NKS 2004

CA and Intermediate Degrees

Klaus Sutner
Carnegie Mellon University
www.cs.cmu.edu/~sutner
sutner@cs.cmu.edu

Overview

- Classical Complexity and Post's Problem
- Intermediate Turing Degrees
- Orbits of Cellular Automata
- The Reversible Case
- Some Questions

Entscheidungsproblem

The Entscheidungsproblem is solved when one knows a procedure by which one can decide in a finite number of operations whether a given logical expression is generally valid or is satisfiable. The solution of the Entscheidungsproblem is of fundamental importance for the theory of all fields, the theorems of which are at all capable of logical development from finitely many axioms.

D. Hilbert, W. Ackermann
Grundzüge der theoretischen Logik, 1928

In modern terminology: find a *decision algorithm* for statements of mathematics (or at least arithmetic, group theory, topology, . . .).

Alas . . .

Theorem. *(Church, Turing)*

Mathematics is undecidable.

In fact, even rather small fragments of mathematics turn out to undecidable. For example,

Theorem. *Novikov, Boone, Adyan, Rabin, . . .*

The Entscheidungsproblem is not solvable for group theory

Theorem. *Y. Matiyasevic, 1970*

It is undecidable whether a Diophantine equation has an integer solution.

No (Computable) Bounds

As a mild example, try to find a solution in positive integers for the quadratic equation

$$x^2 - 991y^2 - 1 = 0.$$

The smallest positive solution here is

$$\begin{aligned} x &= 379516400906811930638014896080 \\ y &= 12055735790331359447442538767 \end{aligned}$$

Much harder is

$$313(x^3 + y^3) = z^3$$

Semi-Decidability

A *semi-algorithm* for A is a procedure which, given any input x:

- halts after finitely many steps if $x \in A$, and
- runs forever otherwise.

A is then *semi-decidable*.

A priori it is not clear that there are semi-decidable sets that fail to be decidable (though undecidable sets must exist by a cardinality argument).

Theorem. *Turing*

The Halting Set K is semi-decidable but fails to be decidable.

Semi-Decidable Sets

The decidable sets are exactly the complemented elements of the lattice of semi-decidable sets.

Lemma. *A is decidable if, and only if, both A and its complement are semi-decidable.*

There are countless examples of semi-decidable sets in mathematics and CS.

Lemma. *Diophantine Equations are semi-decidable (unbounded search).*

Lemma. *The theorems of any axiomatizable theory are semi-decidable.*

Comparing Difficulty

Since there are at least two types of semi-decidable sets, how do we compare the complexity of two (semi-decidable) sets A and B?

The basic idea is simple:

Problem A is easier than B if we could use B as an *oracle* (a subroutine that answers membership questions about B) to construct an algorithm for A.

A is *Turing-reducible* to B ($A \leq_T B$) if there is a decision algorithm for A given an oracle for B.

A and B are *Turing-equivalent* ($A \equiv_T B$) if $A \leq_T B$ and $B \leq_T A$.

Completeness

So we have a natural pre-order on the semi-decidable sets.

$$0 = \text{ decidable}$$
$$1 = \text{ ????}$$

Is there a semi-decidable set that contains as much information as any semi-decidable set?

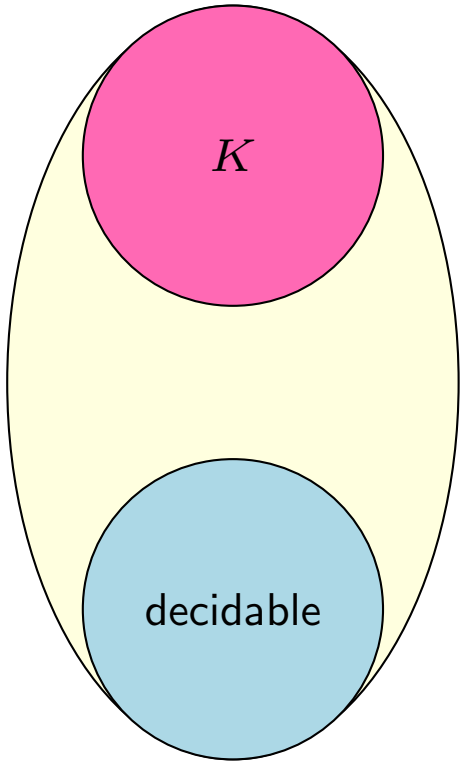
More formally, C is *complete* semi-decidable if

$$\forall\, A \text{ semi-decidable } (A \leq_T C)$$

Note: C itself must be semi-decidable, otherwise this is easy (take the disjoint union of all semi-decidable sets).

