Modularity in Reinforcement Learning

via Algorithmic Independence in Credit Assignment


Michael Chang*


Sidhant Kaushik*


Sergey Levine


Thomas L. Griffiths

You are in the burrito business. And you make some world-class burritos.

Your team is trained to follow this policy:

Heat Tortillas

First: you heat the tortilla.

Heat Tortillas

Add Vegetables

Then: you add the vegetables.

Heat Tortillas

→

Add Vegetables

→

Add Meat

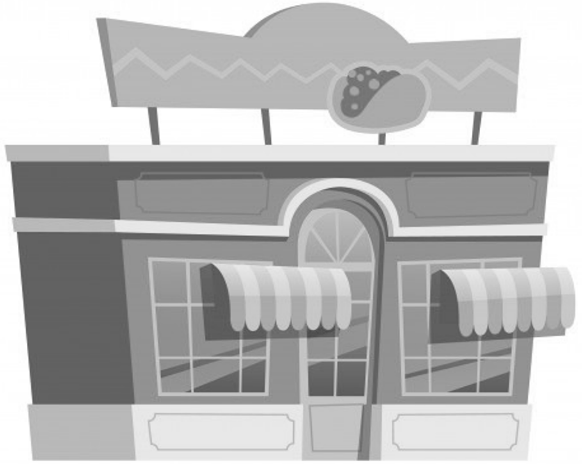
And last: you add the meat.

Heat Tortillas

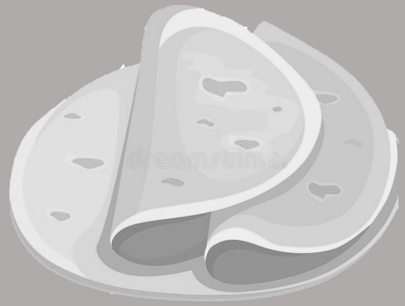
Add Vegetables

Add Meat

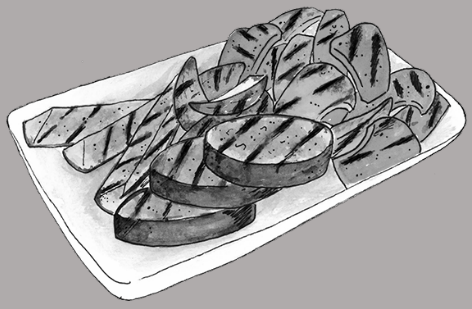
Things are going great.



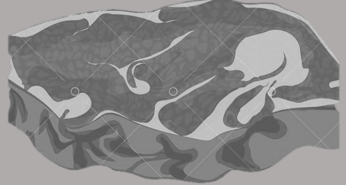
Heat Tortillas



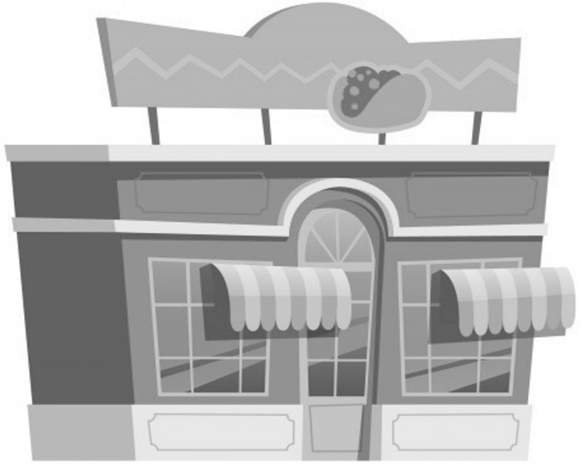
Add Vegetables



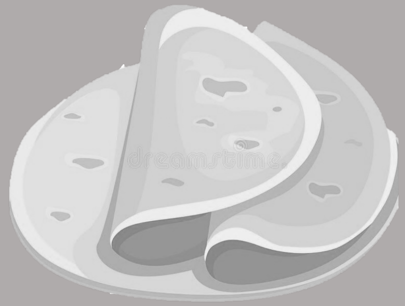
Add Meat



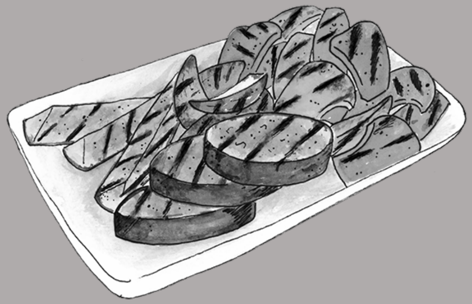
Until this week.



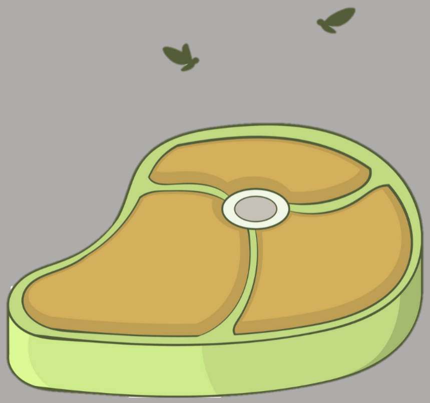
Heat Tortillas



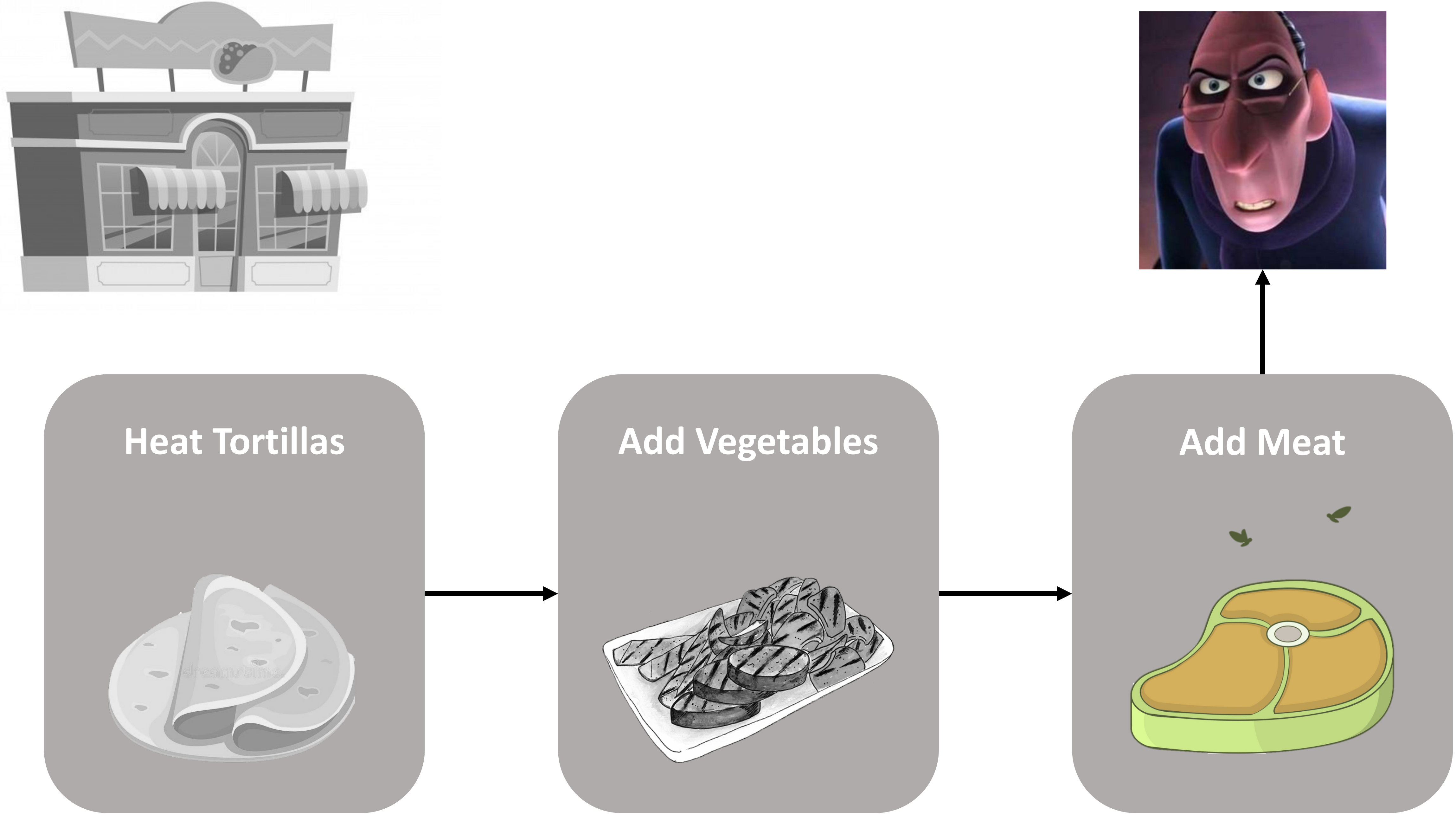
Add Vegetables



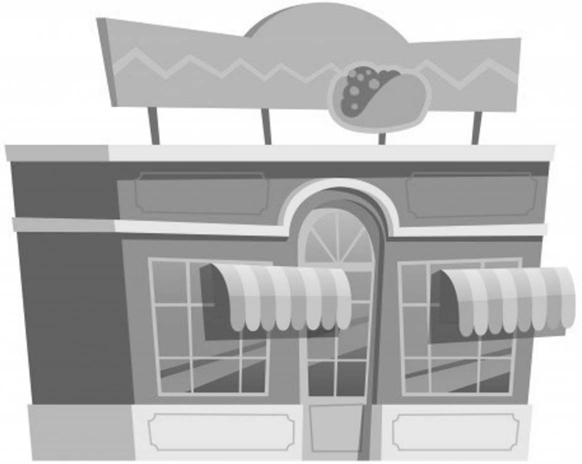
Add Meat



This week the meat was contaminated from the meat supplier.

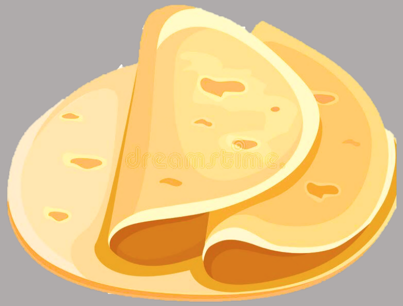


Customers got sick and gave you a lot of angry reviews.



!!!

Heat Tortillas

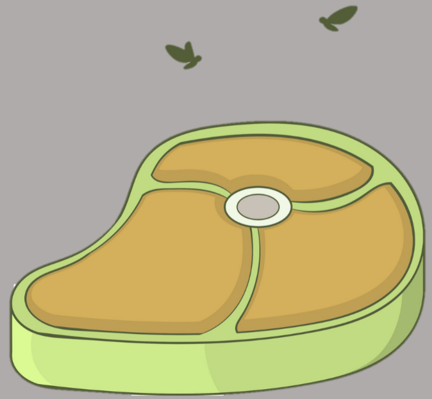


→

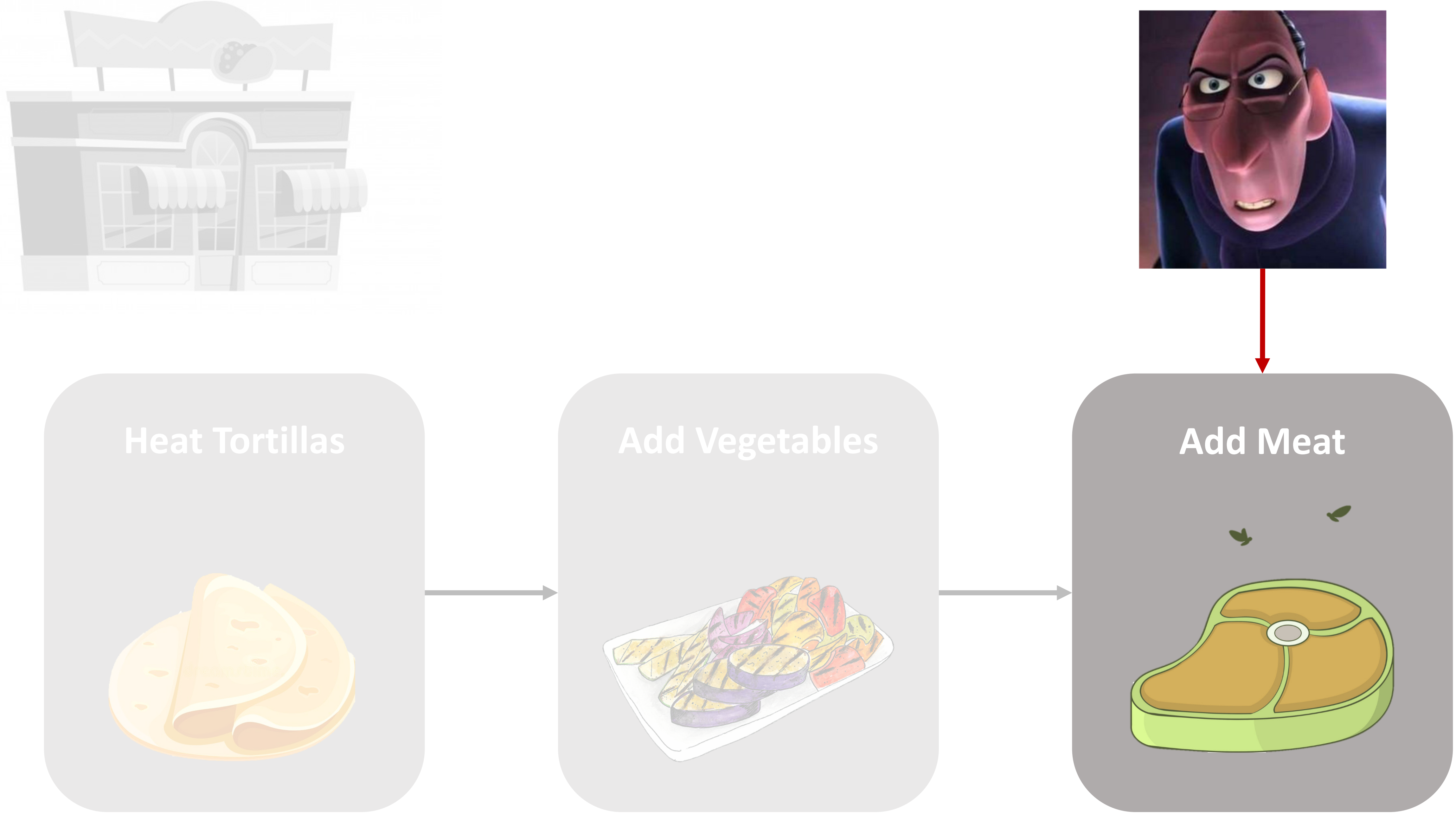
Add Vegetables

→

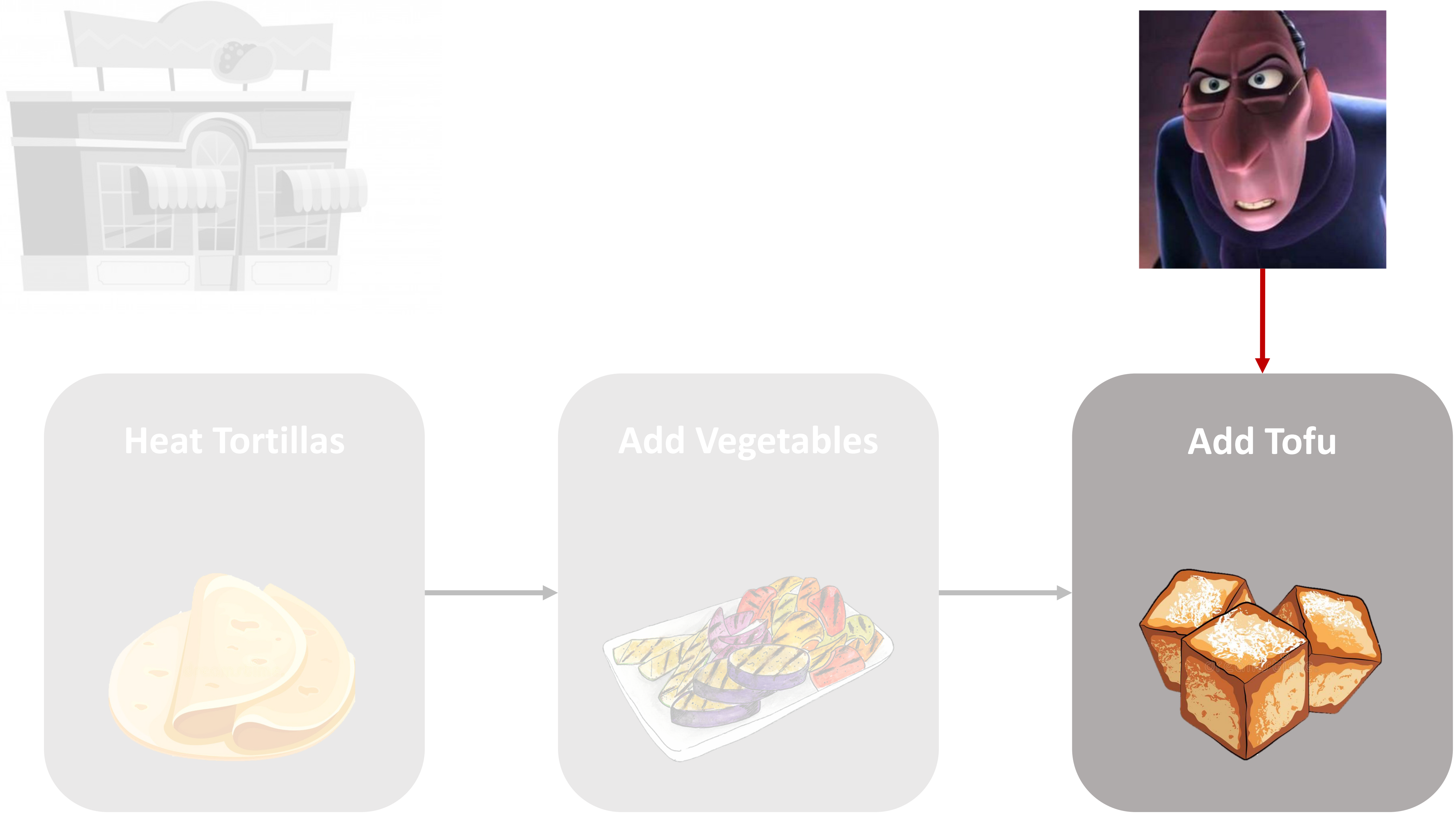
Add Meat



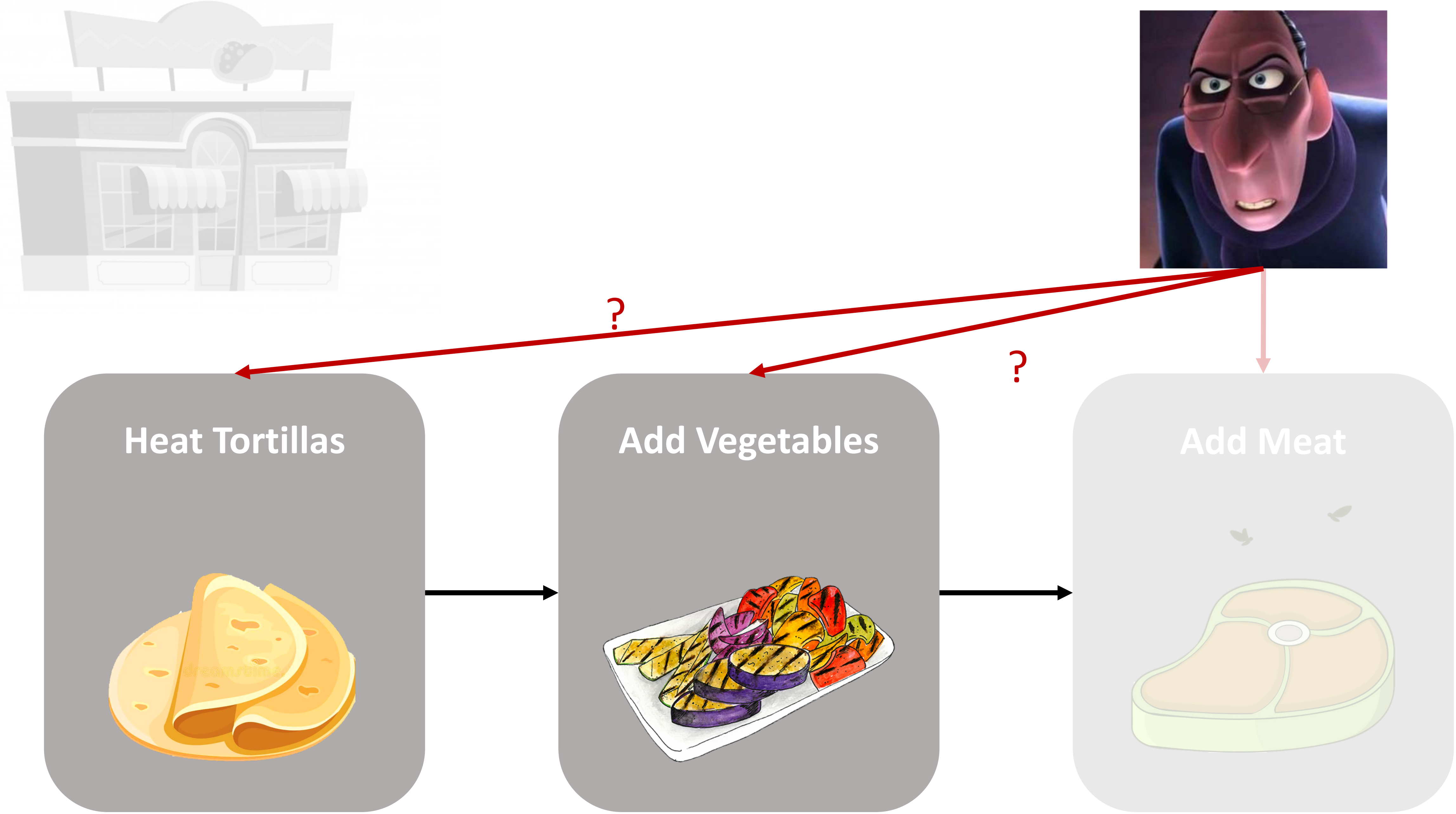
Continuing with the optimal policy from before will only make things worse.



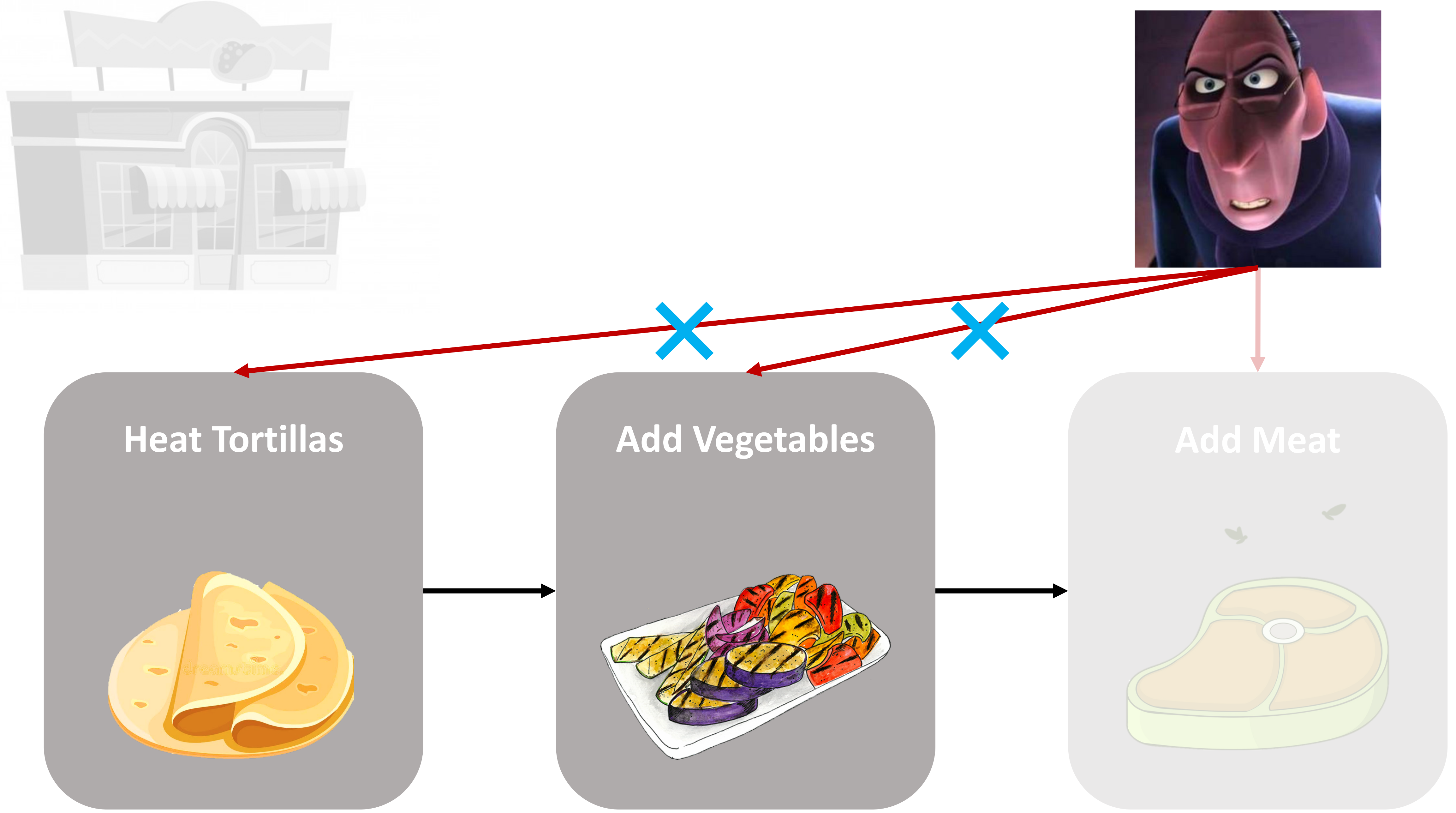
Clearly, credit assignment should modify the decision of adding meat,



perhaps replacing it with tofu.



But how should we perform credit assignment on other decisions from the same decision sequence?



Intuitively, nothing:

we should be isolating credit assignment only to the last step, without affecting anything else.

Heat Tortillas

Add Vegetables

Add Meat

The key intuition here is that if we have a modular way to perform credit assignment,

Heat Tortillas

Add Vegetables

Add Meat

such that it is possible to modify one component without simultaneously modifying others,

Heat Tortillas

Add Vegetables

Add Meat

such that it is possible to modify one component without simultaneously modifying others,

Heat Tortillas

Add Vegetables

Add Meat

such that it is possible to modify one component without simultaneously modifying others,

Heat Tortillas

Add Vegetables

Add Tofu

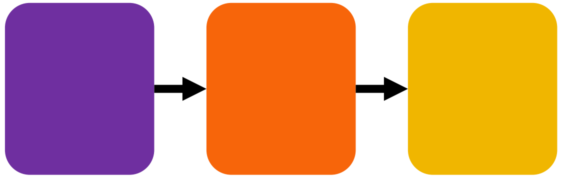
then the system could more efficiently adapt to new contexts that have not been anticipated before.

Problem

Unfortunately, popular learning algorithms do not seem to do a good job at this.

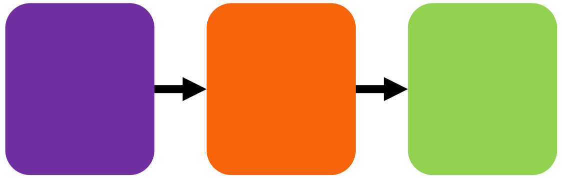
Problem

Training



optimal decision sequence

Transfer

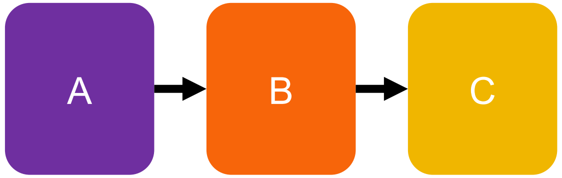


optimal decision sequence

Here is the same scenario, represented as a transfer problem in a simple Markov decision process.

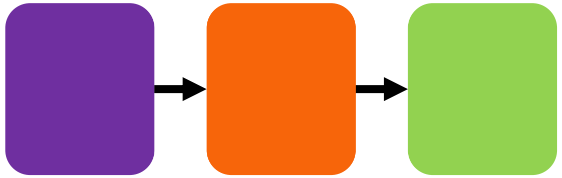
Problem

Training



optimal decision sequence

Transfer

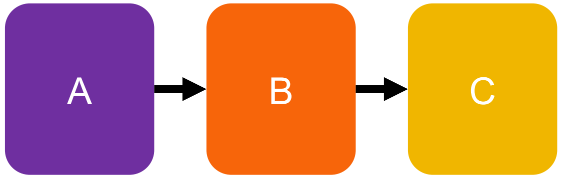


optimal decision sequence

In the training task, the optimal policy is the sequence of actions A, B, then C.

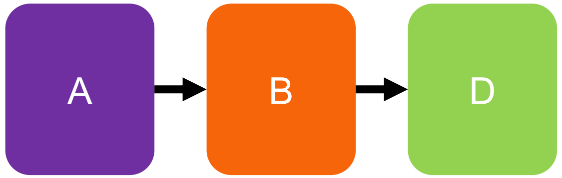
Problem

Training



optimal decision sequence

Transfer

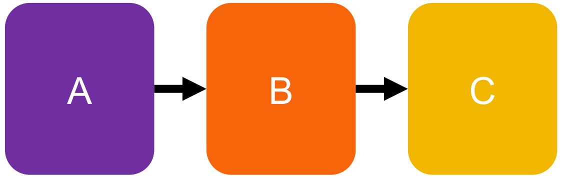


optimal decision sequence

In the transfer task, the optimal last action switches from C to D.

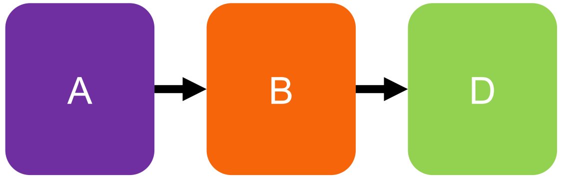
Problem

Training

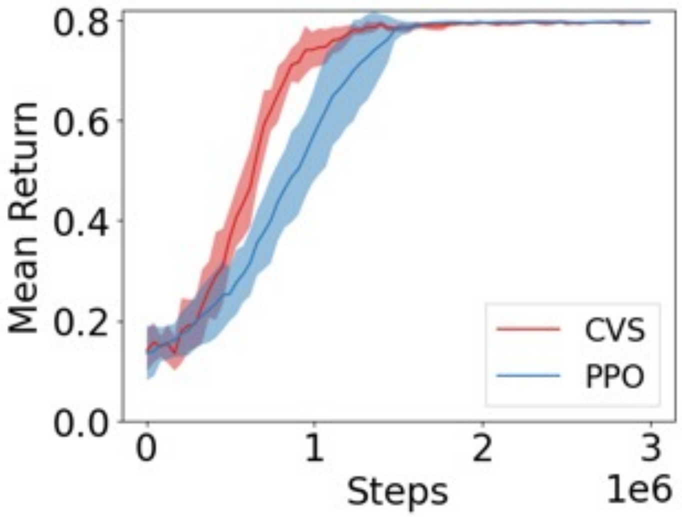


optimal decision sequence

Transfer



optimal decision sequence

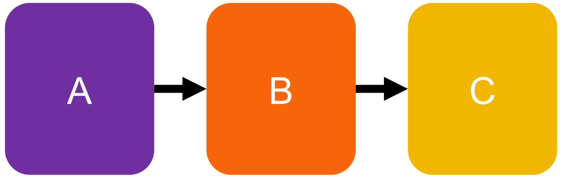


PPO: Schulman, John, et al. "Proximal policy optimization algorithms." (2017).

The blue curve represents PPO, an on-policy policy gradient method.

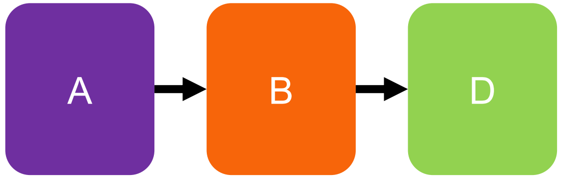
Problem

Training

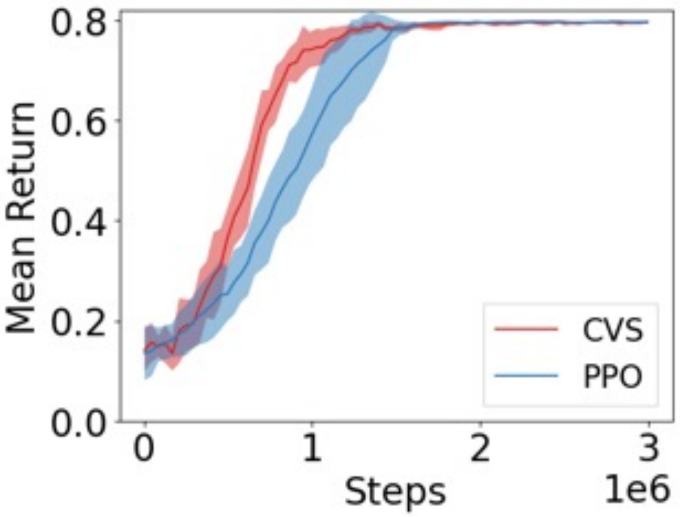


optimal decision sequence

Transfer



optimal decision sequence



PPO: Schulman, John, et al. "Proximal policy optimization algorithms." (2017).

CVS: Chang, Michael, et al. "Decentralized Reinforcement Learning: Global Decision-Making via Local Economic Transactions." *ICML* (2020).

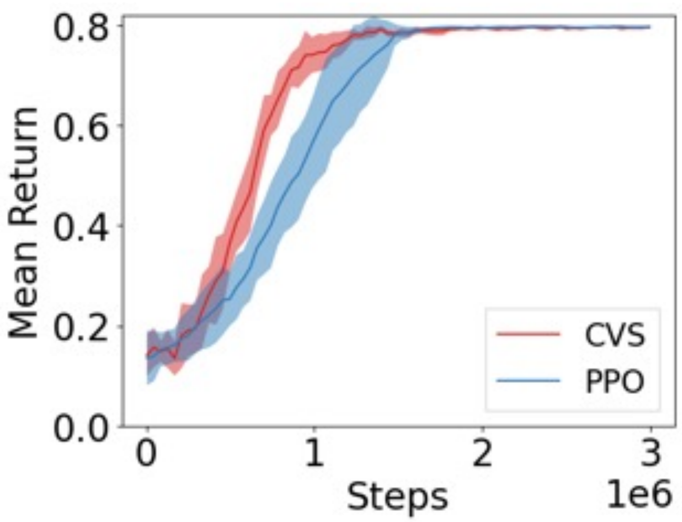
The red curve represents CVS, an on-policy single-step temporal difference method.

Problem

Training

A → B → C

optimal decision sequence



Transfer

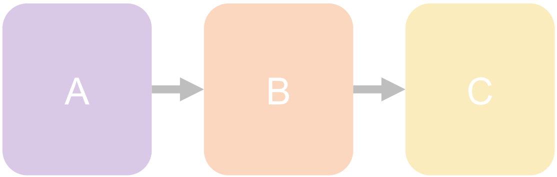
A → B → D

optimal decision sequence

Both have similar learning efficiency on the training task.

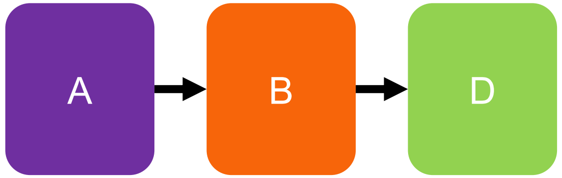
Problem

Training

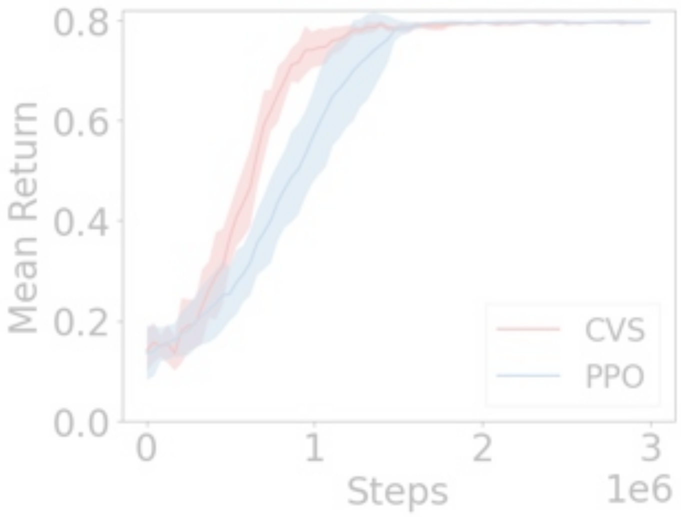


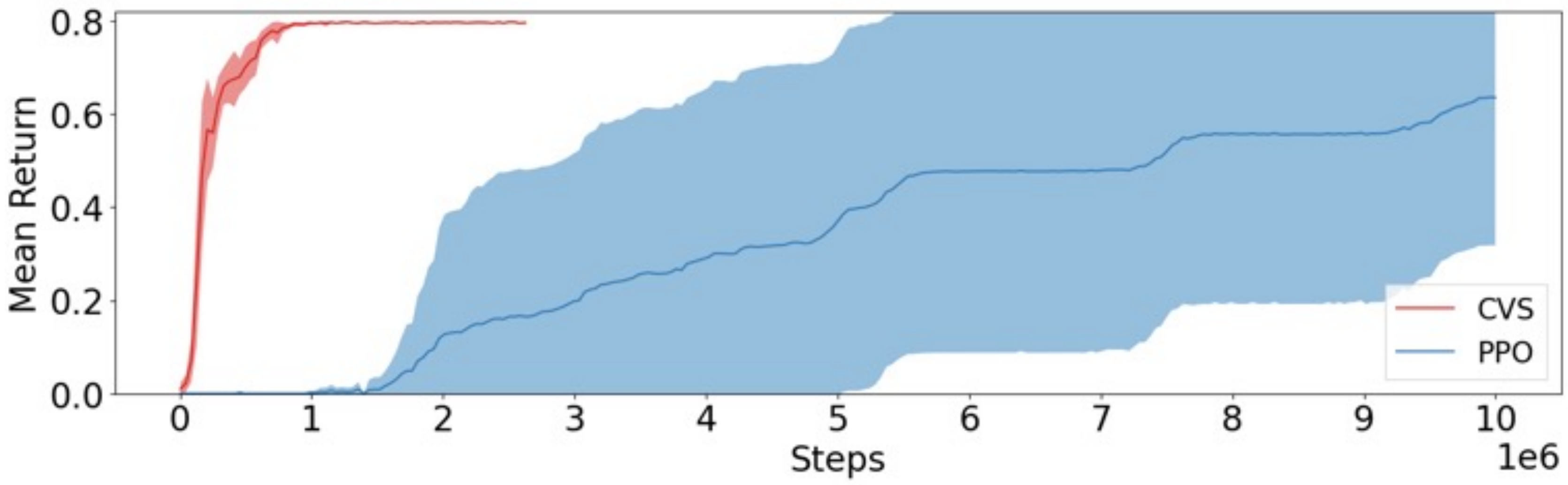
optimal decision sequence

Transfer



optimal decision sequence

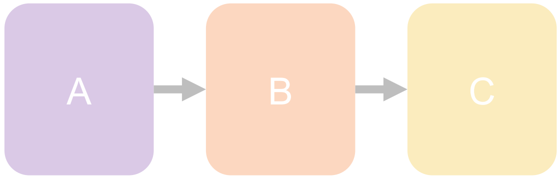




But when we transfer, PPO is not efficient at all at adapting to the new situation.

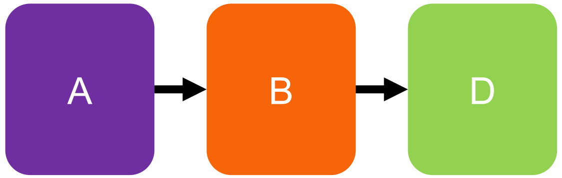
Problem

Training

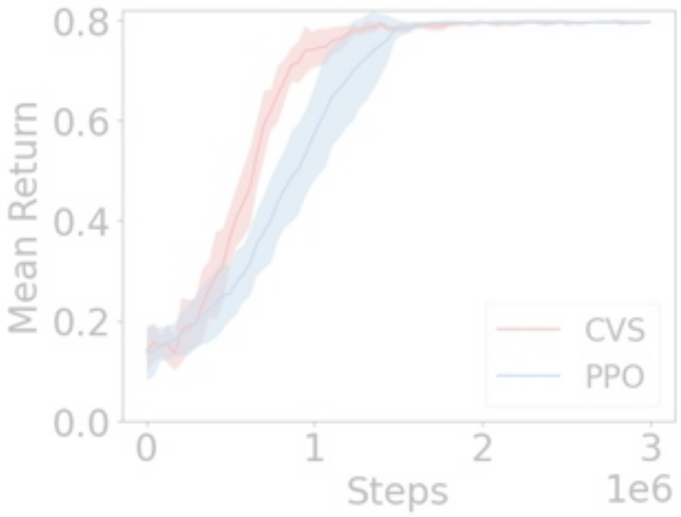


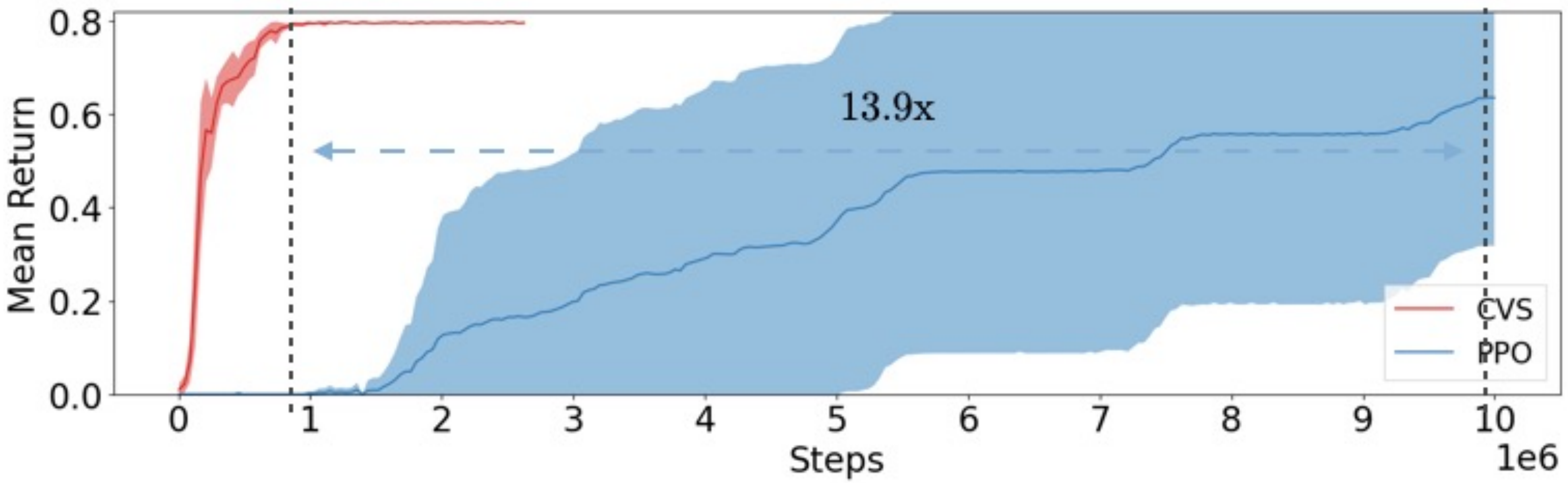
optimal decision sequence

Transfer



optimal decision sequence

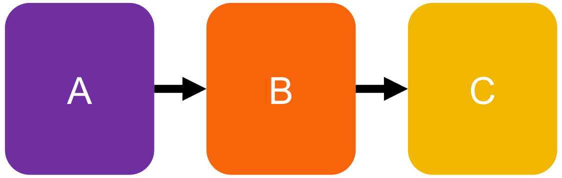




It converges 13.9 times as slow as CVS on average.

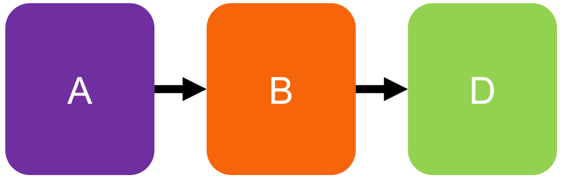
Problem

Training

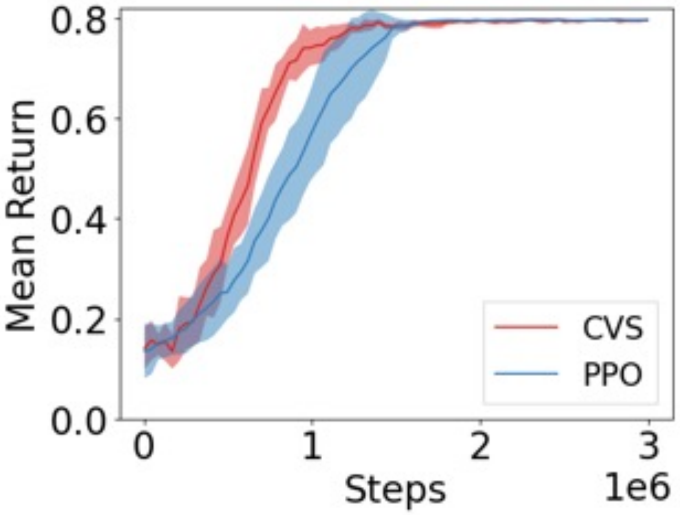


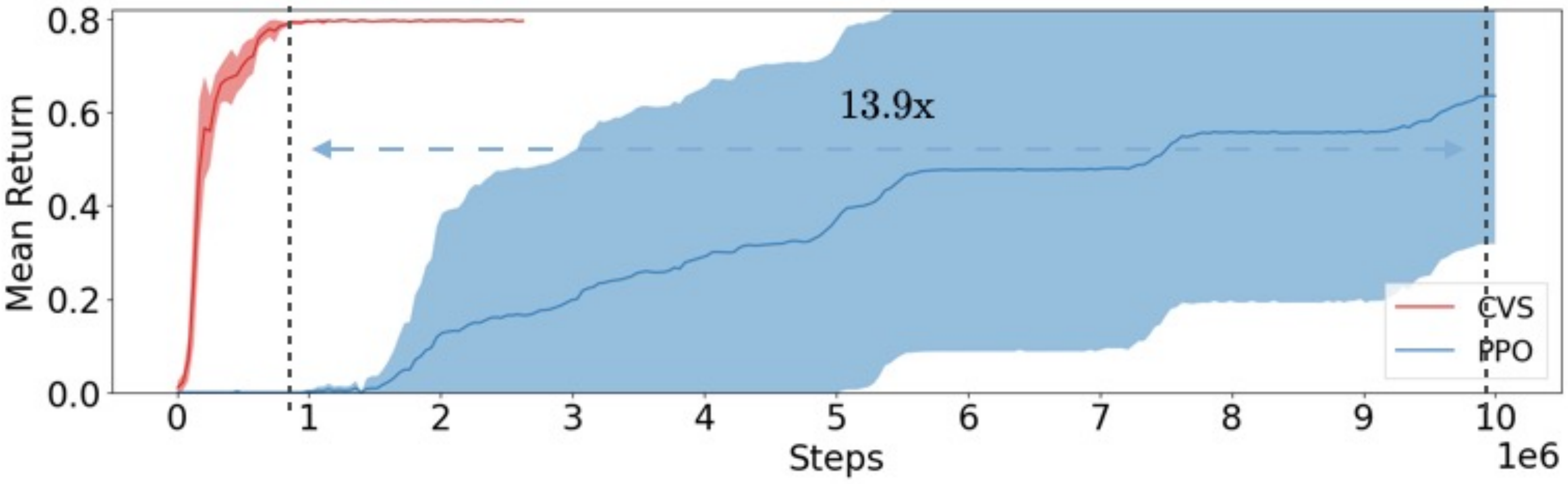
optimal decision sequence

Transfer



optimal decision sequence





This requires explanation. What is the cause of this enormous gap in transfer efficiency?

Independent Credit Assignment

What this talk is about

In this talk, I will talk about independent credit assignment.

Independent Credit Assignment

What this talk is about

What it is, how it could explain this gap in transfer efficiency,

Independent Credit Assignment

What this talk is about

and how it can be used to design and evaluate more modular learning algorithms.

What We Want

Learning algorithms that transfer efficiently

We want algorithms that can transfer efficiently

What We Want

Learning algorithms that transfer efficiently

Re-use previously optimal decisions for solving new tasks

by re-using previously optimal decisions for solving new tasks

What We Want

Learning algorithms that transfer efficiently

Re-use previously optimal decisions for solving new tasks

Modify only what needs to be modified

modifying only what needs to be modified,

What We Want

Learning algorithms that transfer efficiently

- Re-use previously optimal decisions for solving new tasks
  - Modify only what needs to be modified
  - Do not modify what does not need to be modified

and not modifying what does not need to be modified.

What We Want

Learning algorithms that transfer efficiently

Re-use previously optimal decisions for solving new tasks

Modify only what needs to be modified

Do not modify what does not need to be modified

Hypothesis

"Modularity → more efficient transfer"

We have started with a vague hypothesis, also echoed in the machine learning community,

What We Want

Learning algorithms that transfer efficiently

Re-use previously optimal decisions for solving new tasks

Modify only what needs to be modified

Do not modify what does not need to be modified

Hypothesis

“Modularity → more efficient transfer”

that modularity could enable more efficient transfer.

What We Want

Learning algorithms that transfer efficiently

Re-use previously optimal decisions for solving new tasks

Modify only what needs to be modified

Do not modify what does not need to be modified

Hypothesis

“Modularity → more efficient transfer”

↑

What does this mean for learning systems?

But the challenge to precisely expressing this hypothesis, let alone testing it,

What We Want

Learning algorithms that transfer efficiently

Re-use previously optimal decisions for solving new tasks

Modify only what needs to be modified

Do not modify what does not need to be modified

Hypothesis

“Modularity → more efficient transfer”

↑

What does this mean for learning systems?

is that we lack a precise language for describing what modularity means in the context of a learning system

What We Want

Learning algorithms that transfer efficiently

Re-use previously optimal decisions for solving new tasks

Modify only what needs to be modified

Do not modify what does not need to be modified

Hypothesis

“Modularity → more efficient transfer”

↑

What does this mean for learning systems?

and how it depends on the structure of credit assignment.

What We Want

Learning algorithms that transfer efficiently

Re-use previously optimal decisions for solving new tasks

Modify only what needs to be modified

Do not modify what does not need to be modified

Hypothesis

"Modularity → more efficient transfer"

↑

What does this mean for learning systems?

What Currently Exists

Traditional conception of modularity: **independent causal mechanisms**

Defined for static causal graphs describing *static* systems

This is because modularity has traditionally been defined

What We Want

Learning algorithms that transfer efficiently

Re-use previously optimal decisions for solving new tasks

Modify only what needs to be modified

Do not modify what does not need to be modified

Hypothesis

“Modularity → more efficient transfer”

↑

What does this mean for learning systems?

What Currently Exists

Traditional conception of modularity: **independent causal mechanisms**

Defined for static causal graphs describing *static* systems

for static systems whose mechanisms are assumed fixed,

What We Want

Learning algorithms that transfer efficiently

Re-use previously optimal decisions for solving new tasks

Modify only what needs to be modified

Do not modify what does not need to be modified

Hypothesis

"Modularity → more efficient transfer"

↑

What does this mean for learning systems?

Learning systems are *dynamic* systems.

What Currently Exists

Traditional conception of modularity: **independent causal mechanisms**

Defined for static causal graphs describing *static* systems

whereas learning systems are dynamic systems whose mechanisms evolve over time.

What We Want

Learning algorithms that transfer efficiently

Re-use previously optimal decisions for solving new tasks

Modify only what needs to be modified

Do not modify what does not need to be modified

Hypothesis

“Modularity → more efficient transfer”

↑

What does this mean for learning systems?

Learning systems are *dynamic* systems.

Problem

What Currently Exists

Traditional conception of modularity: **independent causal mechanisms**

Defined for static causal graphs describing *static* systems

The problem we face is to extend the static notion of modularity to learning systems,

What We Want

Learning algorithms that transfer efficiently

Re-use previously optimal decisions for solving new tasks

Modify only what needs to be modified

Do not modify what does not need to be modified

Hypothesis

“Modularity → more efficient transfer”

↑

What does this mean for learning systems?

Learning systems are *dynamic* systems.

Problem

What Currently Exists

Traditional conception of modularity: **independent causal mechanisms**

Defined for static causal graphs describing *static* systems

and understand its implications for credit assignment, in particular for reinforcement learning.

What We Want

Learning algorithms that transfer efficiently

Re-use previously optimal decisions for solving new tasks

Modify only what needs to be modified

Do not modify what does not need to be modified

Hypothesis

“Modularity → more efficient transfer”

↑

What does this mean for learning systems?

Learning systems are *dynamic* systems.

Our Work

Problem

What Currently Exists

Traditional conception of modularity: **independent causal mechanisms**

Defined for static causal graphs describing *static* systems

Our work proposes a candidate solution for this problem.

Main Takeaway

To build learning algorithms that transfer efficiently,
we need independently modifiable components.

To get independently modifiable components, we need
a credit assignment mechanism whose causal structure
makes independent modification possible.

My goal for this talk is to convince you that modularity in learning systems

Main Takeaway

To build learning algorithms that transfer efficiently,
we need independently modifiable components.

To get independently modifiable components, we need
a credit assignment mechanism whose causal structure
makes independent modification possible.

requires it to be possible for the credit assignment mechanism to modify the learnable mechanisms independently.

Main Takeaway

To build learning algorithms that transfer efficiently, we need independently modifiable components.

To get independently modifiable components, we need a credit assignment mechanism whose causal structure makes independent modification possible.

Modularity is just as much about independent credit assignment as it is about independent learnable functions.

This talk is organized into three parts.

Modularity for Dynamic Systems

In the first part, I extend the definition of modularity developed in the causal literature to describe dynamic systems.

Dynamic Modularity

Independent Credit Assignment

In the second part, I show that learning algorithms are examples of dynamic systems and

Independent Credit Assignment

propose a test to determine whether the credit assignment mechanisms can modify the learnable components independently.

Modularity for Dynamic Systems

Independent Credit Assignment

Hypothesis

Modularity → more efficient transfer

Expressing the hypothesis precisely

At this point we will be able to test our hypothesis because we now have a precise definition of modularity for learning systems

Modularity for Dynamic Systems

Independent Credit Assignment

Hypothesis

Modularity → more efficient transfer

Expressing the hypothesis precisely

and a practical criterion for testing whether a learning algorithm meets that definition.

Modularity for Dynamic Systems

Independent Credit Assignment

Hypothesis

Modularity → more efficient transfer

Expressing the hypothesis precisely

Testing the hypothesis

Modularity of Reinforcement Learning Algorithms

And finally, I test our hypothesis on discrete-action reinforcement learning algorithms.

Let’s start with modularity for dynamic systems.

Modularity is algorithmic independence of mechanisms.

We first review modularity for static systems, formalized in the causal literature as the algorithmic independence of mechanisms.

Modularity is algorithmic independence of mechanisms.

A dynamic system encompasses a sequence of modifications to the mechanisms.

Modularity in a dynamic system is the conditional algorithmic independence of mechanisms, conditioned on its previous state.

We call a dynamic system the process that encompasses a sequence of modifications to the static system.

Modularity is algorithmic independence of mechanisms.

A dynamic system encompasses a sequence of modifications to the mechanisms.

Modularity in a dynamic system is the conditional algorithmic independence of mechanisms, conditioned on its previous state.

Viewing a system along its sequence of modifications then gives us a natural extension of modularity as

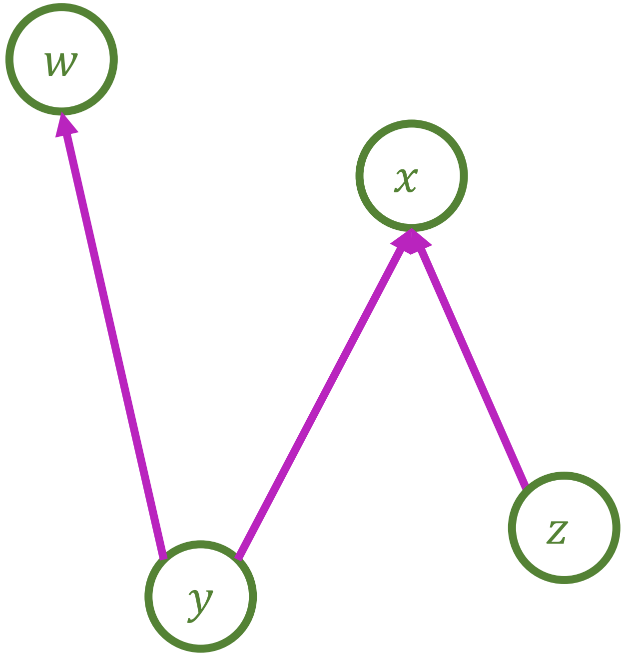
Modularity is algorithmic independence of mechanisms.

A dynamic system encompasses a sequence of modifications to the mechanisms.

Modularity in a dynamic system is the conditional algorithmic independence of mechanisms, conditioned on its previous state.

