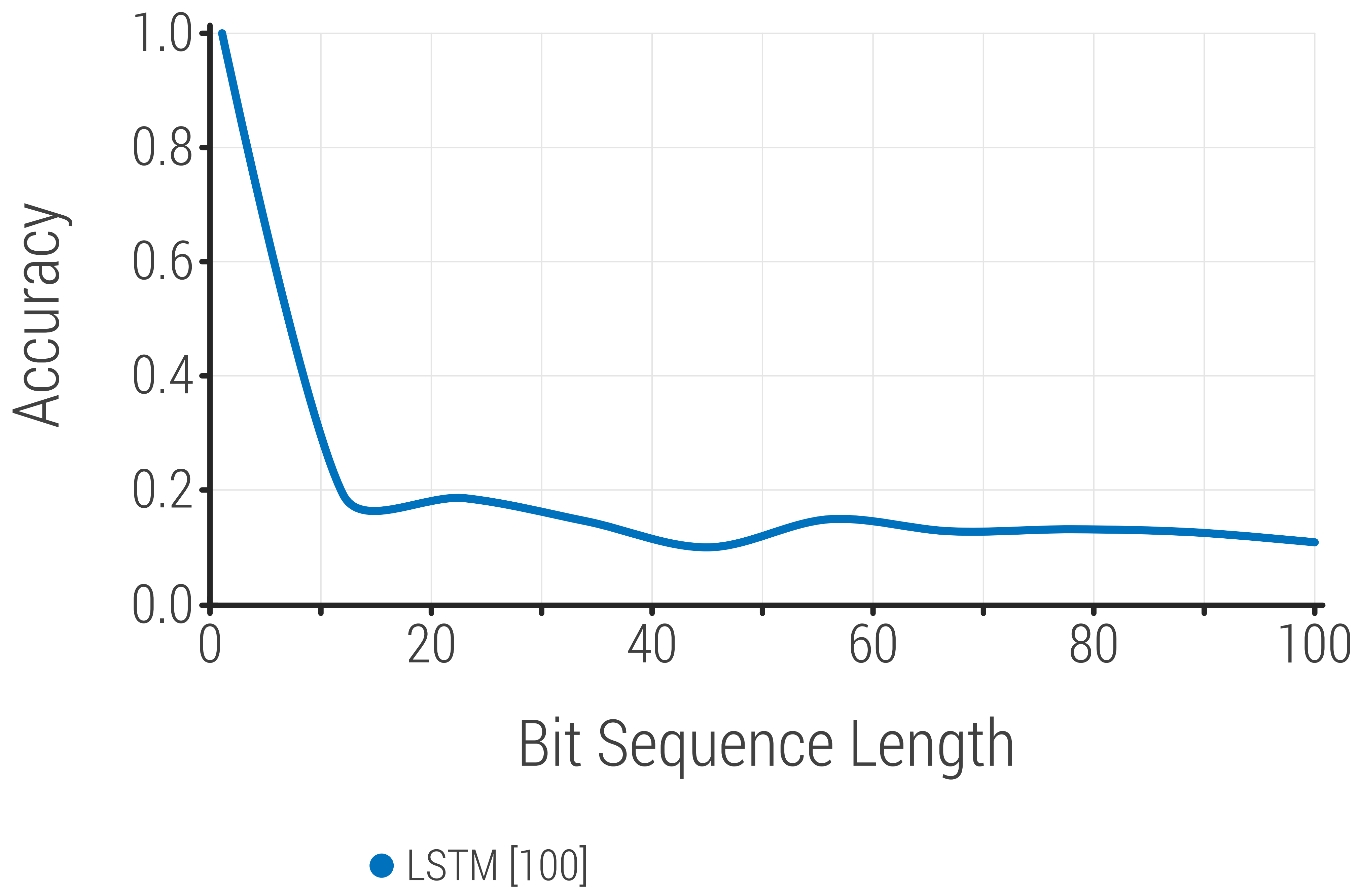
The Parity Function

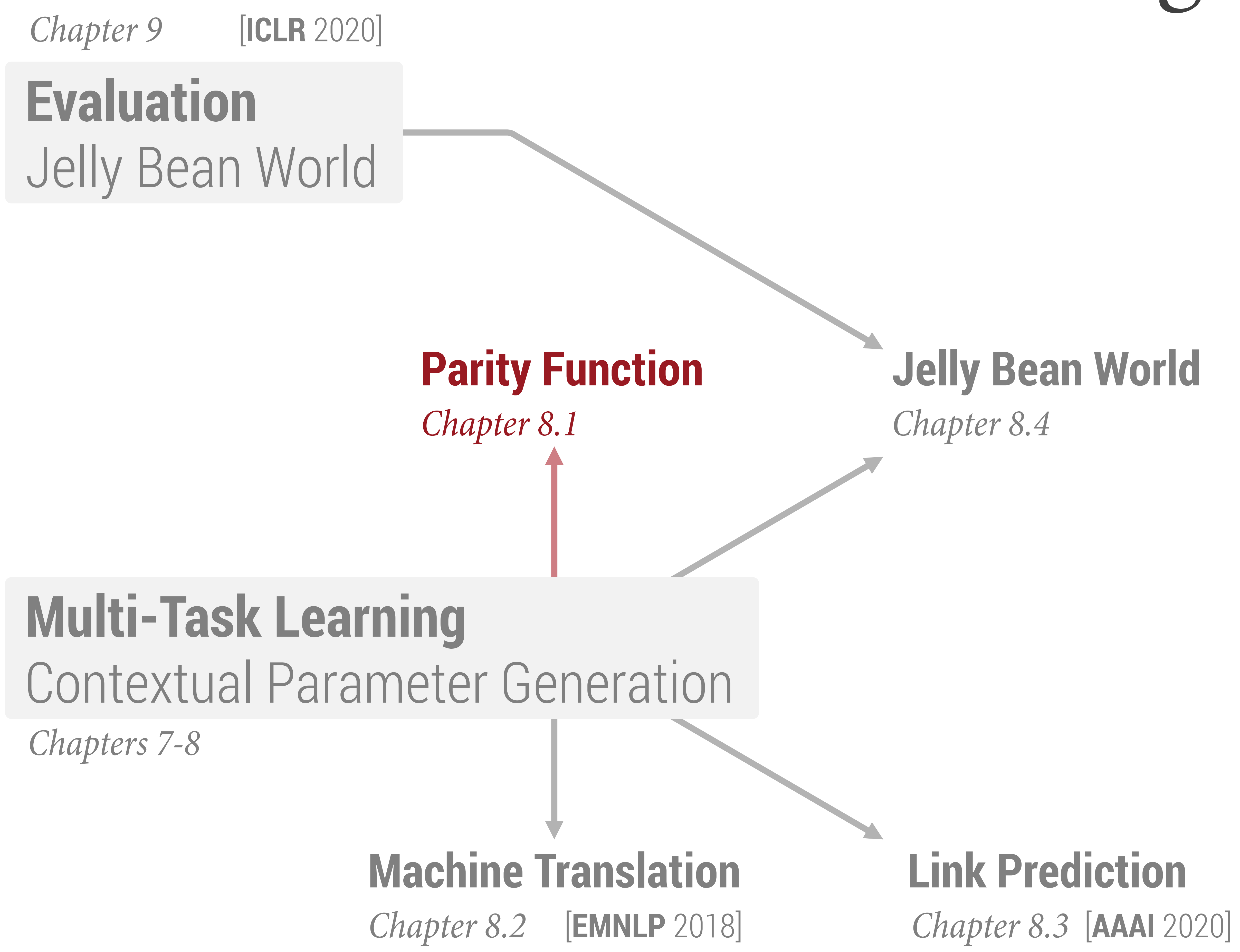
Let us consider the following example:

$$p^n(x) = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

XOR operator

Train over sequences of *length 1 to 3* and evaluate accuracy over sequences of *length 1 to 100.*





Multi-Task Learning

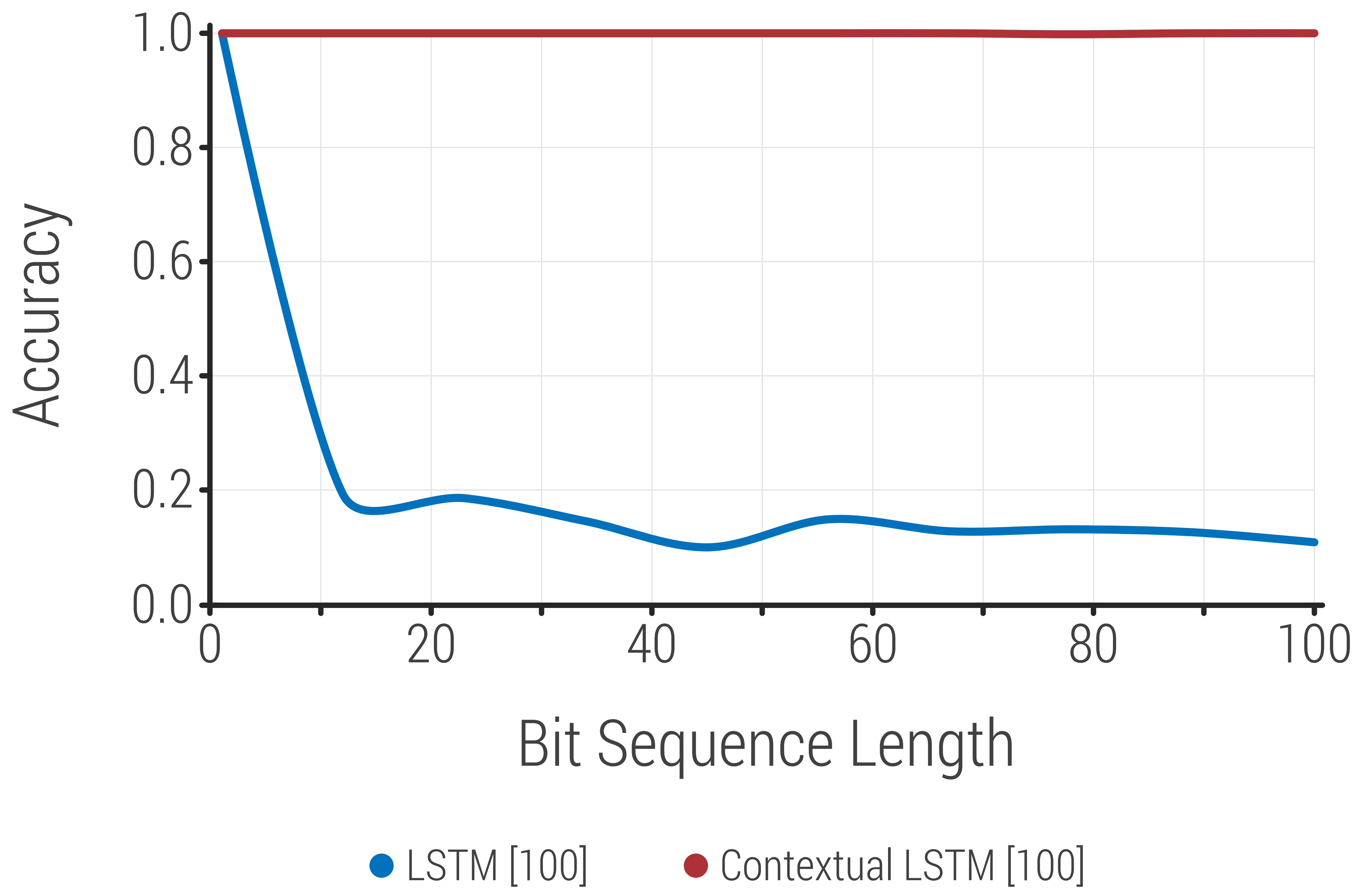
The Parity Function

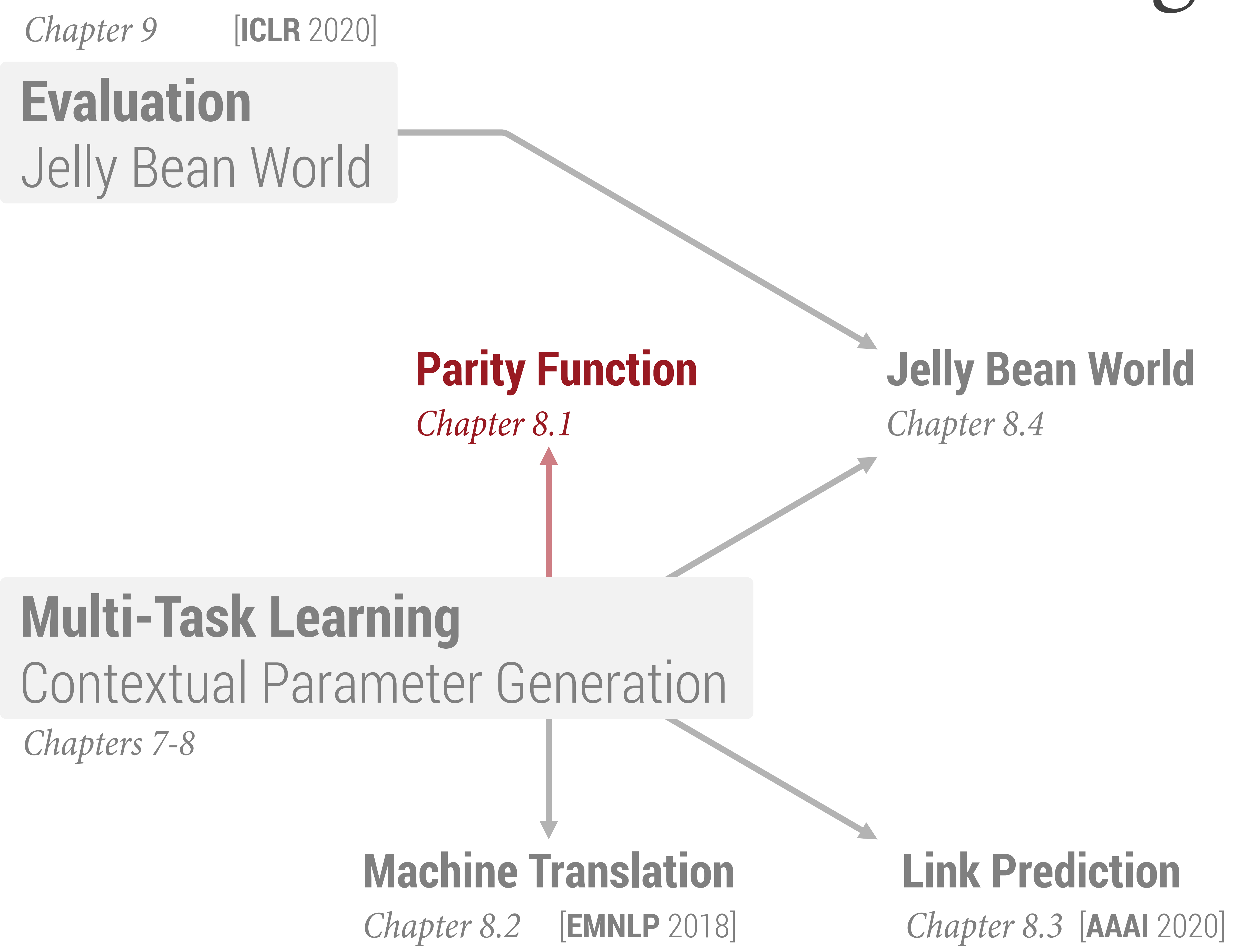
Let us consider the following example:

$$p^n(x) = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

XOR operator

Train over sequences of *length 1 to 3* and evaluate accuracy over sequences of *length 1 to 100.*





Multi-Task Learning

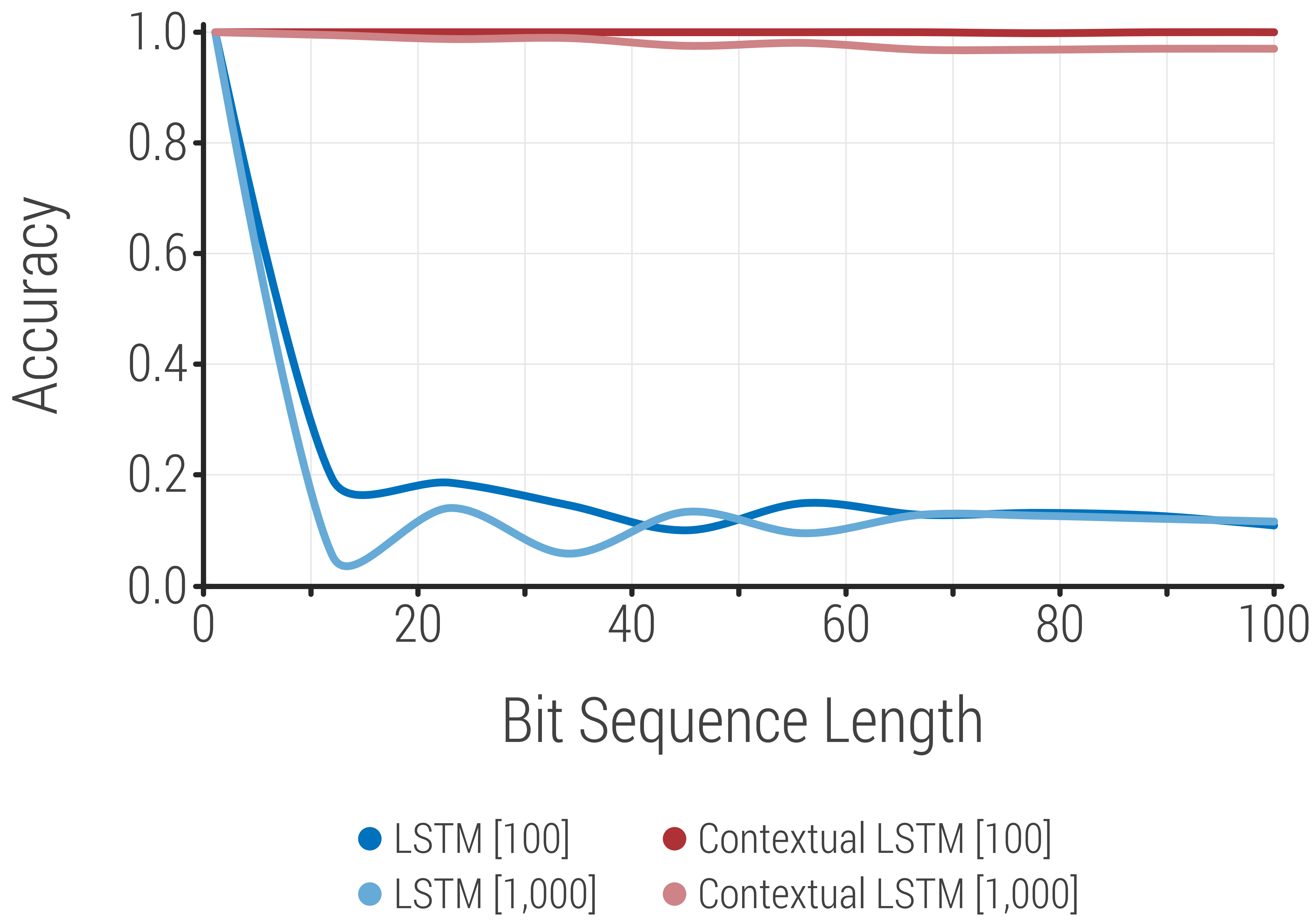
The Parity Function

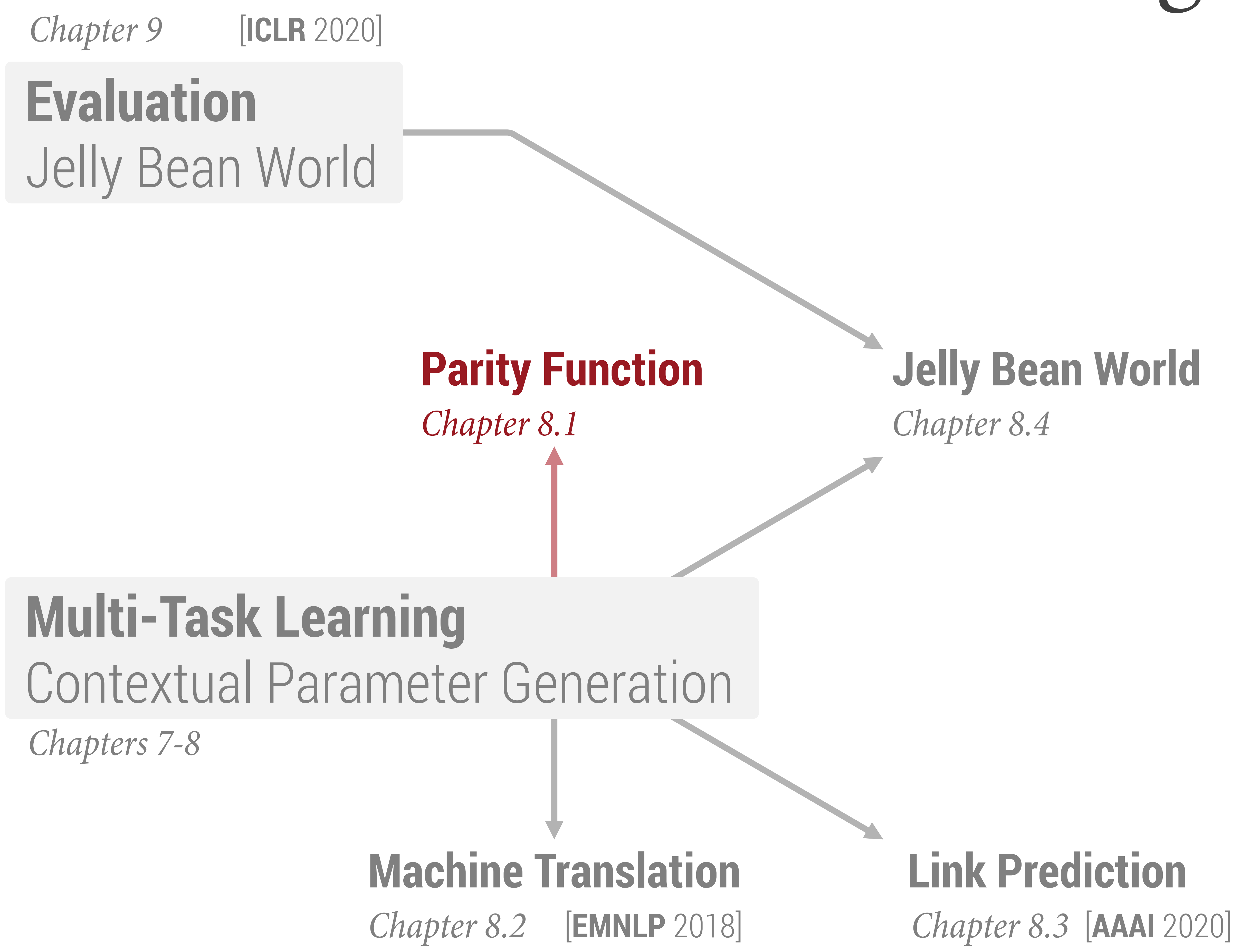
Let us consider the following example:

$$p^n(x) = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

XOR operator

Train over sequences of *length 1 to 3* and evaluate accuracy over sequences of *length 1 to 100*.





