***StarfishDB*:**
Probabilistic Programming Datalog in Action

Niccolò Meneghetti
University of Michigan-Dearborn
niccolom@umich.edu

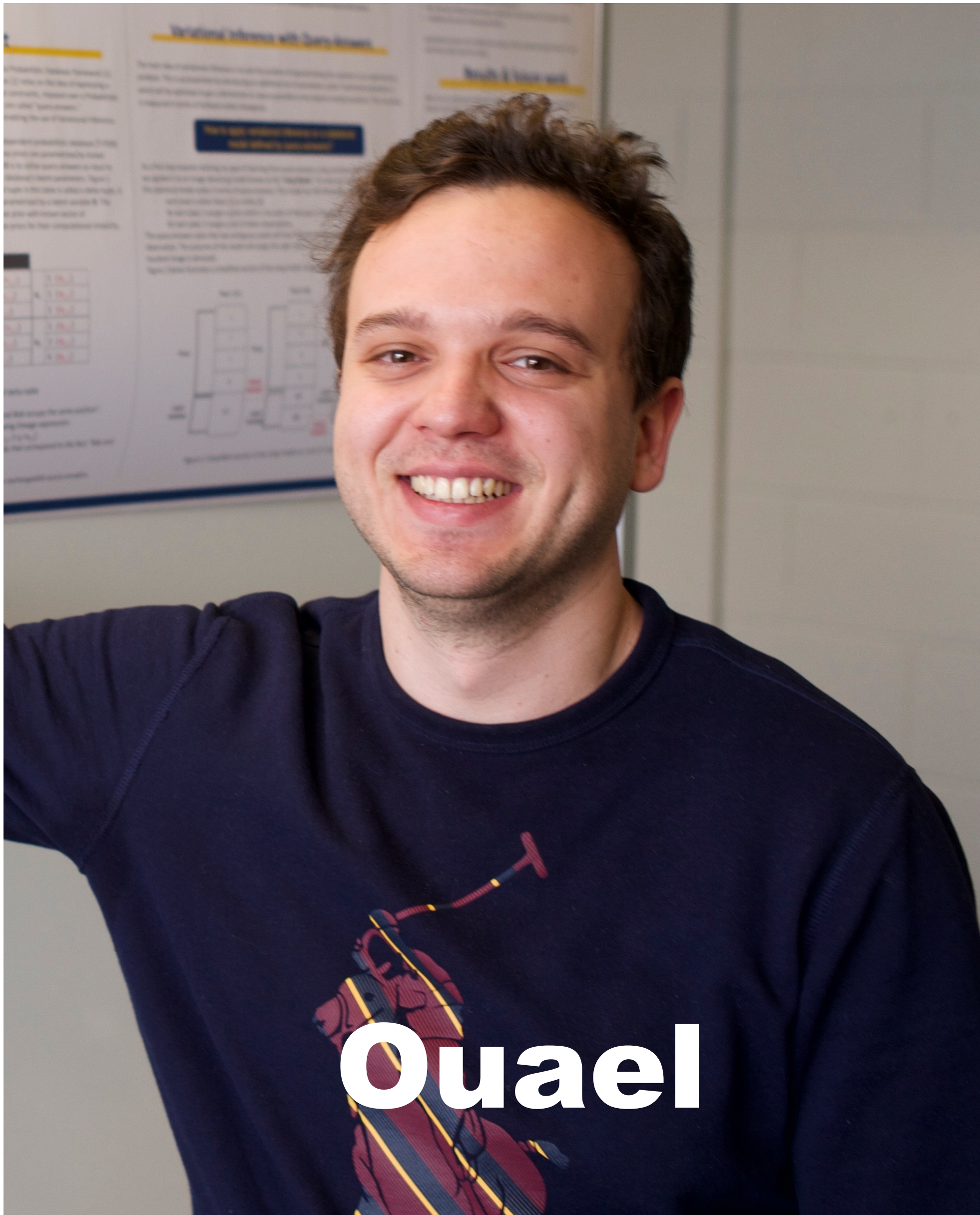
Outline

StarfishDB: a Query Execution Engine for Relational Probabilistic Programming

OUAEL BEN AMARA*, University of Michigan-Dearborn, U.S.A.
SAMI HADOUAJ*, University of Michigan-Dearborn, U.S.A.
NICCOLÒ MENEGHETTI, University of Michigan-Dearborn, U.S.A.

- De Finetti Logic Programming
- Variational Inference
- Probabilistic Programming Datalog

