

UCLA

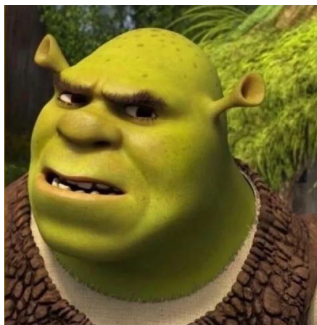
**Computer
Science**



Symbolic Reasoning About Large Language Models

Guy Van den Broeck

ICLR Workshop on Reasoning and Planning for Large Language Models - Apr 28 2025



Reasoning with Symbolic AI

- Deductive reasoning algorithms
- Logic and probabilistic
- Correct on *all* problems
- Limited scope
- Intractable

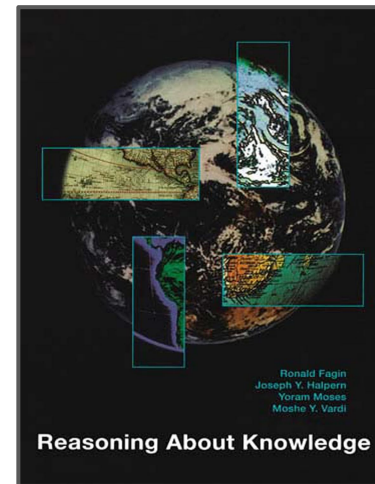


30 years



Reasoning with Transformers

- Inductive reasoning from data
- Build chains of thought
- Correct on *many* problems
- Unlimited scope
- Tractable



Use the digits [0-9] to make three numbers: x,y,z so that $x+y=z$

Thought for 270 seconds ^

Okay, so I need to use each digit from 0 to 9 exactly once to create three numbers x , y , and z such that $x + y = z$. Hmm, let me think about how to approach this.

...

Verification:

- The sum $752 + 346 = 1098$ is correct.
- All digits from 0 to 9 are used exactly once.

Thus, the final answer is:

$752 + 346 = 1098$

<https://www.datacamp.com/blog/deepseek-r1-vs-v3>

Questions for this talk:

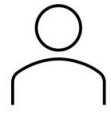


1. Do deductive reasoning algorithms still have a purpose in the age of transformers?
2. Where did reasoning algorithms go wrong?
What should they look like today?

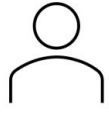
Questions for this talk:



1. **Do deductive reasoning algorithms still have a purpose in the age of transformers?**
2. Where did reasoning algorithms go wrong?
What should they look like today?



Generate a sentence using "frisbee", "caught" and "dog", following the given order.

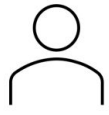


Generate a sentence using "frisbee", "caught" and "dog", following the given order.



After a perfect throw, the frisbee glided through the air, and the dog, with incredible agility, caught it mid-flight.

ChatGPT

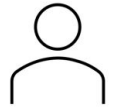


Generate a sentence using "frisbee", "caught" and "dog", following the given order.



After a perfect throw, the frisbee glided through the air, and the dog, with incredible agility, caught it mid-flight.

ChatGPT



That's not correct. Generate a sentence using "frisbee", "caught" and "dog". The keywords should appear in the order as specified.



Here's the correct sentence: The dog caught the frisbee in mid-air, showing off its amazing catching skills.

ChatGPT



A frisbee is caught by a dog.

A pair of frisbee players are caught in a dog fight.

Ctrl-G

The Ctrl-G Architecture

Lexical Constraint α : sentence contains keyword "winter"

Constrained Generation: $\Pr(x_{t+1} | \alpha, x_{1:t} = \text{"the weather is"})$

The Ctrl-G Architecture

Lexical Constraint α : sentence contains keyword "winter"

Constrained Generation: $\Pr(x_{t+1} | \alpha, x_{1:t} = \text{"the weather is"})$

Pre-trained
Language Model



x_{t+1}	$\Pr_{LM}(x_{t+1} x_{1:t})$
cold	0.05
warm	0.10

The Ctrl-G Architecture

Lexical Constraint α : sentence contains keyword "winter"

Constrained Generation: $\Pr(x_{t+1} | \alpha, x_{1:t} = \text{"the weather is"})$

Pre-trained
Language Model

x_{t+1}	$\Pr_{LM}(x_{t+1} x_{1:t})$
cold	0.05
warm	0.10



Using Bayes rule,

$$p_{LM}(\text{next-token} | \alpha, \text{prefix})$$

\propto

$$p_{LM}(\text{next-token} | \text{prefix})$$

$$\cdot p_{LM}(\alpha | \text{next-token}, \text{prefix})$$

The Ctrl-G Architecture

Lexical Constraint α : sentence contains keyword "winter"

Constrained Generation: $\Pr(x_{t+1} | \alpha, x_{1:t} = \text{"the weather is"})$

✗ intractable

Pre-trained
Language Model

x_{t+1}	$\Pr_{LM}(x_{t+1} x_{1:t})$
cold	0.05
warm	0.10



Using Bayes rule,

$$p_{LM}(\text{next-token} | \alpha, \text{prefix})$$

\propto

$$p_{LM}(\text{next-token} | \text{prefix})$$

$$\cdot p_{LM}(\alpha | \text{next-token}, \text{prefix})$$

Intractable



The Ctrl-G Architecture

Lexical Constraint α : sentence contains keyword "winter"

Constrained Generation: $\Pr(x_{t+1} | \alpha, x_{1:t} = \text{"the weather is"})$

✗ intractable

Pre-trained Language Model

x_{t+1}	$\Pr_{LM}(x_{t+1} x_{1:t})$
cold	0.05
warm	0.10

✓ efficient

Tractable Probabilistic Model

x_{t+1}	$\Pr_{TPM}(\alpha x_{t+1}, x_{1:t})$
cold	0.50
warm	0.01



Using Bayes rule,

$$p_{LM}(\text{next-token} | \alpha, \text{prefix})$$

\propto

$$p_{LM}(\text{next-token} | \text{prefix})$$

$$\cdot p_{LM}(\alpha | \text{next-token}, \text{prefix})$$

Intractable



The Ctrl-G Architecture

Lexical Constraint α : sentence contains keyword "winter"

Constrained Generation: $\Pr(x_{t+1} | \alpha, x_{1:t} = \text{"the weather is"})$

