Course Review

Xin Zhang
Peking University

Final Information

- When: 18:40-20:40, June 17
- Where: 117
- Content: From Lecture 6 onwards (including Lecture 6, semantics)
- Form: Allowed to bring two sheets of A4-size/letter-size papers (4 pages)
  - No keys to past exams
  - Shouldn't be print-out of all slides

Topics after Midterm

Probabilistic Programming

- Semantics
- Inference
- Logic programming
- Deep probabilistic programming

Broader AI

- Causal inference
- Explainable AI
- Constrained LLMs

Semantics of Probabilistic Programming

- Formal tools to reason about properties of a program
  - What is the probability that the postcondition is satisfied?
  - What is the probability that this program halts on all inputs?
  - What is the probability that it halts in polynomial time?

Semantics of Probabilistic Programming

- Operational semantics
  - Model the step-by-step executions of a program on an abstract machine
- Denotational semantics
  - Link program concepts with math concepts: measure theory

Operational Semantics: Example

```
x :=0
while x == 0 do
        x:=coin()
```

What is the probability that the program halts?

$$(x \ := 0 \ ; \ e, s, m, p) \xrightarrow{*} (e, s[x \mapsto 0], m, p)$$

$$(e, s[x \mapsto 0], m, p) \xrightarrow{*} (x := \texttt{coin()} \ ; \ e, s[x \mapsto 0], m, p)$$

$$(x \ := \texttt{coin()} \ ; \ e, s[x \mapsto 0], m, p) \xrightarrow{*} (e, [s \mapsto \mathsf{hd}\, m], \mathsf{tl}\, m, p).$$

$$hd(m_1 m_2 \ldots) = m_1$$
$$tl(m_1 m_2 \ldots) = m_2 \ldots$$

The loop continues until it reaches m inf the form of $1m'$

$$(e, s[x \mapsto 1], m', p) \xrightarrow{*} (\texttt{skip}, s[x \mapsto 1], m', p)$$

$$(x \ := 0 \ ; \ e, s, m, p) \xrightarrow{*} (\texttt{skip}, s[x \mapsto 1], m', p)$$

Measurable Spaces and Measures

- (**S**, **B**) is a measurable space
  - **S** is a set
  - **B** is a σ-algebra on **S**, which is a collection of subsets of **S**
    - It contains $\emptyset$
    - Closed under complementation in **S**
    - Closed under countable union
  - The elements of **B** are called measurable sets

- If **F** is a collection of subsets of **S**, $\sigma(\boldsymbol{F})$ is the smallest σ-algebra containing **F**, or $\sigma(\mathcal{F}) \triangleq \bigcap\{\mathcal{A} \mid \mathcal{F} \subseteq \mathcal{A} \textbf{ and } \mathcal{A} \textbf{ is a } \sigma\textbf{-algebra}\}$. We say (S, $\sigma(\boldsymbol{F})$) is generated by **F**.

Denotational Semantics: Example

- $(\boldsymbol{S}, \boldsymbol{B_S})$: x = uniform(0.1, 1.1) $\mu([a,b]) = \mathrm{length}([\mathrm{a},\mathrm{b}] \cap [0.1,1.1])$

- $(\boldsymbol{T}, \boldsymbol{B_T})$: y = uniform(0,x)

- Markov kernel $P(x, \bigcup_{i=1}^{i=M}[a_i, b_i]) = \sum_{i=1}^{i=M} length([a_i, b_i] \cap [0,x])/x$

- μ's pushforward under P is

$$P_*(\mu)(B_T) = \int_{x \in [0.1,1.1]} B_T \cap [0,x] * \mu(dx)$$

Example Question

- What is right about denotational semantics (DS) and operational semantics (OS)?

A. DS can reason about nonterminating programs but OS cannot

B. DS reason about all executions of the program together while OS reason about one execution at a time

C. DS links program concepts with integers

D. OS uses real random numbers

Inference in Probabilistic Programming

- Graph-based inference
  - Compilation-based
  - Can work only with bounded programs

- Evaluation-based inference
  - Evaluation-based
  - Can work with any program

- Important general inference algorithms
  - Hamiltonian Monte-Carlo
  - Sequential Monte-Carlo

Graph-Based Inference: Translation

$$\rho, \phi, G, e \Downarrow \rho', \phi', G'$$

- ρ: environment, which maps a variable to a constant or a node variable
- ϕ: path condition
- e: program

Graph-Based Inference: Example

```
x = guassian(0, 1)
y = uniform(0, x)
if (x>10){
        condition(y >1.5)
}
else{
        condition(y<0.5)
}
```

Example Question

- What kind of programs can graph-based inference handle?
  A. A sorting program that can work with any array of integers
  B. A reactive program that continuously monitors the room temperature
  C. A program that schedules courses of the university
  D. A program that computes Pi with any given precision

Metropolis-Hasting: Single-Site Proposals

- Map $\sigma(X)$, such that $X(x)$ refers to the value of x (only variables in the current execution)

- Map $\sigma(logP)$, where $logP(v)$ evaluates the density for each variable
  - When sampling from a distribution d, we have
$$\sigma\big(logP(x)\big) = LOG - PROB(d, X(x))$$
  - When encounter $condition(b)$, we have
$$\sigma\big(logP(y)\big) = LOG - PROB(b, true)$$

Metropolis-Hasting: Single-Site Proposals

- Pick a variable $x_0 \in dom(X)$ at a random from the current sample

- Construct a proposal X', P' by re-running the program
  - For an expression d that sample from a variable x
    - If $x == x_0$, or $x \notin dom(X)$, then samples from the expression. Otherwise, reuse the value $X'(x) \leftarrow X(x)$
    - Calculate the probability $P'(x) \leftarrow PROB(d, X'(x))$
  - For expression $condition(b)$ with variable y:
    - Calculate the probability $P'(y) \leftarrow PROB(b, y) = 1_{[b==y]}$
  - For expression $observe(e, v)$ with variable y:
    - Calculate the probability $P'(y) \leftarrow PROB(e, v)$

Metropolis-Hasting: Single-Site Proposals

$$\alpha = \frac{|\mathrm{dom}(\mathcal{X})|}{|\mathrm{dom}(\mathcal{X}')|} \frac{\prod_{y \in \mathcal{Y}} \mathcal{P}'(y) \prod_{x \in X'^{\mathrm{reused}}} \mathcal{P}'(x)}{\prod_{y \in \mathcal{Y}} \mathcal{P}(y) \prod_{x \in X^{\mathrm{reused}}} \mathcal{P}(x)}$$

Example Question

- In each iteration of single-site proposals, only the value of the chosen variable need to be sampled, while the values of all the other variables are reused from the last iteration. Is the statement correct?

Hamiltonian Monte Carlo (HMC)

- Augment distribution $p(\boldsymbol{z})$ with $p(\boldsymbol{z}, \boldsymbol{r})$

- Proposal distribution:
  - Update $\boldsymbol{z}, \boldsymbol{r}$ using Hamiltonian dynamics (in practice, a discretized approximation called leapfrog integration)
  - Update $\boldsymbol{r}$ stochastically

- Acceptance probability (After applying Hamiltonian dynamics):

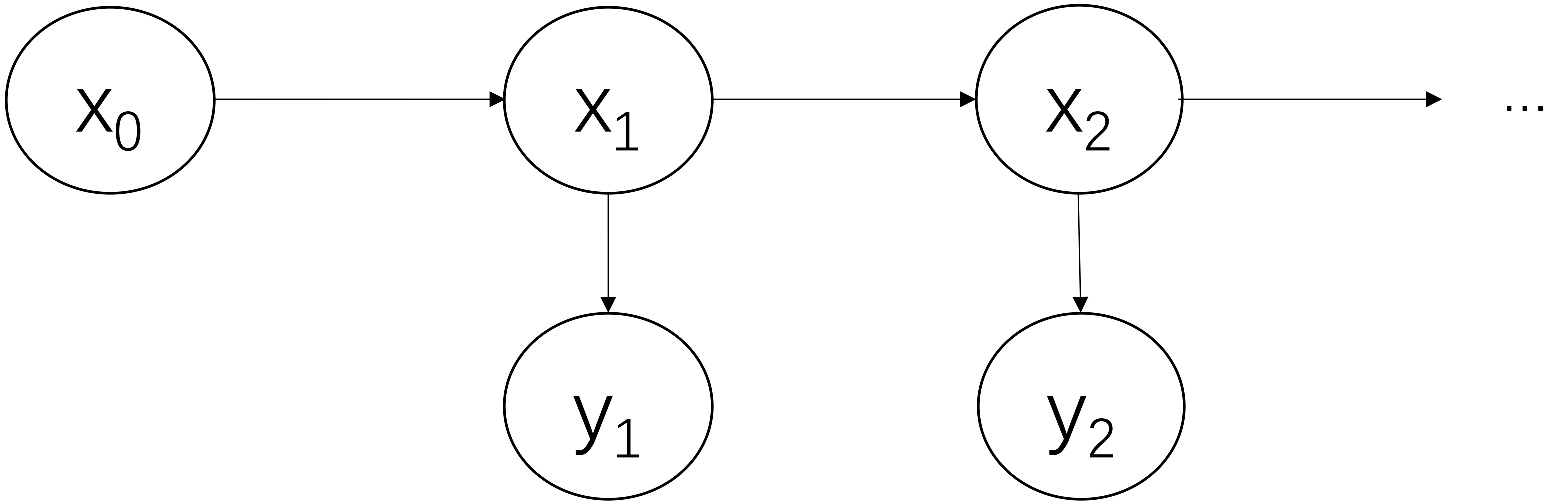
$$\min\left(1, \exp\{H(\mathbf{z}, \mathbf{r}) - H(\mathbf{z}^\star, \mathbf{r}^\star)\}\right)$$

Account for approximation

Example Question

- During HMC, ignoring numerical issues and precision issues, the total energy of the system doesn't change. Is the statement right?