The Authors



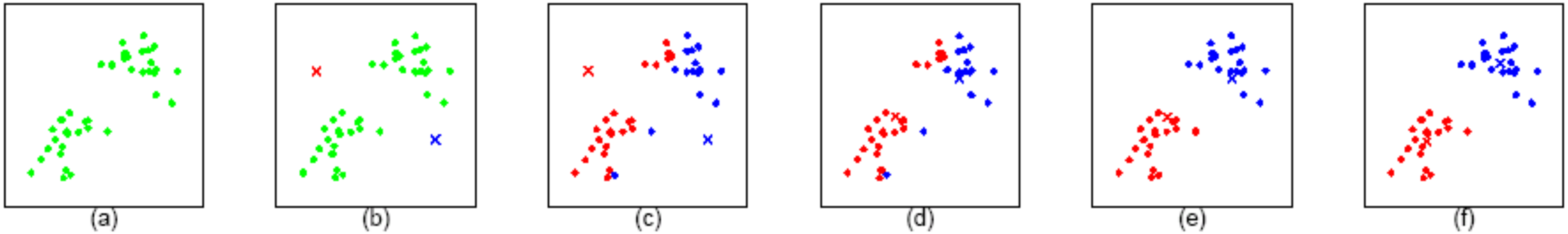
Probabilistic Programming

Generative Story: $z \sim \text{Categorical}(\phi)$

$\mathbf{x} \sim \text{Gaussian}(\boldsymbol{\mu}_z, \boldsymbol{\Sigma}_z)$

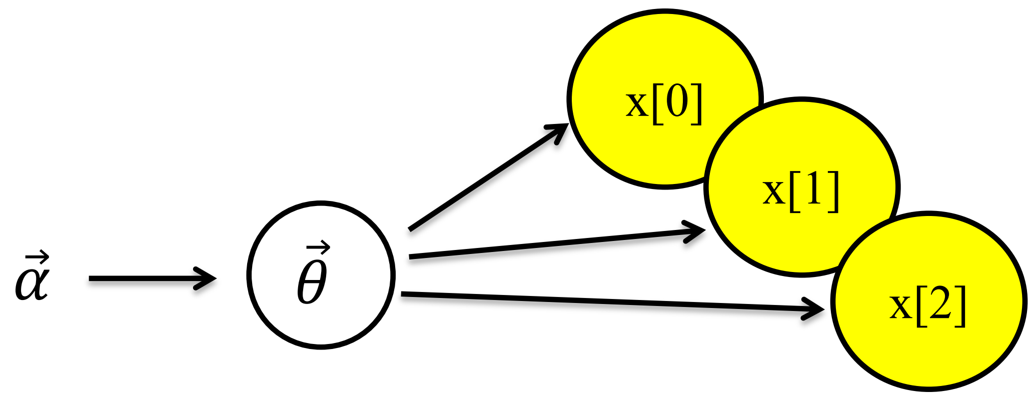
Data: $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$ where $\mathbf{x}^{(i)} \in \mathbb{R}^M$

Goal: compute the posterior density of the generative process w.r.t. the data.



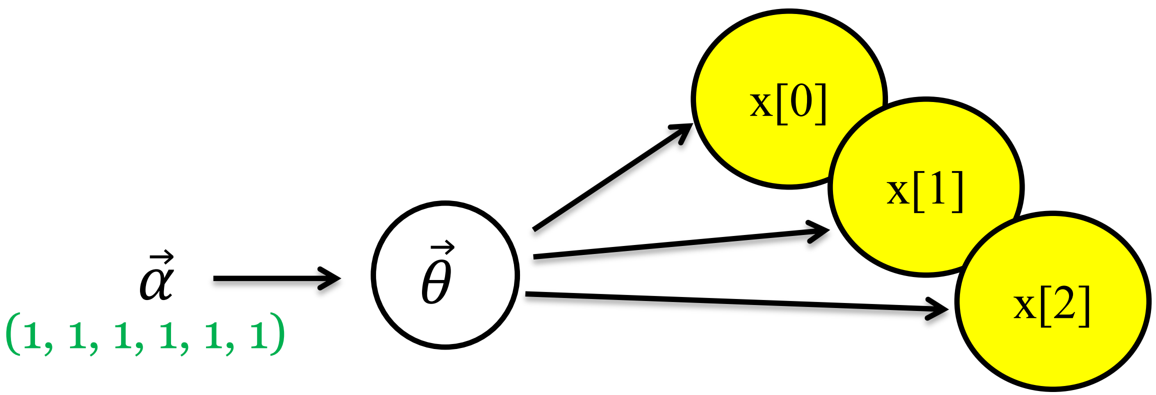
Pólya Dice

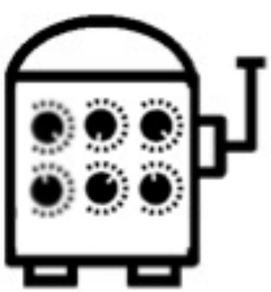
`Pólya die` → a *Categorical* distribution (parametrized by $\vec{\theta}$)
with a *Dirichlet* prior (parametrized by $\vec{\alpha}$)



Pólya Dice

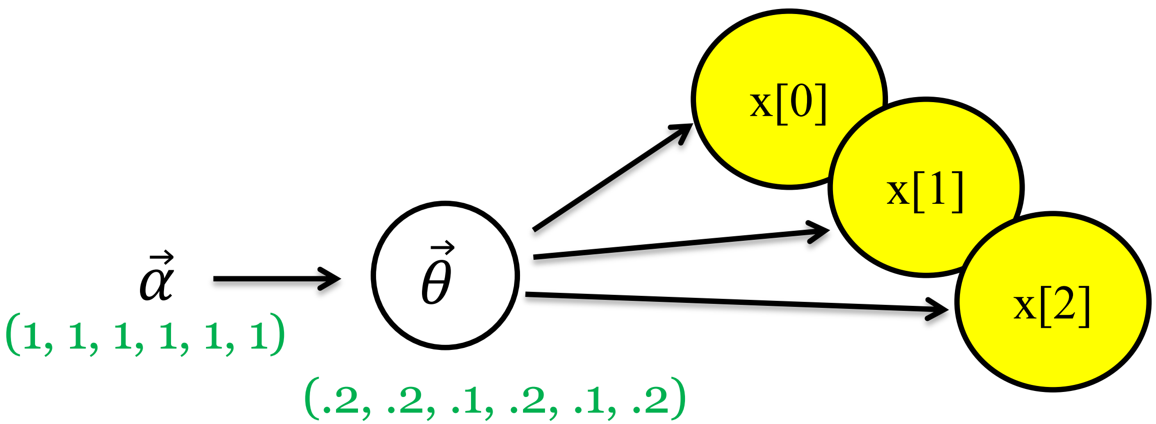
`Pólya die` → a *Categorical* distribution (parametrized by $\vec{\theta}$)
with a *Dirichlet* prior (parametrized by $\vec{\alpha}$)

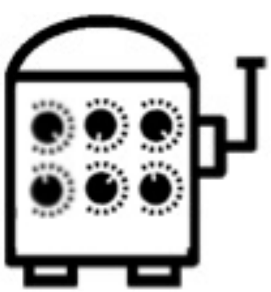


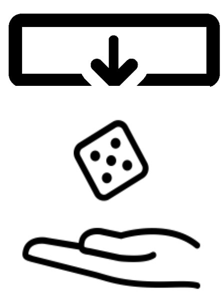


Pólya Dice

`Pólya die` → a *Categorical* distribution (parametrized by $\vec{\theta}$)
with a *Dirichlet* prior (parametrized by $\vec{\alpha}$)

