AUTOMATING MATHEMATICS?

SIDDHARTHA GADGIL

Department of Mathematics

Indian Institute of Science

Bangalore

http://math.iisc.ac.in/~gadgil/

https://github.com/siddhartha-gadgil/ProvingGround

1997

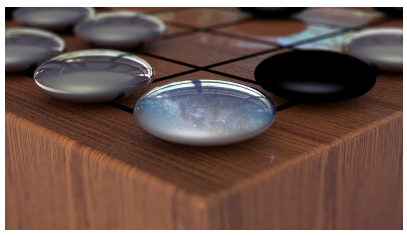
1997

2017

1997

2017

1997



2017

?

1997

2017

?

Whether?
When?
How?

WHAT IS MATHEMATICS?

PRIME NUMBERS

PRIME NUMBERS

- A prime number is a number with exactly two factors, 1 and the number itself.

PRIME NUMBERS

- A prime number is a number with exactly two factors, 1 and the number itself.
- For example, 3, 5 and 7 are primes but 4 is not a prime as it has factor 2.

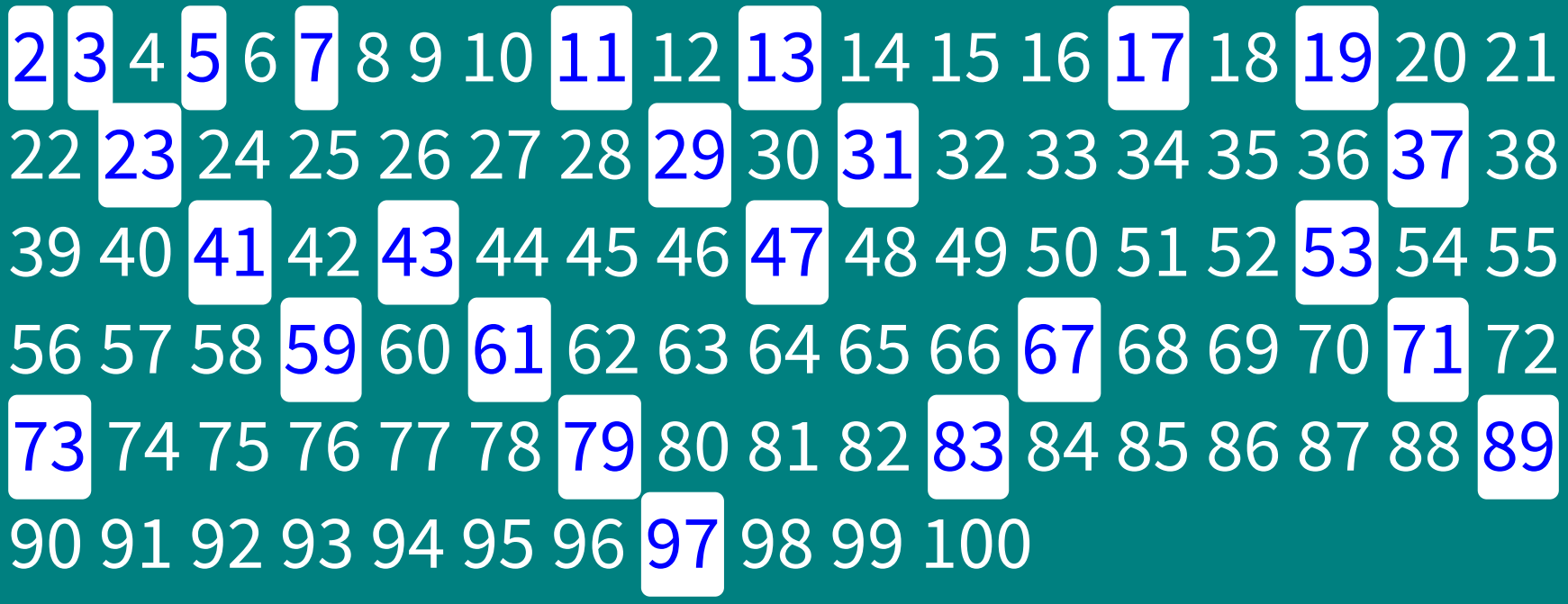
PRIME NUMBERS

- A prime number is a number with exactly two factors, 1 and the number itself.
- For example, 3, 5 and 7 are primes but 4 is not a prime as it has factor 2.
- Primes are a basic object of study in Mathematics.

PRIME NUMBERS

- A prime number is a number with exactly two factors, 1 and the number itself.
- For example, 3, 5 and 7 are primes but 4 is not a prime as it has factor 2.
- Primes are a basic object of study in Mathematics.
- We will discuss what we study first, and then why.

HOW MANY PRIMES?



401 402 403 404 405 406 407 408 409 410 411 412
413 414 415 416 417 418 419 420 421 422 423 424
425 426 427 428 429 430 431 432 433 434 435 436
437 438 439 440 441 442 443 444 445 446 447 448
449 450 451 452 453 454 455 456 457 458 459 460
461 462 463 464 465 466 467 468 469 470 471 472
473 474 475 476 477 478 479 480 481 482 483 484
485 486 487 488 489 490 491 492 493 494 495 496
497 498 499 500

901 902 903 904 905 906 907 908 909 910 911 912
913 914 915 916 917 918 919 920 921 922 923 924
925 926 927 928 929 930 931 932 933 934 935 936
937 938 939 940 941 942 943 944 945 946 947 948
949 950 951 952 953 954 955 956 957 958 959 960
961 962 963 964 965 966 967 968 969 970 971 972
973 974 975 976 977 978 979 980 981 982 983 984
985 986 987 988 989 990 991 992 993 994 995 996
997 998 999 1000

9901 9902 9903 9904 9905 9906 9907 9908 9909
9910 9911 9912 9913 9914 9915 9916 9917 9918
9919 9920 9921 9922 9923 9924 9925 9926 9927
9928 9929 9930 9931 9932 9933 9934 9935 9936
9937 9938 9939 9940 9941 9942 9943 9944 9945
9946 9947 9948 9949 9950 9951 9952 9953 9954
9955 9956 9957 9958 9959 9960 9961 9962 9963
9964 9965 9966 9967 9968 9969 9970 9971 9972
9973 9974 9975 9976 9977 9978 9979 9980 9981
9982 9983 9984 9985 9986 9987 9988 9989 9990
9991 9992 9993 9994 9995 9996 9997 9998 9999
10000

HOW ABOUT TWIN PRIMES?

2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38
39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 55
56 57 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72
73 74 75 76 77 78 79 80 81 82 83 84 85 86 87 88 89
90 91 92 93 94 95 96 97 98 99 100

401 402 403 404 405 406 407 408 409 410 411 412
413 414 415 416 417 418 419 420 421 422 423 424
425 426 427 428 429 430 431 432 433 434 435 436
437 438 439 440 441 442 443 444 445 446 447 448
449 450 451 452 453 454 455 456 457 458 459 460
461 462 463 464 465 466 467 468 469 470 471 472
473 474 475 476 477 478 479 480 481 482 483 484
485 486 487 488 489 490 491 492 493 494 495 496
497 498 499 500

901 902 903 904 905 906 907 908 909 910 911 912
913 914 915 916 917 918 919 920 921 922 923 924
925 926 927 928 929 930 931 932 933 934 935 936
937 938 939 940 941 942 943 944 945 946 947 948
949 950 951 952 953 954 955 956 957 958 959 960
961 962 963 964 965 966 967 968 969 970 971 972
973 974 975 976 977 978 979 980 981 982 983 984
985 986 987 988 989 990 991 992 993 994 995 996
997 998 999 1000

9901 9902 9903 9904 9905 9906 9907 9908 9909
9910 9911 9912 9913 9914 9915 9916 9917 9918
9919 9920 9921 9922 9923 9924 9925 9926 9927
9928 9929 9930 9931 9932 9933 9934 9935 9936
9937 9938 9939 9940 9941 9942 9943 9944 9945
9946 9947 9948 9949 9950 9951 9952 9953 9954
9955 9956 9957 9958 9959 9960 9961 9962 9963
9964 9965 9966 9967 9968 9969 9970 9971 9972
9973 9974 9975 9976 9977 9978 9979 9980 9981
9982 9983 9984 9985 9986 9987 9988 9989 9990
9991 9992 9993 9994 9995 9996 9997 9998 9999
10000

ARITHMETIC PROGRESSIONS IN PRIMES

ARITHMETIC PROGRESSIONS IN PRIMES

- An *arithmetic progression* is a sequence of numbers so that the difference between two consecutive terms is constant, for example
  - 5, 7, 9, 11.
  - 4, 7, 10, 13, 16.
  - 11, 17, 23, 29.

ARITHMETIC PROGRESSIONS IN PRIMES

- An *arithmetic progression* is a sequence of numbers so that the difference between two consecutive terms is constant, for example
  - 5, 7, 9, 11.
  - 4, 7, 10, 13, 16.
  - 11, 17, 23, 29.
- We consider arithmetic progressions all of whose terms are primes, such as the third example.

ARITHMETIC PROGRESSIONS IN PRIMES

- An *arithmetic progression* is a sequence of numbers so that the difference between two consecutive terms is constant, for example
  - 5, 7, 9, 11.
  - 4, 7, 10, 13, 16.
  - 11, 17, 23, 29.
- We consider arithmetic progressions all of whose terms are primes, such as the third example.
- For experiments, we take the first million pairs of primes and extend these while we get primes.

- Some prime arithmetic progressions of length 3:

3 5 7	3 7 11	3 11 19	3 13 23	3 17 31	3 23 43	3 31 59	3 37 71	3 41 79	3 43 83
3 53 103	5 29 53	7 13 19	7 43 79	11 29 47	11 47 83	11 59 107	13 37 61		
17 23 29	17 53 89	17 59 101	19 31 43	19 43 67	19 61 103	23 41 59	23 47 71		
29 41 53	29 59 89	31 37 43	47 53 59						

- There are 110386 of these among the first million.

