

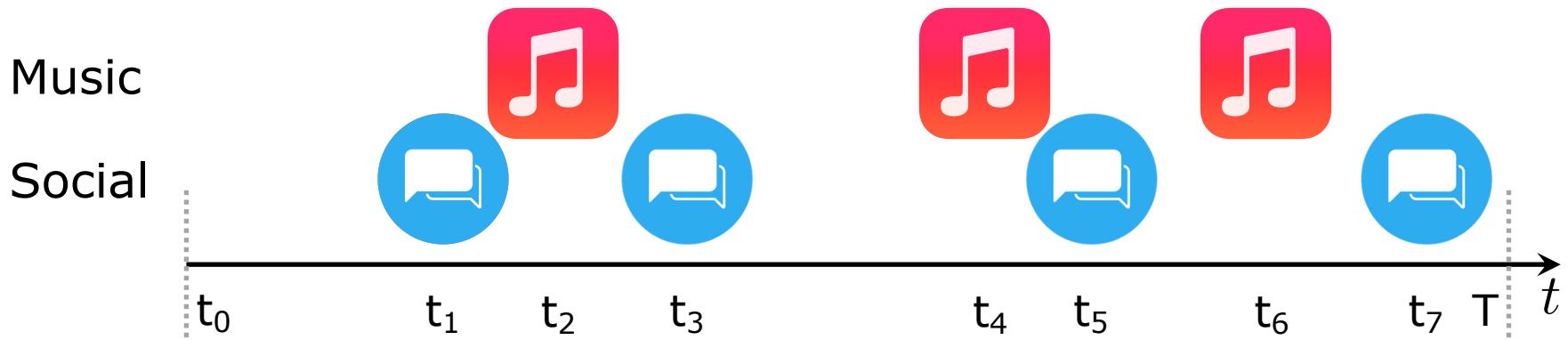
# Modeling Irregular Time Series

Hongyuan Mei

2018–2021 Bloomberg Data Science Ph.D. Fellow

Johns Hopkins University

# Multivariate Time Series



$\propto \lambda_{\text{music}}$



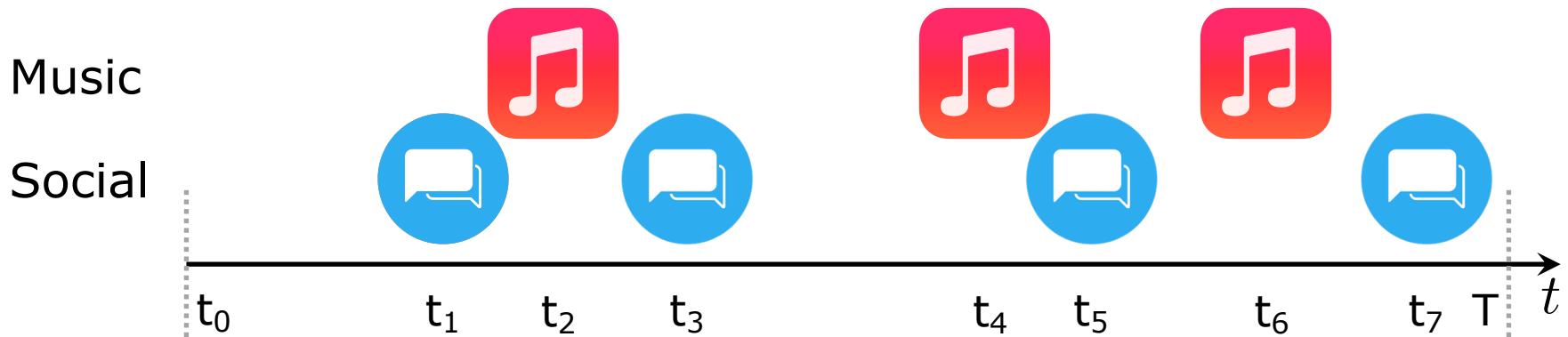
$\propto \lambda_{\text{social}}$

$(\lambda_{\text{music}} + \lambda_{\text{social}})dt = \text{prob of some event at } t$

$\lambda_{\text{social}}$

$\lambda_{\text{music}}$

# Multivariate Time Series: MLE



$$\begin{aligned} & \lambda_{\text{social}} \exp(-\lambda (t_1 - t_0)) \times \lambda_{\text{music}} \exp(-\lambda (t_2 - t_1)) \times \lambda_{\text{social}} \exp(-\lambda (t_3 - t_2)) \dots \\ & = \lambda_{\text{social}}^4 \lambda_{\text{music}}^3 \exp(-\lambda (T - t_0)) \end{aligned}$$

$\lambda = \lambda_{\text{music}} + \lambda_{\text{social}}$

$\lambda_{\text{social}} = 4 / (T - t_0)$

$\lambda_{\text{music}} = 3 / (T - t_0)$

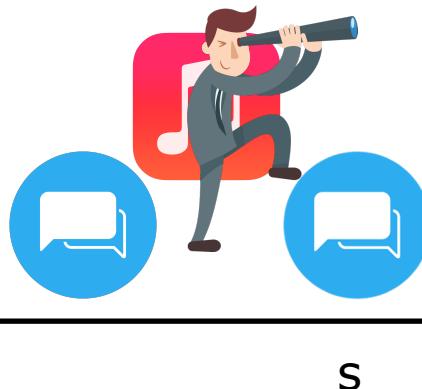
# of events of *that type*

# Multivariate Time Series: MBR

Music

Social

time



$$\lambda_{\text{social}} = 4 / (T - t_0)$$

$$\lambda_{\text{music}} = 3 / (T - t_0)$$

$$t_{\text{pred}} = \int_s^\infty t p(t) dt \quad p(t) = \lambda \exp(-\lambda (t - s))$$

$$\lambda = \lambda_{\text{music}} + \lambda_{\text{social}} = 7 / (T - t_0)$$

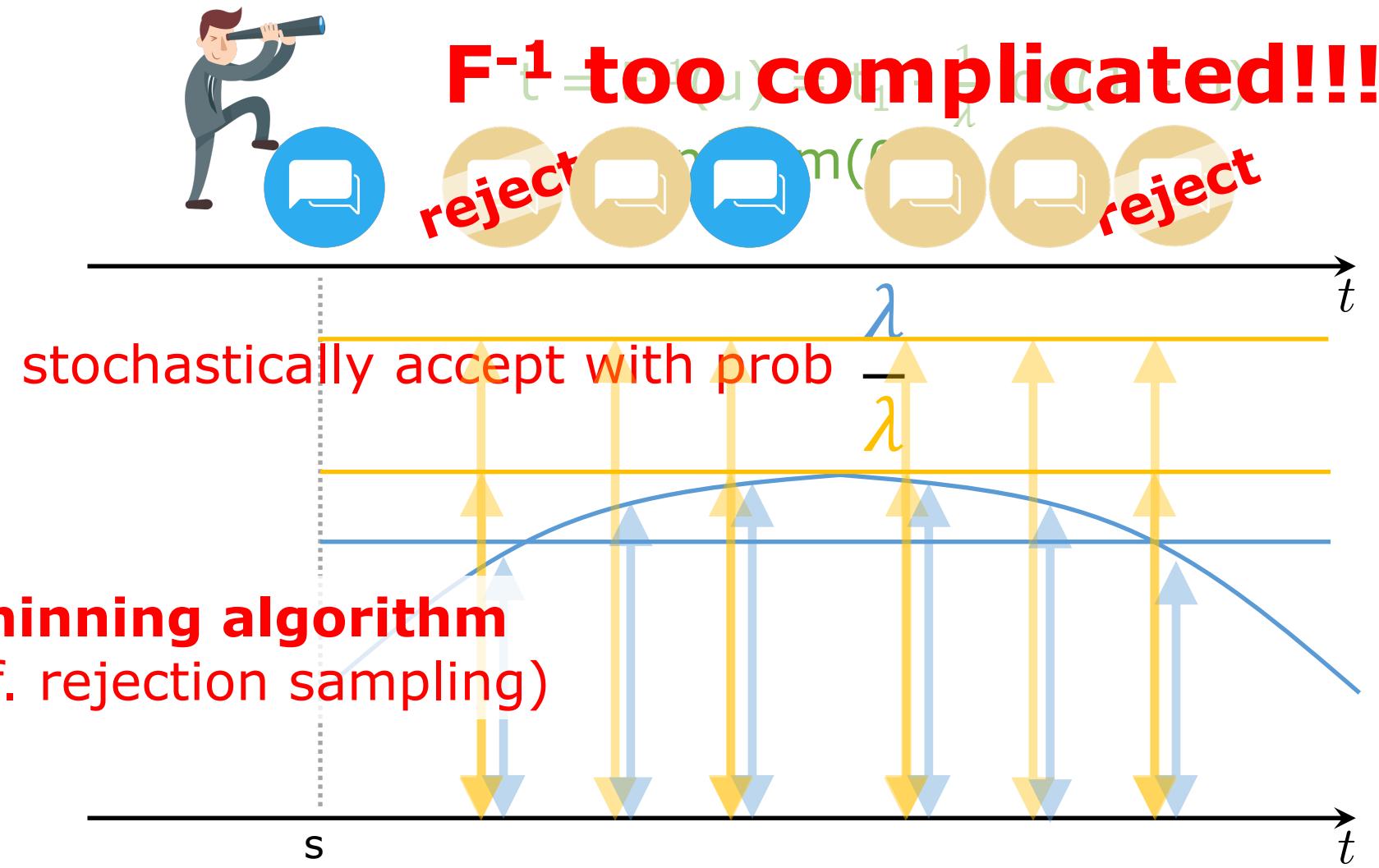
type?

$$k = \operatorname{argmax}_k p(k | t) = \operatorname{argmax}_k \lambda_k$$

$$\operatorname{argmin}_k \sum_j p(j | t) \mathbb{I}(k \neq j) \quad j, k \in \{\text{music}, \text{social}\}$$

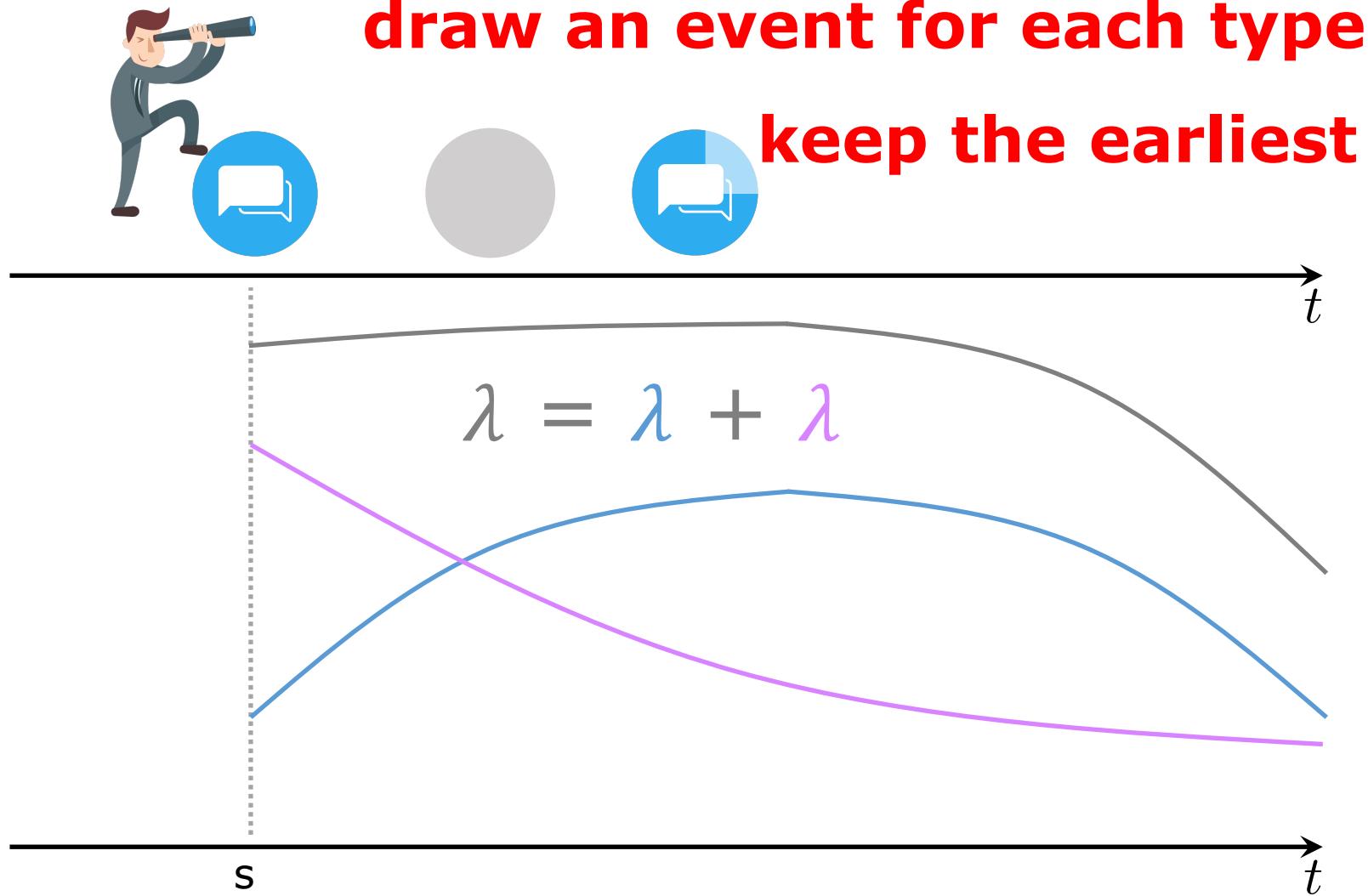
choose type  $k$  that makes  $\sum_{j \neq k} p(j | t)$  small

# When intensities are not constant...



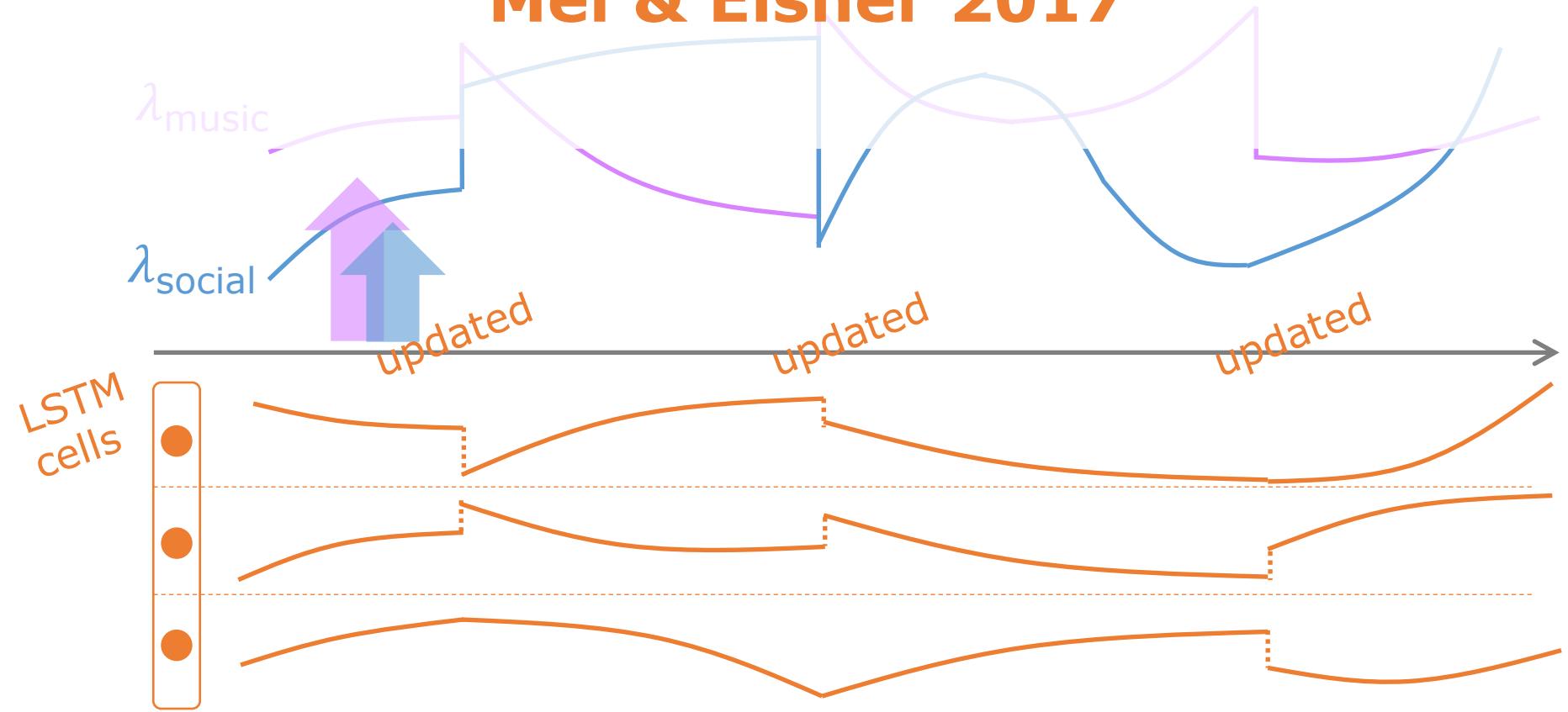
# Different types *race* against one another

**draw an event for each type**



# Let LSTM state evolve with time

## Neural Hawkes Process (NHP) Mei & Eisner 2017



# Lecture Structure

- Lecture-1: concepts
  - Intensity, point process, MLE, thinning, ...
- Lecture-2: fancy models
  - Hawkes, NHP, Neural ODE, ...
- Lecture-3: training + advanced topics
  - MLE & NCE
  - Large event space + domain knowledge

# MLE: Max log prob of *data*

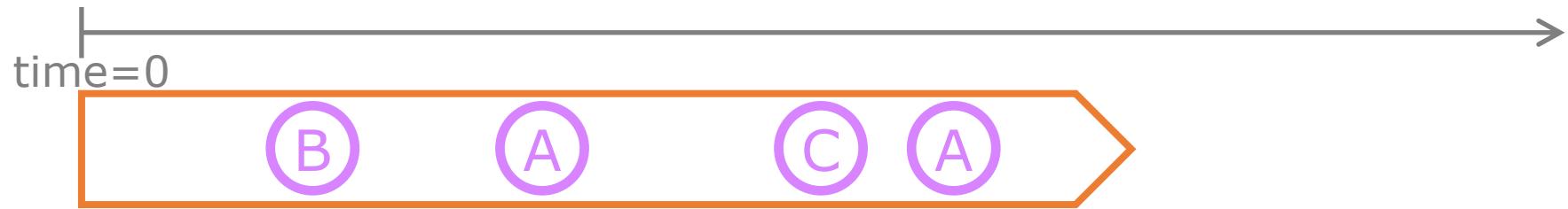
# MLE: Max log prob of *data*



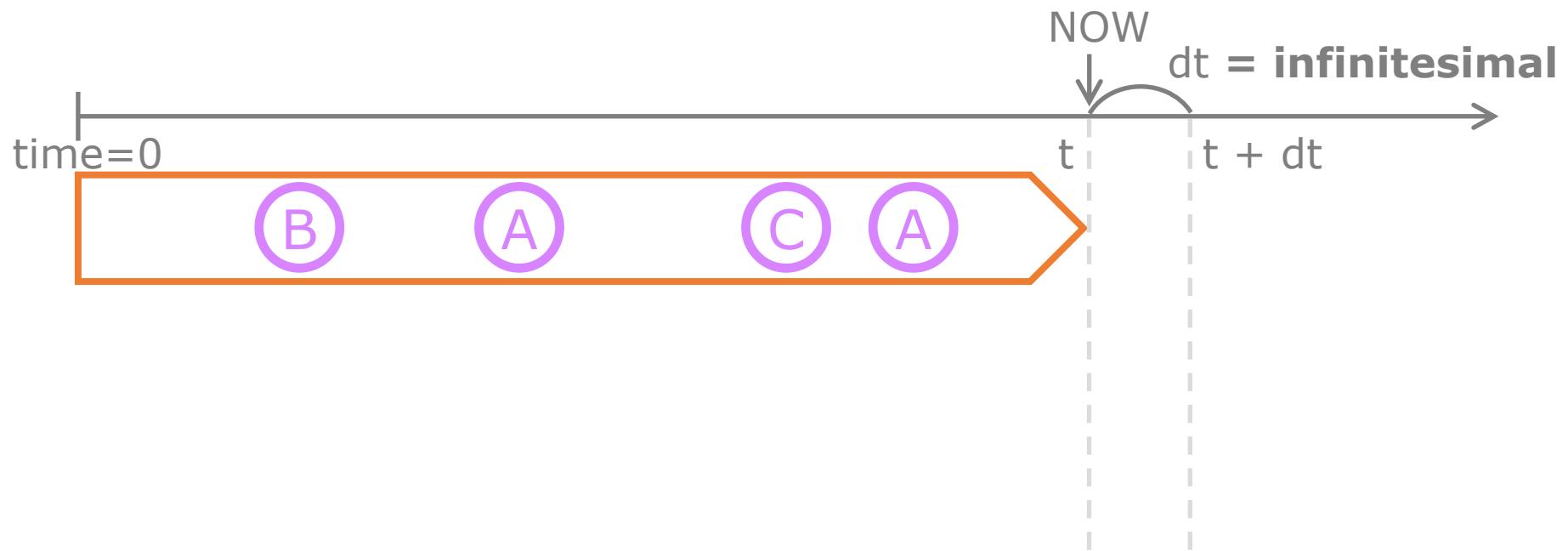
## MLE: Max log prob of *data*



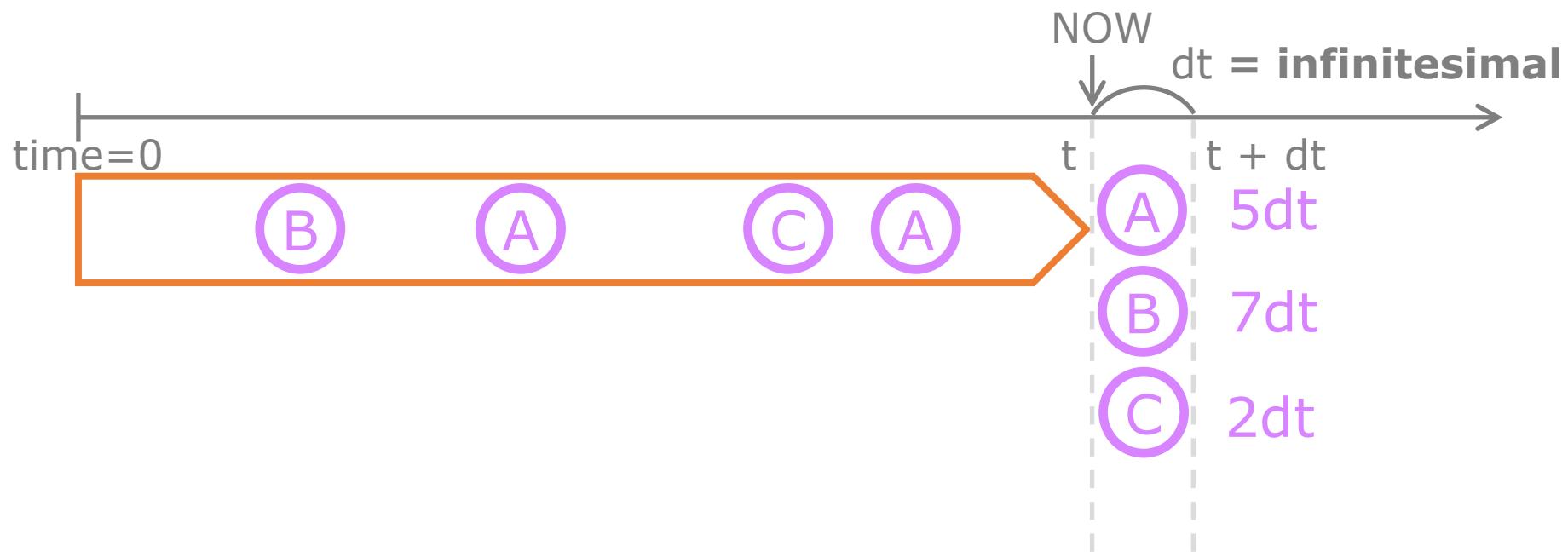
## MLE: Max log prob of *data*



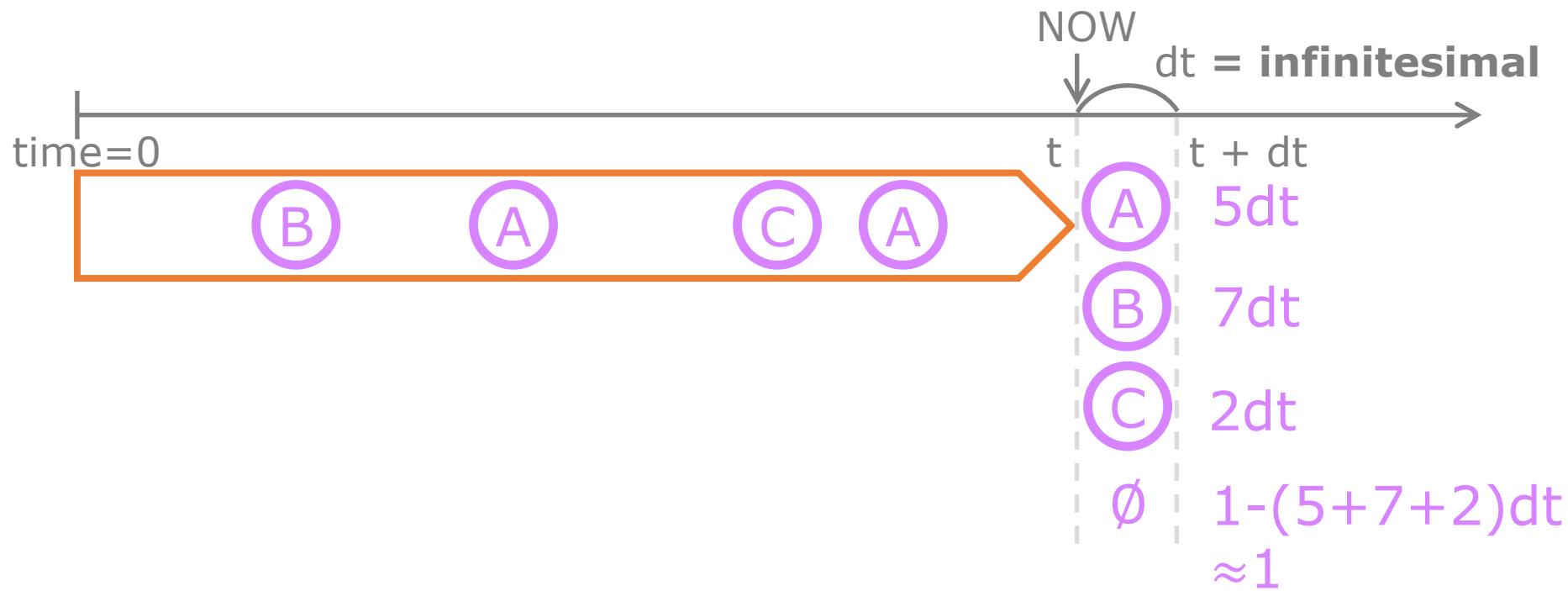
## MLE: Max log prob of *data*



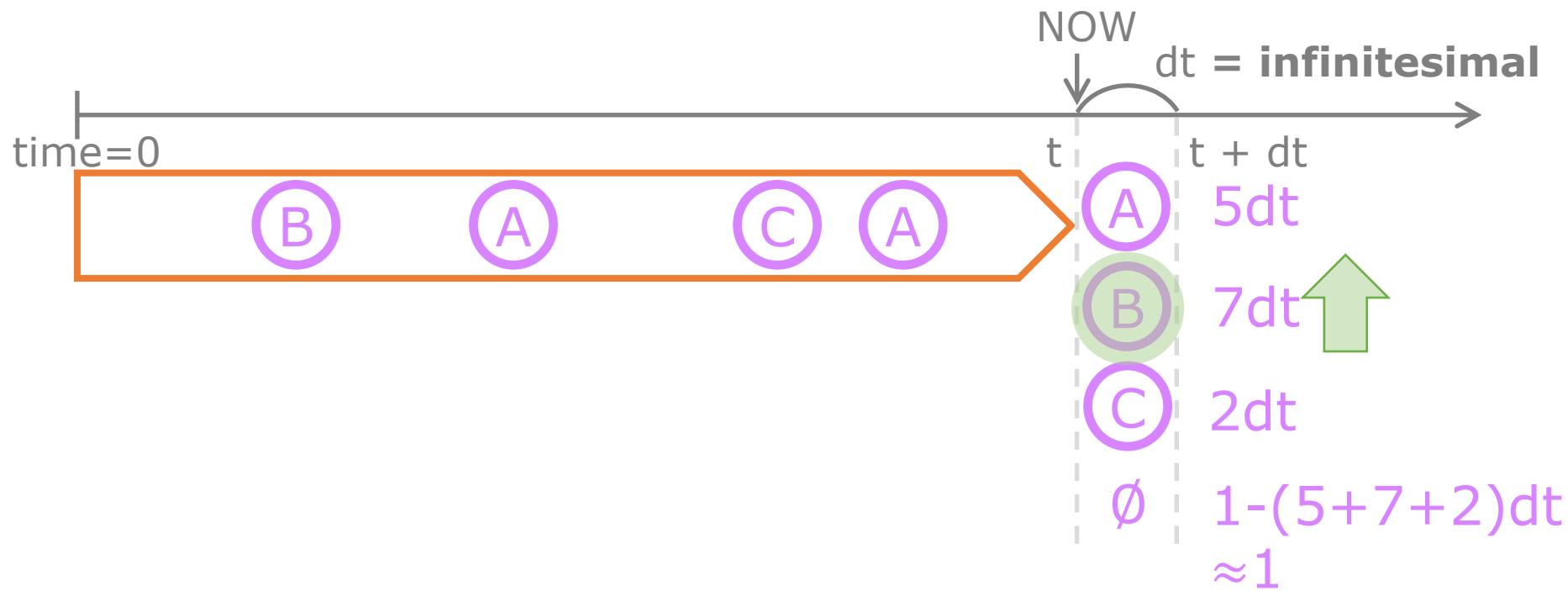
## MLE: Max log prob of *data*



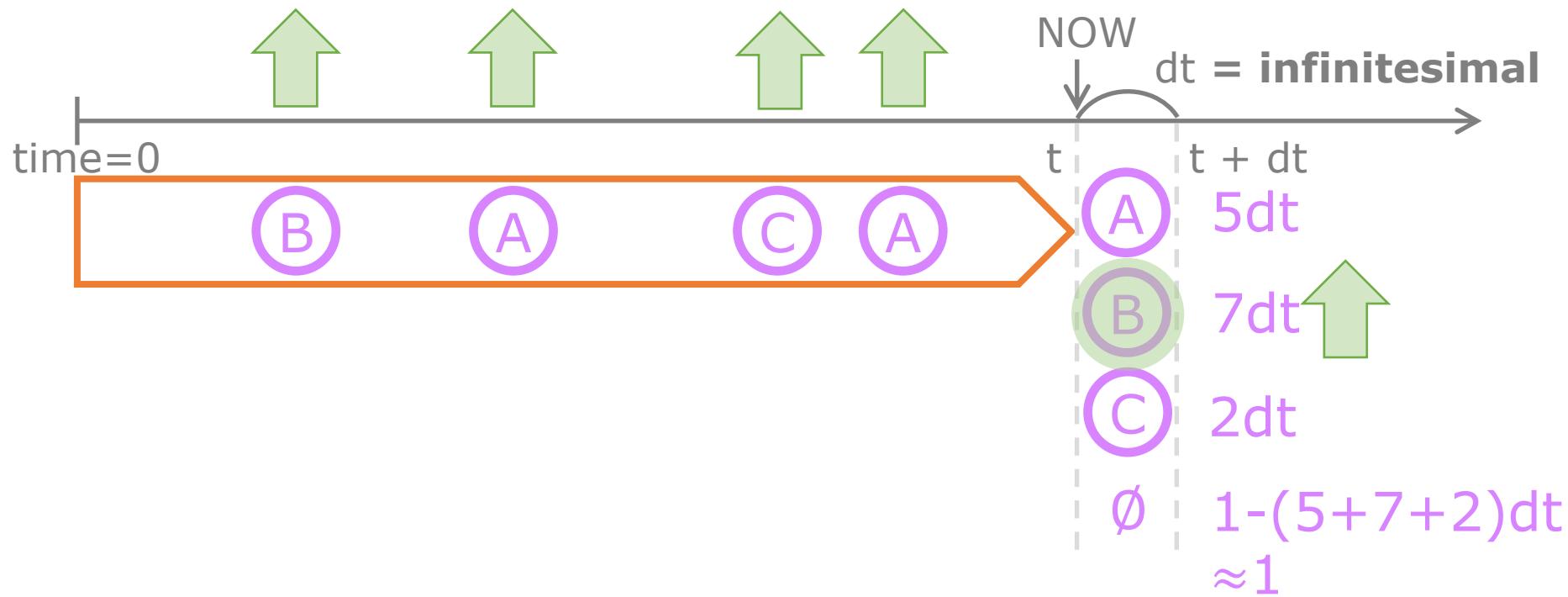
## MLE: Max log prob of *data*



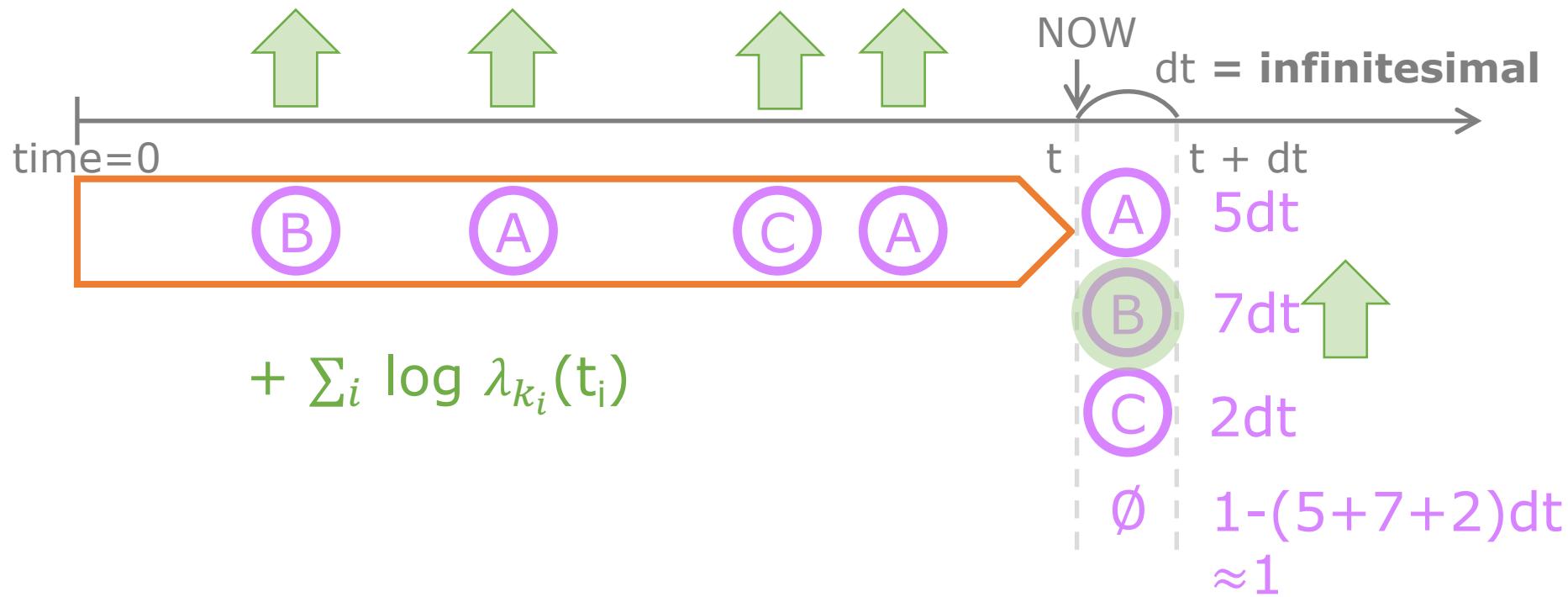
## MLE: Max log prob of *data*



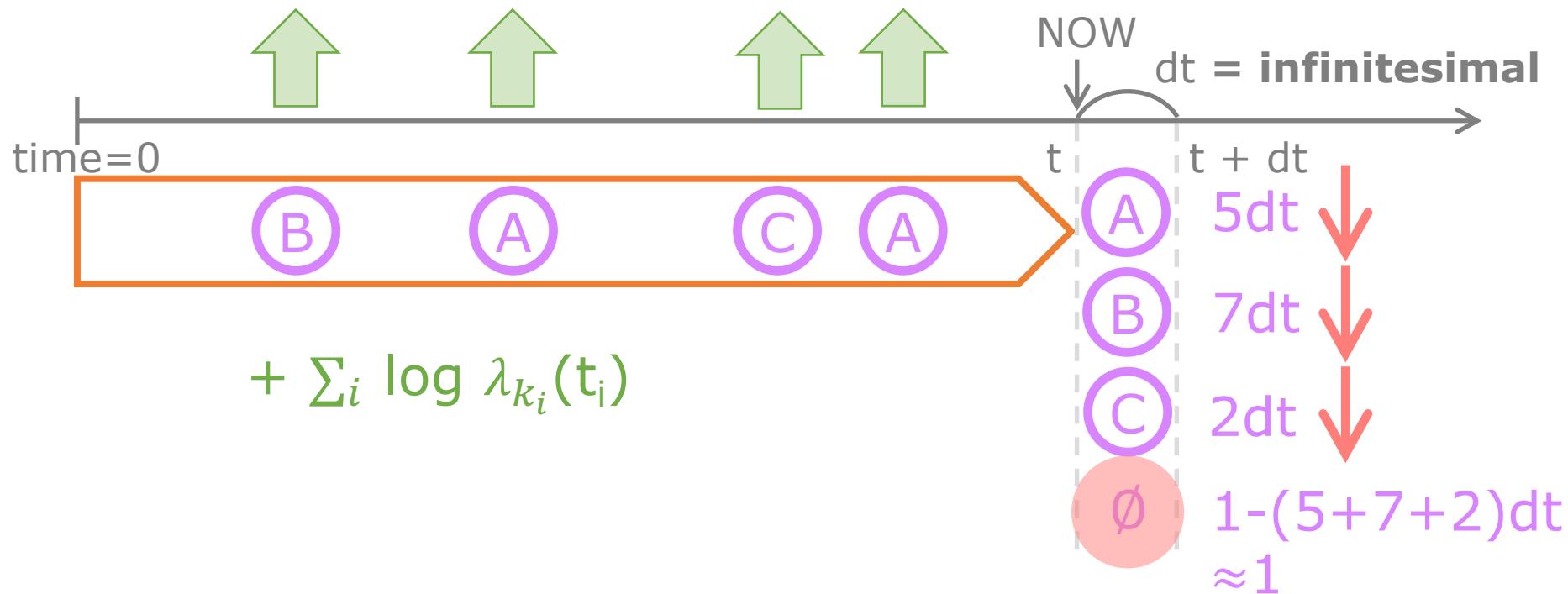
## MLE: Max log prob of *data*



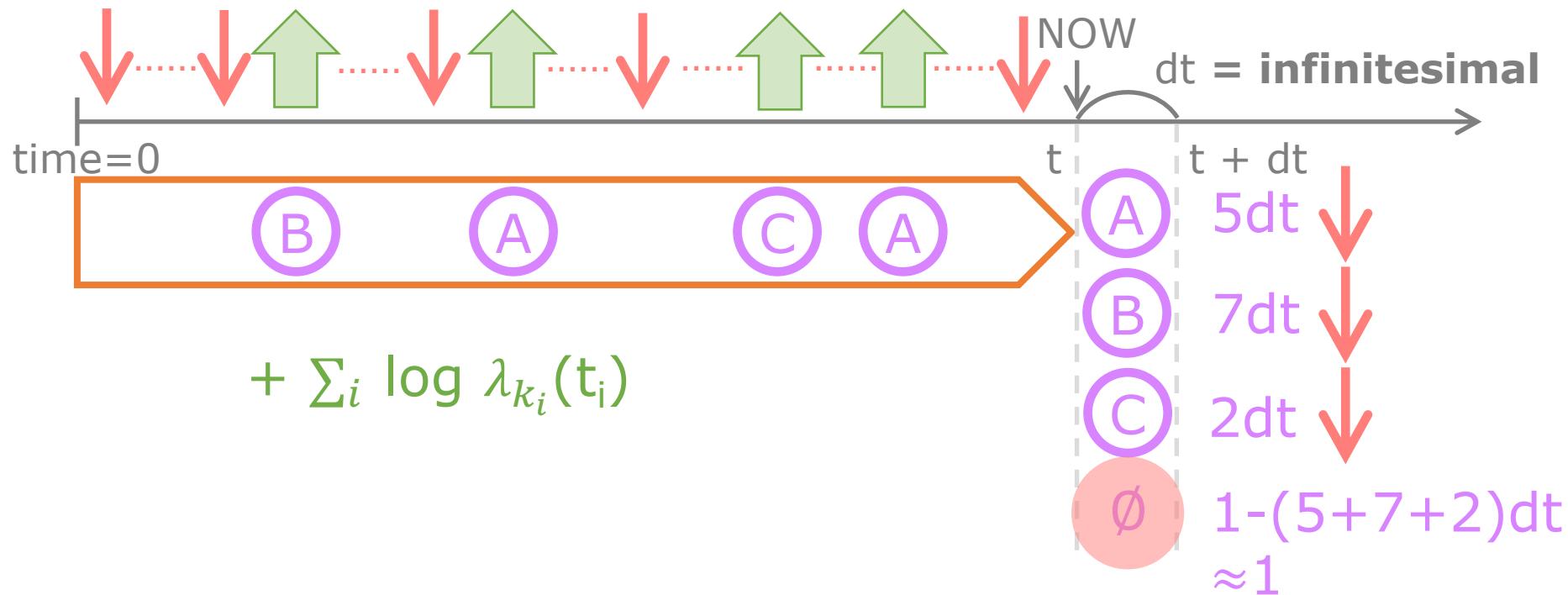
## MLE: Max log prob of *data*



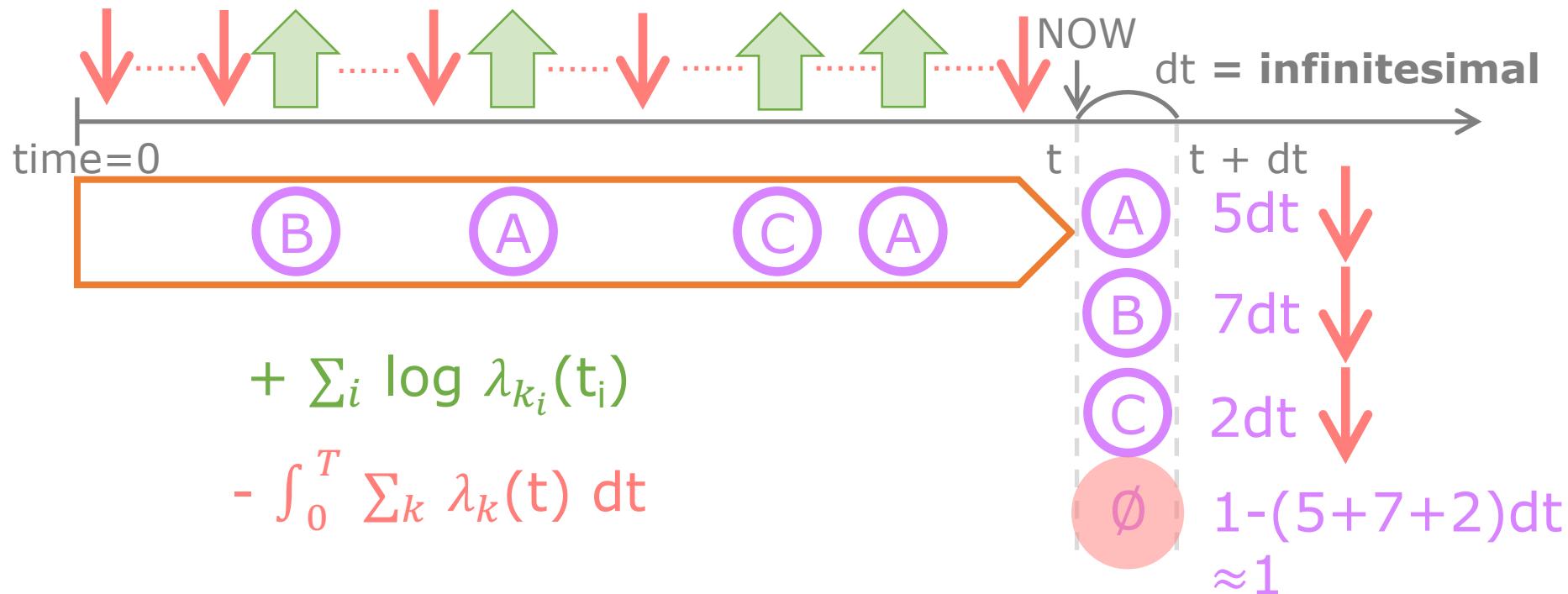
## MLE: Max log prob of *data*



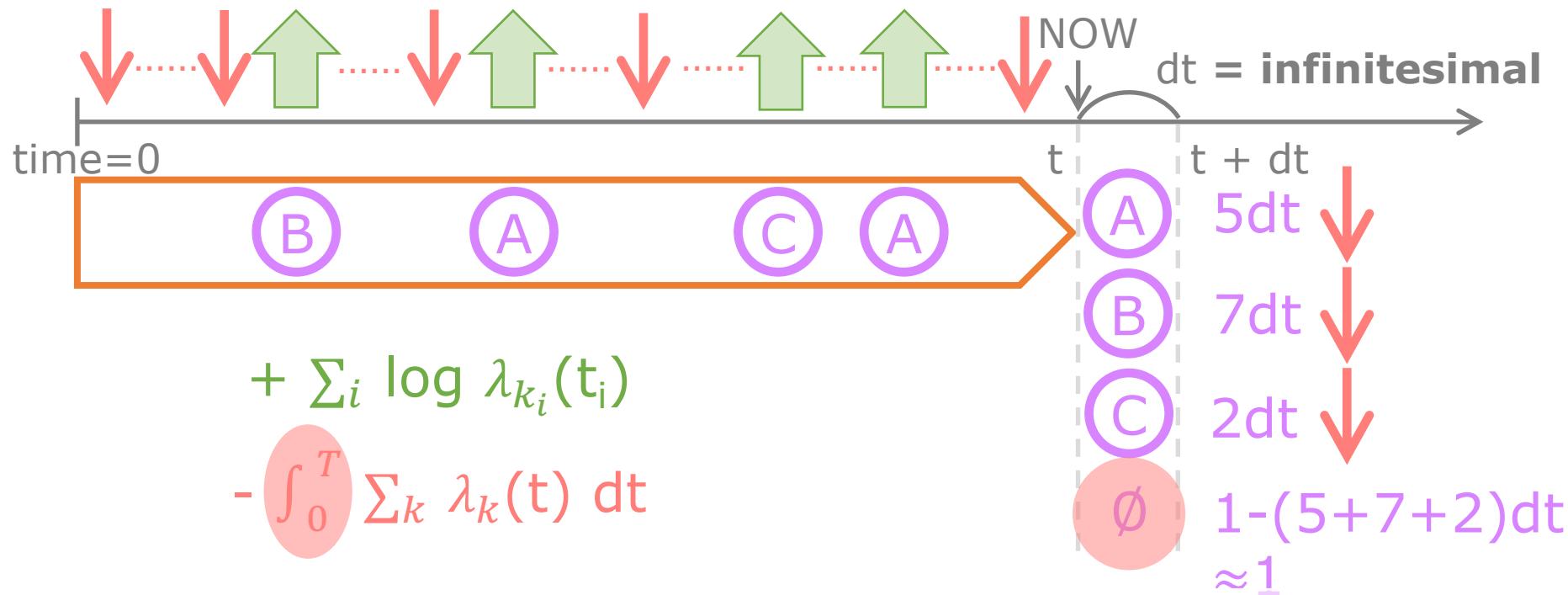
## MLE: Max log prob of *data*



## MLE: Max log prob of *data*

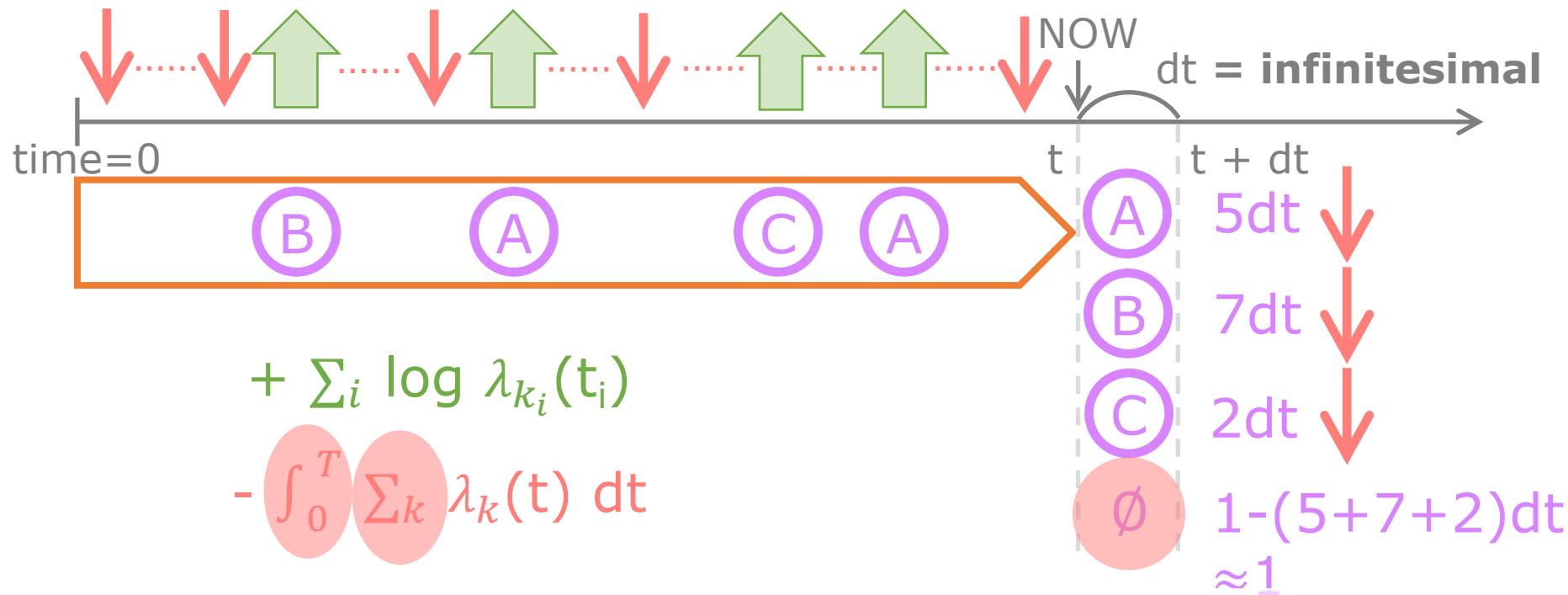


## MLE: Max log prob of *data*



**Integrate over infinitely many *non-events*!**

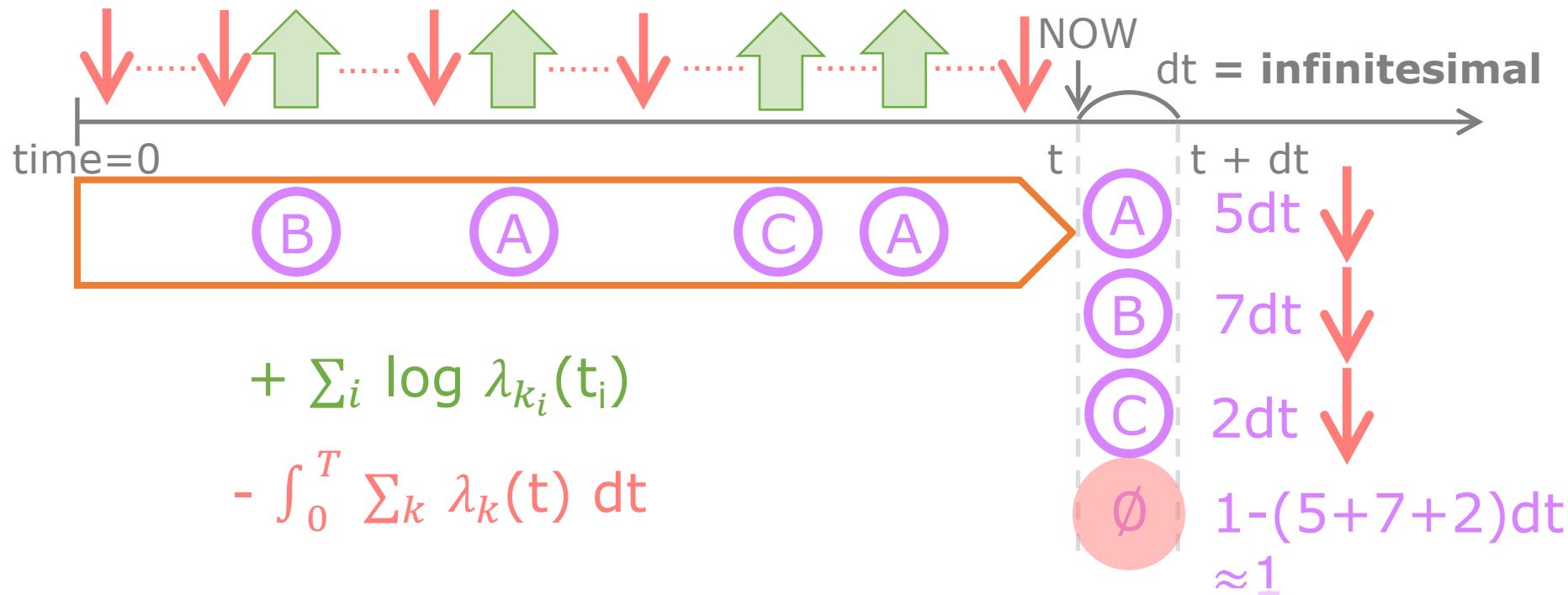
## MLE: Max log prob of *data*



**Integrate over infinitely many *non-events*!**

**Loop over all event types!**

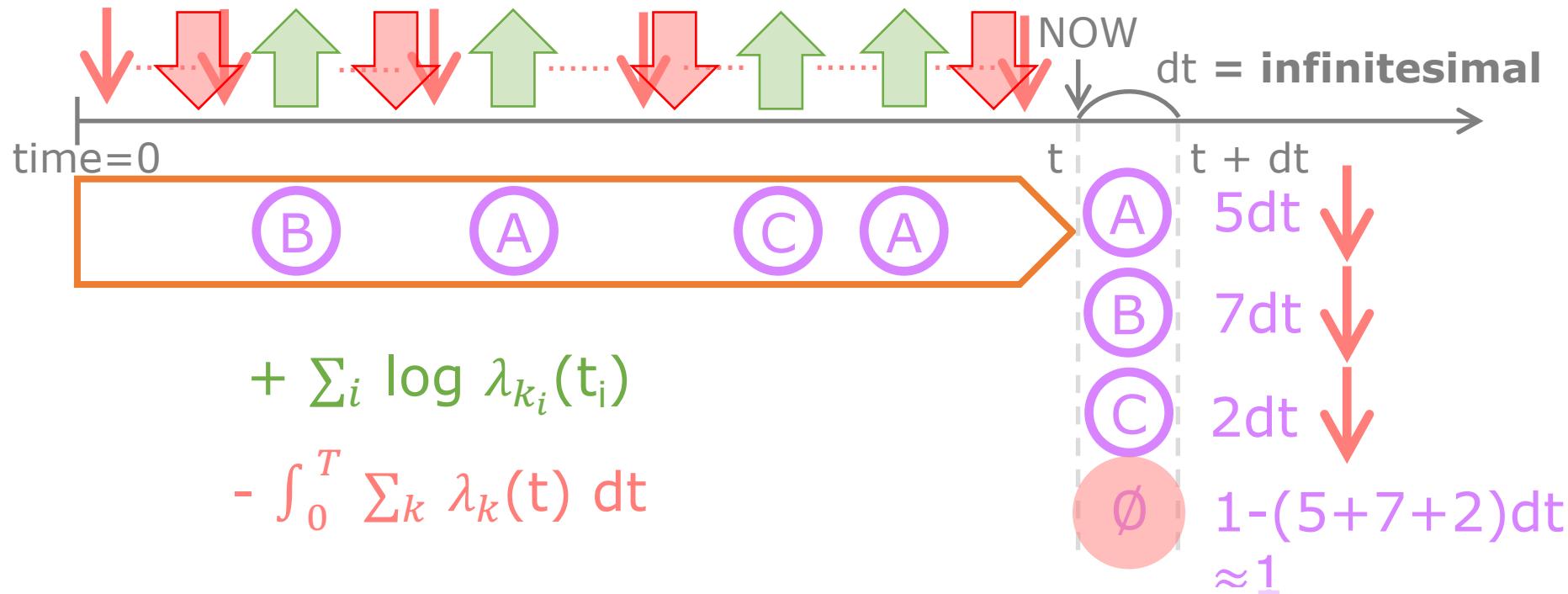
## MLE: Max log prob of *data*



**Integrate over infinitely many *non-events*!**

**Loop over all event types!**

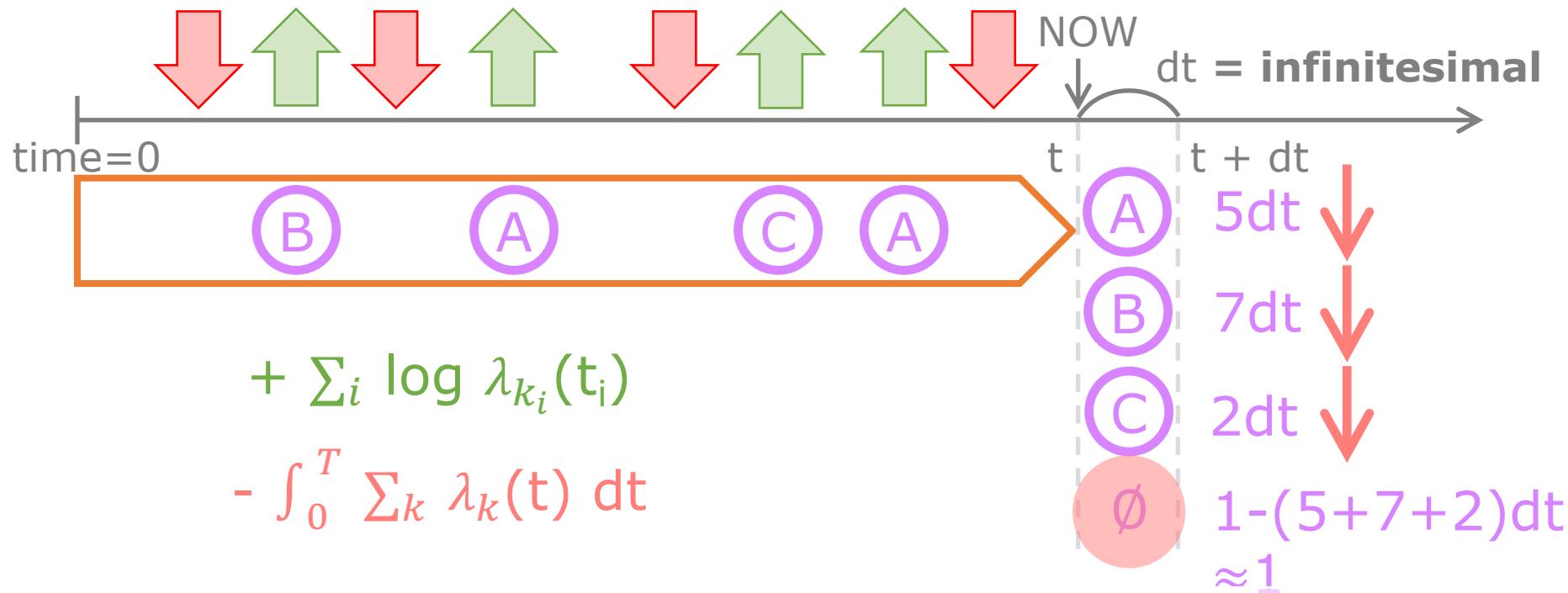
## MLE: Max log prob of *data*



**Integrate over infinitely many *non-events!***

**Loop over all event types!**

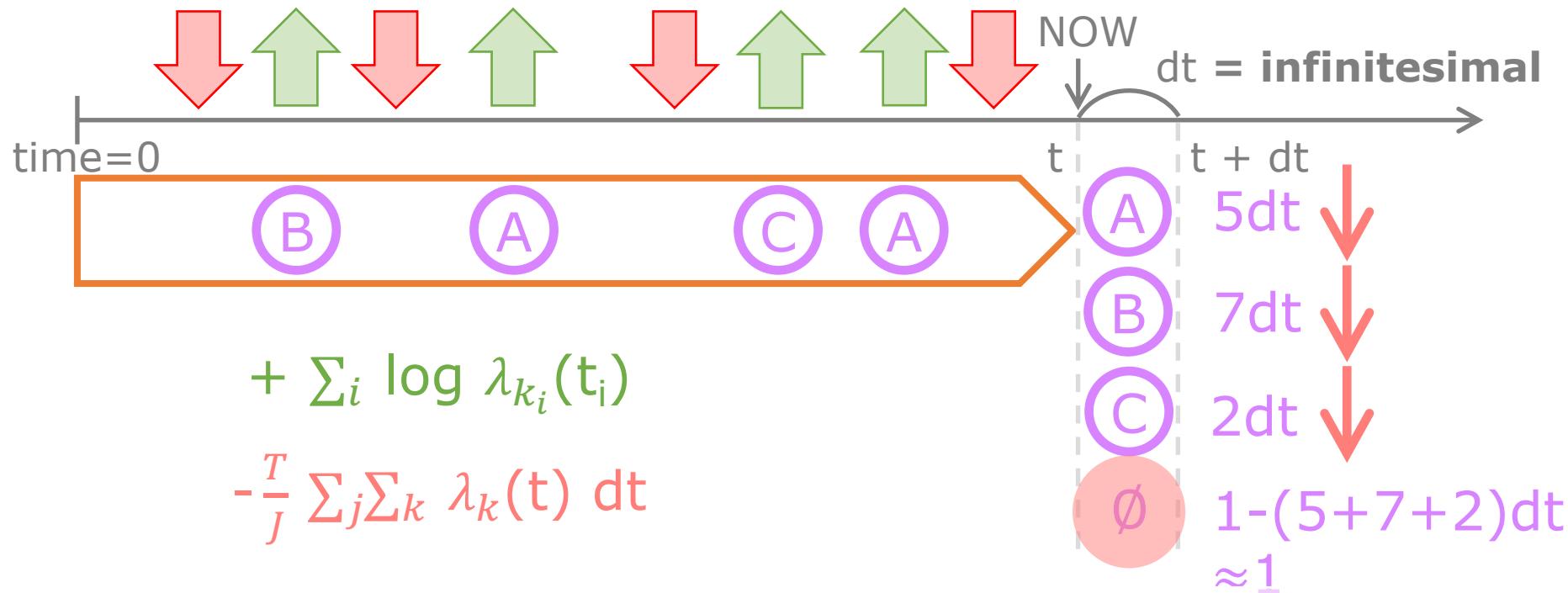
## MLE: Max log prob of *data*



**Integrate over infinitely many *non-events*!**

**Loop over all event types!**

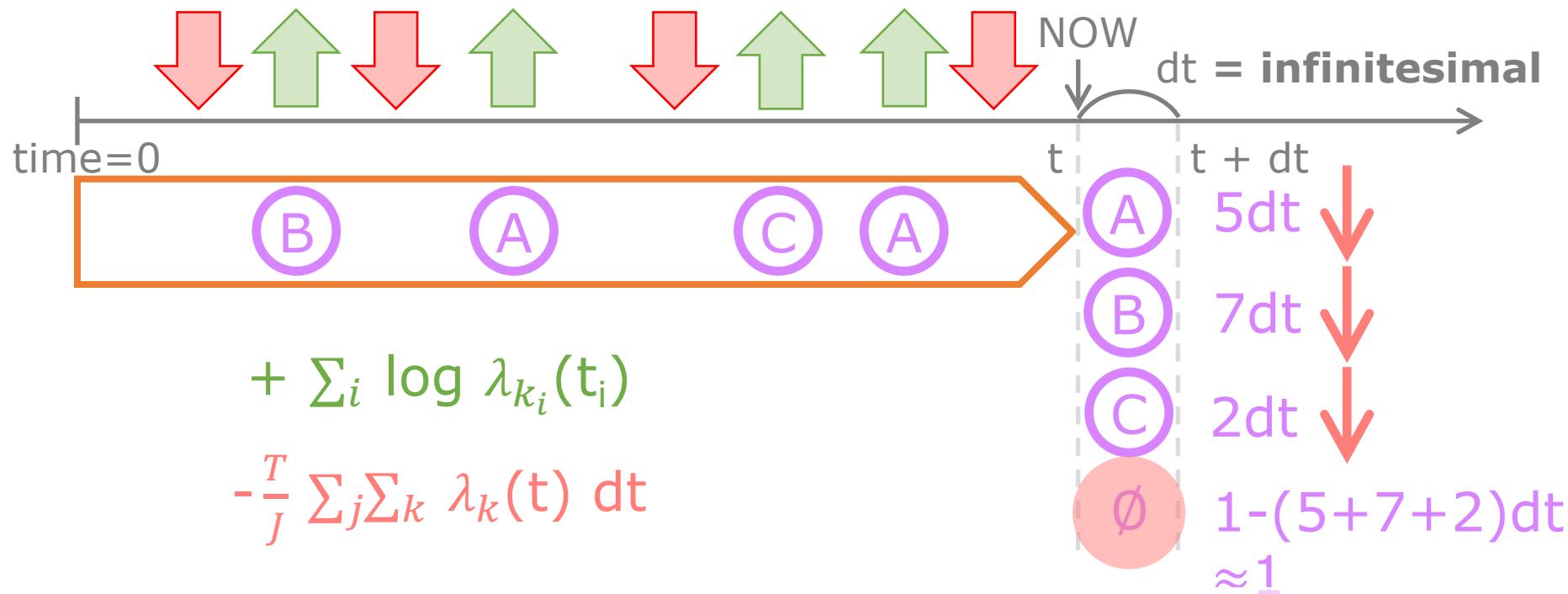
## MLE: Max log prob of *data*



**Integrate over infinitely many *non-events*!**

**Loop over all event types!**

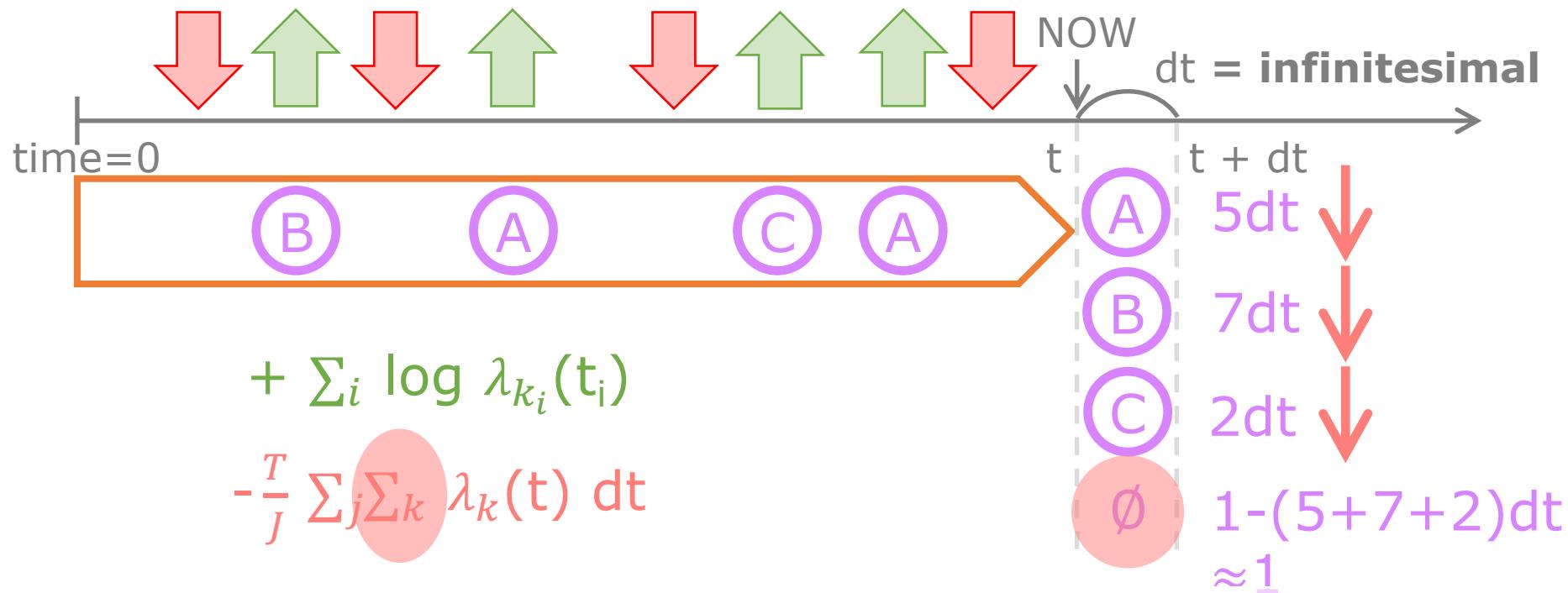
## MLE: Max log prob of *data*



Integrate or Approx by sampling ~~infinitely~~ non-events!

