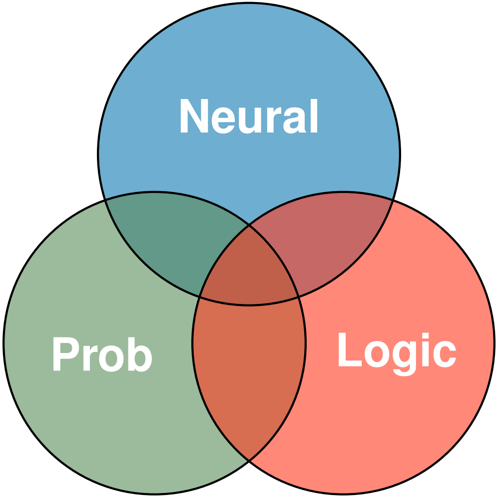
dPASP

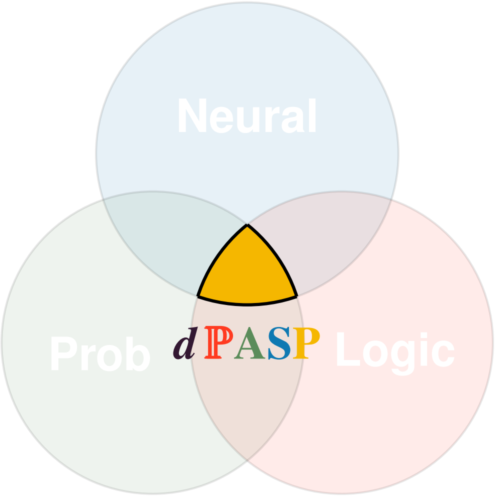
Programming with Logic and Neural Networks

Renato Lui Geh, Jonas Gonçalves, Igor Cataneo Silveira,
Denis Deratani Mauá, Fabio Gagliardi Cozman, Yuka Machino

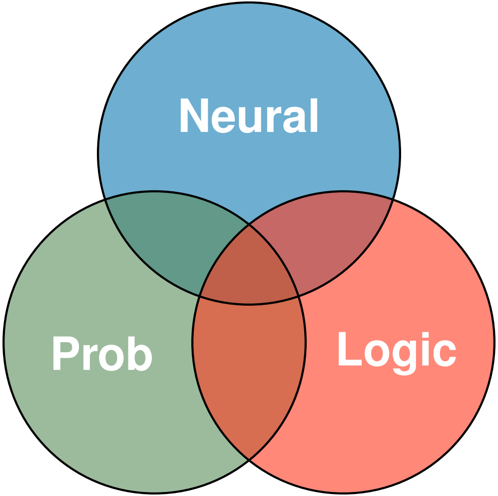
$d\mathbb{P}$ASP for Neurosymbolic Learning and Reasoning



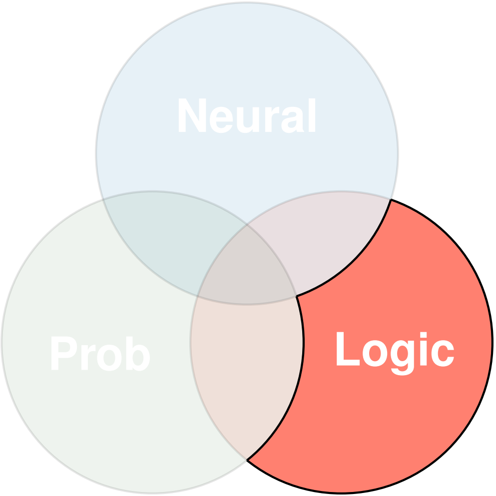
$d\mathbb{P}$ASP for Neurosymbolic Learning and Reasoning



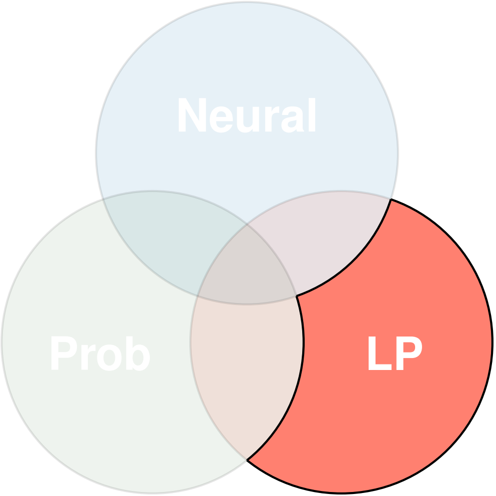
$d\mathbb{P}$ASP for Neurosymbolic Learning and Reasoning



d ℙASP for Neurosymbolic Learning and Reasoning



$d\mathbb{P}$ASP for Neurosymbolic Learning and Reasoning



Horn Logic

Alfred Horn (1918 – 2001)

A rule is...

$$(\neg b_1 \vee \neg b_2 \vee \cdots \vee \neg b_n) \vee (h_1 \vee h_2 \vee \cdots \vee h_m)$$

$$\equiv$$

$$\underbrace{h_1 \vee h_2 \vee \cdots \vee h_m}_{\text{Head}} \leftarrow \underbrace{b_1 \wedge b_2 \wedge \cdots \wedge b_n}_{\text{Body}} .$$

Intuition: *if* $b_1 \wedge \cdots \wedge b_n$, *then* one of $h_1, \cdots, h_m$ must hold.

[Horn, 1951]

Horn Logic

Alfred Horn (1918 – 2001)

A fact is...

$$\top \vee (h_1 \vee h_2 \vee \cdots \vee h_m)$$

$$\equiv$$

$$\underbrace{h_1 \vee h_2 \vee \cdots \vee h_m}_{\text{Head}} \leftarrow \underbrace{\top}_{\text{Body}}.$$

Intuition: $h_1, \cdots, h_m$ are *always* true.

[Horn, 1951]

Horn Logic

Alfred Horn (1918 – 2001)

An integrity constraint is...

$$(\neg b_1 \vee \neg b_2 \vee \cdots \vee \neg b_n) \vee \bot$$

$$\equiv$$

$$\underbrace{\bot}_{\text{Head}} \leftarrow \underbrace{b_1 \wedge b_2 \wedge \cdots \wedge b_n}_{\text{Body}}.$$

Intuition: $b_1 \wedge \cdots \wedge b_n$ *cannot* be true.