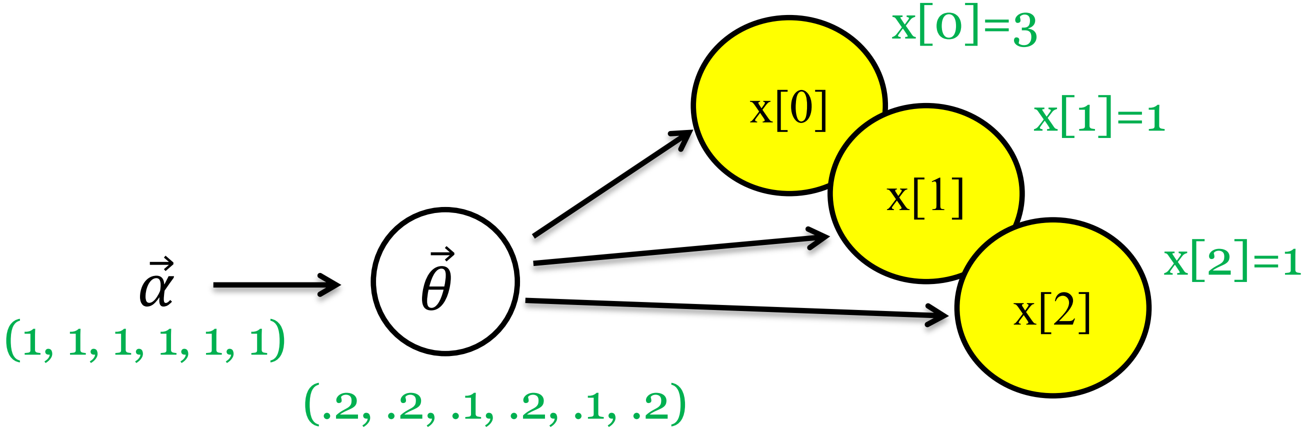


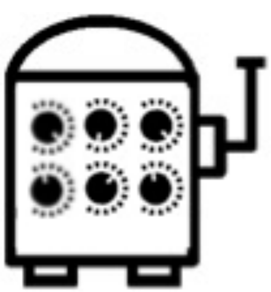


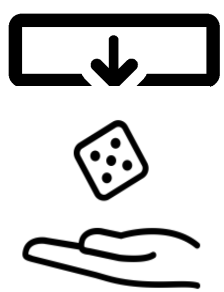


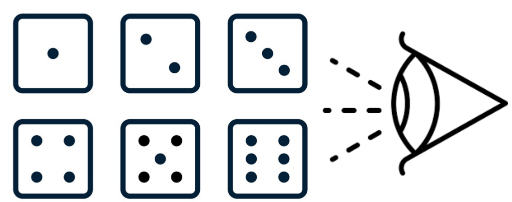
Pólya Dice

`Pólya die` → a *Categorical* distribution (parametrized by $\vec{\theta}$)
with a *Dirichlet* prior (parametrized by $\vec{\alpha}$)