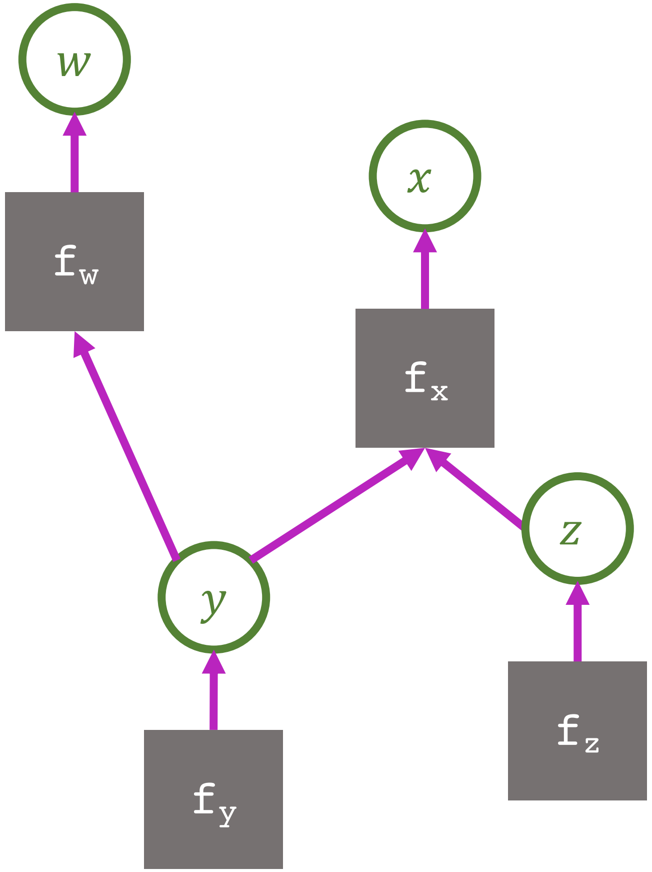
the conditional algorithmic independence of its mechanisms, conditioned on the system's previous state.

Modularity = Algorithmic Independence



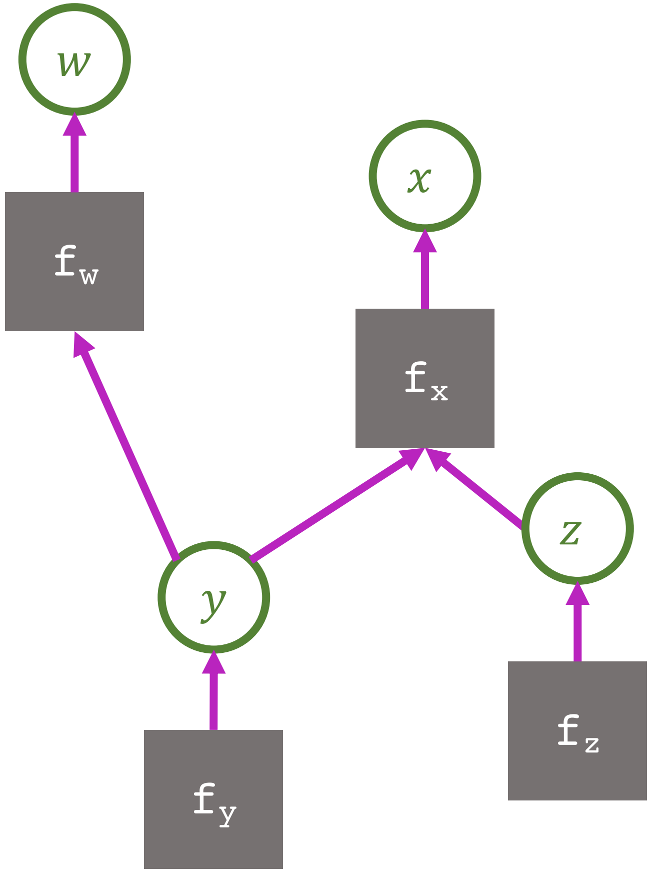
Here is a causal graph.

Modularity = Algorithmic Independence



This is the same graph, with the mechanisms that produce each node drawn explicitly.

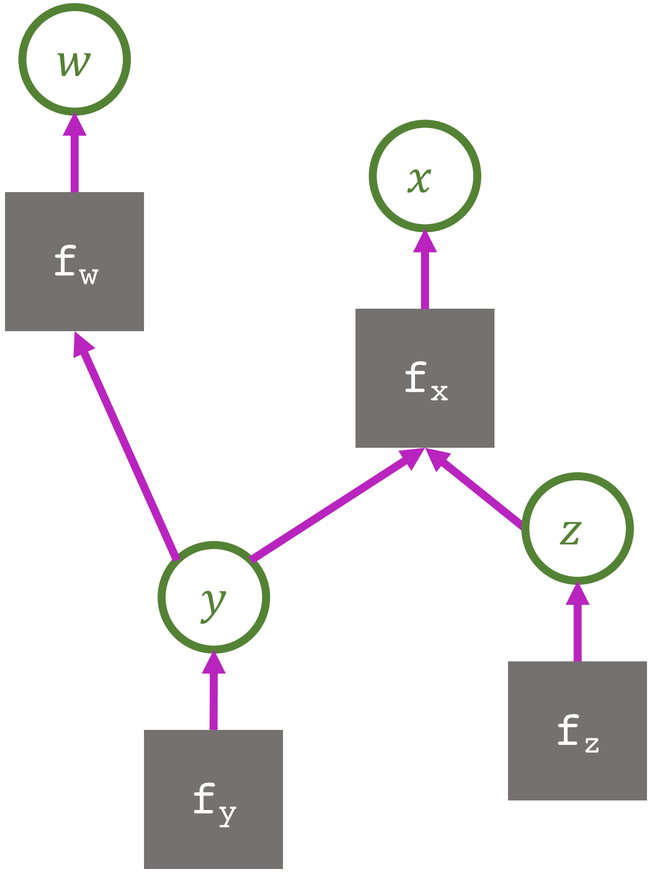
Modularity = Algorithmic Independence



$$\forall j,k: \quad I(f_j : f_k) \stackrel{+}{=} 0$$

Then modularity of this system has been previously formalized as the algorithmic independence of the mechanisms,

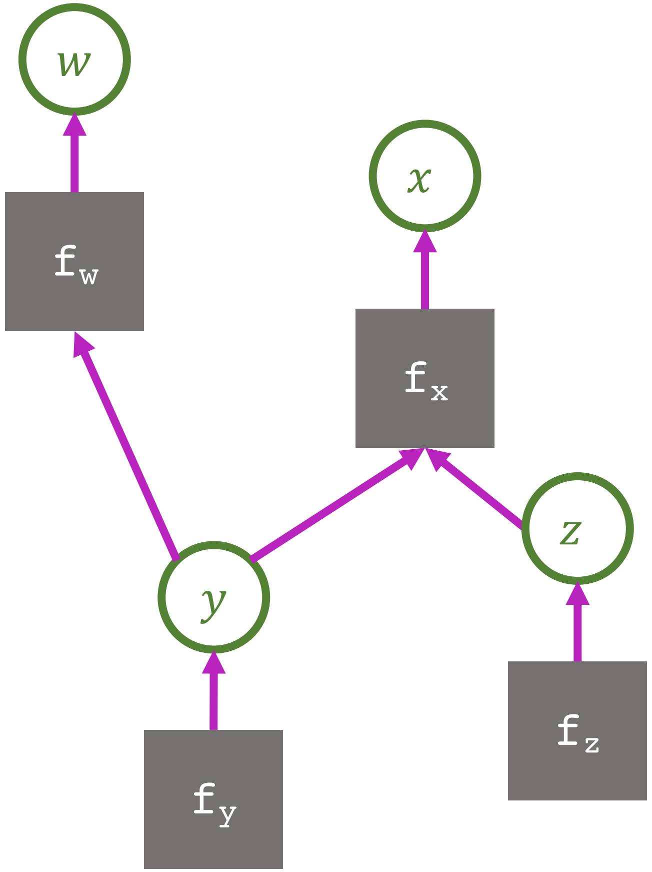
Modularity = Algorithmic Independence



$$\forall j,k: \quad I\left(f_j : f_k\right) \stackrel{+}{=} 0$$

meaning that knowing the source code of the program that computes f_j does not simplify the program for computing f_k.

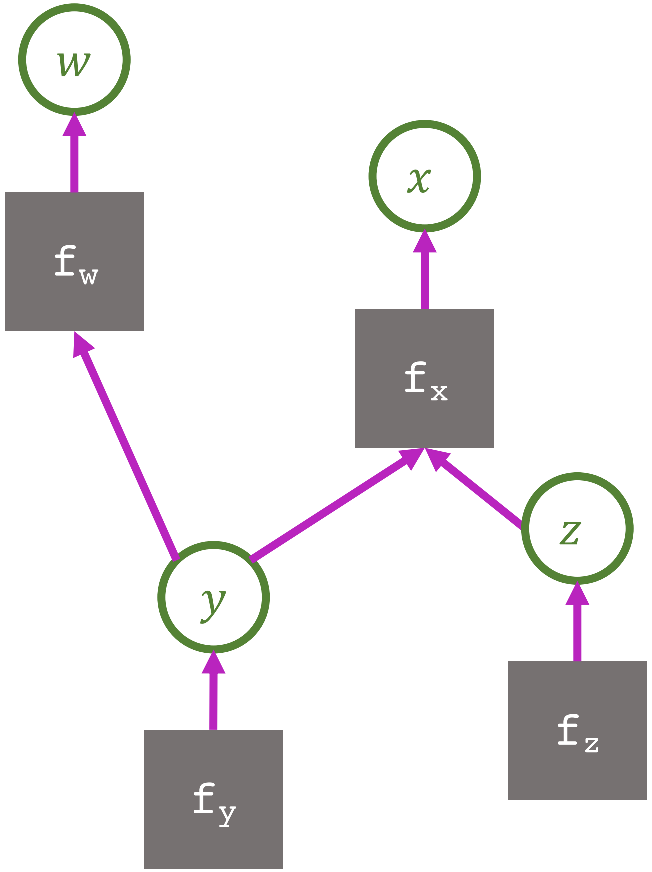
Modularity = Algorithmic Independence



$$\forall j,k: \quad I(f_j : f_k) \stackrel{+}{=} 0$$

See the paper for more background and explanation of notation.

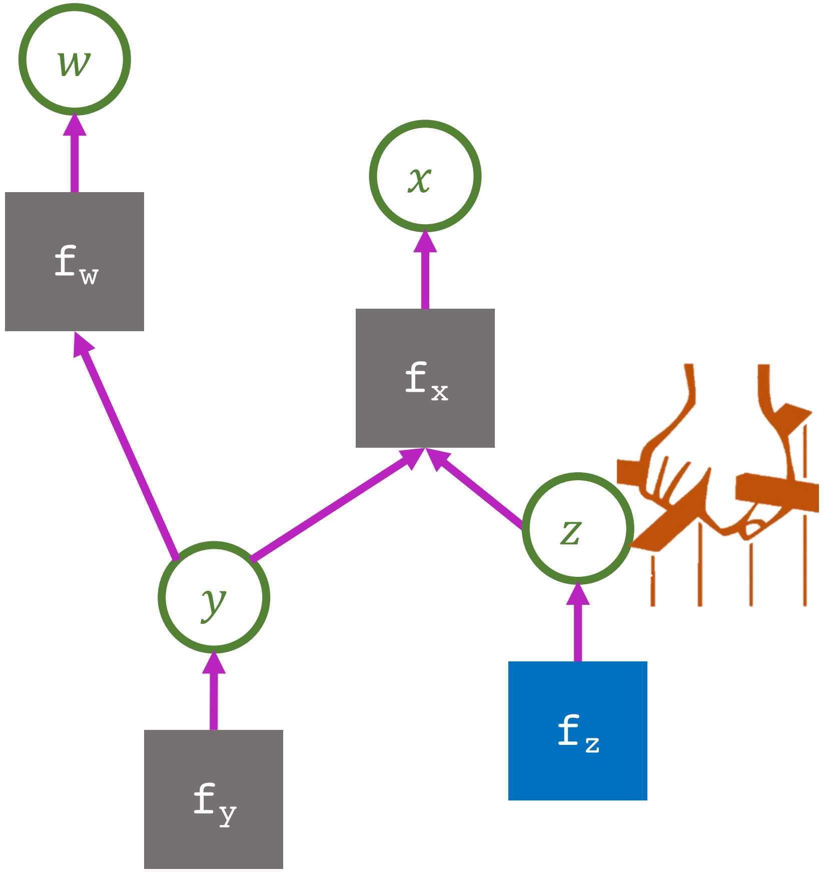
Modularity = Algorithmic Independence



$$\forall j,k: \quad I\left(f_j : f_k\right) \overset{+}{=} 0$$

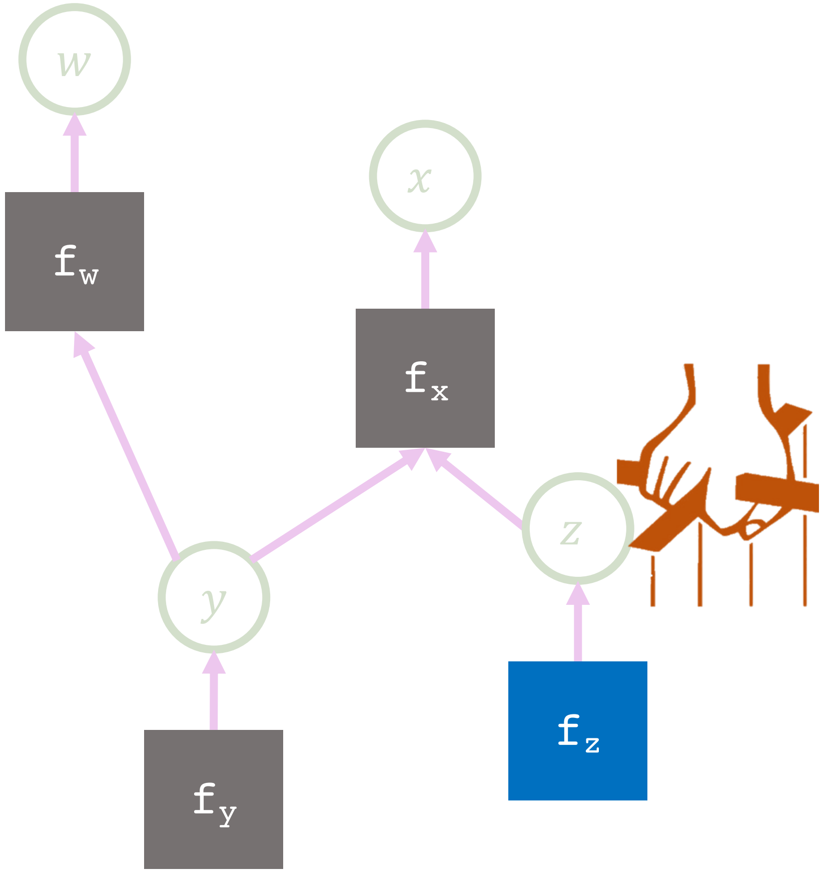
What does algorithmic independence of mechanisms give us?

Modularity = Algorithmic Independence



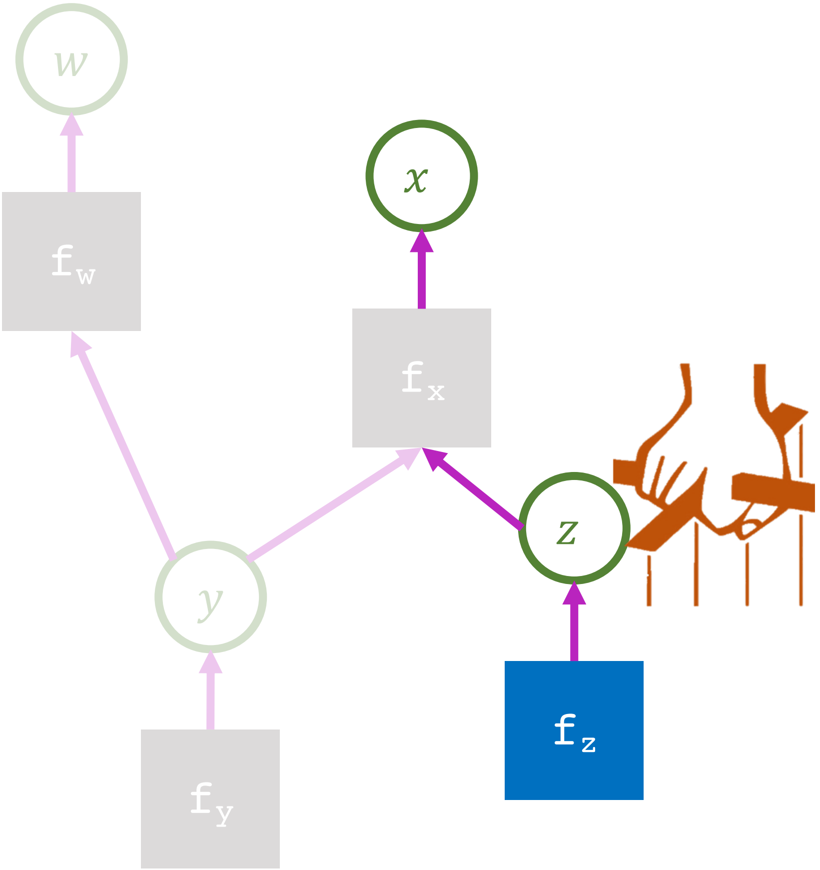
Let's intervene on the graph by modifying one of its mechanisms.

Modularity = Algorithmic Independence



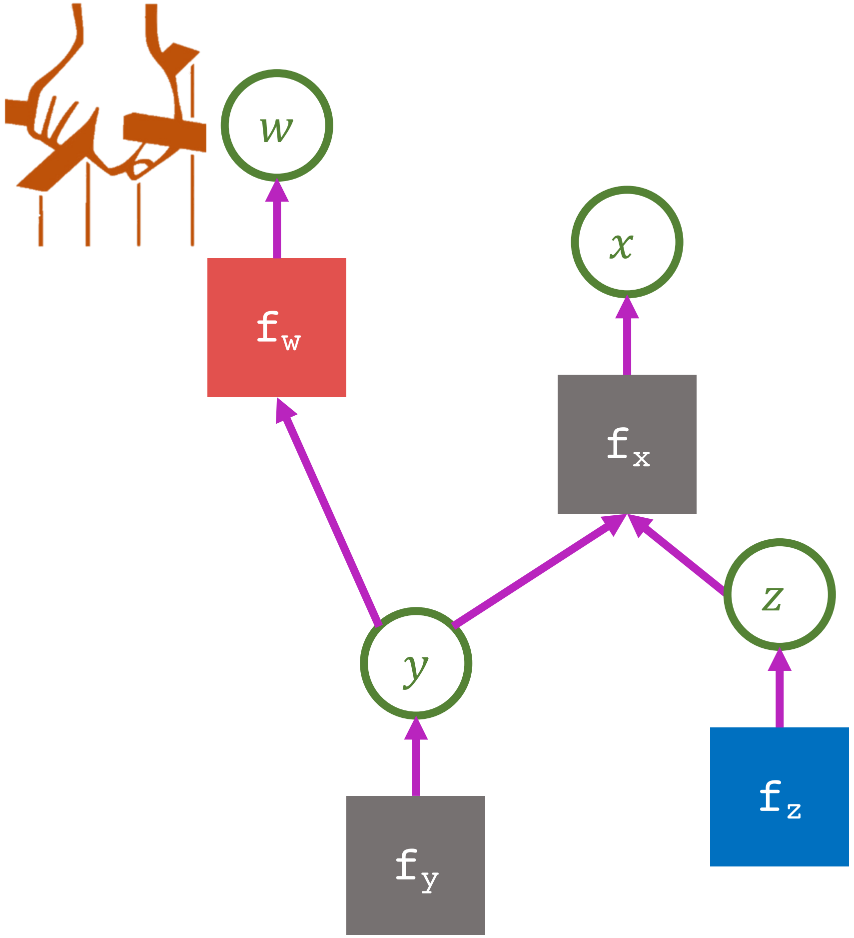
Because of algorithmic independence, this intervention does not affect the other mechanisms,

Modularity = Algorithmic Independence



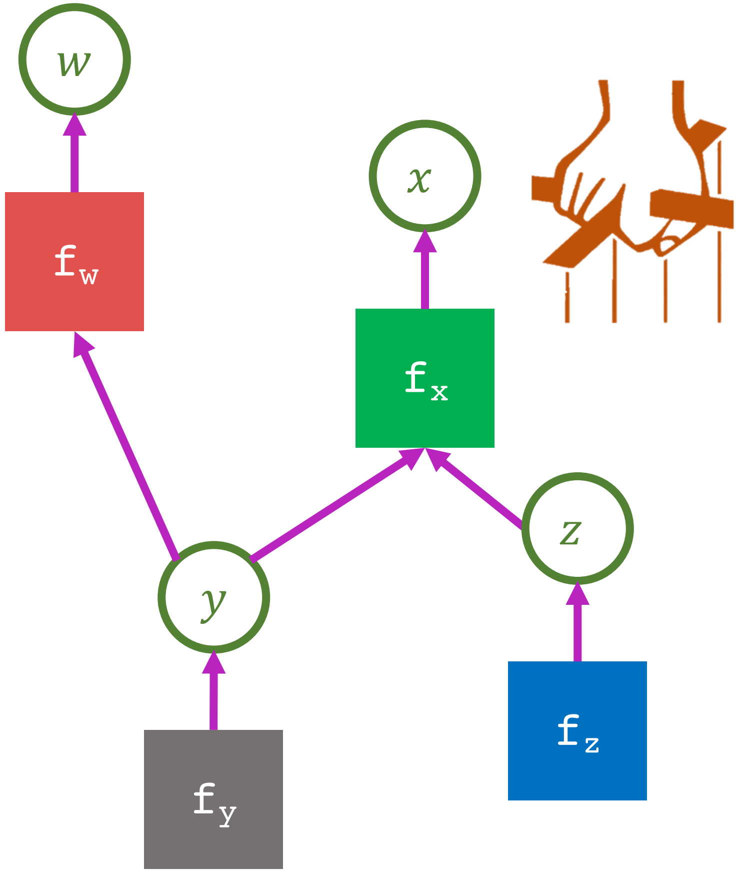
even if it affects the nodes.

Modularity = Algorithmic Independence



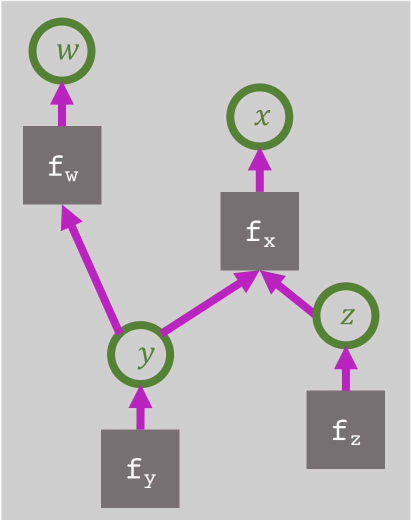
Let's intervene on the graph again by modifying another mechanism.

Modularity = Algorithmic Independence



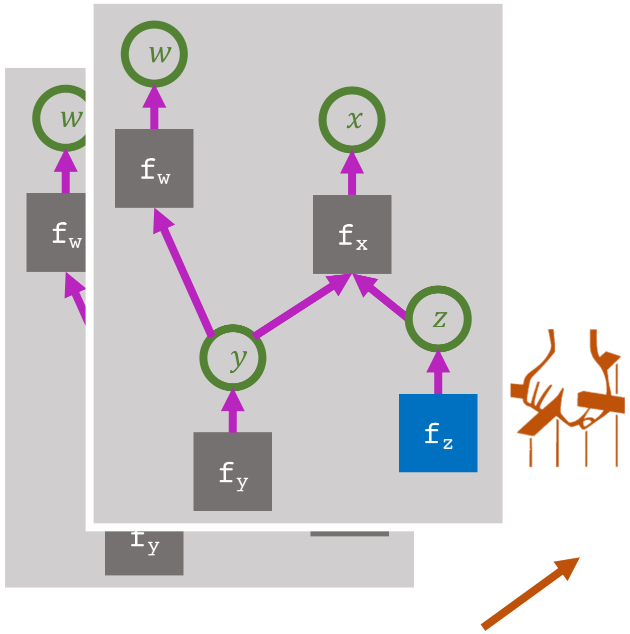
And another.

Dynamic System: A Sequence of Interventions



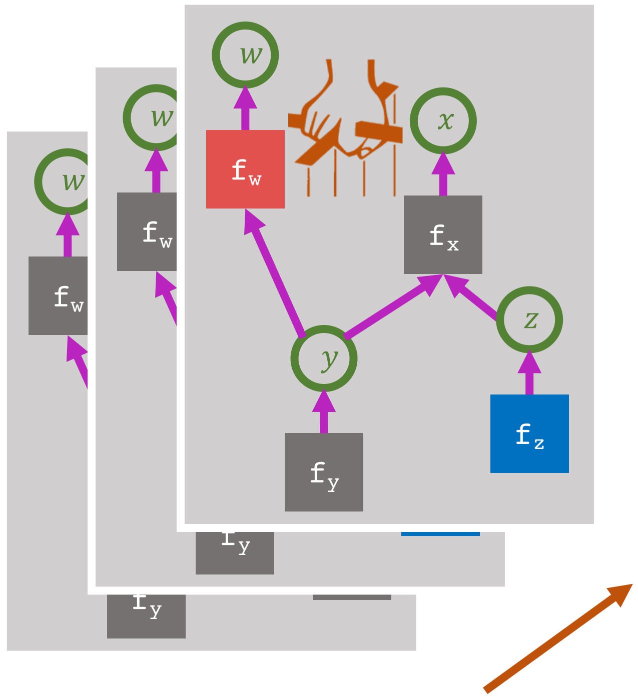
Each time we modify a mechanism we generate a new static system, represented by a new causal graph.

Dynamic System: A Sequence of Interventions



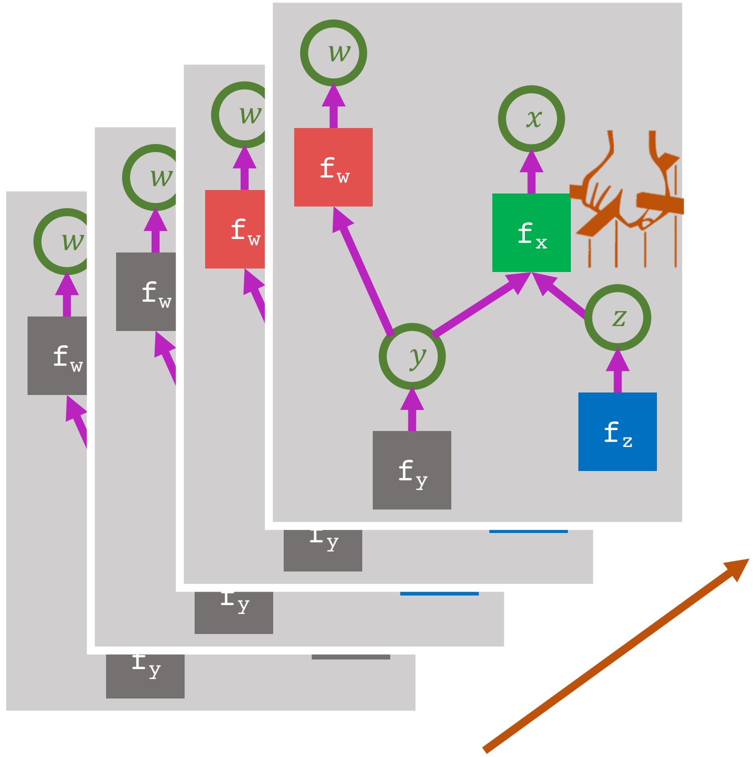
Each time we modify a mechanism we generate a new static system, represented by a new causal graph.

Dynamic System: A Sequence of Interventions



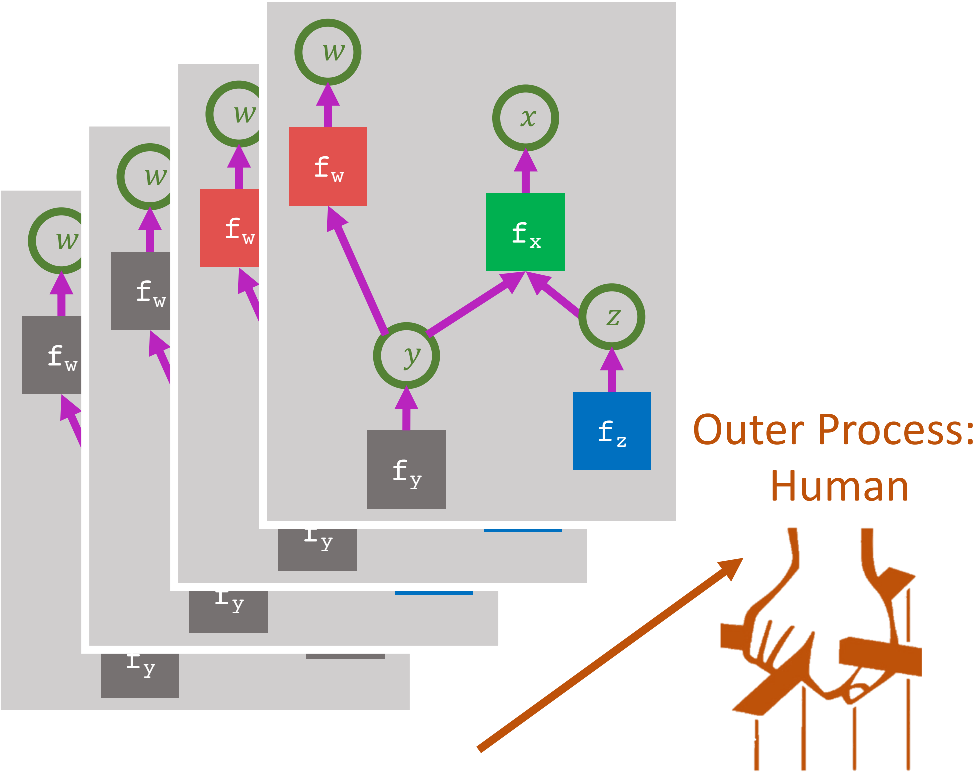
Each time we modify a mechanism we generate a new static system, represented by a new causal graph.

Dynamic System: A Sequence of Interventions



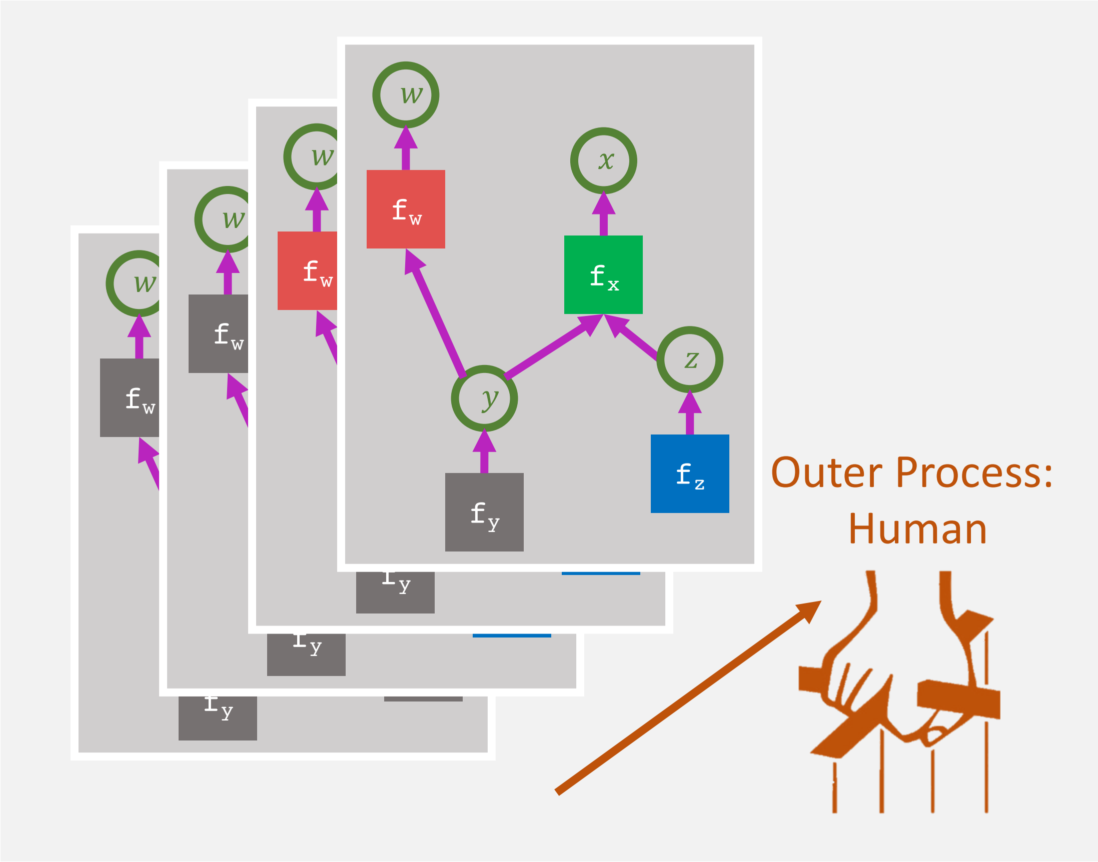
Each time we modify a mechanism we generate a new static system, represented by a new causal graph,

Dynamic System: A Sequence of Interventions



where the outer process that modifies the causal mechanism is the human.

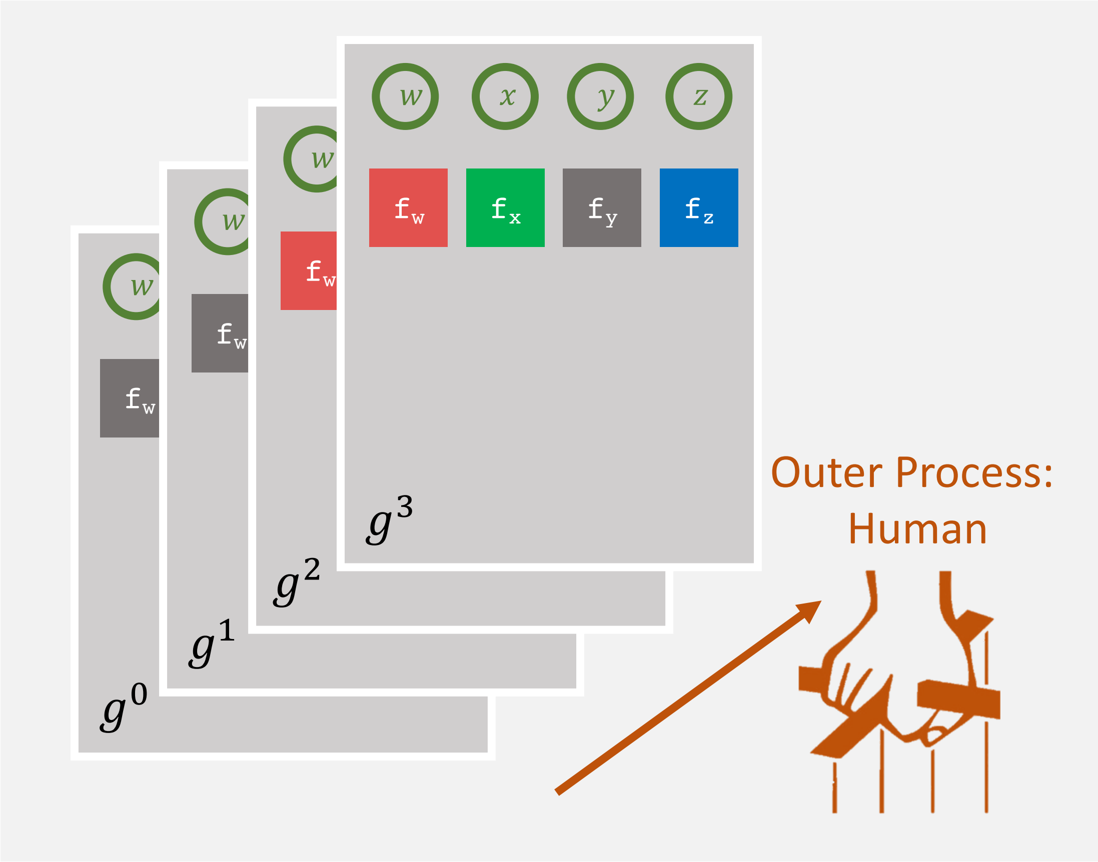
Dynamic System: A Sequence of Interventions



Dynamic system

We call the sequence of static systems, along with the outer process that modifies it, a dynamic system.

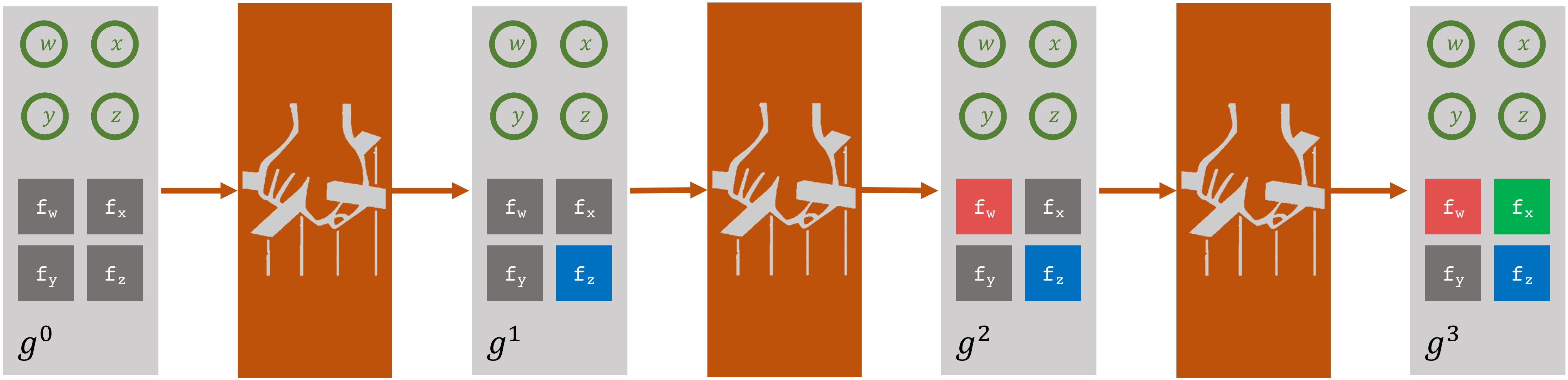
Dynamic System: A Sequence of Interventions



Dynamic system

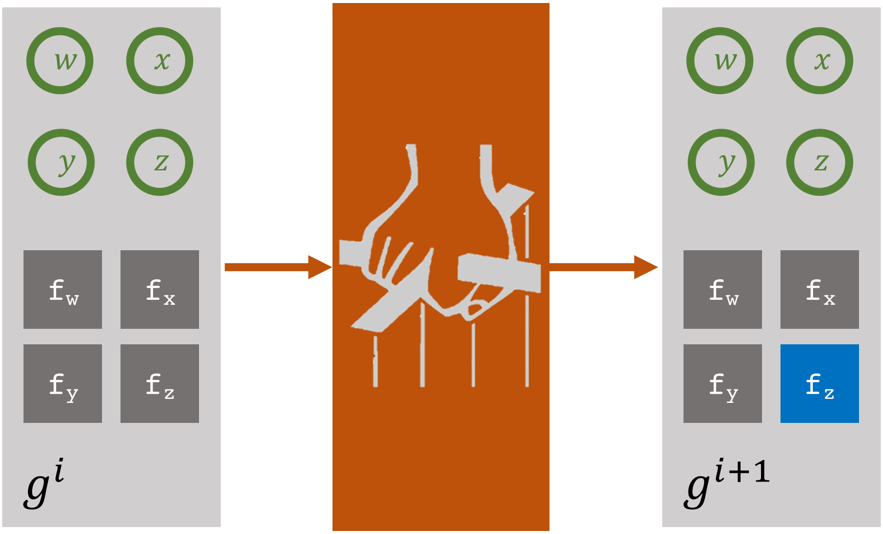
where each time-slice represents a different causal graph.

Dynamic System: A Sequence of Interventions



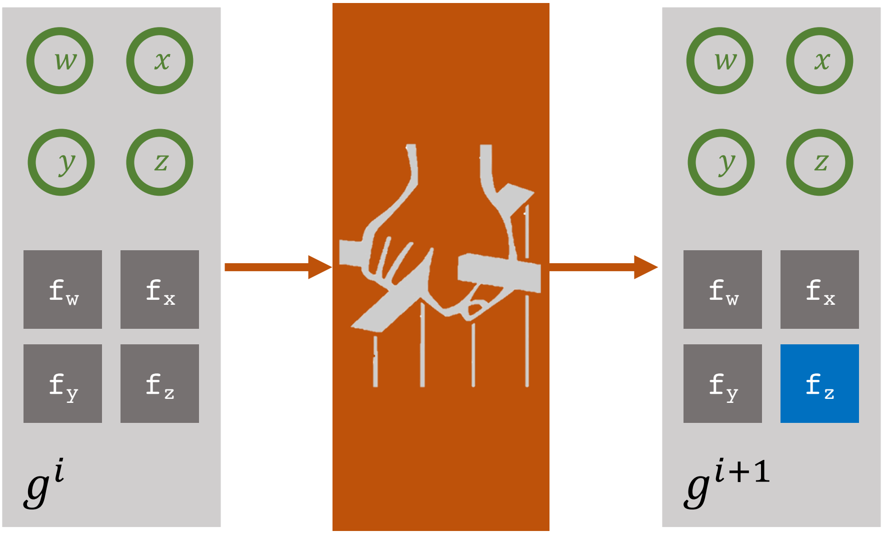
Let's unroll the outer process out more explicitly as a dynamic graph: a Markov chain over graphs.

Modularity in Dynamic Systems



Then we can naturally extend the static notion of modularity to a dynamic system

Modularity in Dynamic Systems



$$\forall j,k: \quad I\left(f_j^{i+1} : f_k^{i+1} \mid g^i\right) \stackrel{+}{=} 0$$

as the algorithmic independence of mechanisms, conditioned on the previous graph before the intervention.

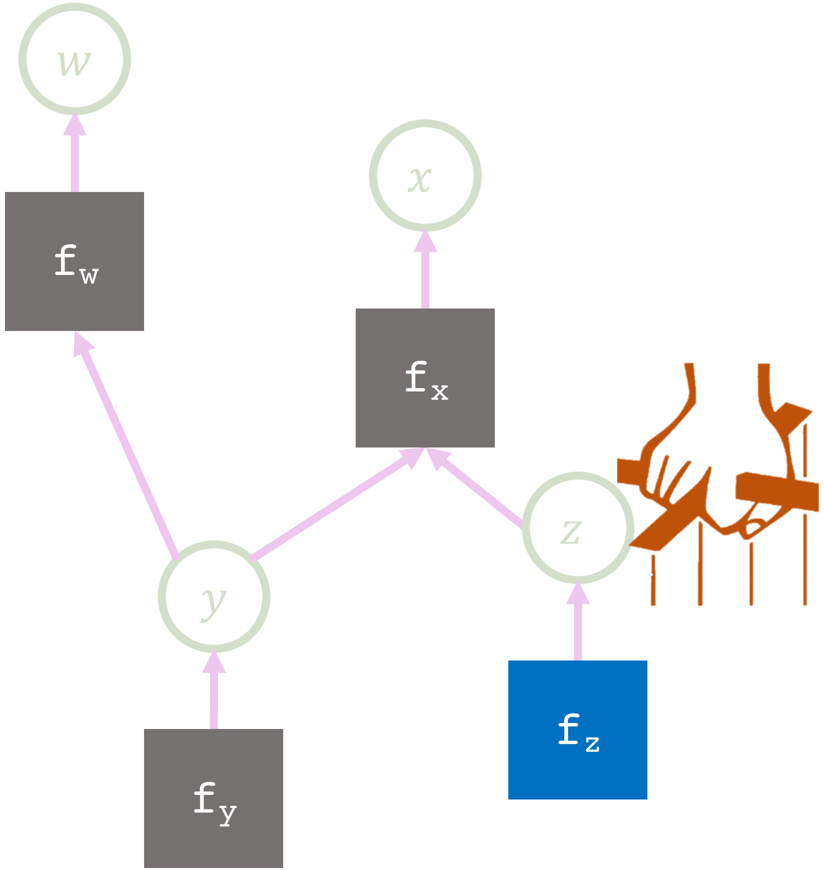
Modularity for Dynamic Systems

Independent Credit Assignment

Modularity of RL Algorithms

To recap,

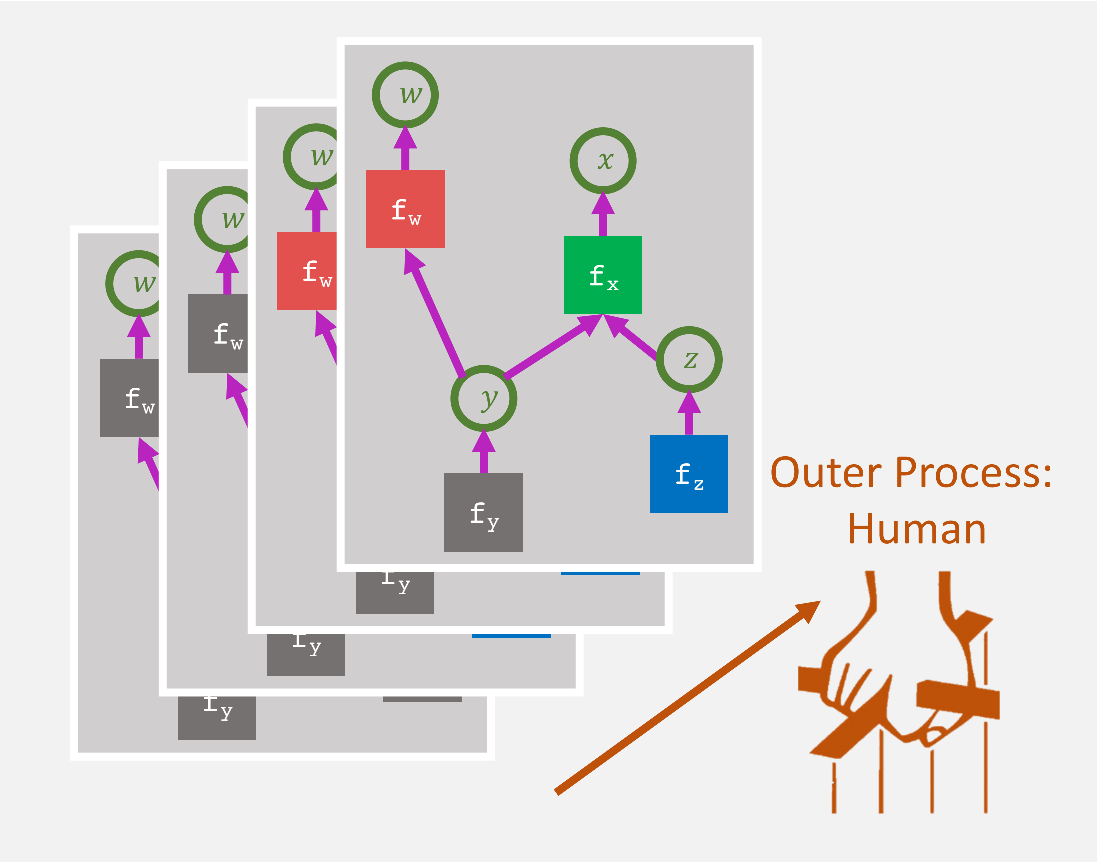
Algorithmic independence



modularity has been previously formalized as the algorithmic independence of the mechanisms of a causal graph.

Algorithmic independence

Sequence of interventions



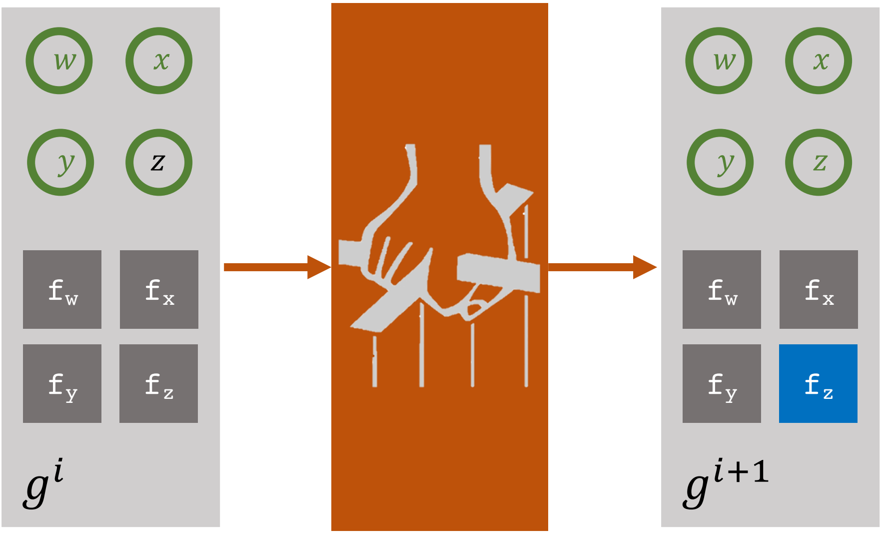
Dynamic system

A dynamic graph represents a sequence of interventions along with the outer process that performs these interventions.

Algorithmic independence

Sequence of interventions

Dynamic systems



Then we formalize modularity in a dynamic system as conditional algorithmic independence of mechanisms.

In the next part of the talk,

Learning algorithms are dynamic systems

I show that learning algorithms are themselves dynamic systems,

Modularity for Dynamic Systems

Independent Credit Assignment

Modularity of RL Algorithms

Learning algorithms are dynamic systems.

Modularity requires independent feedback (e.g. gradients).

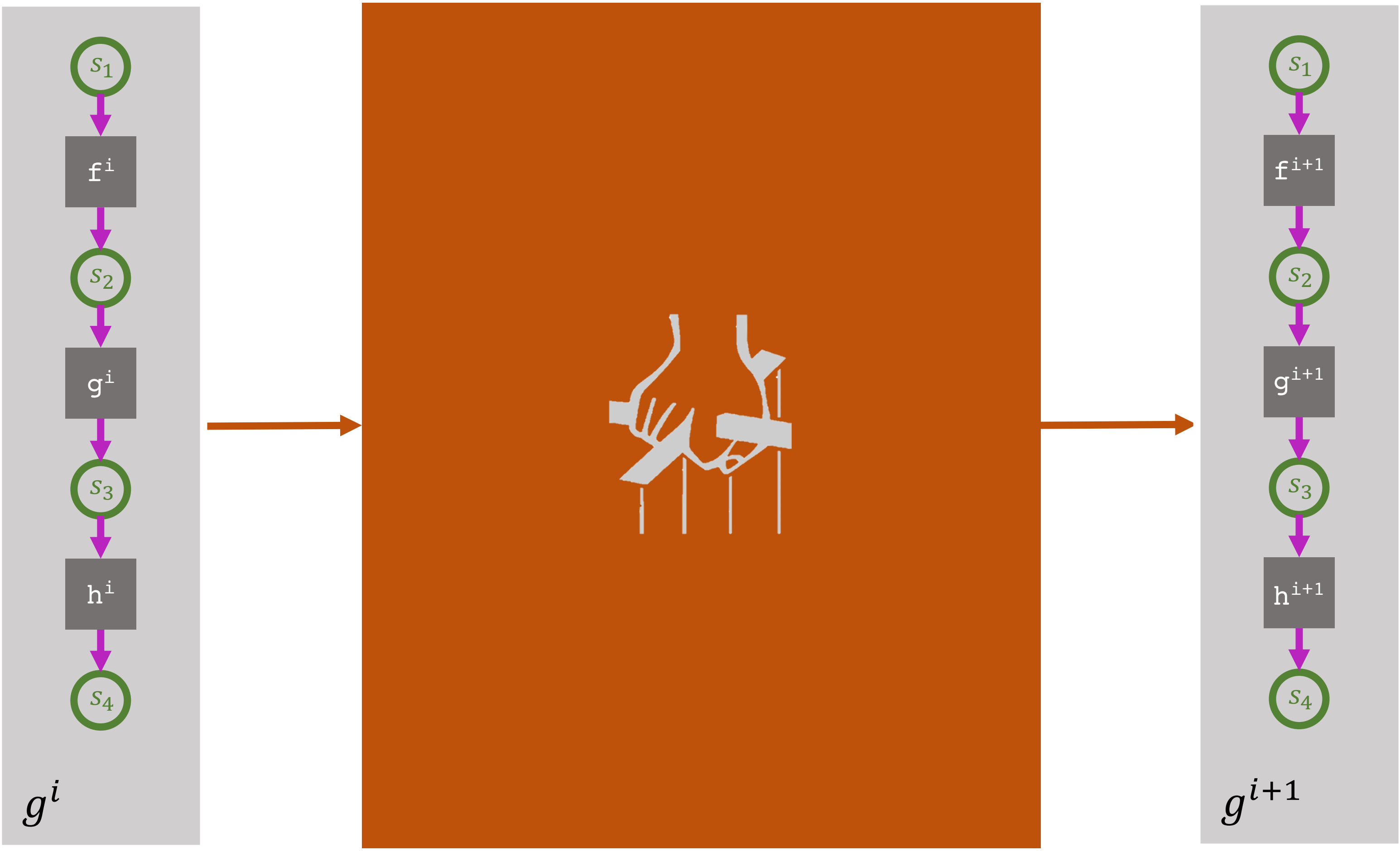
that a modular learning algorithms requires the credit assignment mechanism to produce independent gradients, Learning algorithms are dynamic systems.

Modularity requires independent feedback (e.g. gradients).

Formally represent learning algorithms as algorithmic causal graphs
independence = d-separation.

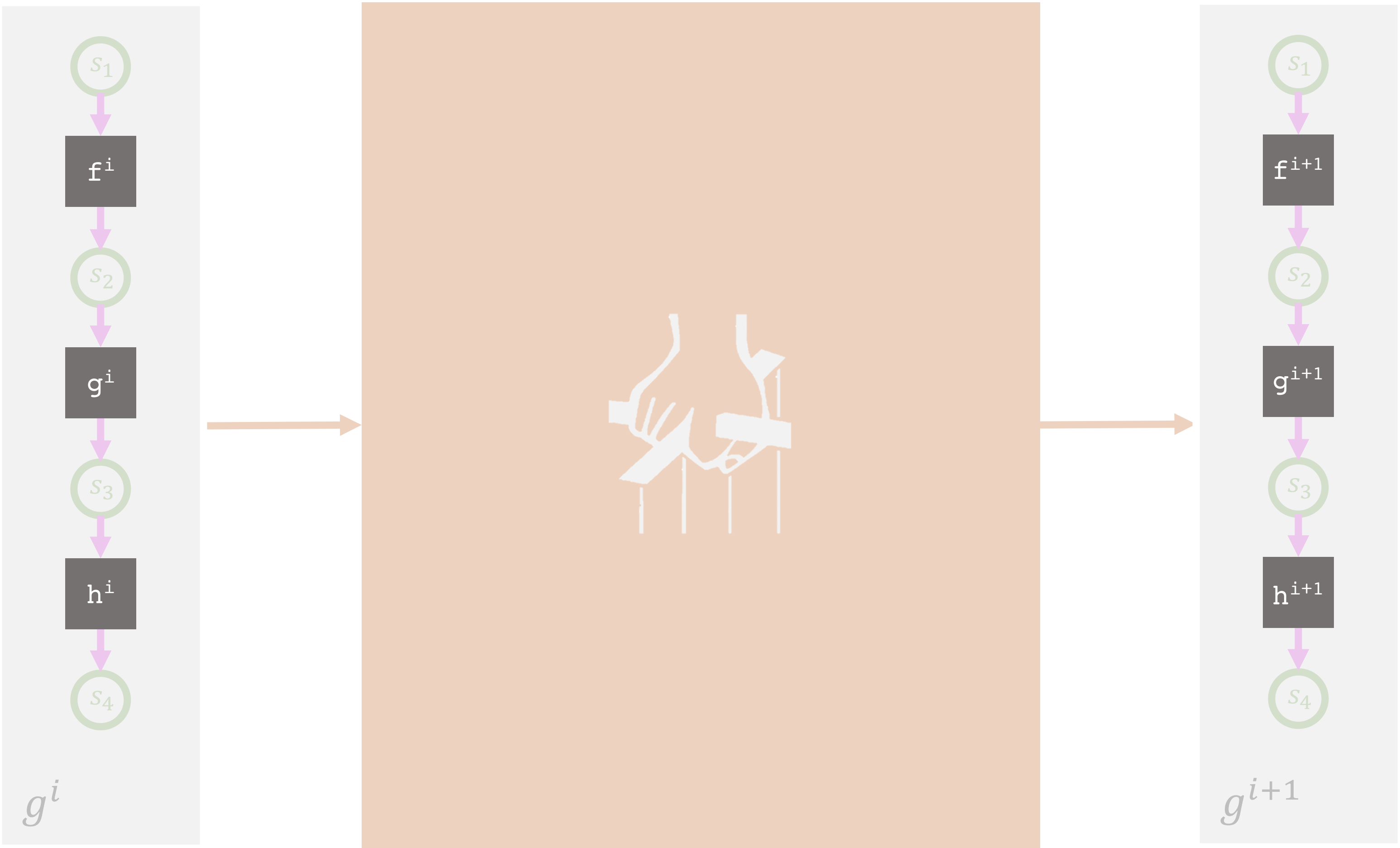
and that you can test for this property by formally treating learning algorithms as causal graphs and checking for d-separation.

