UCT

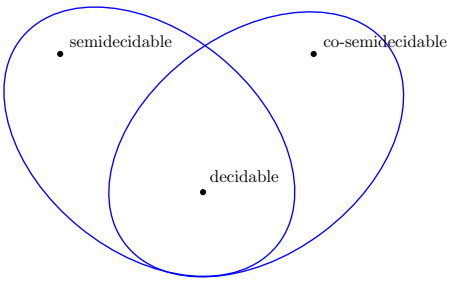
Computability, Reductions

Klaus Sutner
Carnegie Mellon University
Spring 2022

Where Are We?

- We have 3 distinct complexity classes: decidable, semidecidable, co-semidecidable.
- And a collection of decision problems such as TOT that are not decidable.
- But there is no reason why they should fit into the other two classes.

Our World



So far, we have examples of sets in three distinct complexity classes.

Question: Could this be everything?

Probably Not

We know that TOT is not decidable. But it does not look semidecidable either.

A is semidecidable iff we can find a computable membership reason for $x \in A$. We search for this reason, succeed if it exists, and fail otherwise.

For $e \in$ TOT we could find a reason for $\{e\}(0)\downarrow$, later for $\{e\}(1)\downarrow$, then for $\{e\}(2)\downarrow$, and so on. Dovetailing we can figure out reasons for more and more xs.

But how would we ever find a single reason for **all** possible x?

Bad Answer

As usual, we can resort to set theory and counting arguments: there are $2^{\aleph_0}$ subsets of $\mathbb{N}$.

Only countably many of them are decidable, semidecidable and co-semidecidable.

So almost all sets $A \subseteq \mathbb{N}$ must be more complicated.

True, but, like all cardinality based existence arguments, unsatisfactory and really bordering on useless. We want concrete examples, and perhaps a nice hierarchy.

The Misery of Counting

Here is another example where mere counting provides very little information. Cantor proved in 1874 that there are uncountably many transcendental reals: the algebraic reals are countable.

At that time, the only known transcendentals were Liouville numbers and e (C. Hermite), π came a bit later (1882, by F. Lindemann).

$0.11000100000000000000010000000000000000000000000000000000000000000000000000000000000000000000000000000000000001\ldots$

Needless to say, Cantor's result has nothing to say about numbers such as

$$e + \pi \qquad \text{or} \qquad e\,\pi \qquad \text{or} \qquad e^{\pi}$$

Kleene Normal Form

In an effort to find concrete examples, it helps to try to pin down the difference between decidable and semidecidable more carefully.

We claim that there is a trivially decidable relation $T(e, x, t)$ and a simple decoding function D such that

$$\{e\}(x)\downarrow \iff \exists t\, T(e, x, t)$$

$$\{e\}(x) \simeq D\big(\min\big(t \mid T(e, x, t)\big)\big)$$

$T(e, x, t)$ means: t is the complete computation of $\mathcal{U}$ on e and x. In essence, t is the number of steps (we can easily construct the actual computation from the step count).

$D(t)$ simply filters out the actual output of the computation.

Unboundable Search

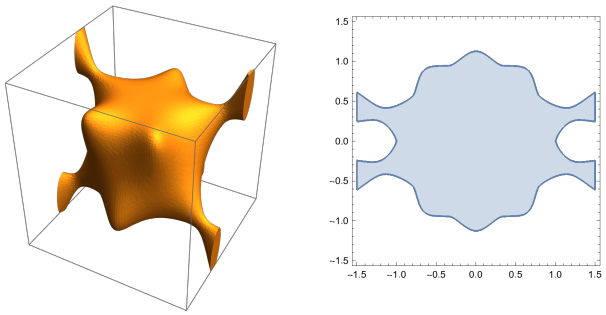
Again, $T(e, x, t)$ is trivial to compute. The problem in deciding whether $x \in W_e$ is that we do not know how large t has to be to produce convergence.

If we can compute a bound $\beta(x)$ for t ahead of time, everything is easy: just compute $T(e, x, \beta(x))$ and check; with the bound, we get a decision algorithm for any semidecidable set W_e.

But without a bound we are reduced to an unbounded search, and we are stuck with semidecidability.

Clever Idea:
Think of the unbounded search as a projection.

Geometric Projection



A 3-dimensional object casts a 2-dimensional shadow.