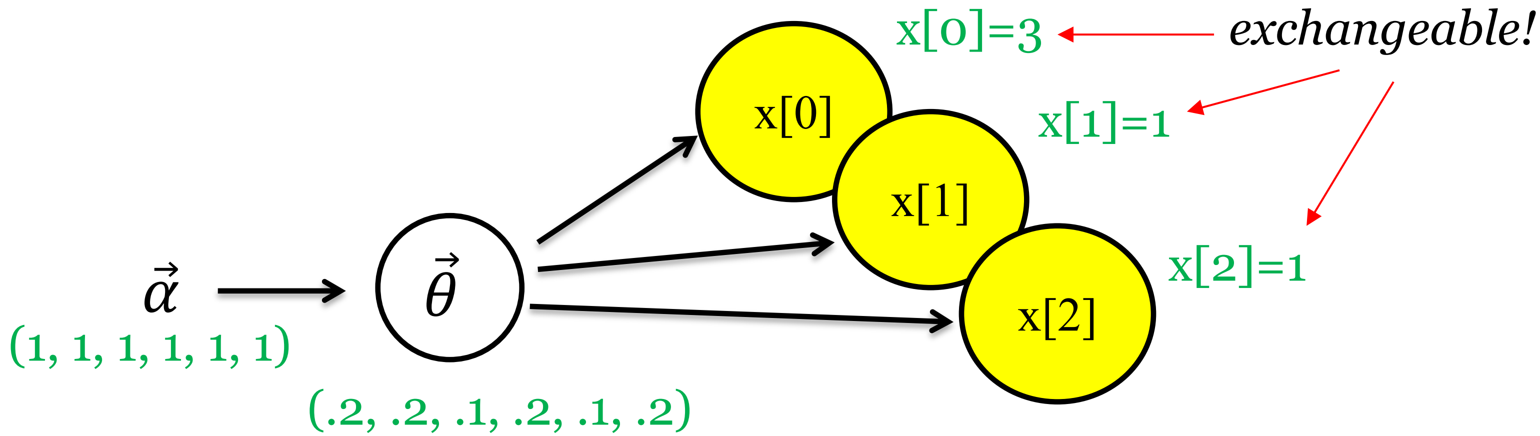


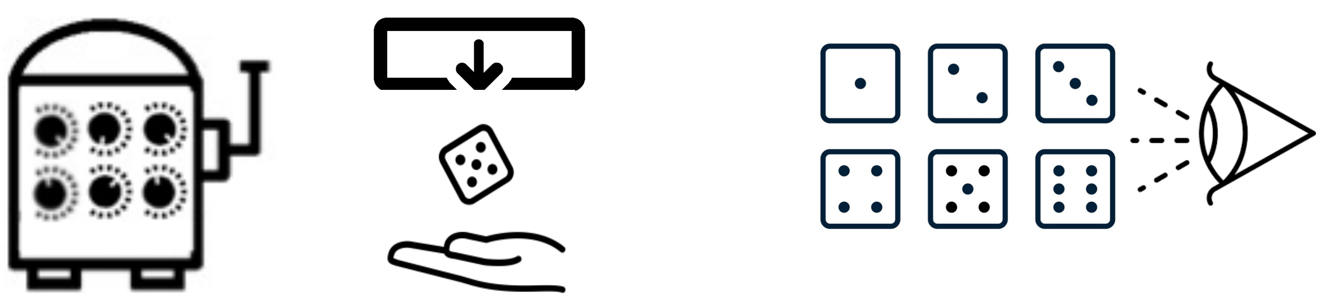




Pólya Dice

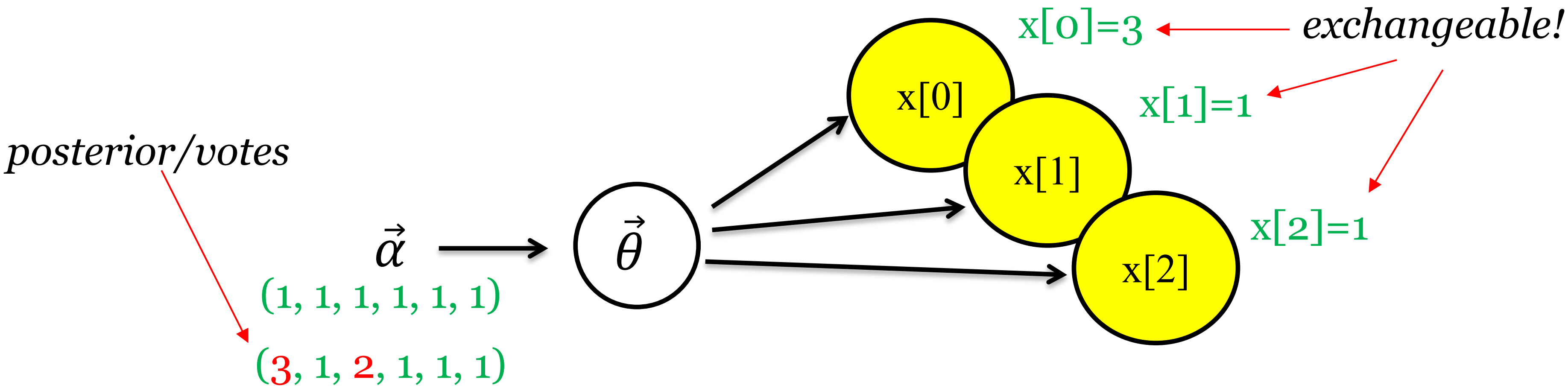
`Pólya die` → a *Categorical* distribution (parametrized by $\vec{\theta}$)
with a *Dirichlet* prior (parametrized by $\vec{\alpha}$)

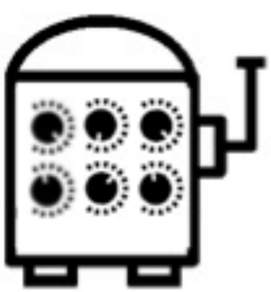


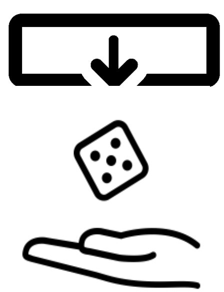


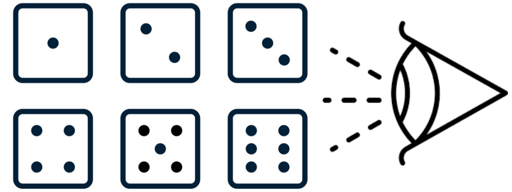
Pólya Dice

`Pólya die` → a *Categorical* distribution (parametrized by $\vec{\theta}$)
with a *Dirichlet* prior (parametrized by $\vec{\alpha}$)