[Horn, 1951]

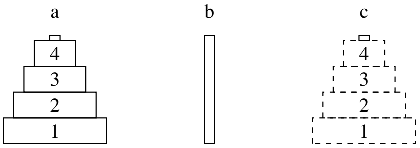
Answer Set Programming

```
% Solving Towers of Hanoi with ASP.

{ move(D,P,T) : disk(D), peg(P) } = 1 :- moves(M), T = 1..M.

move(D,T)    :- move(D,_,T).
on(D,P,0)    :- init_on(D,P).
on(D,P,T)    :- move(D,P,T).
on(D,P,T+1) :- on(D,P,T), not move(D,T+1),
                          not moves(T).
blocked(D-1,P,T+1) :- on(D,P,T), not moves(T).
blocked(D-1,P,T)   :- blocked(D,P,T), disk(D).

:- move(D,P,T), blocked(D-1,P,T).
:- move(D,T), on(D,P,T-1), blocked(D,P,T).
:- goal_on(D,P), not on(D,P,M), moves(M).
:- { on(D,P,T) } != 1, disk(D), moves(M), T = 1..M.
```



[Gebser et al., 2019]

Answer Set Programming – Syntax

Rule

$$\text{Umbrella} \vee \text{Raincoat} \leftarrow \text{Raining} \wedge \text{GoingOutside}$$

```
umbrella; raincoat :- raining, going_outside.
```

Answer Set Programming – Syntax

Rule

$$\text{Umbrella} \lor \text{Raincoat} \leftarrow \text{Raining} \land \text{GoingOutside}$$

```
umbrella; raincoat :- raining, going_outside.
```

Rule w/ vars

$$\forall x(\text{Duck}(x) \leftarrow \text{Quacks}(x) \land \text{Flies}(x) \land \text{Swims}(x))$$

```
duck(X) :- quacks(X), flies(X), swims(X).
```

Answer Set Programming – Syntax

Rule

$$\text{Umbrella} \vee \text{Raincoat} \leftarrow \text{Raining} \wedge \text{GoingOutside}$$

```
umbrella; raincoat :- raining, going_outside.
```

Rule w/ vars

$$\forall x (\text{Duck}(x) \leftarrow \text{Quacks}(x) \wedge \text{Flies}(x) \wedge \text{Swims}(x))$$

```
duck(X) :- quacks(X), flies(X), swims(X).
```

Fact

$$\text{Day}(\text{Monday}) \leftarrow \top$$

```
day(monday).
```

Answer Set Programming – Syntax

Rule

$$\text{Umbrella} \lor \text{Raincoat} \leftarrow \text{Raining} \land \text{GoingOutside}$$

```
umbrella; raincoat :- raining, going_outside.
```

Rule w/ vars

$$\forall x(\text{Duck}(x) \leftarrow \text{Quacks}(x) \land \text{Flies}(x) \land \text{Swims}(x))$$

```
duck(X) :- quacks(X), flies(X), swims(X).
```

Fact

$$\text{Day}(\text{Monday}) \leftarrow \top$$

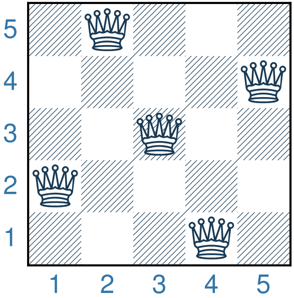
```
day(monday).
```

Integrity constraint

$$\forall x(\bot \leftarrow \text{Penguin}(x), \text{Flies}(x))$$

```
:- penguin(X), flies(X).
```

Answer Set Programming – Example

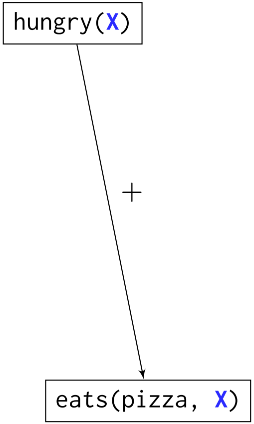


The n-queen problem in ASP

```
% Generate at most one queen row-wise.
{queen(I, 1..n)} = 1 :- I = 1..n.
% Generate at most one queen column-wise.
{queen(1..n, J)} = 1 :- J = 1..n.
% Constrain diagonal attacks.
:- {queen(D-J, J)} > 1, D = 2..2*n.
:- {queen(D+J, J)} > 1, D = 1-n..n-1.
```

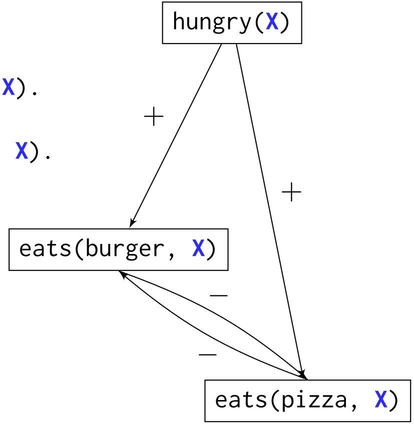
Answer Set Programming – Stable Model Semantics

```
% If we are hungry, we eat pizza.
eats(pizza, X)  :- hungry(X).
% Bruna is hungry.
hungry(bruna).
---
Answer: 1
hungry(bruna) eats(pizza,bruna)
```



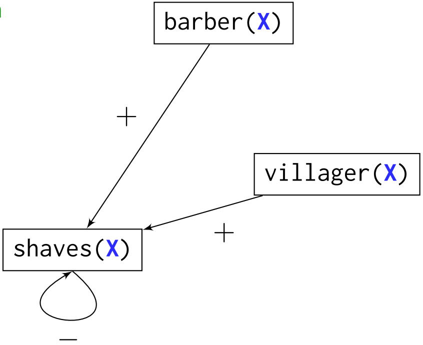
Answer Set Programming – Stable Model Semantics

```
% If we are hungry, either we have burgers...
eats(burger, X) :- hungry(X), not eats(pizza, X).
% ...or pizza.
eats(pizza, X)  :- hungry(X), not eats(burger, X).
% Bruna is hungry.
hungry(bruna).
---
Answer: 1
hungry(bruna) eats(pizza,bruna)
Answer: 2
hungry(bruna) eats(burger,bruna)
```



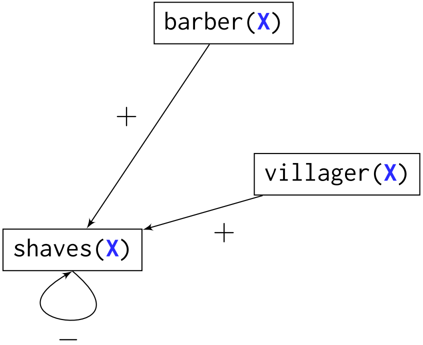
Answer Set Programming – Stable Model Semantics

```
% This barber shaves all those who live in
% the village yet do not shave themselves.
shaves(X, Y) :- barber(X), villager(Y),
                not shaves(Y, Y).
% John is a barber...
barber(john).
% ... who lives in the village.
villager(john).
% Carl lives in the village.
villager(carl).
---
UNSATISFIABLE
```



Answer Set Programming – Least-undefined Stable Semantics

```
% This barber shaves all those who live in
% the village yet do not shave themselves.
shaves(X, Y) :- barber(X), villager(Y),
                not shaves(Y, Y).
% John is a barber...
barber(john).
% ... who lives in the village.
villager(john).
% Carl lives in the village.
villager(carl).
---
Answer: 1
barber(john) villager(john)
villager(carl) shaves(john, carl)
undef shaves(john, john)
```



Answer Set Programming – Limitations

```
% If house is messy, I do my chores.
do(chores) :- house(messy).
% Procrastinate if stressed and not doing chores.
do(procrastinate) :- stressed, not do(chores).
% Study if not procrastinating or doing chores.
do(study) :- not do(procrastinate), not do(chores).
% Oversleep if procrastinated.
overslept :- do(procrastinate).
% I'm late if I overslept or the bus is late.
late :- overslept.
late :- bus_late.
% I'll do the exam if I'm not late.
do(exam) :- not late.
% I'll pass if I study and do the exam.
pass :- do(study), do(exam).
```

Answer Set Programming – Limitations

```
% If house is messy, I do my chores.
do(chores) :- house(messy).
% Procrastinate if stressed and not doing chores.
do(procrastinate) :- stressed, not do(chores).
% Study if not procrastinating or doing chores.
do(study) :- not do(procrastinate), not do(chores).
% Oversleep if procrastinated.
overslept :- do(procrastinate).
% I'm late if I overslept or the bus is late.
late :- overslept.
late :- bus_late.
% I'll do the exam if I'm not late.
do(exam) :- not late.
% I'll pass if I study and do the exam.
pass :- do(study), do(exam).
```

Given that...