Sequential Monte Carlo



Given

$p(x_0)$ and
$p(x_t|x_{t-1})$ and
$p(y_t|x_t)$ and
Observations $y_{1:t}$

Estimate

$p(x_{0:t}|y_{1:t})$ or
$p(x_t|y_{1:t})$ or
$I(f_t) = E_{p(x_{0:t}|y_{1:t})}[f_t(x_{0:t})] = \int f_t(x_{0:t})p(x_{0:t}|y_{1:t})dx_{0:t}$

SMC: Bootstrap Filter

Assume the proposal distribution is $p(x_{1:t})$

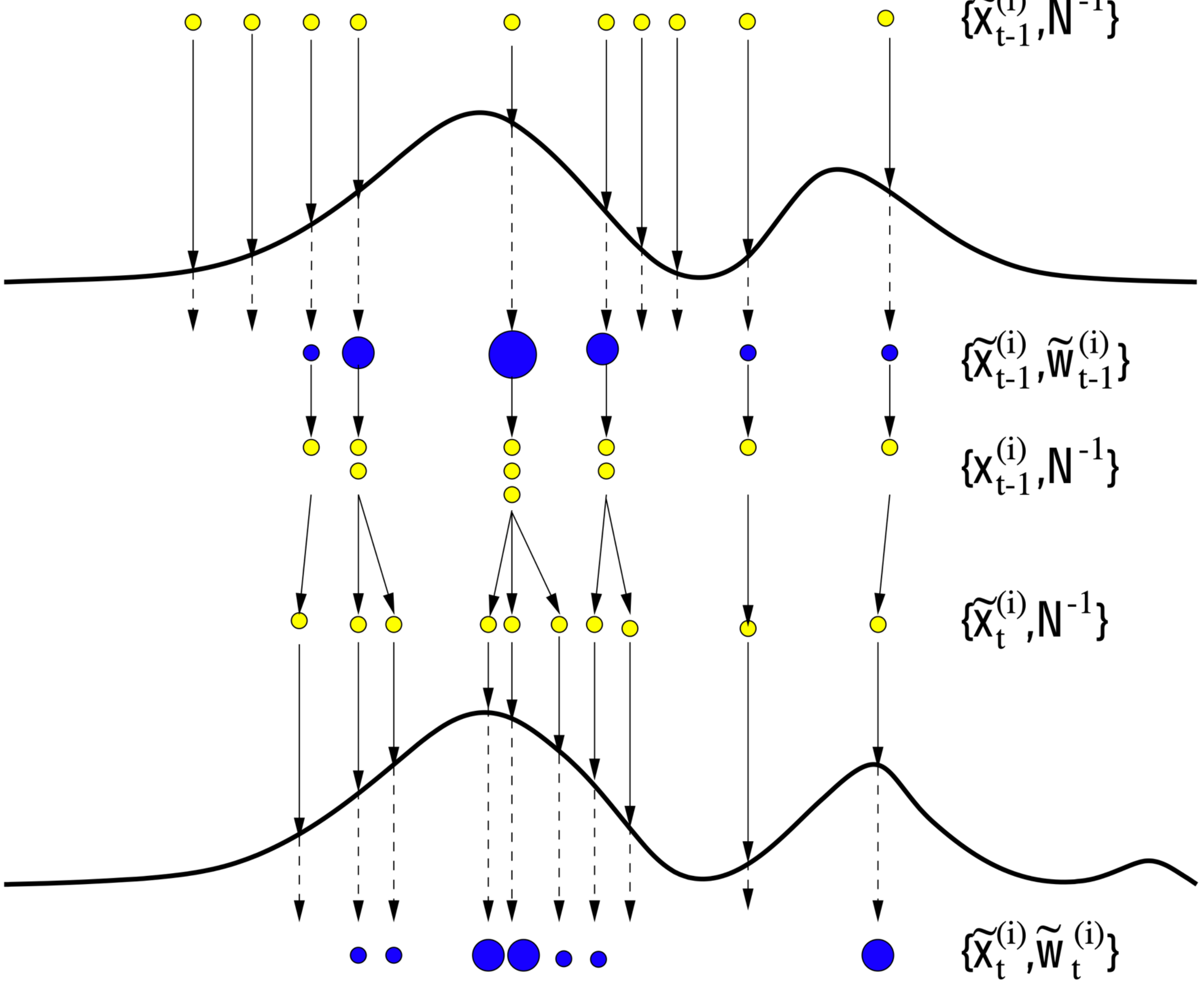
1. Initialization. t = 0
   - For i = 1,…,N, sample $x_0^{(i)} \sim p(x_0)$ and set $t = 1$
2. Importance sampling step.
   - For sample $\tilde{x}_t^{(i)} \sim p(x_t | \tilde{x}_{t-1}^{(i)})$ and set $(\tilde{x}_{0:t-1}^{(i)}, \tilde{x}_t^{(i)})$.
   - For i = 1,…,N, evaluate the importance weights.
   - Normalize the importance weights
3. Selection step
   - Resample with replacement N particles from the current particles according to importance weights
   - Set $t \to t + 1$

Bootstrap Filter: Example

$$
\begin{aligned}
x_t &= \frac{1}{2}x_{t-1} + 25\frac{x_{t-1}}{1+x_{t-1}^2} + 8\cos(1.2t) + v_t \\
y_t &= \frac{x_t^2}{20} + w_t,
\end{aligned}
$$

$$x_1 \sim N(0,10), v_k \sim N(0,10), w_k \sim N(0,1)$$

From "An Introduction to Sequential Monte Carlo Methods" by Arnaud Doucet, Nando De Freitas, and Neil Gordon i=1,...,N=10 particles



From "An Introduction to Sequential Monte Carlo Methods" by Arnaud Doucet, Nando De Freitas, and Neil Gordon

Example Question

- Is SMC a variant of importance sampling?

Probabilistic Logic Programming: Unifying Logic and Probability

- Logic: the ability to describe complex domains concisely in terms of objects and relations

- Probability: the ability to handle uncertainty

- Logic + probability = Probabilistic Logic Programming

Problog: Syntax

Queries:

```
0.5::heads(C).
two_heads :- heads(c1), heads(c2).
query(two_heads).
```

```
0.5::heads(C) :- between(1, 4, C).
query(heads(C)).
```

Evidence:

```
0.5::heads(C).
two_heads :- heads(c1), heads(c2).
evidence(\+ two_heads).
query(heads(c1)).
```

From the documentation of Problog

Semantics of Problog

- From a Problog program, we can sample a Datalog program by sampling the facts

```
0.5 :: stayUp.
0.7 :: drinkCoffee :- stayUp.
0.3 :: fallSleep :- drinkCoffee, stayUp.
```

=

```
0.5 :: stayUp.
0.7 :: r1.
0.3 :: r2.
drinkCoffee :- stayUp, r1.
fallSleep :- drinkCoffee, stayUp, r2.
```

sample →

```
stayUp.
r1.
r2.
drinkCoffee :- stayUp, r1.
fallSleep :- drinkCoffee, stayUp, r2.
```

Probability: 0.5*0.7*0.3

Semantics of Problog

- What about queries?

```
0.5 :: stayUp.
0.7 :: r1.
0.3 :: r2.
drinkCoffee :- stayUp, r1.
fallSleep :- drinkCoffee, stayUp, r2.

query(fallSleep)
```

A query calculates a marginal probability of a fact. Informally,

$$p(f) = \frac{\sum p(any\ program\ that\ derives\ f)}{\sum p(any\ program)}$$

Semantics of Problog

- What about evidence?

```
0.5 :: stayUp.
0.7 :: r1.
0.3 :: r2.
drinkCoffee :- stayUp, r1.
fallSleep :- drinkCoffee, stayUp, r2.

evidence(\+ fallSleep)
query(stayUp)
```

Evidence filters out certain programs. Informally,

$$p(f) = \frac{\sum p(any\ program\ that\ derives\ f|evidence)}{\sum p(any\ program|evidence)}$$

Semantics of Problog

- What about relations and quantified variables?

```
0.9 :: edge(0,1).
0.8 :: edge(1,2).
0.7 :: edge(2,3).
0.8 :: edge(2,4).

path(A,B) :- edge(A,B).
0.8 :: path(A,C) :- path(A,B), edge(B,C).

evidence(\+ path(0,3)).

query(path(0,4)).
```

Semantics of Problog

- Move probabilities to facts

```
0.9 :: edge(0,1).
0.8 :: edge(1,2).
0.7 :: edge(2,3).
0.8 :: edge(2,4).
0.8 :: r(A,B,C).

path(A,B) :- edge(A,B).
path(A,C) :- path(A,B), edge(B,C), r(A,B,C).

evidence(\+ path(0,3)).

query(path(0,4)).
```

Semantics of Problog

- Ground

  Constants: 0, 1, 2, 3 4

  path(A,C) :- path(A,B), edge(B,C), r(A,B,C).

  Generates

  path(0,0) :- path(0,0), edge(0,0), r(0,0,0). A=0, B=0, C=0

  path(0,1) :- path(0,0), edge(0,1), r(0,0,1). A=0, B=0, C=1

  path(0,1) :- path(0,0), edge(0,1), r(0,0,1). A=0, B=0, C=1

  …

Semantics of Problog

- After grounding, each ground term can be seen as a Boolean variable, then the whole program can be solved using the semantics of the Boolean case

path(0,0) -> t1, edge(0,0) -> t2, r(0,0,0) -> t3

path(0,0) :- path(0,0), edge(0,0), r(0,0,0).

t1 :- t1,t2,t3

Example using MaxSAT for Inference

0.6 :: rain.

0.5 :: sprinkle.