Learning Algorithms are Dynamic Systems



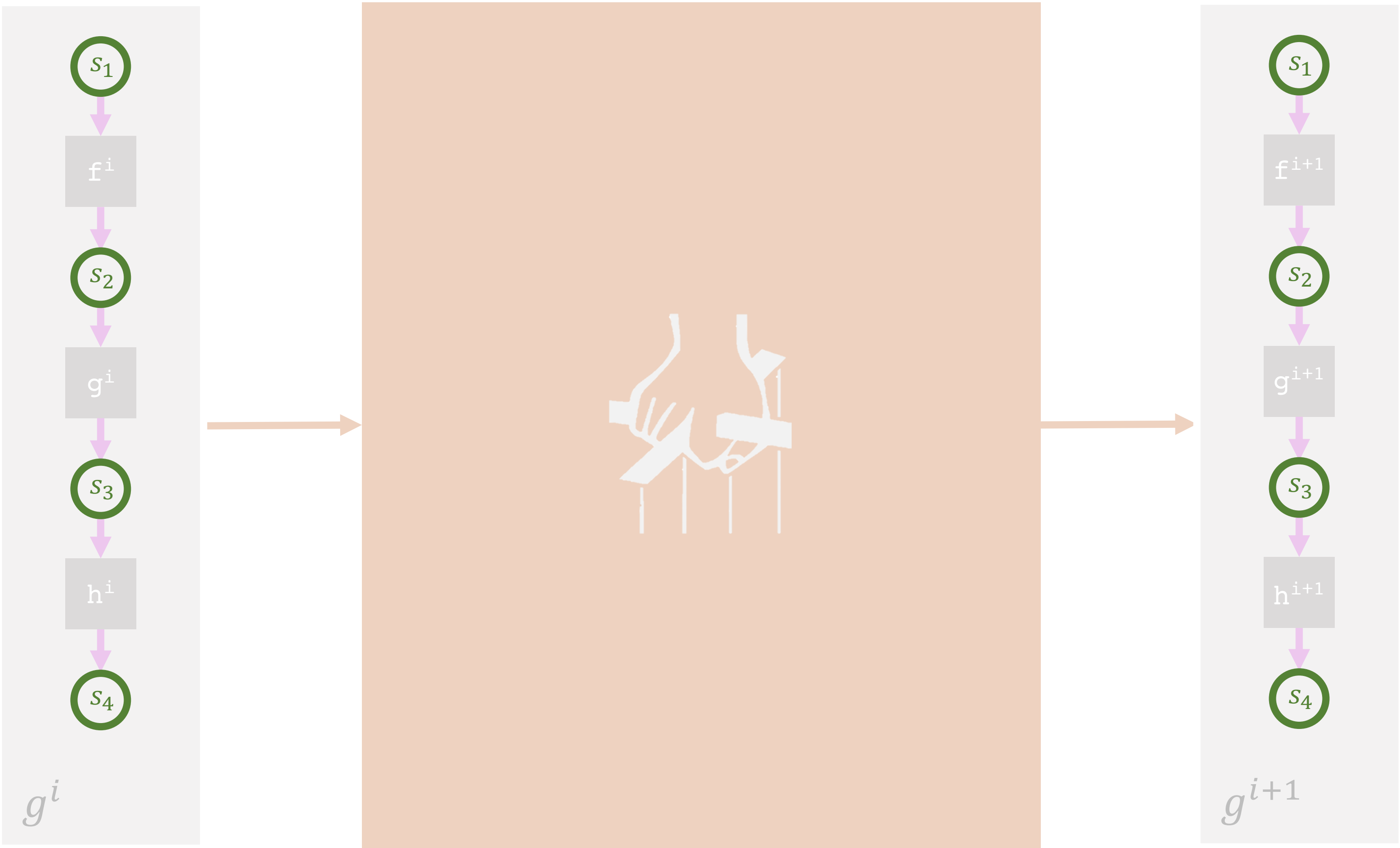
Learning algorithms neatly fall into the dynamic systems framework.

Learning Algorithms are Dynamic Systems



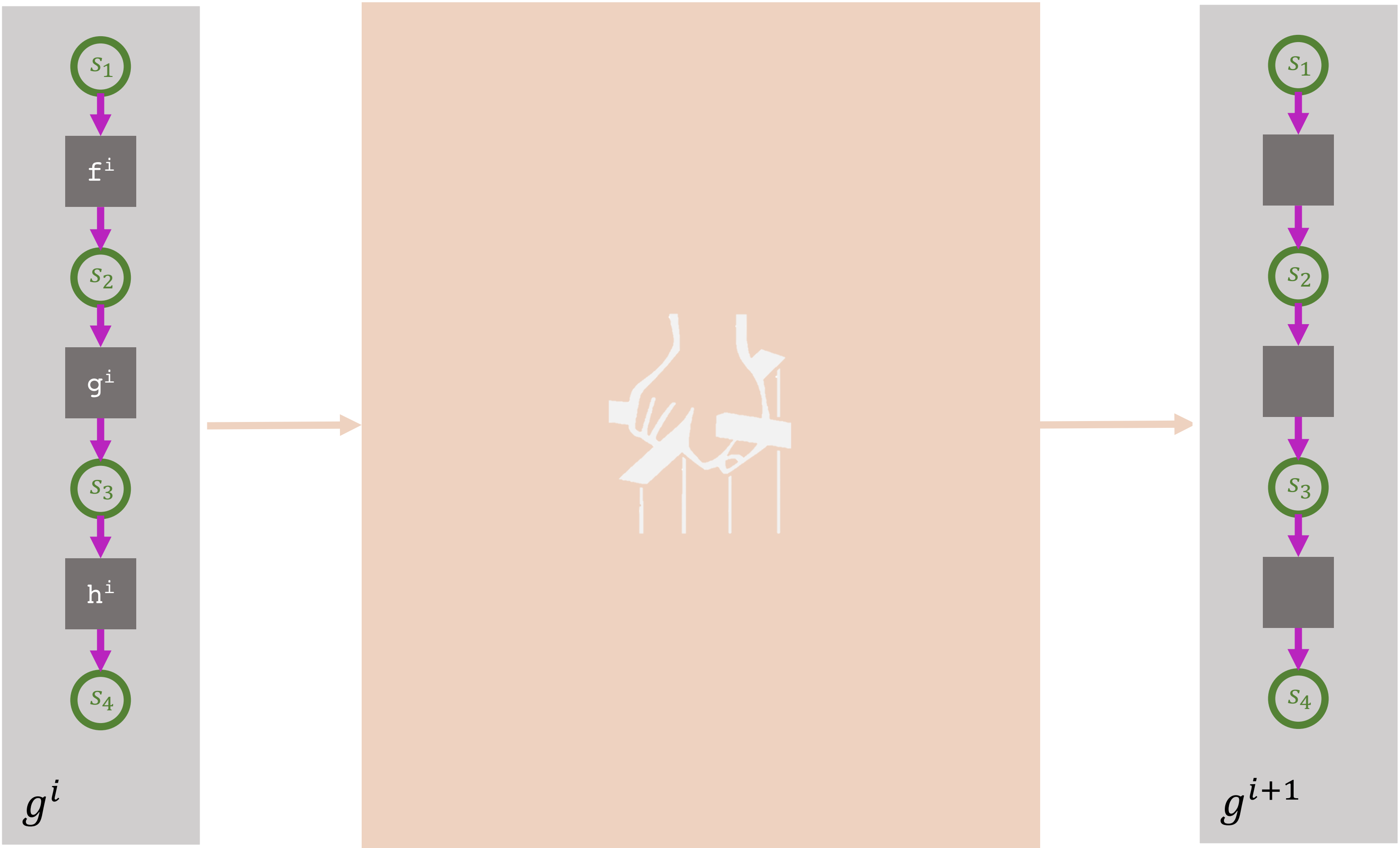
The causal mechanisms correspond to learnable functions.

Learning Algorithms are Dynamic Systems



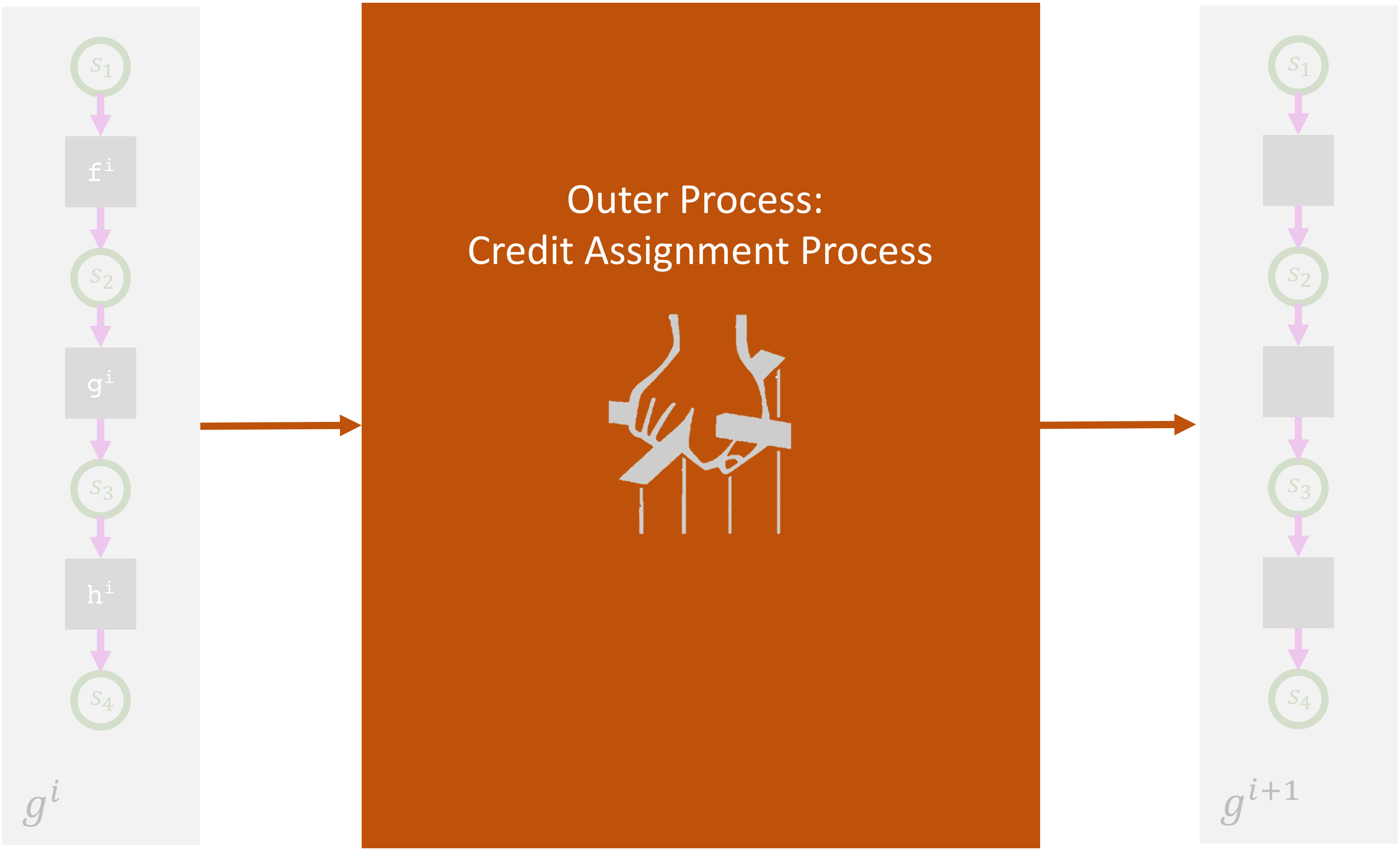
The causal nodes correspond to their inputs and outputs.

Learning Algorithms are Dynamic Systems



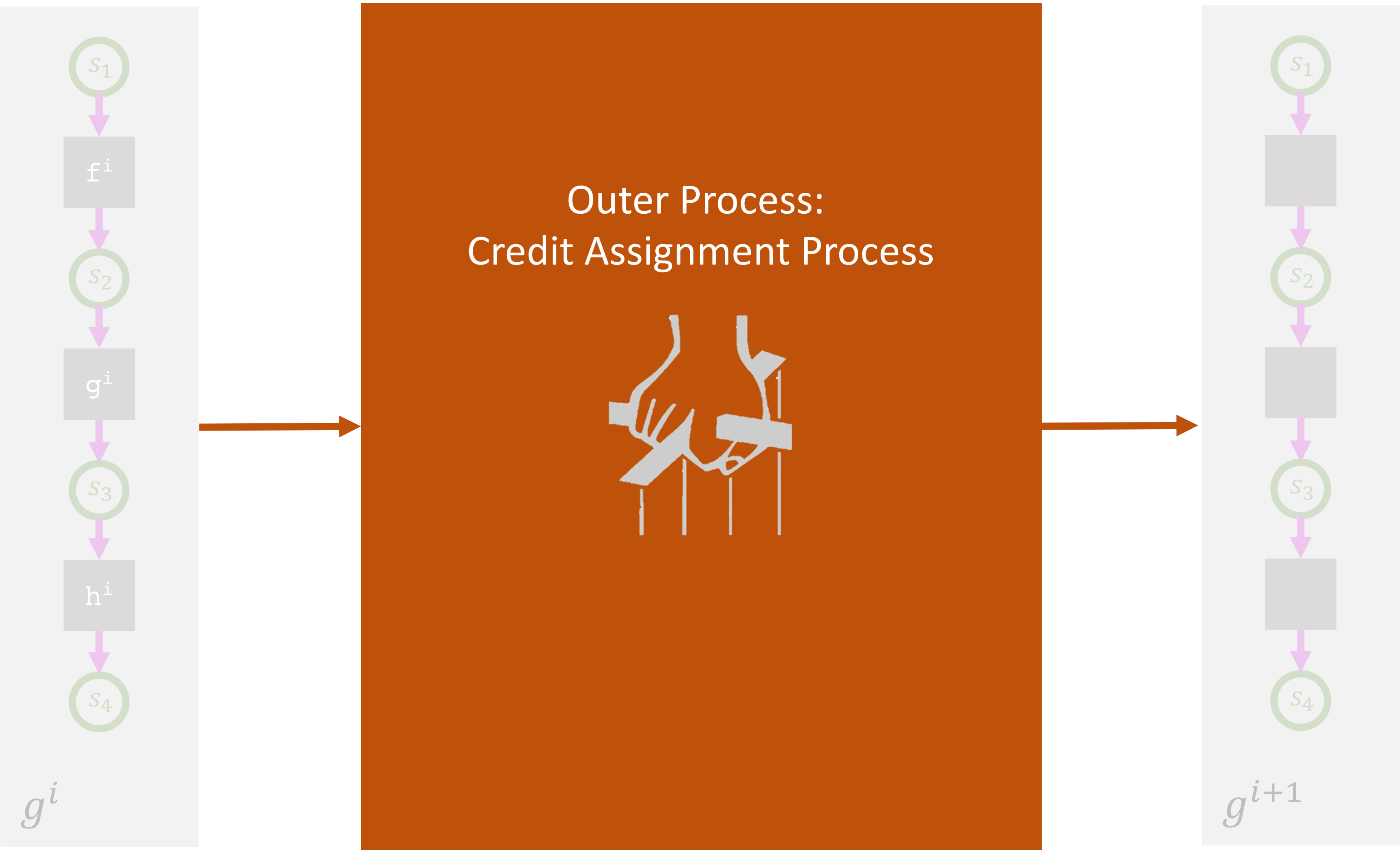
Each static graph represents a forward pass of the learning system.

Learning Algorithms are Dynamic Systems



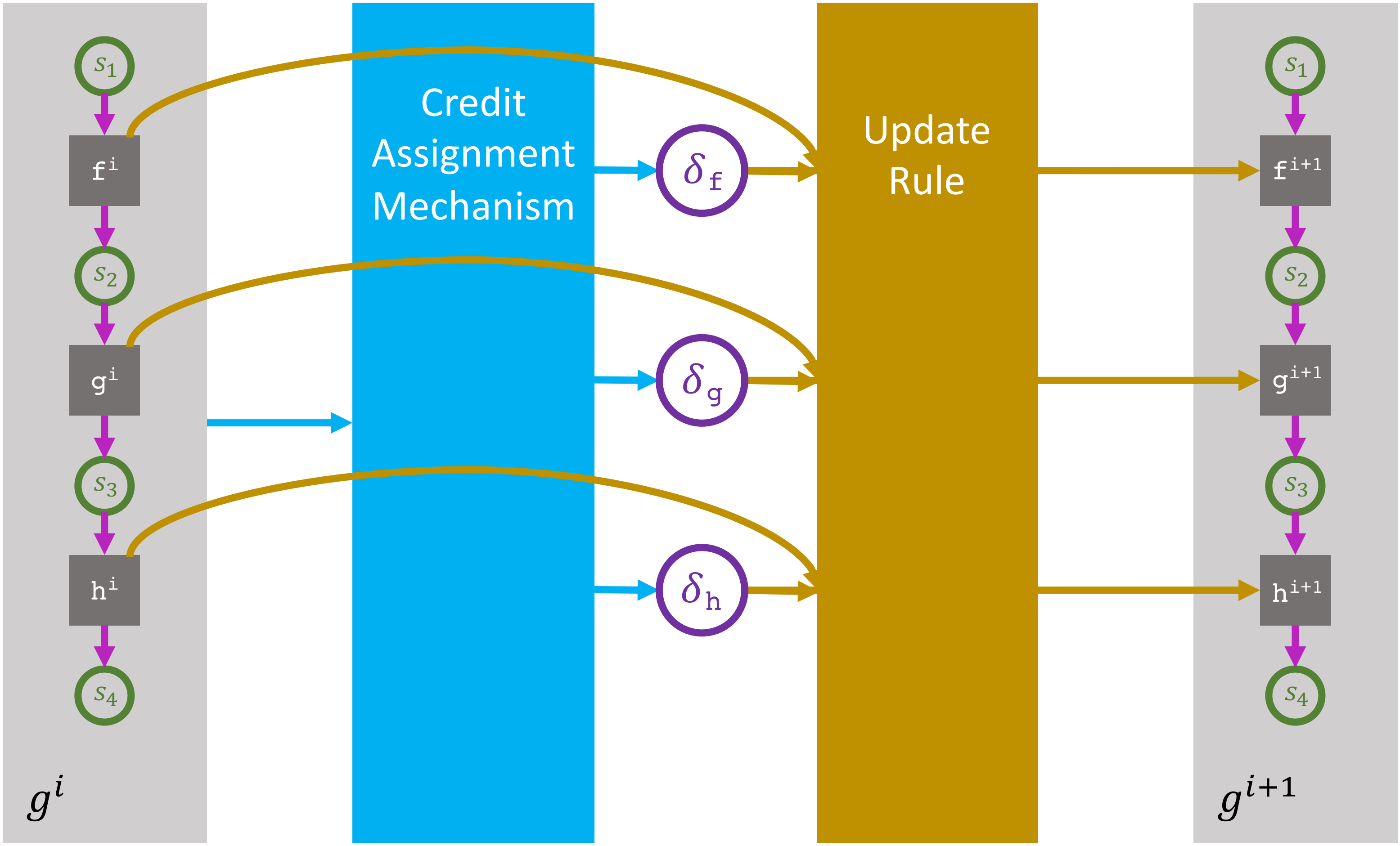
Instead of the outer process being a human who intervenes on the learnable functions,

Learning Algorithms are Dynamic Systems



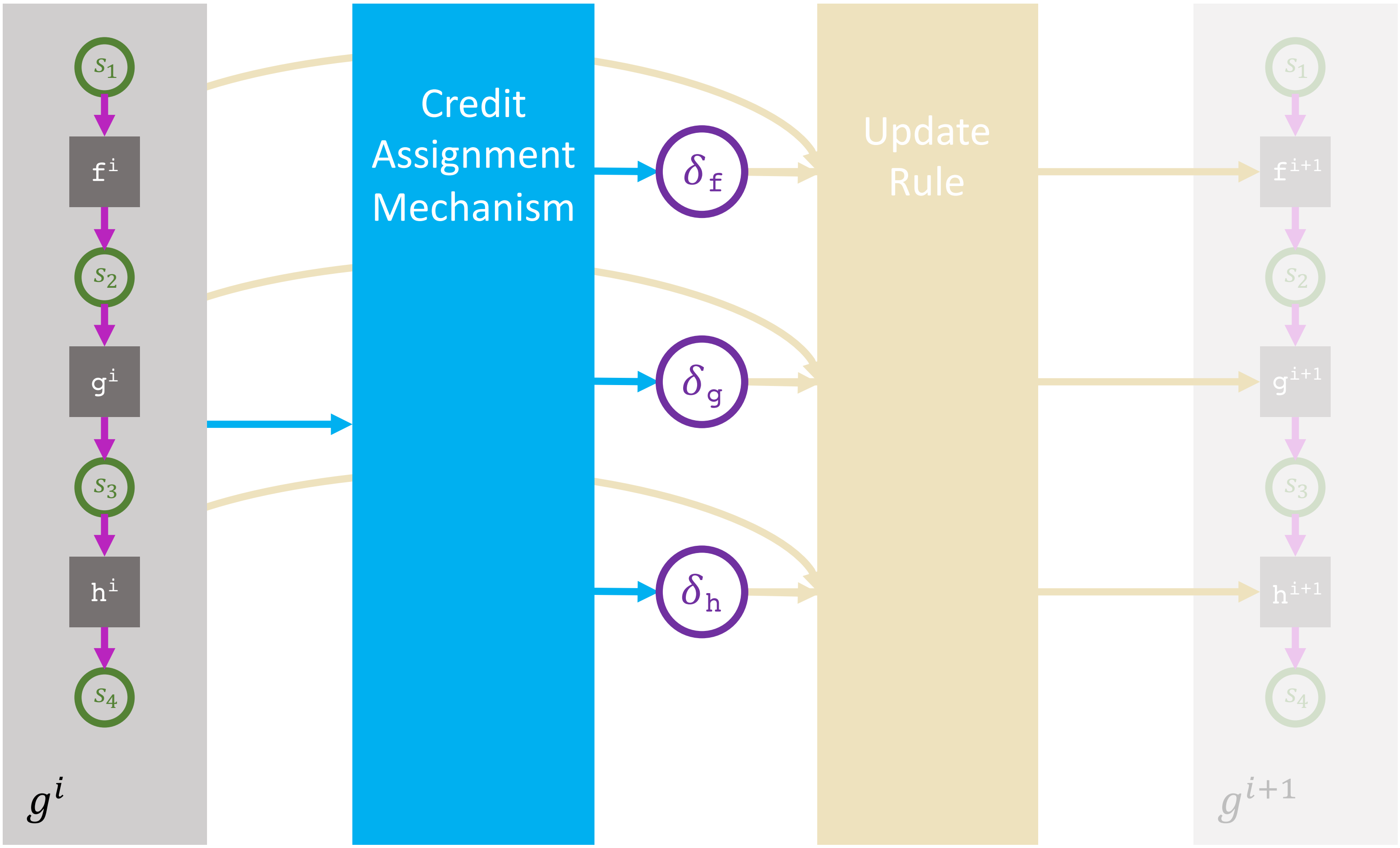
the outer process is the credit assignment process of the learning algorithm,

Learning Algorithms are Dynamic Systems



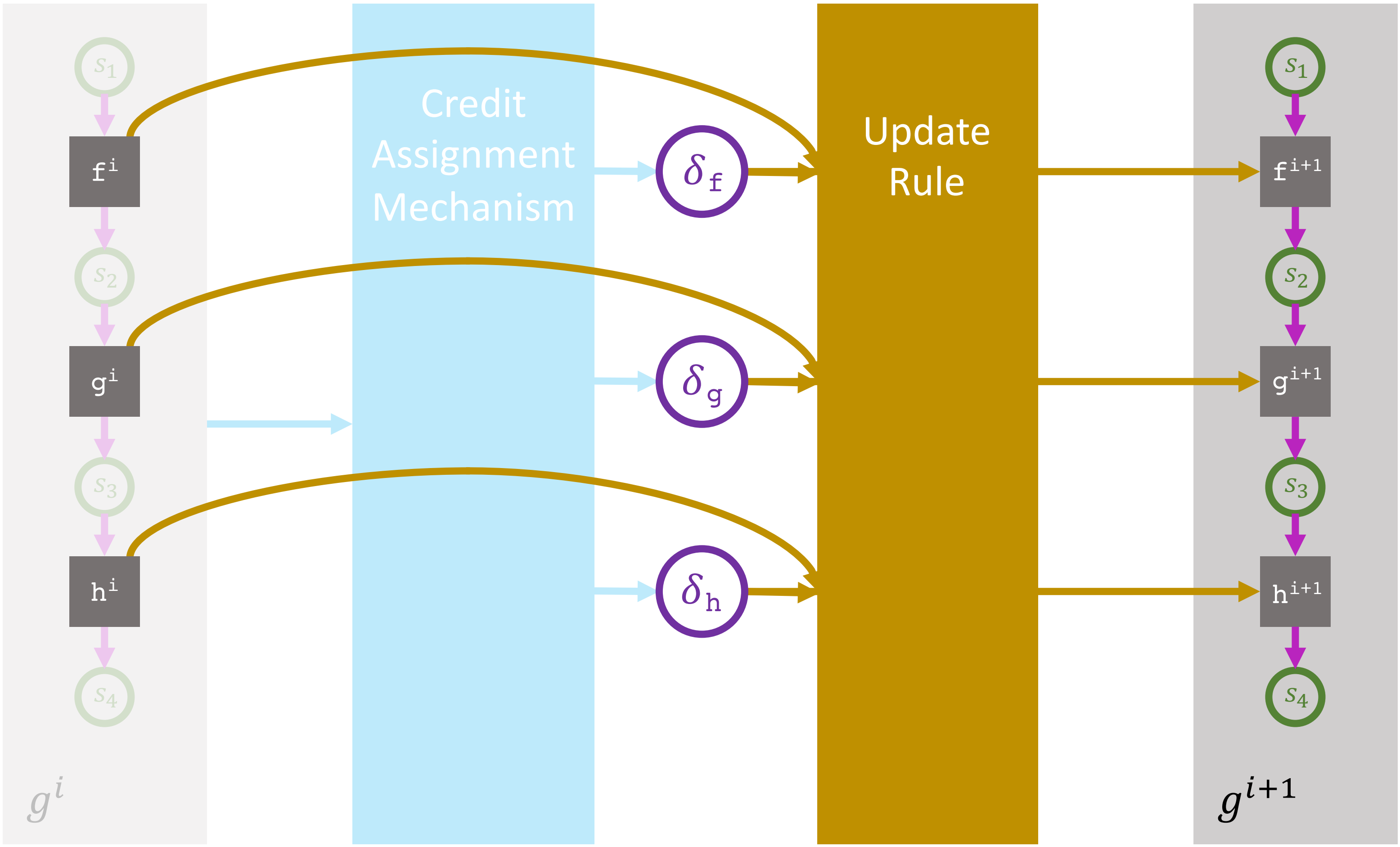
which we can split into two parts:

Learning Algorithms are Dynamic Systems



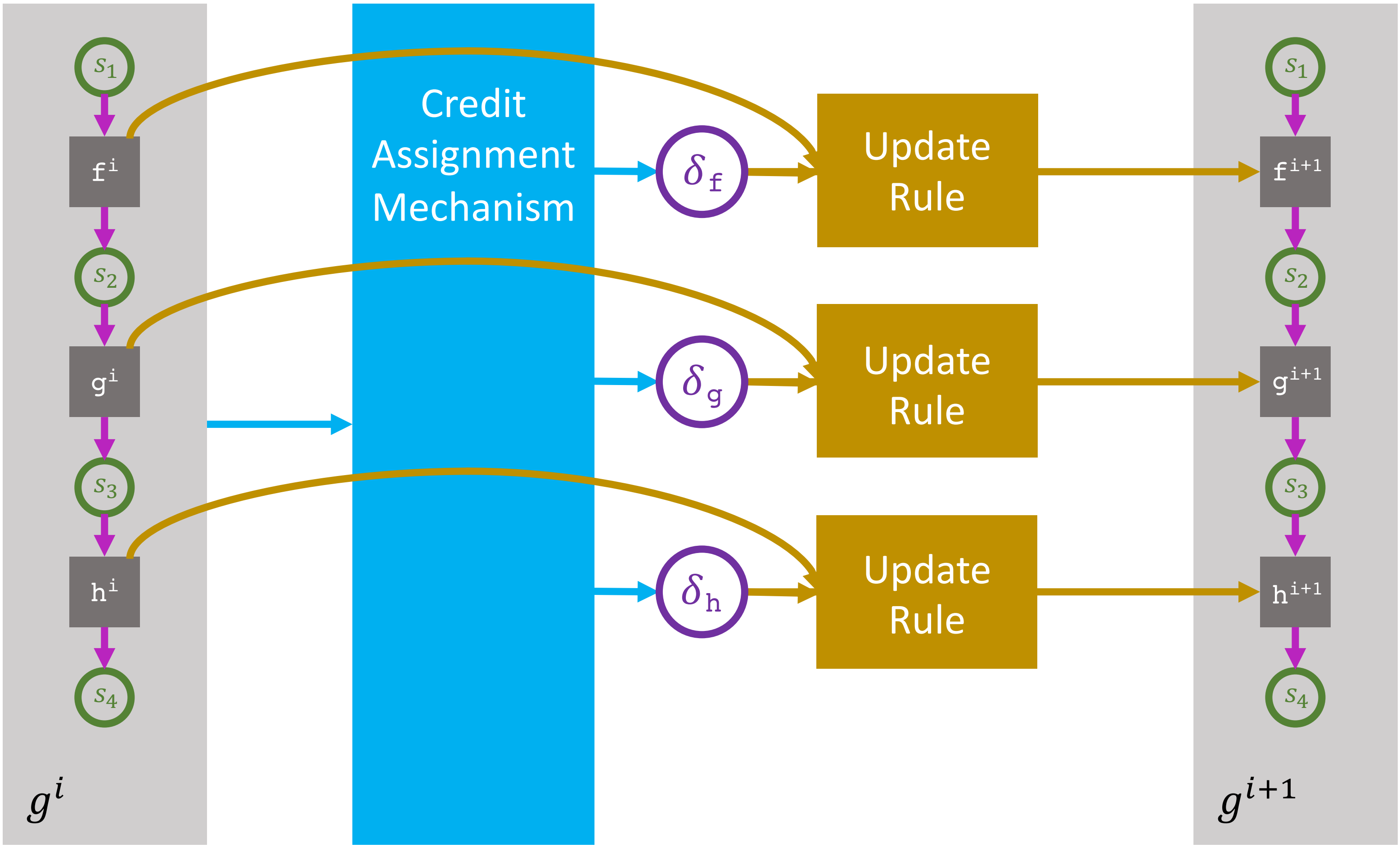
the credit assignment mechanism, which takes the previous graph and produces gradients for the learnable functions,

Learning Algorithms are Dynamic Systems



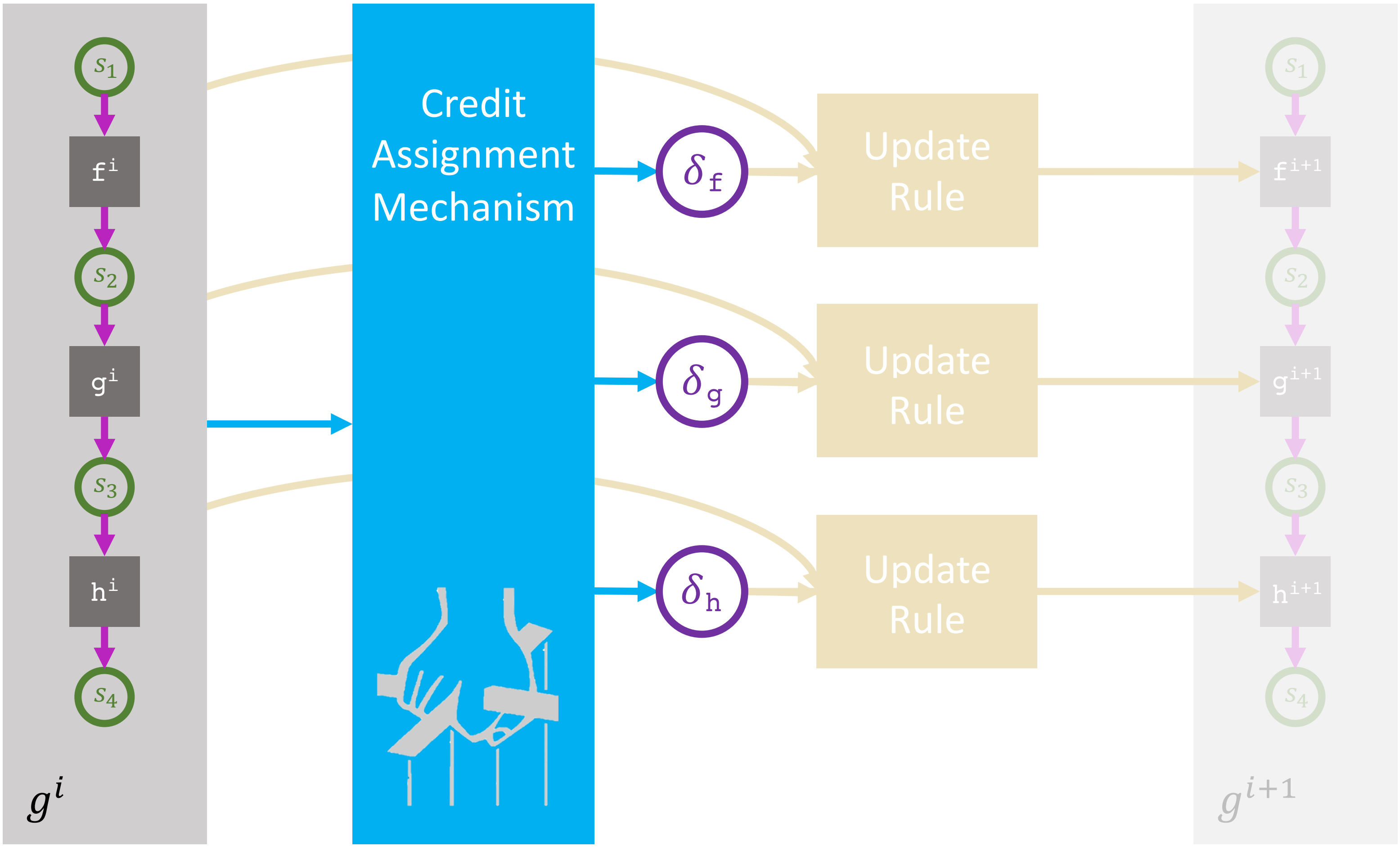
and the update rule, which modifies the learnable functions given the gradients.

Learning Algorithms are Dynamic Systems



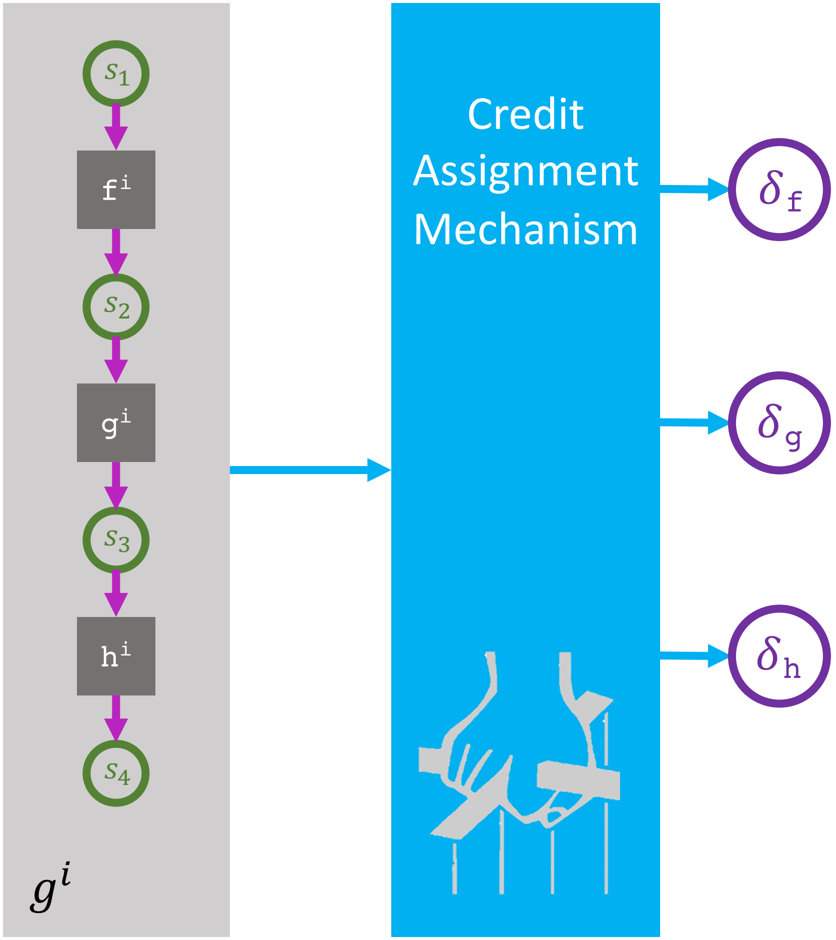
Typically the update rule is assumed to be a fixed operation, like gradient descent, which factorizes over the learnable functions.

Modularity Constraint on Credit Assignment



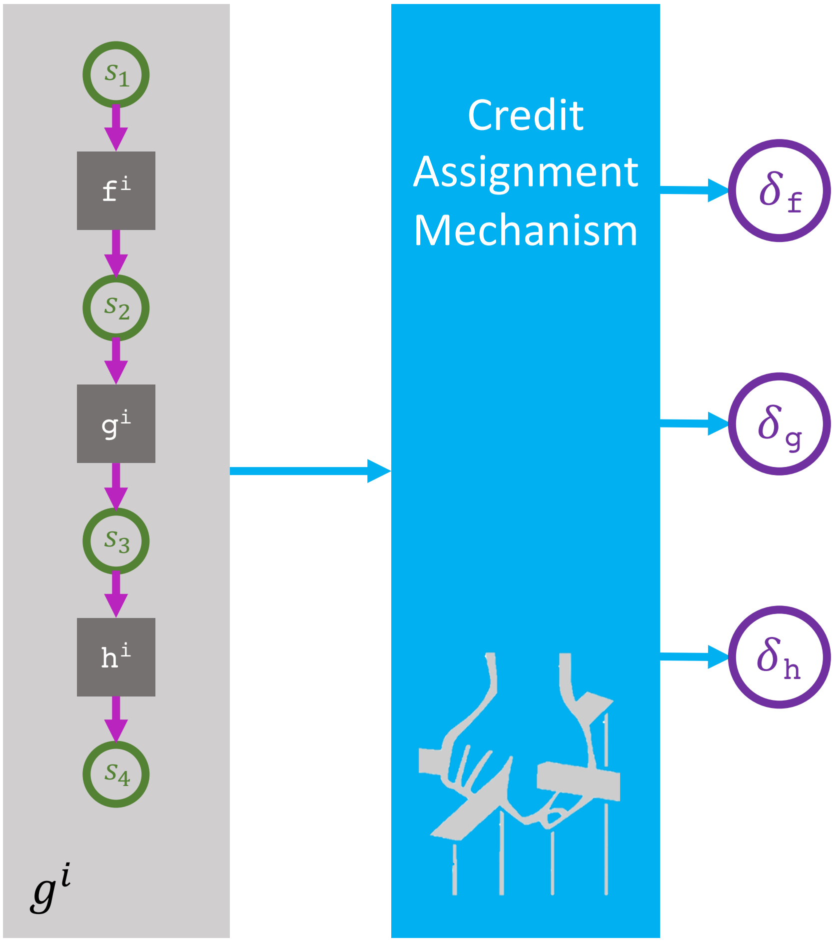
The question is: what constraint should the credit assignment mechanism satisfy

Modularity Constraint on Credit Assignment



to modify the learnable functions independently?

Algorithmically Independent Gradients

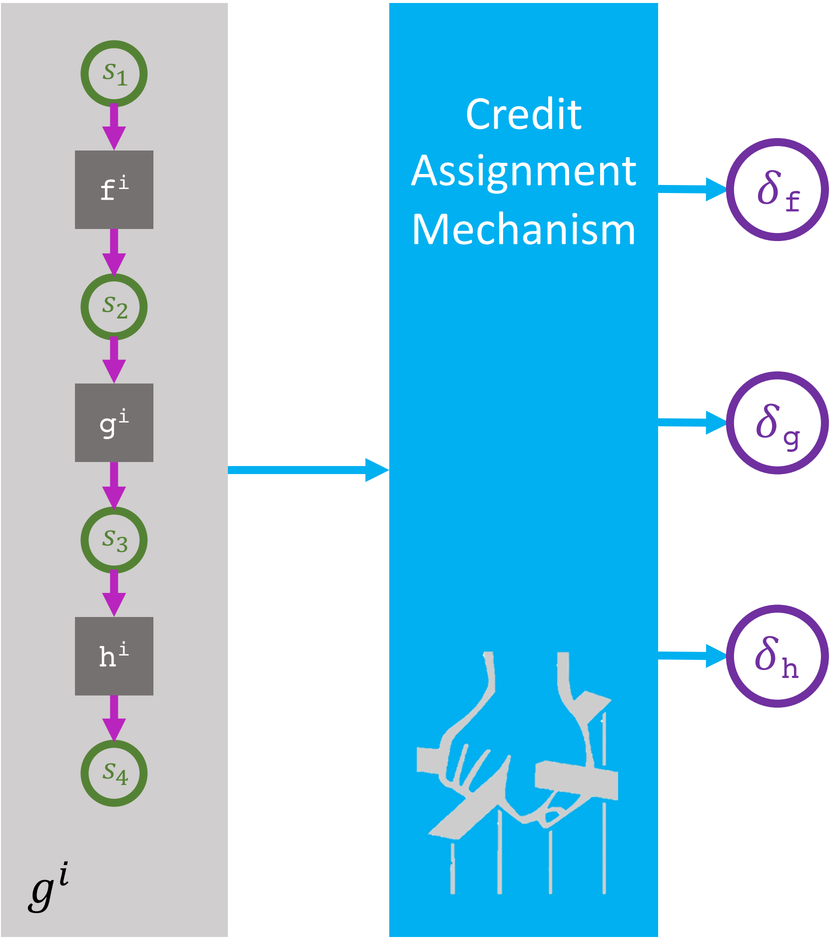


Modularity Constraint:

$$I(\delta_f, \delta_g, \delta_h \mid g^i) \stackrel{+}{=} 0$$

We show in our paper that it must produce gradients that are algorithmically independent

Algorithmically Independent Gradients

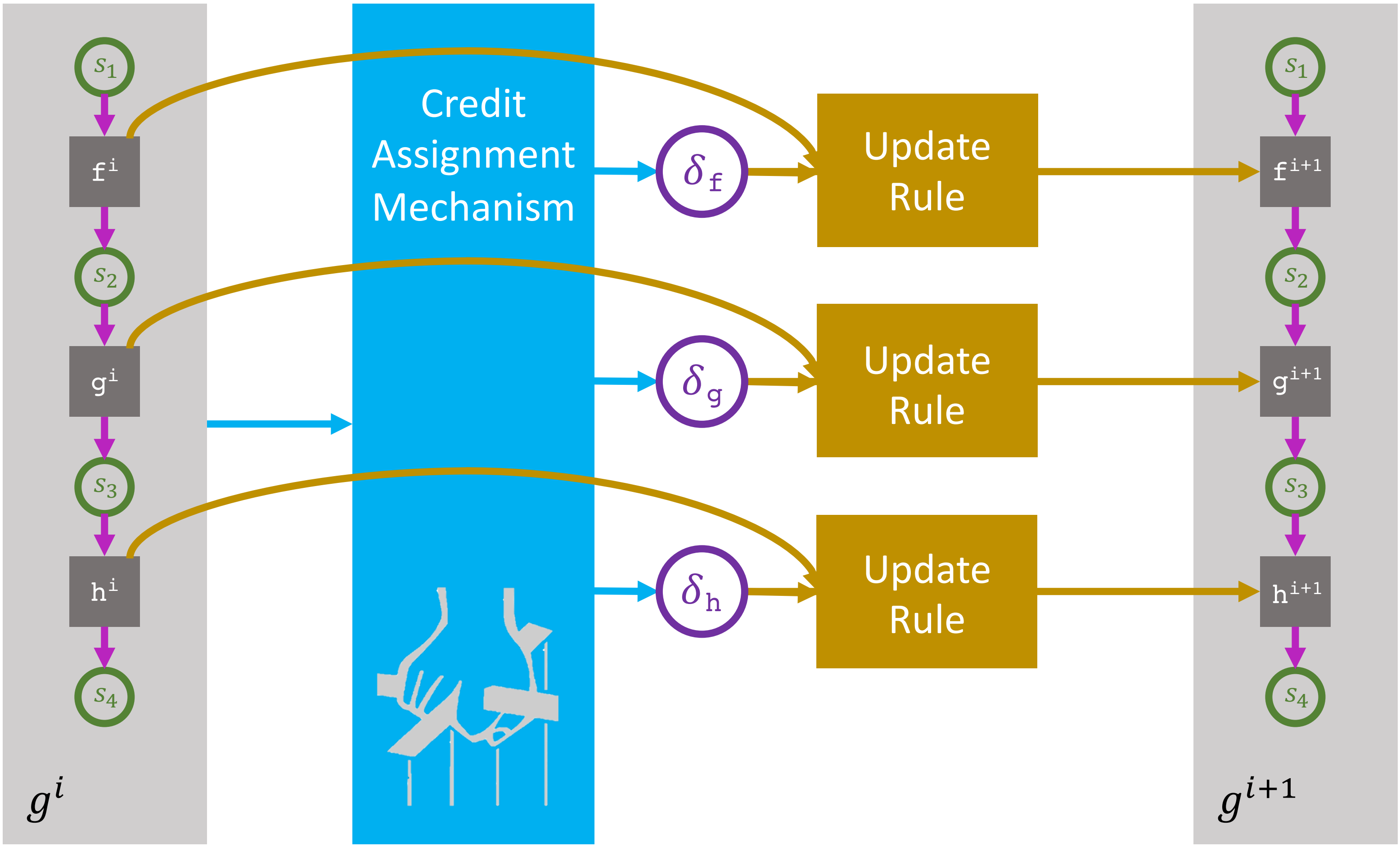


Modularity Constraint:

$$I(\delta_f, \delta_g, \delta_h \mid g^i) \stackrel{+}{=} 0$$

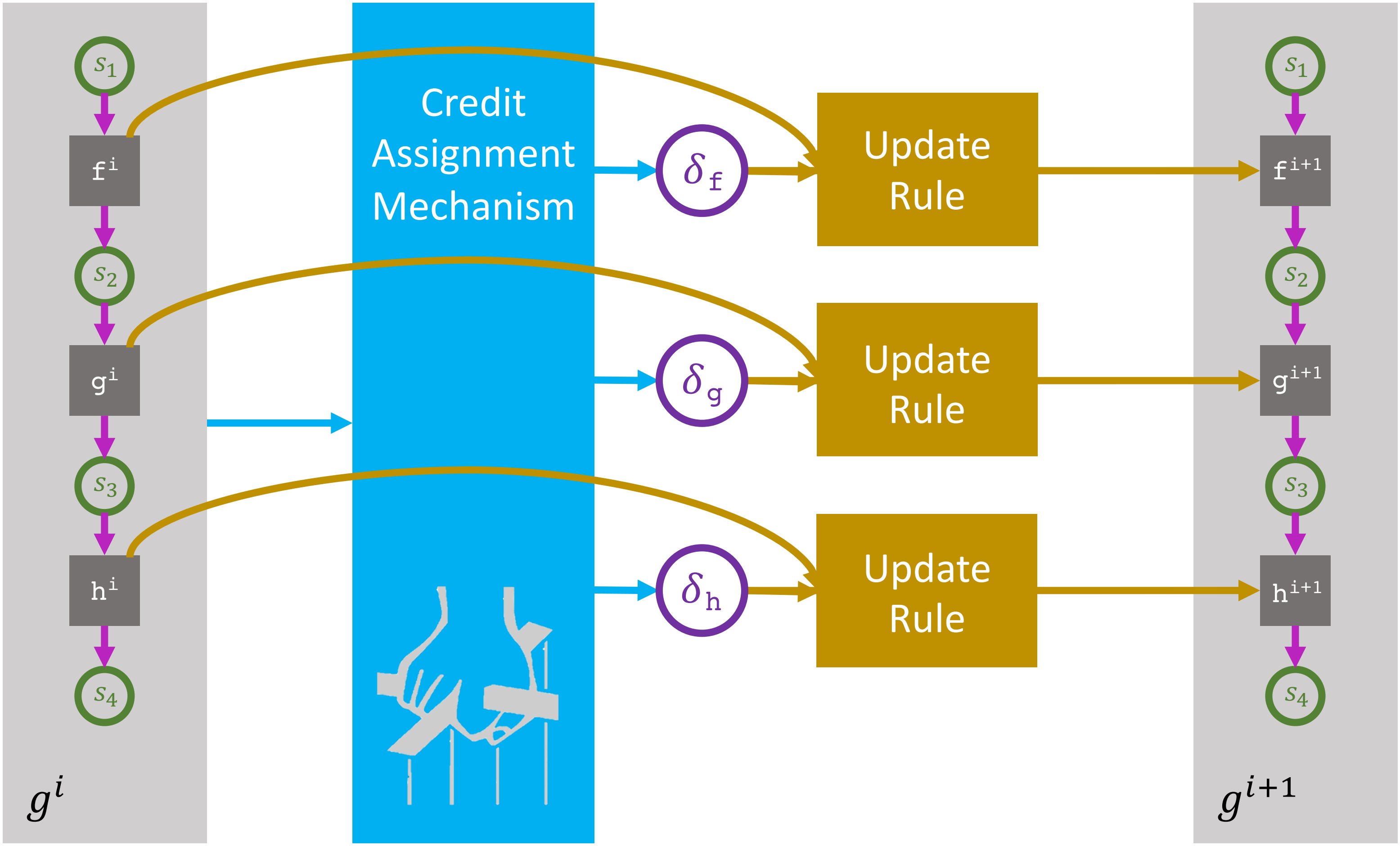
conditioned on the graph in the previous iteration.

Algorithmically Independent Gradients



Then if the gradients are conditionally independent,

Algorithmically Independent Gradients



the learnable functions in the next iteration will also be conditionally independent.

Main Result

Theorem (modularity, informal):
A learning algorithm is modular if its learnable mechanisms do not share weights (i.e. the network is factorized) and if its credit assignment mechanism produces independent gradients.

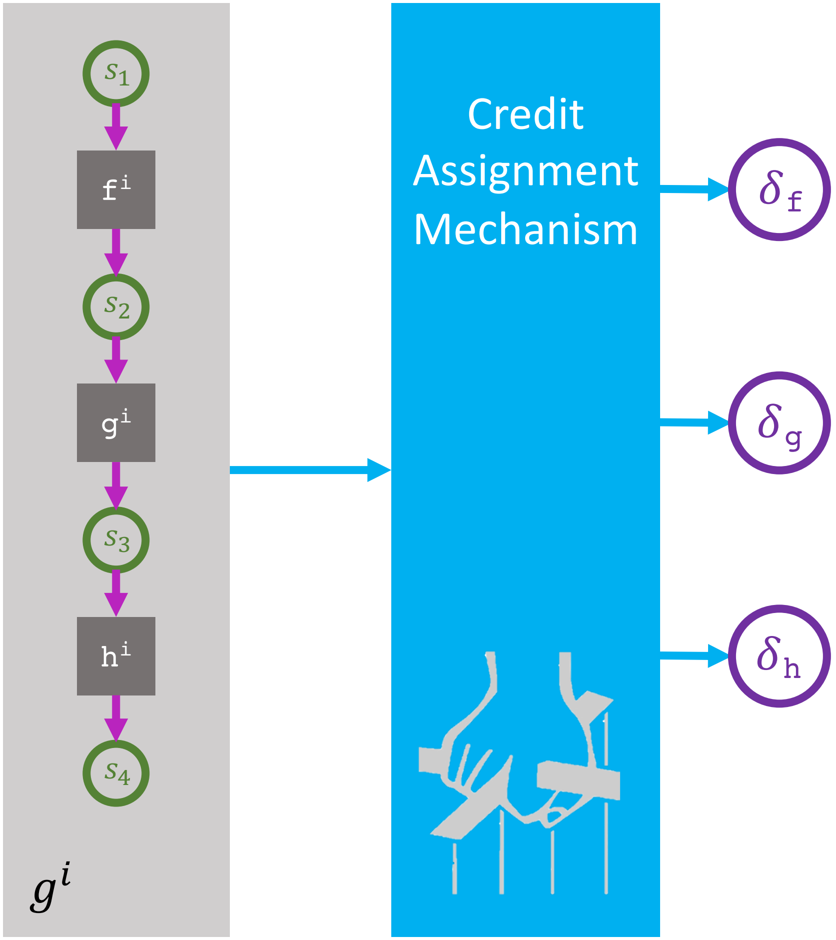
Check out the paper for our theorem that formally states that a learning algorithm is modular

Main Result 1

Theorem (modularity, informal):
A learning algorithm is modular if its learnable mechanisms do not share weights (i.e. the network is factorized) and if its credit assignment mechanism produces independent gradients.

if its learnable mechanisms do not share weights and if its credit assignment mechanism produces independent gradients.

A Criterion for Modularity

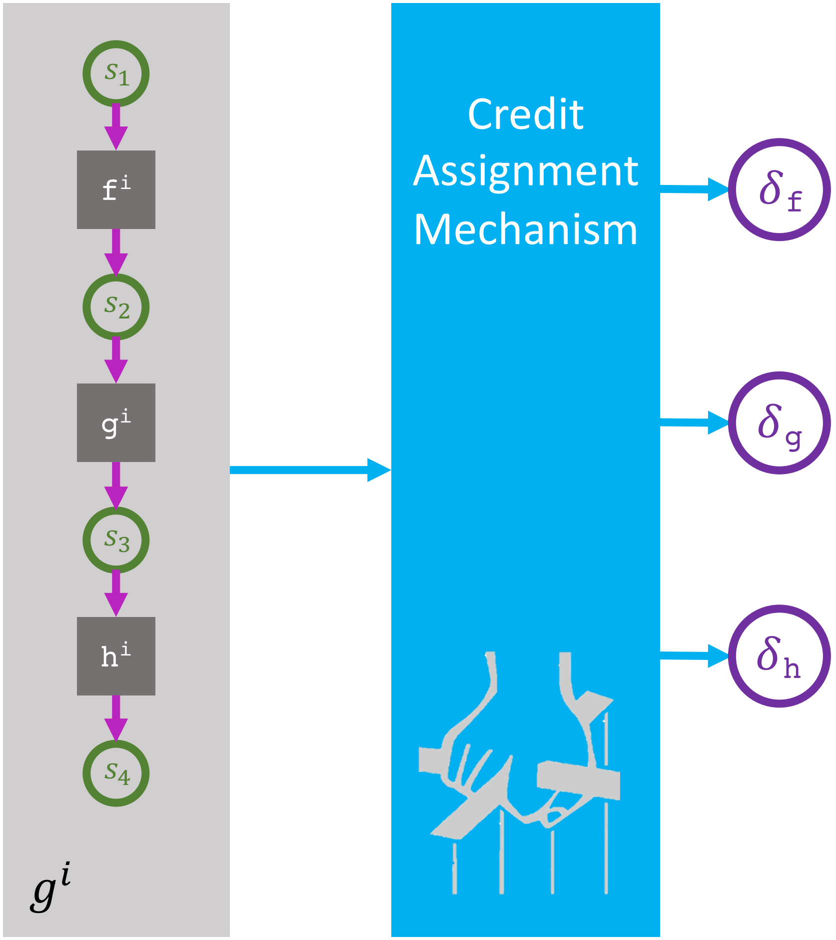


Modularity Constraint:

$$I\left(\delta_\mathtt{f}, \delta_\mathtt{g}, \delta_\mathtt{h} \mid g^i\right) \stackrel{+}{=} 0$$

But algorithmic mutual information is not computable.