Loop over all event types!

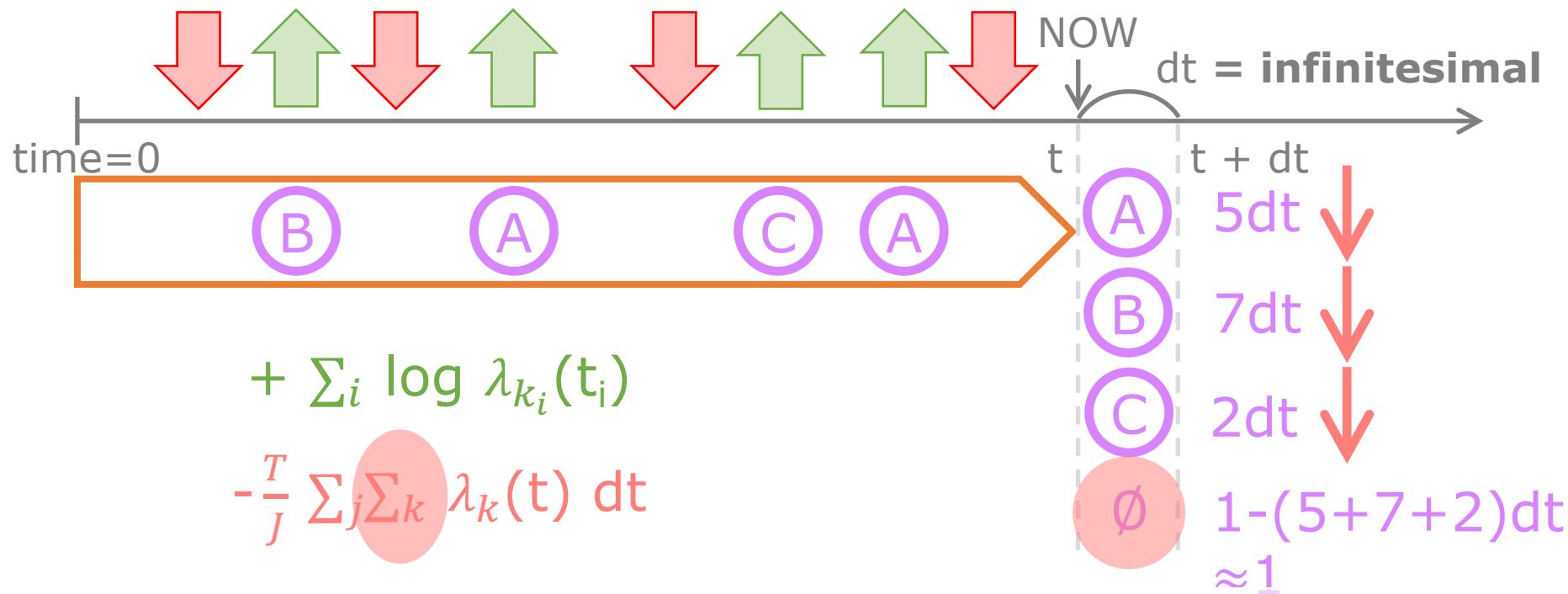
## MLE: Max log prob of *data*



Integrate or ~~Approx by sampling~~ **approximate** non-events!

**Loop over all event types!**

## MLE: Max log prob of *data*



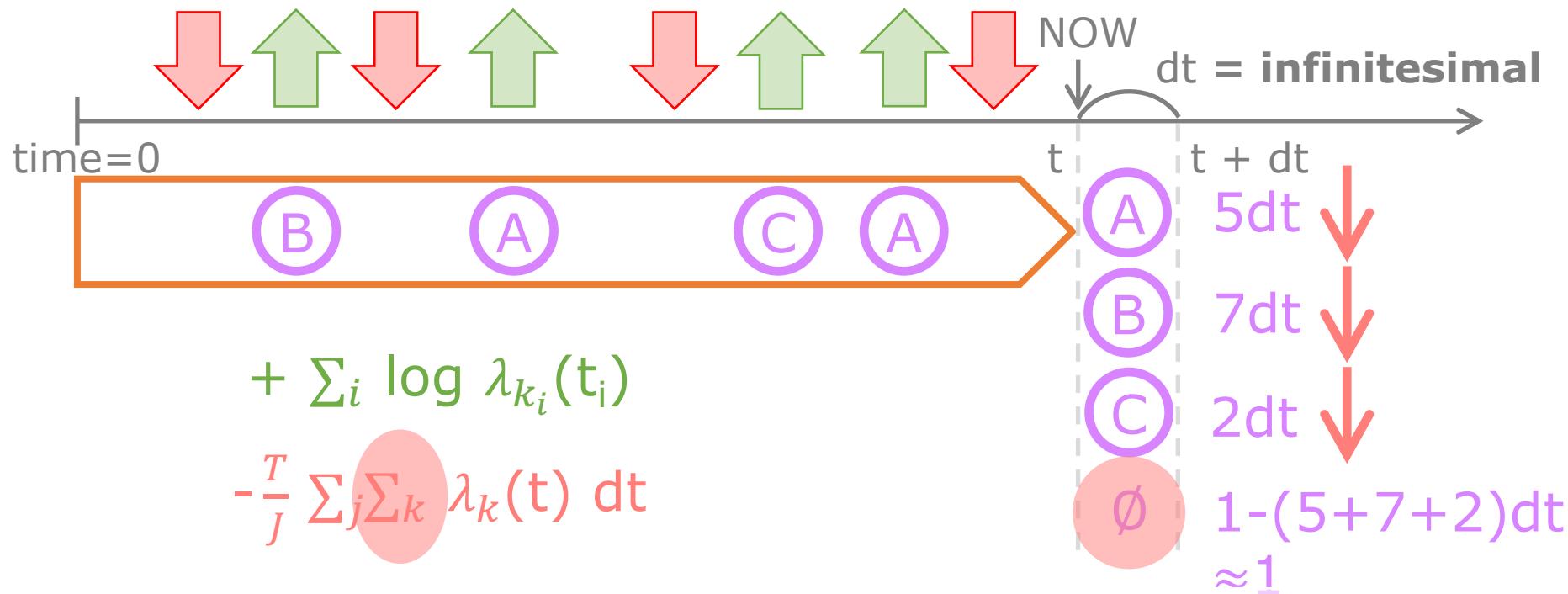
***Still Loop over all event types!***

<http://bburl/tpp-slides-p3>  
<http://bburl/tpp-lab-p3>

## Any Questions?

**<http://bburl/tpp-slides-p3>**

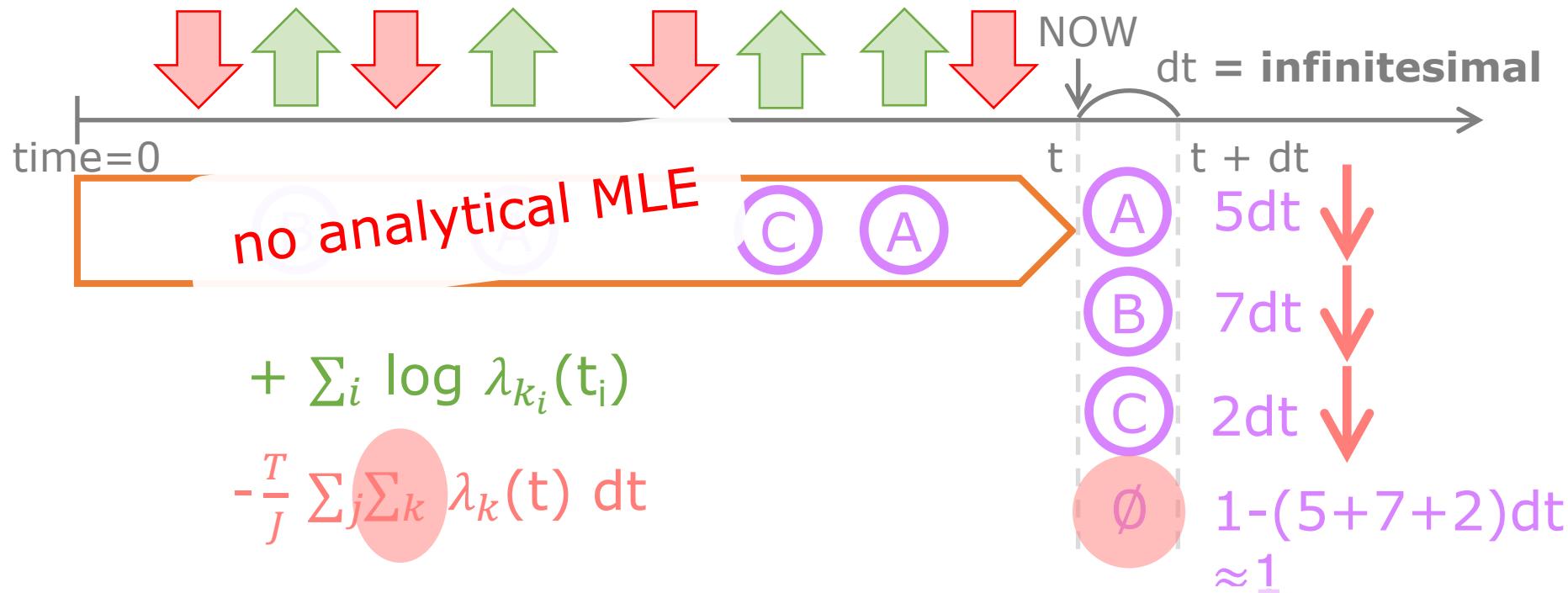
## MLE: Max log prob of *data*



Integrate or ~~Approx by sampling~~ **approximate** non-events!

***Still Loop over all event types!***

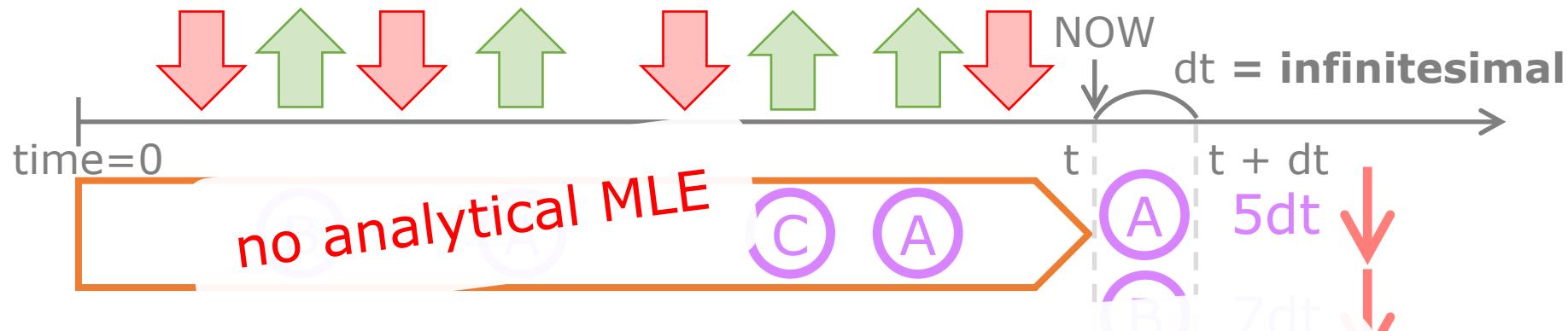
## MLE: Max log prob of *data*



Integrate or ~~Approx by sampling~~ **Approx by sampling non-events!**

***Still Loop over all event types!***

## MLE: Max log prob of *data*



## Stochastic Gradient Descent

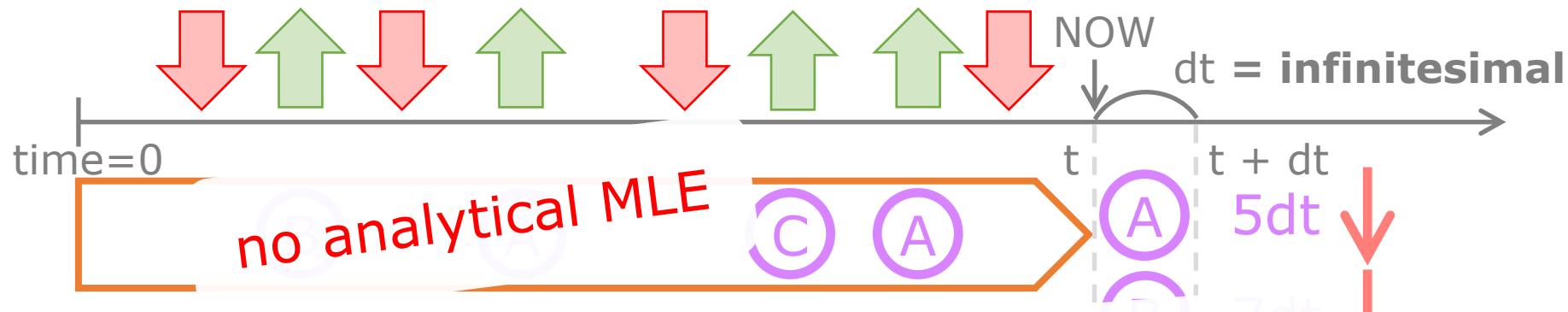
while not converge:

sequence ~ data corpus

*Still* Loop over all event types!

$$\theta \leftarrow \theta + \nabla_{\theta} \text{log-likelihood}$$

## MLE: Max log prob of *data*



## Stochastic Gradient Descent

while not converge:

sequence ~ data corpus

*Still Loop over all event types!*

$$\theta \leftarrow \theta + \nabla_{\theta} \text{log-likelihood}$$

## MLE: Max log prob of *data*



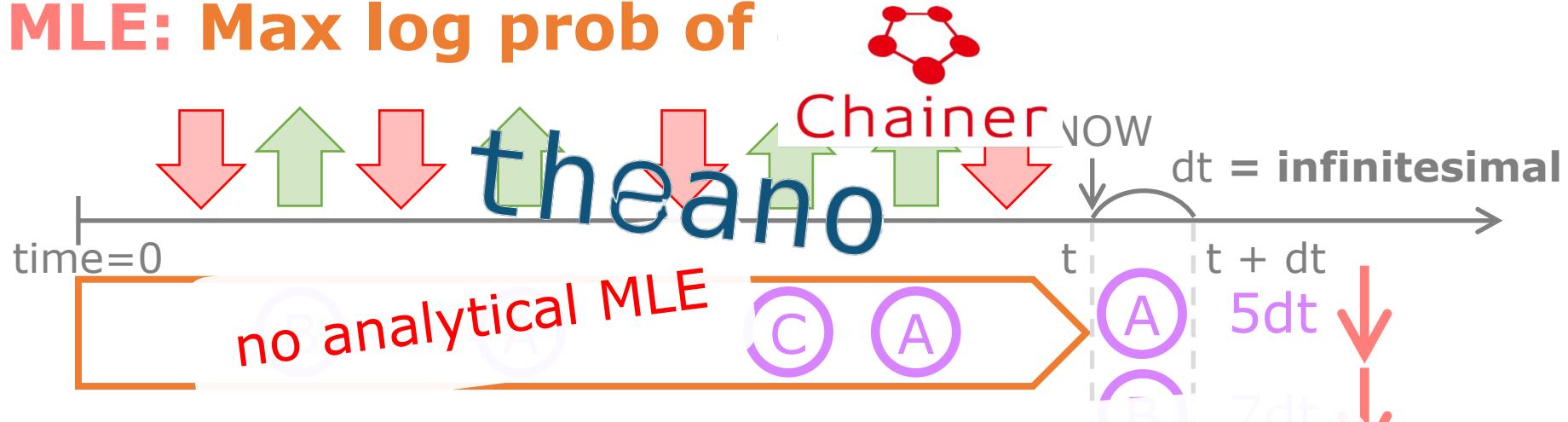
## Stochastic Gradient Descent

while not converge:

sequence ~ data corpus

Still Loop over all event types!  
 $\theta \leftarrow \theta + \nabla_{\theta} \text{log-likelihood}$  ?

## MLE: Max log prob of



## Stochastic Gradient Descent

while not converge:

sequence  $\sim$  data corpus

Still Loop over all event types!  
 $\theta \leftarrow \theta + \nabla_{\theta} \text{log-likelihood}$  ?

## MLE: Max log prob of



## Stochastic Gradient Descent

while not converge:

sequence  $\sim$  data corpus

Still Loop over all event types!  
 $\theta \leftarrow \theta + \nabla_{\theta} \text{log-likelihood}$  ?

## MLE: Max log prob of



## Stochastic Gradient Descent

while not converge:

sequence ~ data corpus

*Still Loop over all event types!*

$$\theta \leftarrow \theta + \nabla_{\theta} \text{log-likelihood}$$


## MLE: Max log prob of



## Stochastic Gradient Descent

while not converge:

sequence ~ data corpus

The Incredible

Still Loop over The Incredibles

$$\theta \leftarrow \theta + \nabla_{\theta} \log \text{likelihood}$$

P Y T ? O R C H

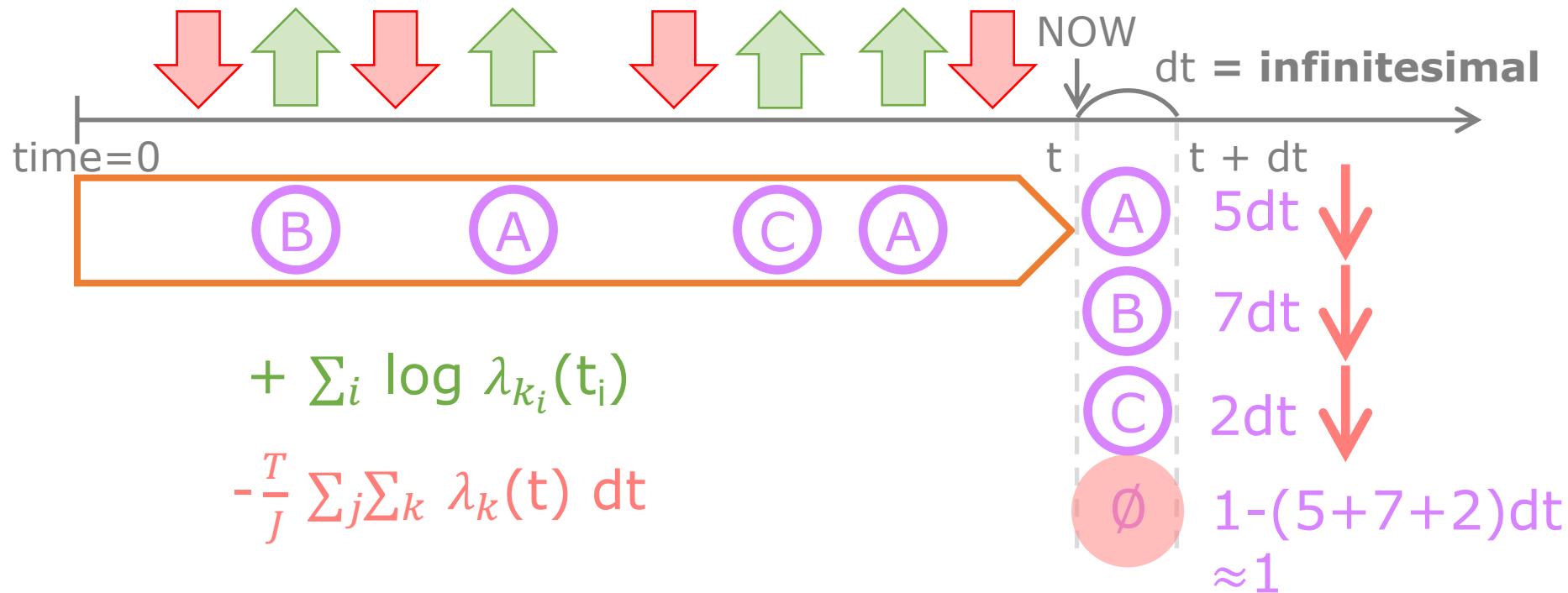


TensorFlow

## Let's write some code

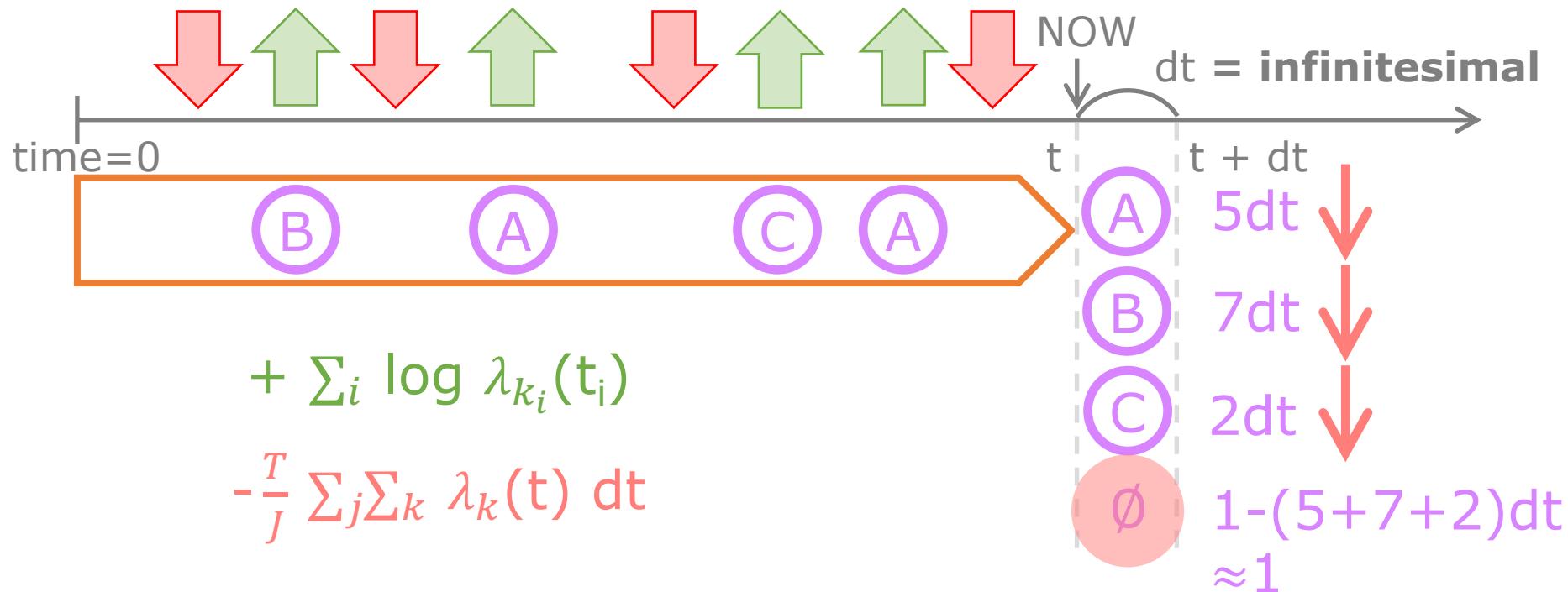
**<http://bburl/tpp-lab-p3>**

## MLE: Max log prob of *data*



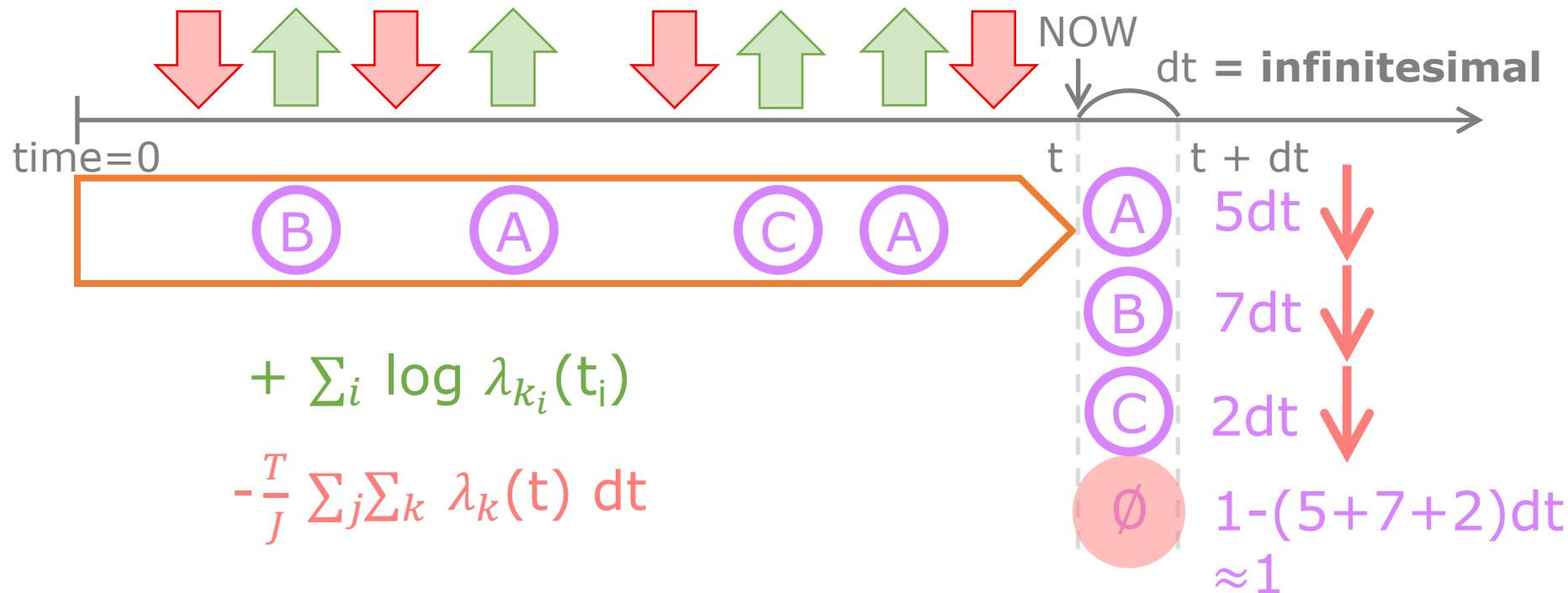
**Loop over all event types!**

## MLE: Max log prob of *data*



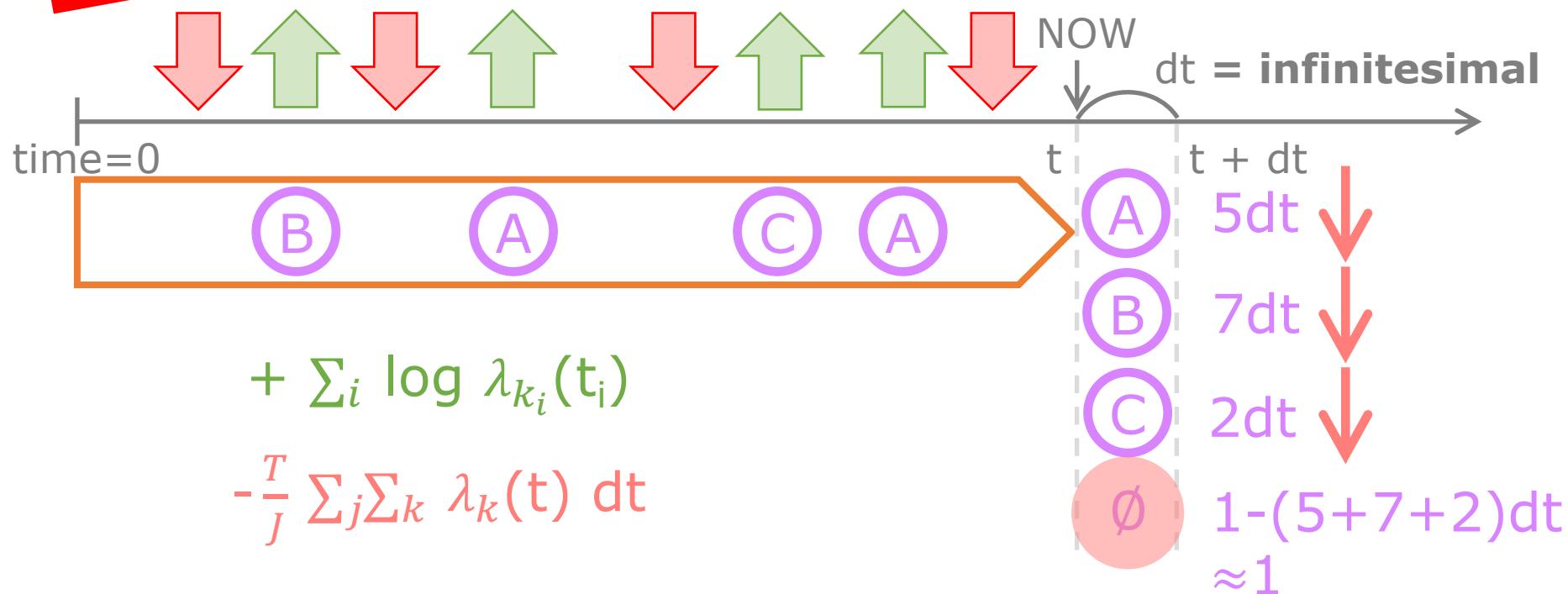
= 50000?  
**Loop over all event types!**

## MLE: Max log prob of *data*



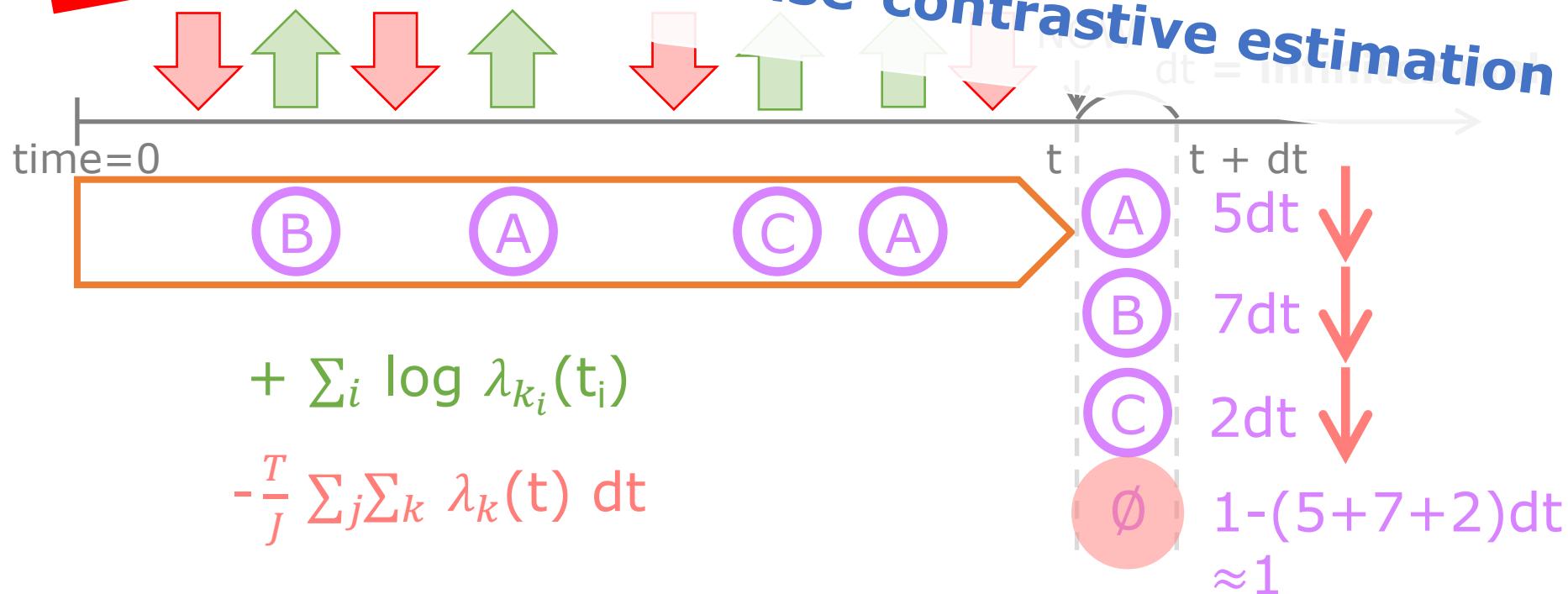
**SLOW!**  
 $= 50000?$   
**Loop over all event types!**

~~MLE: Max log prob of data~~



**SLOW!**  
 $= 50000?$   
**Loop over all event types!**

~~MLE: Max log prob of data~~  
~~noise-contrastive estimation~~



**SLOW!**  
 $= 50000?$   
**Loop over all event types!**

# NCE: Max log prob of *correct discrimination*

# NCE: Max log prob of *correct discrimination*

A

# NCE: Max log prob of *correct discrimination*

log p(A)



# NCE: Max log prob of *correct discrimination*

~~log p( $y_i$ )~~

A

# NCE: Max log prob of *correct discrimination*

~~log p( $y_i$ )~~

A

A

# NCE: Max log prob of *correct discrimination*

~~log p( $y_i$ )~~

A

A

*Is it Real or Noise?*

# NCE: Max log prob of *correct discrimination*

~~$\log p(\mathcal{D})$~~



*Is it Real or Noise?*



# NCE: Max log prob of *correct discrimination*

~~log p(A)~~



*Is it Real or Noise?*



$p(A)$

# NCE: Max log prob of *correct discrimination*

~~$\log p(A)$~~



*Is it Real or Noise?*



$p(A)$



# NCE: Max log prob of *correct discrimination*

~~log p(A)~~



*Is it Real or Noise?*

  $p(A)$

  $q(A)$

# NCE: Max log prob of *correct discrimination*

~~$\log p(A)$~~

A

A

*Is it Real or Noise?*

A       $p(A)$

A       $q(A)$

$q$  = a cheaper model

# NCE: Max log prob of *correct discrimination*

~~log p(A)~~

$$\log \frac{p(A)}{p(A) + q(A)}$$

*Is it Real or Noise?*

$$\boxed{A} \quad p(A)$$

$$\boxed{A} \quad q(A)$$

**q = a cheaper model**

# NCE: Max log prob of *correct discrimination*

~~$\log p(A)$~~

$$\log \frac{p(A)}{p(A) + q(A)}$$

*Is it Real or Noise?*

$$\boxed{A} \quad p(A) \quad \uparrow$$

$$\boxed{A} \quad q(A)$$

$q$  = a cheaper model

# NCE: Max log prob of *correct discrimination*

~~$\log p(A)$~~

$$\log \frac{p(A)}{p(A) + q(A)}$$

B

*Is it Real or Noise?*

$$A \quad p(A) \quad \uparrow$$

$$A \quad q(A)$$

$q = \text{a cheaper model}$

# NCE: Max log prob of *correct discrimination*

~~$\log p(A)$~~

$$\log \frac{p(A)}{p(A) + q(A)}$$

B

B

*Is it Real or Noise?*

$$A \quad p(A) \quad \uparrow$$

$$A \quad q(A)$$

$q = \text{a cheaper model}$

# NCE: Max log prob of *correct discrimination*

~~$\log p(A)$~~

$$\log \frac{p(A)}{p(A) + q(A)}$$

B

B

*Is it Real or Noise?*

$$A \quad p(A) \quad \uparrow$$

B

$$A \quad q(A)$$

$q = \text{a cheaper model}$

# NCE: Max log prob of *correct discrimination*

~~$\log p(A)$~~

$$\log \frac{p(A)}{p(A) + q(A)}$$

B

B

*Is it Real or Noise?*

$$A \quad p(A) \quad \uparrow$$

$$B \quad p(B)$$

$$A \quad q(A)$$

$q = \text{a cheaper model}$

# NCE: Max log prob of *correct discrimination*

~~$\log p(A)$~~

$$\log \frac{p(A)}{p(A) + q(A)}$$

B

B

*Is it Real or Noise?*

$$A \quad p(A) \quad \uparrow$$

$$B \quad p(B)$$

$$A \quad q(A)$$

B

$q = \text{a cheaper model}$

# NCE: Max log prob of *correct discrimination*

~~log p(A)~~

$$\text{A} \quad \log \frac{p(A)}{p(A) + q(A)}$$

B

B

*Is it Real or Noise?*

$$\text{A} \quad p(A) \quad \uparrow$$

B  $p(B)$

$$\text{A} \quad q(A)$$

B  $q(B)$

$q$  = a cheaper model

# NCE: Max log prob of *correct discrimination*

~~$\log p(A)$~~

$$\boxed{A} \quad \log \frac{p(A)}{p(A) + q(A)}$$

$$\boxed{B} \quad \log \frac{q(B)}{p(B) + q(B)}$$

*Is it Real or Noise?*

$$\boxed{A} \quad p(A) \quad \uparrow$$

$$\boxed{B} \quad p(B)$$

$$\boxed{A} \quad q(A)$$

$$\boxed{B} \quad q(B)$$

$q$  = a cheaper model

# NCE: Max log prob of *correct discrimination*

~~$\log p(A)$~~

$$\boxed{A} \quad \log \frac{p(A)}{p(A) + q(A)}$$

$$\boxed{B} \quad \log \frac{q(B)}{p(B) + q(B)}$$

*Is it Real or Noise?*

$$\boxed{A} \quad p(A) \quad \uparrow$$

$$\boxed{B} \quad p(B) \quad \downarrow$$

$$\boxed{A} \quad q(A)$$

$$\boxed{B} \quad q(B)$$

$q$  = a cheaper model

# NCE: Max log prob of *correct discrimination*

~~log p(A)~~

**Binary-Classification Version**  
**Gutmann & Hyvarinen 2010**

$$\boxed{A} \quad \log \frac{p(A)}{p(A) + q(A)}$$

$$\boxed{B} \quad \log \frac{q(B)}{p(B) + q(B)}$$

*Is it Real or Noise?*

$$\boxed{A} \quad p(A) \quad \uparrow$$

$$\boxed{B} \quad p(B) \quad \downarrow$$

$$\boxed{A} \quad q(A)$$

$$\boxed{B} \quad q(B)$$

$q$  = a cheaper model

# NCE: Max log prob of *correct discrimination*

A

A

B

B

## NCE: Max log prob of *correct discrimination*

A

A

B

B

***Which Is Real?***

# NCE: Max log prob of *correct discrimination*

A

A

B

B

A

B

***Which Is Real?***

# NCE: Max log prob of *correct discrimination*

A

A

B

B

***Which Is Real?***

A

$p(A)q(B)$

B

## NCE: Max log prob of *correct discrimination*

A

A

B

B

***Which Is Real?***

A

$p(A)q(B)$

B

A

B

# NCE: Max log prob of *correct discrimination*

A

A

B

B

***Which Is Real?***

A

$p(A)q(B)$

B

A

$p(B)q(A)$

B

## NCE: Max log prob of *correct discrimination*

A

A

B

B

*Which Is Real?*

A

$p(A)q(B)$

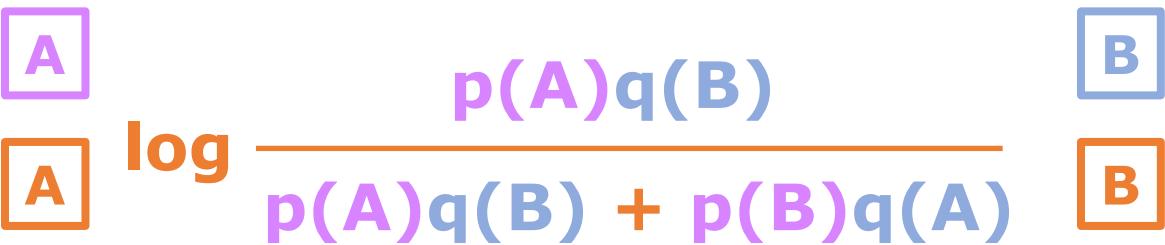
B

A

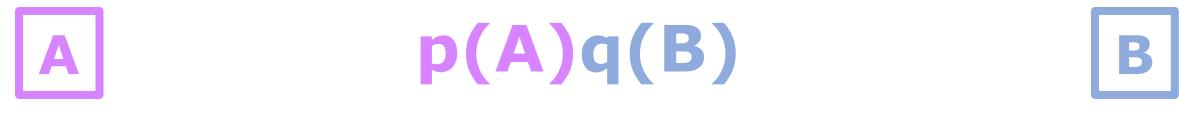
$p(B)q(A)$

B

# NCE: Max log prob of *correct discrimination*

$$\log \frac{p(A)q(B)}{p(A)q(B) + p(B)q(A)}$$


***Which Is Real?***

$$p(A)q(B)$$


$$p(B)q(A)$$


# NCE: Max log prob of *correct discrimination*

$$\log \frac{p(A)q(B)}{p(A)q(B) + p(B)q(A)}$$

A    B  
A    B  
p(A)                                        ↑

*Which Is Real?*

$$p(A)q(B)$$

A    B

$$p(B)q(A)$$

A    B

# NCE: Max log prob of *correct discrimination*

$$\log \frac{p(A)q(B)}{p(A)q(B) + p(B)q(A)}$$

$p(A)$  

*Which Is Real?*

$$p(A)q(B)$$

$p(B)$  

$$p(B)q(A)$$

# NCE: Max log prob of *correct discrimination*

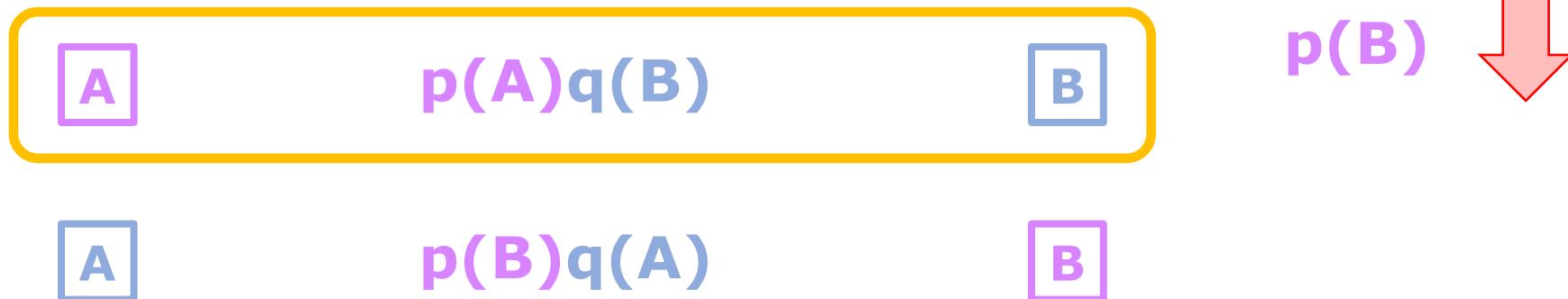
Ranking Version  
Ma & Collins 2018

$$\log \frac{p(A)q(B)}{p(A)q(B) + p(B)q(A)}$$

A                            B  
A                            B

$p(A)$  ↑  
 $p(B)$  ↓

*Which Is Real?*



# NCE: Max log prob of *correct discrimination*

Consistency under  
*weaker assumptions*

Ranking Version  
Ma & Collins 2018

$$\log \frac{p(A)q(B)}{p(A)q(B) + p(B)q(A)}$$

*Which Is Real?*



$$\boxed{A} \quad p(A)q(B) \quad \boxed{B}$$

$$\boxed{A} \quad p(B)q(A) \quad \boxed{B}$$

<http://bburl/tpp-slides-p3>  
<http://bburl/tpp-lab-p3>

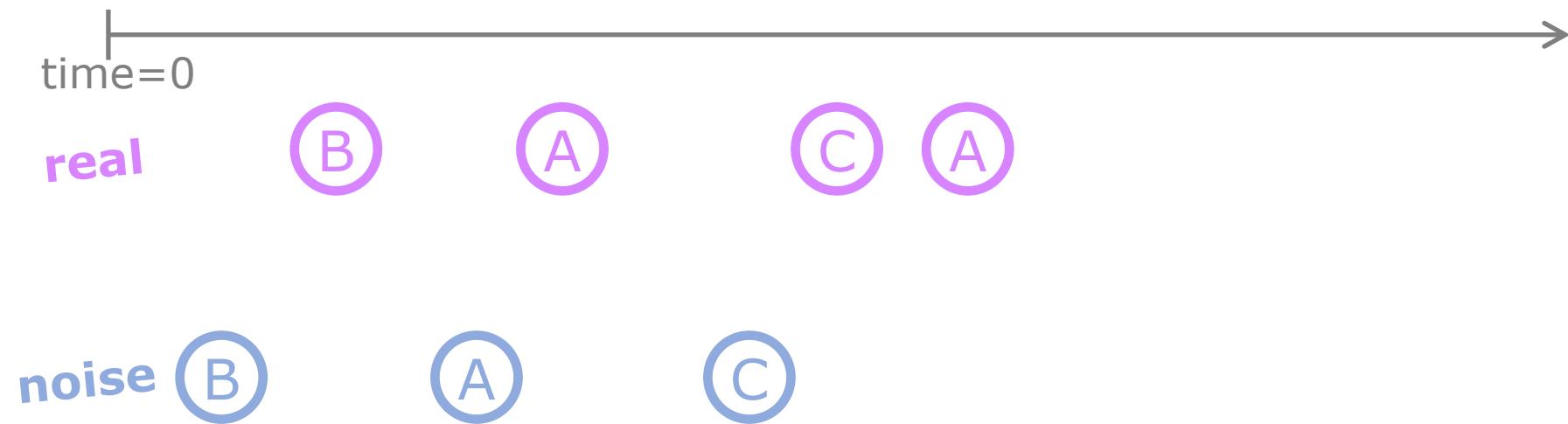
## Any Questions?

**<http://bburl/tpp-slides-p3>**

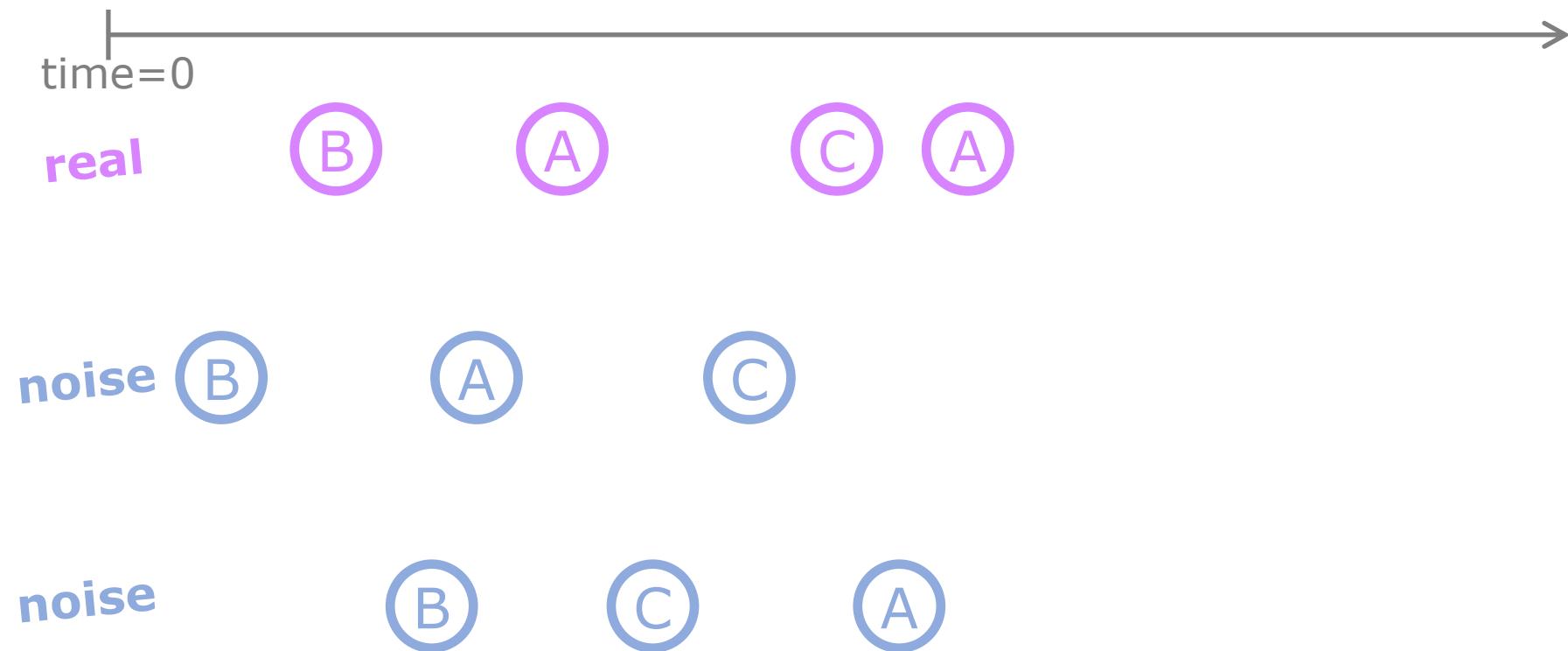
## NCE: Max log prob of *correct discrimination*



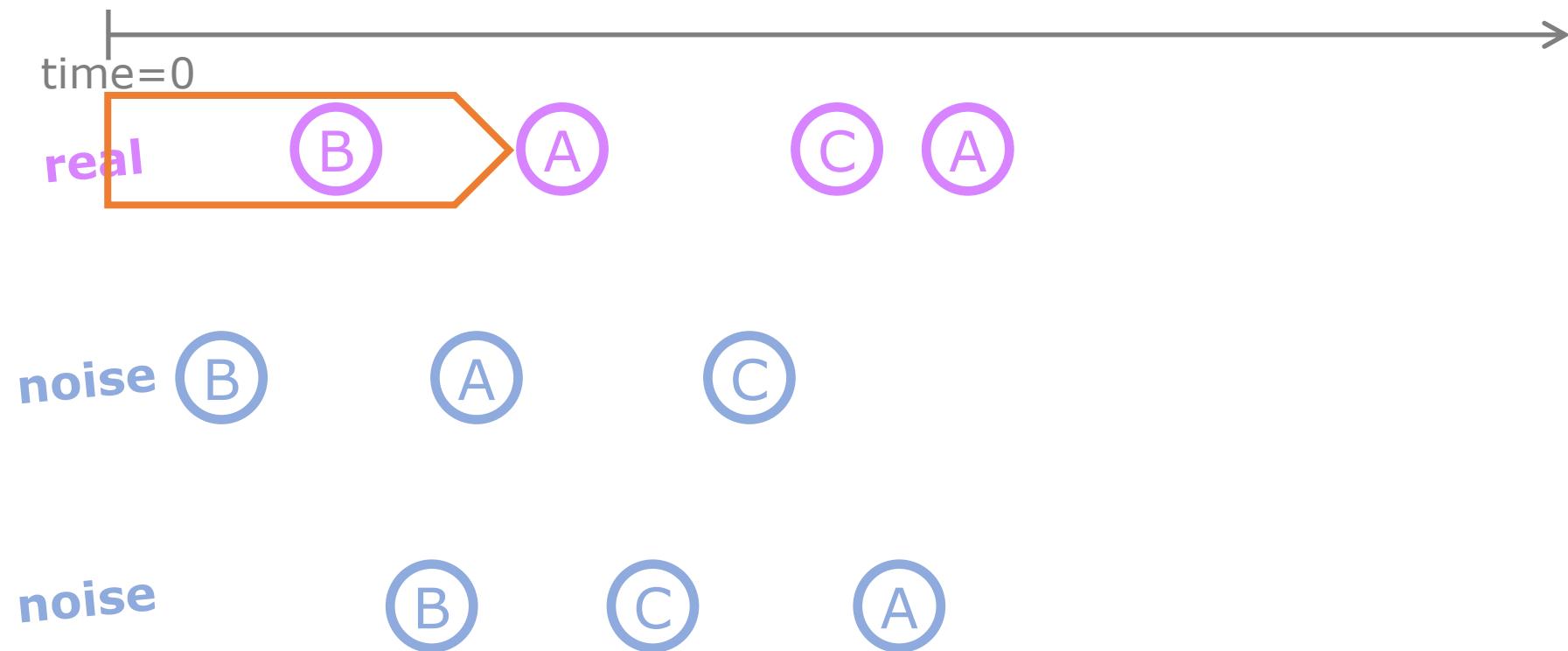
## NCE: Max log prob of *correct discrimination*



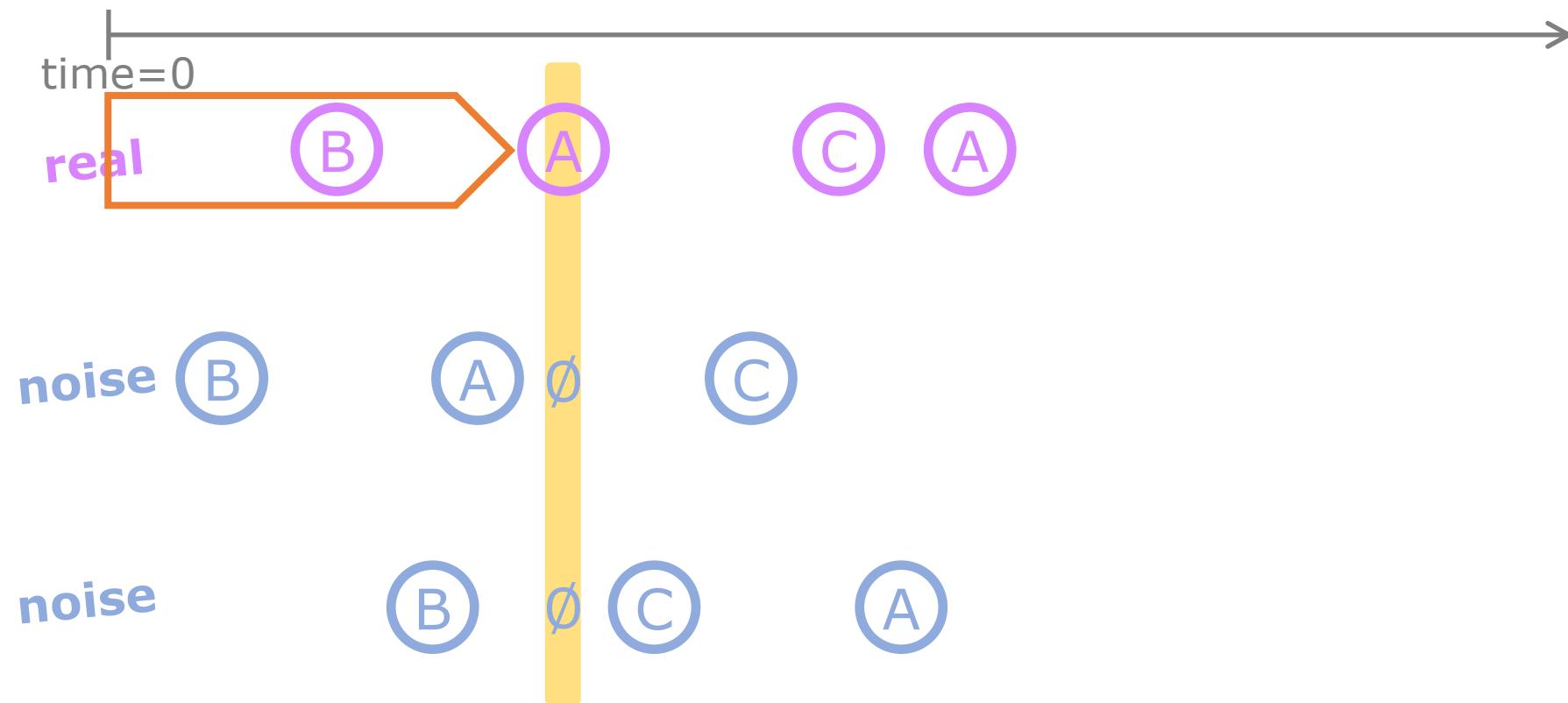
## NCE: Max log prob of *correct discrimination*



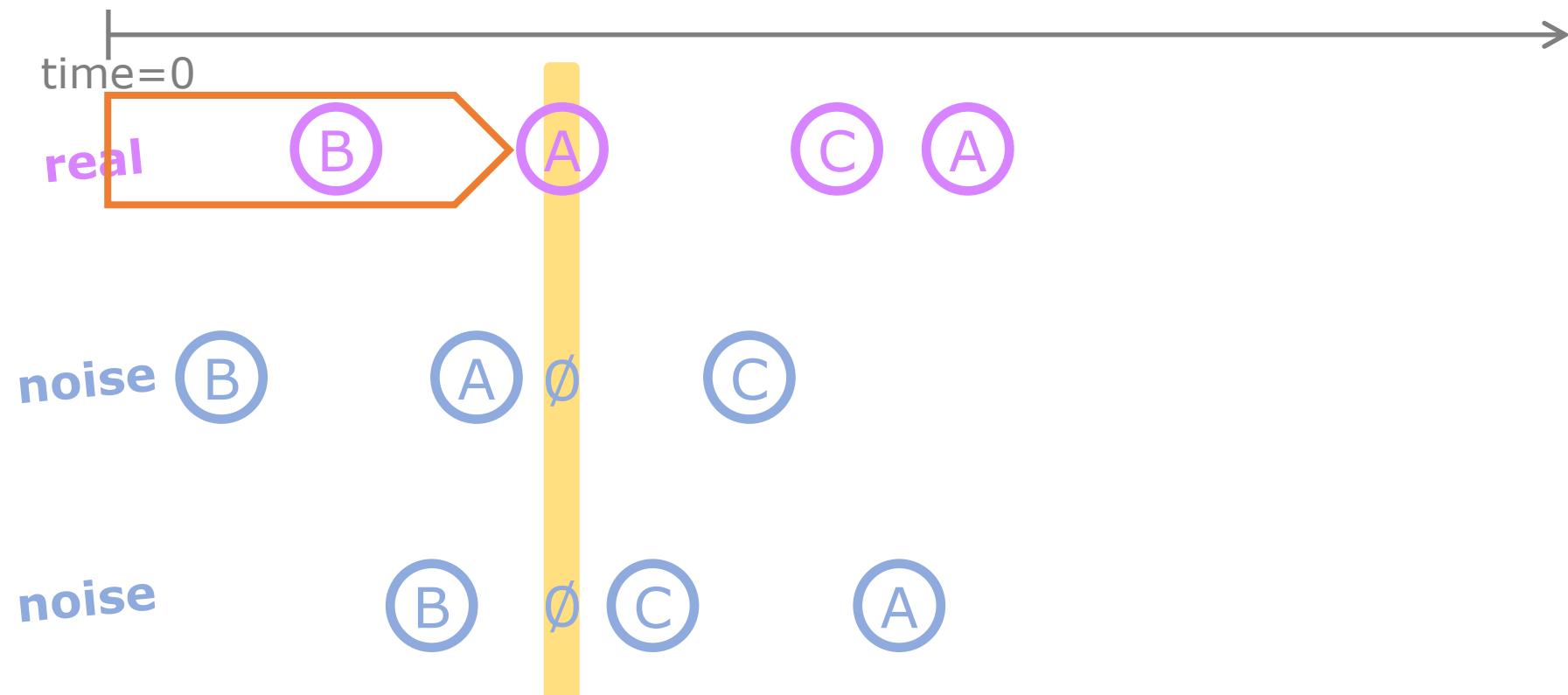
## NCE: Max log prob of *correct discrimination*