Projections

Definition

Suppose $R \subseteq \mathbb{N}^{n+1}$. The projection $S \subseteq \mathbb{N}^n$ of R is defined by

$$S(\boldsymbol{x}) :\Leftrightarrow \exists z\, R(z, \boldsymbol{x})$$

This is just the formal version of unbounded search:

input $\boldsymbol{x}$
foreach $w \in \mathbb{N}$ **do**
  if $(w, \boldsymbol{x}) \in R$
  then return YES
return NO

Note, though, that if the answer is NO, then this takes ω steps.

Who Cares?

In other words: the projection S of R is semidecidable whenever R is decidable: we can search for a witness z.

Much more is true: all semidecidable sets arise in this manner.

Lemma

Every semidecidable set is a projection of a decidable set.

This follows directly from Kleene normal form.

Logic and Geometry

Logically, the step from decidable to semidecidable corresponds to unbounded search, or existential quantification.

The step from semidecidable to co-semidecidable is negation.

These operations are logical, but we can also think of them as geometric:

logic	geometry
negation	complement
existential quantification	projection

Piling On

Wild Idea: What if we close decidable sets under both projections and complements?

Well, we clearly get semidecidable and co-semidecidable ones.

But there is more: we also get projections of co-semidecidable sets, their complements, projections thereof, and so on and so on.

Of course, they might all turn out to be the same, but we will see they're all different.

Projections and Complements

Let's write down careful definitions for these purely set-theoretic operations (which will turn out to make perfects sense computationally):

Let $A \subseteq \mathbb{N}^n$ where $n \geq 1$. We write

$$\text{proj}(A) = \{\, \boldsymbol{x} \in \mathbb{N}^{n-1} \mid \exists\, z\, (z, \boldsymbol{x}) \in A \,\} \subseteq \mathbb{N}^{n-1}$$
$$\text{compl}(A) = \overline{A} = \mathbb{N}^n - A$$

For any collection $\mathcal{C} \subseteq \mathfrak{P}(\mathbb{N}^n)$ of subsets of $\mathbb{N}^n$, $n \geq 1$, define $\text{proj}(\mathcal{C})$ to be the collection of all projections of sets in $\mathcal{C}$. Likewise, define $\text{compl}(\mathcal{C})$ to be the collection of all complements of sets in $\mathcal{C}$.

Arithmetical Hierarchy

Definition

Define classes of subsets of $\mathbb{N}^n$:

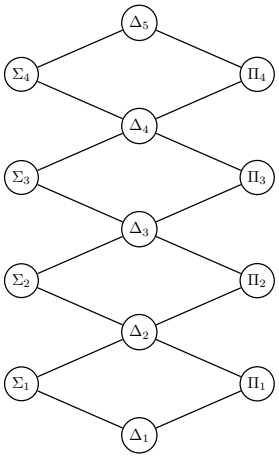
$$\begin{aligned} \Sigma_0 = \Pi_0 &= \text{ all decidable sets} \\ \Sigma_{k+1} &= \text{proj}(\Pi_k) \\ \Pi_k &= \text{compl}(\Sigma_k) \\ \Delta_k &= \Sigma_k \cap \Pi_k \end{aligned}$$

where $k \geq 1$.

Thus, Δ_1 is the class of all decidable sets, Σ_1 is the class of all semidecidable sets, and Π_1 is the class of all co-semidecidable sets.

This is also known as the Kleene-Mostowski hierarchy.

A Diagram



Arithmetical Sets

Definition

A set $A \subseteq \mathbb{N}^n$ is arithmetical if it belongs to some class Σ_k.

It follows by the usual cardinality mumbo-jumbo that the class of arithmetical sets is countable, so we are missing almost all subsets of $\mathbb{N}$.

One interesting example of a set we are missing is arithmetical truth: the set of all (code numbers of) formulas of arithmetic that are true is not an element in this hierarchy (arithmetic truth is a so-called Δ_1^1 set, just outside of our arithmetical sets).

The Bottom of AH

Of practical relevance are mostly the first few levels of the arithmetical hierarchy. We have already seen examples of decidable, semidecidable sets and co-semidecidable sets.

Here are some other examples.

- TOT is Π_2.
- The indices of all finite r.e. sets form a Σ_2 set:

$$\mathsf{FIN} = \{\, e \mid W_e \text{ finite} \,\}$$

- The indices of all cofinite r.e. sets form a Σ_3 set:

$$\mathsf{COF} = \{\, e \mid W_e \text{ cofinite} \,\}$$

None of these sets belong to the next lower Δ_k level of the hierarchy.