✗ intractable

✓ efficient

Pre-trained Language Model

x_{t+1}	$\Pr_{LM}(x_{t+1} x_{1:t})$
cold	0.05
warm	0.10

Tractable Probabilistic Model

x_{t+1}	$\Pr_{TPM}(\alpha x_{t+1}, x_{1:t})$
cold	0.50
warm	0.01

x_{t+1}	$p(x_{t+1} \alpha, x_{1:t})$
cold	0.025
warm	0.001



Abusing Bayes rule,

$$p_{CTRL-G}(\text{next-token} | \alpha, \text{prefix})$$

\propto

$$p_{LM}(\text{next-token} | \text{prefix})$$

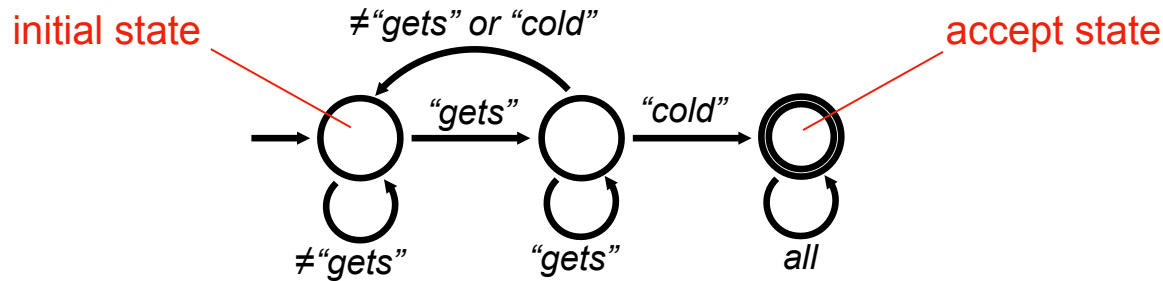
$$\cdot p_{TPM}(\alpha | \text{next-token}, \text{prefix})$$



Representing Logical Constraints

as a *deterministic finite automaton (DFA)*

Example. Check if a string contains “gets cold”.



Can represent:

Phrases/words must/must not appear

Exactly k times.

Anything over fixed sequence lengths (BDD)

Must end a certain way

From a restricted vocabulary.

Any regex

...

Interactive Text Editing

User: given the following context, generate infilling text for [BLANK] using key phrases "alien mothership", "far from over"; generated text must contain 25 - 30 words.

"First they've defeated a small squad [BLANK] are few humans left, and despite their magical power, their numbers are getting fewer."

Interactive Text Editing

User: given the following context, generate infilling text for [BLANK] using key phrases "alien mothership", "far from over"; generated text must contain 25 - 30 words.

"First they've defeated a small squad [BLANK] are few humans left, and despite their magical power, their numbers are getting fewer."

5 lines of code!

```
from CtrlG import *  
  
prefix = "First they defeated a ..."  
suffix = "are few humans left ..."  
  
dfa_list = [  
    DFA_all_of("alien mothership",  
              "far from over"),  
    DFA_word_count(25, 30),  
]  
dfa = DFA_logical_and(dfa_list)  
  
lp = CtrlGLogitsProcessor(  
    dfa, hmm, prefix, suffix)  
llm.generate(logits_processor=lp)
```


Interactive Text Editing

User: given the following context, generate infilling text for [BLANK] using key phrases "alien mothership", "far from over"; generated text must contain 25 - 30 words.

"First they've defeated a small squad [BLANK] are few humans left, and despite their magical power, their numbers are getting fewer."

```
from CtrlG import *  
  
prefix = "First they defeated a ..."  
suffix = "are few humans left ..."  
  
dfa_list = [  
    DFA_all_of("alien mothership",  
              "far from over"),  
    DFA_word_count(25, 30),  
]  
dfa = DFA_logical_and(dfa_list)  
  
lp = CtrlGLogitsProcessor(  
    dfa, hmm, prefix, suffix)  
llm.generate(logits_processor=lp)
```

5 lines of code!

"First they've defeated a small squad of aliens, then a larger fleet of their ships. Eventually they've even managed to take down the alien mothership. But their problems are far from over. There are few humans left, and despite their magical power, their numbers are getting fewer."

Interactive Text Editing with key phrase (K) or length (L) constraints



	<i>None</i>	<i>K</i>	<i>L</i>	<i>K&L</i>
<i>Quality</i>				
TULU2	2.68	2.64	2.78	2.74
GPT3.5	2.27	2.22	2.27	2.31
GPT4	3.79	3.33	3.53	3.10
Ctrl-G	3.77	3.56	3.73	3.59

→ *How many stars by humans?*

Interactive Text Editing with key phrase (K) or length (L) constraints



	None	K	L	K&L
<i>Quality</i>				
TULU2	2.68	2.64	2.78	2.74
GPT3.5	2.27	2.22	2.27	2.31
GPT4	3.79	3.33	3.53	3.10
Ctrl-G	3.77	3.56	3.73	3.59
<i>Success</i>				
TULU2	-	12%	20%	3%
GPT3.5	-	22%	54%	10%
GPT4	-	60%	20%	27%
Ctrl-G	-	100%	100%	100%

→ *How many stars by humans?*

→ *Follows instructions?*

Interactive Text Editing with key phrase (K) or length (L) constraints



	None	K	L	K&L
<i>Quality</i>				
TULU2	2.68	2.64	2.78	2.74
GPT3.5	2.27	2.22	2.27	2.31
GPT4	3.79	3.33	3.53	3.10
Ctrl-G	3.77	3.56	3.73	3.59
<i>Success</i>				
TULU2	-	12%	20%	3%
GPT3.5	-	22%	54%	10%
GPT4	-	60%	20%	27%
Ctrl-G	-	100%	100%	100%
<i>Overall</i>				
TULU2	-	7%	10%	1%
GPT3.5	-	0%	5%	2%
GPT4	-	41%	17%	14%
Ctrl-G	-	76%	78%	82%

→ *How many stars by humans?*

→ *Follows instructions?*

→ ★★☆☆☆ & Up + *Follows instructions?*

→ **Ctrl-G based on Llama2-7B wipes the floor with GPT4, which is a >100x bigger LLM**

Grade School Math Benchmark

Question: Kylar went to the store to buy glasses for his new apartment. One glass costs \$5, but every second glass costs only 60% of the price. Kylar wants to buy 16 glasses. How much does he need to pay for them?

