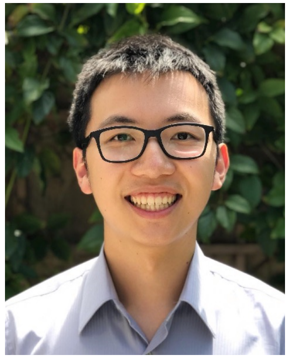
Training and Inference on Any-Order Autoregressive Models the Right Way


Andy Shih


Dorsa Sadigh


Stefano Ermon

Stanford University

This Talk

- Autoregressive Models
  powerful models, but trouble with marginal inference

This Talk

- Autoregressive Models
  powerful models, but trouble with marginal inference

- Any-Order Autoregressive Models (AO-ARMs)
  can do marginal inference, but have some inefficiencies

This Talk

- Autoregressive Models
  powerful models, but trouble with marginal inference

- Any-Order Autoregressive Models (AO-ARMs)
  can do marginal inference, but have some inefficiencies

- MAC: our proposed improvement of AO-ARMs
  address these inefficiencies!

Autoregressive Models

$$\log p(\boldsymbol{x}) = \sum_{i=1}^{N} \log p(x_i | \boldsymbol{x}_{<i})$$

Autoregressive Models

$$\underbrace{\log p(\boldsymbol{x})}_{\text{joint}} = \sum_{i=1}^{N} \underbrace{\log p(x_i | \boldsymbol{x}_{<i})}_{\text{univariate conditional}}$$

joint

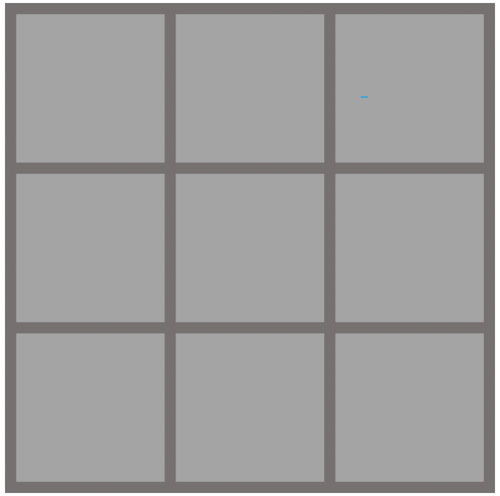
univariate
conditional

Autoregressive Models

$$\underbrace{\log p(\boldsymbol{x})}_{\text{joint}} = \sum_{i=1}^{N} \underbrace{\log p(x_i | \boldsymbol{x}_{<i})}_{\text{univariate conditional}}$$

joint

univariate conditional

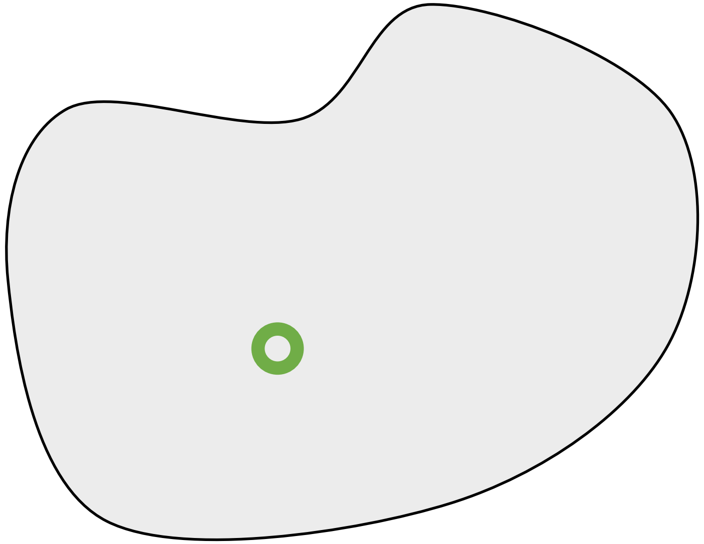


Autoregressive Models

$$\log p(\boldsymbol{x})$$

joint

I like to play sports



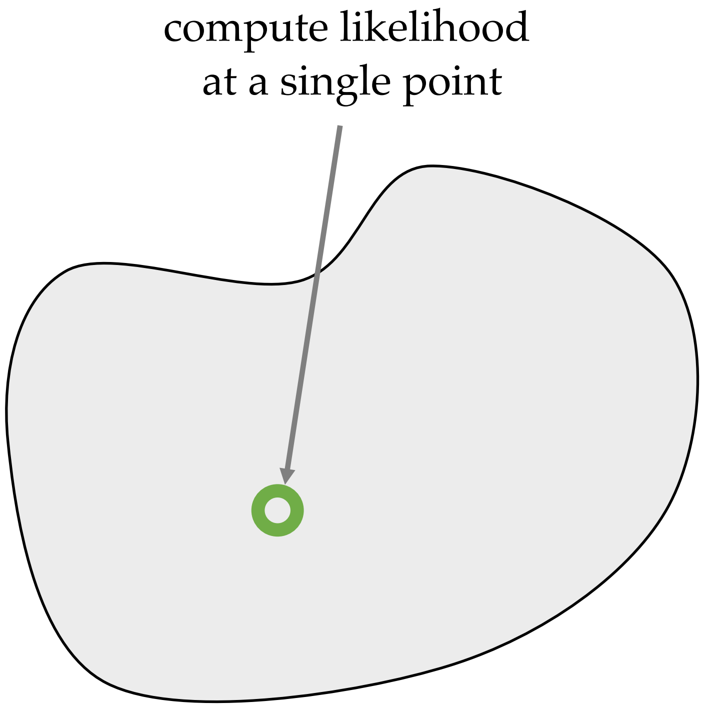
Autoregressive Models

$$\log p(\boldsymbol{x})$$

joint

I like to play sports

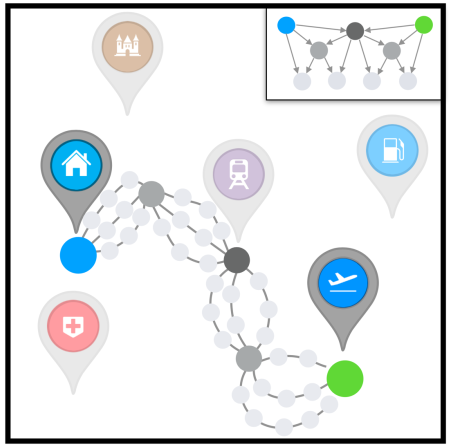
$$\log p(x_1) +$$
$$\log p(x_2|x_1) +$$
$$\log p(x_3|x_1, x_2) +$$
$$\dots$$



But sometimes we have partial evidence

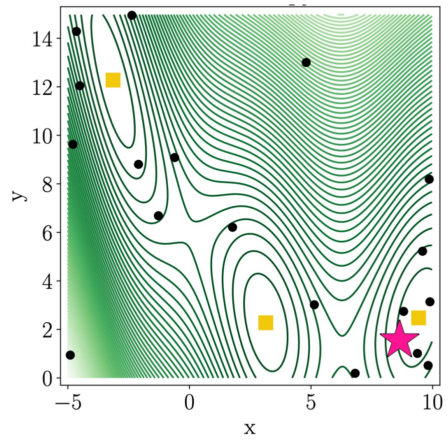
Image

Planning



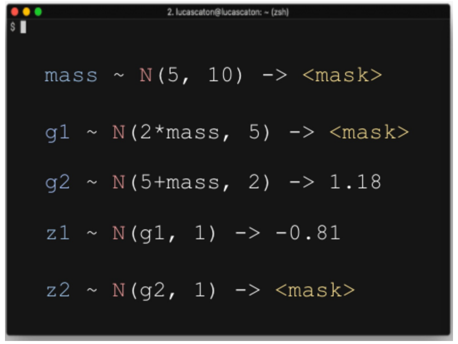
Pertsch et al.

Bayesian Optimization



Neiswanger et al.

Probabilistic Programming



Wu et al.

Language

I l__e _o _l_y s__rts

and more…

Difficulty: The evidence subset is different for each query!

But sometimes we have partial evidence

Image

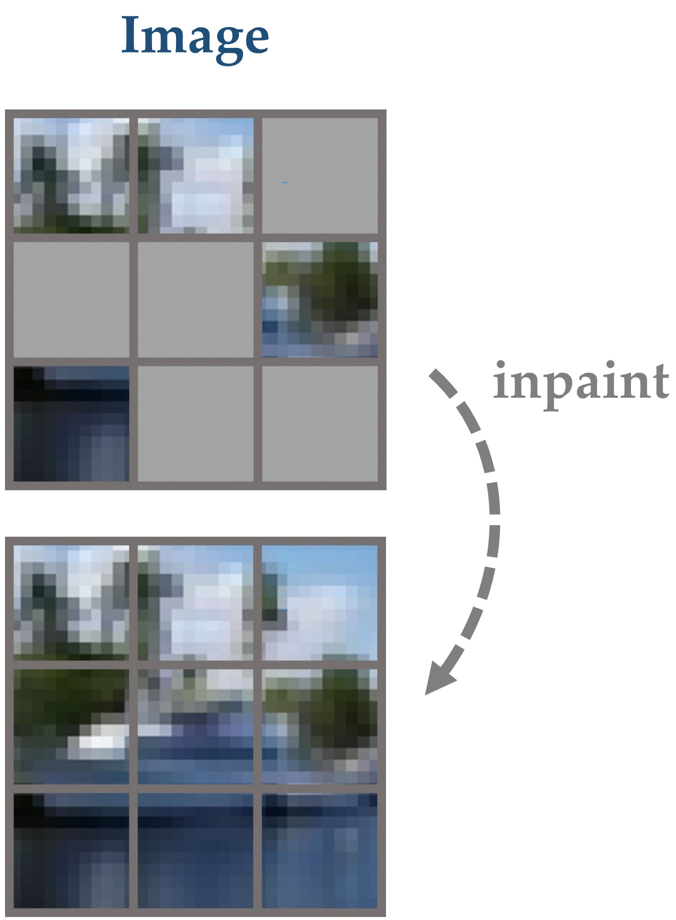
need to compute marginals

$$\log p(\boldsymbol{x}_e)$$

e: subset of variables

Difficulty: The evidence subset is different for each query!

But sometimes we have partial evidence



need to compute marginals

$$\log p(\boldsymbol{x}_e)$$

e: subset of variables

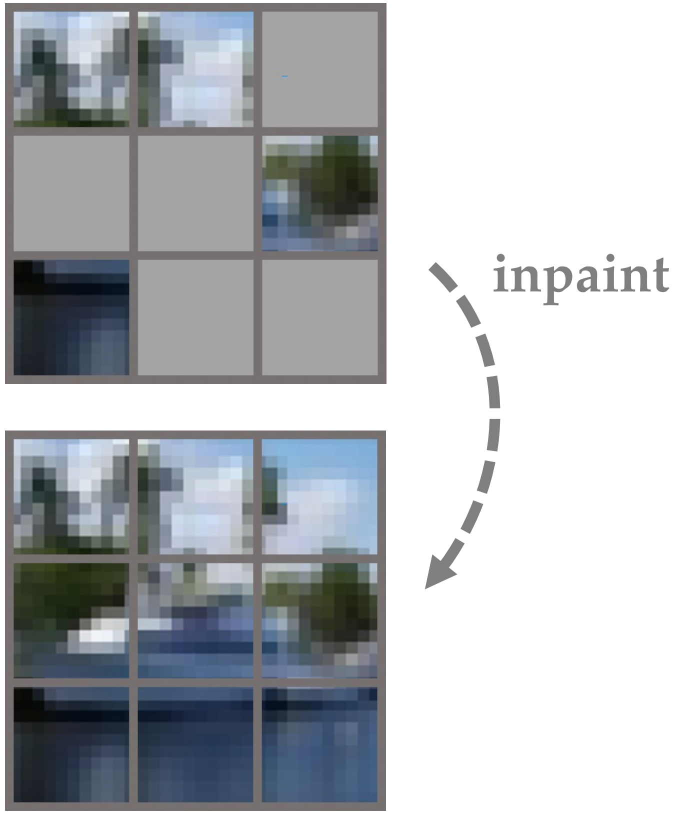
and conditionals

$$\log p(\boldsymbol{x}_q | \boldsymbol{x}_e)$$

Difficulty: The evidence subset is different for each query!

But sometimes we have partial evidence

Image



need to compute marginals

$$\log p(\boldsymbol{x}_e)$$

e: subset of variables

and conditionals

$$\log p(\boldsymbol{x}_q|\boldsymbol{x}_e) = \log p(\boldsymbol{x}_q \boldsymbol{x}_e) - \log p(\boldsymbol{x}_e)$$

Difficulty: The evidence subset is different for each query!

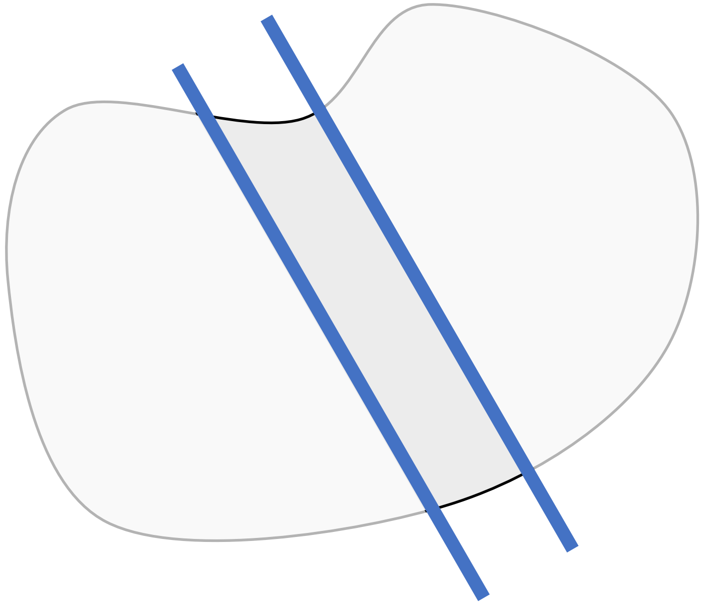
Autoregressive Models

$\log p(\boldsymbol{x}_e)$

marginal

e: subset of variables

I l__e _o _l_y s__rts



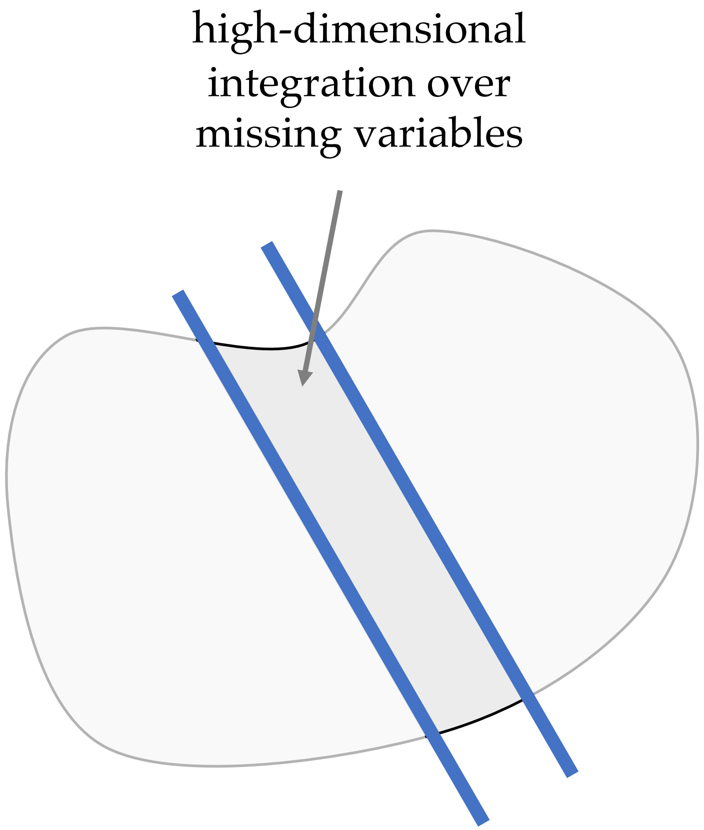
Autoregressive Models

$\log p(\boldsymbol{x}_e)$

marginal

e: subset of variables

I l__e _o _l_y s__rts



$$p(x_1, x_3) = \int_{x_2} \int_{x_4} p(x_1, x_2, x_3, x_4) dx_2 dx_4$$

Autoregressive Models

$$\log p(\boldsymbol{x}_e)$$

marginal

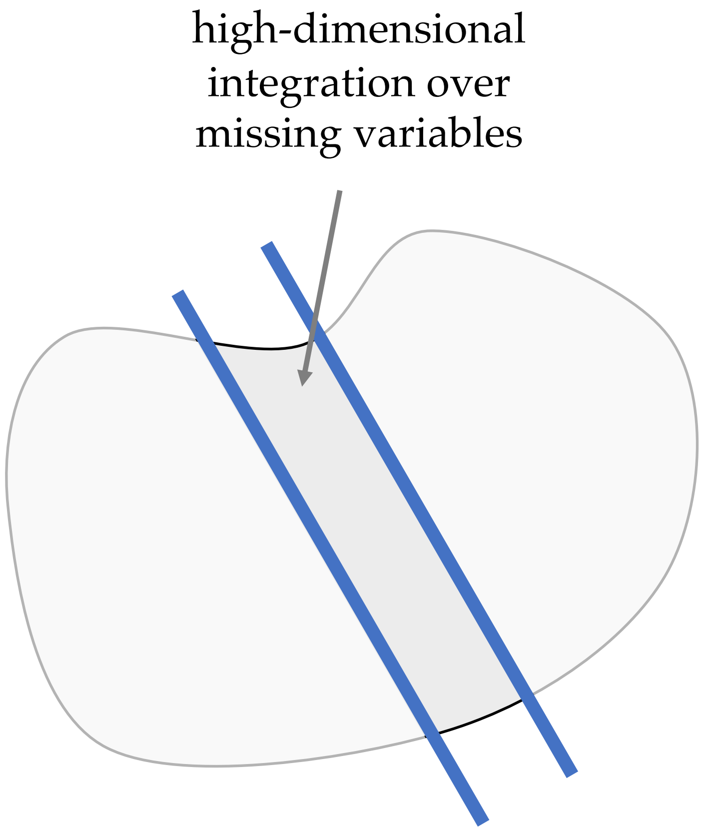
e: subset of variables

If *e* is a prefix of the ordering

✓ $e = \{1, 2\}$

$$\log p(x_1, x_2) = \log p(x_1) + \log p(x_2 | x_1)$$

I l__e _o _l_y s__rts



$$p(x_1, x_3) = \int_{x_2} \int_{x_4} p(x_1, x_2, x_3, x_4) dx_2 dx_4$$

Autoregressive Models

$\log p(\boldsymbol{x}_e)$

marginal

e: subset of variables

If e is a prefix of the ordering

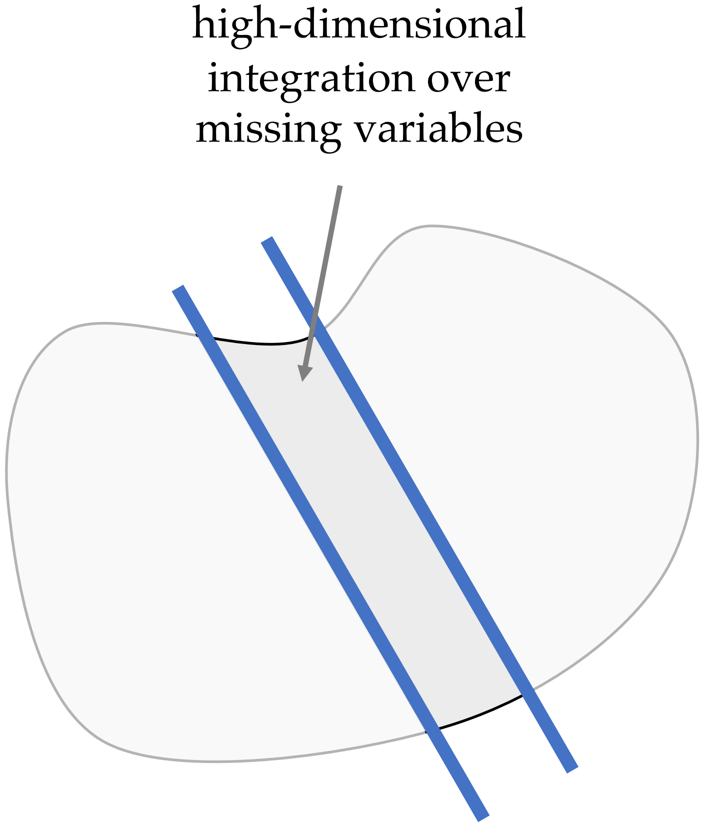
✓ $e = \{1, 2\}$

$$\log p(x_1, x_2) = \log p(x_1) + \log p(x_2 | x_1)$$

I l__e _o _l_y s__rts

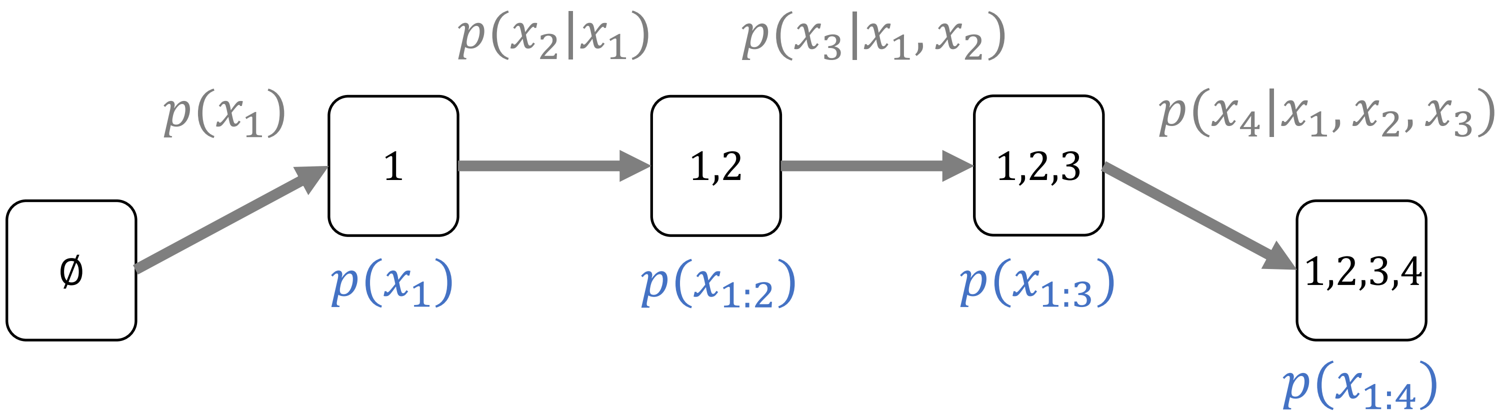
✗ $e = \{1, 3\}$

$$p(x_1, x_3) = \int_{x_2} \int_{x_4} p(x_1, x_2, x_3, x_4) dx_2 dx_4$$



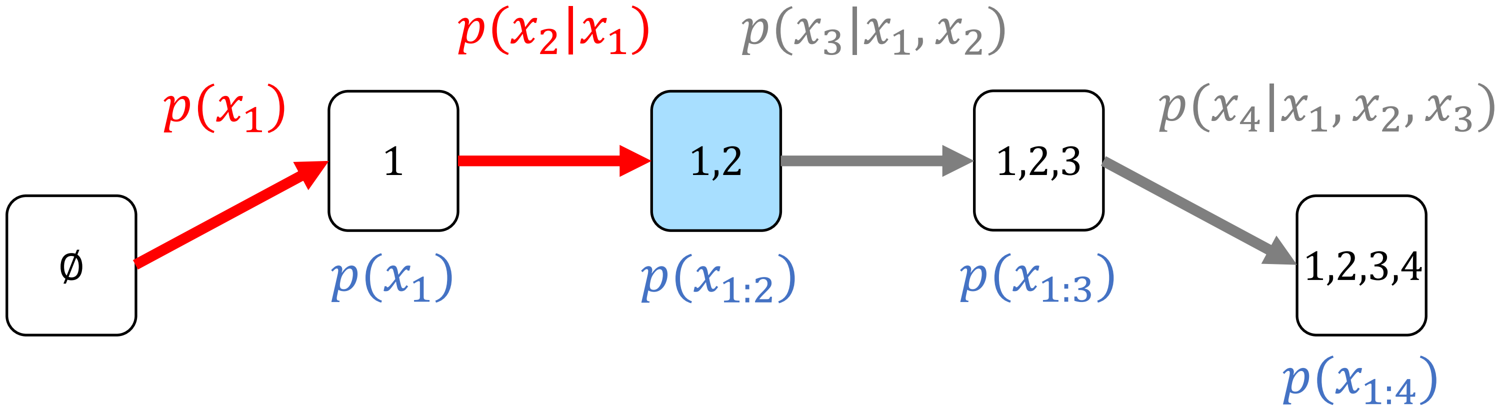
Autoregressive Models

Forward Ordering
1, 2, 3, 4



Autoregressive Models

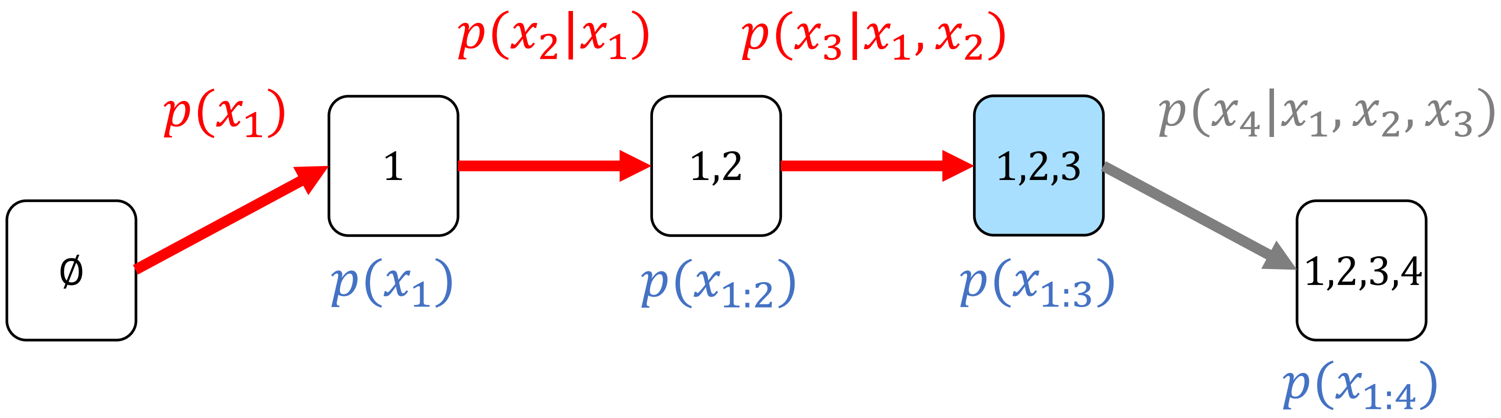
Forward Ordering
1, 2, 3, 4



$$\log p(\boldsymbol{x}_{1:2}) = \log p(x_1) + \log p(x_2|x_1)$$

Autoregressive Models

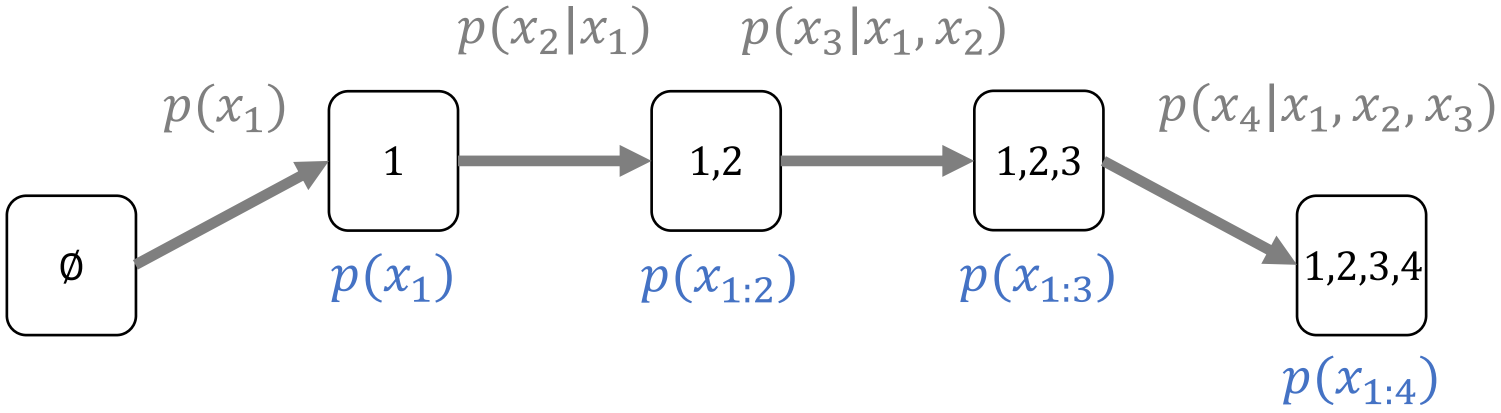
Forward Ordering
1, 2, 3, 4



$$\log p(\boldsymbol{x}_{1:3}) = \log p(x_1) + \log p(x_2|x_1) + \log p(x_3|x_1, x_2)$$

