NeurASP =
Neural Network + ASP

About 40% of the slides in this deck are based on slides by Zhun Yang.

Some neuro-symbolic approaches

- We have some knowledge about the domain that we want to incorporate to our neural learning and reasoning framework. How do we do it?
  - Distillation based approach
  - Compile into the loss function
  - Have a separate symbolic reasoning and filtering layer
- Neural networks that generate code (or a collection of structured facts) which then gets implemented by a program lead to another kind of neuro-symbolic integration.
- Neural models where each neuron has symbolic meaning and the overall network can learn various aspects from noisy data
  - capture logical contradiction
  - learn explainable rules
- Neuro-symbolic approaches in planning and acting domains:
  - partially neural reinforcement learning (RL) framework for the continuous state and action space domain
  - neuro-symbolic architecture that learns state transitions from images

Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence (IJCAI-20)

NeurASP: Embracing Neural Networks into Answer Set Programming

Zhun Yang[1] , **Adam Ishay**[1] and **Joohyung Lee**[1 2 *]
[1] Arizona State University, Tempe, AZ, USA
[2] Samsung Research, Seoul, South Korea
{zyang90, aishay, joolee}@asu.edu

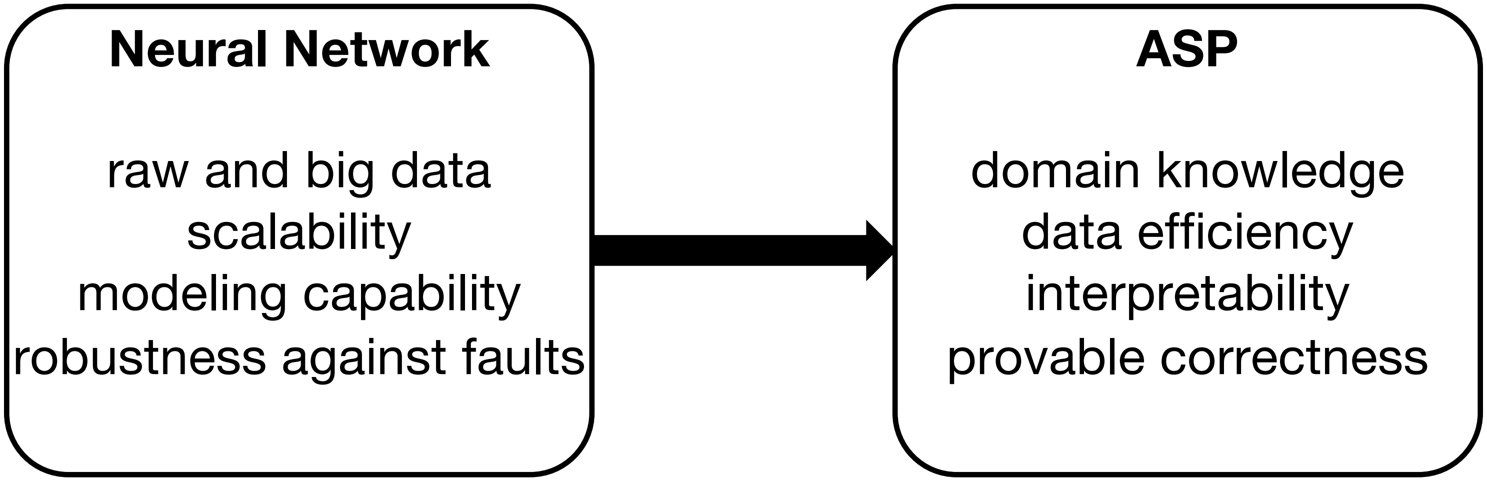
Abstract

We present NeurASP, a simple extension of answer set programs by embracing neural networks. By treating the neural network output as the probability distribution over atomic facts in answer set programs, NeurASP provides a simple and effective way to integrate sub-symbolic and symbolic computation. We demonstrate how NeurASP can make use of a pre-trained neural network in symbolic computation and how it can improve the neural network's perception result by applying symbolic reasoning in answer set programming. Also, NeurASP can make use of ASP rules to train a neural network better so that a neural network not only learns from implicit correlations from the data but also from the explicit complex semantic constraints

of DeepProbLog [Manhaeve *et al.*, 2018], by treating the neural network output as the probability distribution over atomic facts in answer set programs, the proposed NeurASP provides a simple and effective way to integrate sub-symbolic and symbolic computation.

We demonstrate how NeurASP can be useful for some tasks where both perception and reasoning are required. Reasoning can help identify perception mistakes that violate semantic constraints, which in turn can make perception more robust. For example, a neural network for object detection may return a bounding box and its classification "car," but it may not be clear whether it is a real car or a toy car. The distinction can be made by applying reasoning about the relations with the surrounding objects and using commonsense knowledge. When it is unclear whether a round object attached to the car is a wheel or a doughnut, the reasoner could conclude that it is more likely to be a wheel by applying com-

NeurASP



Neural Network

raw and big data
scalability
modeling capability
robustness against faults

ASP

domain knowledge
data efficiency
interpretability
provable correctness

Based on slides by Zhun Yang

Why Answer Set Programs?

- Answer Set Programming (ASP) is a logic programming paradigm.
  - combinatorial search
  - knowledge intensive tasks
- ASP has well-developed foundations, efficient reasoning systems, and a methodology of use tested on a number of industrial applications.
- ASP supports a rich set of KR constructs that allow for convenient representation of complex knowledge.
  - Aggregates
  - Choices rules
  - Default
  - Optimization rules

Based on slides by Zhun Yang

Answer Set Programs – a quick overview

An ASP program is a collection of rules of the form:

$$a_o \leftarrow a_1, \ldots, a_m, \textsf{not } a_{m+1}, \ldots \textsf{not } a_n.$$

where a_is are atoms in the sense of predicate logic. Intuitively, the meaning of the above rule is that if $a_1, \ldots, a_m$ are true and $a_{m+1}, \ldots, a_n$ can be safely assumed to be false, then a_0 must be true. If the right hand side of a rule is empty (i.e., m = n = 0) then we refer to it as a fact. The semantics of ASP programs that do not have **not** in their rules is defined by their unique least model.

ASP Examples

The ancestor program

$$anc(X,Y) \leftarrow par(X,Y).$$
$$anc(X,Y) \leftarrow par(X,Z), anc(Z,Y).$$

$$par(a,b).\ par(b,c).\ par(c,d).\ par(e,f).$$

- Using the above program we not only conclude the facts, but also conclude: *anc(a,b), anc(b,c), anc(c,d), and(e,f), anc(a,c), anc(b,d)*, and *anc(a,d)*.

A key ASP construct – The default “not”

Expressing normal properties: Normally crows are black

$$
\begin{aligned}
&black(X) \leftarrow crow(X), \mathsf{not}\ ab(X).\\
&ab(X) \leftarrow albino\_crow(X).\\
&crow(X) \leftarrow albino\_crow(X).\\
&white(X) \leftarrow albino\_crow(X).
\end{aligned}
$$

$$crow(banjo).$$

$$albino\_crow(willow).$$

we will be able to conclude $black(banjo)$ and $white(willow)$

Semantics of Answer Set Programs

Examples with unique answer sets

- Programs without not: The unique least model. For the following program, it is { }.

  $$p \leftarrow p.$$

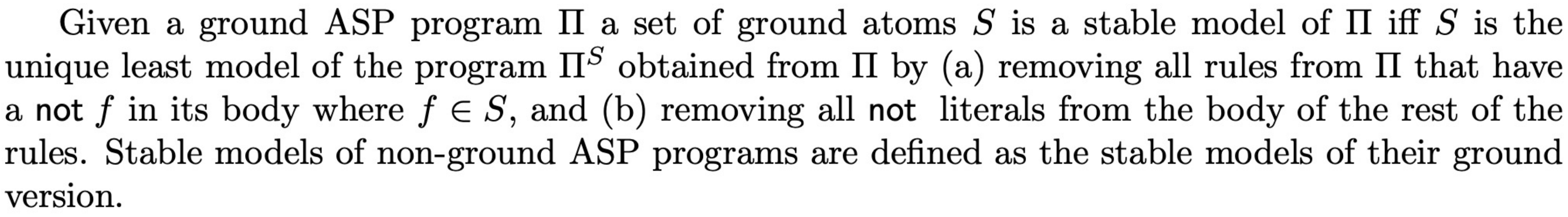
- Some programs with not
  - {a} is the only answer set of the program: $a \leftarrow \textsf{not}\ b.$
  - {q} is the only answer set of the program: $q \leftarrow \textsf{not}\ p.$ $p \leftarrow p.$

Semantics of Answer Set Programs

Examples with multiple answer sets (also called as stable models)

- The following program has two answer sets: {a, p} and {b, p}

$$
\begin{aligned}
&p \leftarrow a.\\
&p \leftarrow b.\\
&a \leftarrow \mathsf{not}\ b.\\
&b \leftarrow \mathsf{not}\ a.
\end{aligned}
$$



Given a ground ASP program Π a set of ground atoms S is a stable model of Π iff S is the unique least model of the program Π^S obtained from Π by (a) removing all rules from Π that have a not f in its body where $f \in S$, and (b) removing all not literals from the body of the rest of the rules. Stable models of non-ground ASP programs are defined as the stable models of their ground version.