- `house(messy)`
- `¬stressed`

...we have that

Answer:
`house(messy) do(chores) do(exam)`

Answer Set Programming – Limitations

```
% If house is messy, I do my chores.
do(chores) :- house(messy).
% Procrastinate if stressed and not doing chores.
do(procrastinate) :- stressed, not do(chores).
% Study if not procrastinating or doing chores.
do(study) :- not do(procrastinate), not do(chores).
% Oversleep if procrastinated.
overslept :- do(procrastinate).
% I'm late if I overslept or the bus is late.
late :- overslept.
late :- bus_late.
% I'll do the exam if I'm not late.
do(exam) :- not late.
% I'll pass if I study and do the exam.
pass :- do(study), do(exam).
```

Given that...

- ¬house(messy)
- stressed

...we have that

Answer:

do(procrastinate) stressed
overslept late

Answer Set Programming – Limitations

```
% If house is messy, I do my chores.
do(chores) :- house(messy).
% Procrastinate if stressed and not doing chores.
do(procrastinate) :- stressed, not do(chores).
% Study if not procrastinating or doing chores.
do(study) :- not do(procrastinate), not do(chores).
% Oversleep if procrastinated.
overslept :- do(procrastinate).
% I'm late if I overslept or the bus is late.
late :- overslept.
late :- bus_late.
% I'll do the exam if I'm not late.
do(exam) :- not late.
% I'll pass if I study and do the exam.
pass :- do(study), do(exam).
```

Given that...

- `house(messy)`
- `stressed`

...we have that

Answer:

```
house(messy) do(chores)
do(exam) stressed
```

Answer Set Programming – Limitations

```
% If house is messy, I do my chores.
do(chores) :- house(messy).
% Procrastinate if stressed and not doing chores.
do(procrastinate) :- stressed, not do(chores).
% Study if not procrastinating or doing chores.
do(study) :- not do(procrastinate), not do(chores).
% Oversleep if procrastinated.
overslept :- do(procrastinate).
% I'm late if I overslept or the bus is late.
late :- overslept.
late :- bus_late.
% I'll do the exam if I'm not late.
do(exam) :- not late.
% I'll pass if I study and do the exam.
pass :- do(study), do(exam).
```

Do I *always* tidy up when messy?

Answer Set Programming – Limitations

```
% If house is messy, I do my chores.
do(chores) :- house(messy).
% Procrastinate if stressed and not doing chores.
do(procrastinate) :- stressed, not do(chores).
% Study if not procrastinating or doing chores.
do(study) :- not do(procrastinate), not do(chores).
% Oversleep if procrastinated.
overslept :- do(procrastinate).
% I'm late if I overslept or the bus is late.
late :- overslept.
late :- bus_late.
% I'll do the exam if I'm not late.
do(exam) :- not late.
% I'll pass if I study and do the exam.
pass :- do(study), do(exam).
```

Do I *always* tidy up when messy?

Do I *always* procrastinate when stressed?

Answer Set Programming – Limitations

```
% If house is messy, I do my chores.
do(chores) :- house(messy).
% Procrastinate if stressed and not doing chores.
do(procrastinate) :- stressed, not do(chores).
% Study if not procrastinating or doing chores.
do(study) :- not do(procrastinate), not do(chores).
% Oversleep if procrastinated.
overslept :- do(procrastinate).
% I'm late if I overslept or the bus is late.
late :- overslept.
late :- bus_late.
% I'll do the exam if I'm not late.
do(exam) :- not late.
% I'll pass if I study and do the exam.
pass :- do(study), do(exam).
```

Do I *always* tidy up when messy?

Do I *always* procrastinate when stressed?

How *often* is the bus late?

Answer Set Programming – Limitations

```
% If house is messy, I do my chores.
do(chores) :- house(messy).
% Procrastinate if stressed and not doing chores.
do(procrastinate) :- stressed, not do(chores).
% Study if not procrastinating or doing chores.
do(study) :- not do(procrastinate), not do(chores).
% Oversleep if procrastinated.
overslept :- do(procrastinate).
% I'm late if I overslept or the bus is late.
late :- overslept.
late :- bus_late.
% I'll do the exam if I'm not late.
do(exam) :- not late.
% I'll pass if I study and do the exam.
pass :- do(study), do(exam).
```

Do I *always* tidy up when messy?

Do I *always* procrastinate when stressed?

How *often* is the bus late?

Am I *really* going to pass if I study?

Answer Set Programming – Limitations

```
% If house is messy, I do my chores.
do(chores) :- house(messy).
% Procrastinate if stressed and not doing chores.
do(procrastinate) :- stressed, not do(chores).
% Study if not procrastinating or doing chores.
do(study) :- not do(procrastinate), not do(chores).
% Oversleep if procrastinated.
overslept :- do(procrastinate).
% I'm late if I overslept or the bus is late.
late :- overslept.
late :- bus_late.
% I'll do the exam if I'm not late.
do(exam) :- not late.
% I'll pass if I study and do the exam.
pass :- do(study), do(exam).
```

Do I *always* tidy up when messy?

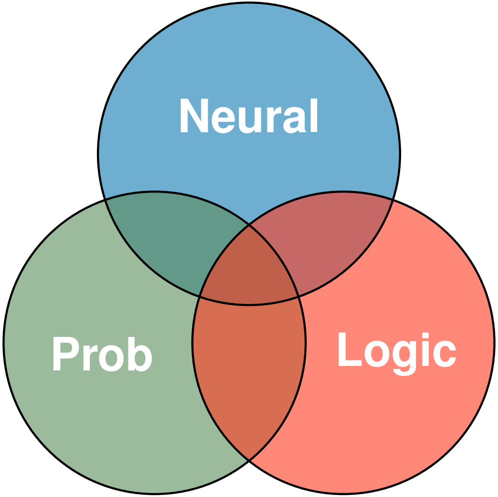
Do I *always* procrastinate when stressed?

How *often* is the bus late?

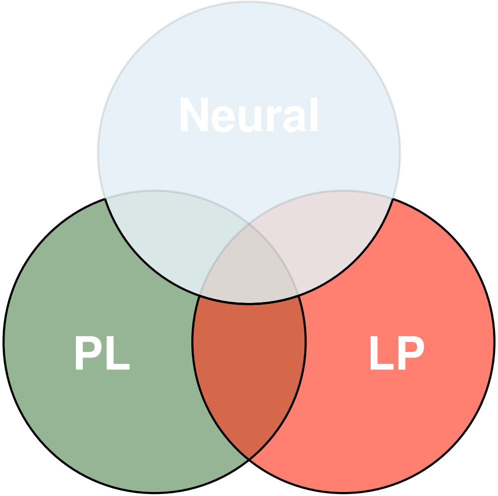
Am I *really* going to pass if I study?

We somehow need to describe *uncertainty*!