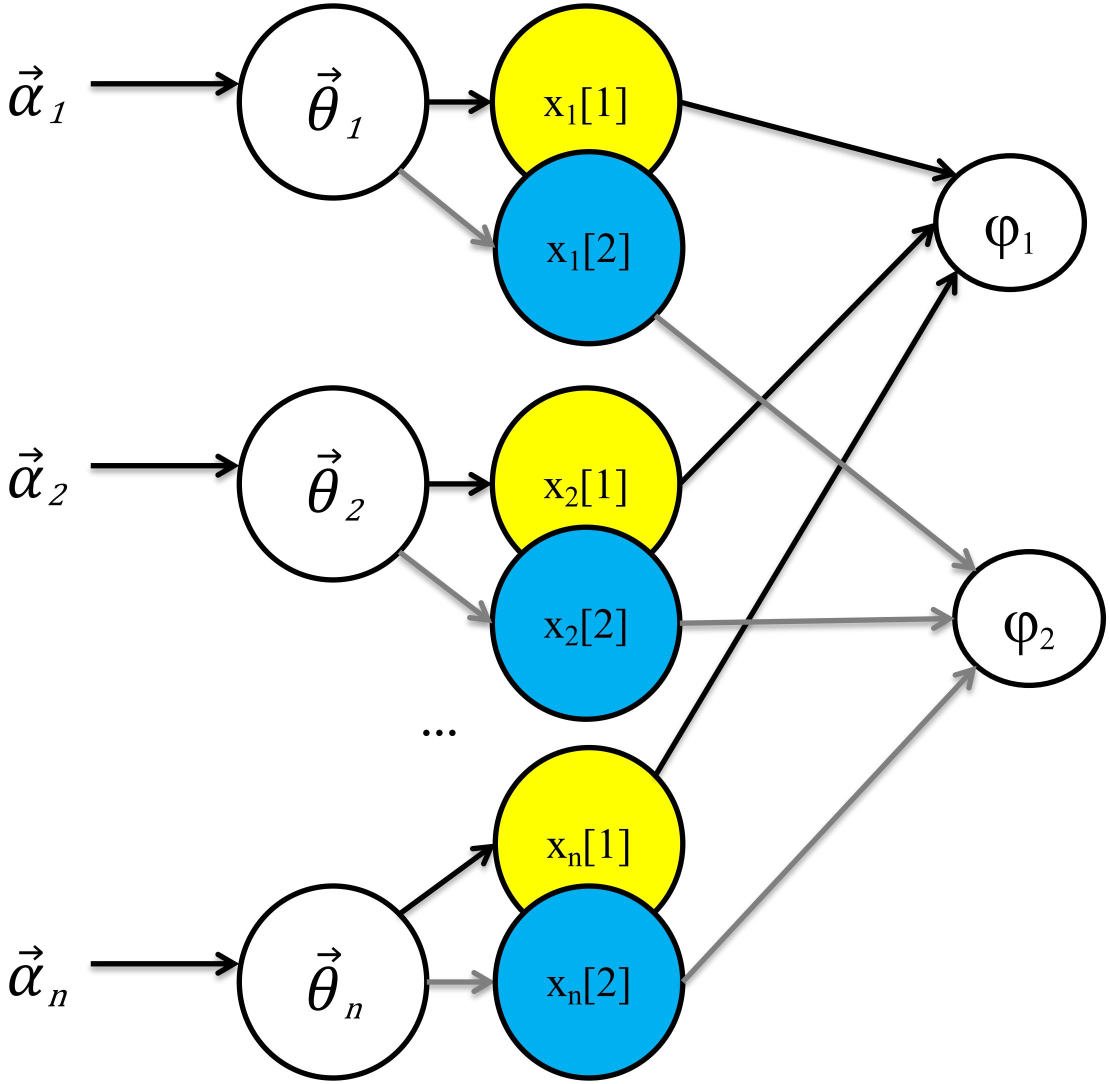
De Finetti Logic Programming

A *DFL Program* consists of:

1. A generative process
   (defined as a set of pairwise independent Pólya dice)

2. A set of exchangeable constraints

Exchangeable Constraints



$x_1[1]=6$
$\vee$
$x_2[1]=6$
$\vee$
$x_3[1]=6$

$x_1[2]=x_2[2]=x_3[2]$

De Finetti Logic Programming

A *DFL Program* consists of:

1. A generative process
   (defined as a set of pairwise independent Pólya dice)

2. A set of exchangeable constraints

product of Dirichlet densities

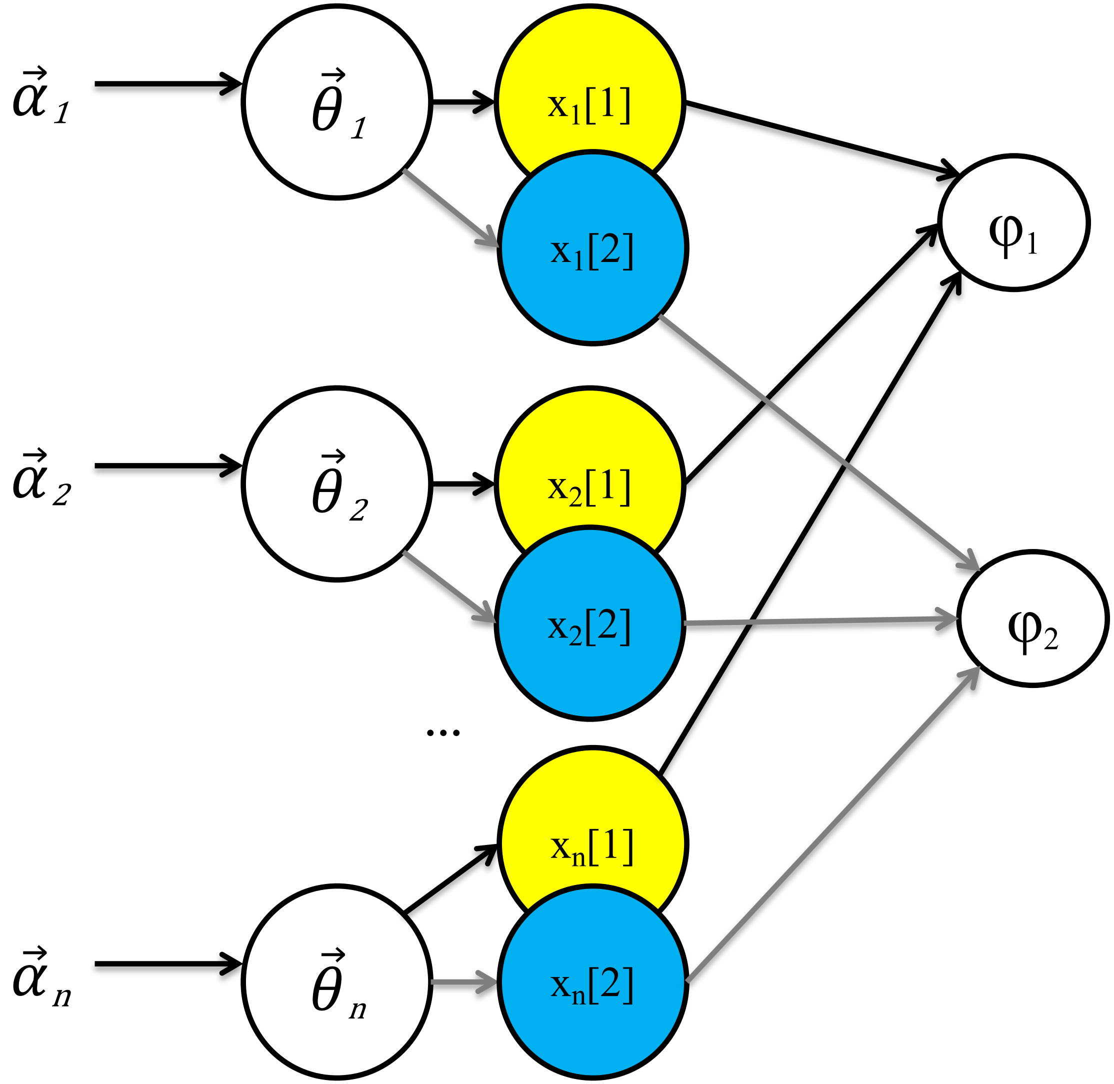
$$p(\Theta \mid \Phi, \mathbb{A}) = \sum_{\tau \in \textsc{Sat}(\Phi, \mathbb{X})} p(\Theta \mid \tau, \mathbb{A}) \cdot P(\tau \mid \Phi, \mathbb{A})$$