ASP Solvers introduce useful syntactic sugar

Choice rules

$$\begin{array}{l} p \leftarrow a. \\ p \leftarrow b. \\ a \leftarrow \text{not } b. \\ b \leftarrow \text{not } a. \end{array}$$

$$\begin{array}{l} p \leftarrow a. \\ p \leftarrow b. \\ 1\{a; b\}1. \end{array}$$

$$\{a; b\} = 1.$$

- More generally: $u\{a_1; a_2; \ldots; a_n\}v.$

- What does this fact mean: $\{val(d) = 0; \ldots; val(d) = 9\} = 1.$

Example Answer Set Program

ASP Program **Π** for Digit Addition

choice rules

```
{ digit(d1)=0 ; … ; digit(d1)=9 }=1.
{ digit(d2)=0 ; … ; digit(d2)=9 }=1.
addition(A, B, N) ← digit(A)=N1,
                    digit(B)=N2,
                    N = N1+N2.
```

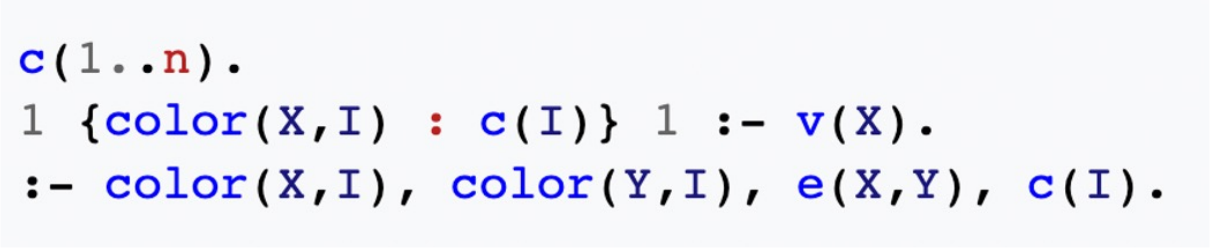
Example Stable Model

```
M = {digit(d1)=3, digit(d2)=0,
     addition(d1,d2,3)}
```

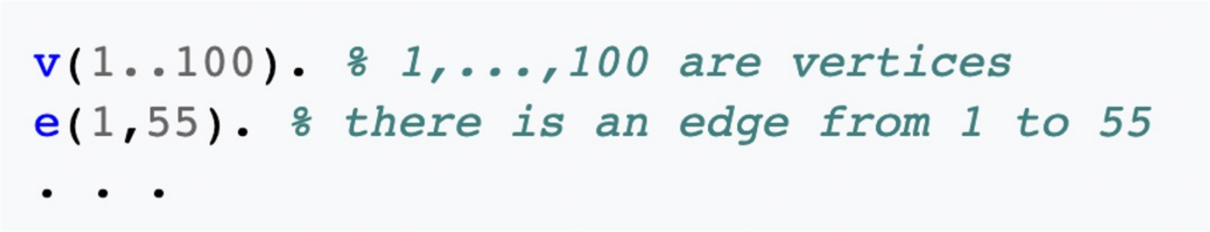
Based on slides by Zhun Yang

Graph Coloring Using ASP

Can the graph be colored by n colors so that no two adjacent vertices have the same color?



```
c(1..n).
1 {color(X,I) : c(I)} 1 :- v(X).
:- color(X,I), color(Y,I), e(X,Y), c(I).
```



```
v(1..100). % 1,...,100 are vertices
e(1,55). % there is an edge from 1 to 55
. . .
```

NeurASP = NN + ASP, but why?

- NeurASP allows one to train a NN under weak supervision.
- Perception and reasoning are separated so as to achieve higher accuracy with less data.
- Reasoning can help identify perception mistakes that violate semantic constraints, which in turn can make perception more robust.
- NeurASP is more elaboration tolerant on visual tasks.
- NeurASP extends classification to context relational classification
- A neural network can be trained together with rules so that it not only learns from implicit correlations from the data but also learns from explicit complex semantic constraints expressed by ASP rules.

Based on slides by Zhun Yang

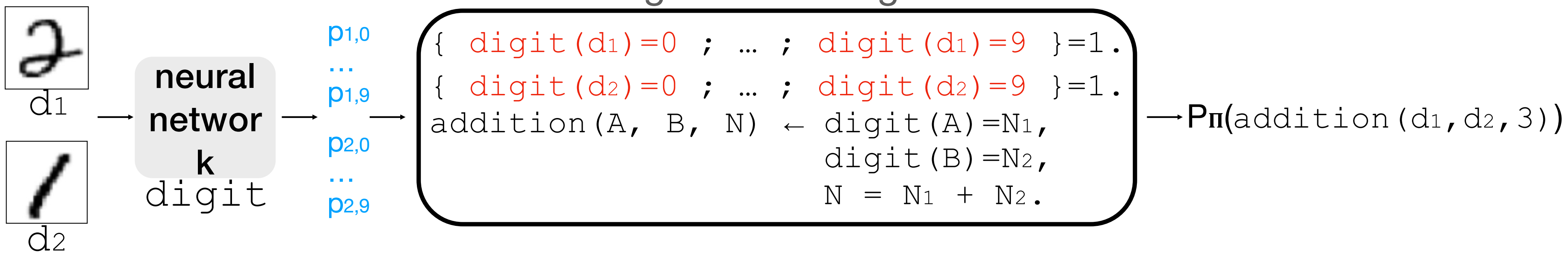
How does NeurASP work?

Based on slides by Zhun Yang

Neural Atom

Links neural network outputs to atoms in an ASP program.

ASP Program **Π** for Digit Addition



d1 → neural network digit → $p_{1,0}$ … $p_{1,9}$ $p_{2,0}$ … $p_{2,9}$ →

```
{ digit(d1)=0 ; … ; digit(d1)=9 }=1.
{ digit(d2)=0 ; … ; digit(d2)=9 }=1.
addition(A, B, N) ← digit(A)=N1,
                    digit(B)=N2,
                    N = N1 + N2.
```

→ Pπ(addition(d1,d2,3))

d2

We use neural network digit to classify the image d1, and the output $p_{1,i}$ for i in {0, …, 9} defines the probability of digit(d1)=i.

neural atom: `nn( digit, d1 ,1, {0,1,2,3,4,5,6,7,8,9} )`

Based on slides by Zhun Yang

NeurASP notations

A NeurASP program combines a set of neural networks with an ASP program. Each neural network is assigned a name, and it takes an input and assigns values (with assigned probabilities) to one or more of its (output) attributes. For example, consider a neural network for MNIST. This neural network will take an image as an input and assign probabilities to the single output attribute *digit* with respect to the values 0 ... 9. In the NeurASP framework this neural network denoted by M_{mnist_nn} will be specified by the following neural atom:

$$nn(mnist\_nn, mnist\_input, 1, \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}).$$

where, *mnist_nn* is the name of the neural network, *mnist_input* refers to an input, 1 denotes the number of attributes, and $\{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ refers to the value the attribute can take. The neural network M_{mnist_nn} will then define a probability function denoted by $M_{mnist\_nn}(mnist\_input)[1, v]$ such that $\sum_{v \in \{0,...,9\}} M_{mnist\_nn}(mnist\_input)[1, v] = 1$.

NeurASP – Sudoku

Now consider a neural network M_{sudoku_nn} that takes an image of the initial state of a Sudoku game and assigns probabilities to the 81 output attributes, each expressing what is the value in one of the 81 squares, with respect to the values *empty*, and 1 … 9. In the NeurASP framework this neural network M_{sudoku_nn} will be specified by the following neural atom:

$$nn(sudoku\_nn, sudoku\_input, 81, \{empty, 1, 2, 3, 4, 5, 6, 7, 8, 9\}).$$

where, *sudoku_nn* is the name of the neural network, *sudoku_input* refers to an input, 81 denotes the number of attributes, and $\{empty, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ refers to the value the attributes can take. The neural network M_{sudoku_nn} will then define a probability function denoted by $M_{sudoku\_nn}(sudoku\_input)[i, v]$ such that
for $1 \leq i \leq 81$, $\sum_{v \in \{empty,1,\ldots,9\}} M_{sudoku\_nn}(sudoku\_input)[i, v] = 1$.