## NCE: Max log prob of *correct discrimination*

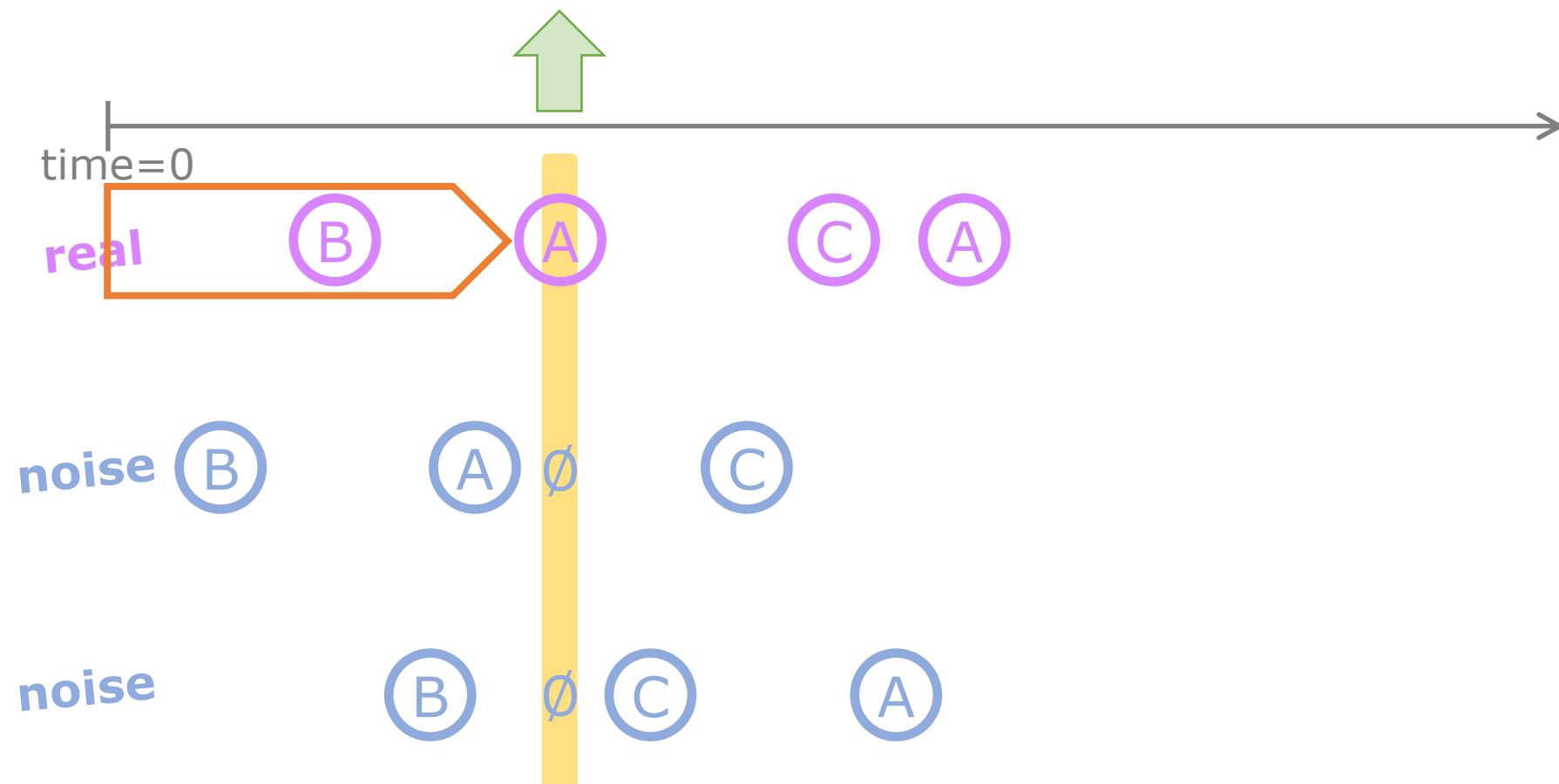


## NCE: Max log prob of *correct discrimination*

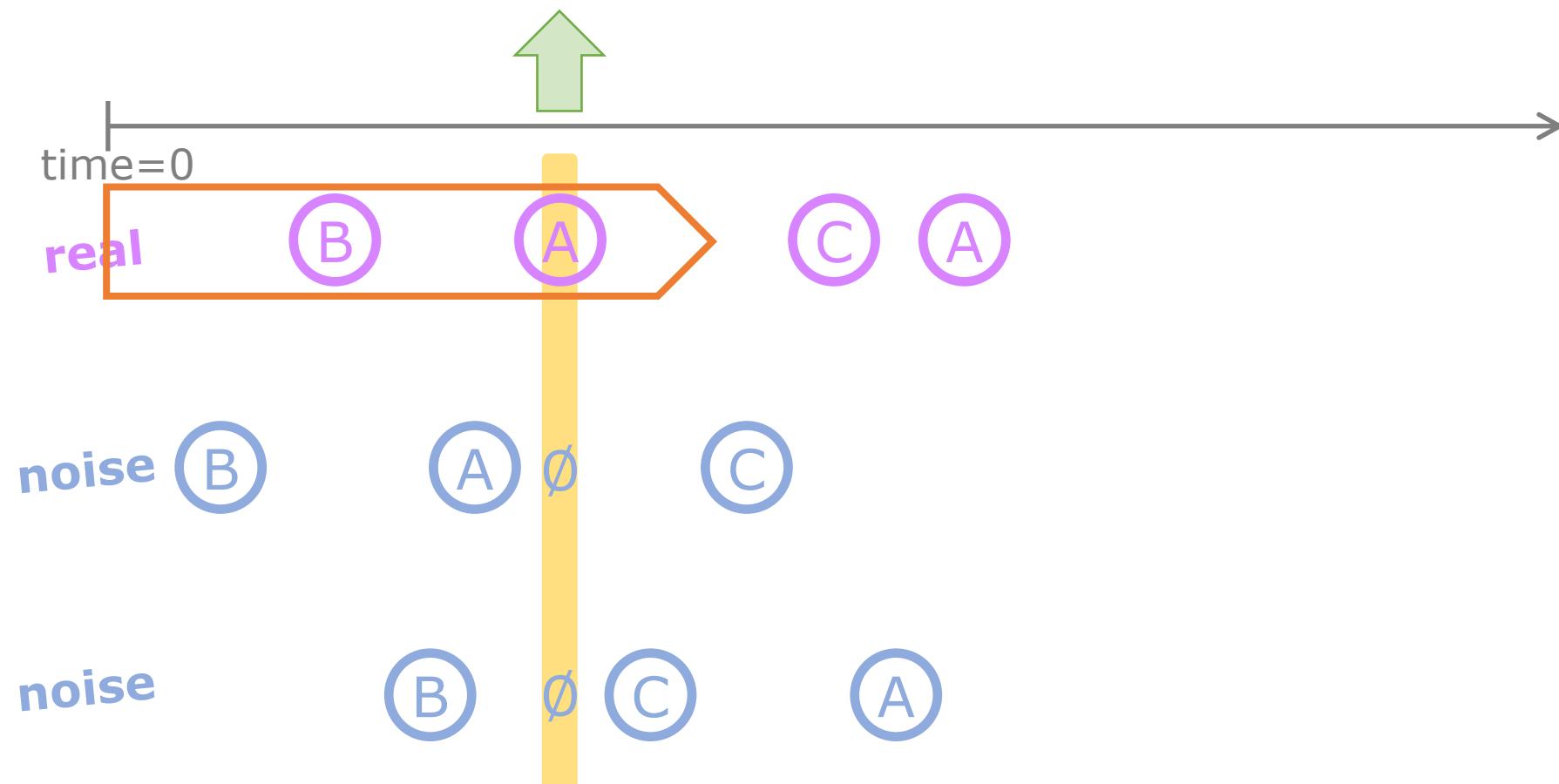


***Which Is Real?***

## NCE: Max log prob of *correct discrimination*

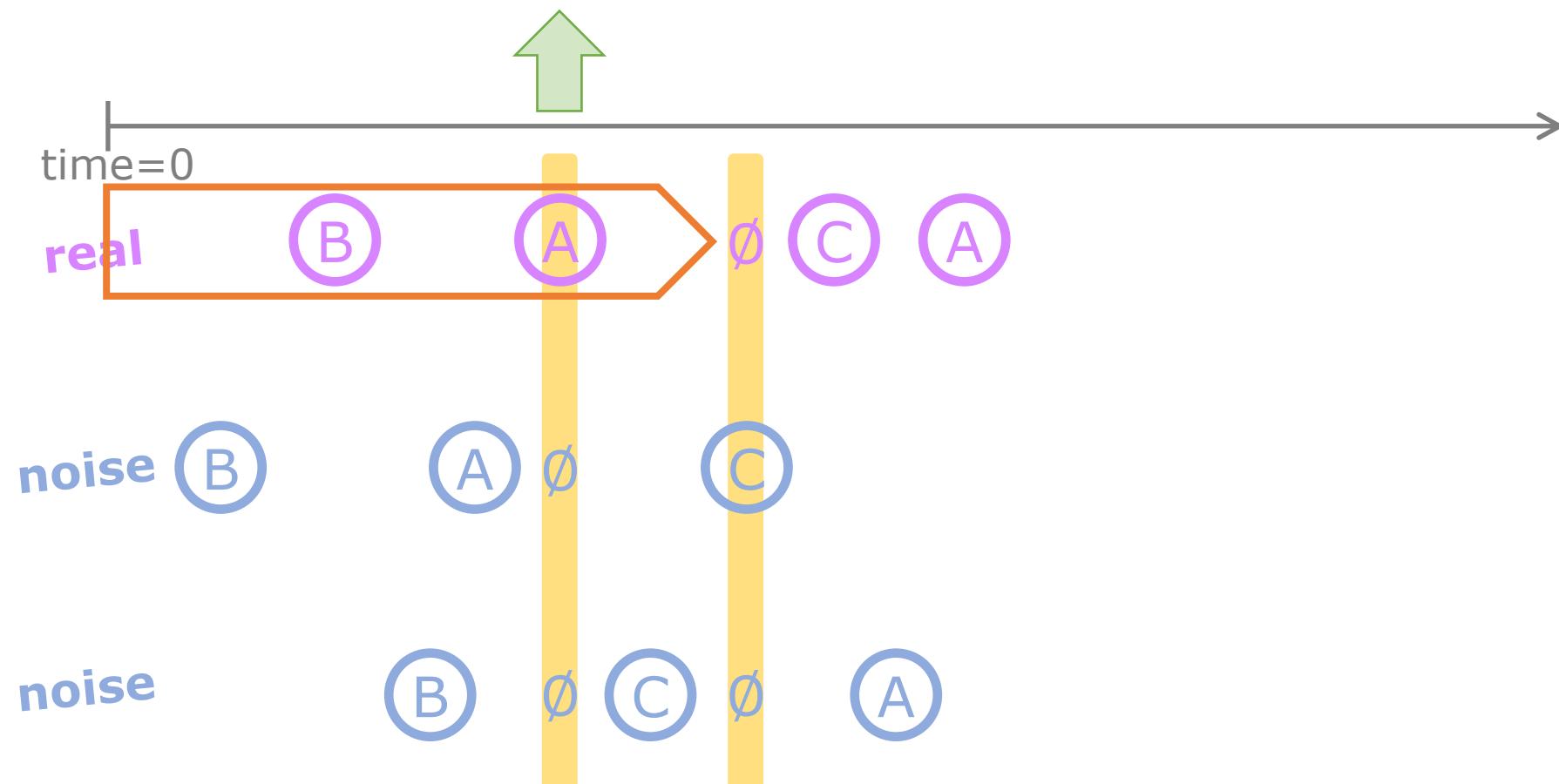


## NCE: Max log prob of *correct discrimination*



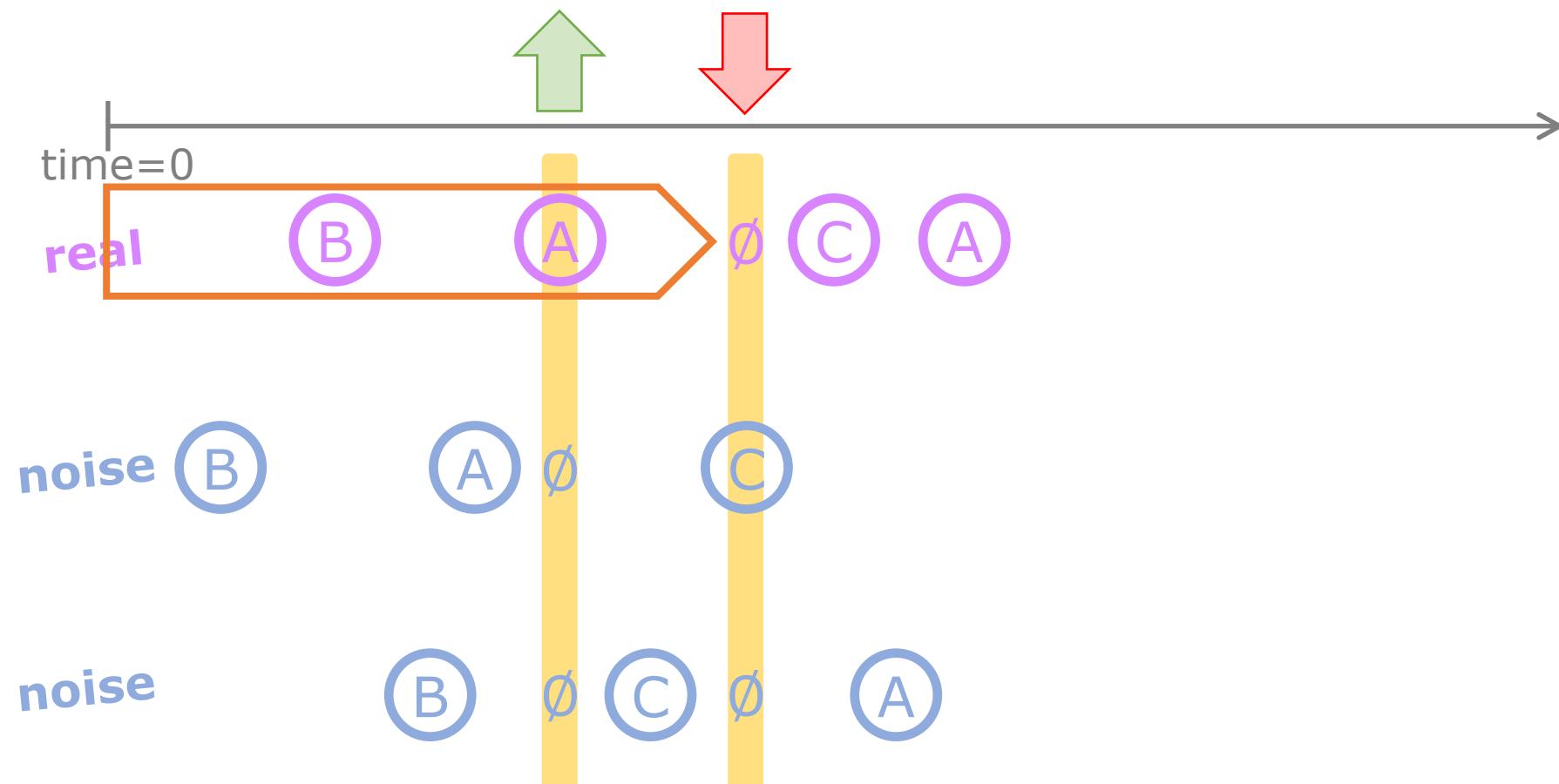
***Which Is Real?***

## NCE: Max log prob of *correct discrimination*

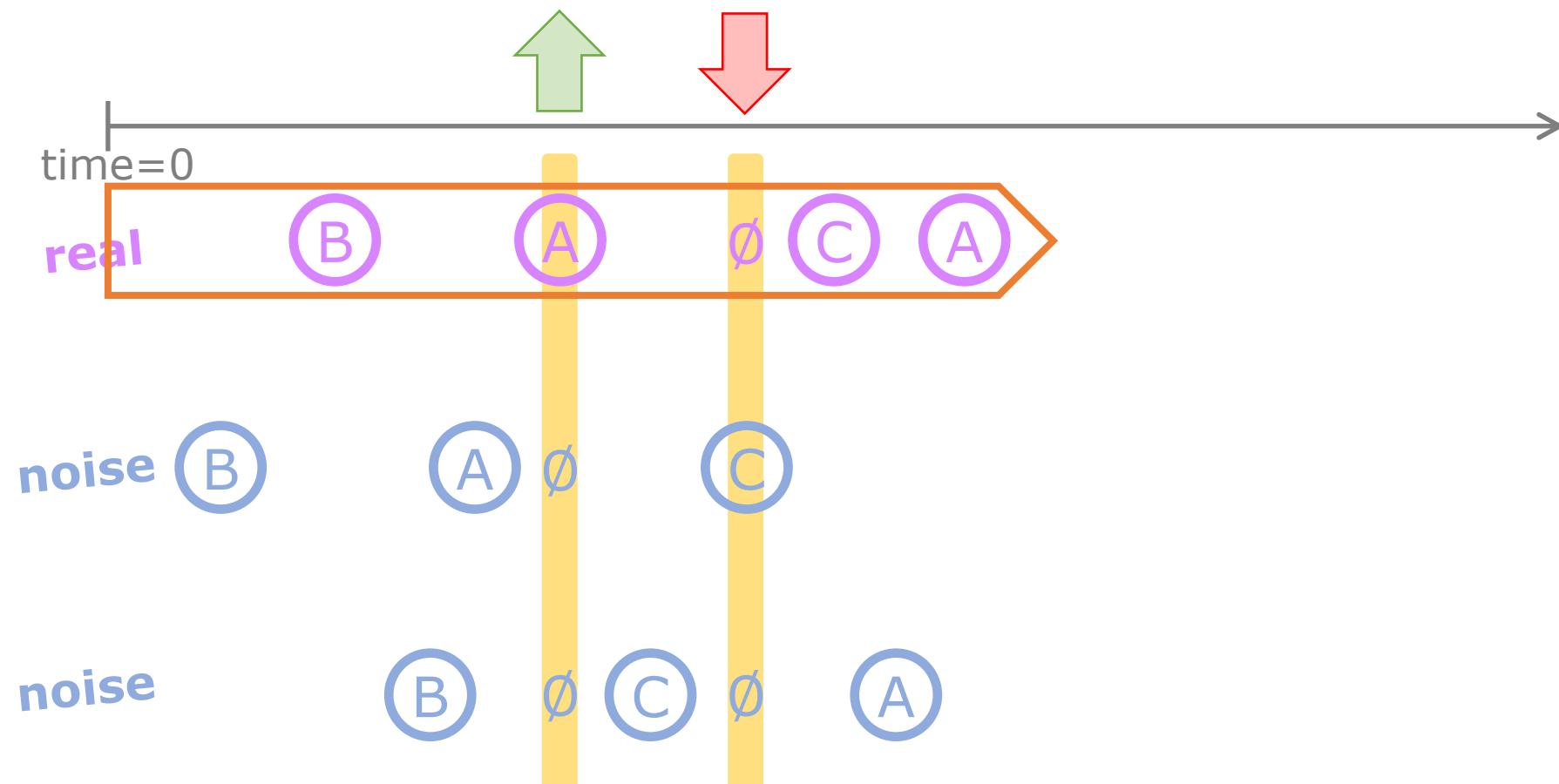


***Which Is Real?***

## NCE: Max log prob of *correct discrimination*

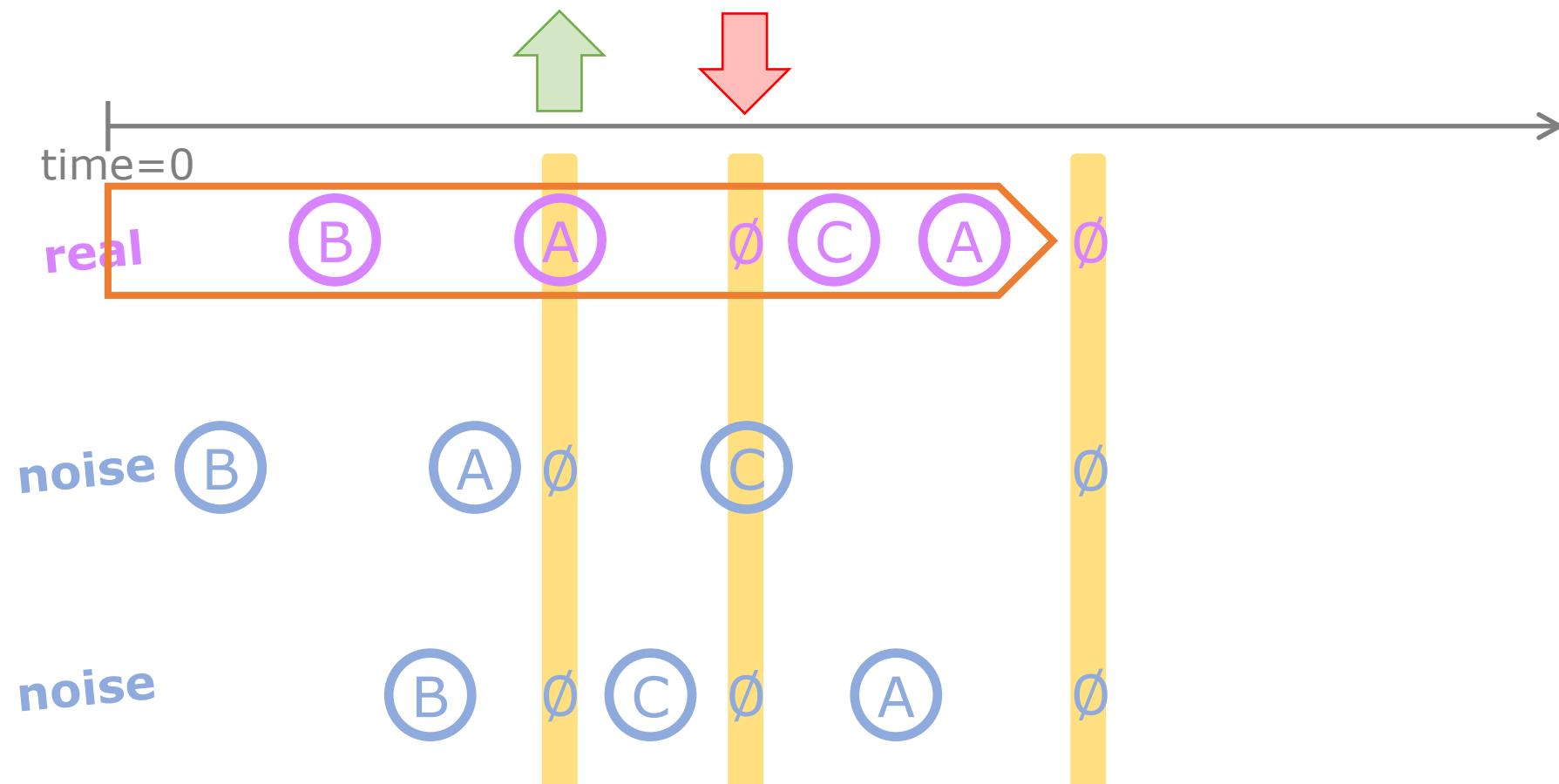


## NCE: Max log prob of *correct discrimination*



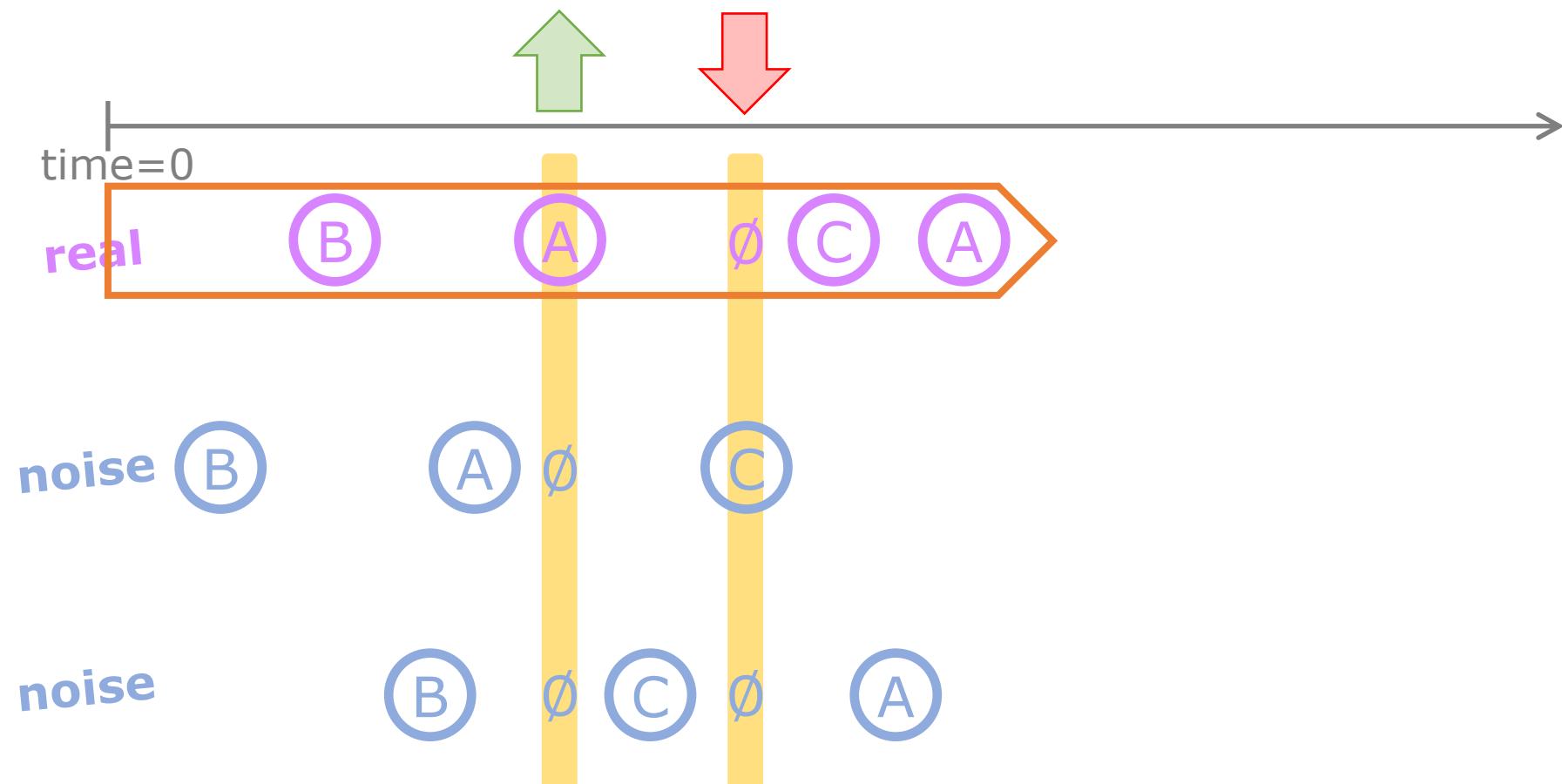
***Which Is Real?***

## NCE: Max log prob of *correct discrimination*



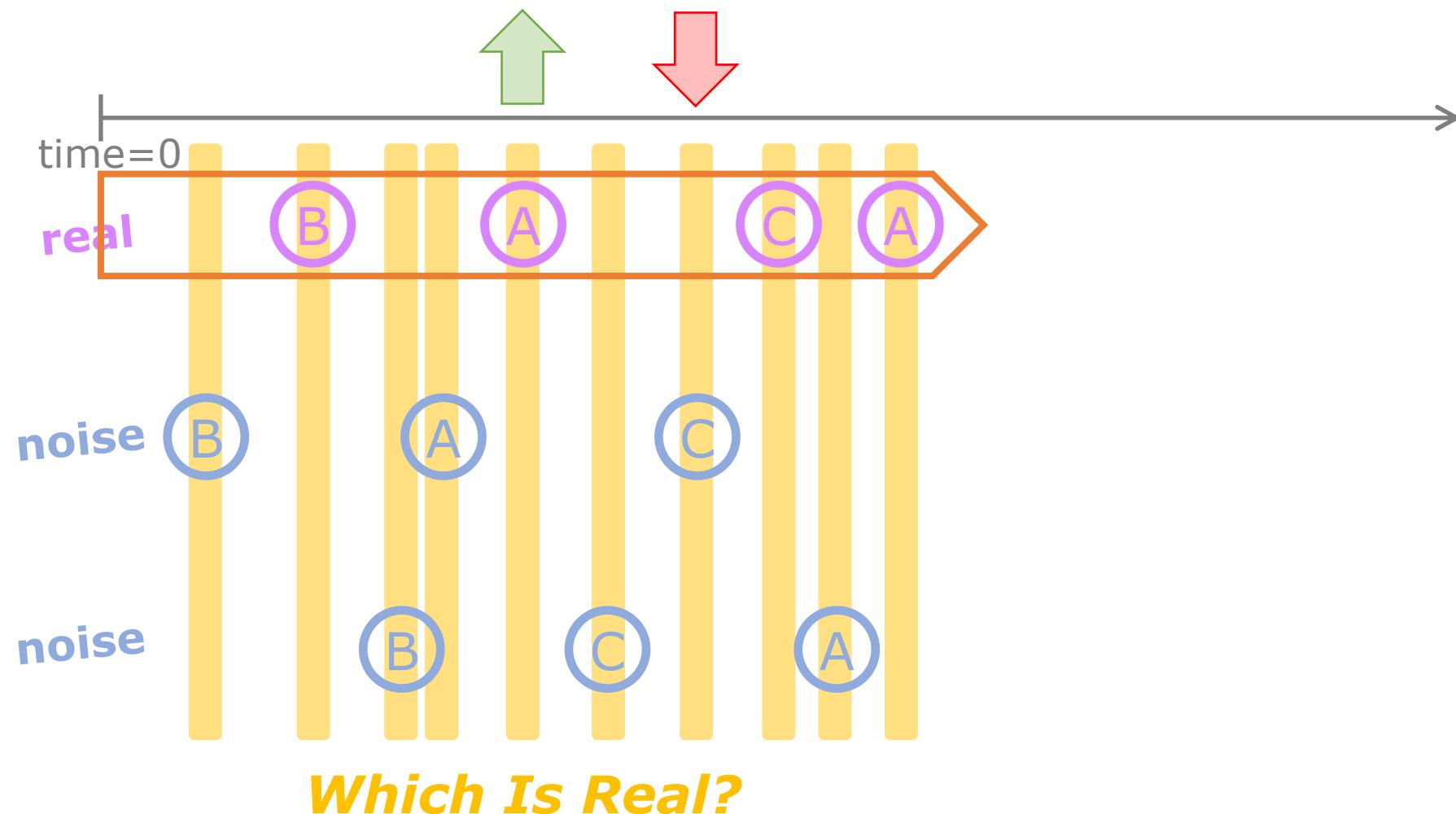
***Which Is Real?***

## NCE: Max log prob of *correct discrimination*

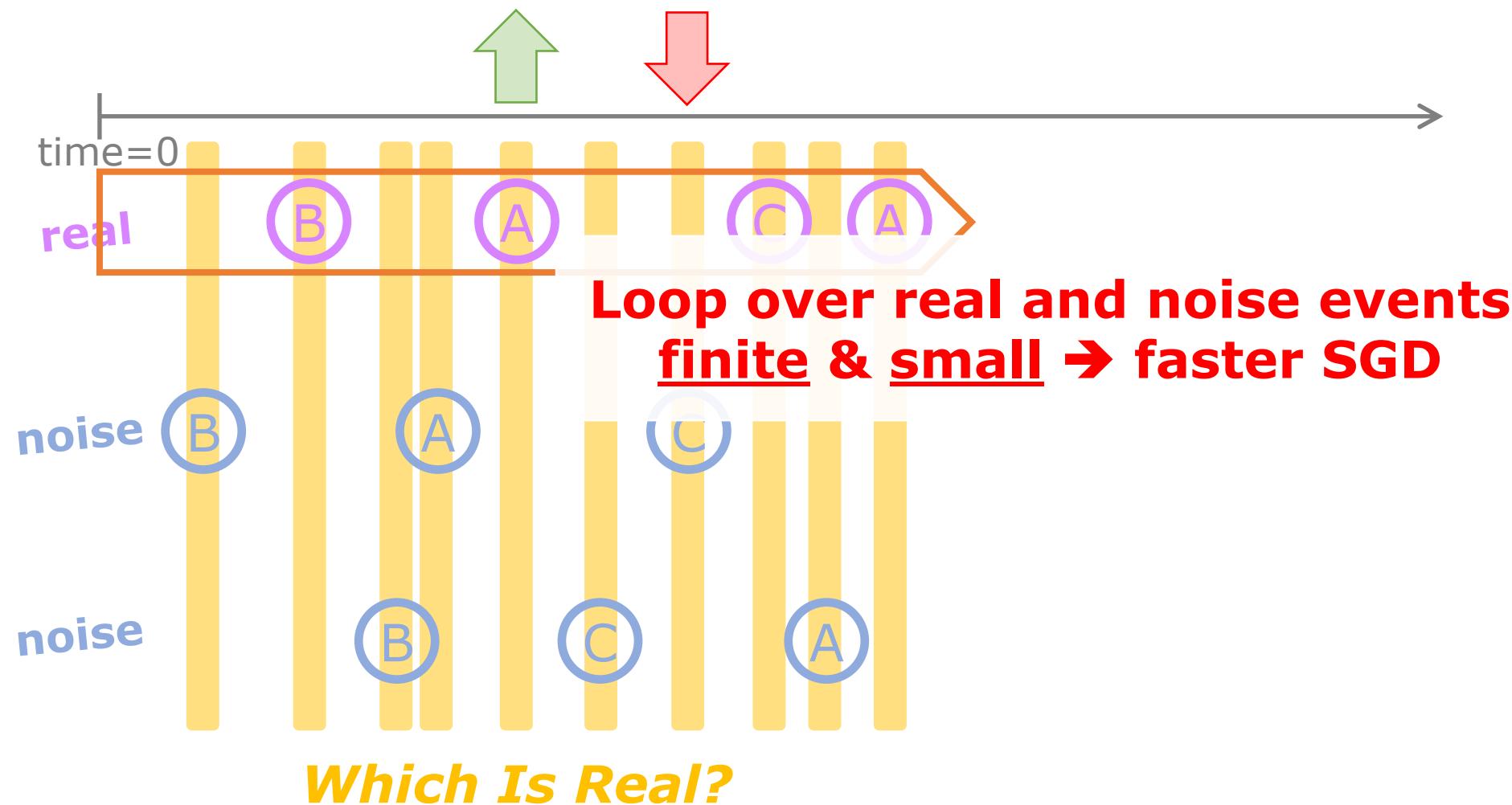


***Which Is Real?***

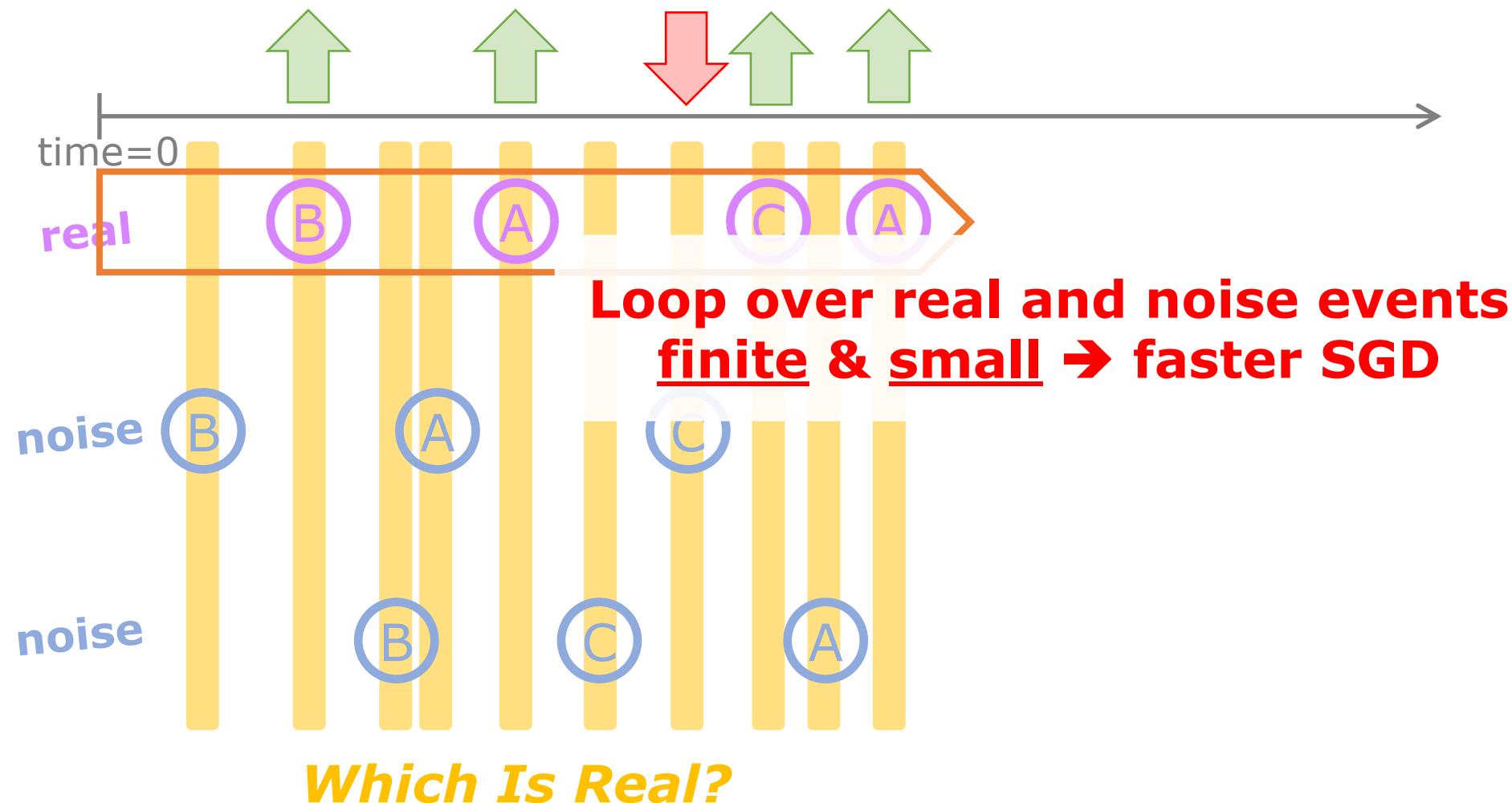
## NCE: Max log prob of *correct discrimination*



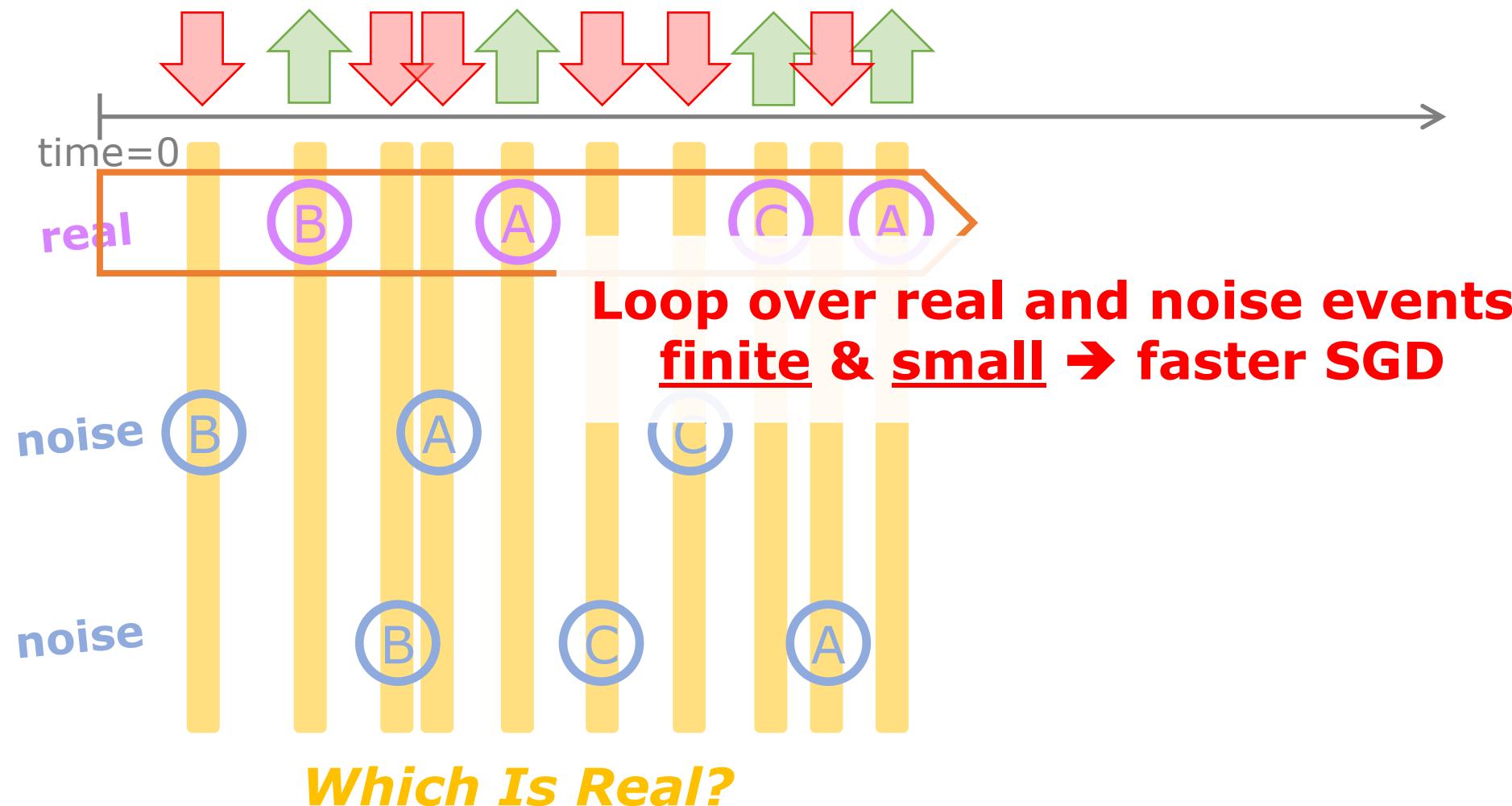
## NCE: Max log prob of *correct discrimination*



# NCE: Max log prob of *correct discrimination*



## NCE: Max log prob of *correct discrimination*



## Let's write some code

**<http://bburl/tpp-lab-p3>**

## NCE vs MLE: consistent & efficient & faster

# NCE vs MLE: consistent & efficient & faster

MLE

NCE

# NCE vs MLE: consistent & efficient & faster

MLE

NCE

consistent?

## NCE vs MLE: consistent & efficient & faster

	MLE	NCE
consistent?	yes	yes

## NCE vs MLE: consistent & efficient & faster

	MLE	NCE
consistent?	yes	yes
sample-efficient?		

## NCE vs MLE: consistent & efficient & faster

	MLE	NCE
consistent?	yes	yes
sample-efficient?	yes	(yes)

## NCE vs MLE: consistent & efficient & faster

	MLE	NCE
consistent?	yes	yes
sample-efficient?	yes	(yes)
sampling for ...		

## NCE vs MLE: consistent & efficient & faster

	MLE	NCE
consistent?	yes	yes
sample-efficient?	yes	(yes)
sampling for ...	integration	noise

## NCE vs MLE: consistent & efficient & faster

	MLE	NCE
consistent?	yes	yes
sample-efficient?	yes	(yes)
sampling for ...	integration	noise
fewer func evals?		

## NCE vs MLE: consistent & efficient & faster

	MLE	NCE
consistent?	yes	yes
sample-efficient?	yes	(yes)
sampling for ...	integration	noise
fewer func evals?		yes

## NCE vs MLE: consistent & efficient & faster

	MLE	NCE
consistent?	yes	yes
sample-efficient?	yes	(yes)
sampling for ...	integration	noise
fewer func evals?		yes
lower runtime?		

## NCE vs MLE: consistent & efficient & faster

	MLE	NCE
consistent?	yes	yes
sample-efficient?	yes	(yes)
sampling for ...	integration	noise
fewer func evals?		yes
lower runtime?		yes

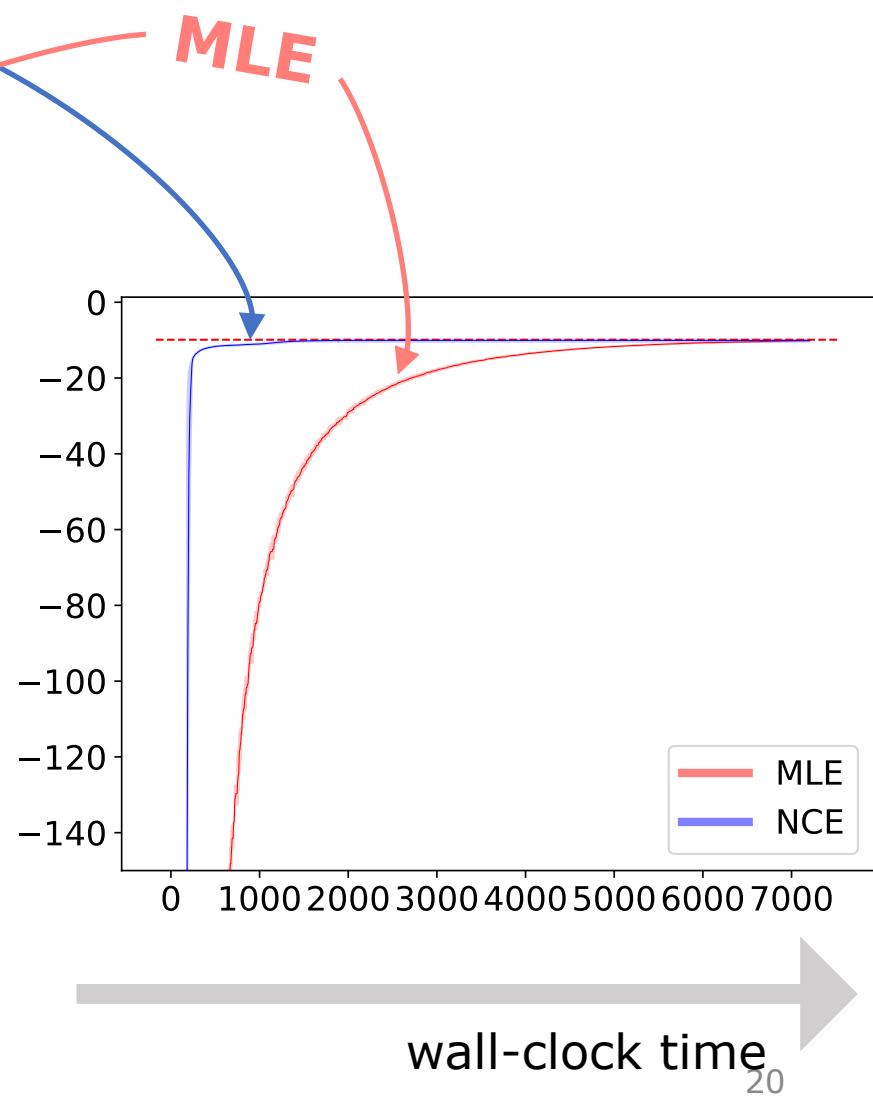
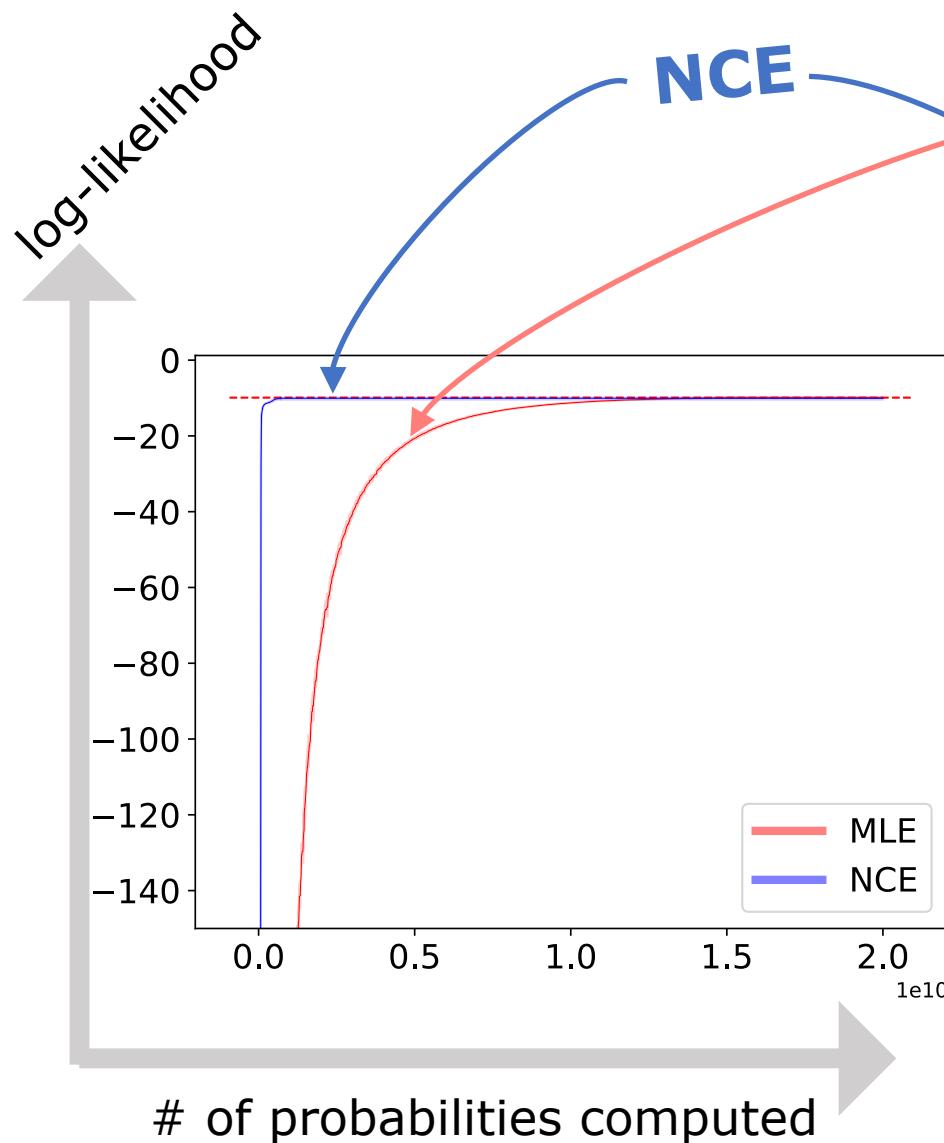
## NCE vs MLE: consistent & efficient & faster

	MLE	NCE
consistent?	yes	yes
sample-efficient?	yes	(yes)
sampling for ...	integration	noise
fewer func evals?		yes
lower runtime?		yes
better learning curve?		

## NCE vs MLE: consistent & efficient & faster

	MLE	NCE
consistent?	yes	yes
sample-efficient?	yes	(yes)
sampling for ...	integration	noise
fewer func evals?		yes
lower runtime?		yes
better learning curve?		yes

# NCE vs MLE: what it typically looks like



# NCE: More in Mei et al. NeurIPS 2020

# NCE: More in Mei et al. NeurIPS 2020

Theorem 1 (Optimality). Under assumptions 1 and 2,  $\theta \in \operatorname{argmax}_\theta J_{NC}(\theta)$  if and only if  $p_\theta = p^*$ .

We first need to highlight the key insight that  $H_\theta(k, t, x_{[0,t]}^0)$  in equation (20) is the negative cross-entropy between the following two discrete distributions over  $\{\emptyset, 1, \dots, K\}$ :

$$\left[ \frac{\lambda_k^*(t|x_{[0,t]}^0)}{\Delta_k^*(t|x_{[0,t]}^0)}, \frac{\lambda_k^q(t|x_{[0,t]}^0)}{\Delta_k^*(t|x_{[0,t]}^0)}, \dots, \frac{\lambda_k^q(t|x_{[0,t]}^0)}{\Delta_k^*(t|x_{[0,t]}^0)} \right] \quad (21a)$$

$$\underbrace{\left[ \frac{\lambda_k(t|x_{[0,t]}^0)}{\Delta_k(t|x_{[0,t]}^0)}, \frac{\lambda_k^q(t|x_{[0,t]}^0)}{\Delta_k(t|x_{[0,t]}^0)}, \dots, \frac{\lambda_k^q(t|x_{[0,t]}^0)}{\Delta_k(t|x_{[0,t]}^0)} \right]}_{\text{length } M} \quad (21b)$$

## theorems & proofs

Theorem 2 (Consistency). Under assumptions 1 and 4, for any  $\theta \in \operatorname{argmax}_\theta J_{NC}(\theta)$  and  $M \geq 1$ , with probability 1, we have  $J_{NC}(\hat{\theta}) \rightarrow J_{NC}(\theta)$  and  $J_{NC}(\hat{\theta})$  will become the same as  $N \rightarrow \infty$  and they are continuous with respect to  $\theta$ . The proof of Theorem 2 in Ma & Collins (2018). But we will still spell it out in our notation for completeness.

The intuition of this theorem is the same as N → ∞ and they are continuous with respect to θ. The proof of Theorem 2 in Ma & Collins (2018). But we will still spell it out in our notation for completeness.

Theorem 3 (Efficiency). Under assumptions 2 and 4–7, there exists an integer  $\bar{M}$  such that for all  $M > \bar{M}$  assumption in Theorem 2, by classical large sample theory (Ferguson, 1996), we have

$$\sqrt{N}(\hat{\theta} - \theta^*) \rightarrow \operatorname{Normal}(0, I_M^{-1}) \text{ as } N \rightarrow \infty \quad (39)$$

$$\|I_M^{-1} - I_*^{-1}\| \leq C/M$$

$$(40)$$

where  $\|I\|$  is the spectral norm of matrix  $I$ .

Proof. We first prove that  $\sqrt{N}(\hat{\theta} - \theta^*)$  is asymptotically normal. By the Mean-Value Theorem, we have

$$\nabla_{\theta} J_{NC}^N(\hat{\theta}) = \nabla_{\theta} J_{NC}^N(\theta^*) + (\hat{\theta} - \theta^*) \int_{u=0}^1 \nabla_{\theta}^2 J_{NC}^N(\theta^* + u(\hat{\theta} - \theta^*)) dt \quad (41)$$

# NCE: More in Mei et al. NeurIPS 2020

Theorem 1 (Optimality). Under assumptions 1 and 2,  $\theta \in \operatorname{argmax}_{\theta} J_{NC}(\theta)$  if and only if  $p_{\theta} = p^*$ .

We first need to highlight the key insight that  $H_{\theta}(k, t, x_{[0,t]}^0)$  in equation (3) is the negative cross-entropy between the following two discrete distributions over  $\{\emptyset, 1\}$ :

$$\left[ \frac{\lambda_k^*(t|x_{[0,t]}^0)}{\Delta_k^*(t|x_{[0,t]}^0)}, \frac{\lambda_k^q(t|x_{[0,t]}^0)}{\Delta_k^q(t|x_{[0,t]}^0)}, \dots, \frac{\lambda_k^q(t|x_t^0)}{\Delta_k^q(t|x_t^0)} \right] \text{ and } \left[ \underbrace{\frac{\lambda_k(t|x_{[0,t]}^0)}{\Delta_k(t|x_{[0,t]}^0)}, \frac{\lambda_k^q(t|x_{[0,t]}^0)}{\Delta_k^q(t|x_{[0,t]}^0)}, \dots, \frac{\lambda_k^q(t|x_t^0)}{\Delta_k^q(t|x_t^0)}}_{\text{length } K} \right]$$

## theorems & proofs

**Theorem 2 (Consistency).** Under assumption 1 and 4, for any  $\theta \in \operatorname{argmax}_{\theta} J_{NC}(\theta)$  and  $M \geq 1$ , with probability  $1 - o(1)$  we have  $J_{NC}(\theta) \rightarrow J_{NC}(\theta^*)$  and  $J_{NC}(\theta)$  will become the same as  $N \rightarrow \infty$  and they are continuous with probability  $1 - o(1)$ . This implies  $J_{NC}(\theta)$  is almost surely a member of the set  $\operatorname{argmax}_{\theta} J_{NC}(\theta)$ . The intuition of this theorem is the same as N → ∞ and they are continuous with probability 1 - o(1). This implies JNC(θ) is almost surely a member of the set argmaxθ JNC(θ).

**Theorem 3 (Efficiency).** Under assumptions 2 and 4–7, there exists an integer  $\bar{M}$  such that for all  $M > \bar{M}$   $\sqrt{N}(\hat{\theta} - \theta^*) \rightarrow \text{Normal}(0, I_M^{-1})$  as  $N \rightarrow \infty$ . Moreover, there exist a constant  $C > 0$  such that for all  $M > \bar{M}$   $\|I_M^{-1} - I_*^{-1}\| \leq C/M$  where  $\|\cdot\|$  is the spectral norm of matrix  $I$ .

**Proof.** We first prove that  $\sqrt{N}(\hat{\theta} - \theta^*)$  is asymptotically normal. By the Mean-Value Theorem, we have

$$\nabla_{\theta} J_{NC}^N(\hat{\theta}) = \nabla_{\theta} J_{NC}^N(\theta^*) + (\hat{\theta} - \theta^*) \int_{u=0}^1 \nabla_{\theta}^2 J_{NC}^N(\theta^* + u(\hat{\theta} - \theta^*)) dt \quad (40)$$

$$\nabla_{\theta} J_{NC}^N(\hat{\theta}) = \nabla_{\theta} J_{NC}^N(\theta^*) + (\hat{\theta} - \theta^*) \int_{u=0}^1 \nabla_{\theta}^2 J_{NC}^N(\theta^* + u(\hat{\theta} - \theta^*)) dt \quad (41)$$

*how to draw noise fast*

**3.1 Efficient Sampling of Noise Events**

The thinning algorithm (Lewis & Shedler, 1979; Liniger, 2009) is a rejection sampling method for drawing an event stream over a given observation interval  $[0, T]$  from a continuous-time autoregressive process. Suppose we have already drawn the first  $i-1$  times, namely  $t_1, \dots, t_{i-1}$ . For every future time  $t \geq t_{i-1}$ , let  $\mathcal{H}(t)$  denote the context  $x_{[0,t]}$  consisting only of the events at those times, and define  $\lambda(t | \mathcal{H}(t)) \stackrel{\text{def}}{=} \sum_{k=1}^K \lambda_k(t | \mathcal{H}(t))$ . If  $\lambda(t | \mathcal{H}(t))$  were constant at  $\bar{\lambda}$ , we could draw the next event time as  $t_i \sim t_{i-1} + \text{Exp}(\bar{\lambda})$ . But what if  $\lambda(t | \mathcal{H}(t))$  is not constant? The thinning algorithm still runs the foregoing method, taking  $\bar{\lambda}$  to be any upper bound:  $\bar{\lambda} \geq \lambda(t | \mathcal{H}(t))$  for all  $t \geq t_{i-1}$ . In this case, there may be “leftover” probability mass not allocated to any  $k$ . This mass is allocated to  $\emptyset$ . A draw of  $x_{t_i} = \emptyset$  means there was no event at time  $t_i$  after all (corresponding to a rejected proposal). Either way, we now continue on to draw  $t_{i+1}$  and  $x_{t_{i+1}}$  using a version of  $\mathcal{H}(t)$  that has been updated to include the event or non-event  $x_{t_i}$ . The update to  $\mathcal{H}(t)$  affects  $\lambda(t | \mathcal{H}(t))$  and the choice of  $\bar{\lambda}$ . How to sample noise streams. To draw a stream  $x_{[0,t]}^0$  of noise events, we run the thinning algorithm, using the noise intensity functions  $\lambda_k^q$ . However, there is a modification:  $\mathcal{H}(t)$  is now defined to be  $x_{[0,t]}^0$ —the history from the observed event stream, rather than the previously sampled noise events—and is updated accordingly. This is because in equation (6), at each time  $t$ , all of  $\{x_t^0, x_t^1, \dots, x_t^M\}$  are conditioned on  $x_{[0,t]}^0$  (akin to the discrete-time case).<sup>7</sup> The full pseudocode is given in Algorithm 1 in the supplementary material.

Coarse-to-fine sampling of event types. Although our NCE method has eliminated the need to integrate over  $t$ , the thinning algorithm above still sums over  $k$  in the definition of  $\Lambda^q(t | \mathcal{H}(t))$ . For large  $K$ , this sum is expensive if we take the noise distribution on each training minibatch to

# NCE: More in Mei et al. NeurIPS 2020

## theorems & proofs

Theorem 1 (Optimality). Under assumptions 1 and 2,  $\theta \in \operatorname{argmax}_{\theta} J_{NC}(\theta)$  if and only if  $p_{\theta} = p^*$ .

We first need to highlight the key insight that  $H_{\theta}(k, t, x_{[0,t]}^0)$  in equation (30) is the negative cross-entropy between the following two discrete distributions over  $\{\emptyset, 1\}$ :

$$\left[ \frac{\lambda_k^*(t|x_{[0,t]}^0)}{\Delta_k^*(t|x_{[0,t]}^0)}, \frac{\lambda_k^q(t|x_{[0,t]}^0)}{\Delta_k^q(t|x_{[0,t]}^0)}, \dots, \frac{\lambda_k^q(t|x_t^0)}{\Delta_k^q(t|x_t^0)} \right]$$

where  $\|\cdot\|$  is the  $L_2$ -norm.

The intuition of this theorem is that  $J_{NC}(\theta)$  will become the same as  $N \rightarrow \infty$  and they are continuous. Theorem 2 (Consistency). Under assumption 1 and 4, for any  $\theta \in \operatorname{argmax}_{\theta} J_{NC}(\theta)$ , we will still have  $\hat{\theta} \rightarrow \theta$  almost surely. This is identical to the proof for complete.

Theorem 3 (Efficiency). Under assumptions 2 and 4–7, there exists an integer  $\bar{M}$  such that for all  $M > \bar{M}$

$$\sqrt{N}(\hat{\theta} - \theta^*) \rightarrow \text{Normal}(0, I_M^{-1}) \text{ as } N \rightarrow \infty$$

for some non-singular matrix  $I_M^{-1}$ . Moreover, there exist a constant  $C > 0$  such that for all  $M > \bar{M}$ ,

$$\|I_M^{-1} - I_*^{-1}\| \leq C/M$$

where  $\|\cdot\|$  is the spectral norm of matrix  $I$ .

Proof. We first prove that  $\sqrt{N}(\hat{\theta} - \theta^*)$  is asymptotically normal. By the Mean-Value Theorem, we have

$$\nabla_{\theta} J_{NC}^N(\hat{\theta}) = \nabla_{\theta} J_{NC}^N(\theta^*) + (\hat{\theta} - \theta^*) \int_{u=0}^1 \nabla_{\theta}^2 J_{NC}^N(\theta^* + u(\hat{\theta} - \theta^*)) du \quad (41)$$

how to draw noise  $t_c$

### 3.1 Efficient Sampling of Noise Events

The thinning algorithm (Lewis & Shedler, 1979) has to draw an event stream over a given observation process. Suppose we have already drawn the future time  $t \geq t_{i-1}$ , let  $\mathcal{H}(t)$  denote the context, and define  $\lambda(t | \mathcal{H}(t)) \stackrel{\text{def}}{=} \sum_{k=1}^K \lambda_k(t | \mathcal{H}(t))$ . If the next event time as  $t_i \sim t_{i-1}$  and  $x_{t_i}^0$  of the

foregoing method, taking  $\bar{\lambda}$  to be any upper bound to any  $k$ . This mass is allocated to  $\mathcal{O}$ . El-

severely, we now continue on to draw  $t_{i+1}$  and  $x_{t_{i+1}}^0$  using a version of  $\mathcal{H}(t)$  that has been updated

to include the event or non-event  $x_{t_i}^0$ . The update to  $\mathcal{H}(t)$  affects  $\lambda(t | \mathcal{H}(t))$  and the choice of  $\bar{\lambda}$ .

How to sample noise streams. To draw a stream  $x_{[0,t]}^0$  of noise events, we run the thinning al-

gorithm, using the noise intensity functions  $\lambda_k^q$ . However, there is a modification:  $\mathcal{H}(t)$  is now

defined to be  $x_{[0,t]}^0$ —the history from the observed event stream, rather than the previously sampled

noise events—and is updated accordingly. This is because in the discrete-time case,<sup>7</sup> the full pseudocode is

given in Algorithm 1 in the supplementary material.