A Criterion for Modularity

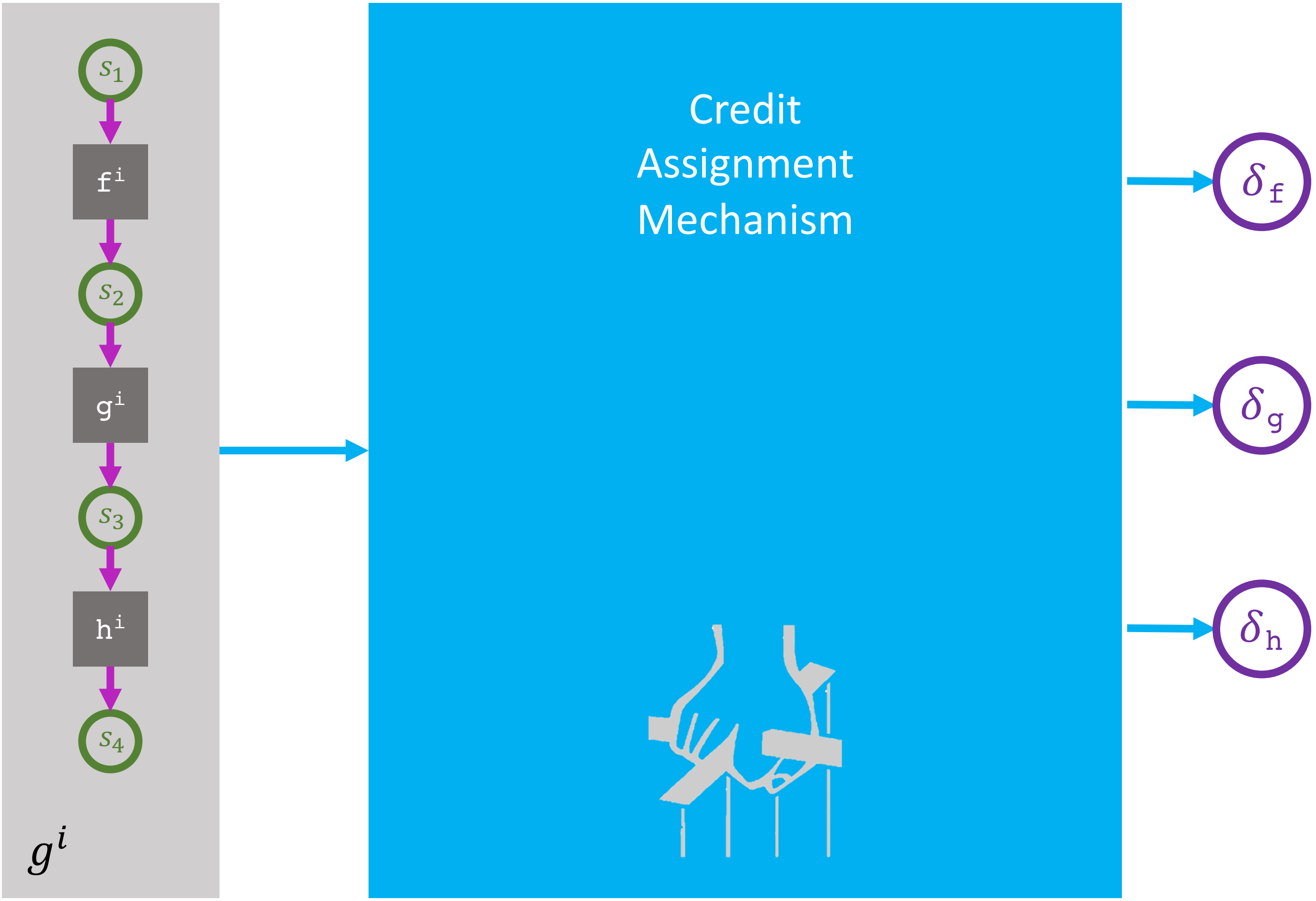


Modularity Constraint:

$$I\left(\delta_{\mathtt{f}}, \delta_{\mathtt{g}}, \delta_{\mathtt{h}} \mid g^i\right) \stackrel{+}{=} 0$$

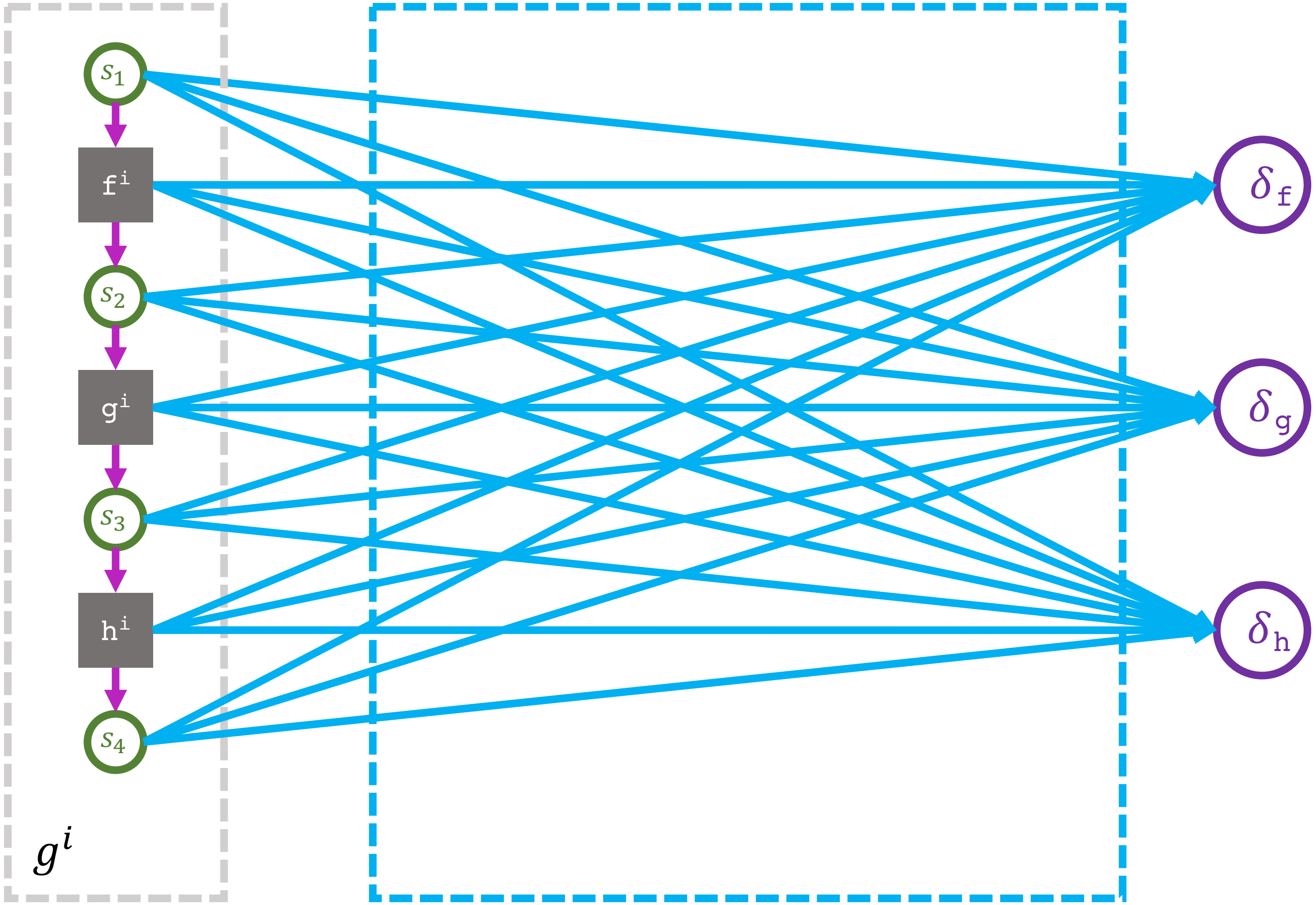
We need a practical criterion to design and evaluate credit assignment mechanisms that produce independent gradients.

A Criterion for Modularity



It turns out we can get this criterion by flattening the graph of the learner, and the graph of credit assignment

A Criterion for Modularity

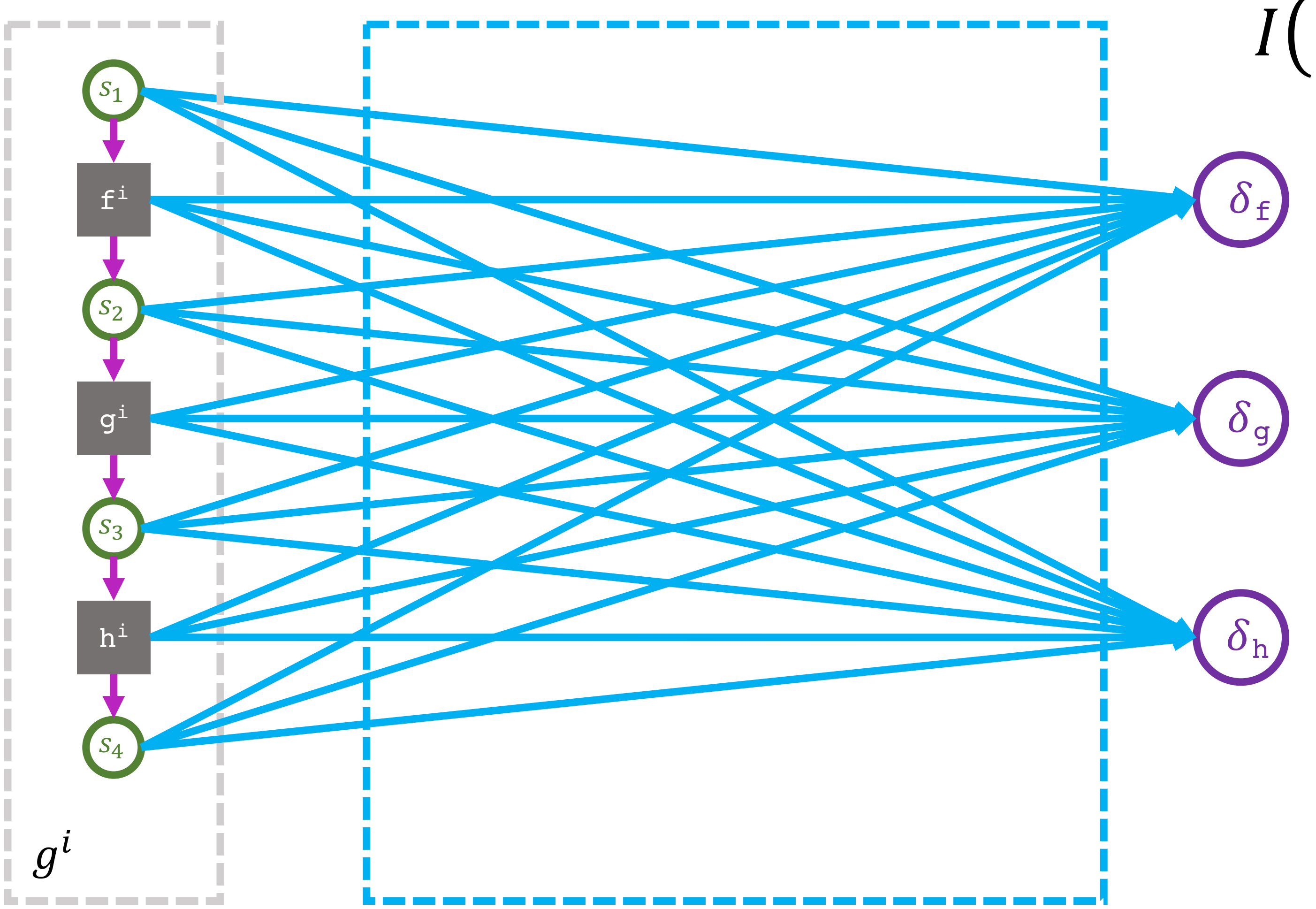


into one big graph, where both the mechanisms and data are treated as nodes.

A Criterion for Modularity

Modularity Criterion:
D-separation implies conditional independence

$$I\left(\delta_{\mathtt{f}}, \delta_{\mathtt{g}}, \delta_{\mathtt{h}} \mid g^i\right) \stackrel{+}{=} 0$$

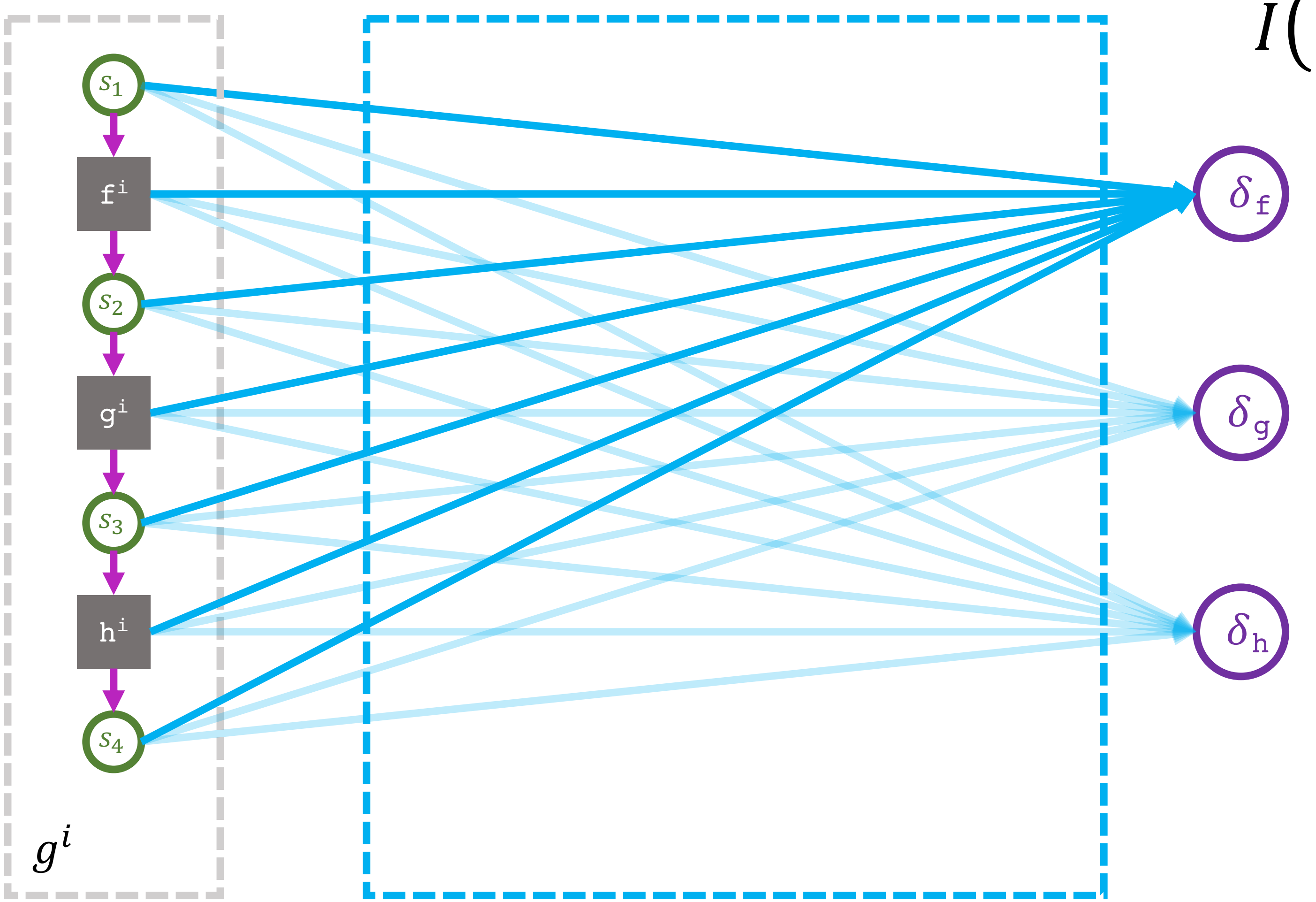


Then we can simply test if the gradients are conditionally independent by inspecting the graph for d-separation.

A Criterion for Modularity

Gradients are *d*-separated ✓

$$I\left(\delta_{\mathtt{f}}, \delta_{\mathtt{g}}, \delta_{\mathtt{h}} \mid g^i\right) \overset{+}{=} 0$$

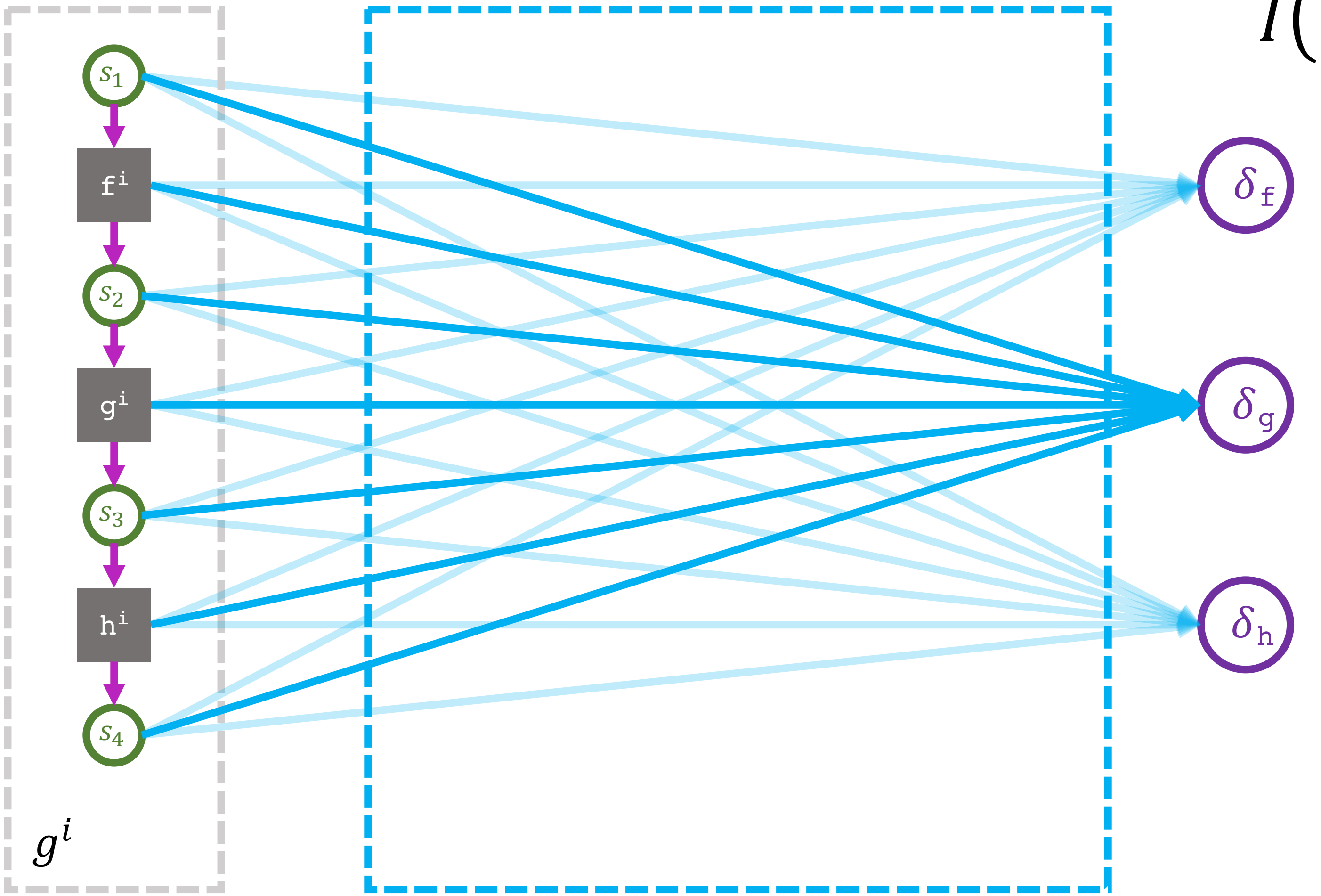


They are d-separated if each gradient is produced by a different function.

A Criterion for Modularity

Gradients are *d*-separated ✓

$$I\left(\delta_{\mathtt{f}}, \delta_{\mathtt{g}}, \delta_{\mathtt{h}} \mid g^i\right) \stackrel{+}{=} 0$$

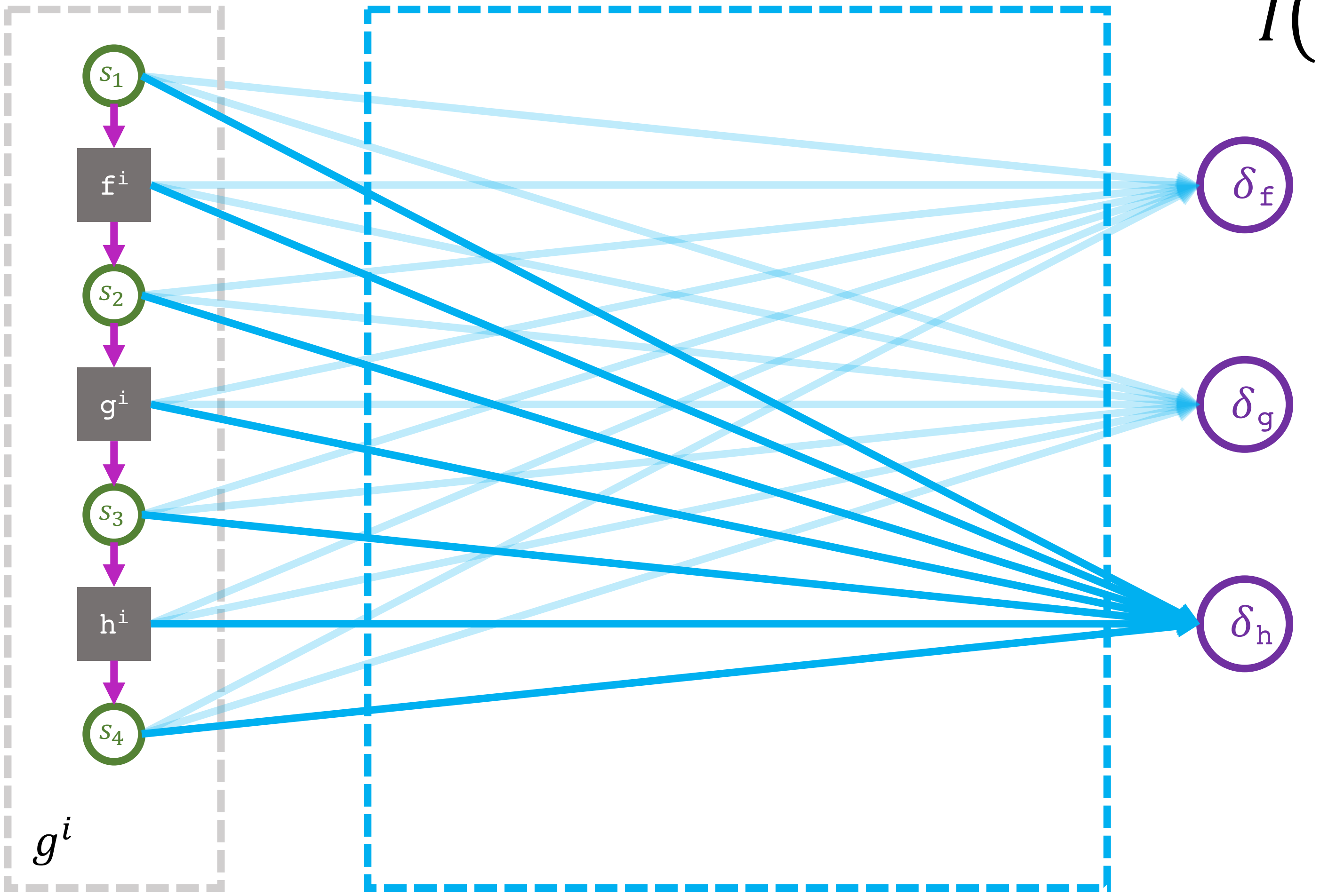


They are d-separated if each gradient is produced by a different function.

A Criterion for Modularity

Gradients are *d*-separated ✓

$$I(\delta_{\mathtt{f}}, \delta_{\mathtt{g}}, \delta_{\mathtt{h}} \mid g^i) \overset{+}{=} 0$$

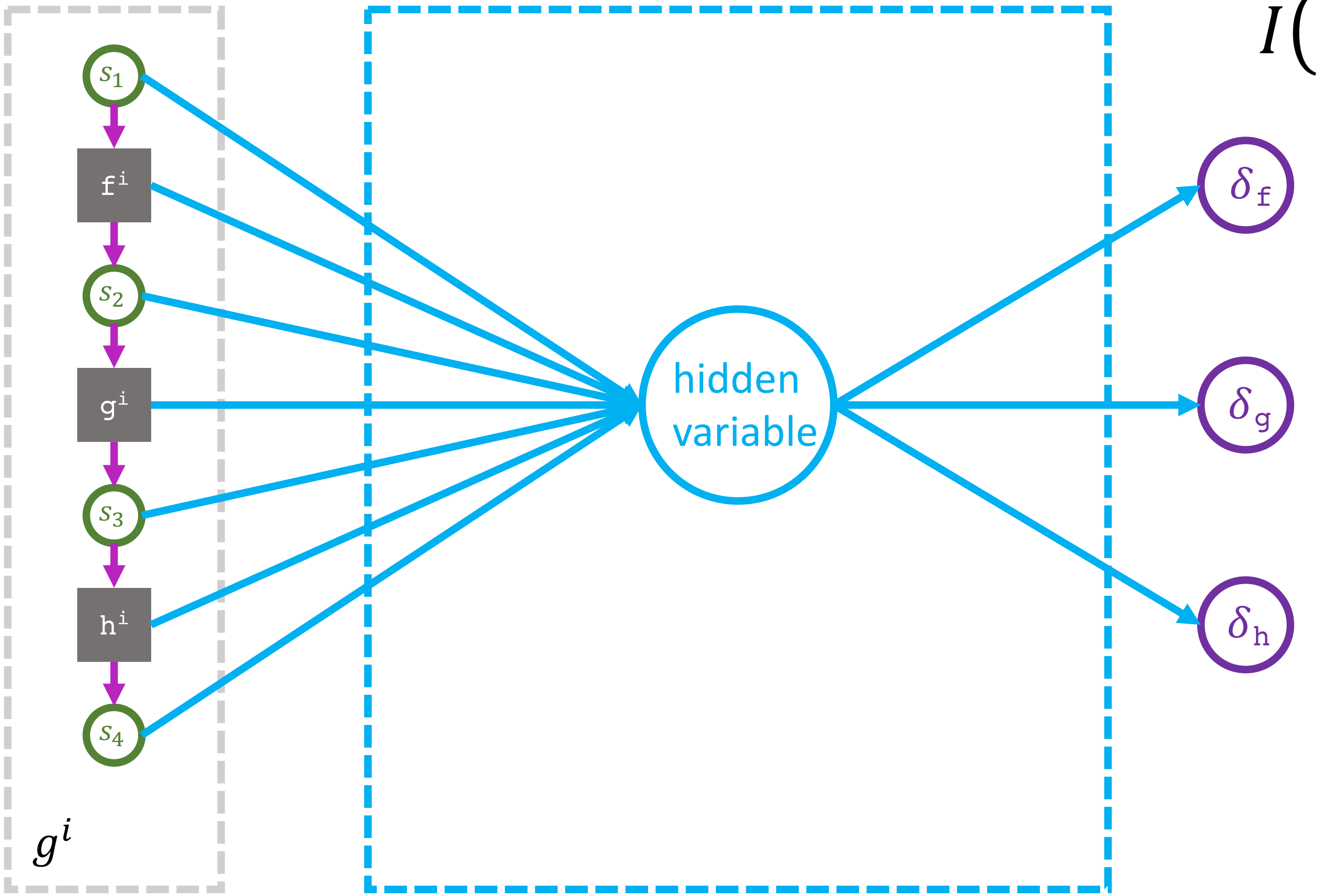


They are d-separated if each gradient is produced by a different function.

A Criterion for Modularity

Gradients are not *d*-separated ✗

$$I\left(\delta_{\mathtt{f}}, \delta_{\mathtt{g}}, \delta_{\mathtt{h}} \mid g^i\right) \overset{+}{\geq} 0$$

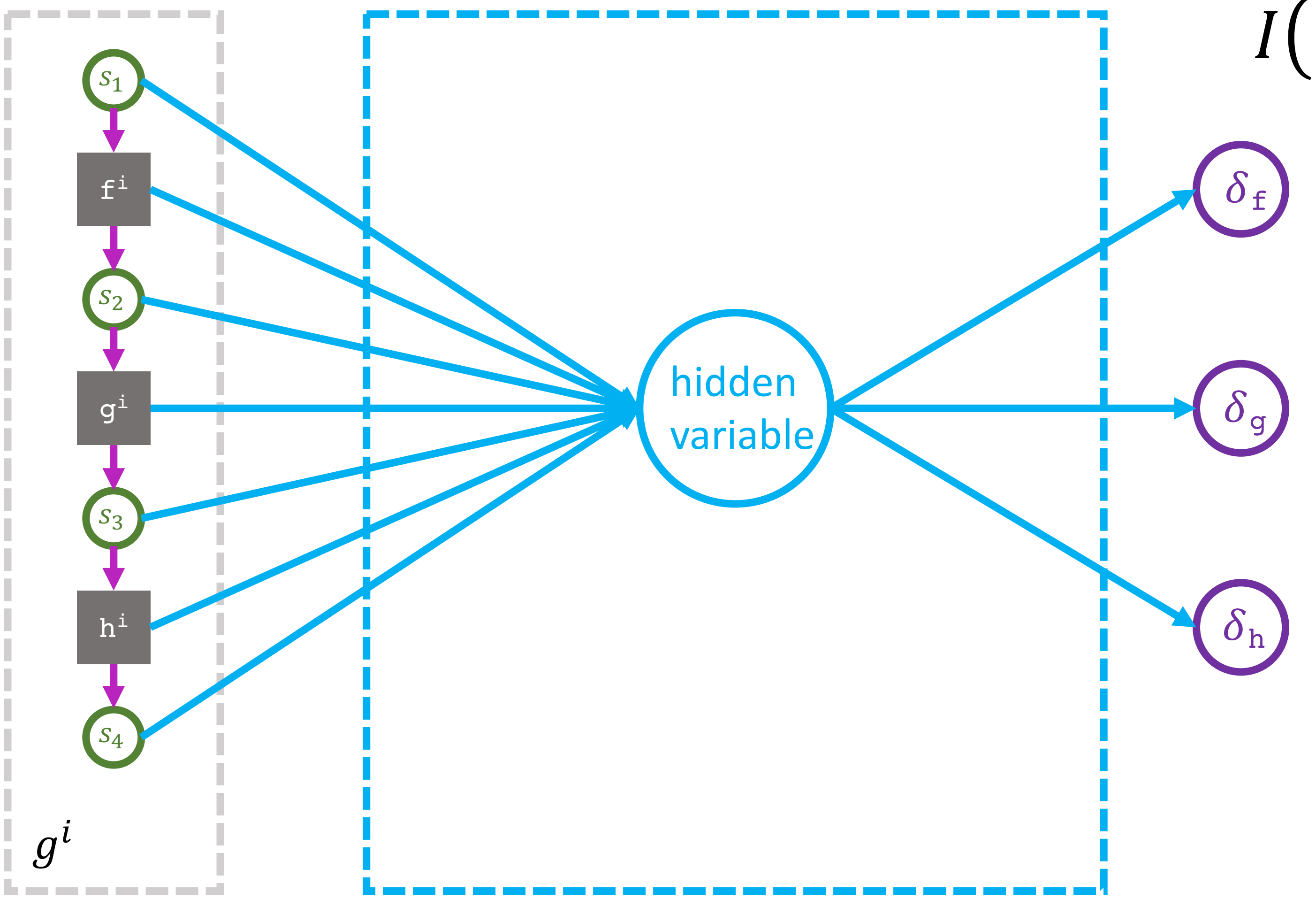


If there is a hidden variable, then the gradients are not d-separated, and thus not conditionally independent.

A Criterion for Modularity

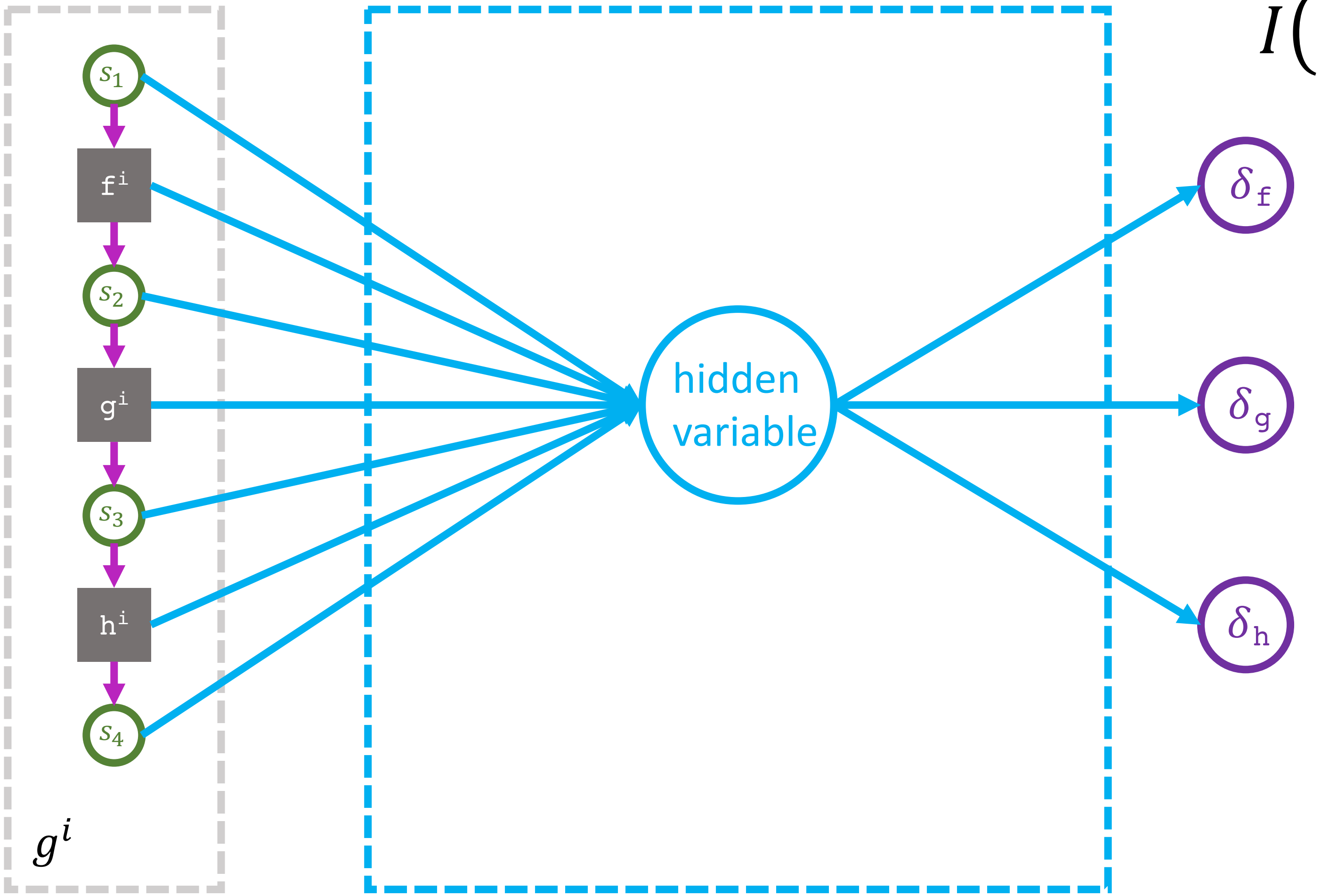
Gradients are not *d*-separated ✗

$$I\left(\delta_{\mathtt{f}}, \delta_{\mathtt{g}}, \delta_{\mathtt{h}} \mid g^i\right) \overset{+}{\geq} 0$$



This makes intuitive sense because if we want the mechanisms to be independent

A Criterion for Modularity



Gradients are not *d*-separated ✕

$$I\left(\delta_{\mathtt{f}}, \delta_{\mathtt{g}}, \delta_{\mathtt{h}} \mid g^i\right) \overset{+}{\geq} 0$$

there can be no dependency introduced through the causal structure internal to the credit assignment mechanism.

Main Result 2

Theorem (modularity criterion, informal):
Assuming faithfulness, the credit assignment mechanism produces independent gradients if and only if the gradients are d-separated by the inputs of the credit assignment mechanism.

Check out the paper for our theorem that formally connects d-separation and independent gradients.

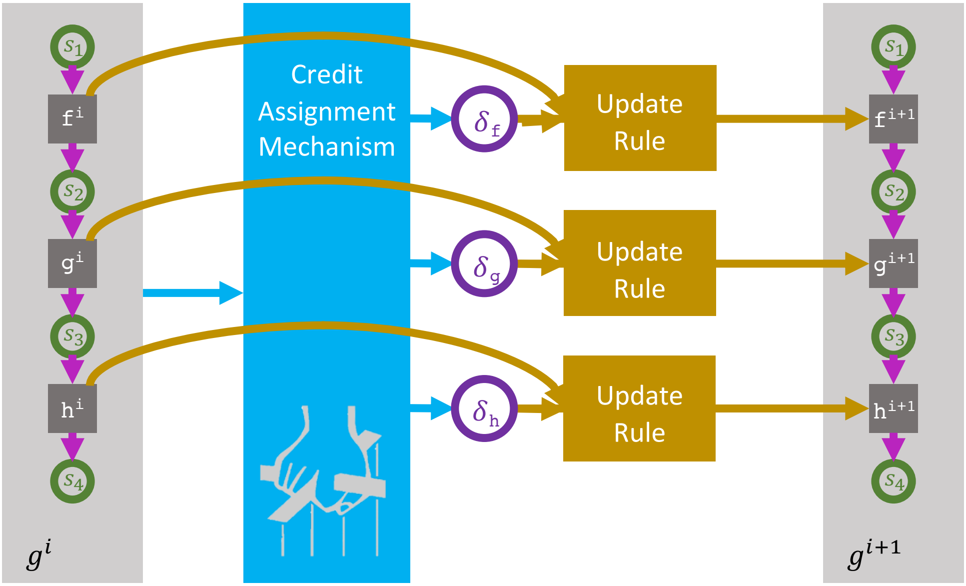
Modularity for Dynamic Systems

Independent Credit Assignment

Modularity of RL Algorithms

To recap,

Learning algorithms are dynamic systems



learning algorithms are examples of dynamic systems.

Learning algorithms are dynamic systems

Modularity requires independent feedback (e.g. gradients).

Modularity Constraint:

$$I\left(\delta_{\mathtt{f}}, \delta_{\mathtt{g}}, \delta_{\mathtt{h}} \mid g^{i}\right) \stackrel{+}{=} 0$$

If we want the learning algorithm to be modular,

Learning algorithms are dynamic systems

Modularity requires independent feedback (e.g. gradients).

Modularity Constraint:

$$I\left(\delta_{\mathtt{f}}, \delta_{\mathtt{g}}, \delta_{\mathtt{h}} \mid g^{i}\right) \overset{+}{=} 0$$

then the credit assignment mechanism needs to produce independent gradients

Learning algorithms are dynamic systems

Modularity requires independent feedback (e.g. gradients).

Modularity Constraint:

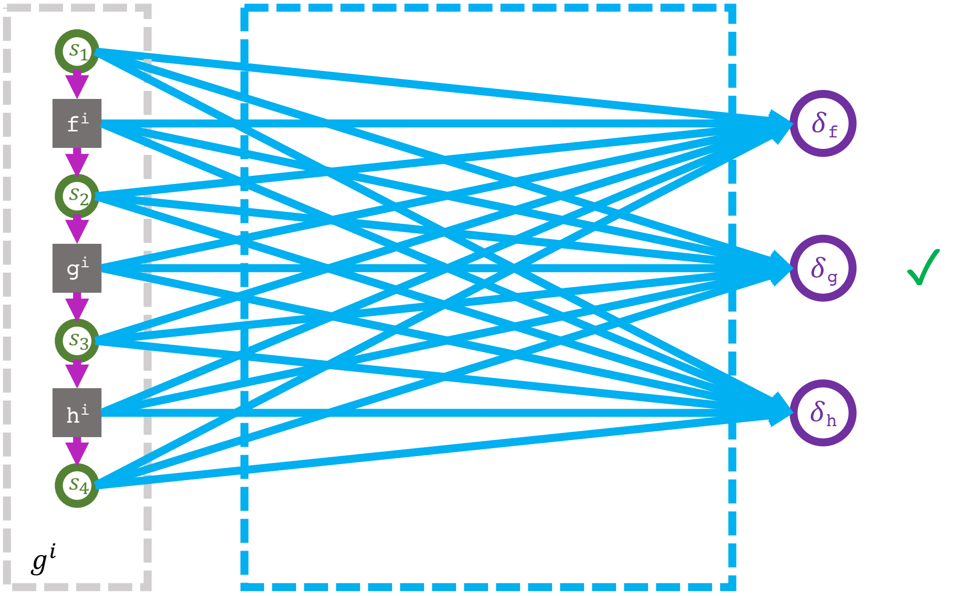
$$I\left(\delta_{\mathtt{f}}, \delta_{\mathtt{g}}, \delta_{\mathtt{h}} \mid g^{i}\right) \overset{+}{=} 0$$

to modify the learnable mechanisms independently.

Modularity for Dynamic Systems

Independent Credit Assignment

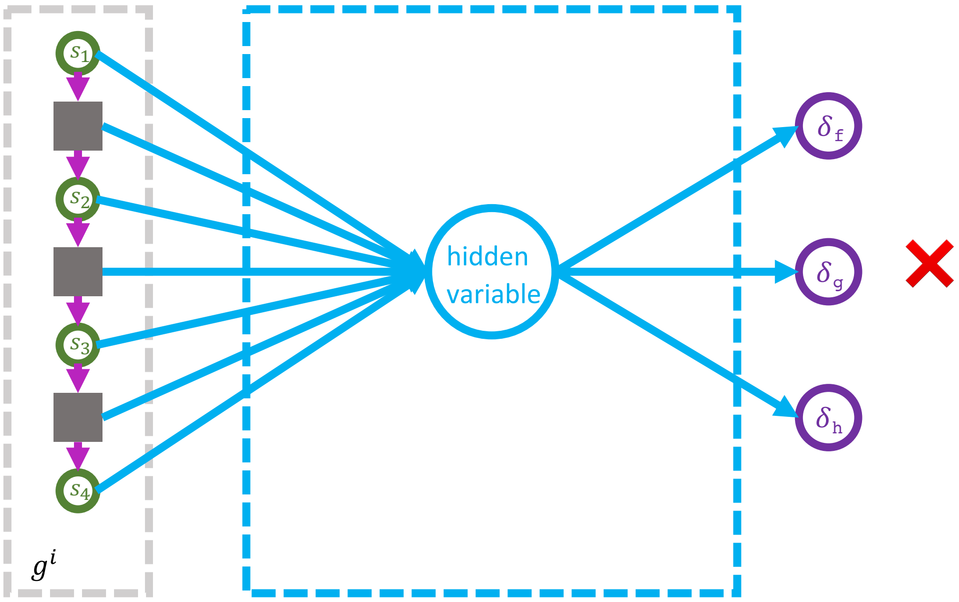
Modularity of RL Algorithms



Learning algorithms are dynamic systems.

Modularity requires independent feedback (e.g. gradients).

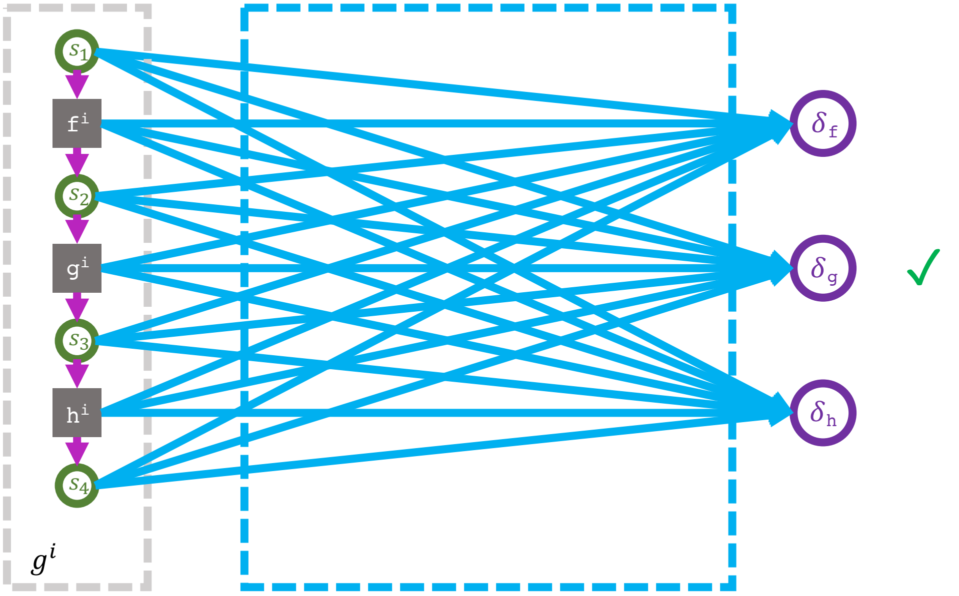
Formally represent learning algorithms as algorithmic causal graphs
independence = d-separation.

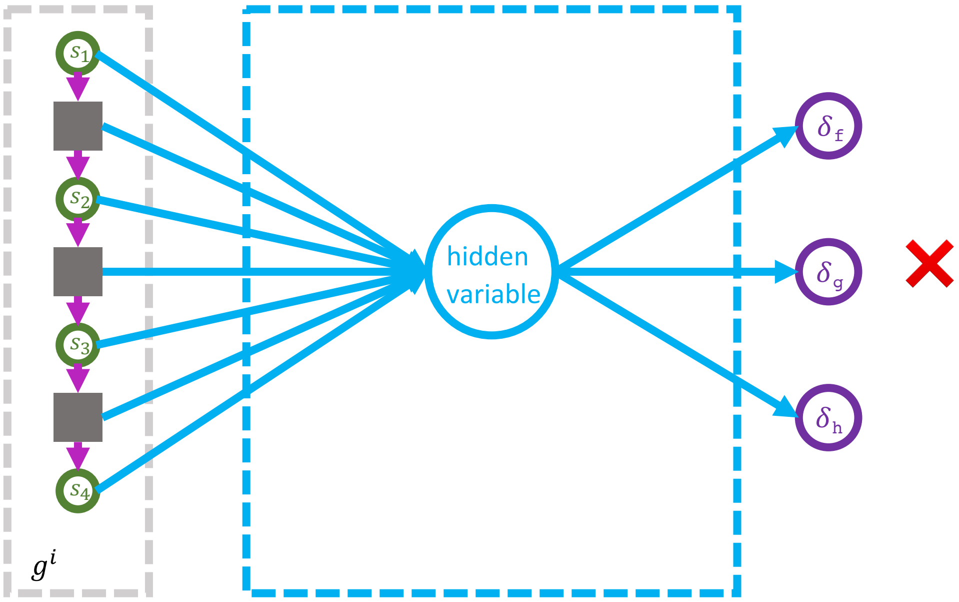


By treating the learning algorithm as itself an algorithmic causal graph, Learning algorithms are dynamic systems.

Modularity requires independent feedback (e.g. gradients).

Formally represent learning algorithms as algorithmic causal graphs
independence = d-separation.





we can test for this property, without any training, by checking whether the gradients are d-separated.

Modularity for Dynamic Systems

Independent Credit Assignment

Modularity of RL Algorithms

Hypothesis

Modularity → more efficient transfer

Expressing the hypothesis precisely

Testing the hypothesis

At this point we have developed a language for precisely expressing our hypothesis

Modularity for Dynamic Systems

Independent Credit Assignment

Hypothesis

Modularity → more efficient transfer

Expressing the hypothesis precisely

Modularity of RL Algorithms

Testing the hypothesis

by presenting a formal definition of modularity in learning systems, as well as a criterion to test for it.

Hypothesis

Modularity → more efficient transfer

Expressing the hypothesis precisely

Testing the hypothesis

In the last part of the talk, we can finally ask whether modularity in reinforcement learning improves transfer efficiency.

Theoretical question: Which reinforcement learning algorithms produce independent gradients?

We first compare major classes of reinforcement learning algorithms

Theoretical question: Which reinforcement learning algorithms produce independent gradients?

on whether they produce independent gradients over a credit assignment update.

Modularity for Dynamic Systems

Independent Credit Assignment

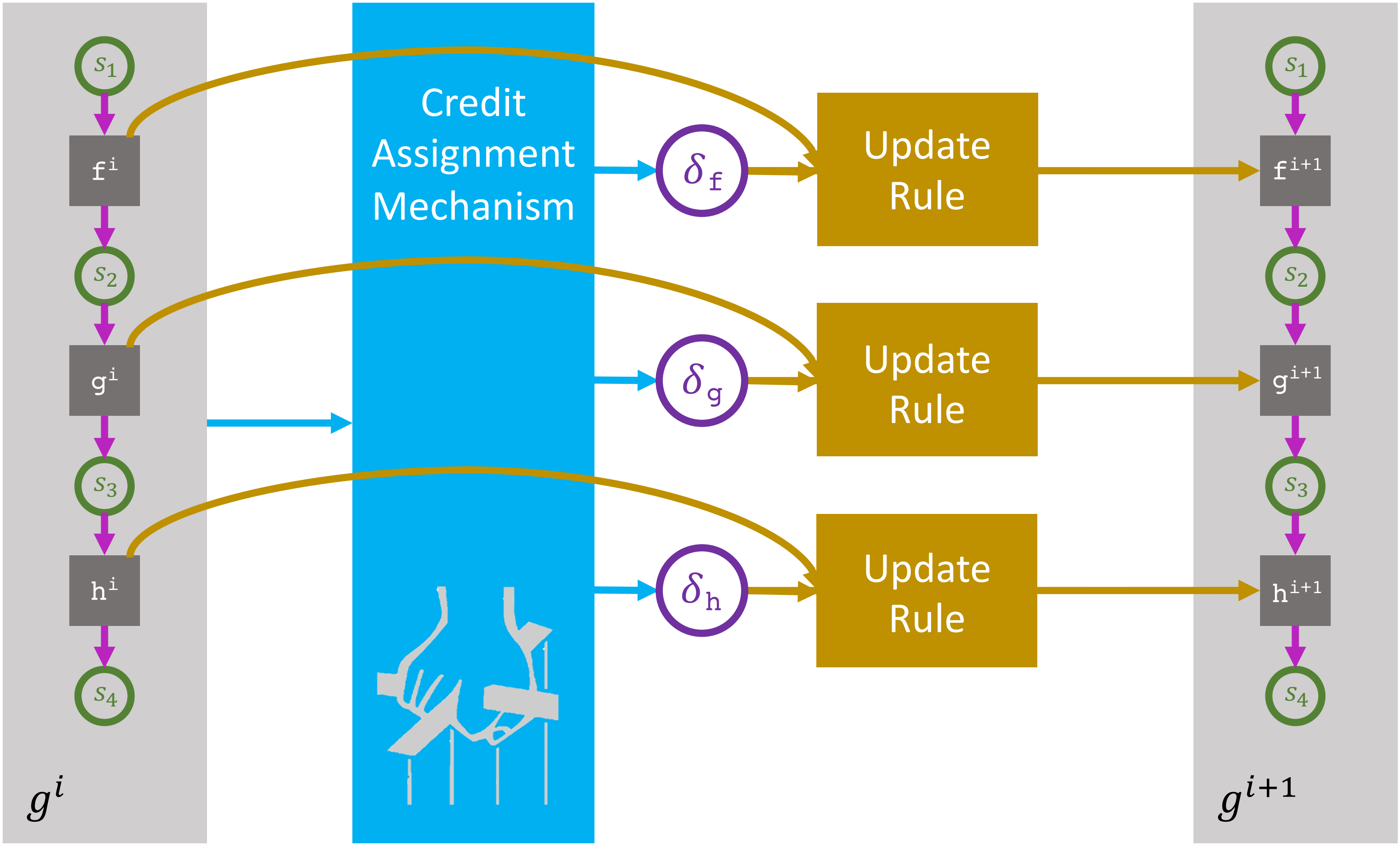
Modularity of RL Algorithms

Theoretical question: Which reinforcement learning algorithms produce independent gradients?

Empirical question: Does modularity improve transfer efficiency?

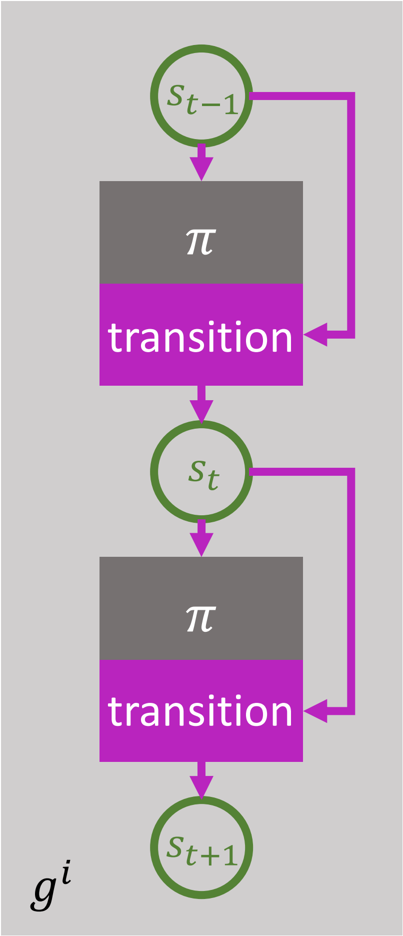
Then we empirically test whether having this property correlates with transfer efficiency.

RL Algorithms as Causal Graphs



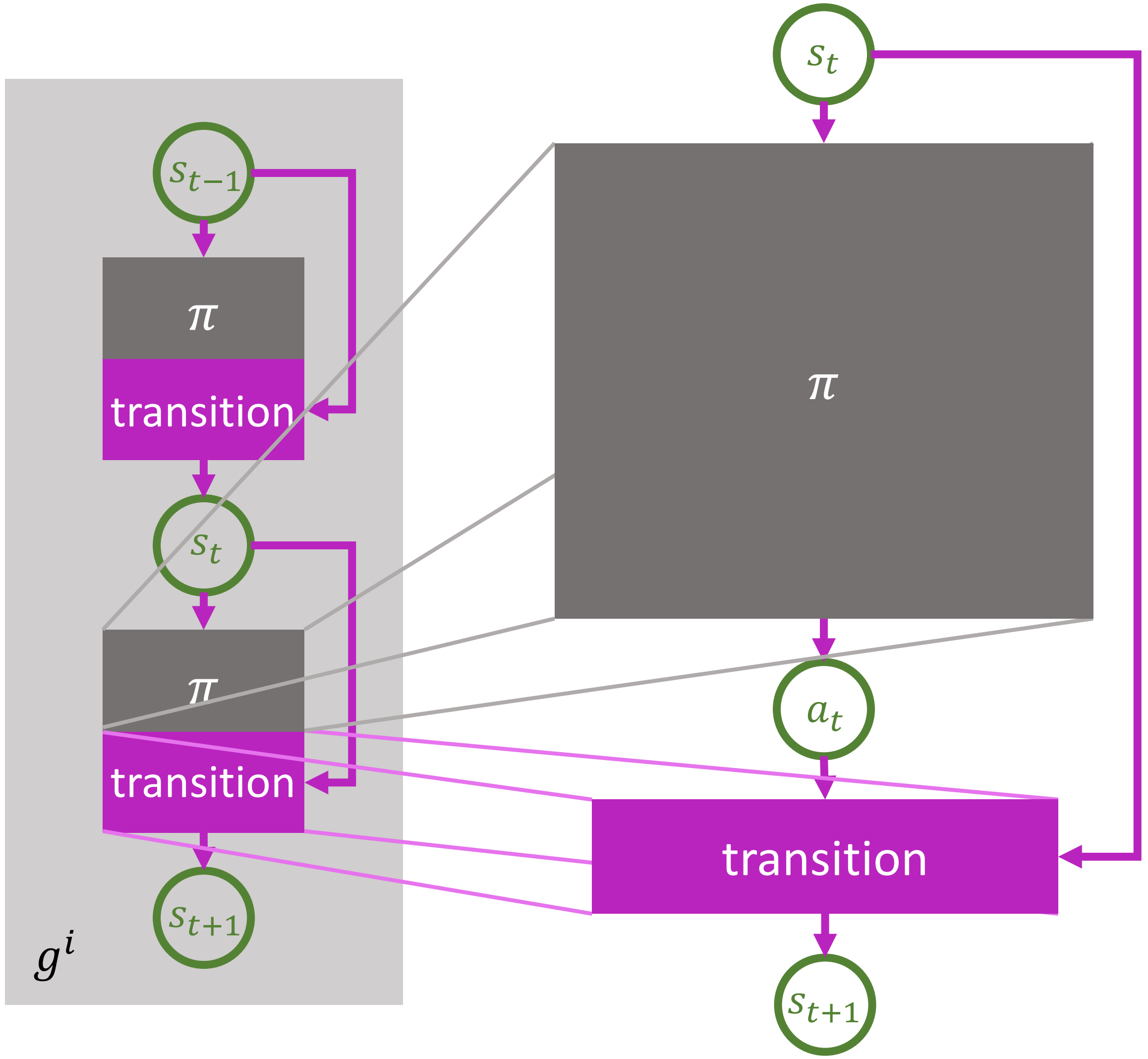
We now apply our framework to reinforcement learning algorithms by representing these algorithms as causal graphs.