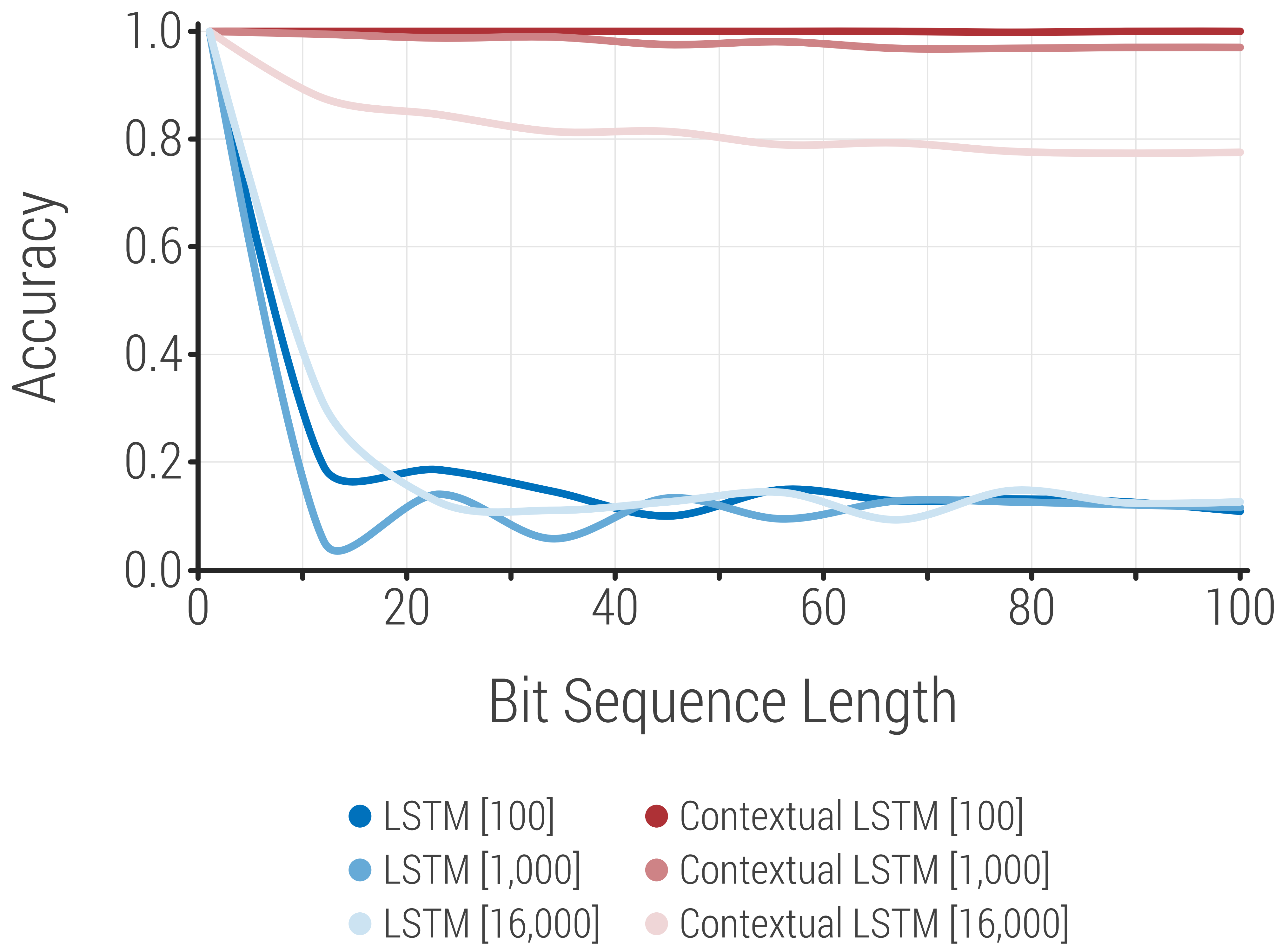
The Parity Function

Let us consider the following example:

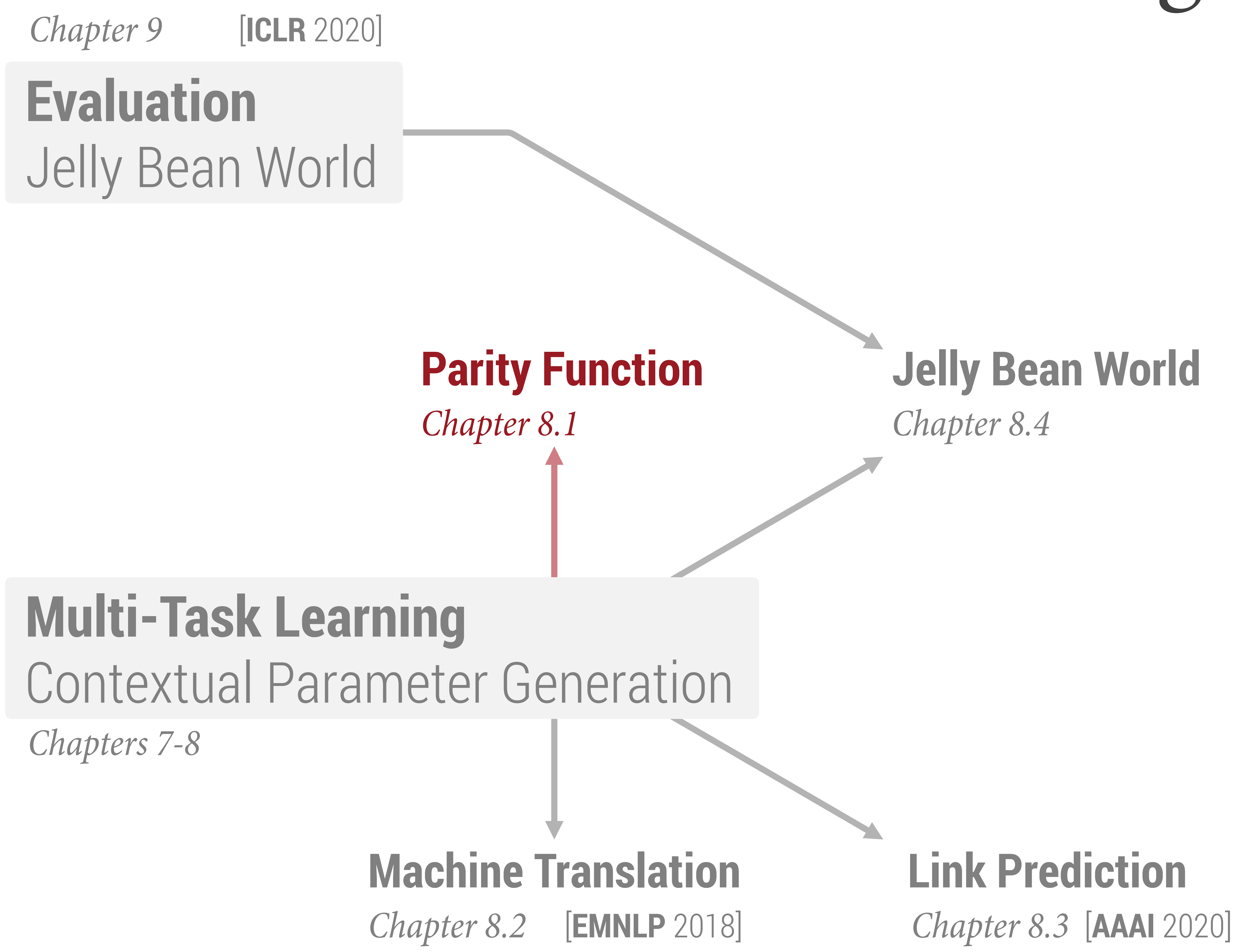
$$p^n(x) = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

XOR operator

Train over sequences of *length 1 to 3* and evaluate accuracy over sequences of *length 1 to 100.*



Multi-Task Learning



The Parity Function

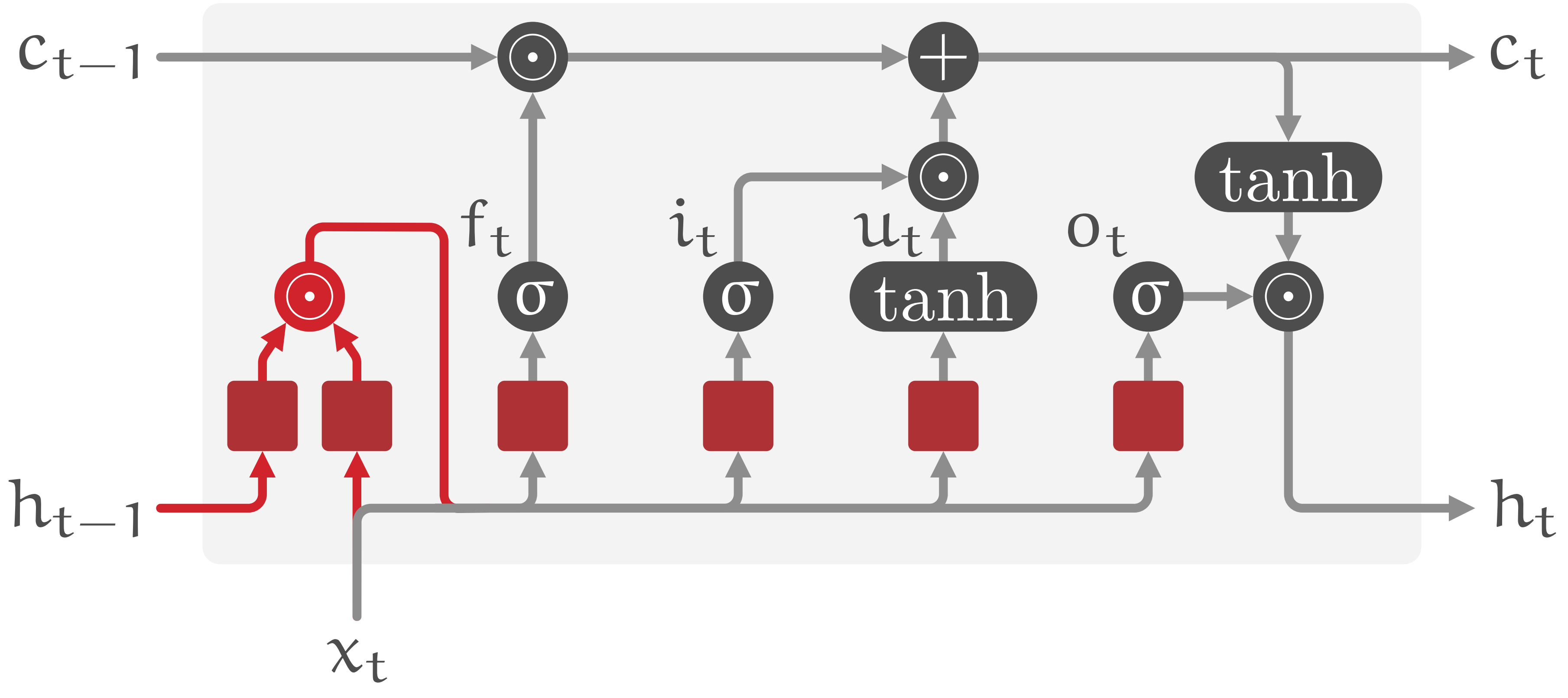
Let us consider the following example:

$$p^n(x) = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

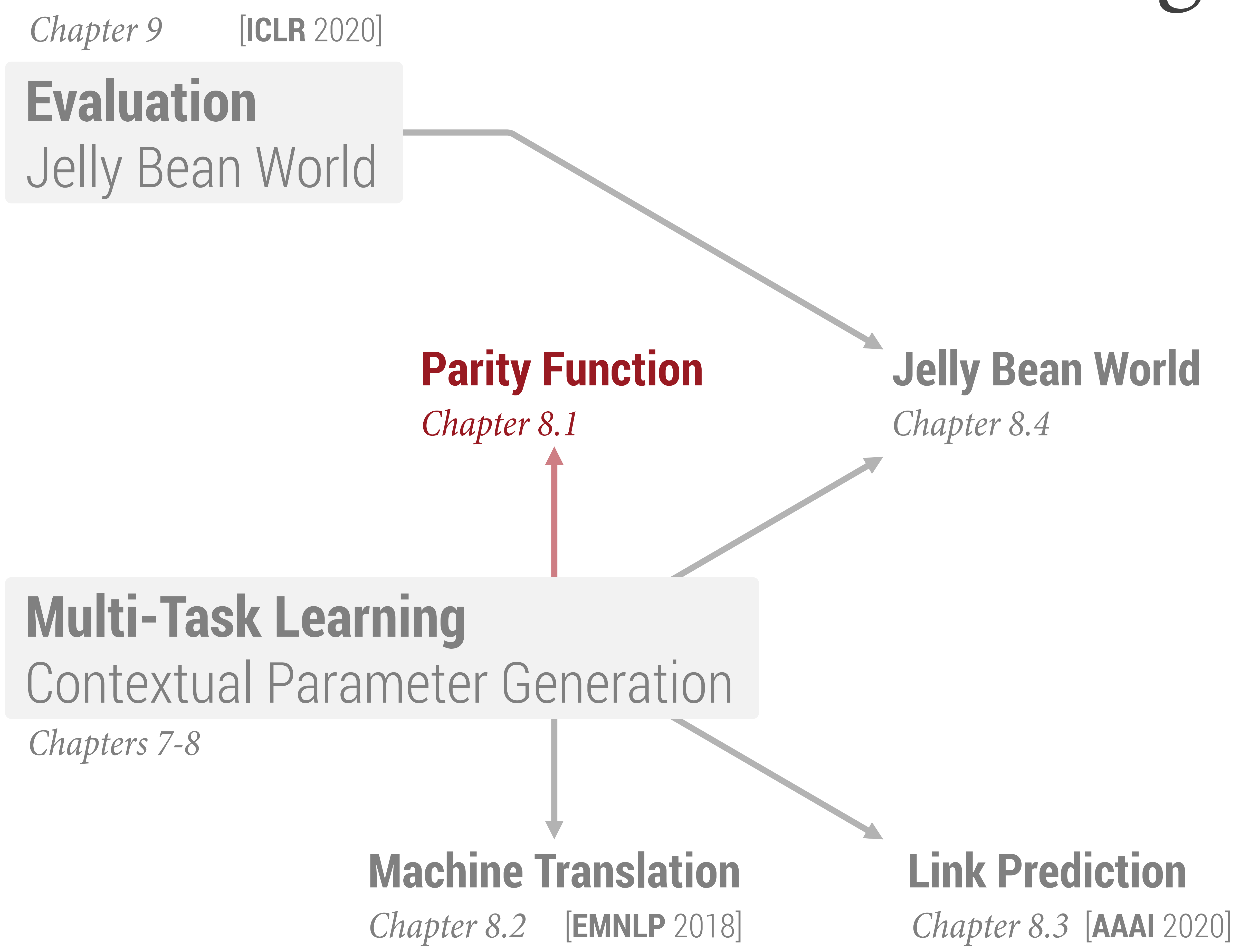
XOR operator

Train over sequences of *length 1 to 3* and evaluate accuracy over sequences of *length 1 to 100.*

Multiplicative Long Short-Term Memory (LSTM) network:



Linear learnable function



Multi-Task Learning

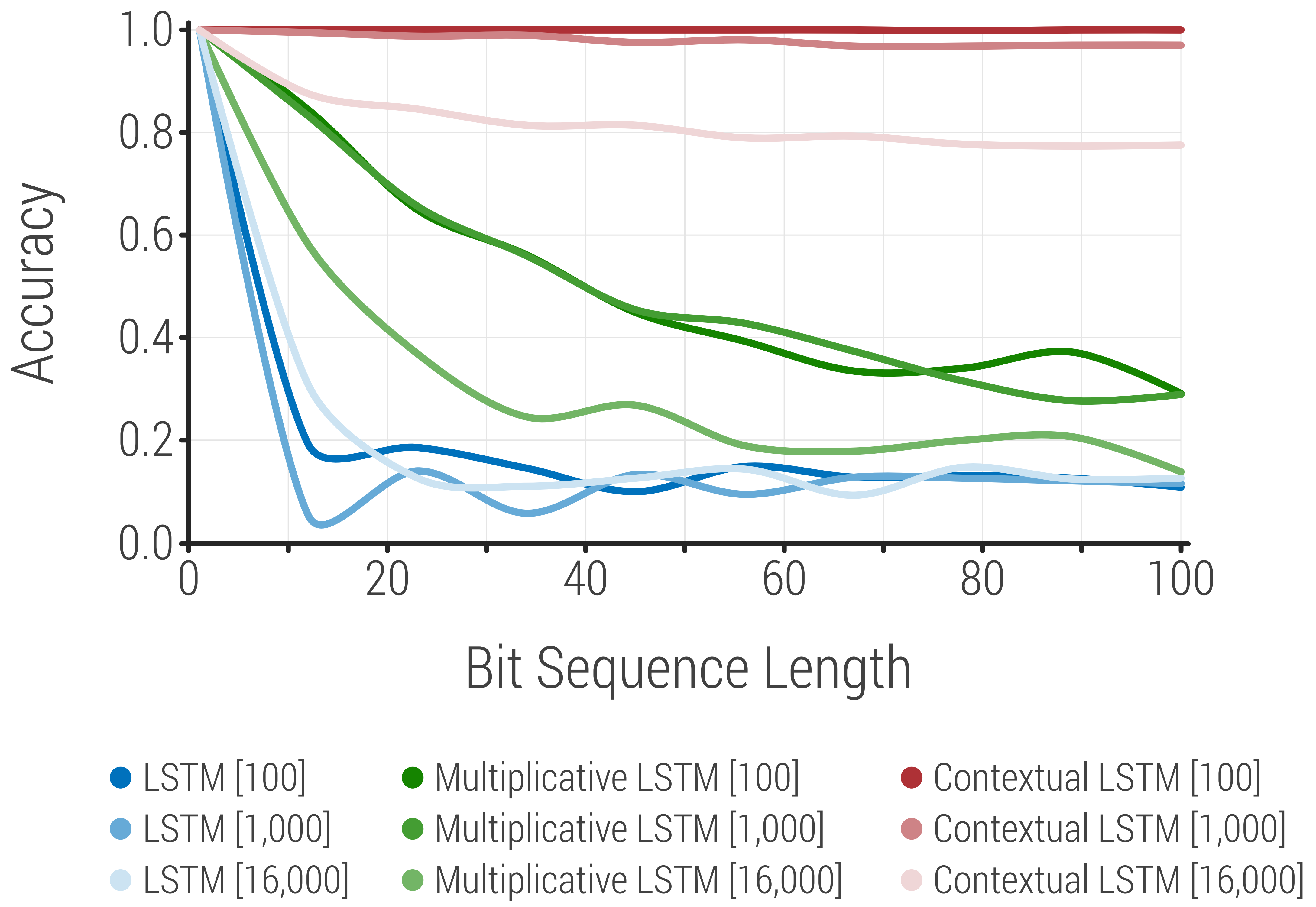
The Parity Function

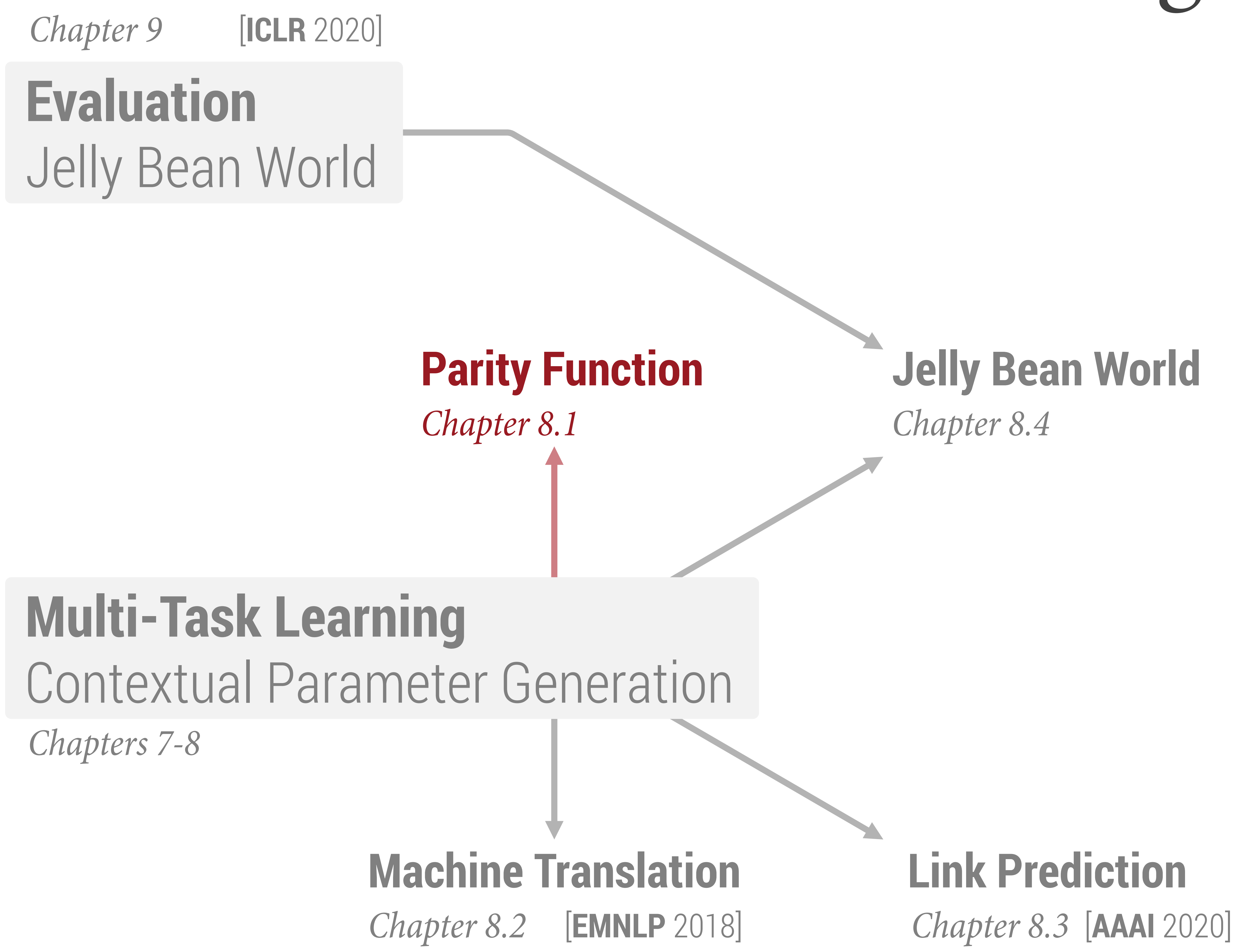
Let us consider the following example:

