1976 Turing Award: Rabin and Scott

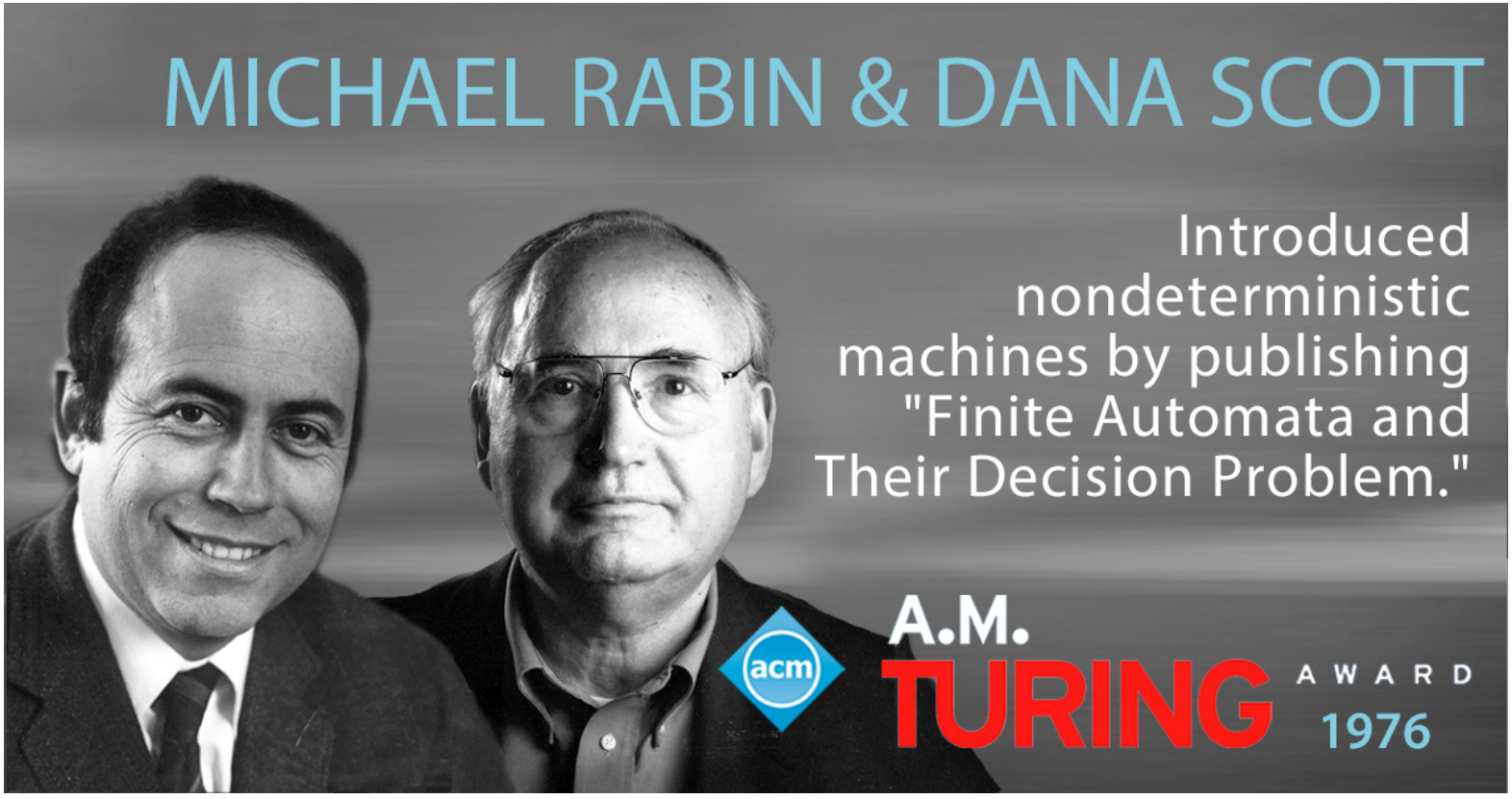
R. Ramanujam

The Institute of Mathematical Sciences, Chennai, India

jam@imsc.res.in

http://www.imsc.res.in/∼jam

Rabin and Scott



1976 A M Turing Award

Michael Rabin and Dana Scott

For their joint paper Finite Automata and Their Decision Problem which introduced the idea of nondeterministic machines, and which has proved to be an enormously valuable concept. Their (Scott and Rabin) classic paper has been a continuous source of inspiration for subsequent work in this field.