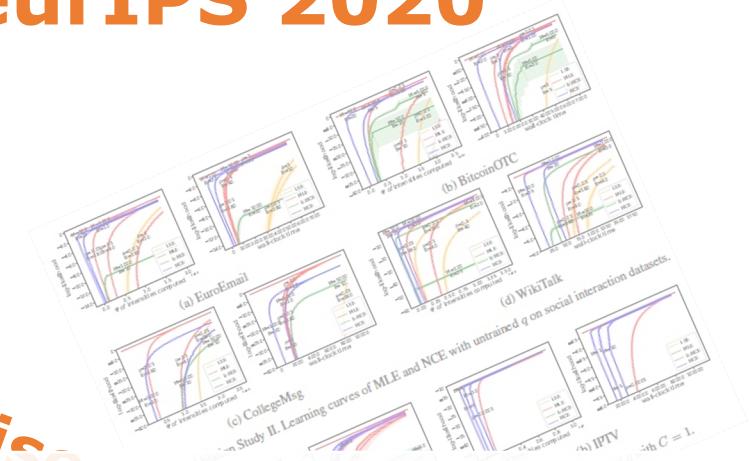
For large  $K$ , this sum is expensive if we take the noise distribution on each training minibatch to

integrate over  $t$ , the thinning algorithm above still sums over  $k$  in the definition of  $\lambda_k^q(t | \mathcal{H}(t))$ .

Coarse-to-fine sampling of event types. Although our NCE method has eliminated the need to

integrate over  $t$ , the thinning algorithm above still sums over  $k$  in the definition of  $\lambda_k^q(t | \mathcal{H}(t))$ .

For large  $K$ , this sum is expensive if we take the noise distribution on each training minibatch to



## more results & analysis

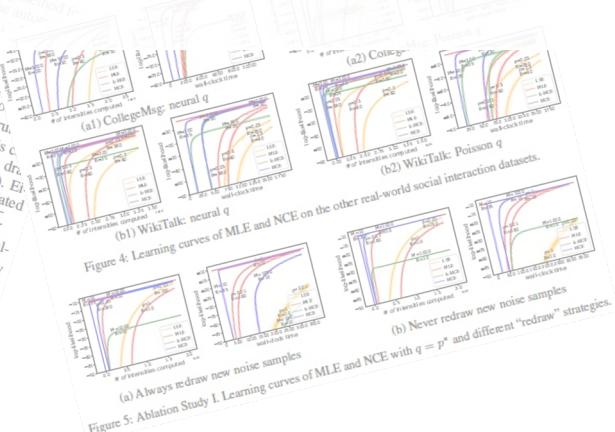


Figure 4: Learning curves of MLE and NCE on the other real-world social interaction datasets.  
 (a1) CollegeMsg: neural q  
 (a2) WikiTalk: Poisson q  
 (b1) WikiTalk: neural q  
 (b2) Never redraw new noise samples  
 (a) Always redraw new noise samples  
 (b) Never redraw new noise samples

Figure 5: Ablation Study I. Learning curves of MLE and NCE with  $q = p^*$  and different “redraw” strategies.

<http://bburl/tpp-slides-p3>  
<http://bburl/tpp-lab-p3>

# More Topics

# More Topics

## Domain Knowledge

KownEvolve: Trivedi et al. ICML 2017

DyRep: Trivedi et al. ICLR 2019

# More Topics

## Domain Knowledge

KownEvolve: Trivedi et al. ICML 2017

DyRep: Trivedi et al. ICLR 2019

## Continuous-Time Reinforcement Learning

Policy-based: Upadhyay et al. NeurIPS 2018

# More Topics

## Domain Knowledge

KownEvolve: Trivedi et al. ICML 2017

DyRep: Trivedi et al. ICLR 2019

## Continuous-Time Reinforcement Learning

Policy-based: Upadhyay et al. NeurIPS 2018

## Granger Causality over event types?

CAUSE: attribution methods, Zhang et al. ICML 2020

# More Topics

## Domain Knowledge

KownEvolve: Trivedi et al. ICML 2017

DyRep: Trivedi et al. ICLR 2019

## Continuous-Time Reinforcement Learning

Policy-based: Upadhyay et al. NeurIPS 2018

## Granger Causality over event types?

CAUSE: attribution methods, Zhang et al. ICML 2020

## Variational Autoencoder (VAE)

NHP as decoder: Boyd et al. NeurIPS 2020

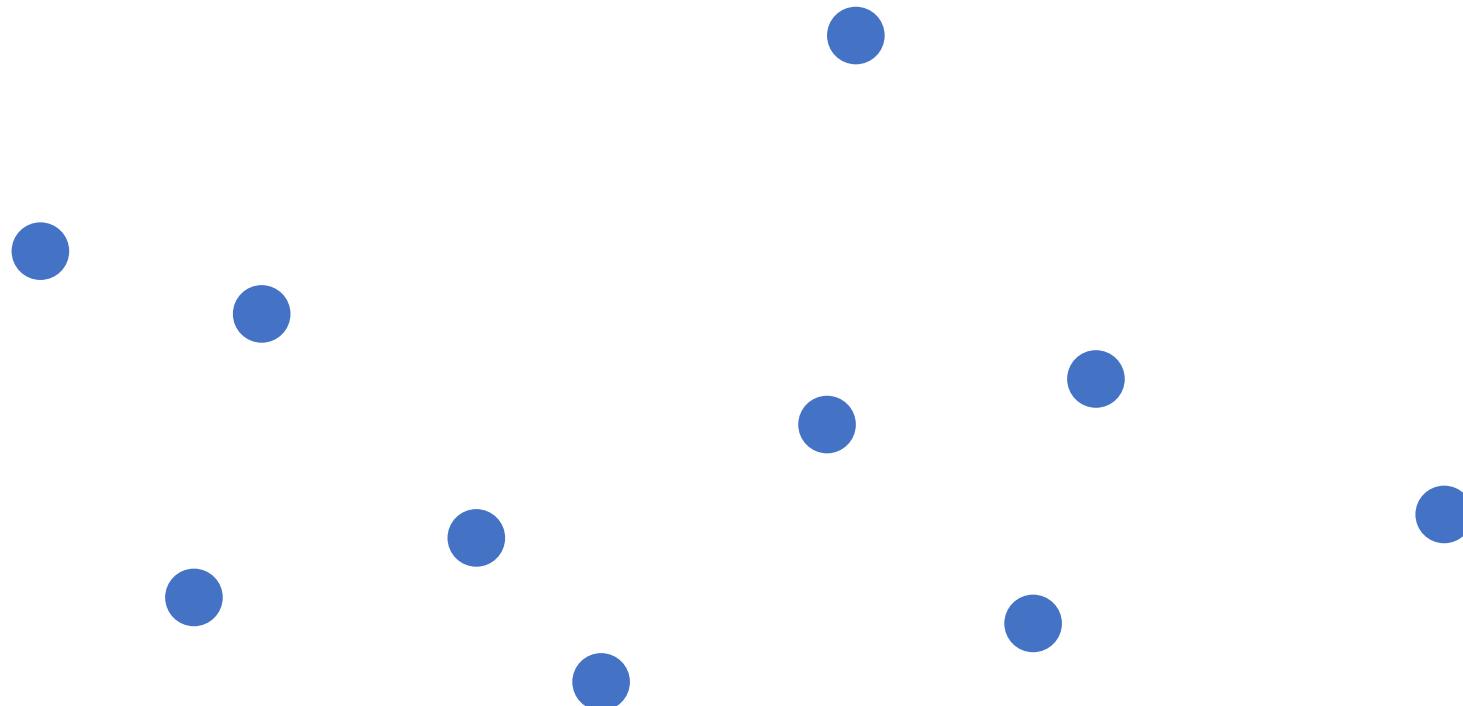
# **Neural Datalog Through Time**

Hongyuan Mei<sup>1</sup>, Guanghui Qin<sup>1</sup>, Minjie Xu<sup>2</sup>, Jason Eisner<sup>1</sup>

<sup>1</sup>Johns Hopkins University

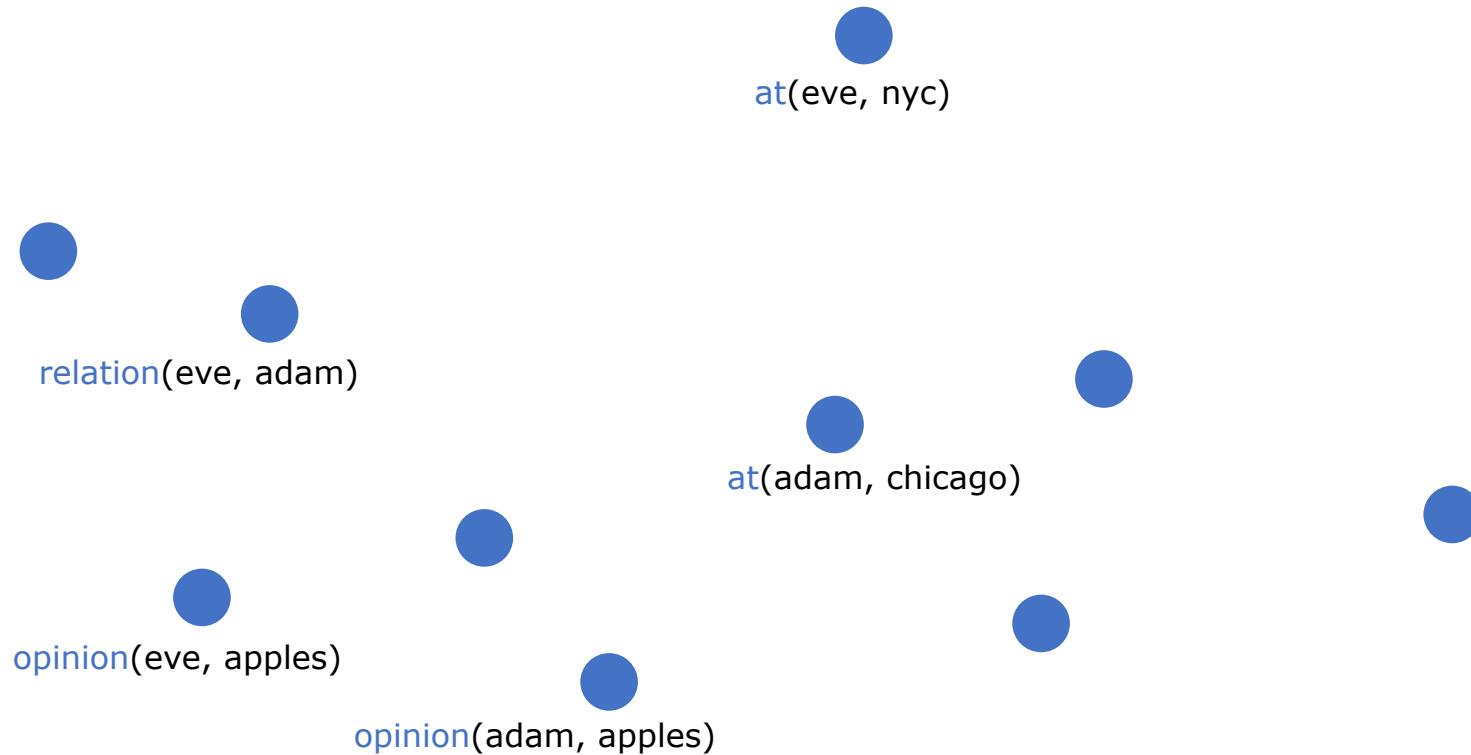
<sup>2</sup>Bloomberg

# Model How a Database Changes Over Time



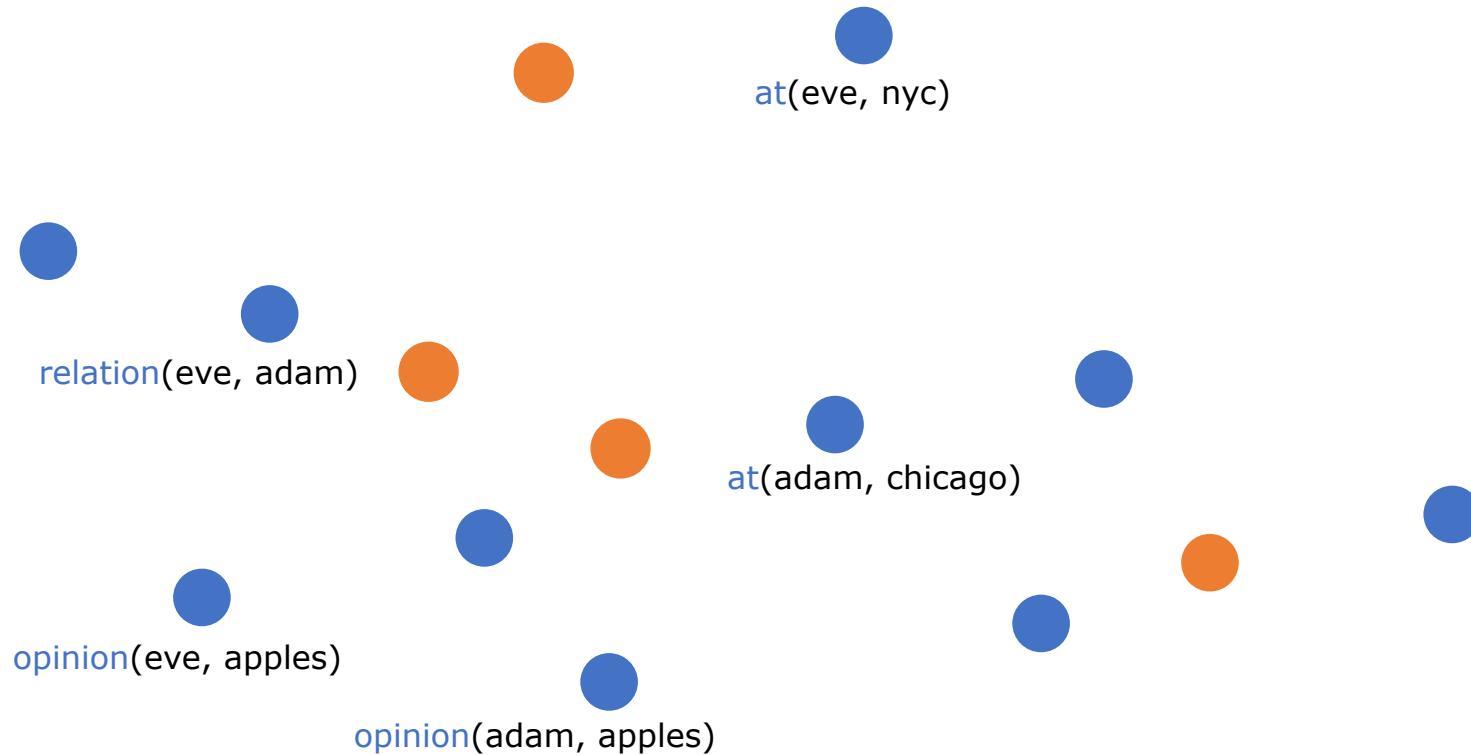
# Model How a Database Changes Over Time

200,000 facts right now



# Model How a Database Changes Over Time

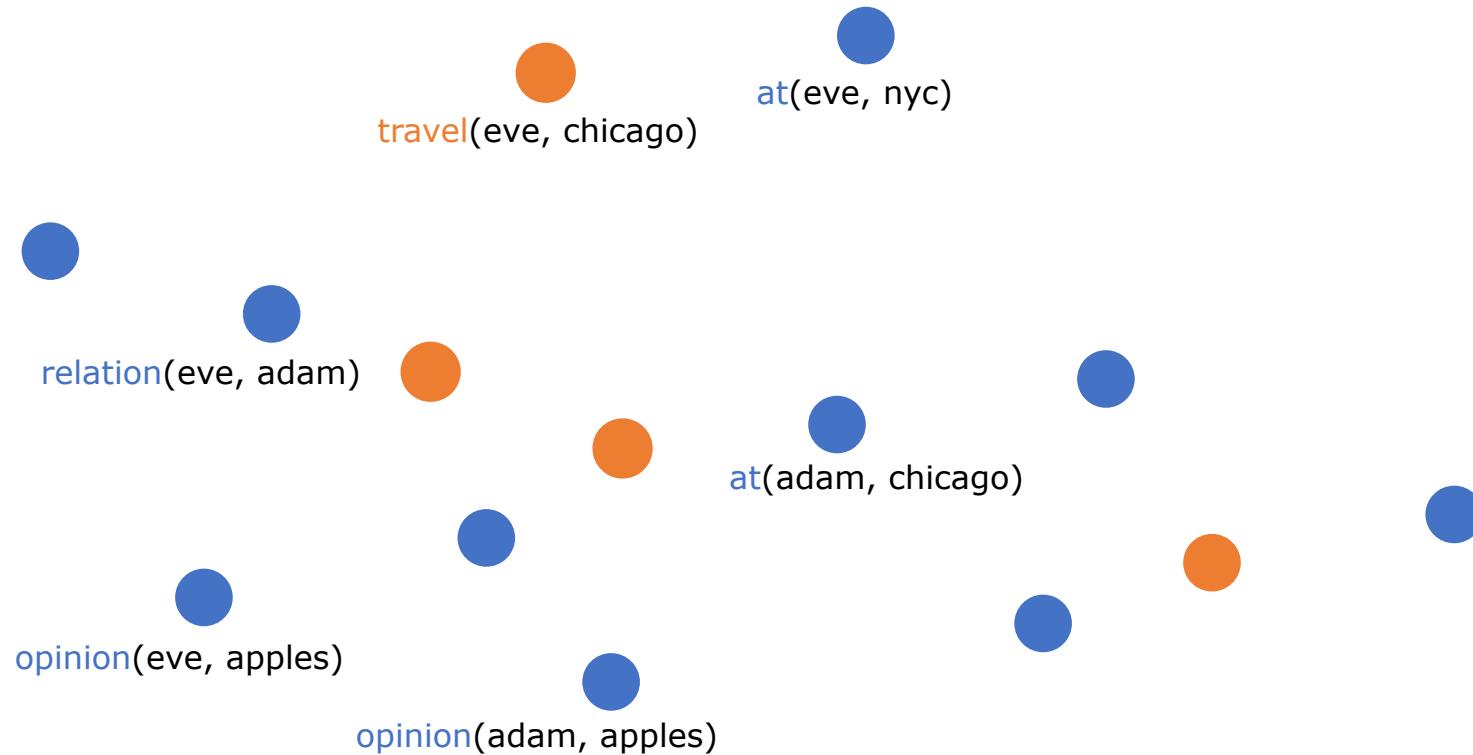
200,000 facts right now



# Model How a Database Changes Over Time

200,000 facts right now

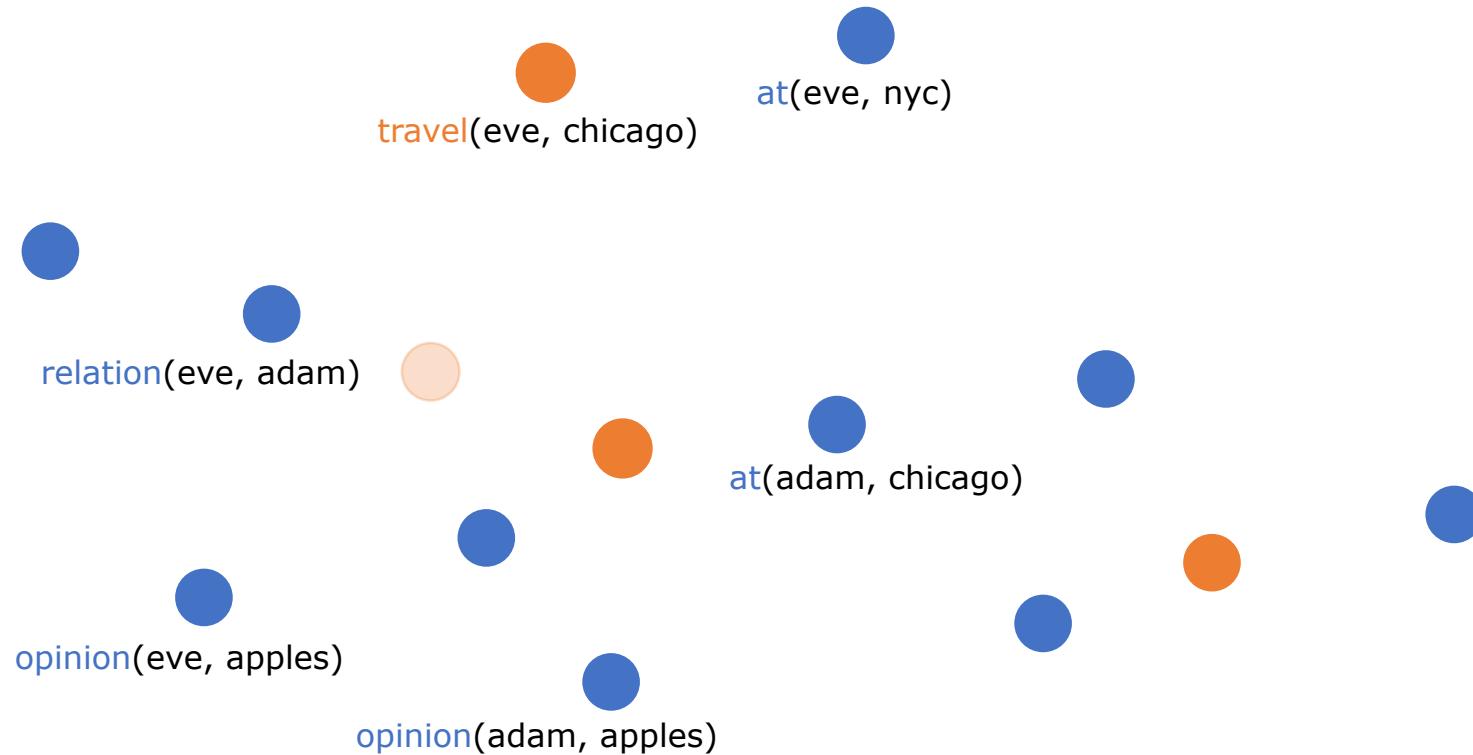
50,000 possible events right now



# Model How a Database Changes Over Time

200,000 facts right now

50,000 possible events right now



# Model How a Database Changes Over Time

200,000 facts right now

50,000 possible events right now

*little language  
to specify a generative model  
of event sequences*



# Model How a Database Changes Over Time

200,000 facts right now

50,000 possible events right now

*little language  
to specify a generative model  
of event sequences*

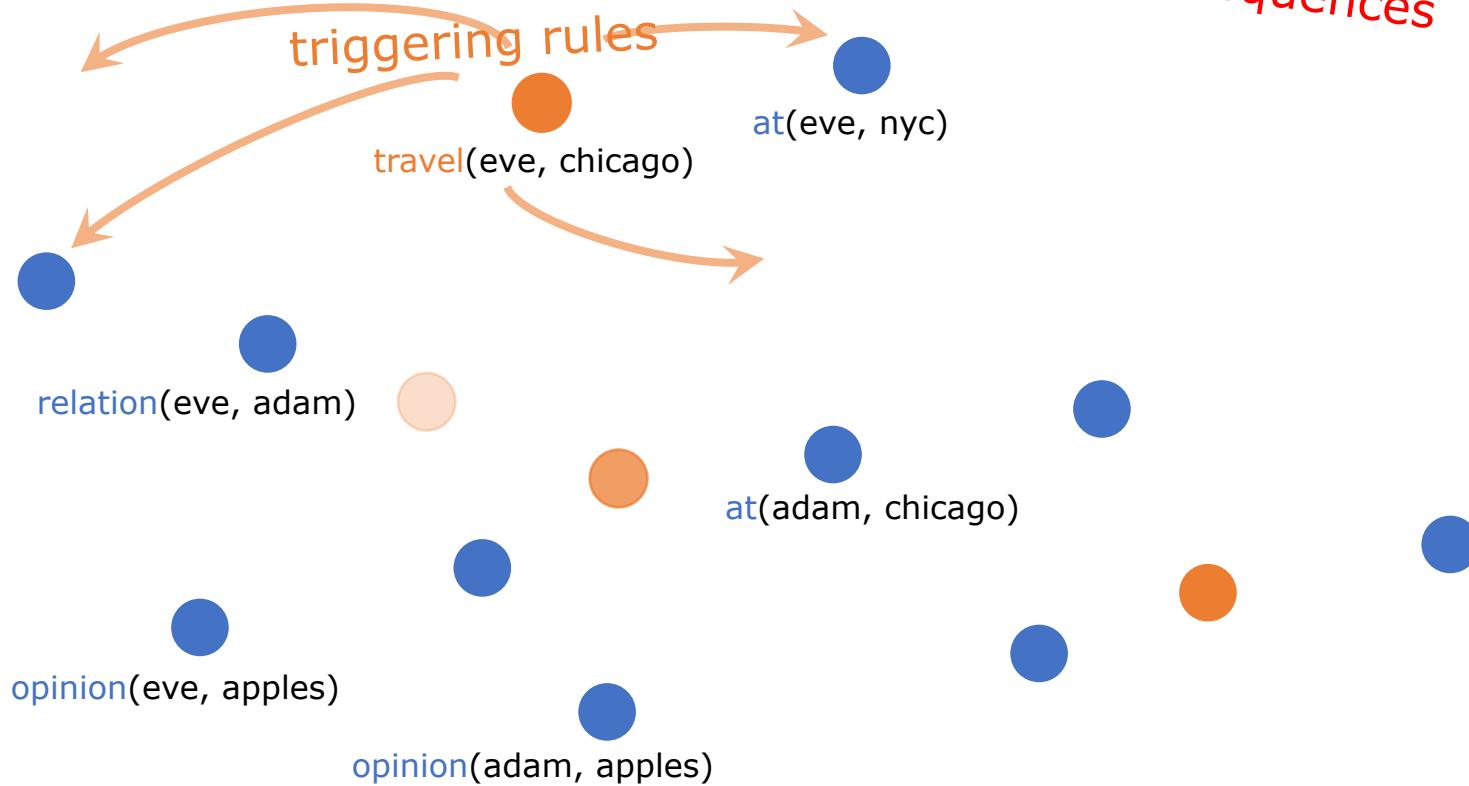


# Model How a Database Changes Over Time

200,000 facts right now

50,000 possible events right now

*little language  
to specify a generative model  
of event sequences*

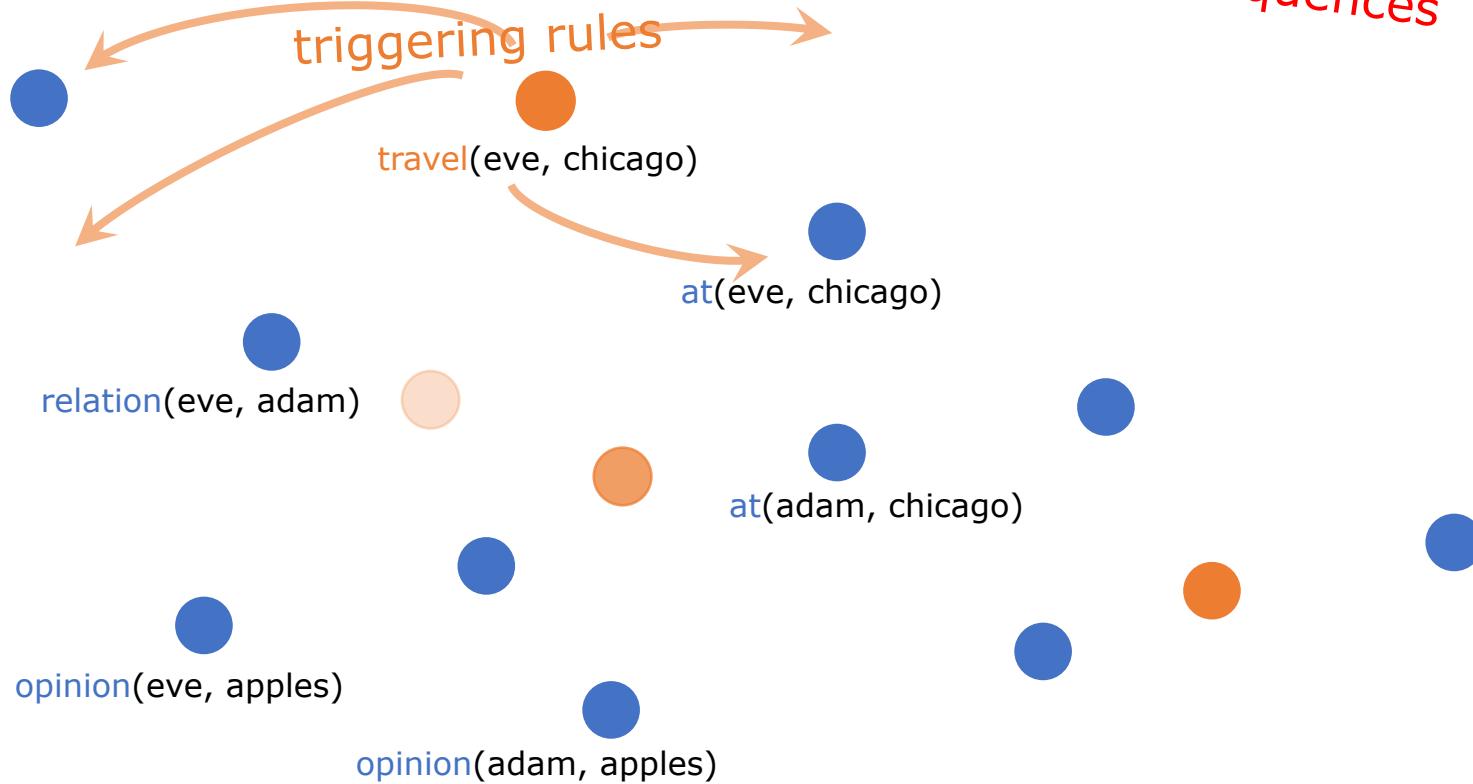


# Model How a Database Changes Over Time

200,000 facts right now

50,000 possible events right now

*little language  
to specify a generative model  
of event sequences*

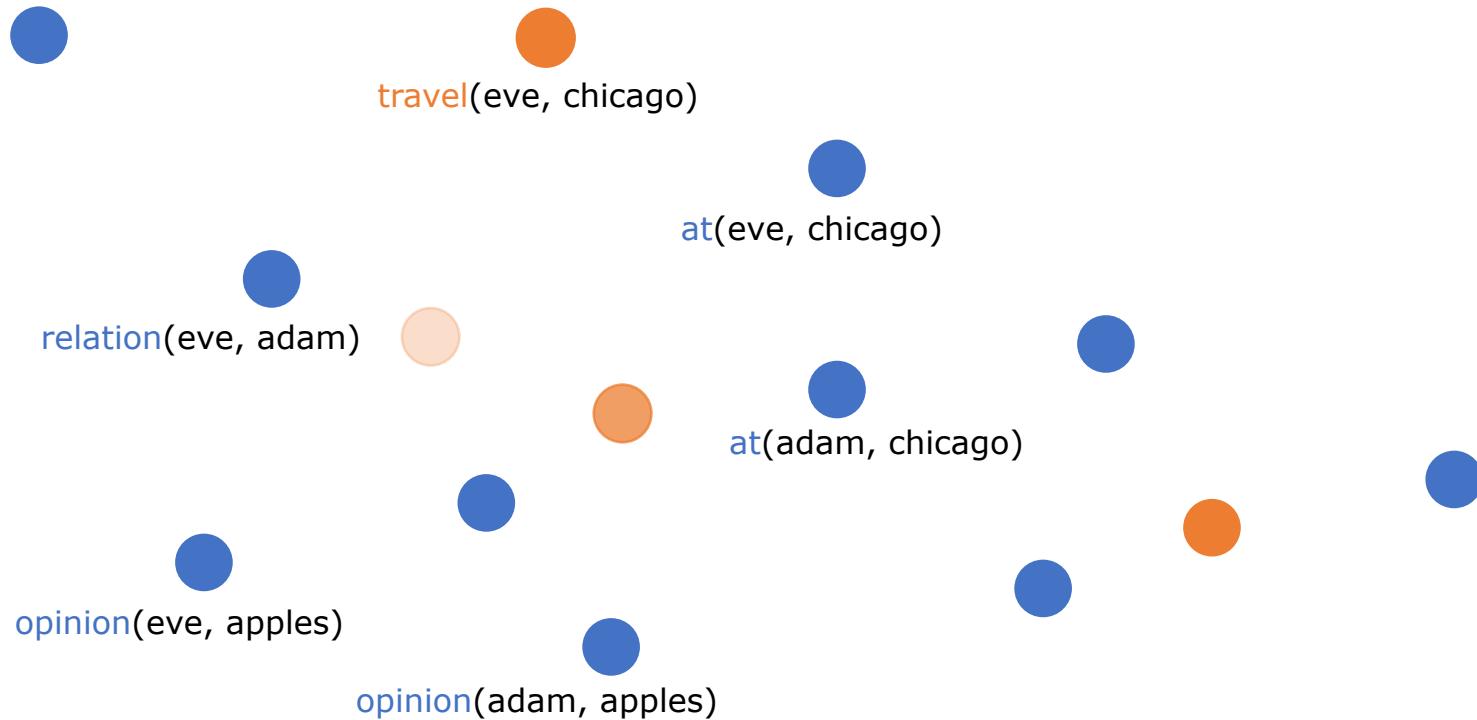


# Model How a Database Changes Over Time

200,000 facts right now

50,000 possible events right now

*little language  
to specify a generative model  
of event sequences*

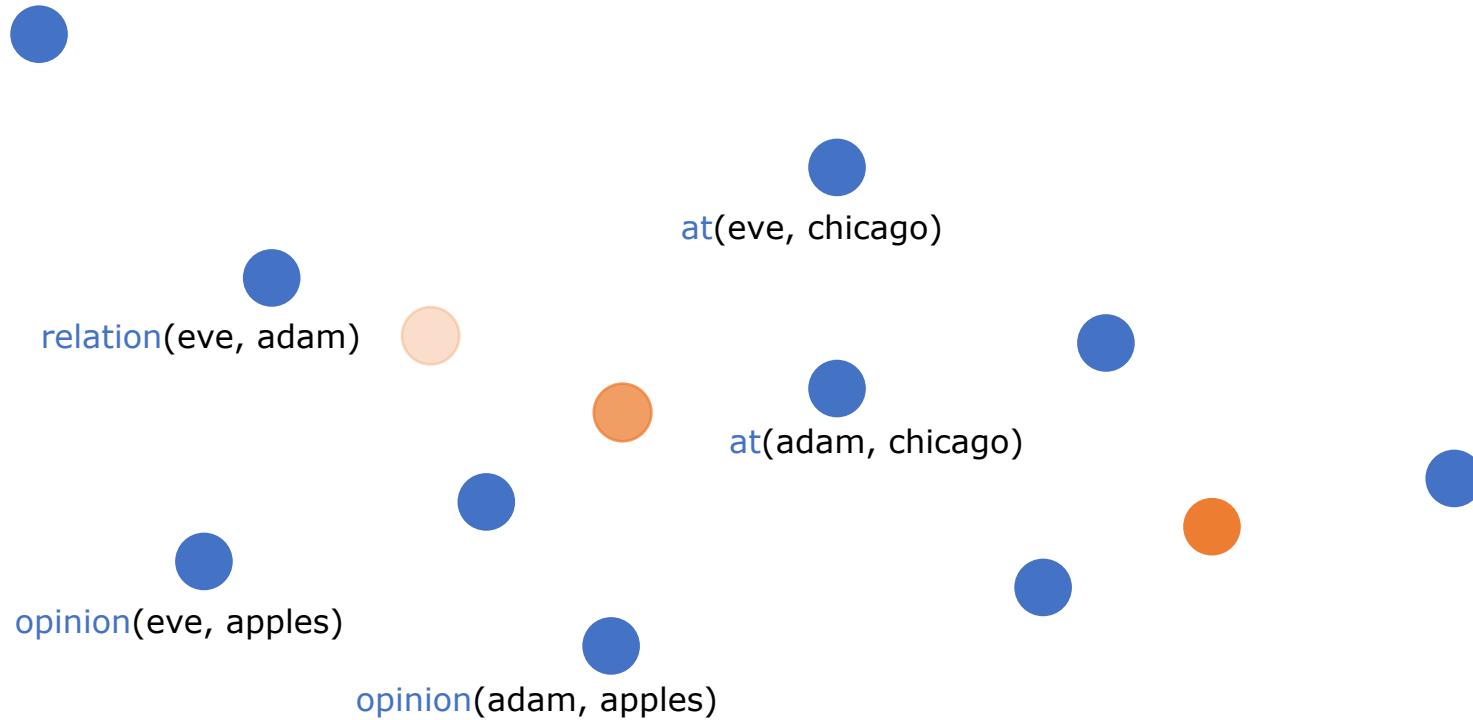


# Model How a Database Changes Over Time

200,000 facts right now

50,000 possible events right now

*little language  
to specify a generative model  
of event sequences*

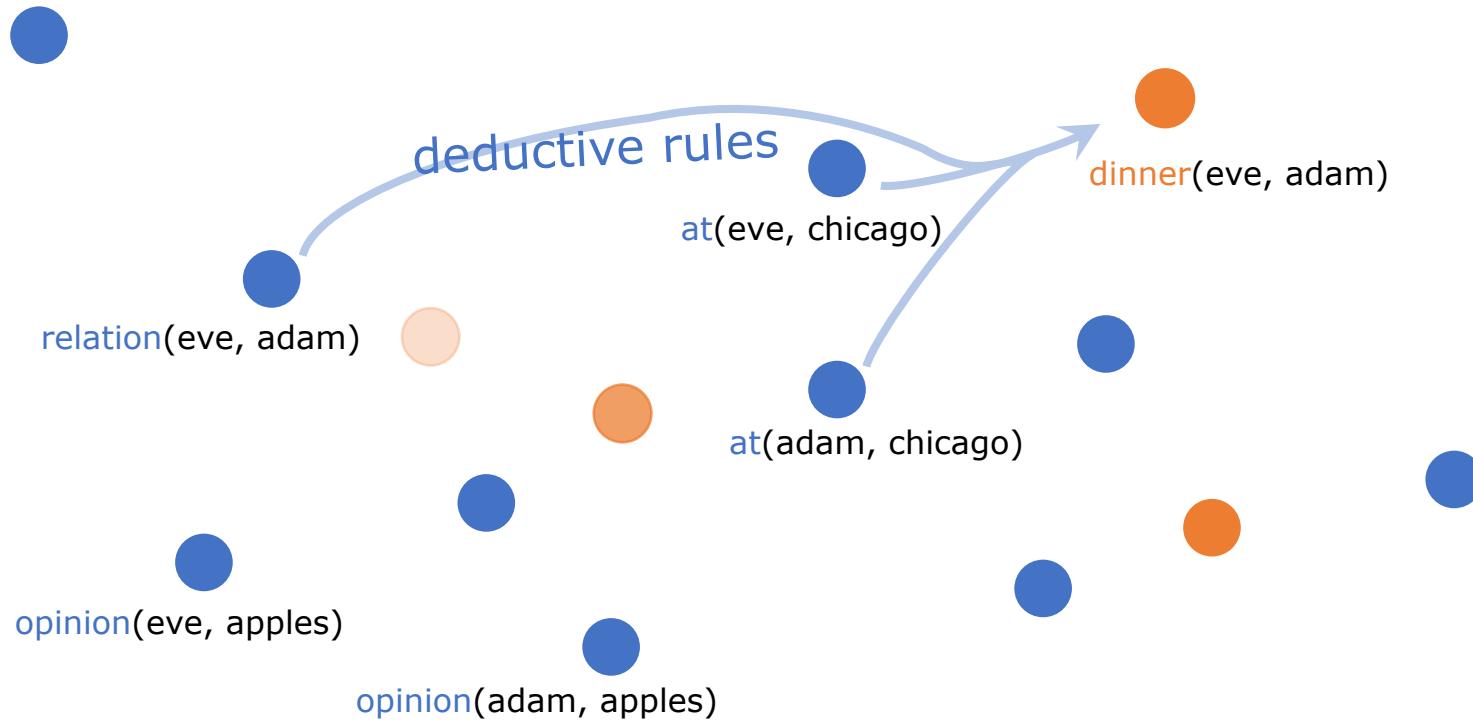


# Model How a Database Changes Over Time

200,000 facts right now

50,000 possible events right now

*little language  
to specify a generative model  
of event sequences*

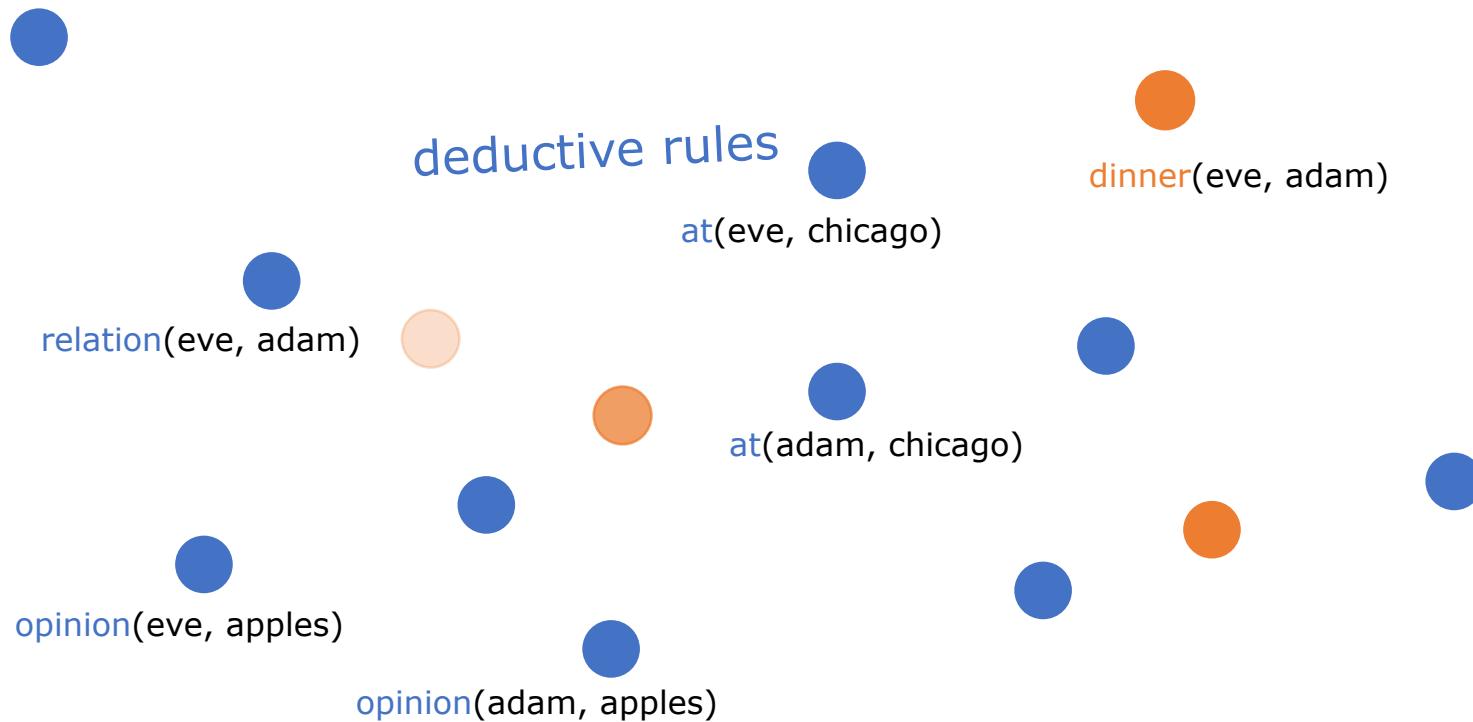


# Model How a Database Changes Over Time

200,000 facts right now

50,000 possible events right now

*little language  
to specify a generative model  
of event sequences*

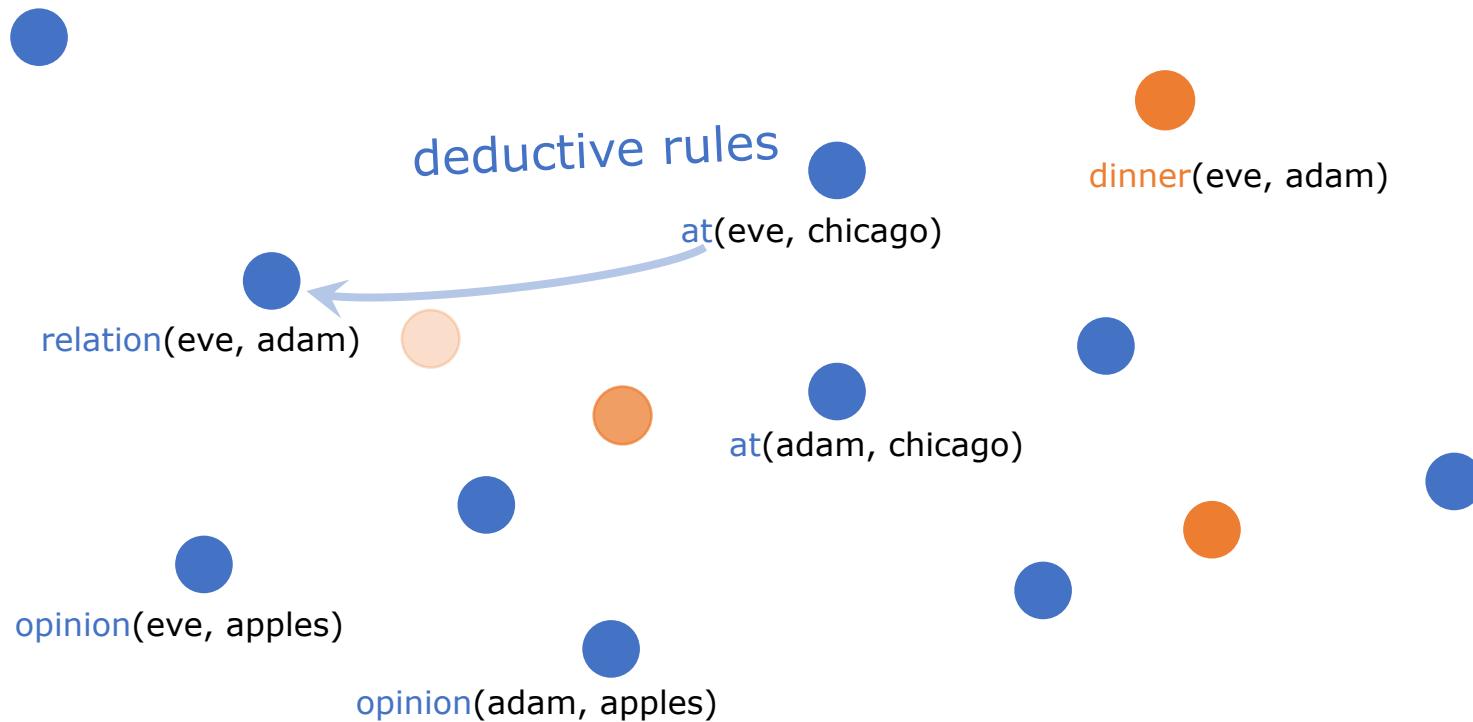


# Model How a Database Changes Over Time

200,000 facts right now

50,000 possible events right now

*little language  
to specify a generative model  
of event sequences*

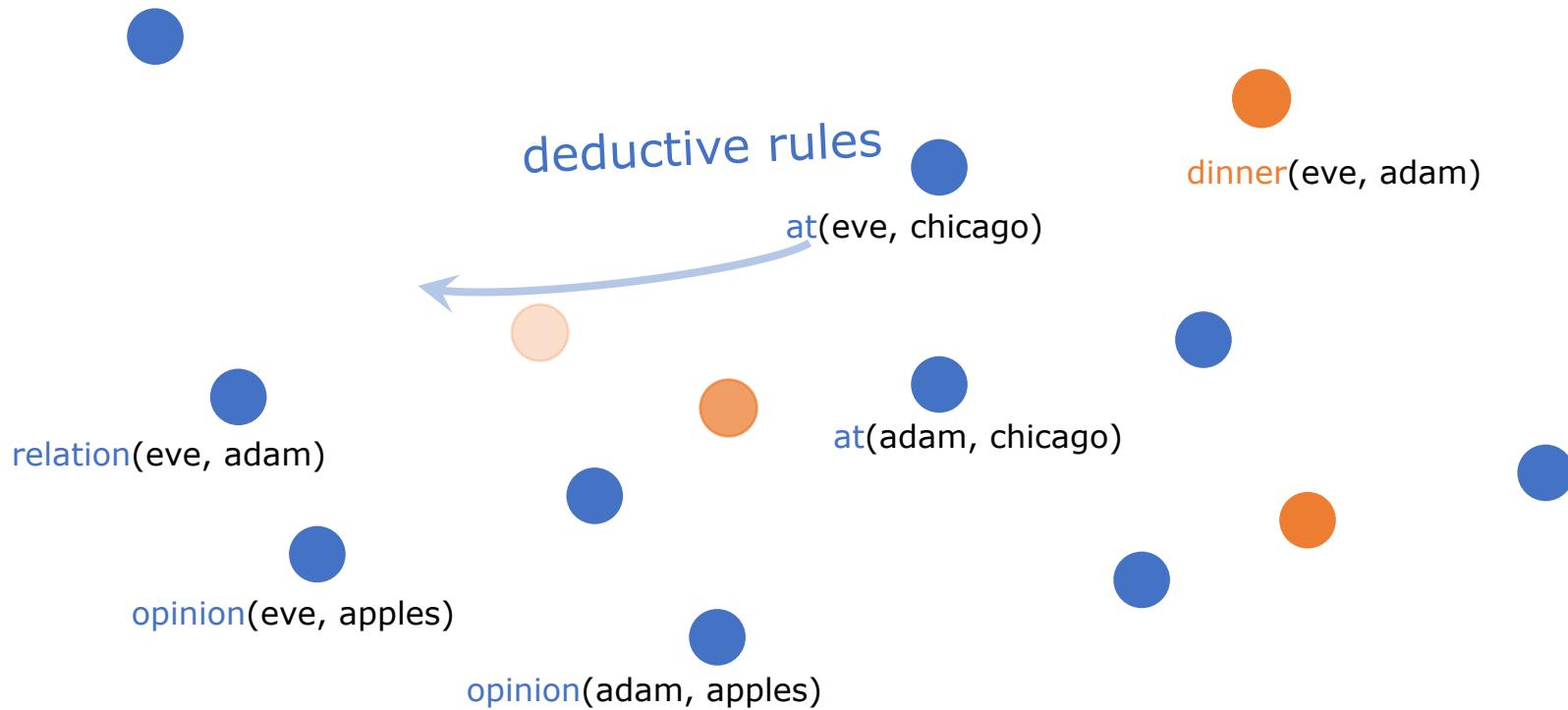


# Model How a Database Changes Over Time

200,000 facts right now

50,000 possible events right now

*little language  
to specify a generative model  
of event sequences*

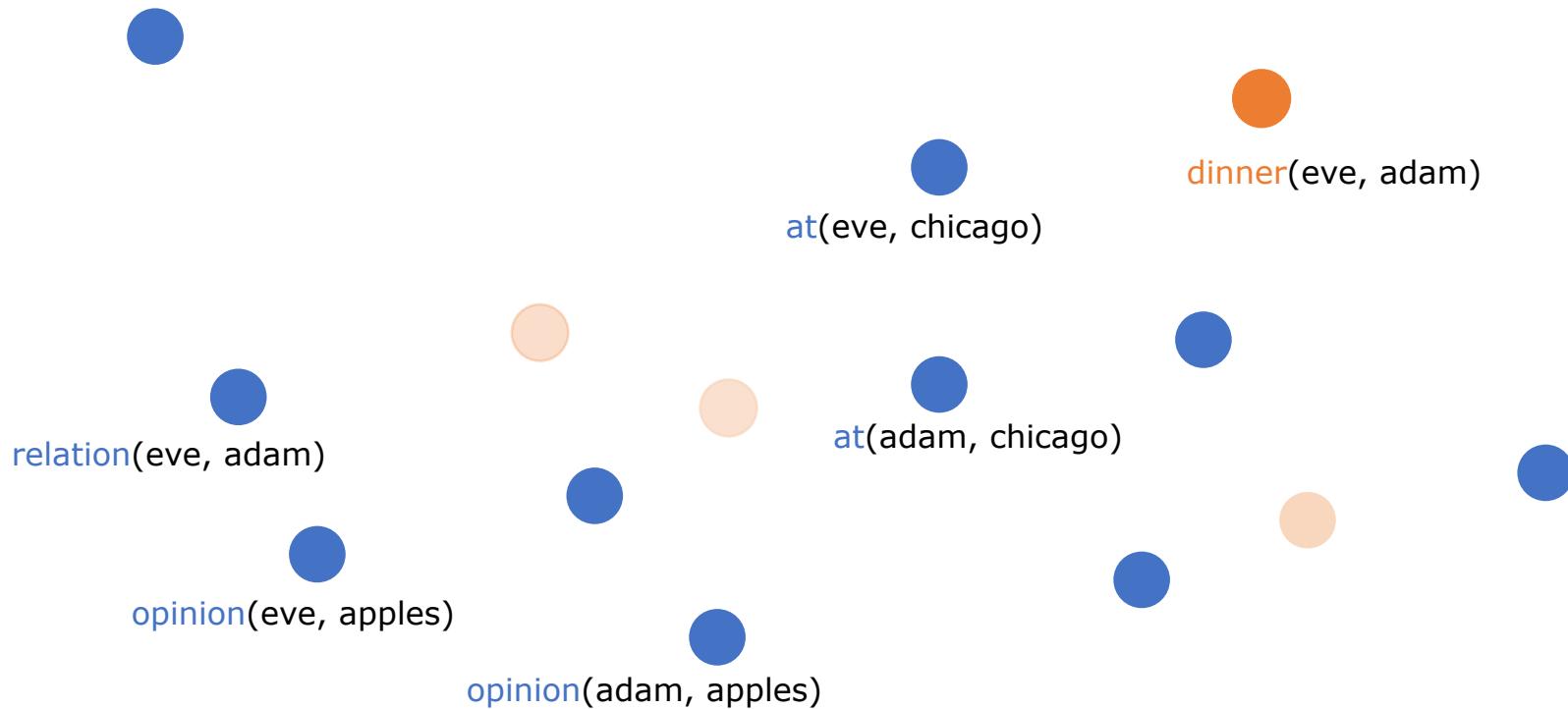


# Model How a Database Changes Over Time

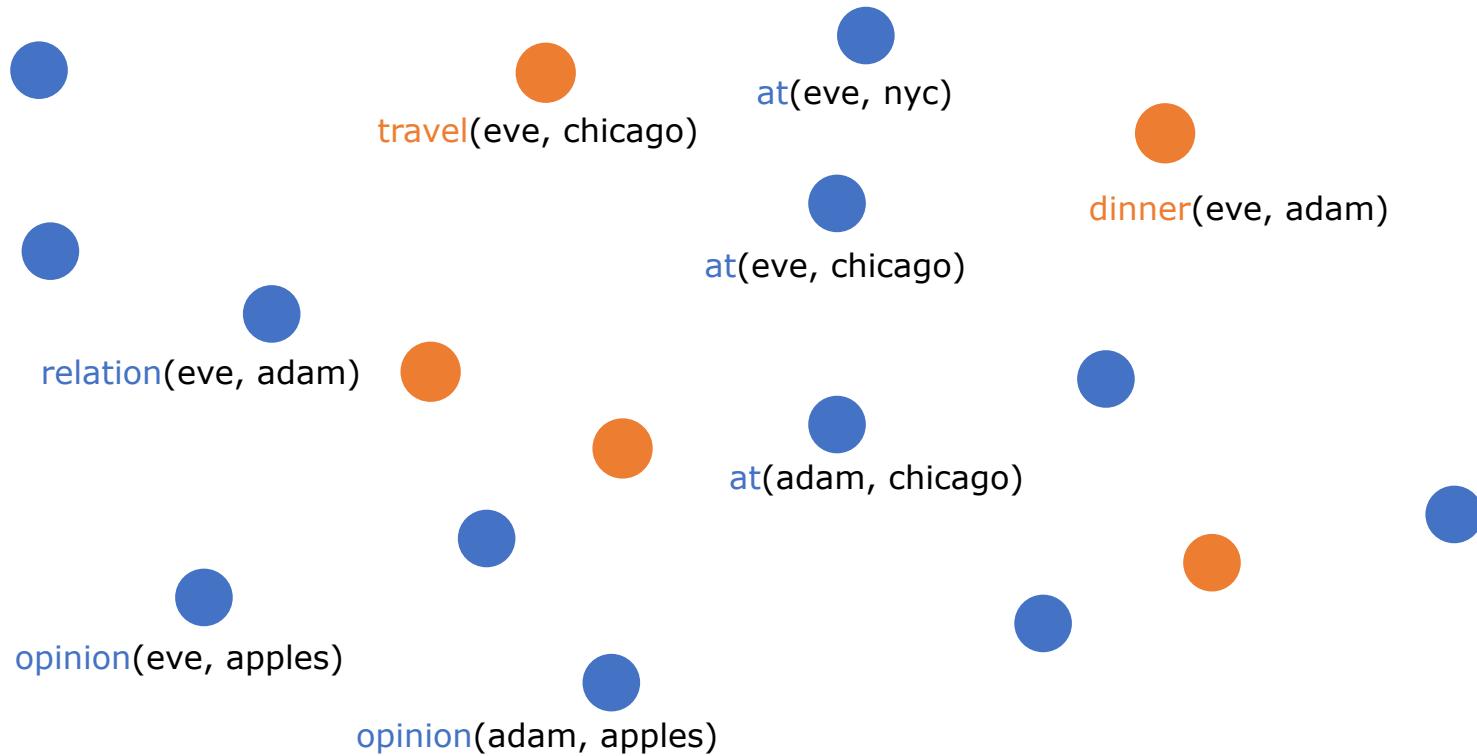
200,000 facts right now

50,000 possible events right now

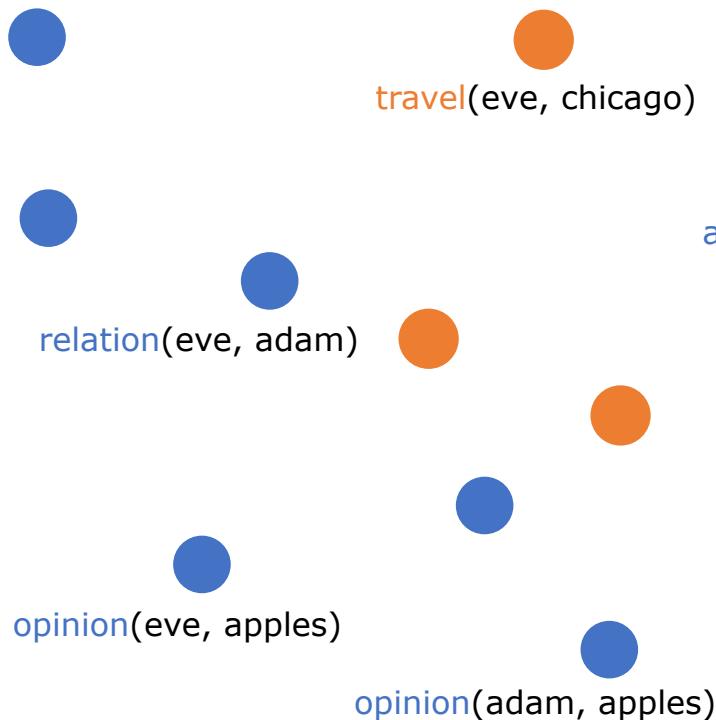
*little language  
to specify a generative model  
of event sequences*



# Deductive Rules, Triggering Rules



# Deductive Rules, Triggering Rules



relation(X, Y)

:- opinion(X, U), opinion(Y, U).

travel(X, P)

:- relation(X, Y), at(Y, P).

!at(X, Q)

← travel(X, P), at(X, Q), P != Q.

at(X, P)

← travel(X, P).

dinner(X, Y)

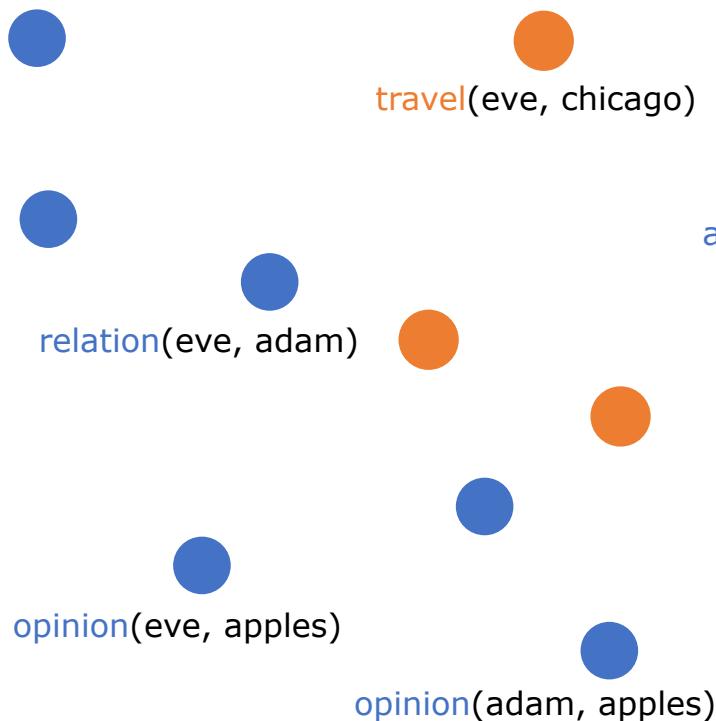
:- relation(X, Y), at(X, P), at(Y, P).

relation(X, Y)

← dinner(X, Y).