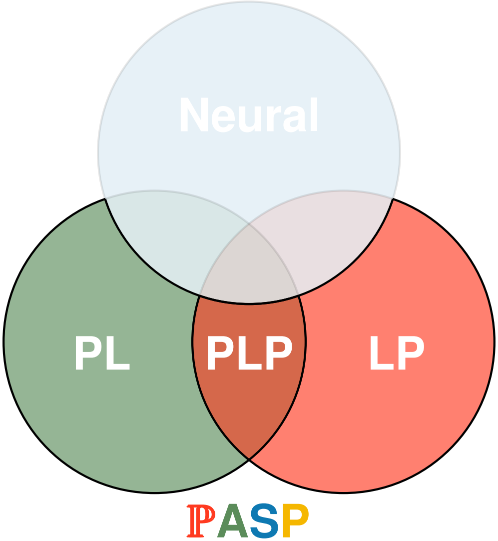
dℙASP for Neurosymbolic Learning and Reasoning



$d\mathbb{P}$ASP for Neurosymbolic Learning and Reasoning



dPASP for Neurosymbolic Learning and Reasoning



Probabilistic Logic Programming

Probabilistic facts

```
% The bus is late once every ten days.
0.1::bus_late.
```

Probabilistic Logic Programming

Probabilistic facts

```
% The bus is late once every ten days.
0.1::bus_late.
```

Probabilistic rules

```
% I only do my chores sometimes...
0.5::do(chores) :- house(messy).
```

Probabilistic Logic Programming

Probabilistic facts

```
% The bus is late once every ten days.
0.1::bus_late.
```

Probabilistic rules

```
% I only do my chores sometimes...
0.5::do(chores) :- house(messy).
```

Annotated disjunctions

```
% My house is either clean, messy or a safety hazard.
0.3::house(clean); 0.6::house(messy); 0.1::house(radioactive).
```

Probabilistic Logic Programming

Probabilistic facts

```
% The bus is late once every ten days.
0.1::bus_late.
```

Probabilistic rules

```
% I only do my chores sometimes...
0.5::do(chores) :- house(messy).
```

Annotated disjunctions

```
% My house is either clean, messy or a safety hazard.
0.3::house(clean); 0.6::house(messy); 0.1::house(radioactive).
```

What is the *probability* I pass?

Probabilistic Logic Programming

```
% Crime rate.
0.2::burglary.
```

Probabilistic Logic Programming

```
% Crime rate.
0.2::burglary.

% Daily earthquake probabilities.
0.05::earthquake(heavy); 0.15::earthquake(mild); 0.8::earthquake(none).
```

Probabilistic Logic Programming

```
% Crime rate.
0.2::burglary.

% Daily earthquake probabilities.
0.05::earthquake(heavy); 0.15::earthquake(mild); 0.8::earthquake(none).

% Error rates.
0.90::alarm :- burglary, earthquake(heavy).
0.85::alarm :- burglary, earthquake(mild).
0.80::alarm :- burglary, earthquake(none).
0.05::alarm :- not burglary, earthquake(mild).
0.10::alarm :- not burglary, earthquake(heavy).
```

Probabilistic Logic Programming

```
% Crime rate.
0.2::burglary.

% Daily earthquake probabilities.
0.05::earthquake(heavy); 0.15::earthquake(mild); 0.8::earthquake(none).

% Error rates.
0.90::alarm :- burglary, earthquake(heavy).
0.85::alarm :- burglary, earthquake(mild).
0.80::alarm :- burglary, earthquake(none).
0.05::alarm :- not burglary, earthquake(mild).
0.10::alarm :- not burglary, earthquake(heavy).

% Help of neighbors.
0.8::calls(X) :- alarm, neighbor(X).
0.1::calls(X) :- not alarm, neighbor(X).
```

Probabilistic Logic Programming

```
% Crime rate.
0.2::burglary.

% Daily earthquake probabilities.
0.05::earthquake(heavy); 0.15::earthquake(mild); 0.8::earthquake(none).

% Error rates.
0.90::alarm :- burglary, earthquake(heavy).
0.85::alarm :- burglary, earthquake(mild).
0.80::alarm :- burglary, earthquake(none).
0.05::alarm :- not burglary, earthquake(mild).
0.10::alarm :- not burglary, earthquake(heavy).

% Help of neighbors.
0.8::calls(X) :- alarm, neighbor(X).
0.1::calls(X) :- not alarm, neighbor(X).

% Bert and Ernie are Elmo's neighbors.
neighbor(bert). neighbor(ernie).
```

Probabilistic Logic Programming

```
% Crime rate.
0.2::burglary.

% Daily earthquake probabilities.
0.05::earthquake(heavy); 0.15::earthquake(mild); 0.8::earthquake(none).

% Error rates.
0.90::alarm :- burglary, earthquake(heavy).
0.85::alarm :- burglary, earthquake(mild).
0.80::alarm :- burglary, earthquake(none).
0.05::alarm :- not burglary, earthquake(mild).
0.10::alarm :- not burglary, earthquake(heavy).

% Help of neighbors.
0.8::calls(X) :- alarm, neighbor(X).
0.1::calls(X) :- not alarm, neighbor(X).

% Bert and Ernie are Elmo's neighbors.
neighbor(bert). neighbor(ernie).

#semantics maxent.
% What is the probability of a burglary given Bert has called?
#query burglary | calls(bert).
---
ℙ(burglary | calls(bert)) = 0.605889
```

Probabilistic Logic Programming

```
% Crime rate.
0.2::burglary.

% Daily earthquake probabilities.
0.05::earthquake(heavy); 0.15::earthquake(mild); 0.8::earthquake(none).

% Error rates.
0.90::alarm :- burglary, earthquake(heavy).
0.85::alarm :- burglary, earthquake(mild).
0.80::alarm :- burglary, earthquake(none).
0.05::alarm :- not burglary, earthquake(mild).
0.10::alarm :- not burglary, earthquake(heavy).

% Help of neighbors.
0.8::calls(X) :- alarm, neighbor(X).
0.1::calls(X) :- not alarm, neighbor(X).

% Bert and Ernie are Elmo's neighbors.
neighbor(bert). neighbor(ernie).

#semantics maxent.
% What is the probability of a burglary given Bert has called?
#query burglary | calls(bert).
---
ℙ(burglary | calls(bert)) = 0.605889
```

But *how* do we compute these probabilities?

Probabilistic Logic Programming

Given probabilistic components...

$\texttt{0.25::}a.\ \texttt{0.70::}b.\ \texttt{0.40::}c.$

Probabilistic Logic Programming

Given probabilistic components...

$\texttt{0.25}::a.\ \texttt{0.70}::b.\ \texttt{0.40}::c.$

A total choice $\theta \in \Theta$ is...

$\Theta = \{\{a, b, c\}, \{a, b, \texttt{not}\ c\}, \ldots, \{\texttt{not}\ a, \texttt{not}\ b, \texttt{not}\ c\}\}.$

Probabilistic Logic Programming

Given probabilistic components...

$\texttt{0.25::}a.\ \texttt{0.70::}b.\ \texttt{0.40::}c.$

A total choice $\theta \in \Theta$ is...

$\Theta = \{\{a, b, c\}, \{a, b, \texttt{not}\ c\}, \ldots, \{\texttt{not}\ a, \texttt{not}\ b, \texttt{not}\ c\}\}.$

If $\theta = \{a, \texttt{not}\ b, c\}$, then...