- Some prime arithmetic progressions of length 4:

5 23 41 59	5 59 113 167	5 89 173 257	7 19 31 43	7 79 151 223	11 17 23 29
13 43 73 103	13 61 109 157	17 29 41 53	19 73 127 181	19 79 139 199	23 53 83 113
29 83 137 191	31 67 103 139	41 47 53 59	41 71 101 131	43 61 79 97	47 59 71 83
47 89 131 173	53 71 89 107	59 83 107 131	61 67 73 79		

- There are 27836 of these among the first million.

- Some prime arithmetic progressions of length 5:

5 11 17 23 29	5 17 29 41 53	5 47 89 131 173	5 53 101 149 197	5 101 197 293 389
5 131 257 383 509	11 41 71 101 131	13 163 313 463 613	17 167 317 467 617	
29 149 269 389 509	37 67 97 127 157	43 103 163 223 283	61 151 241 331 421	
71 131 191 251 311	83 173 263 353 443	89 179 269 359 449	113 173 233 293 353	
137 167 197 227 257				

- There are 3665 of these among the first million.

- Some prime arithmetic progressions of length 6:

7 37 67 97 127 157	11 71 131 191 251 311	13 103 193 283 373 463
13 223 433 643 853 1063	23 263 503 743 983 1223	41 461 881 1301 1721 2141
53 113 173 233 293 353	73 223 373 523 673 823	83 383 683 983 1283 1583
107 137 167 197 227 257	127 457 787 1117 1447 1777	157 307 457 607 757 907
239 359 479 599 719 839	281 401 521 641 761 881	359 389 419 449 479 509

- There are 980 of these among the first million.

- Some prime arithmetic progressions of length 7:

7 157 307 457 607 757 907

47 257 467 677 887 1097 1307

53 1103 2153 3203 4253 5303 6353

71 2381 4691 7001 9311 11621 13931

179 389 599 809 1019 1229 1439

193 613 1033 1453 1873 2293 2713

359 1619 2879 4139 5399 6659 7919

829 1039 1249 1459 1669 1879 2089

1061 1901 2741 3581 4421 5261 6101

1091 1301 1511 1721 1931 2141 2351

1453 1663 1873 2083 2293 2503 2713

- There are 127 of these among the first million.

- Some prime arithmetic progressions of length 8:

619 829 1039 1249 1459 1669 1879 2089

881 1091 1301 1511 1721 1931 2141 2351

1019 3329 5639 7949 10259 12569 14879 17189

1091 3821 6551 9281 12011 14741 17471 20201

1289 2969 4649 6329 8009 9689 11369 13049

1637 2267 2897 3527 4157 4787 5417 6047

1847 3947 6047 8147 10247 12347 14447 16547

2239 2659 3079 3499 3919 4339 4759 5179

2693 4583 6473 8363 10253 12143 14033 15923

3323 4583 5843 7103 8363 9623 10883 12143

- There are 46 of these among the first million.

17 6947 13877 20807 27737 34667 41597 48527 55457

137 8117 16097 24077 32057 40037 48017 55997 63977

409 619 829 1039 1249 1459 1669 1879 2089

433 3583 6733 9883 13033 16183 19333 22483 25633

1699 5689 9679 13669 17659 21649 25639 29629 33619

2063 3323 4583 5843 7103 8363 9623 10883 12143

3499 3709 3919 4129 4339 4549 4759 4969 5179

3823 6133 8443 10753 13063 15373 17683 19993 22303

4721 7451 10181 12911 15641 18371 21101 23831 26561

6043 6883 7723 8563 9403 10243 11083 11923 12763

- There is just 1 arithmetic progressions of length 10 among the first million.

```
199 409 619 829 1039 1249 1459 1669 1879 2089
```

- The largest explicitly known arithmetic progression is of length 27.

LENGTHS OF ARITHMETIC PROGRESSIONS

LENGTHS OF ARITHMETIC PROGRESSIONS

- For the first million pairs of primes, the longest arithmetic progression of primes starting with these have lengths as follows.

LENGTHS OF ARITHMETIC PROGRESSIONS

- For the first million pairs of primes, the longest arithmetic progression of primes starting with these have lengths as follows.

length	2	3	4	5	6	7	8	9	10
number	856944	110386	27836	3665	980	127	46	15	1

LENGTHS OF ARITHMETIC PROGRESSIONS

- For the first million pairs of primes, the longest arithmetic progression of primes starting with these have lengths as follows.

length	2	3	4	5	6	7	8	9	10
number	856944	110386	27836	3665	980	127	46	15	1

- The largest explicitly known arithmetic progression is of length 27.

SOME QUESTIONS

SOME QUESTIONS

- **Question:** Are there infinitely many primes?

SOME QUESTIONS

- **Question:** Are there infinitely many primes?
- **Answer:** Yes, as shown by Euclid.

SOME QUESTIONS

- **Question:** Are there infinitely many primes?
- **Answer:** Yes, as shown by Euclid.
- **Question:** Are there arbitrarily long arithmetic progressions of primes?

SOME QUESTIONS

- **Question:** Are there infinitely many primes?
- **Answer:** Yes, as shown by Euclid.
- **Question:** Are there arbitrarily long arithmetic progressions of primes?
- **Answer:** Yes, by the Green-Tao theorem from 2004.

SOME QUESTIONS

- **Question:** Are there infinitely many primes?
- **Answer:** Yes, as shown by Euclid.
- **Question:** Are there arbitrarily long arithmetic progressions of primes?
- **Answer:** Yes, by the Green-Tao theorem from 2004.
- **Question:** Are there infinitely many twin primes?

SOME QUESTIONS

- **Question:** Are there infinitely many primes?
- **Answer:** Yes, as shown by Euclid.
- **Question:** Are there arbitrarily long arithmetic progressions of primes?
- **Answer:** Yes, by the Green-Tao theorem from 2004.
- **Question:** Are there infinitely many twin primes?
- **Status:** We still do not know.

WHAT IS THE USE?

WHAT IS THE USE?

- Occasionally, a mathematical result finds unexpected applications.

WHAT IS THE USE?

- Occasionally, a mathematical result finds unexpected applications.
- Far more often, in finding the answers to mathematical question we discover **concepts**, i.e., ways of looking at the world, which are useful.

WHAT IS THE USE?

- Occasionally, a mathematical result finds unexpected applications.
- Far more often, in finding the answers to mathematical question we discover **concepts**, i.e., ways of looking at the world, which are useful.
- For example, Gauss developed the concept of *intrinsic curvature* motivated by a geodetic survey.

WHAT IS THE USE?

- Occasionally, a mathematical result finds unexpected applications.
- Far more often, in finding the answers to mathematical question we discover **concepts**, i.e., ways of looking at the world, which are useful.
- For example, Gauss developed the concept of *intrinsic curvature* motivated by a geodetic survey.
- This was further developed, and generalized to higher dimensions by Riemann.

WHAT IS THE USE?

- Occasionally, a mathematical result finds unexpected applications.
- Far more often, in finding the answers to mathematical question we discover **concepts**, i.e., ways of looking at the world, which are useful.
- For example, Gauss developed the concept of *intrinsic curvature* motivated by a geodetic survey.
- This was further developed, and generalized to higher dimensions by Riemann.
- Einstein's **general relativity** was based on intrinsic curvature.

MATHEMATICAL PROOFS

INFINITUDE OF PRIMES

INFINITUDE OF PRIMES

- **Lemma:** Every number $n > 1$ has a prime factor.

INFINITUDE OF PRIMES

- **Lemma:** Every number $n > 1$ has a prime factor.
- **Proof:** Let $p > 1$ be the smallest factor of n that is bigger than 1.

INFINITUDE OF PRIMES

- **Lemma:** Every number $n > 1$ has a prime factor.
- **Proof:** Let $p > 1$ be the smallest factor of n that is bigger than 1.
- Then p must be a prime as

INFINITUDE OF PRIMES

- **Lemma:** Every number $n > 1$ has a prime factor.
- **Proof:** Let $p > 1$ be the smallest factor of n that is bigger than 1.
- Then p must be a prime as
  - Suppose $q > 1$ is a factor of p, then q is a factor of n.

INFINITUDE OF PRIMES

- **Lemma:** Every number $n > 1$ has a prime factor.
- **Proof:** Let $p > 1$ be the smallest factor of n that is bigger than 1.
- Then p must be a prime as
  - Suppose $q > 1$ is a factor of p, then q is a factor of n.
  - But p is the smallest factor of n, so $q = p$.

INFINITUDE OF PRIMES

- **Lemma:** Every number $n > 1$ has a prime factor.
- **Proof:** Let $p > 1$ be the smallest factor of n that is bigger than 1.
- Then p must be a prime as
  - Suppose $q > 1$ is a factor of p, then q is a factor of n.
  - But p is the smallest factor of n, so $q = p$.
  - So the only factors of p are 1 and p, i.e., p is a prime.

INFINITUDE OF PRIMES II

INFINITUDE OF PRIMES II

- **Lemma:** For any positive number n, there is a prime bigger than n.

INFINITUDE OF PRIMES II

- **Lemma:** For any positive number n, there is a prime bigger than n.
- **Proof:** Let $m = n! + 1$, where $n! = 1 \times 2 \times 3 \times \ldots n$ and let p be a *prime* factor of m.