The collaboration

This is the only piece of joint work by Rabin and Scott.

- Both were PhD students of Alonzo Church in Princeton, worked on problems in mathematical logic.

The collaboration

This is the only piece of joint work by Rabin and Scott.

- Both were PhD students of Alonzo Church in Princeton, worked on problems in mathematical logic.
- Rabin graduated in 1957, Scott in 1958. They never worked together when they were in Princeton.

The collaboration

This is the only piece of joint work by Rabin and Scott.

- Both were PhD students of Alonzo Church in Princeton, worked on problems in mathematical logic.
- Rabin graduated in 1957, Scott in 1958. They never worked together when they were in Princeton.
- In 1959, IBM conducted a summer workshop for "young researchers". Rabin and Scott both participated in it, learnt about finite state automata, worked together and published the landmark paper.

The collaboration

This is the only piece of joint work by Rabin and Scott.

- Both were PhD students of Alonzo Church in Princeton, worked on problems in mathematical logic.
- Rabin graduated in 1957, Scott in 1958. They never worked together when they were in Princeton.
- In 1959, IBM conducted a summer workshop for "young researchers". Rabin and Scott both participated in it, learnt about finite state automata, worked together and published the landmark paper.
- Rabin went on to work on probabilistic automata as well as automata on infinite trees. Scott did a little automata theory, but moved on. He thought the emphasis on "finite state" was misplaced.

Automata theory in 1959

Began with the 1943 study of artificial neural networks by Walter Pitts and Warren McCulloch.

- Rabin and Scott moved away from neural networks and returned to a model similar to Turing machines, but which would be “read only”.

Automata theory in 1959

Began with the 1943 study of artificial neural networks by Walter Pitts and Warren McCulloch.

- Rabin and Scott moved away from neural networks and returned to a model similar to Turing machines, but which would be "read only".
- They wanted to study machine that could non-deterministically consider several potential alternatives at each step of the program.

Automata theory in 1959

Began with the 1943 study of artificial neural networks by Walter Pitts and Warren McCulloch.

- Rabin and Scott moved away from neural networks and returned to a model similar to Turing machines, but which would be "read only".
- They wanted to study machine that could non-deterministically consider several potential alternatives at each step of the program.
- In their view, the machine would read an input symbol, replicate itself, and then each machine would proceed with the computation along one of the alternatives.

Why bother?

Non-deterministic machines do not seem "realistic". So why did people consider them to be important ?

- Computational complexity theory is the study of what is possible to calculate given a specific set of resources such as time (number of steps), memory available, etc.

Why bother?

Non-deterministic machines do not seem "realistic". So why did people consider them to be important ?

- Computational complexity theory is the study of what is possible to calculate given a specific set of resources such as time (number of steps), memory available, etc.
- Rather than focusing on a single algorithm and analyzing its requirements, nondeterminism makes it possible to consider several alternatives simultaneously, "guess" one and verify it.

Why bother?

Non-deterministic machines do not seem "realistic". So why did people consider them to be important ?

- Computational complexity theory is the study of what is possible to calculate given a specific set of resources such as time (number of steps), memory available, etc.
- Rather than focusing on a single algorithm and analyzing its requirements, nondeterminism makes it possible to consider several alternatives simultaneously, "guess" one and verify it.
- For instance, there might be different ways to carry out a search, and we might want to proceed without committing to a specific strategy.

Why bother?

Non-deterministic machines do not seem "realistic". So why did people consider them to be important ?

- Computational complexity theory is the study of what is possible to calculate given a specific set of resources such as time (number of steps), memory available, etc.
- Rather than focusing on a single algorithm and analyzing its requirements, nondeterminism makes it possible to consider several alternatives simultaneously, "guess" one and verify it.
- For instance, there might be different ways to carry out a search, and we might want to proceed without committing to a specific strategy.
- Nondeterministic machines provide for precisely such abstractions.

Search

This is the Google era, search is a big part of life.

- Nondeterminism is best understood as a primitive instruction: Search for x.

Search

This is the Google era, search is a big part of life.

- Nondeterminism is best understood as a primitive instruction: Search for x.
- Important conceptual role: expose the existence of the search step explicitly.