The World of Semi-Decidability

Theorem. *The Halting Set K is complete semi-decidable.*



semi-decidable

Universality

The key ingredient for complete problems in *universality*: there are Turing machines that are capable of simulating every possible Turing machines.

Fairly unexiting today, every PC is a universal computer in this sense (disregarding resource bounds).

Real proof requires quite a bit of coding machinery.

Universality

Many simple systems are known that can perform universal computations and give rise to complete problems.

- Alonzo Church, λ calculus, 1931.
- Emil Post, production systems, 1943.
- Newell, Simon, and Shaw, IPL-1, 1956.
- John McCarthy, LISP, 1959.
- Davis, Putnam and Robinson, exponential Diophantine equations, 1961.
- Marvin Minsky, two-counter machine, 1961.
- John Conway, "Game of Life", 1970.
- Edwin Bank, 4-state CA, 1971.
- Wolfram, Cook, ECA rule 110.

An Empirical 0/1-Law

Any *natural* semi-decidable set is either decidable or Turing-equivalent to the Halting problem. In other words, the yellow area appears to be empty.

There are analogues of this in computational complexity theory: For example, every (more or less) natural set in $\mathbb{NP}$ is either in $\mathbb{P}$ or $\mathbb{NP}$-complete.

However, unlike with unconstrained computability, it seems extremely difficult to prove that $\mathbb{P} \neq \mathbb{NP}$.

It might be the case that $\mathbb{P} = \mathbb{NP}$ so the distinction disappears.

Post's Problem

Are there any intermediate semi-decidable sets?

I.e., is there a semi-decidable set A such that

$$\emptyset <_T A <_T K$$

Theorem. *Friedberg, Muchnik 1956/7*

There are intermediate semi-decidable sets.

Construction quite complicated and very different from previously known methods, so-called *priority argument*.

Tip of the Iceberg

Theorem. *Sack's Density Theorem*

Given semi-decidable sets $A <_T B$ there is another semi-decidable set C such that $A <_T C <_T B$.

The Turing degrees of all semi-decidable sets form a semi-lattice $\mathcal{D}$ (meet is partial). Has a very rich and fairly well-understood structure.

Theorem. *Harrington, Shelah, Slaman*

The Entscheidungsproblem for $\mathcal{D}$ is highly undecidable (it has degree $\emptyset^{(\omega)}$).

So How About Cellular Automata?

- How do we measure the computational complexity of a CA?
- What does a CA naturally compute?
- How does it perform (universal) computations?

The standard answer is to *simulate* a classical machine by a cellular automaton via some I/O coding conventions.

But a CA really is a discrete dynamical system, so unlike with Turing machines, Minsky machines, RAMs, etc. there is no simple notion of halting or input/output behavior.

Is there some intrinsic property that pinpoints the computational power of a CA?

Defining Universality

M. Davis proposes an answer to the question: what is a good definition for a *universal Turing machine*?

In "Automata Studies", 1956, eds. J. McCarthy and C. Shannon.

Good here means: using intrinsic properties, but not involving any arbitrary coding conventions.

A Turing machine M is *computationally universal (Davis-universal)* if the set $\mathrm{ID}_{\mathrm{fin}}$ of instantaneous descriptions of the Turing machine that gives rise to finite (ultimately halting) computations has the same Turing degree as K.

Note that $\mathrm{ID}_{\mathrm{fin}}$ is always semi-decidable, so in a Davis-universal TM $\mathrm{ID}_{\mathrm{fin}}$ is as complicated as possible.

Observations

- Every I/O universal machine is Davis-universal.
- However, some Davis-universal machines are not I/O universal: simply erase the tape and write 0.
- One can compute arbitrary partial recursive functions with $\mathrm{ID}_{\mathrm{fin}}$ as oracle. Multiple queries to test whether $\{e\}\,(x)\downarrow$ and to find the right y exploiting the fact that $\mathrm{ID}_{\mathrm{fin}}$ is complete.
- The mediating functions are very simple.
- Fundamental tension between transducers and acceptors.

Davis Stability

Theorem. *Davis*

Every total recursive function can be computed by a stable *Turing machine: all instantaneous descriptions yield finite computations.*

Hence every total recursive function can be computed by a highly non-universal machine ($\mathrm{ID}_{\mathrm{fin}} = \mathrm{ID}$).

Technical point: Uses a normal form for total recursive functions. Alternatively, one can modify the Turing machine directly (slightly more general).

Interesting for CA classification: need to deal with all configurations, not just some specially constructed ones.