RL Algorithms as Causal Graphs



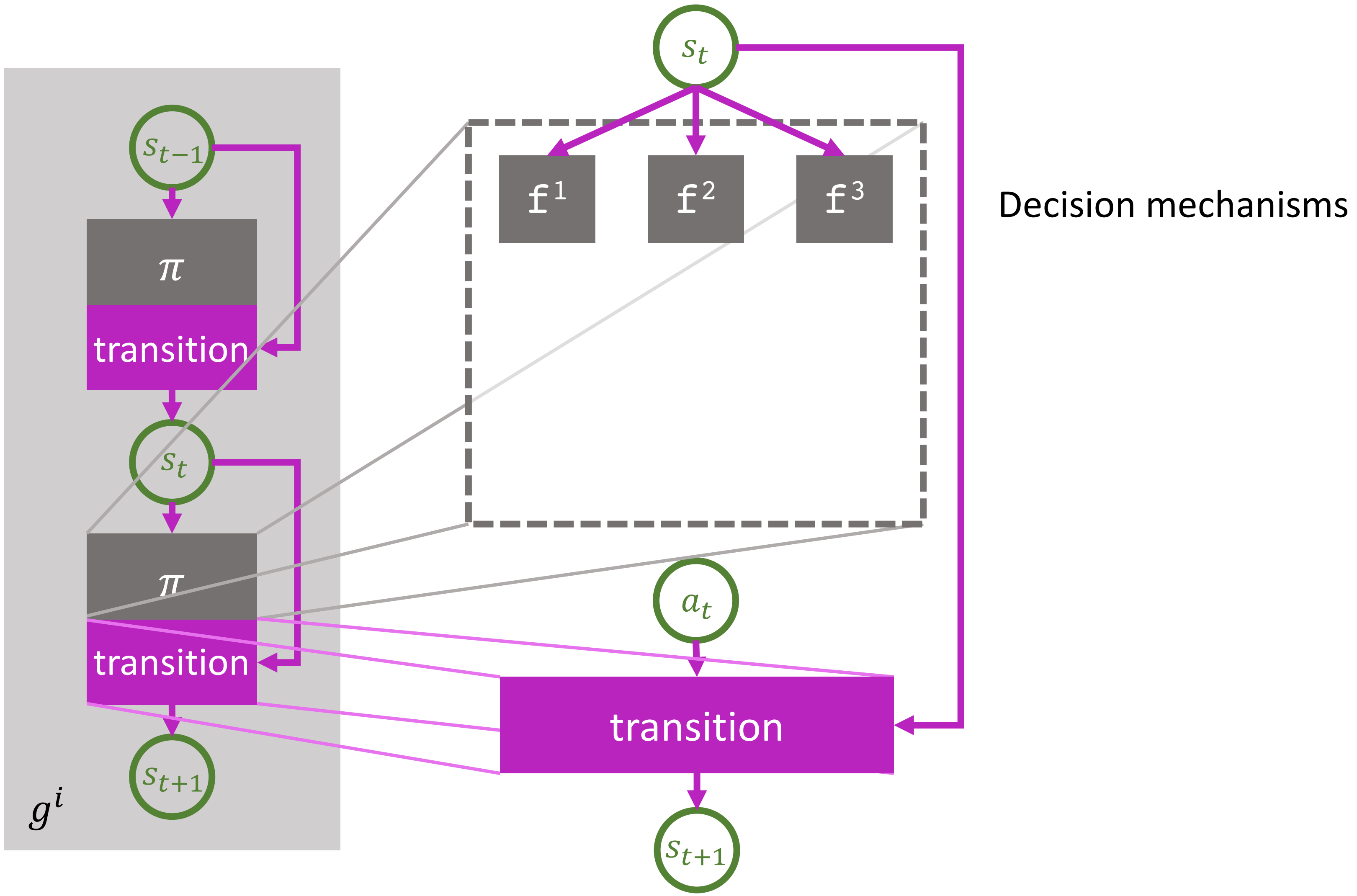
In reinforcement learning, the forward pass of the learner is a rollout in the MDP.

RL Algorithms as Causal Graphs



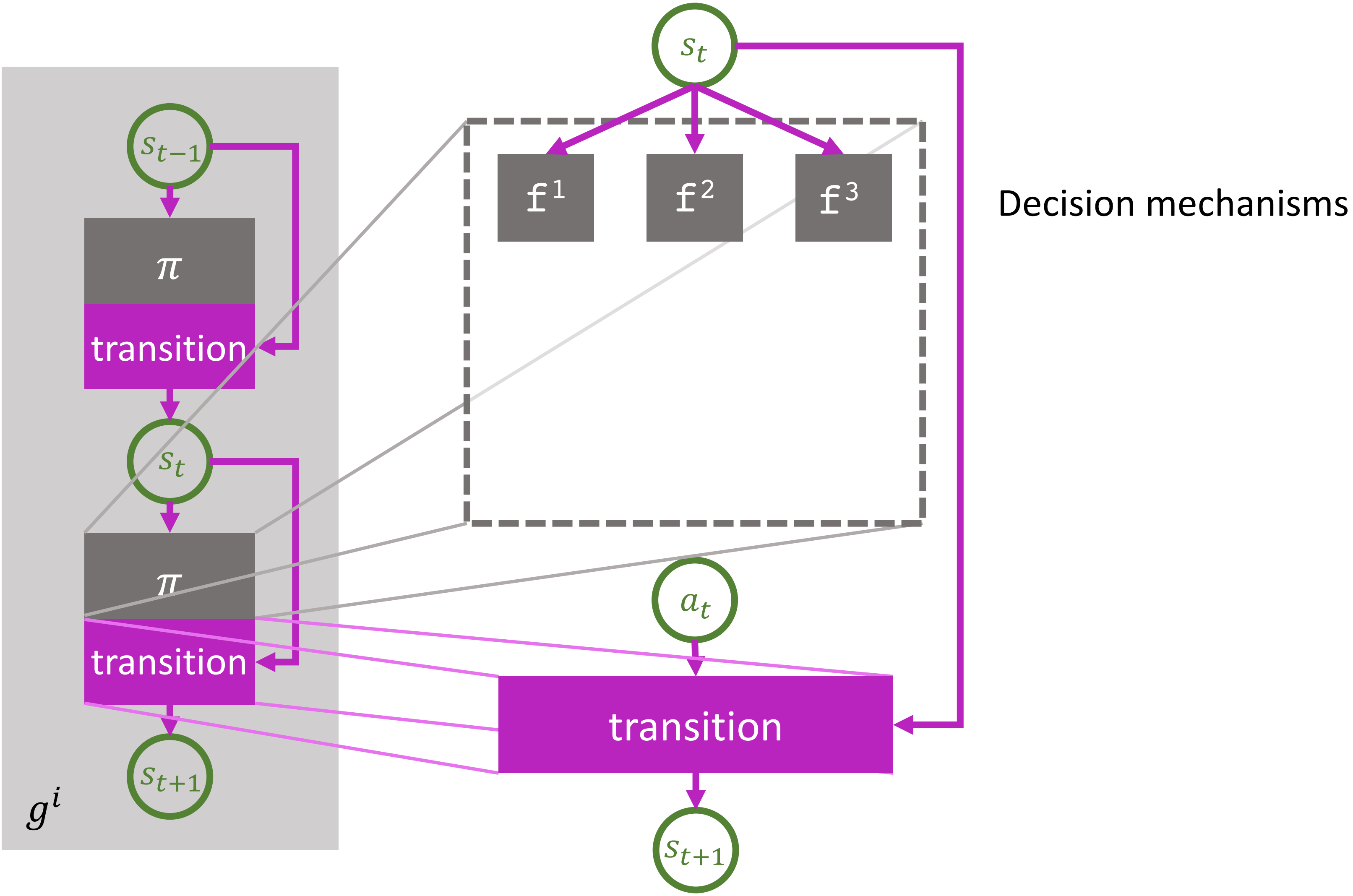
When we consider modularity in reinforcement learning,

RL Algorithms as Causal Graphs



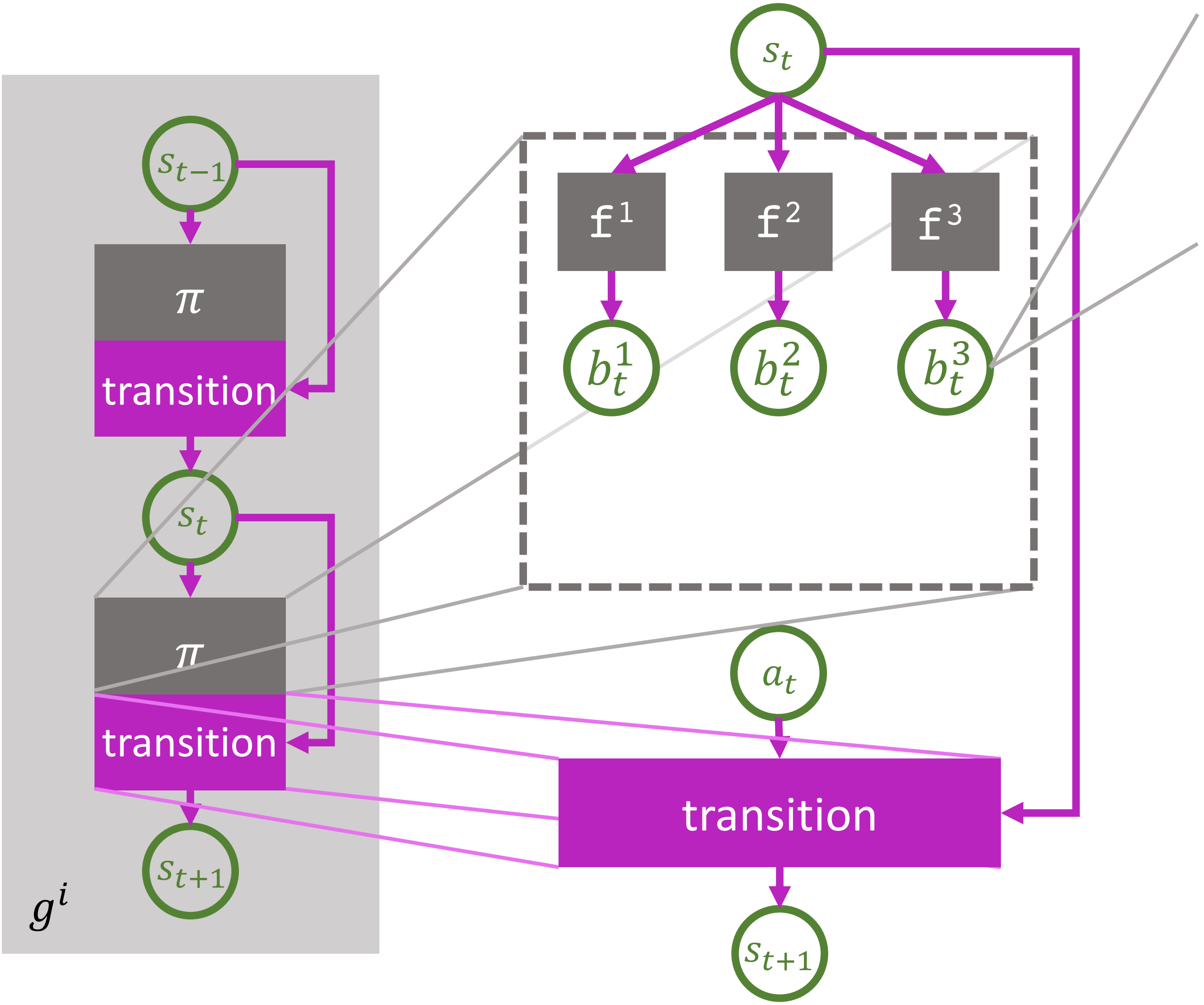
we are interested in the independence of the decision mechanisms that control each value that the action can take on.

RL Algorithms as Causal Graphs



In this case there are three possible values of the action variable.

RL Algorithms as Causal Graphs

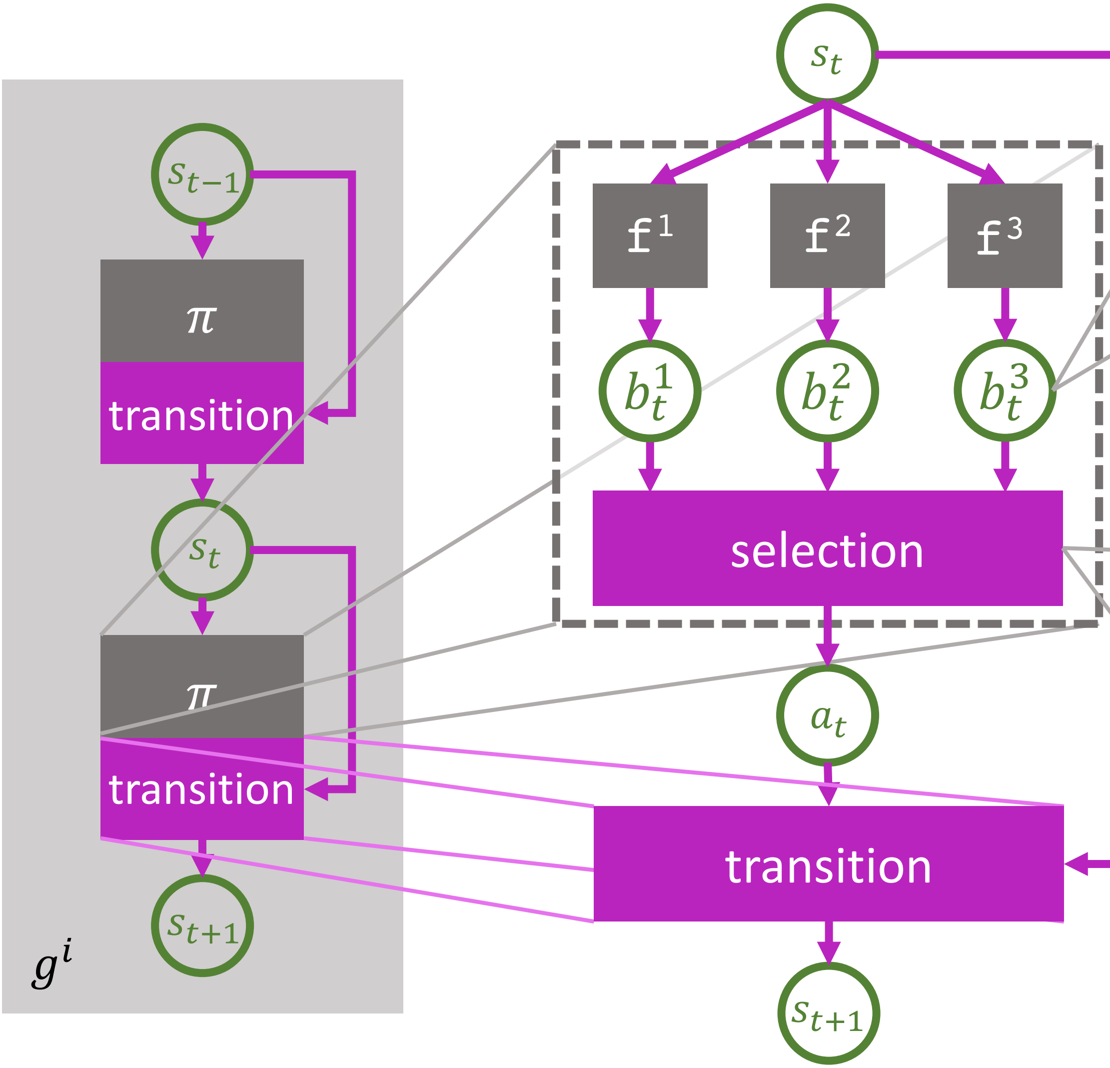


Policy gradient methods:
Action probability $p(a_t | \cdot)$

Action value methods:
Estimated Q-value $Q(\cdot, a_t)$

Each decision mechanism produces a "bid," which could correspond to an action probability or estimated Q-value.

RL Algorithms as Causal Graphs



Policy gradient methods:
Action probability $p(a_t | \cdot)$

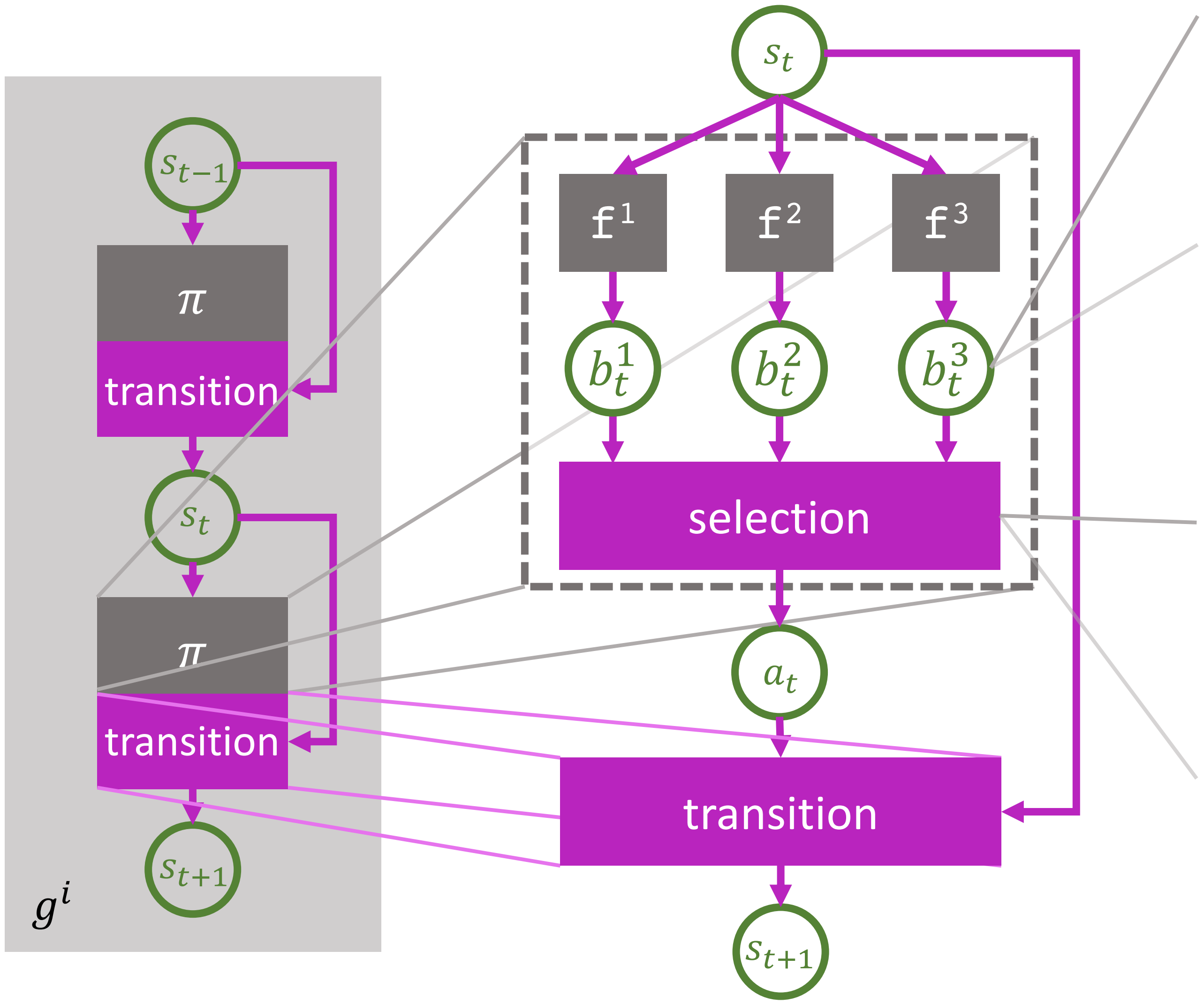
Action value methods:
Estimated Q-value $Q(\cdot, a_t)$

Policy gradient methods:
Categorical sampler

Action value methods:
Epsilon-greedy
Vickrey auction (Chang et al., ICML 2020)

These bids get filtered by a selection mechanism,

RL Algorithms as Causal Graphs



Policy gradient methods:
Action probability $p(a_t | \cdot)$

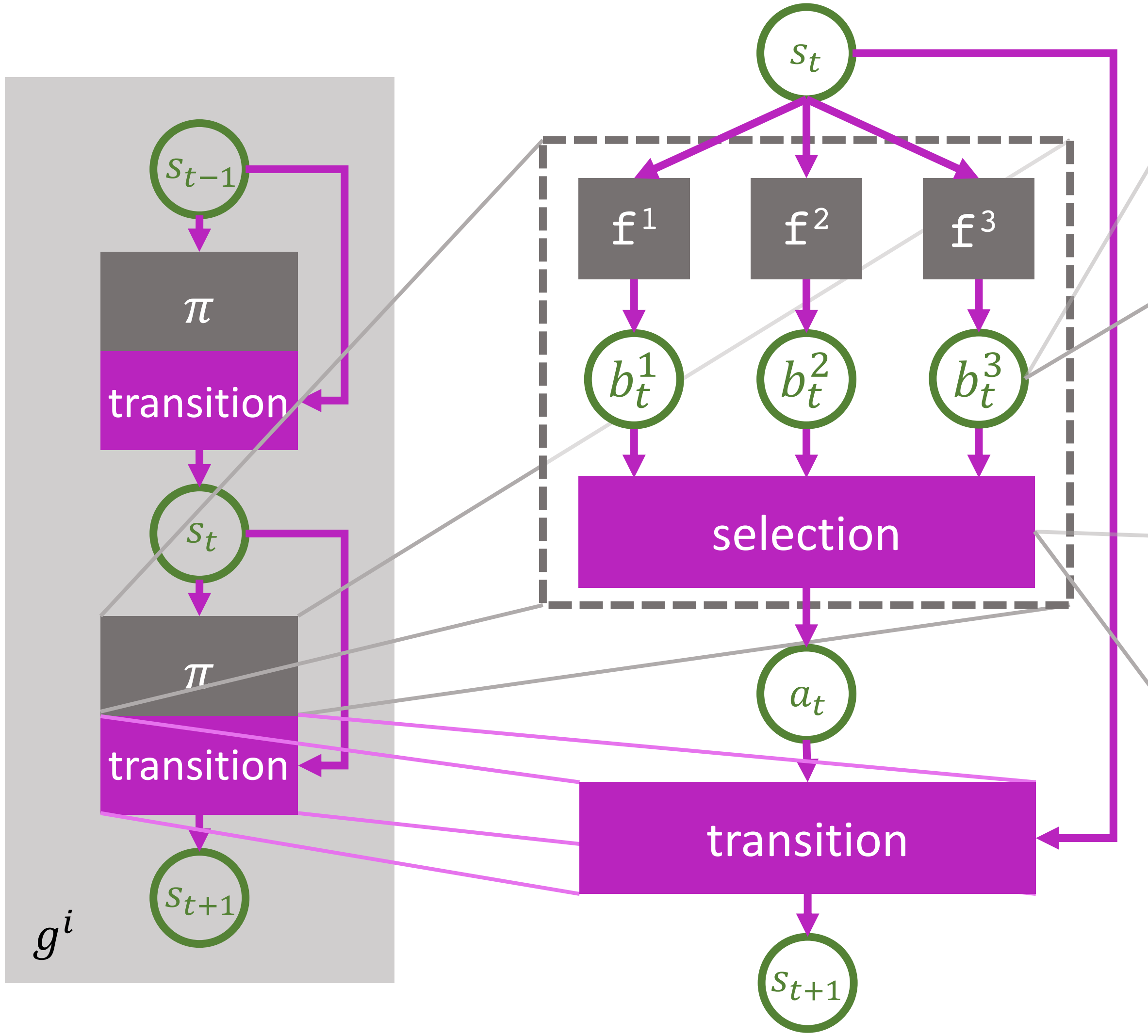
Action value methods:
Estimated Q-value $Q(\cdot, a_t)$

Policy gradient methods:
Categorical sampler

Action value methods:
Epsilon-greedy
Vickrey auction (Chang et al., ICML 2020)

such as a categorical sampler for policy gradient methods,

RL Algorithms as Causal Graphs



Policy gradient methods:
Action probability $p(a_t | \cdot)$

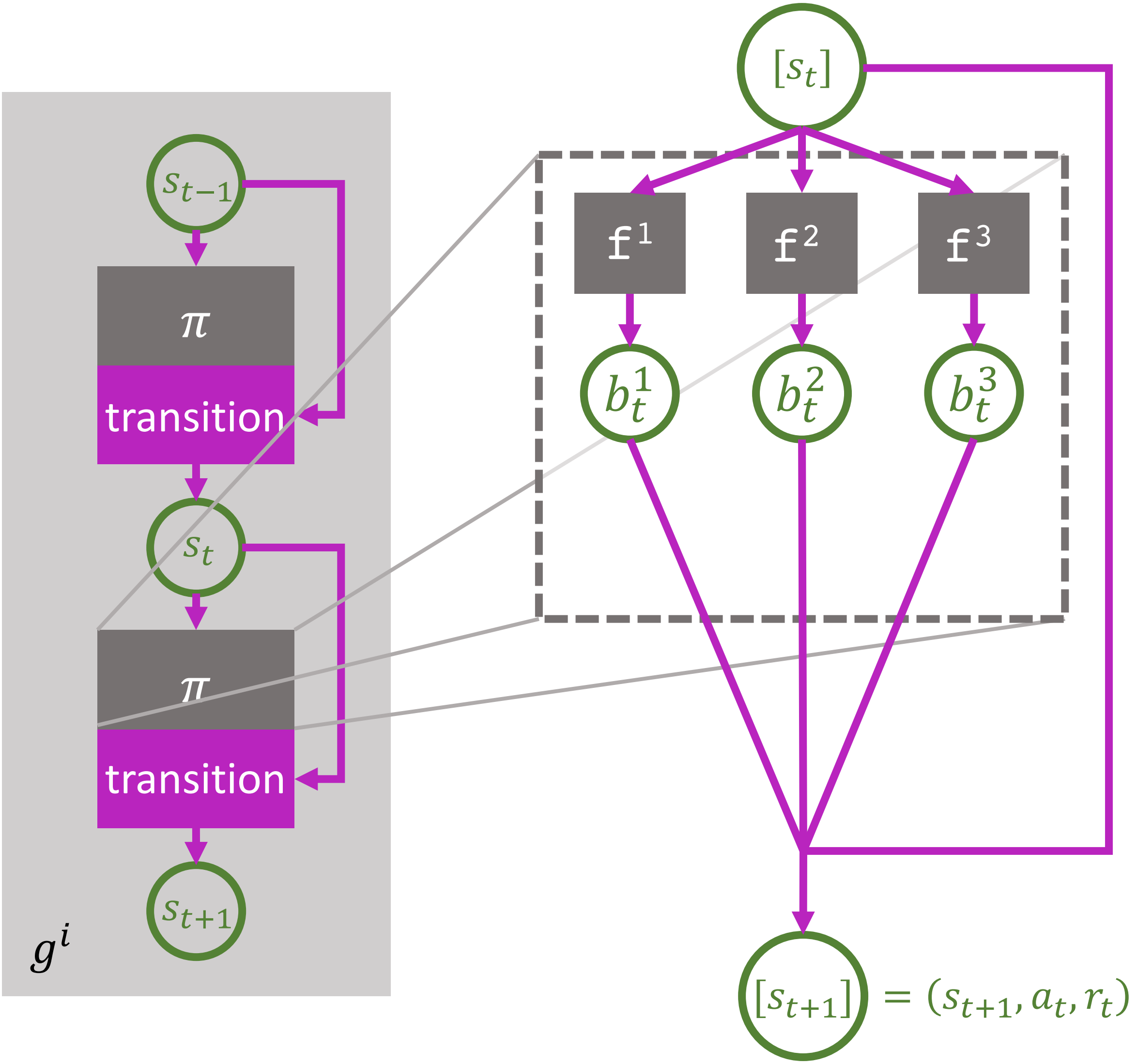
Action value methods:
Estimated Q-value $Q(\cdot, a_t)$

Policy gradient methods:
Categorical sampler

Action value methods:
Epsilon-greedy
Vickrey auction (Chang et al., ICML 2020)

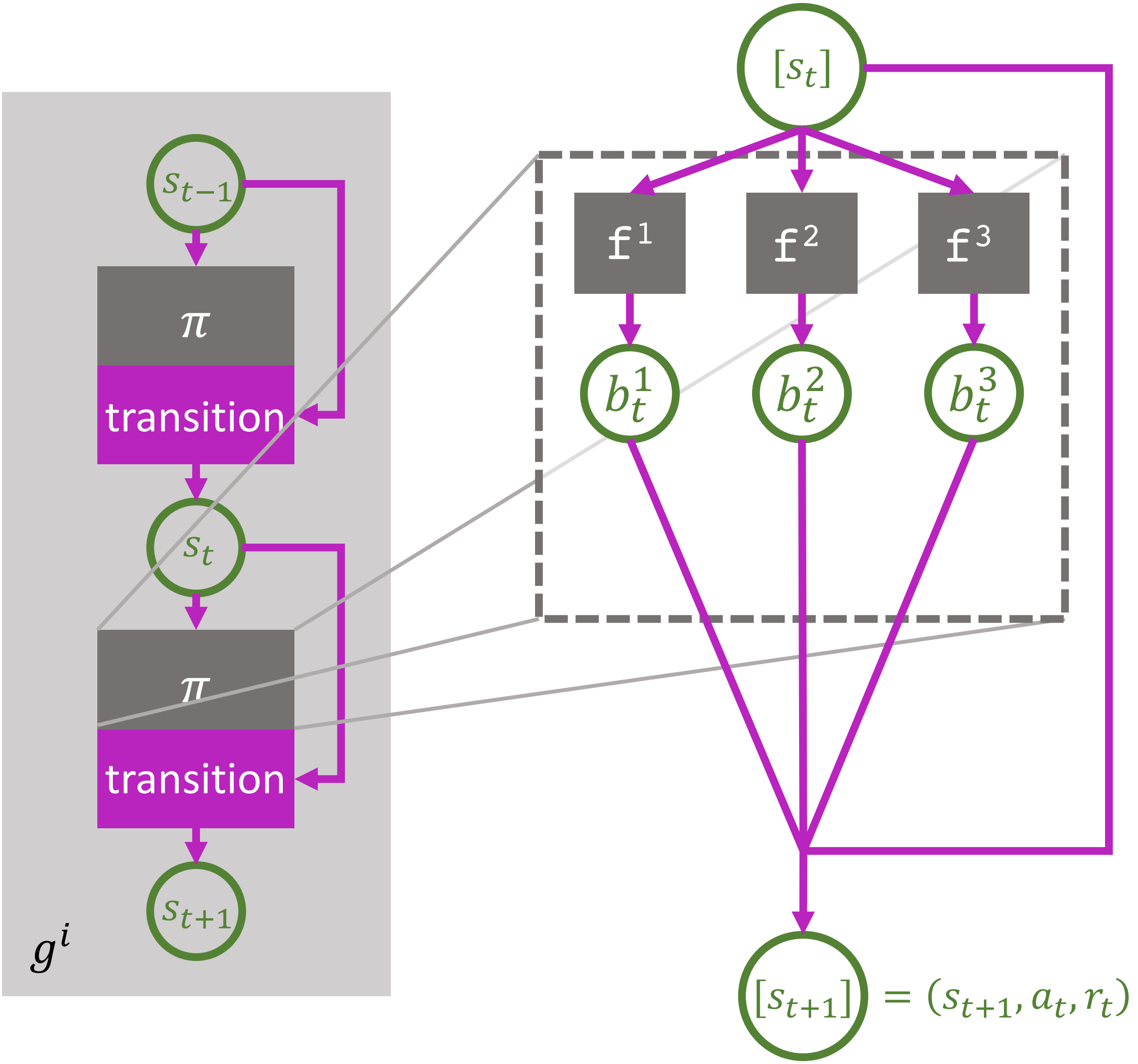
or an epsilon greedy sampler or Vickrey auction for action-value methods.

RL Algorithms as Causal Graphs



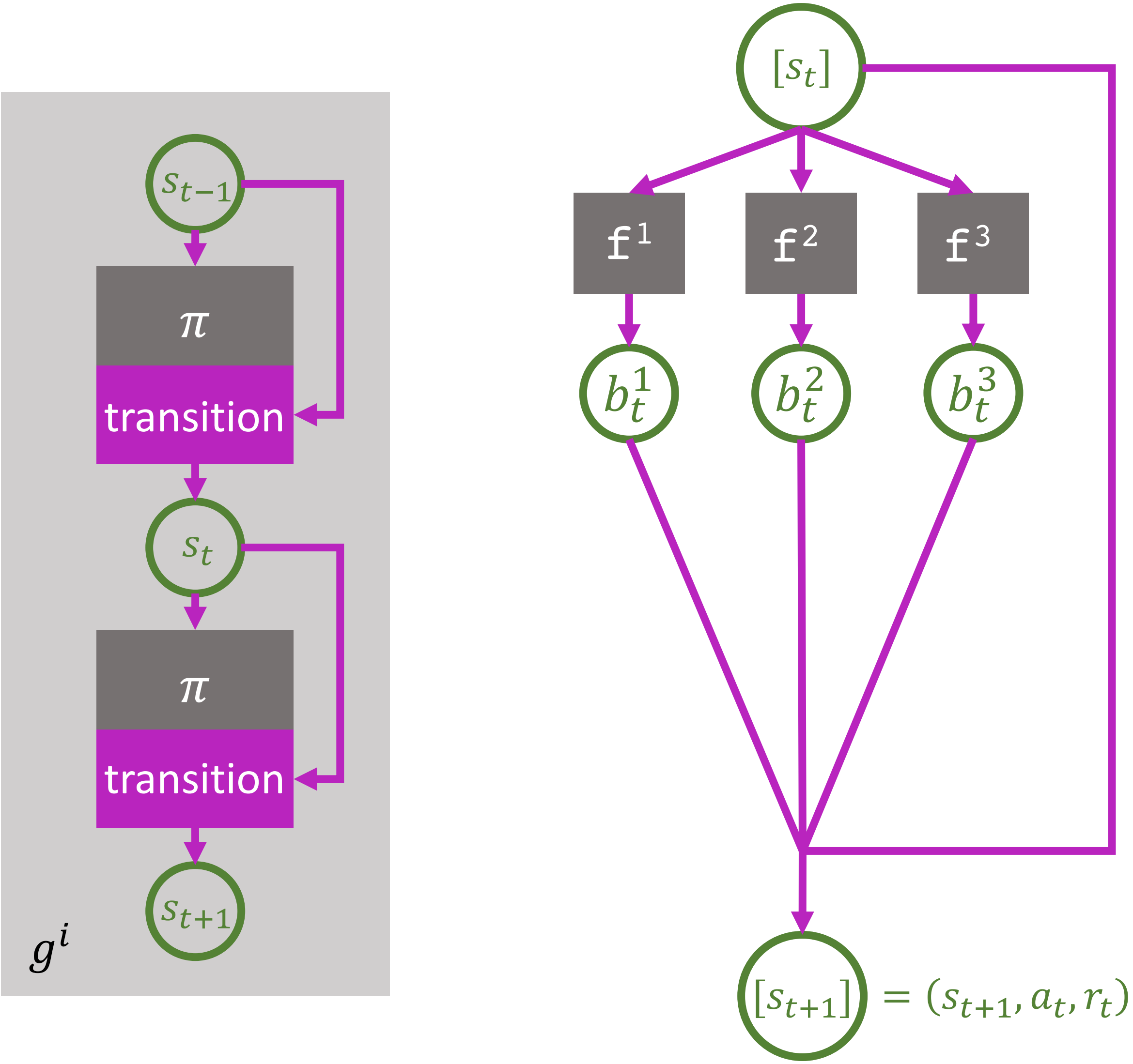
Since we are only interested in the modularity of these decision mechanisms, we can absorb the other operations into the edges,

RL Algorithms as Causal Graphs



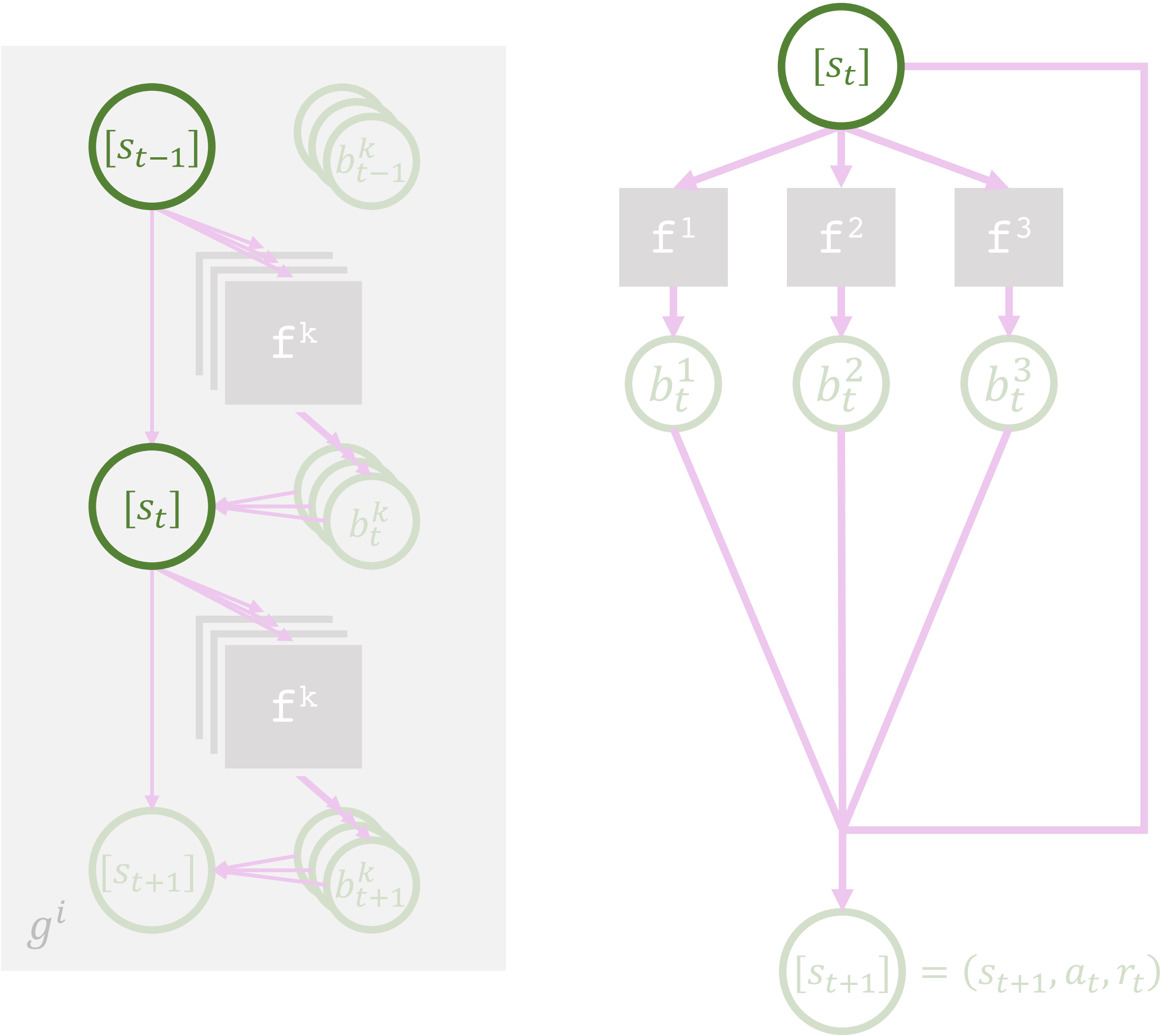
and use brackets to denote that we bundle the states, actions, and rewards together.

RL Algorithms as Causal Graphs



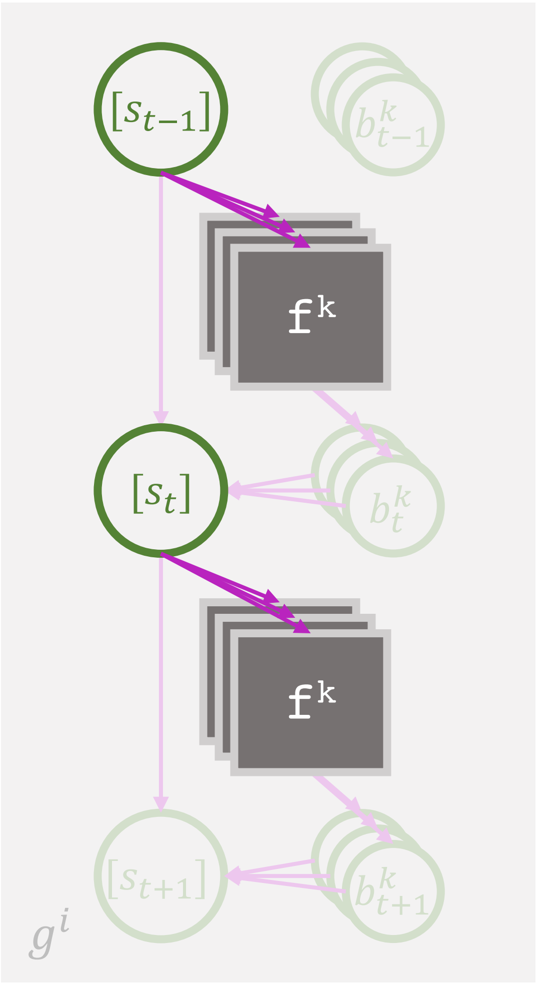
We have now decomposed the causal graph of a single step of the forward pass of the RL algorithm.

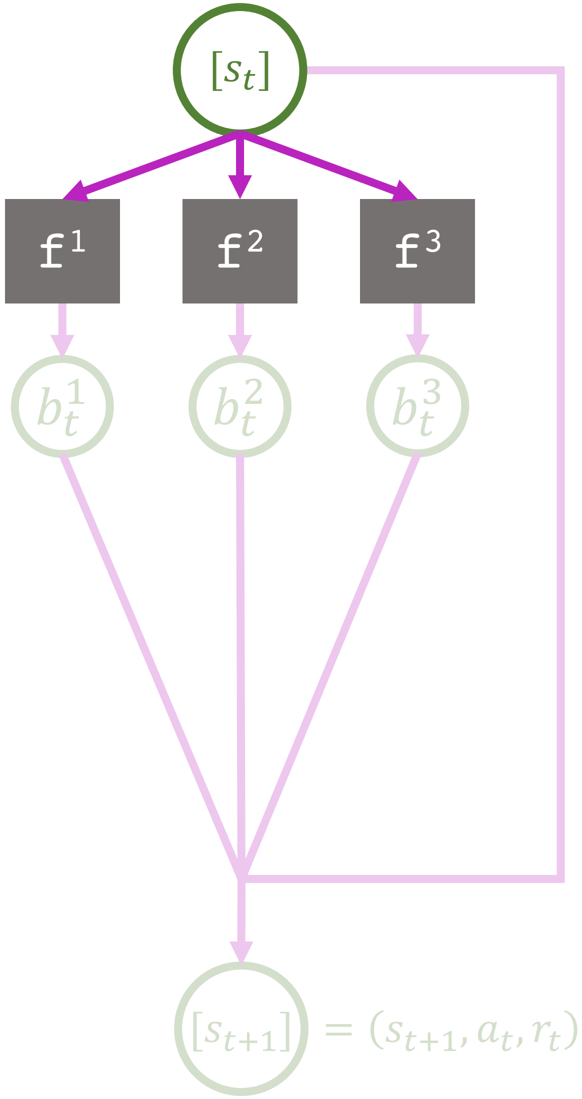
RL Algorithms as Causal Graphs



Here is the causal decomposition again, shown across time.

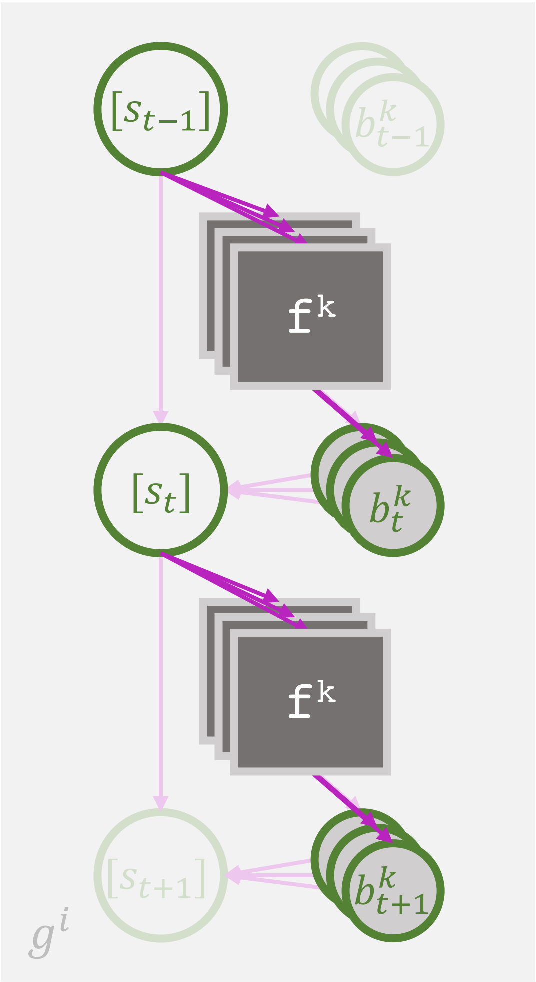
RL Algorithms as Causal Graphs

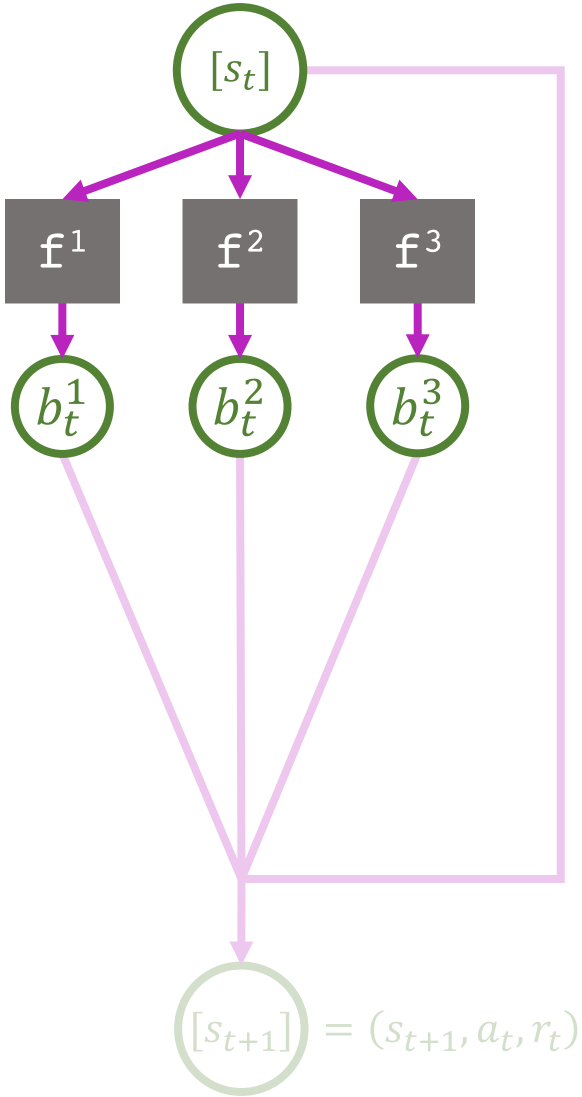




Here is the causal decomposition again, shown across time.

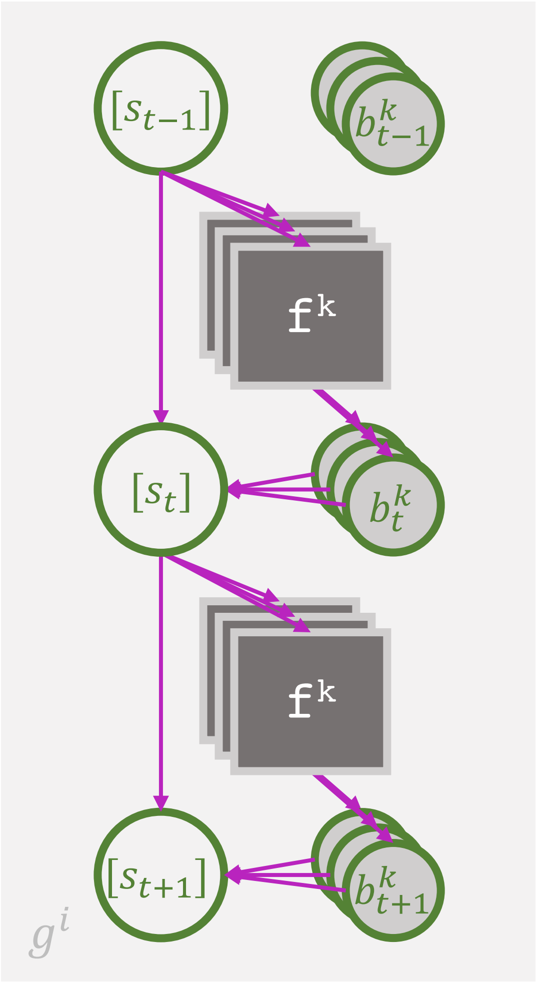
RL Algorithms as Causal Graphs

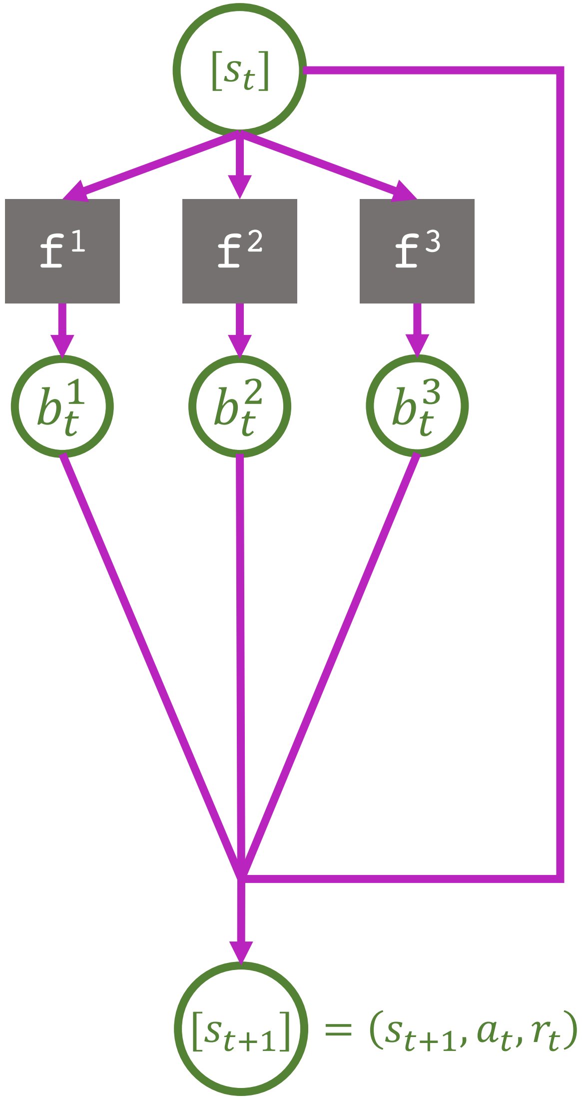




Here is the causal decomposition again, shown across time.

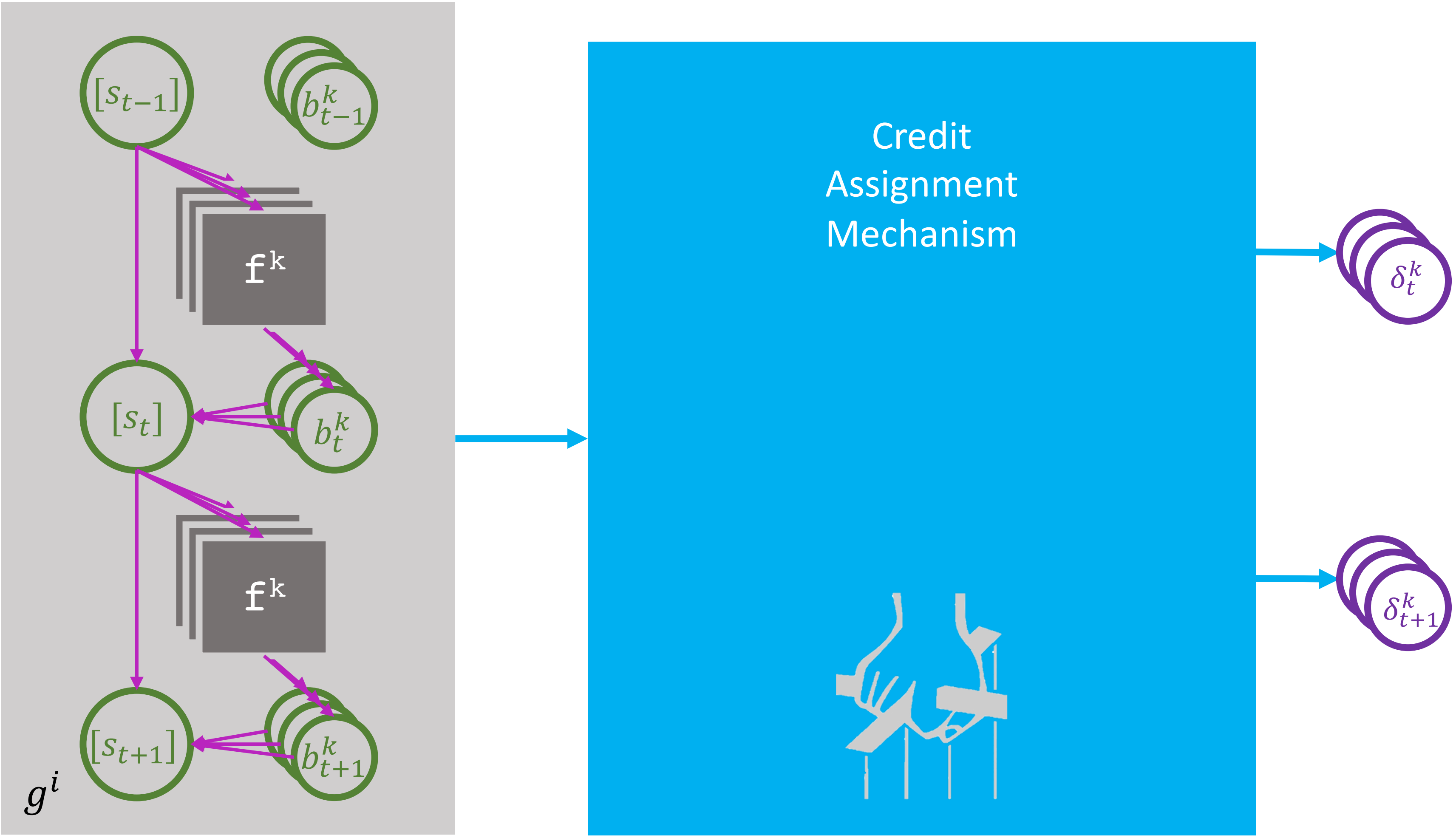
RL Algorithms as Causal Graphs





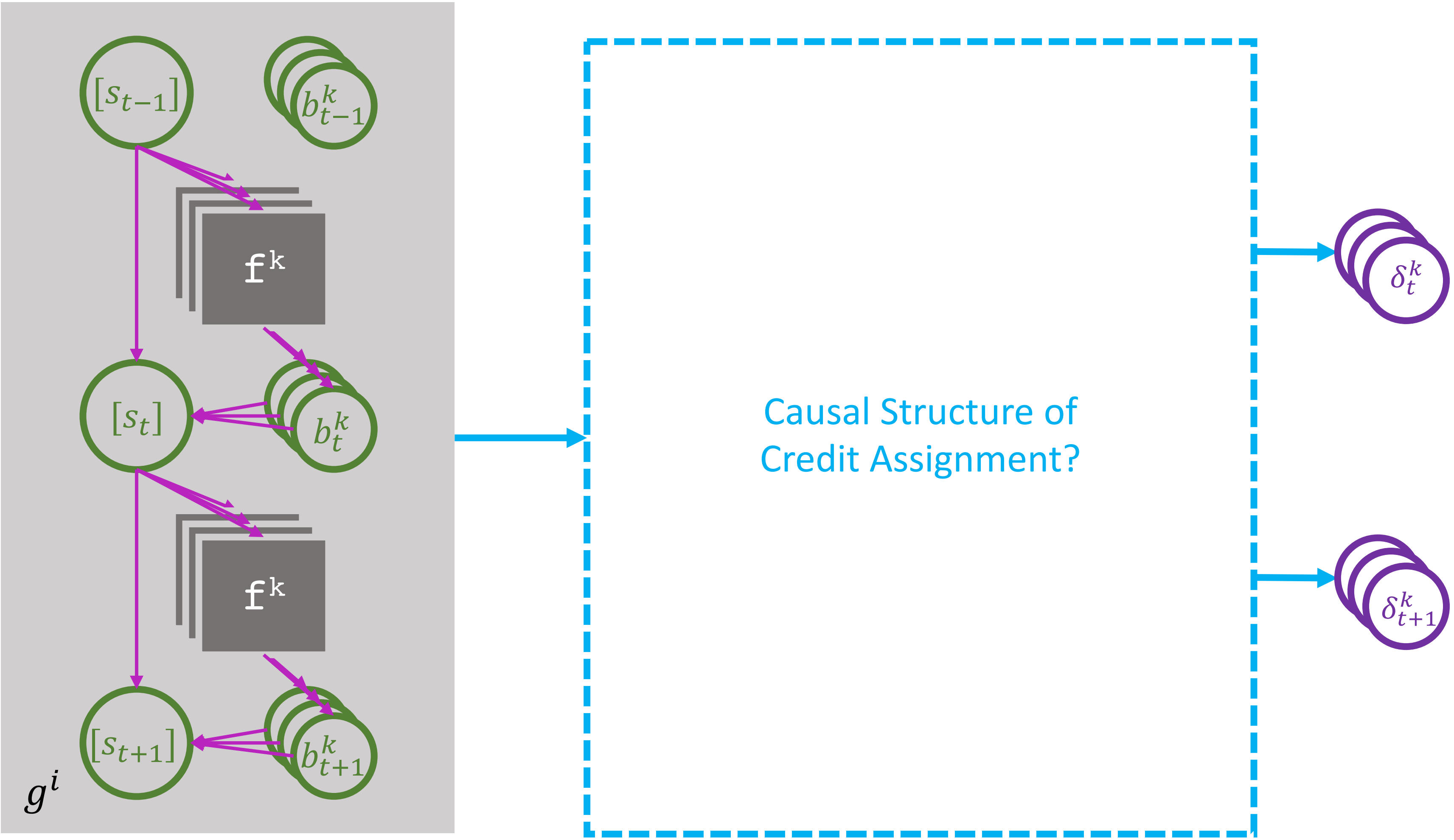
Here is the causal decomposition again, shown across time.