# Deductive Rules, Triggering Rules

**logic!**



`relation(X, Y)`

`:- opinion(X, U), opinion(Y, U).`

`travel(X, P)`

`at(eve, nyc)`  
`:- relation(X, Y), at(Y, P).`

`!at(X, Q)`

`at(eve, nyc)`  
`← travel(X, P), at(X, Q), P != Q.`

`at(X, P)`

`← travel(X, P).`

`dinner(X, Y)`

`at(eve, nyc)`  
`:- relation(X, Y), at(X, P), at(Y, P).`

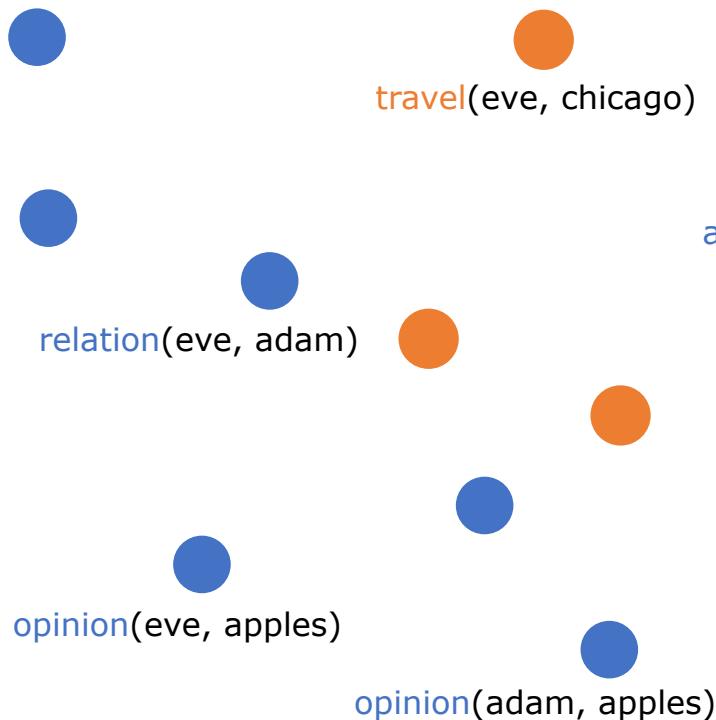
`relation(X, Y)`

`← dinner(X, Y).`

# Deductive Rules, Triggering Rules

which facts are in the database

**logic!**



relation(X, Y)

`:- opinion(X, U), opinion(Y, U).`

travel(X, P)

`:- relation(X, Y), at(Y, P).`

!at(X, Q)

`← travel(X, P), at(X, Q), P != Q.`

at(X, P)

`← travel(X, P).`

dinner(X, Y)

`:- relation(X, Y), at(X, P), at(Y, P).`

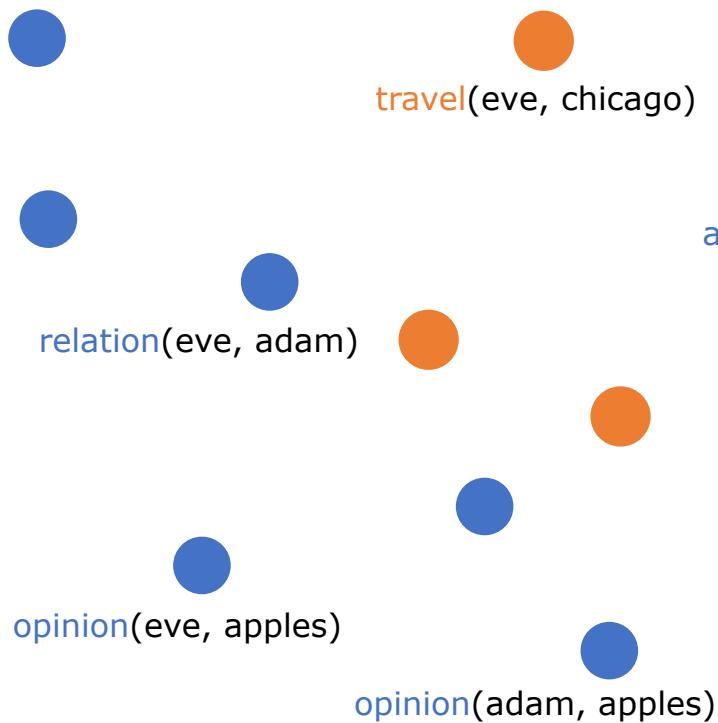
relation(X, Y)

`← dinner(X, Y).`

# Deductive Rules, Triggering Rules

which facts are in the database  
define a trainable neural architecture

**logic!**



`relation(X, Y)`

`:- opinion(X, U), opinion(Y, U).`

`travel(X, P)`

`:- relation(X, Y), at(Y, P).`

`!at(X, Q)`

`← travel(X, P), at(X, Q), P != Q.`

`at(X, P)`

`← travel(X, P).`

`dinner(X, Y)`

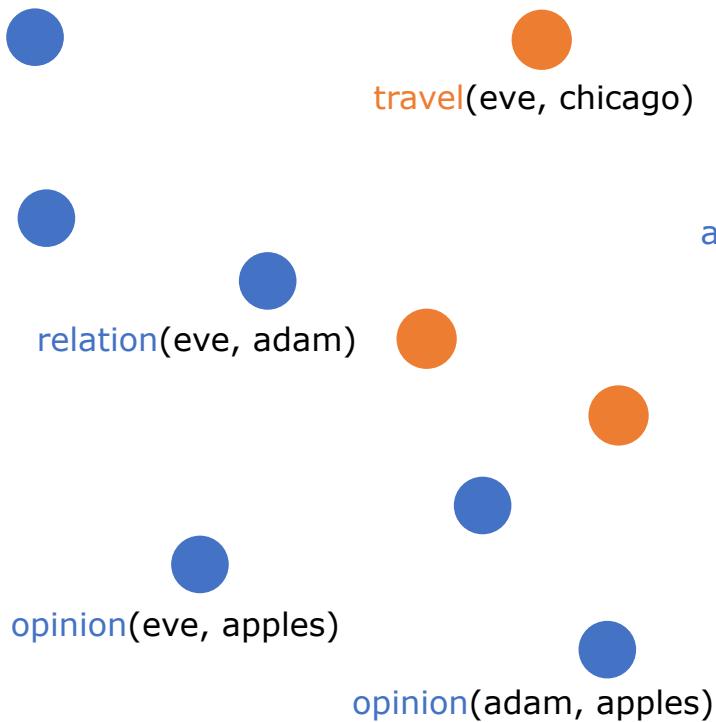
`:- relation(X, Y), at(X, P), at(Y, P).`

`relation(X, Y)`

`← dinner(X, Y).`

# Deductive Rules, Triggering Rules

which facts are in the database  
define a trainable neural architecture  
that computes embeddings of the facts



**logic!**

relation(X, Y)

`:- opinion(X, U), opinion(Y, U).`

travel(X, P)

`:- relation(X, Y), at(Y, P).`

!at(X, Q)

`← travel(X, P), at(X, Q), P != Q.`

at(X, P)

`← travel(X, P).`

dinner(X, Y)

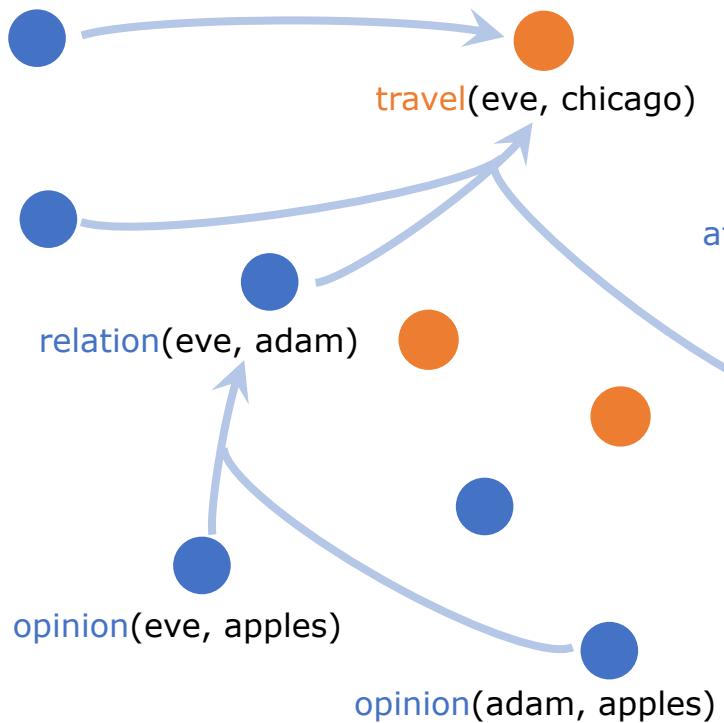
`:- relation(X, Y), at(X, P), at(Y, P).`

relation(X, Y)

`← dinner(X, Y).`

# Deductive Rules, Triggering Rules

which facts are in the database  
define a trainable neural architecture  
that computes embeddings of the facts



**logic!**

`relation(X, Y)`

`:- opinion(X, U), opinion(Y, U).`

`travel(X, P)`

`:- relation(X, Y), at(Y, P).`

`!at(X, Q)`

`← travel(X, P), at(X, Q), P != Q.`

`at(X, P)`

`← travel(X, P).`

`dinner(X, Y)`

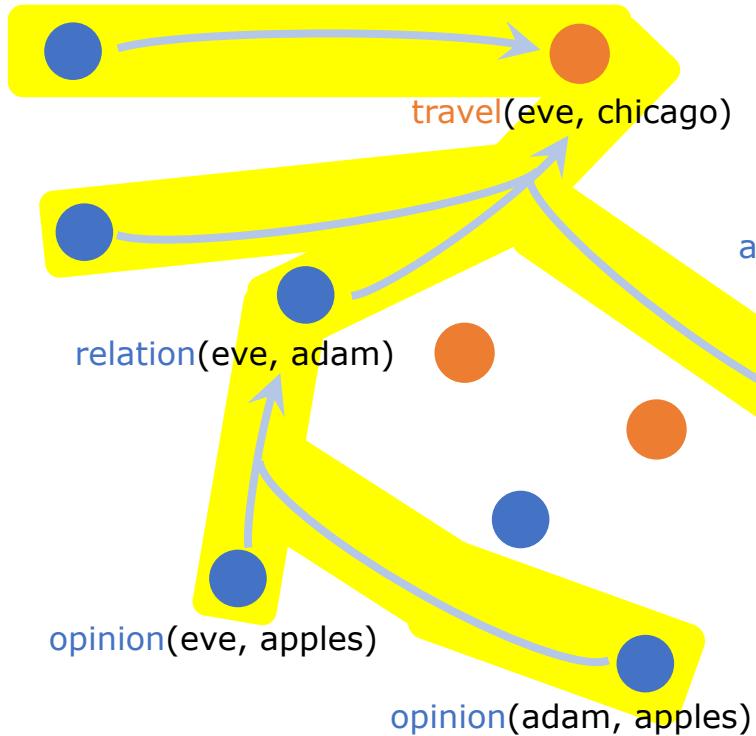
`:- relation(X, Y), at(X, P), at(Y, P).`

`relation(X, Y)`

`← dinner(X, Y).`

# Deductive Rules, Triggering Rules

which facts are in the database  
define a trainable neural architecture  
that computes embeddings of the facts



**logic!**

relation(X, Y)

$\text{:- } \text{opinion}(X, U), \text{opinion}(Y, U).$

travel(X, P)

$\text{:- } \text{relation}(X, Y), \text{at}(Y, P).$

!at(X, Q)

$\leftarrow \text{travel}(X, P), \text{at}(X, Q), P \neq Q.$

at(X, P)

$\leftarrow \text{travel}(X, P).$

dinner(X, Y)

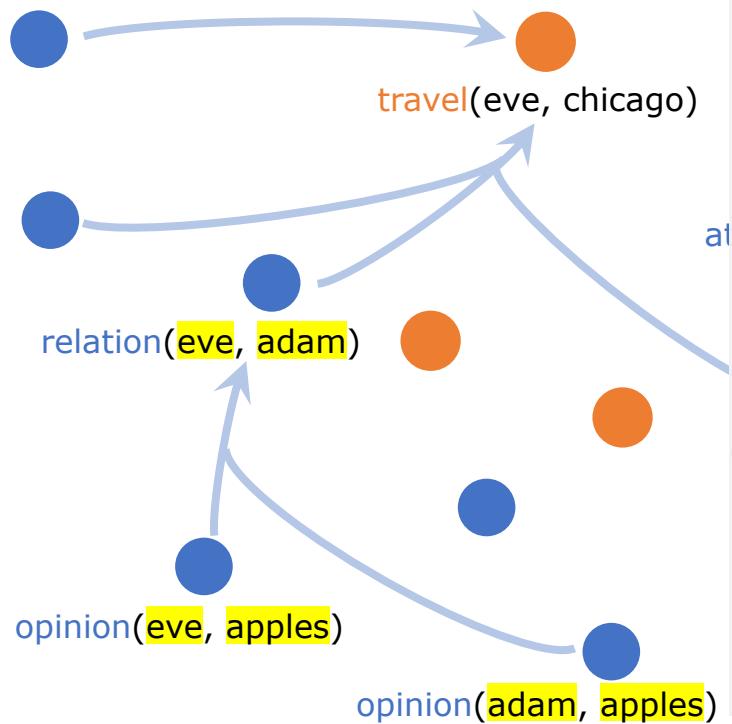
$\text{:- } \text{relation}(X, Y), \text{at}(X, P), \text{at}(Y, P).$

relation(X, Y)

$\leftarrow \text{dinner}(X, Y).$

# Deductive Rules, Triggering Rules

which facts are in the database  
define a trainable neural architecture  
that computes embeddings of the facts



**logic!**

```
relation(X, Y)
:- opinion(X, U), opinion(Y, U).

travel(X, P)
:- relation(X, Y), at(Y, P).

!at(X, Q)
← travel(X, P), at(X, Q), P != Q.

at(X, P)
← travel(X, P).

dinner(X, Y)
:- relation(X, Y), at(X, P), at(Y, P).

relation(X, Y)
← dinner(X, Y).
```

# Datalog → Neural Datalog Through Time

# Datalog → Neural Datalog Through Time

deductive rule

new fact :- old fact<sub>1</sub>, old fact<sub>2</sub>, ...

# Datalog → Neural Datalog Through Time

deductive rule

add to database

new fact :- old fact<sub>1</sub>, old fact<sub>2</sub>, ...

# Datalog → Neural Datalog Through Time

deductive rule

add to database if

new fact :- old fact<sub>1</sub>, old fact<sub>2</sub>, ...

# Datalog → Neural Datalog Through Time

deductive rule

add to database if these are in database

new fact :- old fact<sub>1</sub>, old fact<sub>2</sub>, ...

# Datalog → Neural Datalog Through Time

deductive rule

add to database if these are in database  
new fact :- old fact<sub>1</sub>, old fact<sub>2</sub>, ...  
likes(X, U),

# Datalog → Neural Datalog Through Time

deductive rule

add to database if these are in database  
new fact :- old fact<sub>1</sub>, old fact<sub>2</sub>, ...  
likes(X, U), likes(Y, U)

# Datalog → Neural Datalog Through Time

deductive rule

add to database if these are in database  
new fact :- old fact<sub>1</sub>, old fact<sub>2</sub>, ...  
                          :- likes(X, U), likes(Y, U)

# Datalog → Neural Datalog Through Time

deductive rule

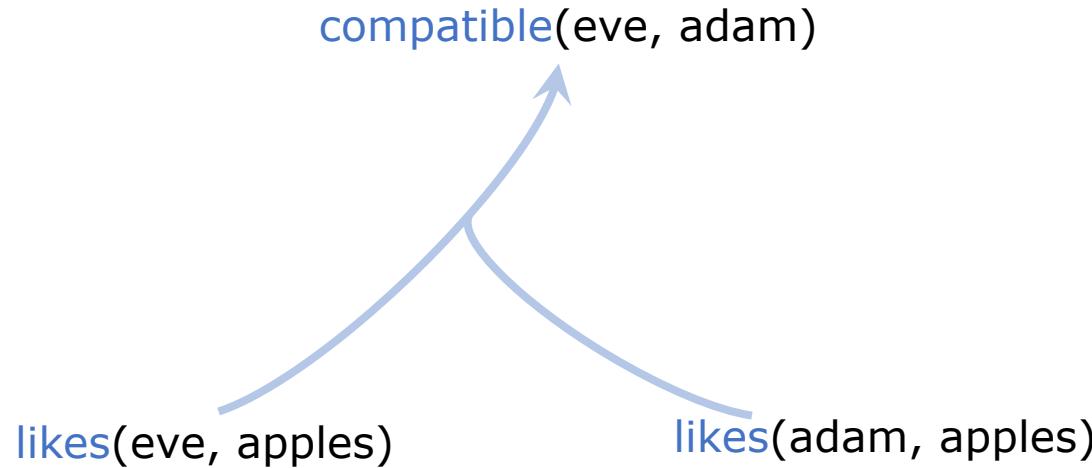
add to database if these are in database  
new fact :- old fact<sub>1</sub>, old fact<sub>2</sub>, ...  
compatible(X, Y) :- likes(X, U), likes(Y, U)

# Datalog → Neural Datalog Through Time

deductive rule

add to database if these are in database  
new fact :- old fact<sub>1</sub>, old fact<sub>2</sub>, ...

compatible(X, Y) :- likes(X, U), likes(Y, U)

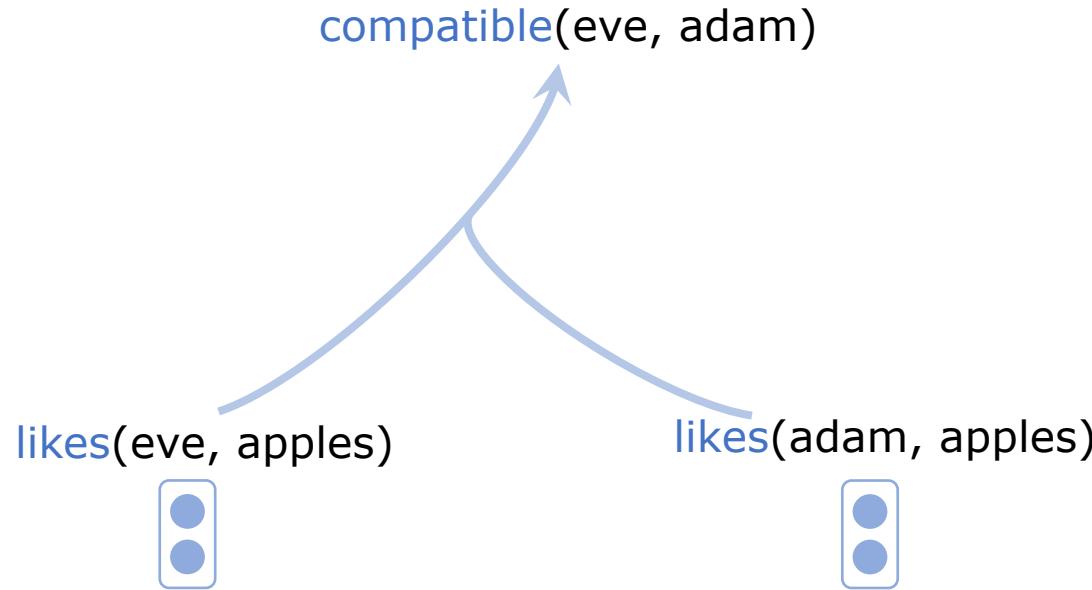


# Datalog → Neural Datalog Through Time

deductive rule

add to database if these are in database  
new fact :- old fact<sub>1</sub>, old fact<sub>2</sub>, ...

compatible(X, Y) :- likes(X, U), likes(Y, U)

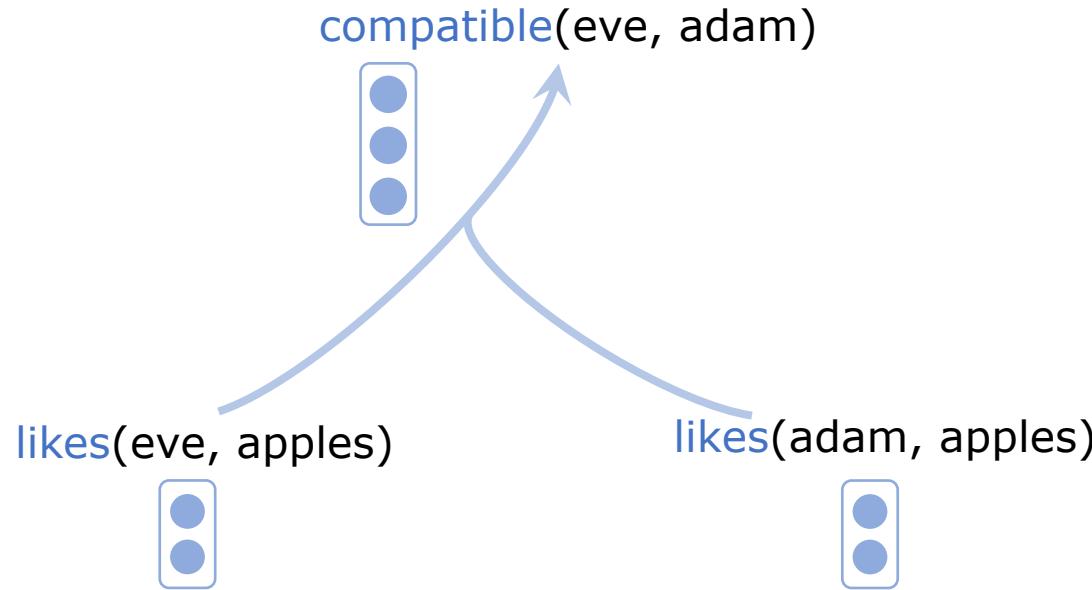


# Datalog → Neural Datalog Through Time

deductive rule

add to database if these are in database  
new fact :- old fact<sub>1</sub>, old fact<sub>2</sub>, ...

compatible(X, Y) :- likes(X, U), likes(Y, U)

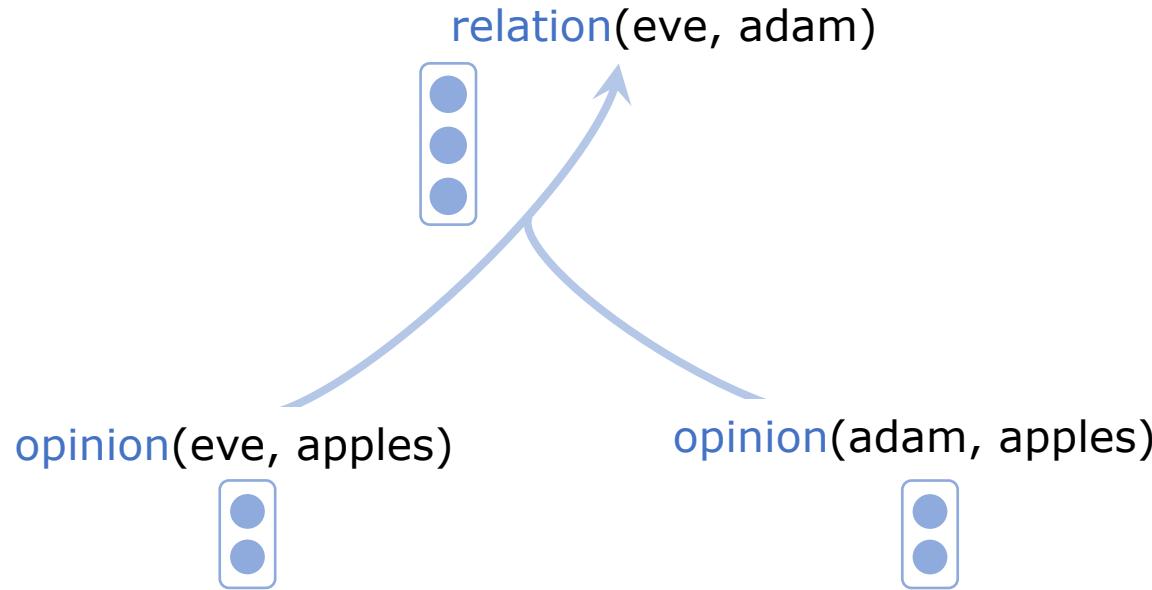


# Datalog → Neural Datalog Through Time

deductive rule

add to database if these are in database  
new fact :- old fact<sub>1</sub>, old fact<sub>2</sub>, ...

compatible(X, Y) :- likes(X, U), likes(Y, U)



# Datalog → Neural Datalog Through Time

deductive rule

add to database if these are in database  
new fact :- old fact<sub>1</sub>, old fact<sub>2</sub>, ...

triggering rule

new fact ← event, old fact<sub>1</sub>, old fact<sub>2</sub>, ...

# Datalog → Neural Datalog Through Time

deductive rule

add to database if these are in database  
new fact :- old fact<sub>1</sub>, old fact<sub>2</sub>, ...

triggering rule

when  
new fact ← event, old fact<sub>1</sub>, old fact<sub>2</sub>, ...

# Datalog → Neural Datalog Through Time

deductive rule

add to database if these are in database  
new fact :- old fact<sub>1</sub>, old fact<sub>2</sub>, ...

triggering rule

this happens  
when  
new fact ← event, old fact<sub>1</sub>, old fact<sub>2</sub>, ...

# Datalog → Neural Datalog Through Time

deductive rule

add to database if these are in database  
new fact :- old fact<sub>1</sub>, old fact<sub>2</sub>, ...

triggering rule

this happens when  
new fact ← event, old fact<sub>1</sub>, old fact<sub>2</sub>, ... while these are in database

# Datalog → Neural Datalog Through Time

deductive rule

add to database if these are in database  
new fact :- old fact<sub>1</sub>, old fact<sub>2</sub>, ...

triggering rule

add to database when this happens while these are in database  
new fact ← event, old fact<sub>1</sub>, old fact<sub>2</sub>, ...

# Datalog → Neural Datalog Through Time

deductive rule

add to database if these are in database  
new fact :- old fact<sub>1</sub>, old fact<sub>2</sub>, ...

triggering rule

add to database when this happens while these are in database  
new fact ← event, old fact<sub>1</sub>, old fact<sub>2</sub>, ...

! old fact ← event, old fact<sub>1</sub>, old fact<sub>2</sub>, ...

# Datalog → Neural Datalog Through Time

deductive rule

add to database if these are in database  
new fact :- old fact<sub>1</sub>, old fact<sub>2</sub>, ...

triggering rule

add to database when this happens while these are in database  
new fact ← event, old fact<sub>1</sub>, old fact<sub>2</sub>, ...

when  
! old fact ← event, old fact<sub>1</sub>, old fact<sub>2</sub>, ...

# Datalog → Neural Datalog Through Time

deductive rule

add to database if these are in database  
new fact :- old fact<sub>1</sub>, old fact<sub>2</sub>, ...

triggering rule

add to database when this happens while these are in database  
new fact ← event, old fact<sub>1</sub>, old fact<sub>2</sub>, ...

when this happens  
! old fact ← event, old fact<sub>1</sub>, old fact<sub>2</sub>, ...

# Datalog → Neural Datalog Through Time

deductive rule

add to database if these are in database  
new fact :- old fact<sub>1</sub>, old fact<sub>2</sub>, ...

triggering rule

add to database when this happens while these are in database  
new fact ← event, old fact<sub>1</sub>, old fact<sub>2</sub>, ...

when this happens while these are in database  
! old fact ← event, old fact<sub>1</sub>, old fact<sub>2</sub>, ...

# Datalog → Neural Datalog Through Time

deductive rule

add to database if these are in database  
new fact :- old fact<sub>1</sub>, old fact<sub>2</sub>, ...

triggering rule

add to database when this happens while these are in database  
new fact ← event, old fact<sub>1</sub>, old fact<sub>2</sub>, ...

delete when this happens while these are in database  
! old fact ← event, old fact<sub>1</sub>, old fact<sub>2</sub>, ...

# Computing the Embeddings

# Computing the Embeddings

relation(eve, adam)

# Computing the Embeddings

`relation(X, Y) :- opinion(X, U), opinion(Y, U)`

`relation(eve, adam)`

# Computing the Embeddings

`relation(X, Y) :- opinion(X, U), opinion(Y, U)`



`relation(eve, adam)`

# Computing the Embeddings

`relation(X, Y) :- opinion(X, U), opinion(Y, U)`



`relation(eve, adam)`  
opinion(adam, apples)  
opinion(eve, apples)

# Computing the Embeddings

`relation(X, Y) :- opinion(X, U), opinion(Y, U)`



`relation(eve, adam)`

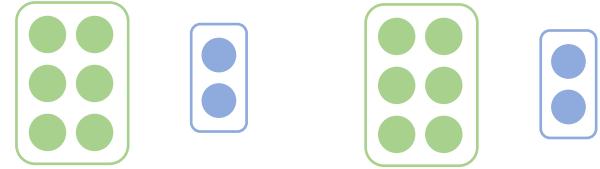
`opinion(adam, apples)`

`opinion(eve, apples)`



# Computing the Embeddings

`relation(X, Y) :- opinion(X, U), opinion(Y, U)`



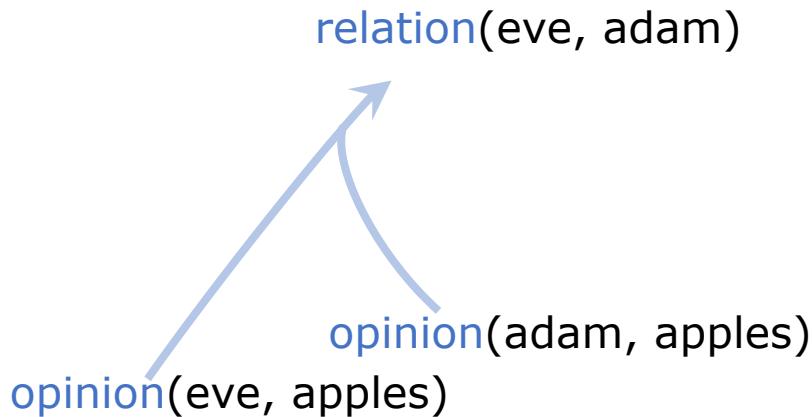
`relation(eve, adam)`  
opinion(eve, apples)  
opinion(adam, apples)



# Computing the Embeddings

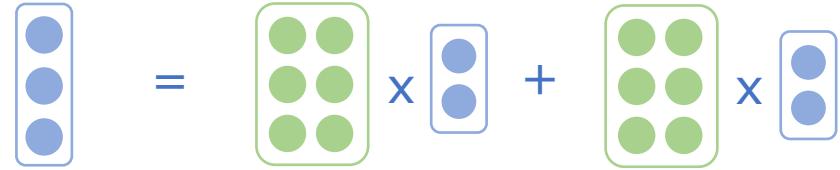
`relation(X, Y) :- opinion(X, U), opinion(Y, U)`

$$= \begin{array}{c} \text{green square grid} \\ \times \end{array} + \begin{array}{c} \text{green square grid} \\ \times \end{array}$$

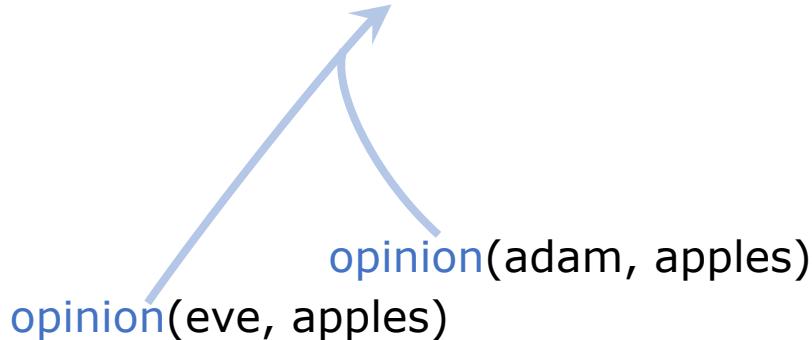


# Computing the Embeddings

`relation(X, Y) :- opinion(X, U), opinion(Y, U)`



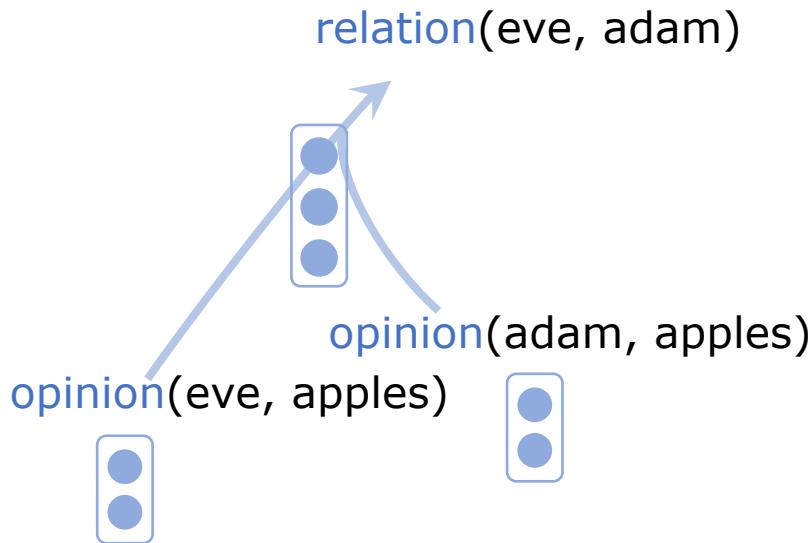
`relation(eve, adam)`



# Computing the Embeddings

`relation(X, Y) :- opinion(X, U), opinion(Y, U)`

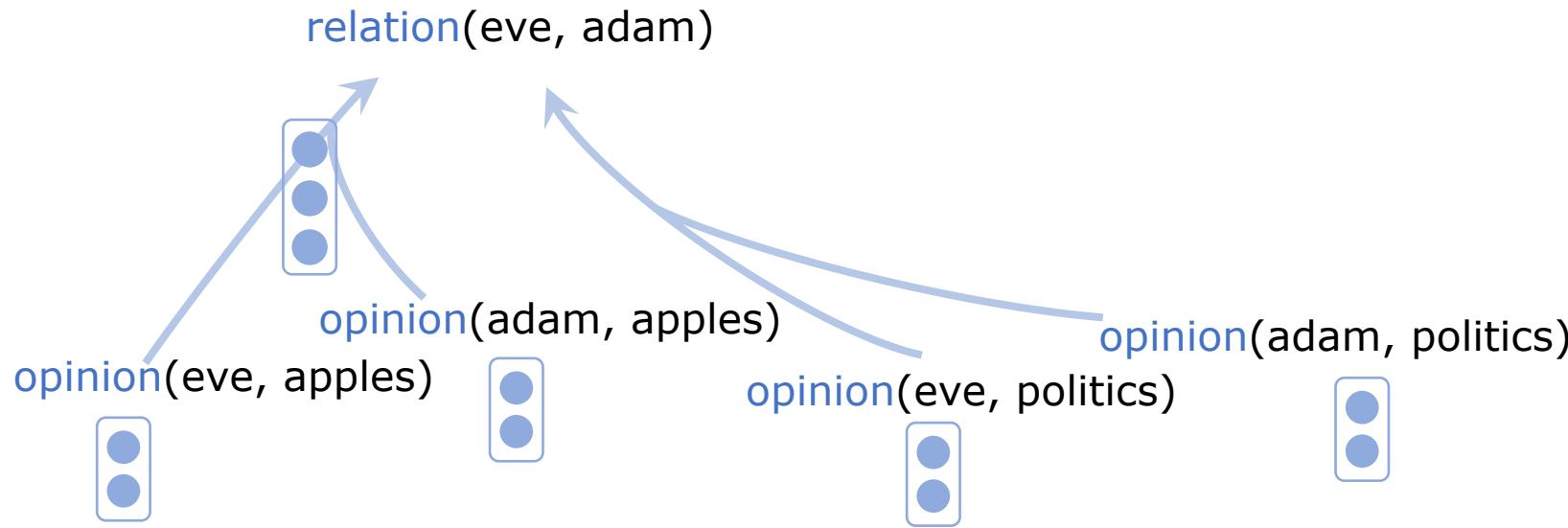
$$= \begin{matrix} \text{green circles} \\ \times \end{matrix} + \begin{matrix} \text{green circles} \\ \times \end{matrix}$$



# Computing the Embeddings

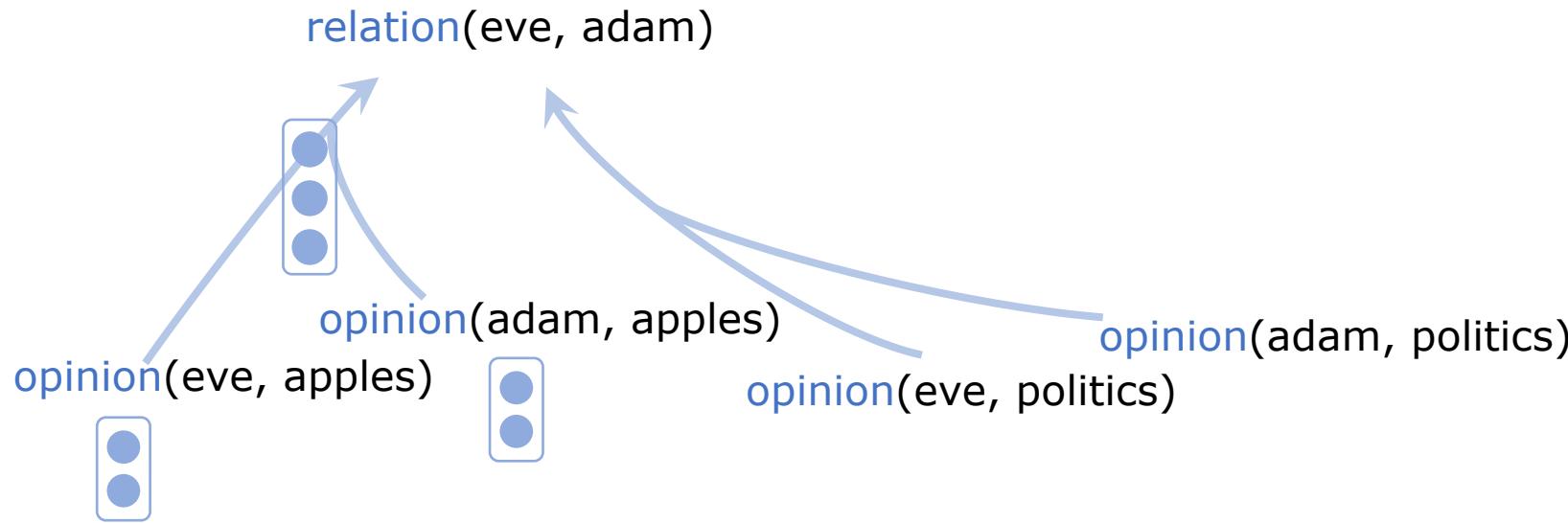
`relation(X, Y) :- opinion(X, U), opinion(Y, U)`

$$= \begin{matrix} \text{green circles} \\ \times \end{matrix} x + \begin{matrix} \text{green circles} \\ \times \end{matrix} x$$



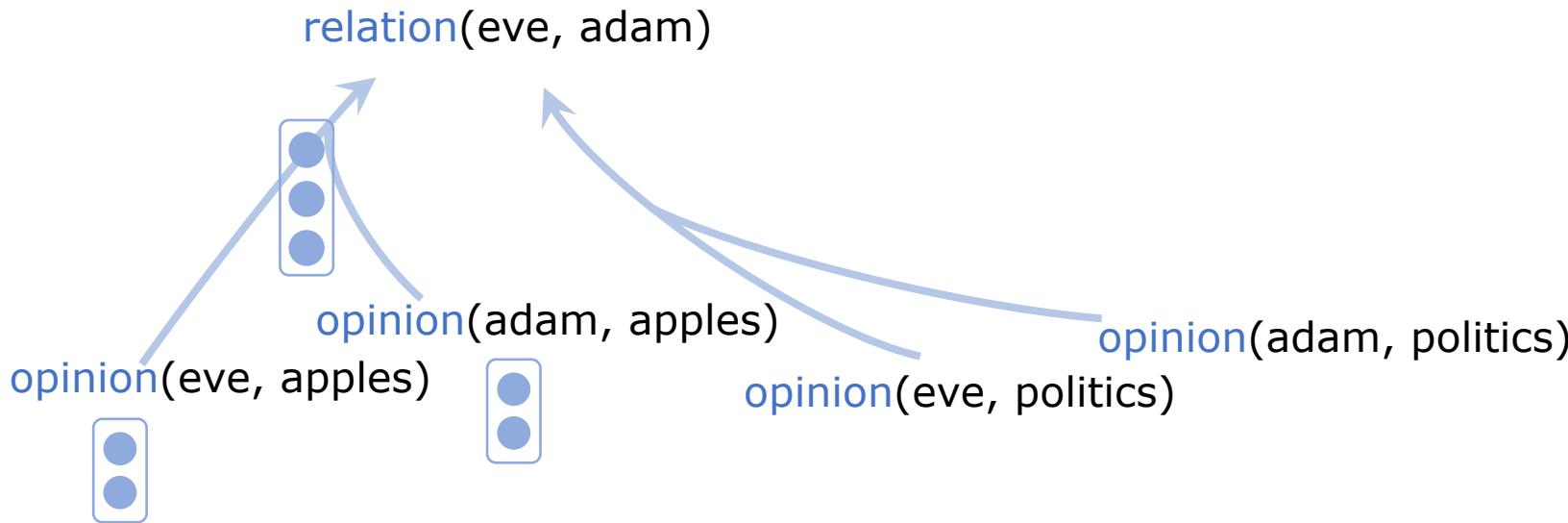
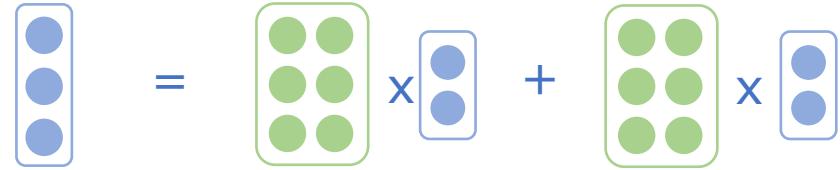
# Computing the Embeddings

$$\text{relation}(X, Y) :- \text{opinion}(X, U), \text{opinion}(Y, U)$$
$$= \begin{array}{c} \text{green 3x3 grid} \\ \times \end{array} + \begin{array}{c} \text{green 3x3 grid} \\ \times \end{array}$$



# Computing the Embeddings

`relation(X, Y) :- opinion(X, U), opinion(Y, U)`

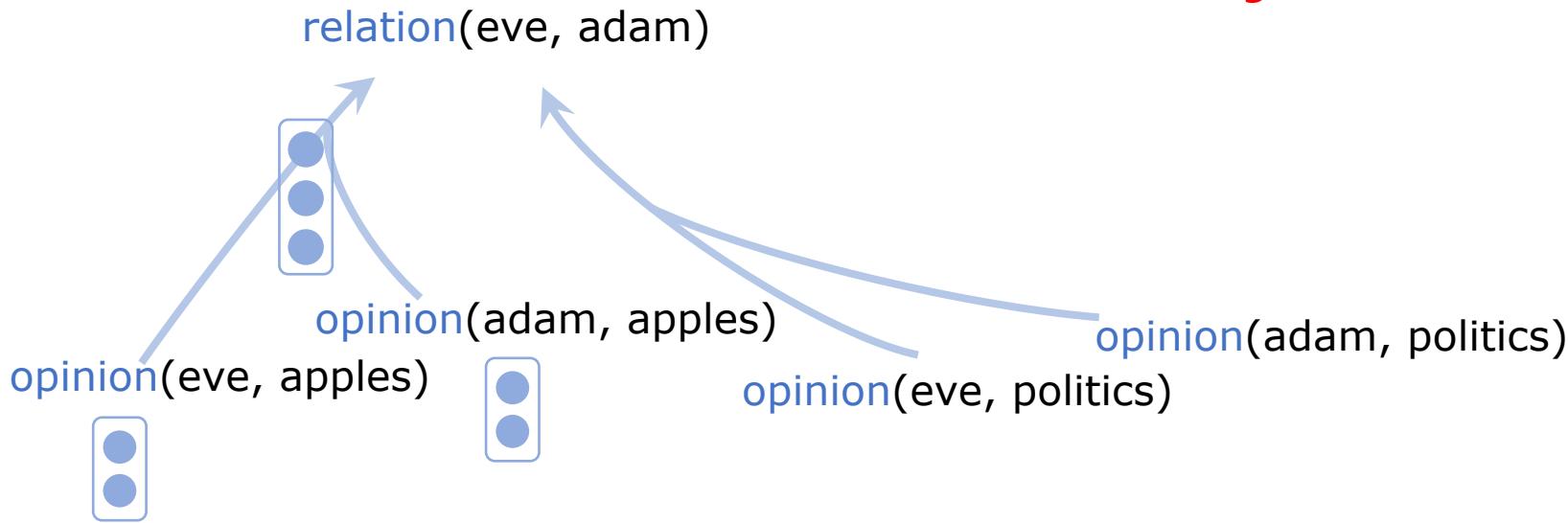


# Computing the Embeddings

`relation(X, Y) :- opinion(X, U), opinion(Y, U)`

$$\begin{array}{c} \text{blue box} \\ \text{---} \\ \text{blue box} \end{array} = \begin{array}{c} \text{green box} \\ \times \quad \text{blue box} \\ \text{green box} \end{array} + \begin{array}{c} \text{green box} \\ \times \quad \text{blue box} \\ \text{green box} \end{array}$$

different inputs  
same params

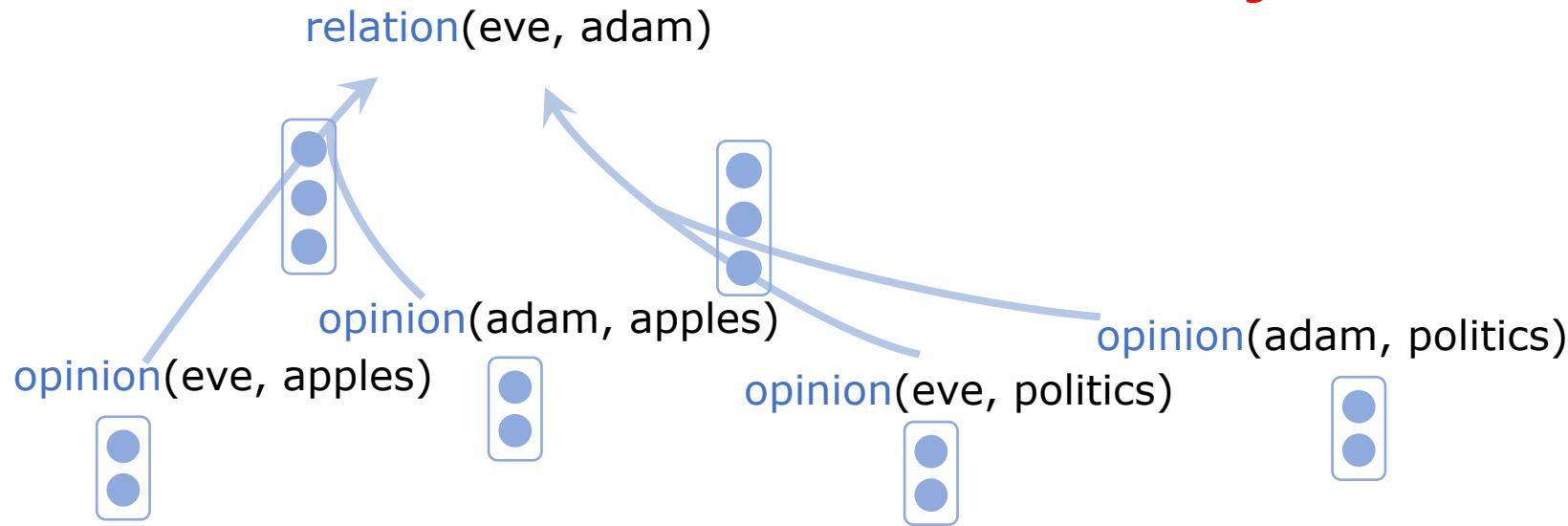


# Computing the Embeddings

`relation(X, Y) :- opinion(X, U), opinion(Y, U)`

$$= \begin{matrix} \text{green box} \\ \times \end{matrix} + \begin{matrix} \text{green box} \\ \times \end{matrix}$$

different inputs  
same params

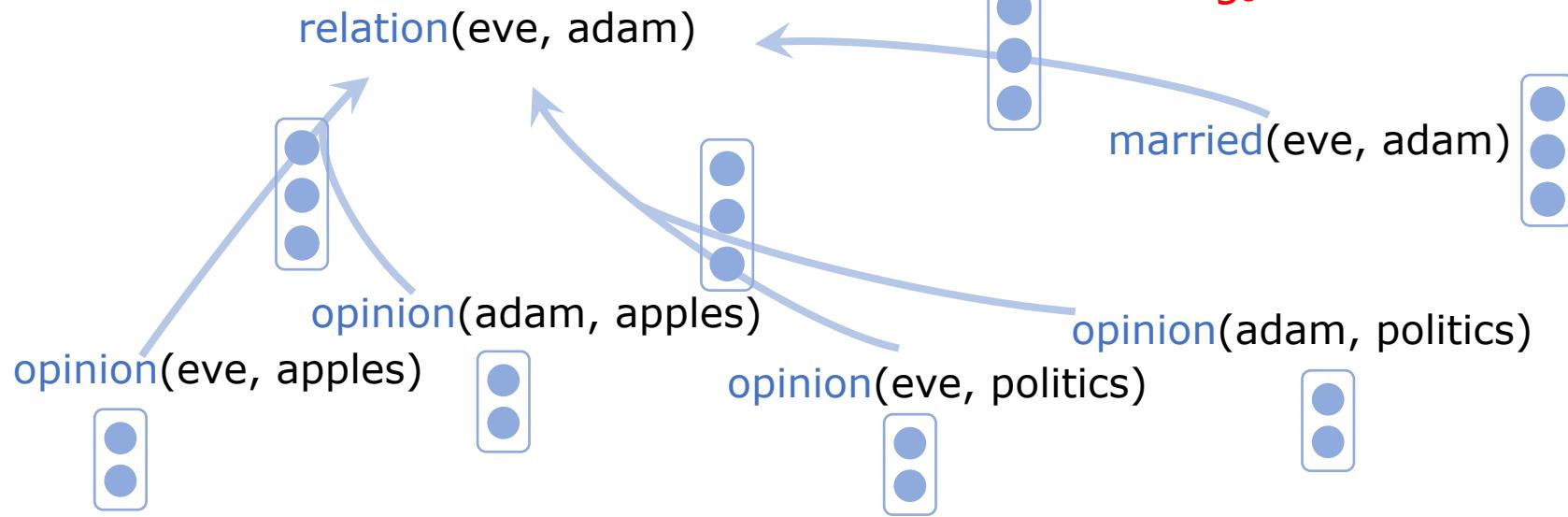


# Computing the Embeddings

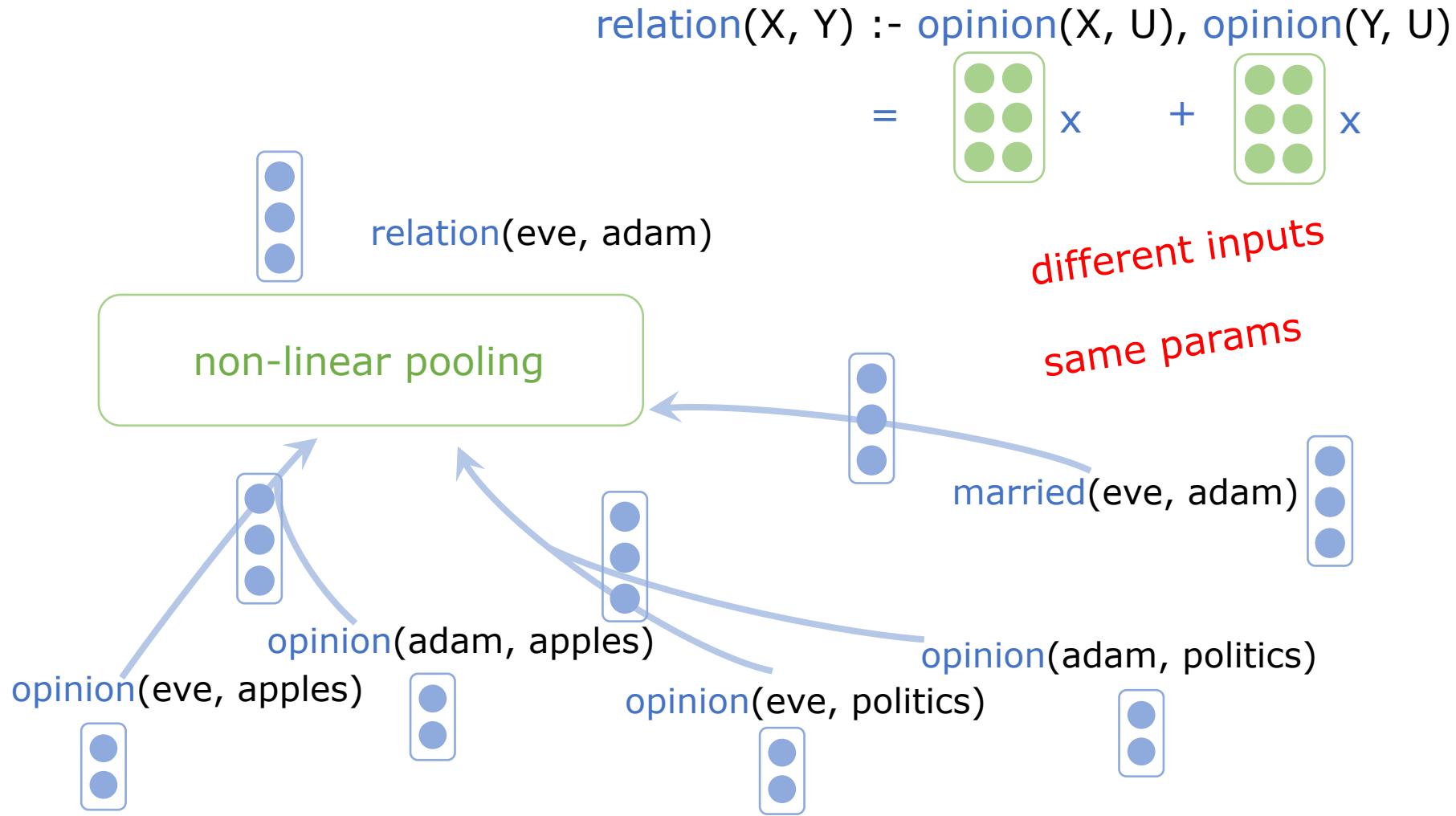
`relation(X, Y) :- opinion(X, U), opinion(Y, U)`

$$= \begin{matrix} \text{green circles} \\ \times \end{matrix} x + \begin{matrix} \text{green circles} \\ \times \end{matrix} x$$

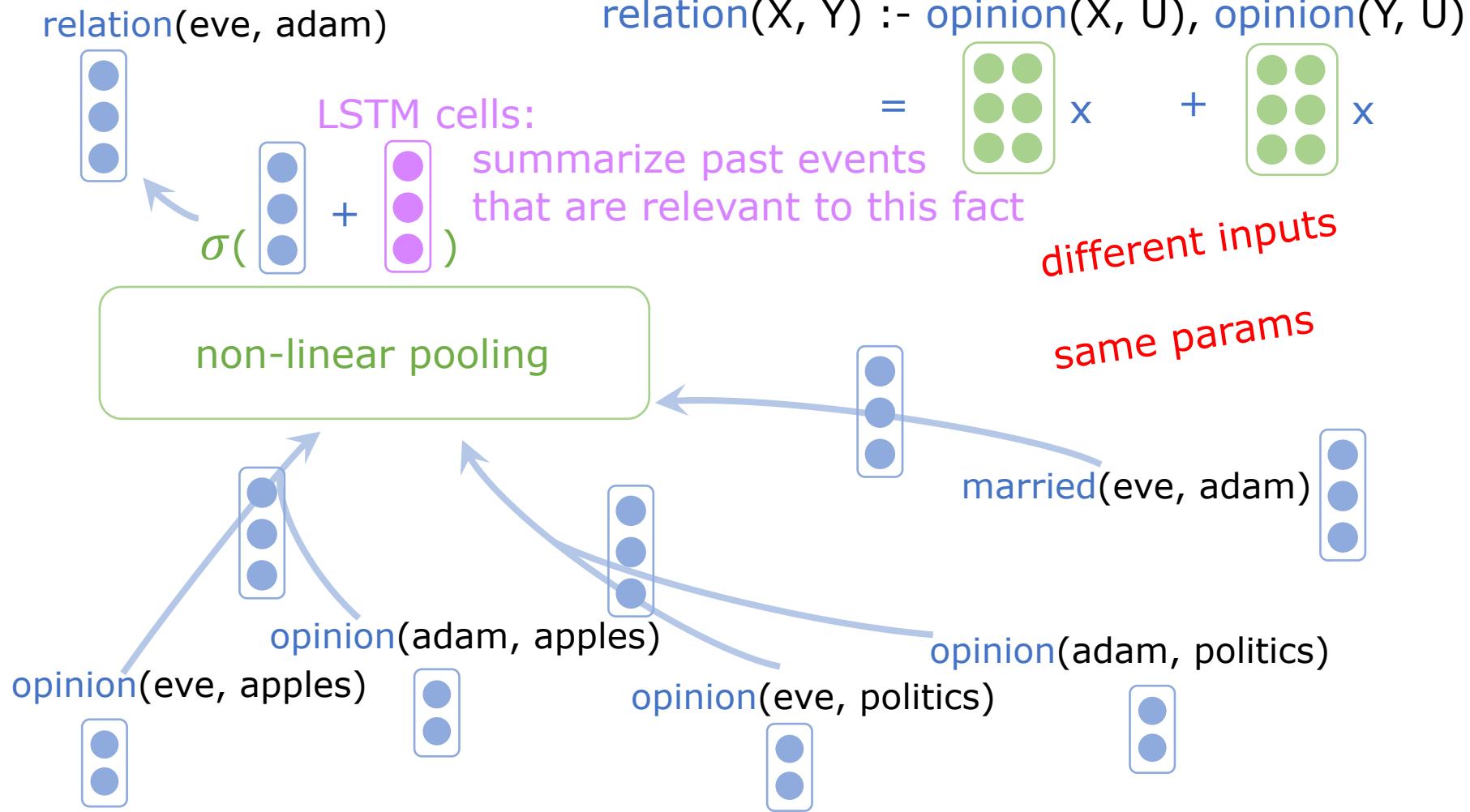
different inputs  
same params



# Computing the Embeddings



# Computing the Embeddings



# Computing **Embeddings** & **Probabilities**

# Computing Embeddings & Probabilities

relation(eve, adam)



relation(eve, cain)



...

# Computing Embeddings & Probabilities

travel(X, P) :- relation(X, Y), at(Y, P).



...

relation(eve, adam)

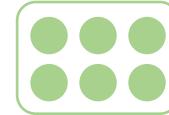


relation(eve, cain)



# Computing Embeddings & Probabilities

travel(X, P) :- relation(X, Y), at(Y, P).



...

relation(eve, adam)



relation(eve, cain)



at(adam, chicago)



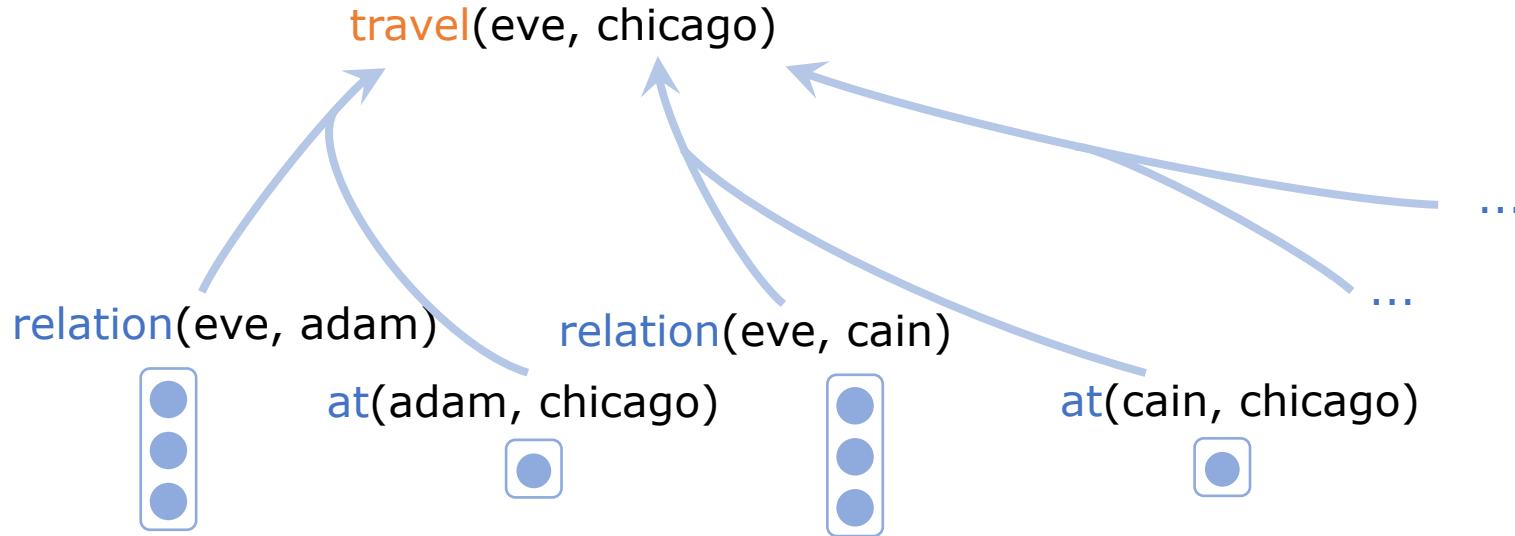
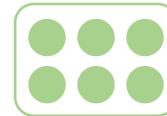
at(cain, chicago)



...