$\mathbb{P}(\theta) = \mathbb{P}(a) \cdot \mathbb{P}(\texttt{not}\ b) \cdot \mathbb{P}(c) = 0.25 \times 0.3 \times 0.4 = 0.03.$

Probabilistic Logic Programming

Given probabilistic components...

$\texttt{0.25::}a.\ \texttt{0.70::}b.\ \texttt{0.40::}c.$

A total choice $\theta \in \Theta$ is...

$\Theta = \{\{a, b, c\}, \{a, b, \texttt{not}\ c\}, \ldots, \{\texttt{not}\ a, \texttt{not}\ b, \texttt{not}\ c\}\}.$

If $\theta = \{a, \texttt{not}\ b, c\}$, then...

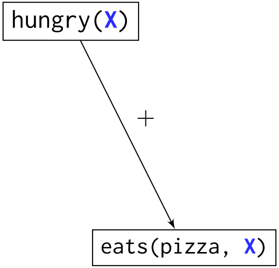
$\mathbb{P}(\theta) = \mathbb{P}(a) \cdot \mathbb{P}(\texttt{not}\ b) \cdot \mathbb{P}(c) = 0.25 \times 0.3 \times 0.4 = 0.03.$

The probability of query q is...

$$\mathbb{P}(q) = \sum_{\theta \in \Theta} \mathbb{P}(\theta) \cdot [\![I_\theta \models q]\!]$$

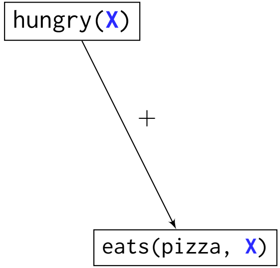
Probabilistic Logic Programming – Example

```
% If we are hungry, we eat pizza.
eats(pizza, X)  :- hungry(X).
% Bruna is hungry 70% of the time.
0.7::hungry(bruna).
---
Answer: 1
∅
Probability: 0.3
---
Answer: 2
hungry(bruna) eats(pizza,bruna)
Probability: 0.7
```



Probabilistic Logic Programming – Example

```
% If we are hungry, we eat pizza.
eats(pizza, X)  :- hungry(X).
% Bruna is hungry 70% of the time.
0.7::hungry(bruna).
---
Answer: 1
∅
Probability: 0.3
---
Answer: 2
hungry(bruna) eats(pizza,bruna)
Probability: 0.7
```



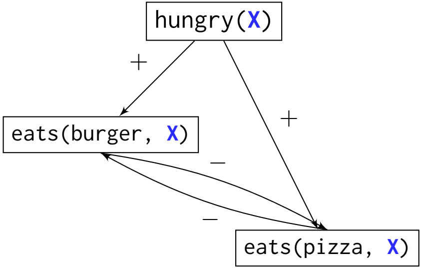
What is $\mathbb{P}$(eats(pizza,bruna))?

$$\mathbb{P}(q) = \sum_{\theta \in \Theta} \mathbb{P}(\theta) \cdot [\![I_\theta \models q]\!]$$

$$= 0.3 \times 0 + 0.7 \times 1 = 0.7$$

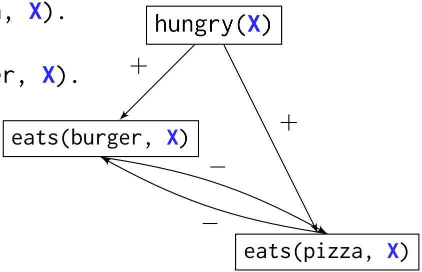
Probabilistic Logic Programming – Example

```
% If we are hungry, either we have burgers...
eats(burger, X) :- hungry(X), not eats(pizza, X).
% ...or pizza, but not both.
eats(pizza, X)  :- hungry(X), not eats(burger, X).
% Bruna is hungry 70% of the time.
0.7::hungry(bruna).
---
Answer: 1
∅
Probability: 0.3
---
Answer: 2
hungry(bruna) eats(pizza,bruna)
Probability: 0.7
---
Answer: 3
hungry(bruna) eats(burger,bruna)
Probability: 0.7
```



Probabilistic Logic Programming – Example

```
% If we are hungry, either we have burgers...
eats(burger, X) :- hungry(X), not eats(pizza, X).
% ...or pizza, but not both.
eats(pizza, X)  :- hungry(X), not eats(burger, X).
% Bruna is hungry 70% of the time.
0.7::hungry(bruna).
---
Answer: 1
∅
Probability: 0.3
---
Answer: 2
hungry(bruna) eats(pizza,bruna)
Probability: 0.7
---
Answer: 3
hungry(bruna) eats(burger,bruna)
Probability: 0.7
```



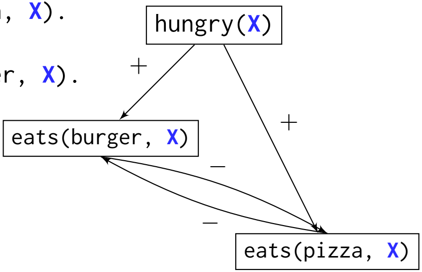
What is $\mathbb{P}($eats(pizza,bruna)$)$?

$$\mathbb{P}(q) = \sum_{\theta \in \Theta} \mathbb{P}(\theta) \cdot [\![I_\theta \models q]\!]$$

$$= 0.3 \times 0 + 0.7 \times 1 + 0.7 \times 1 =$$

Probabilistic Logic Programming – Example

```
% If we are hungry, either we have burgers...
eats(burger, X) :- hungry(X), not eats(pizza, X).
% ...or pizza, but not both.
eats(pizza, X)  :- hungry(X), not eats(burger, X).
% Bruna is hungry 70% of the time.
0.7::hungry(bruna).
---
Answer: 1
∅
Probability: 0.3
---
Answer: 2
hungry(bruna) eats(pizza,bruna)
Probability: 0.7
---
Answer: 3
hungry(bruna) eats(burger,bruna)
Probability: 0.7
```



What is $\mathbb{P}$(eats(pizza,bruna))?

$$\mathbb{P}(q) = \sum_{\theta \in \Theta} \mathbb{P}(\theta) \cdot [\![I_\theta \models q]\!]$$

$$= 0.3 \times 0 + 0.7 \times 1 + 0.7 \times 1 = 1.4 \text{ ???}$$

Probabilistic Answer Set Programming

Given probabilistic components...

$\texttt{0.25::}a.\ \texttt{0.70::}b.\ \texttt{0.40::}c.$

A total choice $\theta \in \Theta$ is...

$\Theta = \{\{a, b, c\}, \{a, b, \texttt{not}\ c\}, \ldots, \{\texttt{not}\ a, \texttt{not}\ b, \texttt{not}\ c\}\}.$

If $\theta = \{a, \texttt{not}\ b, c\}$, then...

$\mathbb{P}(\theta) = \mathbb{P}(a) \cdot \mathbb{P}(\texttt{not}\ b) \cdot \mathbb{P}(c) = 0.25 \times 0.3 \times 0.4 = 0.03.$

The probability of query q is...

$$\mathbb{P}(q) = \sum_{\theta \in \Theta} \mathbb{P}(\theta) \cdot [\![I_\theta \models q]\!]$$

Probabilistic Answer Set Programming

Given probabilistic components...

$\texttt{0.25::}a\texttt{.}\ \texttt{0.70::}b\texttt{.}\ \texttt{0.40::}c\texttt{.}$

A total choice $\theta \in \Theta$ is...