Structure of Credit Assignment in RL



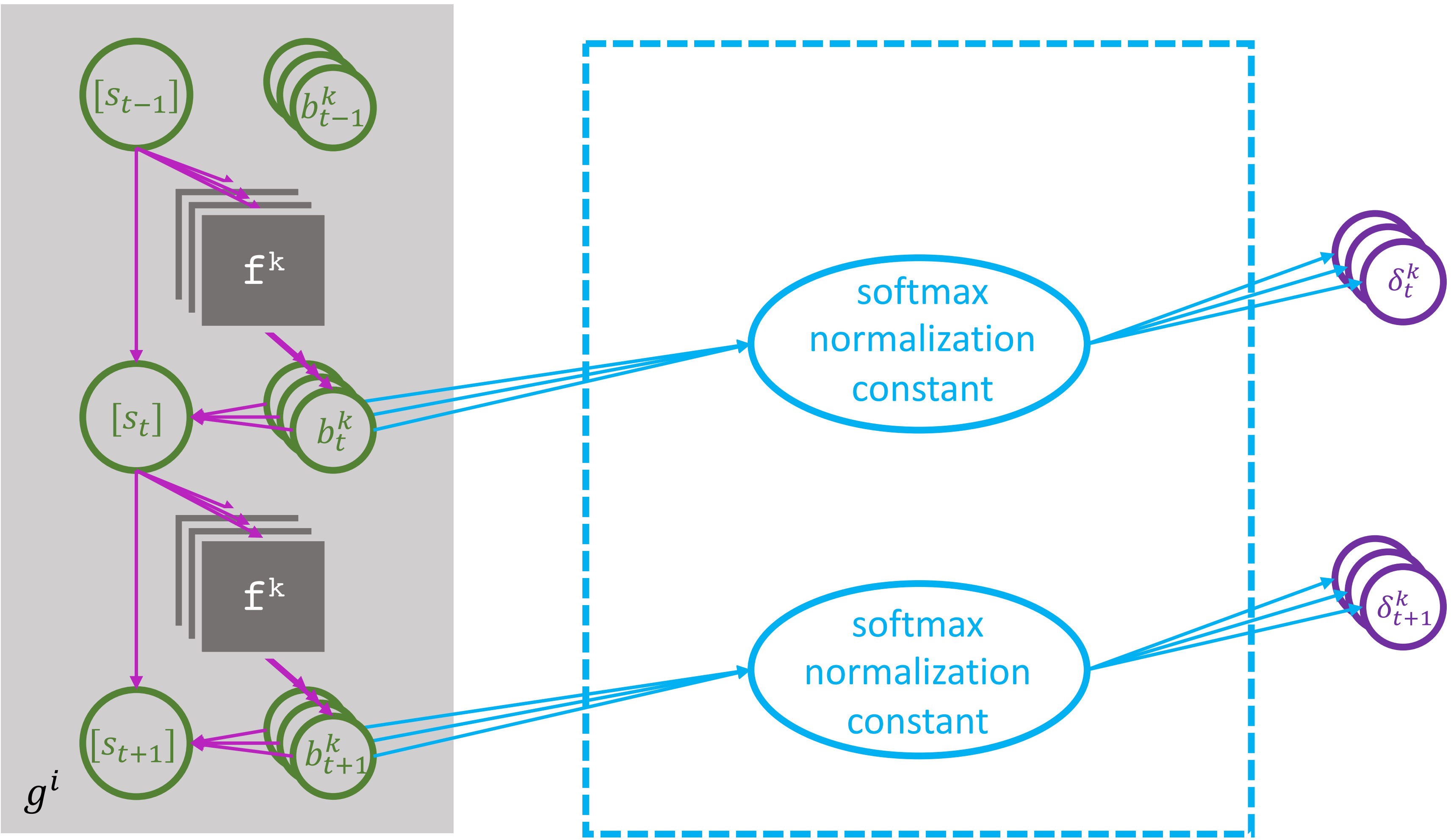
As before, the credit assignment mechanism is the outer process that produces gradients for the graph.

Structure of Credit Assignment in RL



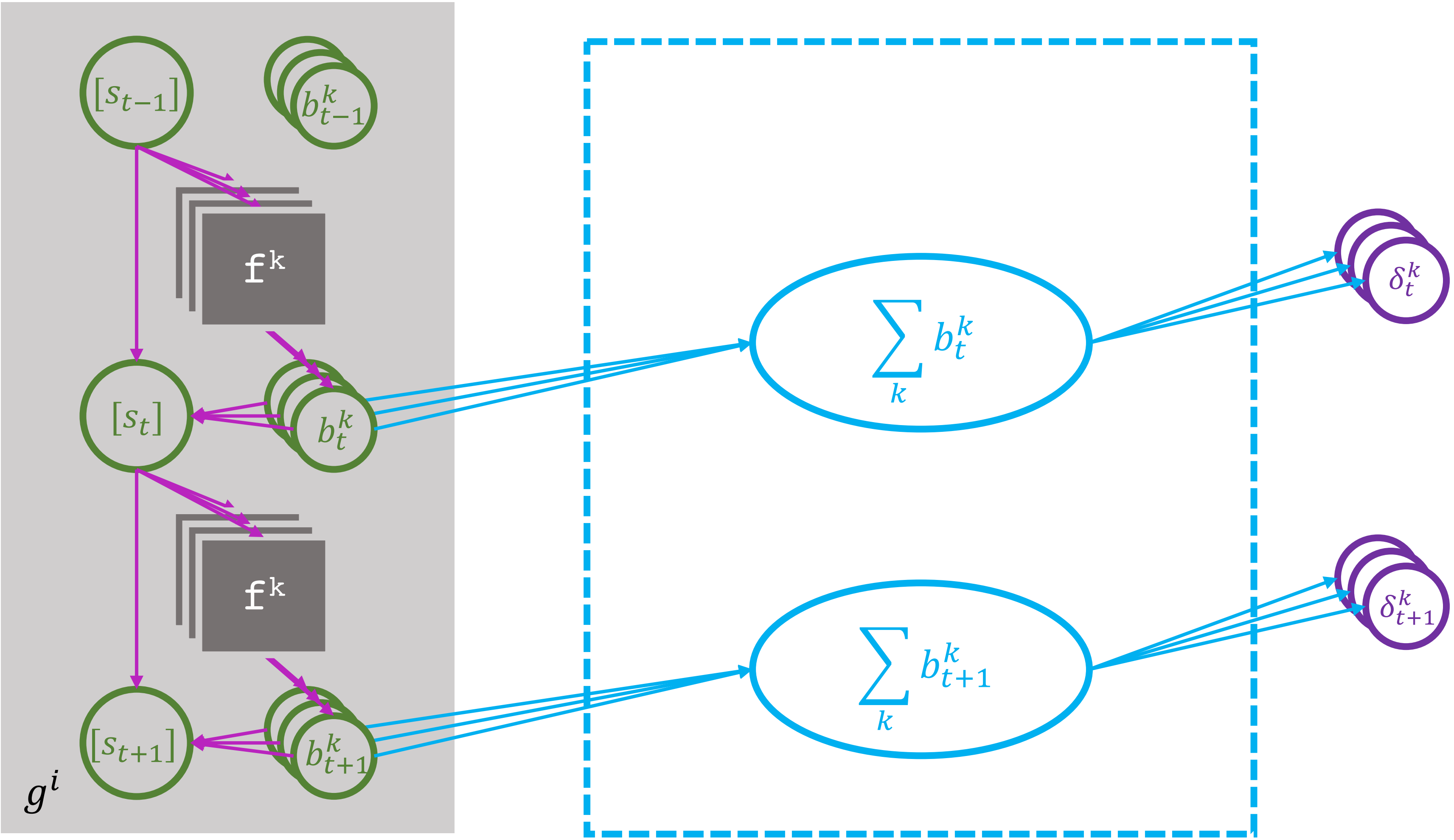
We now analyze the causal structure of the credit assignment mechanism for various RL algorithms.

Policy Gradients: Not Modular



For policy gradient methods, because of the softmax in the policy, the gradients of the decision mechanisms

Policy Gradients: Not Modular



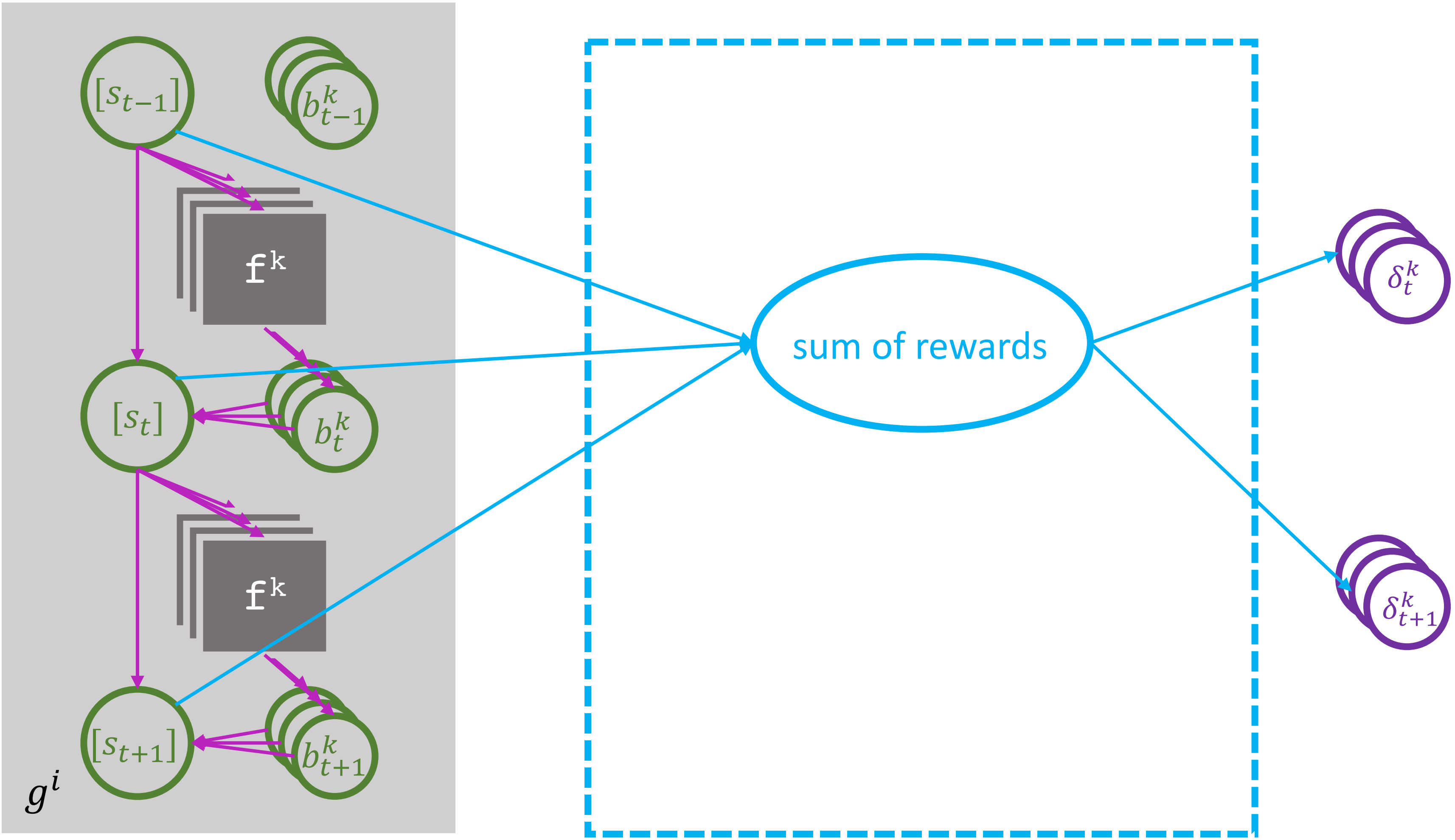
all share a normalization constant as a hidden variable, so the gradients produced are not d-separated.

Policy Gradients: Not Modular

Corollary (**policy gradient**): Policy gradient methods do not produce independent gradients.

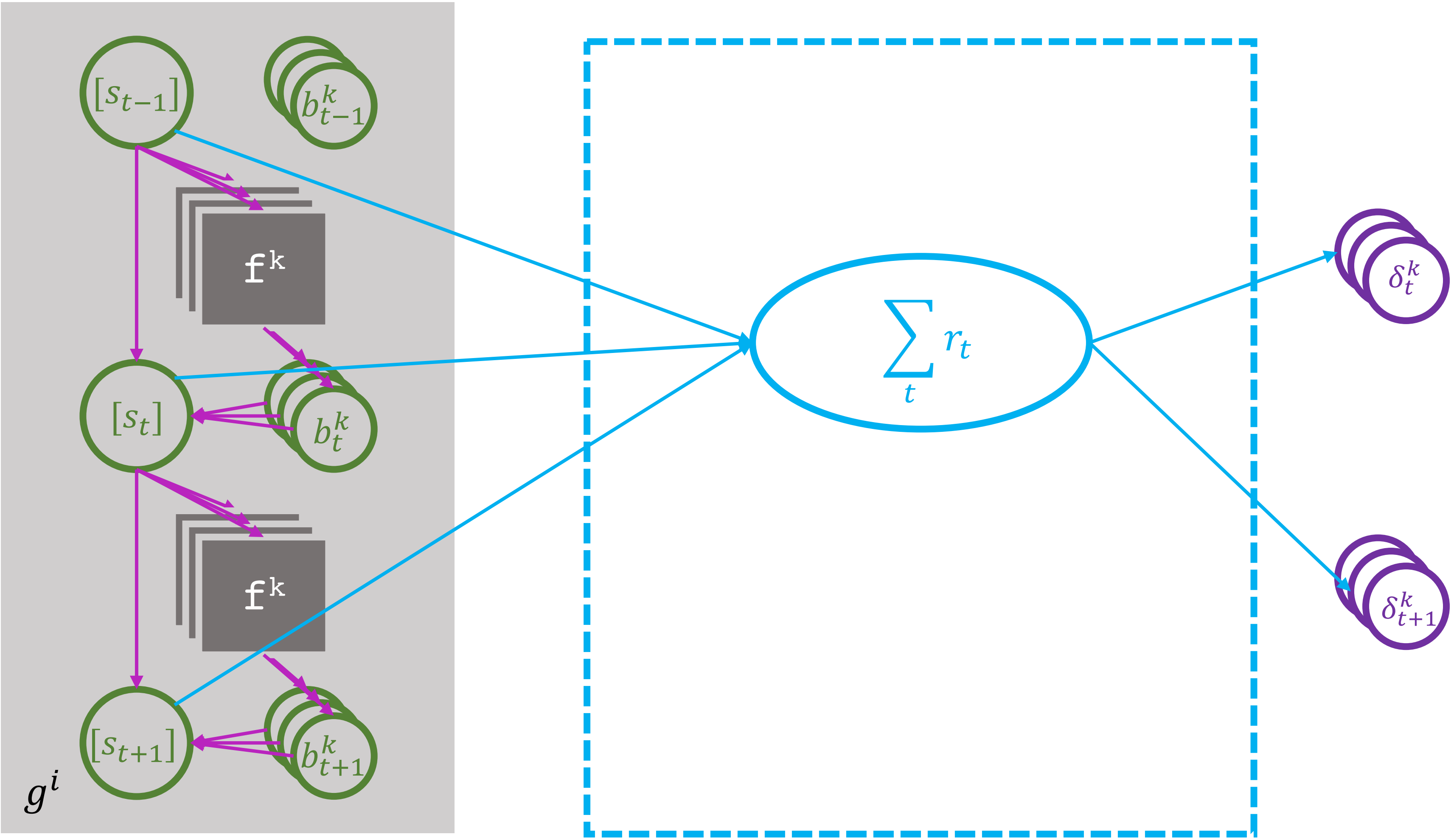
Therefore, policy gradient algorithms are not modular.

N-Step Temporal Difference: Not Modular



For n-step temporal difference methods, the gradients all share some sum over sampled rewards as a hidden variable,

N-Step Temporal Difference: Not Modular



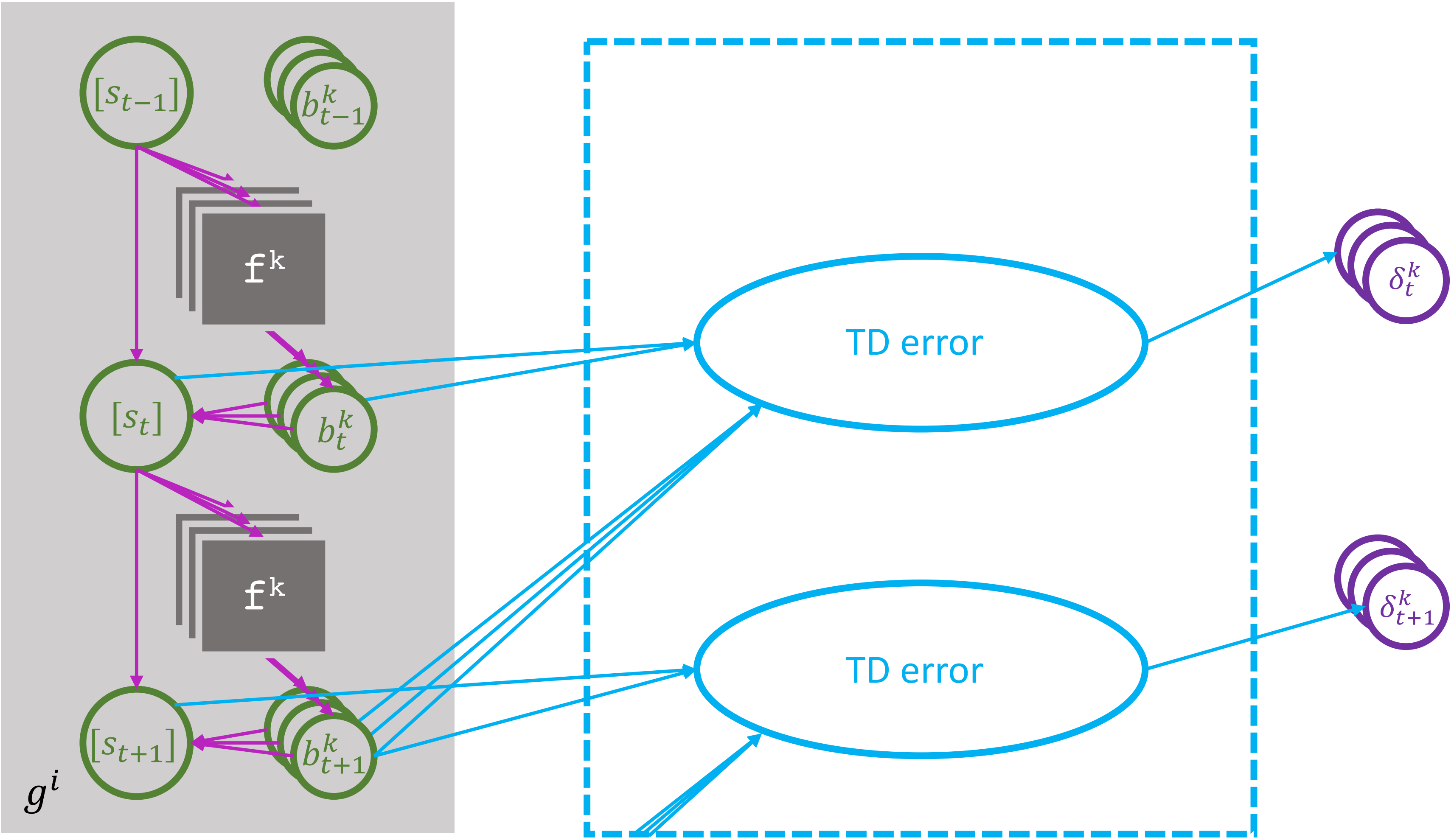
so the gradients produced are not d-separated,

N-Step Temporal Difference: Not Modular

Corollary (n-step TD): n-step temporal difference methods do not produce independent gradients.

so n-step temporal difference algorithms are not modular either.

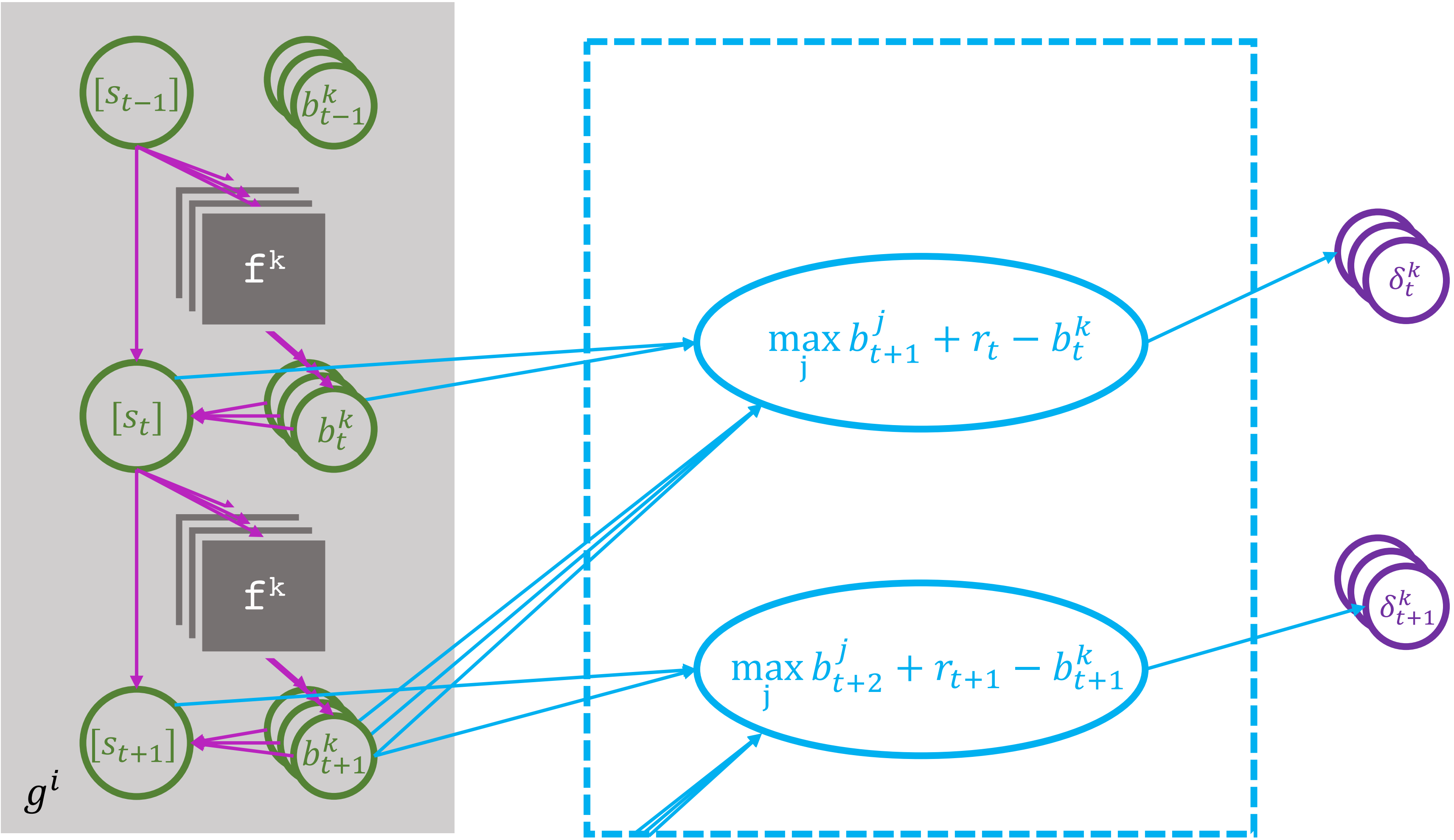
Single-Step Temporal Difference: Modular*



*For acyclic trajectories, see paper for more details.

This leaves single-step temporal difference methods, which also have an intermediate variable: the TD error.

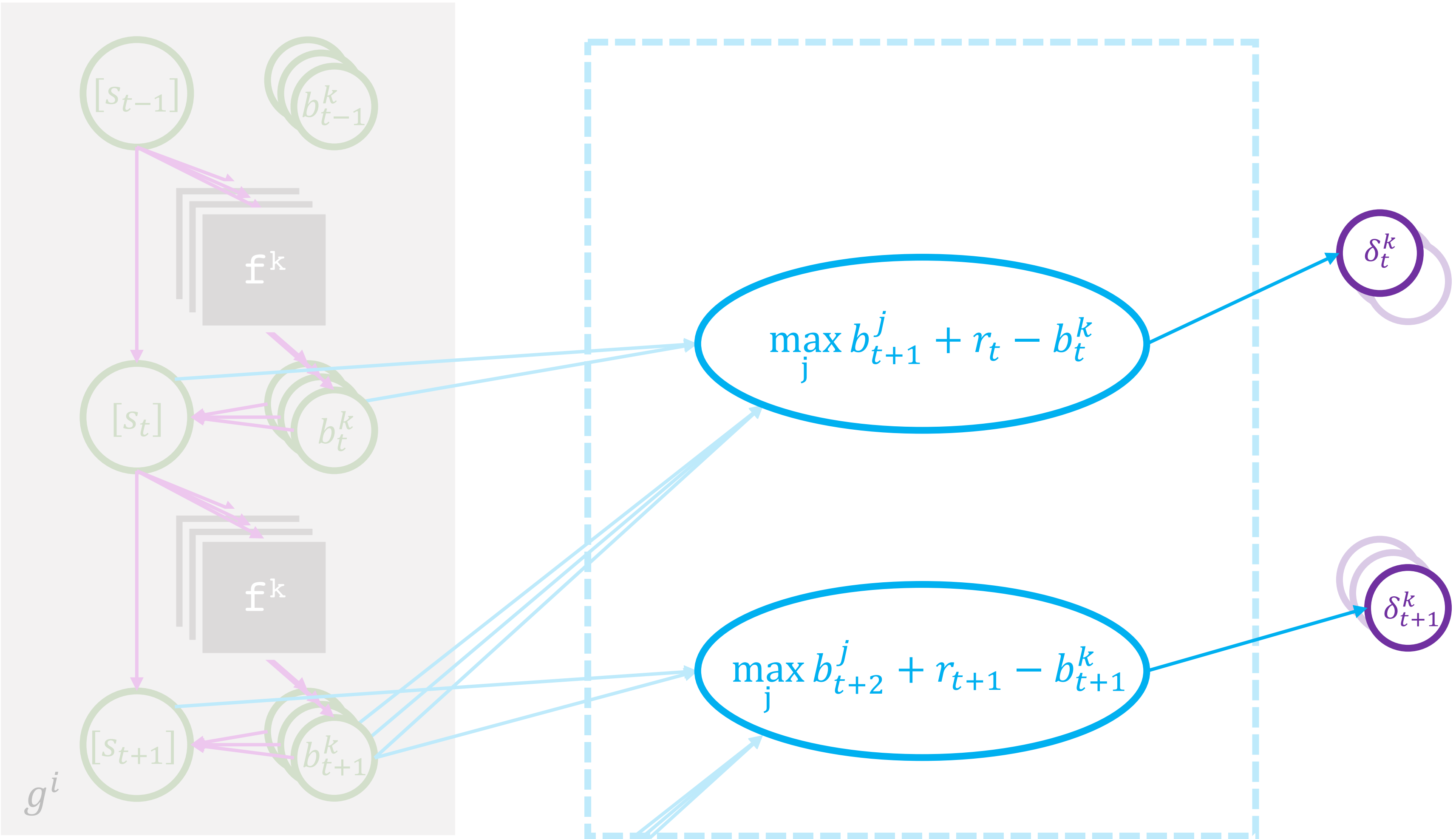
Single-Step Temporal Difference: Modular*



*For acyclic trajectories, see paper for more details.

But unlike policy gradient or n-step temporal difference methods, this hidden variable is not shared

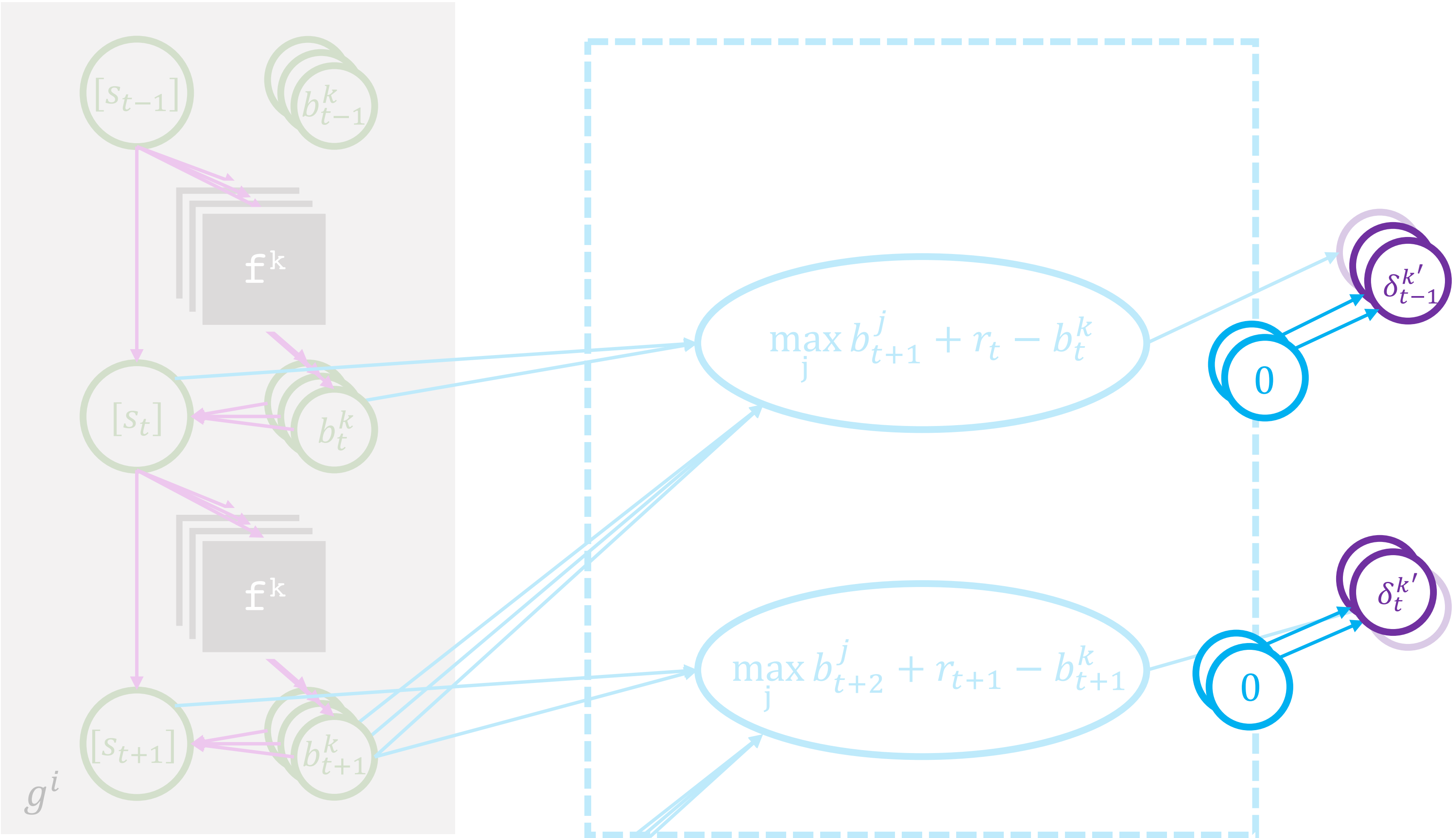
Single-Step Temporal Difference: Modular*



*For acyclic trajectories, see paper for more details.

because it only is connected to the gradient of the decision mechanism of the action that actually was taken

Single-Step Temporal Difference: Modular*



*For acyclic trajectories, see paper for more details.

while the gradients of the other decision mechanisms remain zero.

Single-Step Temporal Difference: Modular*

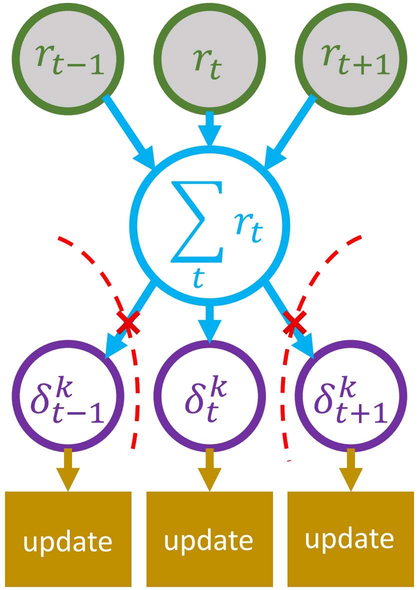
Corollary (**single-step TD**): single-step temporal difference methods produce independent gradients*.

*For acyclic trajectories, see paper for more details.

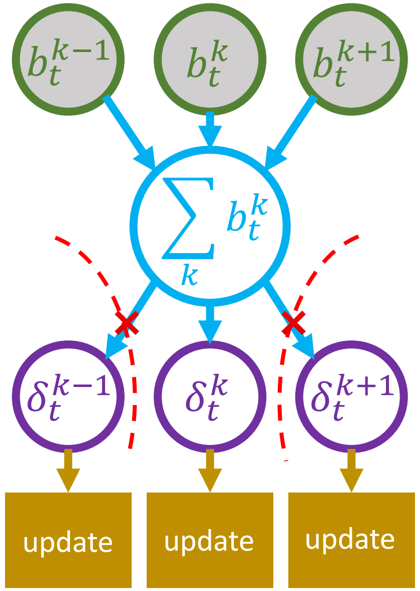
Therefore, single-step temporal difference algorithms are modular, when the trajectories are acyclic. See paper for more details.

Credit Assignment in RL: Summary

shaded nodes = conditioned on



policy gradient

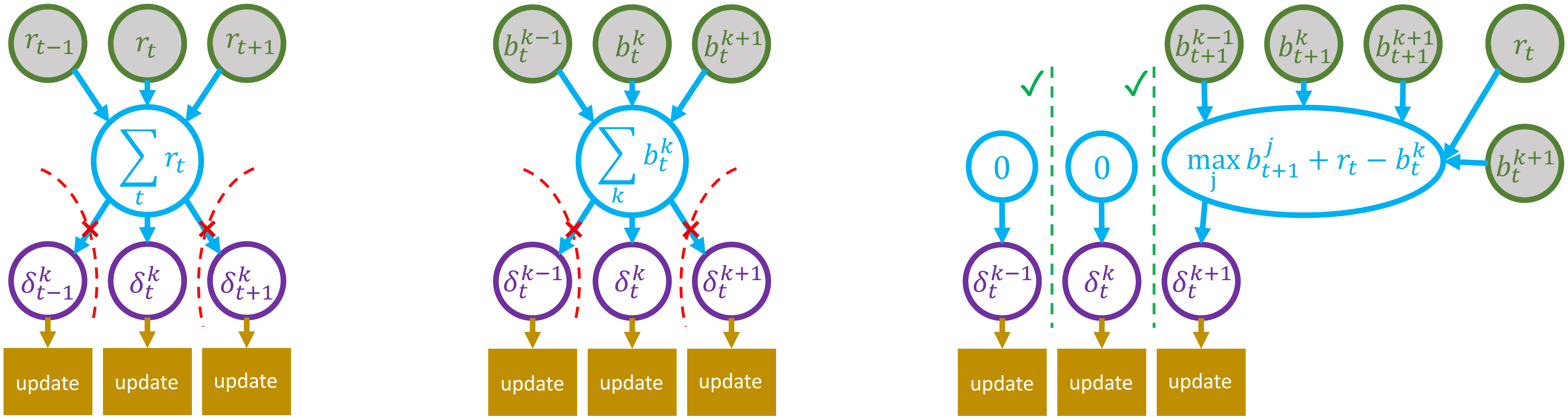


n-step temporal difference

Again, the gradients for policy gradient and n-step temporal difference methods are not d-separated,

Credit Assignment in RL: Summary

shaded nodes = conditioned on



policy gradient

n-step temporal difference

single-step temporal difference

whereas the gradients for single-step temporal difference methods are, in generic cases.

Hypothesis

Modularity → more efficient transfer

Expressing the hypothesis precisely

Testing the hypothesis

To recap, we want to test the hypothesis of whether modularity improves transfer.

Modularity for Dynamic Systems

Independent Credit Assignment

Modularity of RL Algorithms

Hypothesis

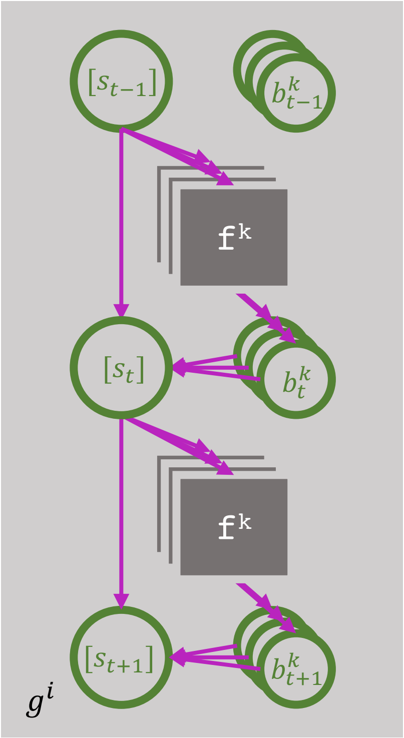
Modularity → more efficient transfer

Expressing the hypothesis precisely

Theoretical question: Which reinforcement learning algorithms produce independent gradients?

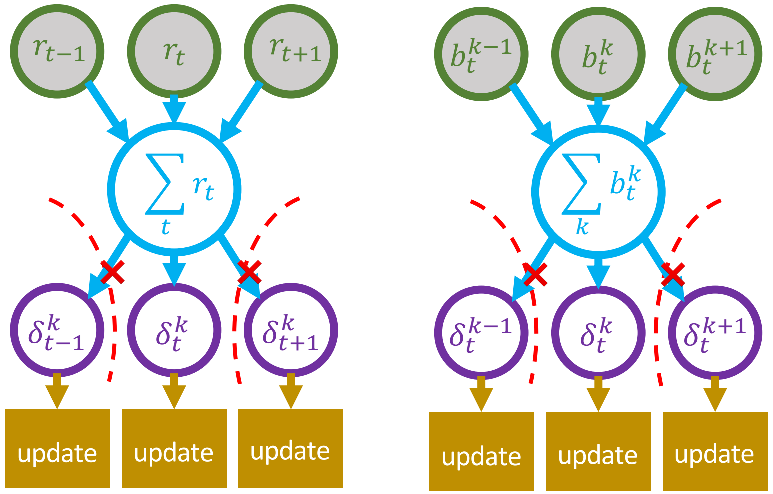
Testing the hypothesis

To do that, we need to determine which reinforcement learning algorithms are indeed modular.



Theoretical question: Which reinforcement learning algorithms produce independent gradients?

To answer this question, we represented reinforcement learning algorithms as causal graphs.

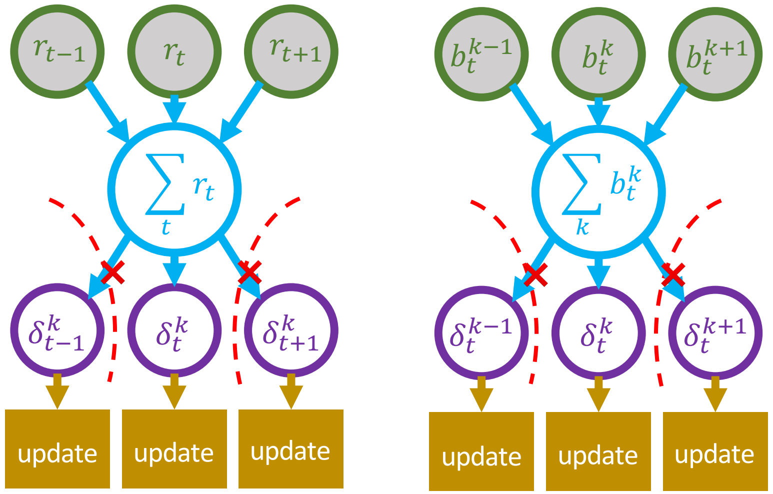


Theoretical question: Which reinforcement learning algorithms produce independent gradients?

policy gradients ✗

n-step temporal difference algorithms ✗

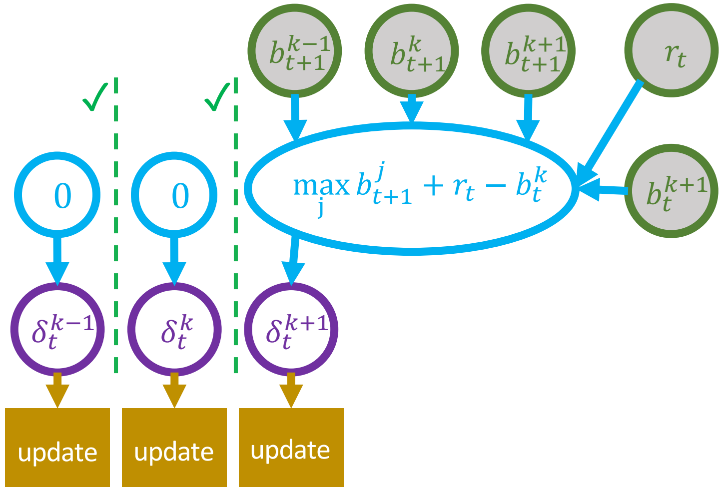
We identified that all policy gradient methods and n-step temporal difference methods do not produce independent gradients



Theoretical question: Which reinforcement learning algorithms produce independent gradients?

- policy gradients ❌
- n-step temporal difference algorithms ❌

because the causal structure of their credit assignment mechanisms contain a shared hidden variable.



Theoretical question: Which reinforcement learning algorithms produce independent gradients?

- policy gradients ❌
- n-step temporal difference algorithms ❌
- single-step temporal difference algorithms ✓

In contrast, in generic cases single-step temporal difference methods do produce independent gradients, and thus are modular.

Hypothesis

Modularity → more efficient transfer

Expressing the hypothesis precisely

Testing the hypothesis

Theoretical question: Which reinforcement learning algorithms produce independent gradients?

- policy gradients ✗
- n-step temporal difference algorithms ✗
- single-step temporal difference algorithms ✓

Empirical question: Does modularity improve transfer efficiency?

Having identified which RL algorithms are modular and which are not,

Hypothesis

Modularity → more efficient transfer

Expressing the hypothesis precisely

Testing the hypothesis

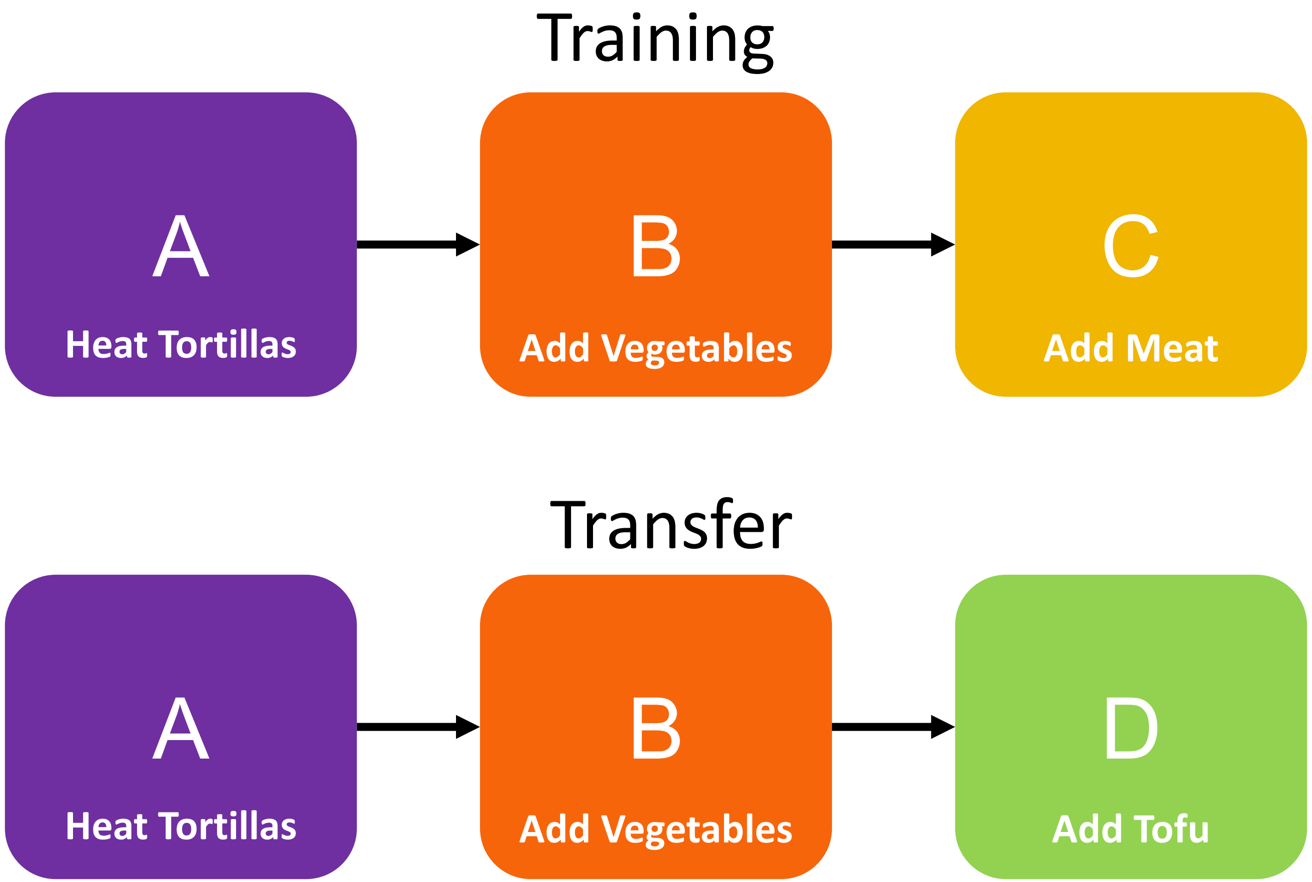
Theoretical question: Which reinforcement learning algorithms produce independent gradients?

- policy gradients ✗
- n-step temporal difference algorithms ✗
- single-step temporal difference algorithms ✓

Empirical question: Does modularity improve transfer efficiency?

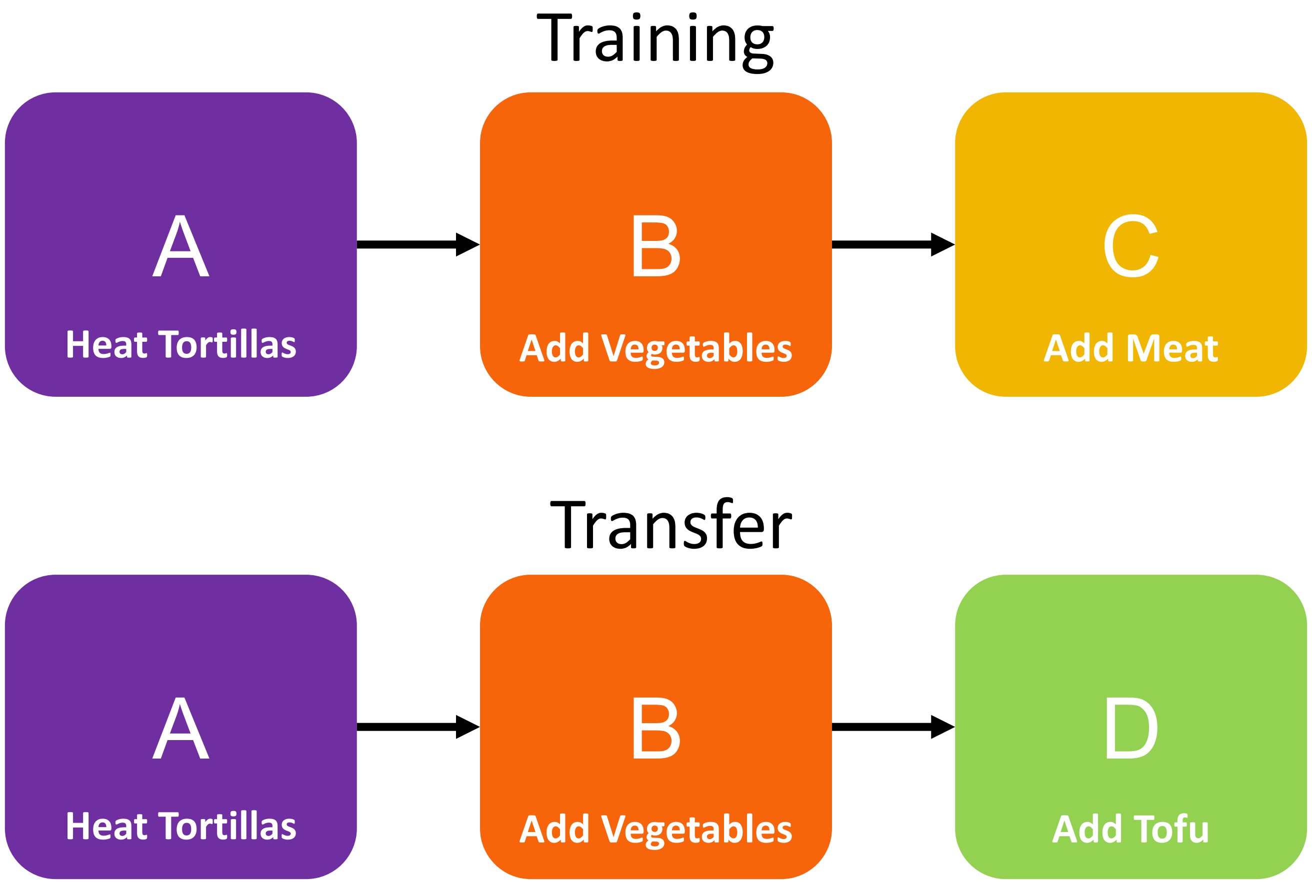
we now are in the position to test whether modularity improves transfer efficiency.

Transfer via Sparse Modifications



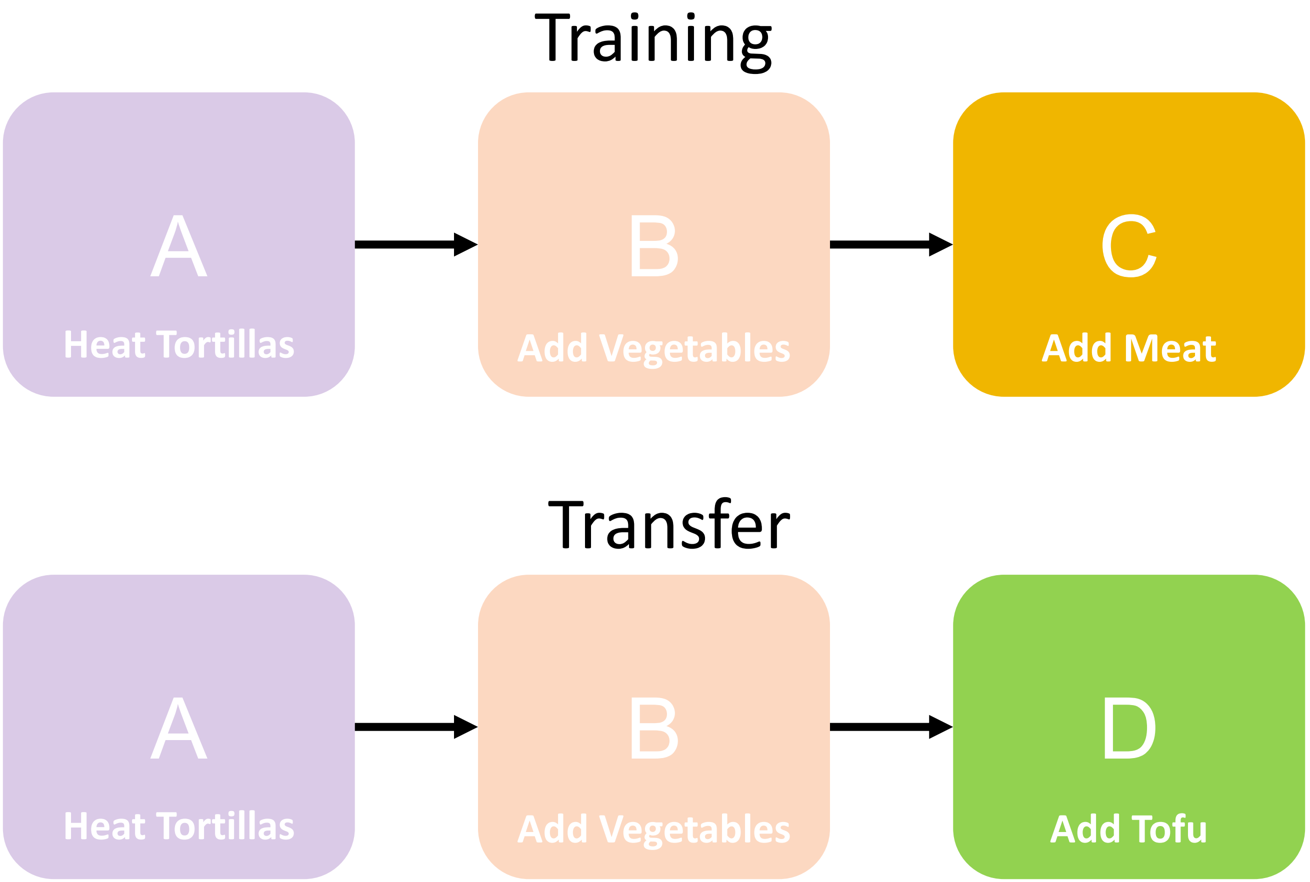
As illustrated by our motivational example, we are interested in transfer problems that require only sparse changes

Transfer via Sparse Modifications



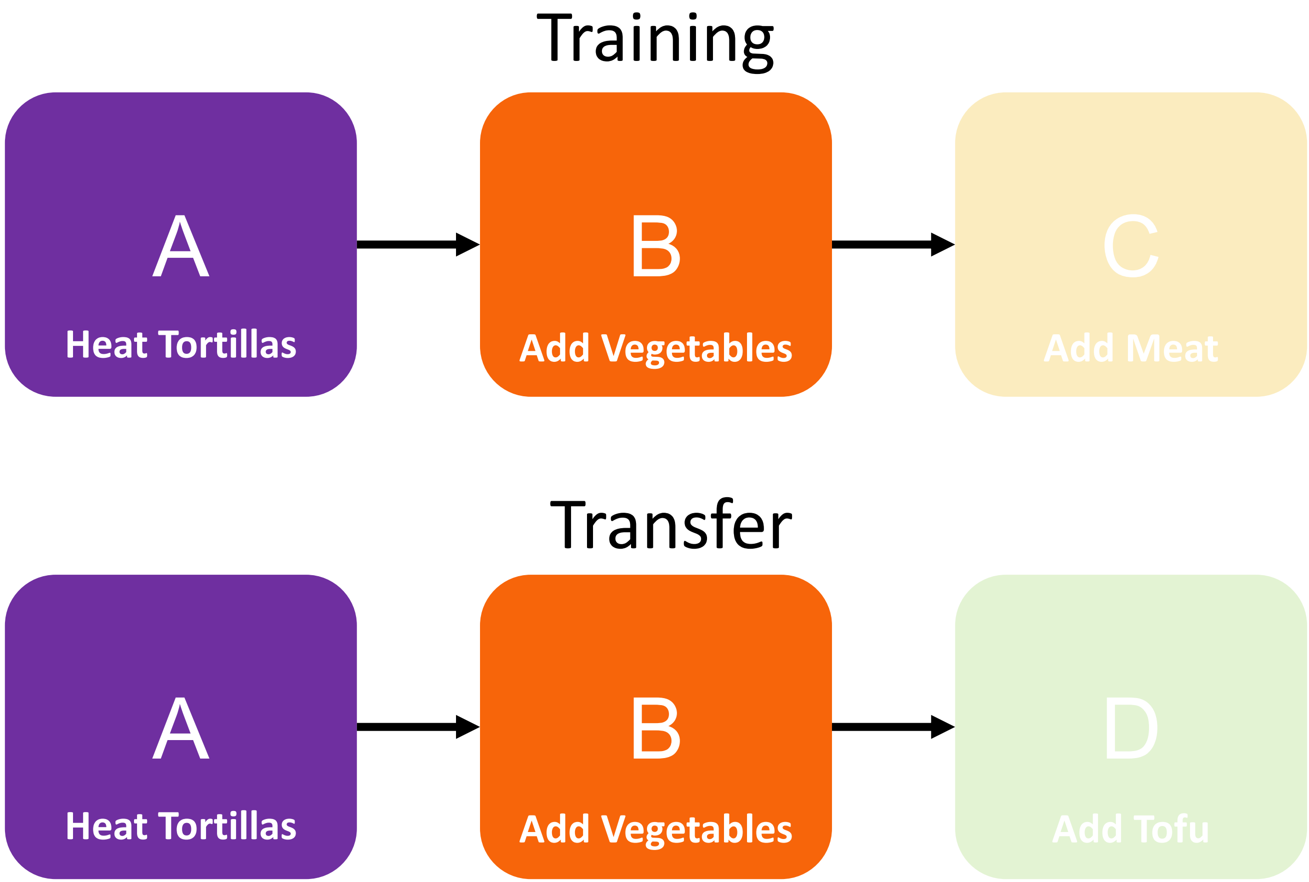
to a sequence of previously optimal decisions, because that tests to what extent an algorithm can separate

Transfer via Sparse Modifications



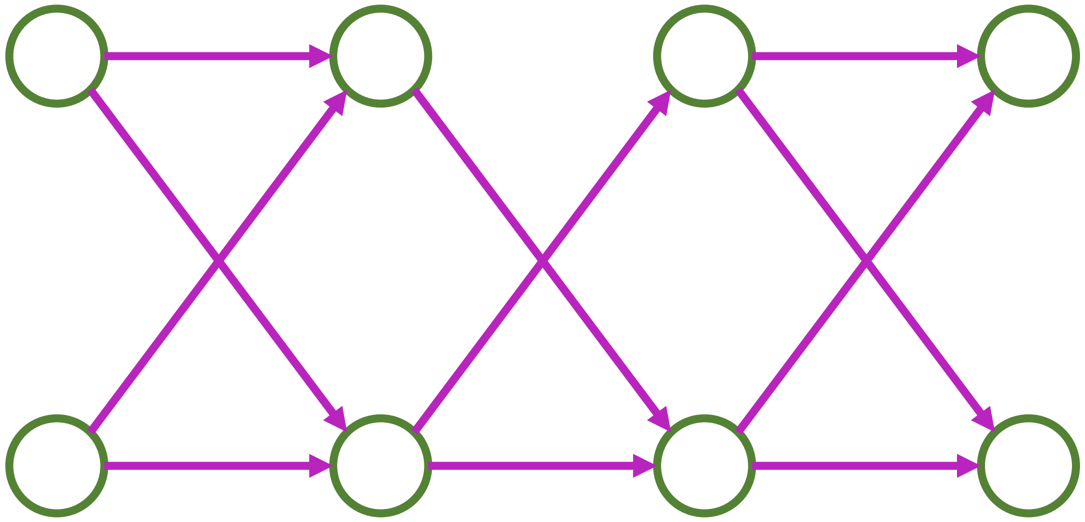
what needs to be modified

Transfer via Sparse Modifications



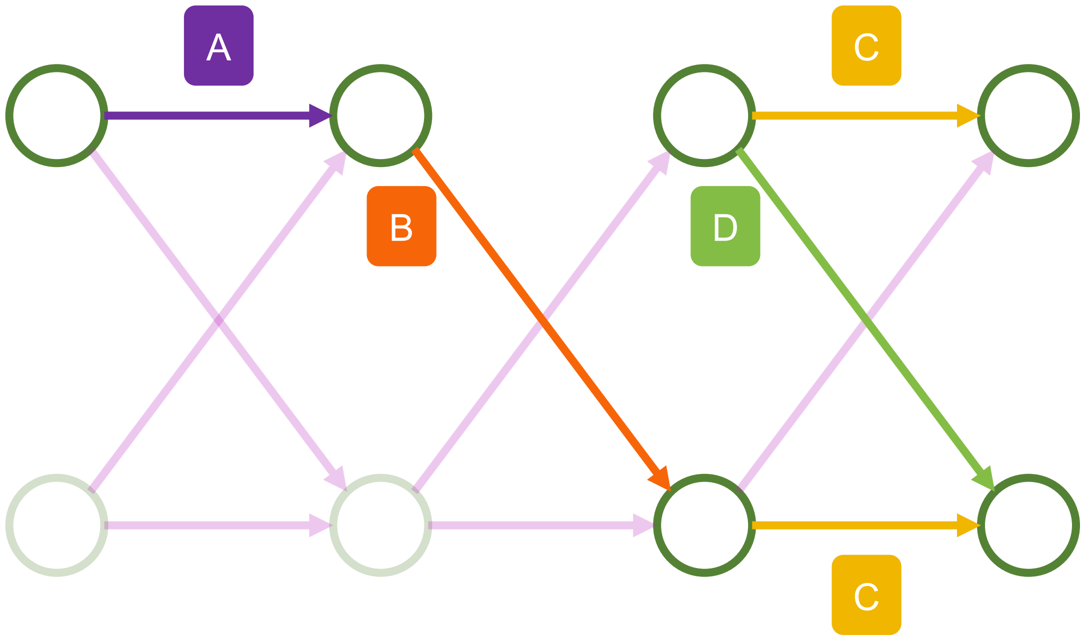
from what does not need to be modified.