Heuristic

To say that $A \subseteq \mathbb{N}$ is Π_2 comes down to this: there is a decidable property P such that

$$\begin{aligned} x \in A &\iff \forall\, u \exists\, v\ P(u,v,x) \\ &\iff \neg \exists\, u\ \neg \exists\, v\ P(u,v,x) \end{aligned}$$

corresponding to a sequence project–complement–project–complement.

Empirical Fact: Writing down the set in terms of alternating quantifiers usually produces the right spot in the arithmetical hierarchy.

This really is a form of Murphy's Law: things are always as bad as they can possibly be.

Decidable Sets

Consider the index set of all decidable (recursive) sets:

$$\mathsf{REC} = \{\, e \mid W_e \text{ decidable} \,\}$$

To locate REC in the arithmetical hierarchy note that we only need to make sure that $\mathbb{N} - W_e$ is semidecidable:

$$\begin{aligned} e \in \mathsf{REC} &\Leftrightarrow \exists\, e' \,(W_e \cap W_{e'} = \emptyset \wedge W_e \cup W_{e'} = \mathbb{N}) \\ &\Leftrightarrow \exists\, e' \,(\forall\, \sigma\, (W_{e,\sigma} \cap W_{e',\sigma} = \emptyset) \wedge \forall\, x \exists\, \tau\, (x \in W_{e,\tau} \cup W_{e',\tau})) \\ &\Leftrightarrow \exists\, e' \,\forall\, x, \sigma \exists\, \tau\, (W_{e,\sigma} \cap W_{e',\sigma} = \emptyset \wedge x \in W_{e,\tau} \cup W_{e',\tau}) \end{aligned}$$

So, $\mathsf{REC} \in \Sigma_3$. In fact, one can show that $\mathsf{REC} \notin \Delta_3$.

Separation

Theorem (Post's Hierarchy Theorem)

All the inclusions $\Delta_k \subsetneqq \Sigma_k, \Pi_k \subsetneqq \Delta_{k+1}$ are proper, $k \geq 1$.

We will not prove this theorem, but we will use it as a natural introduction to hardness and completeness.

Later we will encounter a similar hierarchy in complexity theory (just add a polynomial time bound everywhere), but there it is not known whether the hierarchy is proper.

Exercises

Exercise

Show that FIN *is Σ_2 by constructing this set using projections and complementation.*

Exercise

Show that COF *is Σ_3 by constructing this set using projections and complementation.*

Exercise

Find the position in the hierarchy of

$$\mathsf{INF} = \{\, e \mid W_e \textit{ infinite} \,\}.$$

Separation

Note that the arithmetical hierarchy is cumulative, $\Sigma_k \subseteq \Sigma_{k+1}$.

How would we go about proving $\Sigma_k \subsetneq \Sigma_{k+1}$?

General Problem: We have two complexity classes $\mathcal{C}_1 \subseteq \mathcal{C}_2$. We want to separate the two classes: show $\mathcal{C}_1 \neq \mathcal{C}_2$.

OK, but how? A natural approach is to find a particularly difficult element in $\mathcal{C}_2$, and hope it won't fit into $\mathcal{C}_1$.

Next Question: How does one compare the difficulty of problems?

Cop Out

Comparing the intrinsic difficulty (logical depth) of computational problems in general can be quite tricky, it often looks like the old apples-and-oranges situation.

Fortunately, for decision problems we can weasel around this issue: we can always compare Yes/No answers, even if the questions are totally unrelated.

Restricting one's attention to decision problems is less of a loss than one might think, one can often rephrase more general questions in terms of decision problems, without losing much information.

Reductions

Suppose $\mathcal{C} \subseteq \mathfrak{P}(\mathbb{N})$ is some collection of decision problems.

Definition

A reduction is a pre-order $\preceq$ (reflexive and transitive) on $\mathfrak{P}(\mathbb{N})$.

B is $\mathcal{C}$-hard (for $\preceq$) if $A \preceq B$ for all $A \in \mathcal{C}$.

B is $\mathcal{C}$-complete (for $\preceq$) if B is $\mathcal{C}$-hard and $B \in \mathcal{C}$.

Very often we also want $\preceq$ to be compatible with $\mathcal{C}$:

$$A \preceq B, B \in \mathcal{C} \quad \text{implies} \quad A \in \mathcal{C}$$

Keeping It Simple

$A \preceq B$ is supposed to mean that B contains enough information to determine membership in A, modulo a little auxiliary computation expressed by $\preceq$.

For this to be interesting, $\preceq$ needs to be fairly weak computationally. The heavy lifting should happen in B, not in the auxiliary computation. Otherwise we could get weird situations like $A \preceq \emptyset$.