NeurASP Syntax

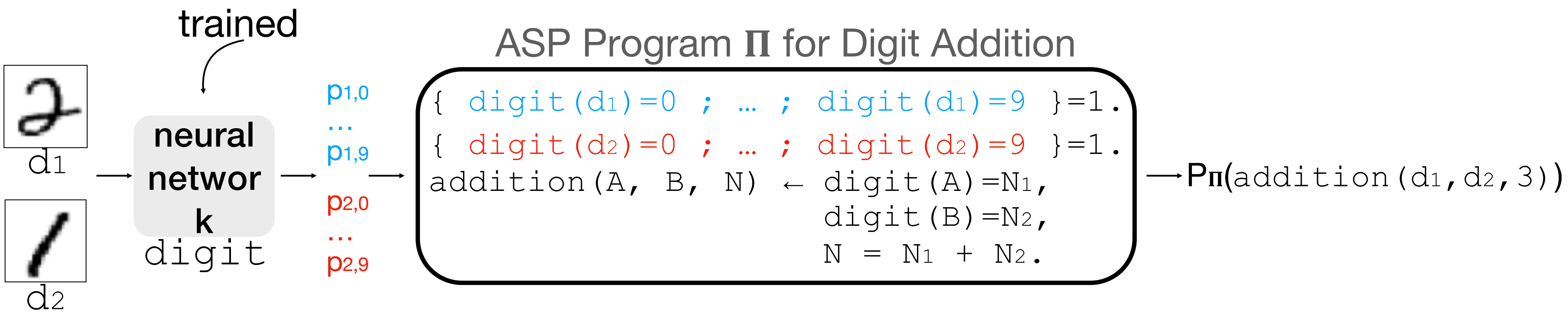
In general, a neural network M_{name} is specified by a neural atom of the form

$$nn(name, input, no\_attr, value\_set)$$

where $name$ is the name of the neural network, $input$ is the input to the neural network, no_attr refers to the number of attributes that neural network is trying to assign values, and $value\_set$ is a set of values that each of the attributes can take. The neural network M_{name} then defines a probability function denoted by $M_{name}(input)[no\_attr, v]$ such that
for $1 \leq i \leq no\_attr$, $\sum_{v \in value\_set} M_{name}(input)[i, v] = 1$.

NeurASP Inference

Infer the probabilities of stable models and formulas.



```
{ digit(d1)=0 ; … ; digit(d1)=9 }=1.
{ digit(d2)=0 ; … ; digit(d2)=9 }=1.
addition(A, B, N) ← digit(A)=N1,
                    digit(B)=N2,
                    N = N1 + N2.
```

$p_{1,i}$ $p_{2,j}$

- Suppose $M_{i,j} = \{\mathtt{digit}(d_1)=i,\ \mathtt{digit}(d_2)=j,\ \mathtt{addition}(d_1,d_2,i+j)\}$
  - $P_\Pi(M_{i,j}) = p_{1,i} \times p_{2,j}$ (for i, j in {0, …, 9})
  - $P_\Pi(\mathtt{addition}(d_1,d_2,3)) = P_\Pi(M_{0,3}) + P_\Pi(M_{1,2}) + P_\Pi(M_{2,1}) + P_\Pi(M_{3,0})$

Based on slides by Zhun Yang

Semantics of NeurASP — Stable Model

The semantics of NeurASP defines a ***stable model*** and its ***probability*** orienting from the NN outputs.

For each NeurASP program Π, we obtain its ASP counterpart Π' by replacing each neural atom

$$nn(m, t, e, \{v_1, \ldots, v_n\})$$

with the set of choice rules

$$\{\ m_i(t)=v_1\ ;\ \ldots\ ;\ m_i(t)=v_n\ \}=1 \qquad \text{for } i \in \{1, \ldots, e\}.$$

- The ***stable models*** of Π are the stable models of Π'.

EX.

Let $\mathcal{M}_{digit}$ be a neural network that classifies an MNIST digit image. Consider the NeurASP program Π below.

```
nn( digit,d,1,{0,1,2,3,4,5,6,7,8,9})
```

Its ASP counterpart Π' is

$$\{\ digit_1(d)=0\ ;\ \ldots\ ;\ digit_1(d)=9\ \}=1$$

Thus the stable models of Π are the 10 stable models of Π' below.

$$\{digit_1(d)=0\} \quad \{digit_1(d)=1\} \quad \ldots \quad \{digit_1(d)=9\}$$

Based on slides by Zhun Yang

Semantics of NeurASP — Probability

Recall that we turn each neural atom $nn(m, t, e, \{v_1, \dots, v_n\})$ into the set of choice rules

$\{ m_i(t)=v_1 ; \dots ; m_i(t)=v_n \}=1$ for $i \in \{1, \dots, e\}$.

- Let σ^{nn} denote the set of atoms $m_i(t)=v_j$ obtained from neural atoms as described above.
- With the input tensor (identified by) t, we assume neural network $\mathcal{M}$ (identified by m) outputs a matrix in $\mathbb{R}^{e \times n}$. The n numbers in the i-th row define the probability distribution of the following n atoms.

$$m_i(t)=v_1 , m_i(t)=v_2 , \dots , m_i(t)=v_n$$

- The ***probability*** of a stable model $\mathcal{I}$ of $\mathbf{\Pi}$ is the product of the probabilities of all atoms in $\mathcal{I} \cap \sigma^{nn}$.

EX.

Consider the NeurASP program $\mathbf{\Pi}$ below.

$nn(digit, d, 1, \{0,1,2,3,4,5,6,7,8,9\})$

Suppose the output of $\mathcal{M}$ given input tensor d is $[0, 0, 0.3, 0, 0, 0, 0, 0.6, 0, 0.1]$, then

the probability of $\mathcal{I}_0=\{digit_1(d)=0\}$ is $P_{\Pi}(\mathcal{I}_0) = 0$

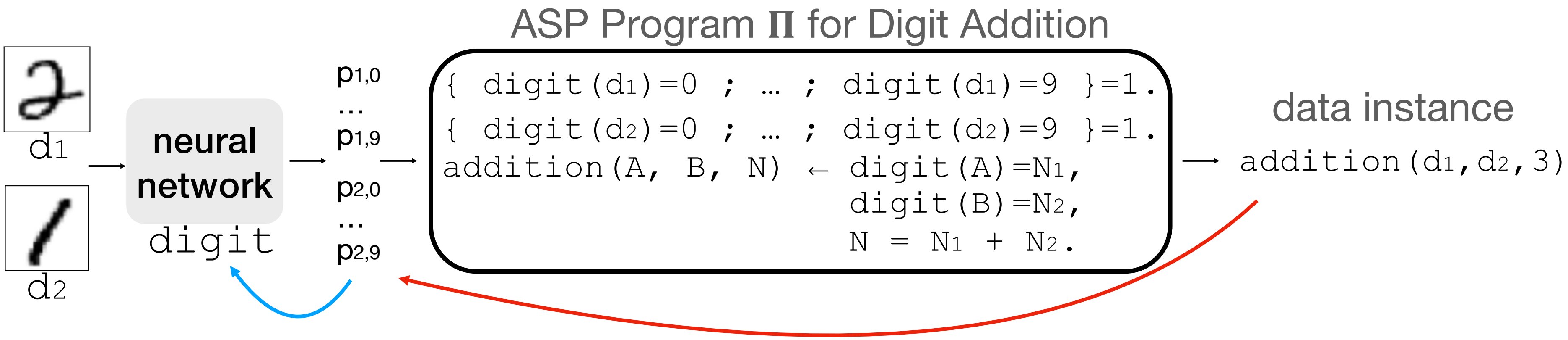
…

the probability of $\mathcal{I}_9=\{digit_1(d)=9\}$ is $P_{\Pi}(\mathcal{I}_9) = 0.1$

Based on slides by Zhun Yang

NeurASP Learning

Back-propagate gradients to the neural network through the chain rule.



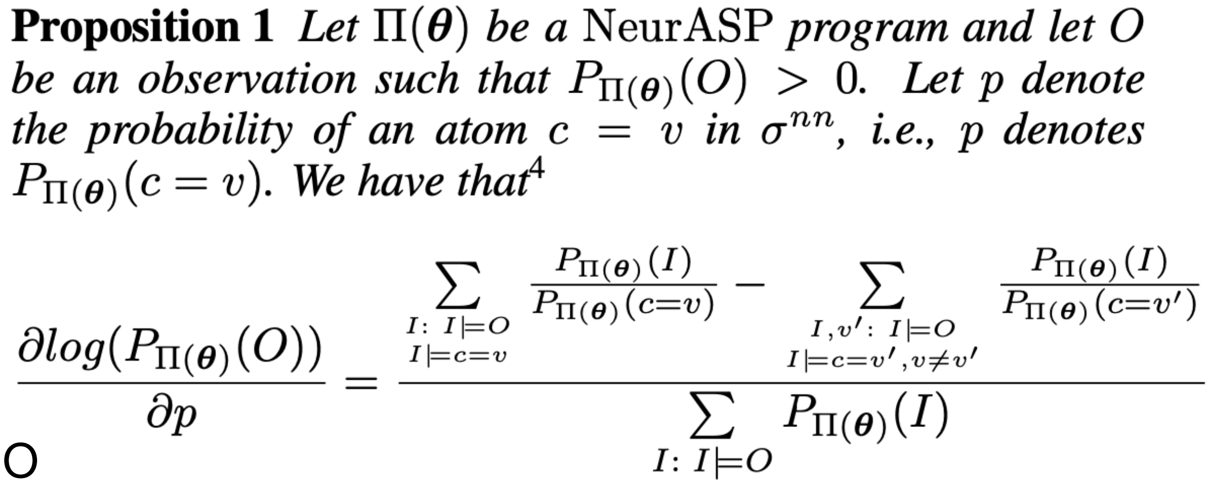
ASP Program Π for Digit Addition

```
{ digit(d1)=0 ; … ; digit(d1)=9 }=1.
{ digit(d2)=0 ; … ; digit(d2)=9 }=1.
addition(A, B, N) ← digit(A)=N1,
                    digit(B)=N2,
                    N = N1 + N2.
```

$$\frac{\partial P_{\Pi}(addition(d_1,d_2,3))}{\partial \boldsymbol{\theta}} = \sum_{\substack{i\in\{1,2\} \\ j\in\{0,\dots,9\}}} \boxed{\frac{\partial P_{\Pi}(addition(d_1,d_2,3))}{\partial p_{i,j}}} \times \boxed{\frac{\partial p_{i,j}}{\partial \boldsymbol{\theta}}}$$