Vanilla LLM Answer: The price of the 2nd glass is $(16 / 2) * 60\% = 8$ dollars. So one pair of glasses costs $16 + 8 = 24$ dollars. **So the answer is 24.**

Grade School Math Benchmark

Question: Kylar went to the store to buy glasses for his new apartment. One glass costs \$5, but every second glass costs only 60% of the price. Kylar wants to buy 16 glasses. How much does he need to pay for them?

Vanilla LLM Answer: The price of the 2nd glass is $(16 / 2) * 60\% = 8$ dollars. So one pair of glasses costs $16 + 8 = 24$ dollars. **So the answer is 24.**

Ctrl-G Answer: The second glass costs $5 * .6 = \$3$. So each set of two glasses actually costs $5 + 3 = \$8$. He wants $16 / 2 = 8$ sets of two. That means he needs to pay $8 * 8 = \$64$. **So the answer is 64.**

Which constraint improves accuracy?

Grade School Math Benchmark

Question: *Kylar went to the store to buy glasses for his new apartment. One glass costs \$5, but every second glass costs only 60% of the price. Kylar wants to buy 16 glasses. How much does he need to pay for them?*

Vanilla LLM Answer: The price of the 2nd glass is $(16 / 2) * 60\% = 8$ dollars. So one pair of glasses costs $16 + 8 = 24$ dollars. So the answer is 24.

Ctrl-G Answer: The second glass costs $5 * .6 = \$3$. So each set of two glasses actually costs $5 + 3 = \$8$. He wants $16 / 2 = 8$ sets of two. That means he needs to pay $8 * 8 = \$64$. So the answer is 64.

Use all the numbers in the problem statement!

Advantages of Ctrl-G:

1. Constraint α is guaranteed to be satisfied:
for any next-token x_{t+1} that would make α unsatisfiable, $p(x_{t+1} | x_{1:t}, \alpha) = 0$.
2. Training the tractable deep generative model does not depend on α , which is only imposed at inference (generation) time.
3. Generalizes well to unseen reasoning tasks, because all tasks are unseen :-)
(baselines train on a distribution over reasoning tasks – slow and brittle!)

You can control an intractable generative model using a generative model that is *tractable for reasoning*.

Questions for this talk:



1. Do deductive reasoning algorithms still have a purpose in the age of transformers?
2. **Where did reasoning algorithms go wrong?
What should they look like today?**

Probabilistic Reasoning Task

Marginal inference:

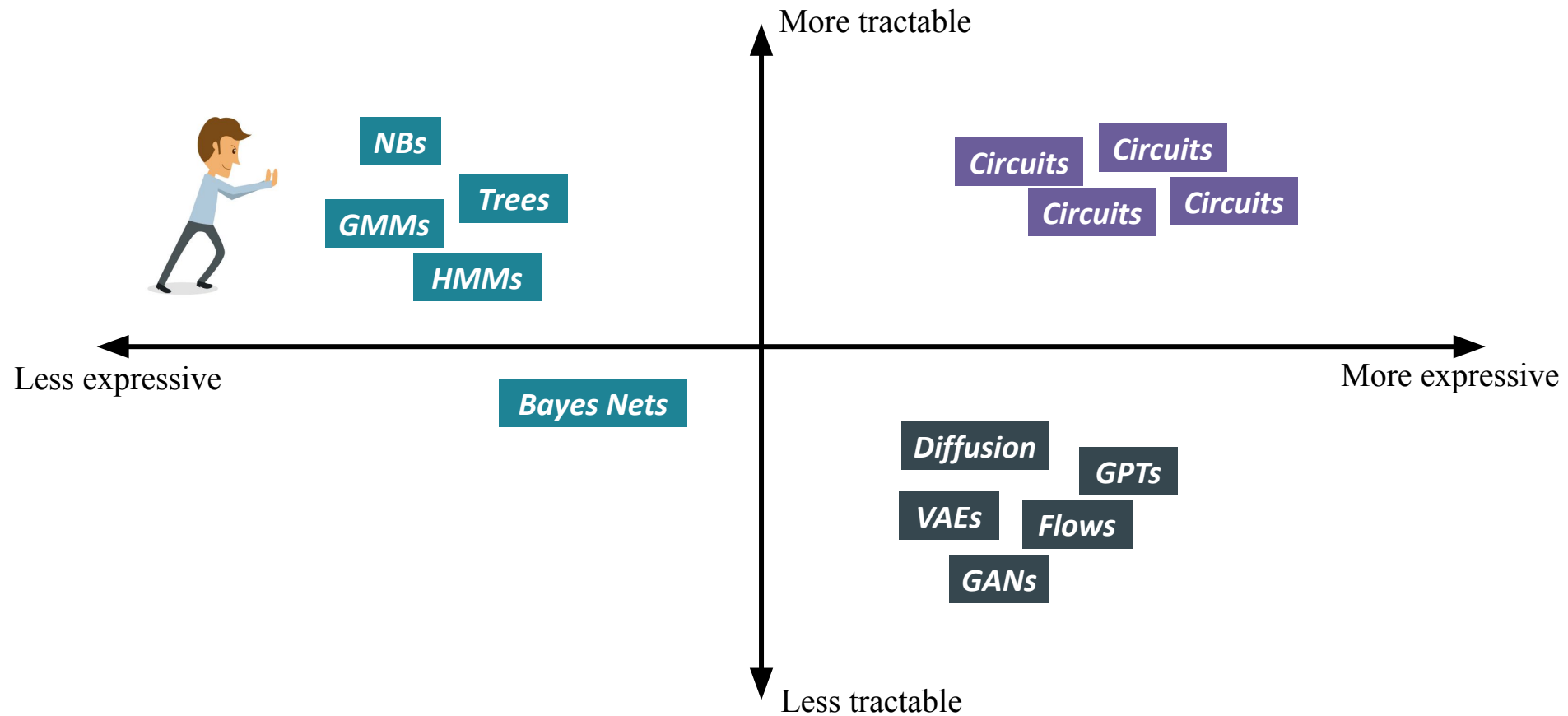
X_1	X_2	Pr
0	0	.1
0	1	.2
1	0	.3
1	1	.4

$$\begin{aligned}\Pr[X_1 = 1] &= \Pr[X_1 = 1, X_2 = 0] + \Pr[X_1 = 1, X_2 = 1] \\ &= 0.3 + 0.4 \\ &= 0.7\end{aligned}$$