posterior

assignments to X that satisfy Φ

Pólya urn

Exchangeable Constraints



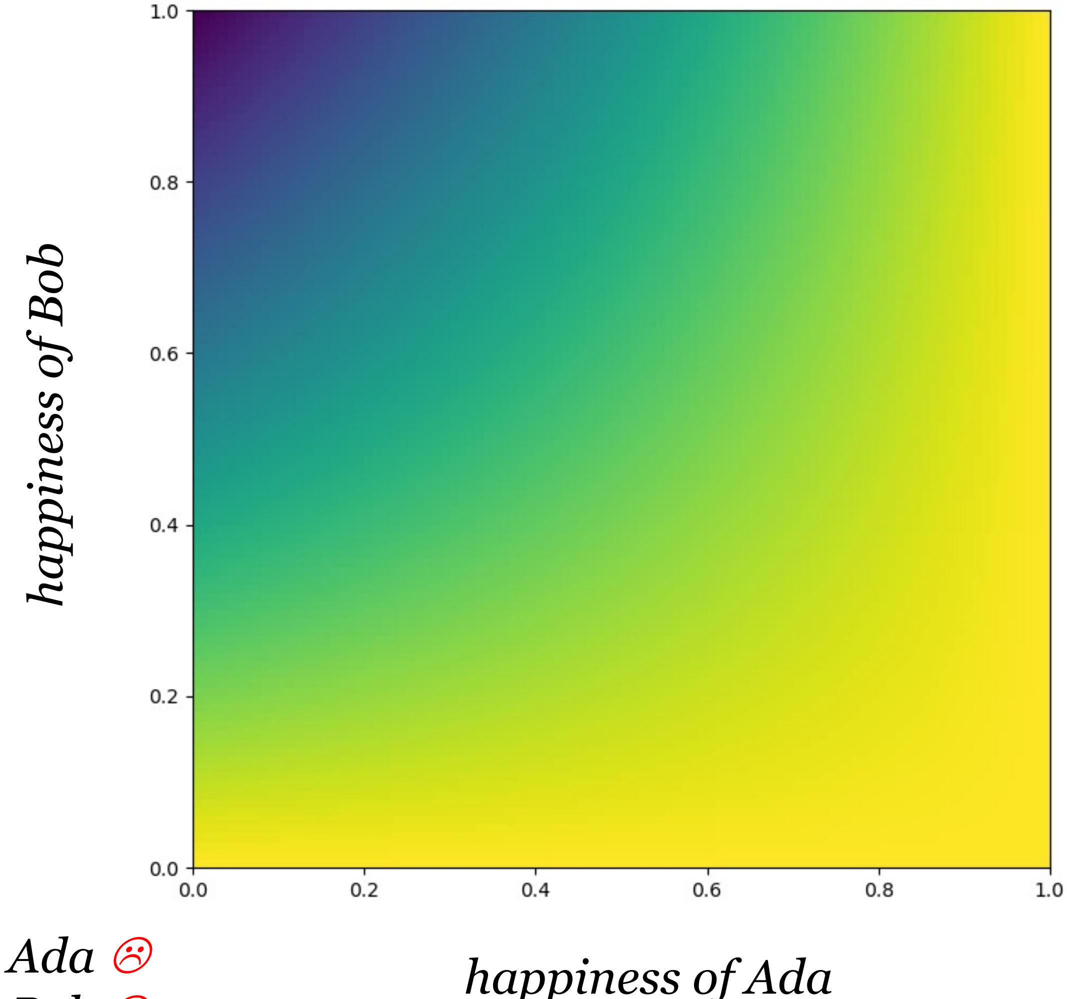
$x_1[1]=6$
$\vee$
$x_2[1]=6$
$\vee$
$x_3[1]=6$

$x_1[2]=x_2[2]=x_3[2]$

Posterior of two Pólya coins

Ada ☹
Bob ☺

Ada ☺
Bob ☺



C1: If Bob is happy, so is Ada

Ada ☹
Bob ☺

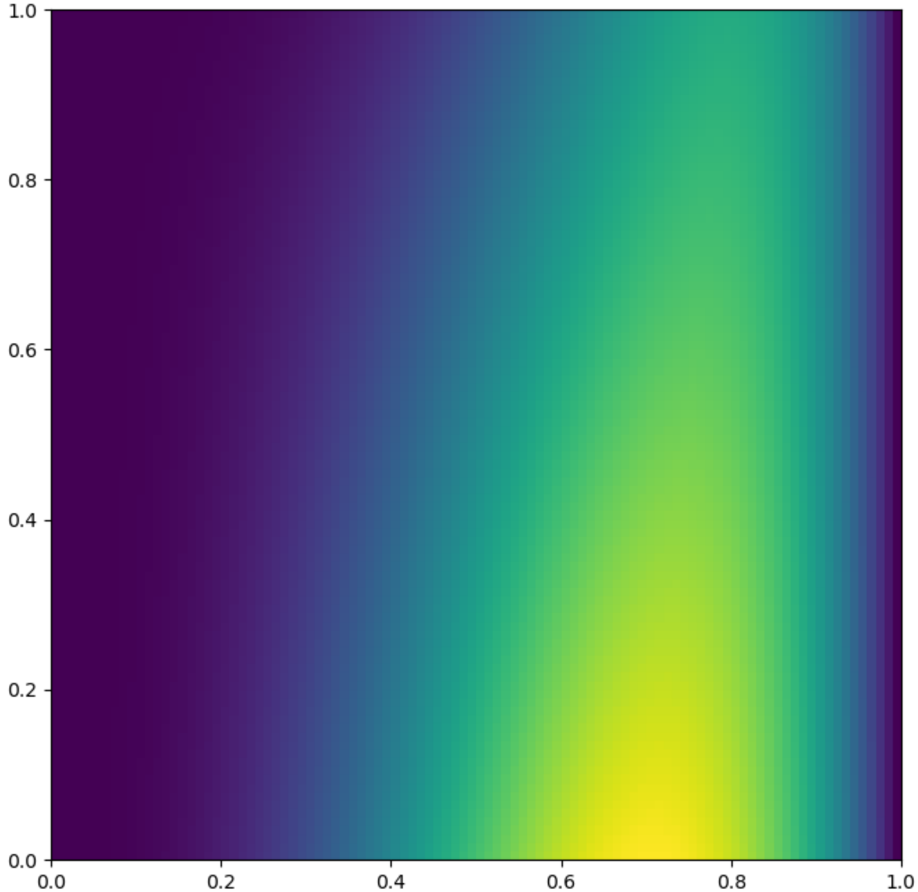
Ada ☺
Bob ☹

Posterior of two Pólya coins

Ada ☹
Bob ☺

Ada ☺
Bob ☺

happiness of Bob



C1: If Bob is happy, so is Ada
C2: Ada is happy
C3: Ada is happy
C4: Ada is happy
C5: Ada is sad