INFINITUDE OF PRIMES II

- **Lemma:** For any positive number n, there is a prime bigger than n.
- **Proof:** Let $m = n! + 1$, where $n! = 1 \times 2 \times 3 \times \ldots n$ and let p be a *prime* factor of m.
- Then p must be greater than n, as, if $p \leq n$,

INFINITUDE OF PRIMES II

- **Lemma:** For any positive number n, there is a prime bigger than n.
- **Proof:** Let $m = n! + 1$, where $n! = 1 \times 2 \times 3 \times \ldots n$ and let p be a *prime* factor of m.
- Then p must be greater than n, as, if $p \leq n$,
  - p divides $n!$, and

INFINITUDE OF PRIMES II

- **Lemma:** For any positive number n, there is a prime bigger than n.
- **Proof:** Let $m = n! + 1$, where $n! = 1 \times 2 \times 3 \times \ldots n$ and let p be a *prime* factor of m.
- Then p must be greater than n, as, if $p \leq n$,
  - p divides $n!$, and
  - p divides $n! + 1$, which means

INFINITUDE OF PRIMES II

- **Lemma:** For any positive number n, there is a prime bigger than n.
- **Proof:** Let $m = n! + 1$, where $n! = 1 \times 2 \times 3 \times \ldots n$ and let p be a *prime* factor of m.
- Then p must be greater than n, as, if $p \leq n$,
  - p divides $n!$, and
  - p divides $n! + 1$, which means
  - p divides $1 = (n! + 1) - n!$, which is absurd.

FORMAL MATHEMATICAL PROOFS

FORMAL MATHEMATICAL PROOFS

- A formal proof or derivation is a finite sequence of sentences, each of which:
  - is an axiom (something we believe true about the universe), or
  - is an assumption, or
  - follows from the earlier sentences by a rule of inference.

FORMAL MATHEMATICAL PROOFS

- A formal proof or derivation is a finite sequence of sentences, each of which:
  - is an axiom (something we believe true about the universe), or
  - is an assumption, or
  - follows from the earlier sentences by a rule of inference.
- A formal proof can be checked mechanically.

FORMAL PROOF IN MIZAR

FORMAL PROOF IN MIZAR

```
reserve n,p for Nat;
theorem Euclid: ex p st p is prime & p > n
proof
set k = n! + 1;
n! > 0 by NEWTON:23;
then n! >= 0 + 1 by NAT_1:38;
then k >= 1 + 1 by REAL_1:55;
then consider p such that
A1: p is prime & p divides k by INT_2:48;
A2: p <> 0 & p > 1 by A1,INT_2:def 5;
take p;
thus p is prime by A1;
assume p <= n;
then p divides n! by A2,NAT_LAT:16;
then p divides 1 by A1,NAT_1:57;
hence contradiction by A2,NAT_1:54;
end;
theorem {p: p is prime} is infinite
from Unbounded(Euclid);
```

FORMAL PROOF IN MIZAR

```
reserve n,p for Nat;
theorem Euclid: ex p st p is prime & p > n
proof
set k = n! + 1;
n! > 0 by NEWTON:23;
then n! >= 0 + 1 by NAT_1:38;
then k >= 1 + 1 by REAL_1:55;
then consider p such that
A1: p is prime & p divides k by INT_2:48;
A2: p <> 0 & p > 1 by A1,INT_2:def 5;
take p;
thus p is prime by A1;
assume p <= n;
then p divides n! by A2,NAT_LAT:16;
then p divides 1 by A1,NAT_1:57;
hence contradiction by A2,NAT_1:54;
end;
theorem {p: p is prime} is infinite
from Unbounded(Euclid);
```

Full proof of infinitude of primes is 44 lines long.


PUZZLES, GAMES,
REASONING

PUZZLES

PUZZLES

- **Some puzzles:** jigsaw, sudoku, detective stories, quiz questions, planetary motions, ...

PUZZLES

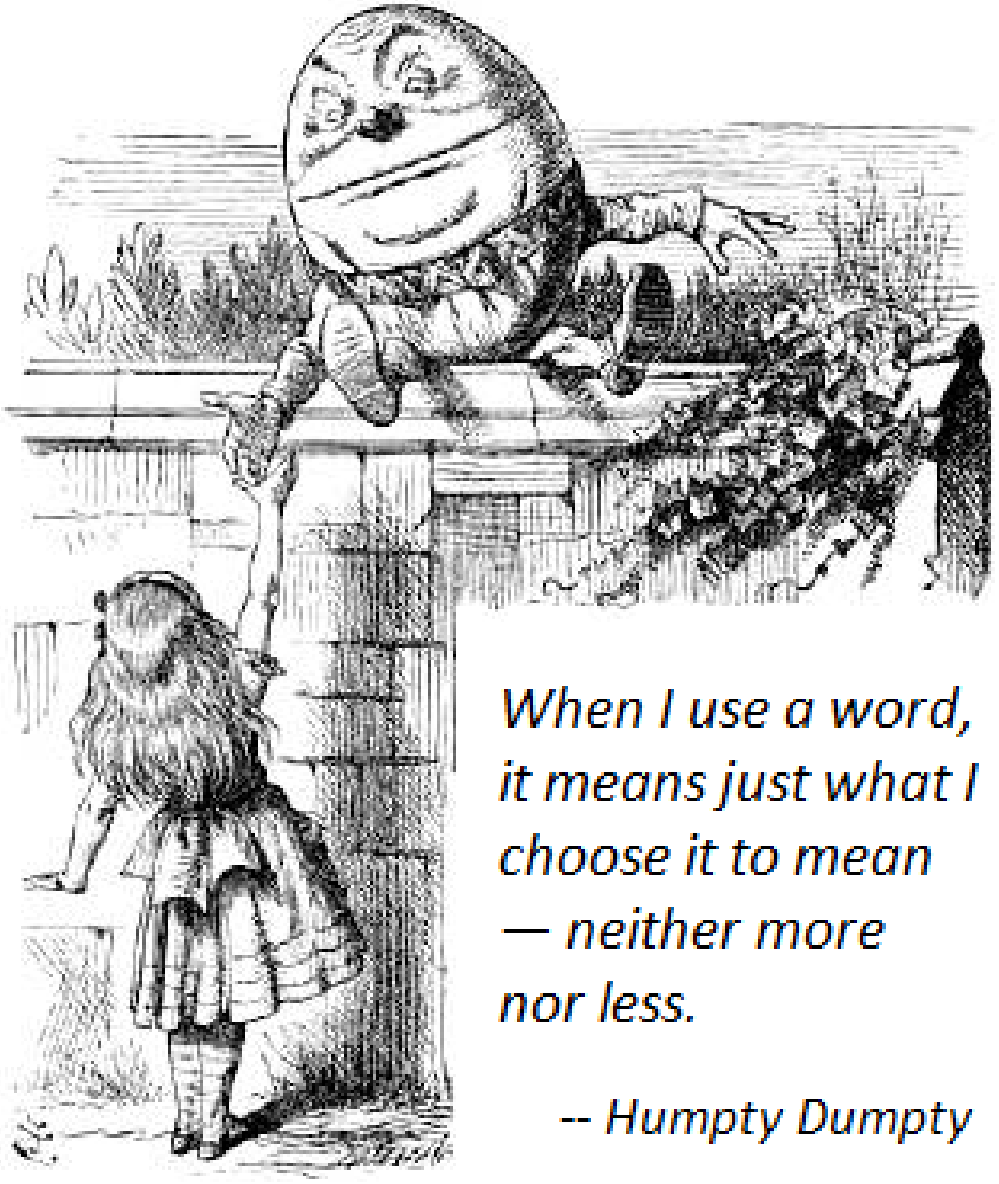
- **Some puzzles:** jigsaw, sudoku, detective stories, quiz questions, planetary motions, ...
- A puzzle is a precisely stated problem for which
  - it is (fairly) easy to *check* that a solution is correct, but
  - it is hard to *find* the solution.

PUZZLES

- **Some puzzles:** jigsaw, sudoku, detective stories, quiz questions, planetary motions, ...
- A puzzle is a precisely stated problem for which
  - it is (fairly) easy to *check* that a solution is correct, but
  - it is hard to *find* the solution.
- A solution may be **formal** or **informal**.

PUZZLES

- **Some puzzles:** jigsaw, sudoku, detective stories, quiz questions, planetary motions, ...
- A puzzle is a precisely stated problem for which
  - it is (fairly) easy to *check* that a solution is correct, but
  - it is hard to *find* the solution.
- A solution may be **formal** or **informal**.
- We solve puzzles by a mixture of *deduction* (algorithms) and *intuition*.



When I use a word, it means just what I choose it to mean — neither more nor less.

-- Humpty Dumpty

DEDUCTIVE REASONING/ALGORITHMS

DEDUCTIVE REASONING/ALGORITHMS

- We perform a small number of steps, each step a calculation or move or deduction.

DEDUCTIVE REASONING/ALGORITHMS

- We perform a small number of steps, each step a calculation or move or deduction.
- The steps are based on a small number of rules (possibly depending on a small number of parameters).

DEDUCTIVE REASONING/ALGORITHMS

- We perform a small number of steps, each step a calculation or move or deduction.
- The steps are based on a small number of rules (possibly depending on a small number of parameters).

- A computer's notion of *small* is very different.