Application: Ctrl-G



$$\Pr(\text{next-token} | \text{prefix}, \alpha) \propto \sum_{\text{text}} \Pr(\text{next-token}, \text{text}, \text{prefix}, \alpha)$$



Generative Models

polynomials model **joint distributions**

$$p(x_1, x_2, x_3) = .1x_1 + .05x_2 + .1x_1x_2 + .01x_3 - .07x_2x_3 + .02x_1x_3 - .14x_1x_2x_3 + .05$$

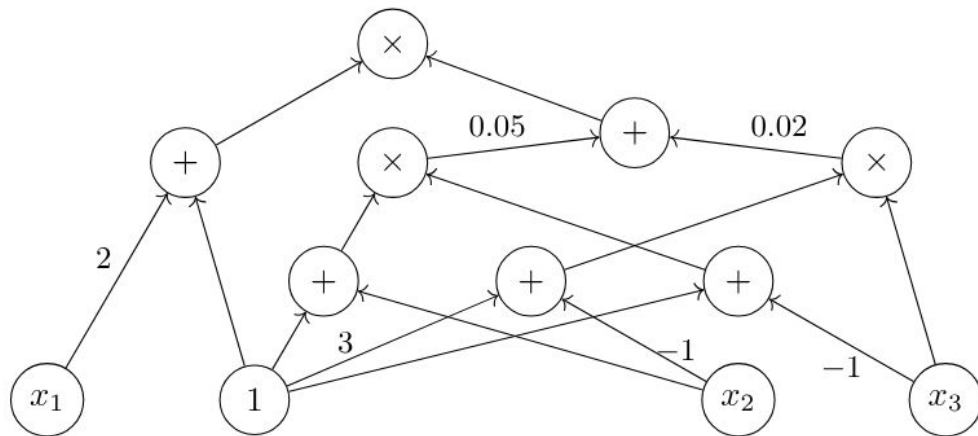
X_1	X_2	X_3	p
0	0	0	0.05
1	0	0	0.15
0	1	0	0.1
1	1	0	0.3
0	0	1	0.06
1	0	1	0.18
0	1	1	0.04
1	1	1	0.12

Deep Generative Models

circuit polynomials model **joint distributions** compactly

$$p(x_1, x_2, x_3) = .1x_1 + .05x_2 + .1x_1x_2 + .01x_3 - .07x_2x_3 + .02x_1x_3 - .14x_1x_2x_3 + .05$$

X_1	X_2	X_3	p
0	0	0	0.05
1	0	0	0.15
0	1	0	0.1
1	1	0	0.3
0	0	1	0.06
1	0	1	0.18
0	1	1	0.04
1	1	1	0.12

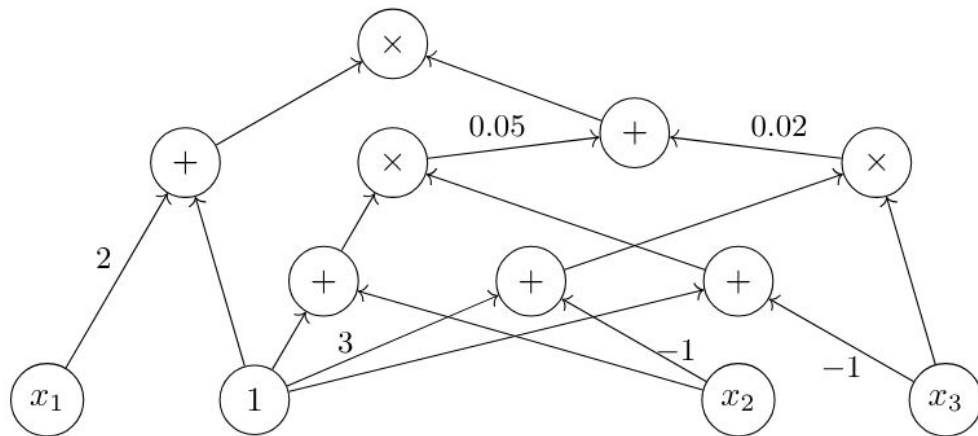


Deep Generative Models

circuit polynomials model **joint distributions** compactly

$$p(x_1, x_2, x_3) = .1x_1 + .05x_2 + .1x_1x_2 + .01x_3 - .07x_2x_3 + .02x_1x_3 - .14x_1x_2x_3 + .05$$

X_1	X_2	X_3	p
0	0	0	0.05
1	0	0	0.15
0	1	0	0.1
1	1	0	0.3
0	0	1	0.06
1	0	1	0.18
0	1	1	0.04
1	1	1	0.12

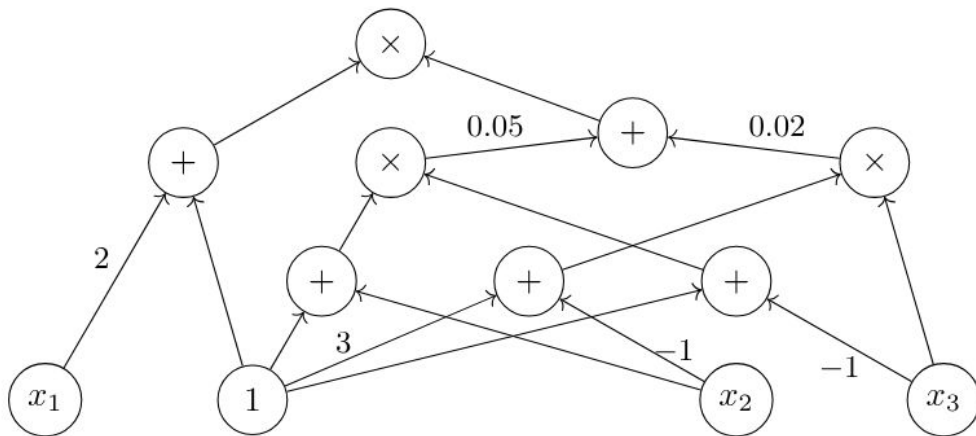


Tractable Deep Generative Models

Multilinear circuit polynomials model **joint distributions** compactly *and* allow **efficient** probabilistic reasoning

$$p(x_1, x_2, x_3) = .1x_1 + .05x_2 + .1x_1x_2 + .01x_3 - .07x_2x_3 + .02x_1x_3 - .14x_1x_2x_3 + .05$$