Based on slides by Zhun Yang

Gradients Computation



Proposition 1 *Let* $\Pi(\boldsymbol{\theta})$ *be a* NeurASP *program and let* O *be an observation such that* $P_{\Pi(\boldsymbol{\theta})}(O) > 0$. *Let* p *denote the probability of an atom* $c = v$ *in* σ^{nn}, *i.e.,* p *denotes* $P_{\Pi(\boldsymbol{\theta})}(c = v)$. *We have that*[4]

$$\frac{\partial log(P_{\Pi(\boldsymbol{\theta})}(O))}{\partial p} = \frac{\sum\limits_{\substack{I:\ I \models O \\ I \models c=v}} \frac{P_{\Pi(\boldsymbol{\theta})}(I)}{P_{\Pi(\boldsymbol{\theta})}(c=v)} - \sum\limits_{\substack{I, v':\ I \models O \\ I \models c=v', v \neq v'}} \frac{P_{\Pi(\boldsymbol{\theta})}(I)}{P_{\Pi(\boldsymbol{\theta})}(c=v')}}{\sum\limits_{I:\ I \models O} P_{\Pi(\boldsymbol{\theta})}(I)}$$

Given

- a NeurASP program Π
- an observation O,

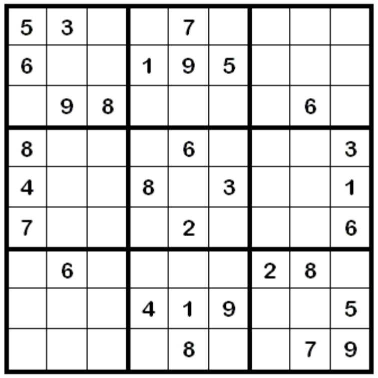
for each NN output p for atom c=v, we

1. find all stable models I of ΠUO
2. compute the probability $P_{Π}$(I)
3. use Prop 1 to compute $\frac{\partial log(P_{\Pi(\boldsymbol{\theta})}(O))}{\partial p}$
4. use the chain rule to further backward to NN parameters

$$\frac{\partial \sum\limits_{O \in \mathbf{O}} log(P_{\Pi(\boldsymbol{\theta})}(O))}{\partial \boldsymbol{\theta}} = \sum_{O \in \mathbf{O}} \frac{\partial log(P_{\Pi(\boldsymbol{\theta})}(O))}{\partial \mathbf{p}} \times \frac{\partial \mathbf{p}}{\partial \boldsymbol{\theta}}$$

Based on slides by Zhun Yang

Example NeurASP Program: Sudoku



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

`img`

Task: given an image of Sudoku board, predict the solution.

- Use NN `identify` to identify the digits in each of the 81 grid cells.

```
nn(identify, img, 81,{empty,1,2,3,4,5,6,7,8,9}).
```

- Assign one number to each cell `i` for `i` ∈ {1, …, 81}.

```
a(R,C,N) ← identify_i(img)=N, R=i/9, C=i\9, N≠empty.
{a(R,C,1); …; a(R,C,9)}=1 ← identify_i(img)=empty, R=i/9, C=i\9.
```

- No number repeats in the same row, column, and 3×3 box.

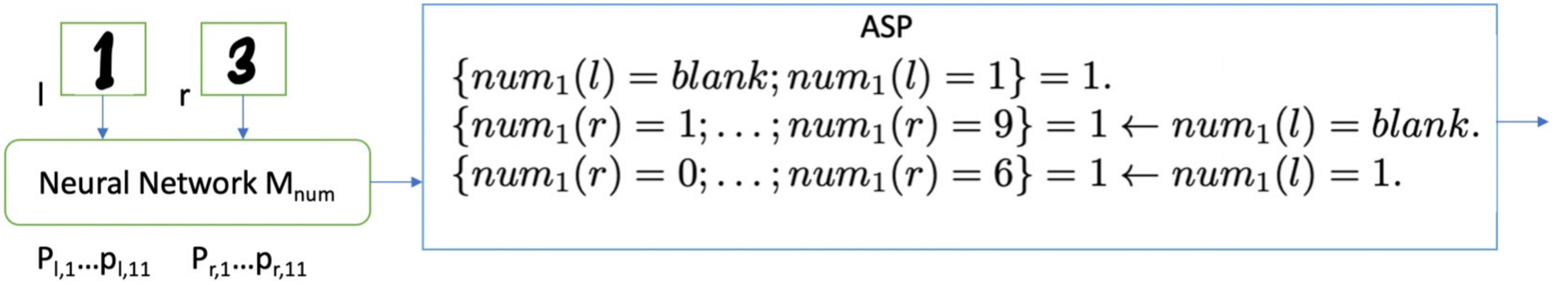
```
← a(R,C_1,N), a(R,C_2,N), C_1≠C_2.
← a(R_1,C,N), a(R_2,C,N), R_1≠R_2.
← a(R_1,C_1,N), a(R_2,C_2,N), R_1≠R_2, C_1≠C_2, ((R_1/3)×3+C_1/3) = ((R_2/3)×3+C_2/3).
```

Based on slides by Zhun Yang

Reading a number between 1 to 16

$nn(num, X, 1, \{blank, 0, \ldots, 9\}) \leftarrow img(X).$

Knowing that the two numerals that the model is reading represents a number between 1 to 16, we can express that knowledge as follows: (a) the image l is either blank, or a one and (b) the image r is (i) between 1 to 9, if the image l is a blank, and (ii) between 0 to 6, if if the image l is a one. This is expressed as the ASP program in Figure 1.



$$\{num_1(l) = blank; num_1(l) = 1\} = 1.$$
$$\{num_1(r) = 1; \ldots; num_1(r) = 9\} = 1 \leftarrow num_1(l) = blank.$$
$$\{num_1(r) = 0; \ldots; num_1(r) = 6\} = 1 \leftarrow num_1(l) = 1.$$

Figure 1: A NeurASP program illustration showing two numeral inputs, a neural network whose output feeds into an ASP program.

In Figure 1, even if the neural network originally had assigned a higher probability to $num_1(r) = 8$, the ASP program will eliminate that possibility in the answer sets where $num_1(l) = 1$.

Kubok-16 puzzle

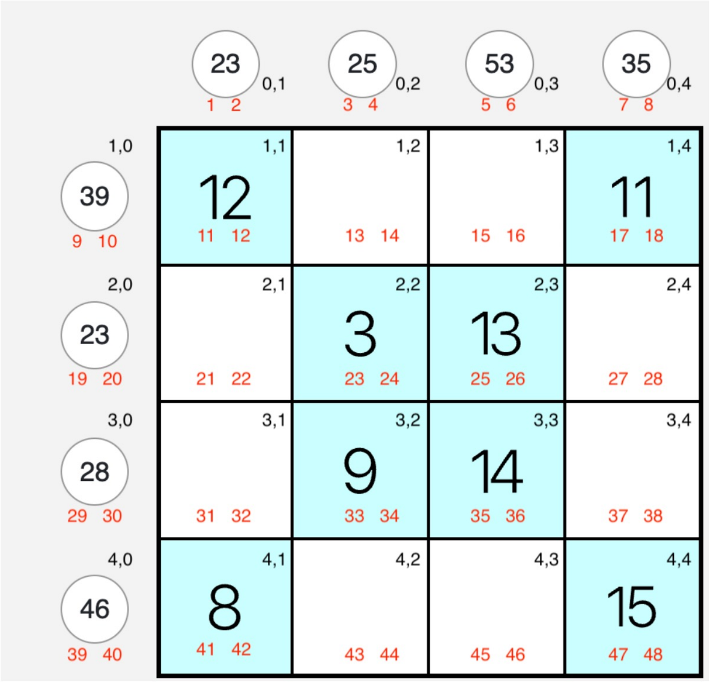


Figure 2: A Kubok-16 puzzle instance with X,Y co-ordinates added to it. The 48 output attributes are also numbered from 1 to 48.

$$nn(kubok, img, 48, \{blank, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}).$$

Kubok-16 example continued

$$nn(kubok, img, 48, \{blank, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}).$$

where, *kubok* is the name of the neural network, *img* refers to an input, 48 denotes the number of attributes, and $\{blank, 0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ refers to the 11 values the attributes can take. The neural network M_{kubok} will then define a probability function denoted by $M_{kubok}(img)[i, v]$ such that
for $1 \leq i \leq 48$, $\sum_{v \in \{blank, 0, \ldots, 9\}} M_{kubok}(img)[i, v] = 1$.

The above neural atom would be translated to the following ASP rules.

$$\{kubok(X, blank); kubok(X, 0); \ldots kubok(X, 9)\} = 1 \leftarrow 1 <= X, X <= 48.$$

ASP Code for Kubok-16

- kubok(X,0) :- kubok(X,blank). % blank is viewed as 0.