Autoregressive Models

Forward Ordering
1, 2, 3, 4



$\log p(\boldsymbol{x}_{3,4})$ ☹

Any-Order Autoregressive Models

Any-Order Autoregressive Models

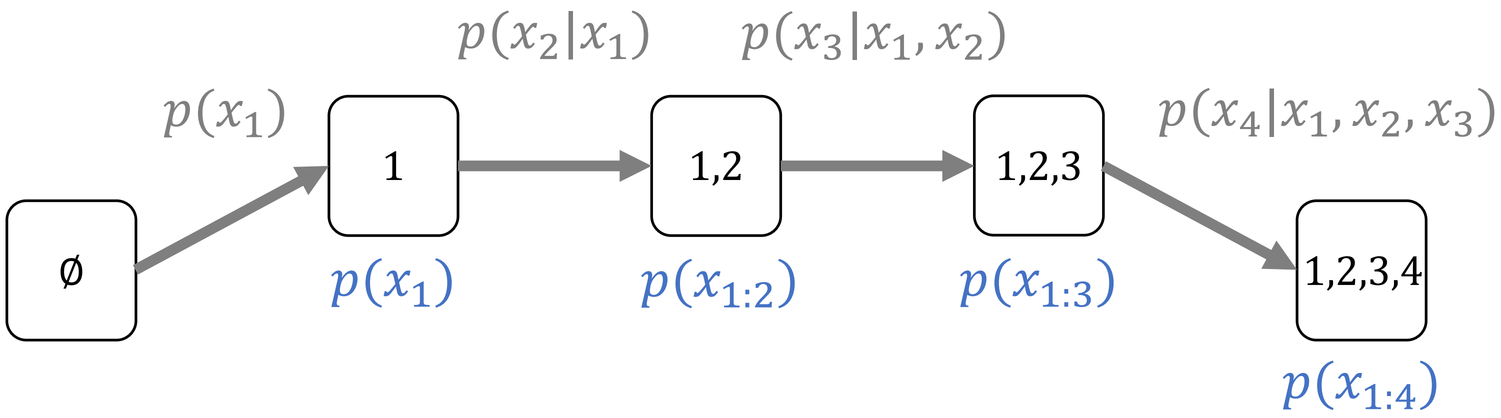
AO-ARMs
for inference on partial evidence

Current AO-ARMs

earliest	A deep and tractable density estimator [Uria et al. 2014]
language	BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding [Devlin et al. 2018]
language	XLNet: Generalized Autoregressive Pretraining for Language Understanding [Yang et al. 2019]
continuous	Arbitrary Conditional Distributions with Energy [Strauss et al. 2021]
text/image	Autoregressive Diffusion Models [Hoogeboom et al. 2022]

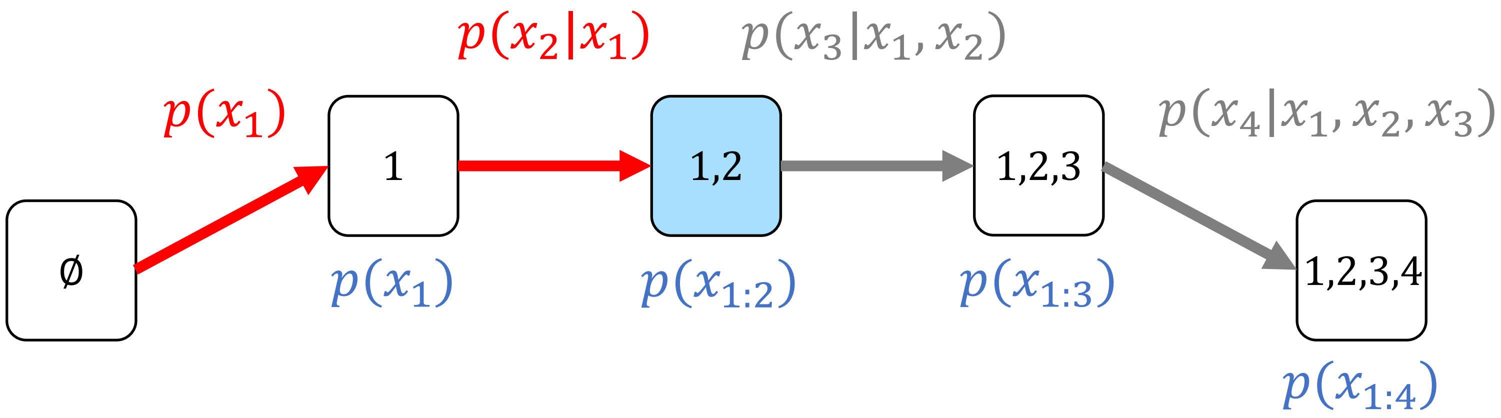
One Autoregressive Model

Forward Ordering
1, 2, 3, 4



One Autoregressive Model

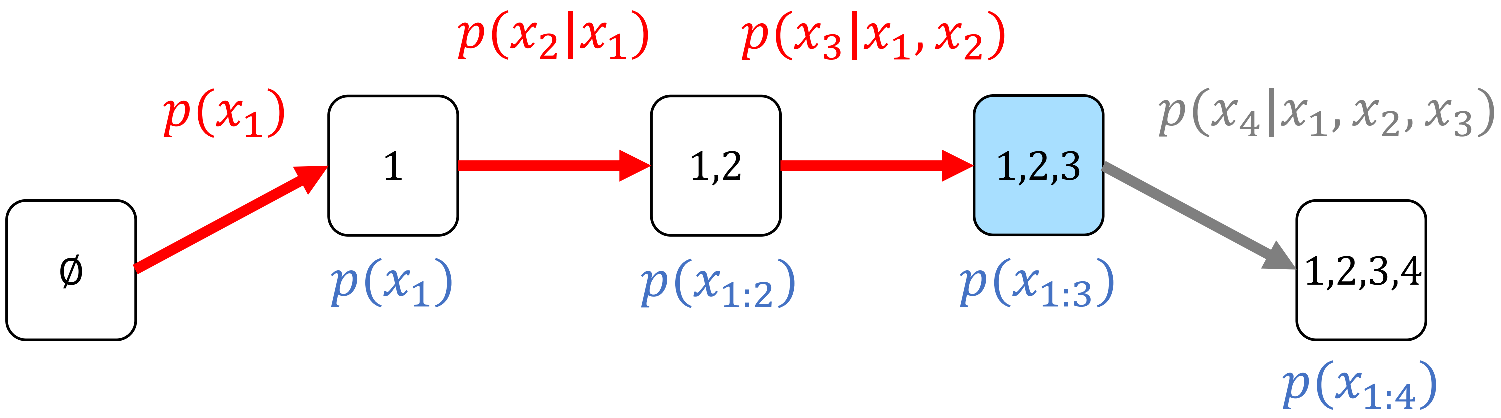
Forward Ordering
1, 2, 3, 4



$$\log p(\boldsymbol{x}_{1:2}) = \log p(x_1) + \log p(x_2|x_1)$$

One Autoregressive Model

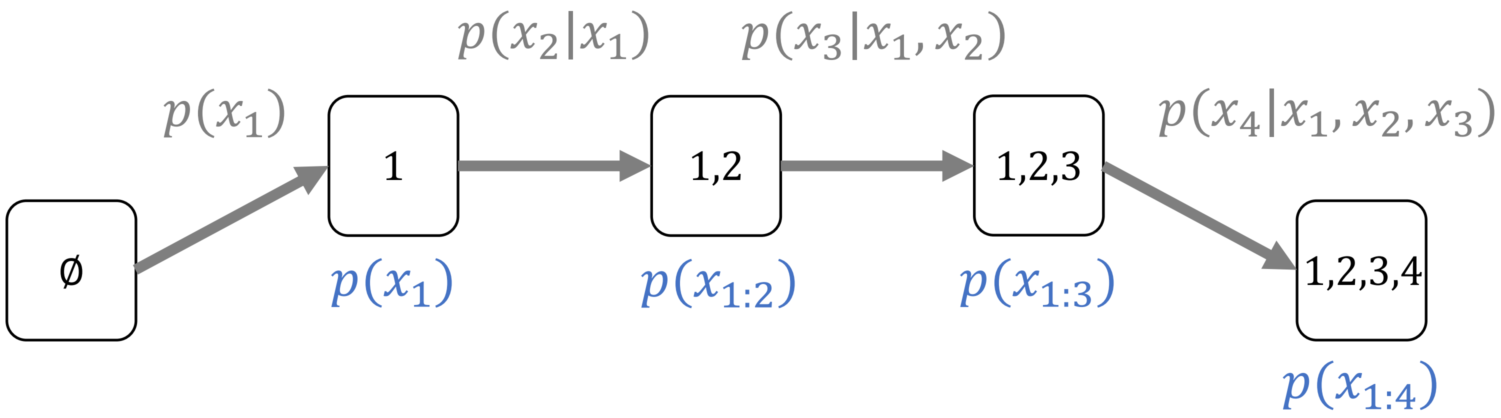
Forward Ordering
1, 2, 3, 4



$$\log p(\boldsymbol{x}_{1:3}) = \log p(x_1) + \log p(x_2|x_1) + \log p(x_3|x_1, x_2)$$

One Autoregressive Model

Forward Ordering
1, 2, 3, 4

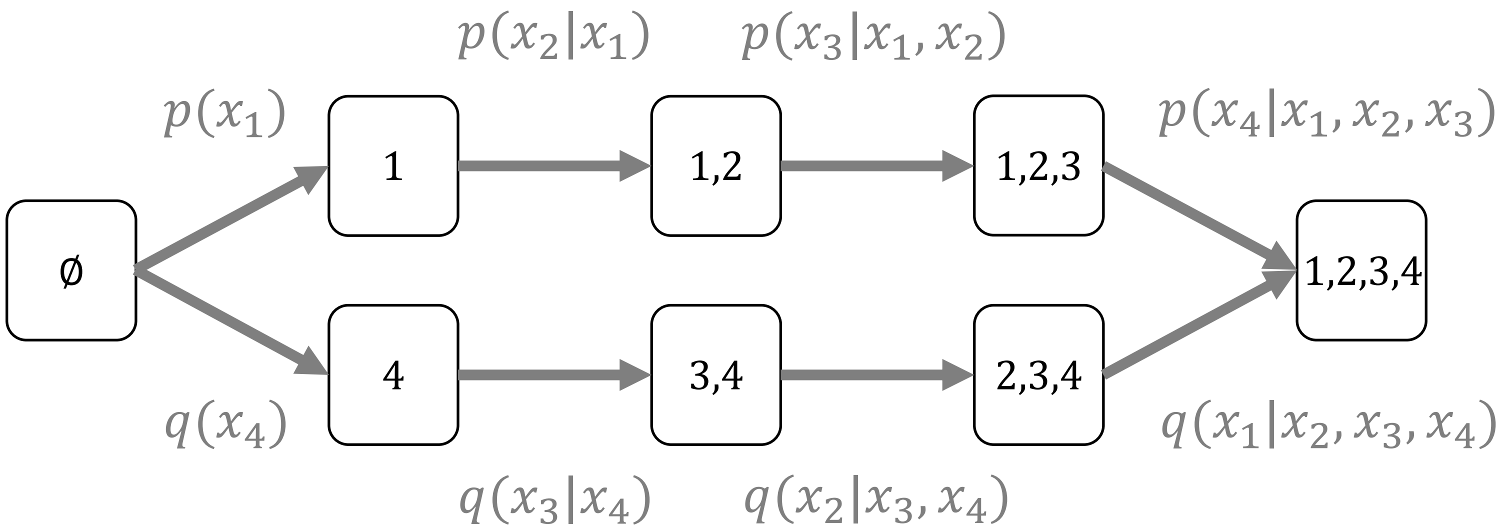


$\log p(\boldsymbol{x}_{3,4})$ ☹

Two Autoregressive Models

Forward Ordering
1, 2, 3, 4

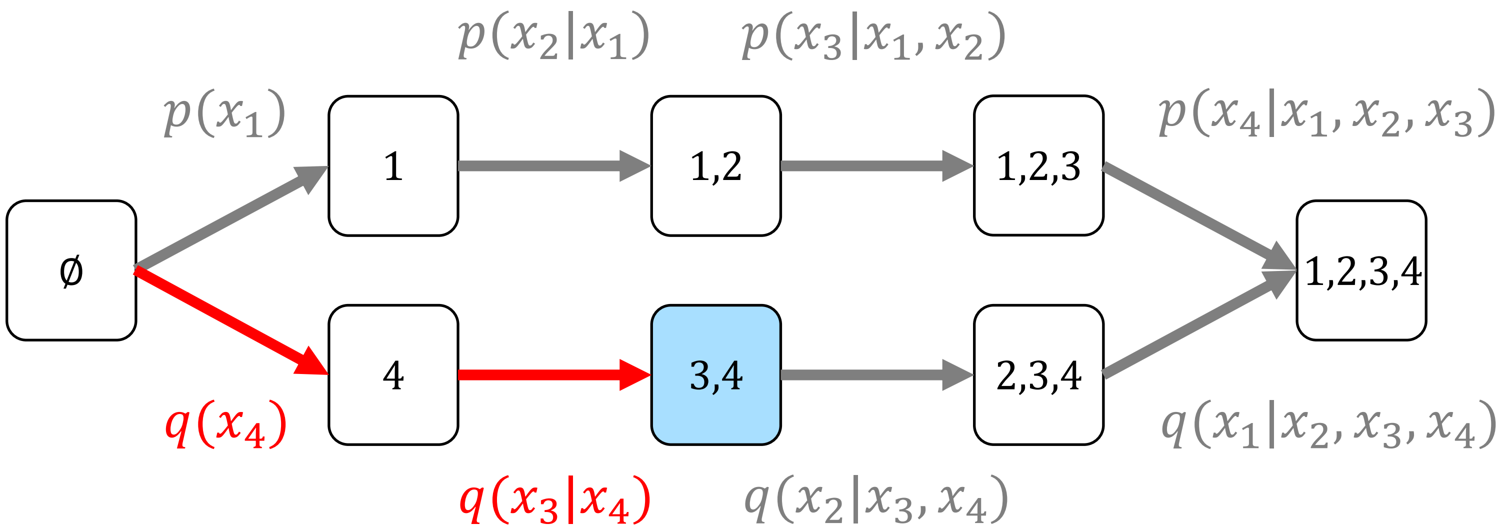
Reverse Ordering
4, 3, 2, 1



Two Autoregressive Models

Forward Ordering
1, 2, 3, 4

Reverse Ordering
4, 3, 2, 1



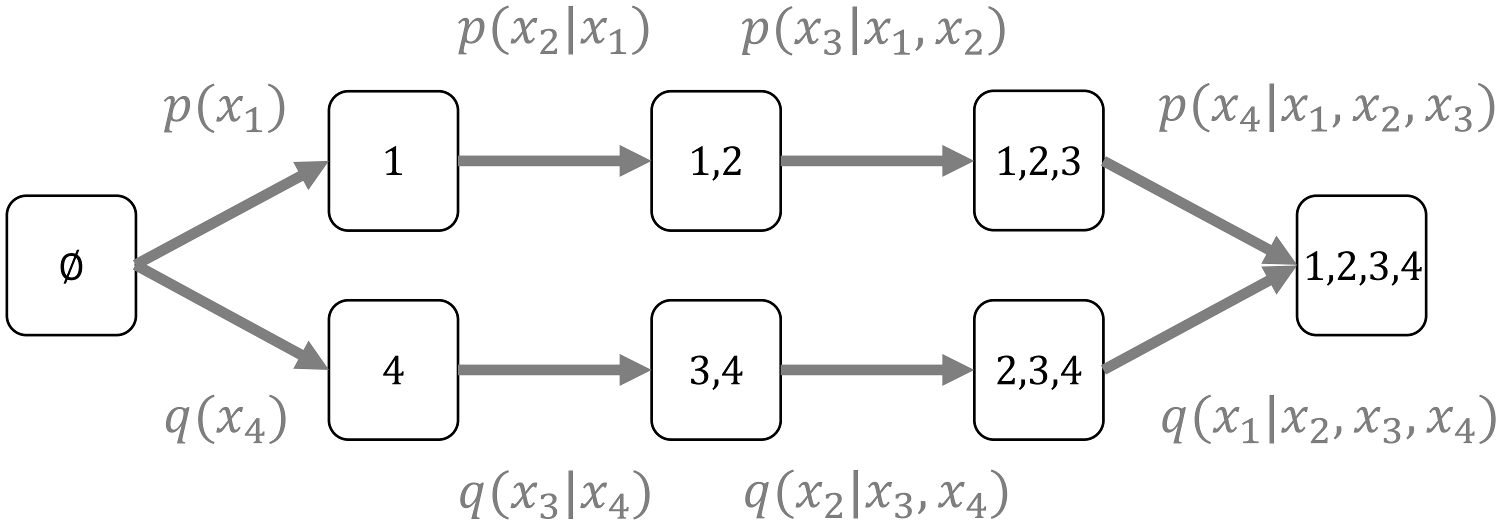
$$\log p(\boldsymbol{x}_{3:4}) = \log p(x_4) + \log p(x_3|x_4)$$

Any-Order Autoregressive Model

Forward Ordering
1, 2, 3, 4

Reverse Ordering
4, 3, 2, 1

… every ordering!



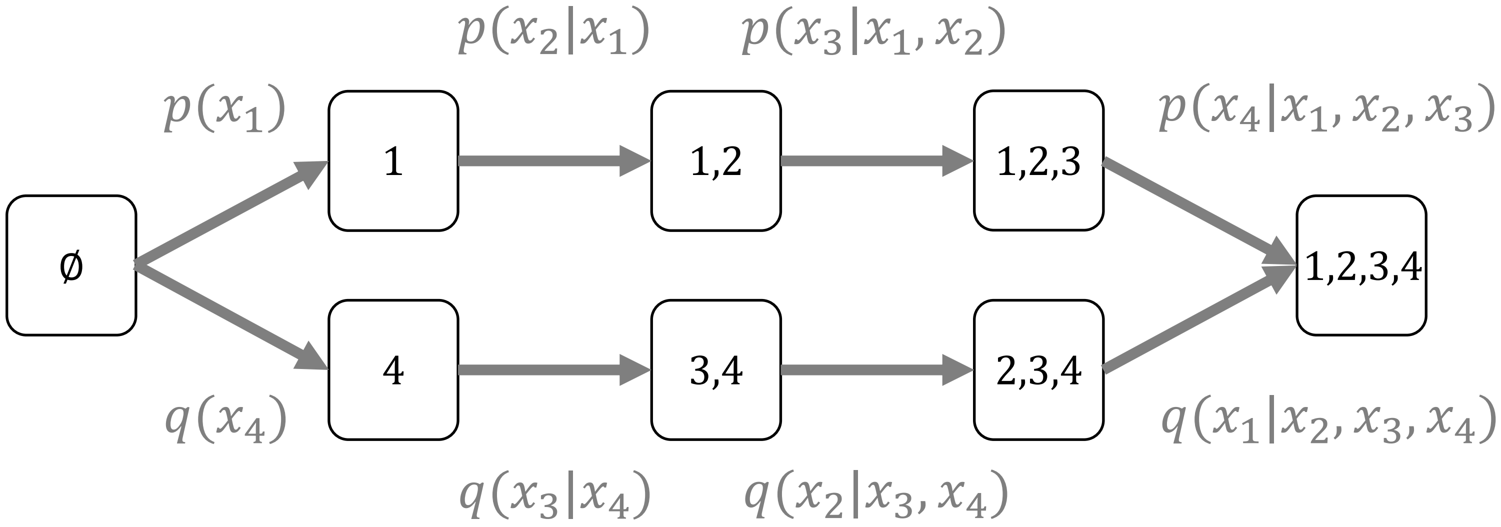
Every mask is a prefix of some order!

Any-Order Autoregressive Model

Forward Ordering
1, 2, 3, 4

Reverse Ordering
4, 3, 2, 1

… every ordering!



Every mask is a prefix of some order!

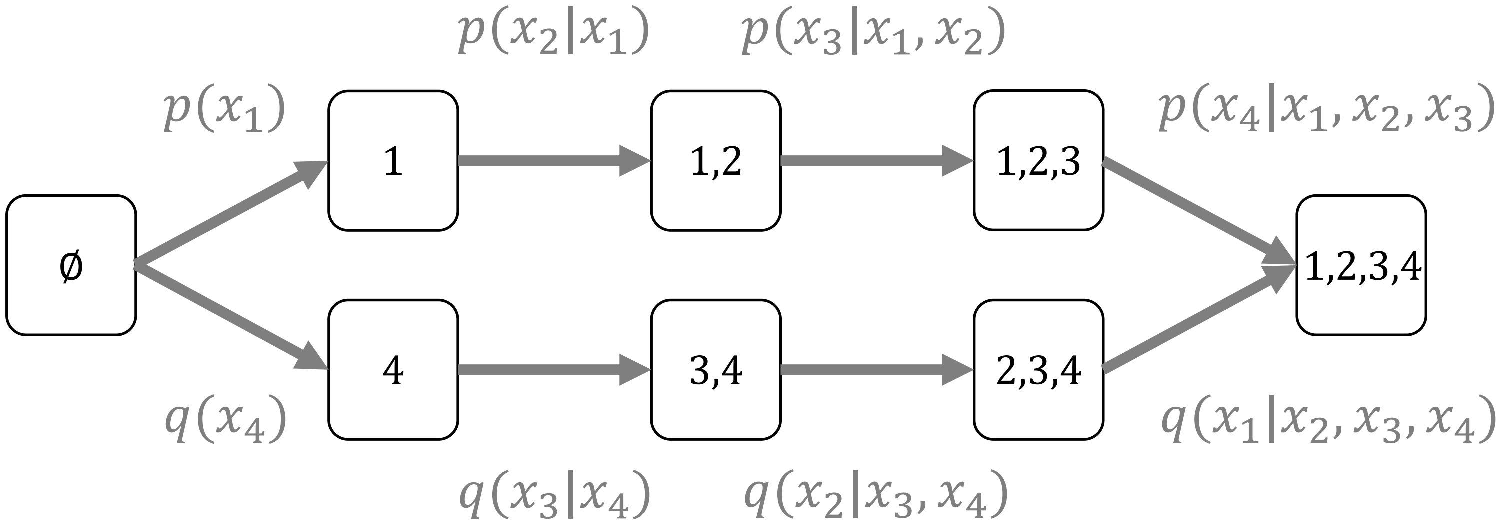
$e = \{1, 3\}$ $\sigma = 1, 3, 2, 4$

Any-Order Autoregressive Model

Forward Ordering
1, 2, 3, 4

Reverse Ordering
4, 3, 2, 1

… every ordering!



Every mask is a prefix of some order!