Ada ☹
Bob ☹

happiness of Ada

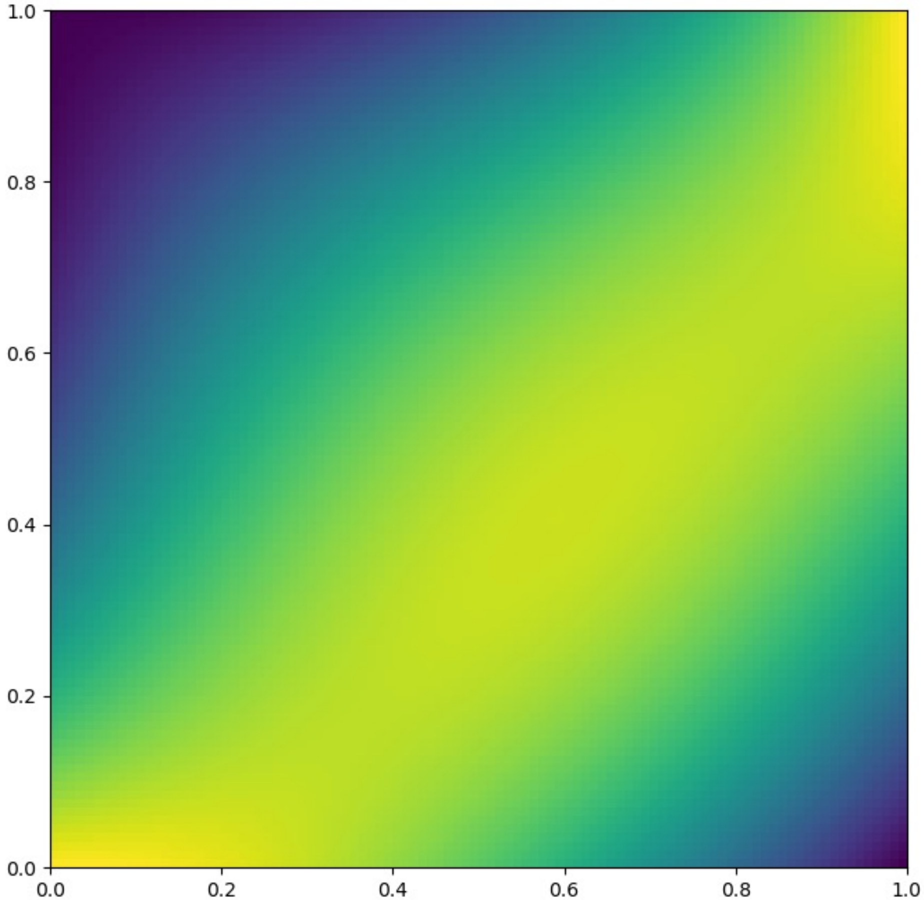
Ada ☺
Bob ☹

Posterior of two Pólya coins

Ada ☹
Bob ☺

Ada ☺
Bob ☺

happiness of Bob



happiness of Ada

C1: If Bob is happy, so is Ada
C2: Ada and Bob have the same mood

Ada ☹
Bob ☺

Ada ☺
Bob ☹

De Finetti Logic Programming

Computing the exact posterior is impractical. Two solutions:

1. Gibbs Sampling

2. Variational Inference

product of Dirichlet densities

$$p(\Theta \mid \Phi, \mathbb{A}) = \sum_{\tau \in \textsc{Sat}(\Phi, \mathbb{X})} p(\Theta \mid \tau, \mathbb{A}) \cdot P(\tau \mid \Phi, \mathbb{A})$$

posterior

assignments to X that satisfy Φ

Pólya urn

Variational Inference

$$q(\Theta, \Phi) \stackrel{\text{def}}{=} q_\Theta \cdot q_\Phi$$

$$q_\Theta \stackrel{\text{def}}{=} \prod_{\boldsymbol{\theta}_i \in \Theta} q_{\boldsymbol{\theta}_i}(\boldsymbol{\theta}_i \mid \boldsymbol{\mu}_i)$$

$$q_\Phi \stackrel{\text{def}}{=} \prod_{\phi_m \in \Phi} q_{\phi_m}(\tau \mid \boldsymbol{\lambda}_m)$$

variational approximation

Dirichlet densities

Categorical distributions

product of Dirichlet densities

$$p(\Theta \mid \Phi, \mathbb{A}) = \sum_{\tau \in \textsc{Sat}(\Phi, \mathbb{X})} p(\Theta \mid \tau, \mathbb{A}) \cdot P(\tau \mid \Phi, \mathbb{A})$$

posterior

assignments to X that satisfy Φ

Pólya urn

Variational Inference

$$q(\Theta, \Phi) \stackrel{\text{def}}{=} q_\Theta \cdot q_\Phi$$

variational approximation

$$q_\Theta \stackrel{\text{def}}{=} \prod_{\boldsymbol{\theta}_i \in \Theta} q_{\boldsymbol{\theta}_i}(\boldsymbol{\theta}_i \mid \boldsymbol{\mu}_i)$$

Dirichlet densities

$$q_\Phi \stackrel{\text{def}}{=} \prod_{\phi_m \in \Phi} q_{\phi_m}(\tau \mid \boldsymbol{\lambda}_m)$$

Categorical distributions

optimize $\boldsymbol{\mu}_i$

$$q_{\boldsymbol{\theta}_i}(\theta_{i,v}) \propto \boldsymbol{\alpha}_i + \sum_{\phi_m \in \Phi} \left[\sum_{\tau \in \textsc{Sat}(\phi_m \wedge (X_i = v))} q_{\phi_m}(\tau) \right]$$

optimize $\boldsymbol{\lambda}_m$

$$q_{\phi_m}(\tau) \propto \prod_{\boldsymbol{\theta}_i \in \Theta} \exp \mathbb{E}_q \left[\log q_{\boldsymbol{\theta}_i}(\theta_{i,\tau[i]}) \right]$$

Where do the constraints come from?