# Computing Embeddings & Probabilities

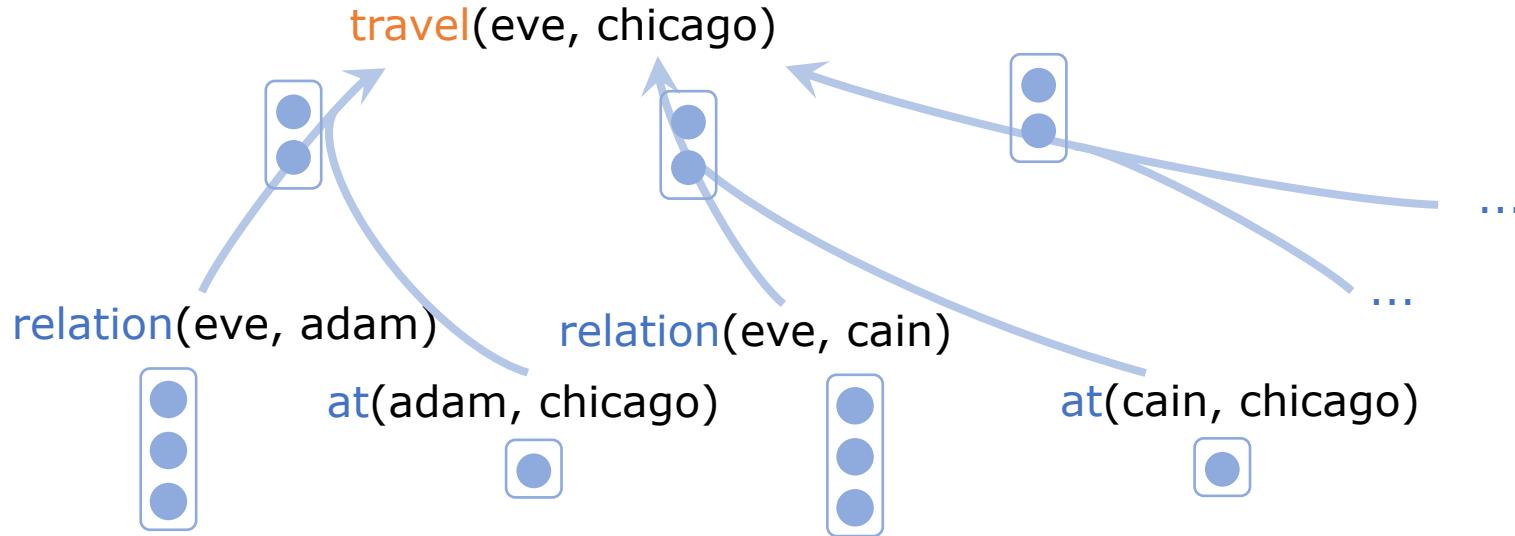
`travel(X, P) :- relation(X, Y), at(Y, P).`



# Computing Embeddings & Probabilities

`travel(X, P) :- relation(X, Y), at(Y, P).`

$$\begin{array}{c} \text{blue} \\ \text{blue} \end{array} = \begin{array}{c} \text{green} \\ \text{green} \\ \text{green} \\ \text{green} \end{array} \times \begin{array}{c} \text{blue} \\ \text{blue} \end{array} + \begin{array}{c} \text{green} \\ \text{green} \end{array} \times \begin{array}{c} \text{blue} \end{array}$$



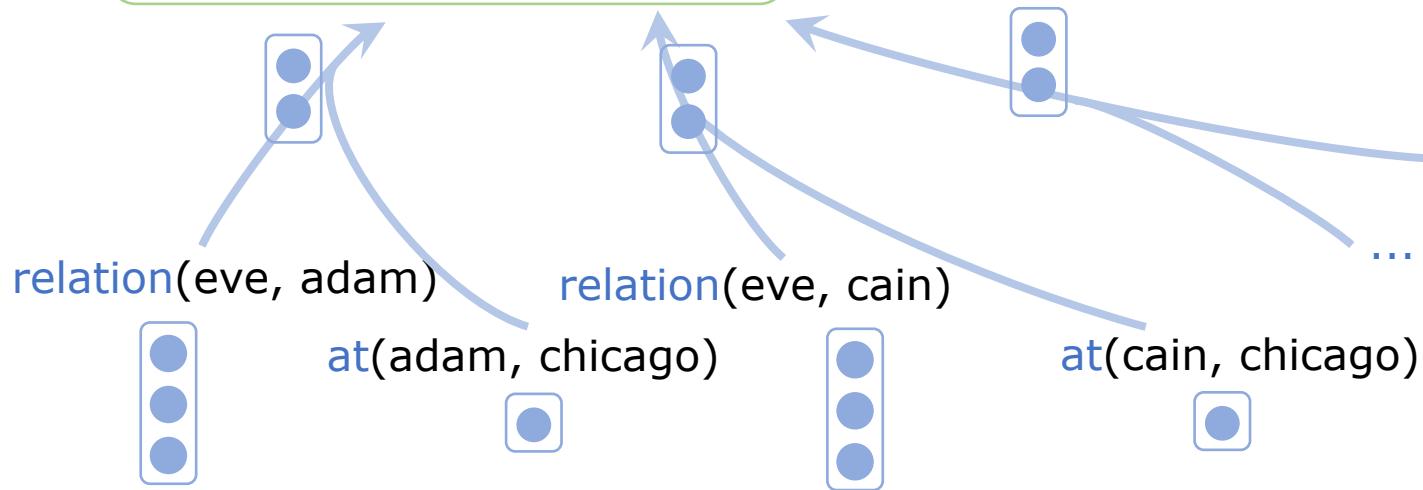
# Computing Embeddings & Probabilities

`travel(X, P) :- relation(X, Y), at(Y, P).`

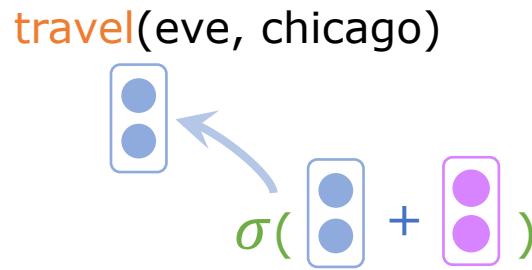
$$\begin{array}{c} \text{blue} \\ \text{blue} \end{array} = \begin{array}{c} \text{green} \\ \text{green} \\ \text{green} \\ \text{green} \end{array} \times \begin{array}{c} \text{blue} \\ \text{blue} \end{array} + \begin{array}{c} \text{green} \\ \text{green} \end{array} \times \begin{array}{c} \text{blue} \end{array}$$

`travel(eve, chicago)`

non-linear pooling

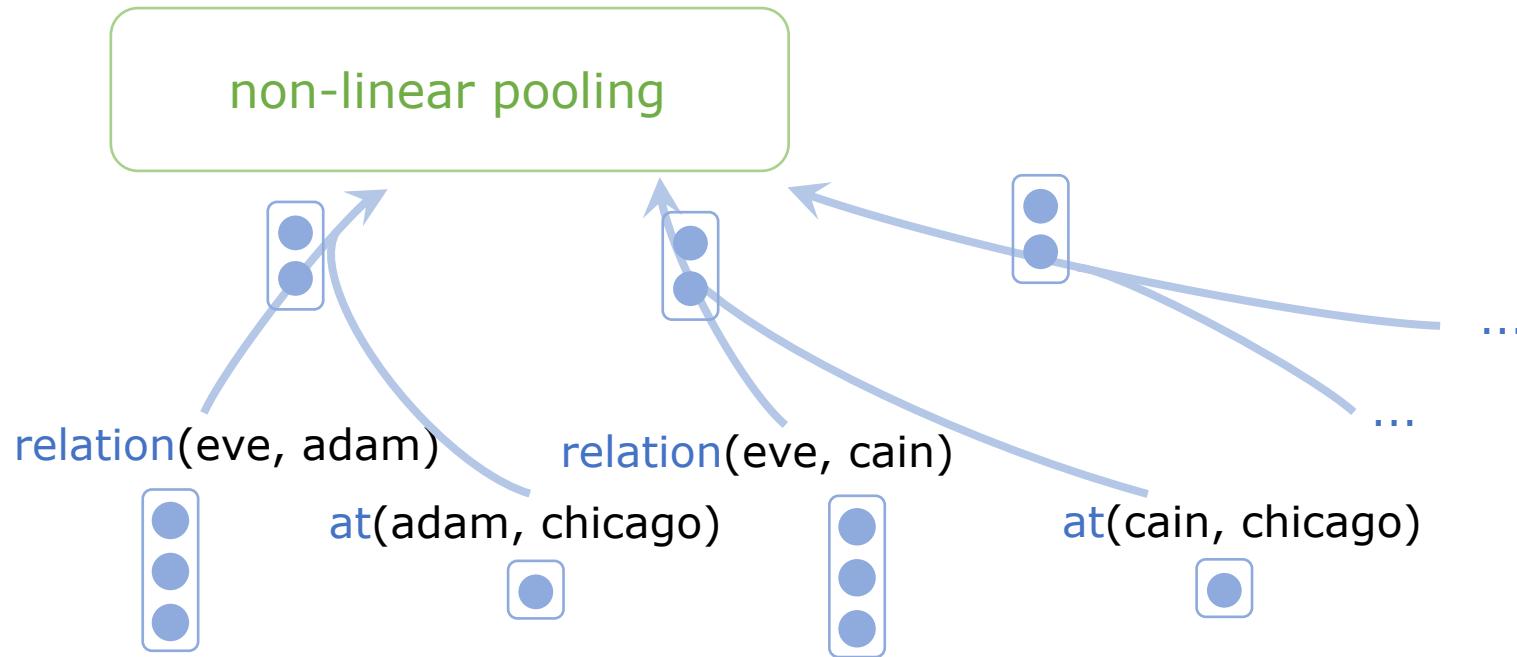


# Computing Embeddings & Probabilities



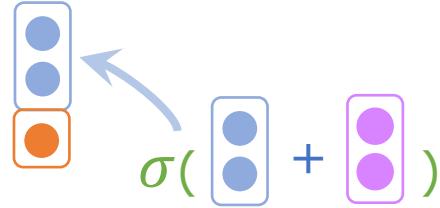
`travel(X, P) :- relation(X, Y), at(Y, P).`

$$\text{blue} = \text{green} \times \text{blue} + \text{green} \times \text{blue}$$



# Computing Embeddings & Probabilities

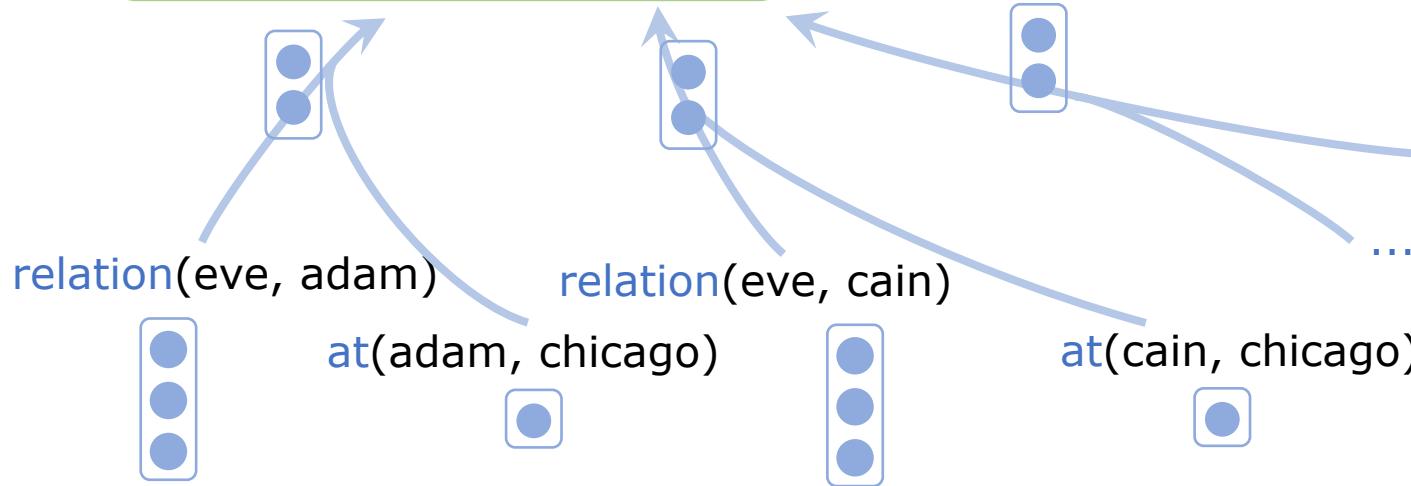
travel(eve, chicago)



travel(X, P) :- relation(X, Y), at(Y, P).

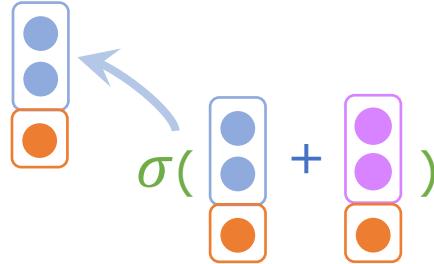
$$\begin{array}{c} \text{blue rectangle} \\ = \end{array} \quad \begin{array}{c} \text{green rectangle} \\ \times \end{array} \quad \begin{array}{c} \text{blue rectangle} \\ + \end{array} \quad \begin{array}{c} \text{green rectangle} \\ \times \end{array} \quad \begin{array}{c} \text{blue rectangle} \end{array}$$

non-linear pooling



# Computing Embeddings & Probabilities

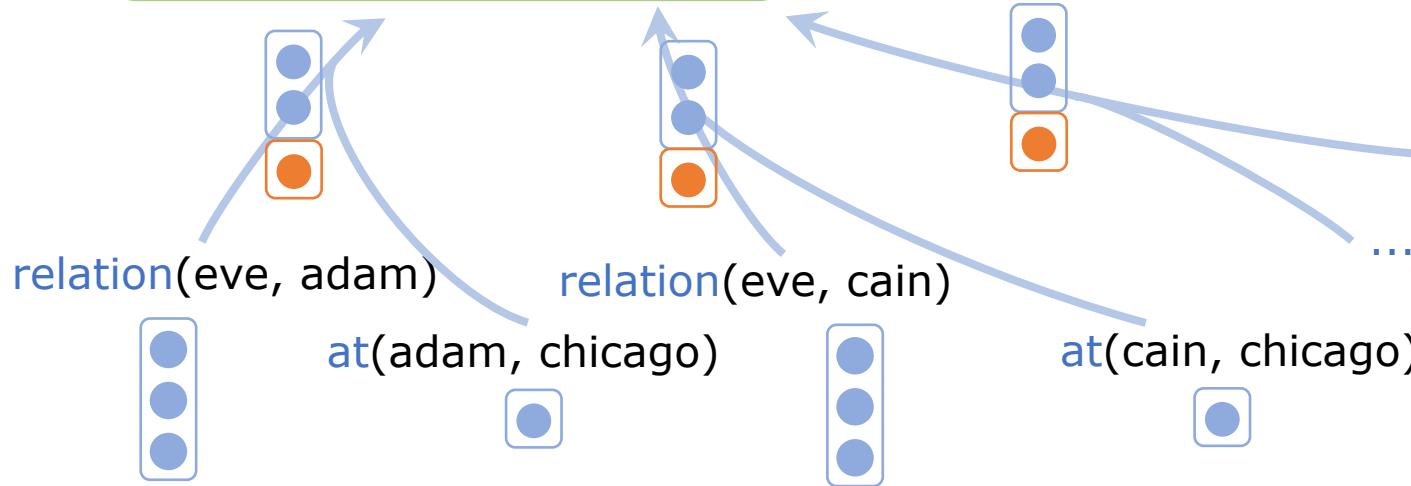
travel(eve, chicago)



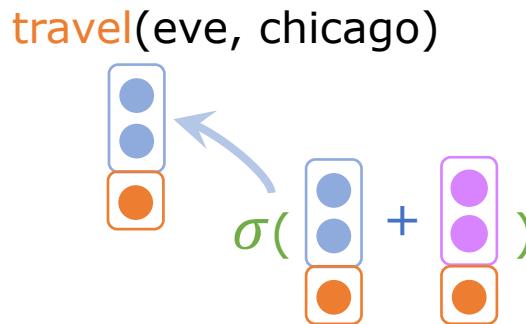
travel( $X, P$ ) :- relation( $X, Y$ ), at( $Y, P$ ).

$$\begin{array}{c} \text{blue rectangle with 3 blue circles} \\ = \end{array} \begin{array}{c} \text{green rectangle with 4 green circles} \\ \times \end{array} \begin{array}{c} \text{blue rectangle with 2 blue circles} \\ + \end{array} \begin{array}{c} \text{green rectangle with 2 green circles} \\ \times \end{array} \begin{array}{c} \text{blue rectangle with 1 blue circle} \end{array}$$

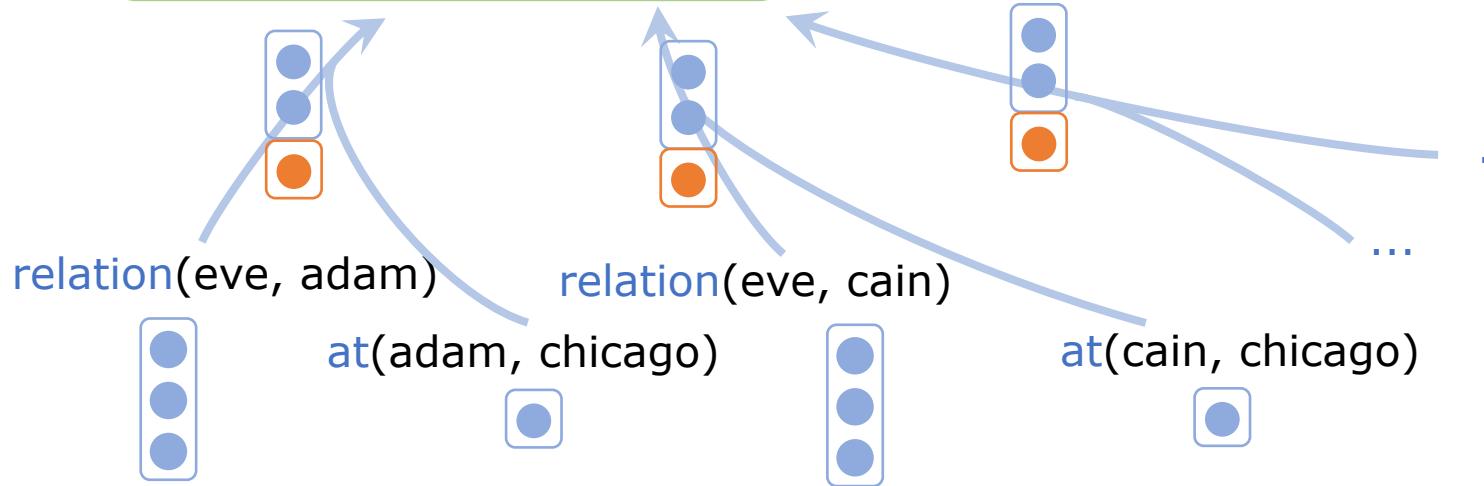
non-linear pooling



# Computing Embeddings & Probabilities



non-linear pooling



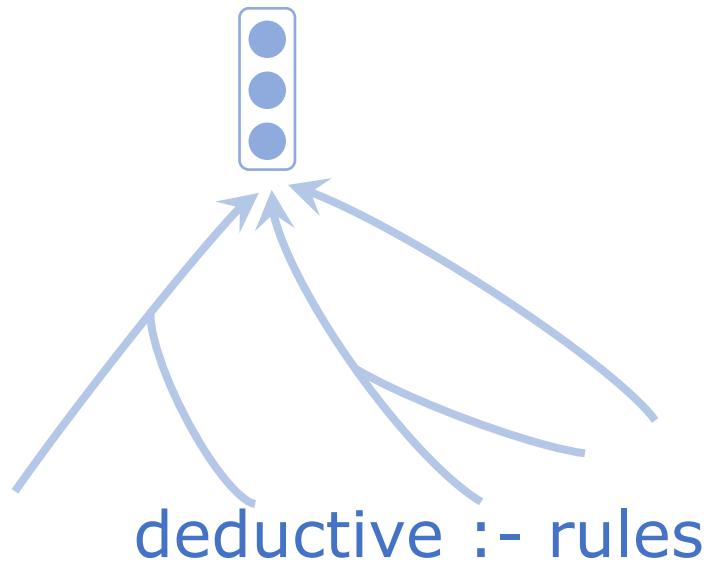
travel(X, P) :- relation(X, Y), at(Y, P).

$$\text{travel}(X, P) = \text{relation}(X, Y) \times \text{at}(Y, P)$$

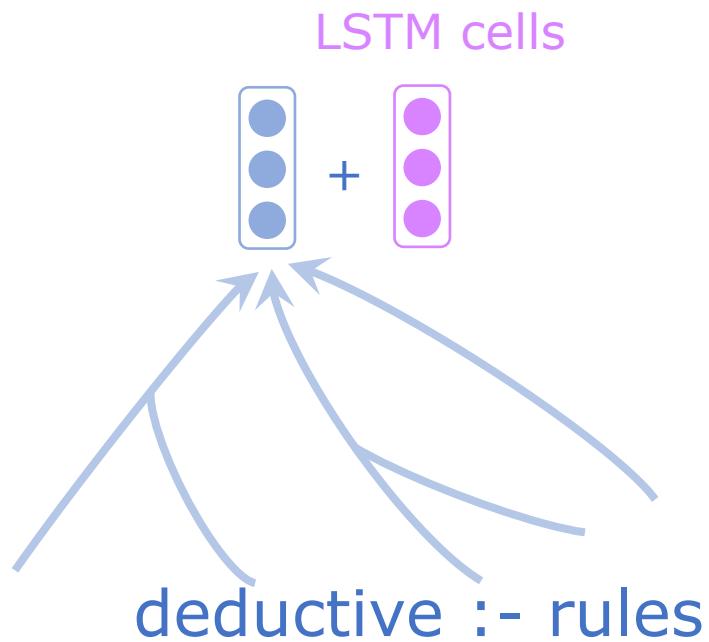
extra dimension for probability

# Rules → Deep Recurrent Neural Net

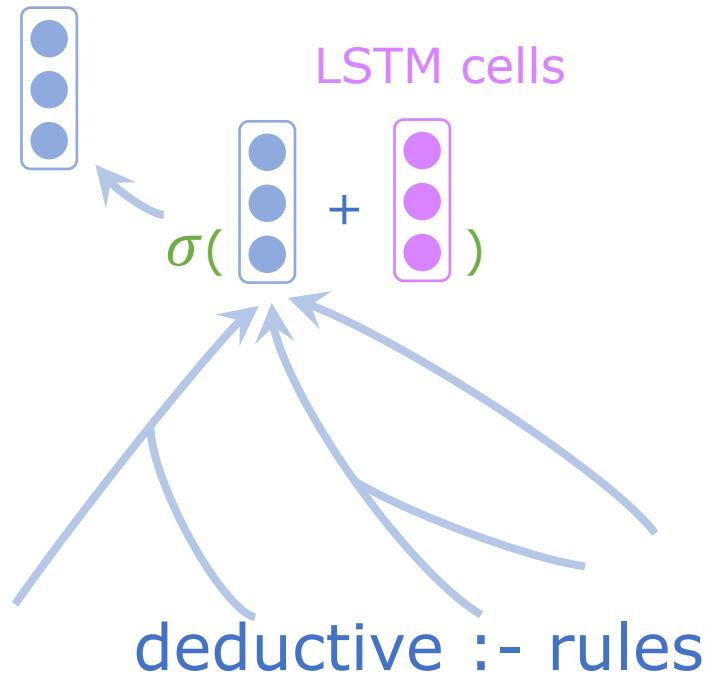
# Rules → Deep Recurrent Neural Net



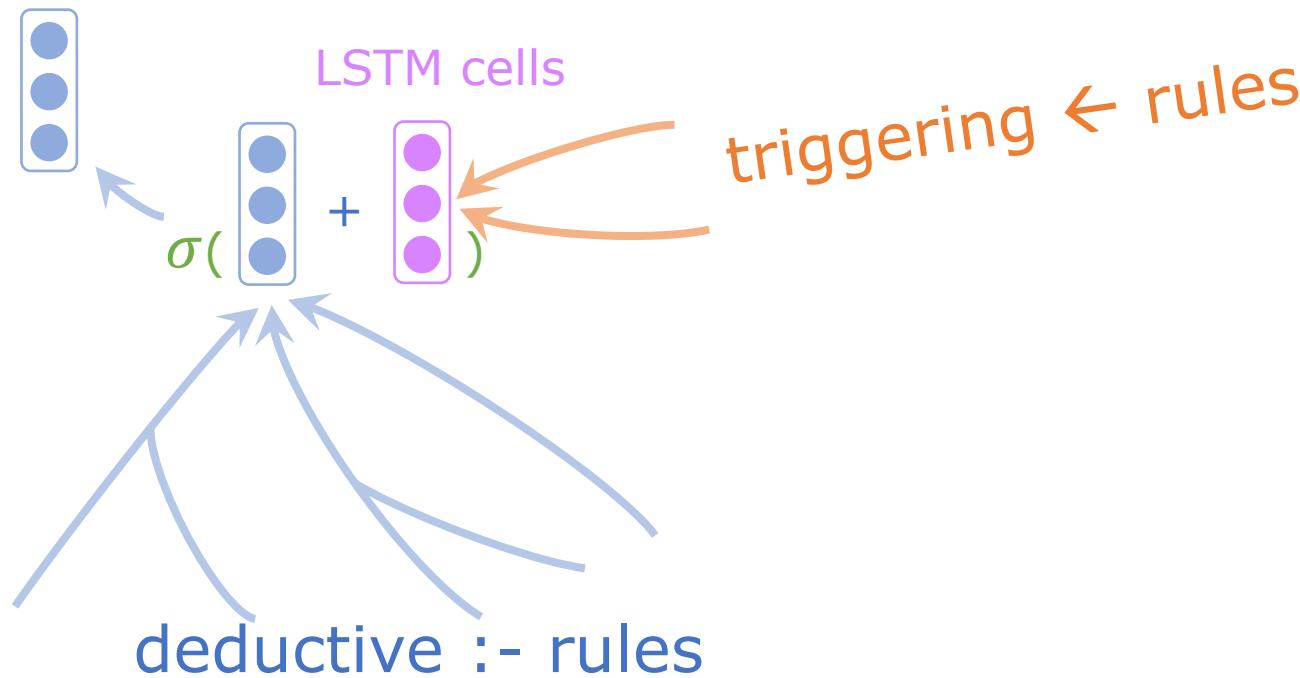
# Rules → Deep Recurrent Neural Net



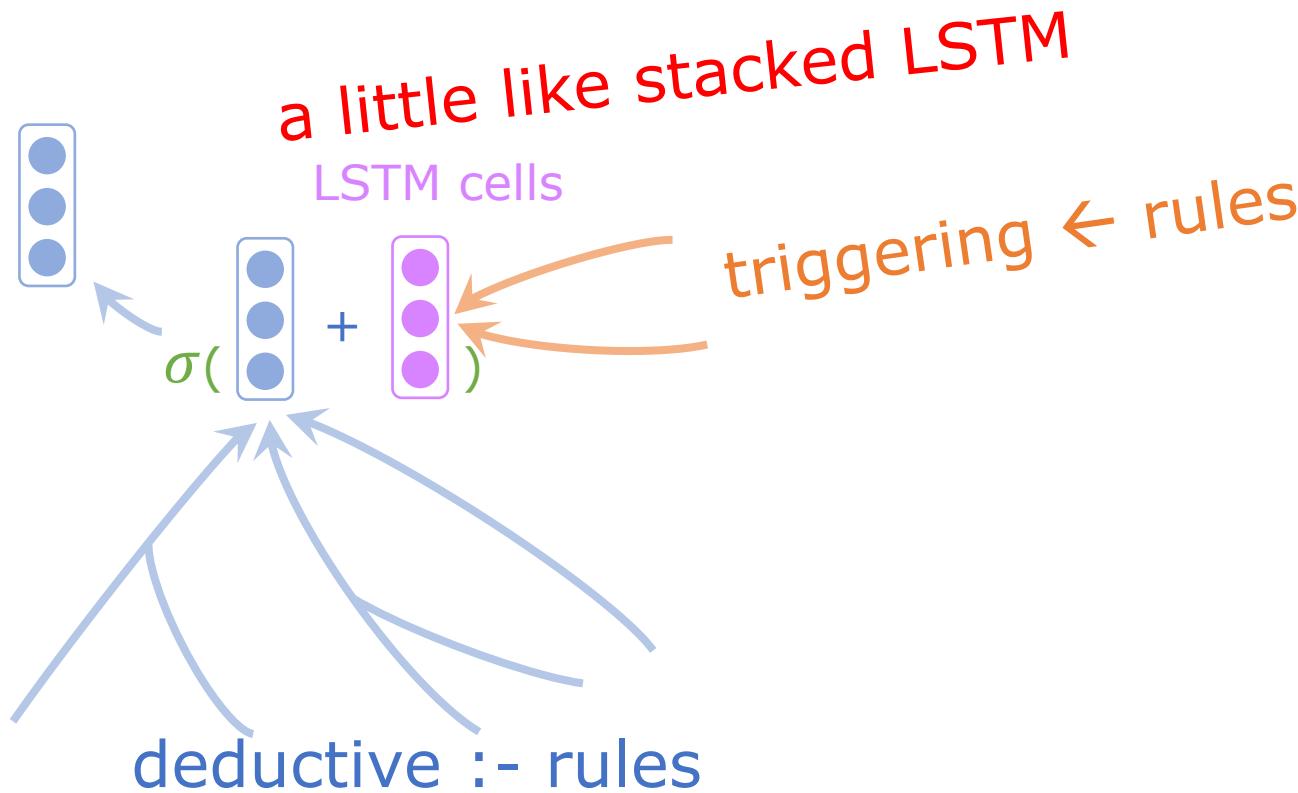
# Rules → Deep Recurrent Neural Net



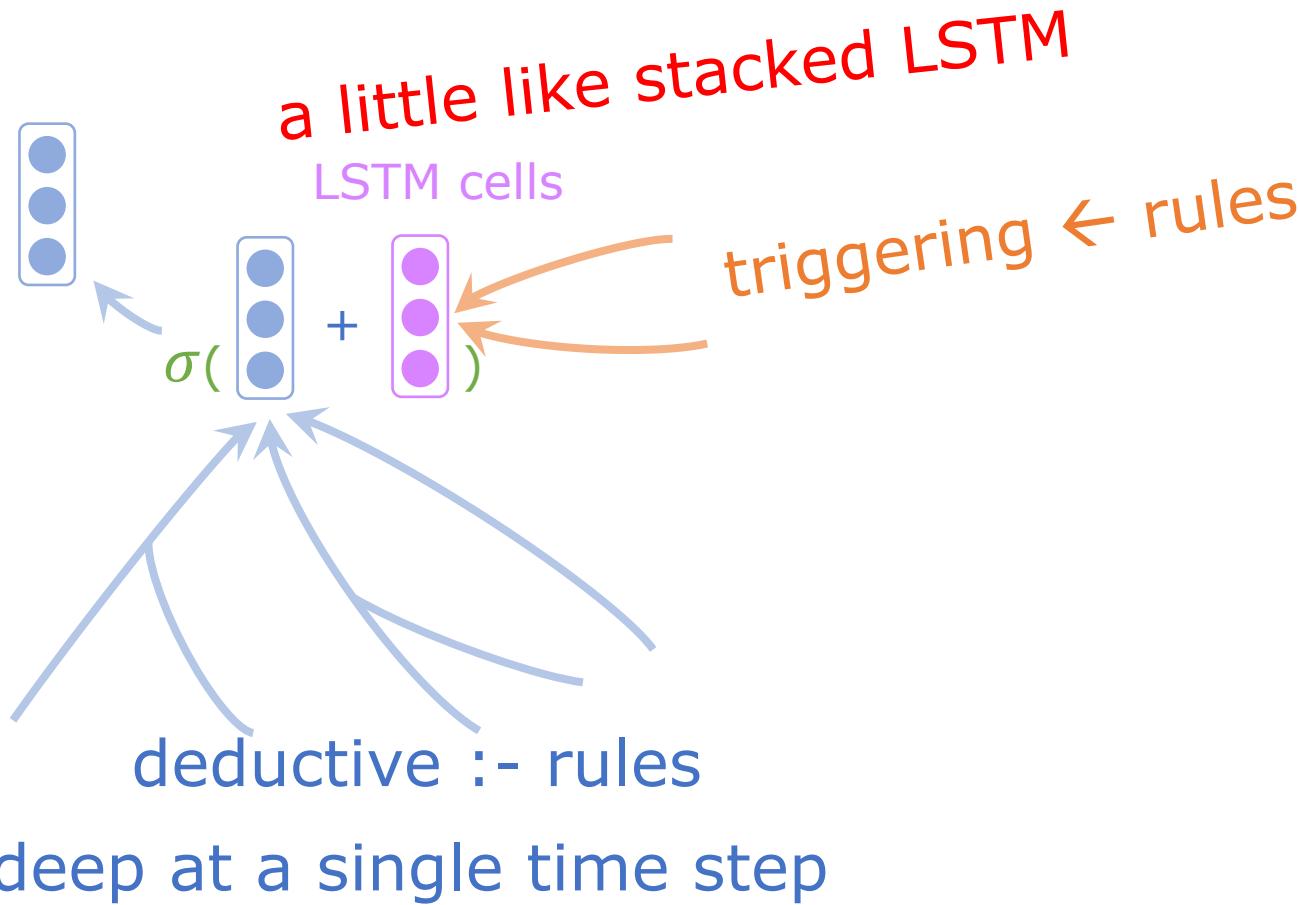
# Rules → Deep Recurrent Neural Net



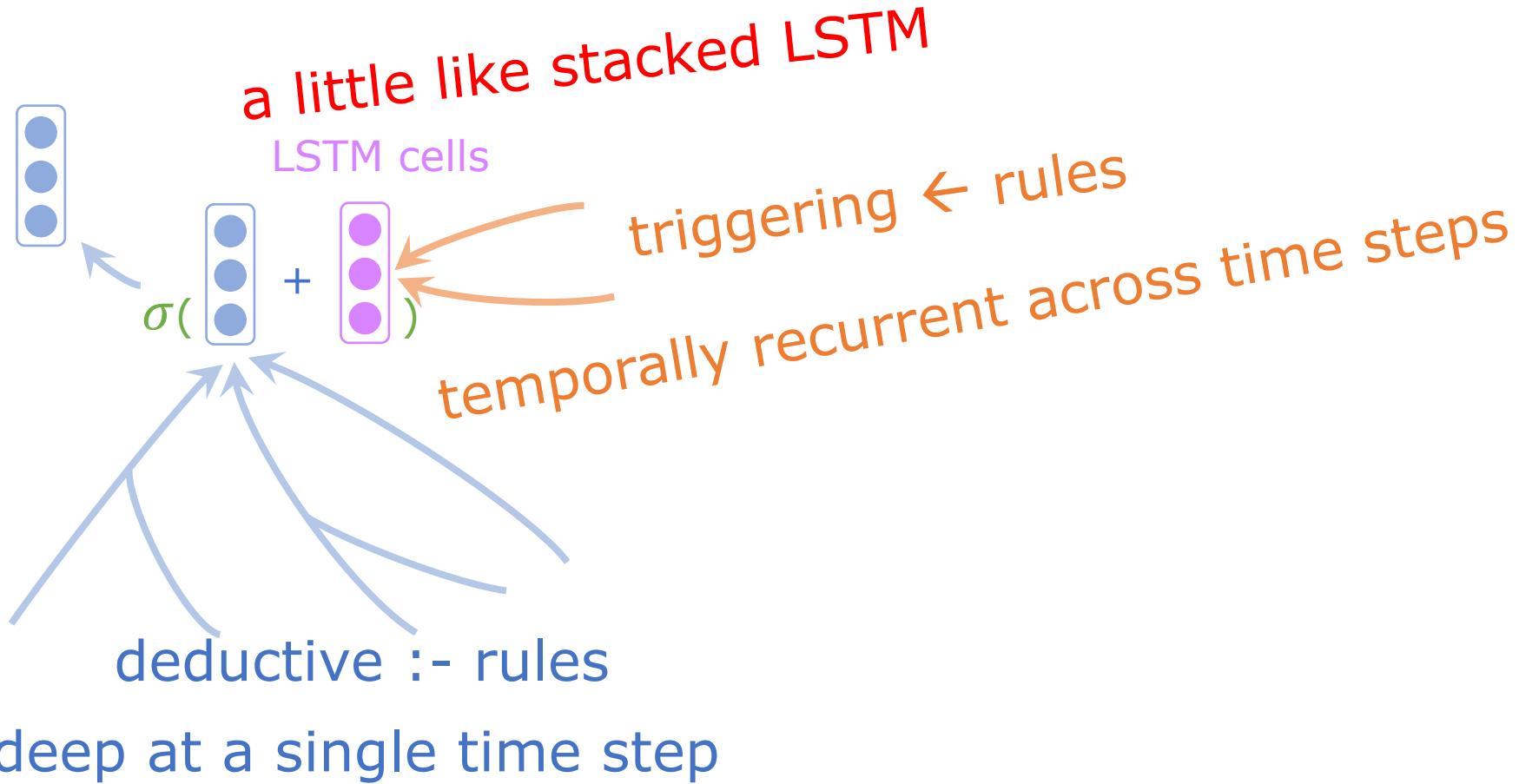
# Rules → Deep Recurrent Neural Net



# Rules → Deep Recurrent Neural Net



# Rules → Deep Recurrent Neural Net



<http://bburl/tpp-slides-p3>  
<http://bburl/tpp-lab-p3>

# Life Story of a Fact

# Life Story of a Fact

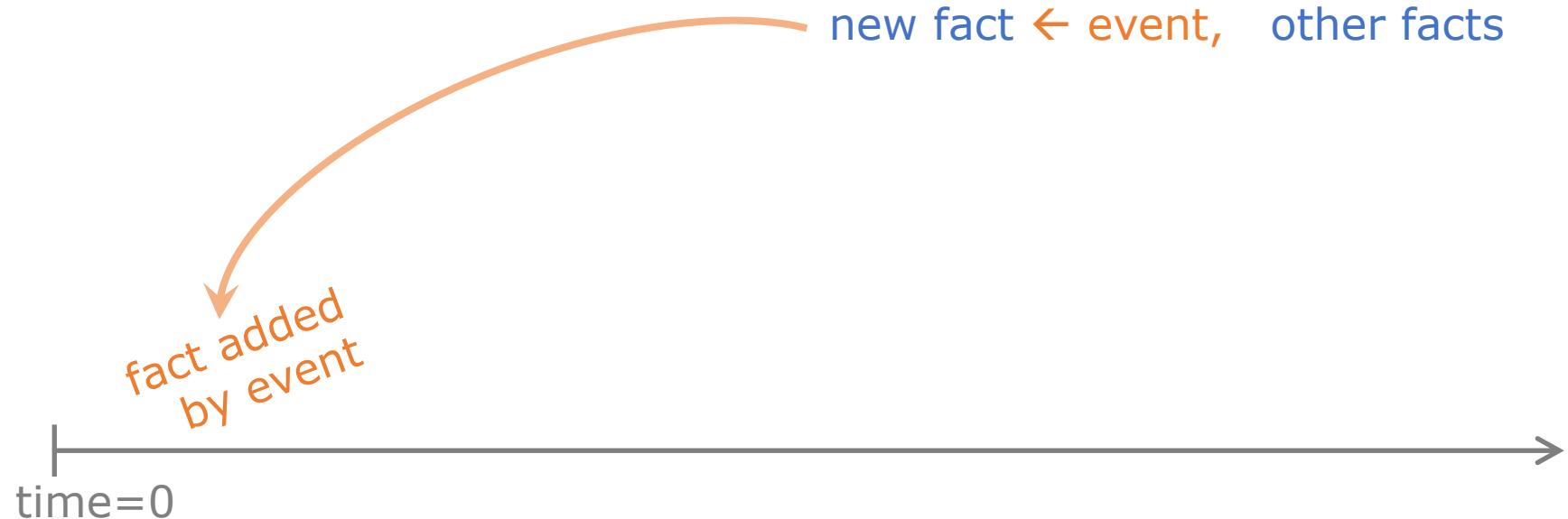


# Life Story of a Fact

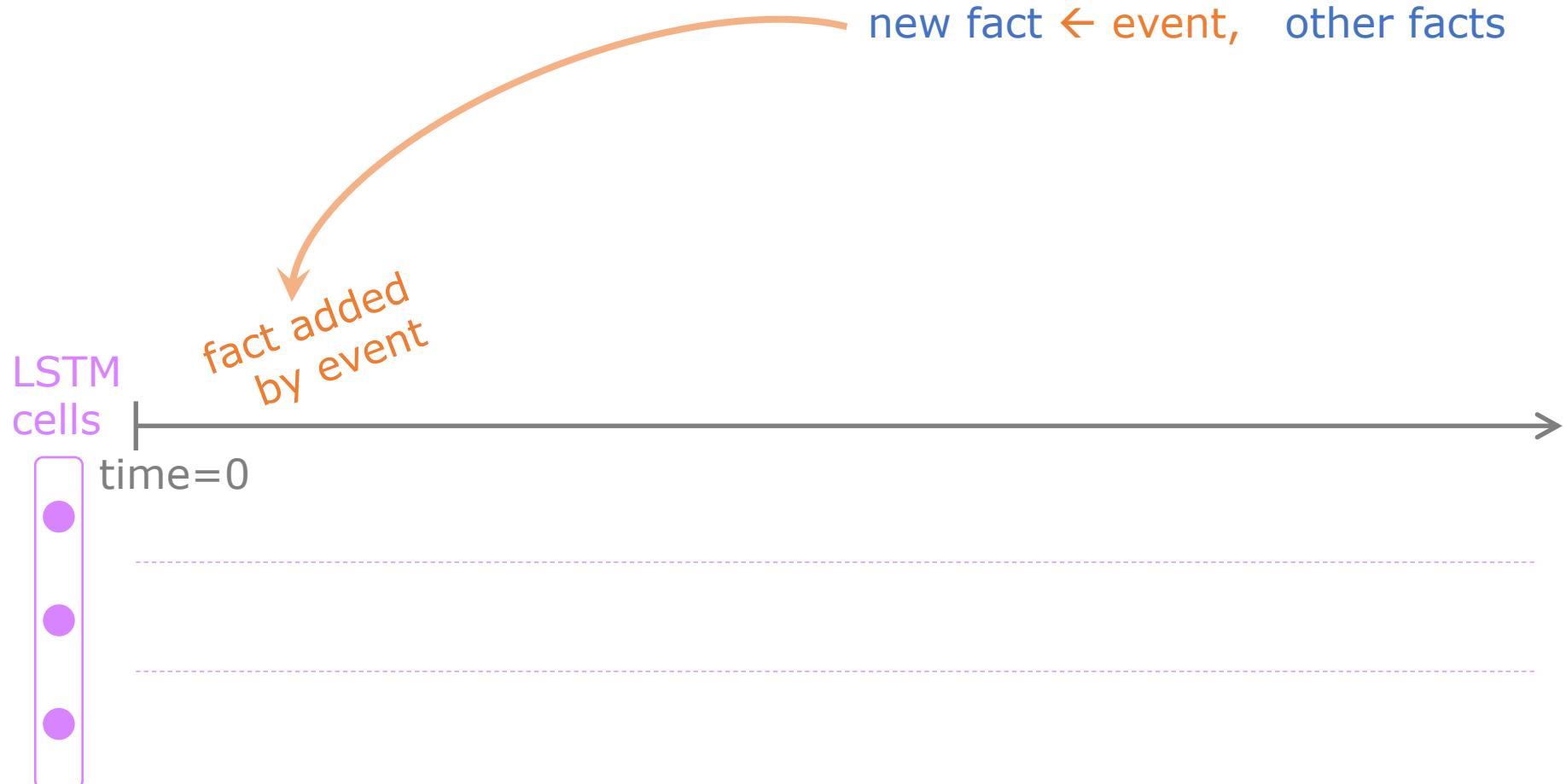
new fact  $\leftarrow$  event, other facts