$e = \{1, 3\}$ $\sigma = 1, 3, 2, 4$

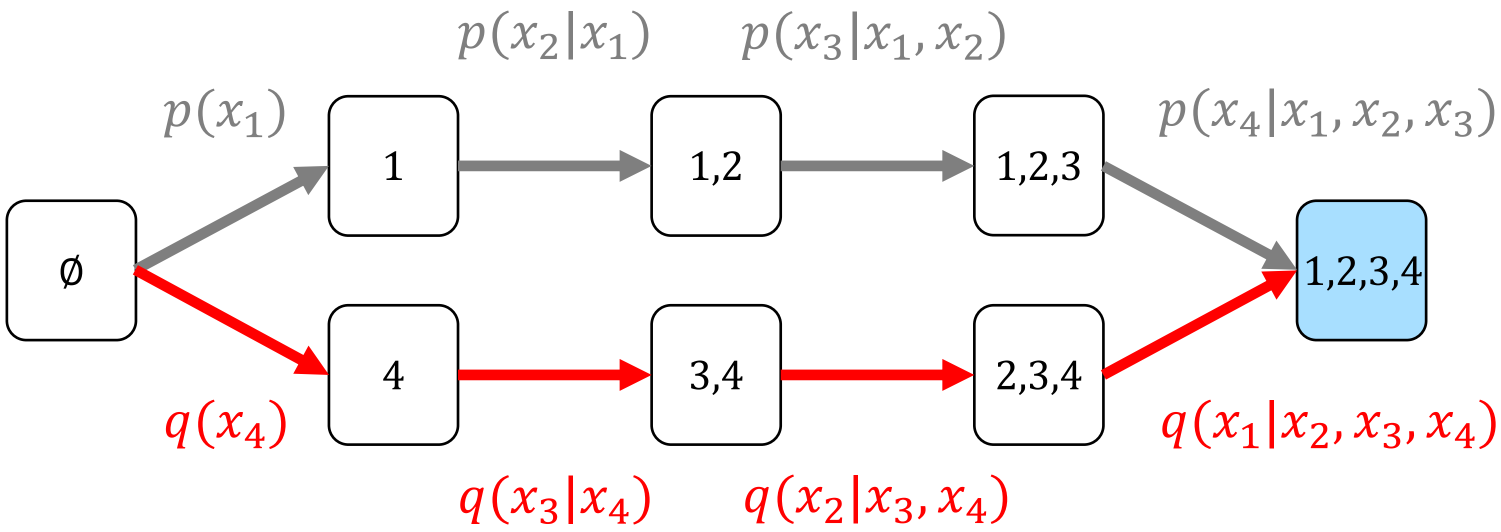
$e = \{1, 2, 4\}$ $\sigma = 1, 2, 4, 3$

Any-Order Autoregressive Model

Forward Ordering
1, 2, 3, 4

Reverse Ordering
4, 3, 2, 1

… every ordering!



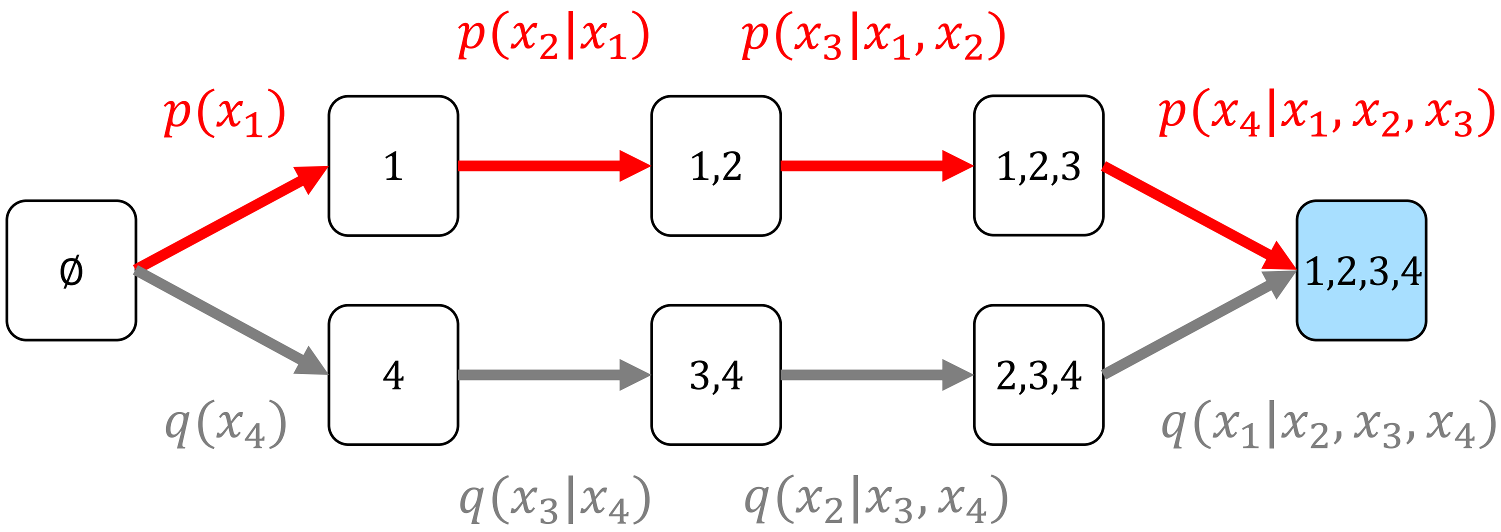
But… redundancy in our model

Any-Order Autoregressive Model

Forward Ordering
1, 2, 3, 4

Reverse Ordering
4, 3, 2, 1

… every ordering!



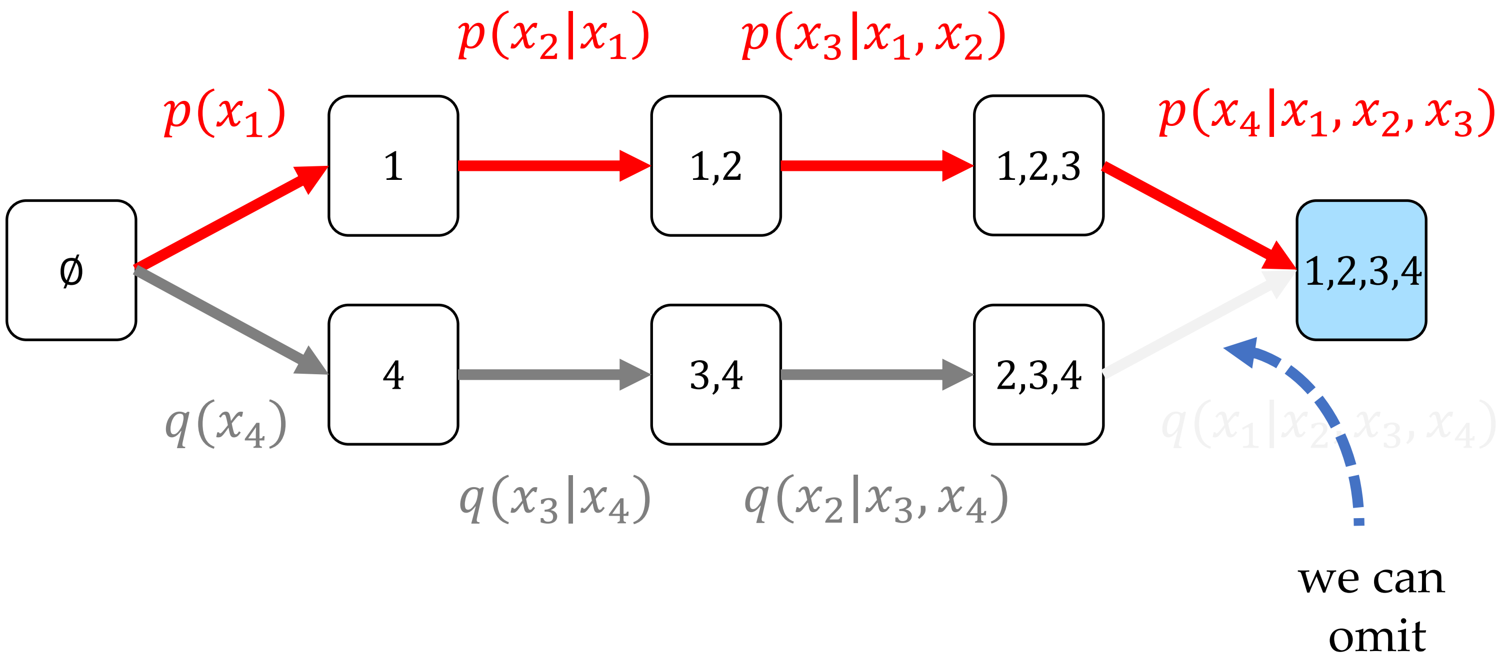
But… redundancy in our model

Any-Order Autoregressive Model

Forward Ordering
1, 2, 3, 4

Reverse Ordering
4, 3, 2, 1

… every ordering!



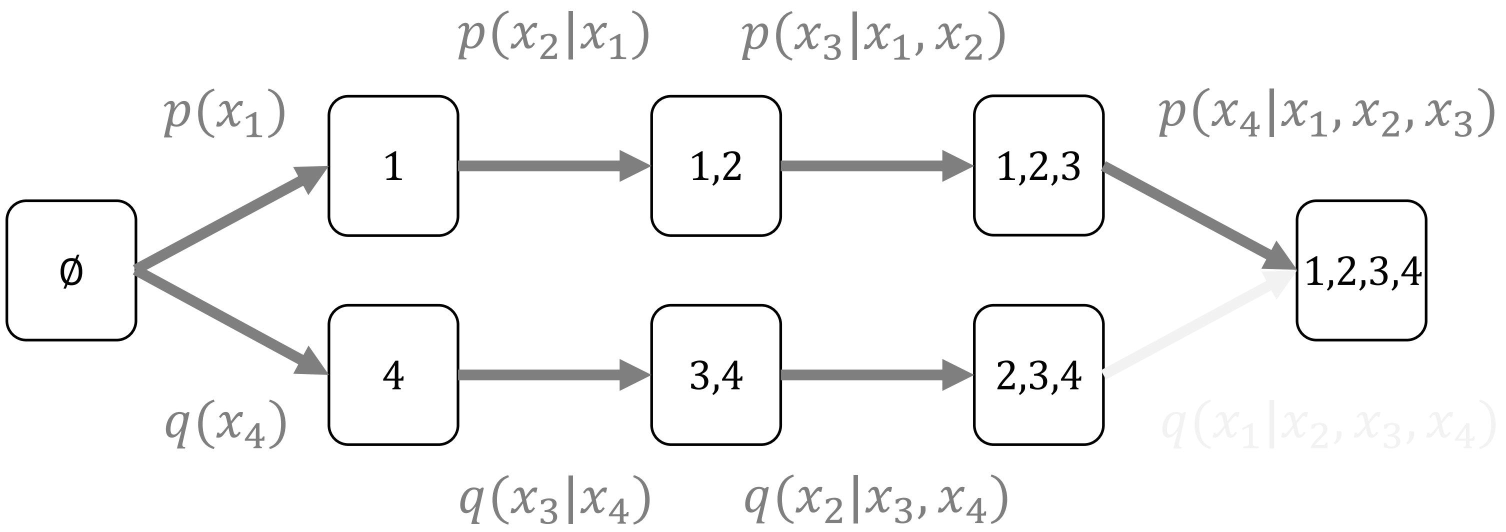
But… redundancy in our model

Any-Order Autoregressive Model

Forward Ordering
1, 2, 3, 4

Reverse Ordering
4, 3, 2, 1

… every ordering!



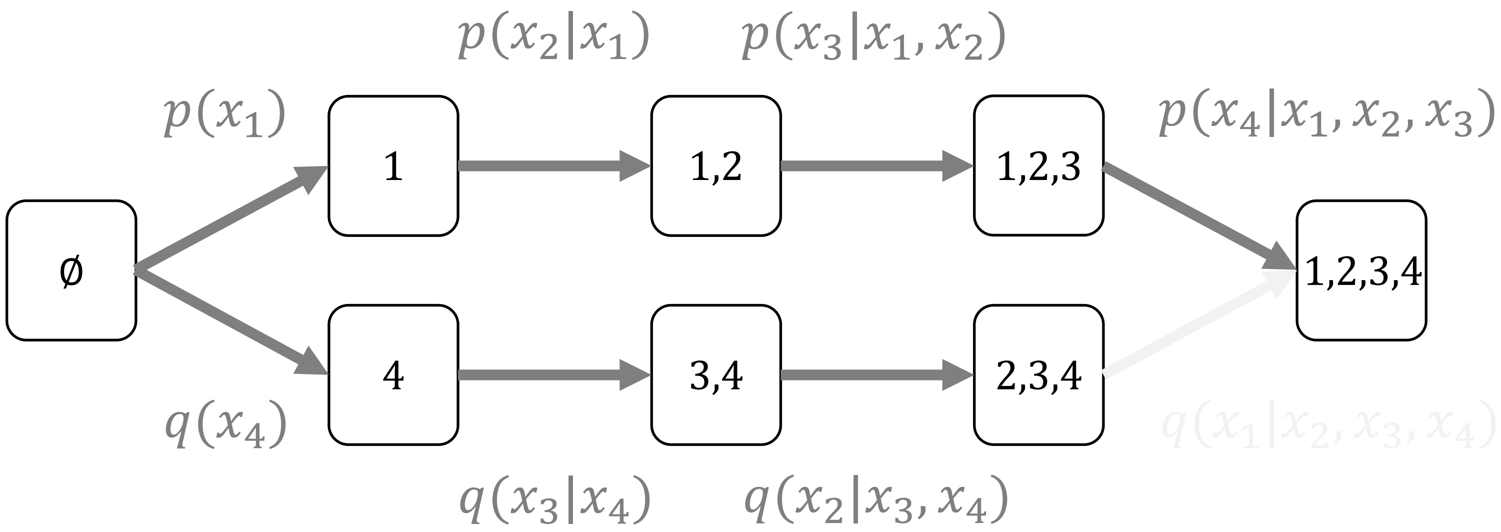
Less redundancy

Any-Order Autoregressive Model

Forward Ordering
1, 2, 3, 4

Reverse Ordering
4, 3, 2, 1

… every ordering!



Problem

How do we reduce redundancy when using all orders?

MAC: Mask-Tuned Arbitrary Conditional Model

MAC: Mask-Tuned Arbitrary Conditional Model

our proposal
for improving AO-ARMs

MAC: an improved version of AO-ARMs

We reduce redundancy, making learning easier.

MAC: an improved version of AO-ARMs

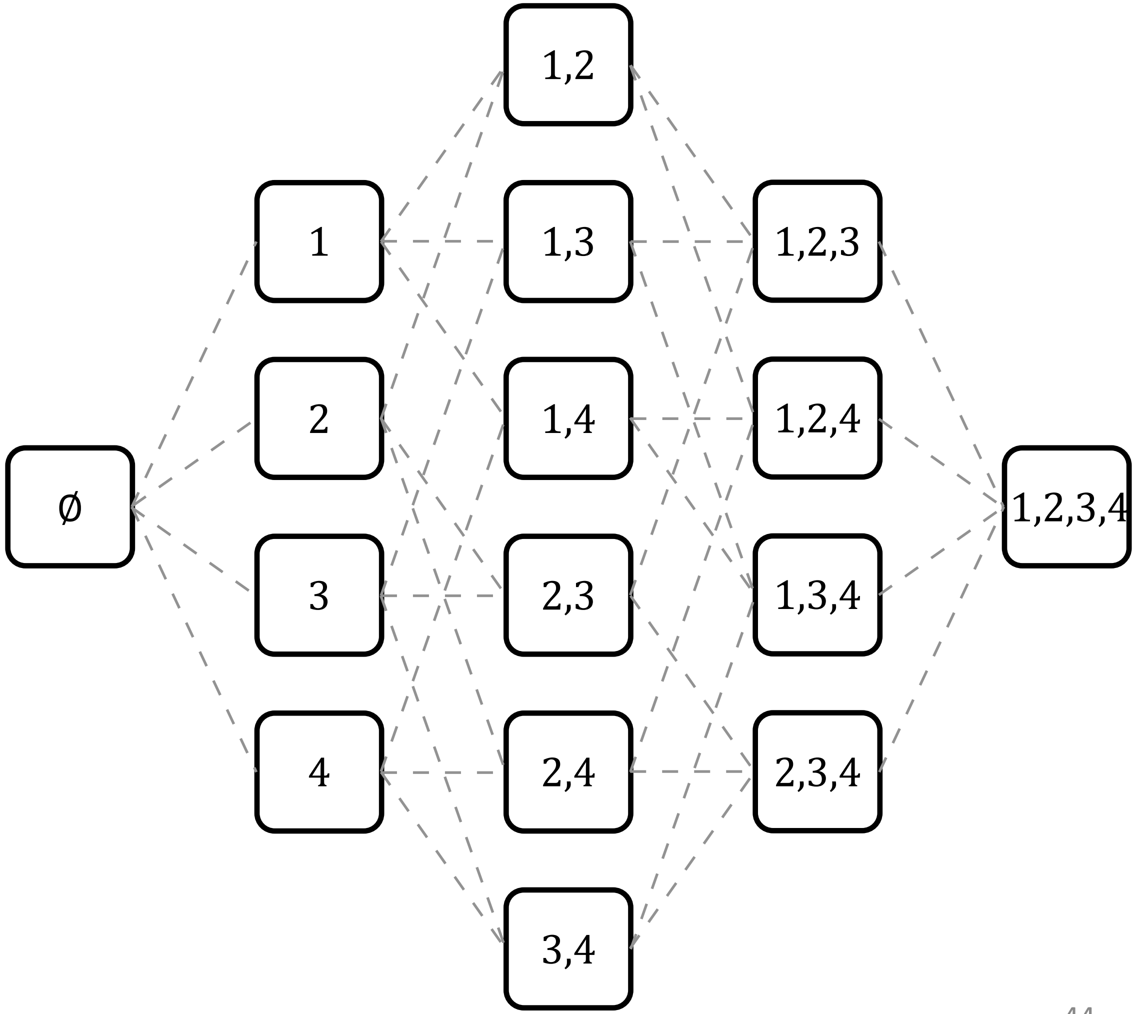
We reduce redundancy, making learning easier.

SOTA likelihoods among arbitrary conditional models!

Text8 dataset (bpd, lower is better)

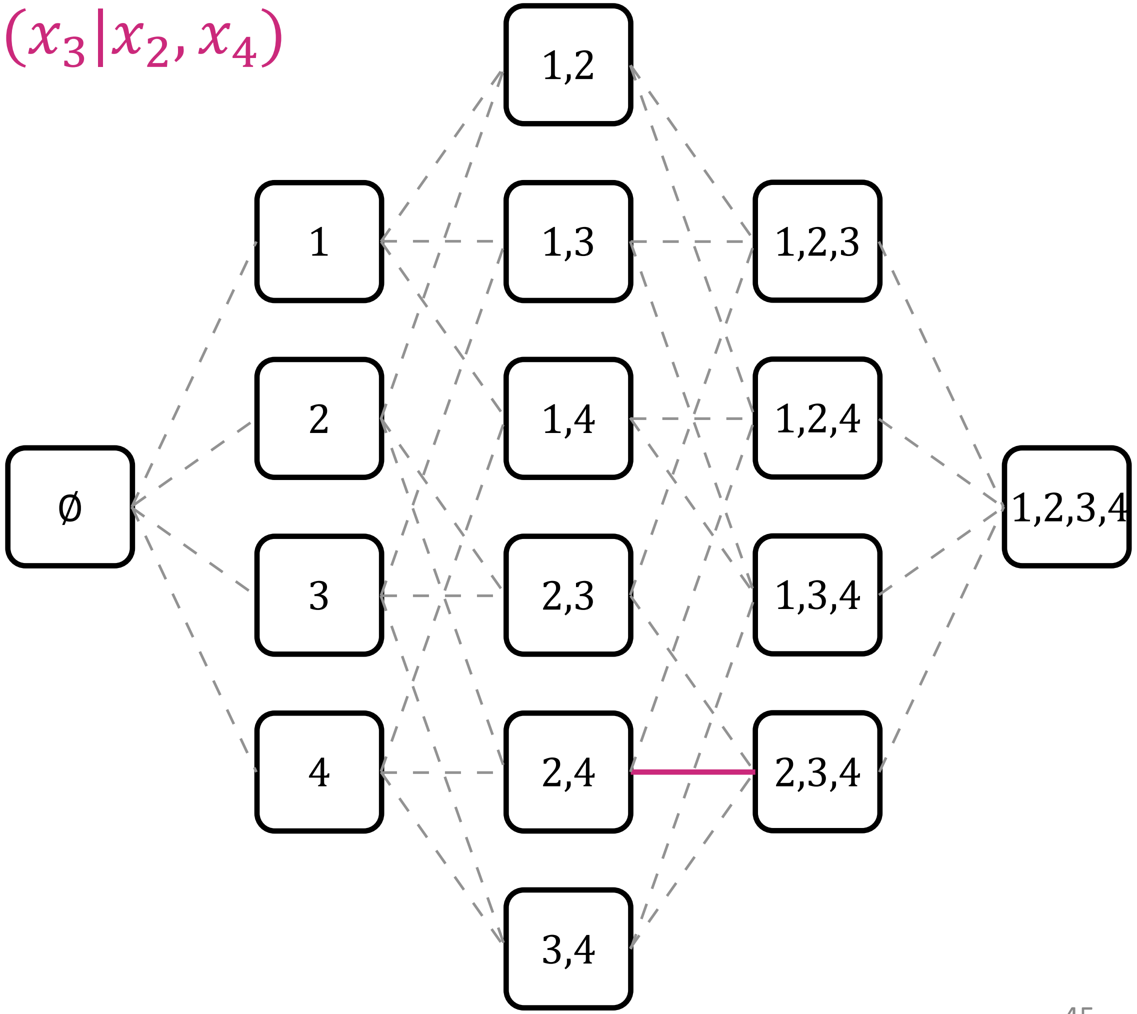
	joint	marginal
ARDM (3000 epochs)	1.48	1.12
MAC (3000 epochs)	1.40	1.09