Classifying CA

At any rate, this suggests to consider the *orbits* of all finite configurations as a measure of complexity of a CA.

$$\mathrm{Orb}_\rho = \{\, (X, Y) \mid Y = \rho^t(X), t \geq 0 \,\}$$

Suppose A is a semi-decidable set and define the class of all CA of complexity A

$$\mathbb{C}_A = \{\, \rho \mid \mathrm{Orb}_\rho \equiv_T A \,\}$$

Note that the Wolfram-Culik-Yu classes 1 and 2 are properly contained in class 3, which is none other than $\mathbb{C}_\emptyset$.

Non-Triviality

Theorem.

For any semi-decidable set A the class $\mathbb{C}_A$ is not empty.

The structure of $\mathcal{D}$ is inherited by cellular automata.

Proof uses a kind of "super-stable" cellular automaton to simulate a stable Turing machine that semi-decides membership in A.

The hard part here is to get the upper bound: the CA knows nothing about well-formed input configurations, it has to work on all (finite) configurations.

Classification is Hard

Theorem.

Decidability is very hard to check: class $\mathbb{C}_\emptyset$ is Σ_3-complete.

Universality is even harder: class $\mathbb{C}_K$ is Σ_4-complete.

Very interesting to find some sufficient criteria for computational universality (see Wolfram-Cook proof for rule 110).

Perhaps some kind of particle mechanism combined with local interactions?

Sufficient criteria for decidability are probably even harder (if interesting):

Pinning down complexity is more difficult that pushing it up.

Reversibility

The evolution of a configuration on a cellular automaton is usually associated with a loss of information: X cannot be recovered from $\rho(X)$.

In other words, the global maps are not injective.

Incidentally, for global maps injective implies surjective.

Can we combine reversibility with computation?

Reversible Computation

Theorem. *Lecerf, Bennett*

Any Turing machine can be simulated by a reversible Turing machine.

Compute $\widehat{f}(x) = \langle f(x), x \rangle$ for any partial recursive function f.

Theorem. *Morita, Harao*

Any reversible Turing machine can be simulated by a reversible cellular automaton.

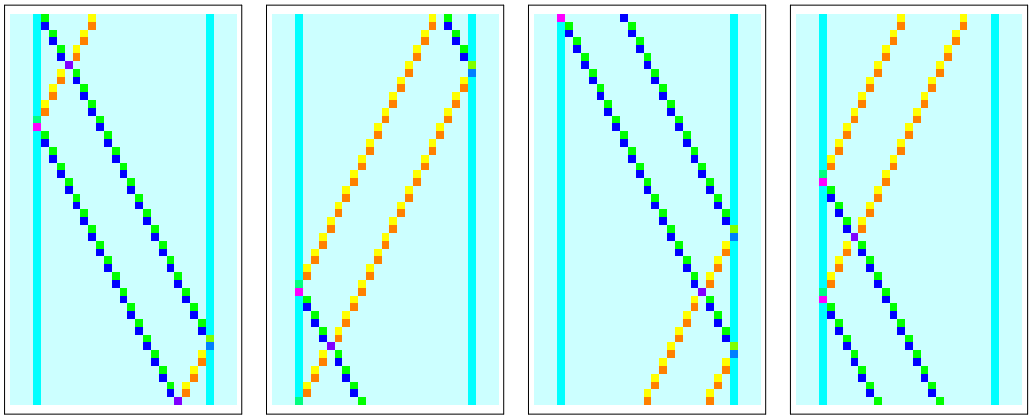
Bennett, Morita, Harao results refer to the input/output behavior of the machines.

But Lecerf actually focuses on orbits. Again, transducers versus acceptors.

Degrees of Reversible CA

Theorem. *For any semi-decidable set A there is a reversible CA in class $\mathbb{C}_A$. Thus, the orbits of a reversible CA can have arbitrary semi-decidable Turing degree.*

Proof again uses a kind of super-stable cellular automaton to simulate a stable Turing machine that semi-decides membership in A. Moreover, the CA uses the Morita/Harao machinery to insure reversibility.



Some Ruminations

- What is the relationship between classical I/O universality and Davis-universality for CAs?
- Rule 110 is universal in the classical sense. Are there simple rules that are universal only in the sense of Davis?
- How does this relate to PCE? Is there any hope to formalize the notion of information hiding?
- The degree result for reversible CA uses finite configurations, can this be pushed to more general cases?
- What is the role of intrinsic universality, in particular in conjunction with reversibility?
- What is the right computation theory to deal with CA? Classical RT or generalized RT?

A Wild Speculation

Recent work by Simpson suggests that natural examples of intermediate degrees can be found if one adopts a different notion of reduction (Muchnik degrees). One of the natural examples is based on randomness.

So how about rule 30?