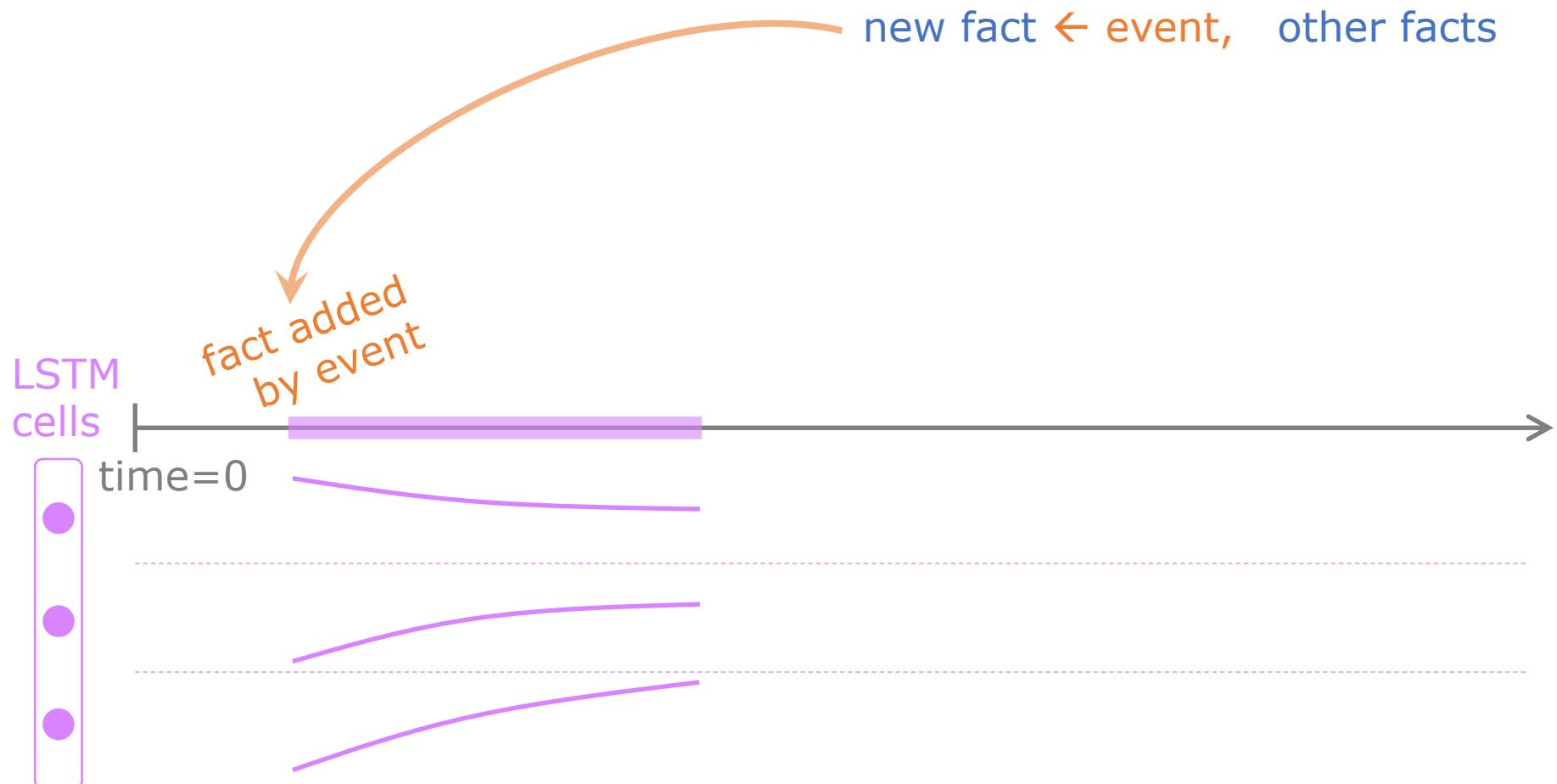
# Life Story of a Fact



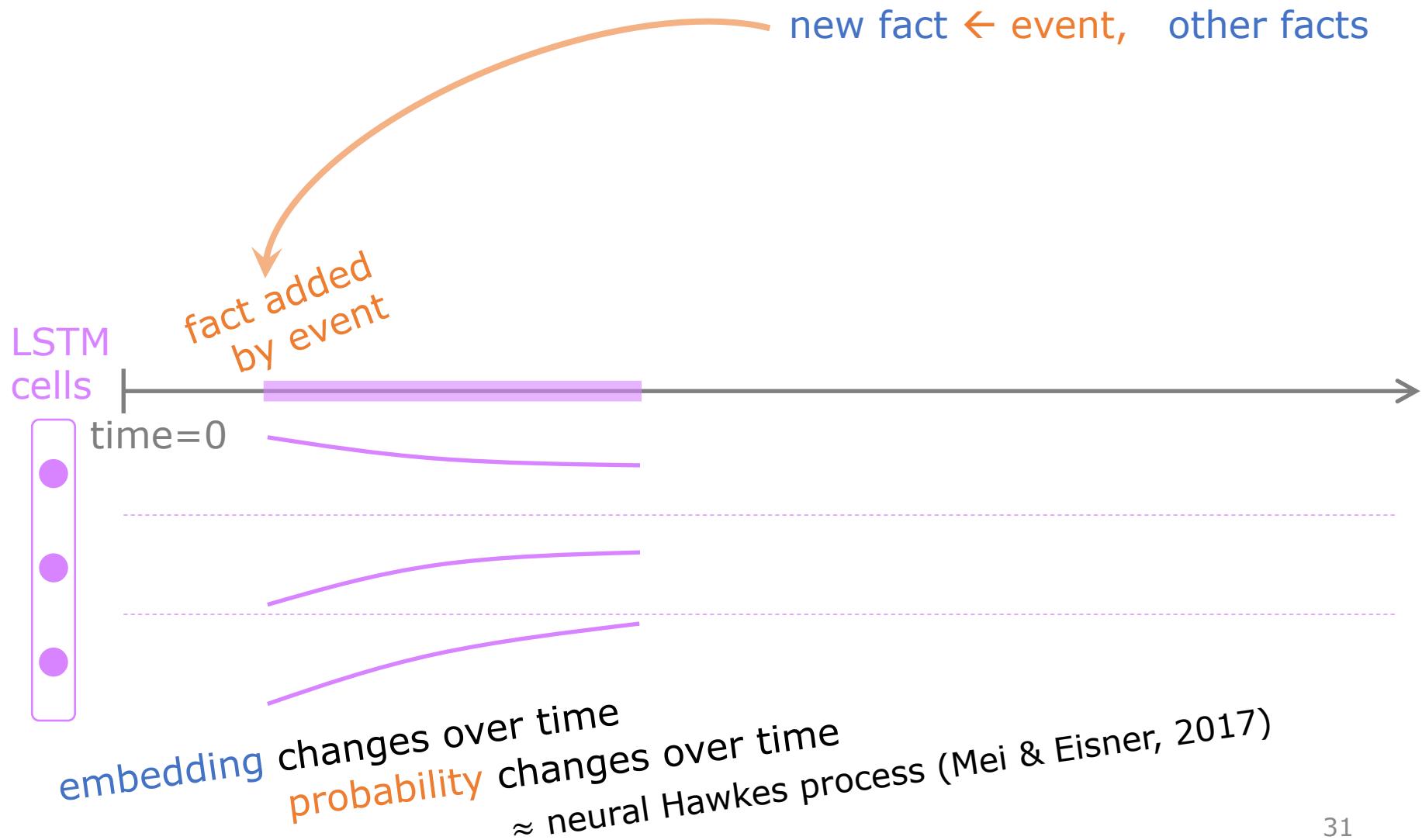
# Life Story of a Fact



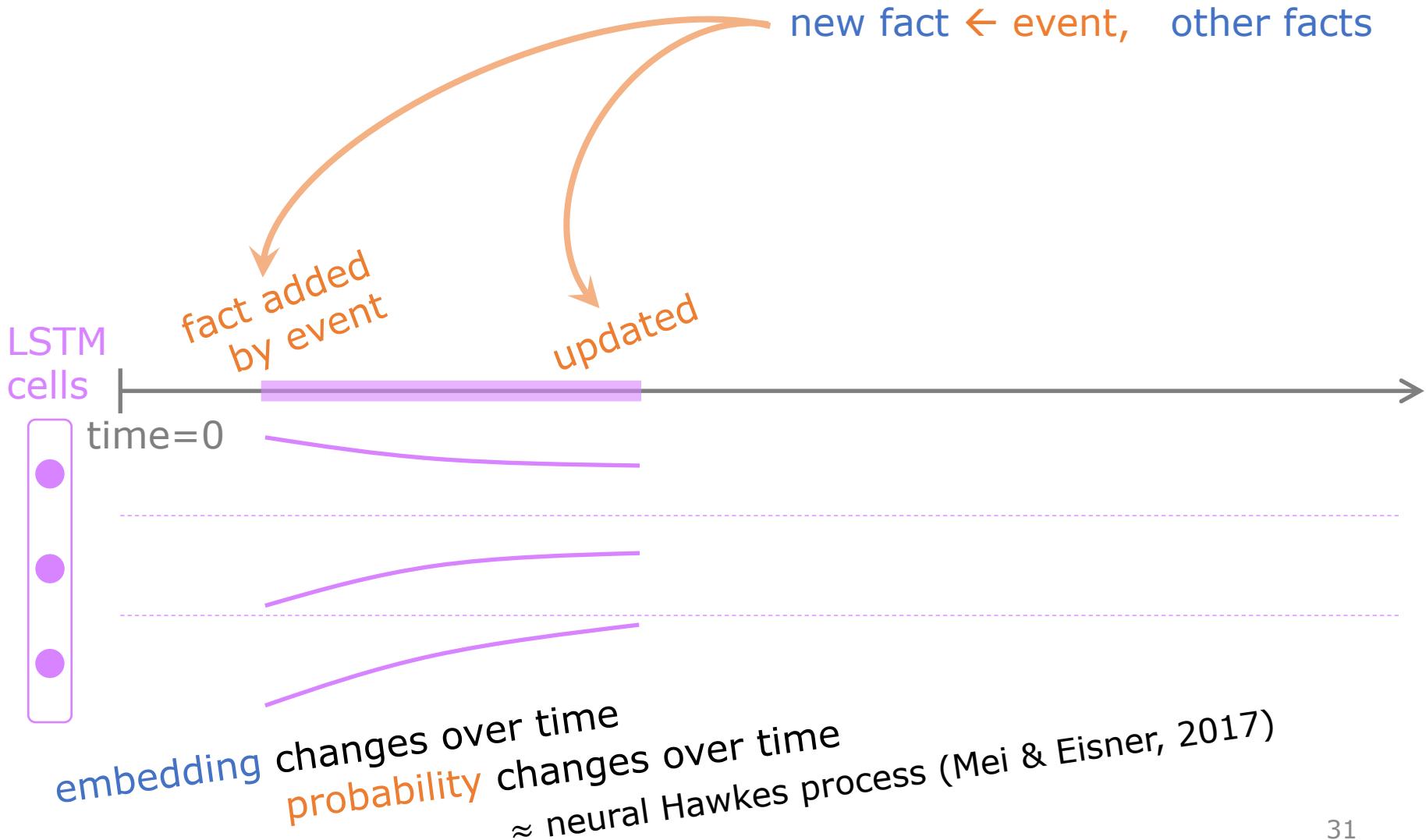
# Life Story of a Fact



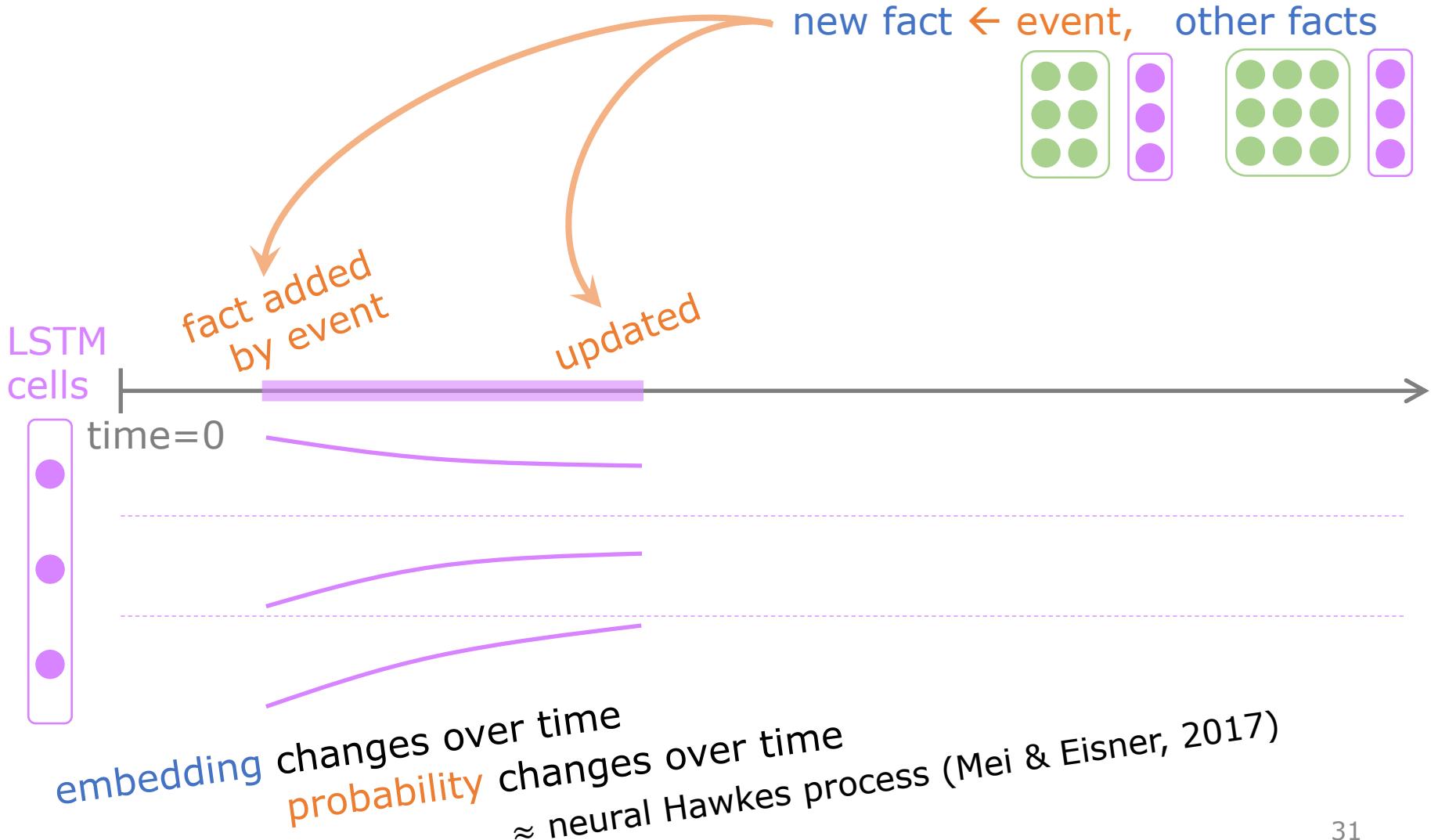
# Life Story of a Fact



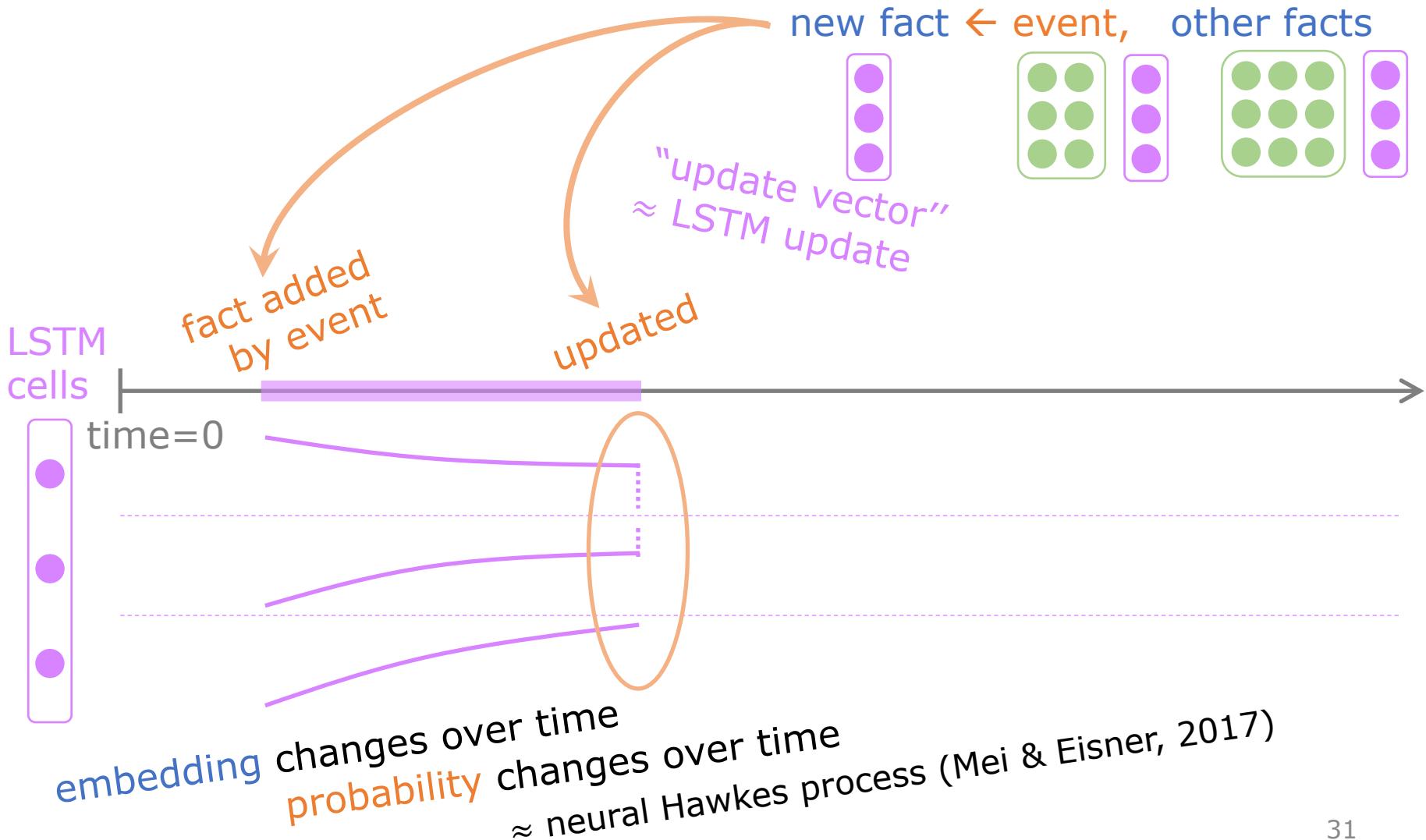
# Life Story of a Fact



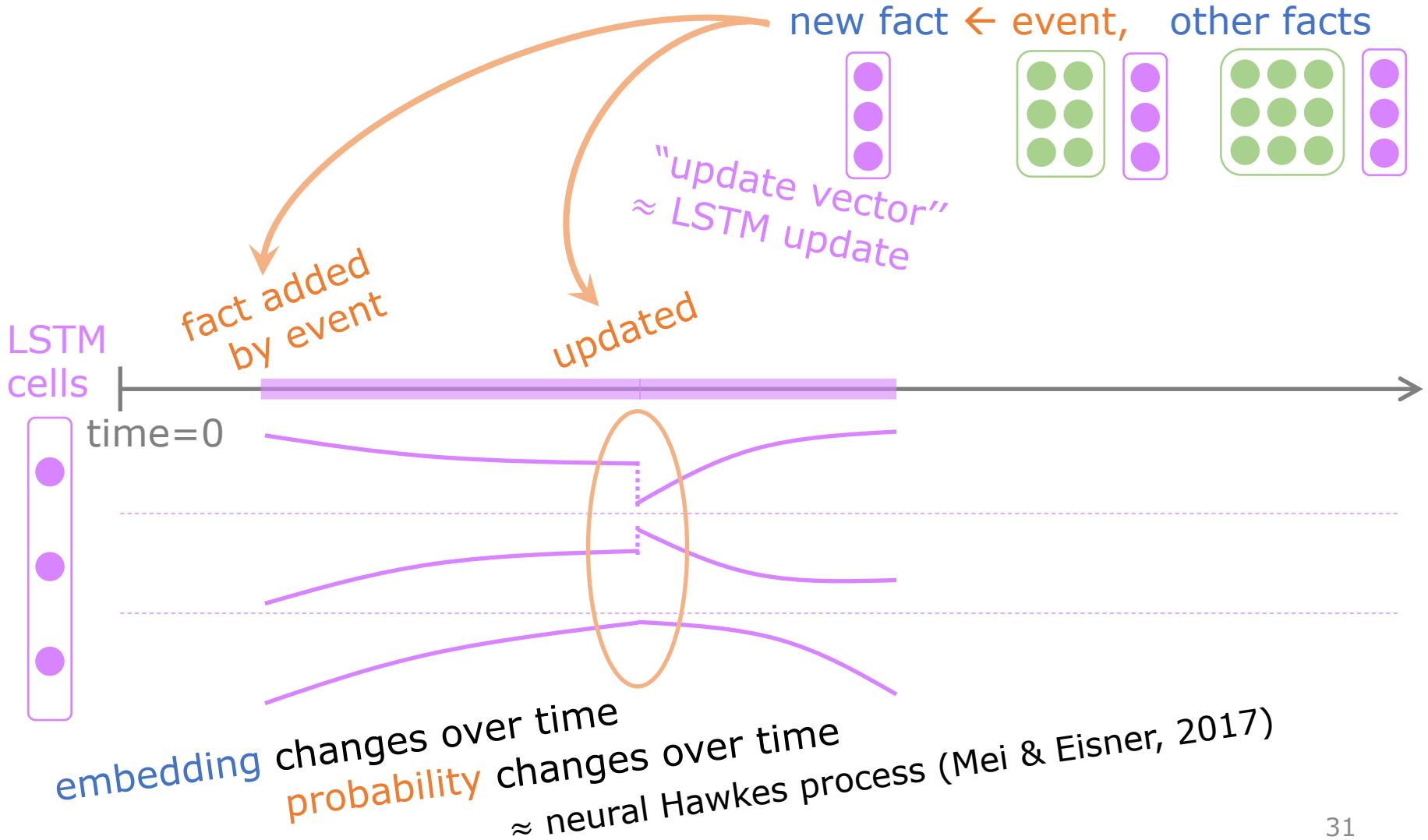
# Life Story of a Fact



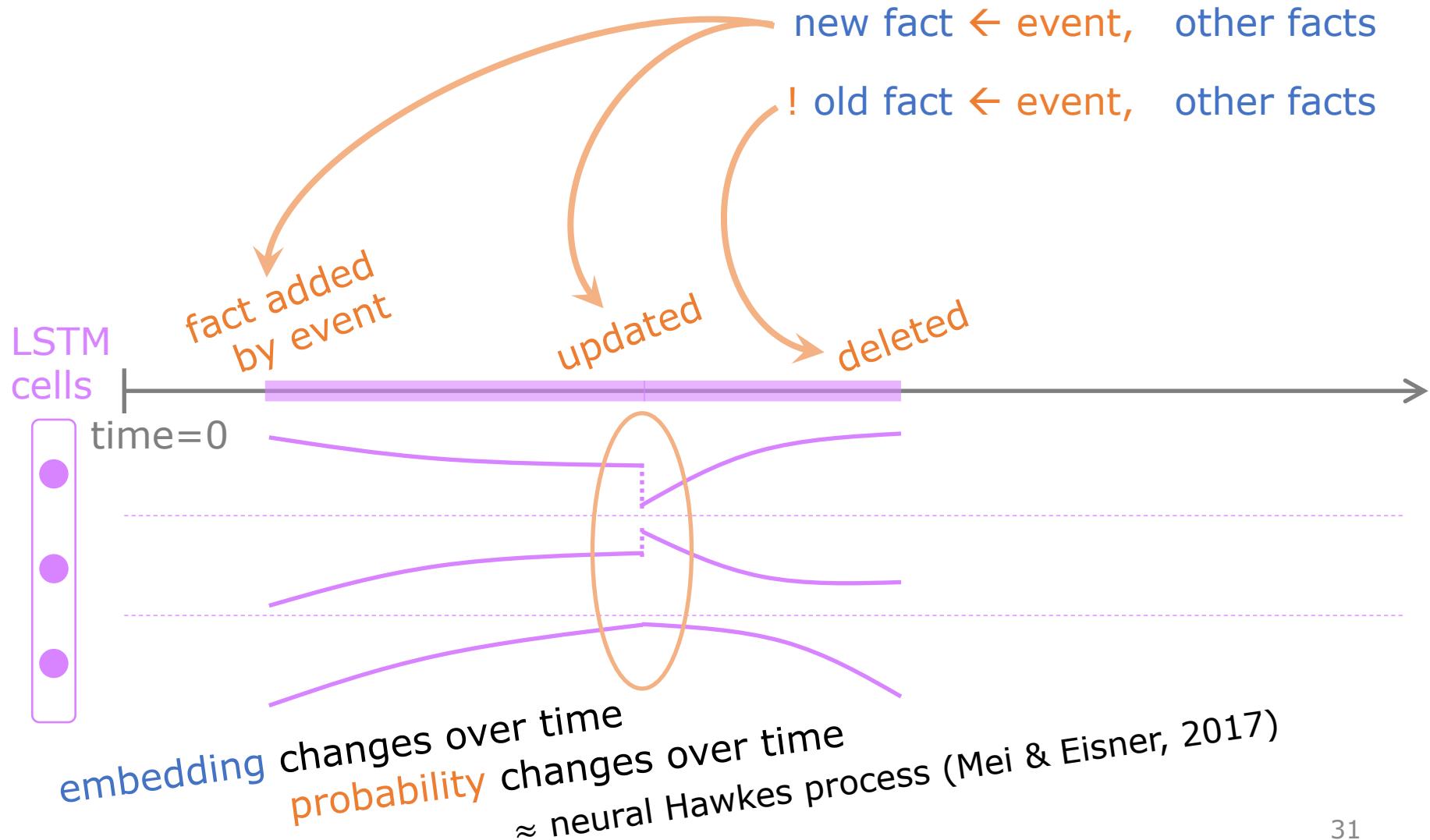
# Life Story of a Fact



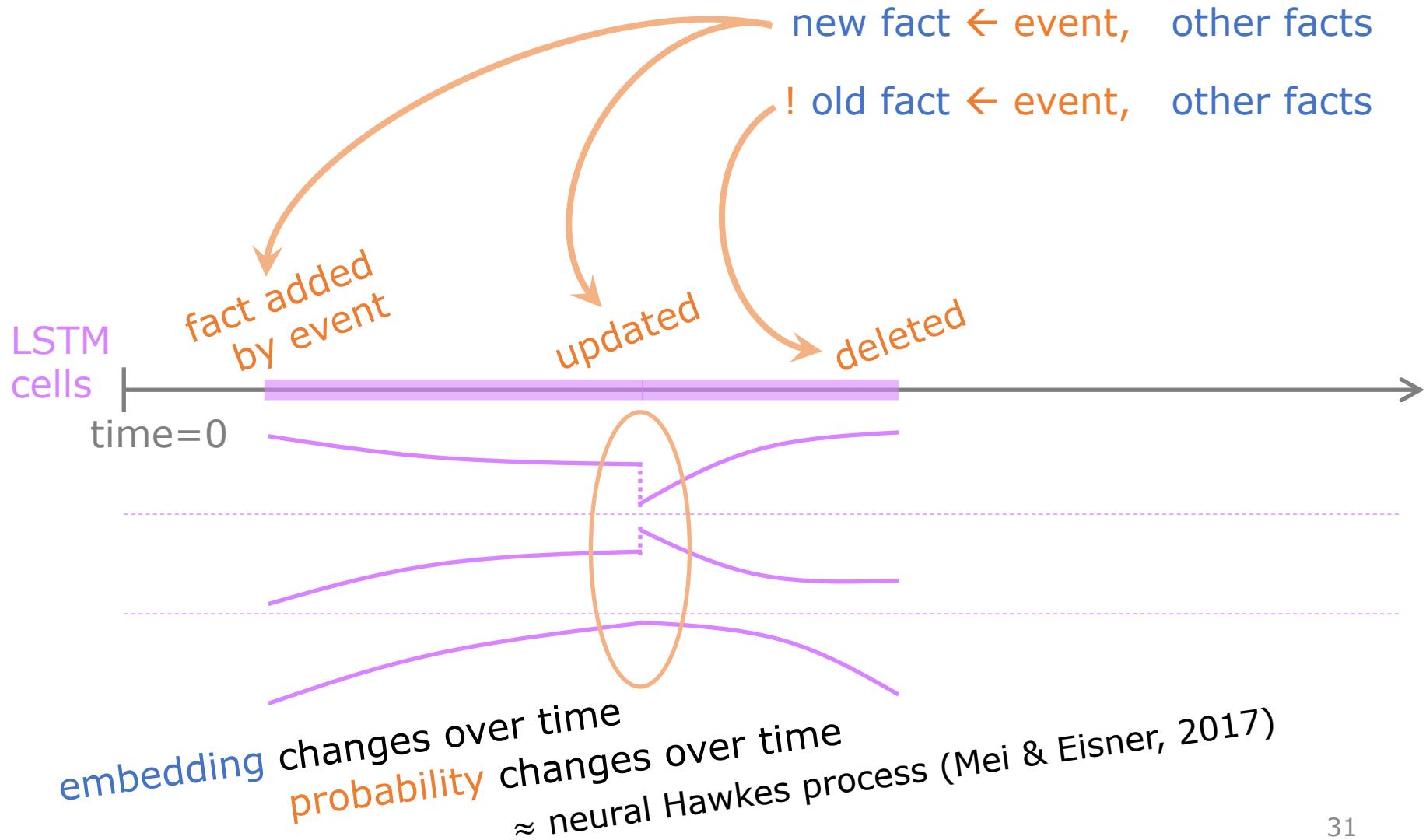
# Life Story of a Fact



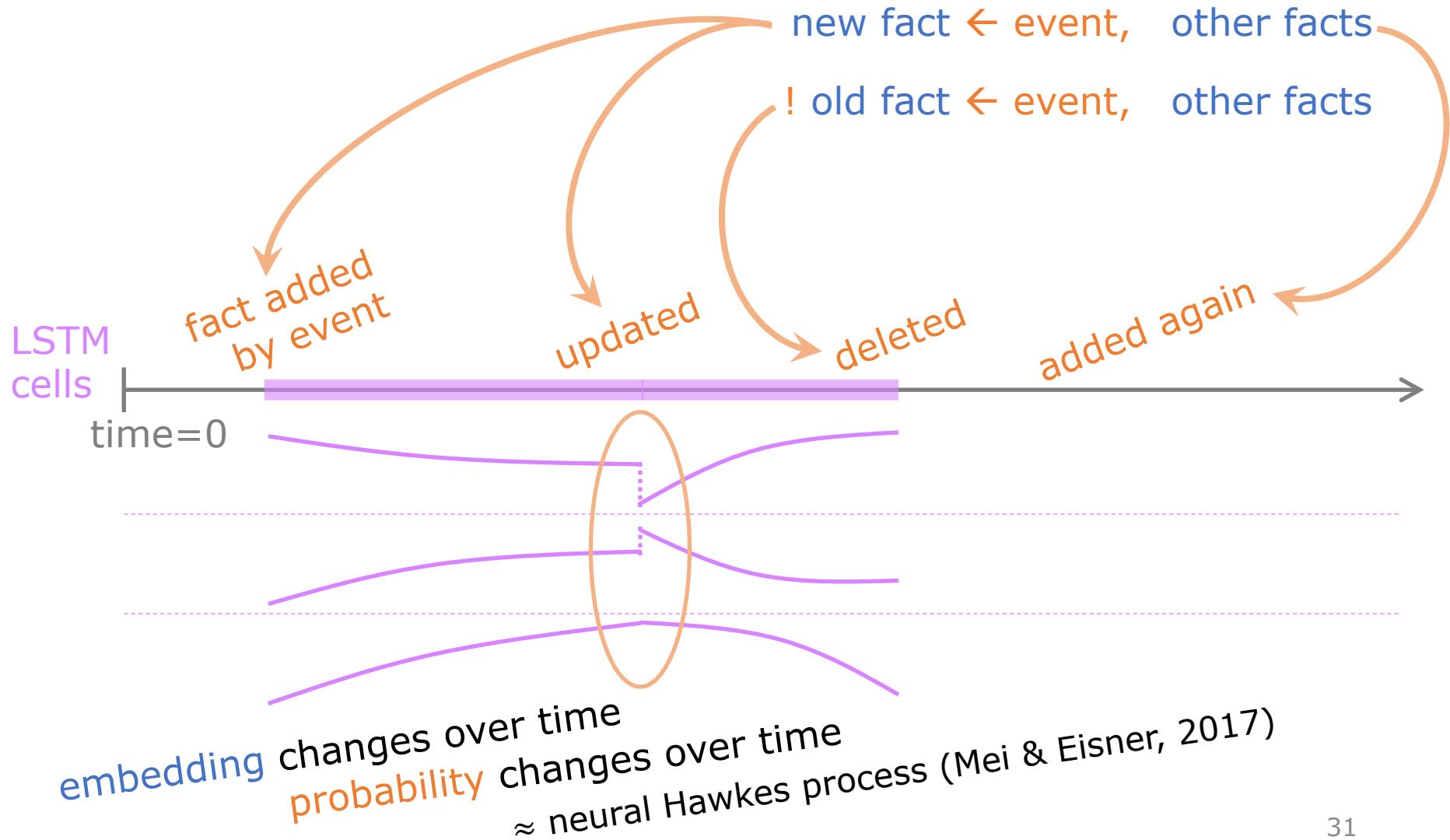
# Life Story of a Fact



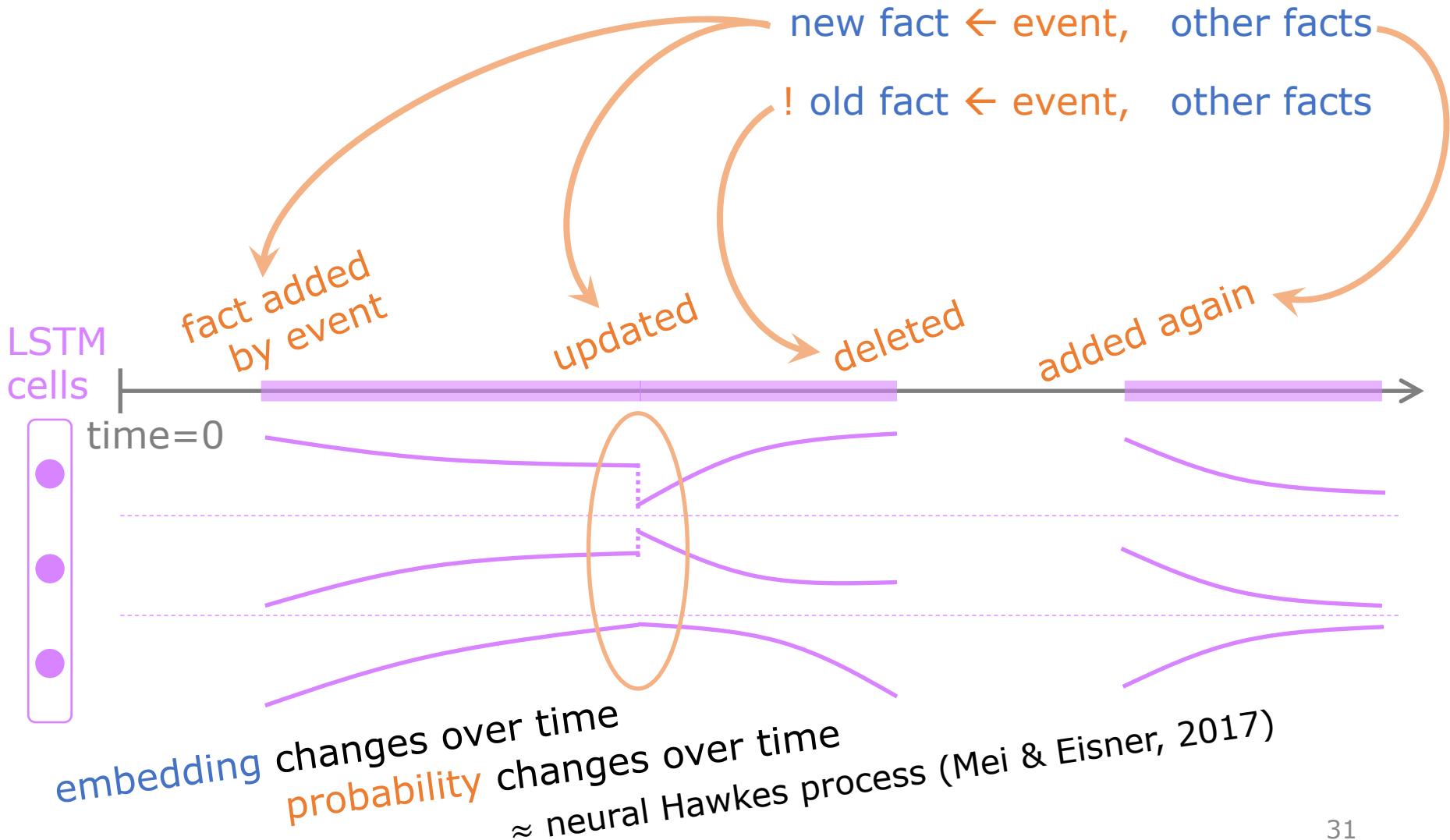
# Life Story of a Fact



# Life Story of a Fact



# Life Story of a Fact



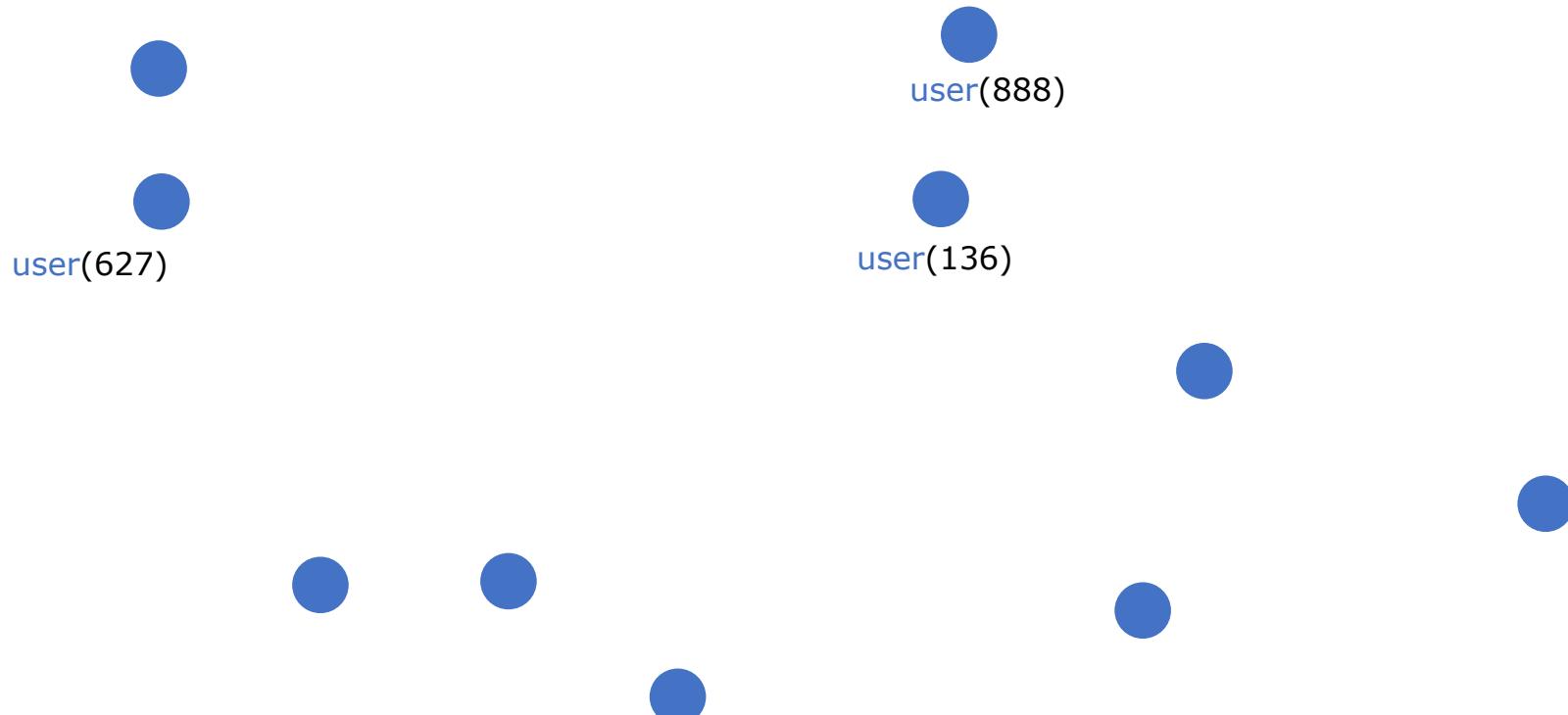
# Experiment: Users watch TV programs

collaborative filtering problem with timing  
who watches what and when?

# Experiment: Users watch TV programs

collaborative filtering problem with timing  
who watches what and when?

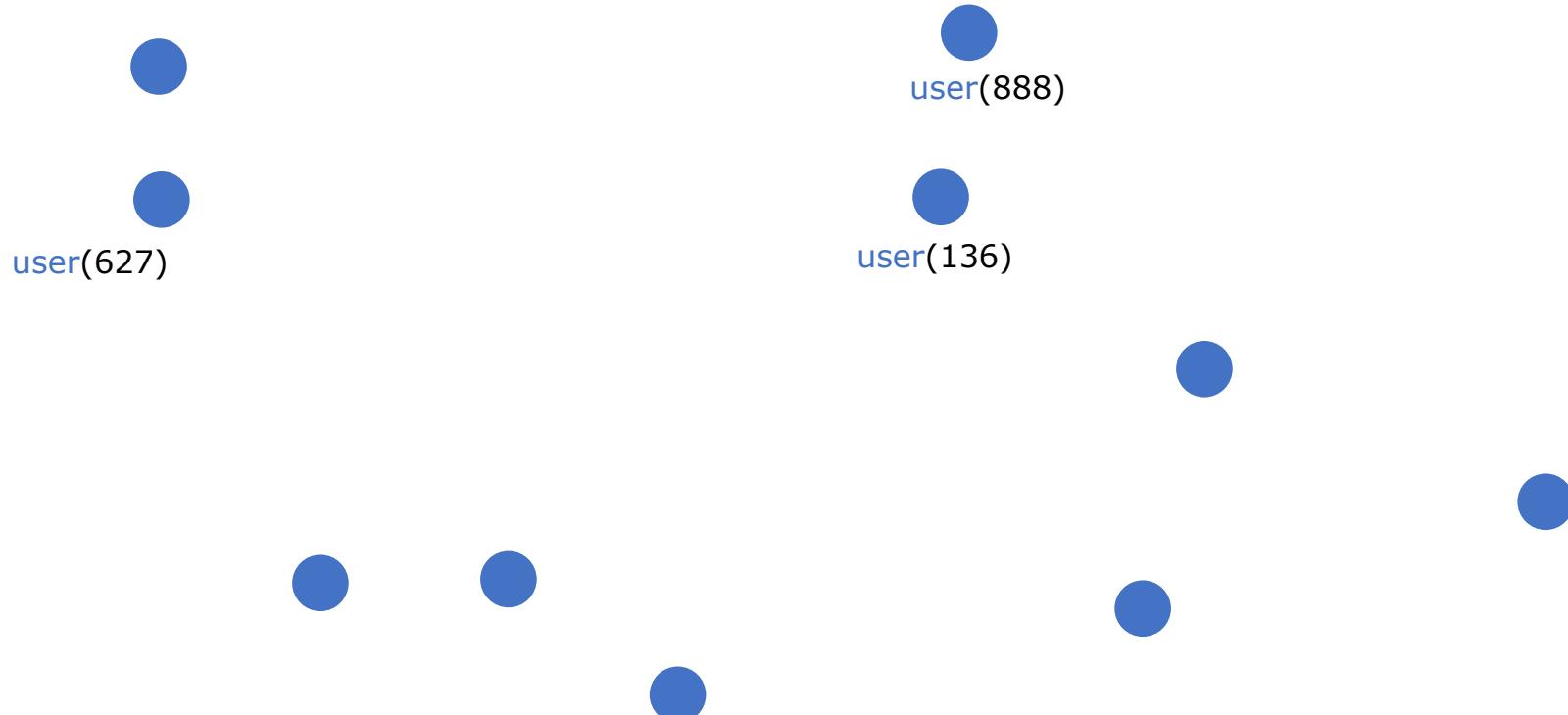
1000 users



# Experiment: Users watch TV programs

collaborative filtering problem with timing  
who watches what and when?

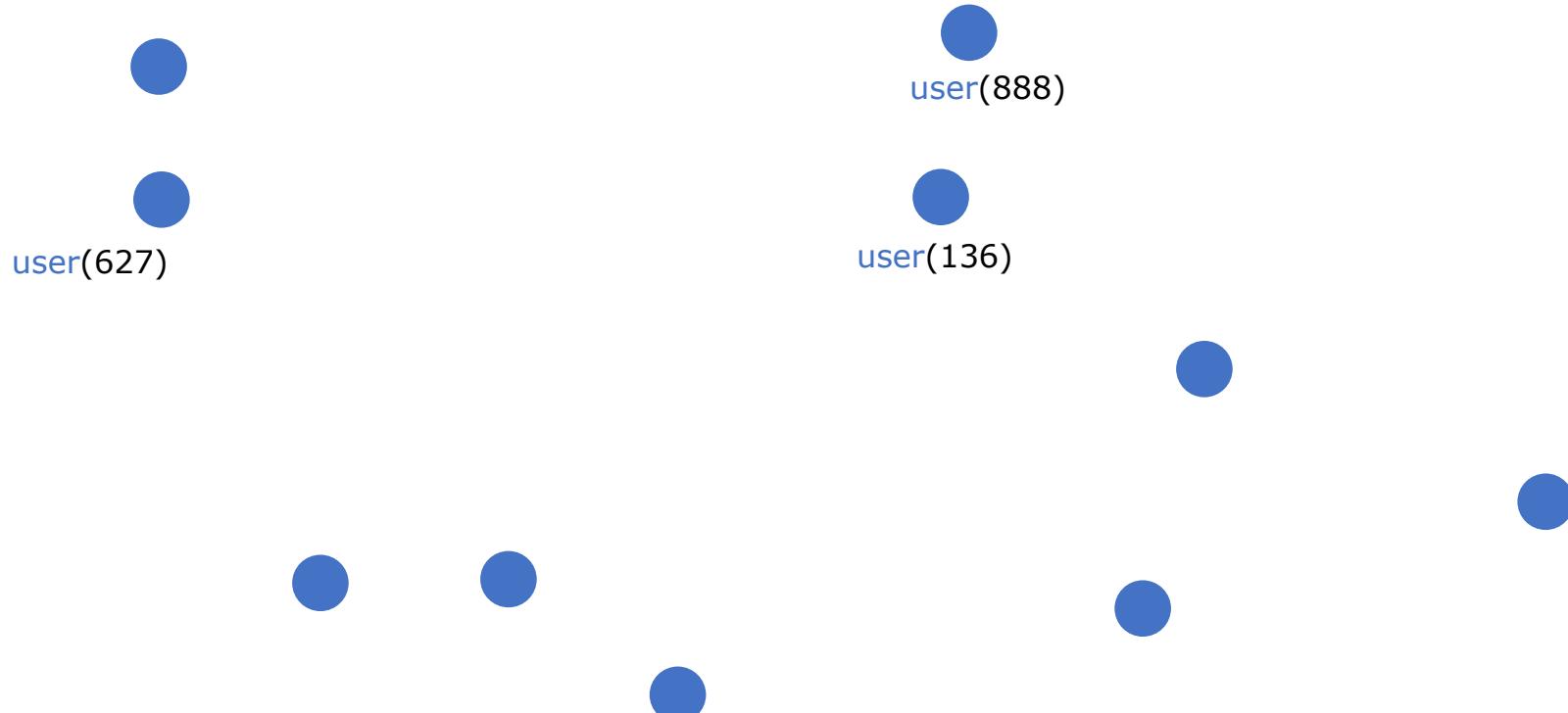
1000 users 49 TV programs to be released



# Experiment: Users watch TV programs

collaborative filtering problem with timing  
who watches what and when?

1000 users 49 TV programs to be released  
49000 possible watch events

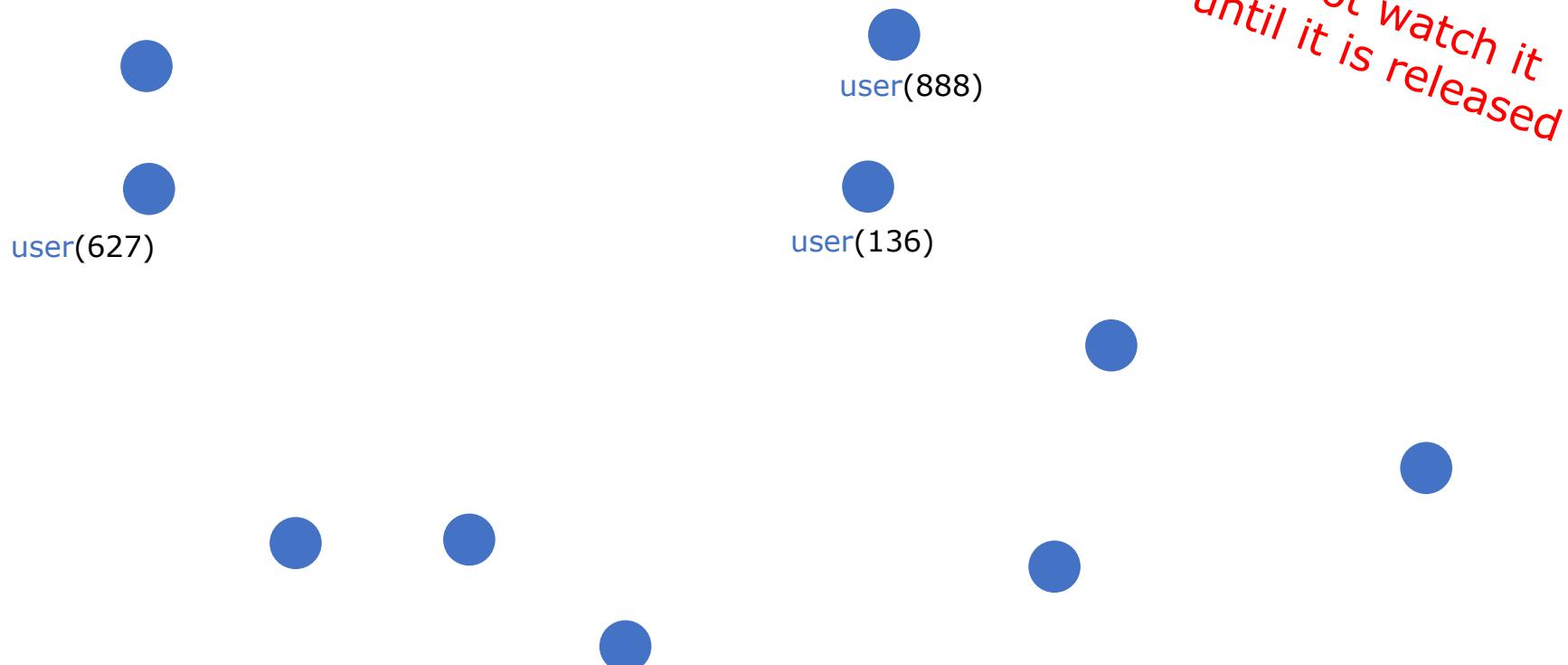


# Experiment: Users watch TV programs

collaborative filtering problem with timing  
who watches what and when?

1000 users 49 TV programs to be released

49000 possible watch events



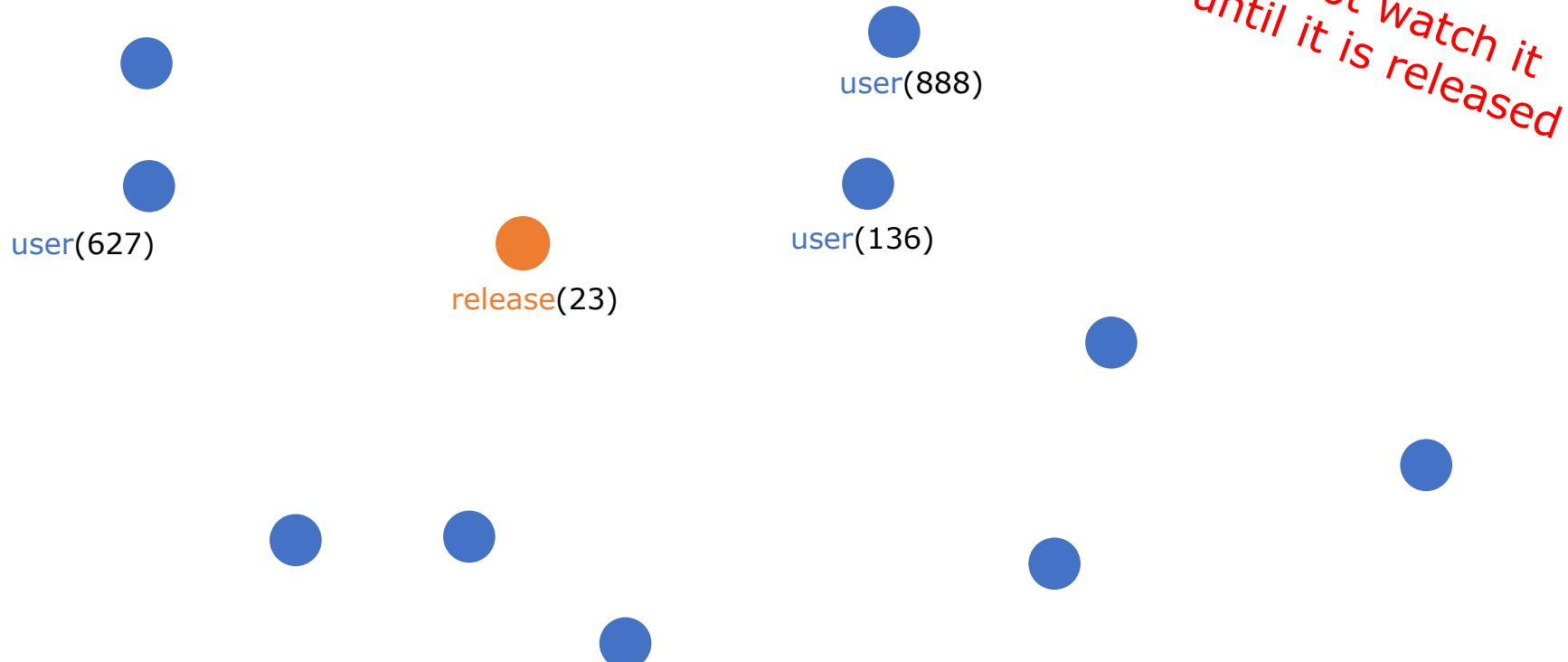
*can not watch it  
until it is released*

# Experiment: Users watch TV programs

collaborative filtering problem with timing  
who watches what and when?

1000 users 49 TV programs to be released

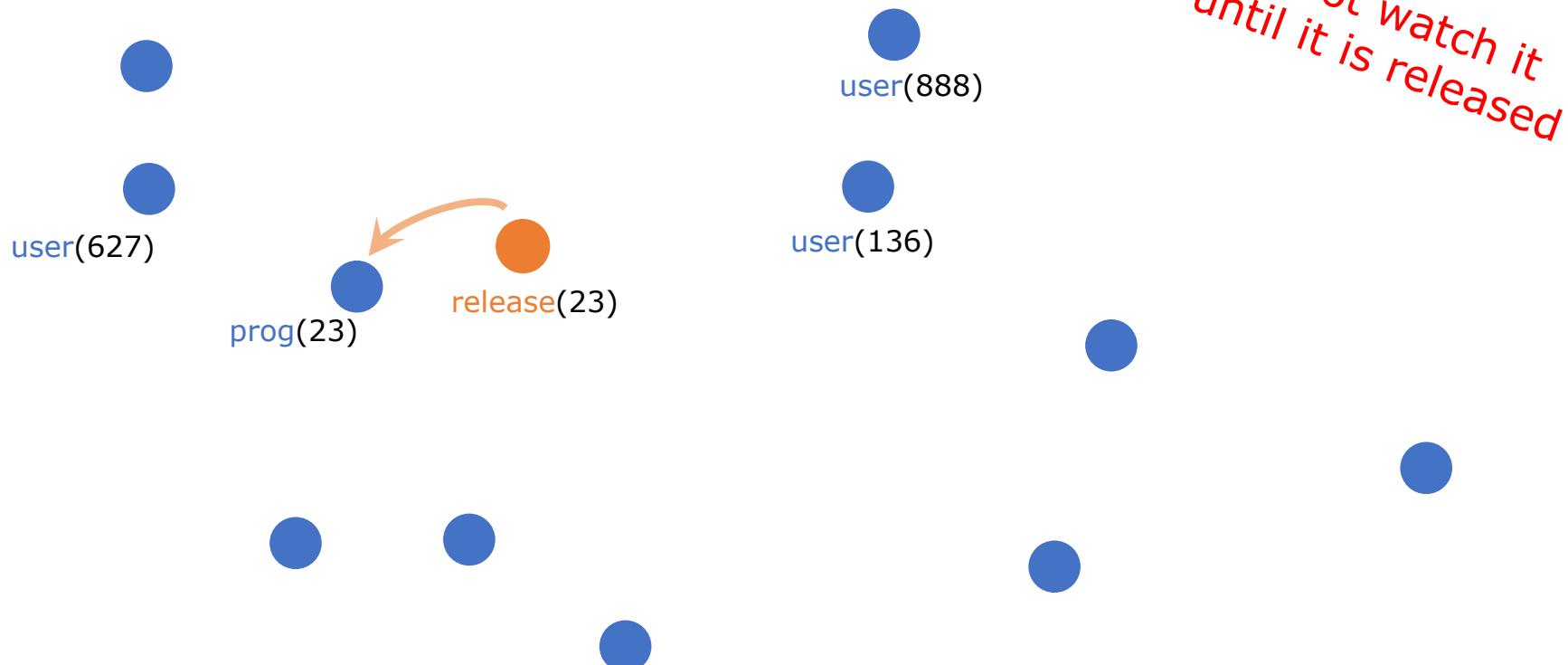
49000 possible watch events



# Experiment: Users watch TV programs

collaborative filtering problem with timing  
who watches what and when?

1000 users    49 TV programs to be released  
49000 possible watch events

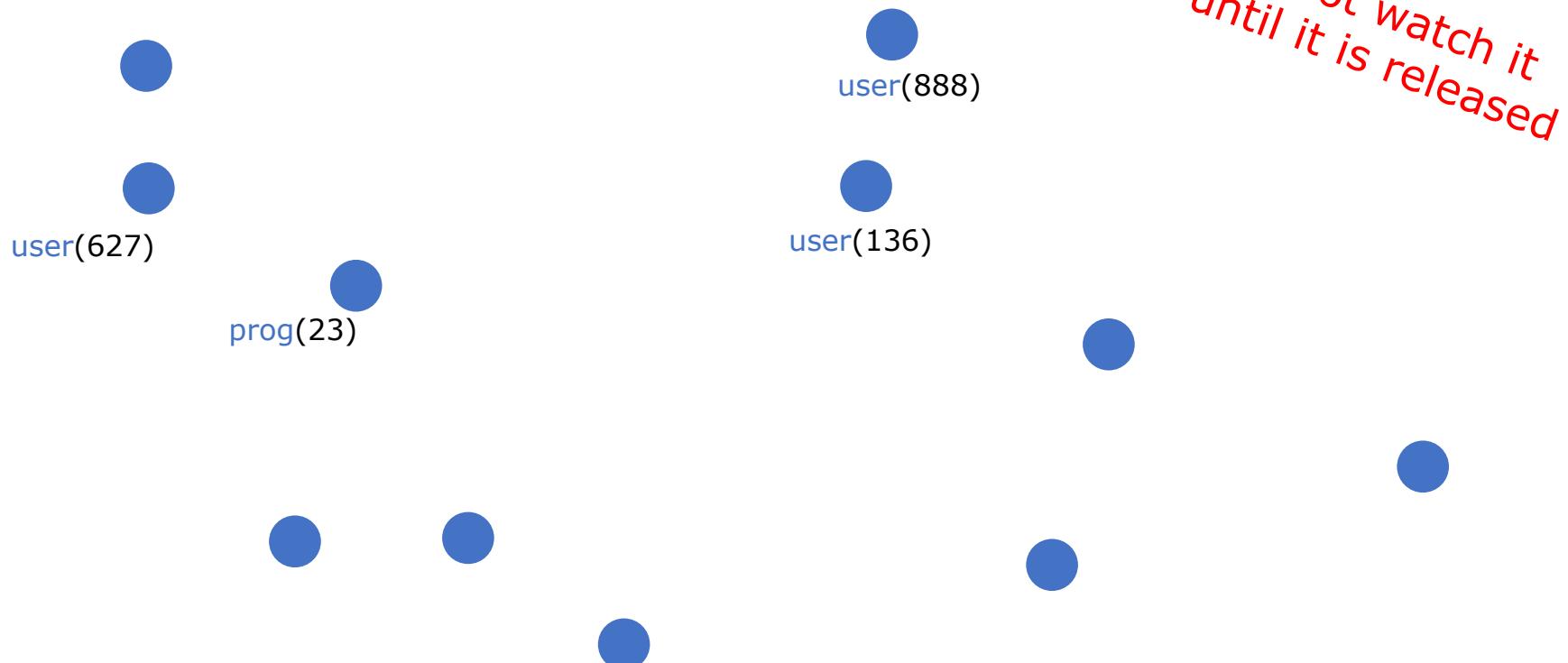


# Experiment: Users watch TV programs

collaborative filtering problem with timing  
who watches what and when?

1000 users 49 TV programs to be released

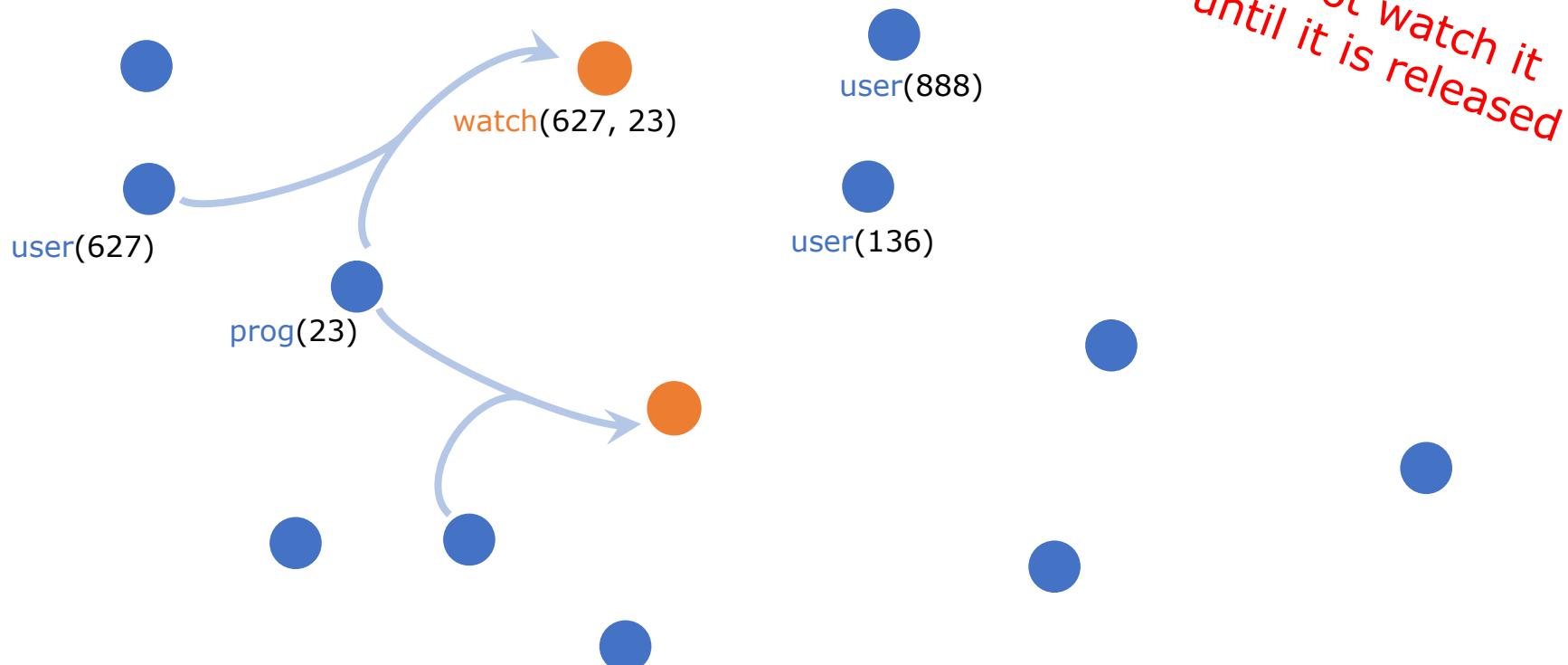
49000 possible watch events



# Experiment: Users watch TV programs

collaborative filtering problem with timing  
who watches what and when?

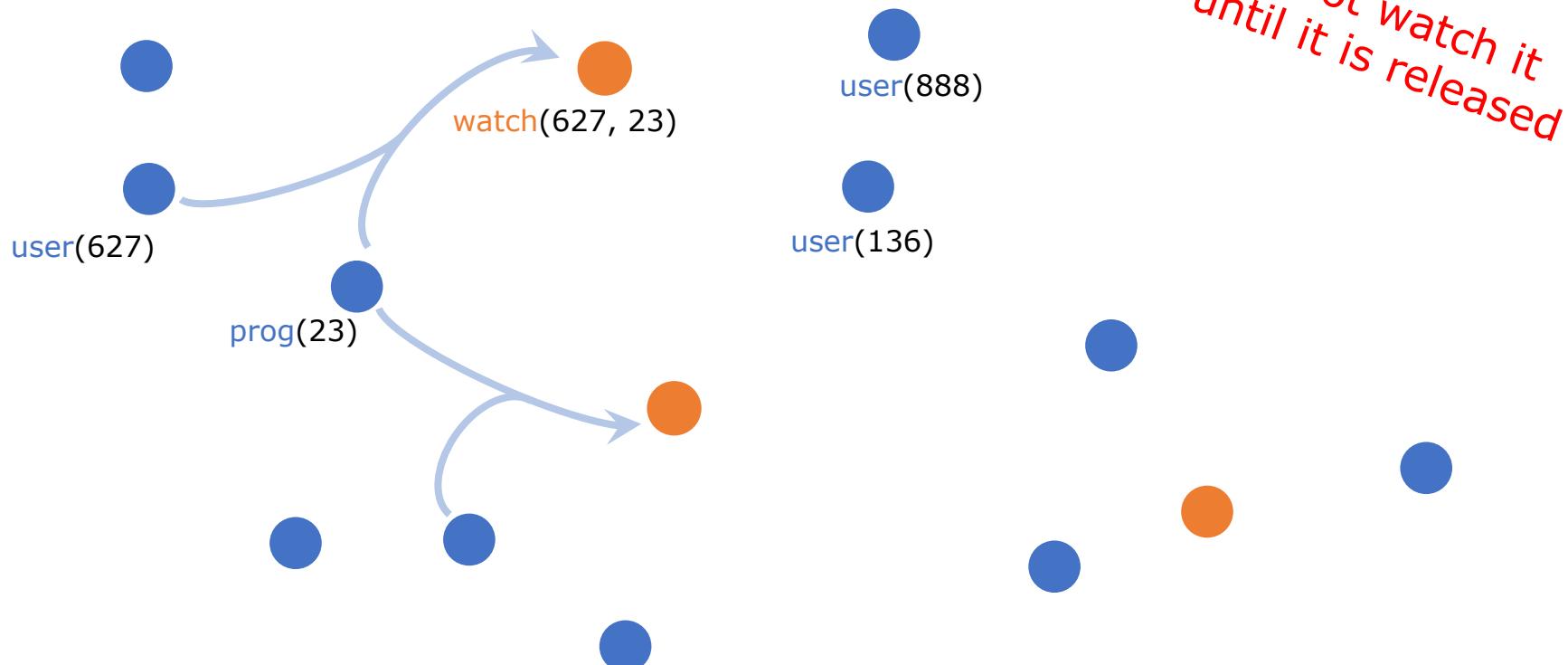
1000 users    49 TV programs to be released  
49000 possible watch events



# Experiment: Users watch TV programs

collaborative filtering problem with timing  
who watches what and when?

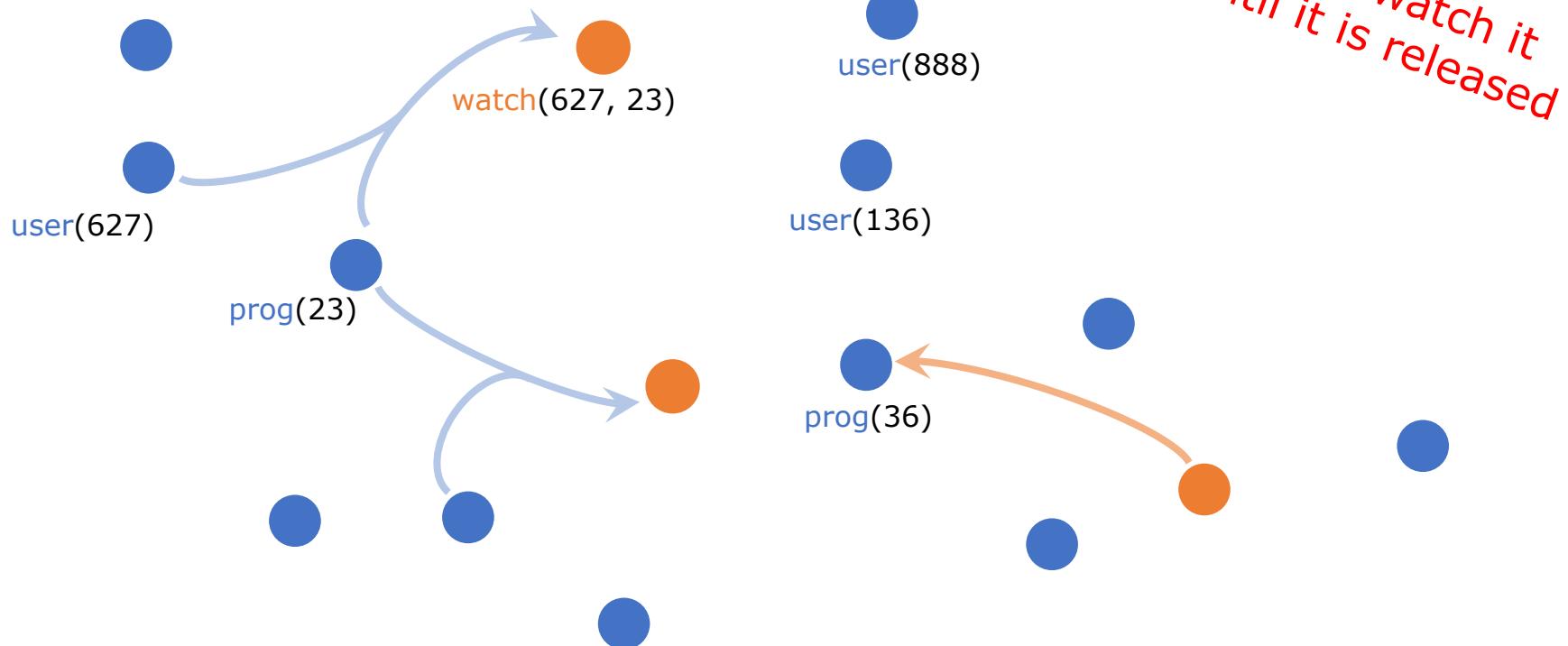
1000 users    49 TV programs to be released  
49000 possible watch events



# Experiment: Users watch TV programs

collaborative filtering problem with timing  
who watches what and when?

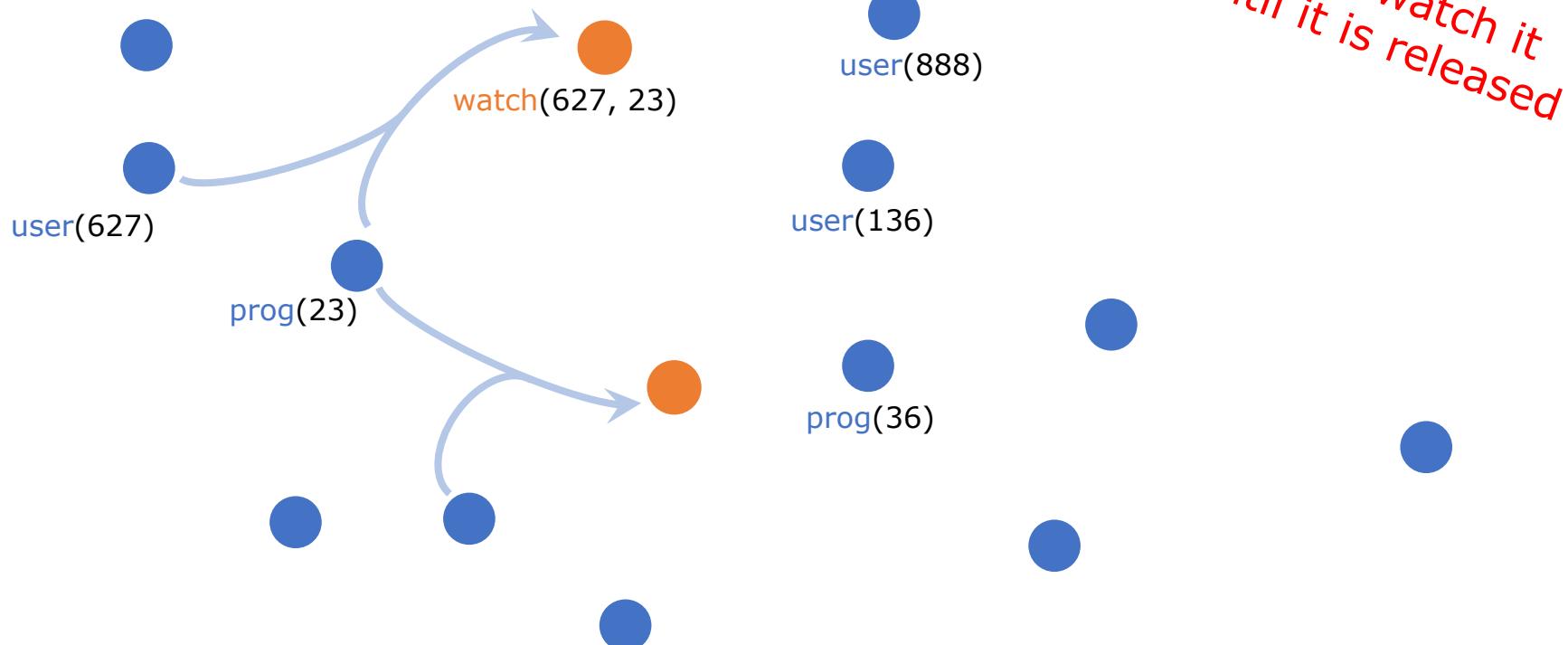
1000 users    49 TV programs to be released  
49000 possible watch events



# Experiment: Users watch TV programs

collaborative filtering problem with timing  
who watches what and when?

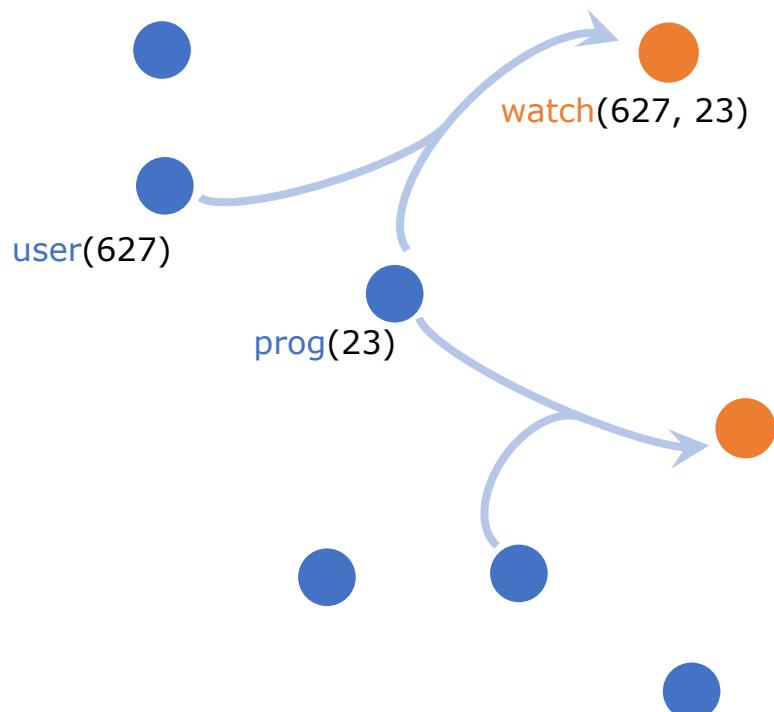
1000 users    49 TV programs to be released  
49000 possible watch events



# Experiment: Users watch TV programs

collaborative filtering problem with timing  
who watches what and when?

1000 users    49 TV programs to be released  
49000 possible watch events



<http://bburl/tpp-slides-p3>

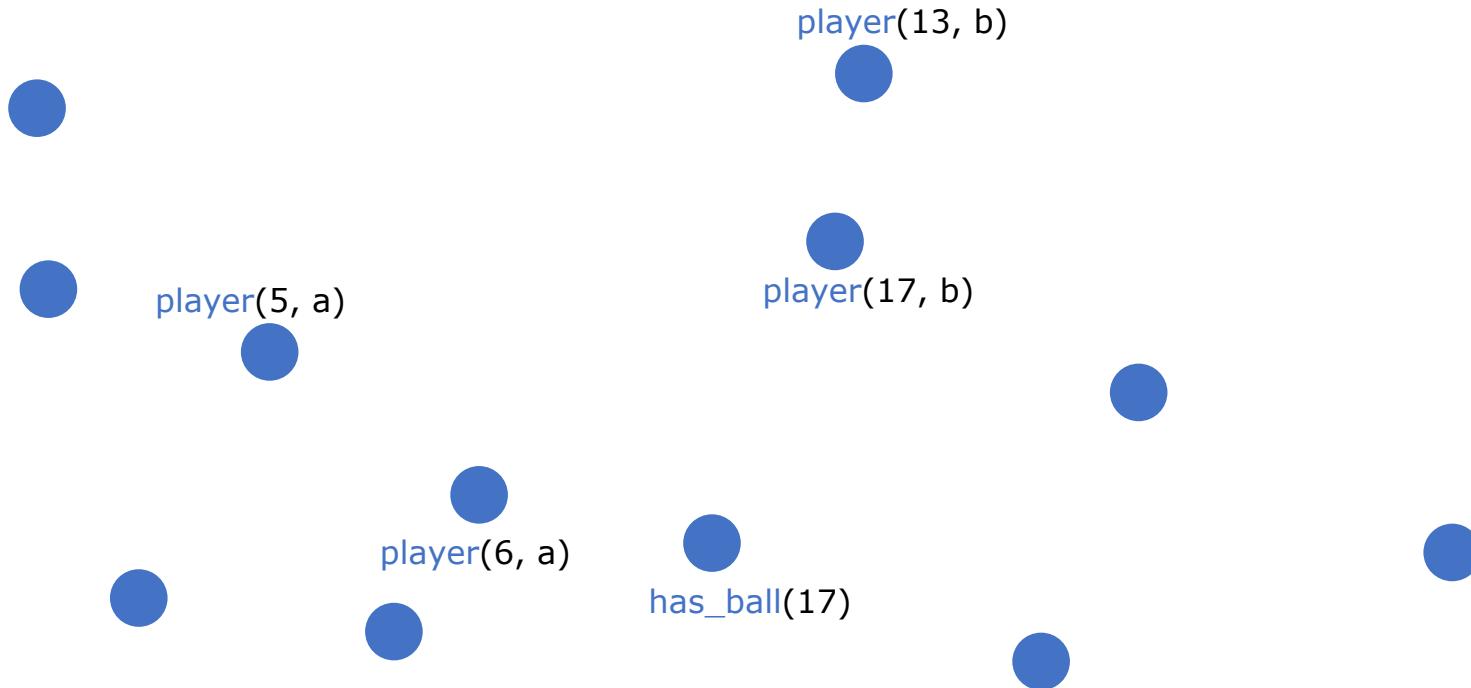
<http://bburl/tpp-lab-p3>

# Experiment: Robots Kick/Pass/Steal

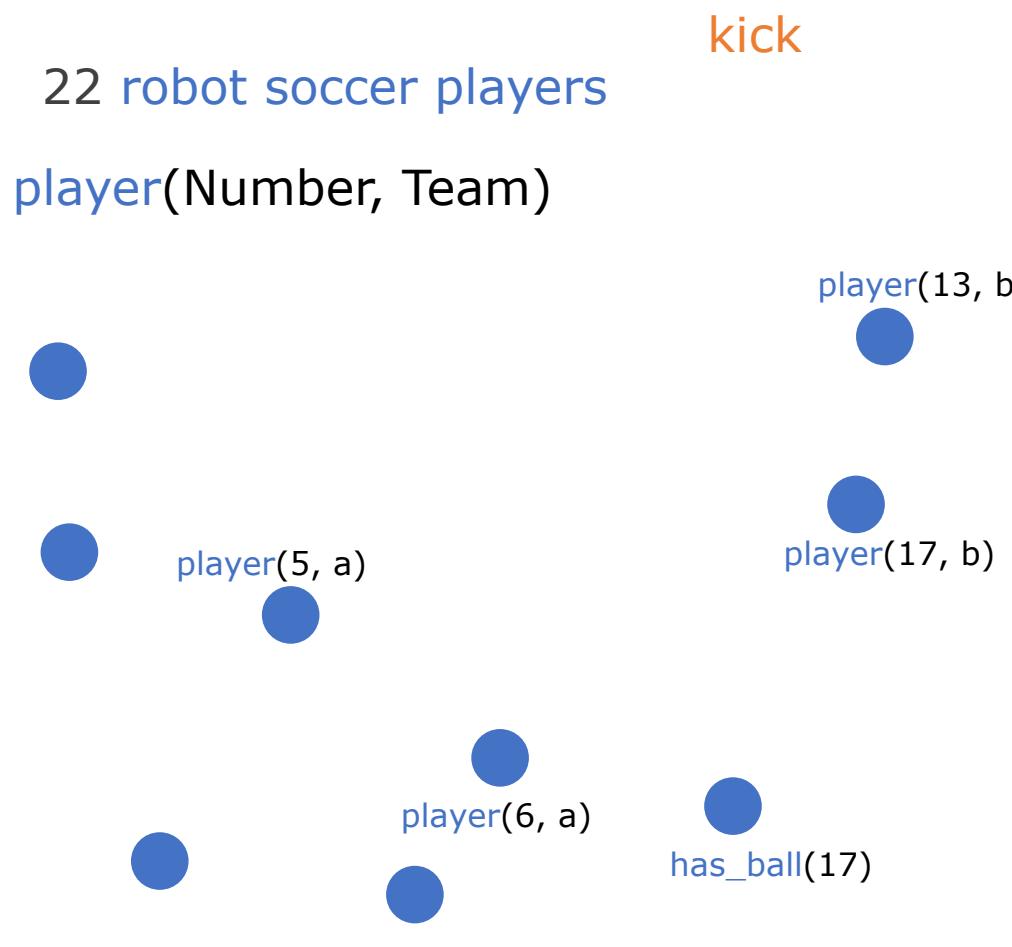
# Experiment: Robots Kick/Pass/Steal

22 robot soccer players

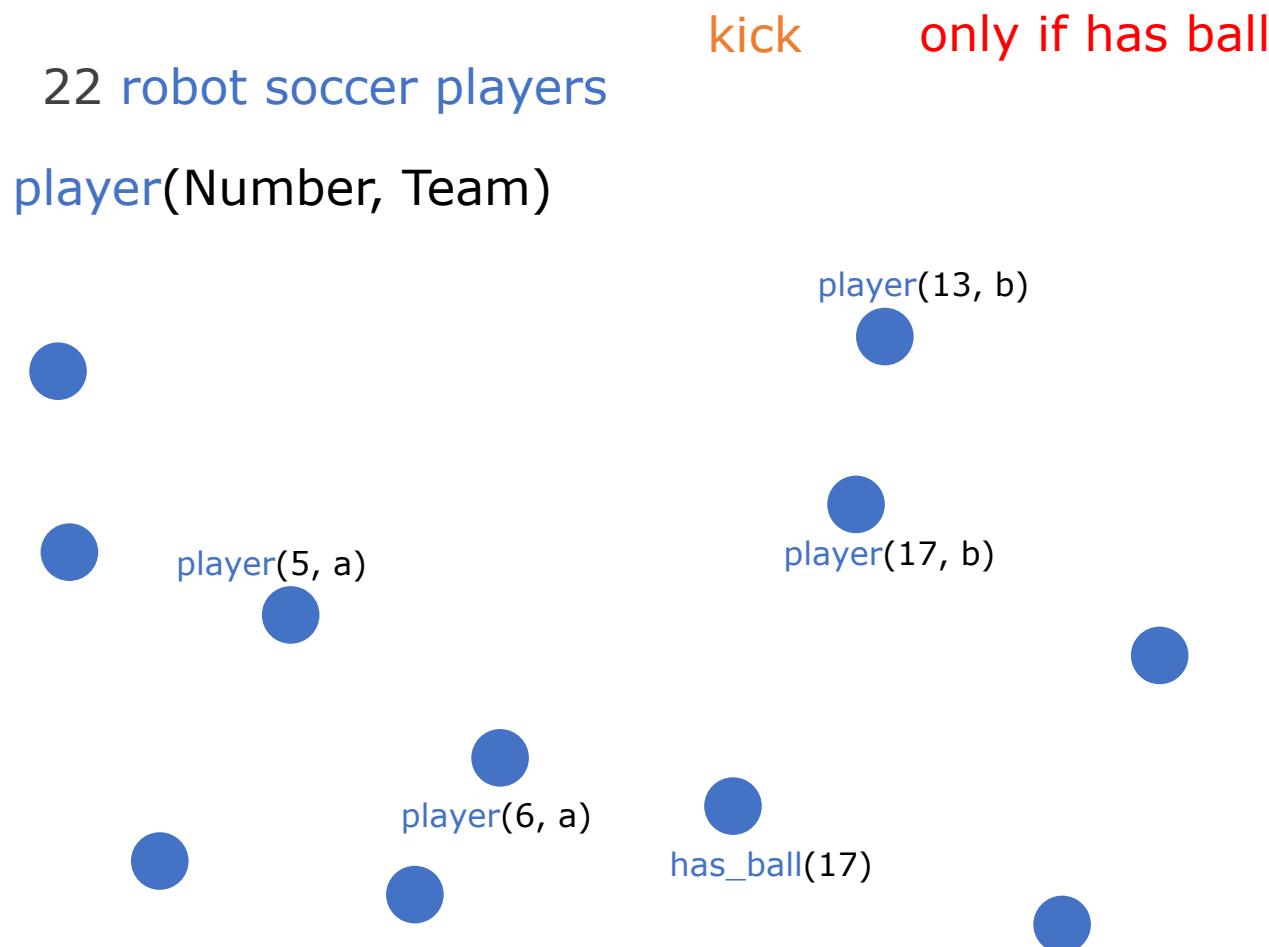
$\text{player}(\text{Number}, \text{Team})$