0.9 :: grass_wet :- rain, sprinkle.

ln0.6 rain

ln0.4 !rain

ln0.5 sprinkle

ln0.5 !sprinkle

ln0.9 r

ln0.1 !r

grass_wet or !rain or !sprinkle or !r

!grass_wet or rain

!grass_wet or sprinle

!grass_wet or r

grass_wet :- rain, sprinkle is translated into

$$grass_{wet} \leftrightarrow rain \land sprinkle \land r$$

Example using WMC for Inference

0.6 rain

0.4 !rain

0.5 sprinkle

0.5 !sprinkle

0.9 r

0.1 !r

grass_wet or !rain or !sprinkle

!grass_wet or rain

!grass_wet or sprinle

w(rain =true) = 0.6
w(rain =false) = 0.4
w(sprinkle = true) = 0.5
w(sprinkle = false) = 0.5
w(r = true) = 0.9
w(r = false) = 0.1

P(grass_wet = true) = WMC(M∧grass_wet=true)

What if we want to evaluate
P(rain | grass_wet = true)?

Using WMC for Marginal Inference

- Let the constructed weighted formula be M, queries be Q, evidence be E, then

$$P(Q) = \frac{WMC(M \land Q \land E)}{WMC(M \land E)}$$

- For more, refer to

Sang, T., Beame, P. and Kautz, H., 2005. Solving Bayesian networks by weighted model counting. In Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05) (Vol. 1, pp. 475-482). AAAI Press.

Example Question

- Is the inference problem of Problog a NP hard problem? Is it decidable?

How about combing PP with DL?

- Making neural networks Bayesian
  - Bayesian neural networks
- Using neural networks to compute probabilistic programs
  - Edwards
- Treat neural networks as input to probabilistic programs
  - Neural-symbolic programming

Neurosymbolic Programs

Symbolic Programs

Interpretable

Verifiable

Structured domain knowledge

Data efficient

Neural Networks

Scalable algorithms

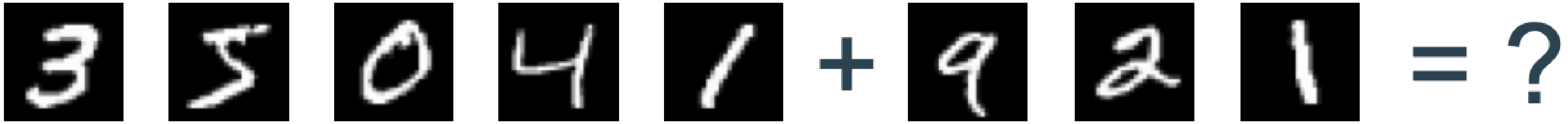
Flexible

Handles messy data

Easy to get started

Slide by Chaudhuri, Sun, Solar-Lezama

Example Task: MNIST Addition



What if we only labeled sums, not single digits?

DeepProbLog Program for MNIST Addition

```
nn(m_digit,[X],Y,[0,1,2,3,4,5,6,7,8,9]) :: digit(X,Y).

addition(X,Y,Z) :- digit(X,X2), digit(Y,Y2),digit(Z,Z2), Z2 is X2+Y2.
```

Neural Annotated Disjunctions

Keyword | Neural Network | Input Variables | Output Variable | Output Domain

nn(m_digit, [X], Y, [0, . . . , 9]) :: digit(X, Y).   Neural Annotated Disjunction

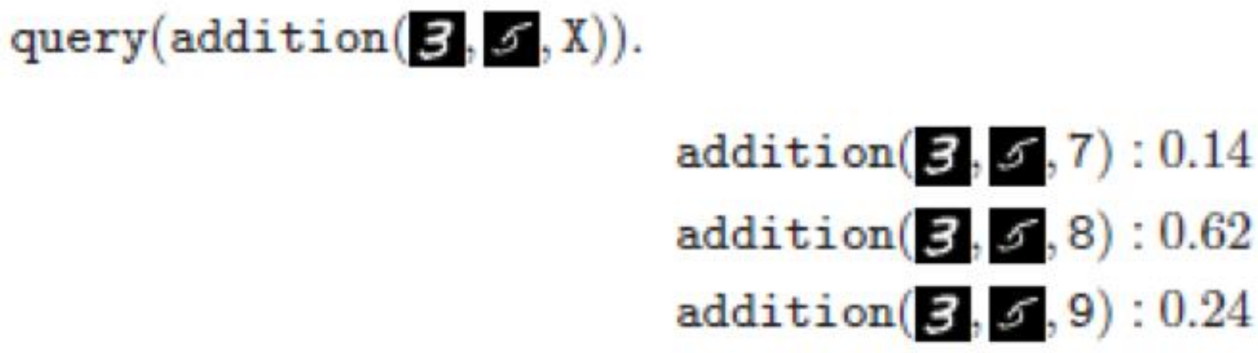
nn(m_digit, [], 0) :: digit(, 0) ; . . . ; nn(m_digit, [], 9) :: digit(, 9).   Grounded Neural Annotated Disjunction

p_0 :: digit(, 0) ; . . . ; p_9 :: digit(, 9).   Grounded Annotated Disjunction

DeepProbLog Program for MNIST Addition

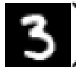
nn(m_digit,[X],Y,[0,1,2,3,4,5,6,7,8,9]) :: digit(X,Y).

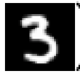
addition(X,Y,Z) :- digit(X,X2), digit(Y,Y2),digit(Z,Z2), Z2 is X2+Y2.



Neural Facts

$$\texttt{nn}(\texttt{m}, [\texttt{X}, \texttt{Y}]) :: \texttt{similar}(\texttt{X}, \texttt{Y}).$$

nn(m, [, ]) :: similar(, ).

p :: similar(, ).

Learning of DeepProblog: Problem

Definition 5

Learning from entailment Given a DeepProbLog program with parameters Θ, a set $\mathcal{Q}$ of pairs (q, p) with q a query and p its desired success probability, and a loss function $\mathcal{L}$, compute:

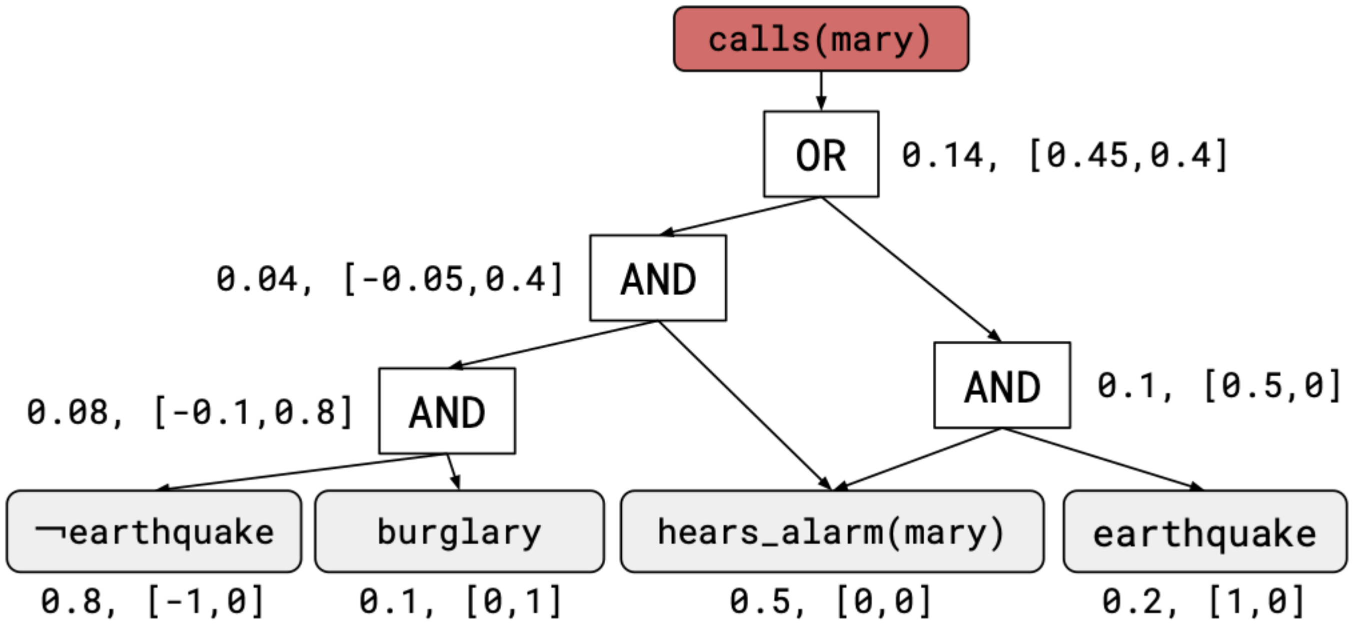
$$\arg\min_{\Theta} \frac{1}{|\mathcal{Q}|} \sum_{(q,p)\in\mathcal{Q}} \mathcal{L}(P(q|\Theta), p)$$

Assuming desired probability $p = 1$, the problem reduces to

$$\arg\min_{\Theta} \frac{1}{|\mathcal{Q}|} \sum_{(q,p)\in\mathcal{Q}} -\log P_{\Theta}(q)$$

Gradient Descent in Problog

```
0.2::earthquake.
0.1::burglary.
0.5::hears_alarm(mary).
0.4::hears_alarm(john).
alarm :- earthquake.
alarm :- burglary.
calls(X):-alarm,hears_alarm(X).
```