$$p^n(x) = x_1 \oplus x_2 \oplus \cdots \oplus x_n$$

XOR operator

Train over sequences of *length 1 to 3* and evaluate accuracy over sequences of *length 1 to 100.*





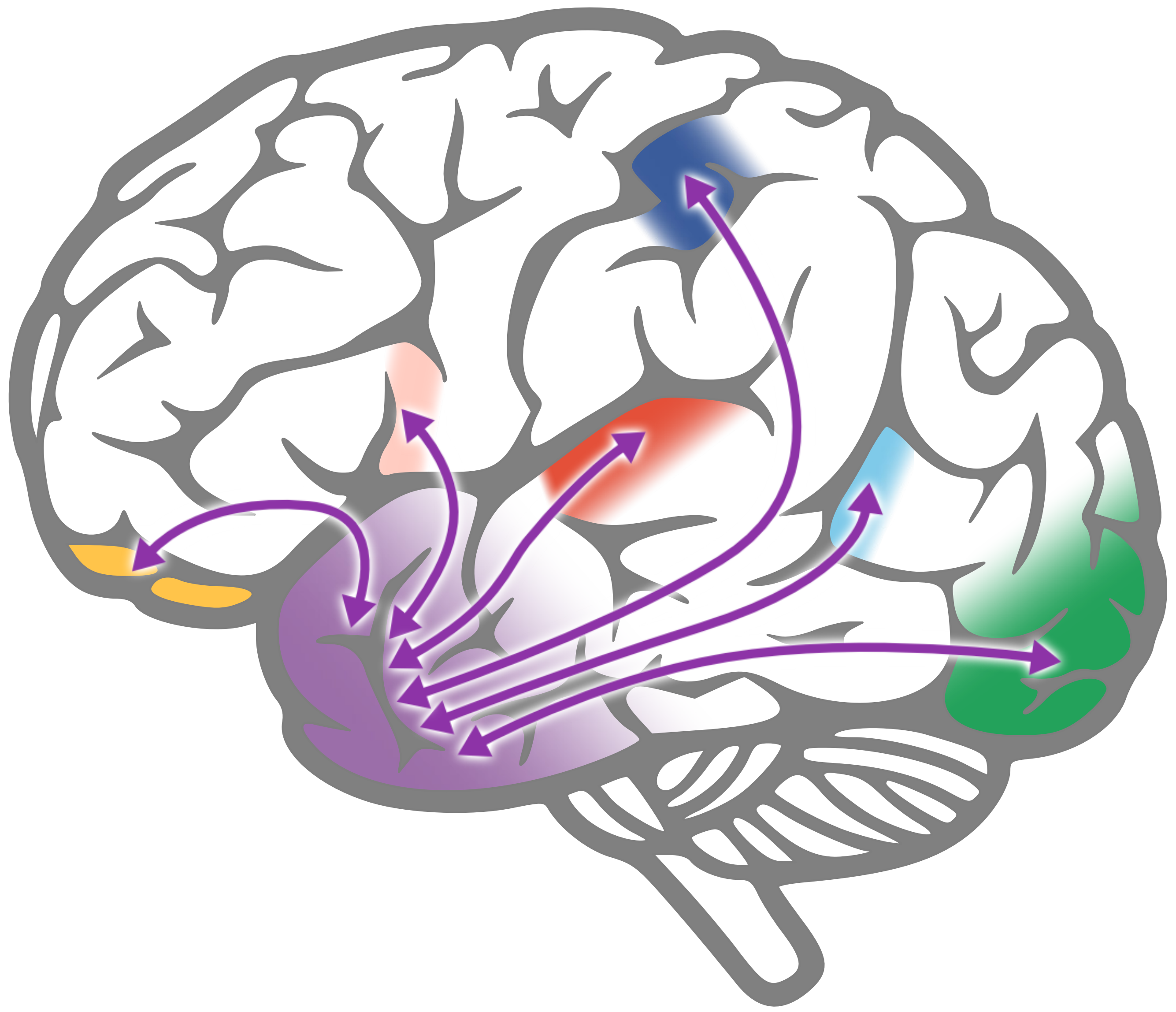
Unified Architecture

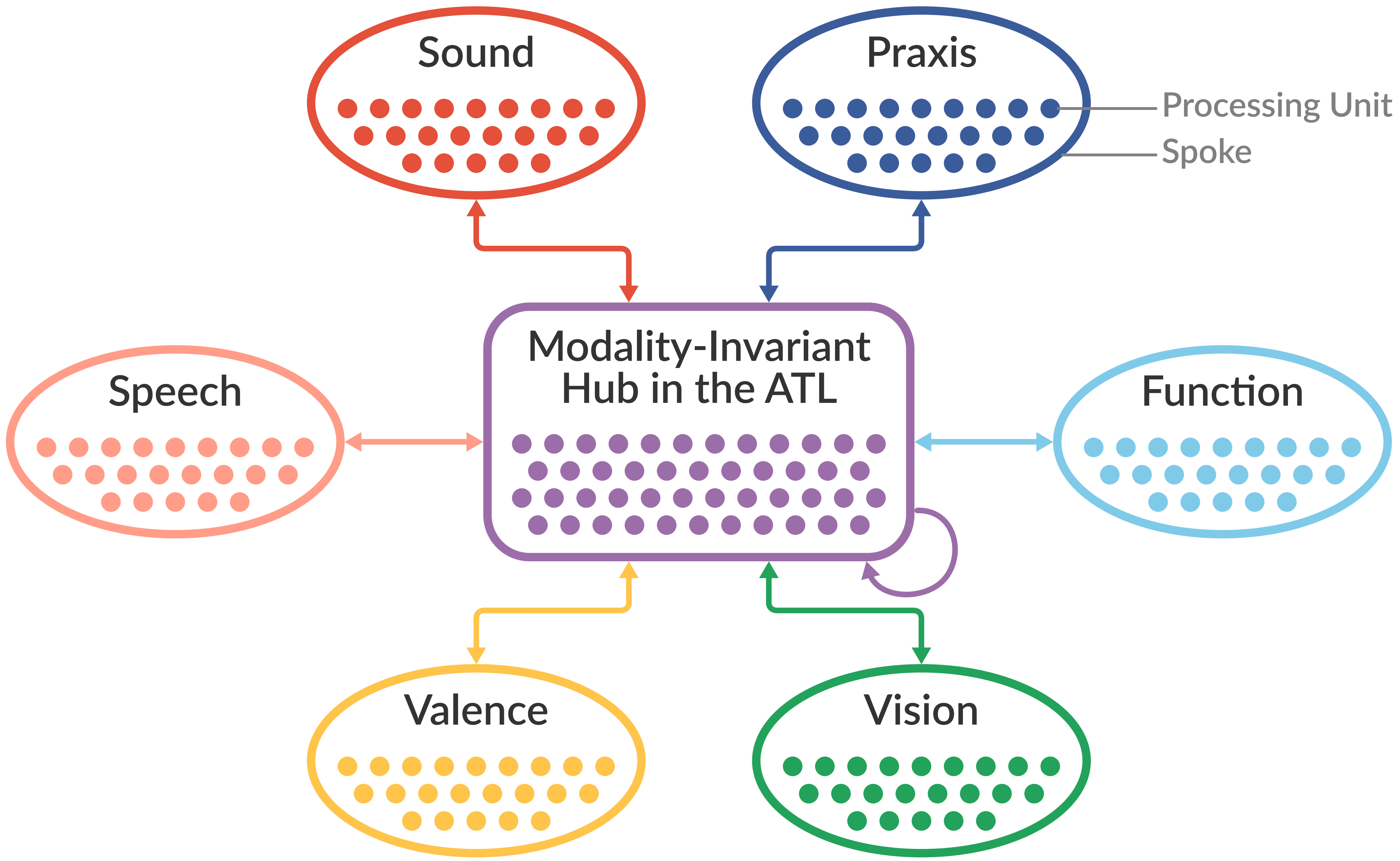
How do all the pieces fit together in the *human brain*?

Unified Architecture

How do all the pieces fit together in the *human brain*?

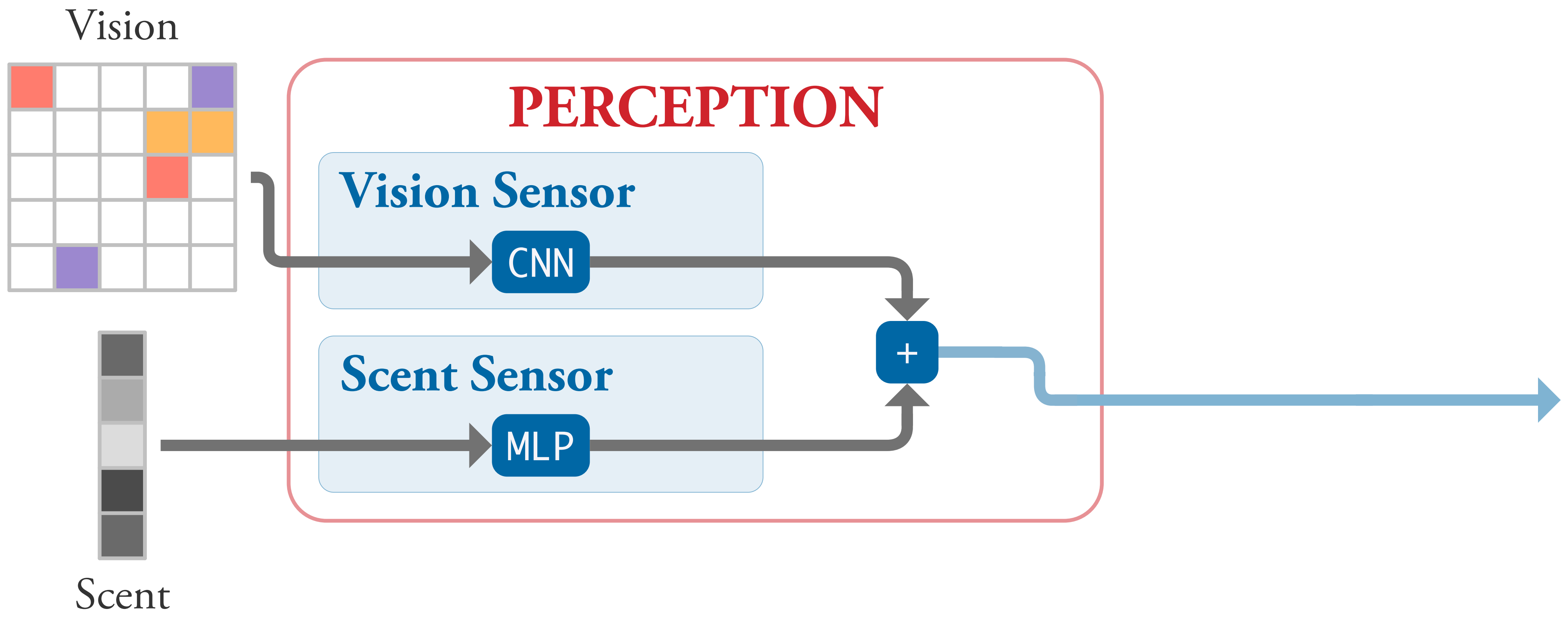
The Hub-and-Spoke Model



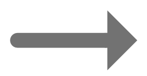


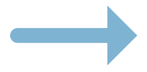
[Rogers et al., Psychological Review 2004]

Unified Architecture: **JBW Example**



LEGEND

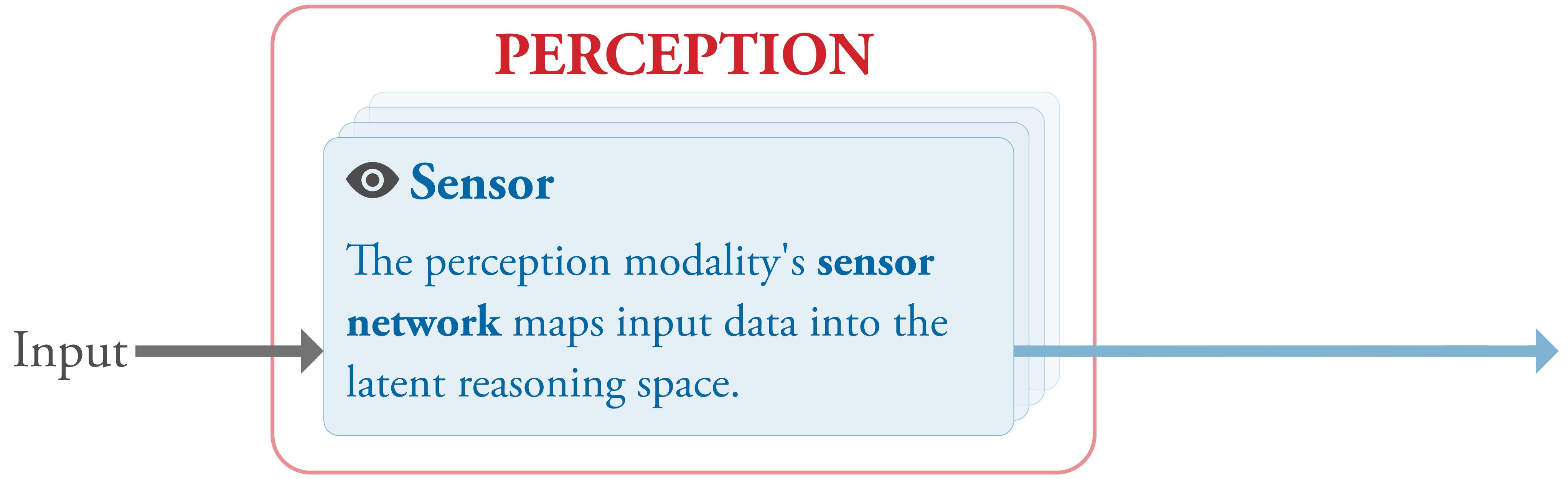

Flow of data whose type is modality/problem dependent


Flow of data whose type is modality/problem independent

Unified Architecture

A **perception modality** is defined as:

- A *data type*, and
- a *sensor network*.



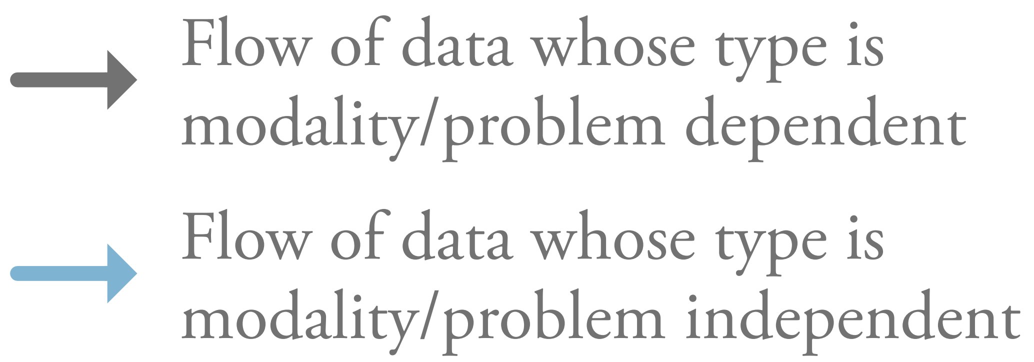
LEGEND

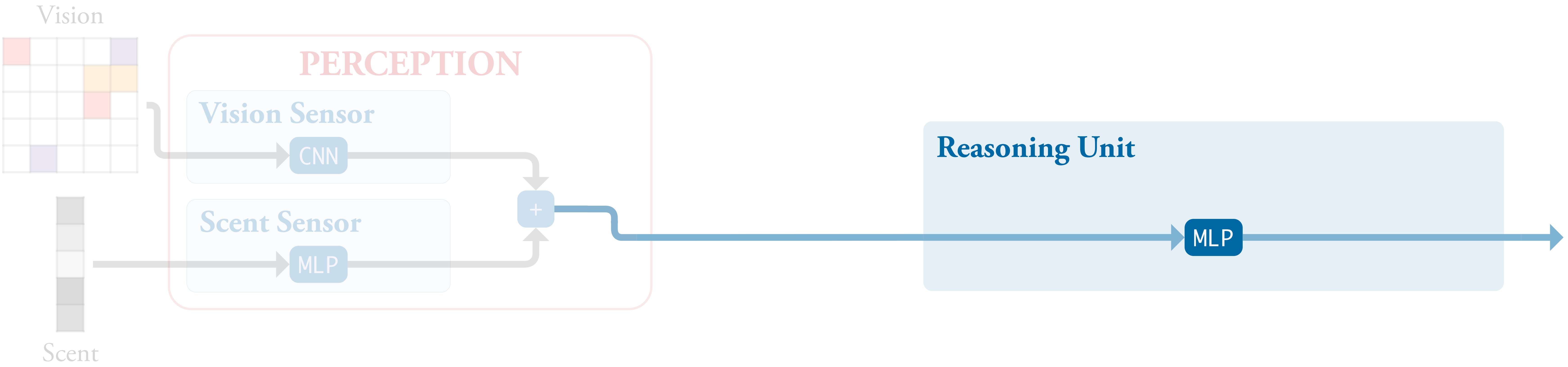
→ Flow of data whose type is modality/problem dependent

