How logic can analyze neural networks

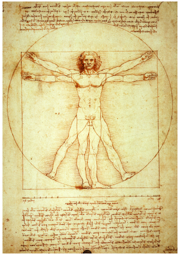
using (another type of) learning

Thomas Schiex

October 2018

TSDL 2018, Toulouse, France

Superhuman performances of AI

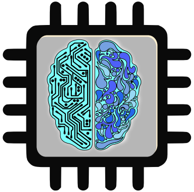


Human beings

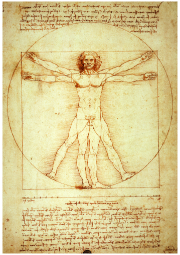
- Easily rely on quick "intuitions" (ill-defined problems)
- Extreme rigor is painful and slow (logic/arithmetic)

AIs (computers)

- Accessible to some "intuition" (problems defined by data)
- Fast and extreme rigor is the default (1 billion op./sec)



Superhuman performances of AI

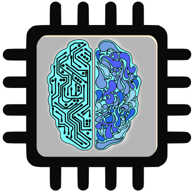


Human beings

- Easily rely on quick "intuitions" (ill-defined problems)
- Extreme rigor is painful and slow (logic/arithmetic)

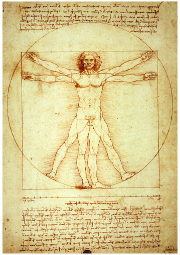
AIs (computers)

- Accessible to some "intuition" (problems defined by data)
- Fast and extreme rigor is the default (1 billion op./sec)



It was expected that machines would show superhuman "logical reasoning" performances

Superhuman performances of AI

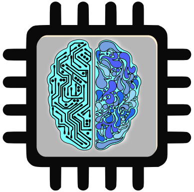


Human beings

- Easily rely on quick "intuitions" (ill-defined problems)
- Extreme rigor is painful and slow (logic/arithmetic)

AIs (computers)

- Accessible to some "intuition" (problems defined by data)
- Fast and extreme rigor is the default (1 billion op./sec)



It was expected that machines would show superhuman "logical reasoning" performances

1955: Newell & Simon "Logic Theorist" proved 38 of the 52 theorems in the *Principia Mathematica* (Russel and Whitehead), and even corrected a proof in it.

Then came complexity theory

NP-hard problems (Cook-Levin, 1970s)

- Some problems seems intrisically hard (for computers at least)
- *Worst case asymptotic* exponential time ($P \neq NP$)

Then came complexity theory

NP-hard problems (Cook-Levin, 1970s)

- Some problems seems intrisically hard (for computers at least)
- *Worst case asymptotic* exponential time (P $\neq$ NP)

$n^2 \times n^2$ Sudoku

- NP-complete, 9×9: 10^{80} cases
- 10^{51} ages of the universe to examine them all
- Fast brute force will fail

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

Then came complexity theory

NP-hard problems (Cook-Levin, 1970s)

- Some problems seems intrisically hard (for computers at least)
- *Worst case asymptotic* exponential time ($P \neq NP$)

$n^2 \times n^2$ Sudoku

- NP-complete, 9×9: 10^{80} cases
- 10^{51} ages of the universe to examine them all
- Fast brute force will fail

	E		1	6					D		7	F		G	B
B				7				A		1		9	D		4
A			2	C			1	B						E	
		7					G	F	2						3
	B			1										7	
	9	A			6			1	E				G	B	
		8	C				5		7					1	6
G					2	A	F			3		5			C
E		5	G				B				9				
	C			4			8	5	6						
		9	F	3	1	C		D		E					
	8	4		5		F	D	3		G	1			6	
		C			7	6					3		F		G
5			D	F			C			8			9		
	F	2					3			4					
		6								D			C		5

Then came complexity theory

NP-hard problems (Cook-Levin, 1970s)

- Some problems seems intrisically hard (for computers at least)
- *Worst case asymptotic* exponential time ($P \neq NP$)

$n^2 \times n^2$ Sudoku

- NP-complete, 9×9: 10^{80} cases
- 10^{51} ages of the universe to examine them all
- Fast brute force will fail