Compatibility tries to address this issue.

Hardness is Easy

It is not difficult to fabricate a hard set B for $\mathcal{C}$: just take the disjoint union over the whole class. For example, for the collection of semidecidable sets $\mathcal{E}$ we can take $K = \{\, \langle e, x \rangle \mid x \in W_e \,\}$. Clearly all membership information about semidecidable sets is contained in K, and K is certainly $\mathcal{E}$-hard (we'll talk about the right reduction in a moment).

Alas, achieving completeness can be difficult: to obtain hardness we want to pack a lot of information into B, but to ensure membership in $\mathcal{C}$ we have to keep B simple–a classical case of clashing requirements.

This works nicely for K, which is itself is also semidecidable, but can require some fairly delicate arguments for other classes.

Tackling AH

So the hope is that we can

- come up with a reasonable notion of reduction for the classes Σ_n,
- and find a Σ_n-complete set for each n,
- and show that these complete sets are all different.

And, it would be nice to have some natural examples of Σ_n-complete sets, at least for the first few levels.

The Mother of All Reductions

For the time being, consider the class of all decision problems $\mathcal{C} = \mathfrak{P}(\mathbb{N})$, so computability is not an issue.

What is the most general way we could translate a problem A into a problem B, both in $\mathcal{C}$?

The idea is to **pretend** that we can solve the decision problem for B, and use this (fictitious) ability to answer questions about A. Note that we may ask repeated questions about B, and the rest of the computation can be arbitrarily complicated.

Turing's Wild Idea

> Let us suppose we are supplied with some unspecified means of solving number-theoretic problems; a kind of oracle as it were . . . this oracle cannot be a machine. With the help of the oracle we could form a new kind of machine (call them o-machines), having as one of its fundamental processes that of solving a given number-theoretic problem.

Classical Oracles

Quoi?

- Oracle machines were introduced by Turing in a 1939 paper under the name of o-machines (as opposed to the original a-machines). Curiously, he never really exploited this idea (Post and Kleene did).
- As we will see, OTMs provide a powerful way to classify problems according to their complexity.
- Think of the oracle as a data base, created by an alien super-civilization a trillion years ago. The oracle Turing machine has access to all their wisdom.

Less Informally

Fix some set $A \subseteq \mathbb{N}$. We want to add knowledge about A to a Turing machine: the machine writes $x \in \mathbb{N}$ on a special tape (alternatively, a special area on the main tape) and then enters a magical query state q_Q.

At this point, a genie takes over and checks whether $x \in A$. If so, the machine state is changed to q_Y, otherwise it is changed to q_N.

Then the normal computation resumes.

Definition

A Turing machine with this added facility is a oracle Turing machine (OTM) with oracle A.

For Real

The last slide is rather vague. But one can make the notion of an oracle Turing machine precise, the same way as an ordinary Turing machine can be defined precisely (in the usual pseudo-set-theory setting).

Exercise

Give a precise definition of oracle Turing machines by redefining the next-step relation.

We write $\mathcal{M}_e^A$ for the eth OTM with oracle A and $\{e\}^A$ for the corresponding partial function.

R. I. Soare

The Turing o-machine is the single most important concept in computability, theoretical or practical.

Comments

- The whole point behind Turing machines (meaning a-machines) is that they are physically realizable, at least in principle. They capture everything that is physically possible, and more.
- But o-machines are **NOT**, though some esteemed members of the hyper-computation community fail to realize that.
- The one exception is when the oracle A is decidable: in this case we can replace magic by another Turing machine.

In other words, decidable oracles are no better than none, we get no additional power (but note that the running time might improve).

The Key Generalization

We can now generalize all the basic notions we have relative to an arbitrary oracle A: we copy our old definitions, replacing TM by OTM everywhere.

- A-computable
- A-decidable
- A-semidecidable

So computable is $\emptyset$-computable and so forth.

More Precisely . . .

Definition

Let $A, B \subseteq \mathbb{N}$. A partial function $f : \mathbb{N}^k \rightharpoonup \mathbb{N}$ is computable in A or A-computable if there is an oracle Turing machine that computes f using A as an oracle.

A set B is Turing-reducible to A or A-decidable if its characteristic function is A-computable.

A set B is semidecidable in A or A-semidecidable if its semi-characteristic function is A-computable.

Turing reducibility is usually written

$$B \leq_T A$$

A Preorder

Proposition

Turing reducibility is a preorder (reflexive and transitive):

- $A \leq_T A$.
- $A \leq_T B$ *and* $B \leq_T C$ *implies* $A \leq_T C$.