DEDUCTIVE REASONING/ALGORITHMS

  - We perform a small number of steps, each step a calculation or move or deduction.
  - The steps are based on a small number of rules (possibly depending on a small number of parameters).
- A computer's notion of *small* is very different.
- There have been huge improvements in algorithms:

DEDUCTIVE REASONING/ALGORITHMS

  - We perform a small number of steps, each step a calculation or move or deduction.
  - The steps are based on a small number of rules (possibly depending on a small number of parameters).
- A computer's notion of *small* is very different.
- There have been huge improvements in algorithms: *multiplication with carry-over,*

DEDUCTIVE REASONING/ALGORITHMS

  - We perform a small number of steps, each step a calculation or move or deduction.
  - The steps are based on a small number of rules (possibly depending on a small number of parameters).
- A computer's notion of *small* is very different.
- There have been huge improvements in algorithms: *multiplication with carry-over, fingerprints,*

DEDUCTIVE REASONING/ALGORITHMS

  - We perform a small number of steps, each step a calculation or move or deduction.
  - The steps are based on a small number of rules (possibly depending on a small number of parameters).
- A computer's notion of *small* is very different.
- There have been huge improvements in algorithms: *multiplication with carry-over, fingerprints, SMT solvers.*

TACIT KNOWLEDGE & INTUITION

TACIT KNOWLEDGE & INTUITION

- *Tacit knowledge* is the kind of knowledge that is difficult to transfer to another person by means of writing it down or verbalizing it.

TACIT KNOWLEDGE & INTUITION

- *Tacit knowledge* is the kind of knowledge that is difficult to transfer to another person by means of writing it down or verbalizing it.
- **Examples:** riding a bicycle, speaking a language.

TACIT KNOWLEDGE & INTUITION

- *Tacit knowledge* is the kind of knowledge that is difficult to transfer to another person by means of writing it down or verbalizing it.
- **Examples:** riding a bicycle, speaking a language.
- Experts have a lot of tacit knowledge, typically learned through experience.

TACIT KNOWLEDGE & INTUITION

- *Tacit knowledge* is the kind of knowledge that is difficult to transfer to another person by means of writing it down or verbalizing it.
- **Examples:** riding a bicycle, speaking a language.
- Experts have a lot of tacit knowledge, typically learned through experience.
- Intuition is based on tacit knowledge.

TACIT KNOWLEDGE & INTUITION

- *Tacit knowledge* is the kind of knowledge that is difficult to transfer to another person by means of writing it down or verbalizing it.
- **Examples:** riding a bicycle, speaking a language.
- Experts have a lot of tacit knowledge, typically learned through experience.
- Intuition is based on tacit knowledge.
- While intuition is sometimes wrong, to be useful it should be correct *often enough*.
    - A *hunch* is sometimes correct.
    - A *judgement* is often correct.

SOLVING PUZZLES

SOLVING PUZZLES

- Checking a solution should be purely deductive.

SOLVING PUZZLES

- Checking a solution should be purely deductive.
- However, finding a solution involves:

SOLVING PUZZLES

- Checking a solution should be purely deductive.
- However, finding a solution involves:
  - deciding what to consider - a *policy*, which may use intuitive *hunches*, and

SOLVING PUZZLES

- Checking a solution should be purely deductive.
- However, finding a solution involves:
  - deciding what to consider - a *policy*, which may use intuitive *hunches*, and
  - deciding how promising the present approach/situation is - the *value*, which may use intuitive *judgements*.

SOLVING PUZZLES

- Checking a solution should be purely deductive.
- However, finding a solution involves:
  - deciding what to consider - a *policy*, which may use intuitive *hunches*, and
  - deciding how promising the present approach/situation is - the *value*, which may use intuitive *judgements*.
- A computer following a purely algorithmic approach can compensate by following up on far more options, and looking more moves ahead before deciding the value.

COMPUTERS AND GAMES

REWARDS, VALUES AND POLICIES

REWARDS, VALUES AND POLICIES

- The rules of a game (e.g. chess) tell us

REWARDS, VALUES AND POLICIES

- The rules of a game (e.g. chess) tell us
  - what moves we can make.

REWARDS, VALUES AND POLICIES

- The rules of a game (e.g. chess) tell us
  - what moves we can make.
  - what *reward* we get at a stage - e.g. win/loss/draw at the end.

REWARDS, VALUES AND POLICIES

- The rules of a game (e.g. chess) tell us
  - what moves we can make.
  - what *reward* we get at a stage - e.g. win/loss/draw at the end.
- In tic-tac-toe, we can simply calculate the reward.

REWARDS, VALUES AND POLICIES

- The rules of a game (e.g. chess) tell us
  - what moves we can make.
  - what *reward* we get at a stage - e.g. win/loss/draw at the end.
- In tic-tac-toe, we can simply calculate the reward.
- In most cases however, we need

REWARDS, VALUES AND POLICIES

- The rules of a game (e.g. chess) tell us
  - what moves we can make.
  - what *reward* we get at a stage - e.g. win/loss/draw at the end.
- In tic-tac-toe, we can simply calculate the reward.
- In most cases however, we need
  - A *policy* - what moves to consider.

REWARDS, VALUES AND POLICIES

- The rules of a game (e.g. chess) tell us
  - what moves we can make.
  - what *reward* we get at a stage - e.g. win/loss/draw at the end.
- In tic-tac-toe, we can simply calculate the reward.
- In most cases however, we need
  - A *policy* - what moves to consider.
  - A (relative) *value* telling us what future reward we can expect based on the present position.

KASPAROV VS DEEP BLUE

KASPAROV VS DEEP BLUE

- In Chess, a basic **value** is obtained by counting pieces and pawns with weights.

KASPAROV VS DEEP BLUE

- In Chess, a basic **value** is obtained by counting pieces and pawns with weights.
- *Standard openings* also give a **policy** during the early stages of the game, as do *endgame tables*.

KASPAROV VS DEEP BLUE

- In Chess, a basic **value** is obtained by counting pieces and pawns with weights.
- *Standard openings* also give a **policy** during the early stages of the game, as do *endgame tables*.
- Deep Blue, and chess theory, extend these to elaborate, rule based values and policies.

KASPAROV VS DEEP BLUE

- In Chess, a basic **value** is obtained by counting pieces and pawns with weights.
- *Standard openings* also give a **policy** during the early stages of the game, as do *endgame tables*.
- Deep Blue, and chess theory, extend these to elaborate, rule based values and policies.
- The value and policy functions of Kasparov were far better, but compensated for by Deep Blue being able to consider far more move sequences.

ALPHAGO VS LEE SEDOL

ALPHAGO VS LEE SEDOL

- In the chinese game Go, the number of legal moves is much larger, so *trying everything* leads to too many moves.

ALPHAGO VS LEE SEDOL

- In the chinese game Go, the number of legal moves is much larger, so *trying everything* leads to too many moves.
- More importantly, it is very hard to describe a good value function.

ALPHAGO VS LEE SEDOL

- In the chinese game Go, the number of legal moves is much larger, so *trying everything* leads to too many moves.
- More importantly, it is very hard to describe a good value function.
- This makes it far harder for computers.

ALPHAGO VS LEE SEDOL

- In the chinese game Go, the number of legal moves is much larger, so *trying everything* leads to too many moves.
- More importantly, it is very hard to describe a good value function.
- This makes it far harder for computers.
- Yet, in March 2016, a Go playing system AlphaGo defeated 18-time world champion Lee Sedol.

ALPHAGO VS LEE SEDOL

- In the chinese game Go, the number of legal moves is much larger, so *trying everything* leads to too many moves.
- More importantly, it is very hard to describe a good value function.
- This makes it far harder for computers.
- Yet, in March 2016, a Go playing system AlphaGo defeated 18-time world champion Lee Sedol.
- In January 2017, AlphaGo defeated the world number one Ke Jie comprehensively.

ALPHAGO AND LEARNING

ALPHAGO AND LEARNING

- The policy and value functions of AlphaGo are deep neural networks that were *trained*.

ALPHAGO AND LEARNING

- The policy and value functions of AlphaGo are deep neural networks that were *trained*.
- The policy network was trained by learning to predict the next move from games of expert players.

ALPHAGO AND LEARNING

- The policy and value functions of AlphaGo are deep neural networks that were *trained*.
- The policy network was trained by learning to predict the next move from games of expert players.
- The value network was trained by AlphaGo playing against versions of itself.

ALPHAGO AND LEARNING

- The policy and value functions of AlphaGo are deep neural networks that were *trained*.
- The policy network was trained by learning to predict the next move from games of expert players.
- The value network was trained by AlphaGo playing against versions of itself.
- AlphaGo considered fewer sequences of moves than Deep Blue.

ALPHAGO AND LEARNING

- The policy and value functions of AlphaGo are deep neural networks that were *trained*.
- The policy network was trained by learning to predict the next move from games of expert players.
- The value network was trained by AlphaGo playing against versions of itself.
- AlphaGo considered fewer sequences of moves than Deep Blue.
- AlphaGo came up with unexpected moves.

ALPHAGO ZERO AND ALPHA ZERO

ALPHAGO ZERO AND ALPHA ZERO

- AlphaGo was succeeded (and defeated) by *AlphaGo Zero*, which learnt purely by self play.

ALPHAGO ZERO AND ALPHA ZERO

- AlphaGo was succeeded (and defeated) by *AlphaGo Zero*, which learnt purely by self play.
- Its successor, *AlphaZero*, could master a variety of similar games starting with just the rules.

ALPHAGO ZERO AND ALPHA ZERO

- AlphaGo was succeeded (and defeated) by *AlphaGo Zero*, which learnt purely by self play.
- Its successor, *AlphaZero*, could master a variety of similar games starting with just the rules.
- AlphaZero took just 4 hours to become the strongest chess player on the planet (beating a traditional chess program, Stockfish).

ALPHAGO ZERO AND ALPHA ZERO

- AlphaGo was succeeded (and defeated) by *AlphaGo Zero*, which learnt purely by self play.
- Its successor, *AlphaZero*, could master a variety of similar games starting with just the rules.
- AlphaZero took just 4 hours to become the strongest chess player on the planet (beating a traditional chess program, Stockfish).
- AlphaZero "had a dynamic, open style", and "prioritizes piece activity over material, preferring positions that looked risky and aggressive."