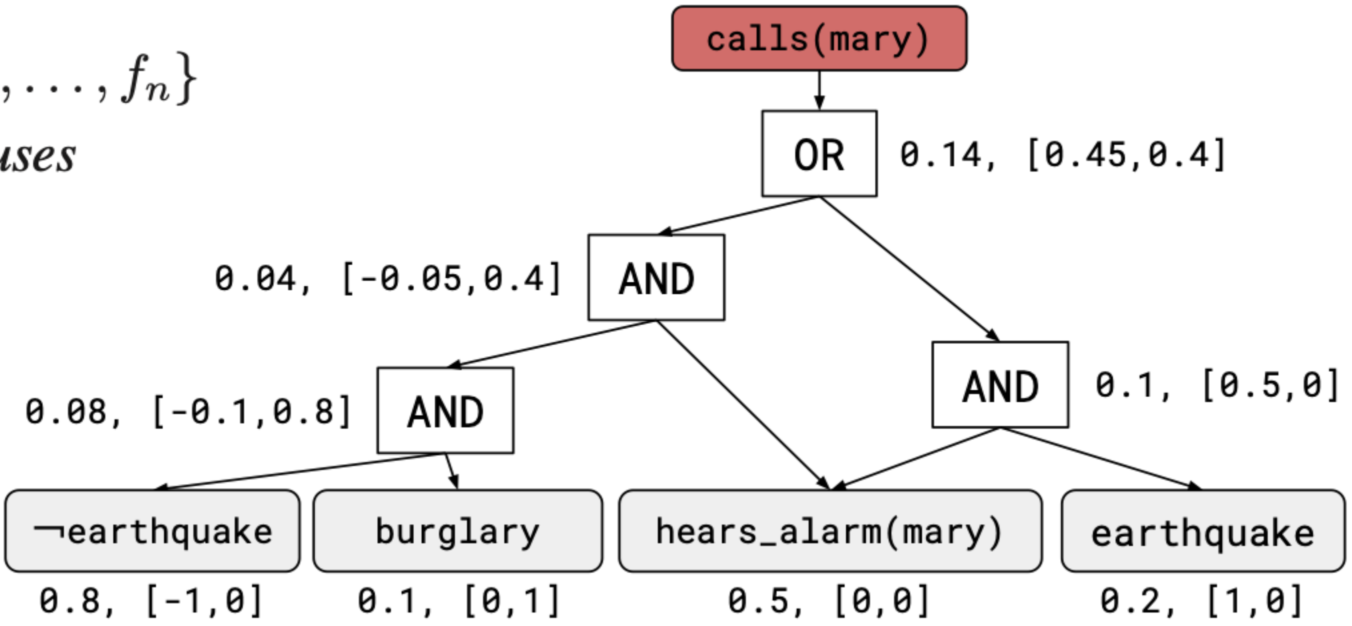
$$\arg\min_{\Theta} \frac{1}{|\mathcal{Q}|} \sum_{(q,p)\in\mathcal{Q}} -\log P_{\Theta}(q)$$

Algebraic Prolog

An algebraic Prolog (aProbLog) program consists of

- a *commutative semiring* $(\mathcal{A}, \oplus, \otimes, e^{\oplus}, e^{\otimes})$[1]
- a finite set of ground *algebraic facts* $\mathrm{F} = \{f_1, \ldots, f_n\}$
- a finite set BK of *background knowledge clauses*
- a *labeling function* $\alpha : \mathrm{L}(\mathrm{F}) \rightarrow \mathcal{A}$

$$(a_1, \vec{a_2}) \oplus (b_1, \vec{b_2}) = (a_1 + b_1, \vec{a_2} + \vec{b_2})$$
$$(a_1, \vec{a_2}) \otimes (b_1, \vec{b_2}) = (a_1 b_1, b_1 \vec{a_2} + a_1 \vec{b_2})$$
$$e^{\oplus} = (0, \vec{0})$$
$$e^{\otimes} = (1, \vec{0})$$

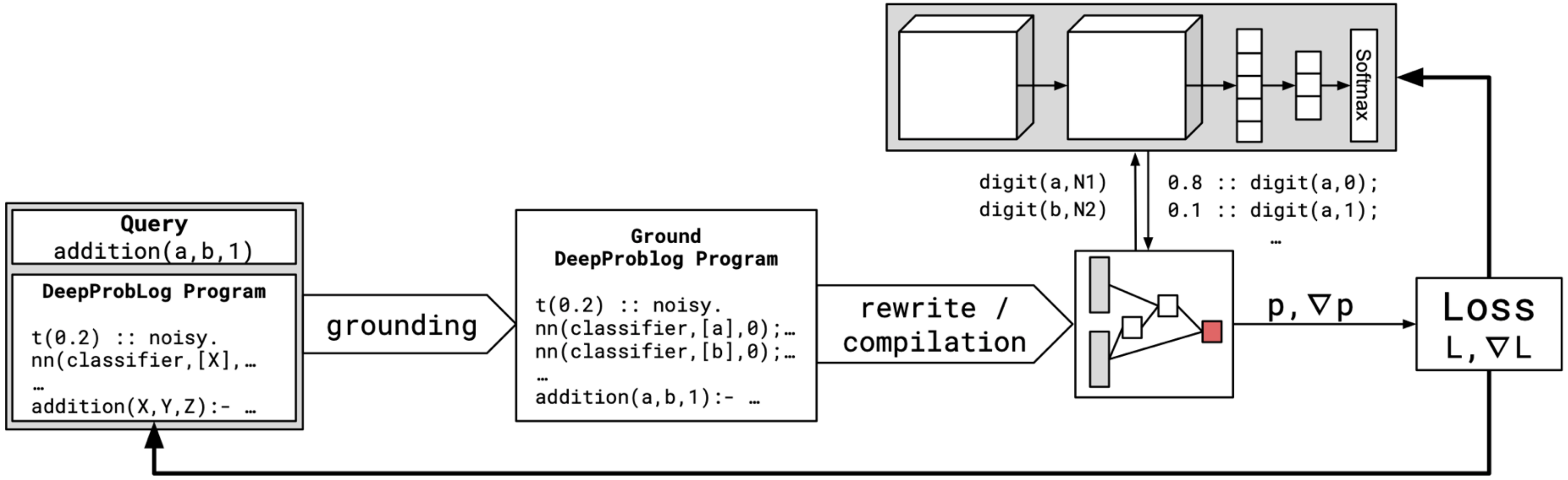


Algebraic Circuit
(Support Efficient Inference based on BDD)

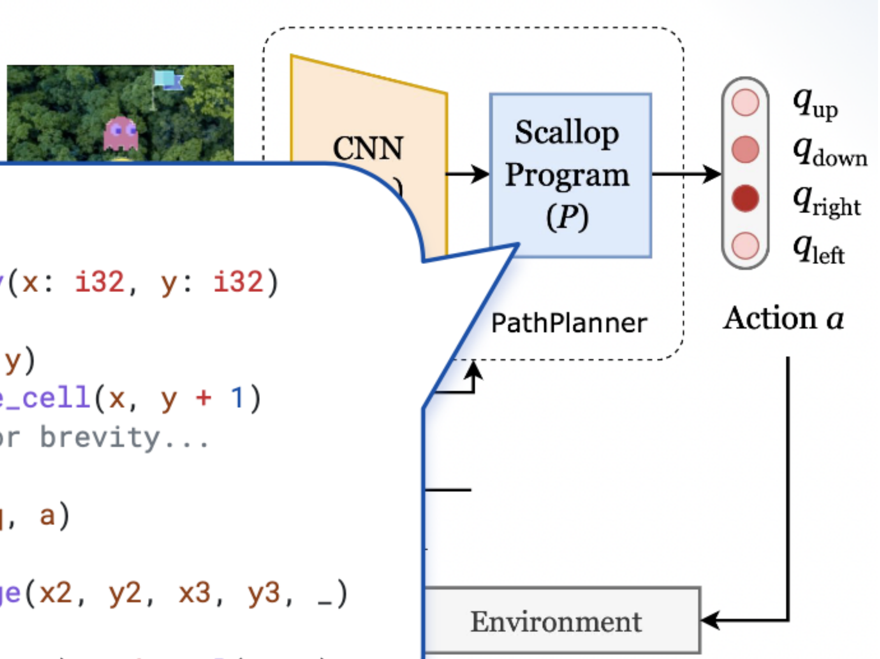
Gradient Descent in DeepProbLog

$$\frac{dz}{dx} = \frac{dz}{dy} \cdot \frac{dy}{dx}$$

$$\frac{d\mathcal{L}}{d\theta_k} = \frac{\partial \mathcal{L}}{\partial P(q)} \sum_i \frac{\partial P(q)}{\partial \hat{p}_i} \frac{\partial \hat{p}_i}{\partial \theta_k}$$



A Motivating Example for Scallop




```
type Action = UP | RIGHT | DOWN | LEFT
type actor(x: i32, y: i32), goal(x: i32, y: i32), enemy(x: i32, y: i32)

rel safe_cell(x, y) = grid_cell(x, y) and not enemy(x, y)
rel edge(x, y, x, y + 1, UP) = safe_cell(x, y) and safe_cell(x, y + 1)
// Rules for RIGHT, DOWN, and LEFT edges are omitted for brevity...

rel next_pos(p, q, a) = actor(x, y) and edge(x, y, p, q, a)
rel path(x, y, x, y) = next_pos(x, y, _)
rel path(x1, y1, x3, y3) = path(x1, y1, x2, y2) and edge(x2, y2, x3, y3, _)

rel next_action(a) = next_pos(p, q, a) and path(p, q, r, s) and goal(r, s)
```