EMP	ROLE
Ada (x_1)	Lead ($v_{1,1}$)
	Dev ($v_{1,2}$)
	QA ($v_{1,3}$)
Bob (x_2)	Lead ($v_{2,1}$)
	Dev ($v_{2,2}$)
	QA ($v_{2,3}$)

EMP	ROLE
Ada	Lead
Bob	Lead

EMP	ROLE
Ada	Dev
Bob	Lead

EMP	ROLE
Ada	QA
Bob	Lead

EMP	ROLE
Ada	Lead
Bob	Dev

EMP	ROLE
Ada	Dev
Bob	Dev

EMP	ROLE
Ada	QA
Bob	Dev

EMP	ROLE
Ada	Lead
Bob	QA

EMP	ROLE
Ada	Dev
Bob	QA

EMP	ROLE
Ada	QA
Bob	QA

query-answer

"Ada and Bob share the same role"

$$\varphi := (x_1=v_{1,1} \wedge x_2=v_{2,1}) \vee (x_1=v_{1,2} \wedge x_2=v_{2,2}) \vee (x_1=v_{1,3} \wedge x_2=v_{2,3})$$

(1) Probabilistic Programming Datalog

Bárány, Vince, Balder Ten Cate, Benny Kimelfeld, Dan Olteanu, and Zografoula Vagena. "Declarative probabilistic programming with datalog." *ACM Transactions on Database Systems (TODS)* 42, no. 4 (2017): 1-35.

Probabilistic Programming Datalog

$$\mathrm{weather}(\underline{C, T}, w \in \{\mathrm{sun}, \mathrm{rain}\} \sim Cat[\![P]\!]) \leftarrow \mathrm{city}(C, P), \mathrm{ts}(T).$$

Probabilistic Programming Datalog

$$\mathrm{weather}(\underline{C, T}, w \in \{\mathrm{sun}, \mathrm{rain}\} \sim Cat[\![P]\!]) \leftarrow \mathrm{city}(C, P), \mathrm{ts}(T).$$

```
city('Fargo', [.1, .9]), ts('noon')
```

Probabilistic Programming Datalog

$$\text{weather}(\underline{C, T}, w \in \{\text{sun}, \text{rain}\} \sim Cat[\![P]\!]) \leftarrow \text{city}(C, P), \text{ts}(T).$$

```
city('Fargo', [.1, .9]), ts('noon')
```

```
weather('Fargo', 'noon', sun) with prob 0.1

weather('Fargo', 'noon', rain) with prob 0.9
```

Probabilistic Programming Datalog

$$\text{weather}(\underline{C, T}, w \in \{\text{sun}, \text{rain}\} \sim \mathcal{C}at[\![P]\!]) \leftarrow \text{city}(C, P), \text{ts}(T).$$

```
city('Fargo', [.1, .9]), ts('noon')
```

```
weather('Fargo', 'noon', sun) with prob 0.1
weather('Fargo', 'noon', rain) with prob 0.9
```

$$\text{obs}(\underline{\mathit{VarId}, \mathit{ObsId}}, v \in D \sim \mathcal{C}at[\![P]\!]) \leftarrow \text{lp}(\mathit{VarId}, D, P),$$
$$\text{sample}(\mathit{VarId}, \mathit{ObsId}).$$

$$\text{lp}(\underline{\mathit{VarId}}, D, p \in \mathcal{S}_{|D|} \sim \mathcal{D}ir[\![H]\!]) \leftarrow \text{dt}(\mathit{VarId}, D, H).$$

Where do the constraints come from?

(Datalog probabilistic program)

$$
\begin{aligned}
\mathrm{dt}([\mathrm{red}, D], \mathrm{ts}, [1, 1, .., 1]) &\leftarrow \mathrm{d}(D, P, W).\\
\mathrm{dt}([\mathrm{blue}, T], \mathrm{ws}, [1, 1, .., 1]) &\leftarrow \mathrm{t}(T).\\
\mathrm{sample}([\mathrm{red}, D], P) &\leftarrow \mathrm{d}(D, P, W).\\
\mathrm{sample}([\mathrm{blue}, T], [D, P]) &\leftarrow \mathrm{d}(D, P, W), \mathrm{obs}([\mathrm{red}, D], P, T).\\
\mathrm{qa}^*(D, P, W) &\leftarrow \mathrm{d}(D, P, W), \mathrm{obs}([\mathrm{blue}, T], [D, P], W).
\end{aligned}
$$

APACHE ARROW Acero

(ground constraints)

$$
\begin{aligned}
&(x_{0,0}[s]=v_0 \wedge x_{0,1}[n]=v_0) \vee (x_{0,0}[s]=v_1 \wedge x_{0,1}[n]=v_1)\\
&(x_{0,0}[e]=v_0 \wedge x_{1,0}[w]=v_0) \vee (x_{0,0}[e]=v_1 \wedge x_{0,1}[w]=v_1)\\
&(x_{1,0}[s]=v_0 \wedge x_{1,1}[n]=v_0) \vee (x_{1,0}[s]=v_1 \wedge x_{1,1}[n]=v_1)\\
&(x_{0,1}[e]=v_0 \wedge x_{1,1}[w]=v_0) \vee (x_{0,1}[e]=v_1 \wedge x_{1,1}[w]=v_1)\\
&(x_{2,0}[s]=v_0 \wedge x_{2,1}[n]=v_0) \vee (x_{2,0}[s]=v_1 \wedge x_{2,1}[n]=v_1)\\
&\ldots\\
&\ldots
\end{aligned}
$$

(many)

LLVM/ClangJIT

Inference in Action

(a) Noisy Image (b) Gibbs sampler denoising (c) Variational inference denoising

Thank you!

(Questions?)