ARTIFICIAL
INTELLIGENCE
ELSEWHERE.

GENERATIVE QUERY NETWORK

GENERATIVE QUERY NETWORK

- In an artifical 3D environment, the network observes 2D images from a few positions.

GENERATIVE QUERY NETWORK

- In an artifical 3D environment, the network observes 2D images from a few positions.
- It has to predict the observed image from a new position.

GENERATIVE QUERY NETWORK

- In an artifical 3D environment, the network observes 2D images from a few positions.
- It has to predict the observed image from a new position.
- To do this, the 2D image was mapped to a *concise representation* by a network, which was then used to predict the image from a different viewpoint.

GENERATIVE QUERY NETWORK

- In an artifical 3D environment, the network observes 2D images from a few positions.
- It has to predict the observed image from a new position.
- To do this, the 2D image was mapped to a *concise representation* by a network, which was then used to predict the image from a different viewpoint.
- The concise representation factorized by colour, shape and size (among other things).

GENERATIVE ADVERSARIAL NETWORK

GENERATIVE ADVERSARIAL NETWORK

- These consist of a pair of networks, contesting with each other.

GENERATIVE ADVERSARIAL NETWORK

- These consist of a pair of networks, contesting with each other.
- One network generates candidates (generative) and the other evaluates them (discriminative).

GENERATIVE ADVERSARIAL NETWORK

- These consist of a pair of networks, contesting with each other.
- One network generates candidates (generative) and the other evaluates them (discriminative).
- For example the discriminative network tries to distinguish between real images and synthetic ones generated by the generative network.

DISTRIBUTIONAL REINFORCEMENT LEARNING

DISTRIBUTIONAL REINFORCEMENT LEARNING

- In *temporal reinforcement learning*, a network tries to predict (average) future rewards.

DISTRIBUTIONAL REINFORCEMENT LEARNING

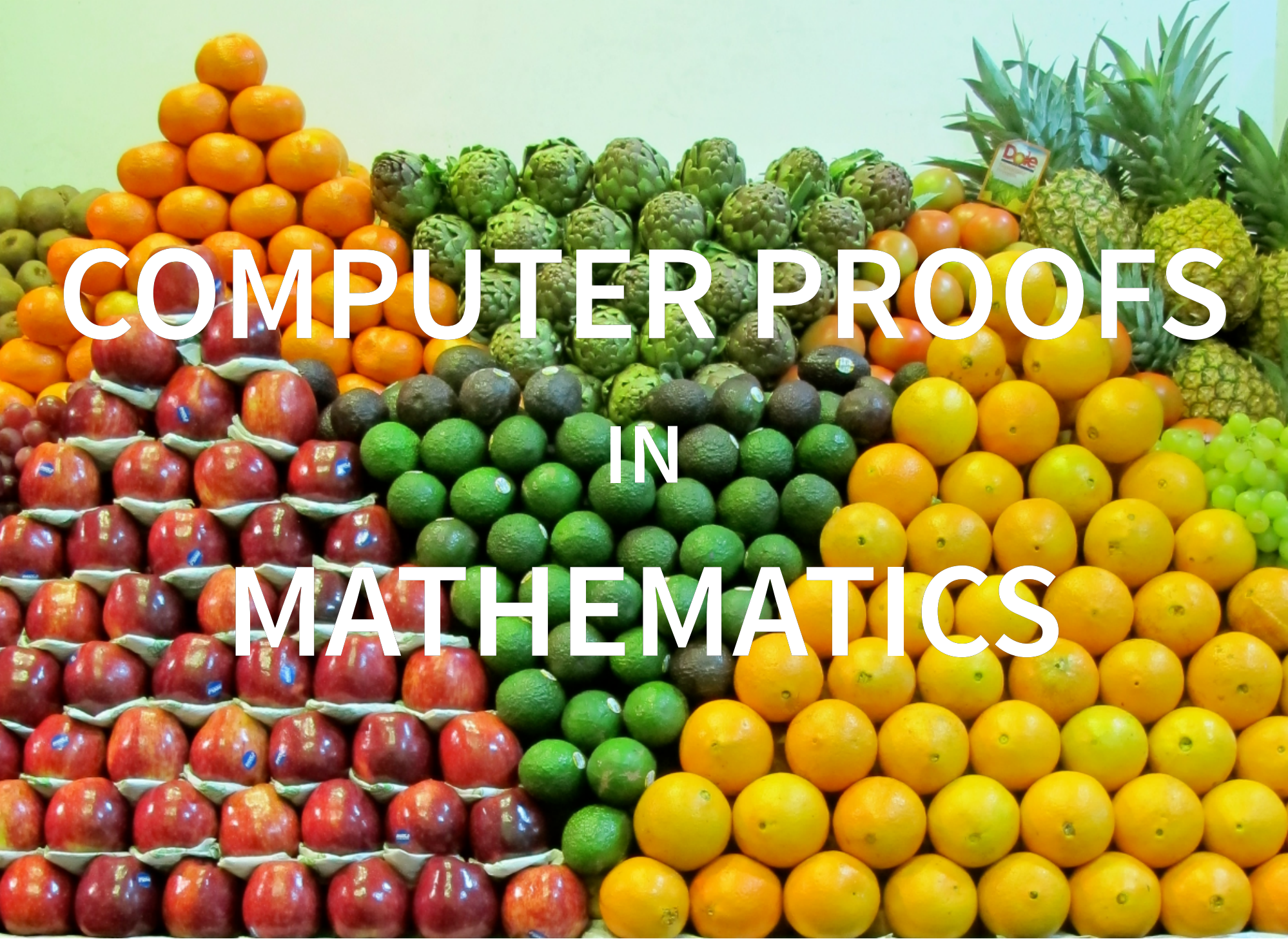
- In *temporal reinforcement learning*, a network tries to predict (average) future rewards.
- However, sometimes the reward is either very big or very small, so the average reward is misleading.

DISTRIBUTIONAL REINFORCEMENT LEARNING

- In *temporal reinforcement learning*, a network tries to predict (average) future rewards.
- However, sometimes the reward is either very big or very small, so the average reward is misleading.
- In *distributional reinforcement learning* we have several predictors, which react differently to *positive* and *negative* errors.

DISTRIBUTIONAL REINFORCEMENT LEARNING

- In *temporal reinforcement learning*, a network tries to predict (average) future rewards.
- However, sometimes the reward is either very big or very small, so the average reward is misleading.
- In *distributional reinforcement learning* we have several predictors, which react differently to *positive* and *negative* errors.
- Recently, similar distributions of *dopamine* cells was found in the brains of mice.


COMPUTER PROOFS
IN
MATHEMATICS

UNIVERSAL DEDUCER?

UNIVERSAL DEDUCER?

- A universal deducer is a program which, given a mathematical statement, either proves it is true or proves it is false.

UNIVERSAL DEDUCER?

- A universal deducer is a program which, given a mathematical statement, either proves it is true or proves it is false.
- By results of Church, Gödel, Turing, such a program is impossible.

UNIVERSAL DEDUCER?

- A universal deducer is a program which, given a mathematical statement, either proves it is true or proves it is false.
- By results of Church, Gödel, Turing, such a program is impossible.
- Practically, we can conclude that there is no best deducer, as any given proof can be found by some deducer but no deducer can find all proofs.

SOME COMPUTER-ASSISTED PROOFS

SOME COMPUTER-ASSISTED PROOFS

- **Four-colour problem:** Any map can be coloured with at most 4 colours.

SOME COMPUTER-ASSISTED PROOFS

- **Four-colour problem:** Any map can be coloured with at most 4 colours.
- **Kepler Conjecture:** The most efficient way to pack spheres is the hexagonal close packing.

SOME COMPUTER-ASSISTED PROOFS

- **Four-colour problem:** Any map can be coloured with at most 4 colours.
- **Kepler Conjecture:** The most efficient way to pack spheres is the hexagonal close packing.
- **Boolean Pythagorean triples problem:** is it possible to colour each of the positive integers either red or blue, so that no Pythagorean triple of integers a, b, c, satisfying $a^2 + b^2 = c^2$ are all the same color.

SOME COMPUTER-ASSISTED PROOFS

- **Four-colour problem:** Any map can be coloured with at most 4 colours.
- **Kepler Conjecture:** The most efficient way to pack spheres is the hexagonal close packing.
- **Boolean Pythagorean triples problem:** is it possible to colour each of the positive integers either red or blue, so that no Pythagorean triple of integers a, b, c, satisfying $a^2 + b^2 = c^2$ are all the same color.
- All these proofs are long (maybe unavoidable).

ROBBINS CONJECTURE

ROBBINS CONJECTURE

- Robbins conjecture was a conjectural characterization of Boolean algebras in terms of associativity and commutativity of $\vee$ and the Robbins equation $\neg(\neg(a \vee b) \vee \neg(a \vee \neg b)) = a$.

ROBBINS CONJECTURE

- Robbins conjecture was a conjectural characterization of Boolean algebras in terms of associativity and commutativity of $\vee$ and the Robbins equation
  $\neg(\neg(a \vee b) \vee \neg(a \vee \neg b)) = a$.
- This was conjectured in the 1930s, and finally proved in 1996 using the automated theorem prover **EQP**, which is a Resolution Theorem Prover with Paramodulation.

ROBBINS CONJECTURE

- Robbins conjecture was a conjectural characterization of Boolean algebras in terms of associativity and commutativity of $\vee$ and the Robbins equation
  $\neg(\neg(a \vee b) \vee \neg(a \vee \neg b)) = a$.
- This was conjectured in the 1930s, and finally proved in 1996 using the automated theorem prover **EQP**, which is a Resolution Theorem Prover with Paramodulation.
- So far, this seems to be the only major success of deductive theorem provers.