X_1	X_2	X_3	p
0	0	0	0.05
1	0	0	0.15
0	1	0	0.1
1	1	0	0.3
0	0	1	0.06
1	0	1	0.18
0	1	1	0.04
1	1	1	0.12



Computing Marginals

Compute $p(x = \blacksquare) = \iint p(x = \blacksquare, y, z) dydz$

- Sum node \oplus_a**

$$\iint p_a(x = \blacksquare, y, z) dydz$$

$$= \iint 0.5 \cdot p_b(x = \blacksquare, y, z) + 0.5 \cdot p_c(x = \blacksquare, y, z) dydz$$

$$= 0.5 \iint p_b(x = \blacksquare, y, z) dydz + 0.5 \iint p_c(x = \blacksquare, y, z) dydz$$

$$\iint \otimes_b dydz$$

$$\iint \otimes_c dydz$$

- Product node \otimes_b**

$$\iint p_b(x = \blacksquare, y, z) dydz$$

$$= \iint p_d(z) \cdot p_e(x = \blacksquare, y) dydz$$

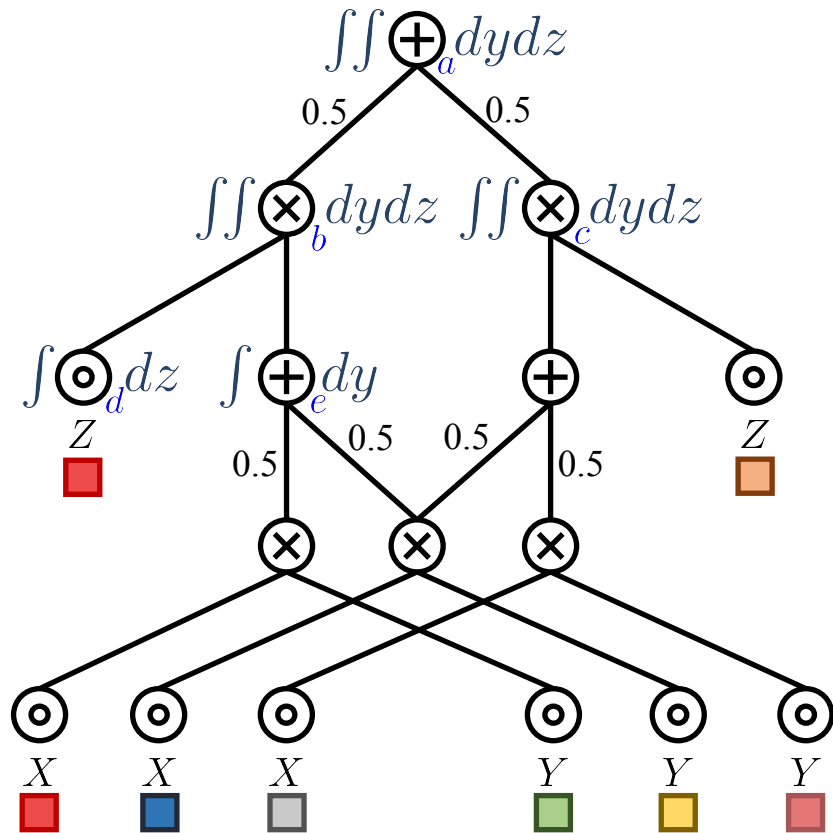
$$= \int p_d(z) dz \cdot \int p_e(x = \blacksquare, y) dy$$

$$\int \otimes_d dz$$

$$\int \otimes_e dy$$

- Input node \odot_d**

$$\int p_d(z) = 1$$

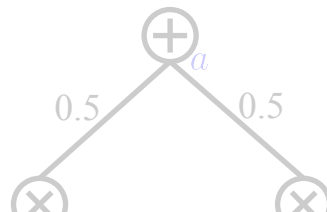


Computing Marginals

Compute $p(x = \square) = \int \int p(x = \square, y, z) dy dz$

- Sum node \oplus_a

$$\int \int p_a(x = \square, y, z) dy dz$$



Theorem. Given

1. a DFA constraint α with m edges and
2. a PC $p(\cdot)$ with h hidden states (representing a Hidden Markov Model), computing $p(\alpha \mid x_{1:t})$ over a sequence of n future tokens takes $O(nmh^2)$ time.



$$= \int \int p_d(z) \cdot p_e(x = \square, y) dy dz$$

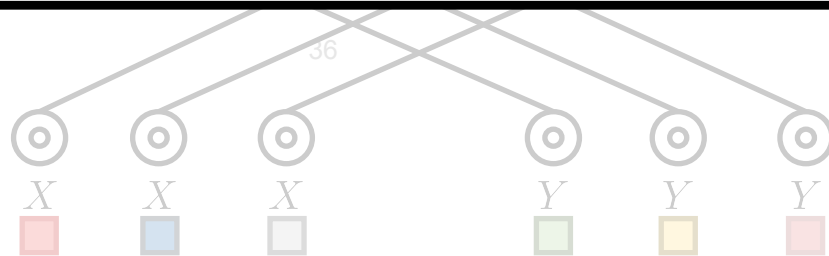
$$= \underbrace{\int p_d(z) dz}_{\int \otimes_d dz} \cdot \underbrace{\int p_e(x = \square, y) dy}_{\int \otimes_e dy}$$

$$\int \otimes_d dz$$

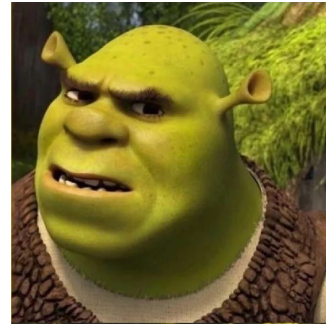
$$\int \otimes_e dy$$

- Input node \odot_d

$$\int p_d(z) = 1$$



You Tricked Us



You promised us reasoning algorithms...

... and all we got was another lousy feedforward neural network!