Search

This is the Google era, search is a big part of life.

- Nondeterminism is best understood as a primitive instruction: Search for x.
- Important conceptual role: expose the existence of the search step explicitly.
- Then, whether the search step can be eliminated, becomes important.

Search

This is the Google era, search is a big part of life.

- Nondeterminism is best understood as a primitive instruction: Search for x.
- Important conceptual role: expose the existence of the search step explicitly.
- Then, whether the search step can be eliminated, becomes important.
- Rabin and Scott: it can indeed be eliminated for finite state automata, but that there is an exponential price to be paid.

Sources

This is a very wikipedia level talk, all information from Internet.

- The Turing award lectures, as articles in CACM.

Sources

This is a very wikipedia level talk, all information from Internet.

- The Turing award lectures, as articles in CACM.
- ACM site on Turing awards and interviews with the awardees.

Sources

This is a very wikipedia level talk, all information from Internet.

- The Turing award lectures, as articles in CACM.
- ACM site on Turing awards and interviews with the awardees.
- Solomon Feferman and Anita Burdman Feferman (2004). Alfred Tarski: life and logic. Cambridge University Press.

Michael Rabin

A short bio

Born on September 1, 1931, in Breslau, Germany (now Wrocĺaw, Poland).

- The family moved to Palestine in 1935.

A short bio

Born on September 1, 1931, in Breslau, Germany (now Wrocĺaw, Poland).

- The family moved to Palestine in 1935.
- Taught mathematics by Elisha Netanyahu, who was a major influence.

A short bio

Born on September 1, 1931, in Breslau, Germany (now Wrocĺaw, Poland).

- The family moved to Palestine in 1935.
- Taught mathematics by Elisha Netanyahu, who was a major influence.
- While doing military service, learnt set theory and wrote to Abraham Fraenkel, who pulled him out to the University of Jerusalem.

A short bio

Born on September 1, 1931, in Breslau, Germany (now Wrocĺaw, Poland).

- The family moved to Palestine in 1935.
- Taught mathematics by Elisha Netanyahu, who was a major influence.
- While doing military service, learnt set theory and wrote to Abraham Fraenkel, who pulled him out to the University of Jerusalem.
- Master's thesis solved an open problem posed by the Emmy Noether, this got him admission for PhD at Princeton to work with Church (thesis on unsolvability of group theoretical problems).

A puzzle

In 1960, Rabin returned to the IBM summer workshop. This time he met John McCarthy who proposed the spies and guards puzzle to him.

- Spies have passwords that allow them to pass from enemy territory to their own.

A puzzle

In 1960, Rabin returned to the IBM summer workshop. This time he met John McCarthy who proposed the spies and guards puzzle to him.

- Spies have passwords that allow them to pass from enemy territory to their own.
- Guards need to know the passwords to allow them in, but cannot be trusted to keep the passwords secret.

A puzzle

In 1960, Rabin returned to the IBM summer workshop. This time he met John McCarthy who proposed the spies and guards puzzle to him.

- Spies have passwords that allow them to pass from enemy territory to their own.
- Guards need to know the passwords to allow them in, but cannot be trusted to keep the passwords secret.
- Some method has to be found to verify that even if the enemy gains knowledge of the password, the spies can safely return but the enemy infiltrators are kept out.

von Neumann's solution

The middle square method, proposed by John von Neumann as a way of generating random numbers.

- Each spy is given a 100 digit number x.

von Neumann's solution

The middle square method, proposed by John von Neumann as a way of generating random numbers.

- Each spy is given a 100 digit number x.
- The guards are given another 100 digit number obtained by taking the middle digits from x^2.

von Neumann's solution

The middle square method, proposed by John von Neumann as a way of generating random numbers.

- Each spy is given a 100 digit number x.
- The guards are given another 100 digit number obtained by taking the middle digits from x^2.
- When returning, the spy gives the guard x.

von Neumann's solution

The middle square method, proposed by John von Neumann as a way of generating random numbers.

- Each spy is given a 100 digit number x.
- The guards are given another 100 digit number obtained by taking the middle digits from x^2.
- When returning, the spy gives the guard x.
- The guard then computes x^2 and compares the middle digits to the number he possesses as a pass code.

von Neumann's solution

The middle square method, proposed by John von Neumann as a way of generating random numbers.