# Experiment: Robots Kick/Pass/Steal



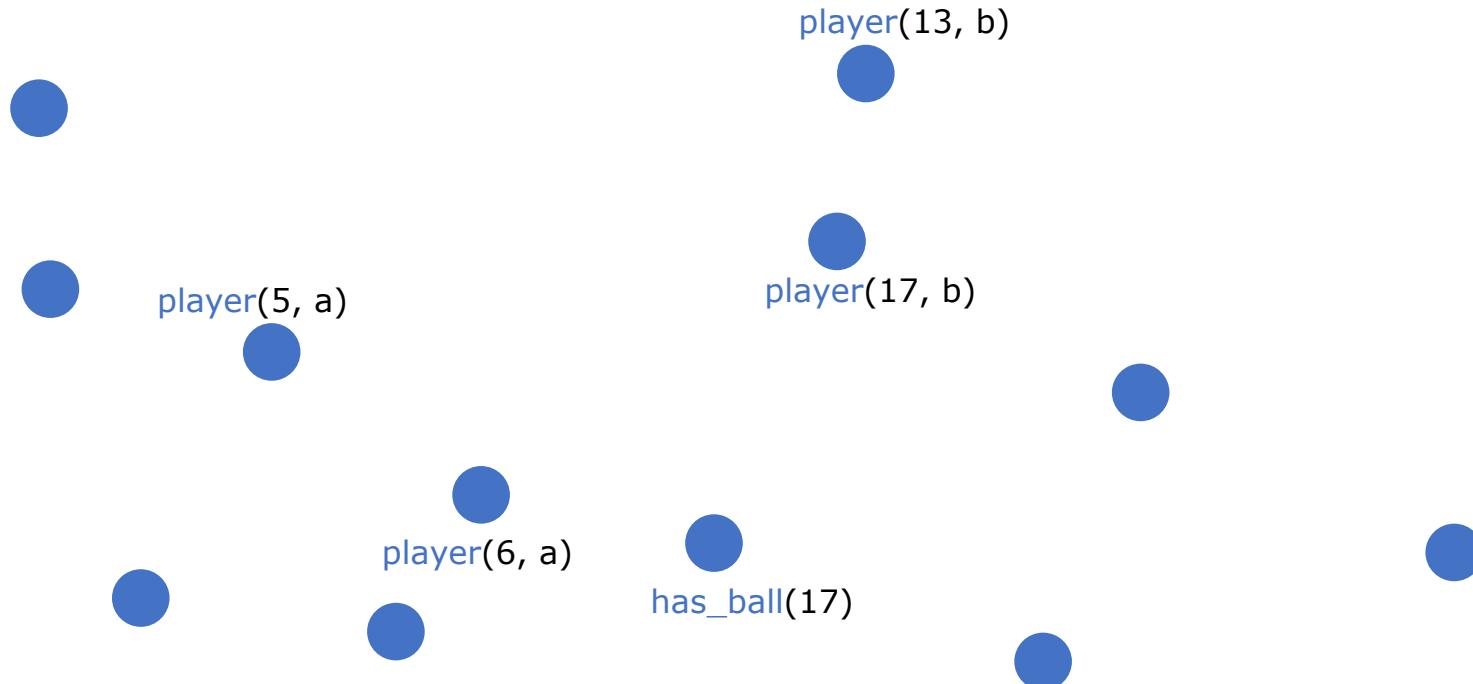
# Experiment: Robots Kick/Pass/Steal



# Experiment: Robots Kick/Pass/Steal

22 robot soccer players  
player(Number, Team)

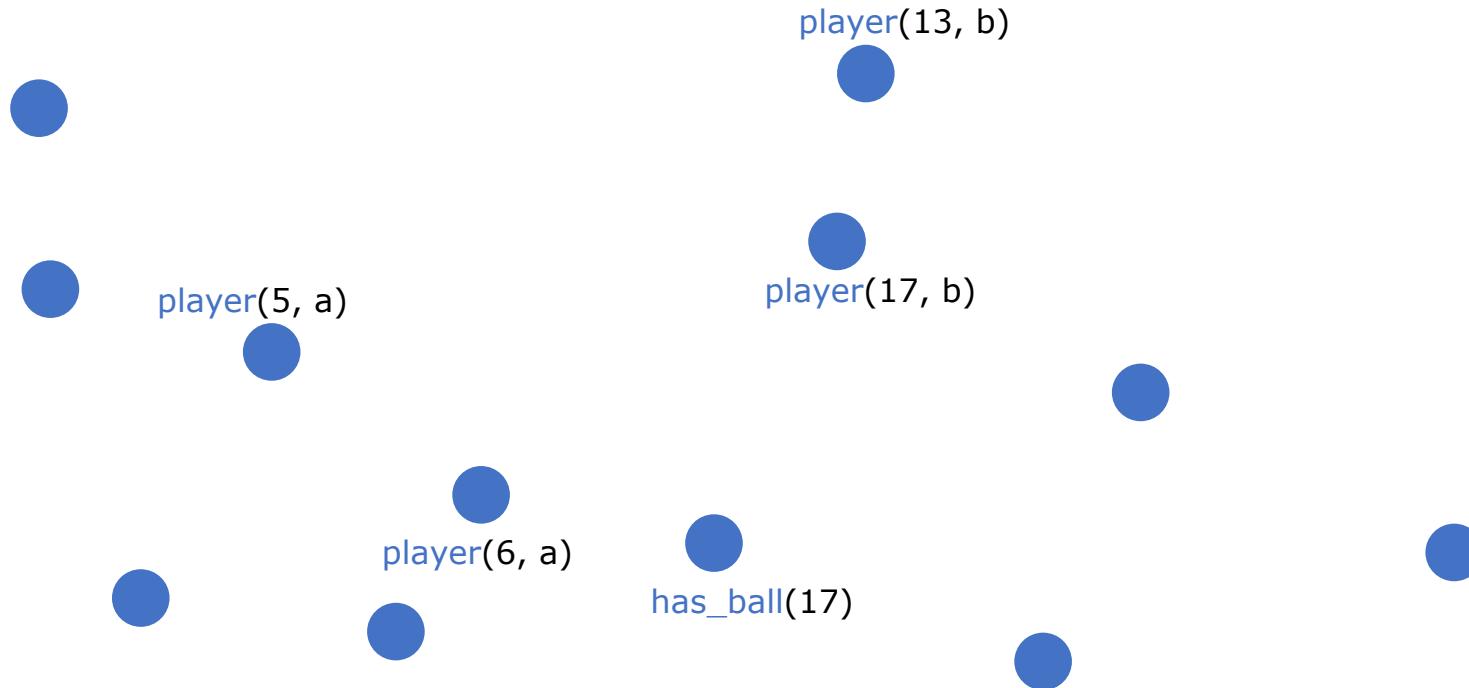
kick      only if has ball  
pass



# Experiment: Robots Kick/Pass/Steal

22 robot soccer players  
player(Number, Team)

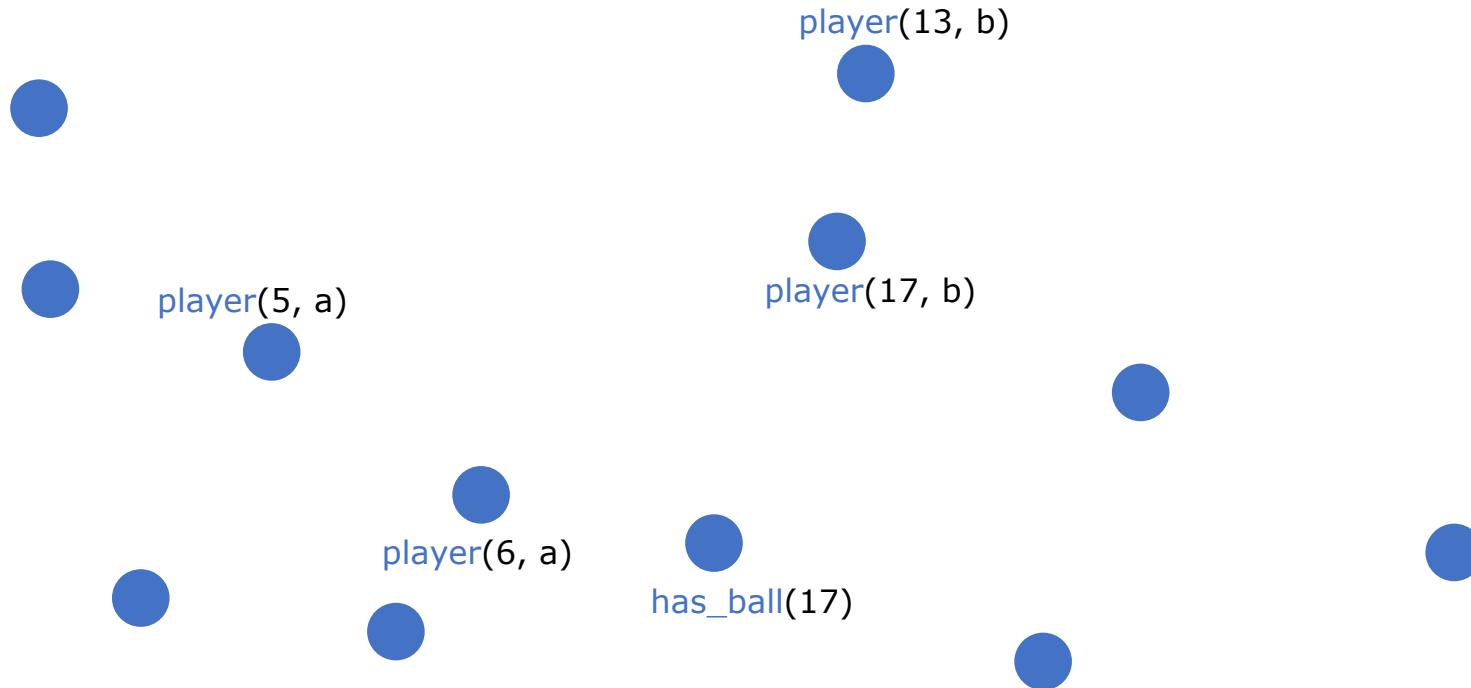
kick      only if has ball  
pass      only to a teammate



# Experiment: Robots Kick/Pass/Steal

22 robot soccer players  
player(Number, Team)

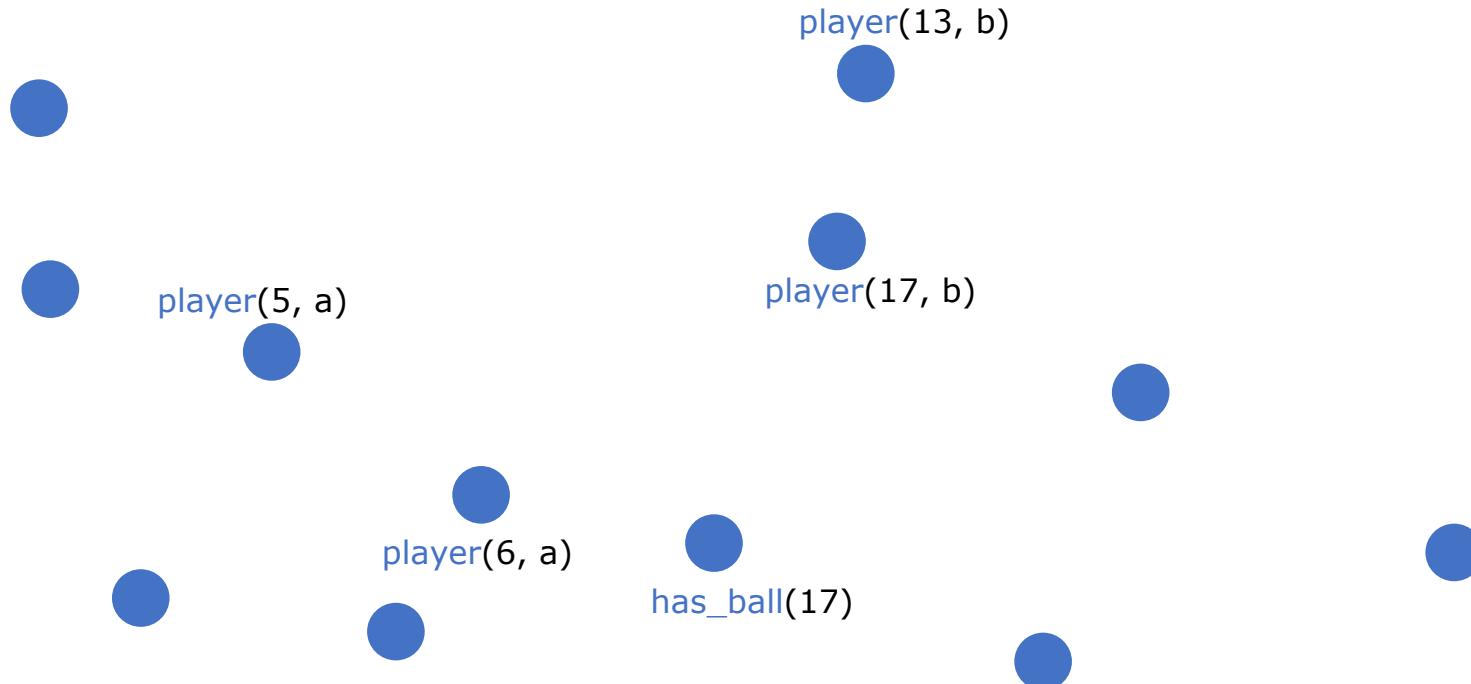
kick      only if has ball  
pass      only to a teammate  
steal



# Experiment: Robots Kick/Pass/Steal

22 robot soccer players  
player(Number, Team)

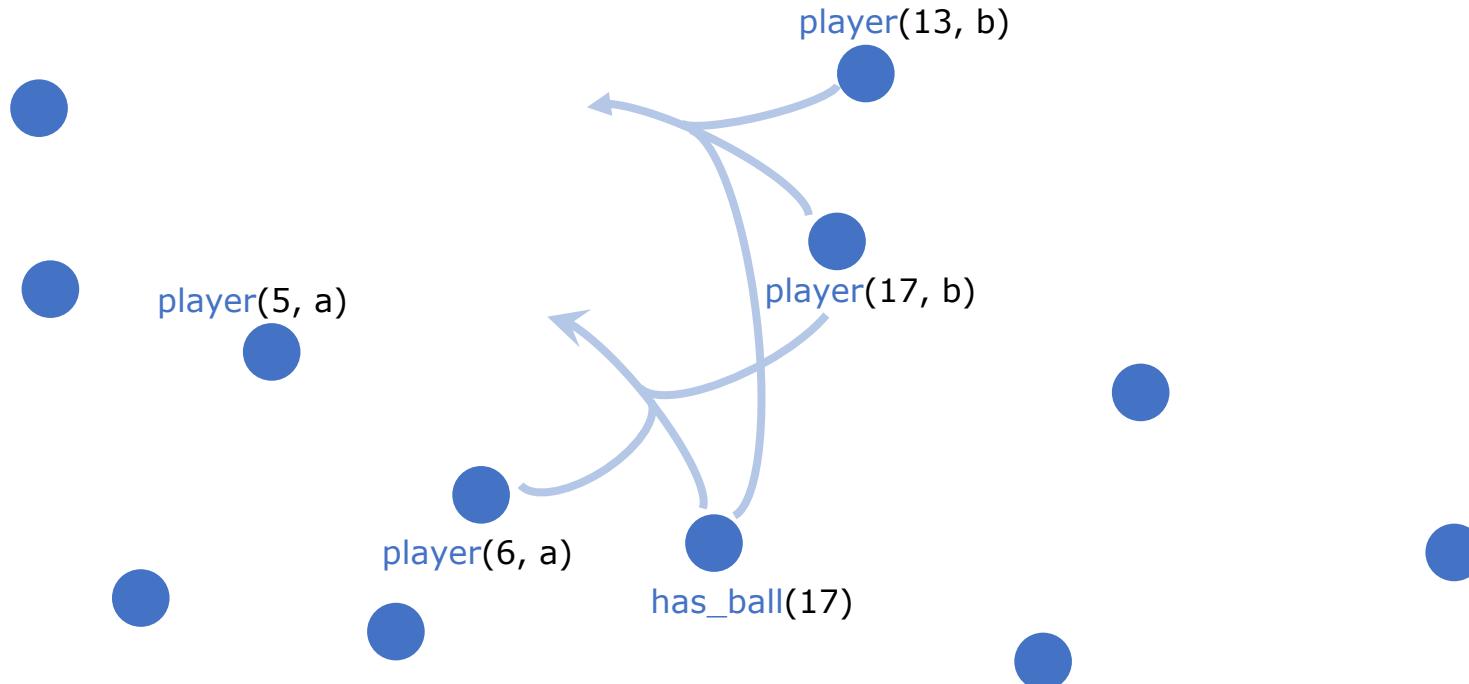
kick      only if has ball  
pass     only to a teammate  
steal    only from opponent



# Experiment: Robots Kick/Pass/Steal

22 robot soccer players  
player(Number, Team)

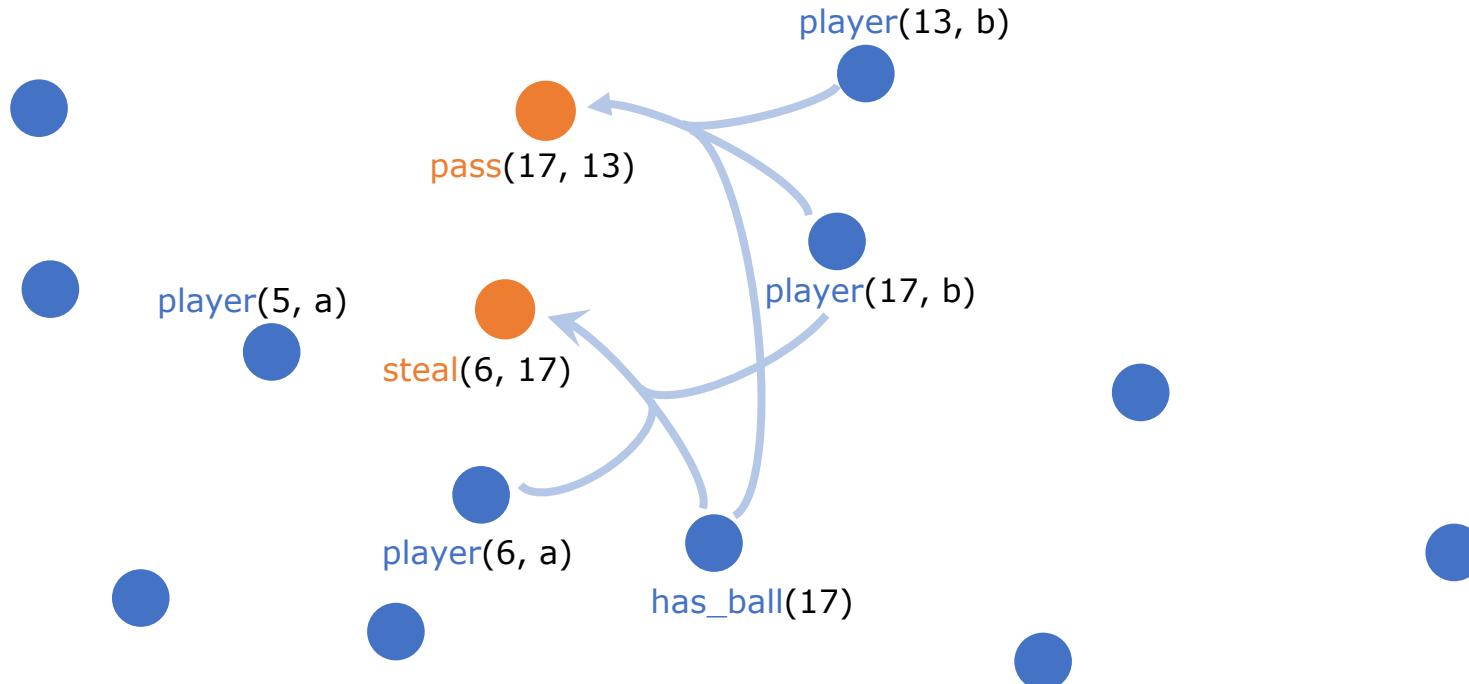
kick      only if has ball  
pass     only to a teammate  
steal    only from opponent



# Experiment: Robots Kick/Pass/Steal

22 robot soccer players  
player(Number, Team)

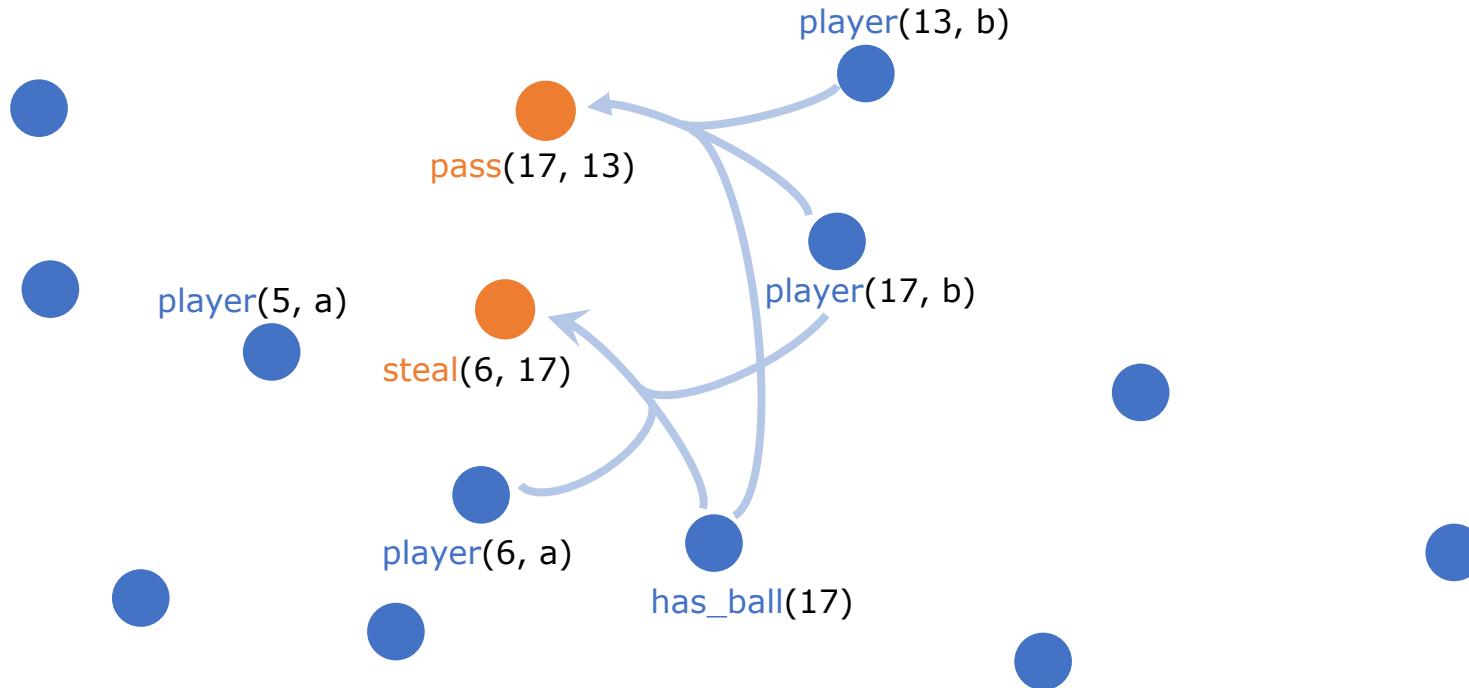
kick	only if has ball
pass	only to a teammate
steal	only from opponent



# Experiment: Robots Kick/Pass/Steal

22 robot soccer players  
player(Number, Team)

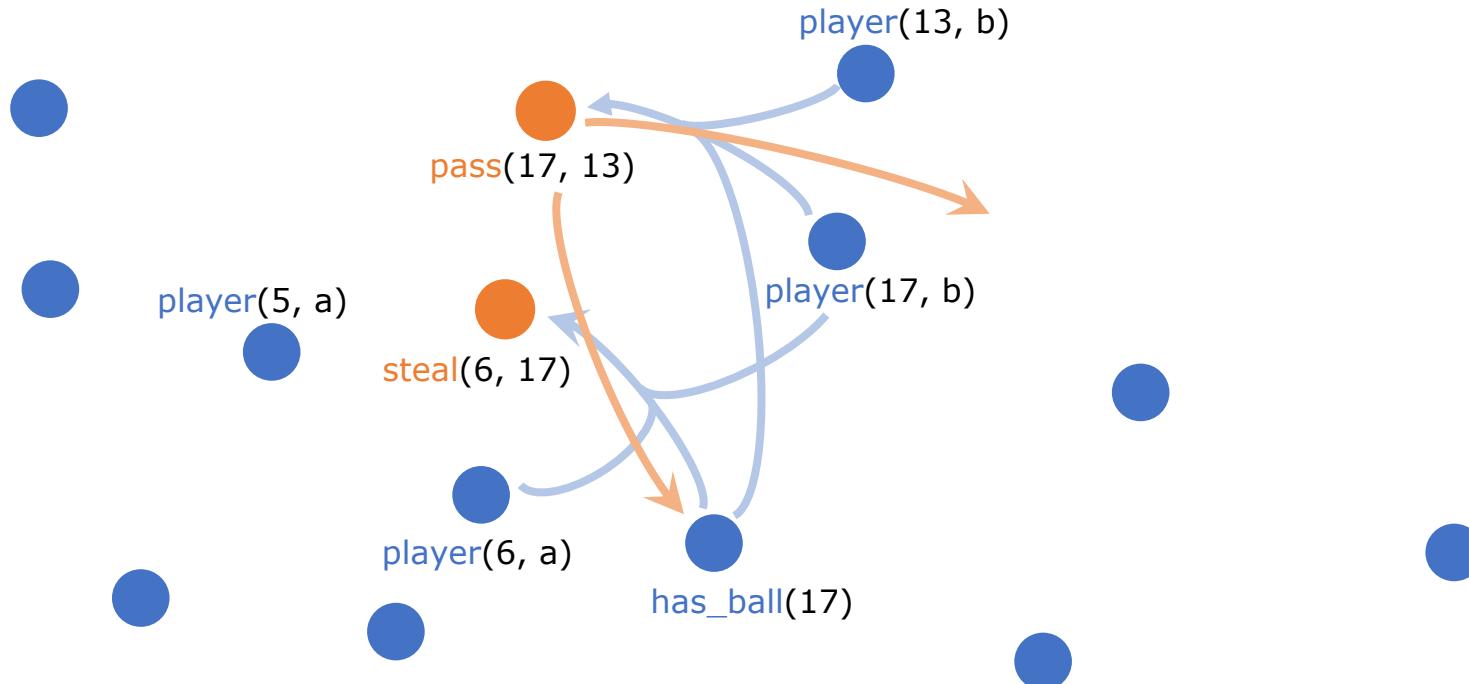
kick	only if has ball
pass	only to a teammate
steal	only from opponent



# Experiment: Robots Kick/Pass/Steal

22 robot soccer players  
player(Number, Team)

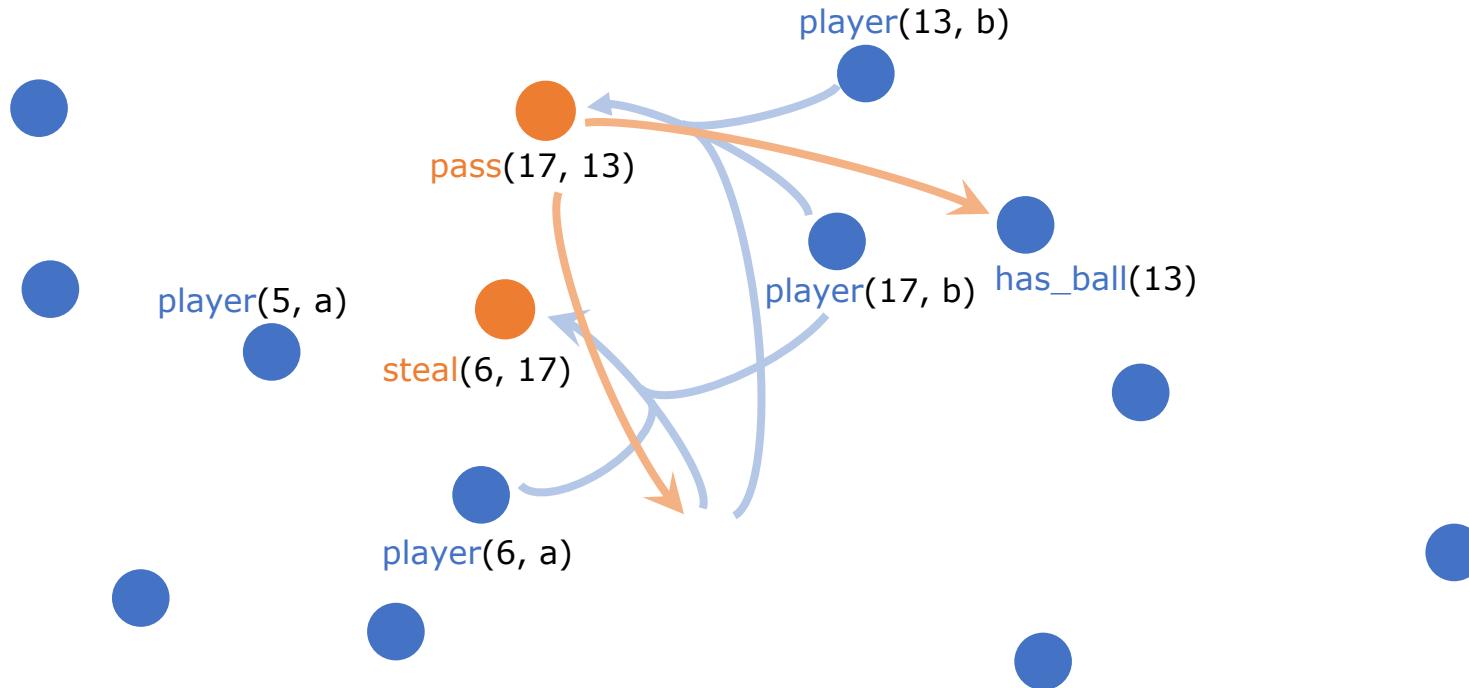
kick	only if has ball
pass	only to a teammate
steal	only from opponent



# Experiment: Robots Kick/Pass/Steal

22 robot soccer players  
player(Number, Team)

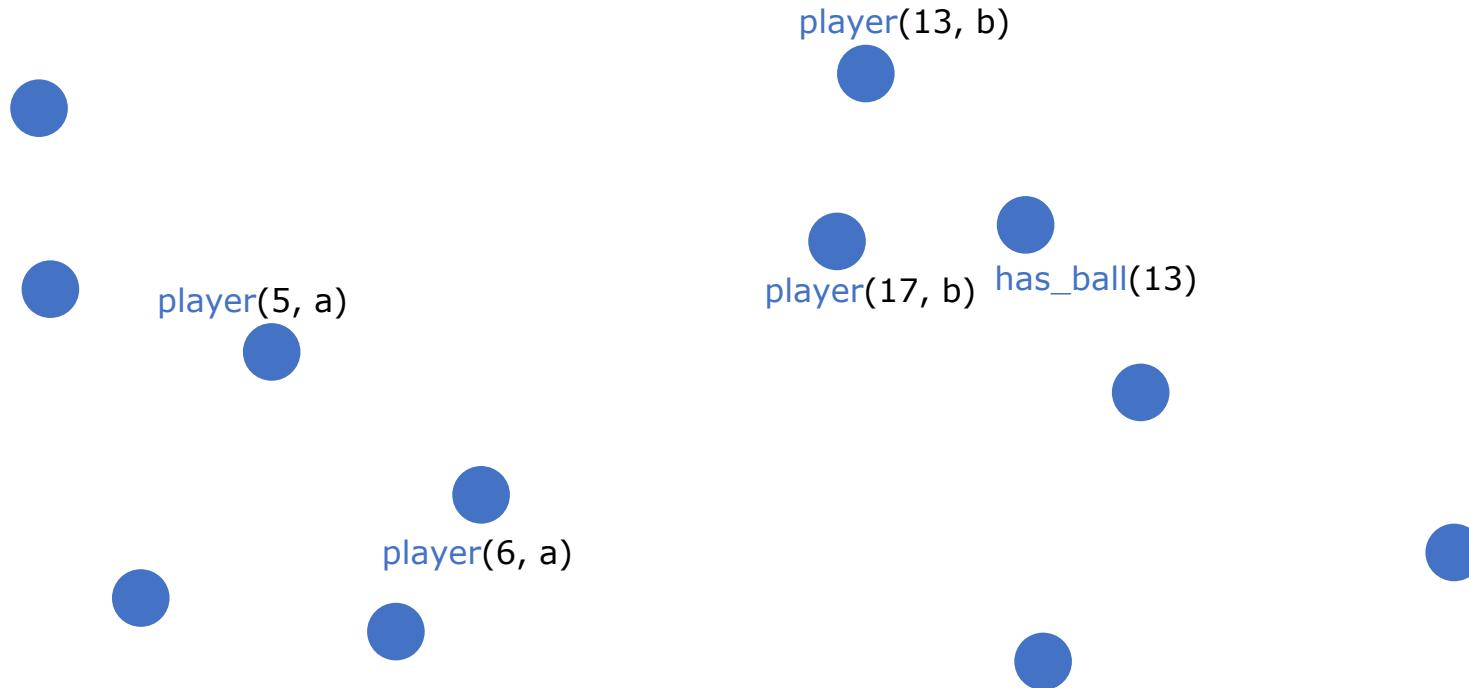
kick	only if has ball
pass	only to a teammate
steal	only from opponent



# Experiment: Robots Kick/Pass/Steal

22 robot soccer players  
player(Number, Team)

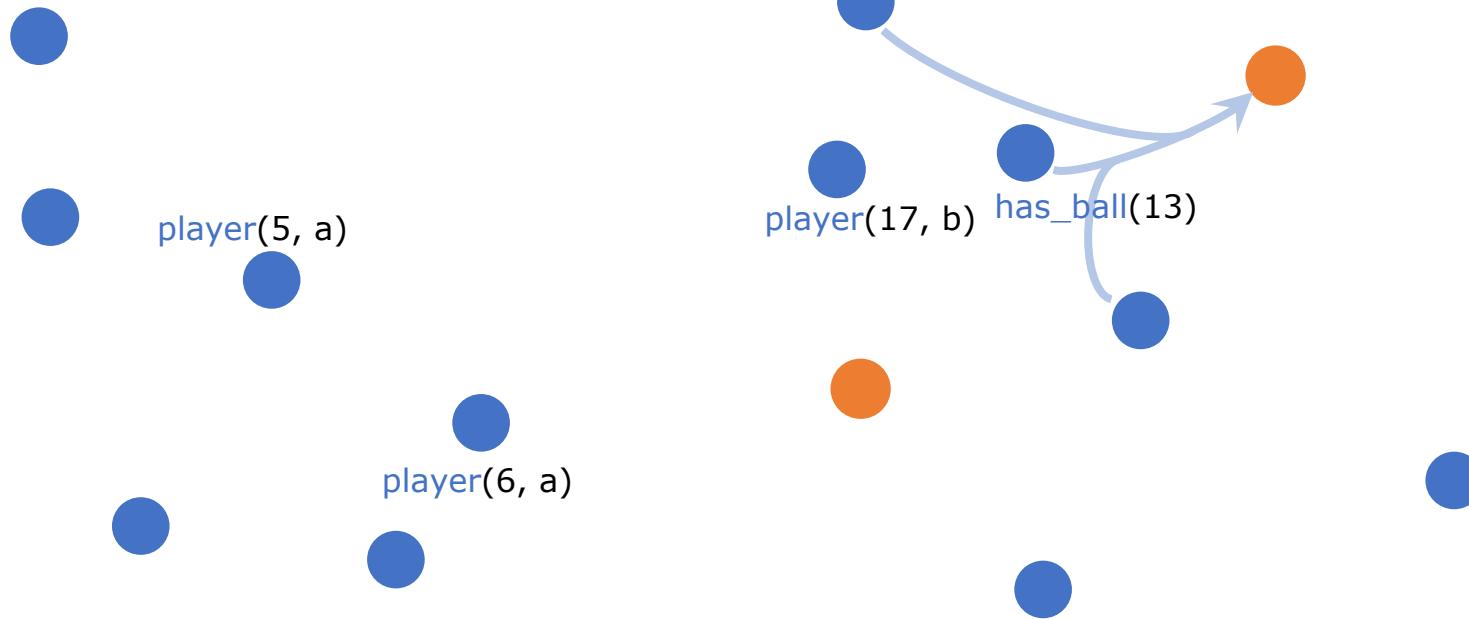
kick      only if has ball  
pass      only to a teammate  
steal      only from opponent



# Experiment: Robots Kick/Pass/Steal

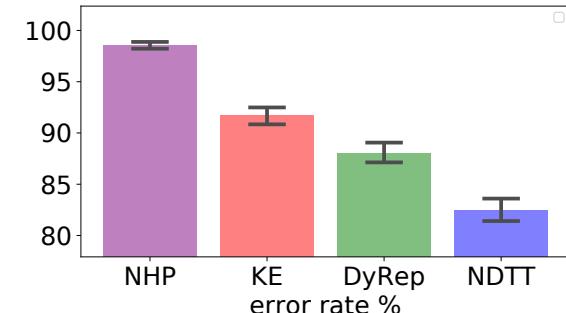
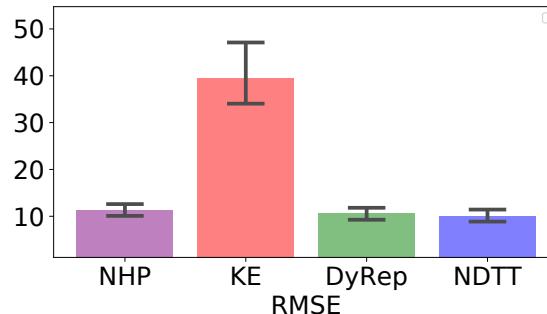
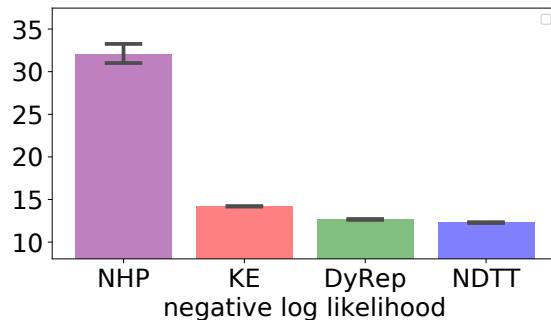
22 robot soccer players  
player(Number, Team)

kick	only if has ball
pass	only to a teammate
steal	only from opponent

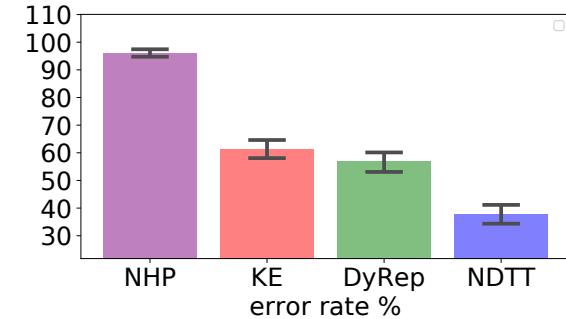
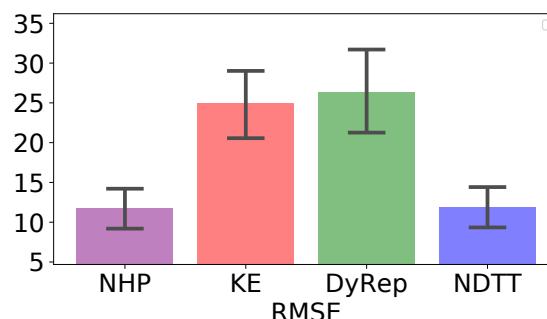
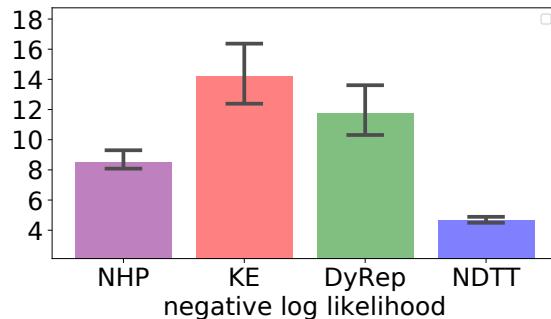


# Results: NDTT > Competitors

3 error metrics (in 3 columns): smaller is better



users watch TV programs



robots kick/pass/steal soccer ball

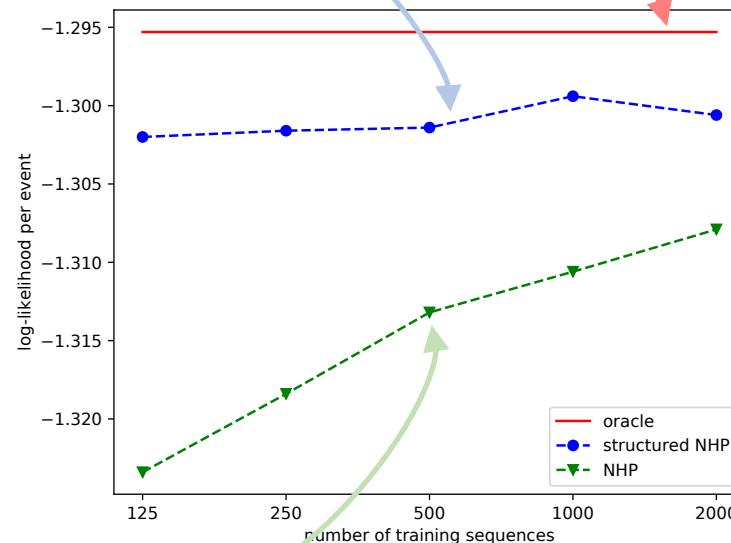
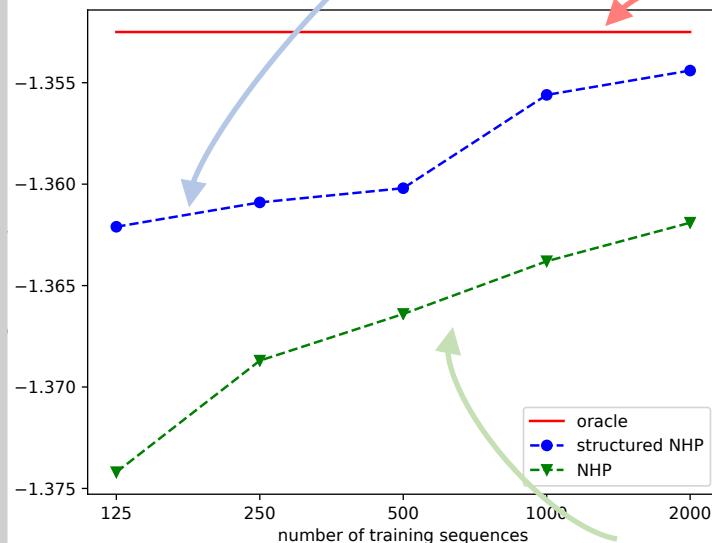
# Good Generalization with Less Data

log-likelihood

Neural Datalog

Through Time

Oracle



Neural Hawkes process

# of training sequences

**Summary:**

**Deep Recurrent Net**

# Summary:

## Deep Recurrent Net

e.g., RNN    discrete-time  
LSTM

# Summary:

## Deep Recurrent Net

e.g., RNN    discrete-time  
LSTM  
neural Hawkes process    continuous-time

# Summary:

## Deep Recurrent Net

hidden  
system  
state

e.g., RNN    discrete-time  
LSTM  
neural Hawkes process    continuous-time

# Summary:

## Deep Recurrent Net

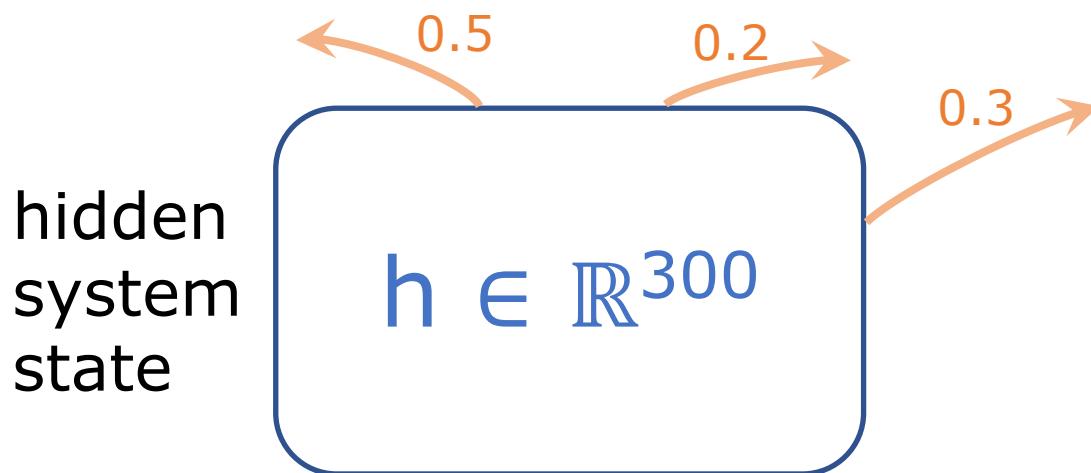
hidden  
system  
state

$$h \in \mathbb{R}^{300}$$

e.g., RNN    discrete-time  
LSTM  
neural Hawkes process    continuous-time

# Summary:

## Deep Recurrent Net



hidden  
system  
state

e.g., RNN    discrete-time  
LSTM  
neural Hawkes process    continuous-time

# Summary:

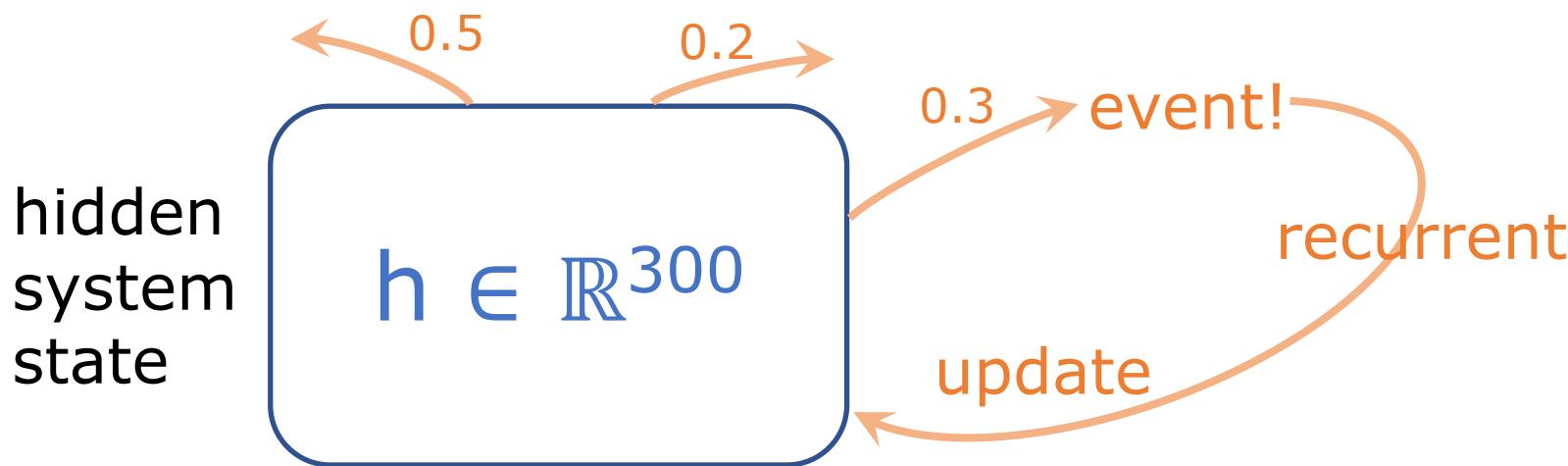
## Deep Recurrent Net



e.g., RNN    discrete-time  
LSTM  
neural Hawkes process    continuous-time

## Summary:

## Deep Recurrent Net



e.g., RNN    discrete-time  
LSTM  
neural Hawkes process    continuous-time

# Summary: Logic → Deep Recurrent Net

hidden  
system  
state



# Summary: Logic → Deep Recurrent Net

**distributed**

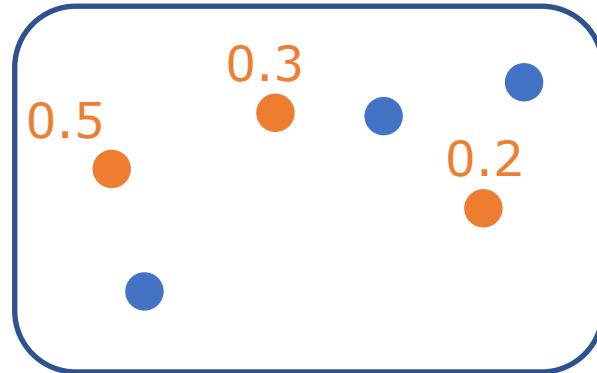
hidden  
system  
state



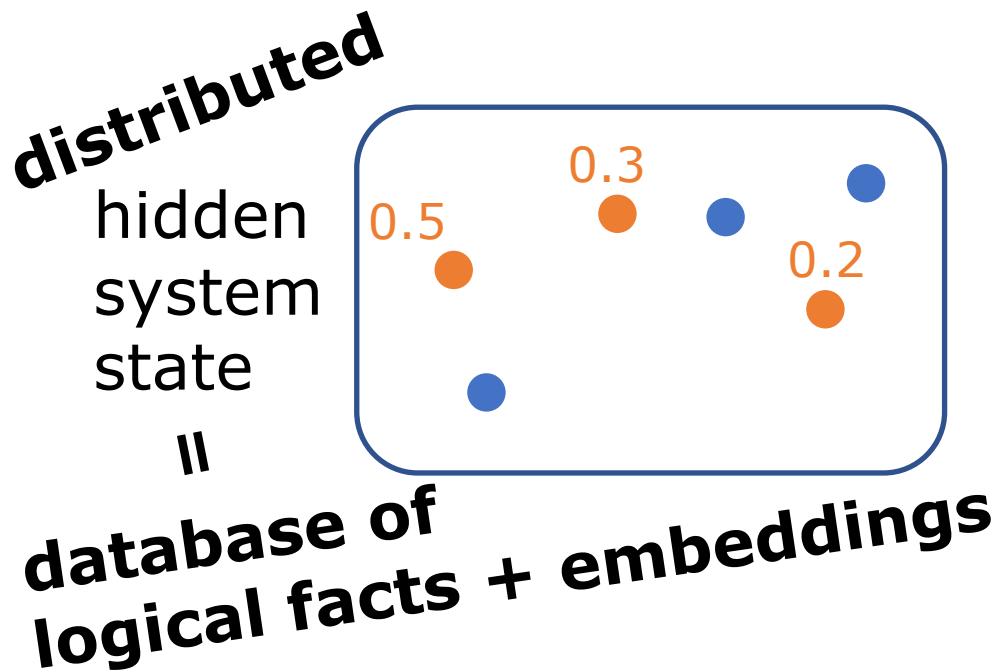
# Summary: Logic → Deep Recurrent Net

*distributed*

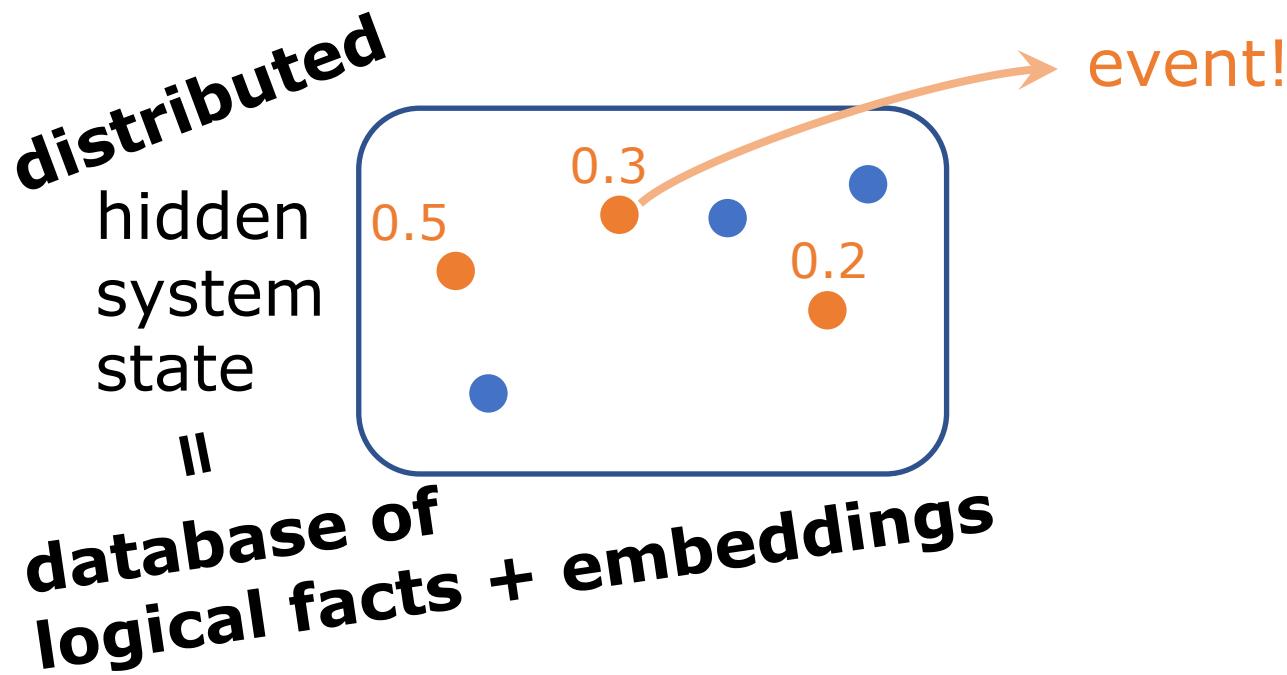
hidden  
system  
state



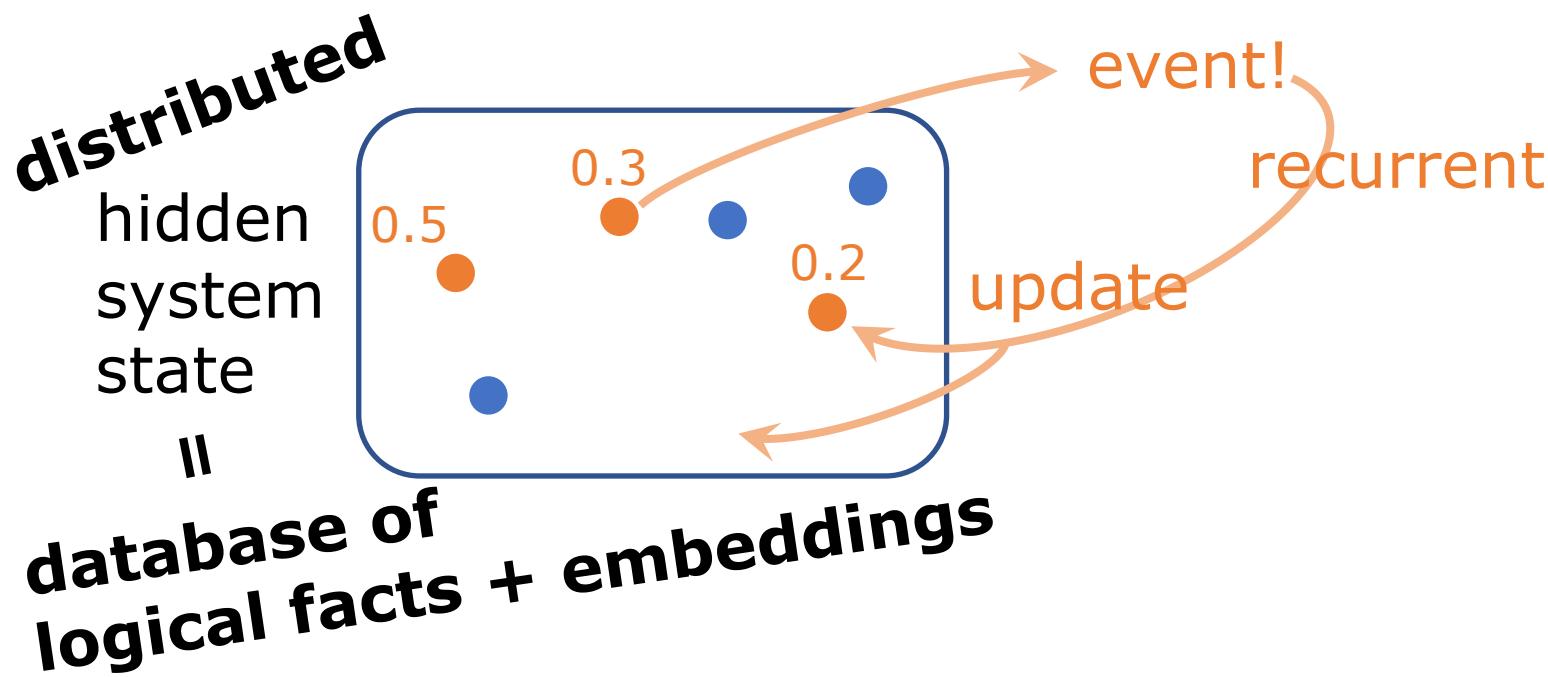
# Summary: Logic → Deep Recurrent Net



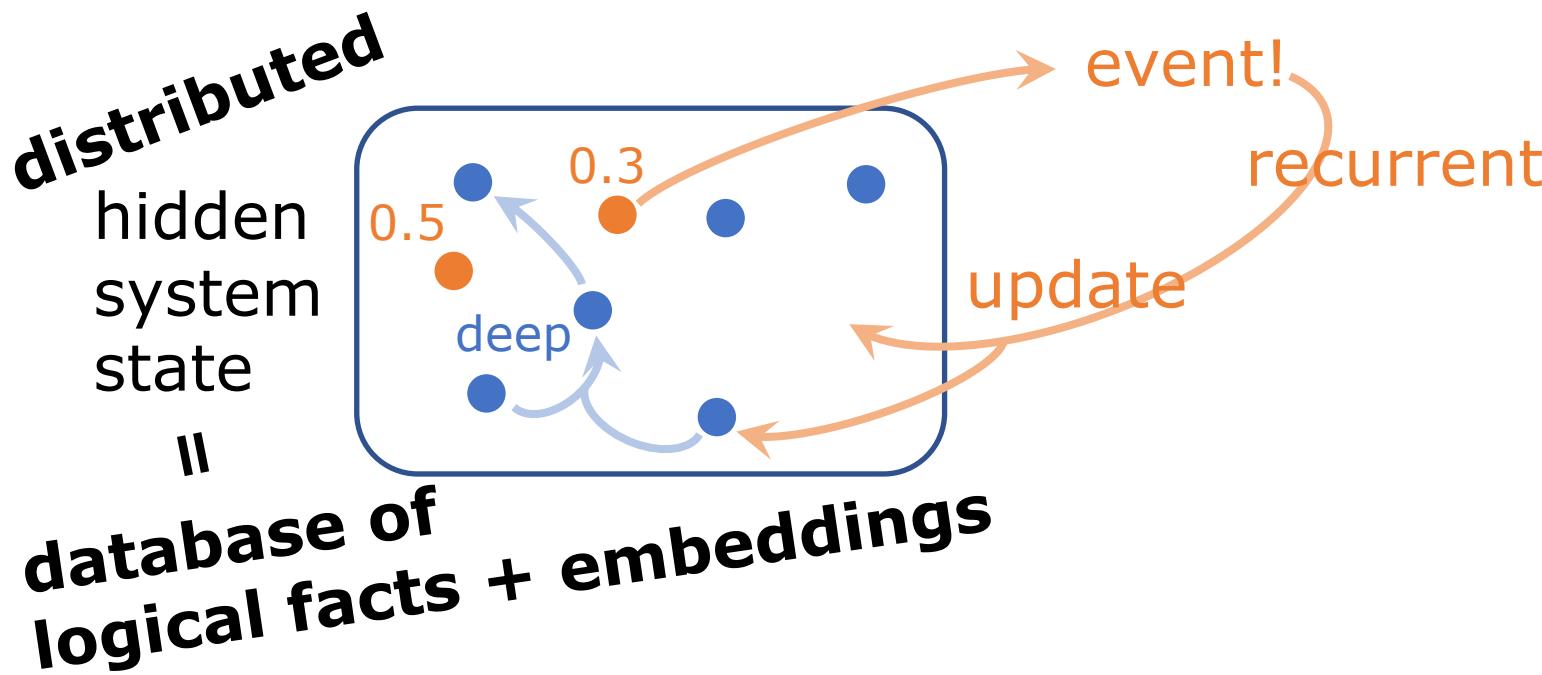
# Summary: Logic → Deep Recurrent Net



# Summary: Logic → Deep Recurrent Net



# Summary: Logic → Deep Recurrent Net



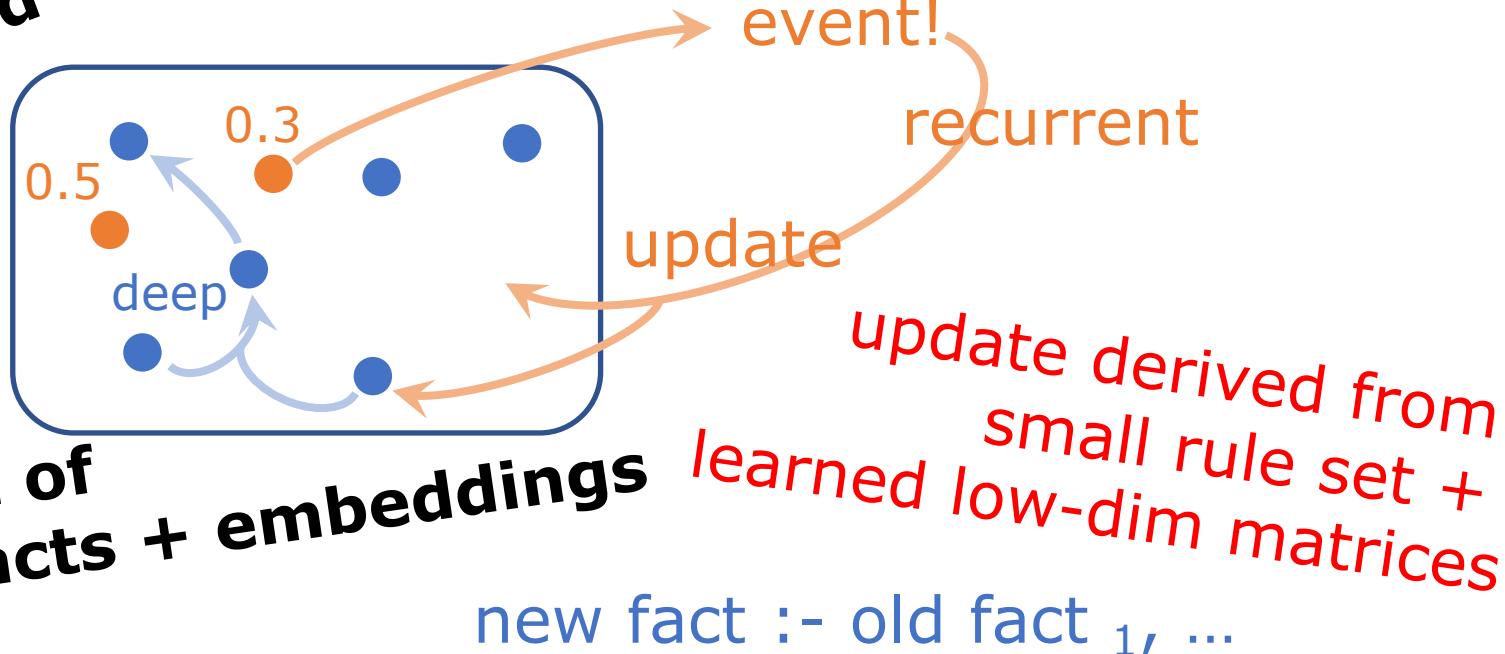
# Summary: Logic → Deep Recurrent Net

**distributed**

hidden  
system  
state

!!

**database of  
logical facts + embeddings**



new fact :- old fact<sub>1</sub>, ...



new fact <- event, ...

! old fact <- event, ...

# Summary: Logic → Deep Recurrent Net

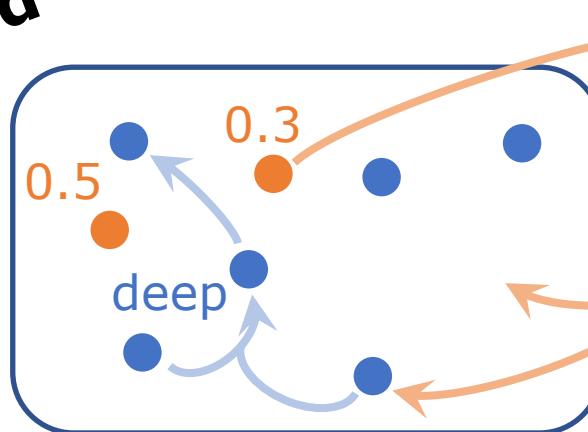
**distributed**

hidden  
system  
state

!!

**database of  
logical facts + embeddings**

try our code  
in your domain!



event!  
recurrent  
update  
update derived from  
small rule set +  
learned low-dim matrices  
new fact :- old fact<sub>1</sub>, ...



new fact ← event, ...  
! old fact ← event, ...

# Neural Datalog Through Time

**Thank You**

Bloomberg PhD Fellowship

Songyun Duan

Yujie Zha

ICLR NSF

Karan Uppal

ICML reviewers

Rakshit Trivedi

MARCC

Hongteng Xu

JHU-CLSP

NVIDIA

Argo Lab

## Any Questions?

**Thank You**

**<http://bburl/tpp-slides-p3>**

**<http://bburl/tpp-lab-p3>**