Proof. Every call to oracle B in the algorithm for A can be replaced by a computation which uses calls to oracle C.

The actual computation can be absorbed by the algorithm, so that all that remains are the calls to oracle C.

□

Of course, $\leq_T$ is not symmetric. There even are incomparable semidecidable sets, but that's more complicated.

Dire Warning

Note that by definition Turing reducibility can handle complements:

$$\overline{A} \leq_T A$$

As a consequence, Turing reducibility is compatible with $\mathfrak{P}(\mathbb{N})$, but not with the class $\mathcal{E}$ of semidecidable sets, or Σ_n in general; it is simply too brutish for these classes.

We will need to fix this later, but for the time being let's just explore Turing reducibility.

A Meta-Theorem

As it turns out, the machinery we developed for computable functions carries over to A-computable functions, just about verbatim.

In particular we still have a Kleene style enumeration

$$\{e\}^A$$

of all A-computable functions.

Then $B \leq_T A$ means

$$B(x) \simeq \{e\}^A(x)$$

for some index e (we abuse notation slightly by writing B for the characteristic function of B).

Reduction to Halting

Claim

Every semidecidable set is K-decidable.

Proof. Let W semidecidable. We will translate a membership query for W into a membership query for K.

Here is an overly careful argument. Define the following function:

$$g(x,z) \simeq \begin{cases} 0 & \text{if } x \in W, \\ \uparrow & \text{otherwise.} \end{cases}$$

Clearly, g is computable and has some index $\widehat{g}$. Then

$$g(x,z) \simeq \{\widehat{g}\}(x,z) \simeq \{S_1^1(\widehat{g},x)\}(z)$$

by the S-m-n theorem.

The map $f(x) := S_1^1(\widehat{g},x)$ is easily computable (primitive recursive).

But then $x \in W \iff f(x) \in K$, done. □

This is the anal-retentive version of the proof, usually this would be abbreviated to something a bit more cryptic like so: let

$$\{f(x)\}(z) \simeq \begin{cases} 0 & \text{if } x \in W, \\ \uparrow & \text{otherwise.} \end{cases}$$

Then f is primitive recursive, blahblahblah, done.

Truth in Advertising

Note that the critical point here is that an index for the last function is easily computable. One can prove this formally by applying the S-m-n theorem, but usually no one bothers.

Since an oracle TM can compute f, it can check membership in W by making just one call to the oracle and returning the same answer true or false (a bit like tail recursion).

This is a very special case, in general multiple calls are needed and the OTM has to do more work than just computing a primitive recursive function.

More on this special type of reduction (many-one reduction) later.

Better Reductions

The fact that Turing reducibility automatically clobbers complements makes is it awkward to use in connection with semidecidability. Essentially, Turing reducibility is just too coarse for semidecidable sets.

We are looking for a weaker reduction that is compatible:

$$A \preceq B, B \text{ semidecidable} \quad \text{implies} \quad A \text{ semidecidable}$$

Forward Pointer: Finding the right notion of reducibility is absolutely critical for lower complexity classes like $\mathbb{NP}$.

Many-One

Here is a type of reduction that is suggested by some of the examples above.

Definition

Let $A, B \subseteq \mathbb{N}$. A is many-one reducible to B if there exists a computable function $f : \mathbb{N} \to \mathbb{N}$ such that

$$x \in A \iff f(x) \in B.$$

In symbols: $A \leq_m B$.

Downward Closure

Proposition

$$A \leq_m B \quad \textit{implies} \quad A \leq_T B$$

The opposite implication is false.

Proposition

- *$A \leq_m B$ and B decidable implies that A is decidable.*
- *$A \leq_m B$ and B semidecidable implies that A is semidecidable.*

Lower Bounds

Recall our proof that all semidecidable sets are many-one reducible to K. There is an easily computable function f such that

$$\{f(x)\}(z) \simeq \begin{cases} 0 & \text{if } x \in W \\ \uparrow & \text{otherwise.} \end{cases}$$

Then $x \in W \iff f(x) \in K$, so we have $W \leq_m K$.

Convention

In conjunction with the arithmetical hierarchy, the default reduction is many-many reducibility.

E.g., when we say that A is Σ_n-complete this means

$$A \in \Sigma_n \qquad \text{and} \qquad B \leq_m A, \text{ all } B \in \Sigma_n$$

In particular $\mathcal{E}$-complete always is defined this way.