- val(X,Y,Z) :- kubok(M,U), kubok(N,V), Z = 10*U + V, N = 10*X+2*Y, M = N-1, U!= blank, V !=blank. % This rule computes the number in the various X,Y co-ordinates based on the numeral corresponding to each of the 48 attributes of the image.

- num4(1..4). num16(1..16).

- {val(X,Y,Z): Z <=16, num16(Z)} = 1 :- X <=4, Y <=4, num4(X), num4(Y). % This choice rule states that each square in the 4 X 4 grid should have a **unique** number between 1 to 16.

- {val(X,Y,Z): X <=4, Y <=4, num4(X), num4(Y)} = 1 :- Z <=16, num16(Z). % This choice rule states that for any number between 1 to 16 **exactly one** square in the 4 X 4 grid will have that number.

- result(X,0, S) :- S = #sum{ Z : val(X,Y,Z) , num4(Y) }, num4(X).

result(0,Y, S) :- S = #sum{ Z : val(X,Y,Z) , num4(X) }, num4(Y). % The above two rules compute the sum of the numbers in each row and column

- :- result(0,Y,Z), val(0,Y,ZZ), Z != ZZ.

:- result(X,0,Z), val(X,0,ZZ), Z != ZZ. % The above two rules filter out the possible assignments where the computed row or the column sum does not match with what is given.

What’s the benefit of NeurASP?

Based on slides by Zhun Yang

Higher Accuracy with Less Data

1. Perception and reasoning are separated so as to achieve higher accuracy with less data. (Neuro-symbolic Concept Learner by Mao et al. 2019)

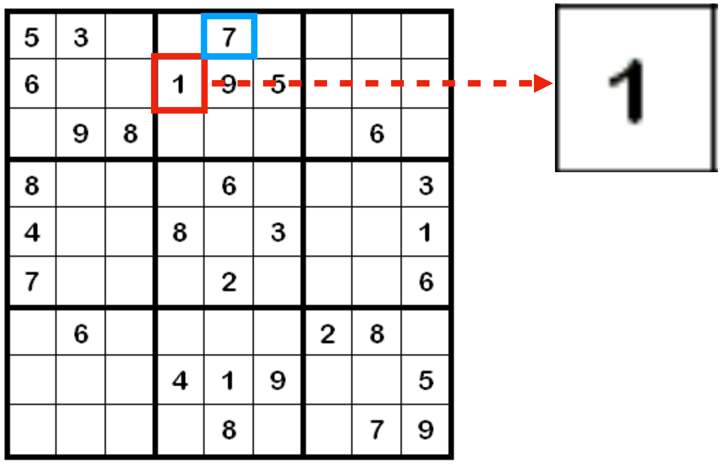
	Method	Input	Number of **Sudoku** Data for Training	Accuracy of Solution
NeurASP	Convolutional Neural Network + ASP	**Image** of Sudoku	**25**	100%
(Park 2018)	Convolutional Neural Network	**Text** Representation of Sudoku (9×9 numbers)	**1 Million**	70.0%
(Palm et al. 2018)	Graph Neural Network	**Text** Representation of Sudoku (9×9 numbers)	**180,000**	96.6%

Based on slides by Zhun Yang

Prune out Perception Errors

2. Reasoning can help identify perception mistakes that violate semantic constraints, which in turn can make perception more robust.

7 v.s. 1



5	3			7				
6			1	9	5			
	9	8					6	
8				6				3
4			8		3			1
7				2				6
	6					2	8	
			4	1	9			5
				8			7	9

Based on slides by Zhun Yang

Based on slides by Zhun Yang

Prune out Perception Errors

2. Reasoning can help identify perception mistakes that violate semantic constraints, which in turn can make perception more robust.

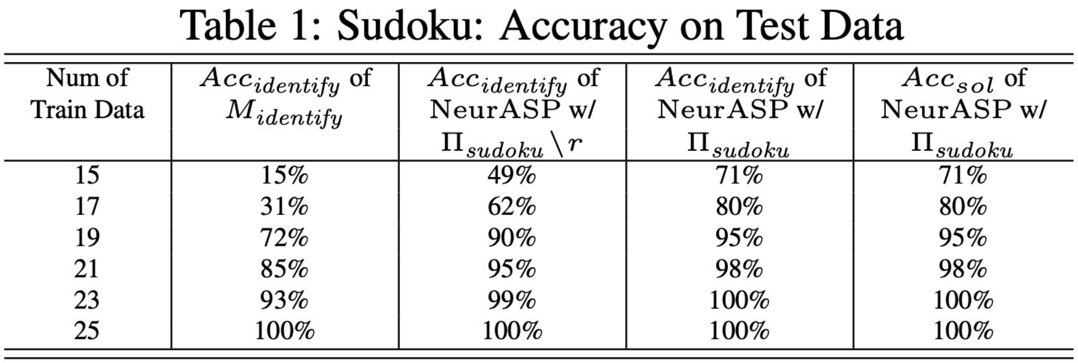
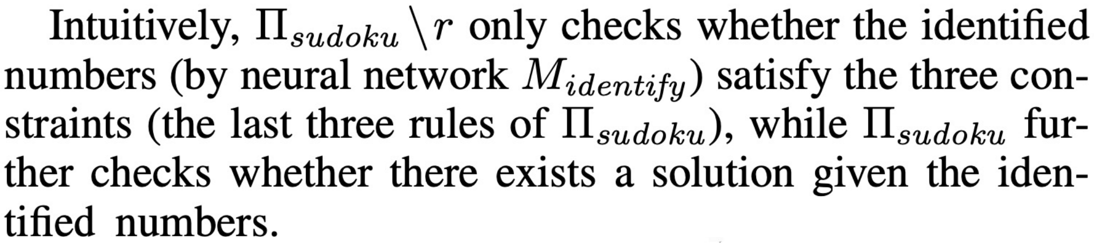


Table 1: Sudoku: Accuracy on Test Data

Num of Train Data	$Acc_{identify}$ of $M_{identify}$	$Acc_{identify}$ of NeurASP w/ $\Pi_{sudoku} \setminus r$	$Acc_{identify}$ of NeurASP w/ Π_{sudoku}	Acc_{sol} of NeurASP w/ Π_{sudoku}
15	15%	49%	71%	71%
17	31%	62%	80%	80%
19	72%	90%	95%	95%
21	85%	95%	98%	98%
23	93%	99%	100%	100%
25	100%	100%	100%	100%



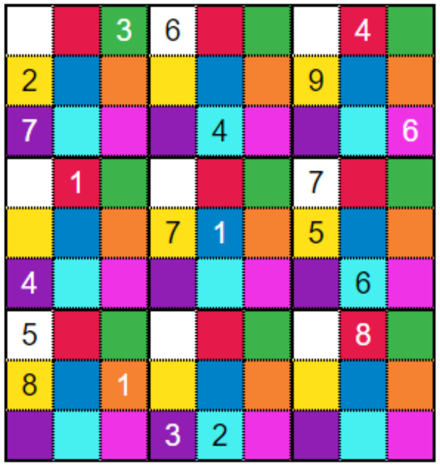
Intuitively, $\Pi_{sudoku} \setminus r$ only checks whether the identified numbers (by neural network $M_{identify}$) satisfy the three constraints (the last three rules of Π_{sudoku}), while Π_{sudoku} further checks whether there exists a solution given the identified numbers.

```
{a(R,C,1); …; a(R,C,9)}=1 ← identifyᵢ(img)=empty, R=i/9, C=i\9.
```

Elaboration Tolerant

3. NeurASP can be easily applied to elaborations of a task.

Offset Sudoku



		3	6				4	
2						9		
7				4				6
	1					7		
			7	1		5		
4							6	
5							8	
8		1						
			3	2				

[Offset Sudoku] No number repeats at the same relative position in 3*3 boxes

```
:- a(R1,C1,N), a(R2,C2,N), R1\3 = R2\3,
   C1\3 = C2\3, R1 != R2, C1 != C2.
```

[Anti-knight Sudoku] No number repeats at a knight move

```
:- a(R1,C1,N), a(R2,C2,N), |R1-R2|+|C1-C2|=3.
```

[Sudoku-X] No number repeats at the diagonals

```
:- a(R1,C1,N), a(R2,C2,N), R1=C1, R2=C2, R1!=R2.
:- a(R1,C1,N), a(R2,C2,N), R1+C1=8, R2+C2=8, R1!=R2.
```

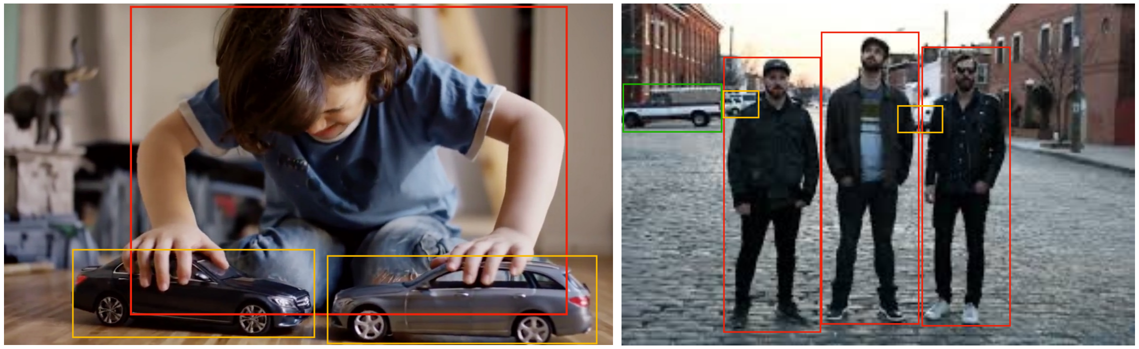
Based on slides by Zhun Yang

Based on slides by Zhun Yang

Context Relational Classification

4. NeurASP extends classification to context relational classification

Q: What are the cars and toy-cars in these images?