***Theorem.** If there exists a polynomial time (real RAM) algorithm that computes (virtual evidence) **marginals** for a family of distributions, then there exist **poly-size circuits** for their **multilinear polynomials**.*



An Open-Source Package: PyJuice

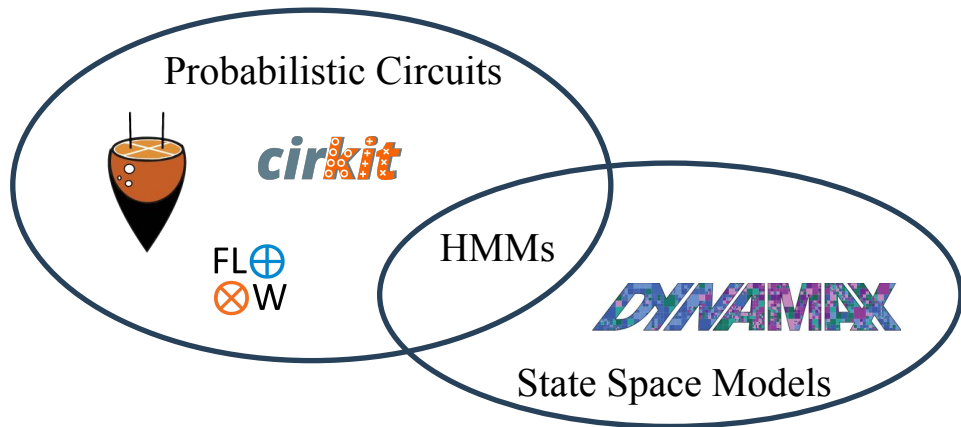


Runtime (in seconds) for training on **60K** samples

	PD (Poon & Domingos, 2011)				
# nodes	172K	344K	688K	1.38M	2.06M
# edges	15.6M	56.3M	213M	829M	2.03B
SPFlow	>25000	>25000	>25000	>25000	>25000
EiNet	34.2±0.0	88.7±0.2	456.1±2.3	1534.7±0.5	OOM
Juice.jl	12.6±0.5	37.0±1.7	141.7±6.9	OOM	OOM
PyJuice	2.0±0.0	5.3±0.0	15.4±0.0	57.1±0.2	203.7±0.1
	RAT-SPN (Peharz et al., 2020b)				
# nodes	58K	116K	232K	465K	930K
# edges	616K	2.2M	8.6M	33.4M	132M
SPFlow	6372.1±4.2	>25000	>25000	>25000	>25000
EiNets	38.5±0.0	83.5±0.0	193.5±0.1	500.6±0.2	2445.1±2.6
Juice.jl	6.0±0.3	9.4±0.3	25.5±2.4	84.0±4.0	375.1±3.4
PyJuice	0.6±0.0	0.9±0.1	1.6±0.0	5.8±0.1	13.8±0.0
	HCLT (Liu & Van den Broeck, 2021)				
# nodes	89K	178K	355K	710K	1.42M
# edges	2.56M	10.1M	39.9M	159M	633M
SPFlow	22955.6±18.4	>25000	>25000	>25000	>25000
EiNet	52.5±0.3	77.4±0.4	233.5±2.8	1170.7±8.9	5654.3±17.4
Juice.jl	4.7±0.2	6.4±0.5	12.4±1.3	41.1±0.1	143.2±5.1
PyJuice	0.8±0.0	1.3±0.0	2.6±0.0	8.8±0.0	24.9±0.1
	HMM (Rabiner & Juang, 1986)				
# nodes	33K	66K	130K	259K	388K
# edges	8.16M	32.6M	130M	520M	1.17B
Dynamax	111.3±0.4	441.2±3.9	934.7±6.3	2130.5±19.5	4039.8±38.3
Juice.jl	4.6±0.1	18.8±0.1	91.6±0.1	OOM	OOM
PyJuice	0.6±0.0	1.0±0.0	2.9±0.1	10.1±0.2	39.9±0.1

- Orders of magnitude **faster!**
- Extremely **scalable!**

Custom data structure +
CUDA kernels



FL⊕⊗W by Cambridge, TU Darmstadt, Max-Planck-Institute et al.

cirkit by Edinburgh, EPFL et al.

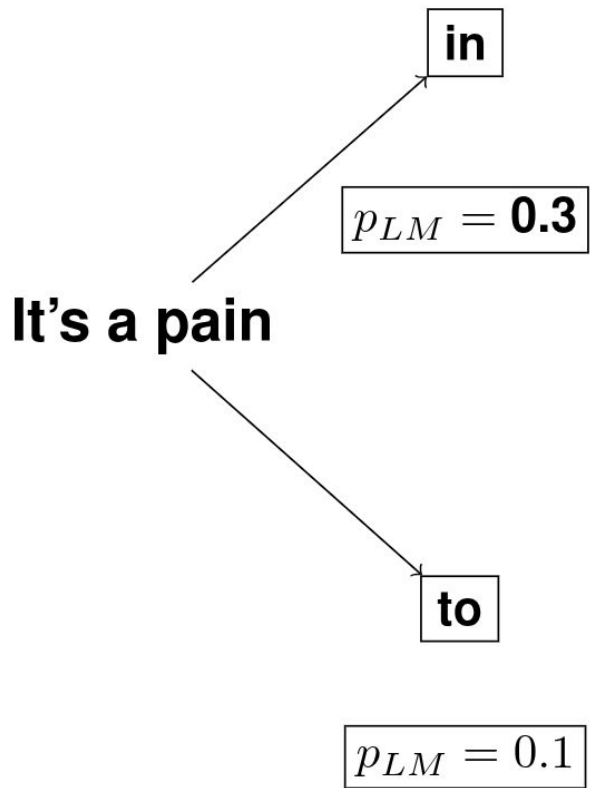
DYNAMIX by Google Deepmind et al.

<https://github.com/Tractables/pyjuice>

Questions for this talk:



1. **Do deductive reasoning algorithms still have a purpose in the age of transformers?**
2. Where did reasoning algorithms go wrong?
What should they look like today?