B		A	P			L				C			M				1		J	H	4	7		
			G	F			A		O		7	9	3	L	C			P	M				B	K
N	M			I					4	P			6	A	E	F	G				C	D	3	
5		K										4		D	L		O		6					
C	H	1			D	F	G	N		B	E												9	P
7	A									6	4					G	H			2	1	F		
G			J	N	L	E	8				B		C	7	1		P		F			H	O	
2				6	K					J	1	P		G	N			A		B	3	M		
	F	I		8	6	C	7		3	D	O	N		H				M		K	E	P		
		3			P						I	2	F		K		J				9	N		
L				K	C	9	N	E	6	H		A	B	M	3		8					2	7	J
	N		3		B		M						D			P			K					O
		5		B	I	O			K	9				3	A			J	2	6		L		
P					F			L			6						4		G		H		K	
1	8	G					2		5	4	L	J		K	I	6	C	B	D	E				9
		F	6				4		7		9	H	G						8			C		
		C	L	2		1				N		F	K	O	7		D	G	3	I		9	E	
		P	A	4		H			L	E		1	J	2					C	O				8
	9	D			8		C		E	L	M		P				F	I	B	1	K			7
		B	8	M			D	6					7	C									G	H
3	G												H	6		O	M	C	E			I	P	1
					1		L		P	3		B										K		F
	C	8	2				5	O	J	G	A			F	9					3			4	B
D	6				M	3			G	8	2	5	9		H		I			A	O			
		H	I	O	7		9				J			E				5			L	6		M

Then came complexity theory

NP-hard problems (Cook-Levin, 1970s)

- Some problems seems intrisically hard (for computers at least)
- *Worst case asymptotic* exponential time (P $\neq$ NP)

$n^2 \times n^2$ Sudoku

- NP-complete, 9×9: 10^{80} cases
- 10^{51} ages of the universe to examine them all
- Fast brute force will fail
- Can be solved in milliseconds

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

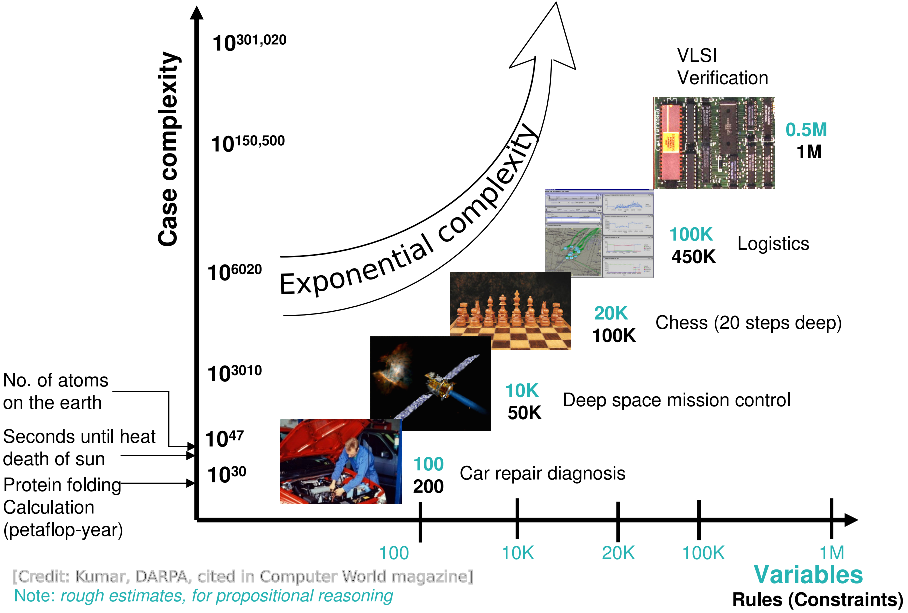
The modern world needs rigourous logic, more than ever

Technological progress

- Increasingly complex useful objects planes, computers, software, cars, AIs
- That must be highly reliable (lives at stake)
- We cannot fully get them under control anymore

Increasing system complexity

- Hardware: Pentium FDIV bug (1994, 3.1 million transistors)
- Software: the Therac-25 (radiation-therapy) kills 6 patients



[Credit: Kumar, DARPA, cited in Computer World magazine]

Note: *rough estimates, for propositional reasoning*

Propositional logic as an example (SAT)

SAT

1. A set of Boolean variables x_i
2. A set of clauses (*disjunction* of variables or negation of) $(\neg x_1 \vee x_7)$
3. Must satisfy all clauses (or prove impossible)

Propositional logic as an example (SAT)

SAT

1. A set of Boolean variables x_i
2. A set of clauses (*disjunction* of variables or negation of) $(\neg x_1 \vee x_7)$
3. Must satisfy all clauses (or prove impossible)