Semantics and Provance Framework

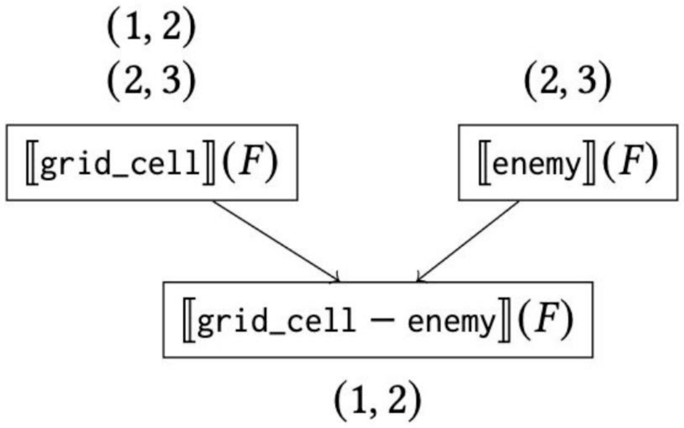
- The formal semantics of SCLRAM is parameterized by a provenance structure inspired by the theory of Provenance Semirings [PODS'07]
- A Provenance Structure is an algebraic structure that specifies:
  - Tag Space: the space of additional information associated with each tuple
  - Operations: how tags propagate during execution

	Abstract Provenance			max-min-prob(mmp)
(Tag Space)	t	$\in$	T	$[0, 1]$
(False)	$\mathbf{0}$	$\in$	T	0
(True)	$\mathbf{1}$	$\in$	T	1
(Disjunction)	$\oplus$	:	$T \times T \to T$	max
(Conjunction)	$\otimes$	:	$T \times T \to T$	min
(Negation)	$\ominus$	:	$T \rightharpoonup T$	$\lambda p.(1 - p)$
(Saturation)	$\circleddash$	:	$T \times T \to \mathrm{Bool}$	==

Semantics and Provance Framework

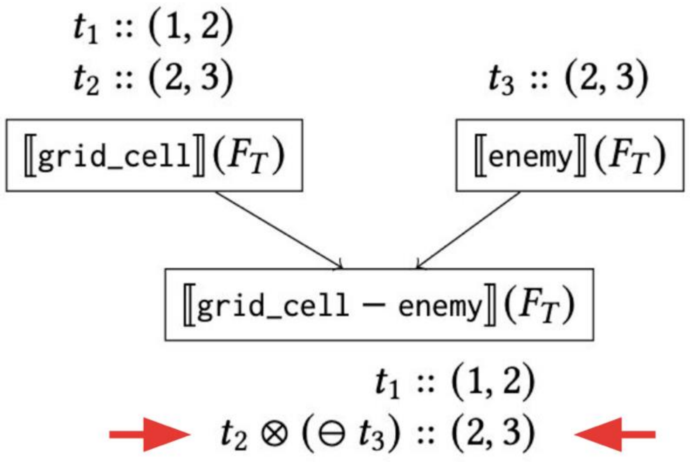
Scallop program `rel safe_cell(x, y) = grid_cell(x, y) and not enemy(x, y)`

SclRam program `safe_cell ← grid_cell - enemy`



Untagged Semantics

vs.

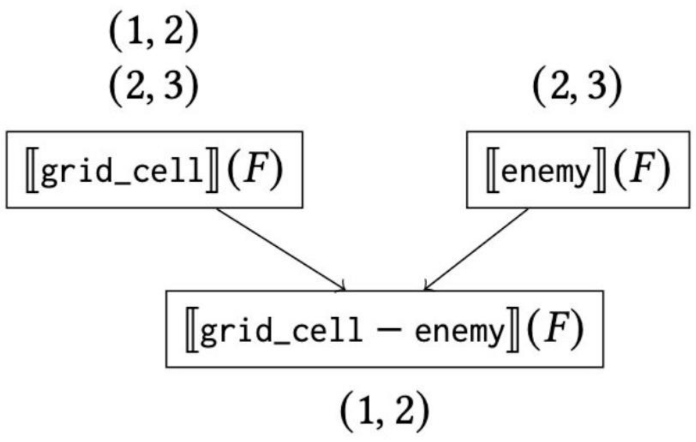


Tagged Semantics

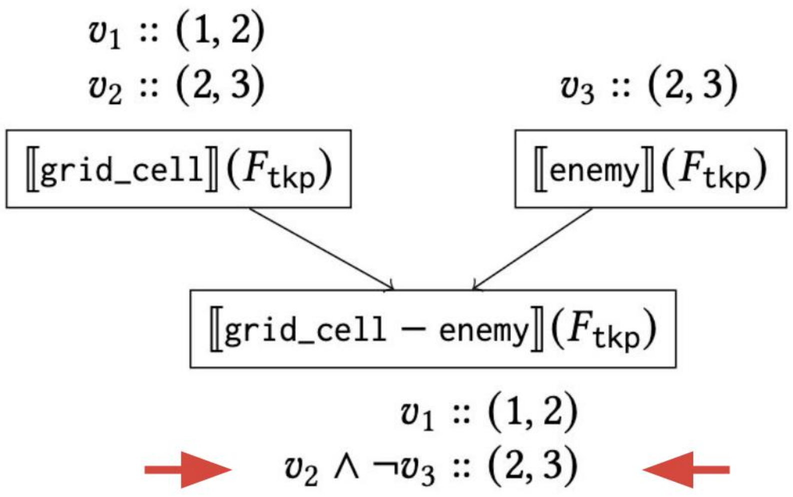
Semantics and Provance Framework

Scallop program `rel safe_cell(x, y) = grid_cell(x, y) and not enemy(x, y)`

SclRam program `safe_cell ← grid_cell - enemy`



vs.



Untagged Semantics

Tagged Semantics with `top-k-proofs`

Semantics and Provance Framework

Scallop program `rel safe_cell(x, y) = grid_cell(x, y) and not enemy(x, y)`

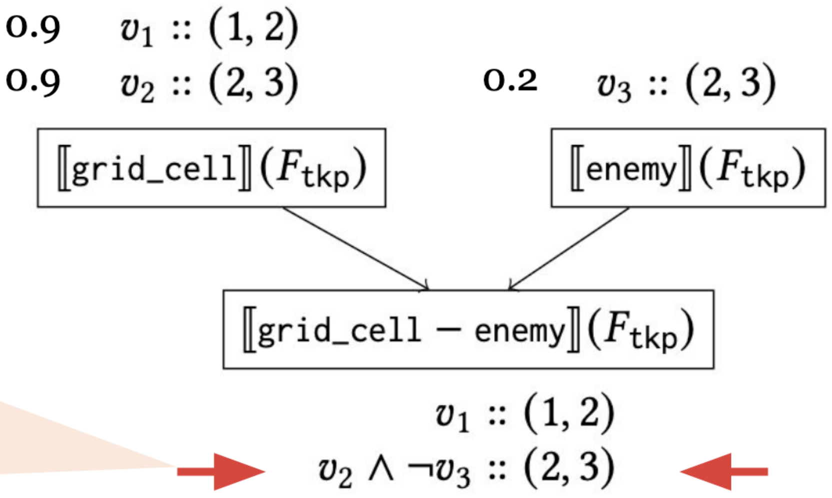
SclRam program `safe_cell ← grid_cell - enemy`

Recover Probability from Bool Formula

Using Weighted Model Counting (WMC)

$$\Pr(v_1) = 0.9, \Pr(v_2) = 0.9, \Pr(v_3) = 0.2$$

$$\Pr(v_2 \wedge \neg v_3) = \Pr(v_2) \cdot (1 - \Pr(v_3)) = 0.9 \cdot (1 - 0.2) = 0.72$$



Tagged Semantics with `top-k-proofs`

Built-in Library of Provenance Structures

Kind	Provenance	T	$\mathbf{0}$	$\mathbf{1}$	$\oplus$	$\otimes$	$\ominus$	$\circleddash$	τ	ρ
Discrete	`unit`	$\{()\}$	$()$	$()$	$\lambda t_1, t_2.()$	$\lambda t_1, t_2.()$	$\lambda a.\text{FAIL}$	$==$	$\lambda i.()$	$\lambda t.()$
	`bool`	$\{\top, \bot\}$	$\bot$	$\top$	$\vee$	$\wedge$	$\neg$	$==$	id	id
	`natural`	$\mathbb{N}$	0	1	$+$	$\times$	$\lambda n.\mathbb{1}[n > 0]$	$==$	id	id
Probabilistic	`max-min-prob`	$[0, 1]$	0	1	max	min	$\lambda t.1 - t$	$==$	id	id
	`add-mult-prob`	$[0, 1]$	0	1	$\lambda t_1, t_2.\text{clamp}(t_1 + t_2)$	$\lambda t_1, t_2.(t_1 \cdot t_2)$	$\lambda t.1 - t$	$\lambda t.\top$	id	id
	`nand-min-prob`	$[0, 1]$	0	1	$\lambda t_1, t_2. -(1 - t_1)(1 - t_2)$	min	$\lambda t.1 - t$	$\lambda t.\top$	id	id
	`nand-mult-prob`	$[0, 1]$	0	1	$\lambda t_1, t_2. -(1 - t_1)(1 - t_2)$	$\lambda t_1, t_2.t_1 \cdot t_2$	$\lambda t.1 - t$	$\lambda t.\top$	id	id
	`top-k-proofs`	Φ	$\emptyset$	$\{\emptyset\}$	$\vee_{\text{top-}k}$	$\wedge_{\text{top-}k}$	$\neg_{\text{top-}k}$	$==$	$\lambda p_i.\{\{\text{pos}(i)\}\}$	$\lambda \varphi.\text{WMC}(\varphi, \Gamma)$
	`sample-k-proofs`	Φ	$\emptyset$	$\{\emptyset\}$	$\vee_{\text{sample-}k}$	$\wedge_{\text{sample-}k}$	$\neg_{\text{sample-}k}$	$==$	$\lambda p_i.\{\{\text{pos}(i)\}\}$	$\lambda \varphi.\text{WMC}(\varphi, \Gamma)$
Differentiable	`diff-max-min-prob`	$\mathbb{D}$	$\hat{0}$	$\hat{1}$	max	min	$\lambda \hat{t}.\hat{1} - \hat{t}$	$==$	id	id
	`diff-add-mult-prob`	$\mathbb{D}$	$\hat{0}$	$\hat{1}$	$\lambda \hat{t}_1, \hat{t}_2.\text{clamp}(\hat{t}_1 + \hat{t}_2)$	$\lambda \hat{t}_1, \hat{t}_2.\hat{t}_1 \cdot \hat{t}_2$	$\lambda \hat{t}.\hat{1} - \hat{t}$	$\lambda \hat{t}.\top$	id	id
	`diff-nand-min-prob`	$[\hat{0}, \hat{1}]$	$\hat{0}$	$\hat{1}$	$\lambda \hat{t}_1, \hat{t}_2. -(\hat{1} - \hat{t}_1)(\hat{1} - \hat{t}_2)$	min	$\lambda \hat{t}.\hat{1} - \hat{t}$	$\lambda \hat{t}.\top$	id	id
	`diffnand-mult-prob`	$[\hat{0}, \hat{1}]$	$\hat{0}$	$\hat{1}$	$\lambda \hat{t}_1, \hat{t}_2. -(\hat{1} - \hat{t}_1)(\hat{1} - \hat{t}_2)$	$\lambda \hat{t}_1, \hat{t}_2.\hat{t}_1 \cdot \hat{t}_2$	$\lambda \hat{t}.\hat{1} - \hat{t}$	$\lambda \hat{t}.\top$	id	id
	`diff-top-k-proofs`	Φ	$\emptyset$	$\{\emptyset\}$	$\vee_{\text{top-}k}$	$\wedge_{\text{top-}k}$	$\neg_{\text{top-}k}$	$==$	$\lambda \hat{p}_i.\{\{\text{pos}(i)\}\}$	$\lambda \varphi.\text{WMC}(\varphi, \hat{\Gamma})$
	`diff-sample-k-proofs`	Φ	$\emptyset$	$\{\emptyset\}$	$\vee_{\text{sample-}k}$	$\wedge_{\text{sample-}k}$	$\neg_{\text{sample-}k}$	$==$	$\lambda \hat{p}_i.\{\{\text{pos}(i)\}\}$	$\lambda \varphi.\text{WMC}(\varphi, \hat{\Gamma})$
…	…	…	…	…	…	…	…	…	…	…