- Each spy is given a 100 digit number x.
- The guards are given another 100 digit number obtained by taking the middle digits from x^2.
- When returning, the spy gives the guard x.
- The guard then computes x^2 and compares the middle digits to the number he possesses as a pass code.
- Even if the guard passes his number to an enemy, it is very difficult for the enemy to determine what initial number resulted in those 100 middle digits of x^2.

Complexity and cryptography

Rabin was greatly intrigued by the middle square method.

- It is not impossible to find x knowing the middle digits of x^2, but certainly difficult. But difficult for whom ?

Complexity and cryptography

Rabin was greatly intrigued by the middle square method.

- It is not impossible to find x knowing the middle digits of x^2, but certainly difficult. But difficult for whom ?
- Rabin wanted to quantify such "difficulty to invert" for any algorithm.

Complexity and cryptography

Rabin was greatly intrigued by the middle square method.

- It is not impossible to find x knowing the middle digits of x^2, but certainly difficult. But difficult for whom ?
- Rabin wanted to quantify such "difficulty to invert" for any algorithm.
- Paper: "Degree of Difficulty of Computing a Function and a Partial Ordering of Recursive Sets".

Complexity and cryptography

Rabin was greatly intrigued by the middle square method.

- It is not impossible to find x knowing the middle digits of x^2, but certainly difficult. But difficult for whom ?
- Rabin wanted to quantify such "difficulty to invert" for any algorithm.
- Paper: "Degree of Difficulty of Computing a Function and a Partial Ordering of Recursive Sets".
- A groundbreaking paper, which was the starting point for later advances in the study of complexity in relation to cryptography.

Another visit

After several years at the University of Jerusalem, Rabin visited MIT in 1975 and met Gary Miller.

- Miller had devised an algorithm to check if a given number was prime, and it relied on an unproven hypothesis due to Riemann.

Another visit

After several years at the University of Jerusalem, Rabin visited MIT in 1975 and met Gary Miller.

- Miller had devised an algorithm to check if a given number was prime, and it relied on an unproven hypothesis due to Riemann.
- Rabin had earlier worked on probabilistic automata, for which a random number is used to determine which transition to take from each state.

Another visit

After several years at the University of Jerusalem, Rabin visited MIT in 1975 and met Gary Miller.

- Miller had devised an algorithm to check if a given number was prime, and it relied on an unproven hypothesis due to Riemann.
- Rabin had earlier worked on probabilistic automata, for which a random number is used to determine which transition to take from each state.
- He used this concept to develop a primality testing algorithm.

Another visit

After several years at the University of Jerusalem, Rabin visited MIT in 1975 and met Gary Miller.

- Miller had devised an algorithm to check if a given number was prime, and it relied on an unproven hypothesis due to Riemann.
- Rabin had earlier worked on probabilistic automata, for which a random number is used to determine which transition to take from each state.
- He used this concept to develop a primality testing algorithm.
- Adding randomness to algorithms was to be a theme for much of Rabin's later work, especially in distributed computing, and cryptography.

Hostility

Mathematicians apparently did not like such "trading certainty for speed".

- Rabin speaks of hostile referees: "if he had attached a pen to his dog's tail he would have proved Riemann's hypothesis with positive probability" !

Hostility

Mathematicians apparently did not like such "trading certainty for speed".

- Rabin speaks of hostile referees: "if he had attached a pen to his dog's tail he would have proved Riemann's hypothesis with positive probability" !
- Rabin: "The probability of an error is smaller than the probability that none of us is awake and we are all dreaming this."

Hostility

Mathematicians apparently did not like such "trading certainty for speed".

- Rabin speaks of hostile referees: "if he had attached a pen to his dog's tail he would have proved Riemann's hypothesis with positive probability" !
- Rabin: "The probability of an error is smaller than the probability that none of us is awake and we are all dreaming this."
- Rabin in later years: "I must admit that after many years of work in this area, the efficacy of randomness for so many algorithmic problems is absolutely mysterious to me. It is efficient, it works; but why and how is absolutely mysterious."

The dining philosophers

Paradigm problem due to Edsger Dijkstra, to illustrate the difficulty of coordination in distributed systems.

- Gabbay, Pnueli, Shelah and Stavi (1980) showed that there is no symmetric, deterministic, fair and distributed solution to the DPP.

The dining philosophers

Paradigm problem due to Edsger Dijkstra, to illustrate the difficulty of coordination in distributed systems.