Sudoku

1. cell (i, j) contains k — x_{ijk} true
2. At least one number per cell i, j — $(x_{ij1} \vee \ldots \vee x_{ij9})$
3. At most one number per cell i, j — $(\forall k > k' \neg x_{ijk} \vee \neg x_{ijk'})$
4. Cell (i, j) and (i, j') must be different — $(\neg x_{ijk} \vee \neg x_{ij'k})$

SAT is the simplest canonical NP-complete problem

More sophisticated/practical function description

- propositions over theories — SAT Modulo Theory (SMT)[9]
- non Boolean variables — Constraint Satisfaction, Constraint Programming[26]
- numerical output — Weighted MaxSAT[20]/CSP,[5] Graphical models[14]
- Mathematical programming/OR — Mixed Integer Linear Programming ((M)ILP, QP,...)

What can we embrace with NP-complete problems?

NP-complete: can express all NP-complete problems

- the logical puzzles you like (Sudoku, Nonograms...)
- or not (configuration, scheduling, test pattern generation...)
- robot planning
- digital circuit verification (Bounded Model Checking)
- or software verification (FOL, grounding, abstraction)

What can we embrace with NP-complete problems?

NP-complete: can express all NP-complete problems

- the logical puzzles you like (Sudoku, Nonograms...)
- or not (configuration, scheduling, test pattern generation...)
- robot planning
- digital circuit verification (Bounded Model Checking)
- or software verification (FOL, grounding, abstraction)

NP-complete, so intractable

Standard argument for less realistic problem reformulation, heuristics or stochastic search

What can we embrace with NP-complete problems?

NP-complete: can express all NP-complete problems

- the logical puzzles you like (Sudoku, Nonograms...)
- or not (configuration, scheduling, test pattern generation...) SIEMENS THALES
- robot planning (Rosetta-Philæ probe plan, CP, LAAS/Toulouse) cnes
- digital circuit verification (Bounded Model Checking) intel IBM
- or software verification (FOL, grounding, abstraction) cadence

NP-complete, so intractable

Standard argument for less realistic problem reformulation, heuristics or stochastic search

Real SAT instances with millions of variables/clauses can be solved (with a proof)

IBM Bounded Model Checking SAT instance (SATLIB)

```
p cnf 51639 368352
-1 7 0
-1 6 0
-1 5 0
-1 -4 0
-1 3 0
-1 2 0
-1 -8 0
-9 15 0
-9 14 0
-9 13 0
-9 -12 0
-9 11 0
-9 10 0
-9 -16 0
```

$51,639$ variables, $368,352$ constraints

$\neg x_1 \vee x_7$

$\neg x_1 \vee x_6$

$\ldots$

10 Pages later

```
185 -9 0
185 -1 0
177 169 161 153 145 137 129
121 113 105 97 89 81 73 65 57
49 41 33 25 17 9 1 -185 0
186 -187 0
186 -188 0
```

...

$$(x_{177} \vee x_{169} \vee x_{161} \vee x_{153} \vee \cdots \vee x_{17} \vee x_9 \vee x_1 \vee \neg x_{185})$$

4,000 Pages later

```
10236 -10050 0
10236 -10051 0
10236 -10235 0
10008 10009 10010 10011 10012 10013 10014 10015 10016 10017 10018
10019 10020 10021 10022 10023 10024 10025 10026 10027 10028 10029
10030 10031 10032 10033 10034 10035 10036 10037 10086 10087 10088
10089 10090 10091 10092 10093 10094 10095 10096 10097 10098 10099
10100 10101 10102 10103 10104 10105 10106 10107 10108 -55 -54 53 -52
-51 50 10047 10048 10049 10050 10051 10235 -10236 0
10237 -10008 0
10237 -10009 0
10237 -10010 0
...
```

Finally 15,000 Pages later

```
-7 260 0
7 -260 0
1072 1070 0
-15 -14 -13 -12 -11 -10 0
-15 -14 -13 -12 -11 10 0
-15 -14 -13 -12 11 -10 0
-15 -14 -13 -12 11 10 0
-7 -6 -5 -4 -3 -2 0
-7 -6 -5 -4 -3 2 0
-7 -6 -5 -4 3 -2 0
-7 -6 -5 -4 3 2 0
185 0
```

Finally 15,000 Pages later

```
-7 260 0
7 -260 0
1072 1070 0
-15 -14 -13 -12 -11 -10 0
-15 -14 -13 -12 -11 10 0
-15 -14 -13 -12 11 -10 0
-15 -14 -13 -12 11 10 0
-7 -6 -5 -4 -3 -2 0
-7 -6 -5 -4 -3 2 0
-7 -6 -5 -4 3 -2 0
-7 -6 -5 -4 3 2 0
185 0
```