AO-ARM as a Binary Lattice



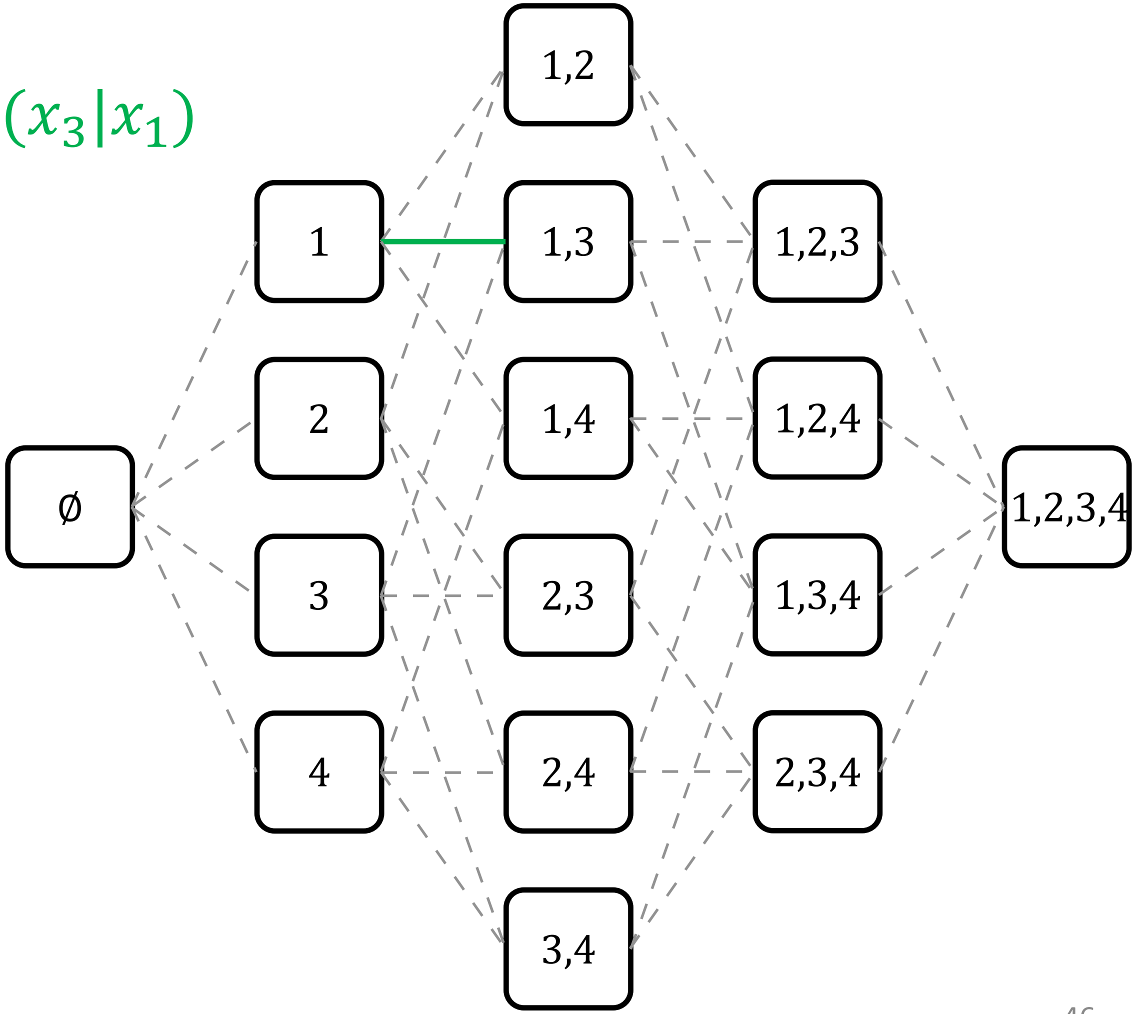
AO-ARM as a Binary Lattice

$p(x_3|x_2, x_4)$

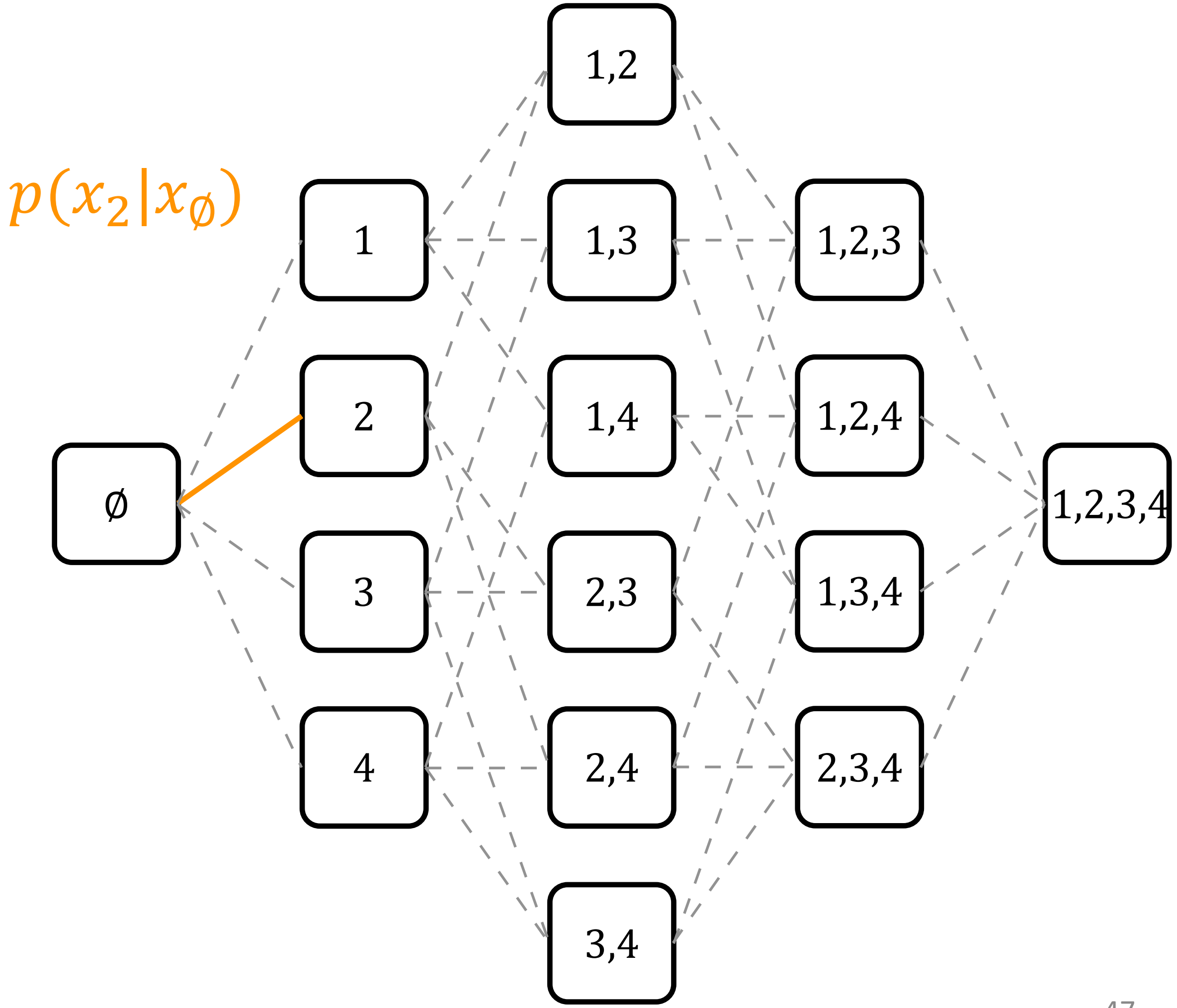


AO-ARM as a Binary Lattice

$p(x_3|x_1)$



AO-ARM as a Binary Lattice



Prior Work's Training Routine

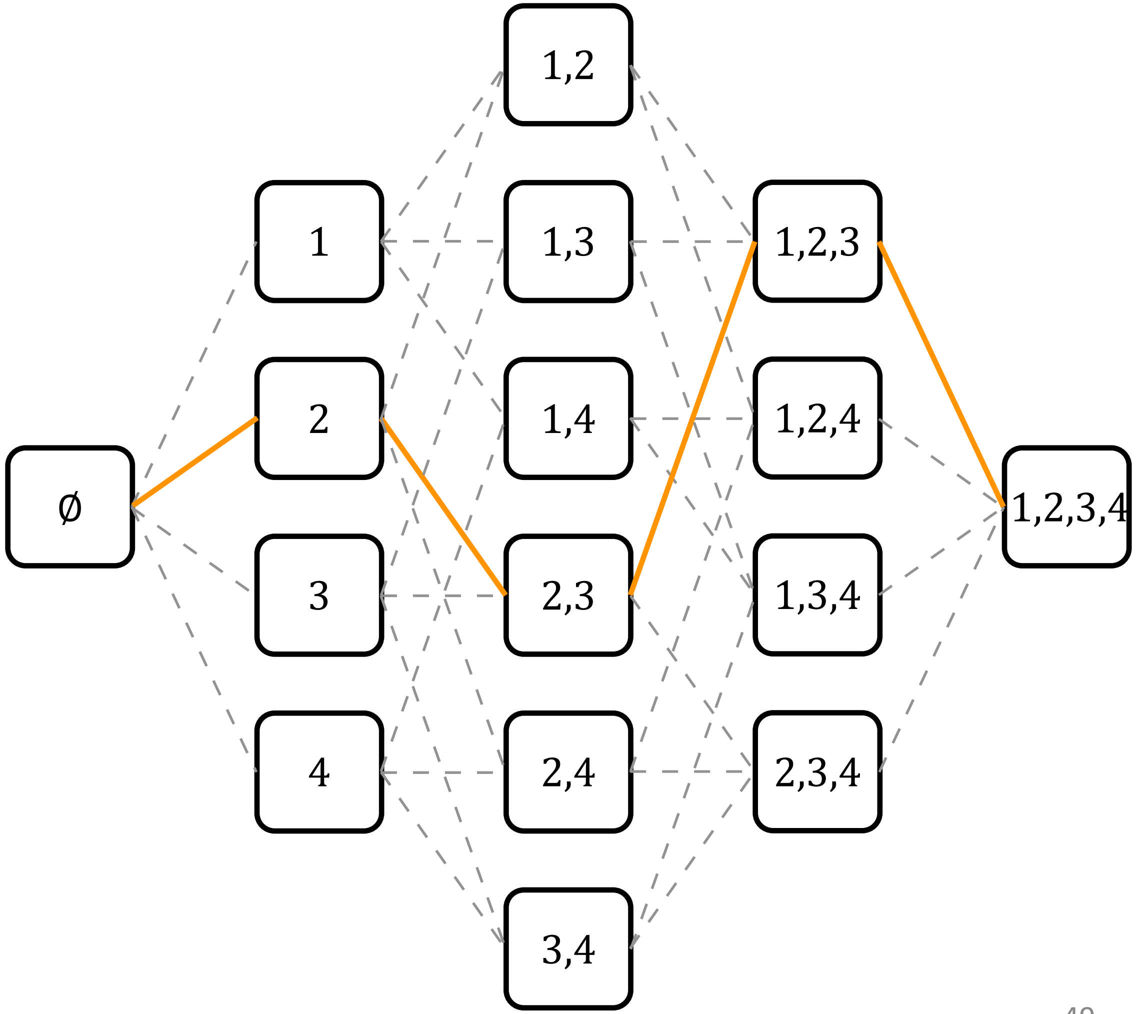
$\log p(\boldsymbol{x})$

joint

1. sample an order

   2, 3, 1, 4

2. then train (maximize log-ll)



Prior Work's Training Routine

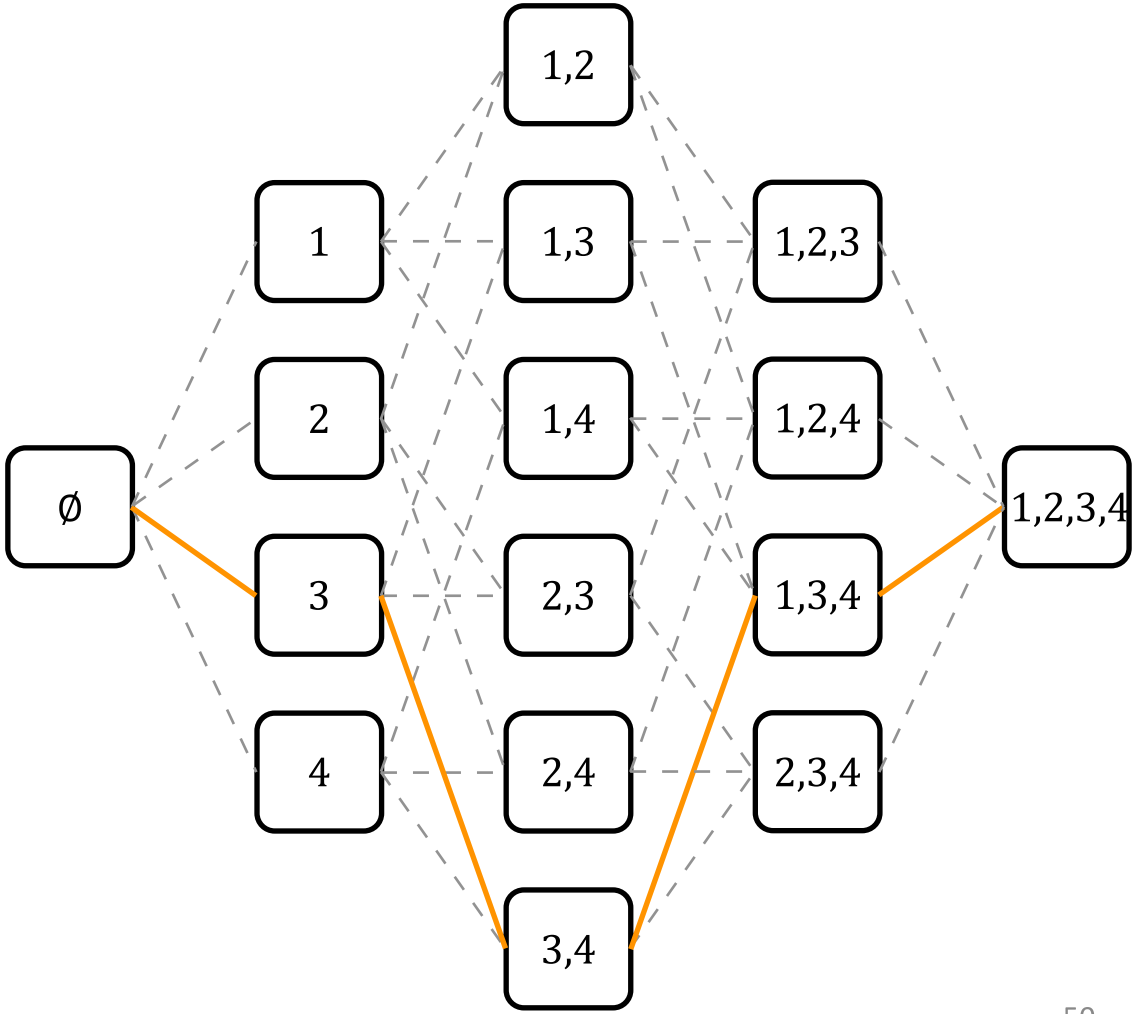
$\log p(\boldsymbol{x})$

joint

1. sample an order

   3, 4, 1, 2

2. then train (maximize log-ll)



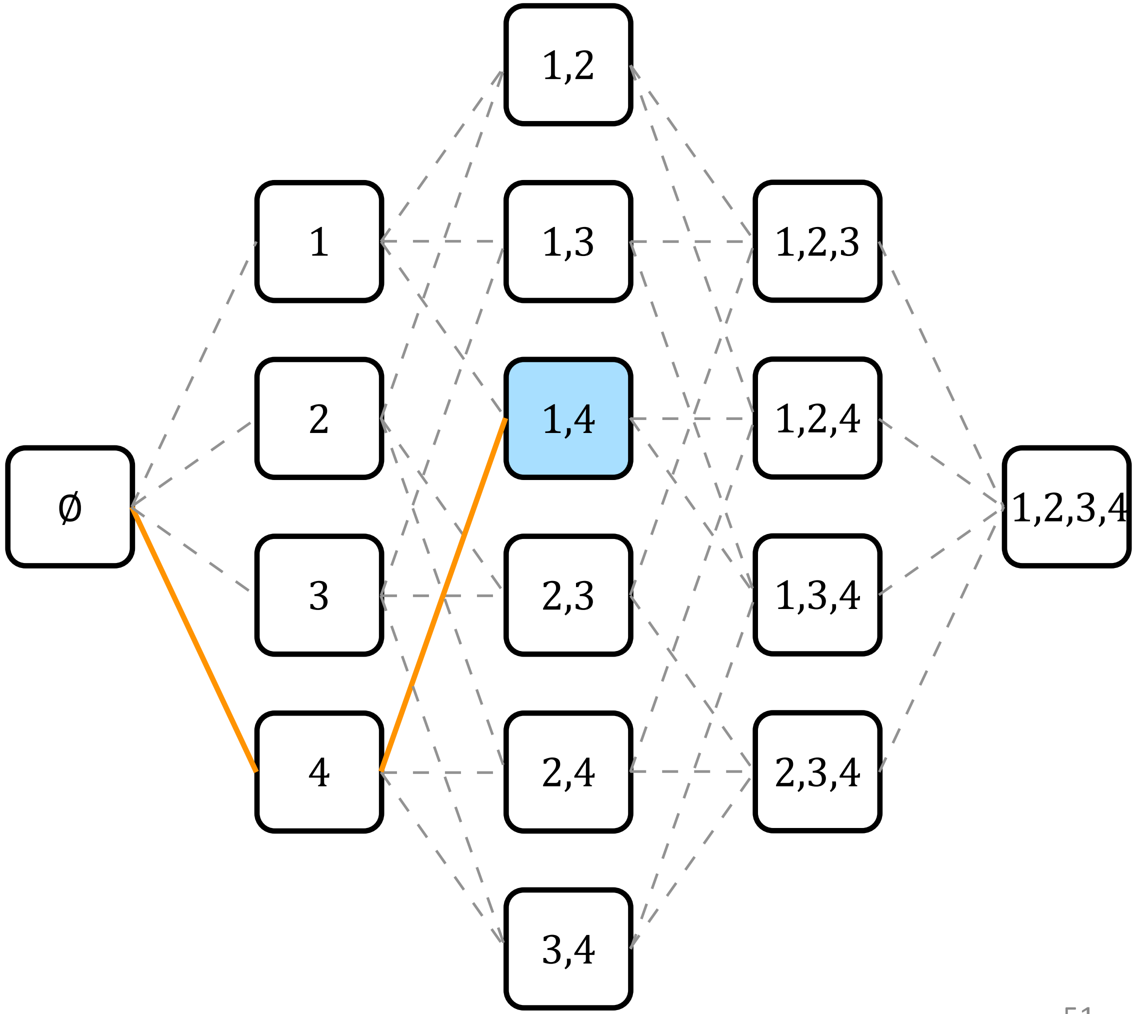
Prior Work's Inference Routine

$\log p(\boldsymbol{x}_e)$

1. sample a compatible order

   4, 1

2. then evaluate



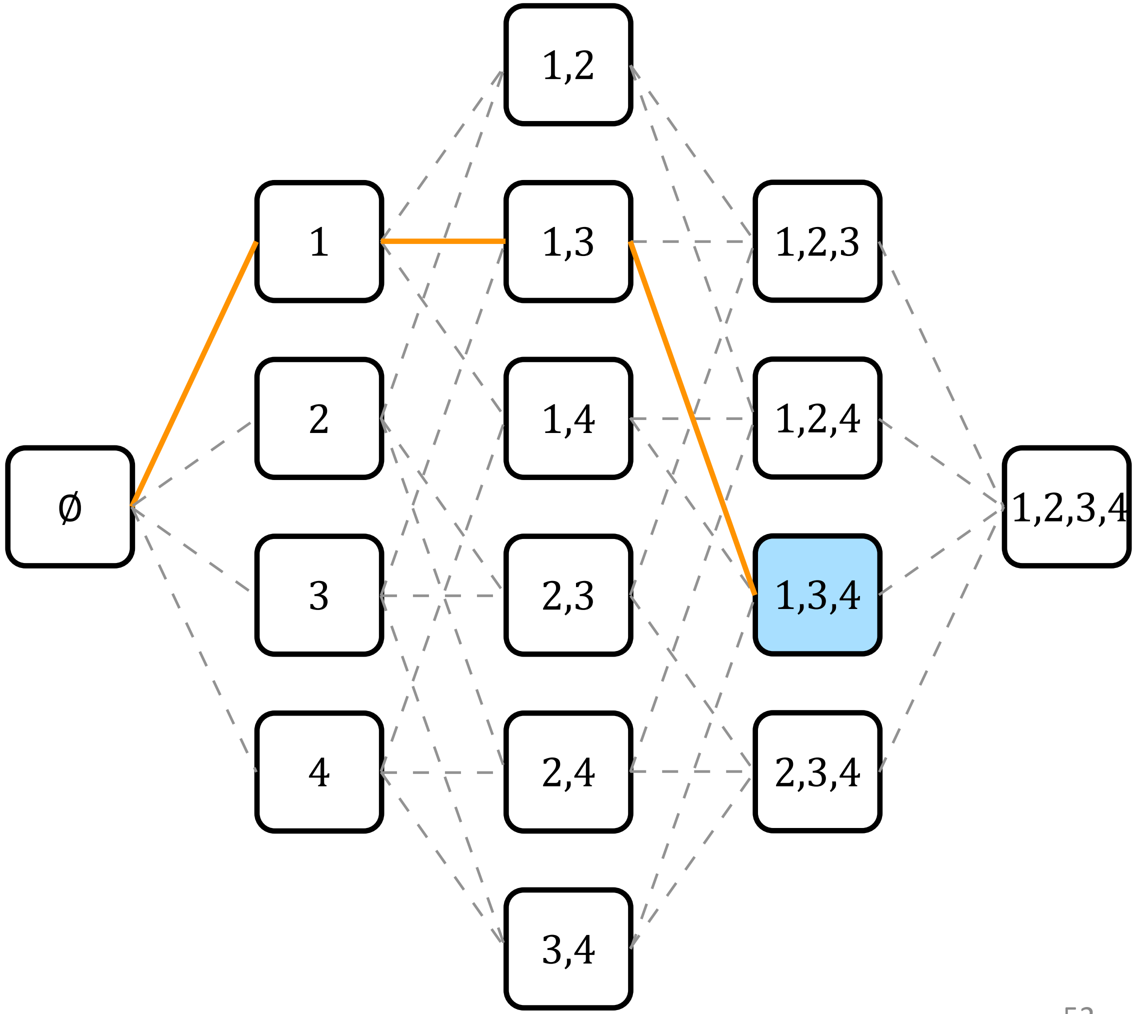
Prior Work's Inference Routine

$\log p(\boldsymbol{x}_e)$

1. sample a compatible order

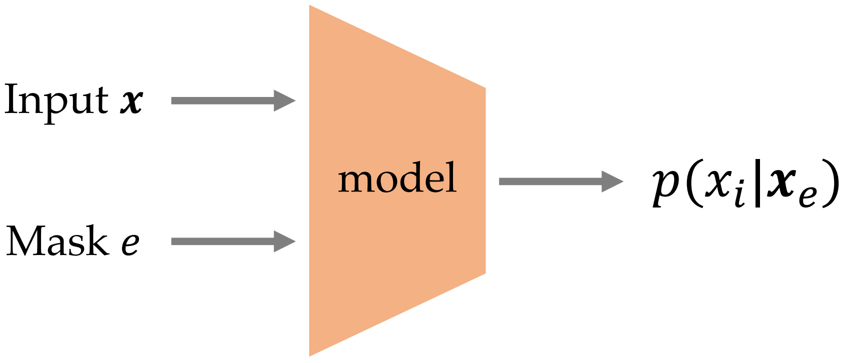
    1, 3, 4

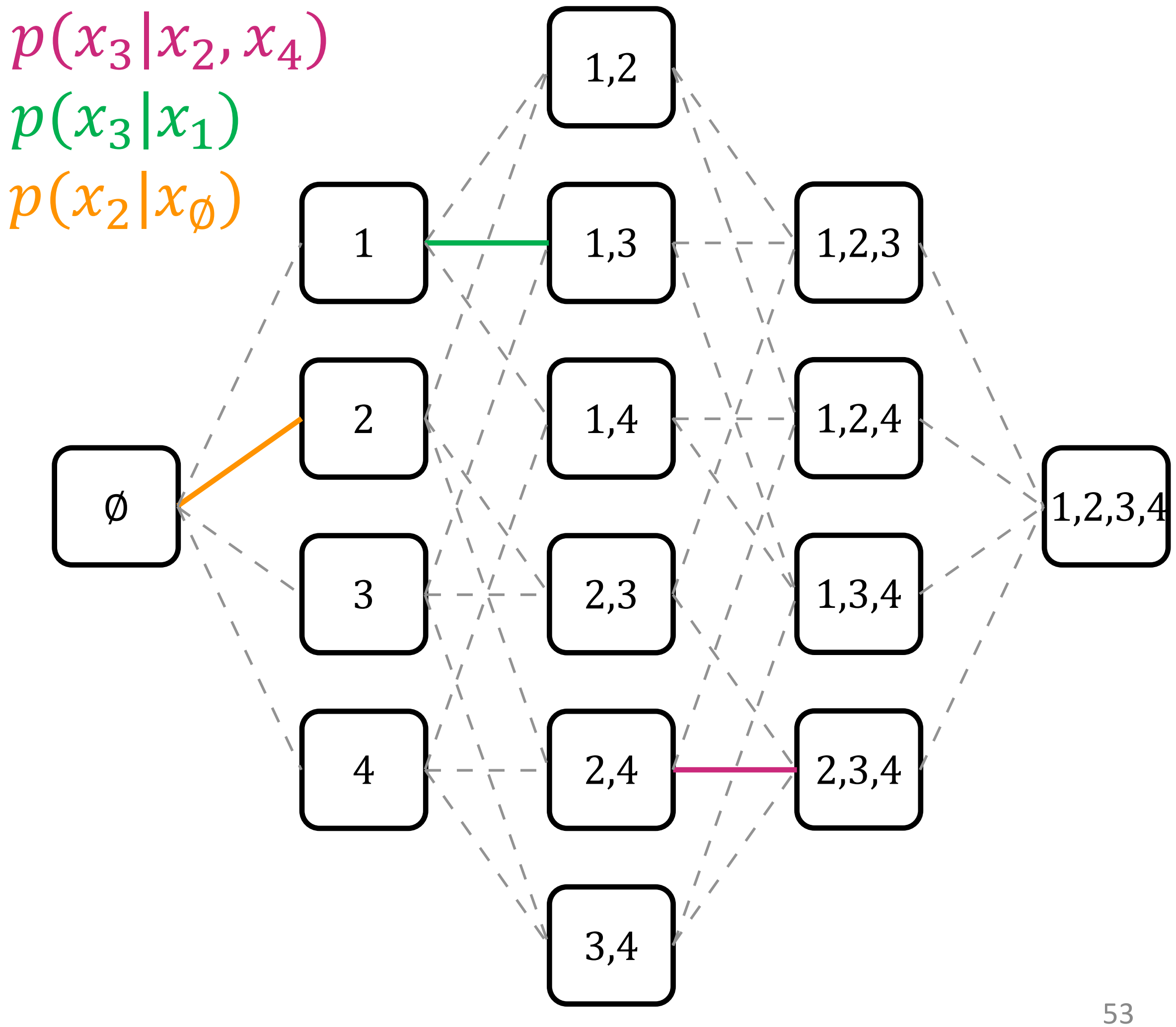
2. then evaluate



AO-ARM as a Binary Lattice

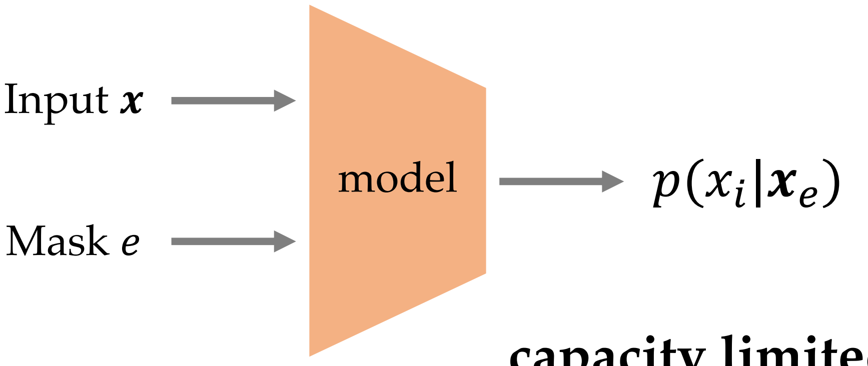
All univariate conditionals (edges) learned with a **single** weight-tied neural network