- Gabbay, Pnueli, Shelah and Stavi (1980) showed that there is no symmetric, deterministic, fair and distributed solution to the DPP.
- Lehmann and Rabin (1981) gave an extremely simple and beautiful randomized algorithm for DPP that is symmetric, fair and distributed.

The dining philosophers

Paradigm problem due to Edsger Dijkstra, to illustrate the difficulty of coordination in distributed systems.

- Gabbay, Pnueli, Shelah and Stavi (1980) showed that there is no symmetric, deterministic, fair and distributed solution to the DPP.
- Lehmann and Rabin (1981) gave an extremely simple and beautiful randomized algorithm for DPP that is symmetric, fair and distributed.
- In the process, they gave a paradigm solution to achieve fairness and breaking symmetry in distributed algorithms.

Contributions to logic

Michael Rabin made many fundamental contributions to logic. Perhaps the most significant of them for computer science is the Rabin Tree Theorem.

- Gödel showed that the first order theory of arithmetic with addition and multiplication is decidable.

Contributions to logic

Michael Rabin made many fundamental contributions to logic. Perhaps the most significant of them for computer science is the Rabin Tree Theorem.

- Gödel showed that the first order theory of arithmetic with addition and multiplication is decidable.
- Büchi (1960) showed that the theory of natural numbers with order is decidable, and he used automata to solve the problem.

Contributions to logic

Michael Rabin made many fundamental contributions to logic. Perhaps the most significant of them for computer science is the Rabin Tree Theorem.

- Gödel showed that the first order theory of arithmetic with addition and multiplication is decidable.
- Büchi (1960) showed that the theory of natural numbers with order is decidable, and he used automata to solve the problem.
- Rabin (1969) extended the techniques to show that the theory of the infinite binary tree is decidable.

Contributions to logic

Michael Rabin made many fundamental contributions to logic. Perhaps the most significant of them for computer science is the Rabin Tree Theorem.

- Gödel showed that the first order theory of arithmetic with addition and multiplication is decidable.
- Büchi (1960) showed that the theory of natural numbers with order is decidable, and he used automata to solve the problem.
- Rabin (1969) extended the techniques to show that the theory of the infinite binary tree is decidable.
- The consensus today is that, in a deep sense, it represents the "boundary" of decidability in such theories.

The tree theorem

Importance principally because graphs seem to be "hard", trees seem to be "easy" and Rabin showed a way to identify "tree interpretations" and get algorithms.

- Rabin: "I consider this to be the most difficult research I have ever done."

The tree theorem

Importance principally because graphs seem to be "hard", trees seem to be "easy" and Rabin showed a way to identify "tree interpretations" and get algorithms.

- Rabin: "I consider this to be the most difficult research I have ever done."
- Wolfgang Thomas: "a fundamental decidability result that appears in hundreds of applications in theoretical computer science."

A quote

Michael Rabin on research and teaching.

> *There is this misconception that there is a conflict and maybe even a contradiction between great teaching and being able to do great science. I think this is completely incorrect, and that wonderful teaching flows from a deep understanding of the subject matter.*

Dana Scott

A short bio

Born on October 11, 1932, in Berkeley, California, USA.

- Tarski was a major influence during undergraduate days.

A short bio

Born on October 11, 1932, in Berkeley, California, USA.

- Tarski was a major influence during undergraduate days.
- Learnt $\lambda-$calculus of Church: "which, literally, gave me nightmares at first".

A short bio

Born on October 11, 1932, in Berkeley, California, USA.

- Tarski was a major influence during undergraduate days.
- Learnt $\lambda-$calculus of Church: "which, literally, gave me nightmares at first".
- He worked on theories of measurement and on the continuum hypothesis at Princeton.

A short bio

Born on October 11, 1932, in Berkeley, California, USA.

- Tarski was a major influence during undergraduate days.
- Learnt $\lambda-$calculus of Church: "which, literally, gave me nightmares at first".
- He worked on theories of measurement and on the continuum hypothesis at Princeton.
- Met Christopher Strachey during his stay in Amsterdam; started work on semantics of programming languages, moved to Oxford.

A Mathematical ToC

It was Roger Penrose who drew Strachey's attention to the $\lambda-$calculus, as a functional abstraction of programs.