$\Theta = \{\{a, b, c\}, \{a, b, \texttt{not}\ c\}, \ldots, \{\texttt{not}\ a, \texttt{not}\ b, \texttt{not}\ c\}\}\texttt{.}$

If $\theta = \{a, \texttt{not}\ b, c\}$, then...

$\mathbb{P}(\theta) = \mathbb{P}(a) \cdot \mathbb{P}(\texttt{not}\ b) \cdot \mathbb{P}(c) = 0.25 \times 0.3 \times 0.4 = 0.03\texttt{.}$

The probability of query q is...

$$\mathbb{P}(q) = \sum_{\theta \in \Theta} \mathbb{P}(\theta) \cdot \frac{N(I_\theta \models q)}{N(\theta)}$$

Probabilistic Answer Set Programming

Given probabilistic components...

$\texttt{0.25::}a.\ \texttt{0.70::}b.\ \texttt{0.40::}c.$

A total choice $\theta \in \Theta$ is...

$$\Theta = \{\{a, b, c\}, \{a, b, \texttt{not}\ c\}, \ldots, \{\texttt{not}\ a, \texttt{not}\ b, \texttt{not}\ c\}\}.$$

If $\theta = \{a, \texttt{not}\ b, c\}$, then...

$$\mathbb{P}(\theta) = \mathbb{P}(a) \cdot \mathbb{P}(\texttt{not}\ b) \cdot \mathbb{P}(c) = 0.25 \times 0.3 \times 0.4 = 0.03.$$

The probability of query q is...

$$\mathbb{P}(q) = \sum_{\theta \in \Theta} \mathbb{P}(\theta) \cdot \underbrace{\frac{N(I_\theta \models q)}{N(\theta)}}_{\text{\# of models where } \theta \text{ is true}}$$

Probabilistic Answer Set Programming

Given probabilistic components...

$\texttt{0.25}::a.\ \texttt{0.70}::b.\ \texttt{0.40}::c.$

A total choice $\theta \in \Theta$ is...

$$\Theta = \{\{a, b, c\}, \{a, b, \texttt{not}\ c\}, \ldots, \{\texttt{not}\ a, \texttt{not}\ b, \texttt{not}\ c\}\}.$$

If $\theta = \{a, \texttt{not}\ b, c\}$, then...

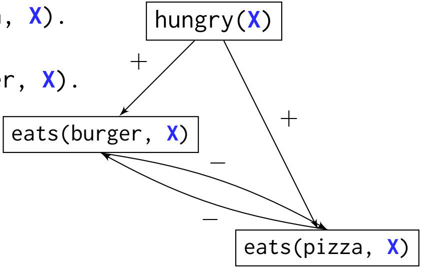
$$\mathbb{P}(\theta) = \mathbb{P}(a) \cdot \mathbb{P}(\texttt{not}\ b) \cdot \mathbb{P}(c) = 0.25 \times 0.3 \times 0.4 = 0.03.$$

The probability of query q is...

$$\mathbb{P}(q) = \sum_{\theta \in \Theta} \mathbb{P}(\theta) \cdot \frac{\overbrace{N(I_\theta \models q)}^{\text{\# of models where } \theta \text{ is true and } q \text{ is consistent}}}{N(\theta)}$$

Probabilistic Answer Set Programming – Example

```
% If we are hungry, either we have burgers...
eats(burger, X) :- hungry(X), not eats(pizza, X).
% ...or pizza, but not both.
eats(pizza, X)  :- hungry(X), not eats(burger, X).
% Bruna is hungry 70% of the time.
0.7::hungry(bruna).
---
Answer: 1
∅
Probability: 0.3
---
Answer: 2
hungry(bruna) eats(pizza,bruna)
Probability: 0.7
---
Answer: 3
hungry(bruna) eats(burger,bruna)
Probability: 0.7
```



What is $\mathbb{P}(\texttt{eats(pizza,bruna)})$?

$$\mathbb{P}(q) = \sum_{\theta \in \Theta} \mathbb{P}(\theta) \cdot \frac{N(I_\theta \models q)}{N(\theta)}$$

$$= 0.3 \times \frac{0}{1} + 0.7 \times \frac{1}{2} + 0.7 \times \frac{0}{2} = 0.35$$

Learning with ℙASP

```
% Crime rate.
?::burglary.
% Daily earthquake probabilities.
0.05::earthquake(heavy); 0.15::earthquake(mild); 0.8::earthquake(none).
% Error rates.
0.90::alarm :- burglary, earthquake(heavy).
0.85::alarm :- burglary, earthquake(mild).
0.80::alarm :- burglary, earthquake(none).
0.05::alarm :- not burglary, earthquake(mild).
0.10::alarm :- not burglary, earthquake(heavy).
% Help of neighbors.
0.8::calls(X) :- alarm, neighbor(X).
0.1::calls(X) :- not alarm, neighbor(X).
% Bert and Ernie are Elmo's neighbors.
neighbor(bert). neighbor(ernie).
% What is the probability of a burglary at Elmo's?
#query burglary.
% We learn from a CSV containing Bert and Ernie's calls.
#learn "https://www.ime.usp.br/~renatolg/dpasp/elmo.csv", niters = 10.
% Select the max-ent semantics.
#semantics maxent.
```

Output:

```
Learning [=========] ETA: 0h00m13s | LL=-7.07434
ℙ(burglary) = 0.201862
```

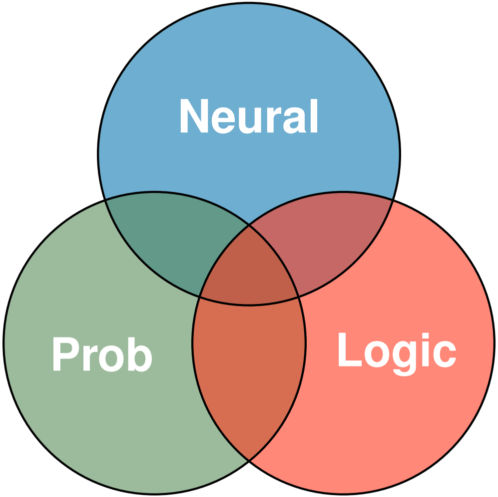
Learning with ℙASP

```
% Crime rate.
?::burglary.
% Daily earthquake probabilities.
0.05::earthquake(heavy); 0.15::earthquake(mild); 0.8::earthquake(none).
% Error rates.
0.90::alarm :- burglary, earthquake(heavy).
0.85::alarm :- burglary, earthquake(mild).
0.80::alarm :- burglary, earthquake(none).
0.05::alarm :- not burg
0.10::alarm :- not burg
% Help of neighbors.
0.8::calls(X) :- alarm, neighbor(X).
0.1::calls(X) :- not alarm, neighbor(X).
% Bert and Ernie are Elmo's neighbors.
neighbor(bert). neighbor(ernie).
% What is the probability of a burglary at Elmo's?
#query burglary.
% We learn from a CSV containing Bert and Ernie's calls.
#learn "https://www.ime.usp.br/~renatolg/dpasp/elmo.csv", niters = 10.
% Select the max-ent semantics.
#semantics maxent.
```