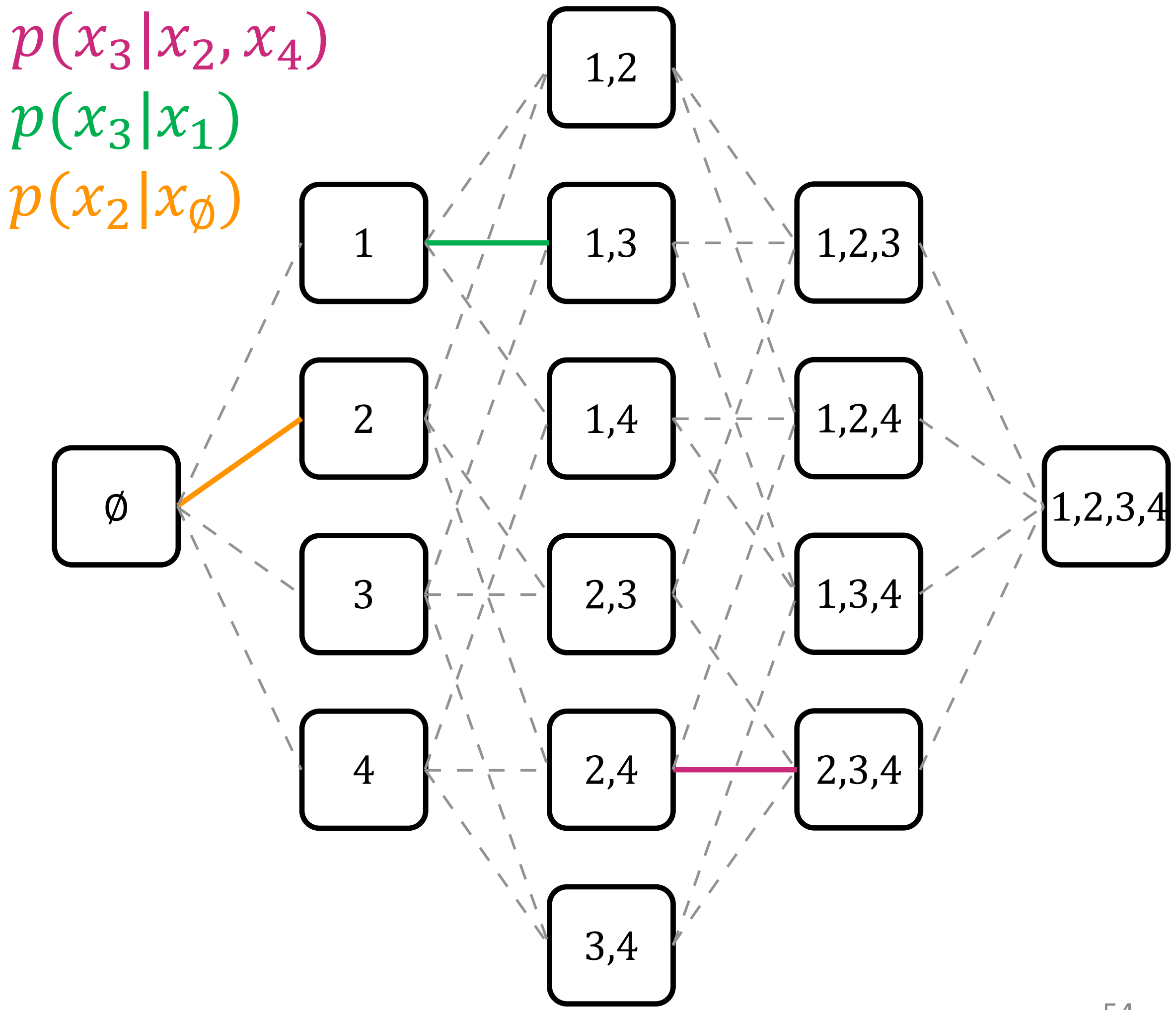


AO-ARM as a Binary Lattice

All univariate conditionals (edges) learned with a **single** weight-tied neural network

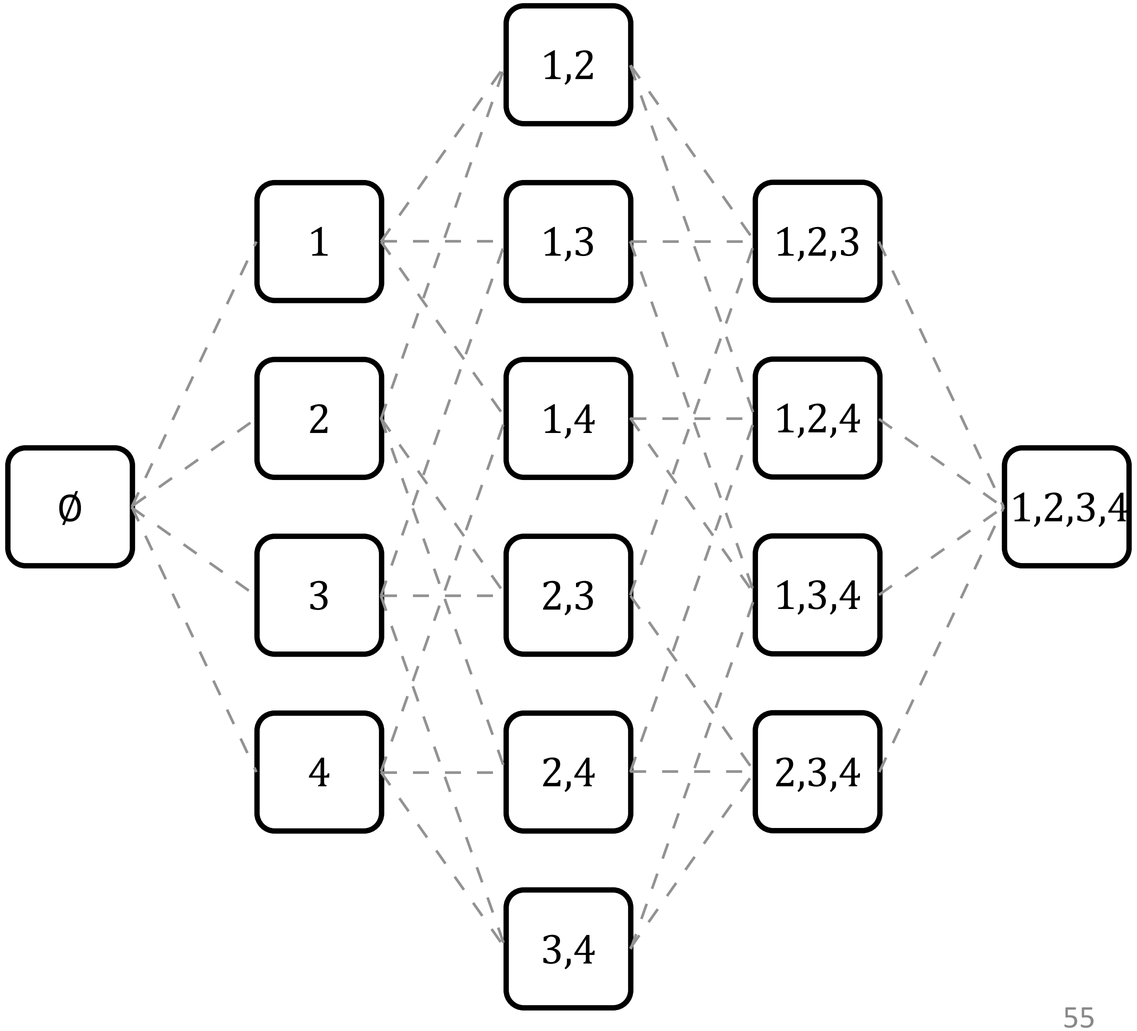


capacity limited!

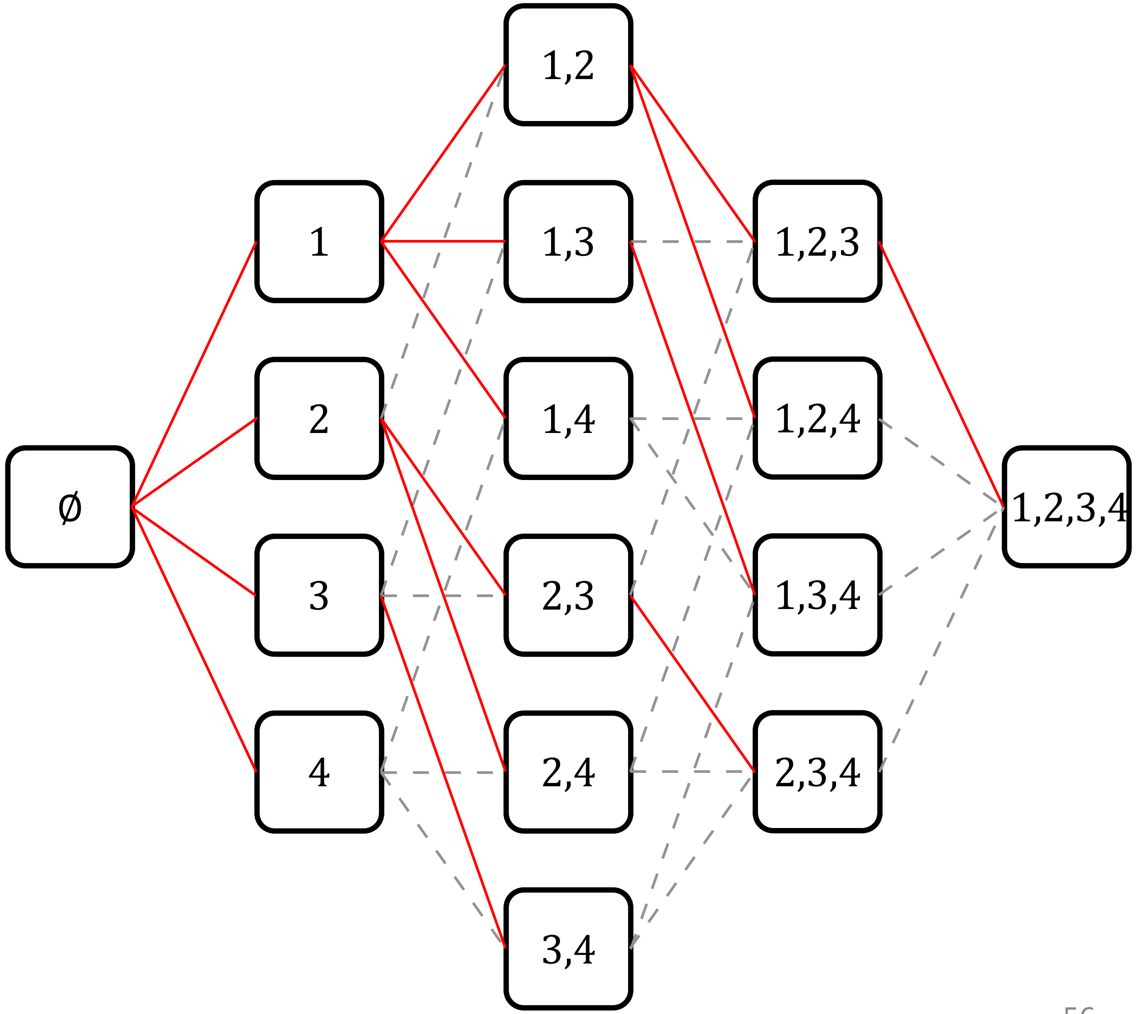


Improvement 1: Reduce Redundancy

For each node, we only need one path from ∅



Improvement 1: Reduce Redundancy

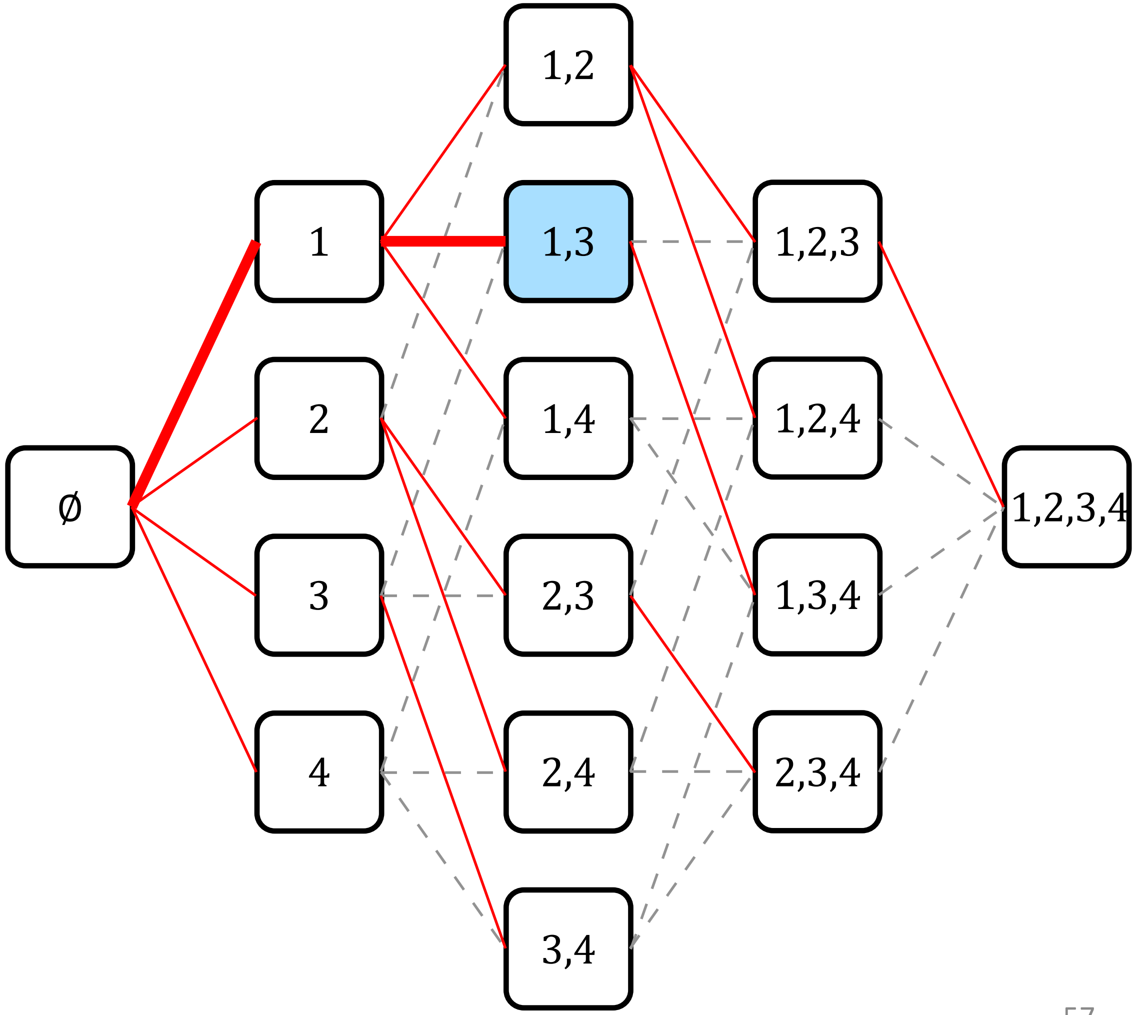


For each node, we only need one path from ∅

Only need the red edges!

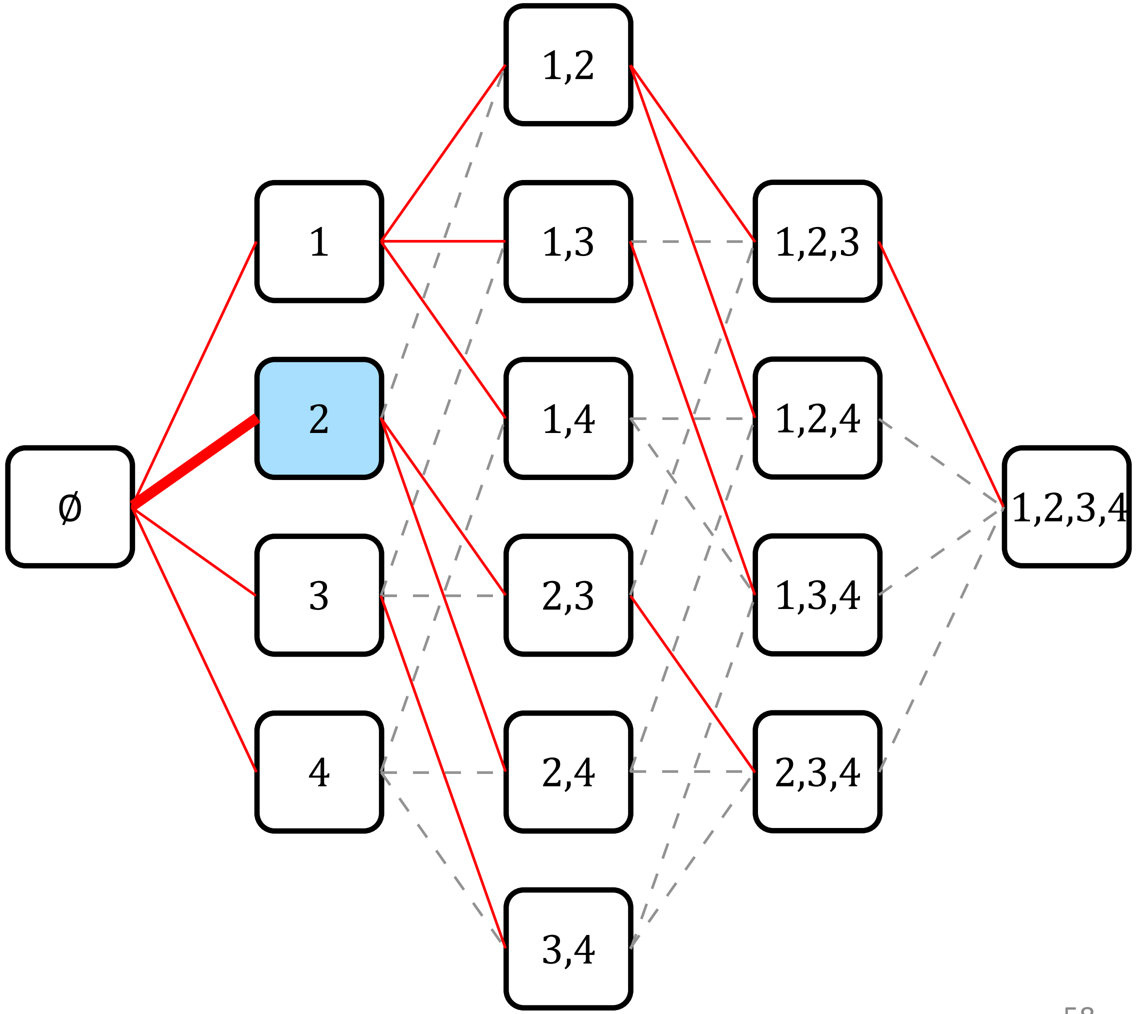
Improvement 2: Weight by Frequency

At inference time, we evaluate the red paths to answer queries



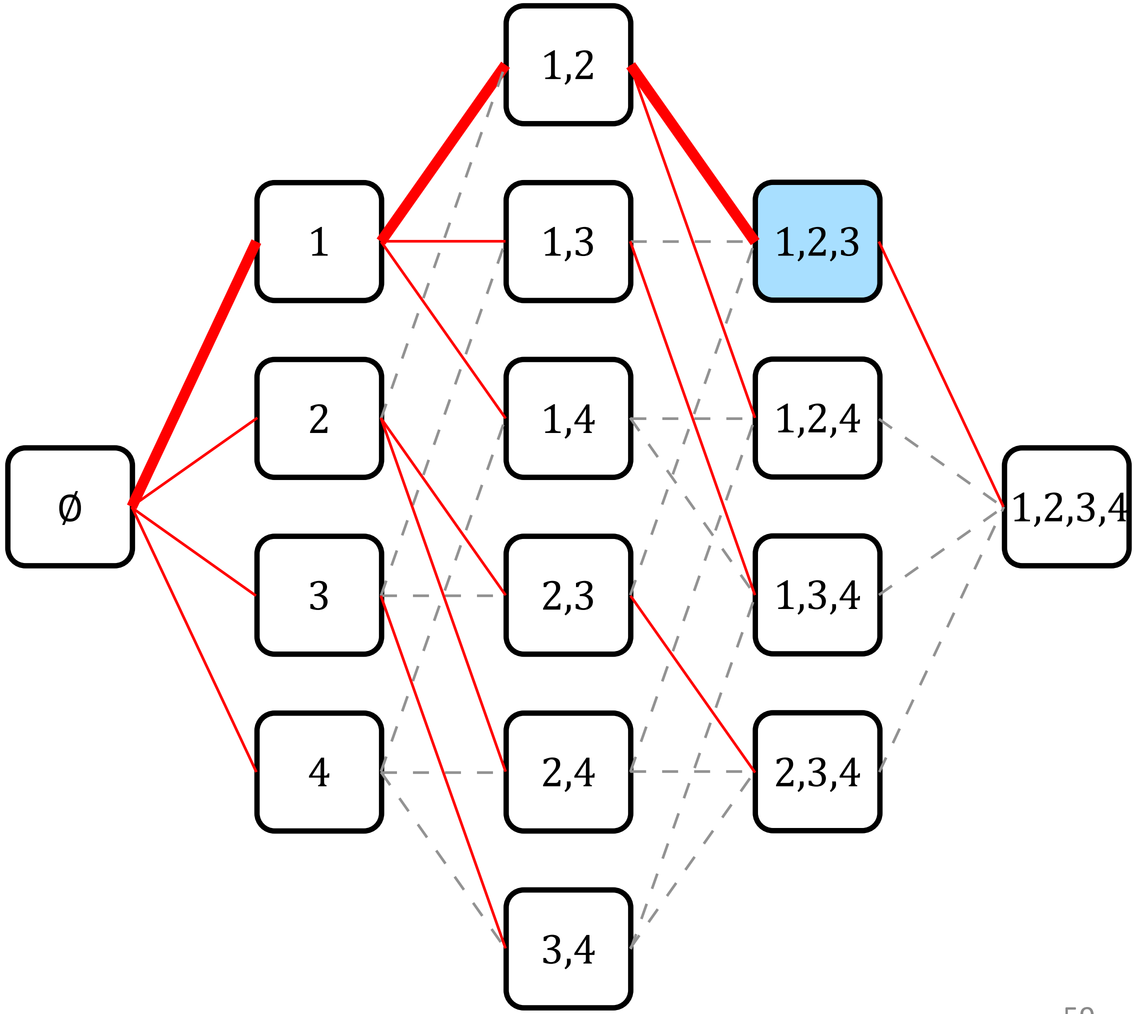
Improvement 2: Weight by Frequency

At inference time, we evaluate the red paths to answer queries



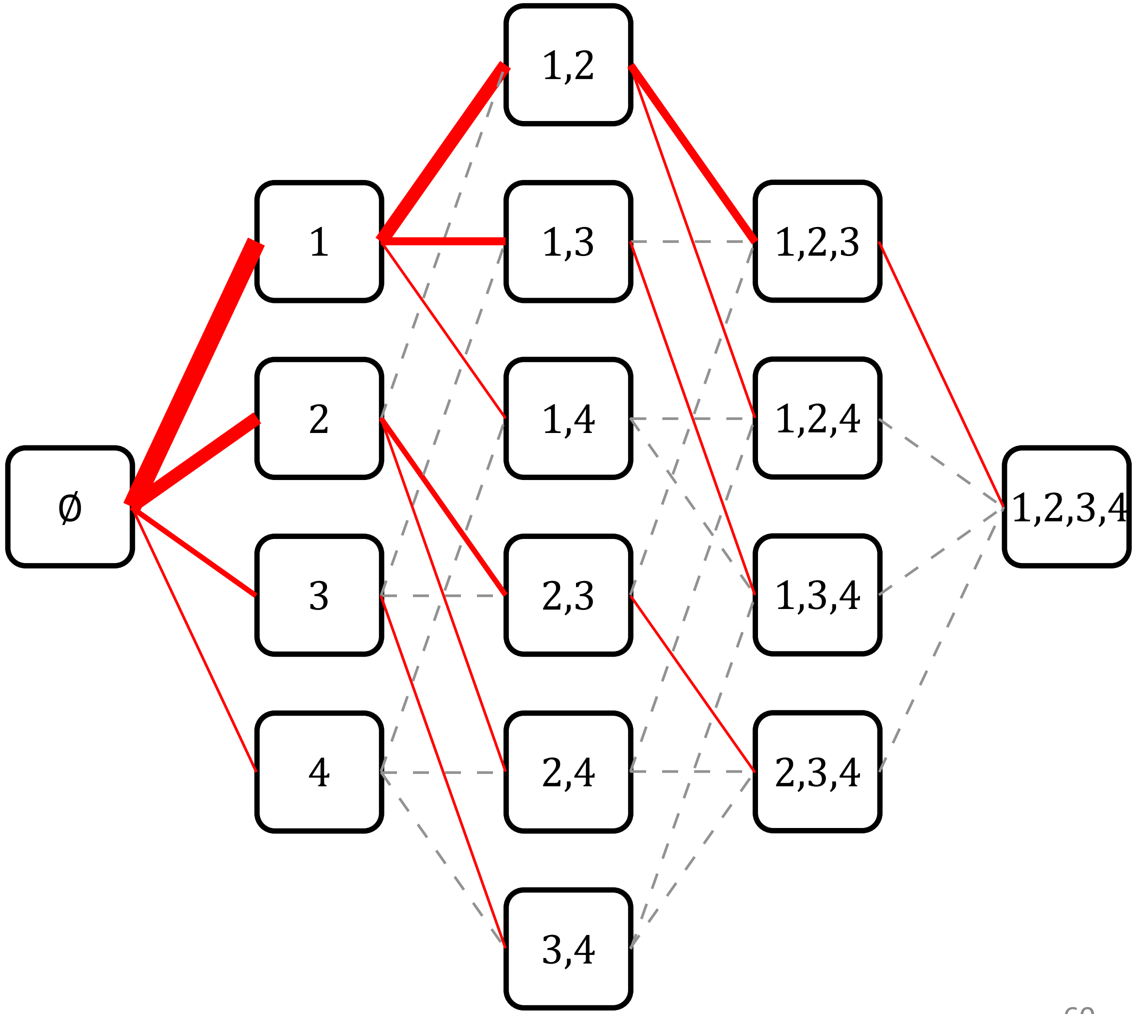
Improvement 2: Weight by Frequency

At inference time, we evaluate the red paths to answer queries



Improvement 2: Weight by Frequency

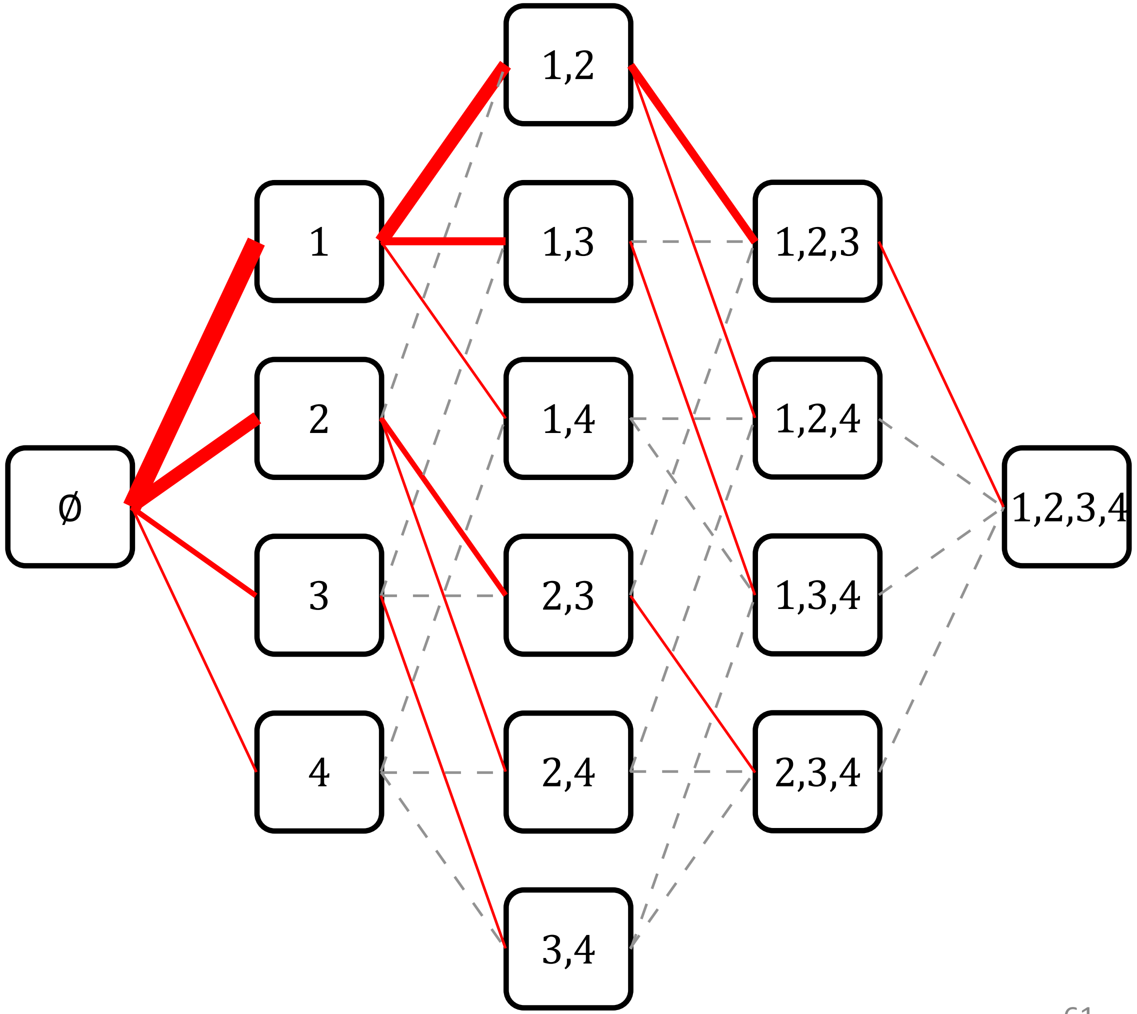
Some edges will be evaluated more frequently!