Exercise

Show that $\leq_m$ is compatible with Σ_n:

$$B \leq_m A, A \in \Sigma_n \qquad \textit{implies} \qquad B \in \Sigma_n$$

Some Index Sets

Here is a small collection of natural index sets.

$$\begin{aligned} \text{FIN} &= \{\, e \in \mathbb{N} \mid W_e \text{ is finite} \,\} \\ \text{INF} &= \{\, e \in \mathbb{N} \mid W_e \text{ is infinite} \,\} \\ \text{TOT} &= \{\, e \in \mathbb{N} \mid W_e = \mathbb{N} \,\} \\ \text{REC} &= \{\, e \in \mathbb{N} \mid W_e \text{ is decidable } \,\} \\ \text{CMP} &= \{\, e \in \mathbb{N} \mid W_e \text{ is complete } \,\} \end{aligned}$$

By Rice's theorem, they are all undecidable. But we would like to understand their relative complexity more precisely.

Relative Complexity

Theorem

$$K <_T \text{FIN} <_T \text{REC}$$

$$\text{FIN} \equiv_T \text{INF} \equiv_T \text{TOT}$$

FIN $\equiv_T$ INF is clear, but the others require work.

Also, we would like to have many-one reductions whenever possible.

We'll just hint at what needs to be done for a proof.

How Bad Can It Be? 54

Intuitively, FIN is not semidecidable since by definition

$$e \in \mathsf{FIN} \iff \exists b \forall x, \sigma\, (x \in W_{e,\sigma} \Rightarrow x < b)$$

The matrix of the formula is easily decidable, but we are not just doing an infinite search (that would be the $\exists b$ part). Because of the universal quantifier, the only obvious bound we get from this is Σ_2.

But note: this is just an upper bound, not a lower bound. Experience shows, though, that unless we are obviously wasting quantifiers these upper bounds are tight.

Lower Bound 55

Claim: FIN is Σ_2-complete.

To see this, suppose $A \in \Sigma_2$ arbitrary. We need to show that $A \leq_m$ FIN.

First note that by the definition of Σ_2 we have

$$x \in A \Leftrightarrow \exists s \forall t\ R(x, s, t)$$

for some decidable relation R.

Define $f(x) = e$ where e is an index such that

$$W_e = \{\, z \mid \forall s < z \exists t \neg R(x, s, t) \,\}$$

One can check that the set on the right is indeed semidecidable.

Then f is the desired many-one reduction.

Even Worse 56

Unsurprisingly, REC is even worse. Quantifier counting produces an upper bound of Σ_3

$$\begin{aligned} e \in \mathsf{REC} &\iff \exists e'\, (W_e = \mathbb{N} - W_{e'}) \\ &\iff \exists e' \forall x\, (\forall \sigma\, (W_{e,\sigma} \cap W_{e',\sigma} = \emptyset) \wedge \exists \tau\, (x \in W_{e,\tau} \cup W_{e',\tau})) \end{aligned}$$

As before, this upper bound is already tight, one can show that $A \leq_m$ REC for any Σ_3 set A.

Alas, this requires more work ...

Another Proof 57

We will show that

$$\mathsf{INF} \leq_m \mathsf{TOT}$$

As a warmup exercise, consider the downward closure operation

$$\mathsf{dc}(A) = \{\, x \in \mathbb{N} \mid \exists a \in A\, (a \geq x) \,\}$$

So $\mathsf{dc}(A)$ fills in the holes in A.

In general

$$\mathsf{dc}(A) = \begin{cases} \{\, z \mid z \leq \max A \,\} & \text{if } A \text{ is finite,} \\ \mathbb{N} & \text{otherwise.} \end{cases}$$

Effectivizing the Argument 58

So $\mathsf{dc}(A)$ translates from infinite to total, fairly close to what we want.

But note that $\mathsf{dc}(W)$ is r.e. whenever W is r.e. Moreover, we can compute an index for $\mathsf{dc}(W)$ from an index for W. In fact, there is a primitive recursive function f that does it:

$$\{f(e)\}(z) \simeq \begin{cases} 0 & \text{if } \exists u\, (u \geq z \wedge u \in W_e), \\ \uparrow & \text{otherwise.} \end{cases}$$

Hence $\{f(e)\}$ is either constant 0, or undefined for all sufficiently large z. More precisely, W_e is infinite iff $W_{f(e)} = \mathbb{N}$.

Hence we have the desired reduction: $e \in \mathsf{INF} \iff f(e) \in \mathsf{TOT}$, done.