Experimental Setup: MDP



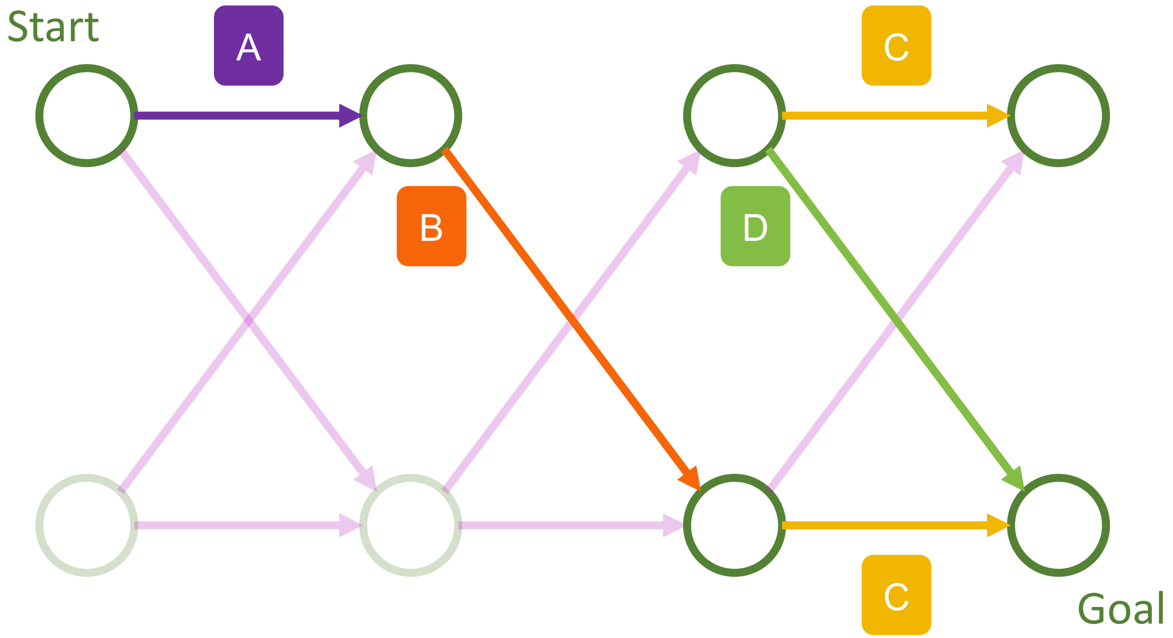
We can represent this kind of transfer problem as a simple MDP. Circles represent states. Edges represent state transitions.

Experimental Setup: MDP



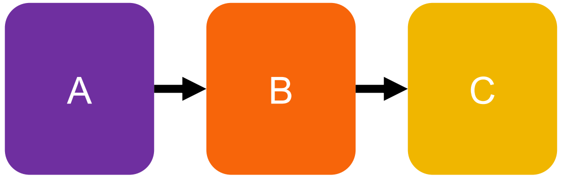
Here we label some transitions with the actions that cause them.

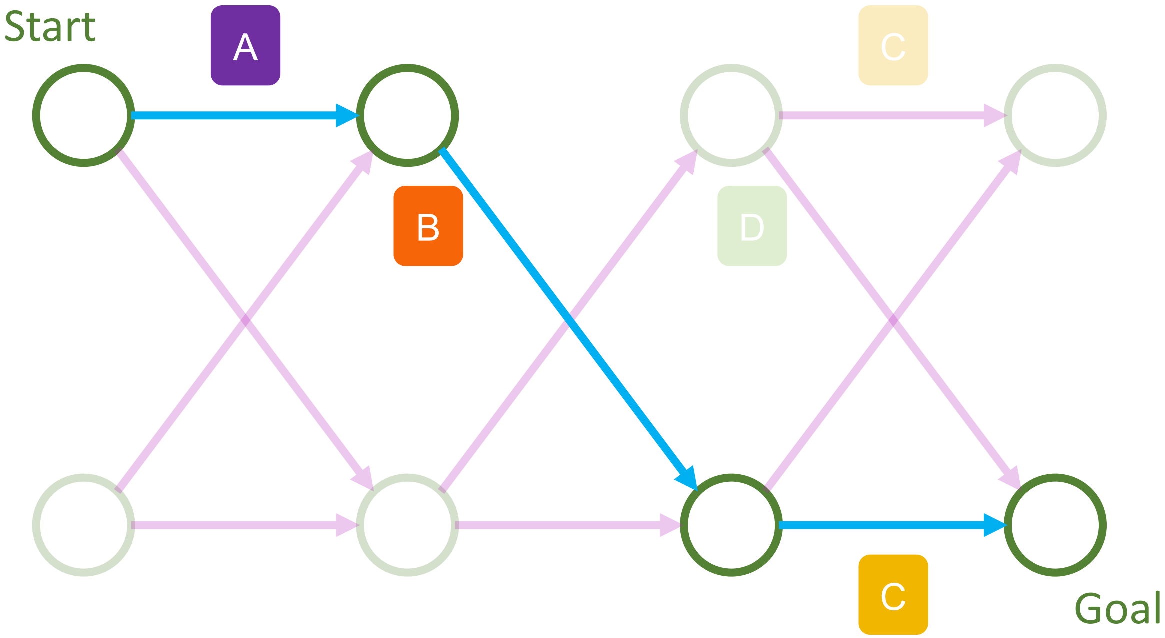
Experimental Setup: MDP



We have a start state and goal state.

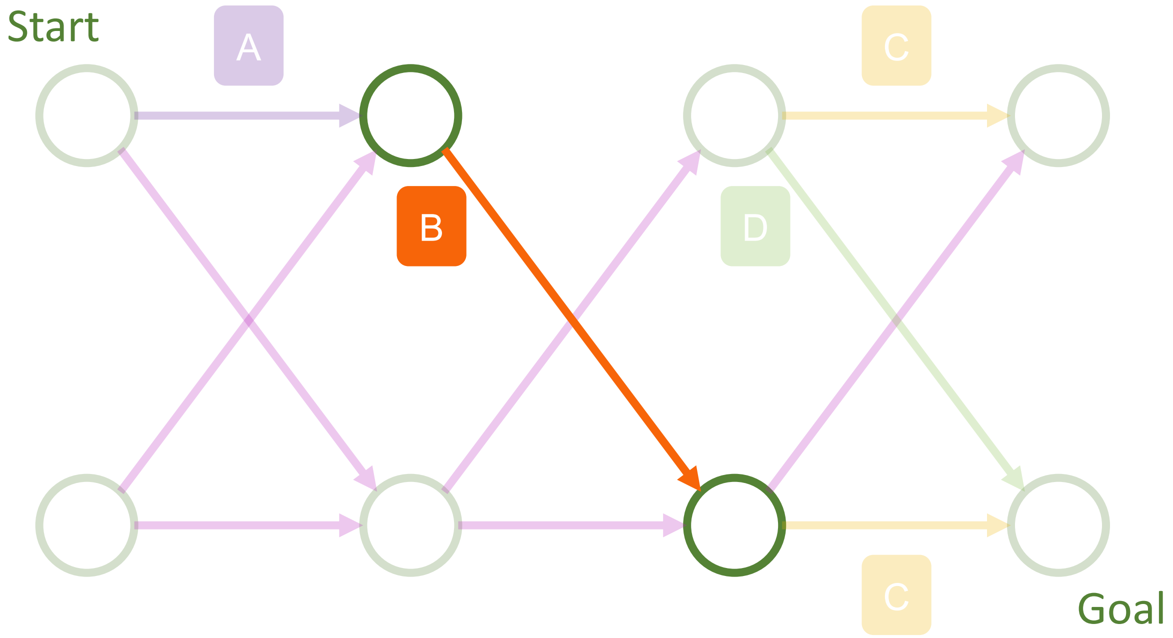
Experimental Setup: MDP





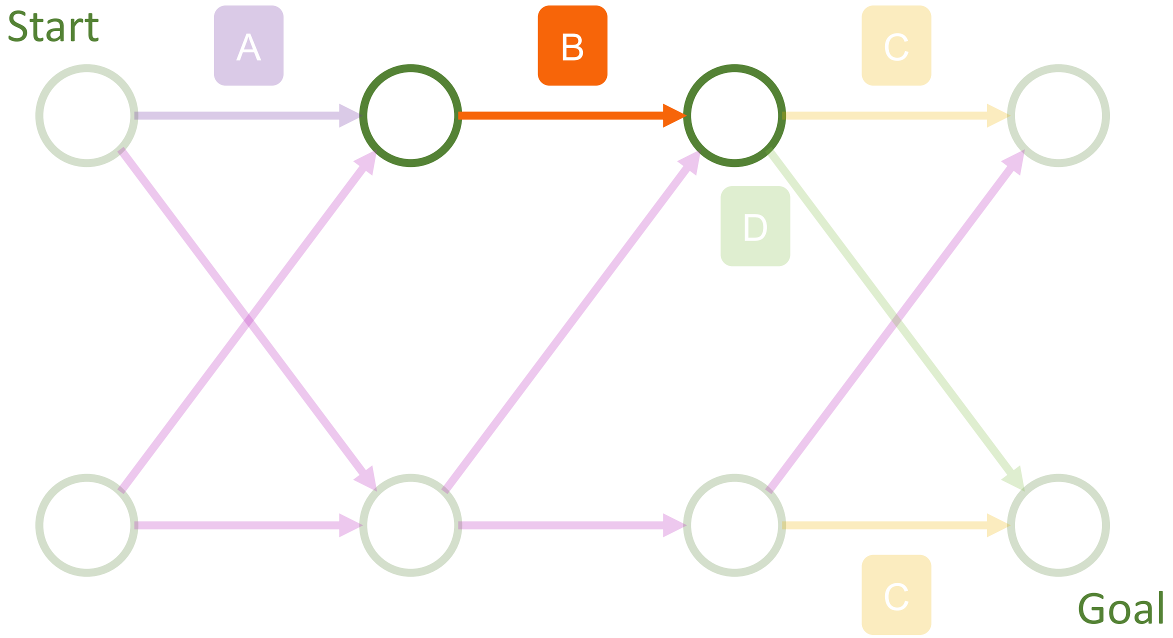
During training, the optimal decision sequence is A, B then C.

Experimental Setup: MDP



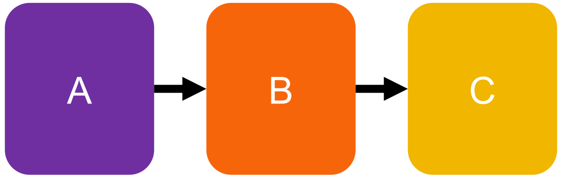
To generate the transfer task from the training task,

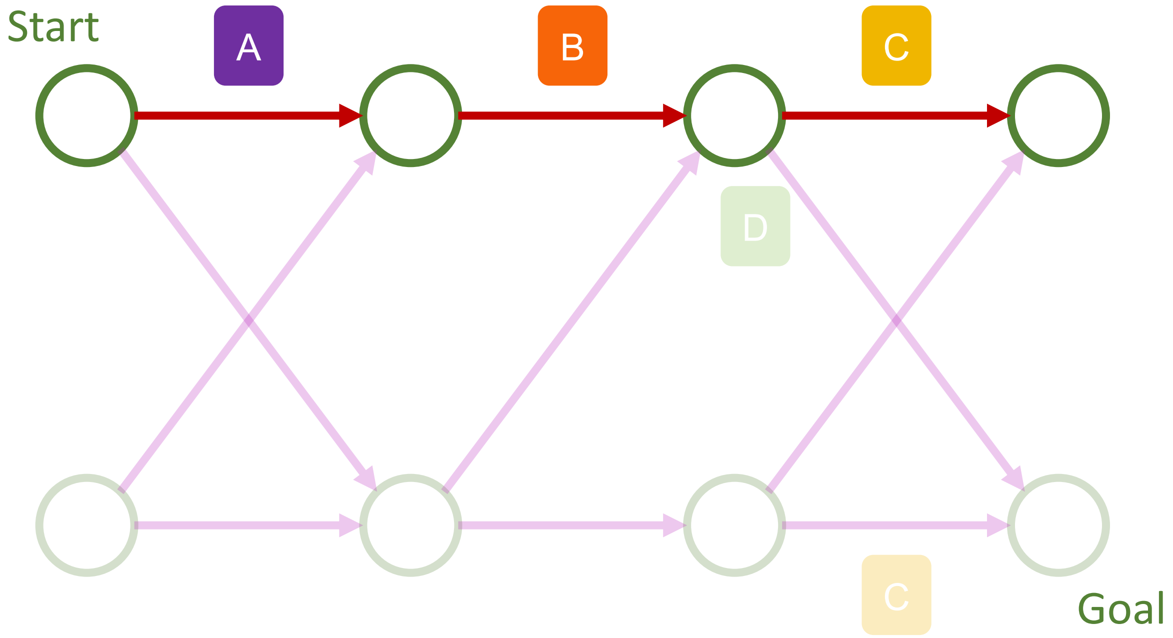
Experimental Setup: MDP



we modify the transition that B corresponds to.

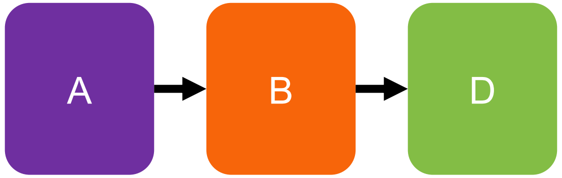
Experimental Setup: MDP

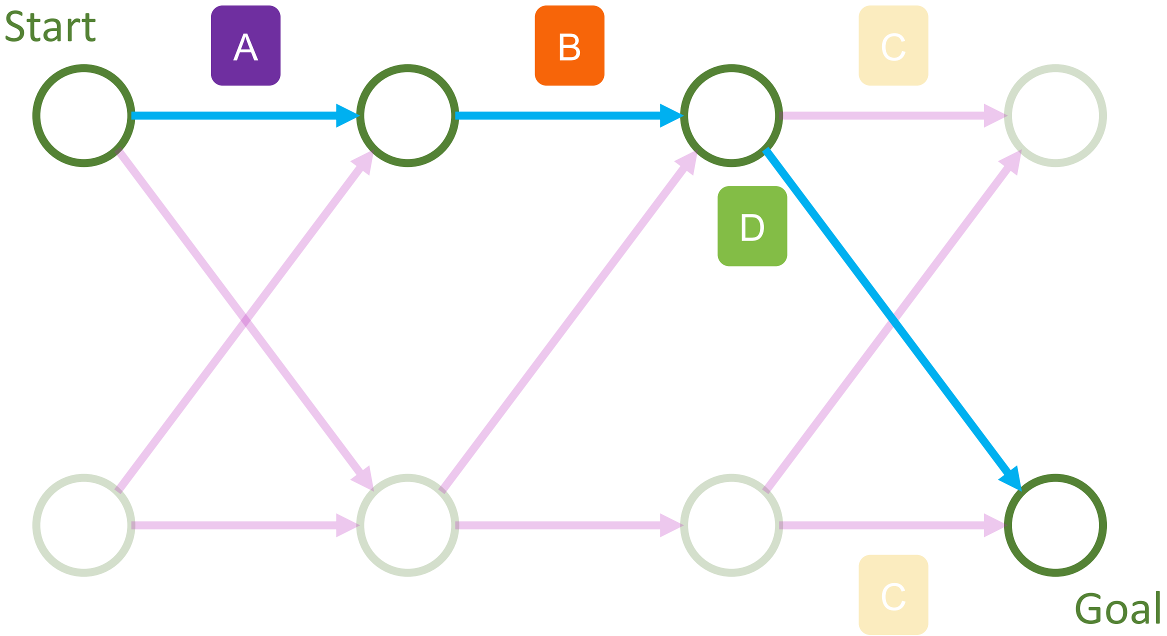




Now the original sequence of decisions is suboptimal,

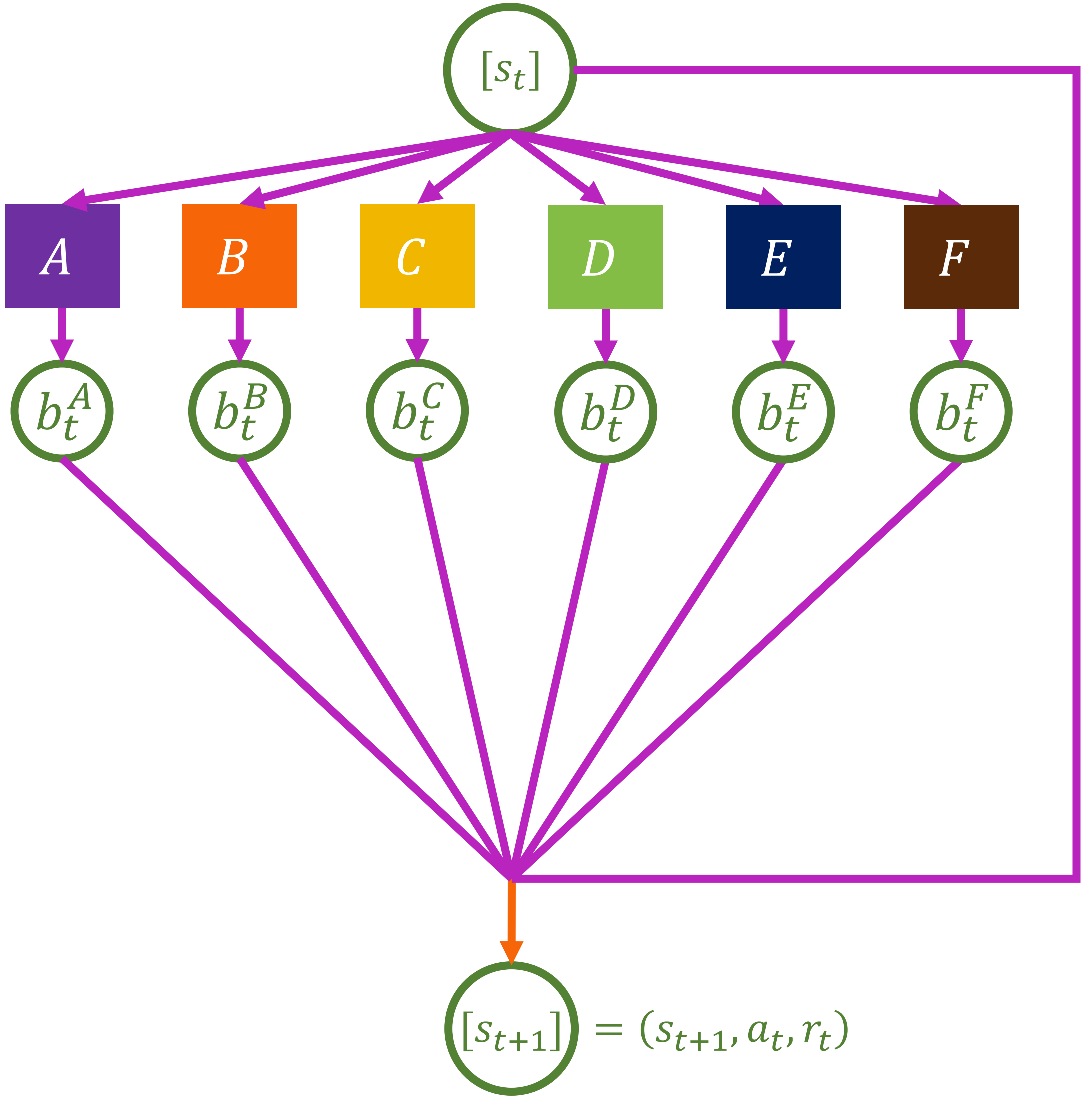
Experimental Setup: MDP





but swapping action C for action D, in this case, is now the optimal thing to do.

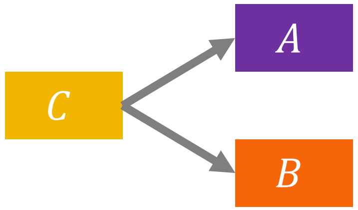
Experimental Setup: MDP

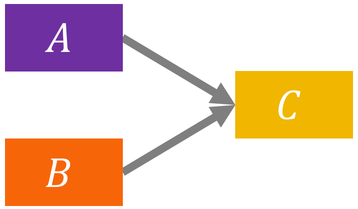


Our MDP has six possible values for the action variable, so we will have six decision mechanisms.

Various Topologies of Transfer Problems

Training Task





Linear Chain

single task, 3 time-steps

Common Ancestor

multi-task, 2 time-steps each

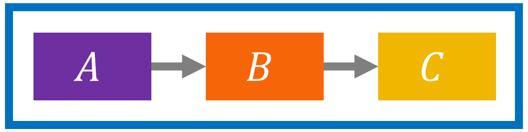
Common Descendant

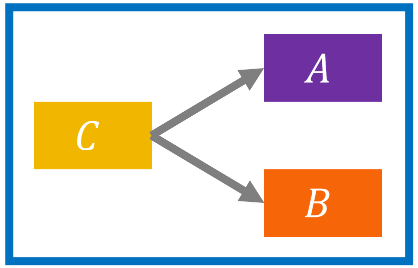
multi-task, 2 time-steps each

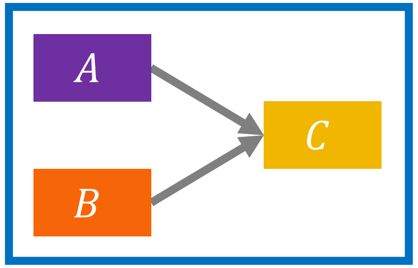
Similar to how analysis of d-separation is conducted with triplets of nodes,

Various Topologies of Transfer Problems

Training Task







Linear Chain

single task, 3 time-steps

Common Ancestor

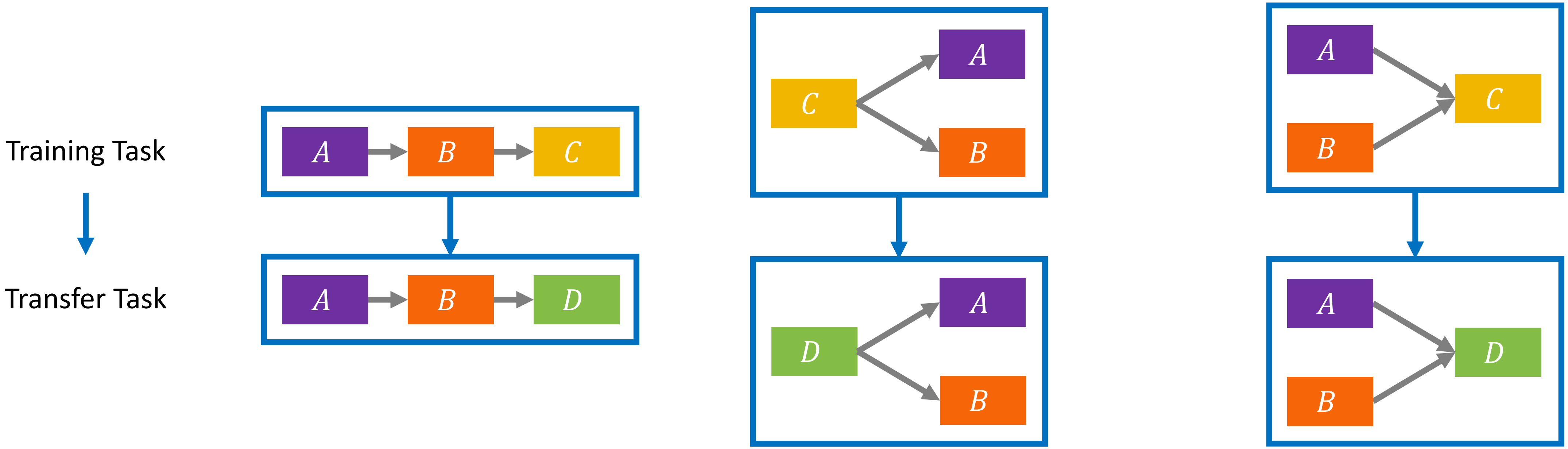
multi-task, 2 time-steps each

Common Descendant

multi-task, 2 time-steps each

we enumerated all possible topologies of triplets of decisions.

Various Topologies of Transfer Problems



Linear Chain

single task, 3 time-steps

Common Ancestor

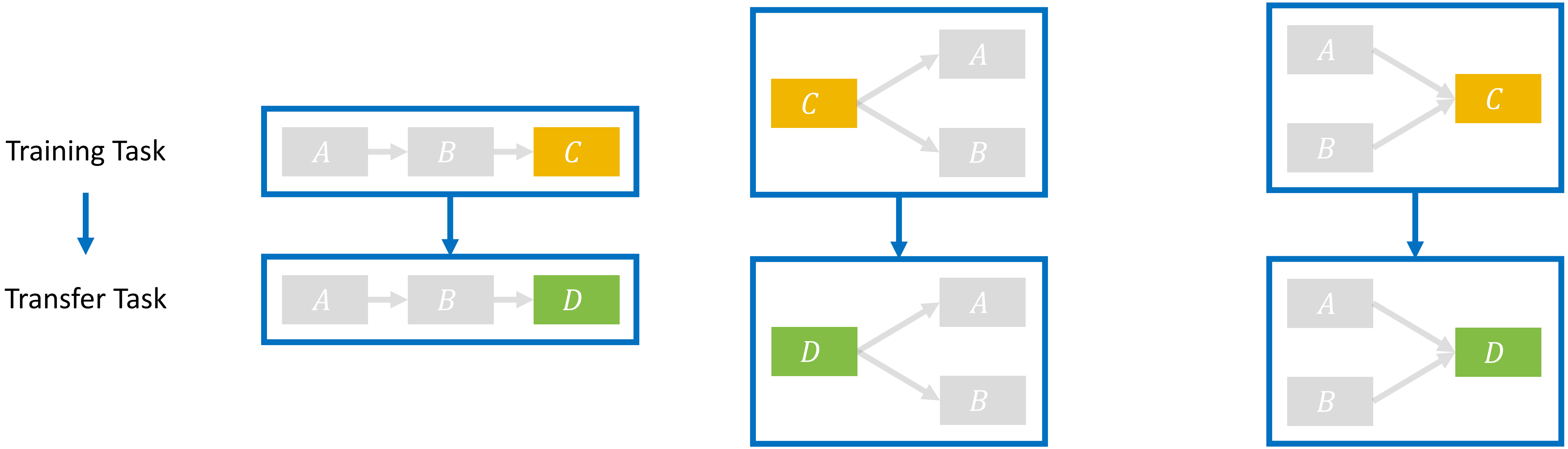
multi-task, 2 time-steps each

Common Descendant

multi-task, 2 time-steps each

And for each topology we generated three transfer tasks

Various Topologies of Transfer Problems



Linear Chain

single task, 3 time-steps

Common Ancestor

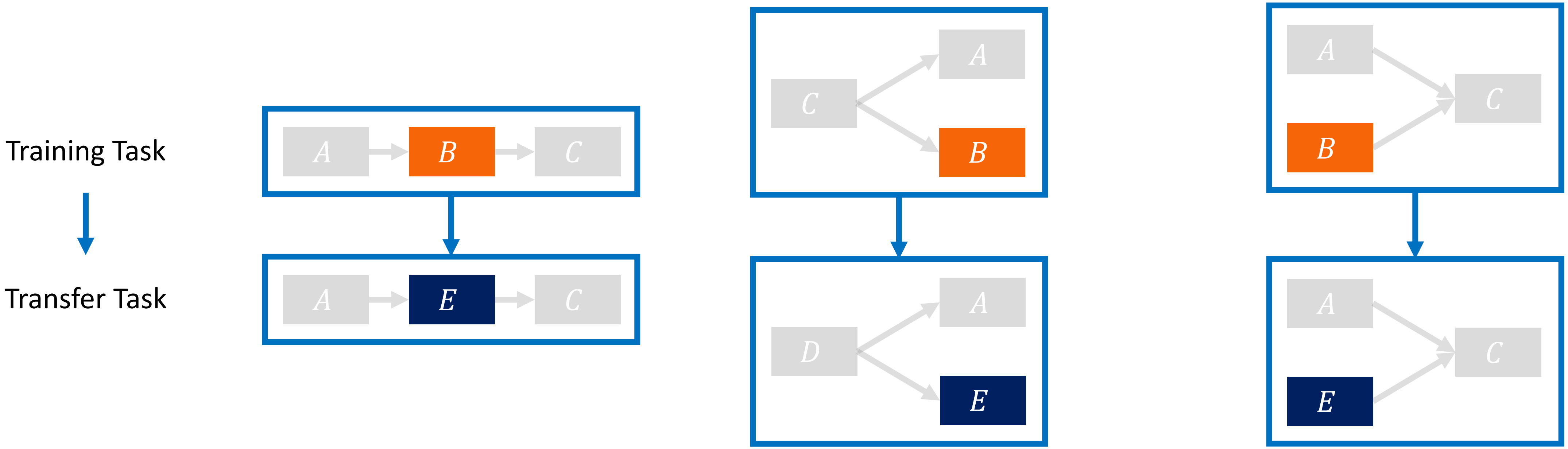
multi-task, 2 time-steps each

Common Descendant

multi-task, 2 time-steps each

by enumerating all the ways of making an isolated change to the optimal decision sequences.

Various Topologies of Transfer Problems



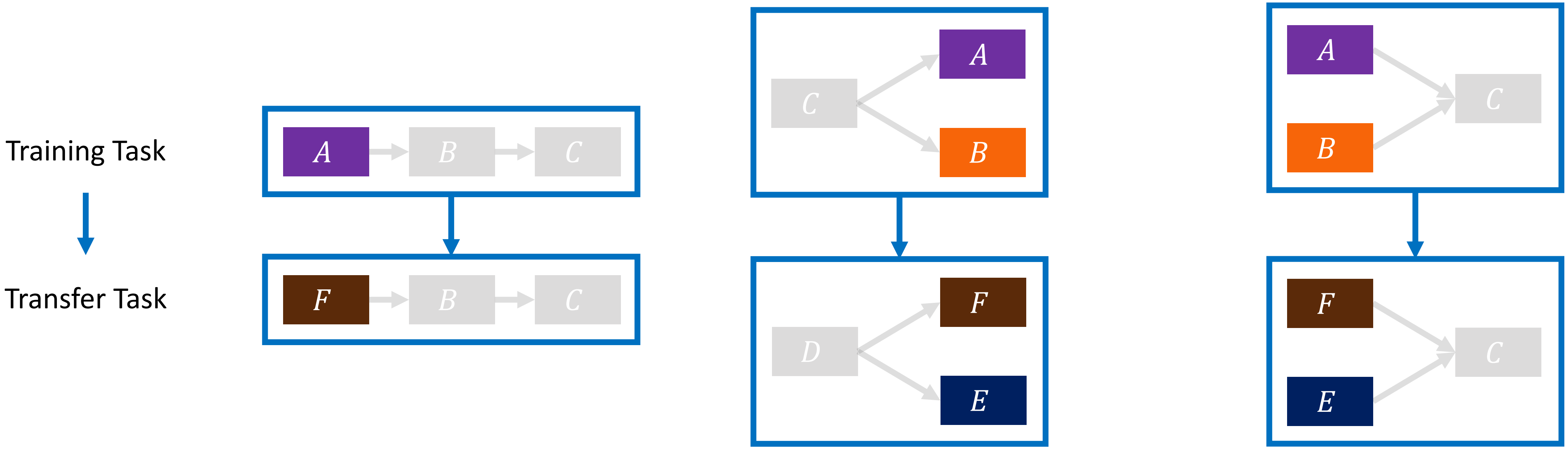
Linear Chain — single task, 3 time-steps

Common Ancestor — multi-task, 2 time-steps each

Common Descendant — multi-task, 2 time-steps each

by enumerating all the ways of making an isolated change to the optimal decision sequences.

Various Topologies of Transfer Problems



Linear Chain

single task, 3 time-steps

Common Ancestor

multi-task, 2 time-steps each

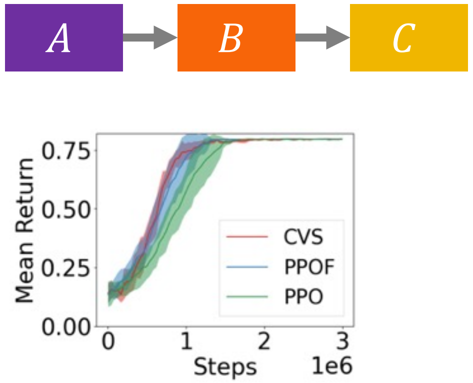
Common Descendant

multi-task, 2 time-steps each

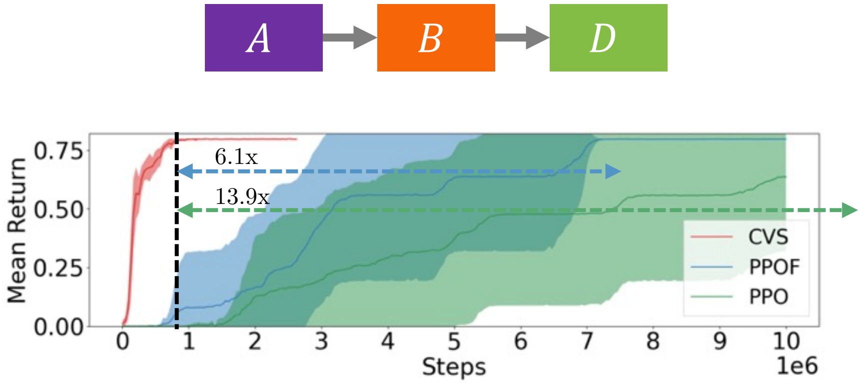
by enumerating all the ways of making an isolated change to the optimal decision sequences.

Empirical Results

Training Task



Transfer Task



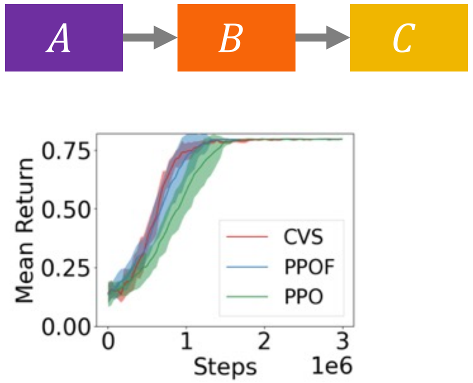
This includes our motivating example, where the last action should change from C to D. We compare three algorithms:

Empirical Results

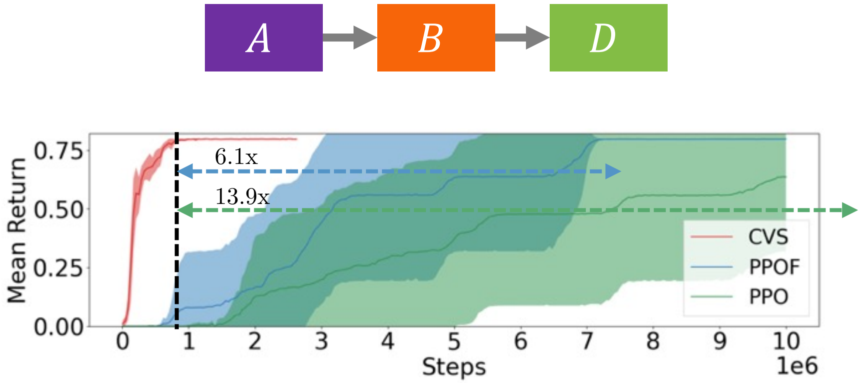
CVS
Independent gradients? ✓
Network factorization? ✓

Chang, Kaushik, Weinberg, Griffiths, Levine (ICML 2020)

Training Task



Transfer Task



the Cloned Vickrey Society, or CVS, which is a modular algorithm,

Empirical Results

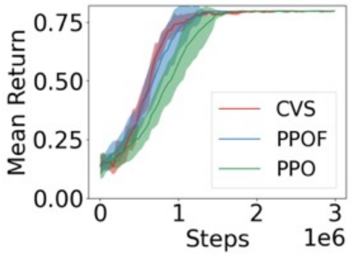
CVS
Independent gradients? ✓
Network factorization? ✓

Chang, Kaushik, Weinberg, Griffiths, Levine (ICML 2020)

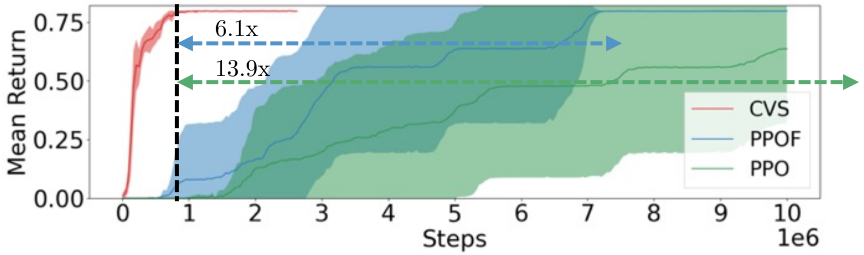
PPO
Independent gradients? ✗
Network factorization? ✗

Schulman et al. (arXiv 2017)

Training Task



Transfer Task



PPO, which is not modular because the policy is not factorized along the actions and its gradients are not independent,

Empirical Results

CVS
Independent gradients? ✓
Network factorization? ✓

Chang, Kaushik, Weinberg, Griffiths, Levine (ICML 2020)

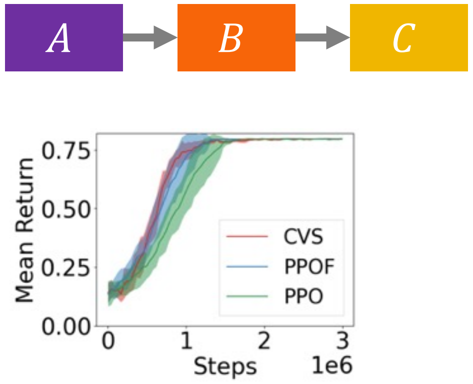
PPOF
Independent gradients? ✗
Network factorization? ✓

PPO with a separate network for each action

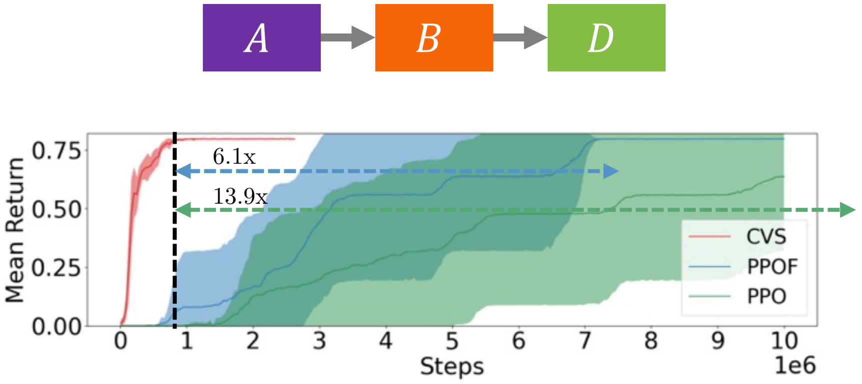
PPO
Independent gradients? ✗
Network factorization? ✗

Schulman et al. (arXiv 2017)

Training Task



Transfer Task



and PPOF, a variant of PPO with a policy network that is factorized over the action variable.

Empirical Results

CVS
Independent gradients? ✓
Network factorization? ✓

Chang, Kaushik, Weinberg, Griffiths, Levine (ICML 2020)

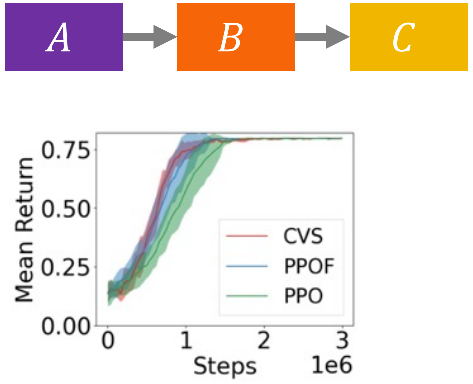
PPOF
Independent gradients? ✗
Network factorization? ✓

PPO with a separate network for each action

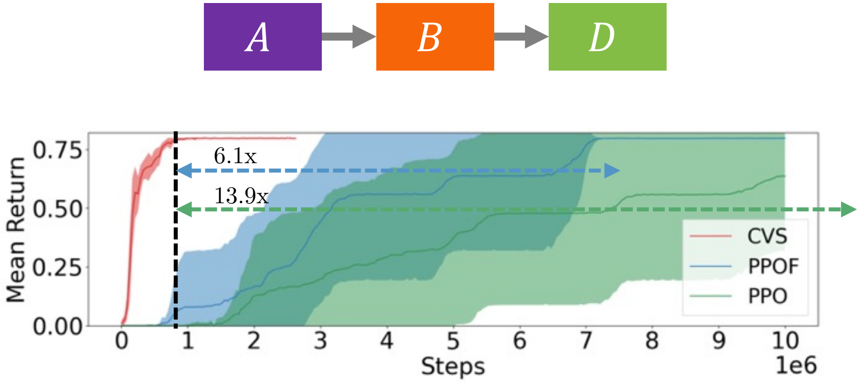
PPO
Independent gradients? ✗
Network factorization? ✗

Schulman et al. (arXiv 2017)

Training Task



Transfer Task



PPOF is also not modular because its gradients are not independent.

Empirical Results

CVS
Independent gradients? ✓
Network factorization? ✓

Chang, Kaushik, Weinberg, Griffiths, Levine (ICML 2020)

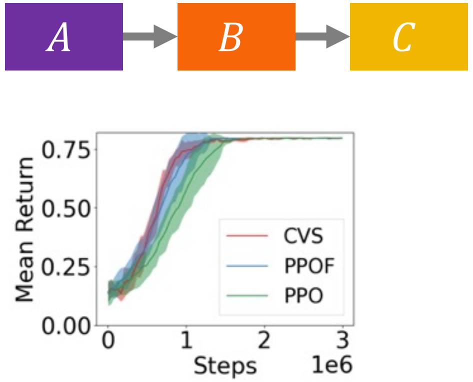
PPOF
Independent gradients? ✗
Network factorization? ✓

PPO with a separate network for each action

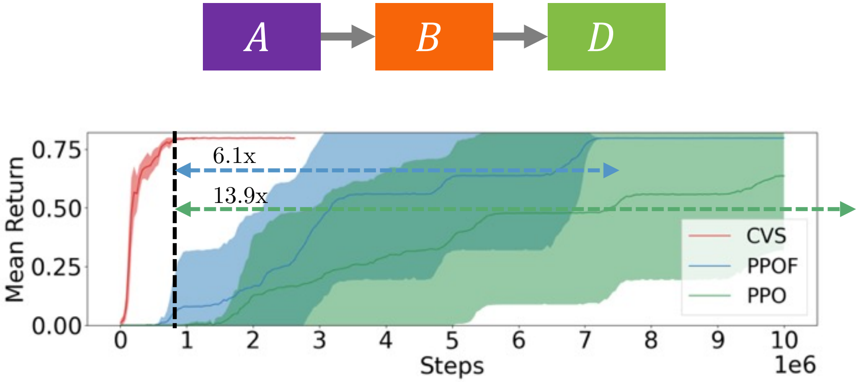
PPO
Independent gradients? ✗
Network factorization? ✗

Schulman et al. (arXiv 2017)

Training Task



Transfer Task



We chose PPOF as a baseline because while it is intuitive how network factorization contributes to modularity,

Empirical Results

CVS
Independent gradients? ✓
Network factorization? ✓

Chang, Kaushik, Weinberg, Griffiths, Levine (ICML 2020)

PPOF
Independent gradients? ✗
Network factorization? ✓

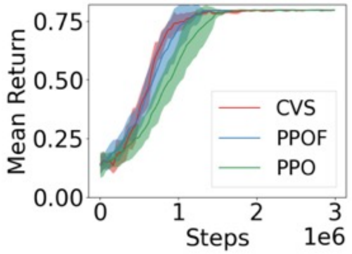
PPO with a separate network for each action

PPO
Independent gradients? ✗
Network factorization? ✗

Schulman et al. (arXiv 2017)

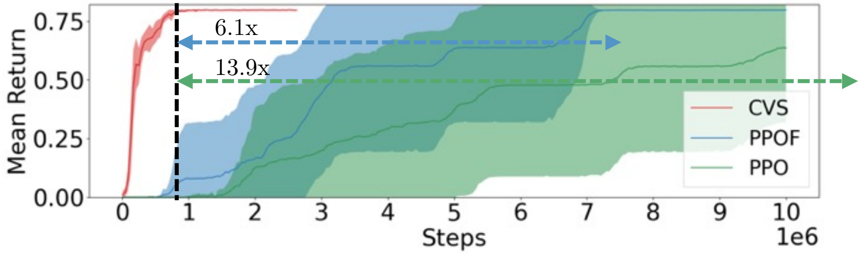
Training Task

$A \rightarrow B \rightarrow C$



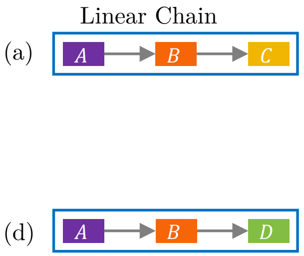
Transfer Task

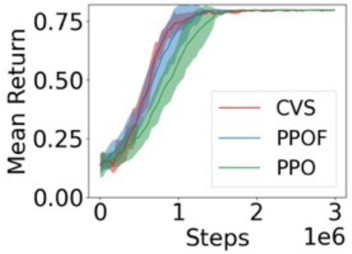
$A \rightarrow B \rightarrow D$

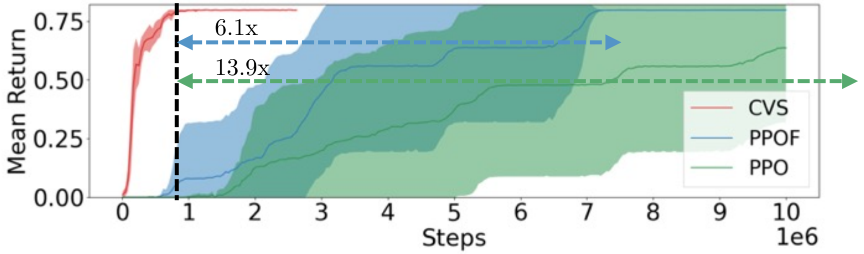


comparing CVS with PPOF specifically tests the role of independent gradients in enabling transfer efficiency.

Empirical Results

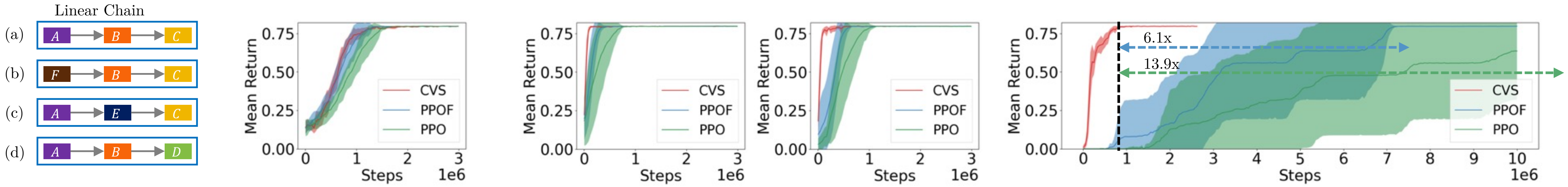






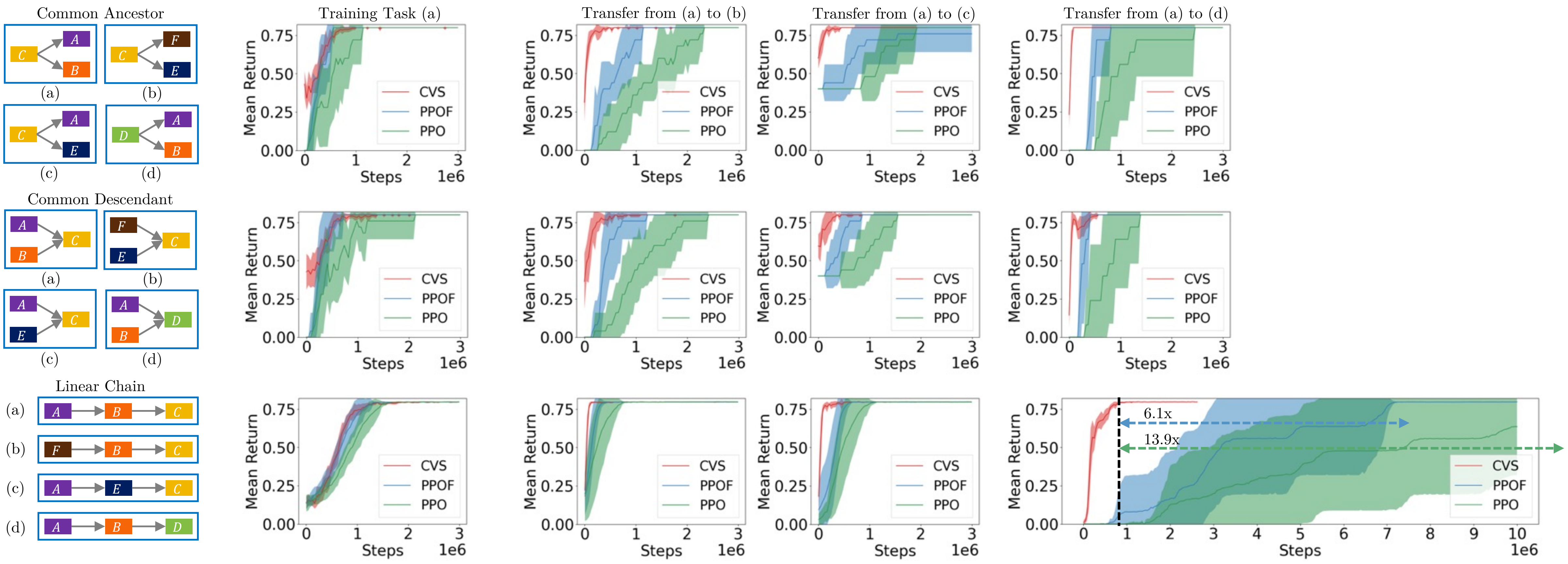
When we consider all nine transfer settings, across the board, CVS is consistently more sample efficient in the transfer task

Empirical Results



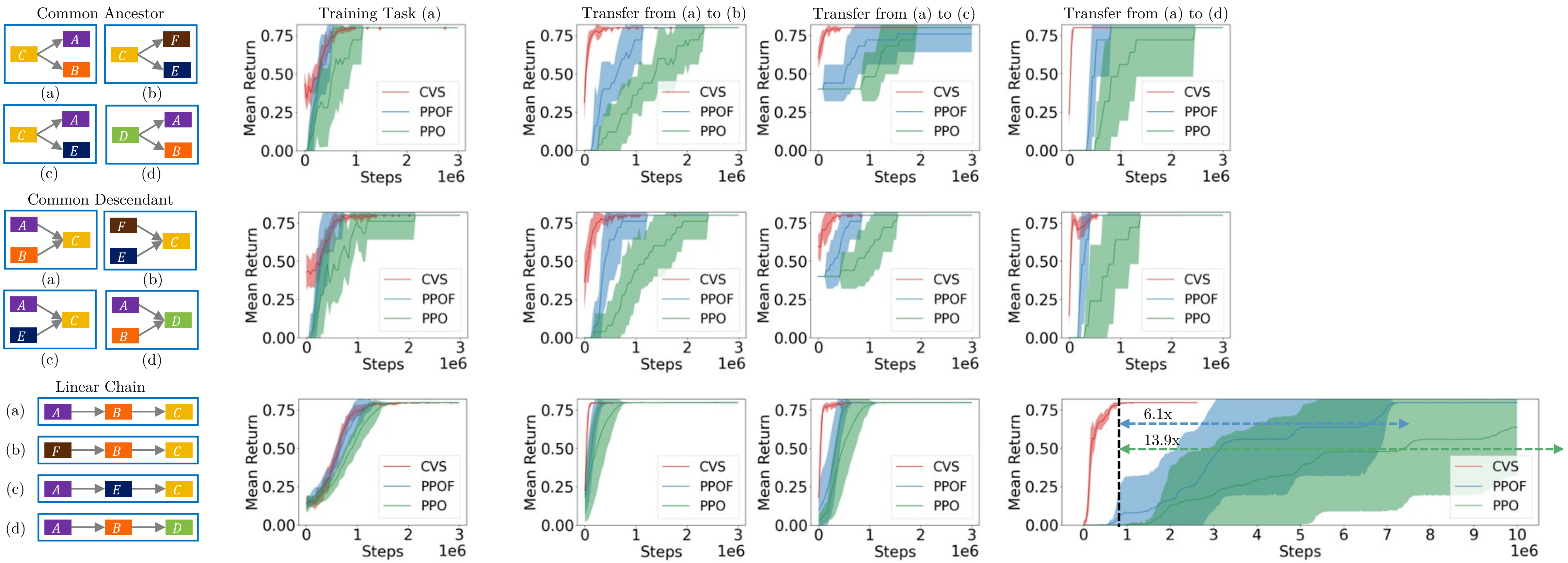
When we consider all nine transfer settings, across the board, CVS is consistently more sample efficient in the transfer task

Empirical Results



When we consider all nine transfer settings, across the board, CVS is consistently more sample efficient in the transfer task

Empirical Results

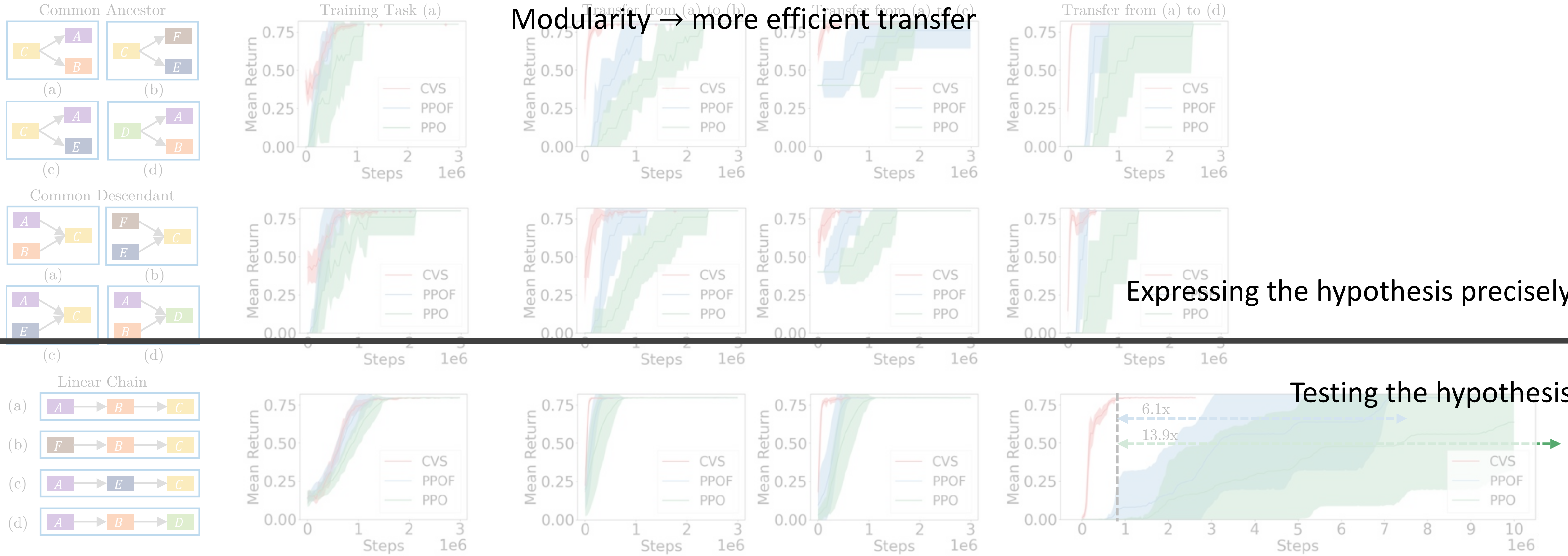


despite having comparable training efficiency in the training task.