→ Flow of data whose type is modality/problem independent

Unified Architecture: **JBW Example**

LEGEND



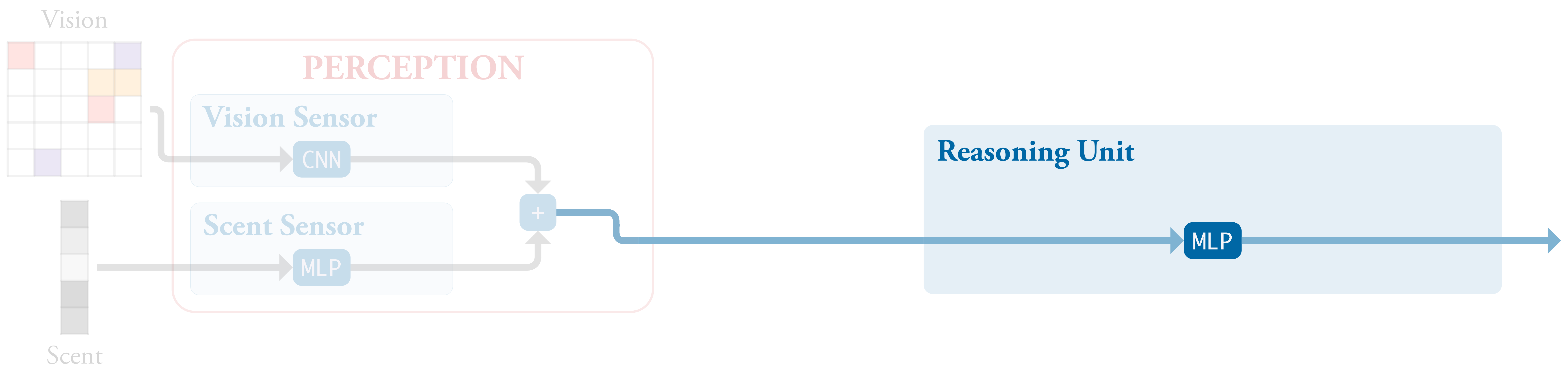


Unified Architecture: **JBW Example**

LEGEND

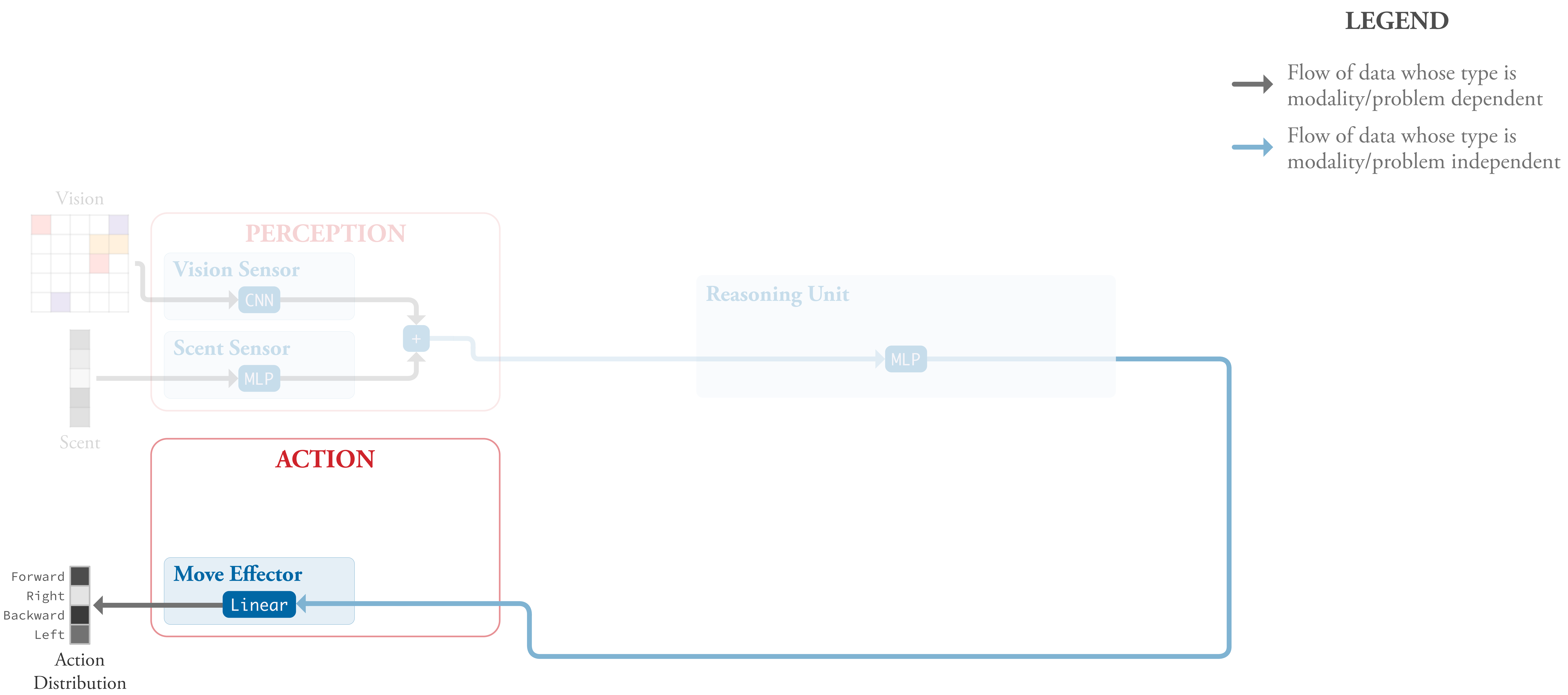
Flow of data whose type is modality/problem dependent

Flow of data whose type is modality/problem independent



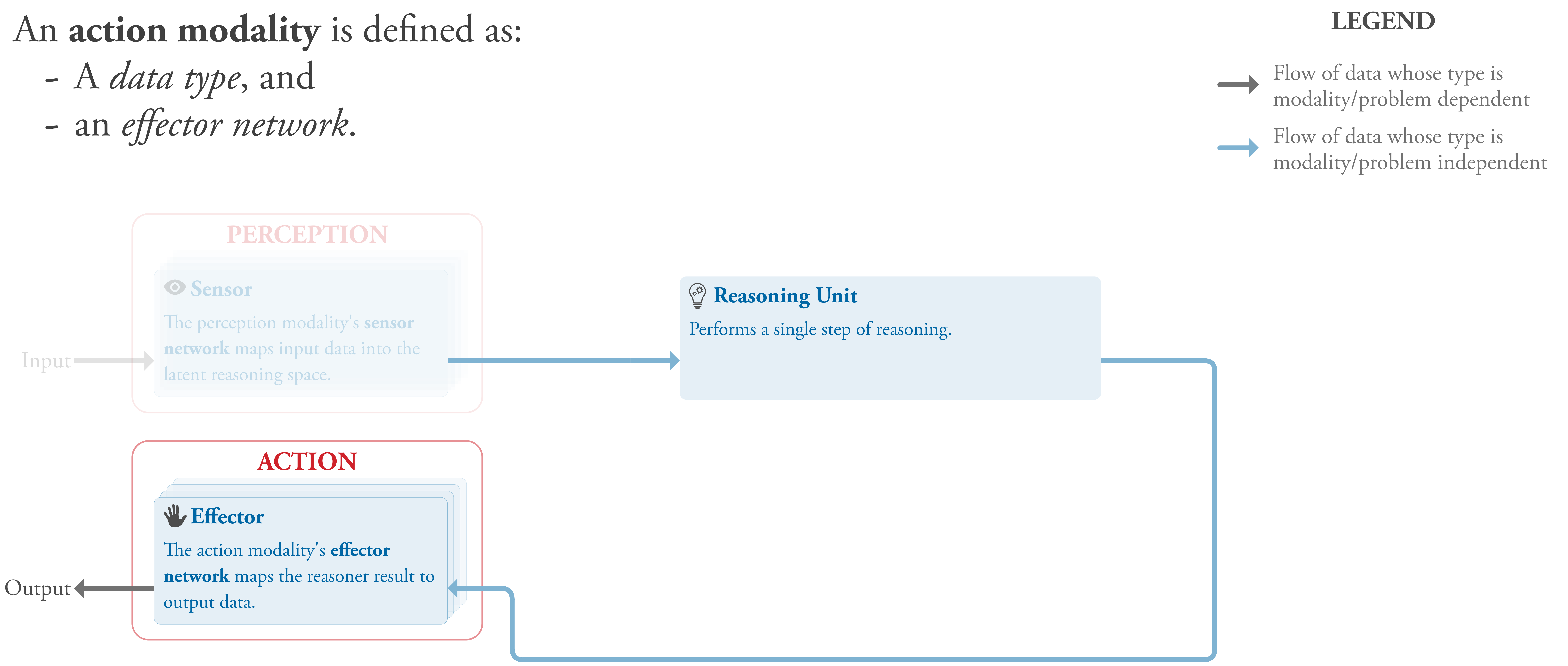
Much of the complexity of deep learning models lies in perception, rather than reasoning.

Unified Architecture: JBW Example

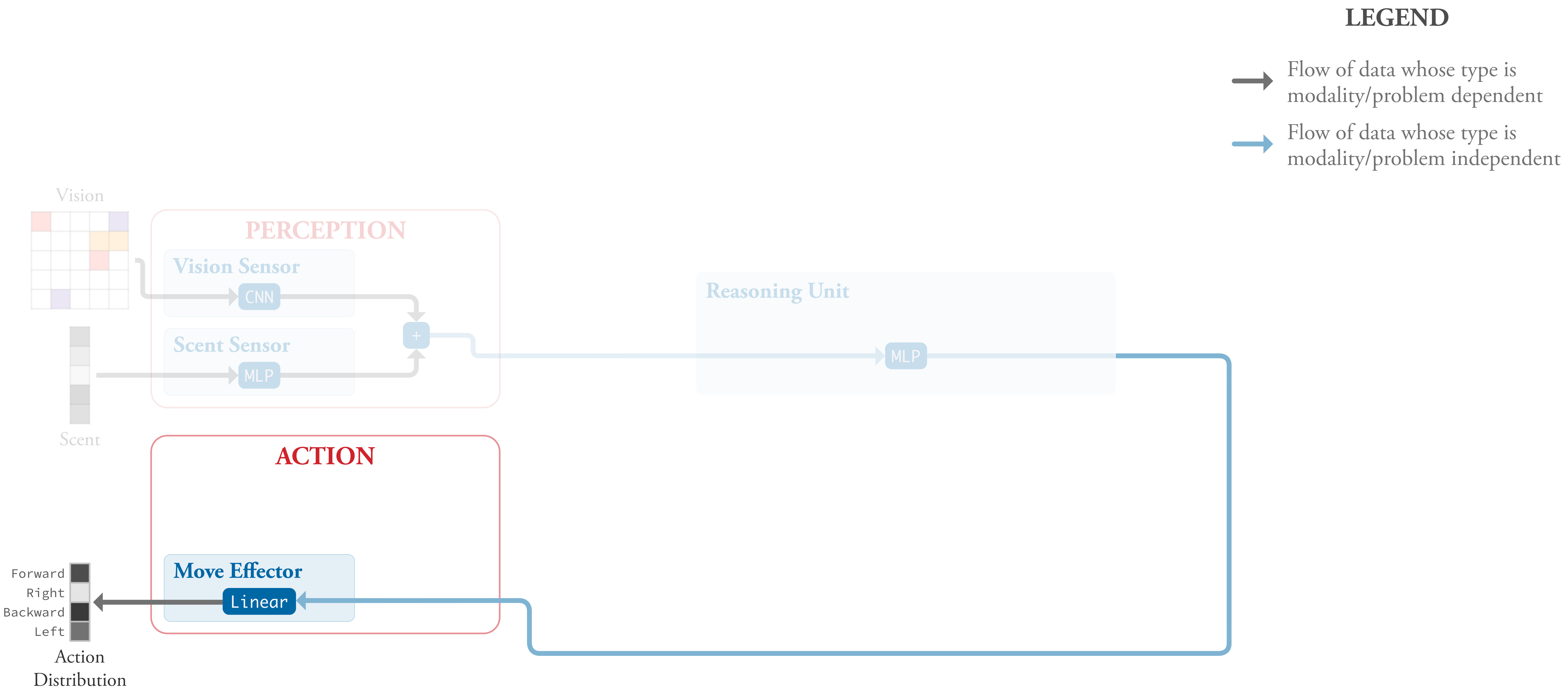


Unified Architecture

An **action modality** is defined as:
- A *data type*, and
- an *effector network*.



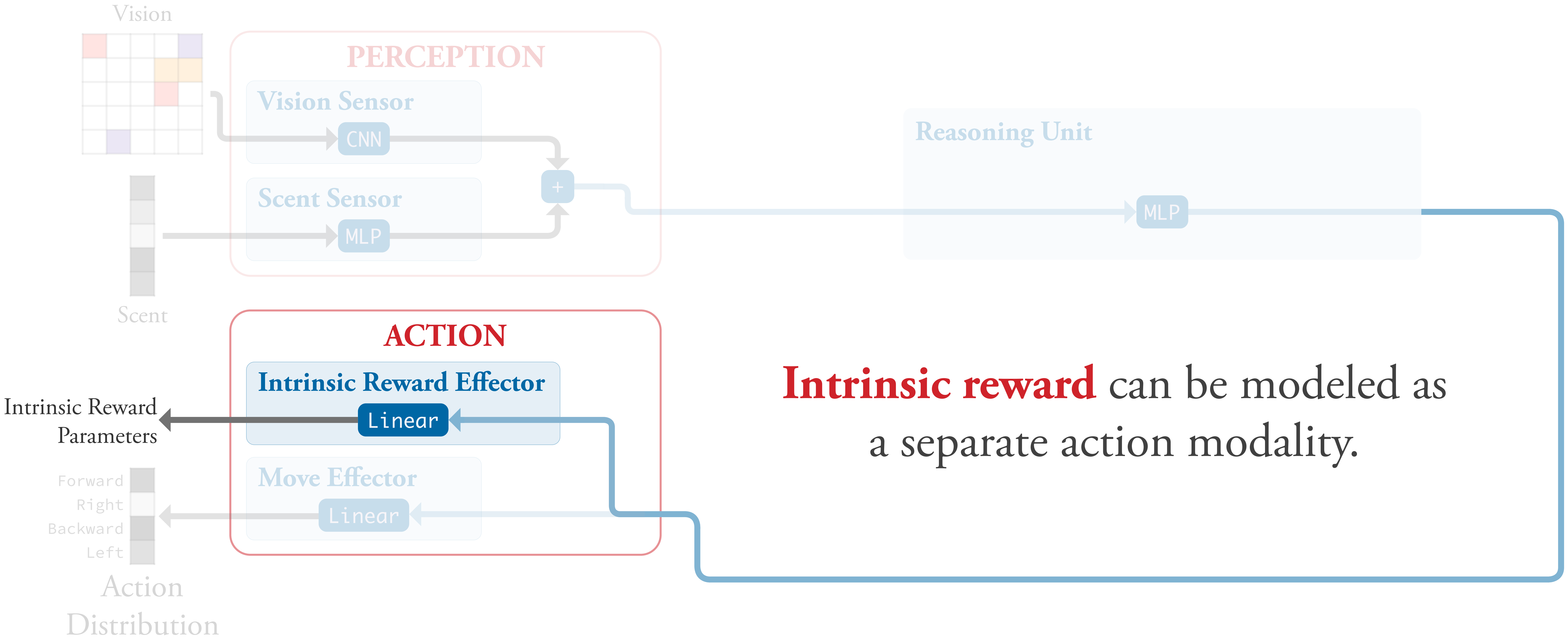
Unified Architecture: JBW Example



Unified Architecture: JBW Example

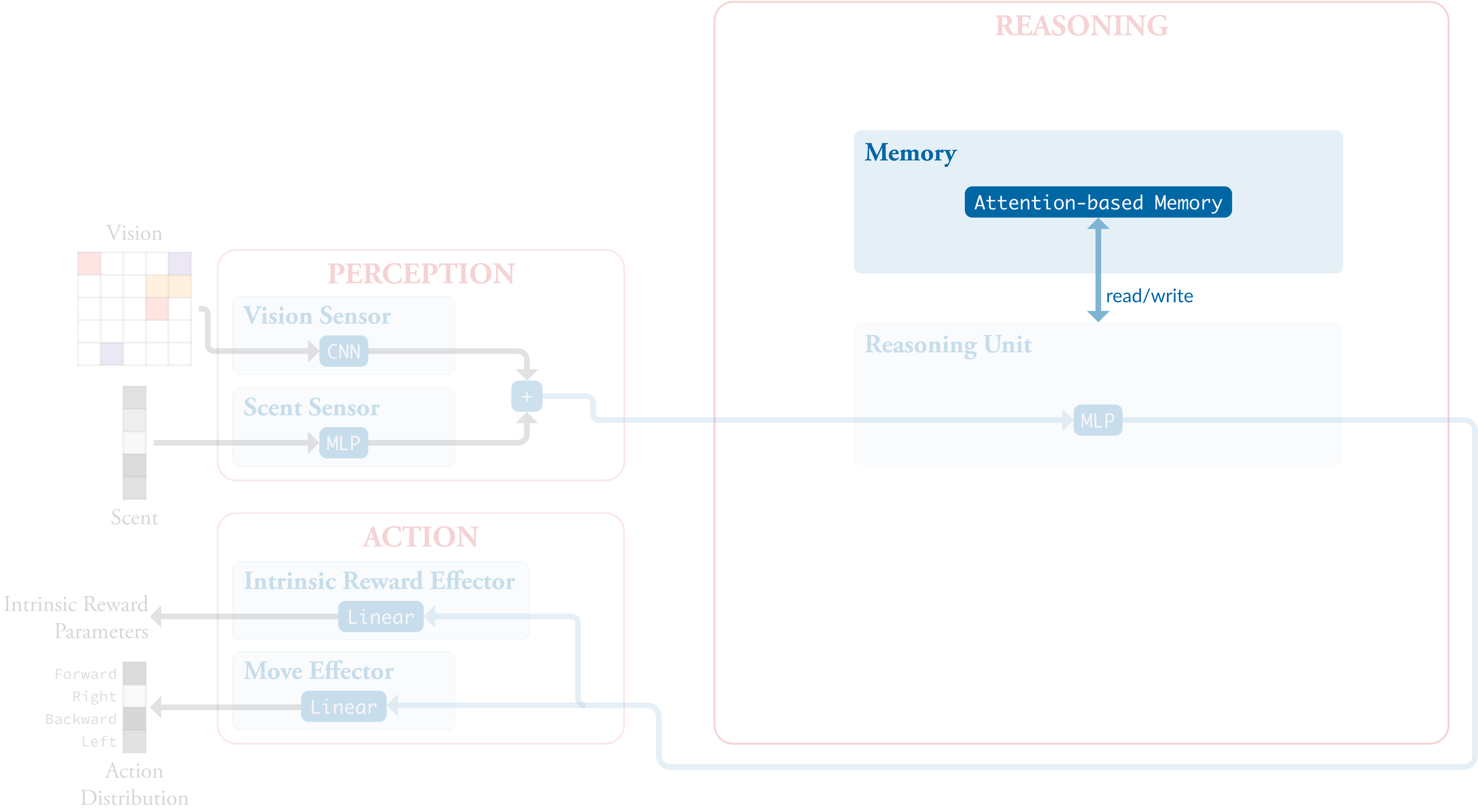
LEGEND

- Flow of data whose type is modality/problem dependent
- Flow of data whose type is modality/problem independent



Intrinsic reward can be modeled as a separate action modality.