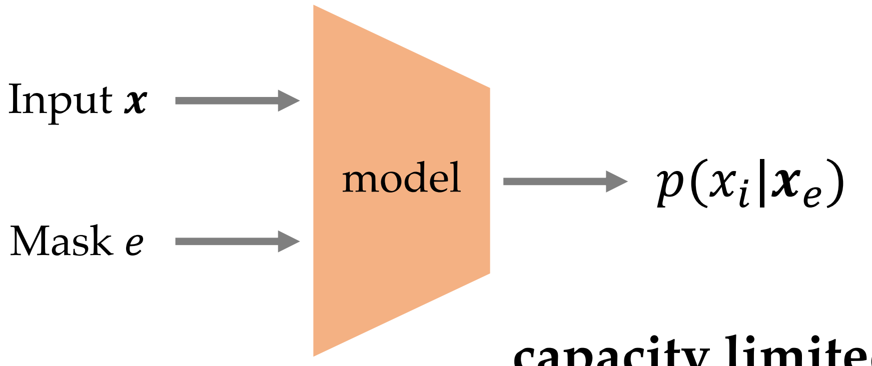
Improvement 2: Weight by Frequency

Some edges will be evaluated more frequently!

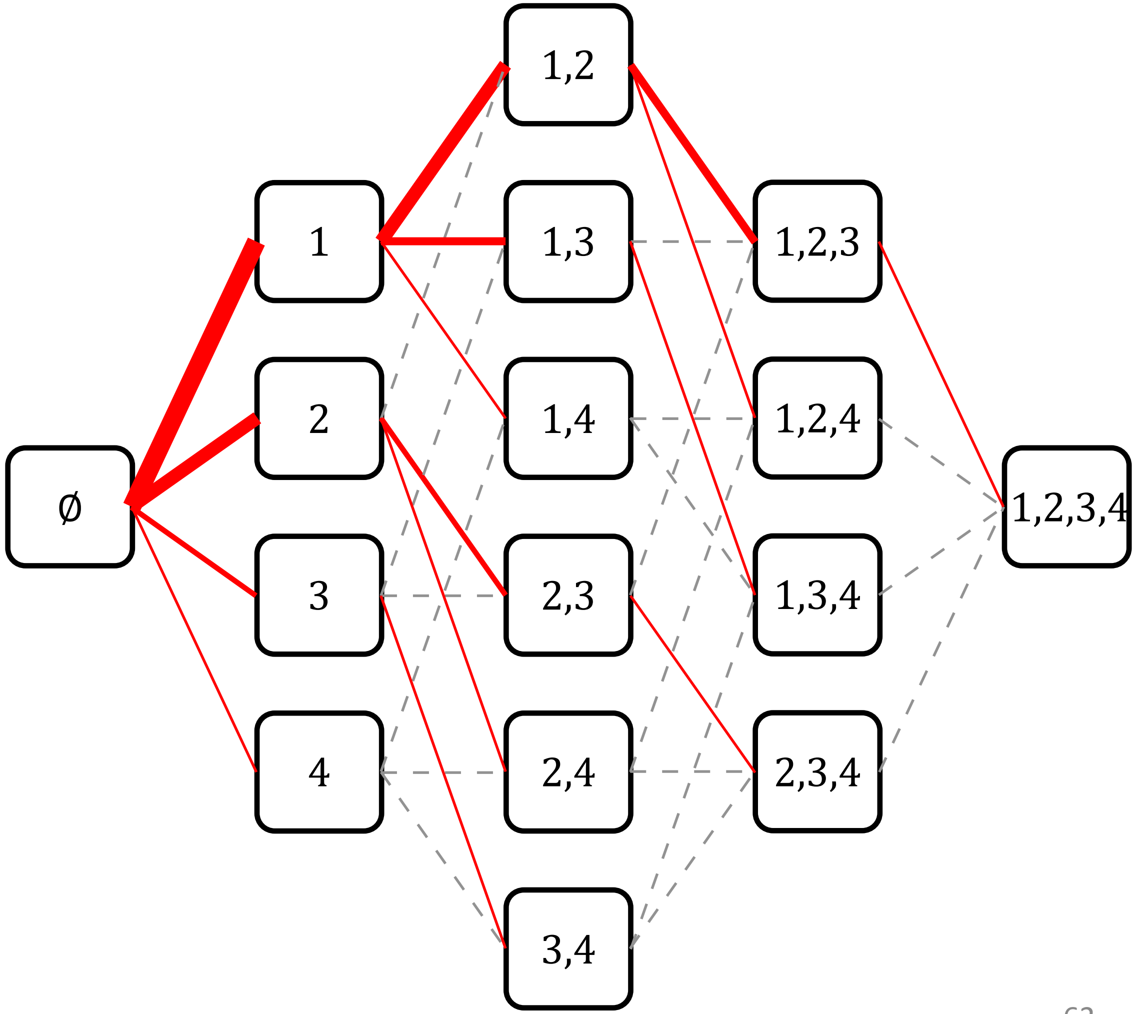
Thick edges carry "more descendants"



MAC

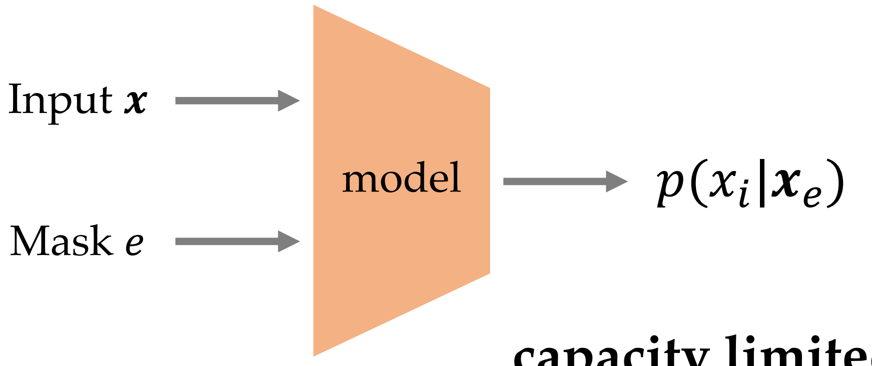


capacity limited!

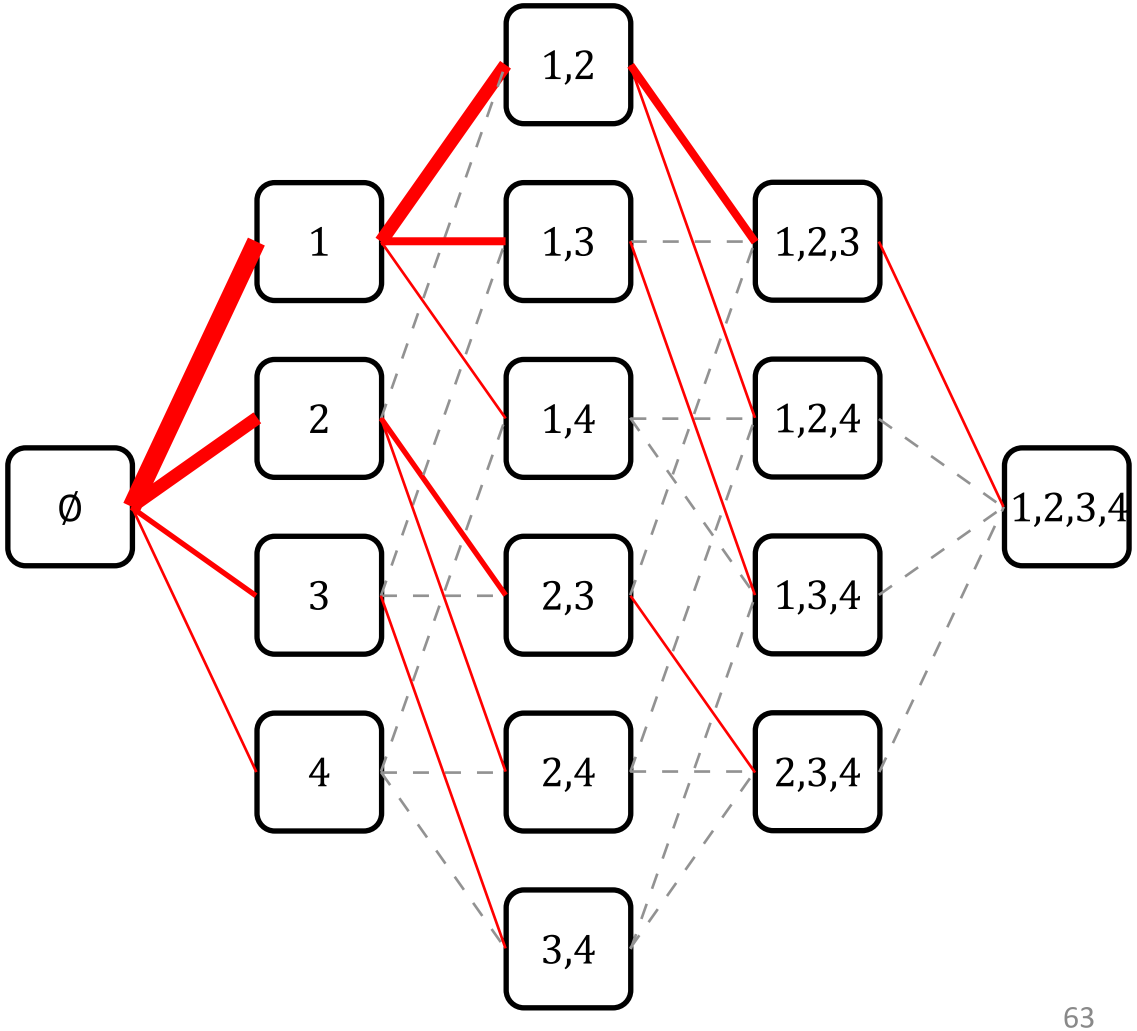


MAC

Reduces redundancy



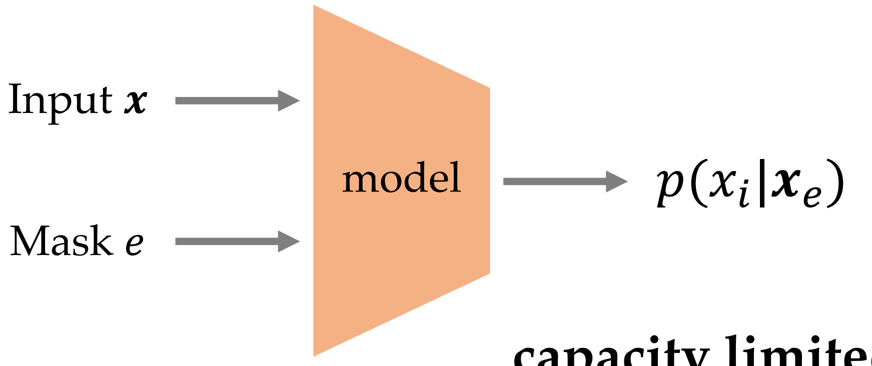
capacity limited!



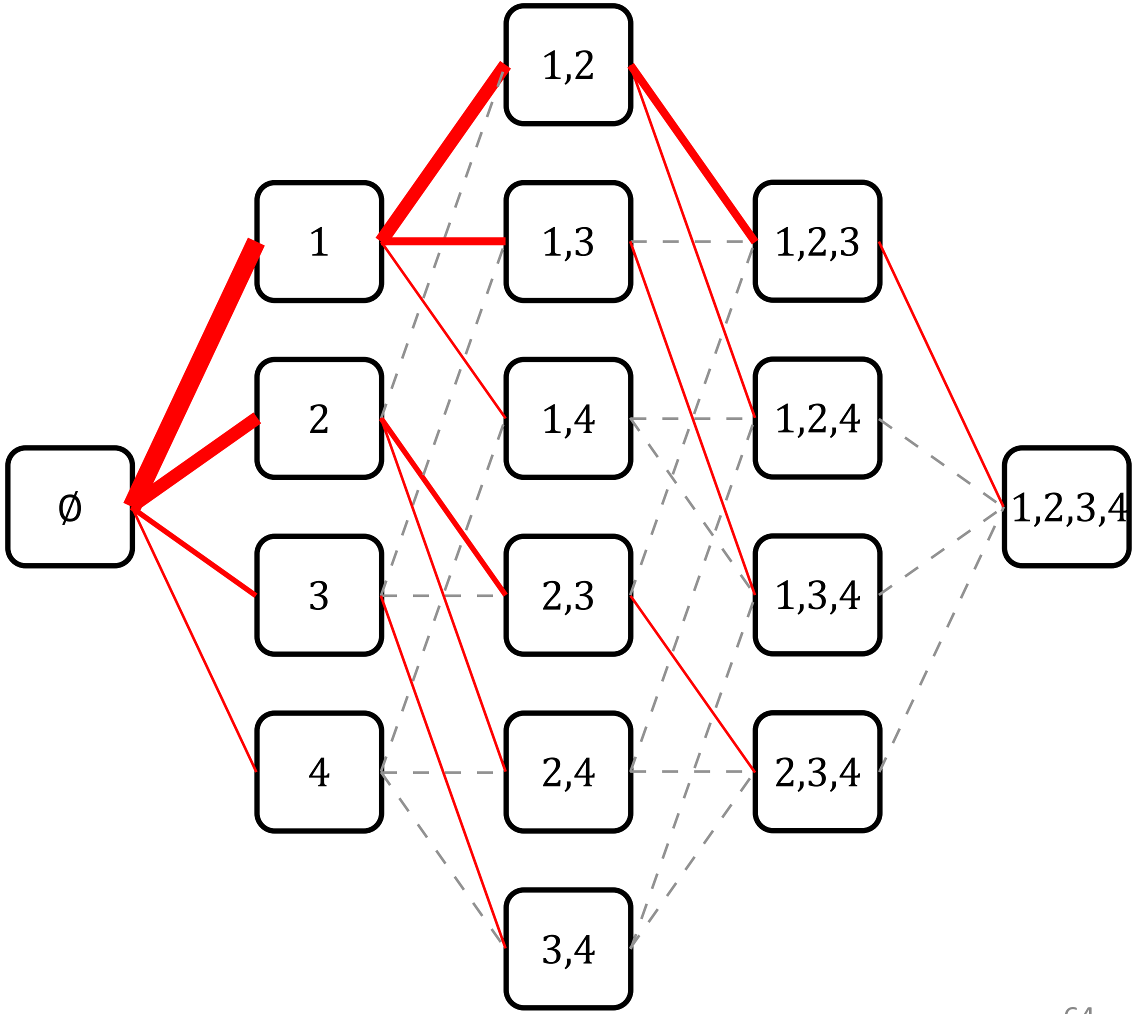
MAC

Reduces redundancy

Trains on "important" edges



capacity limited!

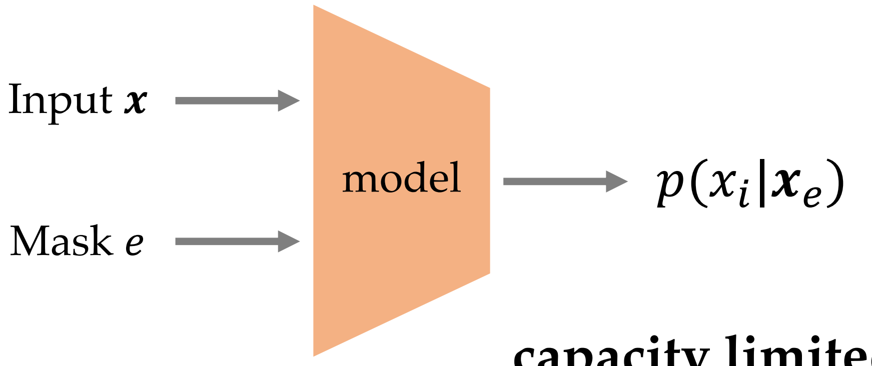


MAC

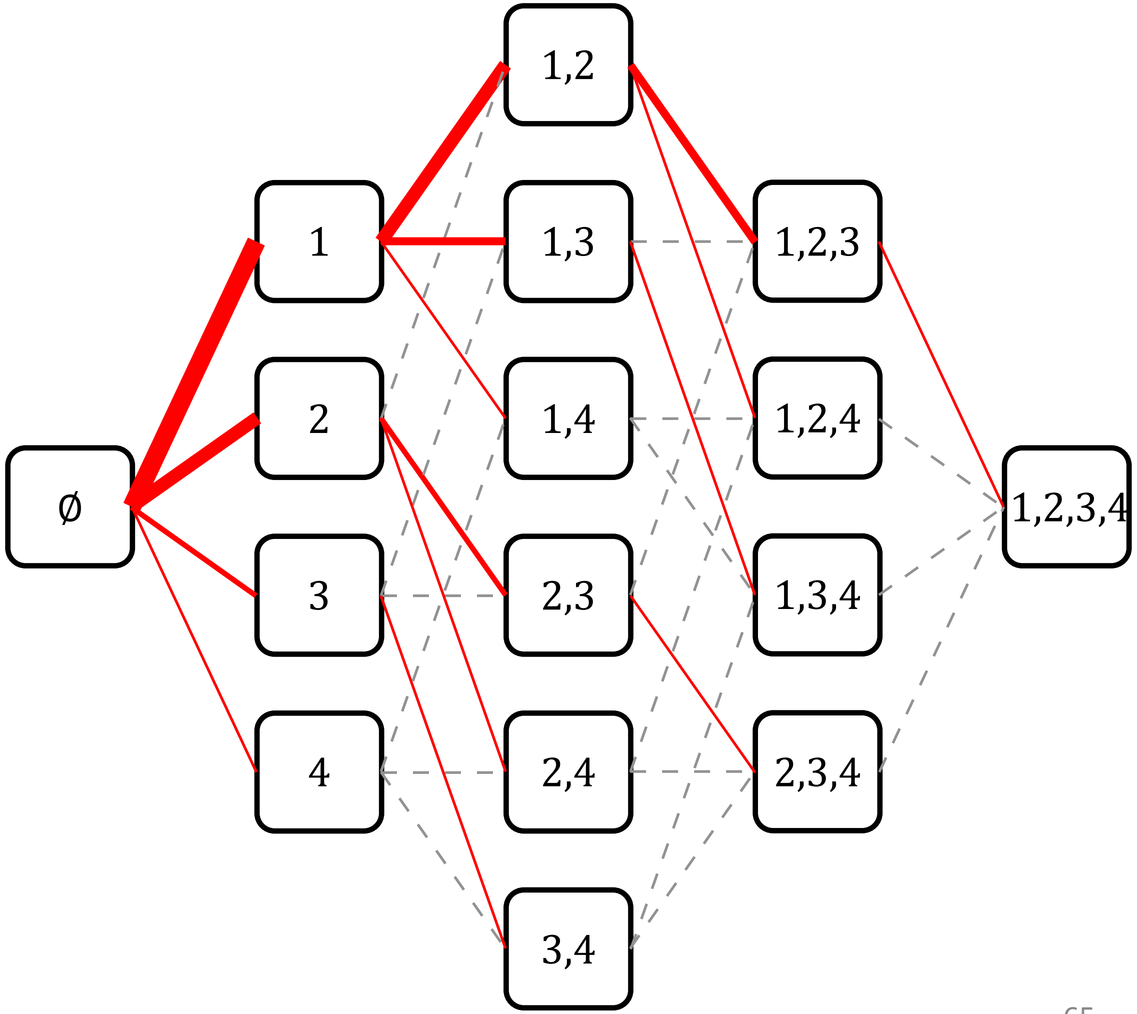
Reduces redundancy

Trains on "important" edges

Helps with limited capacity



capacity limited!

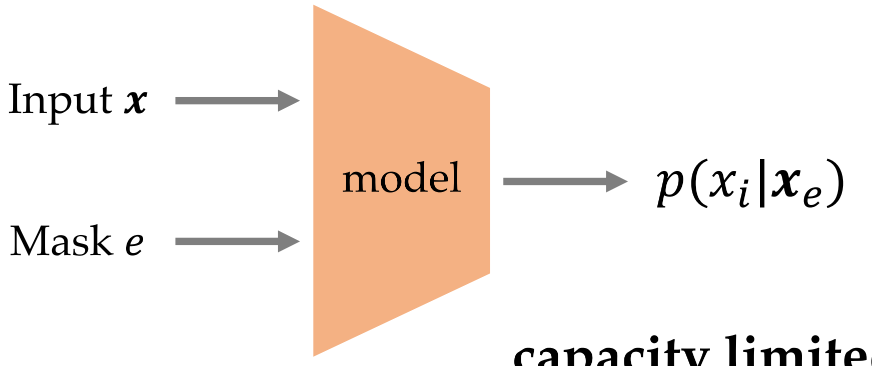


MAC

Reduces redundancy

Trains on "important" edges

Helps with limited capacity



capacity limited!

previous objective

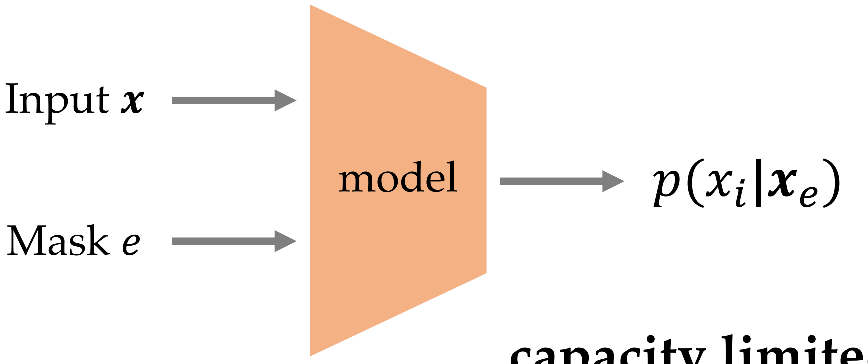
$$\log p(x_1|x_2)$$
$$\log p(x_1|x_3, x_4)$$
$$\log p(x_2|x_1)$$
$$\log p(x_2|x_1, x_4)$$
$$\log p(x_3|x_1, x_2)$$
$$\log p(x_3|x_1, x_2, x_4)$$
$$\log p(x_4)$$
$$\log p(x_4|x_3)$$
$$\ldots$$

MAC

Reduces redundancy

Trains on "important" edges

Helps with limited capacity



capacity limited!

new objective

~~$\log p(x_1|x_2)$~~

~~$\log p(x_1|x_3, x_4)$~~

$\log p(x_2|x_1)$

~~$\log p(x_2|x_1, x_4)$~~

$\log p(x_3|x_1, x_2)$

~~$\log p(x_3|x_1, x_2, x_4)$~~

$\log p(x_4)$

$\log p(x_4|x_3)$

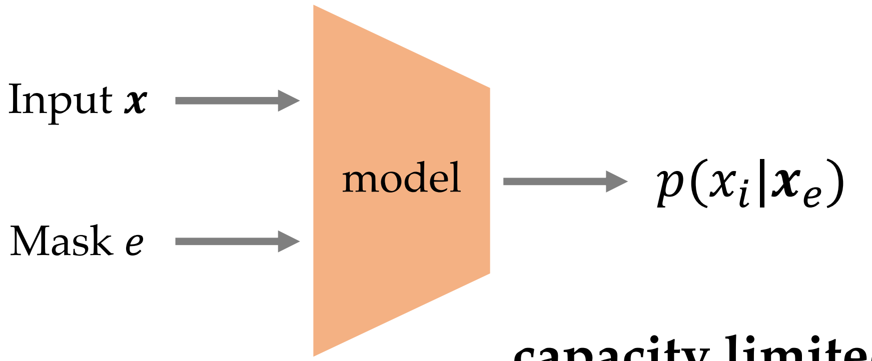
$\dots$

MAC

Reduces redundancy

Trains on "important" edges

Helps with limited capacity



capacity limited!