- Scott asked, what are the spaces that computable functions live in ?

A Mathematical ToC

It was Roger Penrose who drew Strachey's attention to the $\lambda-$calculus, as a functional abstraction of programs.

- Scott asked, what are the spaces that computable functions live in ?
- The trouble with programs is that there is no difference between programs and the data they operate on.

A Mathematical ToC

It was Roger Penrose who drew Strachey's attention to the $\lambda-$calculus, as a functional abstraction of programs.

- Scott asked, what are the spaces that computable functions live in ?
- The trouble with programs is that there is no difference between programs and the data they operate on.
- Functions can take functions as arguments; recursive functions can "call" themselves.

A Mathematical ToC

It was Roger Penrose who drew Strachey's attention to the $\lambda-$calculus, as a functional abstraction of programs.

- Scott asked, what are the spaces that computable functions live in ?
- The trouble with programs is that there is no difference between programs and the data they operate on.
- Functions can take functions as arguments; recursive functions can "call" themselves.
- The 'operational' meaning is given by a machine; what is the 'denotational' (mathematical) meaning ?

Domain theory

Scott's theorem: The computable functions on a space D are exactly the continuous functions over D (in a suitable topology).

- Scott showed a solution to the equation $F \cong [F \to F]$, where the right hand side consists of continuous functions from F to F.

Domain theory

Scott's theorem: The computable functions on a space D are exactly the continuous functions over D (in a suitable topology).

- Scott showed a solution to the equation $F \approxeq [F \to F]$, where the right hand side consists of continuous functions from F to F.
- This is a *tour de force* and lays the foundation for giving a formal semantics to full programming languages with rich features.

Domain theory

Scott's theorem: The computable functions on a space D are exactly the continuous functions over D (in a suitable topology).

- Scott showed a solution to the equation $F \approxeq [F \to F]$, where the right hand side consists of continuous functions from F to F.
- This is a *tour de force* and lays the foundation for giving a formal semantics to full programming languages with rich features.
- In the process, Scott gave us notions of approximation, limits, projections and other tools similar to that of mathematical analysis.

Domain theory

Scott's theorem: The computable functions on a space D are exactly the continuous functions over D (in a suitable topology).

- Scott showed a solution to the equation $F \cong [F \to F]$, where the right hand side consists of continuous functions from F to F.
- This is a *tour de force* and lays the foundation for giving a formal semantics to full programming languages with rich features.
- In the process, Scott gave us notions of approximation, limits, projections and other tools similar to that of mathematical analysis.
- Scott's dream: program analysis should achieve the sophistication of calculus.

A quick intro

Consider a program outputting a stream of numbers.

- Initially we know nothing about what it does. Denote this by $\perp$.

A quick intro

Consider a program outputting a stream of numbers.

- Initially we know nothing about what it does. Denote this by $\perp$.
- After a while it outputs number n_1. We have an information ordering $\perp \preceq n_1$.

A quick intro

Consider a program outputting a stream of numbers.

- Initially we know nothing about what it does. Denote this by $\perp$.
- After a while it outputs number n_1. We have an information ordering $\perp \preceq n_1$.
- Proceeding further, we may get $\perp \preceq n_1 \preceq n_1 n_2 \preceq \ldots n_1 n_2 \ldots n_k \preceq < n_1 n_2 \ldots n_k >$ when it terminates.

A quick intro

Consider a program outputting a stream of numbers.

- Initially we know nothing about what it does. Denote this by $\perp$.
- After a while it outputs number n_1. We have an information ordering $\perp \preceq n_1$.
- Proceeding further, we may get $\perp \preceq n_1 \preceq n_1 n_2 \preceq \ldots n_1 n_2 \ldots n_k \preceq < n_1 n_2 \ldots n_k >$ when it terminates.
- What if it never stops and produces an infinite stream ? Then it is a complete output as well. Infinite streams that are least upper bounds of finite streams are the limits in this information ordering.

Continuity

The output of a program is determined only by a finite part of the input.

- Suppose the input to a program P is a stream approximated by: $v_0 \preceq v_1 \preceq v_2 \preceq \ldots$.

Continuity

The output of a program is determined only by a finite part of the input.

- Suppose the input to a program P is a stream approximated by: $v_0 \preceq v_1 \preceq v_2 \preceq \ldots$.
- P can only produce successive approximations of output $f(v_0) \preceq f(v_1) \preceq f(v_2) \preceq \ldots$.