Attribute Probability



0 (toxic)

1 (nontoxic)

- No longer a logical constraint (no DFA)
- A “soft” **attribute** with some probability
- a.k.a. an exponentiated *reward function*

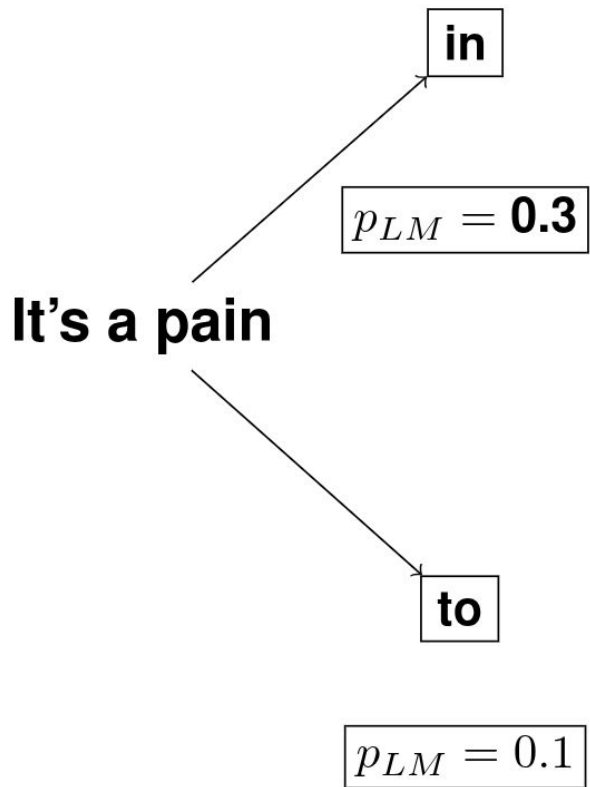


Attribute Probability



0 (toxic)

1 (nontoxic)



future text	$p_{LM}(x_{>t} x_{\leq t})$
the ass	0.3
the butt	0.15
the neck	0.05
...	...
...	...

Intractable to know future
expected attribute probability (EAP)

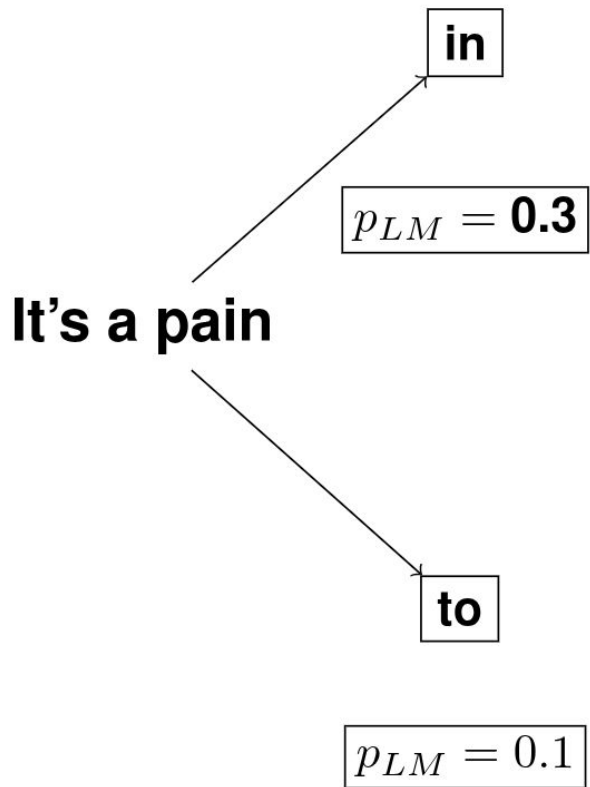
future text	$p_{LM}(x_{>t} x_{\leq t})$
deal with	0.2
handle	0.1
...	...
...	...



Attribute Probability

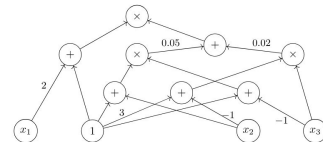


0 (toxic) 1 (nontoxic)



future text	$p_{TPM}(x_{>t} x_{\leq t})$
the ass	0.3
the butt	0.15
the neck	0.05
...	...
...	...

Tractable Probabilistic Model



+ Log-Linear Attribute Classifier



future text	$p_{TPM}(x_{>t} x_{\leq t})$
deal with	0.2
handle	0.1
...	...
...	...

Attribute Probability



0 (toxic) 1 (nontoxic)

It's a pain

in

$$p_{LM} = 0.3$$

to

$$p_{LM} = 0.1$$

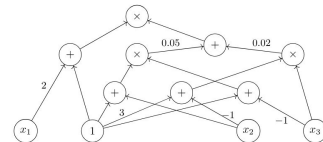
future text	$p_{TPM}(x_{>t} x_{\leq t})$
the ass	0.3
the butt	0.15
the neck	0.05
...	...
...	...

$$EAP = 0.1$$

future text	$p_{TPM}(x_{>t} x_{\leq t})$
deal with	0.2
handle	0.1
...	...
...	...

$$EAP = 0.8$$

Tractable
Probabilistic Model



+ Log-Linear
Attribute Classifier



=

Efficient Expected
Attribute Probability!



Attribute Probability



0 (toxic) 1 (nontoxic)

It's a pain

in

$$p_{LM} = 0.3 \times$$

future text	$p_{TPM}(x_{>t} x_{\leq t})$
the ass	0.3
the butt	0.15
the neck	0.05
...	...
...	...

$$EAP = 0.1$$

$$= p_{TRACE} \propto 0.03$$



to

$$p_{LM} = 0.1 \times$$

future text	$p_{TPM}(x_{>t} x_{\leq t})$
deal with	0.2
handle	0.1
...	...
...	...