new objective

~~$\log p(x_1|x_2)$~~

~~$\log p(x_1|x_3, x_4)$~~

$\alpha_1 \quad \log p(x_2|x_1)$

~~$\log p(x_2|x_1, x_4)$~~

$\alpha_2 \quad \log p(x_3|x_1, x_2)$

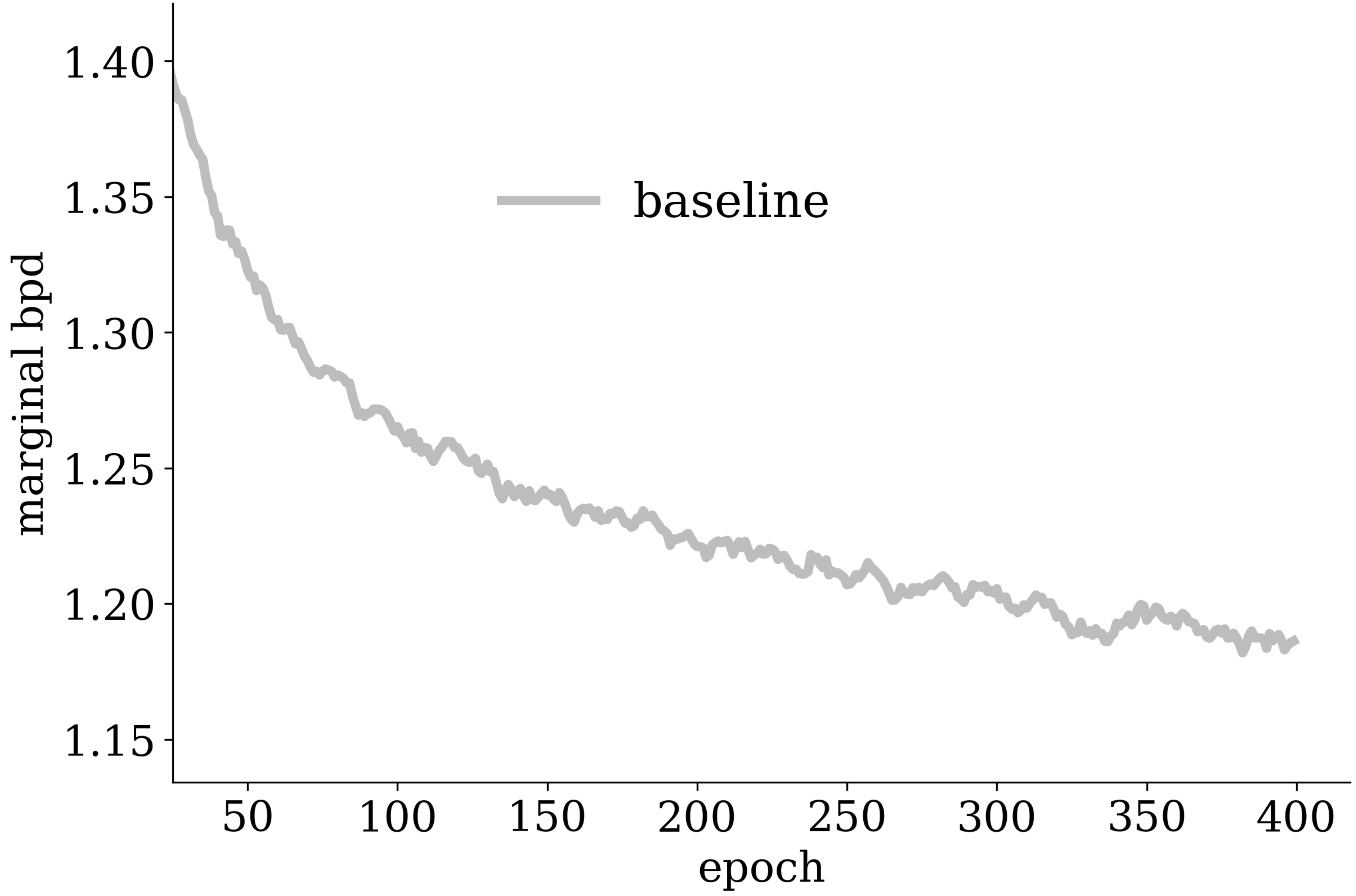
~~$\log p(x_3|x_1, x_2, x_4)$~~

$\alpha_3 \quad \log p(x_4)$

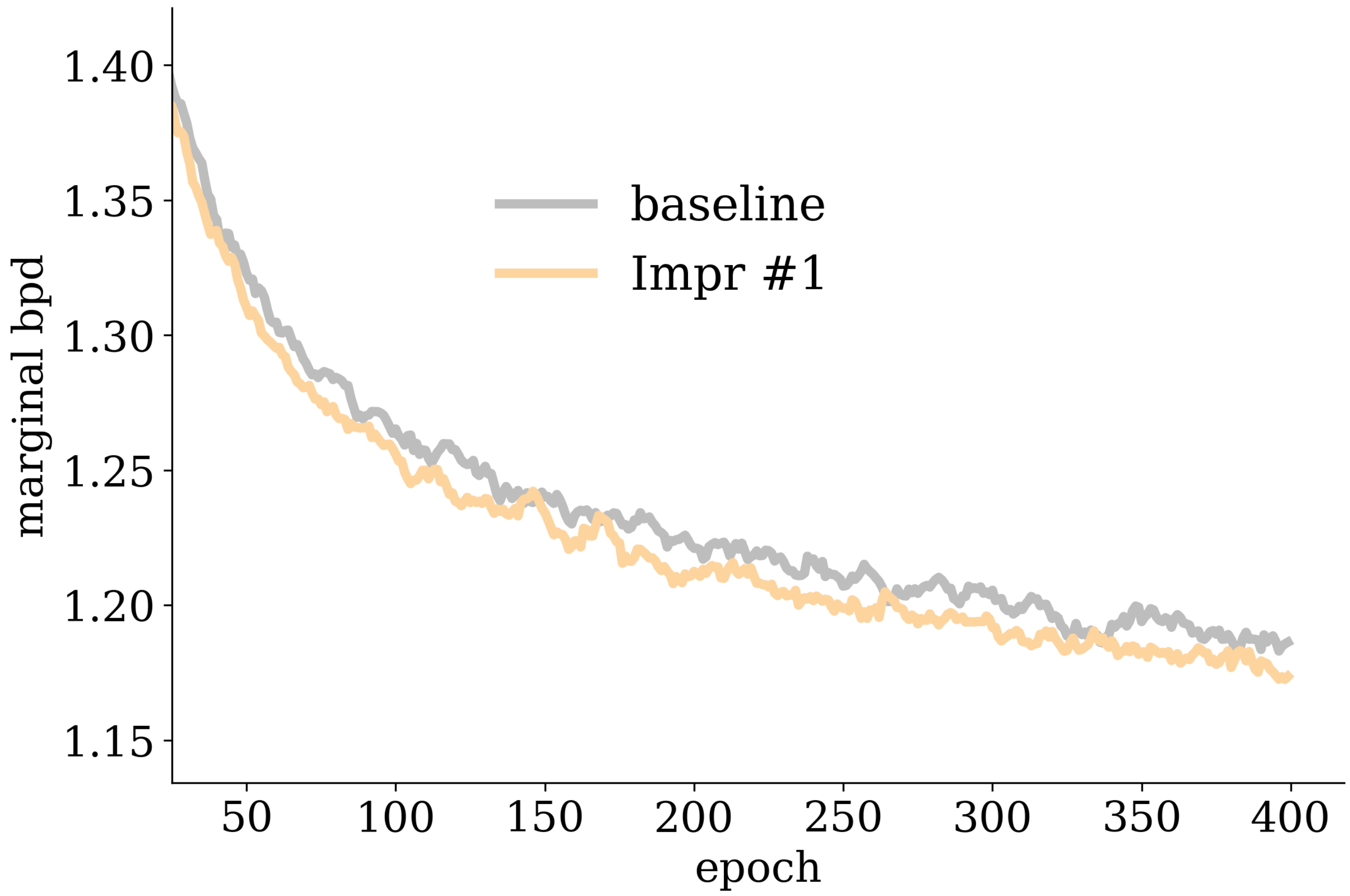
$\alpha_4 \quad \log p(x_4|x_3)$

...

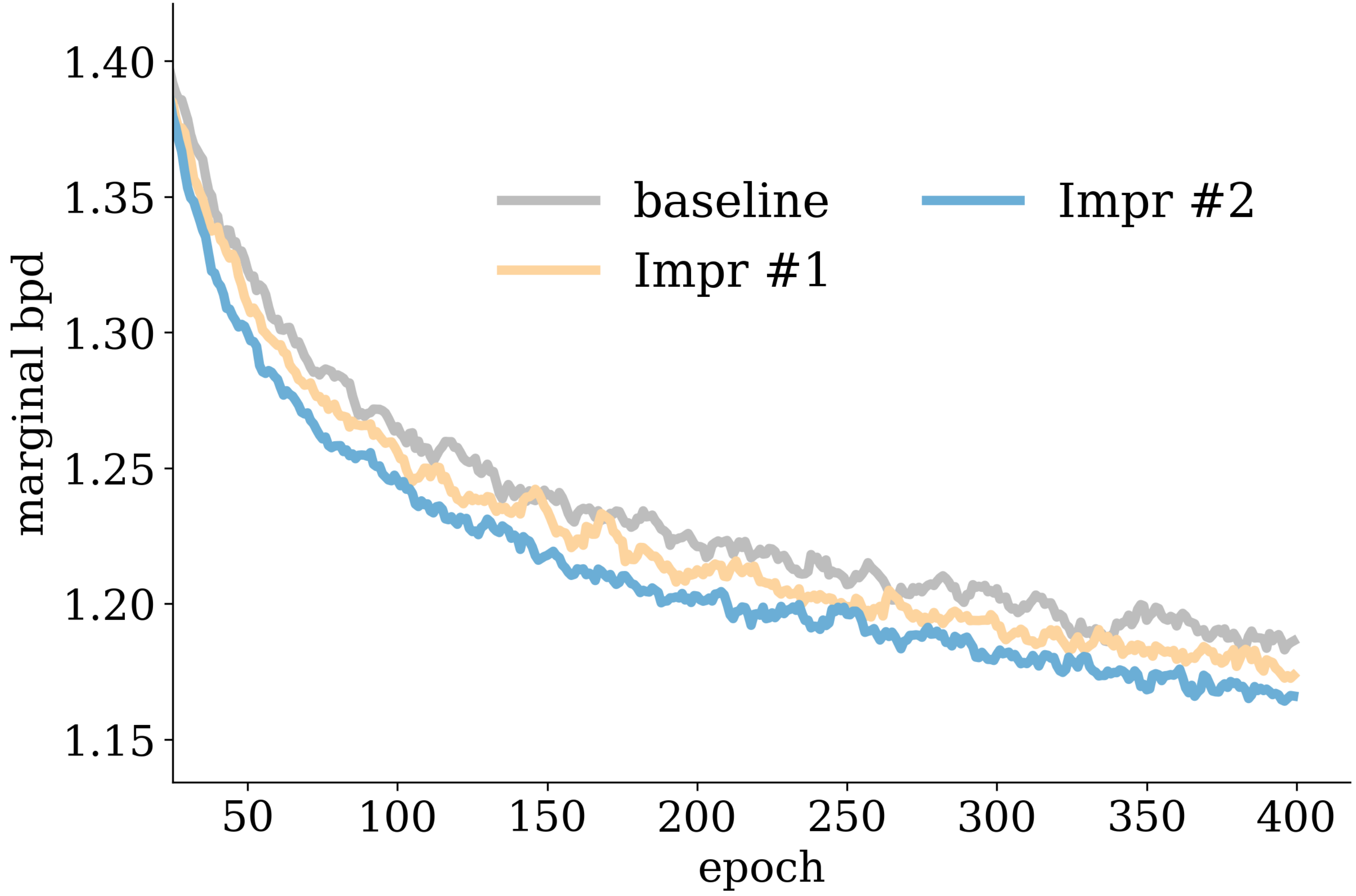
Ablations


baseline
marginal bpd
1.40
1.35
1.30
1.25
1.20
1.15
50
100
150
200
250
300
350
400
epoch

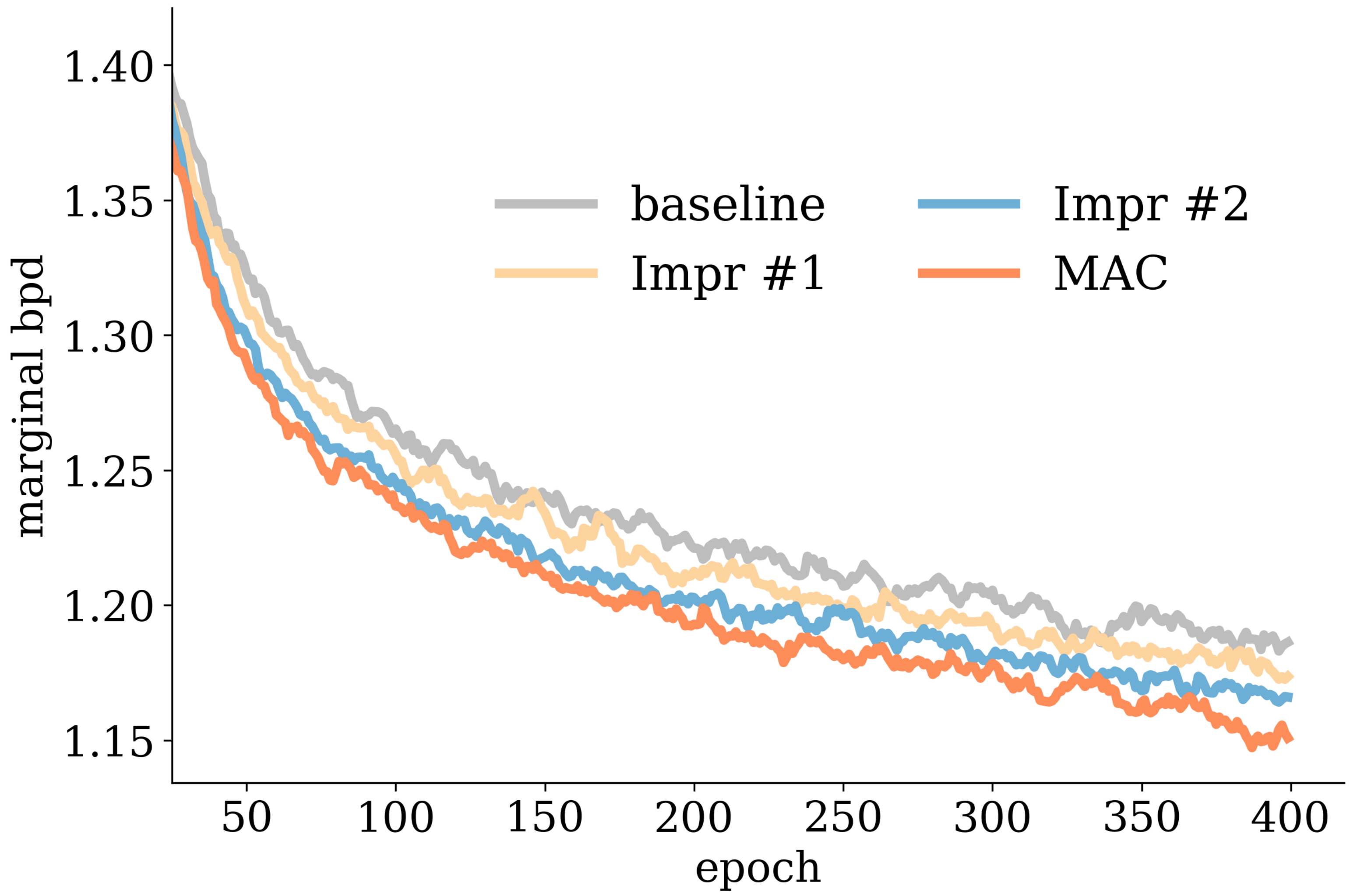
Ablations


baseline
Impr #1
marginal bpd
1.40
1.35
1.30
1.25
1.20
1.15
50
100
150
200
250
300
350
400
epoch

Ablations


baseline
Impr #1
Impr #2
marginal bpd
1.40
1.35
1.30
1.25
1.20
1.15
50
100
150
200
250
300
350
400
epoch

Ablations



marginal bpd
1.40
1.35
1.30
1.25
1.20
1.15
50
100
150
200
250
300
350
400
epoch
baseline
Impr #1
Impr #2
MAC

Text8 character modeling

Text8 dataset (bpd, lower is better)

	joint	marginal
ARDM (3000 epochs)	1.48	1.12
MAC (3000 epochs)	1.40	1.09

Text8 character modeling

Text8 dataset (bpd, lower is better)

	joint	marginal
OA-Transformer	1.64	
D3PM	1.47	
ARDM (14000 epochs)	1.43	
ARDM (3000 epochs)	1.48	1.12
MAC (3000 epochs)	1.40	1.09

Transformer: 1.35

Text8 character modeling

```
 prophylactic drugs several drugs most of which are also used for trea
tment of malaria can be taken preventatively generally these drugs are
 taken daily or weekly at a lower dose than would be used for treatmen
t of a person who had actually contracte
```

Text8 character modeling

```
 prophylactic drugs several drugs most of which are also used for trea
tment of malaria can be taken preventatively generally these drugs are
 taken daily or weekly at a lower dose than would be used for treatmen
t of a person who had actually contracte

 pr_p_y__cti__dr__s _eve__l drug_ _o_t of__h__h are __so u_ed __r _re_
tment__f mal__i__c__ b_ _a_en _re_enta__vely ge__ra_l_ t_es_ d_ugs_are
 _ake___aily or _ee_ly _t a lo_er dose t__n _o_ld _e_used_f_r _re_tme_
t__f__ _e__on_w____ad_a_tuall___on_ra_te
```

Text8 character modeling

```
 prophylactic drugs several drugs most of which are also used for trea
tment of malaria can be taken preventatively generally these drugs are
 taken daily or weekly at a lower dose than would be used for treatmen
t of a person who had actually contracte

 pr_p_y__cti__dr__s _eve__l drug_ _o_t of__h__h are __so u_ed __r _re_
tment__f mal__i__c__ b_ _a_en _re_enta__vely ge__ra_l_ t_es_ d_ugs_are
 _ake___aily or _ee_ly _t a lo_er dose t__n _o_ld _e_used_f_r _re_tme_
t__f__ _e__on_w____ad_a_tuall___on_ra_te

 prophylactic drugs several drugs most of which are also used for trea
tment of malaria can be taken preventatively generally these drugs are
 taken daily or weekly at a lower dose than would be used for treatmen
t of a lesion who had actually contracte
```

ImageNet32

ImageNet32 dataset (bpd, lower is better)

	joint	marginal
ARDM (16 epochs)	3.60	2.10
MAC (16 epochs)	3.58	2.08

Image Transformer: 3.77

ImageNet32

ImageNet32

ImageNet32

CIFAR10

CIFAR10 dataset (bpd, lower is better) w/ rotation, flipping

	joint	marginal
ARDM (1200 epochs)	2.86	1.84
MAC (1200 epochs)	2.81	1.81

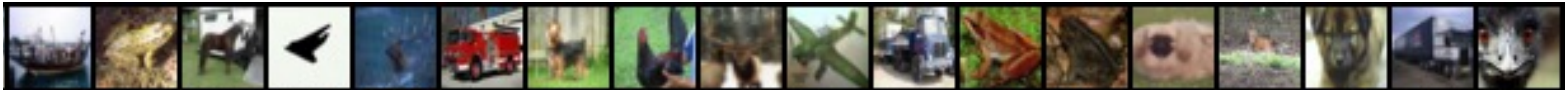
CIFAR10

CIFAR10 dataset (bpd, lower is better) w/ rotation, flipping

	joint	marginal
D3PM	3.44	
ARDM (3000 epochs)	2.69	
ARDM (1200 epochs)	2.86	1.84
MAC (1200 epochs)	2.81	1.81

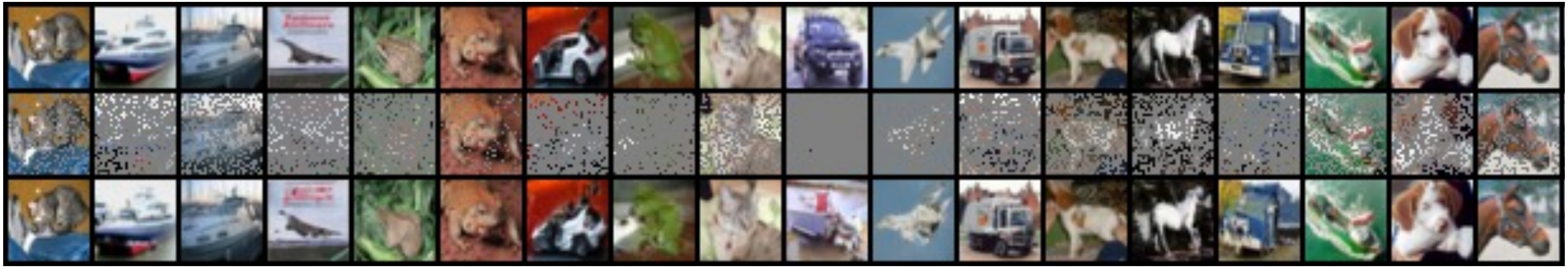
Sparse Transformer: 2.56

CIFAR10



CIFAR10

CIFAR10

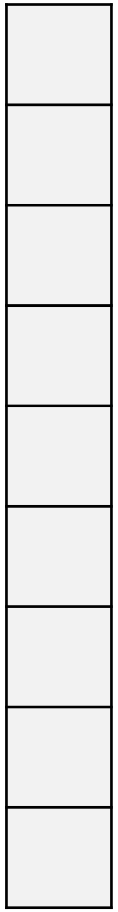


Continuous Tabular Benchmarks

Marginal log-likelihood on continuous tabular datasets (higher is better)

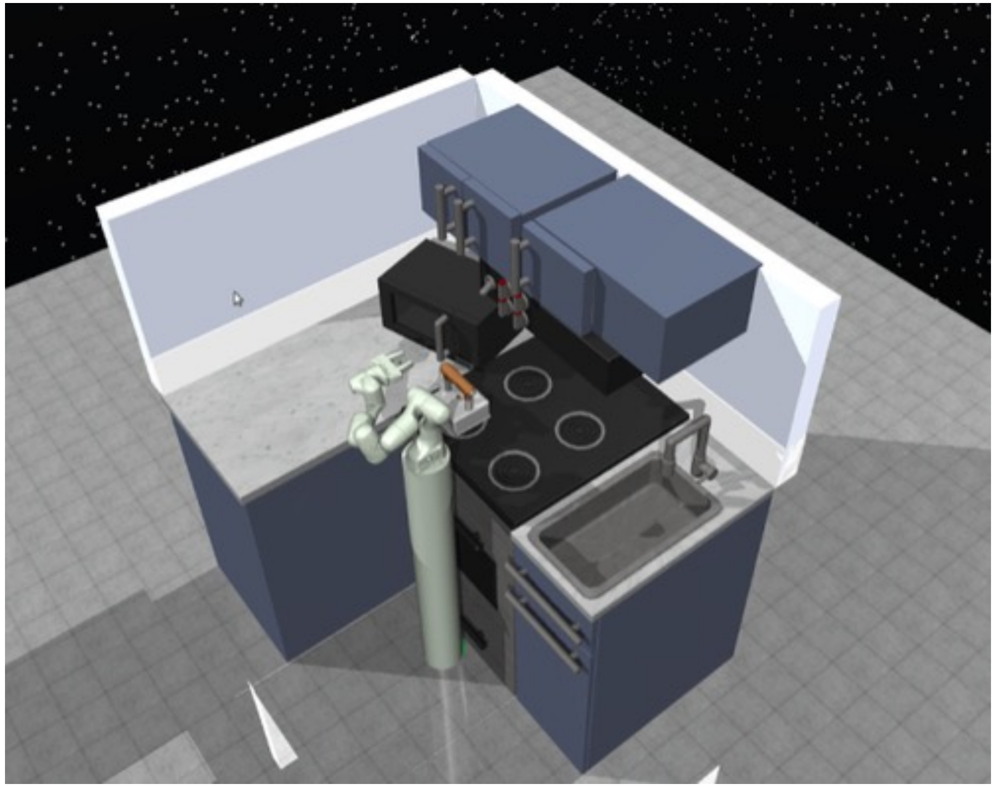
	power	gas	hepmass	miniboone	bsds
SPFlow	-0.12	4.81	-13.38	-9.85	-8.15
ACFlow	0.42	10.13	-11.58	-10.36	19.60
ACE	0.58	12.20	-10.72	-7.94	20.31
MAC	**0.61**	**13.02**	**-10.69**	**-7.76**	**20.33**