Continuity

The output of a program is determined only by a finite part of the input.

- Suppose the input to a program P is a stream approximated by: $v_0 \preceq v_1 \preceq v_2 \preceq \ldots$.
- P can only produce successive approximations of output $f(v_0) \preceq f(v_1) \preceq f(v_2) \preceq \ldots$.
- The function computed by P should map the limit of the v_i's to the limit of the $f(v_i)$'s. That is, it should be continuous.

Continuity

The output of a program is determined only by a finite part of the input.

- Suppose the input to a program P is a stream approximated by: $v_0 \preceq v_1 \preceq v_2 \preceq \ldots$.
- P can only produce successive approximations of output $f(v_0) \preceq f(v_1) \preceq f(v_2) \preceq \ldots$.
- The function computed by P should map the limit of the v_i's to the limit of the $f(v_i)$'s. That is, it should be continuous.
- Intuitively, discontinous functions need infinite amount of information from the input and hence are not computable.

Contributions to Logic

Dana Scott is one of the most influential logicians in the last half-century.

- 1960's: Infinitary logic, boolean valued models of set theory, Scott's trick !

Contributions to Logic

Dana Scott is one of the most influential logicians in the last half-century.

- 1960's: Infinitary logic, boolean valued models of set theory, Scott's trick !
- 1970's: Montague-Scott semantics of intensional logic, Scott-Lemmon filtrations for modal logic, Scott normal form for 2-variable logic.

Contributions to Logic

Dana Scott is one of the most influential logicians in the last half-century.

- 1960's: Infinitary logic, boolean valued models of set theory, Scott's trick !
- 1970's: Montague-Scott semantics of intensional logic, Scott-Lemmon filtrations for modal logic, Scott normal form for 2-variable logic.
- 1980's: Continuous lattices; domain theory.

Contributions to Logic

Dana Scott is one of the most influential logicians in the last half-century.

- 1960's: Infinitary logic, boolean valued models of set theory, Scott's trick !
- 1970's: Montague-Scott semantics of intensional logic, Scott-Lemmon filtrations for modal logic, Scott normal form for 2-variable logic.
- 1980's: Continuous lattices; domain theory.
- 1990's: Equilogical spaces.

Pentominoes!

Little known: Dana Scott's contributions to recreational mathematics.

- A polygon in the plane made of 5 equal-sized squares connected edge-to-edge. When rotations and reflections are not considered to be distinct shapes, there are 12 different free pentominoes.

Pentominoes!

Little known: Dana Scott's contributions to recreational mathematics.

- A polygon in the plane made of 5 equal-sized squares connected edge-to-edge. When rotations and reflections are not considered to be distinct shapes, there are 12 different free pentominoes.
- A standard pentomino puzzle is to tile a rectangular box with the pentominoes: cover it without overlap and without gaps. Question: how many different solutions can we have ?

Pentominoes!

Little known: Dana Scott's contributions to recreational mathematics.