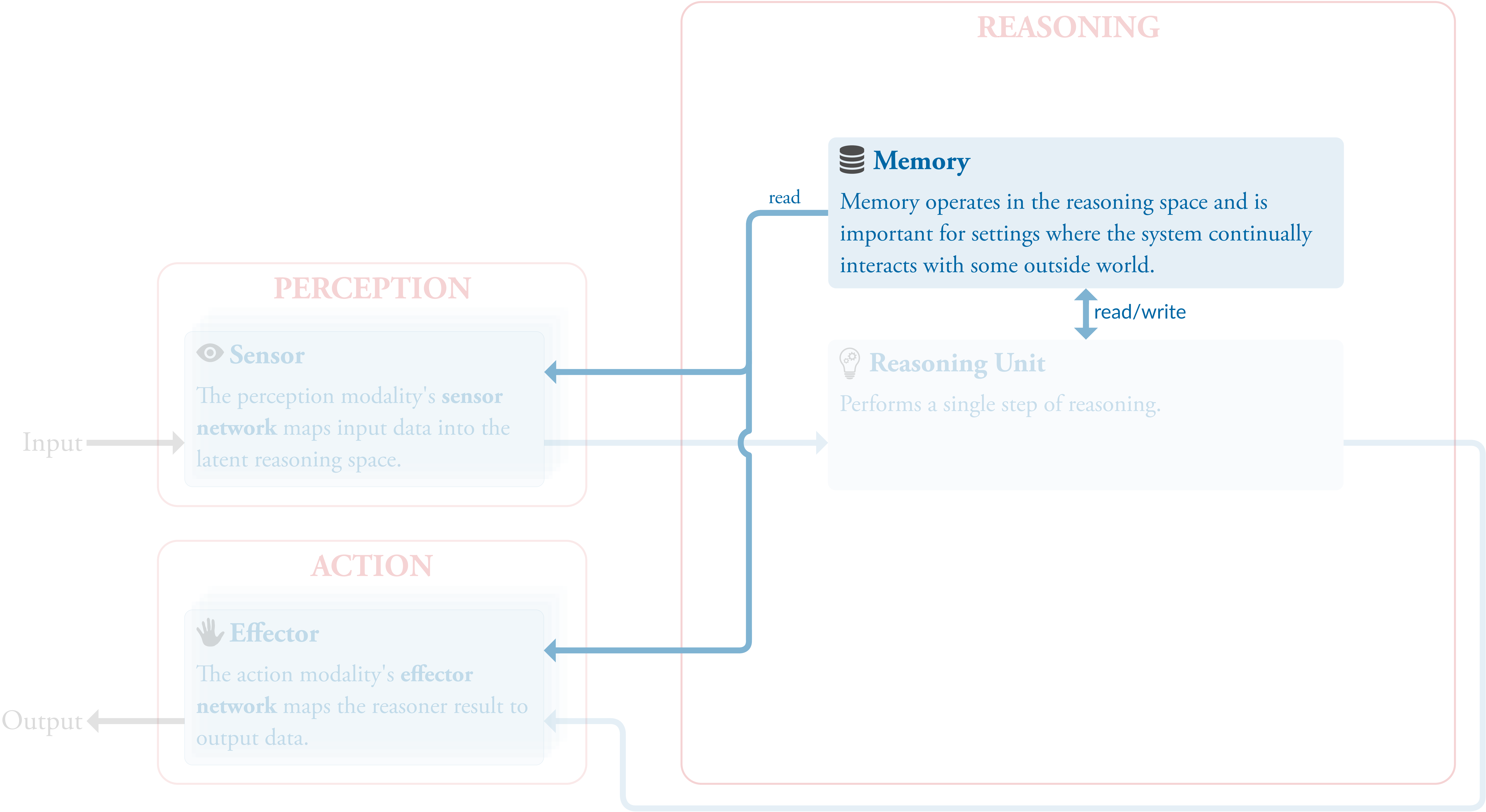
Unified Architecture: JBW Example



LEGEND

- Flow of data whose type is modality/problem dependent
- Flow of data whose type is modality/problem independent

Unified Architecture

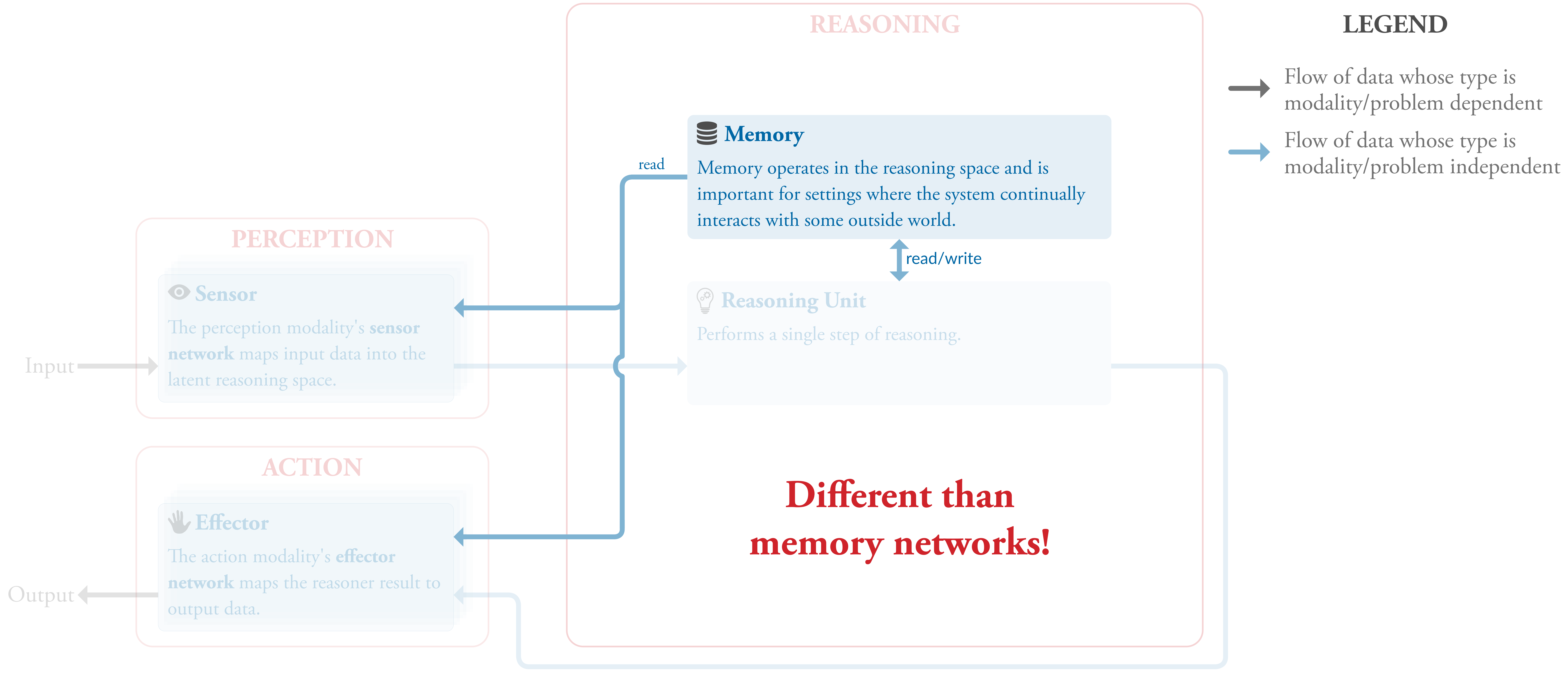


LEGEND

Flow of data whose type is modality/problem dependent

Flow of data whose type is modality/problem independent

Memory


REASONING
Memory
Memory operates in the reasoning space and is important for settings where the system continually interacts with some outside world.
read
read/write
Reasoning Unit
Performs a single step of reasoning.
PERCEPTION
Sensor
The perception modality's **sensor network** maps input data into the latent reasoning space.
Input
ACTION
Effector
The action modality's **effector network** maps the reasoner result to output data.
Output
Different than memory networks!
LEGEND
Flow of data whose type is modality/problem dependent
Flow of data whose type is modality/problem independent

Unified Architecture

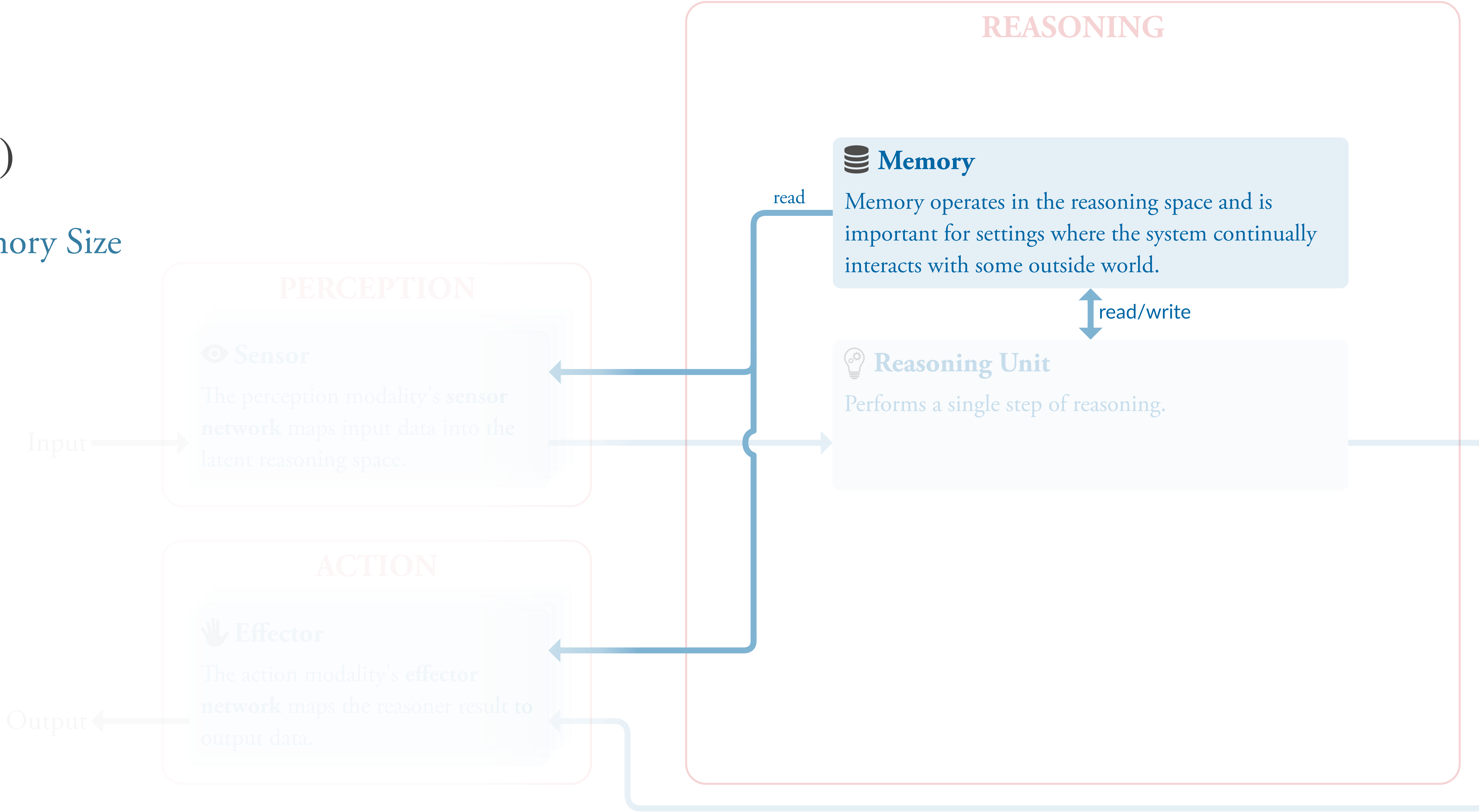
$$M_{\text{read}} : K \mapsto V$$

$$M_{\text{write}} : (K, V) \mapsto ()$$

$$M_k \in \mathbb{R}^{M \times D_k}$$

Memory Size

$$M_v \in \mathbb{R}^{M \times D_v}$$



Unified Architecture

$$M_{\text{read}} : K \mapsto V$$

$$M_{\text{write}} : (K, V) \mapsto ()$$

$$M_k \in \mathbb{R}^{M \times D_k}$$

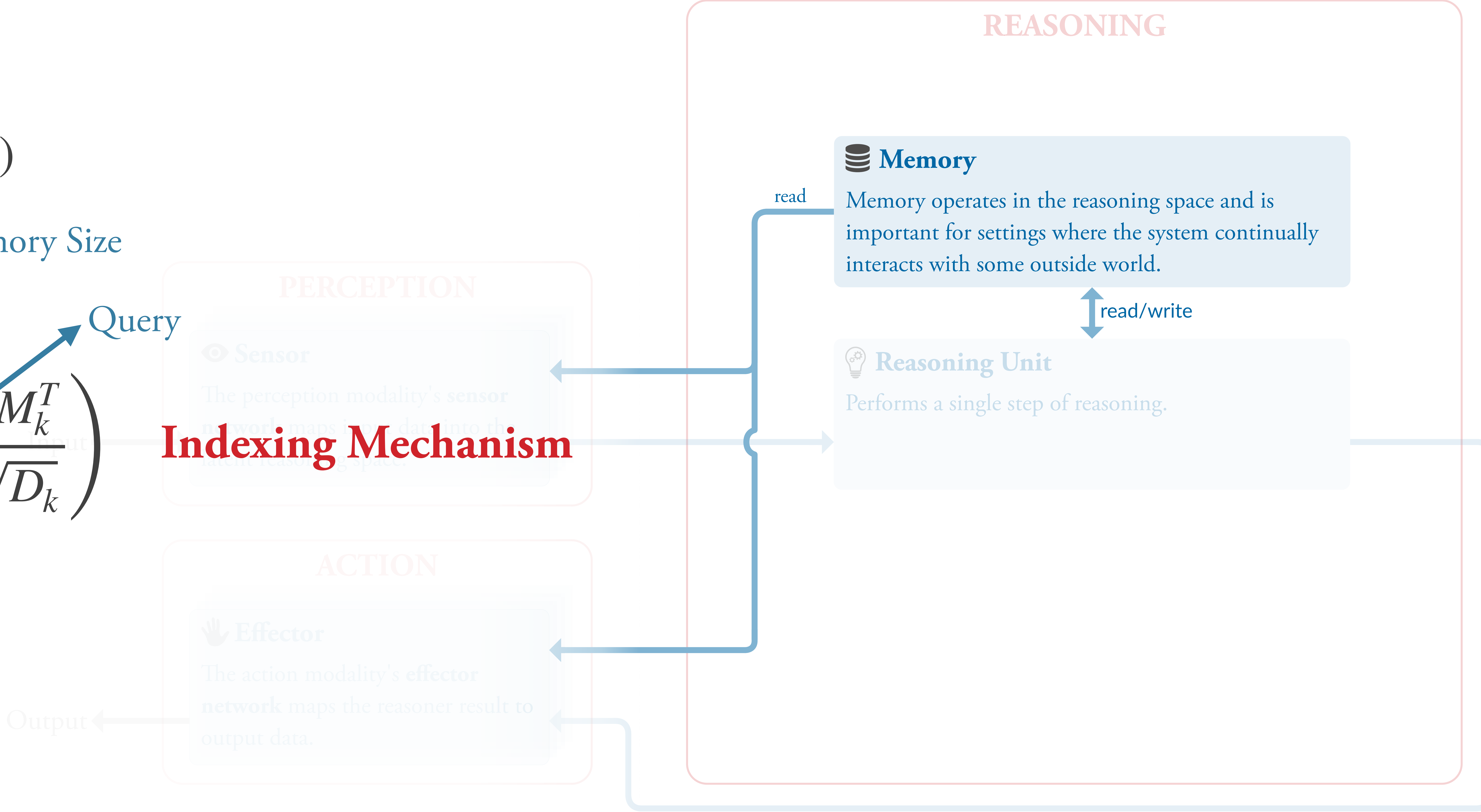
Memory Size

$$M_v \in \mathbb{R}^{M \times D_v}$$

Query

$$I(q) = \text{Softmax}\left(\frac{qM_k^T}{\sqrt{D_k}}\right)$$

Indexing Mechanism



$$M_{\text{read}} : K \mapsto V$$

$$M_{\text{write}} : (K, V) \mapsto ()$$

$$M_k \in \mathbb{R}^{M \times D_k}$$

Memory Size

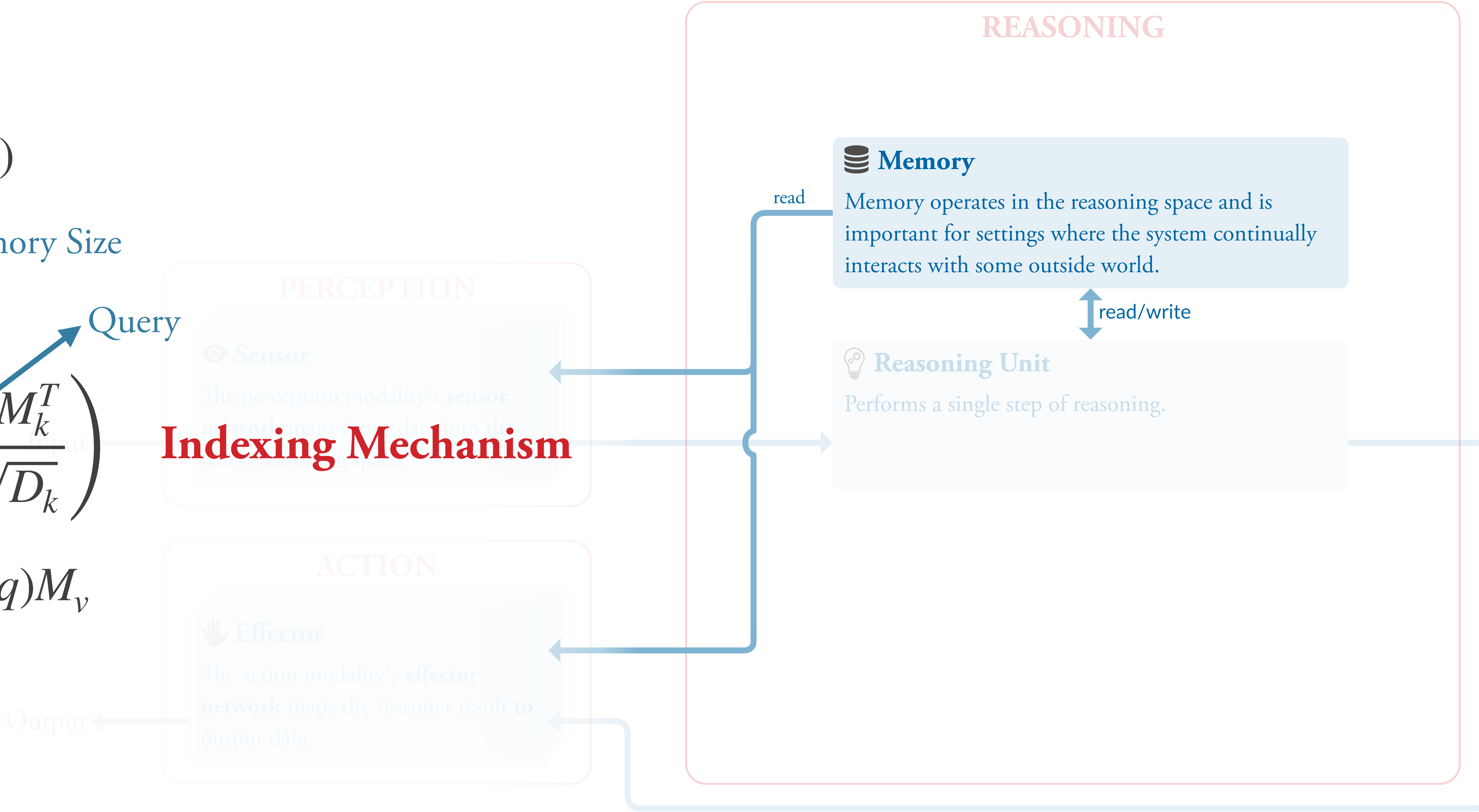
$$M_v \in \mathbb{R}^{M \times D_v}$$

Query

$$I(q) = \text{Softmax}\left(\frac{qM_k^T}{\sqrt{D_k}}\right)$$

Indexing Mechanism

$$M_{\text{read}}(q) : \text{return } I(q)M_v$$



$$M_{\text{read}} : K \mapsto V$$

$$M_{\text{write}} : (K, V) \mapsto ()$$

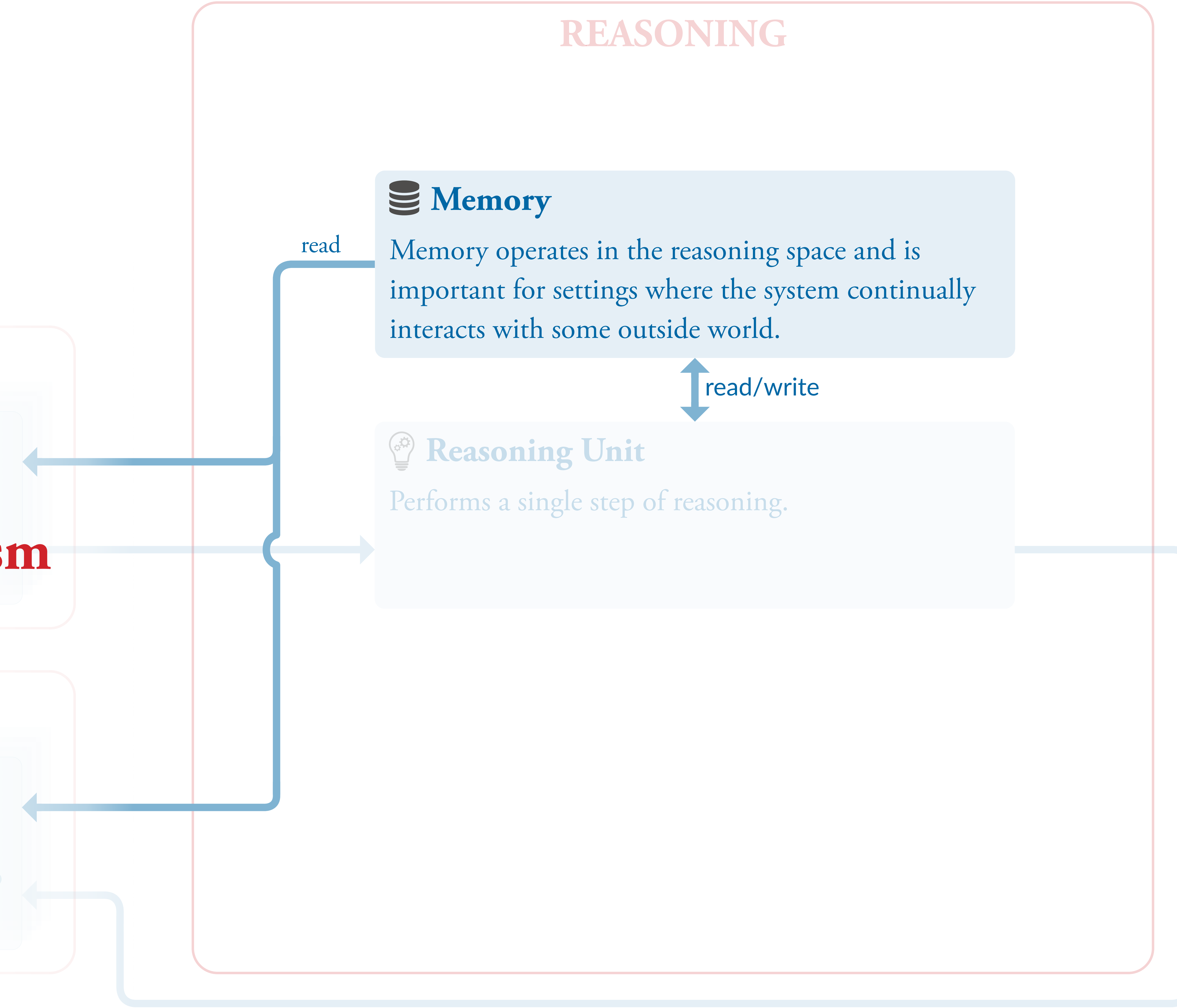
$$M_k \in \mathbb{R}^{M \times D_k}$$ Memory Size