Search space

$2^{50,000} \approx 3.1\ 10^{15,051}$

Finally 15,000 Pages later

```
-7 260 0
7 -260 0
1072 1070 0
-15 -14 -13 -12 -11 -10 0
-15 -14 -13 -12 -11 10 0
-15 -14 -13 -12 11 -10 0
-15 -14 -13 -12 11 10 0
-7 -6 -5 -4 -3 -2 0
-7 -6 -5 -4 -3 2 0
-7 -6 -5 -4 3 -2 0
-7 -6 -5 -4 3 2 0
185 0
```

Search space

$2^{50,000} \approx 3.1\ 10^{15,051}$

Solved in one second

Finally 15,000 Pages later

```
-7 260 0
7 -260 0
1072 1070 0
-15 -14 -13 -12 -11 -10 0
-15 -14 -13 -12 -11 10 0
-15 -14 -13 -12 11 -10 0
-15 -14 -13 -12 11 10 0
-7 -6 -5 -4 -3 -2 0
-7 -6 -5 -4 -3 2 0
-7 -6 -5 -4 3 -2 0
-7 -6 -5 -4 3 2 0
185 0
```

Search space

$2^{50,000} \approx 3.1\ 10^{15,051}$

Solved in one second

How dœs it work?

NP-complete problem solvers: huge progress

SAT: Conflict Directed Clause Learning

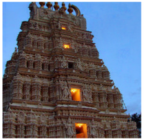
- Massive efficient reasoning[7,8,25] + making assumptions
- Lot of heuristics[1,21]
- Safe learning from failure[2,10,18,19,28,29] with Backward resolution
- Efficient cache-friendly data-structures

- International competitions ($> 50,000$ benchmarks with many real problems)
- Open source solvers (autocatalytic)
- Strong European/French/Toulouse presence in theory, algorithms, solvers, applications[1,4,12]

(Deep) neural nets and safety critical settings

It dœsn't seem too hard to fool a standard Convolutional Neural Net[a]

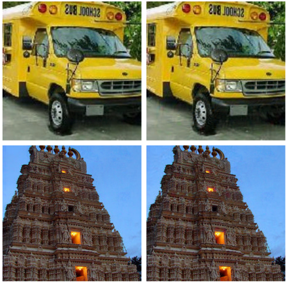
[a] Christian Szegedy et al. "Intriguing properties of neural networks". In: *arXiv preprint arXiv:1312.6199* (2013).



(Deep) neural nets and safety critical settings

It dœsn't seem too hard to fool a standard Convolutional Neural Net[a]

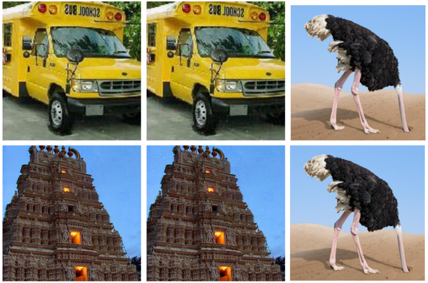
[a] Christian Szegedy et al. "Intriguing properties of neural networks". In: *arXiv preprint arXiv:1312.6199* (2013).



(Deep) neural nets and safety critical settings

It dœsn't seem too hard to fool a standard Convolutional Neural Net[a]

[a] Christian Szegedy et al. "Intriguing properties of neural networks". In: *arXiv preprint arXiv:1312.6199* (2013).

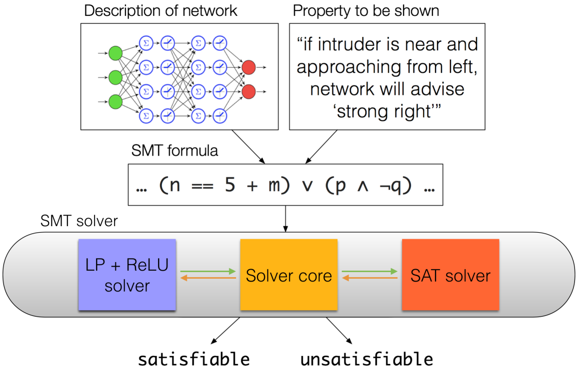


Checking Neural nets with logic

- First results in 2010 on “shallow” networks (robotics)[23,24]
- More recent results on deep non naive convolutional neural nets[3,22]
- Some in avionics with ReLU (ACAS Xu aircraft collision avoidance system[13,16])
- (M)ILP, SMT(LI), global optimization[27] and pure SAT[22]