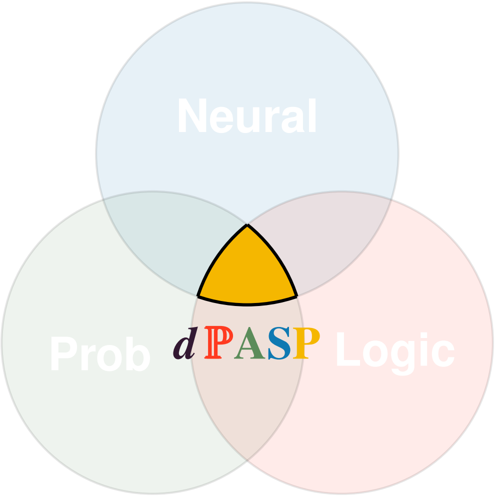
Continuous data? High dimensional data?

```
0h00m13s | LL=-7.07434
ℙ(burglary) = 0.201862
```

dPASP for Neurosymbolic Learning and Reasoning



$d\mathbb{P}$ASP for Neurosymbolic Learning and Reasoning



Neural Probabilistic Logic Programming

Neural facts

```
% Prob bus being late given traffic information.
?::bus_late(X) as @bus_net :- traffic_info(X).
```

Neural Probabilistic Logic Programming

Neural facts

```
% Prob bus being late given traffic information.
?::bus_late(X) as @bus_net :- traffic_info(X).
```

Neural rules

```
% Whether I do my chores depends on complex bio and neural data...
?::do_chores(X) as @brain_net :- bioneural_data(X), house(messy).
```

Neural Probabilistic Logic Programming

Neural facts

```
% Prob bus being late given traffic information.
?::bus_late(X) as @bus_net :- traffic_info(X).
```

Neural rules

```
% Whether I do my chores depends on complex bio and neural data...
?::do_chores(X) as @brain_net :- bioneural_data(X), house(messy).
```

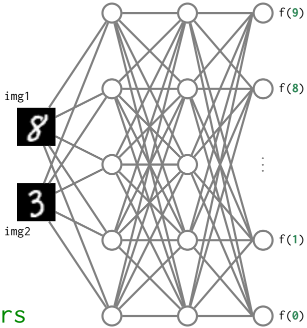
Neural annotated disjunctions

```
% The state of my house depends on my roommate.
?::house(X, {clean,messy,radioactive}) :- roommate_data(X).
```

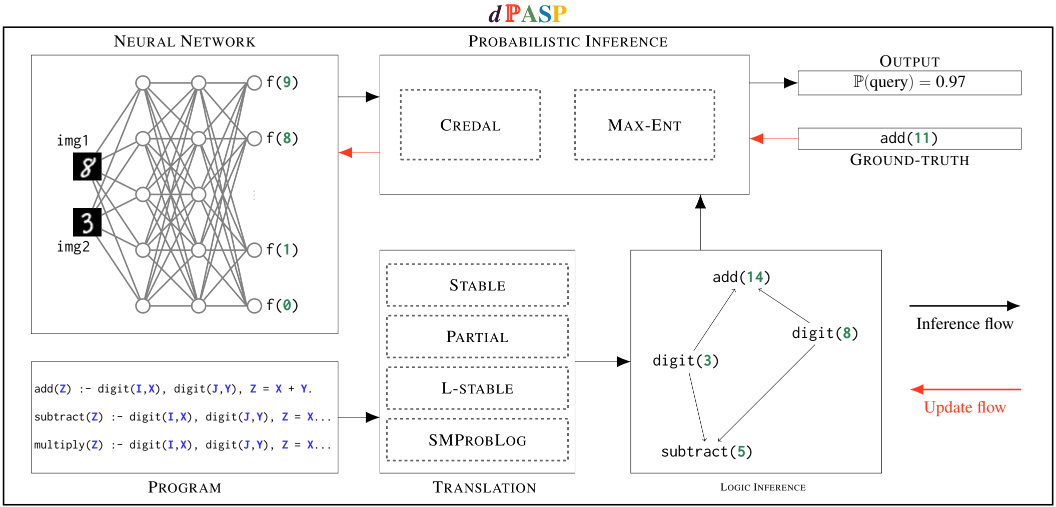
Neural Probabilistic Logic Programming

Example: Parsing arithmetic expressions, e.g. X + Y $= f($$) + f($$) = ?$

```
% neural rule
?::digit(Image, {0..9}) :- data(Image).
% data loaders -- interact with Python code
data(img1) ~ test(@mnist_test), train(@mnist_train).
data(img2) ~ test(@mnist_test), train(@mnist_train).
% prob. answer set program
add(Z) :- digit(I, X), digit(J, Y), Z = X + Y.
subtract(Z) :- digit(I, X), digit(J, Y), Z = X - Y.
multiply(Z) :- digit(I, X), digit(J, Y), Z = X * Y.
% learn the program end-to-end and pass learning parameters
#learn @mnist_sum, lr = 1., niters = 5, ..., batch = 1000.
% inference: what is the probability of X + Y = 14 given X = 8?
#query add(11) | digit(img1, 8).
```

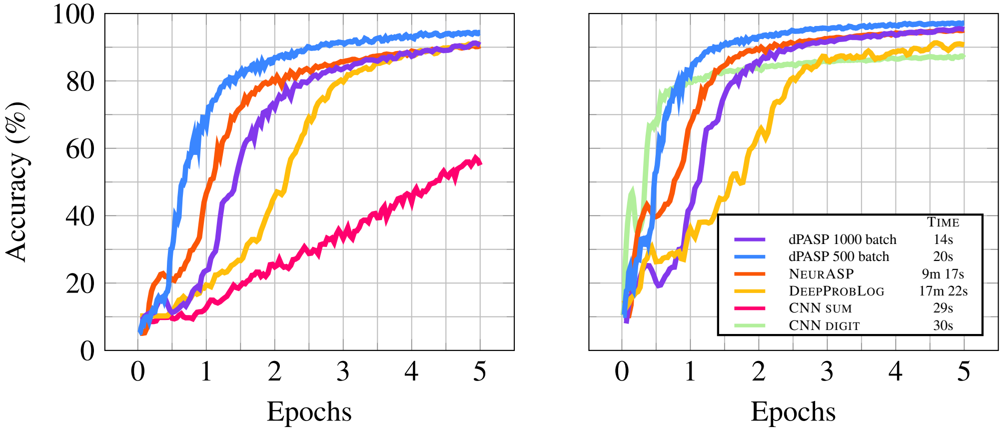


A Bird's Eye View of *d*ℙASP



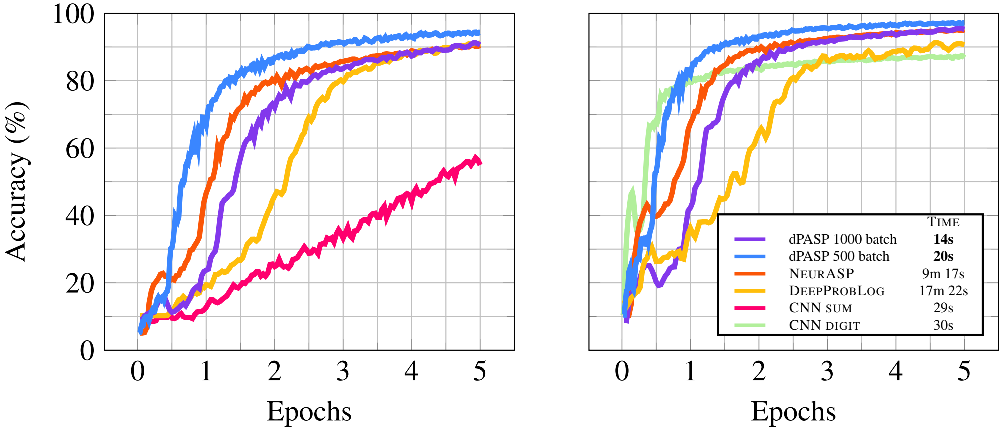
Experiments

How much **faster** is dPASP on the MNIST Add?