Shared-Autonomy

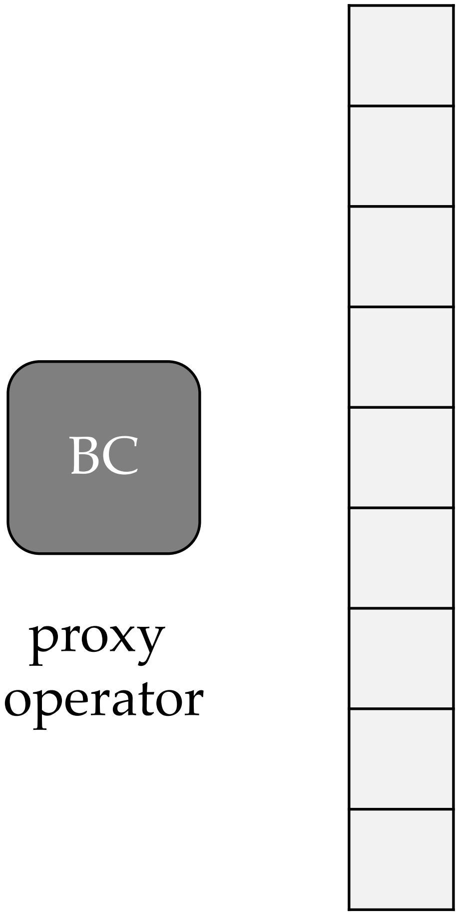


action dim: 9

FrankaKitchen
Kitchen-mixed0



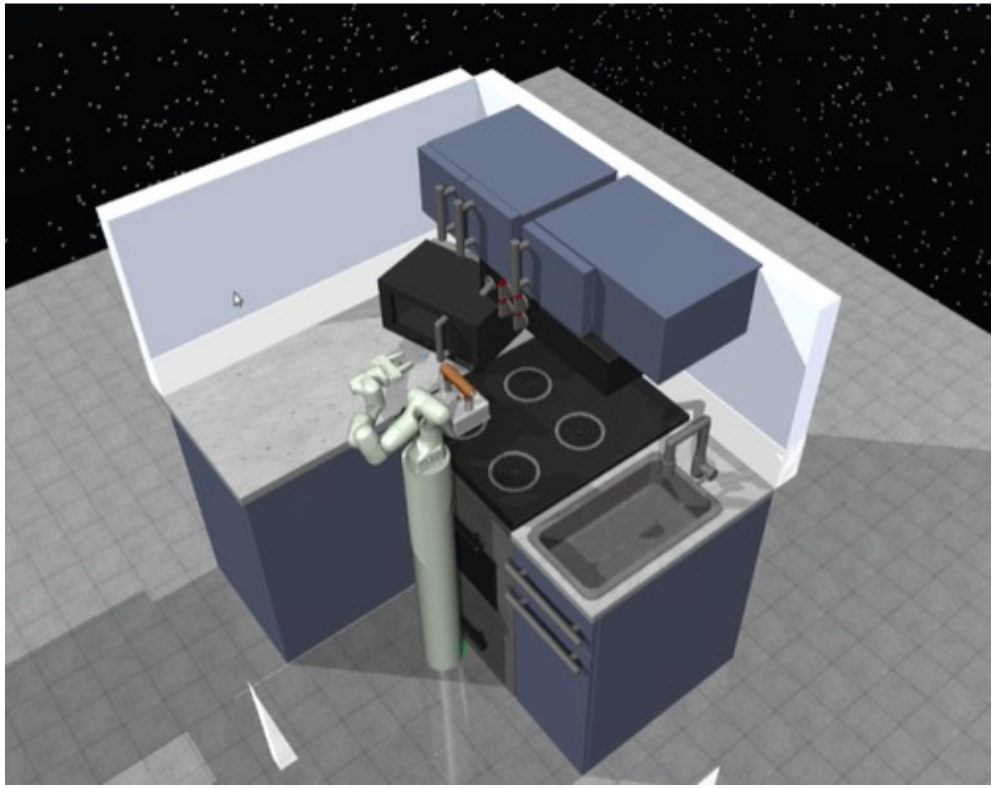
Shared-Autonomy



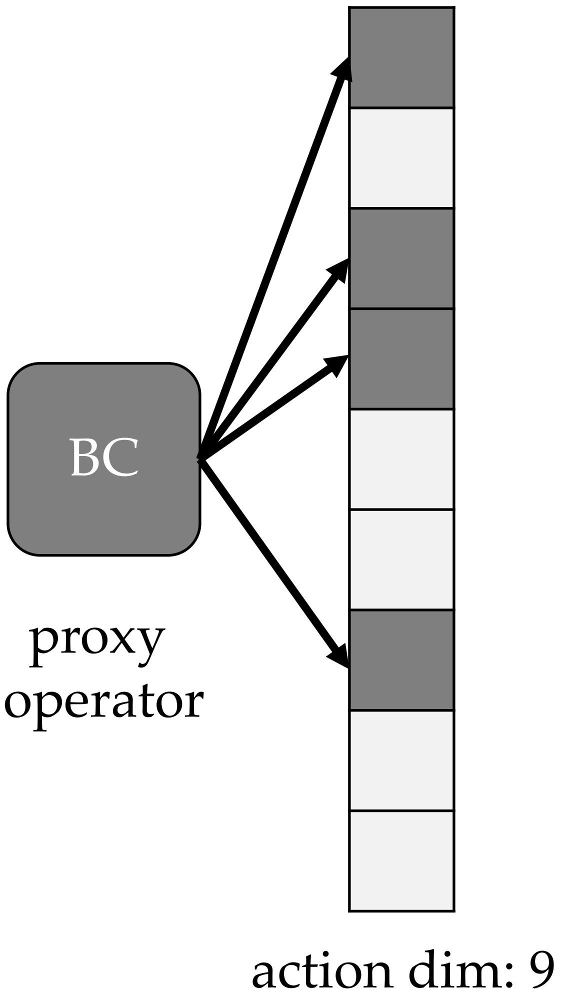
proxy
operator

action dim: 9

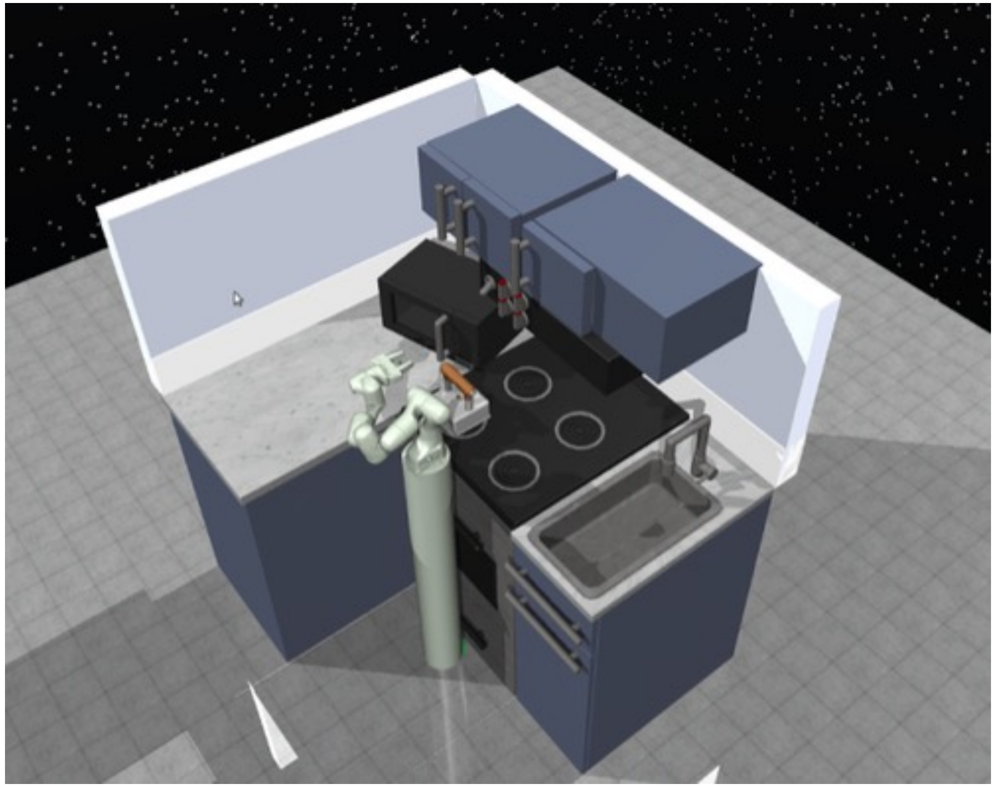
FrankaKitchen
Kitchen-mixed0



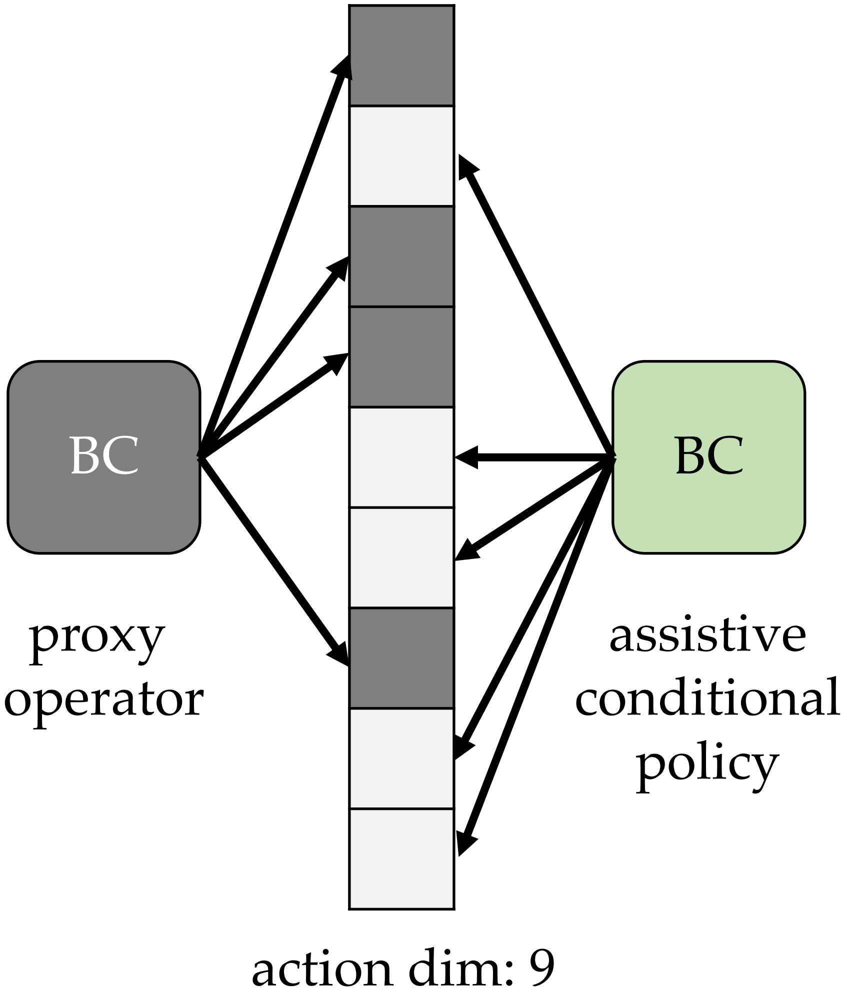
Shared-Autonomy



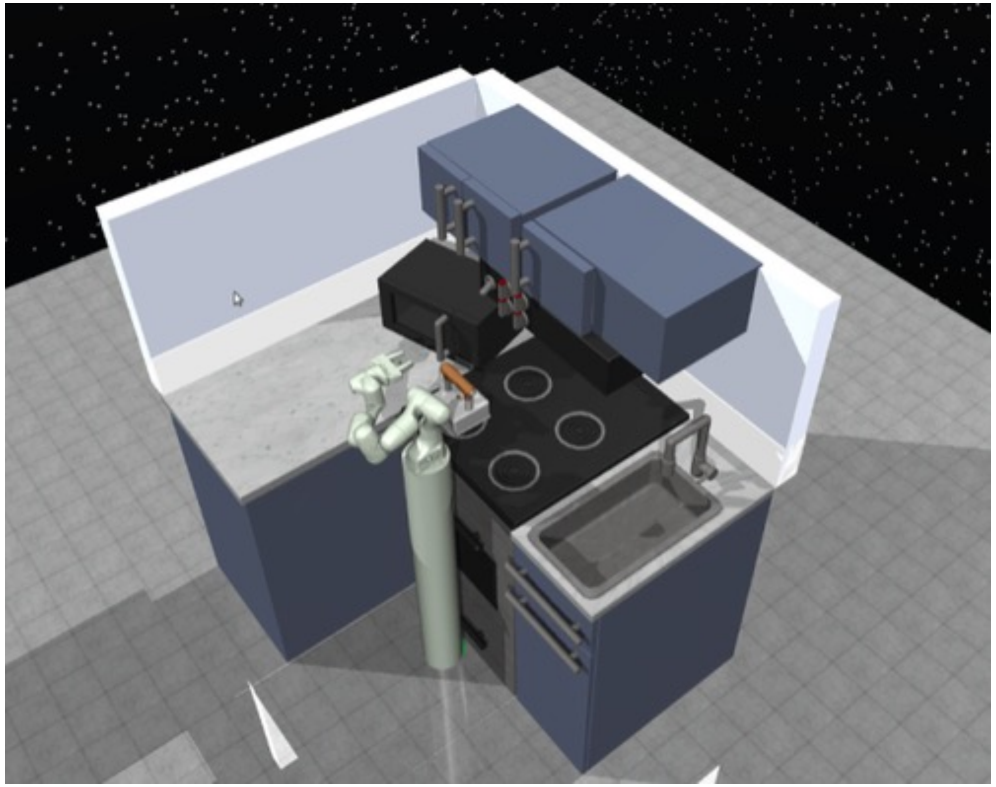
FrankaKitchen
Kitchen-mixed0



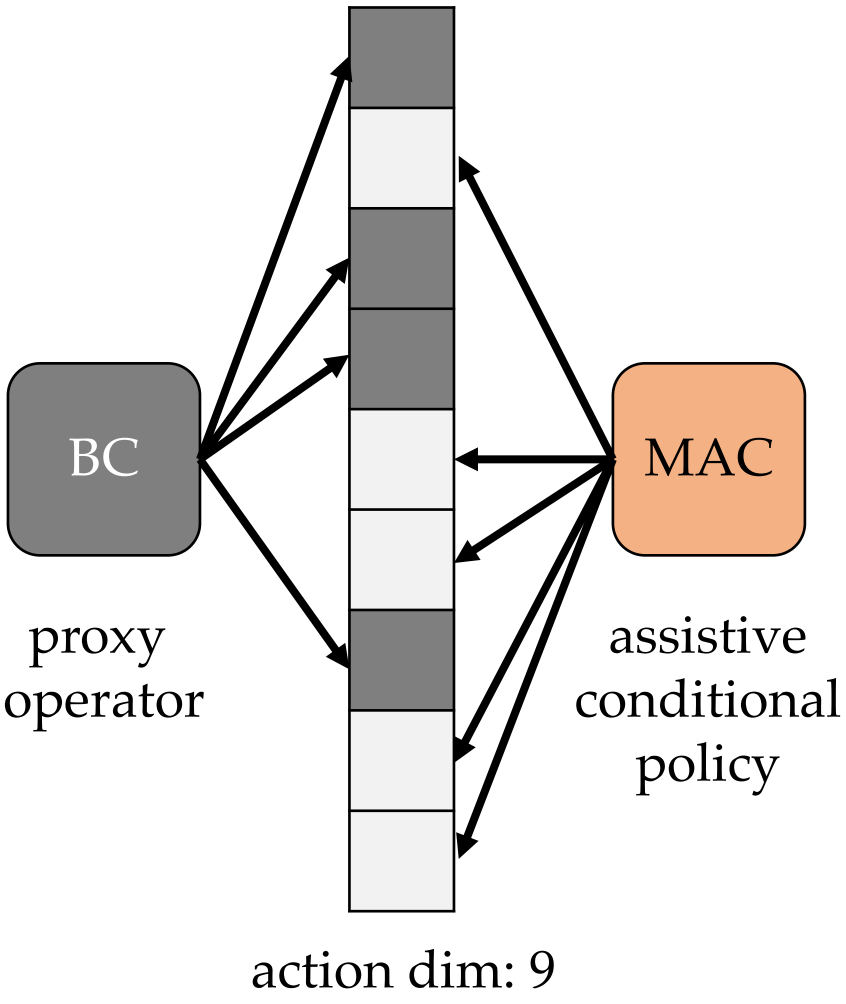
Shared-Autonomy



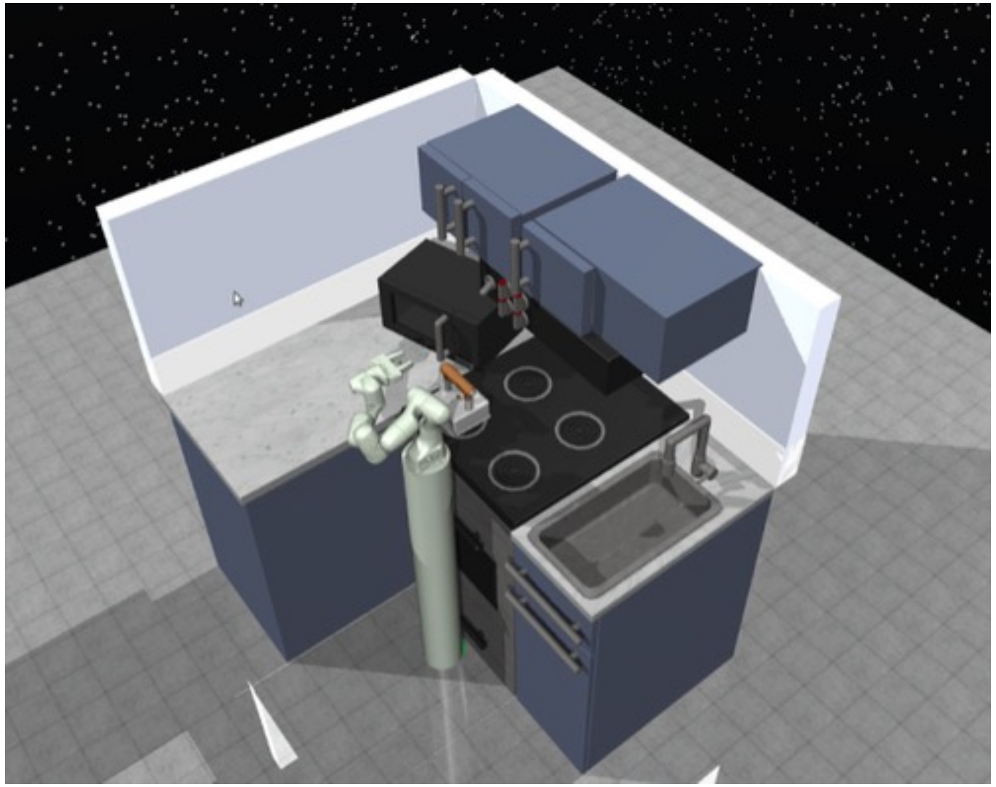
FrankaKitchen
Kitchen-mixed0



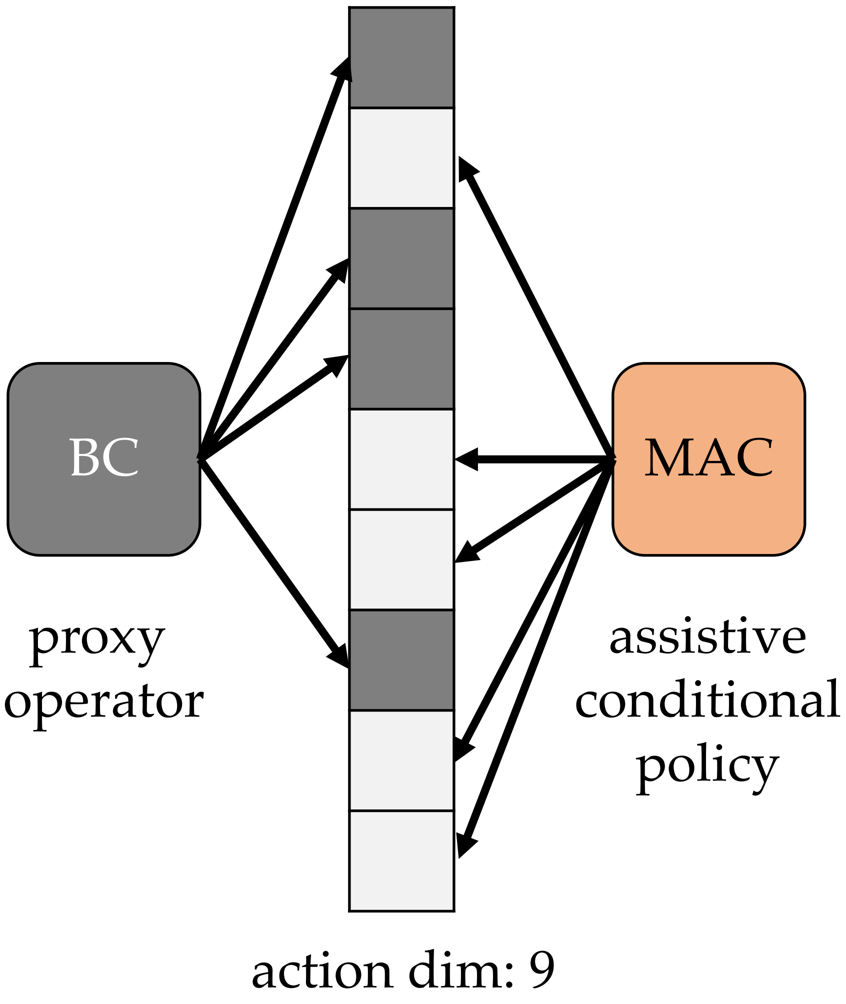
Shared-Autonomy



FrankaKitchen
Kitchen-mixed0



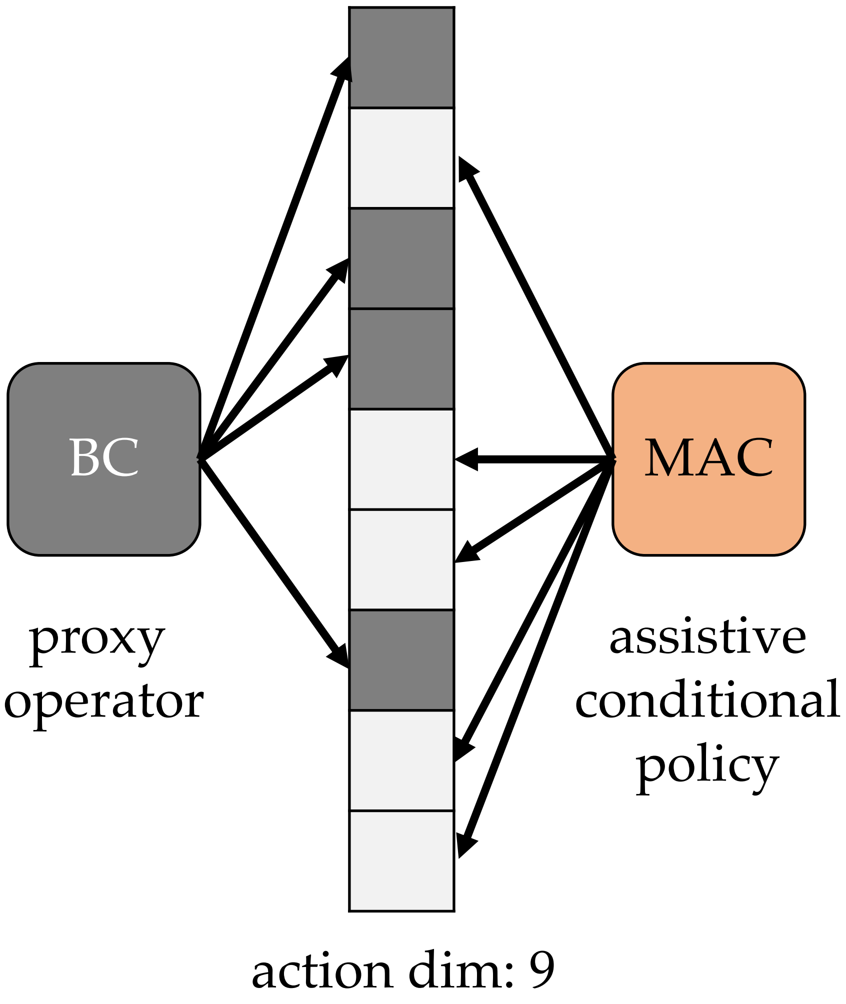
Shared-Autonomy



FrankaKitchen
Kitchen-mixed0

	Reward
BC	1.81 ± 0.08
MAC	**2.00** ± 0.05

Shared-Autonomy



FrankaKitchen
Kitchen-mixed0

	Reward
BC	1.81 ± 0.08
MAC	**2.00** ± 0.05

Full autonomy
IBC: 2.15 ± 0.06

Just a few lines

```python
def sample_test_masks(batch: int, xdim: int):
    sigma = rand(size=(batch, xdim)).argsort(dim=-1)
    t = randint(low=1, high=xdim+1, size=(batch, 1))
    masks = sigma < t
    return masks, t
```

Just a few lines

```
def sample_test_masks(batch: int, xdim: int):
    sigma = rand(size=(batch, xdim)).argsort(dim=-1)
    t = randint(low=1, high=xdim+1, size=(batch, 1))
    masks = sigma < t
    return masks, t

def sample_train_masks(batch: int, xdim: int):
    test_masks, test_t = sample_test_masks(batch, xdim)

    # sample intermediate prefix by taking random int in [0, test_t)
    batch_arange = arange(xdim).reshape(1, xdim).repeat(batch, 1)
    nonzero_weights = (batch_arange < test_t).float()
    t = multinomial(nonzero_weights, num_samples=1)

    # double argsort trick to get ranks, but we need:
    # 1. descending=True to order 1s before 0s of the bitmask
    # 2. stable=True to keep the relative ordering between the 1s
    sigma = test_masks.long().sort(descending=True,
                                   stable=True).indices.argsort()
    masks = sigma < t
    return masks, t
```

Takeaways

AO-ARMs are SOTA for arbitrary conditional modeling

Takeaways

AO-ARMs are SOTA for arbitrary conditional modeling

applications: masked language modeling, inpainting,
hierarchical planning, Prob.Prog. Inference

Takeaways

AO-ARMs are SOTA for arbitrary conditional modeling

applications: masked language modeling, inpainting, hierarchical planning, Prob.Prog. Inference

But AO-ARMs try to learn *too* much

Takeaways

AO-ARMs are SOTA for arbitrary conditional modeling

applications: masked language modeling, inpainting, hierarchical planning, Prob.Prog. Inference

But AO-ARMs try to learn *too* much

→ redundancy → worse performance due to finite capacity

Takeaways

AO-ARMs are SOTA for arbitrary conditional modeling

applications: masked language modeling, inpainting, hierarchical planning, Prob.Prog. Inference

But AO-ARMs try to learn *too* much

→ redundancy → worse performance due to finite capacity

MAC learns *just enough* to support arbitrary conditionals

→ reduce redundancy → better performance

Takeaways

AO-ARMs are SOTA for arbitrary conditional modeling

applications: masked language modeling, inpainting, hierarchical planning, Prob.Prog. Inference

But AO-ARMs try to learn *too* much

→ redundancy → worse performance due to finite capacity

https://arxiv.org/abs/2205.13554

MAC learns *just enough* to support arbitrary conditionals

→ reduce redundancy → better performance