Example Question

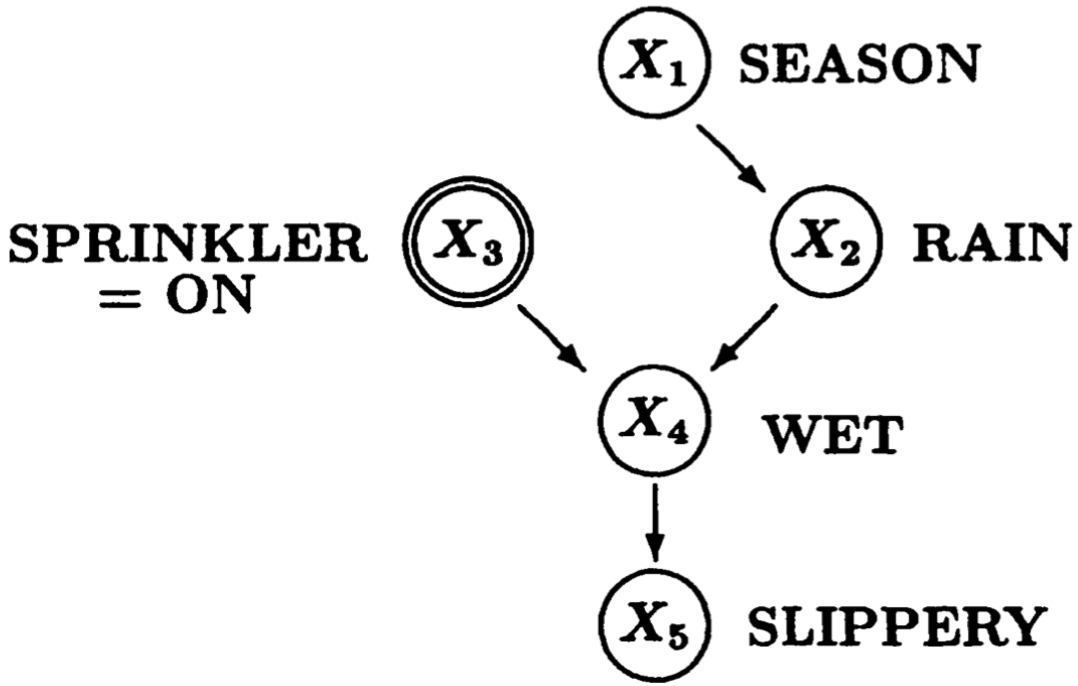
- In terms of training, which one of DeepProblog and Scallop is more efficient?

Pearl's Causal Hierarchy

- L1: Predictions: What if I observe … ?
- L2: Interventions: What if I change … ?
- L3: Counterfactuals: What if we did … given … ?

What models can be used to answer these questions?

Causal Bayesian Network: Handling Interventions



$$P_{X_3=\mathrm{On}}(x_1, x_2, x_4, x_5) = P(x_1)\,P(x_2 \mid x_1)\,P(x_4 \mid x_2, X_3 = \mathrm{On})P(x_5 \mid x_4),$$

Example Question

- Can any Bayesian network used for causal inference?

Structural Equation (Functional) Model

- Functional causal model
  - Can answer all three questions
- Expressed using deterministic functional equations
  - Probabilities are introduced by assuming certain variables are unobserved
  - Follows Laplace's conception of natural phenomena
- Advantages over stochastic representations
  - More general
  - More in tune with human intuition
  - Counterfactuals

Structural Equations

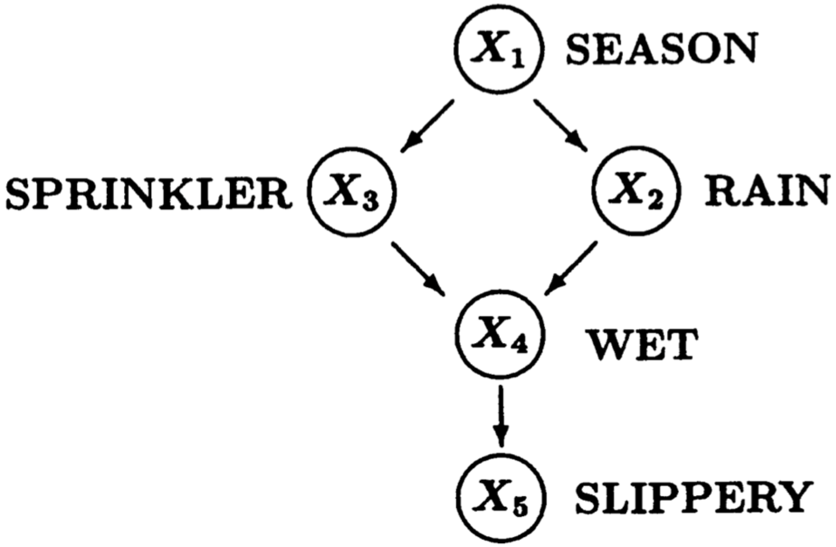
- A functional causal model consists a set of equations:

$$x_i = f_i(\underbrace{pa_i}_{\text{parents}}, \underbrace{u_i}_{\substack{\text{Errors due to} \\ \text{omitted factors.} \\ \text{Random.}}}), \quad i = 1, \ldots, n,$$

Counterfactuals in Functional Models

- Causal Bayesian networks have trouble dealing with counterfactuals
  - The simplest example:
    - Consider two independent boolean variables x and y, we have $P(x|y) = 0.5$, given $y = 1$, what is $P(y = 1 \mid do(x)= 0, y =1)$?
  - A more complex example:

$do(x_3{=}ON)$, $X_5{=}True$



Three Steps for Computing

For computing $P(Y= y \mid do(X = x), e)$:

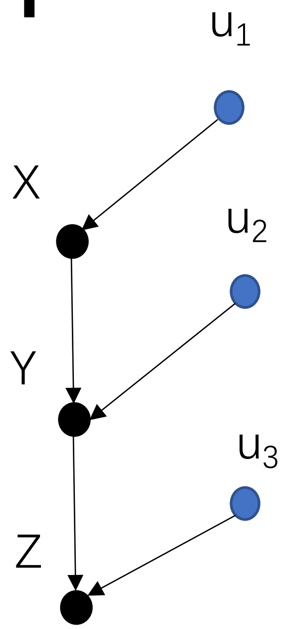
1. (abduction): Update the probability $P(u)$ to obtain $P(u|e)$

2. (action): Perform intervention $do(X) = x$

3. (prediction) Use the modified model to compute $P(Y=y)$

The Twin Network Approach

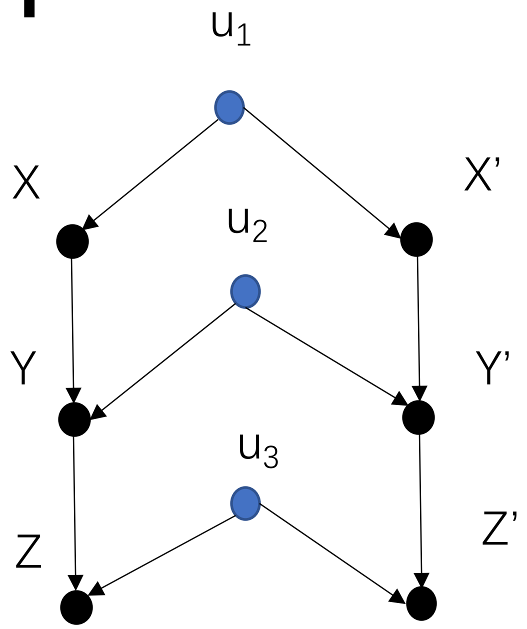
- Consider the following example
  - $X = u_1, Y = X + u_2, Z = Y + u_3$



- How to compute P(Z|do(X) = x, Z=z)?