- By default, we believe person is smaller than car.

```
smaller(B,B′) ← label(B)=person, label(B′)=car, not ~smaller(B,B′).
```

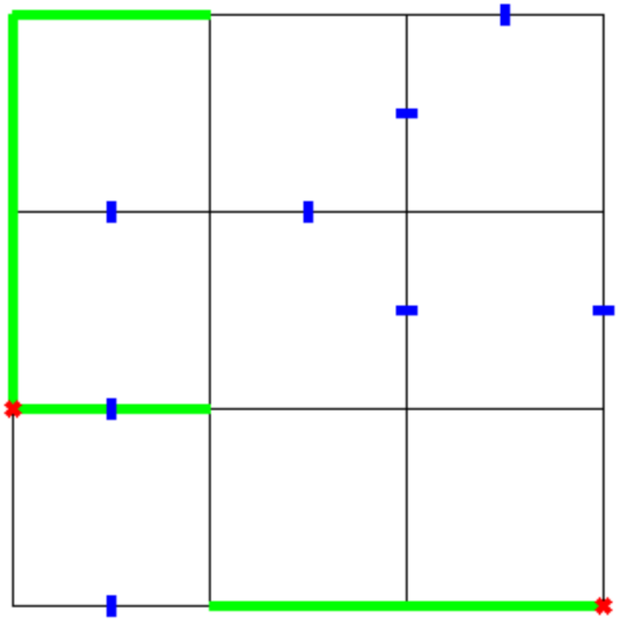
- On the other hand, there are some exceptions.

```
~smaller(B,B′) ← box(B,X1,Y1,X2,Y2), box(B′,X1′,Y1′,X2′,Y2′),
                 Y2 ≤ Y2′, |X1−X2|×|Y1−Y2| > |X1′−X2′|×|Y1′−Y2′|.
toy(B′) ← label(B)=person, label(B′)=car, smaller(B′,B).
```

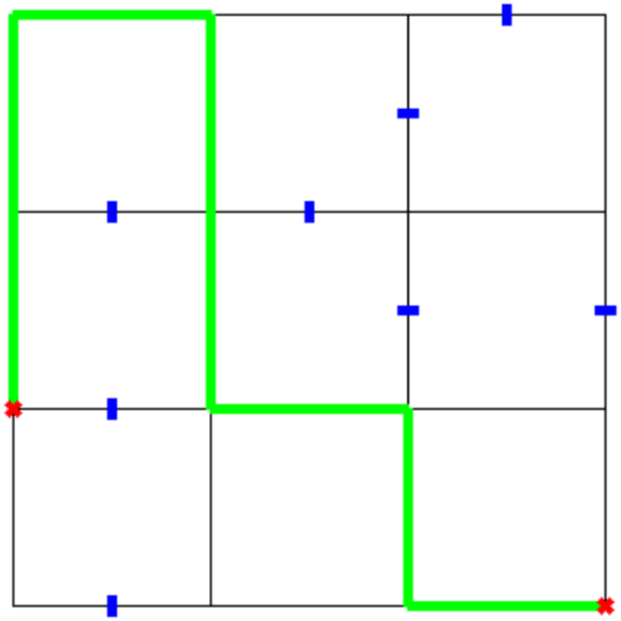
Based on slides by Zhun Yang

Semantic Regularizer

5. A neural network can be trained together with rules so that it not only learns from implicit correlations from the data but also learns from explicit complex semantic constraints expressed by ASP rules.



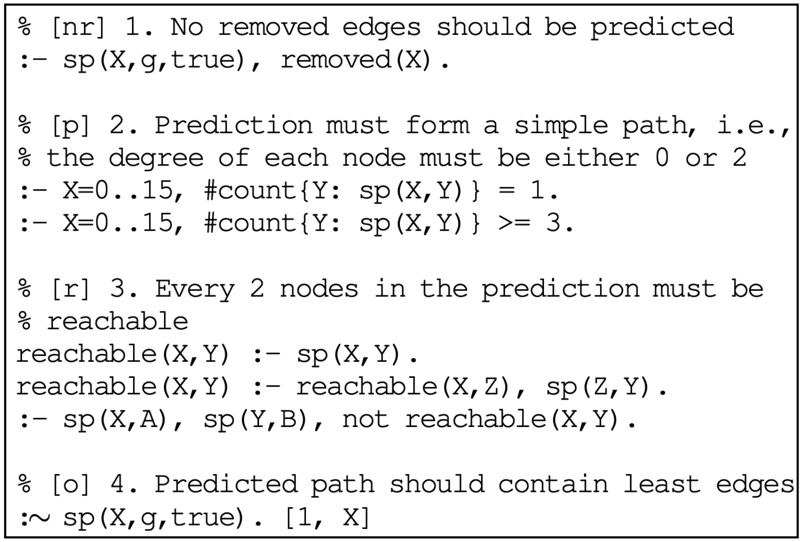
Multi-Layer Perception
(Cross-Entropy)



Multi-Layer Perception
(NeurASP)

Semantic Regularizer

5. A neural network can be trained together with rules so that it not only learns from implicit correlations from the data but also learns from explicit complex semantic constraints expressed by ASP rules.



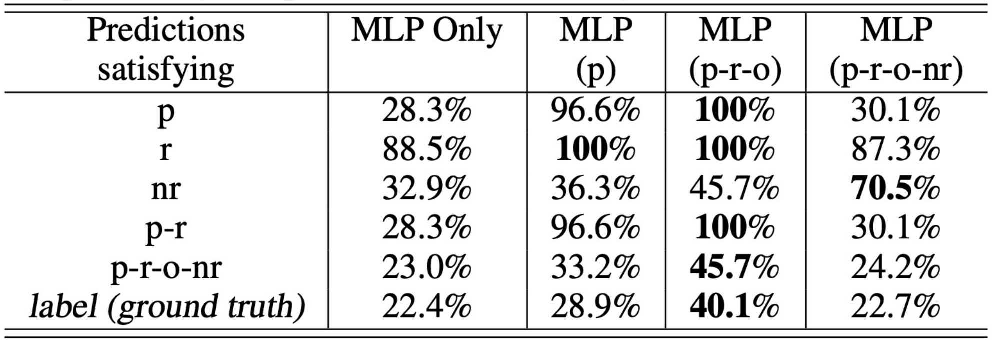
```
% [nr] 1. No removed edges should be predicted
:- sp(X,g,true), removed(X).

% [p] 2. Prediction must form a simple path, i.e.,
% the degree of each node must be either 0 or 2
:- X=0..15, #count{Y: sp(X,Y)} = 1.
:- X=0..15, #count{Y: sp(X,Y)} >= 3.

% [r] 3. Every 2 nodes in the prediction must be
% reachable
reachable(X,Y) :- sp(X,Y).
reachable(X,Y) :- reachable(X,Z), sp(Z,Y).
:- sp(X,A), sp(Y,B), not reachable(X,Y).

% [o] 4. Predicted path should contain least edges
:~ sp(X,g,true). [1, X]
```

Table 2: Shortest Path: Accuracy on Test Data: columns denote MLPs trained with different rules; each row represents the percentage of predictions that satisfy the constraints



Predictions satisfying	MLP Only	MLP (p)	MLP (p-r-o)	MLP (p-r-o-nr)
p	28.3%	96.6%	**100%**	30.1%
r	88.5%	**100%**	**100%**	87.3%
nr	32.9%	36.3%	45.7%	**70.5%**
p-r	28.3%	96.6%	**100%**	30.1%
p-r-o-nr	23.0%	33.2%	**45.7%**	24.2%
label (ground truth)	22.4%	28.9%	**40.1%**	22.7%