Experiments

How much **faster** is dPASP on the MNIST Add?



Challenges in $d\,\mathbb{P}$ASP

The woes of exact inference:

$$\mathbb{P}(q) = \sum_{\theta \in \Theta} \mathbb{P}(\theta) \cdot \frac{N(I_\theta \models q)}{N(\theta)}$$

Challenges in $d\mathbb{P}$ASP

The woes of exact inference:

$$\mathbb{P}(q) = \underbrace{\sum_{\theta \in \Theta}}_{\text{Exponential!}} \mathbb{P}(\theta) \cdot \frac{N(I_\theta \models q)}{N(\theta)}$$

Challenges in $d\mathbb{P}$ASP

The woes of exact inference:

$$\mathbb{P}(q) = \underbrace{\sum_{\theta \in \Theta}}_{\text{Exponential!}} \mathbb{P}(\theta) \cdot \frac{\overbrace{N(I_\theta \models q)}^{\text{\#P-complete!}}}{N(\theta)}$$

Challenges in $d\mathbb{P}$ASP

The woes of exact inference:

$$\mathbb{P}(q) = \underbrace{\sum_{\theta\in\Theta}}_{\text{Exponential!}} \mathbb{P}(\theta) \cdot \frac{\overbrace{N(I_\theta \models q)}^{\text{\#P-complete!}}}{N(\theta)}$$

Approximate inference by optimality:

$$\mathbb{P}(q) = \underbrace{\sum_{\theta\in\Theta^*}}_{\text{Linear!}} \mathbb{P}(\theta) \cdot \frac{\overbrace{N^*(I_\theta \models q)}^{\text{Linear!}}}{N^*(\theta)}$$

Challenges in $d\mathbb{P}$ASP

The woes of exact inference:

$$\mathbb{P}(q) = \underbrace{\sum_{\theta \in \Theta}}_{\text{Exponential!}} \mathbb{P}(\theta) \cdot \frac{\overbrace{N(I_\theta \models q)}^{\text{\#P-complete!}}}{N(\theta)}$$

Approximate inference by optimality:

$$\mathbb{P}(q) = \underbrace{\sum_{\theta \in \Theta^*}}_{\text{Linear!}} \mathbb{P}(\theta) \cdot \frac{\overbrace{N^*(I_\theta \models q)}^{\text{Linear!}}}{N^*(\theta)}$$

But biased towards high prob models!

Challenges in $d\mathbb{P}$ASP

The woes of exact inference:

$$\mathbb{P}(q) = \underbrace{\sum_{\theta \in \Theta}}_{\text{Exponential!}} \mathbb{P}(\theta) \cdot \frac{\overbrace{N(I_\theta \models q)}^{\text{\#P-complete!}}}{N(\theta)}$$

Approximate inference by sampling (the θ's):

$$\mathbb{P}(q) = \underbrace{\sum_{\theta \in \Theta'}}_{\text{Linear!}} \mathbb{P}(\theta) \cdot \frac{\overbrace{N(I_\theta \models q)}^{\text{Still \#P-complete!}}}{N(\theta)}$$

Challenges in $d\mathbb{P}$ASP

The woes of exact inference:

$$\mathbb{P}(q) = \underbrace{\sum_{\theta \in \Theta}}_{\text{Exponential!}} \mathbb{P}(\theta) \cdot \frac{\overbrace{N(I_\theta \models q)}^{\text{\#P-complete!}}}{N(\theta)}$$

Approximate inference by sampling (the θ's):

$$\mathbb{P}(q) = \underbrace{\sum_{\theta \in \Theta'}}_{\text{Linear!}} \mathbb{P}(\theta) \cdot \frac{\overbrace{N(I_\theta \models q)}^{\text{Still \#P-complete!}}}{N(\theta)}$$

No guarantees on approximation!

Challenges in $d\mathbb{P}$ASP

The woes of exact inference:

$$\mathbb{P}(q) = \underbrace{\sum_{\theta \in \Theta}}_{\text{Exponential!}} \mathbb{P}(\theta) \cdot \frac{\overbrace{N(I_\theta \models q)}^{\text{\#P-complete!}}}{N(\theta)}$$

Inference by compilation:

$$\mathbb{P}(q) = \underbrace{\sum_{\theta \in \Theta}}_{\text{Linear!}} \mathbb{P}(\theta) \cdot \frac{\overbrace{N(I_\theta \models q)}^{\text{Linear!}}}{N(\theta)}$$

Challenges in $d\mathbb{P}$ASP

The woes of exact inference:

$$\mathbb{P}(q) = \underbrace{\sum_{\theta \in \Theta}}_{\text{Exponential!}} \mathbb{P}(\theta) \cdot \overbrace{\frac{N(I_\theta \models q)}{N(\theta)}}^{\text{\#P-complete!}}$$

Inference by compilation:

$$\mathbb{P}(q) = \underbrace{\sum_{\theta \in \Theta}}_{\text{Linear!}} \mathbb{P}(\theta) \cdot \overbrace{\frac{N(I_\theta \models q)}{N(\theta)}}^{\text{Linear!}}$$

Compiling to probabilistic circuits is hard!

Challenges in $d\mathbb{P}$ASP

The woes of exact inference:

$$\mathbb{P}(q) = \underbrace{\sum_{\theta \in \Theta}}_{\text{Exponential!}} \mathbb{P}(\theta) \cdot \frac{\overbrace{N(I_\theta \models q)}^{\text{\#P-complete!}}}{N(\theta)}$$

Inference by approximate compilation:

$$\mathbb{P}(q) = \underbrace{\sum_{\theta \in \Theta}}_{\text{Linear!}} \mathbb{P}(\theta) \cdot \frac{\overbrace{N(I_\theta \models q)}^{\text{Linear!}}}{N(\theta)}$$

Challenges in $d\mathbb{P}$ASP

The woes of exact inference:

$$\mathbb{P}(q) = \underbrace{\sum_{\theta \in \Theta}}_{\text{Exponential!}} \mathbb{P}(\theta) \cdot \frac{\overbrace{N(I_\theta \models q)}^{\text{\#P-complete!}}}{N(\theta)}$$

Inference by approximate compilation:

$$\mathbb{P}(q) = \underbrace{\sum_{\theta \in \Theta}}_{\text{Linear!}} \mathbb{P}(\theta) \cdot \frac{\overbrace{N(I_\theta \models q)}^{\text{Linear!}}}{N(\theta)}$$

Compiling to probabilistic circuits is hard $\times 10$!

dPASP

Programming with Logic and Neural Networks

Renato Lui Geh, Jonas Gonçalves, Igor Cataneo Silveira,
Denis Deratani Mauá, Fabio Gagliardi Cozman, Yuka Machino

GitHub

Learn dPASP!