$$M_v \in \mathbb{R}^{M \times D_v}$$ Query

$$I(q) = \text{Softmax}\left(\frac{qM_k^T}{\sqrt{D_k}}\right)$$ **Indexing Mechanism**

$$M_{\text{read}}(q) : \text{return } I(q)M_v$$ Forgetting Factor

$$M_{\text{write}}(q, v) : M_v := \lambda I(q)v + (1 - \lambda I(q))M_v$$



Unified Architecture

Memory

$$M_{\text{read}} : K \mapsto V$$

$$M_{\text{write}} : (K, V) \mapsto ()$$

$$M_k \in \mathbb{R}^{M \times D_k}$$

Memory Size

$$M_v \in \mathbb{R}^{M \times D_v}$$

Query

$$I(q) = \text{Softmax}\left(\frac{qM_k^T}{\sqrt{D_k}}\right)$$

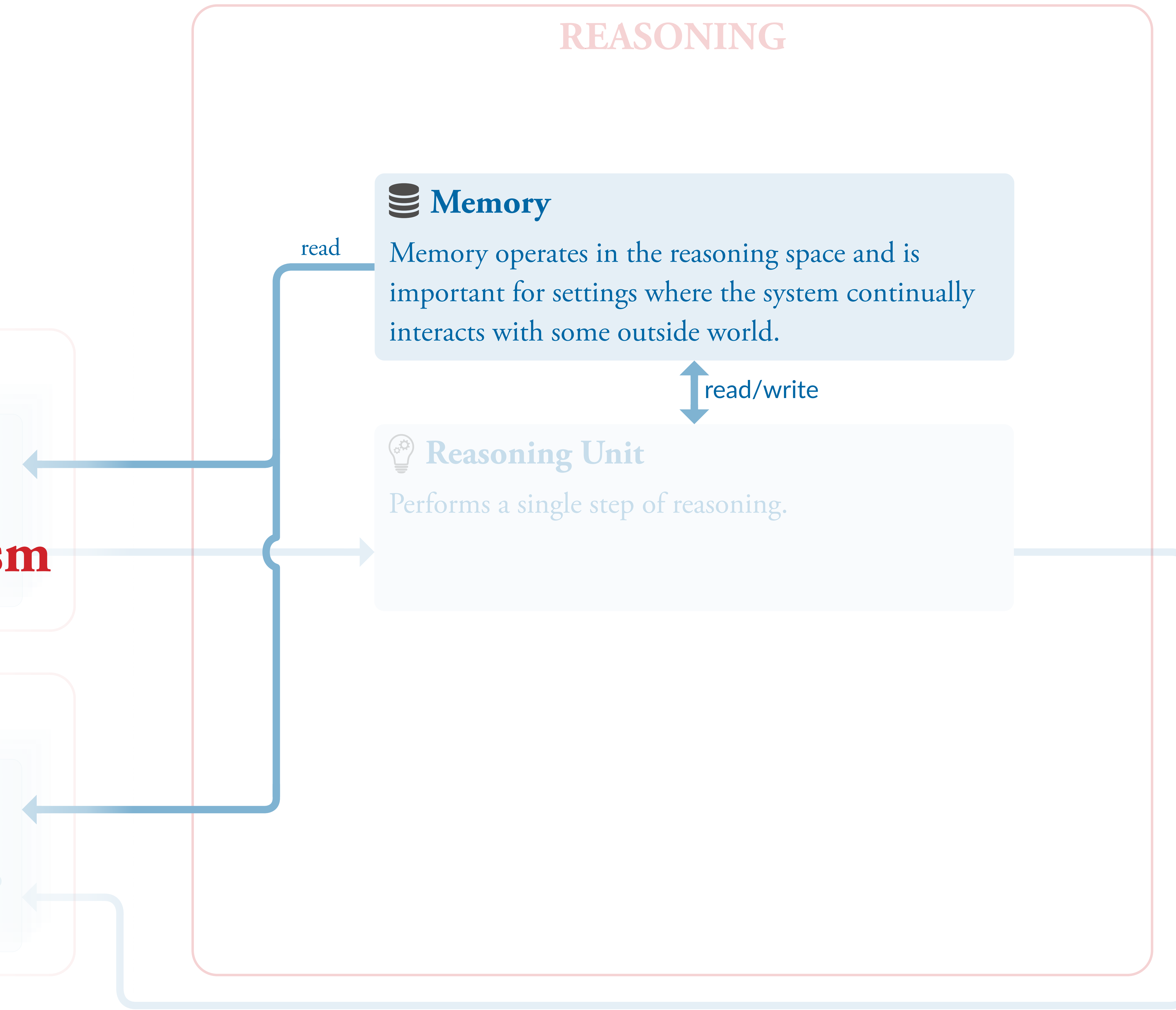
Indexing Mechanism

$$M_{\text{read}}(q) : \text{return } I(q)M_v$$

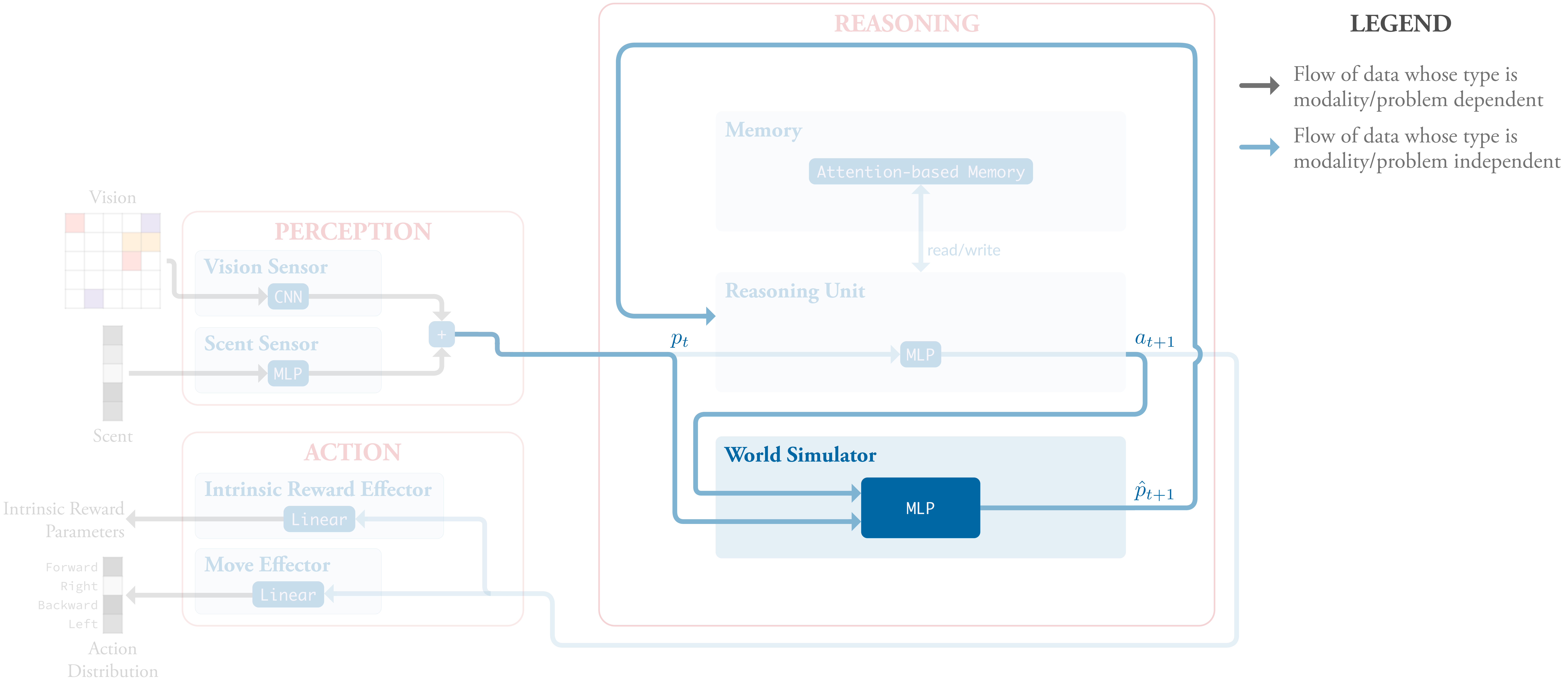
Forgetting Factor

$$M_{\text{write}}(q, v) : M_v := \lambda I(q)v + (1 - \lambda I(q))M_v$$

Enables associative learning and memories!



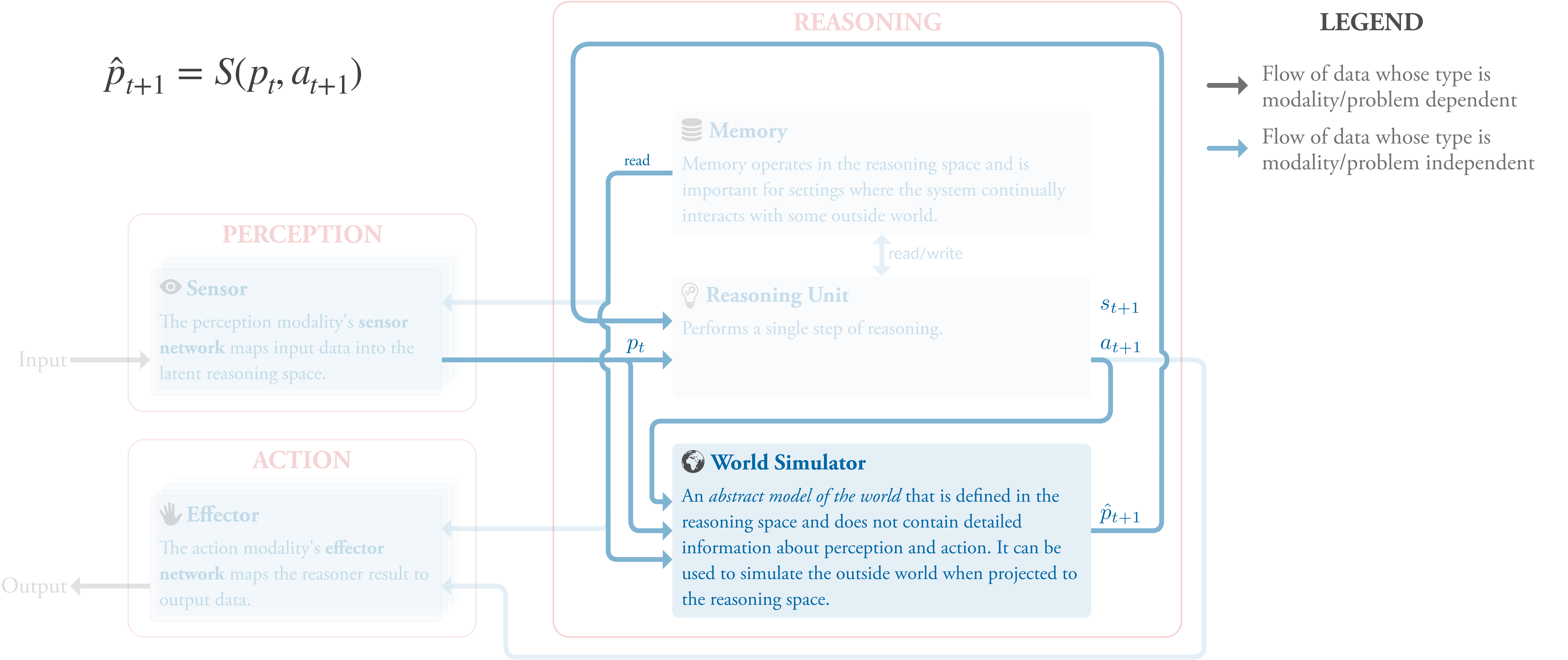
Unified Architecture: JBW Example



Unified Architecture

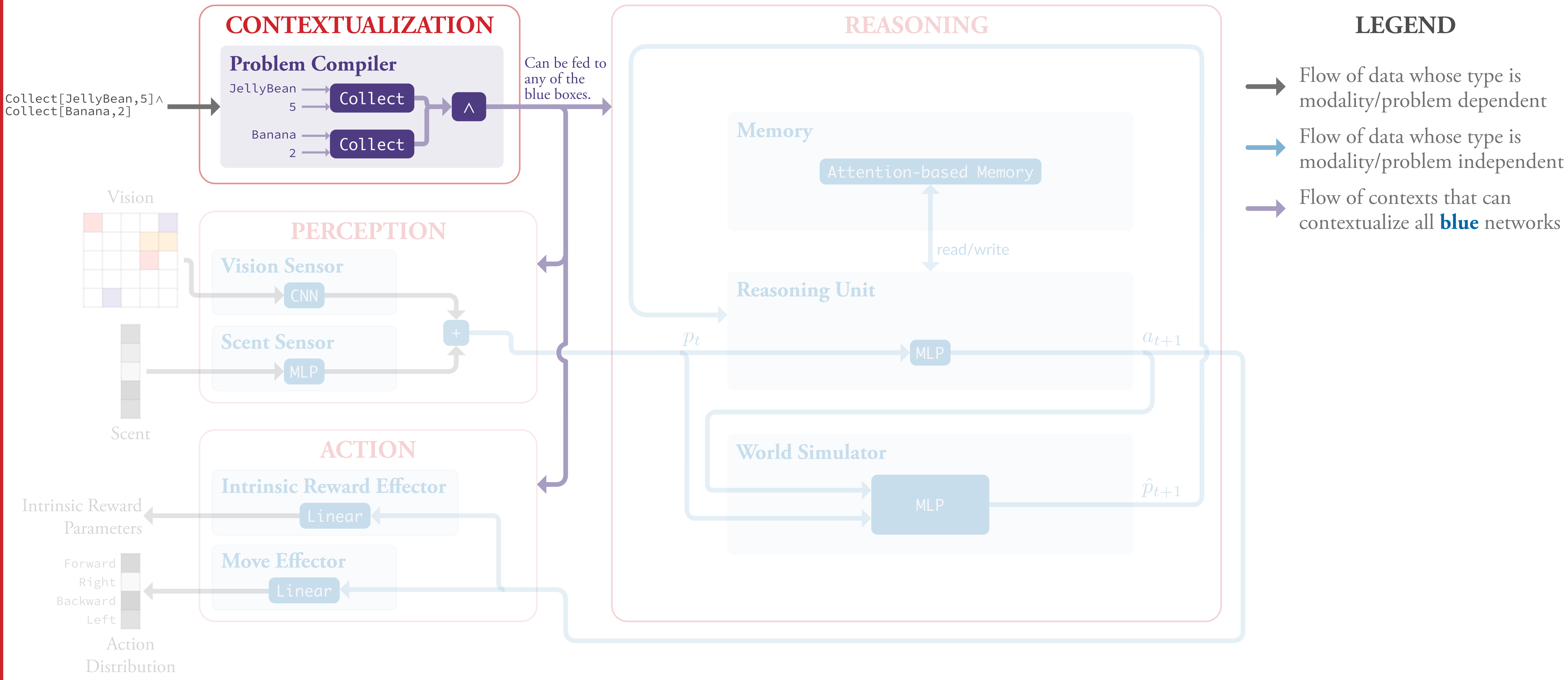
World Simulator

$$\hat{p}_{t+1} = S(p_t, a_{t+1})$$



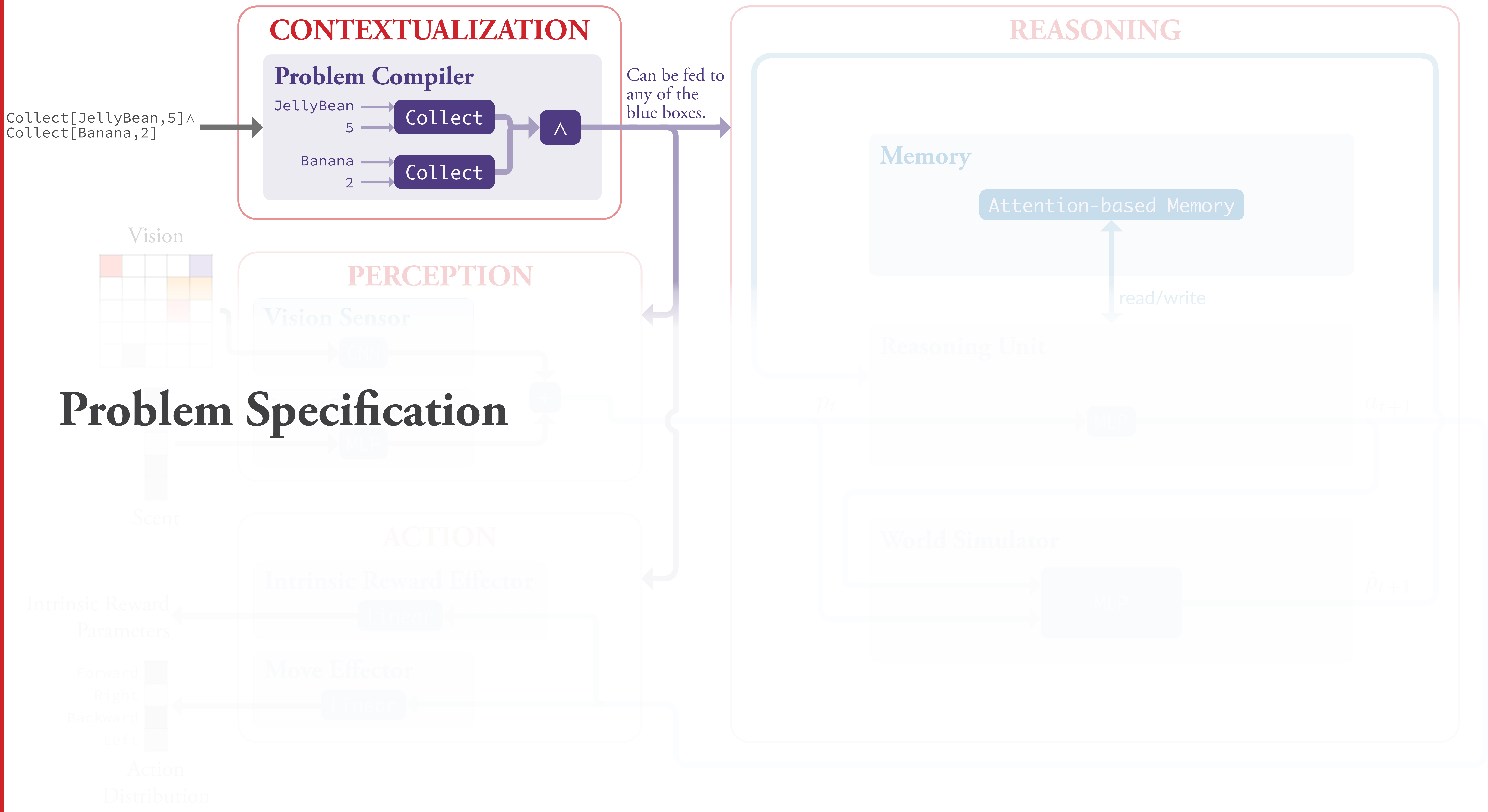
Unified Architecture: JBW Example

Goal Contextualization



Unified Architecture: JBW Example

Goal Contextualization



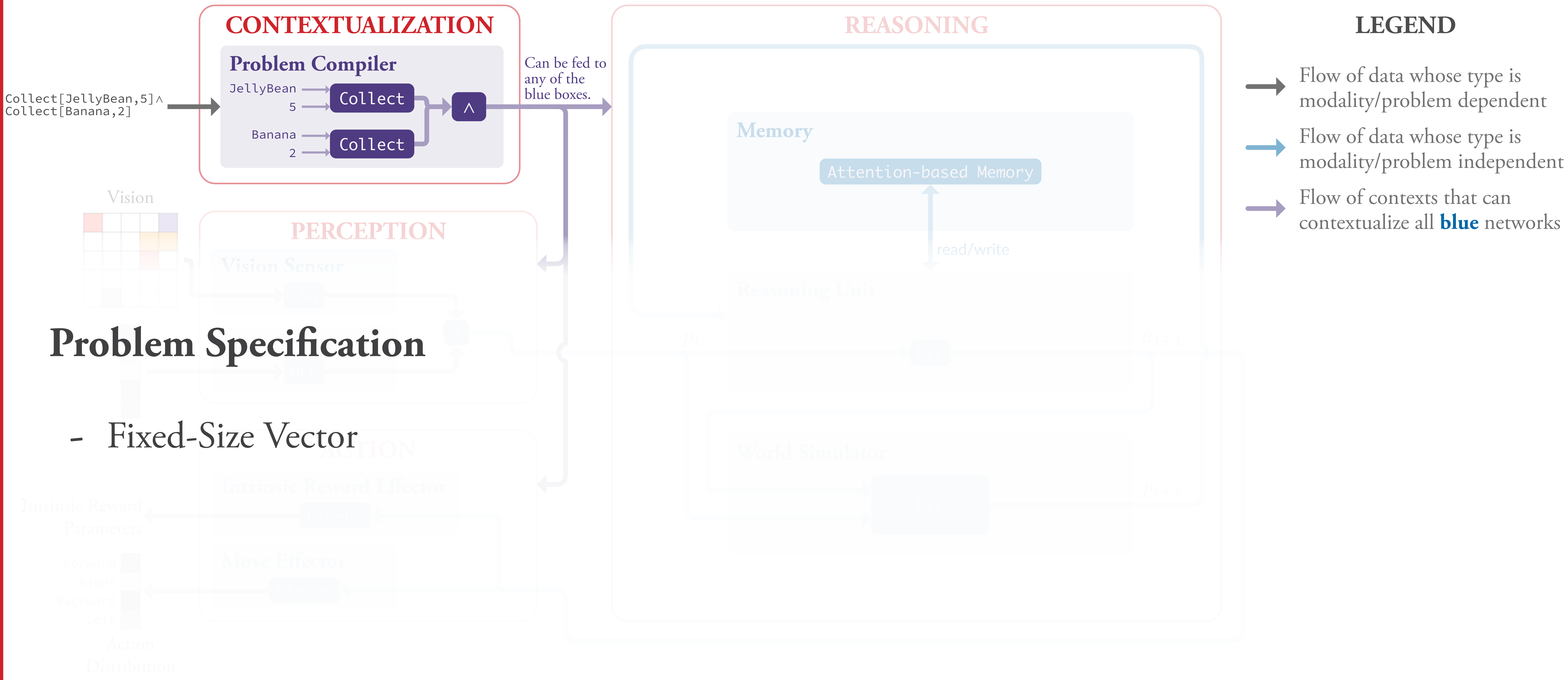
Problem Specification

LEGEND

- Flow of data whose type is modality/problem dependent
- Flow of data whose type is modality/problem independent
- Flow of contexts that can contextualize all **blue** networks