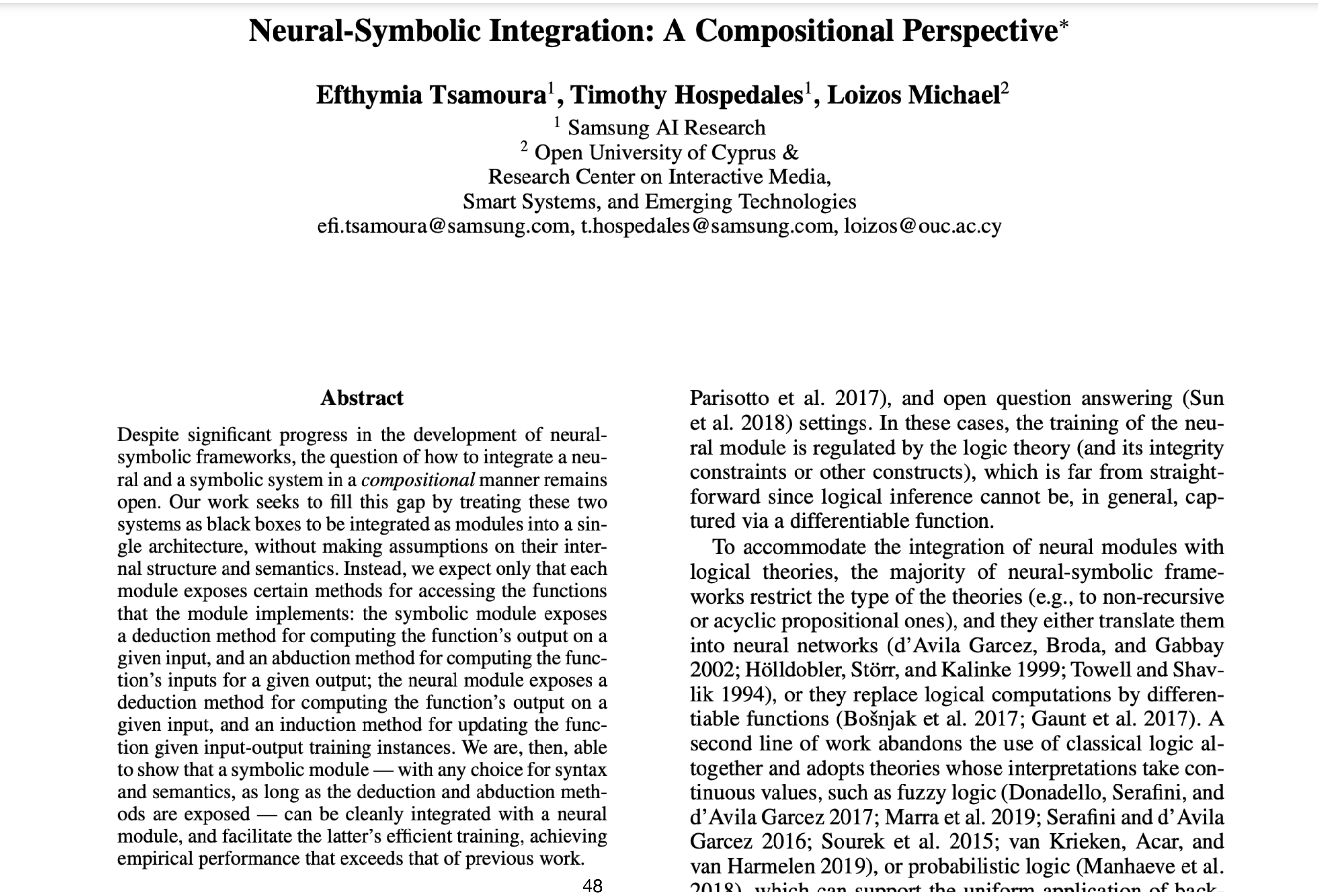
Based on slides by Zhun Yang

Challenges of NeurASP

- Training with NeurASP takes much more time than pure NN training due to exact inference — NeurASP uses clingo to ground the whole program and enumerate all stable models.
- Scalability can be addressed using approximate inference instead of exact inference
- The interface between the neural part and the ASP part needs to be manually developed
- Codes for NeurASP and experiments are available at
  - https://github.com/azreasoners/NeurASP

Based on slides by Zhun Yang

Follow-up works by others



Neural-Symbolic Integration: A Compositional Perspective*

Efthymia Tsamoura[1], Timothy Hospedales[1], Loizos Michael[2]
[1] Samsung AI Research
[2] Open University of Cyprus &
Research Center on Interactive Media,
Smart Systems, and Emerging Technologies
efi.tsamoura@samsung.com, t.hospedales@samsung.com, loizos@ouc.ac.cy

Abstract

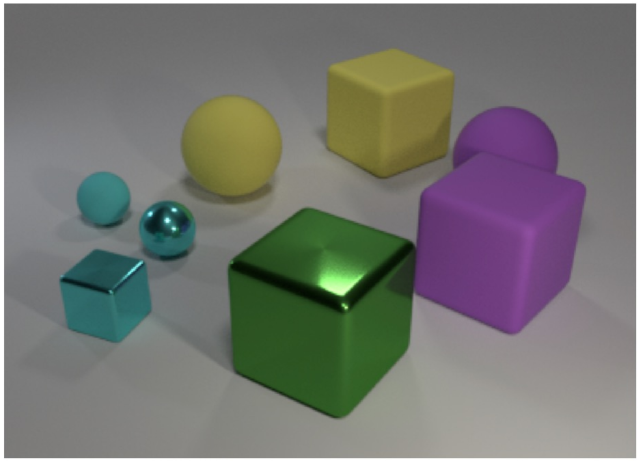
Despite significant progress in the development of neural-symbolic frameworks, the question of how to integrate a neural and a symbolic system in a *compositional* manner remains open. Our work seeks to fill this gap by treating these two systems as black boxes to be integrated as modules into a single architecture, without making assumptions on their internal structure and semantics. Instead, we expect only that each module exposes certain methods for accessing the functions that the module implements: the symbolic module exposes a deduction method for computing the function's output on a given input, and an abduction method for computing the function's inputs for a given output; the neural module exposes a deduction method for computing the function's output on a given input, and an induction method for updating the function given input-output training instances. We are, then, able to show that a symbolic module — with any choice for syntax and semantics, as long as the deduction and abduction methods are exposed — can be cleanly integrated with a neural module, and facilitate the latter's efficient training, achieving empirical performance that exceeds that of previous work.

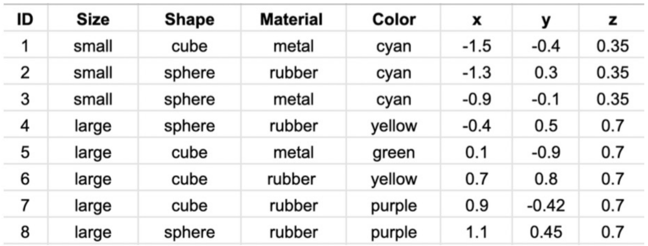
Parisotto et al. 2017), and open question answering (Sun et al. 2018) settings. In these cases, the training of the neural module is regulated by the logic theory (and its integrity constraints or other constructs), which is far from straightforward since logical inference cannot be, in general, captured via a differentiable function.

To accommodate the integration of neural modules with logical theories, the majority of neural-symbolic frameworks restrict the type of the theories (e.g., to non-recursive or acyclic propositional ones), and they either translate them into neural networks (d'Avila Garcez, Broda, and Gabbay 2002; Hölldobler, Störr, and Kalinke 1999; Towell and Shavlik 1994), or they replace logical computations by differentiable functions (Bošnjak et al. 2017; Gaunt et al. 2017). A second line of work abandons the use of classical logic altogether and adopts theories whose interpretations take continuous values, such as fuzzy logic (Donadello, Serafini, and d'Avila Garcez 2017; Marra et al. 2019; Serafini and d'Avila Garcez 2016; Sourek et al. 2015; van Krieken, Acar, and van Harmelen 2019), or probabilistic logic (Manhaeve et al. 2018), which can support the uniform application of back-

Additional Applications of Neuro-Symbolic Reasoning

Visual Question Answering (Yi et al. 2018)



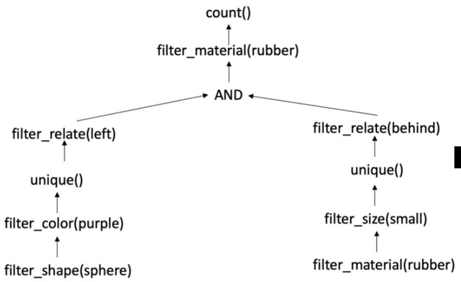


ID	Size	Shape	Material	Color	x	y	z
1	small	cube	metal	cyan	-1.5	-0.4	0.35
2	small	sphere	rubber	cyan	-1.3	0.3	0.35
3	small	sphere	metal	cyan	-0.9	-0.1	0.35
4	large	sphere	rubber	yellow	-0.4	0.5	0.7
5	large	cube	metal	green	0.1	-0.9	0.7
6	large	cube	rubber	yellow	0.7	0.8	0.7
7	large	cube	rubber	purple	0.9	-0.42	0.7
8	large	sphere	rubber	purple	1.1	0.45	0.7

Structural scene representation of the CLEVR image

Program Executor

Question in Natural Language



Answering Compositional Questions

(Gupta et al. ICLR 2019)