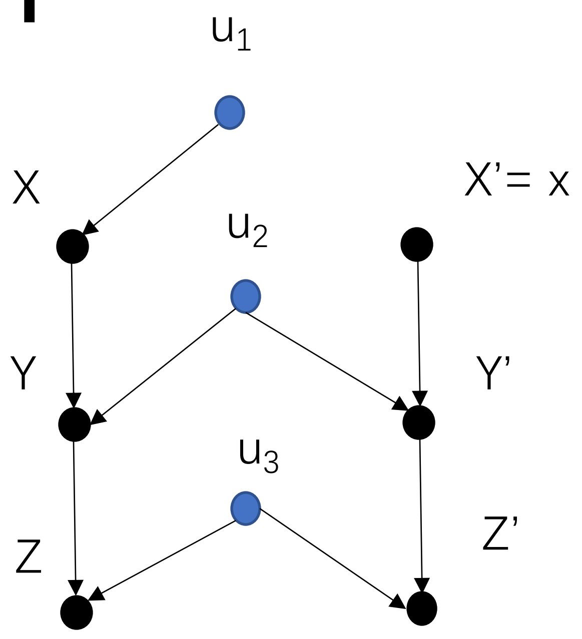
The Twin Network Approach

- P(Z|do(X) = x, Z=z) becomes P(Z'|do(X') = x, Z=z)



The Twin Network Approach

- P(Z|do(X) = x, Z=z) becomes P(Z'|X' = x, Z=z)



Pearl and Halpern's Definition of Actual Causality

- $\vec{X} = \vec{x}$ is an actual cause of ϕ in situation $(M, \vec{u})$ if

- AC1. $(M, \vec{u}) \vDash (\vec{X} = \vec{x}) \wedge \phi$
  - Both $(\vec{X} = \vec{x})$ and ϕ are true in the actual world

- AC2. Complicated. Captures counterfactuals

- AC3. $\vec{X}$ is minimal; no subset of $\vec{X}$ satisfies AC1 and AC2.
  - No irrelevant conjuncts

Pearl and Halpern's Definition

- AC2. There is a set of $\overrightarrow{W}$ of variables in V and a setting $\vec{x}'$ of the variables in $\vec{X}$ such that if $(M, \vec{u}) \vDash (\overrightarrow{W} = \vec{w})$, then
$$(M, \vec{u}) \vDash \left(\vec{X} \leftarrow \overrightarrow{x'}, \overrightarrow{W} \rightarrow \vec{w}\right) \wedge \neg\phi$$

In words: keeping the variables in $\overrightarrow{W}$ fixed at their actual values, changing $\vec{X}$ can change the outcome ϕ

Example

- JimmyThrows = u1, SuzyThrows = u2,
  SuzyShatters = SuzyThrows,
  JimmyShatters = JimmyThrows & !SuzyShatters,
  BottleShatters = SuzyShatters | JimmyShatters

Let $\vec{X} = \{SuzyThrows\}, \overrightarrow{W} = \{\text{JimmyShatters}\}, \phi = BottleShatters$,
then $(M, \vec{u}) \vDash \left(\vec{X} \leftarrow \vec{x}, \overrightarrow{W} \rightarrow \vec{w}\right) \wedge \neg\phi$

Example Question

- What predicate is the actual cause depending on how you model the problem. Is the statement right?

AI Explainability: Motivation

Utility

Debugging

Bias Detection

Recourse

If and when to trust model predictions

Vet models to assess suitability for deployment

Stakeholders

End users (e.g., loan applicants)

Decision makers (e.g., doctors, judges)

Regulatory agencies (e.g., FDA, European commission)

Researchers and engineers

Overview of Explainability Techniques

- Explainable models

- Post hoc explanations
  - Global vs. local

Approaches for Post hoc Explanations

Local Explanations

- Feature Importances
- Rule Based
- Saliency Maps
- Prototypes/Example Based
- Counterfactuals

Global Explanations

- Collection of Local Explanations
- Model Distillation
- Summaries of Counterfactuals
- Representation Based

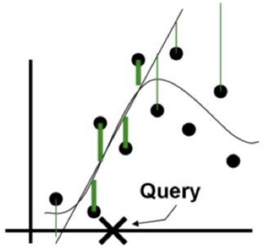
LIME Example

Original Image

P(labrador) = 0.21

Perturbed Instances	P(Labrador)
	0.92
	0.001
	0.34

Maybe to a fault?



Locally weighted regression

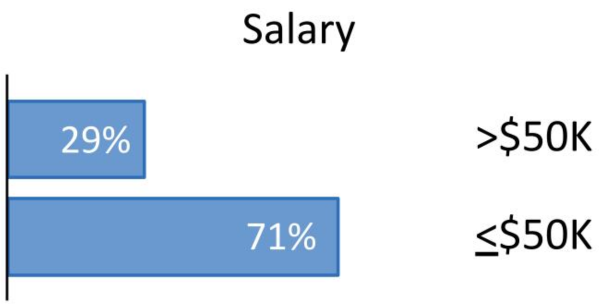
Explanation

LIME is quite customizable:

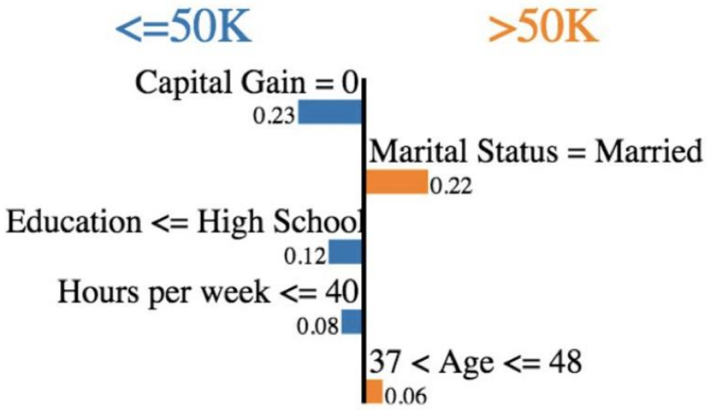
- How to perturb?
- Distance/similarity?
- How *local* you want it to be?
- How to express explanation

Anchors Example

Feature	Value
Age	$37 < \text{Age} \leq 48$
Workclass	Private
Education	$\leq$ High School
Marital Status	Married
Occupation	Craft-repair
Relationship	Husband
Race	Black
Sex	Male
Capital Gain	0
Capital Loss	0
Hours per week	≤ 40
Country	United States