Unified Architecture: JBW Example

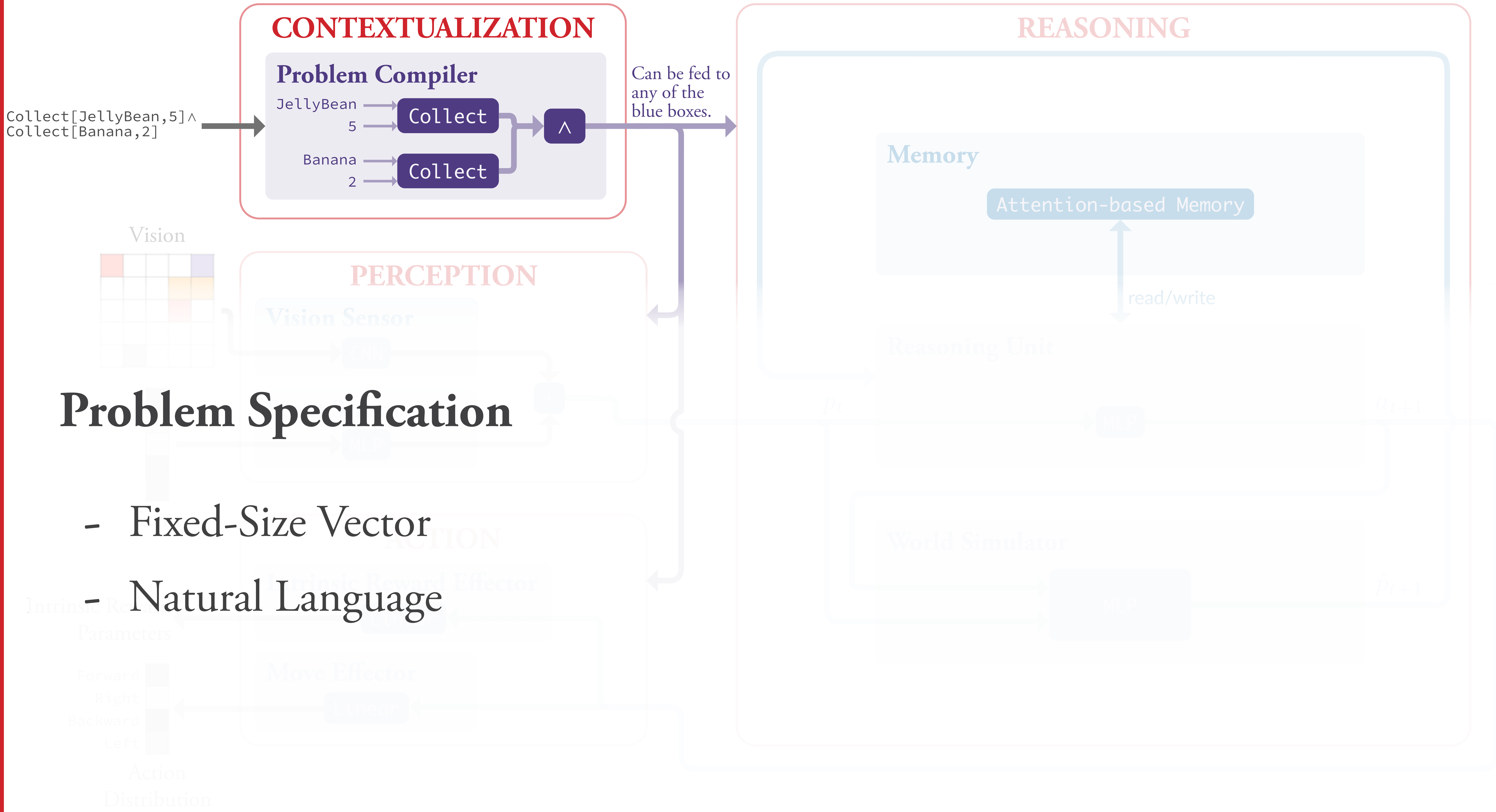
Goal Contextualization



Problem Specification

- Fixed-Size Vector

Unified Architecture: **JBW Example** Goal Contextualization



LEGEND

- Flow of data whose type is modality/problem dependent
- Flow of data whose type is modality/problem independent
- Flow of contexts that can contextualize all **blue** networks

Problem Specification

- Fixed-Size Vector
- Natural Language

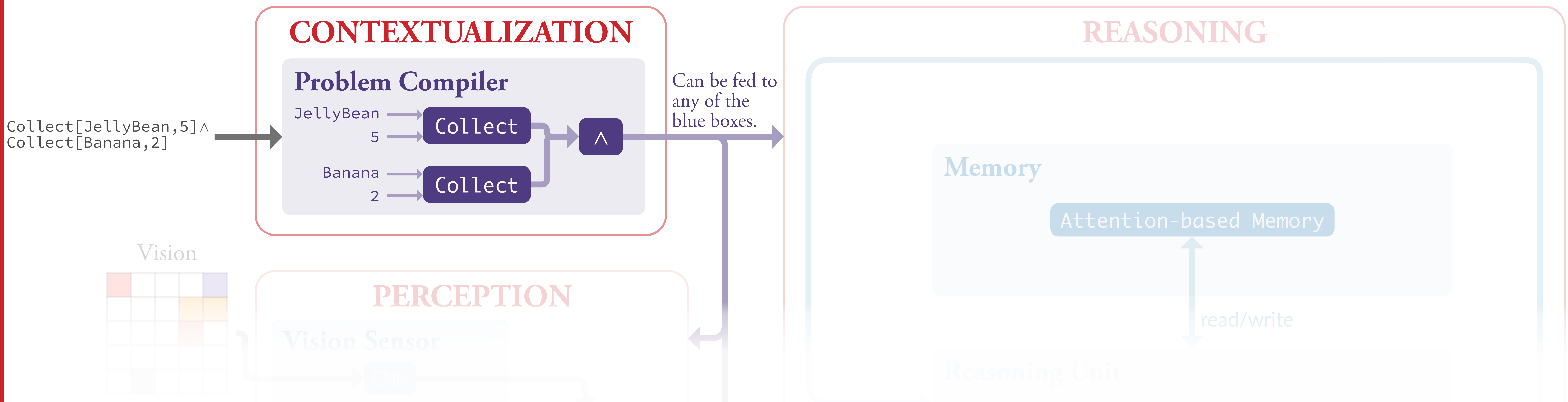
Unified Architecture: JBW Example

Goal Contextualization

Problem Specification

- Fixed-Size Vector
- Natural Language
- Structured Language

Unified Architecture: **JBW Example** Goal Contextualization



LEGEND

- Flow of data whose type is modality/problem dependent
- Flow of data whose type is modality/problem independent
- Flow of contexts that can contextualize all **blue** networks

Problem Specification

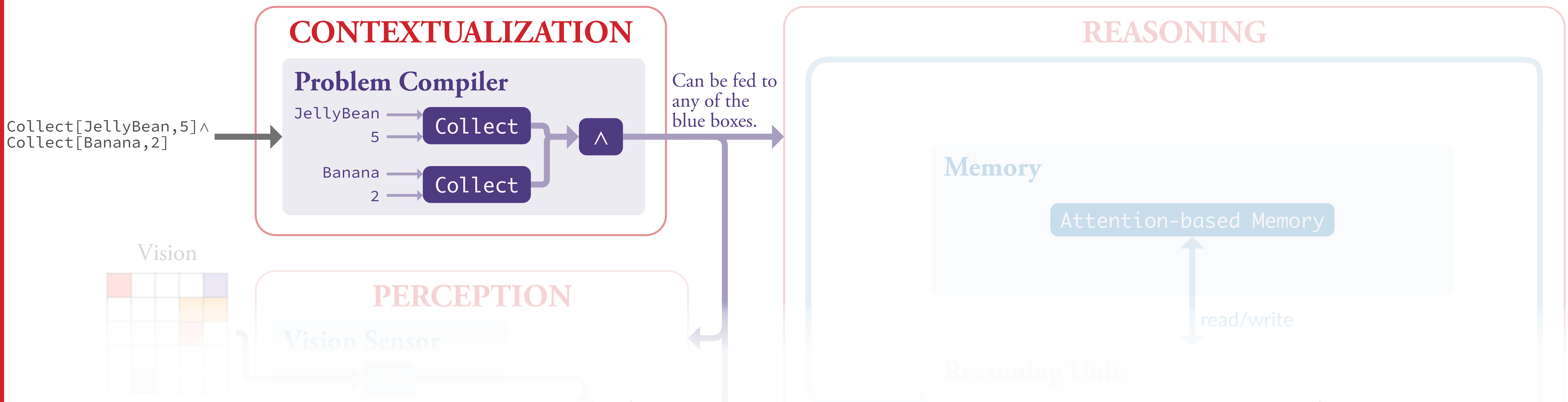
- Fixed-Size Vector
- Natural Language
- Structured Language

Problem Compilation

Problem Specification → Structured Representation (e.g., a vector).

Unified Architecture: JBW Example

Goal Contextualization



LEGEND

- Flow of data whose type is modality/problem dependent
- Flow of data whose type is modality/problem independent
- Flow of contexts that can contextualize all **blue** networks

Problem Specification

- Fixed-Size Vector
- Natural Language
- Structured Language

Problem Compilation

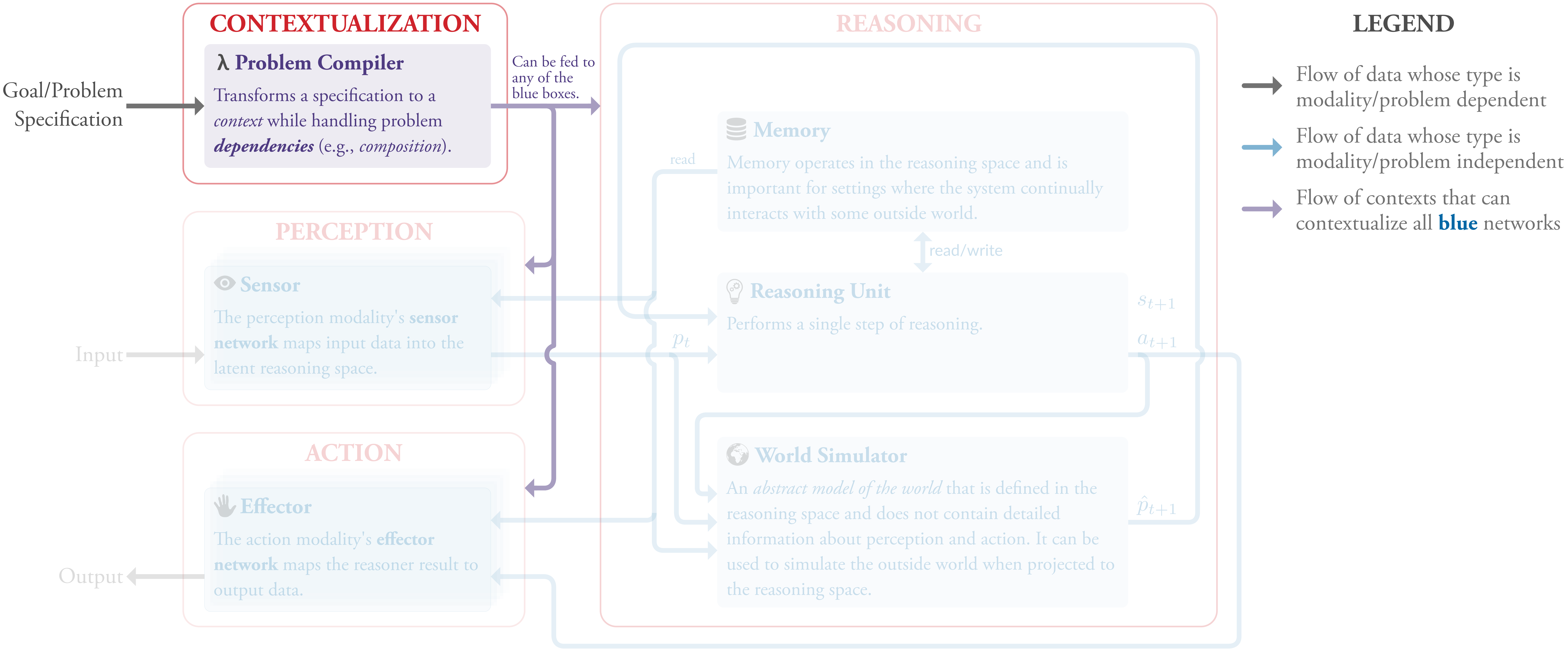
Problem Specification → Structured Representation (e.g., a vector).

Problem Generation

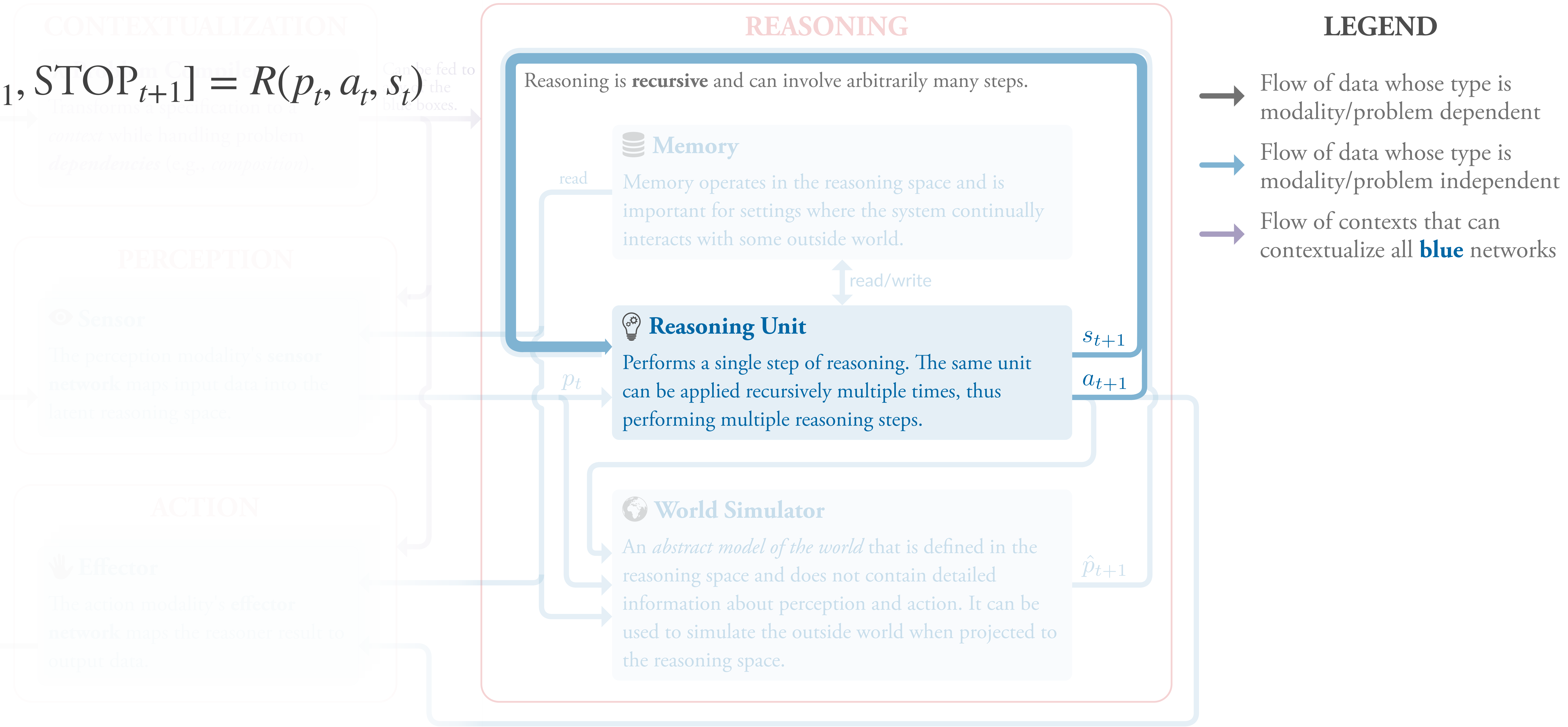
Action modalities can generate problems that are fed to the compiler.