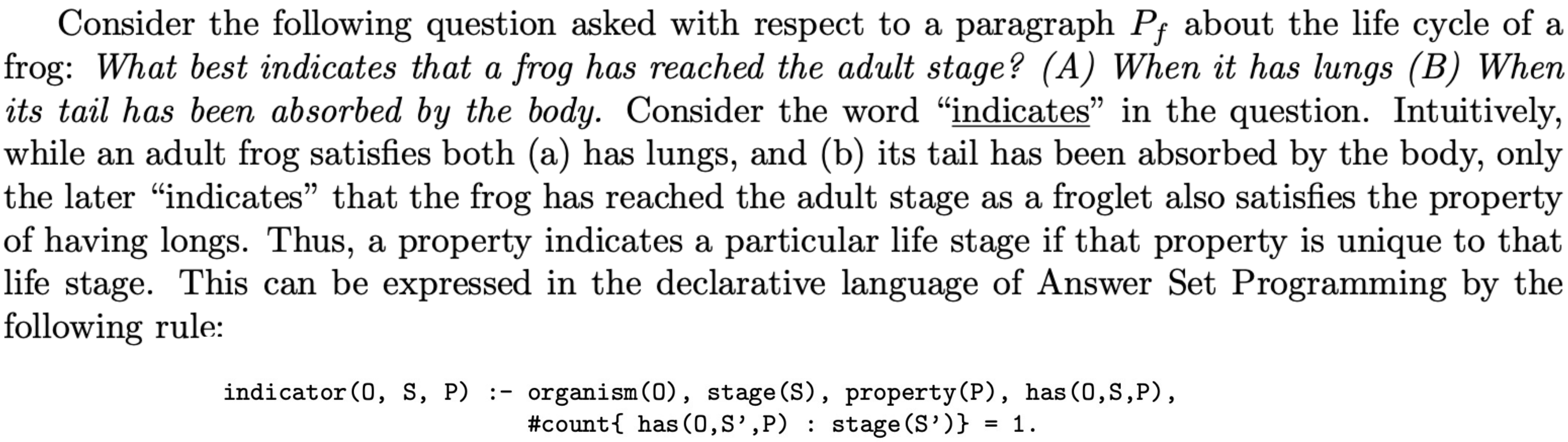
- Suppose “Who kicked the longest field goal in the second quarter” (Q) is asked with respect to a paragraph P.
- To answer this question, the question is translated to the following program using a neural module:

  relocate(Q,find-max-num(filter(Q,P))).

- Symbolic Program Executor executes the above:
  - first filter(Q,P) is executed to find a span P1 of P relevant to the question Q.
  - Next when find-max-num(P1) is executed a span P2 of P1 is identified that is associated with the largest number.
  - Then relocate(Q,P2) is executed to find who kicked the field goal, which earlier had been identified as the longest.

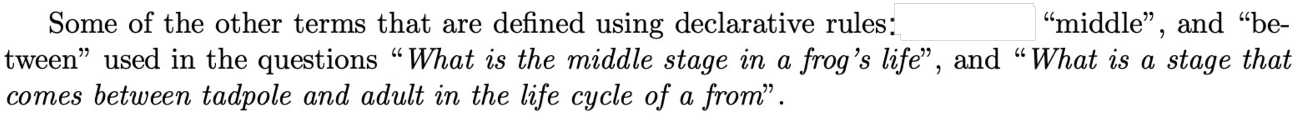
Declarative Reasoning over Text

(Mitra et al. AAAI 2019)



Consider the following question asked with respect to a paragraph P_f about the life cycle of a frog: *What best indicates that a frog has reached the adult stage? (A) When it has lungs (B) When its tail has been absorbed by the body.* Consider the word "indicates" in the question. Intuitively, while an adult frog satisfies both (a) has lungs, and (b) its tail has been absorbed by the body, only the later "indicates" that the frog has reached the adult stage as a froglet also satisfies the property of having longs. Thus, a property indicates a particular life stage if that property is unique to that life stage. This can be expressed in the declarative language of Answer Set Programming by the following rule:

```
indicator(O, S, P) :- organism(O), stage(S), property(P), has(O,S,P),
                      #count{ has(O,S',P) : stage(S')} = 1.
```



Some of the other terms that are defined using declarative rules: "middle", and "between" used in the questions "*What is the middle stage in a frog's life*", and "*What is a stage that comes between tadpole and adult in the life cycle of a from*".

Problem Solving – Grid Puzzles

Mitra & Baral, EMNLP 2015

- Answer Set Programming (ASP) has been used to formulate various problem solving tasks such as planning, constraint satisfaction problems (such as Nqueens), games (such as Sudoku, Kubok-16), and combinatorial grid puzzles
  - The clues of the Zebra puzzle are:
    - The Englishman lives in the red house.
    - The Spaniard owns the dog.
    - Coffee is drunk in the green house. …
  - Who owns the zebra?
- Problem solving tasks given in text or images can be converted to formal inputs using a neural module that can then use ASP (specific to a domain, but generic to all problems in that domain) to solve the problem.

Mathematical Reasoning

Disentangle language understanding from mathematical reasoning; Mishra et al. EMNLP 2022

- Translate Math problems to code in a programming language using a neural approach
- Symbolically execute the code

Neuro-symbolic Reinforcement Learning

RL, Deep RL and need for symbolic reasoning

- Reinforcement learning (RL) is about learning what action to take in a state in an environment so as to maximize the expected cumulative reward.
- The dynamics of the environment is modeled as a Markov decision process (MDP), with a set of states S, a set of actions A, a probabilistic transition function P that defines the probability of transitioning from one state to another due to an action, and a reward function associated with each transition.
- The goal of the reinforcement learning algorithm is then to learn a policy of what action to take in which state so as to maximize the expected cumulative reward.
- Deep RL is often used when the dimension of a state becomes large, such as when the state is expressed as a screenshot of a game or are images of a camera. In case of a game, the game score is often used as the reward.
- Symbolic reasoning is useful
    - Often there are additional conditions: safety, verifiability
    - Need for explainability at some level: use of rule learning, ILP

Neuro-symbolic Reinforcement Learning

Use of symbolic reasoning - Safety via shielding; formal verification

- Safety via shielding: safety during learning or execution phases
  - Enforcing properties expressed in temporal logic
  - Alshiekh et al. 2017: A reactive system called a shield is synthesized (a-priori)
    - First way: the shield acts each time the learning agent is about to make a decision and provides a list of safe actions.
    - Second way: The shield monitors the actions from the learner and corrects them only if the chosen action causes a violation of the specification.
  - Anderson et al. (REVEL): allows learning over continuous state and action spaces, supports (partially) neural policy representations and contemporary policy gradient methods for learning, while also ensuring that every intermediate policy that the learner constructs during exploration is safe on worst-case inputs.
    - Unlike in prior work, the monitor and the shield are updated as learning progresses.

Neuro-symbolic Reinforcement Learning

Use of symbolic reasoning – Interpretability via rule learning, ILP

- In hierarchical RL: higher level may use rule learning, ILP for better interpretability, while lower level uses Deep RL
  - Mitchener et al. 2022 learned metapolicies using Inductive Answer Set Learning and learned meta policies such as the following in the Animal AI environment.
    - If a ramp is available then climb it.
    - If the agent is on a platform and there is lava near the goal, then observe the arena dynamics.
    - If there are more goals on one side of a platform you are on, then go to that side.
    - If there's no lava, collect multi-goals.
    - If there's no lava around the goal and the goal is not on a platform go get it.
- Explainable (logical) rules can also be learned in a neural framework

References

- Yang, Zhun, Adam Ishay, and Joohyung Lee. "Neurasp: Embracing neural networks into answer set programming." 29th International Joint Conference on Artificial Intelligence (IJCAI 2020). 2020.
- Tsamoura, E., Hospedales, T., & Michael, L. (2021, May). Neural-symbolic integration: A compositional perspective. In Proceedings of the AAAI Conference on Artificial Intelligence (Vol. 35, No. 6, pp. 5051-5060).
- Paulo Shakarian, Gerardo I. Simari, Chitta Baral, Bowen Xi, Lahari Pokala. Neuro Symbolic Reasoning and Learning Current Advances and Future Directions. Springer Nature. 2023.

References (cont.)

- Yi, Kexin, et al. "Neural-symbolic vqa: Disentangling reasoning from vision and language understanding." Advances in neural information processing systems 31 (2018).
- Gupta, Nitish, Kevin Lin, Dan Roth, Sameer Singh, and Matt Gardner. "Neural Module Networks for Reasoning over Text." In International Conference on Learning Representations.
- Arindam Mitra, Peter Clark, Oyvind Tafjord and Chitta Baral. Declarative Question Answering over Knowledge Bases containing Natural Language Text with Answer Set Programming. AAAI 2019.
- Arindam Mitra and Chitta Baral. Learning to automatically solve logic grid puzzles. EMNLP 2015
- Swaroop Mishra, Matthew Finlayson, Pan Lu, Leonard Tang, Sean Welleck, Chitta Baral, Tanmay Rajpurohit, Oyvind Tafjord, Ashish Sabharwal, Peter Clark and Ashwin K. Kalyan. LILA: A Unified Benchmark for Mathematical Reasoning. EMNLP 2022.

References (cont.)

- Alshiekh, Mohammed, et al. "Safe reinforcement learning via shielding." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 32. No. 1. 2018.

- Anderson, G., Verma, A., Dillig, I., & Chaudhuri, S. (2020). Neurosymbolic reinforcement learning with formally verified exploration. Advances in neural information processing systems, 33, 6172-6183.

- Mitchener, L., Tuckey, D., Crosby, M., & Russo, A. (2022). Detect, understand, act: A neuro-symbolic hierarchical reinforcement learning framework. Machine Learning, 111(4), 1523-1549.

Thank you!