- A question on a blog of Terence Tao, asked to him by Apoorva Khare, was answered in PolyMath 14.

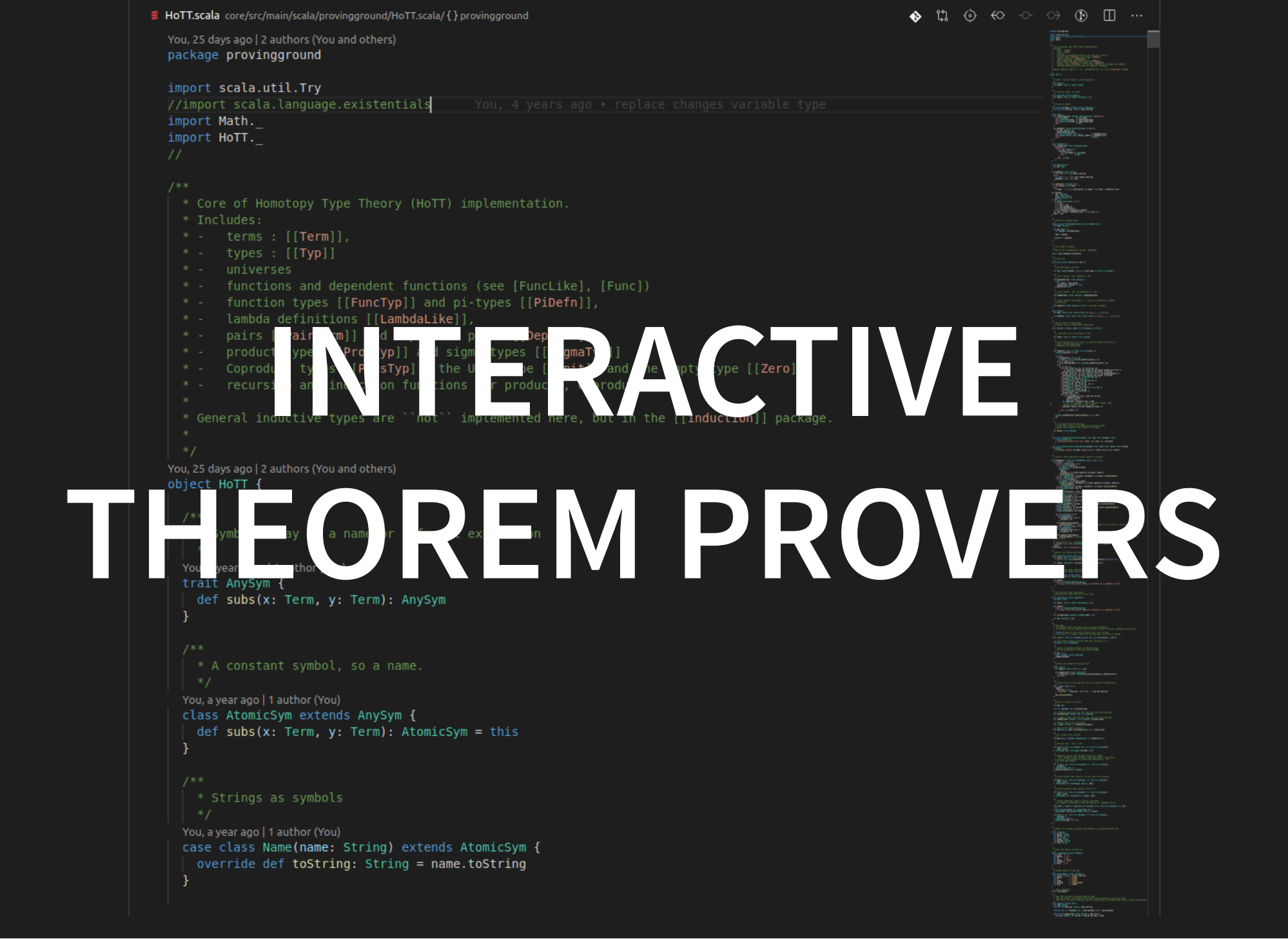
- A question on a blog of Terence Tao, asked to him by Apoorva Khare, was answered in PolyMath 14.
- A crucial step in the discovery was a computer generated but human readable proof I posted.

- A question on a blog of Terence Tao, asked to him by Apoorva Khare, was answered in PolyMath 14.
- A crucial step in the discovery was a computer generated but human readable proof I posted.

Gross proof time

Here is a computer generated proof of a bound on the length of the commutator $ab\bar{a}\bar{b}$ for a *linear* norm on the free group with the lengths of the generators bounded above by 1.

1. $|\bar{a}| \leq 1.0$
2. $|\bar{b}\bar{a}b| \leq 1.0$ using $|\bar{a}| \leq 1.0$
3. $|\bar{b}| \leq 1.0$
4. $|a\bar{b}\bar{a}| \leq 1.0$ using $|\bar{b}| \leq 1.0$
5. $|\bar{a}\bar{b}ab\bar{a}\bar{b}| \leq 2.0$ using $|\bar{a}\bar{b}a| \leq 1.0$ and $|b\bar{a}\bar{b}| \leq 1.0$
6. $|a| \leq 1.0$
7. $|ba\bar{b}| \leq 1.0$ using $|a| \leq 1.0$
8. $|b| \leq 1.0$
9. $|\bar{a}ba| \leq 1.0$ using $|b| \leq 1.0$
10. $|ab\bar{a}\bar{b}ab| \leq 2.0$ using $|ab\bar{a}| \leq 1.0$ and $|\bar{b}ab| \leq 1.0$
11. $|\bar{a}ab\bar{a}\bar{b}aba| \leq 2.0$ using $|ab\bar{a}\bar{b}ab| \leq 2.0$
12. $|b\bar{a}\bar{b}aab\bar{a}\bar{b}ab\bar{a}| \leq 3.0$ using $|b\bar{a}\bar{b}| \leq 1.0$ and $|aab\bar{a}\bar{b}ab\bar{a}| \leq 2.0$
13. $|\bar{a}\bar{b}ab\bar{a}\bar{b}aab\bar{a}\bar{b}ab\bar{a}| \leq 4.0$ using $|\bar{a}\bar{b}a| \leq 1.0$ and $|b\bar{a}\bar{b}aab\bar{a}\bar{b}ab\bar{a}| \leq 3.0$
14. $|\bar{b}\bar{a}\bar{b}aab\bar{a}\bar{b}ab\bar{a}\bar{b}| \leq 4.0$ using $|\bar{b}\bar{a}\bar{b}aab\bar{a}\bar{b}ab\bar{a}| \leq 3.0$


HoTT.scala core/src/main/scala/provinggroundHoTT.scala/{} provinggroundYou, 25 days ago | 2 authors (You and others)
package provinggroundimport scala.util.Try//import scala.language.existentialsYou, 4 years ago • replace changes variable typeimport Math._import HoTT._//
/**
* Core of Homotopy Type Theory (HoTT) implementation.
* Includes:
* - terms : [[Term]],
* - types : [[Typ]]
* - universes
* - functions and dependent functions (see [FuncLike], [Func])
* - function types [[FuncTyp]] and pi-types [[PiDefn]],
* - lambda definitions [[LambdaLike]],
* - pairs
* - product
* - Coprodu
* - recursi
*
* General inductive types are ``not`` implemented here, but in the [[Induction]] package.
*
*/
You, 25 days ago | 2 authors (You and others)
object HoTT {
/*
trait AnySym {
def subs(x: Term, y: Term): AnySym
}
/**
* A constant symbol, so a name.
*/
You, a year ago | 1 author (You)
class AtomicSym extends AnySym {
def subs(x: Term, y: Term): AtomicSym = this
}
/**
* Strings as symbols
*/
You, a year ago | 1 author (You)
case class Name(name: String) extends AtomicSym {
override def toString: String = name.toString
}
INTERACTIVE
THEOREM PROVERS

INTERACTIVE THEOREM PROVERS

INTERACTIVE THEOREM PROVERS

- These are software systems where proofs are obtained by human-machine collaboration.

INTERACTIVE THEOREM PROVERS

- These are software systems where proofs are obtained by human-machine collaboration.
- The computer both finds (parts of) proofs and checks correctness.

INTERACTIVE THEOREM PROVERS

- These are software systems where proofs are obtained by human-machine collaboration.
- The computer both finds (parts of) proofs and checks correctness.
- Some very large mathematical proofs have been verified by such systems.

INTERACTIVE THEOREM PROVERS

- These are software systems where proofs are obtained by human-machine collaboration.
- The computer both finds (parts of) proofs and checks correctness.
- Some very large mathematical proofs have been verified by such systems.
- Formal methods are also based on interactive provers.

WHO GUARDS THE GUARDS?

WHO GUARDS THE GUARDS?

- A computer verified proof is only as trustworthy as the system that verified the proof.

WHO GUARDS THE GUARDS?

- A computer verified proof is only as trustworthy as the system that verified the proof.
- Following the *de Bruijn* principle, proofs are *verified* by a small *trusted kernel*, which can be thoroughly checked.

WHO GUARDS THE GUARDS?

- A computer verified proof is only as trustworthy as the system that verified the proof.
- Following the *de Bruijn* principle, proofs are *verified* by a small *trusted kernel*, which can be thoroughly checked.
- For example, the *lean theorem prover* has three (small) proof checkers written in three languages.

RULES OF THE GAME?

RULES OF THE GAME?

- Around 100 years ago, after battling paradoxes, foundations of mathematics were built on Set Theory and First-order logic.

RULES OF THE GAME?