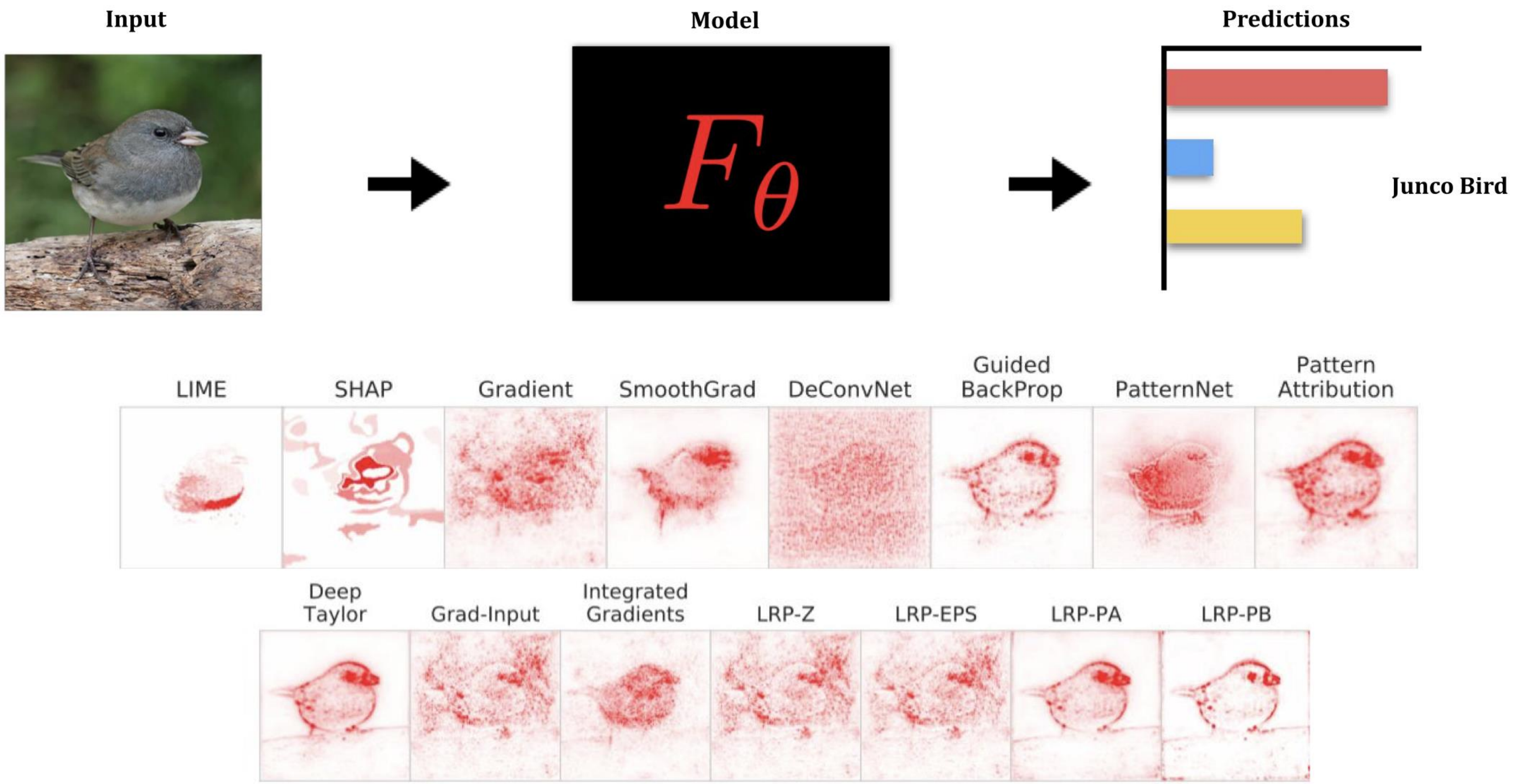
LIME



Anchors

IF Education ≤ High School
Then Predict Salary ≤ 50K

Saliency Maps



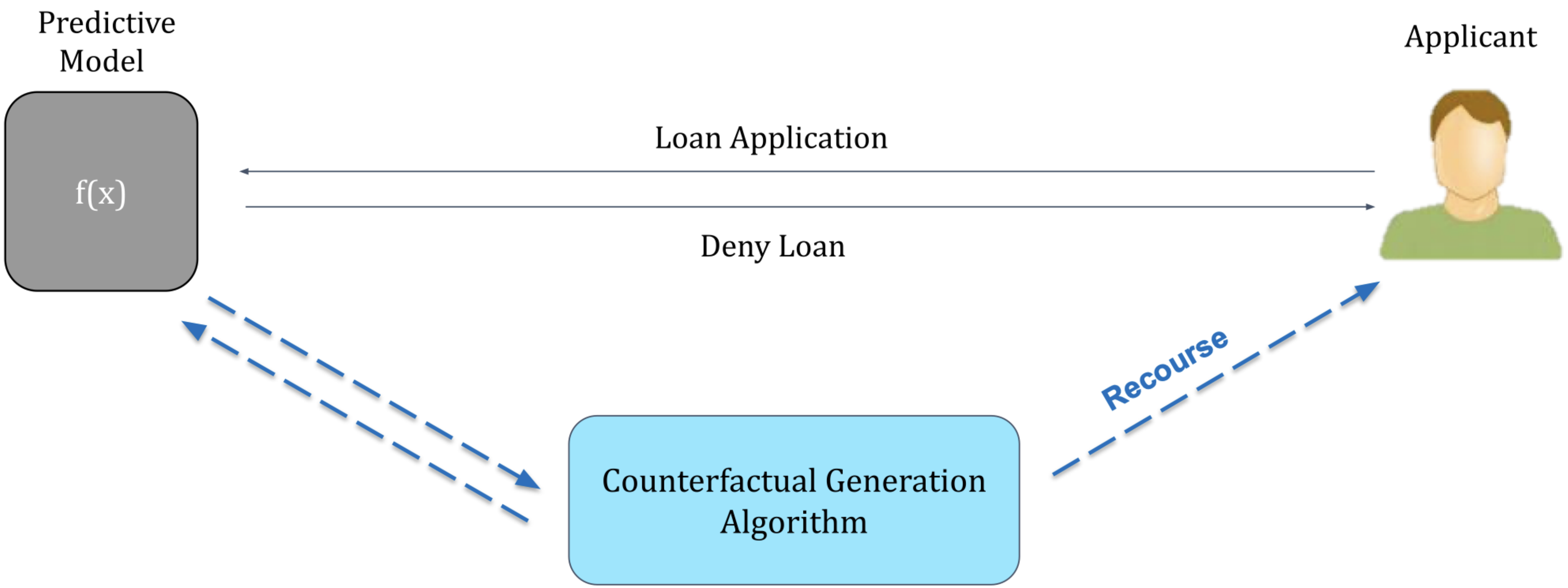
Prototype Approaches

Explain a model with synthetic or natural input ‘**examples**’.

Insights

- What kind of input is the model **most likely to misclassify**?
- Which training samples are **mislabelled**?
- Which input **maximally activates** an intermediate neuron?

Counterfactual Explanations



Recourse: Increase your salary by 50K & pay your credit card bills on time for next 3 months

Example Question

- Which technique below is an attribution-based explanation technique?

A. Anchors

B. Shapley Value

C. Counterfactual explanations

D. Influence function

LLM with Language Control

- Reduce ambiguity

- Enforce additional constraints
  - Correctness

- Key ideas
  - Use programming languages to interact with LLMs
  - Force LLMs to output structured sentences

Showcases

```
# instructions + few-shot samples
""""
A list of good dad jokes. A indicates the punchline
Q: How does a penguin build its house?
A: Igloos it together.
Q: Which knight invented King Arthur's Round Table?
A: Sir Cumference.
""""

# generate a joke
"Q:[JOKE]\n" where len(TOKENS(JOKE)) < 120 and STOPS_AT(JOKE, "?")
"A:[PUNCHLINE]" where STOPS_AT(PUNCHLINE, "\n") and len(TOKENS(PUNCHLINE)) > 1
```

Follow Semantics: Look Ahead

`Follow [<expr>](v, t) =`

- `True`, if vt satisfies <expr>
- `False`, if vt violates <expr>

Current Value

Next Vocabulary Token

Problem: What if a constraint is only momentarily violated? e.g., `len(PUNCHLINE) > 10`

From Luca Beurer-Kellner’s slides.

Follow and Final Semantics

Follow [<expr>](v, t) =

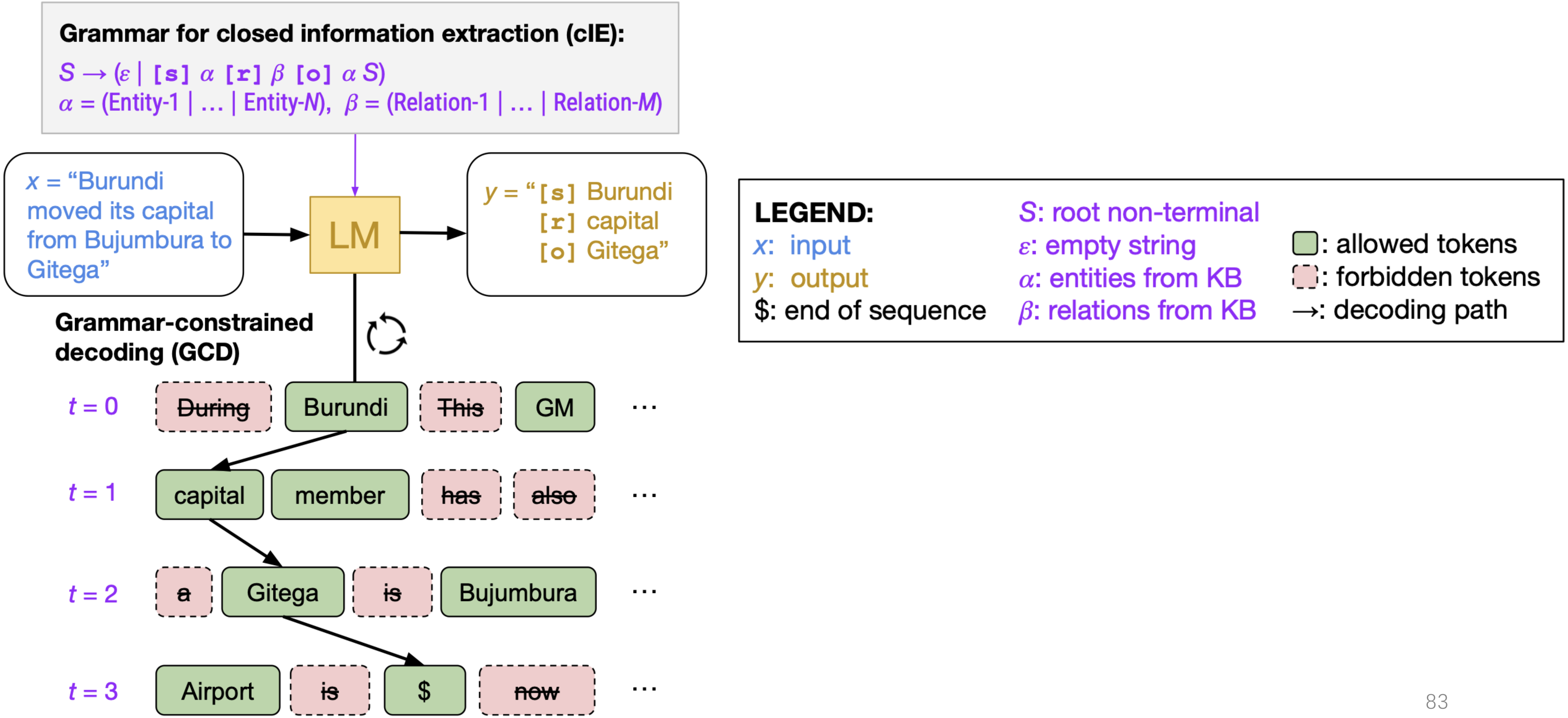
- **fin(True)**, if vt* satisfies <expr>
- **fin(False)**, if vt* violates <expr>
- **var(True)**, if vt satisfies <expr>
- **var(False)**, if vt satisfies <expr>

Current Value

Next Vocabulary Token

From Luca Beurer-Kellner’s slides.

Grammar-Constrained Decoding



The GCD Framework in EMNLP 2023

Assume parsing works

Parsing let us know if a sentence is valid according to a grammar.
It provides a `IsSentenceValid` function: str $->$ bool.

How GCD works

The high-level algorithm of GCD is:

1. Given an existing sentence(not necessarily complete) s
2. Get a probability distribution over the next token $P(w_i|s)$
3. For each candidate token w_i in the distribution:
4. Check if the sentence $s + w_i$ is valid according to the parser
5. If valid, add w_i to the whitelist
6. sample from the whitelist
7. Repeat until the sentence is complete

Example Question

- Which approach employs an incremental parser?

A. LMQL

B. Grammar-Constrained Decoding