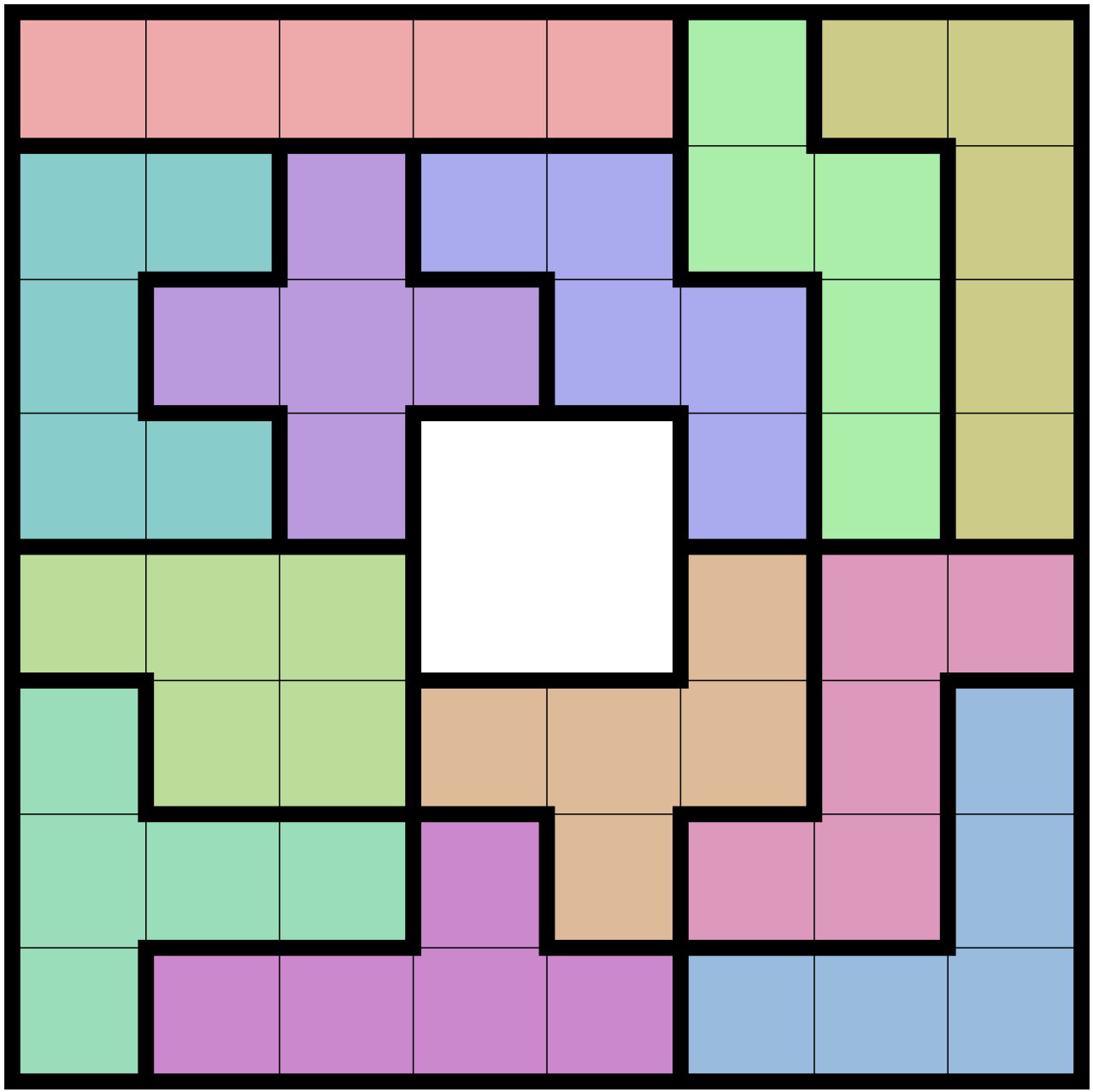
- A polygon in the plane made of 5 equal-sized squares connected edge-to-edge. When rotations and reflections are not considered to be distinct shapes, there are 12 different free pentominoes.
- A standard pentomino puzzle is to tile a rectangular box with the pentominoes: cover it without overlap and without gaps. Question: how many different solutions can we have ?
- What about an 8×8 rectangle with a 2×2 hole in the centre ?

Pentominoes!

Little known: Dana Scott's contributions to recreational mathematics.

- A polygon in the plane made of 5 equal-sized squares connected edge-to-edge. When rotations and reflections are not considered to be distinct shapes, there are 12 different free pentominoes.
- A standard pentomino puzzle is to tile a rectangular box with the pentominoes: cover it without overlap and without gaps. Question: how many different solutions can we have ?
- What about an 8×8 rectangle with a 2×2 hole in the centre ?
- Dana Scott (1958) showed that there are exactly 65 solutions. Scott's algorithm was one of the first applications of a backtracking computer program.

Pentomino



A quote

Scott is known to be liberal with advice for researchers.

- Learn as much as you can while you are young, since life becomes too busy later.

A quote

Scott is known to be liberal with advice for researchers.

- Learn as much as you can while you are young, since life becomes too busy later.
- Try to regard mathematics as an experimental science.

A quote

Scott is known to be liberal with advice for researchers.

- Learn as much as you can while you are young, since life becomes too busy later.
- Try to regard mathematics as an experimental science.
- My favourite:

> *For God's sake, somebody solve the P-NP question ! We need to get on with all the serious pending work . . .*

Vienna, Summer of Logic, 2014.

Discussion time

Thank you.
Questions, comments, suggestions welcome; also, please write to jam@imsc.res.in.