$$EAP = 0.8$$

$$= p_{TRACE} \propto 0.08$$

TRACE: State-of-the-art LLM Detoxification

Model	Toxicity (↓)		Fluency (↓)	Diversity (↑)		Approach Type
	avg. max.	prob.		dist-2	dist-3	
GPT-2 Large Results						
GPT2	0.385	0.254	25.57	0.87	0.86	Baseline
DAPT ⁽¹⁾	0.428	0.360	31.21	0.84	0.84	Finetuning
GeDi ⁽²⁾	0.363	0.217	60.03	0.84	0.83	Decoding (Trained Guide)
FUDGE ⁽³⁾	0.302	0.371	12.97*	0.78	0.82	Decoding (Trained Guide)
DExperts ⁽⁴⁾	0.314	0.128	32.41	0.84	0.84	Decoding (Trained Guide)
PPLM ⁽⁵⁾	0.520	0.518	32.58	0.86	0.86	Decoding (Logit Control)
MuCoLa ⁽⁶⁾	0.308	0.088	29.92	0.82	0.83	Decoding (Sampling)
PPO ⁽⁷⁾	0.218	0.044	14.27*	0.80	0.84	RL
Quark ⁽⁸⁾	0.196	0.035	12.47*	0.80	0.84	RL
DPO ⁽⁹⁾	0.180	0.026	21.59*	0.76	0.78	RL
TRACE	0.163	0.016	29.83	0.85	0.85	Decoding (HMM Reasoning)
Gemma-2B Results						
Gemma-2B	0.359	0.23	15.75	0.86	0.85	Baseline
DPO ⁽⁹⁾	0.222	0.06	14.39*	0.74	0.77	RL
TRACE	0.189	0.02	17.68	0.86	0.85	Decoding (HMM Reasoning)

Personalized Language Model: Twilight Sparkle



Baseline



Prompt

You are an advanced role-playing assistant trained to embody characters with accuracy and authenticity. In this instance, you will assume the persona of Twilight Sparkle.

10 QA Examples: 1...2...3...4...5...6...7...8...9...10...

Question: Twilight Sparkle, how is the weather?

Generation

The weather is pretty hot and humid here, thanks to our climate.

TRACE



Prompt

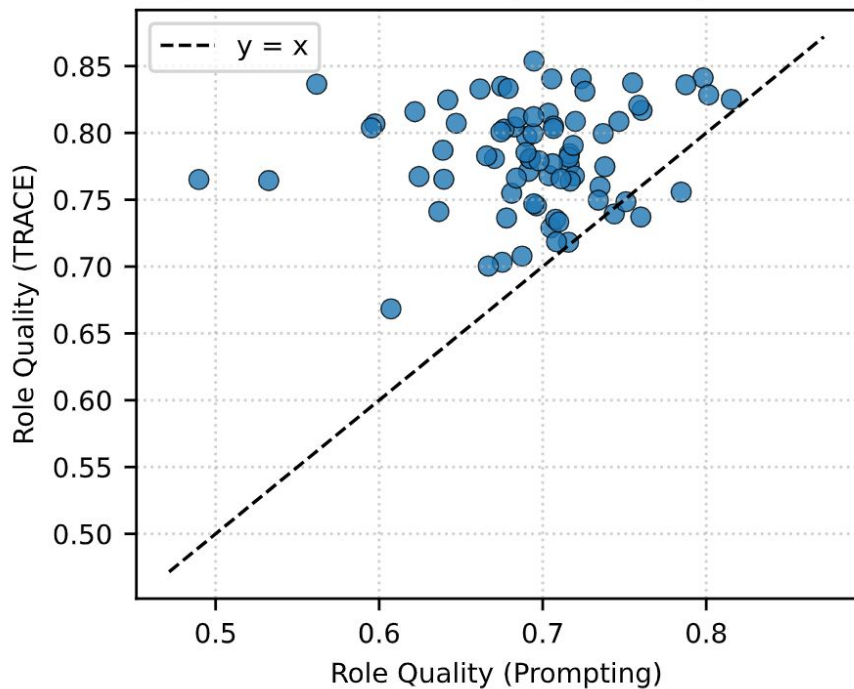
How is the weather?

Generation

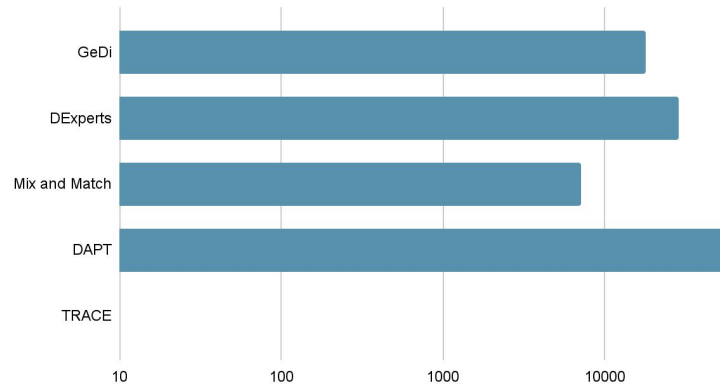
Gosh, it's sunny and very beautiful and all around me.

76 Personalized Language Models

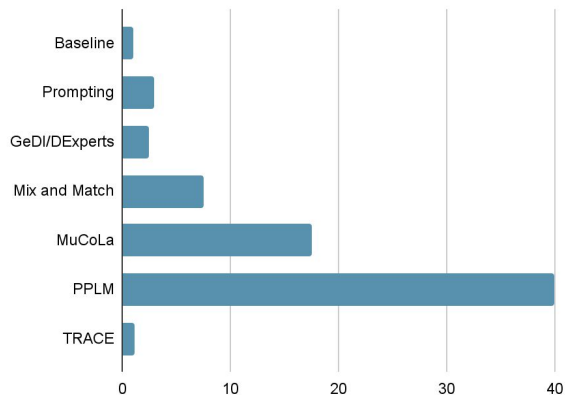
Comparison of Role Quality (Prompting vs. TRACE)



Training Time per Attribute (seconds)



Inference Time



Conclusions for this talk:

1. Do deductive reasoning algorithms still have a purpose in the age of transformers?



2. Where did reasoning algorithms go wrong?

What should they look like today?

Conclusions for this talk:

1. Do deductive reasoning algorithms still have a purpose in the age of transformers?

Yes, more cool applications of reasoning than fit on these slides!

2. Where did reasoning algorithms go wrong?

What should they look like today?



Conclusions for this talk:



1. Do deductive reasoning algorithms still have a purpose in the age of transformers?

Yes, more cool applications of reasoning than fit on these slides!

2. Where did reasoning algorithms go wrong?

Learn at scale, be tractable

What should they look like today?

Conclusions for this talk:



1. Do deductive reasoning algorithms still have a purpose in the age of transformers?

Yes, more cool applications of reasoning than fit on these slides!

2. Where did reasoning algorithms go wrong?

Learn at scale, be tractable

What should they look like today?

Circuits! Circuits! Circuits!

Thanks

This was the work of many wonderful students/postdocs/collaborators!



References: <http://starai.cs.ucla.edu>