Empirical Results

Hypothesis

Modularity → more efficient transfer



Expressing the hypothesis precisely

Testing the hypothesis

Thus, we have tested our hypothesis, and it survives the experimental test.

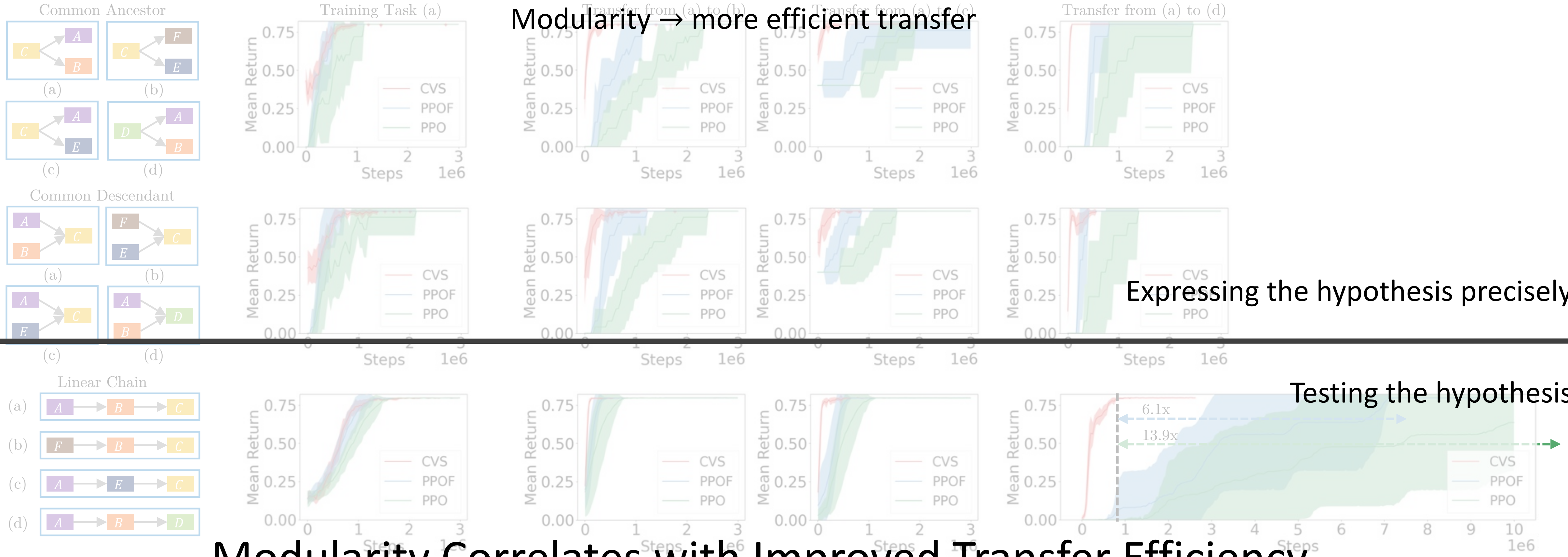
Empirical Results

Hypothesis

Modularity → more efficient transfer

Expressing the hypothesis precisely

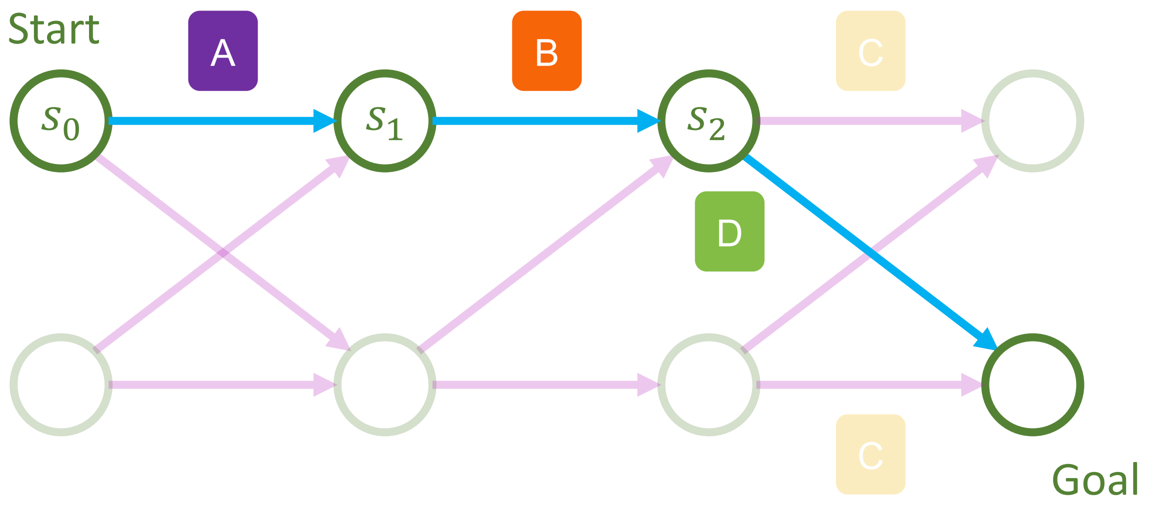
Testing the hypothesis



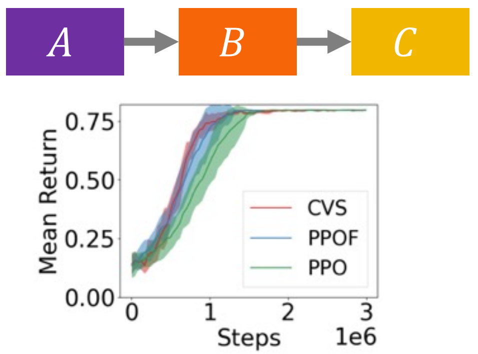
Modularity Correlates with Improved Transfer Efficiency.

Modularity does seem to correlate with improvements in transfer efficiency.

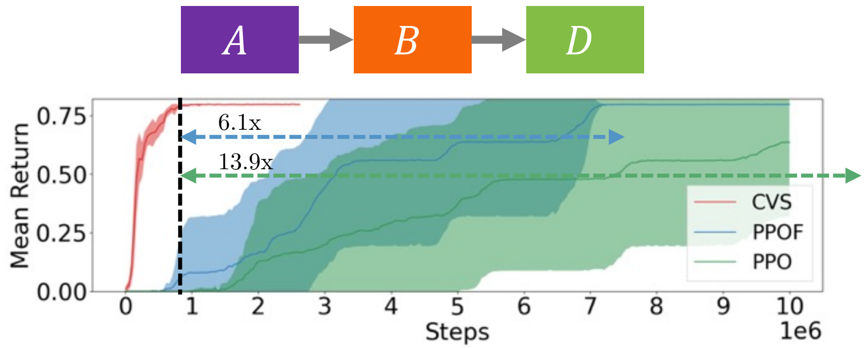
Analysis



Training Task

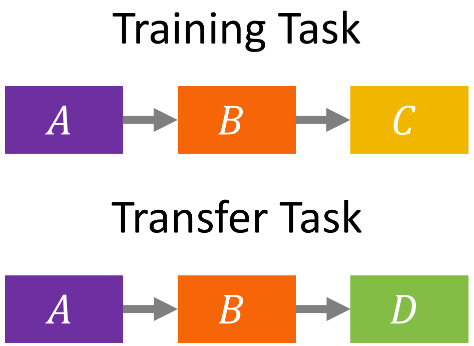


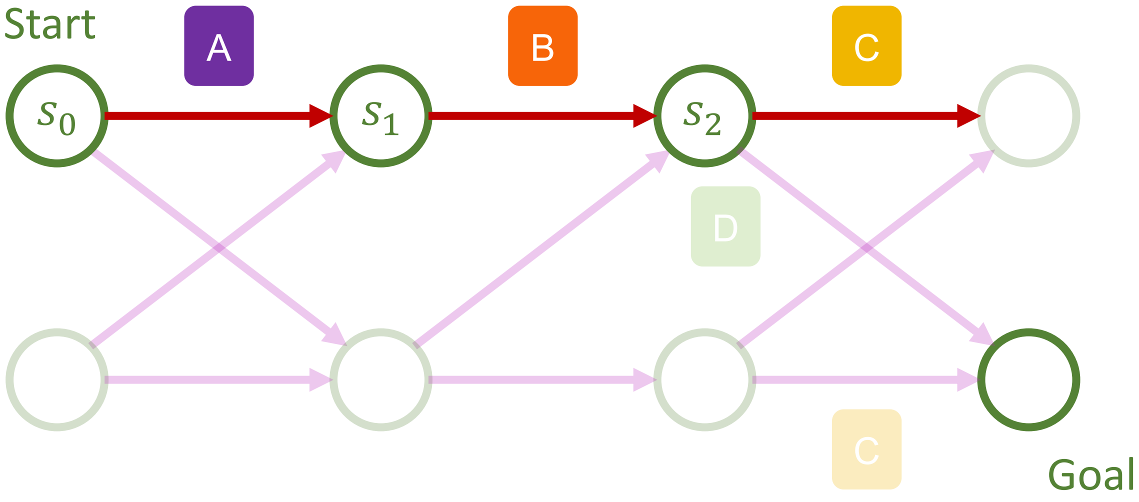
Transfer Task



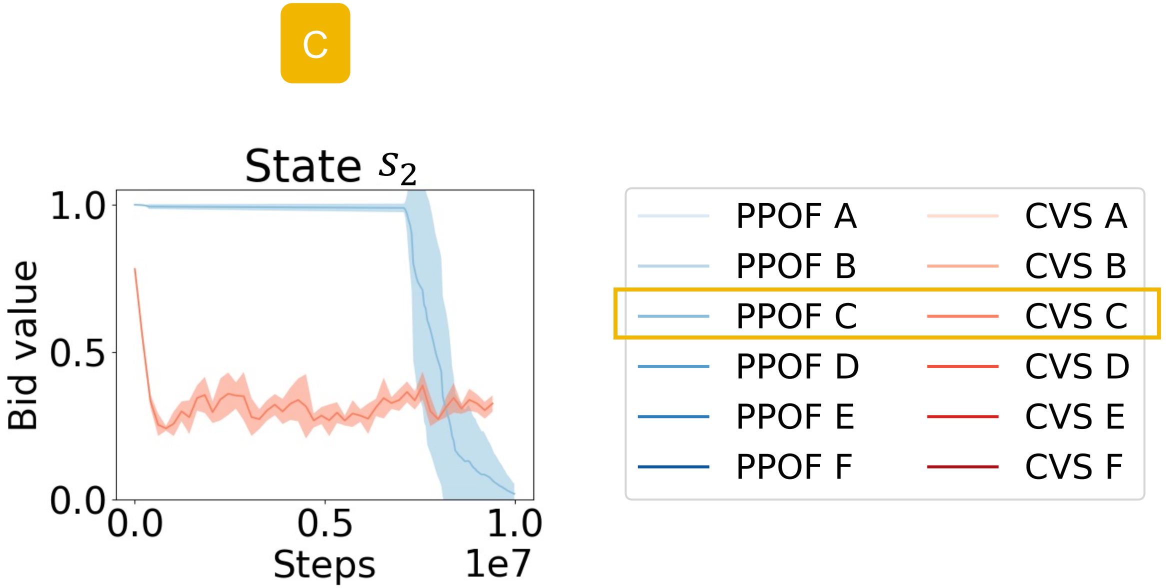
We dove deeper to find an explanation for why this hypothesis seems to hold.

Analysis





Bid values during transfer:

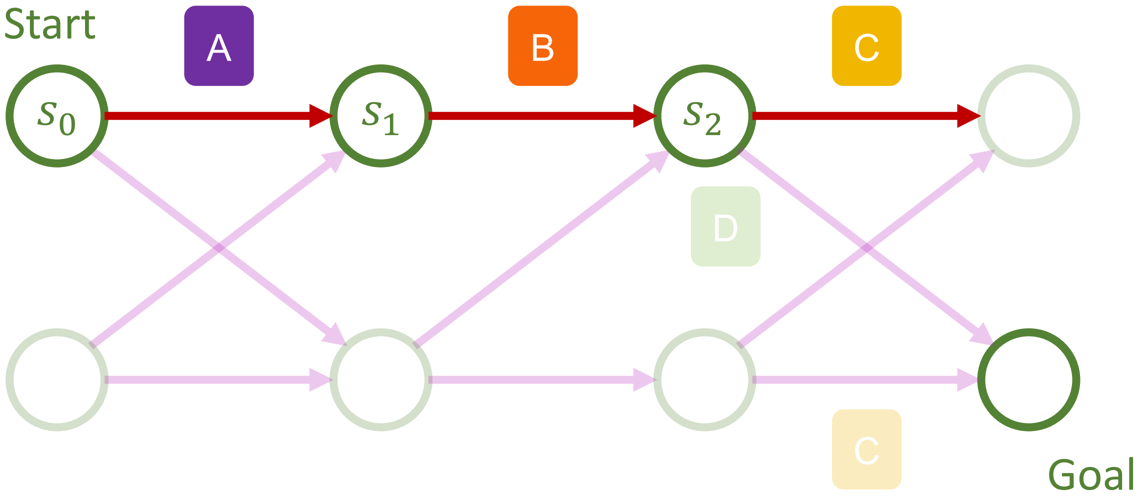


We measured the outputs of the decision mechanisms and observed how they changed between training and transfer.

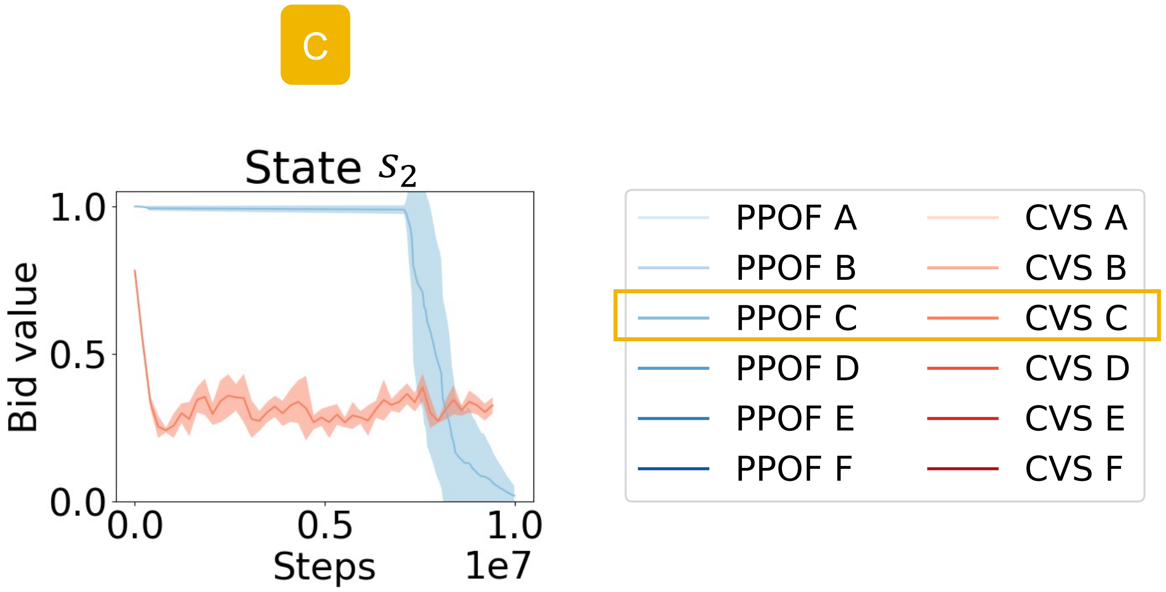
Analysis

Training Task

Transfer Task

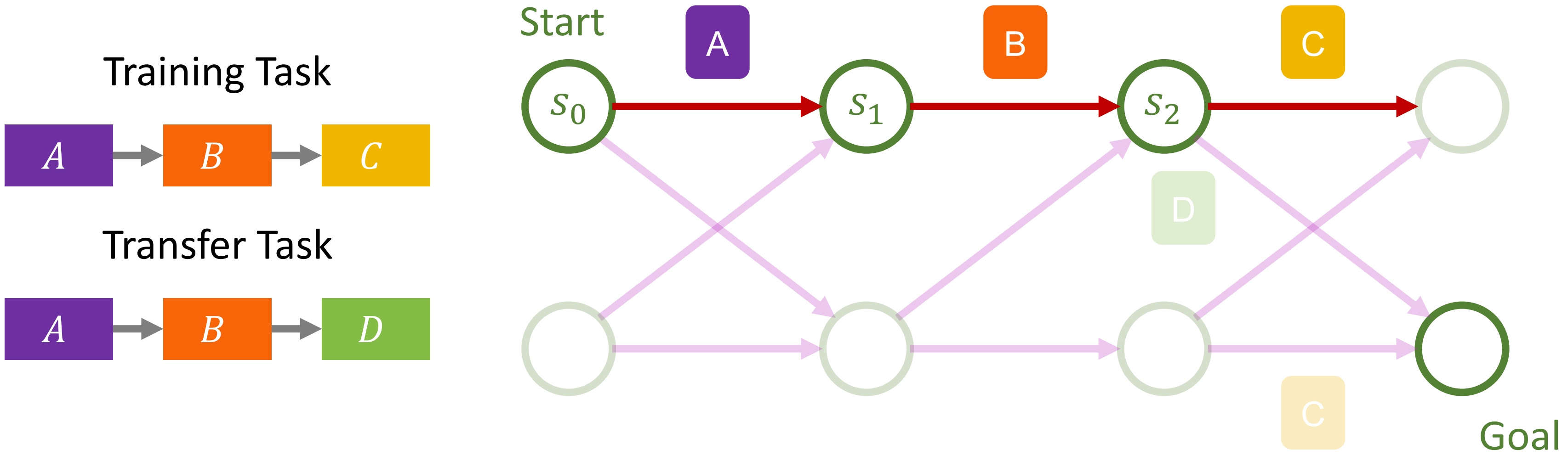


Bid values during transfer:

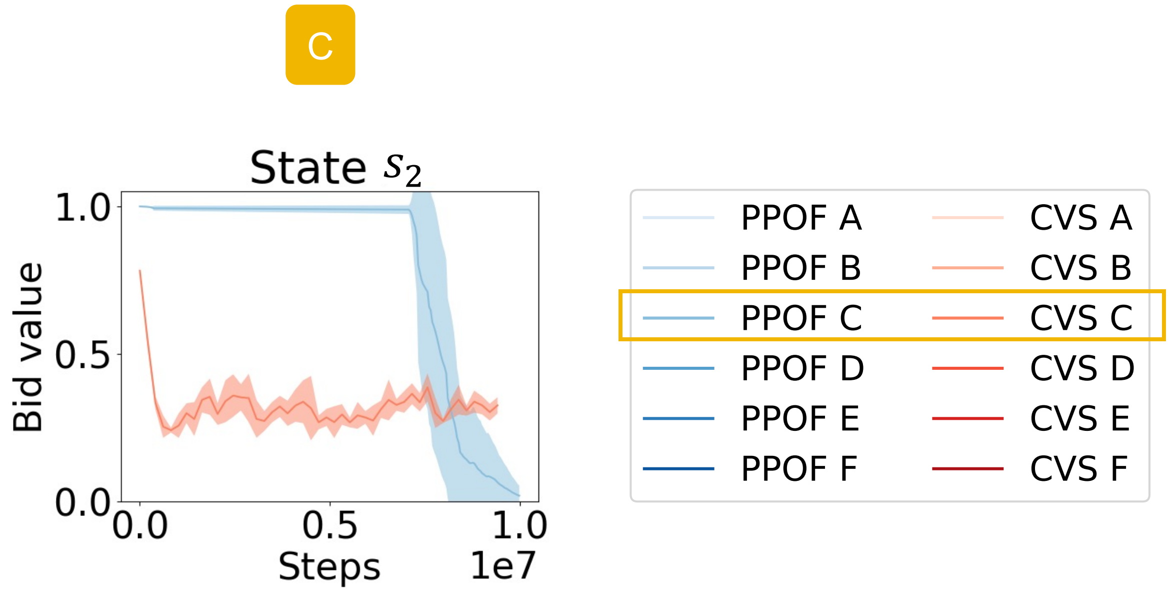


These outputs are the bids: the higher the bid, the more likely the action for that decision mechanism will be selected.

Analysis

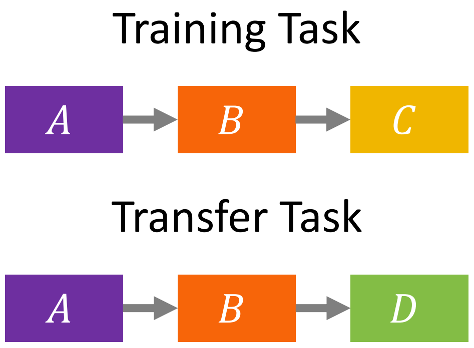


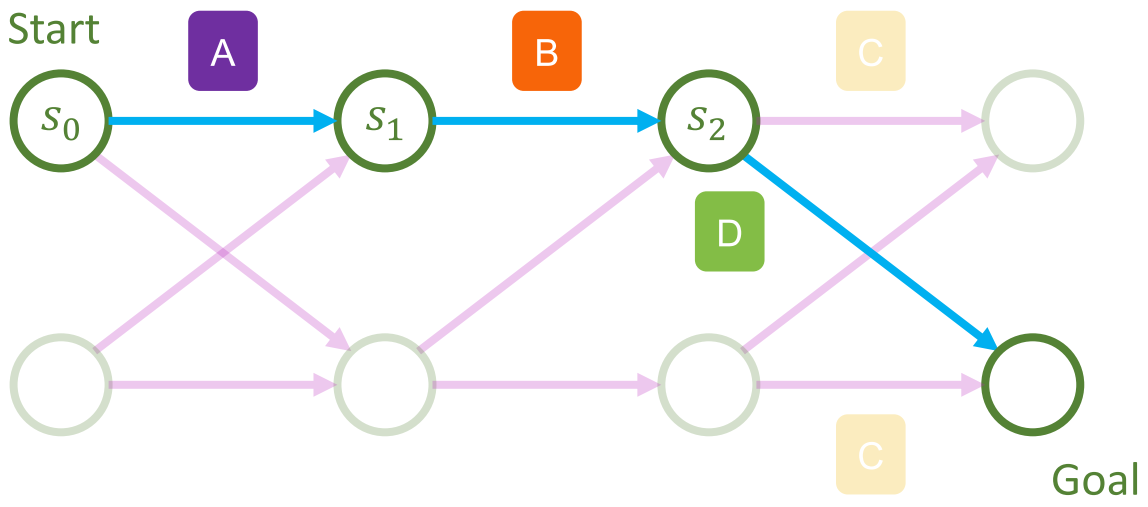
Bid values during transfer:



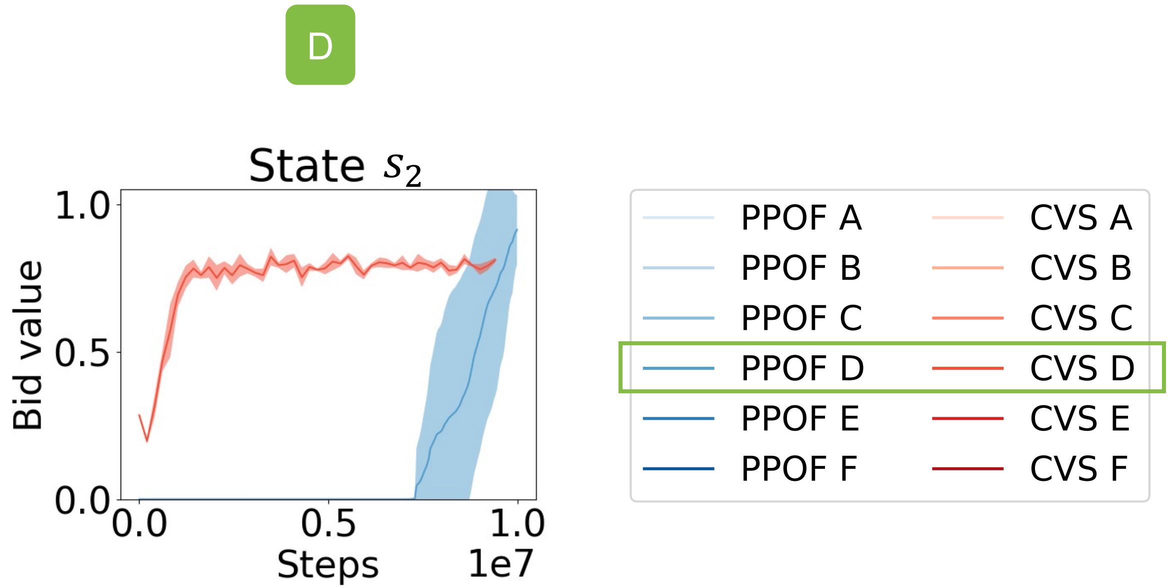
Here we see the bids for action C both drop because they are suboptimal in the transfer task,

Analysis





Bid values during transfer:



while the bids for action D rise because it is now the new optimal action.

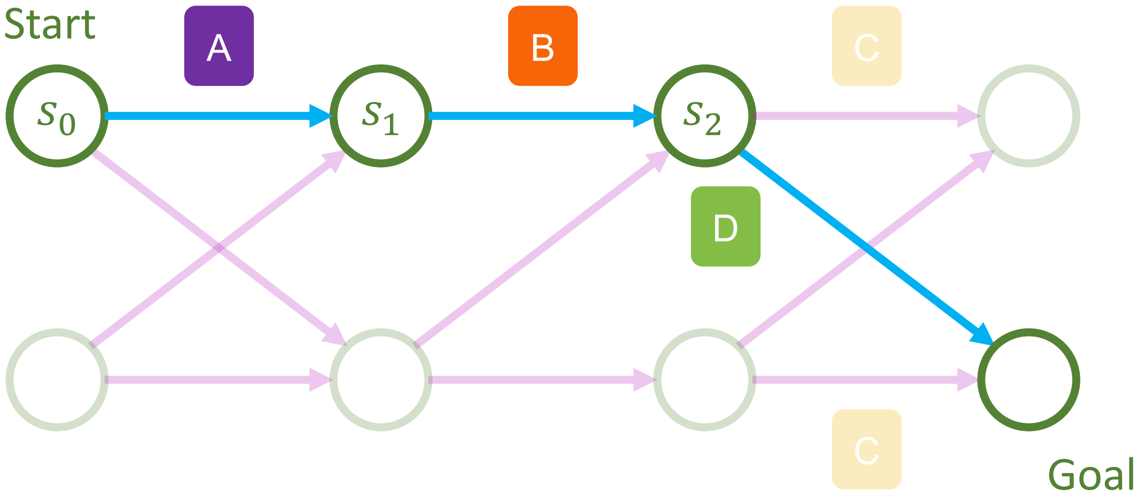
Analysis

Training Task

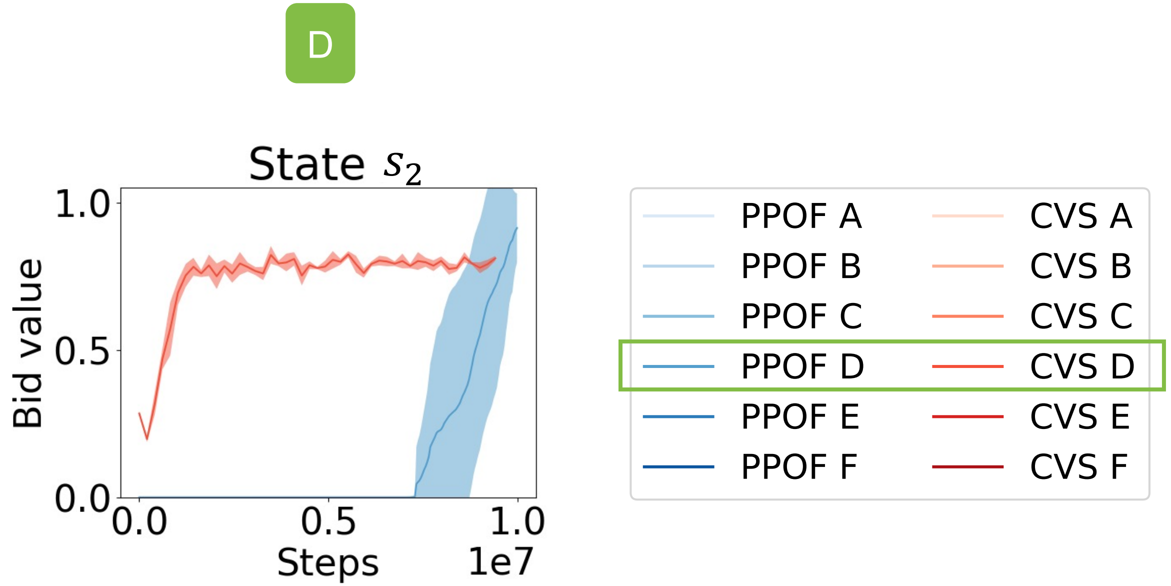
A → *B* → *C*

Transfer Task

A → *B* → *D*

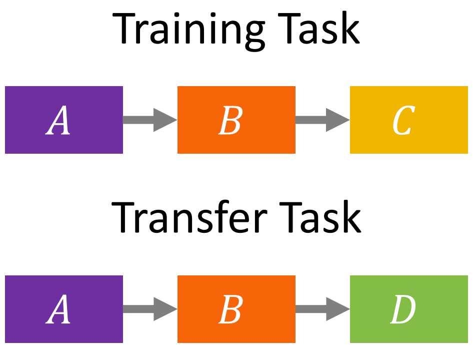


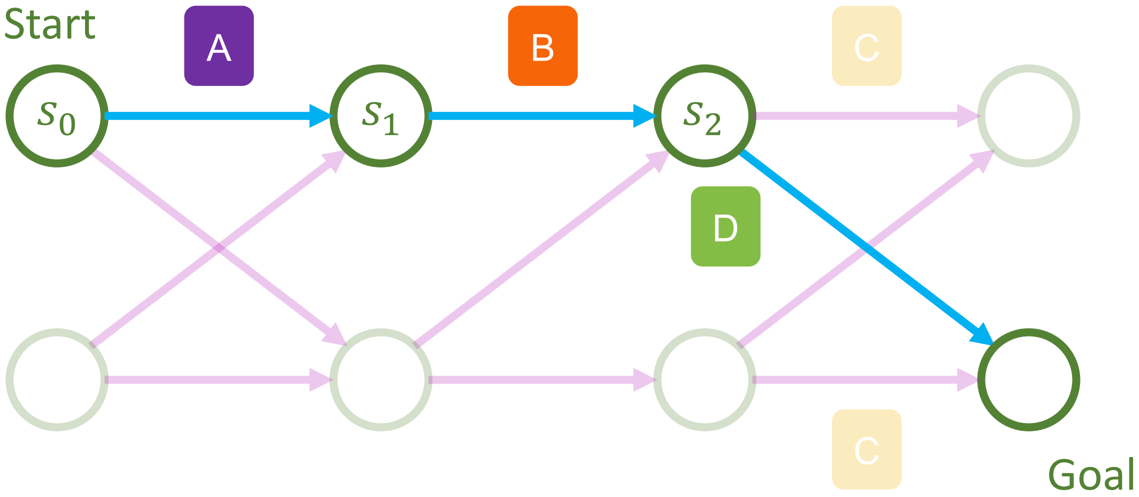
Bid values during transfer:



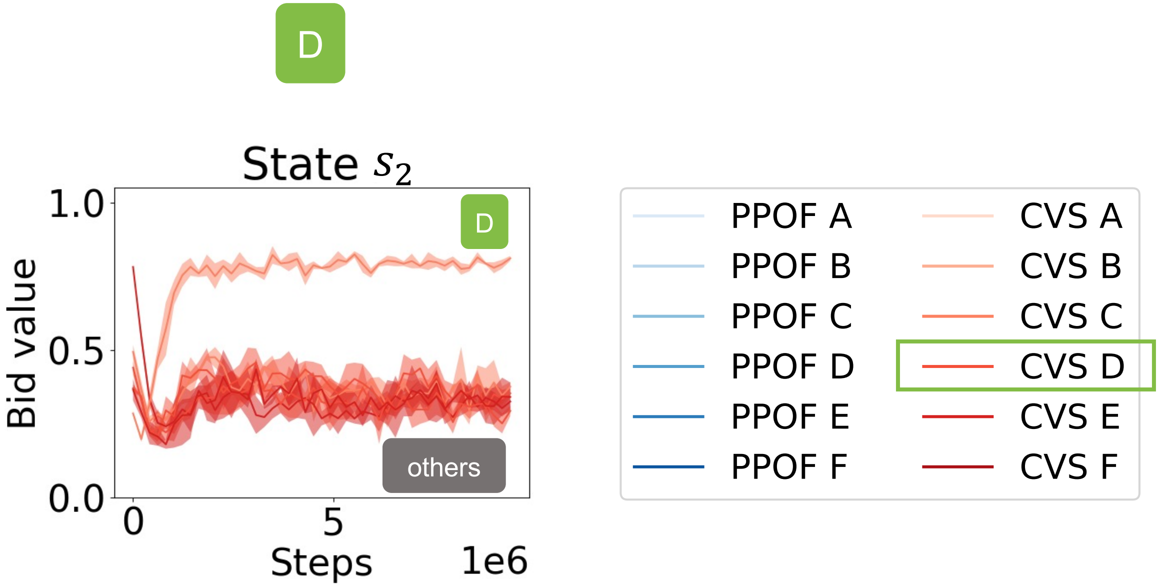
Notice how much faster CVS, in red, switches compared to PPOF, in blue,

Analysis



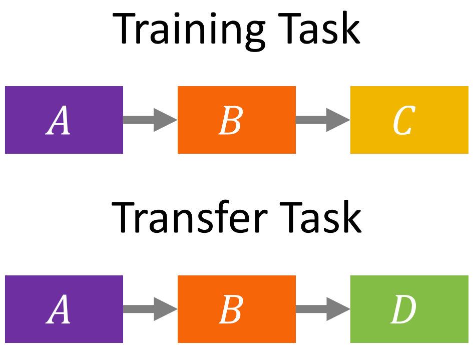


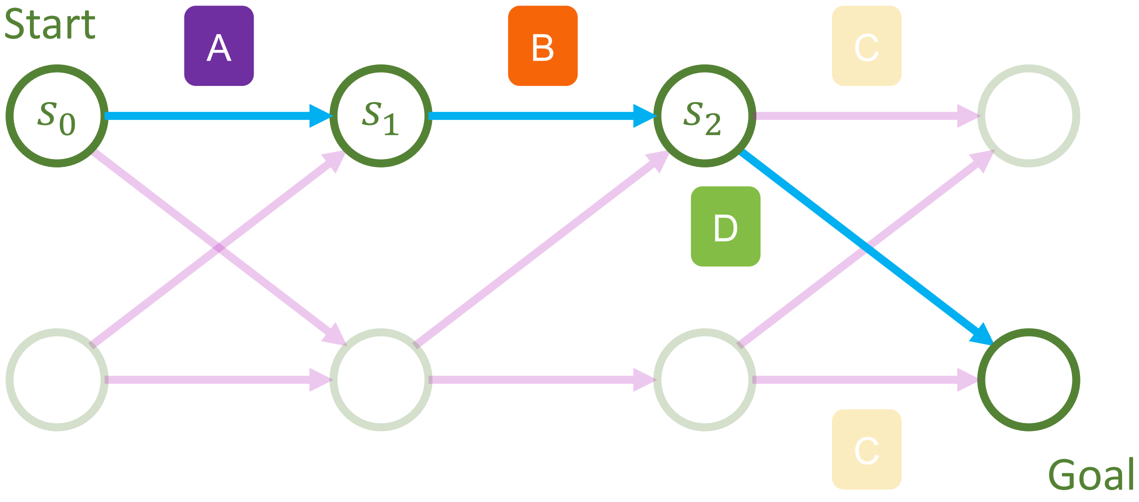
Bid values during transfer:



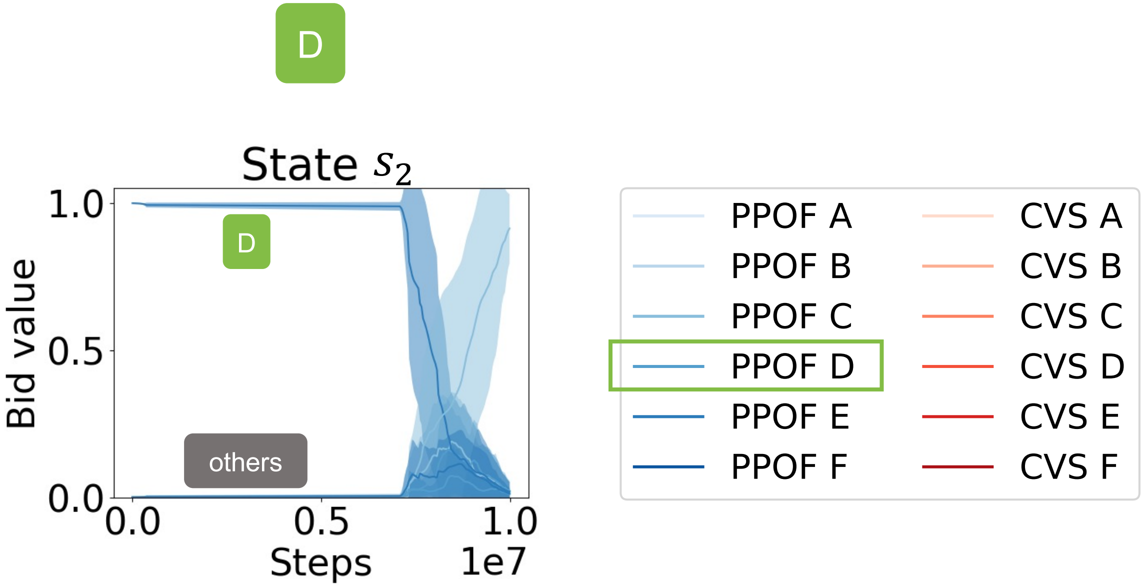
possibly because the decisions mechanisms of CVS can be adjusted independently,

Analysis



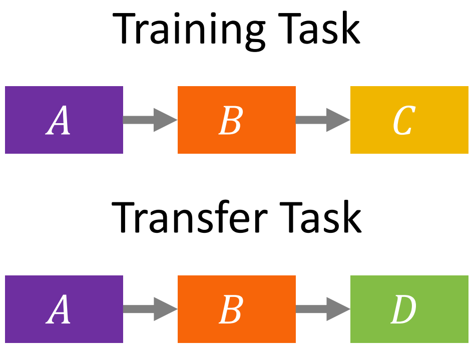


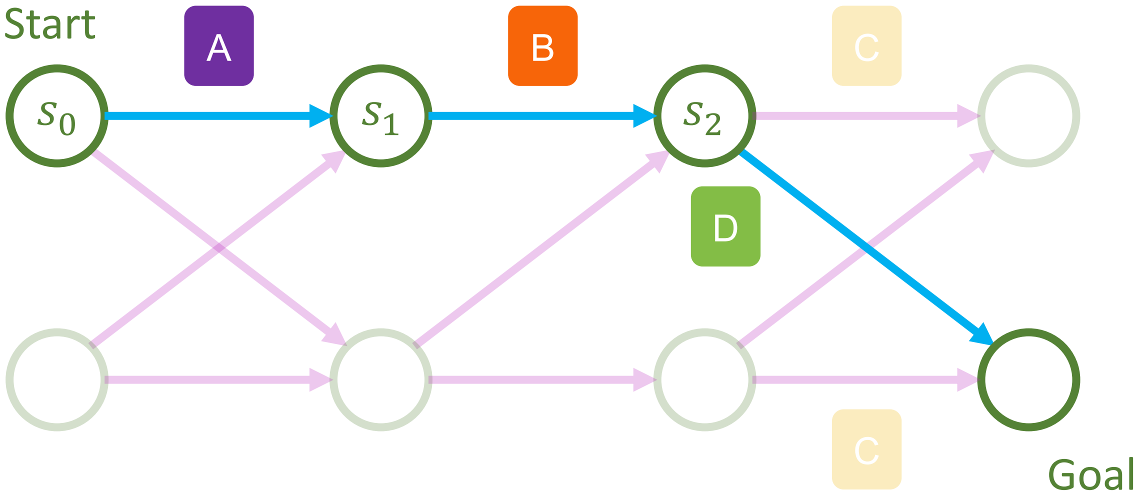
Bid values during transfer:



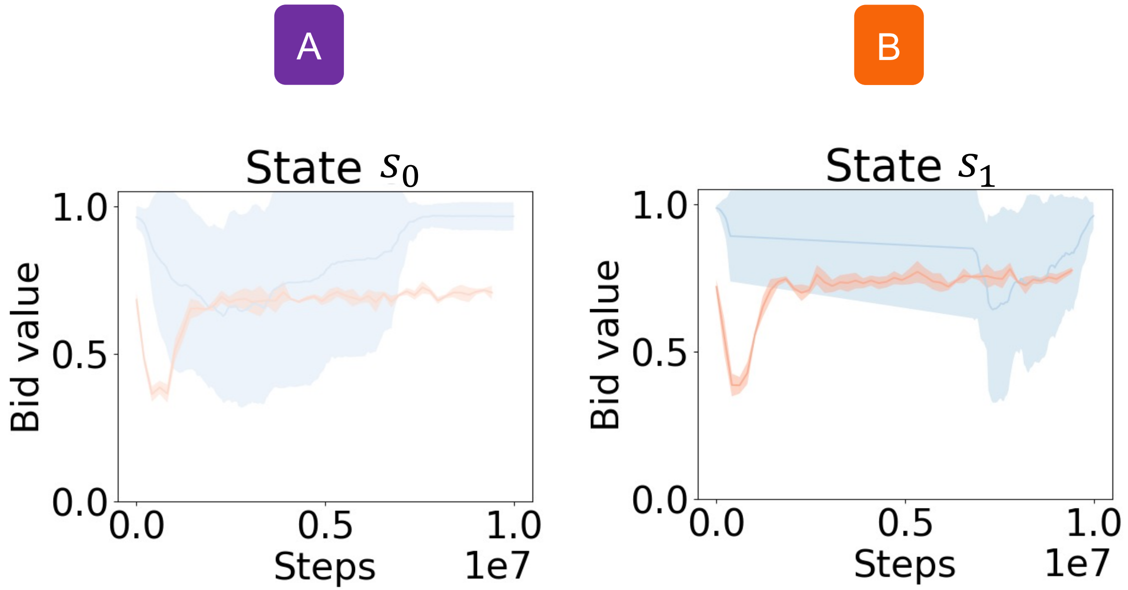
whereas for PPOF they are tied together by a softmax.

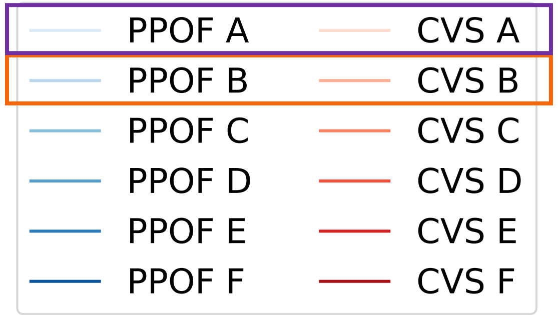
Analysis





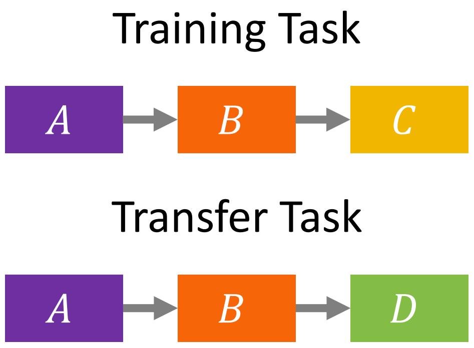
Bid values during transfer:

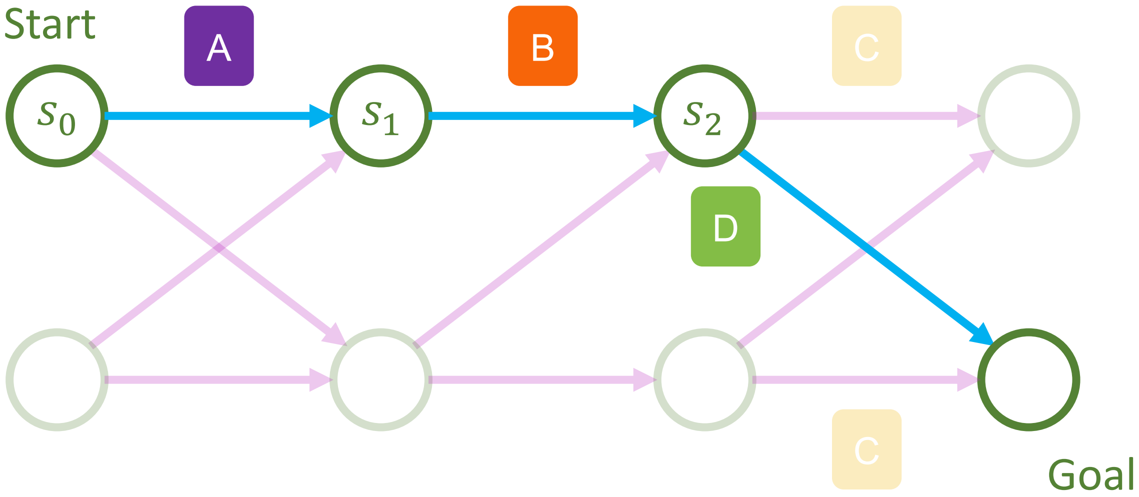




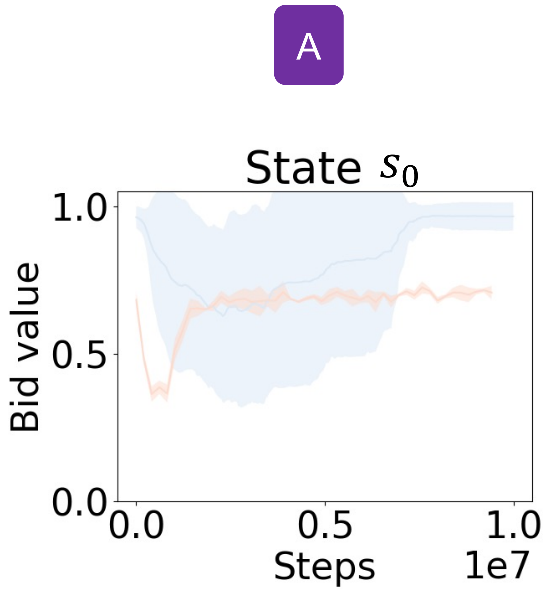
Actions A and B are the optimal actions at states 0 and 1, and we see that both of them get affected by the transfer task

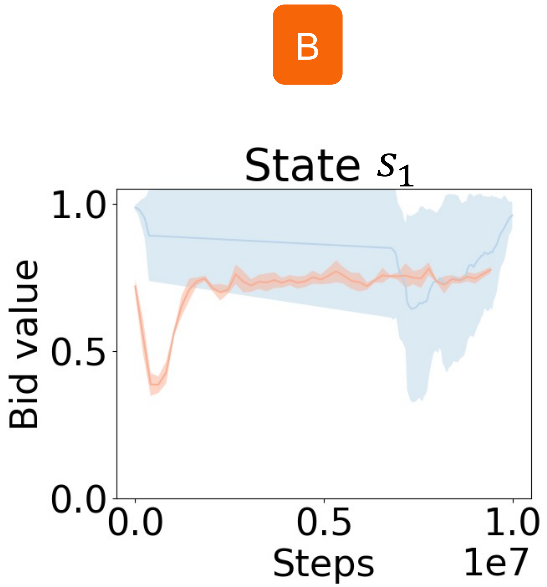
Analysis

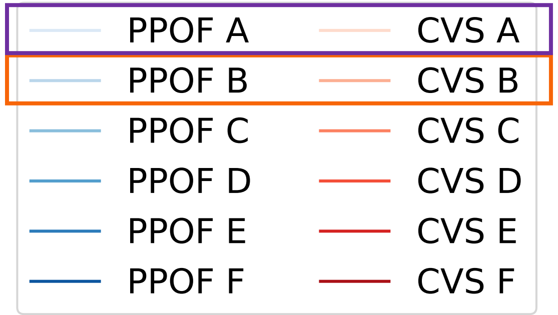




Bid values during transfer:







but CVS recovers much faster than PPOF.

Hypothesis
Modularity → more efficient transfer

To conclude, we started with the hypothesis that modularity improves transfer.

Modularity for Dynamic Systems

Independent Credit Assignment

Modularity of RL Algorithms

Hypothesis

Modularity → more efficient transfer

But we had to build up some formalism to even get to the point where we could test this hypothesis.

Hypothesis
Modularity → more efficient transfer

Modularity is algorithmic independence of mechanisms.

A dynamic system encompasses a sequence of modifications to the mechanisms.

Modularity in a dynamic system is the conditional algorithmic independence of mechanisms, conditioned on its previous state.

First we had to extend the definition of modularity developed in the causal literature to describe dynamic systems.

Hypothesis
Modularity → more efficient transfer

Modularity is algorithmic independence of mechanisms.

A dynamic system encompasses a sequence of modifications to the mechanisms.

Modularity in a dynamic system is the conditional algorithmic independence of mechanisms, conditioned on its previous state.

Learning algorithms are dynamic systems.

Modularity requires independent feedback (e.g. gradients).

Formally represent learning algorithms as algorithmic causal graphs
    independence = d-separation.

Then, we connected learning algorithms to causal graphs and

Modularity for Dynamic Systems

Independent Credit Assignment

Modularity of RL Algorithms

Hypothesis
Modularity → more efficient transfer

Modularity is algorithmic independence of mechanisms.

A dynamic system encompasses a sequence of modifications to the mechanisms.

Modularity in a dynamic system is the conditional algorithmic independence of mechanisms, conditioned on its previous state.

Learning algorithms are dynamic systems.

Modularity requires independent feedback (e.g. gradients).

Formally represent learning algorithms as algorithmic causal graphs

independence = d-separation.

proposed how to test whether the credit assignment mechanism can modify the learnable components independently.

Modularity for Dynamic Systems

Independent Credit Assignment

Modularity of RL Algorithms

Hypothesis
Modularity → more efficient transfer

Modularity is algorithmic independence of mechanisms.

A dynamic system encompasses a sequence of modifications to the mechanisms.

Modularity in a dynamic system is the conditional algorithmic independence of mechanisms, conditioned on its previous state.

Learning algorithms are dynamic systems.

Modularity requires independent feedback (e.g. gradients).

Formally represent learning algorithms as algorithmic causal graphs
independence = d-separation.

Theoretical question: Which reinforcement learning algorithms produce independent gradients?
- policy gradients ✗
- n-step temporal difference algorithms ✗
- single-step temporal difference algorithms ✓

Finally we applied this framework to determine which RL algorithms are modular

Modularity for Dynamic Systems

Independent Credit Assignment

Modularity of RL Algorithms

Hypothesis
Modularity → more efficient transfer

Modularity is algorithmic independence of mechanisms.

A dynamic system encompasses a sequence of modifications to the mechanisms.

Modularity in a dynamic system is the conditional algorithmic independence of mechanisms, conditioned on its previous state.

Learning algorithms are dynamic systems.

Modularity requires independent feedback (e.g. gradients).

Formally represent learning algorithms as algorithmic causal graphs
independence = d-separation.

Theoretical question: Which reinforcement learning algorithms produce independent gradients?
- policy gradients ❌
- n-step temporal difference algorithms ❌
- single-step temporal difference algorithms ✓

Empirical question: Does modularity improve transfer efficiency?
- empirical evidence suggests so

and showed that the hypothesis survives a suite of empirical tests.

Modularity for Dynamic Systems

Independent Credit Assignment

Modularity of RL Algorithms

Hypothesis

Modularity → more efficient transfer

Modularity is algorithmic independence of mechanisms.

A dynamic system encompasses a sequence of modifications to the mechanisms.

Modularity in a dynamic system is the conditional algorithmic independence of mechanisms, conditioned on its previous state.

Main Takeaway

Learning algorithms are dynamic systems.

Modularity requires independent feedback (e.g. gradients).

Formally represent learning algorithms as algorithmic causal graphs

independence = d-separation.

Theoretical question: Which reinforcement learning algorithms produce independent gradients?

- policy gradients ✗
- n-step temporal difference algorithms ✗
- single-step temporal difference algorithms ✓

Empirical question: Does modularity improve transfer efficiency?

- empirical evidence suggests so

What we learned from all of this is the following takeaway message:

Modularity for Dynamic Systems

Independent Credit Assignment

Modularity of RL Algorithms

Hypothesis

Modularity → more efficient transfer

Modularity is algorithmic independence of mechanisms.

A dynamic system encompasses a sequence of modifications to the mechanisms.

Modularity in a dynamic system is the conditional algorithmic independence of mechanisms, conditioned on its previous state.

Main Takeaway

To build learning algorithms that transfer efficiently, we need independently modifiable components.

Learning algorithms are dynamic systems.

Modularity requires independent feedback (e.g. gradients).

Formally represent learning algorithms as algorithmic causal graphs
independence = d-separation.

Theoretical question: Which reinforcement learning algorithms produce independent gradients?

- policy gradients ✗
- n-step temporal difference algorithms ✗
- single-step temporal difference algorithms ✓

Empirical question: Does modularity improve transfer efficiency?

- empirical evidence suggests so

Modularity for Dynamic Systems

Independent Credit Assignment

Modularity of RL Algorithms

Modularity is algorithmic independence of mechanisms.

Hypothesis

Modularity → more efficient transfer

A dynamic system encompasses a sequence of modifications to the mechanisms.

Modularity in a dynamic system is the conditional algorithmic independence of mechanisms, conditioned on its previous state.

Learning algorithms are dynamic systems.

Modularity requires independent feedback (e.g. gradients).

Formally represent learning algorithms as algorithmic causal graphs
independence = d-separation.

Main Takeaway

To build learning algorithms that transfer efficiently, we need independently modifiable components.

To get independently modifiable components, we need a credit assignment mechanism whose causal structure makes independent modification possible.

Theoretical question: Which reinforcement learning algorithms produce independent gradients?
- policy gradients ✗
- n-step temporal difference algorithms ✗
- single-step temporal difference algorithms ✓

Empirical question: Does modularity improve transfer efficiency?
- empirical evidence suggests so

To get independently modifiable components,

Modularity for Dynamic Systems

Independent Credit Assignment

Modularity of RL Algorithms

Hypothesis
Modularity → more efficient transfer

Modularity is algorithmic independence of mechanisms.

A dynamic system encompasses a sequence of modifications to the mechanisms.

Modularity in a dynamic system is the conditional algorithmic independence of mechanisms, conditioned on its previous state.

Main Takeaway

To build learning algorithms that transfer efficiently, we need independently modifiable components.

To get independently modifiable components, we need a credit assignment mechanism whose causal structure makes independent modification possible.

Learning algorithms are dynamic systems.

Modularity requires independent feedback (e.g. gradients).

Formally represent learning algorithms as algorithmic causal graphs
 independence = d-separation.

Theoretical question: Which reinforcement learning algorithms produce independent gradients?
- policy gradients ✗
- n-step temporal difference algorithms ✗
- single-step temporal difference algorithms ✓

Empirical question: Does modularity improve transfer efficiency?
- empirical evidence suggests so

we need a credit assignment mechanism whose causal structure makes independent modification possible.