Properties

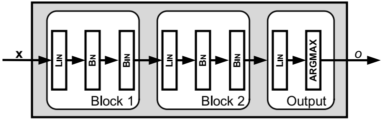
Reachability, Local/global robustness (adversarial manipulations), Invertibility, Equivalence



From deep Neural Nets to SAT

Binarized Deep NN: ± 1 activations/weights[6]

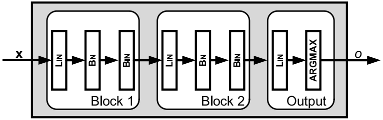
- Still powerful, used in embedded systems for their speed
- Lin: affine transformation with learnt binary weights (float bias).
- Bn: (Batch normalization) rescaling with learnt floats.
- Bin: binarization using the *Sign* function.



From deep Neural Nets to SAT

Binarized Deep NN: ± 1 activations/weights[6]

- Still powerful, used in embedded systems for their speed
- Lin: affine transformation with learnt binary weights (float bias).
- Bn: (Batch normalization) rescaling with learnt floats.
- Bin: binarization using the *Sign* function.



A learnt block can be described as a MILP/ILP/SMT(LI), SAT formula[a]

[a] Nina Narodytska et al. "Verifying properties of binarized deep neural networks". In: *arXiv preprint arXiv:1709.06662* (2017).

How can we check robustness to adversarial manipulation?

Check satisfiability of a SAT formula that combines

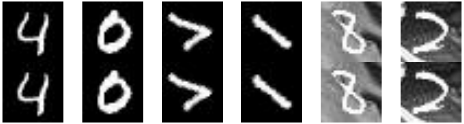
- a SAT description of the NN behavior
- a SAT description of a valid input image
- a SAT description of bounded manipulation of it
- a SAT formula that forces the output to a wrong class

Robustness checking

- if the formula has a solution: this is a certificate of manipulability (repair)
- else we have a proof of robustness

In practice on MINST

- 4 blocks BNN with 100 to 200 neurons per layer, L_∞ norm
- Millions of clauses: Glucose[1] certifies local (non) robustness for most input in $< 5'$ CPU time



Can also prove that some network are locally robust

Take away messages

NP is not exactly what we tend to think

- AI, OR and CS have made drastic progress in their reasoning capacities
- For SAT, this progress also comes from logical learning

Differentiable and non differentiable AI together

- Logic can analyze and exploit learnt models (not only Neural Nets)
- Intuition can help logic without tainting it (guidance)

References I

[1] Gilles Audemard and Laurent Simon. "Predicting Learnt Clauses Quality in Modern SAT Solvers.". In: *International Joint Conference in AI*. Vol. 9. 2009, pp. 399–404.

[2] Maurice Bruynooghe and Luıs Moniz Pereira. "Deduction revision by intelligent backtracking". In: (1984).

[3] Chih-Hong Cheng, Georg Nührenberg, and Harald Ruess. "Verification of binarized neural networks via Inter-Neuron Factoring". In: *arXiv preprint arXiv:1710.03107* (2017).

[4] Association for Constraint Programming. *Publication statistics in CP per country every year*. URL `http://www.a4cp.org/cparchive/countries_by_year`.

[5] Martin C Cooper et al. "Soft arc consistency revisited". In: *Artificial Intelligence* 174.7 (2010), pp. 449–478.

[6] Matthieu Courbariaux et al. "Binarized neural networks: Training deep neural networks with weights and activations constrained to+ 1 or-1". In: *arXiv preprint arXiv:1602.02830* (2016).

[7] Martin Davis, George Logemann, and Donald Loveland. "A machine program for theorem-proving". In: *Communications of the ACM* 5.7 (1962), pp. 394–397.

[8] Martin Davis and Hilary Putnam. "A computing procedure for quantification theory". In: *Journal of the ACM (JACM)* 7.3 (1960), pp. 201–215.

[9] Leonardo De Moura and Nikolaj Bjørner. "Satisfiability modulo theories: introduction and applications". In: *Communications of the ACM* 54.9 (2011), pp. 69–77.

[10] Rina Dechter. "Enhancement schemes for constraint processing: Backjumping, learning, and cutset decomposition". In: *Artificial Intelligence* 41.3 (1990), pp. 273–312.