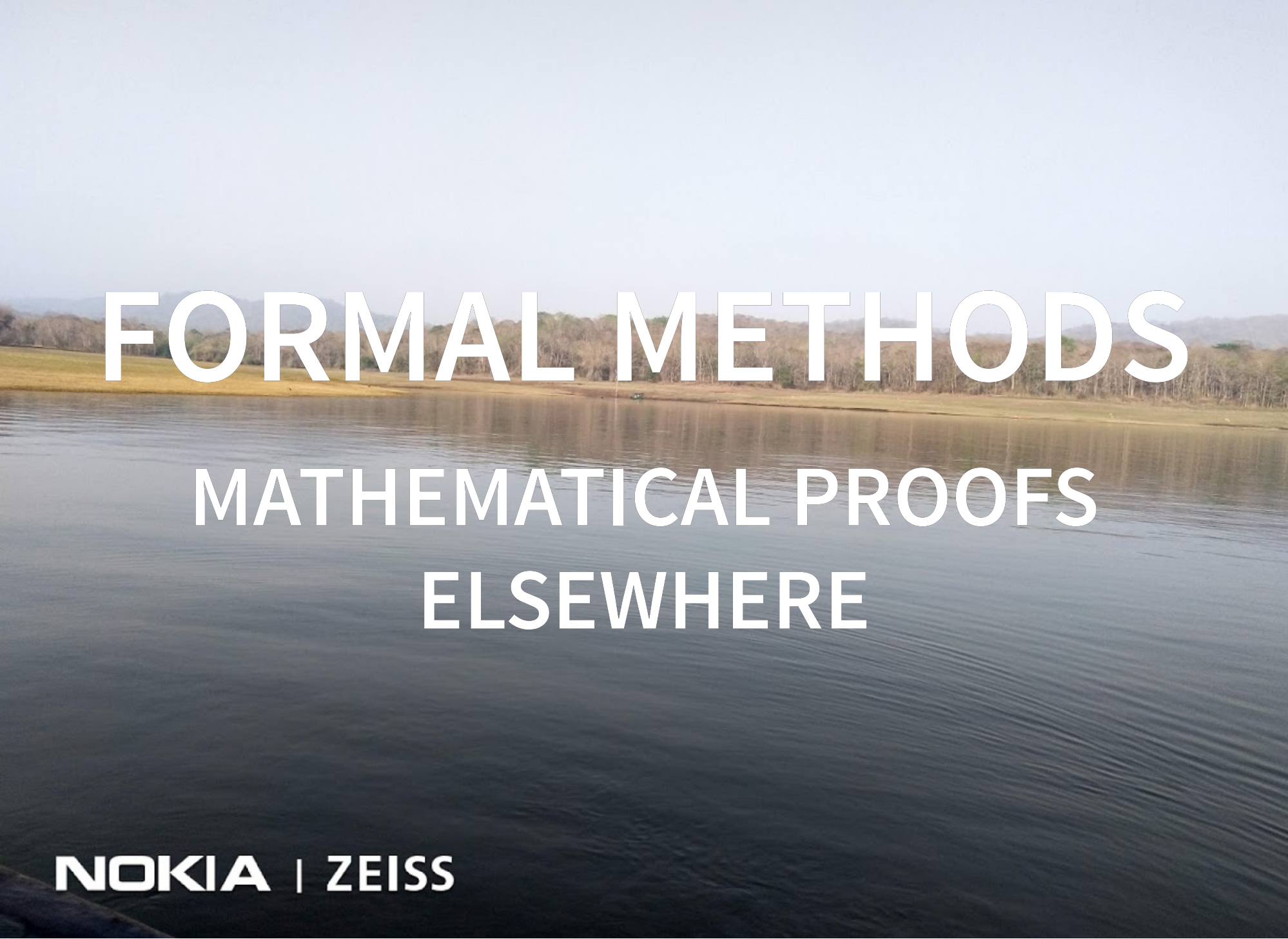
- Around 100 years ago, after battling paradoxes, foundations of mathematics were built on Set Theory and First-order logic.
- However, formal proofs in these foundation become long and opaque, and don't resemble real life mathematics.

RULES OF THE GAME?

- Around 100 years ago, after battling paradoxes, foundations of mathematics were built on Set Theory and First-order logic.
- However, formal proofs in these foundation become long and opaque, and don't resemble real life mathematics.
- Instead, interactive proof systems use richer foundations that have evolved over time.

RULES OF THE GAME?

- Around 100 years ago, after battling paradoxes, foundations of mathematics were built on Set Theory and First-order logic.
- However, formal proofs in these foundation become long and opaque, and don't resemble real life mathematics.
- Instead, interactive proof systems use richer foundations that have evolved over time.
- A dramatic advance in the new foundations came about with the recent discovery of connections with *topology*, a branch of mathematics.


FORMAL METHODS
MATHEMATICAL PROOFS
ELSEWHERE
NOKIA | ZEISS

FORMAL METHODS

FORMAL METHODS

- *Formal methods* mean

FORMAL METHODS

- *Formal methods* mean
  - we specify and describe software, hardware etc. in precise mathematical terms, and

FORMAL METHODS

- *Formal methods* mean
  - we specify and describe software, hardware etc. in precise mathematical terms, and
  - use *mathematical proof* to ensure correct behavior.

FORMAL METHODS

- *Formal methods* mean
  - we specify and describe software, hardware etc. in precise mathematical terms, and
  - use *mathematical proof* to ensure correct behavior.
- This gives a much greater certainty of correctness.

FORMAL METHODS

- *Formal methods* mean
  - we specify and describe software, hardware etc. in precise mathematical terms, and
  - use *mathematical proof* to ensure correct behavior.
- This gives a much greater certainty of correctness.
- However, proofs are much harder than tests.

FORMAL METHODS

- *Formal methods* mean
  - we specify and describe software, hardware etc. in precise mathematical terms, and
  - use *mathematical proof* to ensure correct behavior.
- This gives a much greater certainty of correctness.
- However, proofs are much harder than tests.
- Formal proofs use interactive theorem provers.

DO WE NEED *COMPLETELY CORRECT ALWAYS*?

DO WE NEED *COMPLETELY CORRECT ALWAYS*?

Pentium FDIV Bug

DO WE NEED *COMPLETELY CORRECT ALWAYS*?

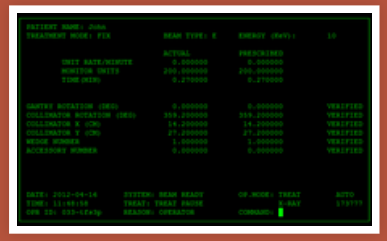
Pentium FDIV Bug

Fixing an error is very costly

DO WE NEED *COMPLETELY CORRECT ALWAYS*?

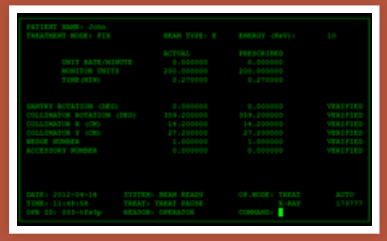
Pentium FDIV Bug

Fixing an error is very costly



Therac 25
radiation machine

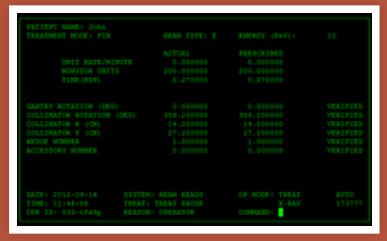
DO WE NEED *COMPLETELY CORRECT ALWAYS*?

	Pentium FDIV Bug	Fixing an error is very costly
	Therac 25 radiation machine	Safety critical

DO WE NEED *COMPLETELY CORRECT ALWAYS*?

Pentium FDIV Bug

Fixing an error is very costly

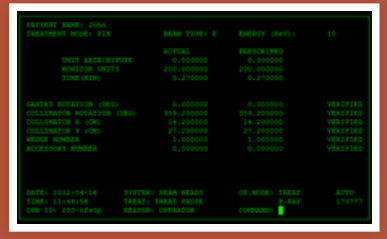


Therac 25 radiation machine

Safety critical

WhatsApp Pegasus attack

DO WE NEED *COMPLETELY CORRECT ALWAYS*?

	Pentium FDIV Bug	Fixing an error is very costly
	Therac 25 radiation machine	Safety critical
	WhatsApp Pegasus attack	A bug is a *vulnerability*

USERS OF FORMAL METHODS

USERS OF FORMAL METHODS

Intel Chips

USERS OF FORMAL METHODS

Intel Chips

Fixing an error is very costly

USERS OF FORMAL METHODS

Intel Chips

Fixing an error is very costly

Paris driverless metro

USERS OF FORMAL METHODS

	Intel Chips	Fixing an error is very costly
	Paris driverless metro	Safety critical