Unified Architecture

Goal Contextualization

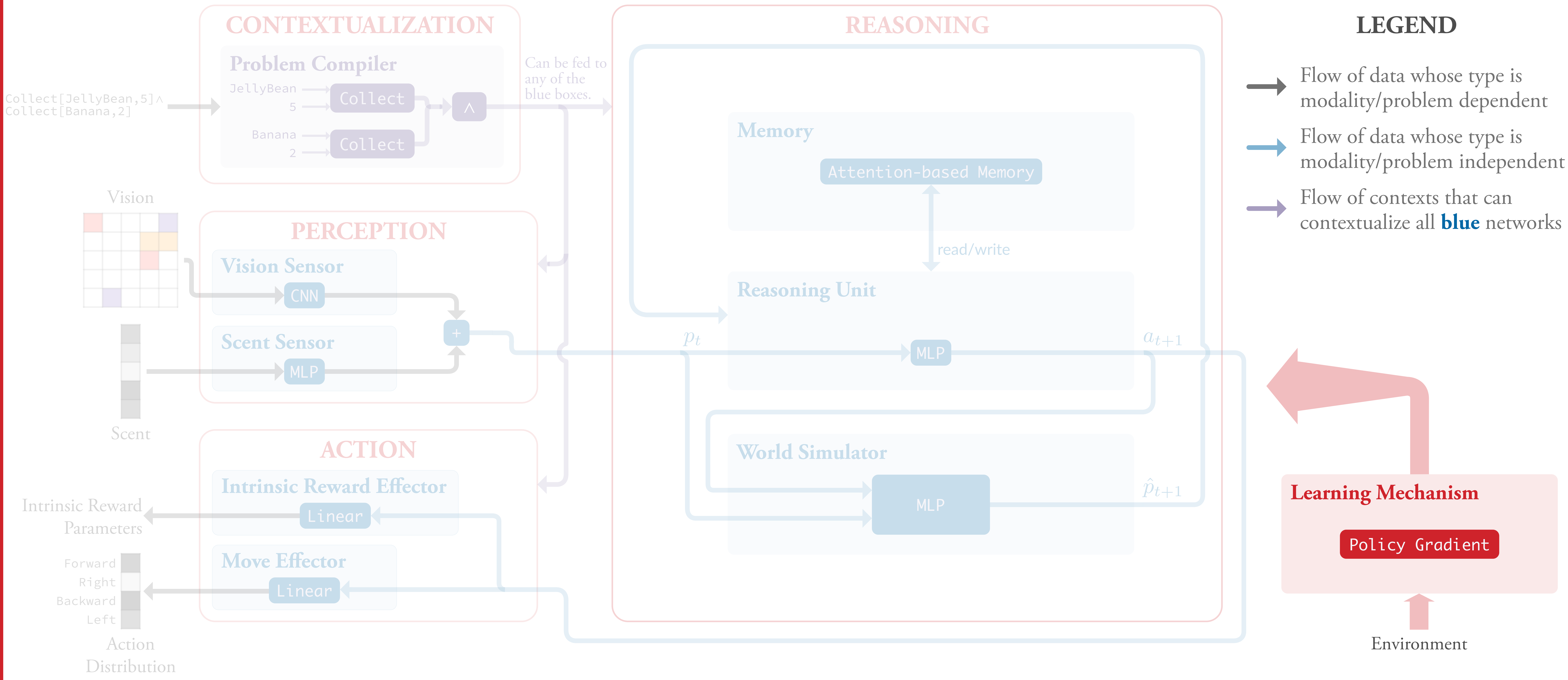


$$[a_{t+1}, s_{t+1}, \mathrm{STOP}_{t+1}] = R(p_t, a_t, s_t)$$



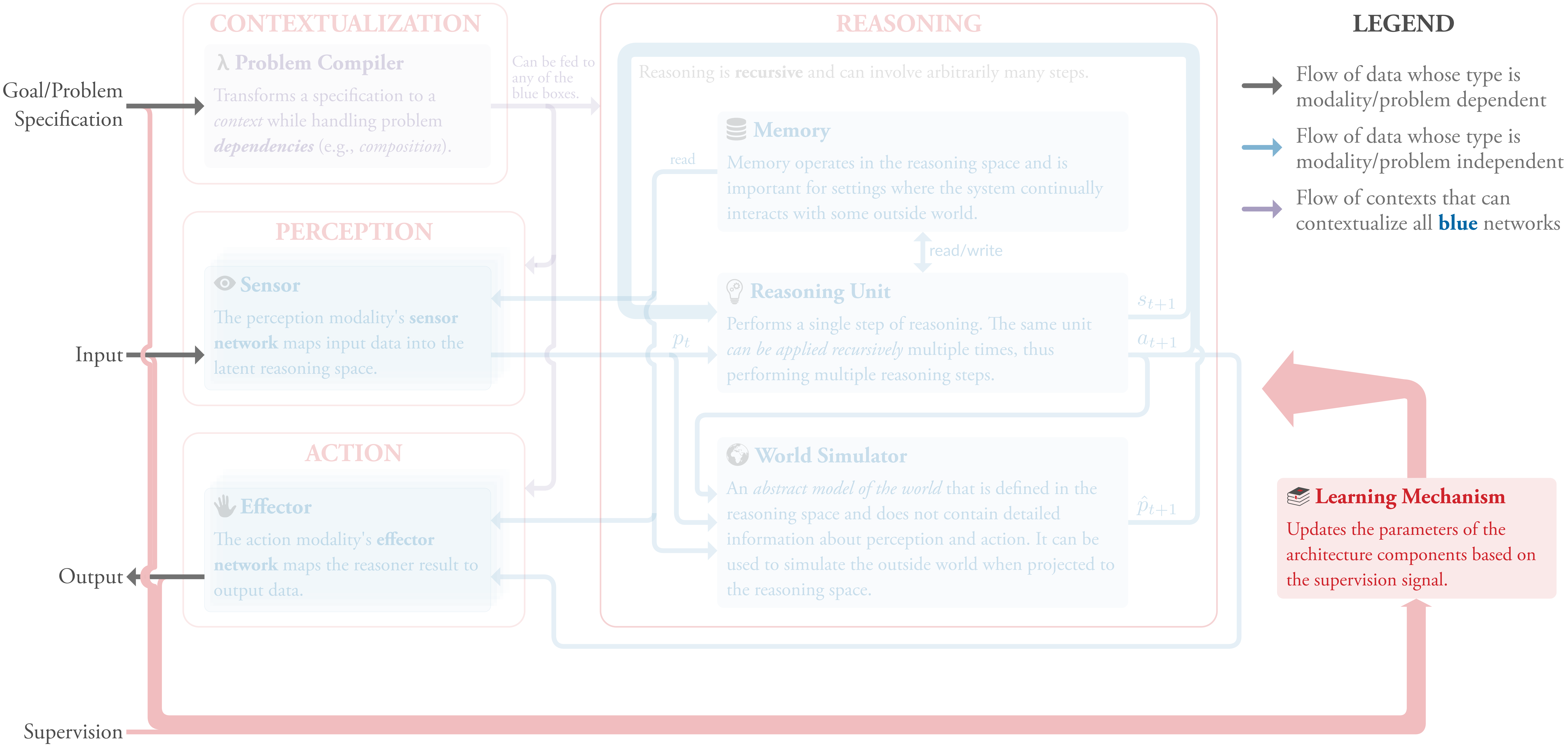
Unified Architecture: JBW Example

Learning Mechanism

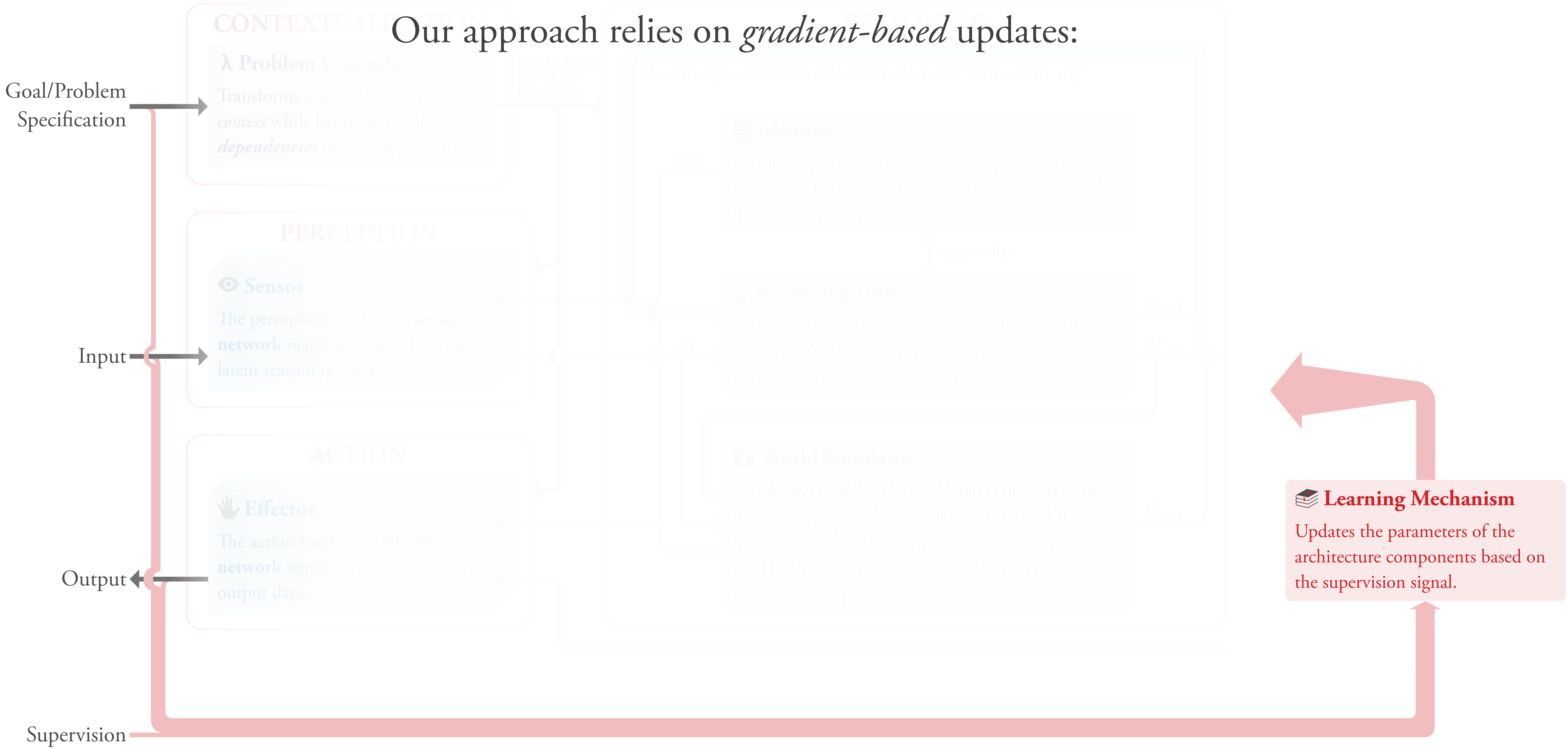


Unified Architecture

Learning Mechanism



Our approach relies on *gradient-based* updates:

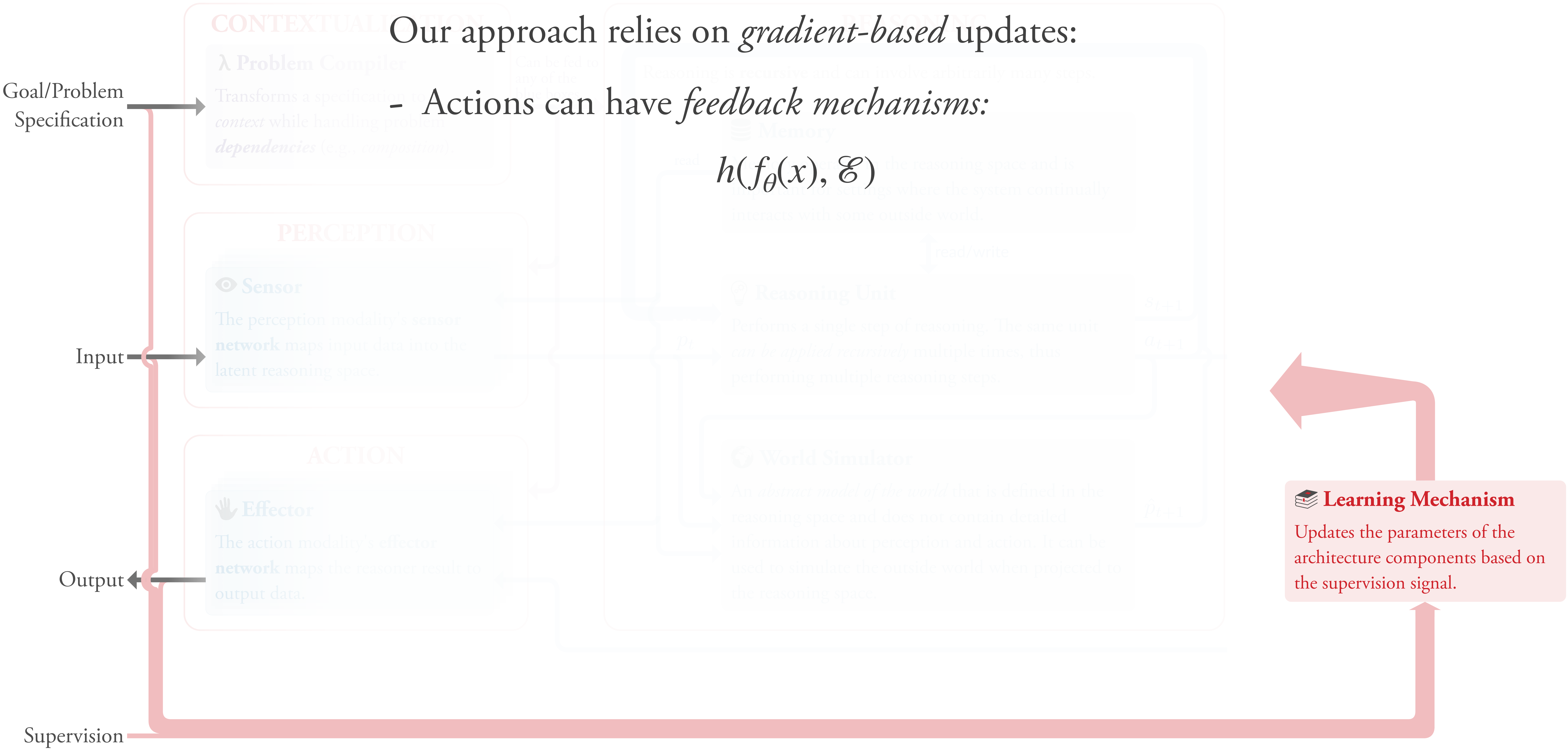


Unified Architecture

Our approach relies on *gradient-based* updates:

- Actions can have *feedback mechanisms:*

$$h(f_\theta(x), \mathcal{E})$$



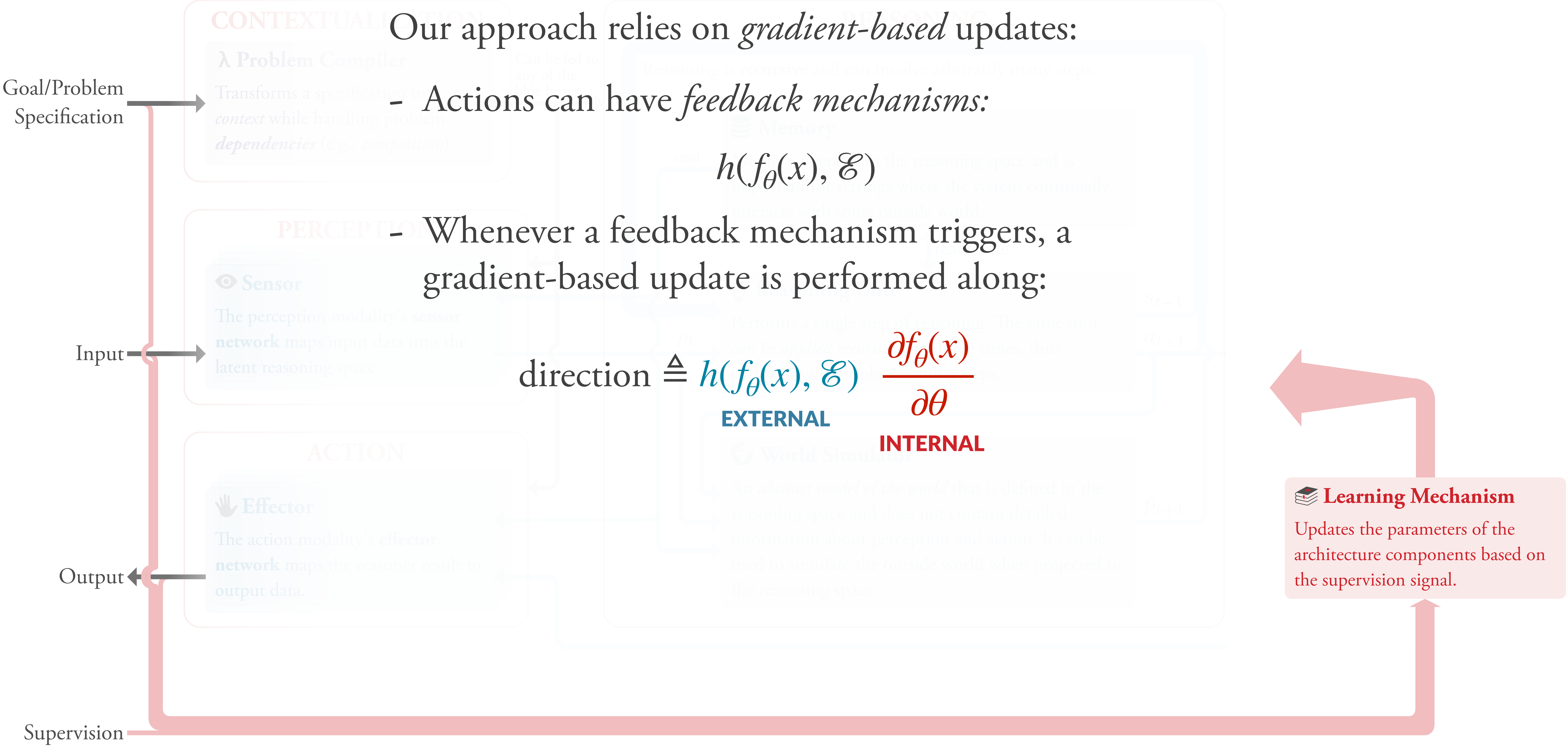
Our approach relies on *gradient-based* updates:

- Actions can have *feedback mechanisms:*

$$h(f_\theta(x), \mathscr{E})$$

- Whenever a feedback mechanism triggers, a gradient-based update is performed along:

$$\text{direction} \triangleq \underbrace{h(f_\theta(x), \mathscr{E})}_{\text{EXTERNAL}} \; \underbrace{\frac{\partial f_\theta(x)}{\partial \theta}}_{\text{INTERNAL}}$$



Our approach relies on *gradient-based* updates:

- Actions can have *feedback mechanisms:*

$$h(f_\theta(x), \mathcal{E})$$

- Whenever a feedback mechanism triggers, a gradient-based update is performed along:

$$\text{direction} \triangleq \underbrace{h(f_\theta(x), \mathcal{E})}_{\text{EXTERNAL}} \underbrace{\frac{\partial f_\theta(x)}{\partial \theta}}_{\text{INTERNAL}}$$

SUPERVISED LEARNING

$$\frac{\partial \mathcal{L}(f_\theta(x), \mathcal{E})}{\partial f_\theta(x)}$$

Goal/Problem Specification

Input

Output

Supervision

Learning Mechanism

Updates the parameters of the architecture components based on the supervision signal.

Learning Mechanism

Our approach relies on *gradient-based* updates:

- Actions can have *feedback mechanisms:*

$$h(f_\theta(x), \mathscr{E})$$

- Whenever a feedback mechanism triggers, a gradient-based update is performed along:

$$\text{direction} \triangleq \underbrace{h(f_\theta(x), \mathscr{E})}_{\text{EXTERNAL}} \ \underbrace{\frac{\partial f_\theta(x)}{\partial \theta}}_{\text{INTERNAL}}$$

SUPERVISED LEARNING

$$\frac{\partial \mathscr{L}(f_\theta(x), \mathscr{E})}{\partial f_\theta(x)}$$

REINFORCEMENT LEARNING

Advantage Function

Goal/Problem Specification

Input

Output

Supervision

Learning Mechanism

Updates the parameters of the architecture components based on the supervision signal.

Learning Mechanism

Our approach relies on *gradient-based* updates:

- Actions can have *feedback mechanisms:*

$$h(f_\theta(x), \mathcal{E})$$

- Whenever a feedback mechanism triggers, a gradient-based update is performed along:

$$\text{direction} \triangleq \underbrace{h(f_\theta(x), \mathcal{E})}_{\text{EXTERNAL}} \; \underbrace{\frac{\partial f_\theta(x)}{\partial \theta}}_{\text{INTERNAL}}$$

SUPERVISED LEARNING

$$\frac{\partial \mathcal{L}(f_\theta(x), \mathcal{E})}{\partial f_\theta(x)}$$

REINFORCEMENT LEARNING

Advantage Function

We also plan to explore *staged learning* and *curriculum learning.*

[**Platanios**, Stretcu, Neubig, Poczos, Mitchell, NAACL 2019]

Goal/Problem Specification

Input

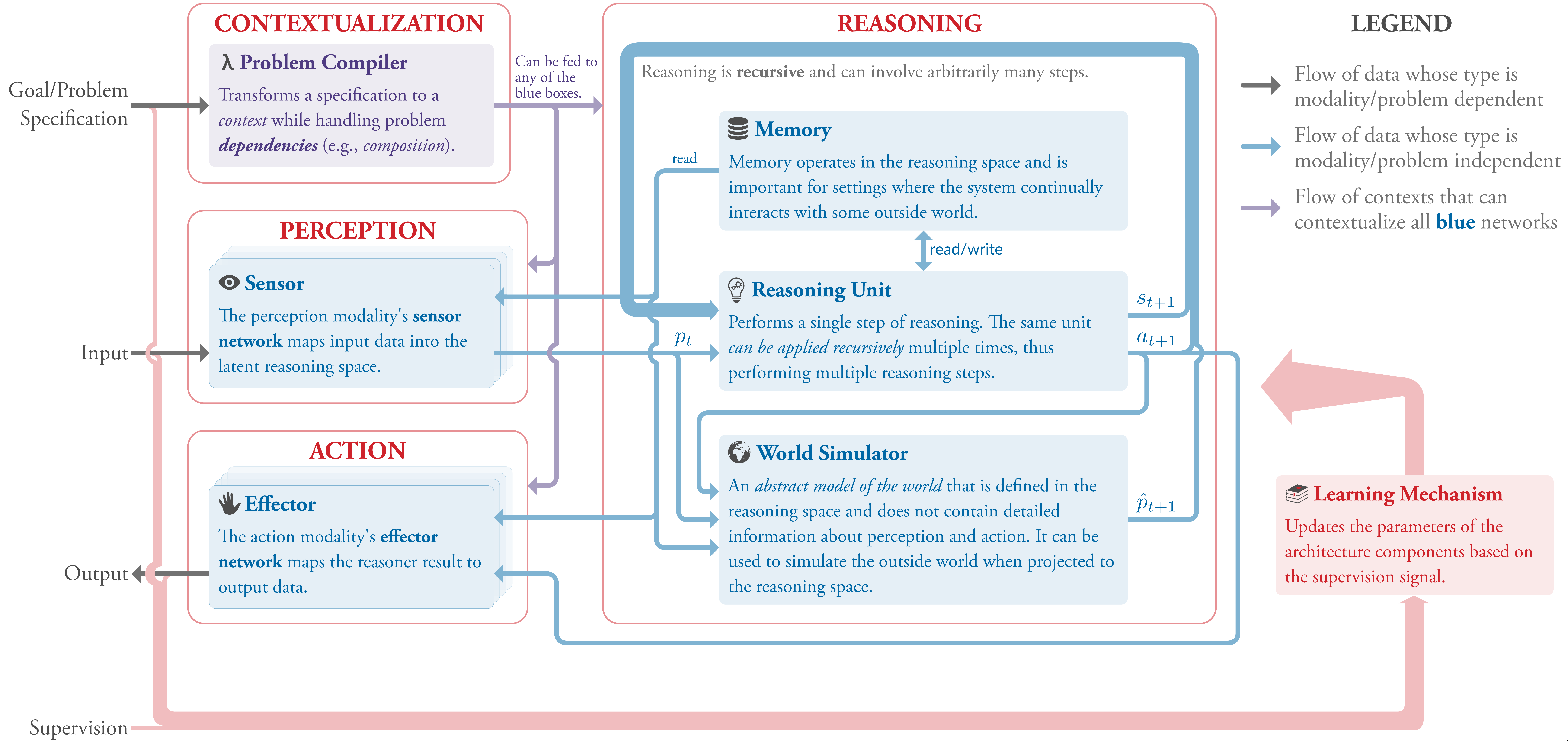
Output

Supervision

Learning Mechanism

Updates the parameters of the architecture components based on the supervision signal.

Unified Architecture



Unified Architecture: JBW Example