[11] Marijn JH Heule, Oliver Kullmann, and Victor W Marek. “Solving and verifying the boolean pythagorean triples problem via cube-and-conquer”. In: *International Conference on Theory and Applications of Satisfiability Testing*. Springer. 2016, pp. 228–245.

[12] Barry Hurley et al. “Multi-language evaluation of exact solvers in graphical model discrete optimization”. In: *Constraints* (2016), pp. 1–22.

[13] Guy Katz et al. “Reluplex: An efficient SMT solver for verifying deep neural networks”. In: *International Conference on Computer Aided Verification*. Springer. 2017, pp. 97–117.

[14] Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

[15] Oliver Kullmann. “The Science of Brute Force”. In: *Communications of the ACM* (2017).

[16] Lindsey Kuper et al. “Toward Scalable Verification for Safety-Critical Deep Networks”. In: *Proc. of SysML’2018 conference.* Stanford, USA, 2018. URL: `arXiv%20preprint%20arXiv:1801.05950`.

[17] Evelyn Lamb. “Maths proof smashes size record: supercomputer produces a 200-terabyte proof–but is it really mathematics?” In: *Nature* 534.7605 (2016), pp. 17–19.

[18] João P Marques-Silva and Karem A Sakallah. “GRASP: A search algorithm for propositional satisfiability”. In: *IEEE Transactions on Computers* 48.5 (1999), pp. 506–521.

[19] David A McAllester. *An Outlook on Truth Maintenance*. Tech. rep. Massachusetts Inst Of Tech, Cambridge, Artificial Intelligence Lab., 1980.

[20] Antonio Morgado et al. “Iterative and core-guided MaxSAT solving: A survey and assessment”. In: *Constraints* 18.4 (2013), pp. 478–534.

References III

[21] Matthew W Moskewicz et al. "Chaff: Engineering an efficient SAT solver". In: *Proceedings of the 38th annual Design Automation Conference*. ACM. 2001, pp. 530–535.

[22] Nina Narodytska et al. "Verifying properties of binarized deep neural networks". In: *arXiv preprint arXiv:1709.06662* (2017).

[23] Luca Pulina and Armando Tacchella. "An abstraction-refinement approach to verification of artificial neural networks". In: *International Conference on Computer Aided Verification*. Springer. 2010, pp. 243–257.

[24] Luca Pulina and Armando Tacchella. "Challenging SMT solvers to verify neural networks". In: *AI Communications* 25.2 (2012), pp. 117–135.

[25] John Alan Robinson. "A machine-oriented logic based on the resolution principle". In: *Journal of the ACM (JACM)* 12.1 (1965), pp. 23–41.

[26] Francesca Rossi, Peter Van Beek, and Toby Walsh. *Handbook of constraint programming*. Elsevier, 2006.

[27] Wenjie Ruan, Xiaowei Huang, and Marta Kwiatkowska. "Reachability analysis of deep neural networks with provable guarantees". In: *arXiv preprint arXiv:1805.02242* (2018).

[28] Thomas Schiex and Gérard Verfaillie. "Nogood recording for static and dynamic constraint satisfaction problems". In: *International Journal on Artificial Intelligence Tools* 3.02 (1994), pp. 187–207.

[29] Richard M Stallman and Gerald J Sussman. "Forward reasoning and dependency-directed backtracking in a system for computer-aided circuit analysis". In: *Artificial intelligence* 9.2 (1977), pp. 135–196.

[30] Christian Szegedy et al. "Intriguing properties of neural networks". In: *arXiv preprint arXiv:1312.6199* (2013).

2017: proving an "alien" theorem?

A conjecture in combinatorics ∞

When one splits $\mathbb{N}$ in 2, one part must contain a Pythagorean triple $(a^2 = b^2 + c^2)$

2017: proving an “alien” theorem?

A conjecture in combinatorics ∞

When one splits $\mathbb{N}$ in 2, one part must contain a Pythagorean triple $(a^2 = b^2 + c^2)$

No known proof, puzzled mathematicians for decades (one offered a 100 $ reward)

2017: proving an "alien" theorem?

A conjecture in combinatorics ∞

When one splits $\mathbb{N}$ in 2, one part must contain a Pythagorean triple $(a^2 = b^2 + c^2)$

No known proof, puzzled mathematicians for decades (one offered a 100 $ reward)

SAT solver proof[11,17]

$200\,TB$ proof, compressed to $86\,GB$ (stronger proof system)[a]

[a] Oliver Kullmann. "The Science of Brute Force". In: *Communications of the ACM* (2017).