USERS OF FORMAL METHODS

Intel Chips

Fixing an error is very costly

Paris driverless metro

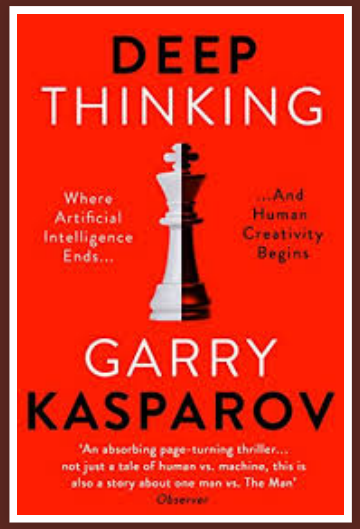
Safety critical

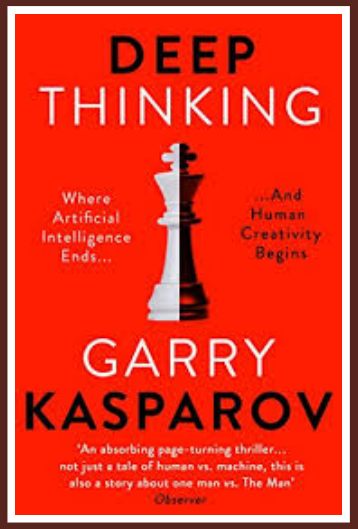
Scala dotty compiler

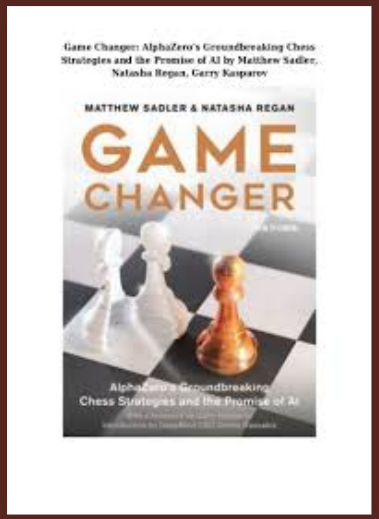
USERS OF FORMAL METHODS

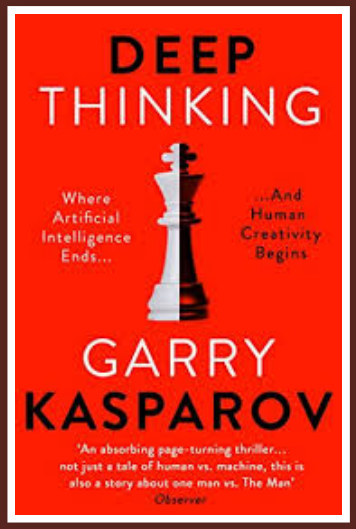
	Intel Chips	Fixing an error is very costly
	Paris driverless metro	Safety critical
	Scala dotty compiler	A bug is a vulnerability

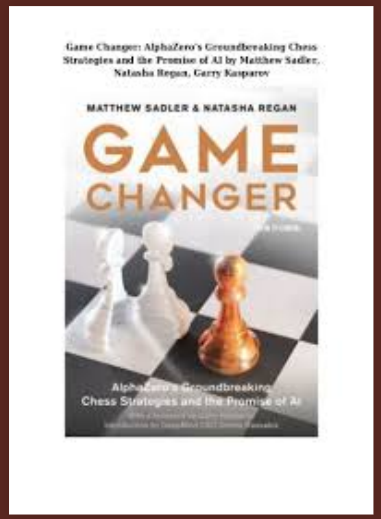

THE FUTURE OF
COMPUTER PROOFS?

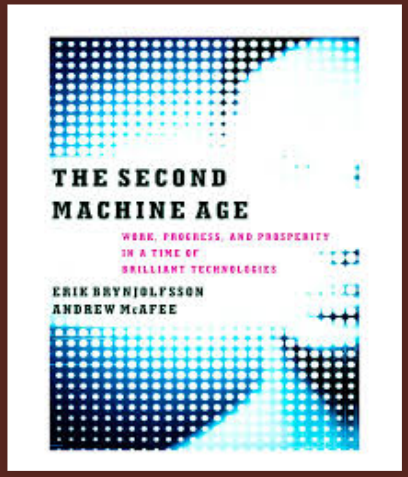

DEEP
THINKING
Where Artificial Intelligence Ends...
...And Human Creativity Begins
GARRY
KASPAROV
'An absorbing page-turning thriller... not just a tale of human vs. machine, this is also a story about one man vs. The Man'
Observer

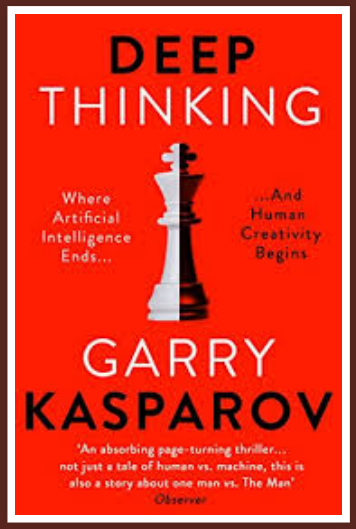

DEEP
THINKING
Where Artificial Intelligence Ends...
...And Human Creativity Begins
GARRY KASPAROV
'An absorbing page-turning thriller... not just a tale of human vs. machine, this is also a story about one man vs. The Man'
Observer

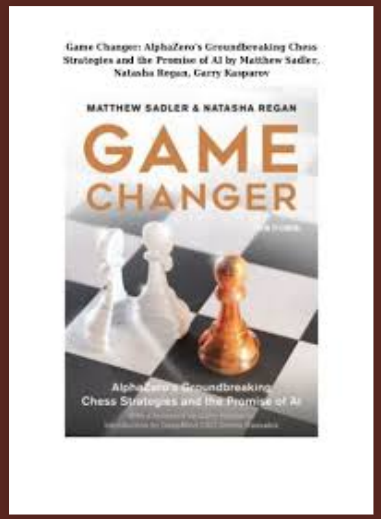

Game Changer: AlphaZero's Groundbreaking Chess Strategies and the Promise of AI by Matthew Sadler, Natasha Regan, Garry Kasparov
MATTHEW SADLER & NATASHA REGAN
GAME CHANGER

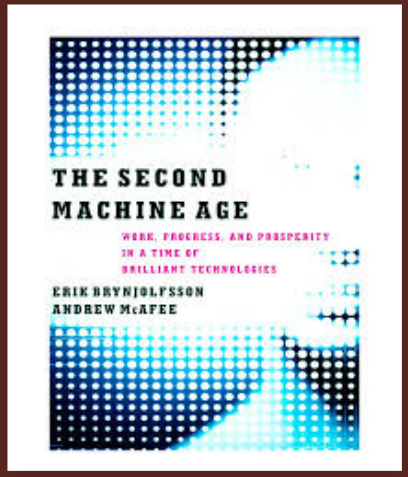

DEEP
THINKING
Where Artificial Intelligence Ends...
...And Human Creativity Begins
GARRY KASPAROV
'An absorbing page-turning thriller... not just a tale of human vs. machine, this is also a story about one man vs. The Man'
Observer

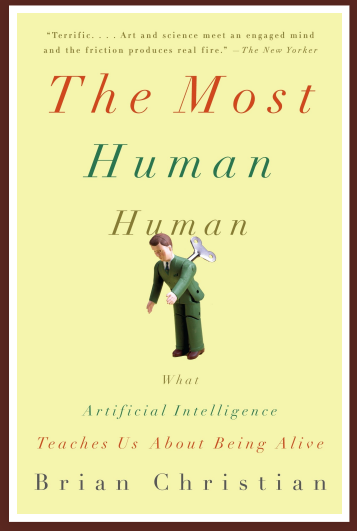

Game Changer: AlphaZero's Groundbreaking Chess Strategies and the Promise of AI by Matthew Sadler, Natasha Regan, Garry Kasparov
MATTHEW SADLER & NATASHA REGAN
GAME CHANGER
Chess Strategies and the Promise of AI


THE SECOND MACHINE AGE
WORK, PROGRESS, AND PROSPERITY IN A TIME OF BRILLIANT TECHNOLOGIES
ERIK BRYNJOLFSSON
ANDREW McAFEE


DEEP
THINKING
Where Artificial Intelligence Ends...
...And Human Creativity Begins
GARRY KASPAROV
'An absorbing page-turning thriller... not just a tale of human vs. machine, this is also a story about one man vs. The Man'
Observer


Game Changer: AlphaZero's Groundbreaking Chess Strategies and the Promise of AI by Matthew Sadler, Natasha Regan, Garry Kasparov
MATTHEW SADLER & NATASHA REGAN
GAME CHANGER


THE SECOND MACHINE AGE
WORK, PROGRESS, AND PROSPERITY IN A TIME OF BRILLIANT TECHNOLOGIES
ERIK BRYNJOLFSSON
ANDREW McAFEE


"Terrific. . . . Art and science meet an engaged mind and the friction produces real fire." —The New Yorker
The Most Human Human
What Artificial Intelligence Teaches Us About Being Alive
Brian Christian

- Artificial Intelligence can:

- Artificial Intelligence can:
  - Generate solutions that we can see are good.

- Artificial Intelligence can:
  - Generate solutions that we can see are good.
  - Judge value based on future rewards.

- Artificial Intelligence can:
  - Generate solutions that we can see are good.
  - Judge value based on future rewards.
  - Show originality.

- Artificial Intelligence can:
  - Generate solutions that we can see are good.
  - Judge value based on future rewards.
  - Show originality.
  - Learn things for which we depend on tacit knowledge.

- Artificial Intelligence can:
  - Generate solutions that we can see are good.
  - Judge value based on future rewards.
  - Show originality.
  - Learn things for which we depend on tacit knowledge.
  - Work with limited and/or unstructured data.

- Artificial Intelligence can:
  - Generate solutions that we can see are good.
  - Judge value based on future rewards.
  - Show originality.
  - Learn things for which we depend on tacit knowledge.
  - Work with limited and/or unstructured data.
  - Organize observations naturally and efficiently.

- Artificial Intelligence can:
  - Generate solutions that we can see are good.
  - Judge value based on future rewards.
  - Show originality.
  - Learn things for which we depend on tacit knowledge.
  - Work with limited and/or unstructured data.
  - Organize observations naturally and efficiently.
- **Automating mathematics?**

- Artificial Intelligence can:
  - Generate solutions that we can see are good.
  - Judge value based on future rewards.
  - Show originality.
  - Learn things for which we depend on tacit knowledge.
  - Work with limited and/or unstructured data.
  - Organize observations naturally and efficiently.
- **Automating mathematics?** It appears that there are clear approaches and **no evident barriers**.