Halting (A_{TM}) is recognizable

While the Halting Problem is not solvable, it is re, recognizable or semi-decidable.

To see this, consider the following semi-decision procedure. Let *P* be an arbitrary procedure and let *x* be an arbitrary natural number. Run the procedure *P* on input *x* until it stops. If it stops, say "yes." If P does not stop, we will provide no answer. This semi-decides the Halting Problem. Here is a procedural description.

```
Semi_Decide_Halting() {
    Read P, x;
    P(x);
    Print "yes";
}
```

Enumeration Theorem

- Define

  $$\mathbf{W_n} = \{\ \mathbf{x} \in \mathcal{N} \mid \varphi(\mathbf{n},\mathbf{x})\downarrow\ \}$$

- Theorem: A set **B** is re iff there exists an **n** such that $\mathbf{B} = \mathbf{W_n}$.
  Proof: Follows from definition of $\varphi(\mathbf{n},\mathbf{x})$.
- This gives us a way to enumerate the recursively enumerable (semi-decidable) sets.

Non-re Problems

- There are even "practical" problems that are worse than unsolvable -- they're not even semi-decidable.
- The classic non-re problem is the Uniform Halting Problem, that is, the problem to decide of an arbitrary effective procedure P, whether or not P is an algorithm.
- Assume that the set of algorithms (TOTAL) can be enumerated, and that F accomplishes this. Then

  $F(x) = F_x$

  where $F_0, F_1, F_2, \ldots$ is a list of indexes of all and only the algorithms

The Contradiction

- Define $G(x) = \text{Univ}(F(x), x) + 1 = \varphi_{F(x)}(x) = F_x(x) + 1$
- But then G is itself an algorithm. Assume it is the g-th one

$$F(g) = F_g = G$$

Then, $G(g) = F_g(g) + 1 = G(g) + 1$

- But then G contradicts its own existence since G would need to be an algorithm.
- This cannot be used to show that the effective procedures are non-enumerable, since the above is not a contradiction when G(g) is undefined. In fact, we already have shown how to enumerate the (partial) recursive functions.

The Set TOTAL

- The listing of all algorithms can be viewed as
  TOTAL = $\{ f \in \boldsymbol{N} \mid \forall x\, \varphi_f(x)\downarrow \}$
- We can also note that
  TOTAL = $\{ f \in \boldsymbol{N} \mid W_f = \boldsymbol{N} \}$, where W_f is the domain of φ_f
- Theorem: TOTAL is not re.
  Proof: Shown earlier.

Insights

Non-re nature of algorithms

- No generative system (e.g., grammar) can produce descriptions of all and only algorithms
- No parsing system (even one that rejects by divergence) can accept all and only algorithms

- Of course, if you buy Church's Theorem, the set of all procedures can be generated. In fact, we can build an algorithmic acceptor of such programs.

Many unbounded ways

- How do you achieve divergence, i.e., what are the various means of unbounded computation in each of our models?
- GOTO: Turing Machines and Register Machines
- Minimization: Recursive Functions
  - Why not primitive recursion/iteration?
- Fixed Point: (Ordered) Factor Replacement Systems

Non-determinism

- It sometimes doesn’t matter
  - Turing Machines, Finite-State Automata, Linear Bounded Automata
- It sometimes helps
  - Push Down Automata
- It sometimes hinders
  - Factor Replacement Systems, Petri Nets

HOW HARD IS IT TO ANALYZE PETRI NETS?

TO DETERMINE IF SOME MARKING CAN EVENTUALLY ARISE IS IN

EXPSPACE (N)

SOLVABLE, BUT TAKES EXPONENTIAL SPACE

TIME IS ACTUALLY 2^{2^N}

IF PRIORITY ADDED TO TRANSITIONS, PETRI NETS ARE COMPLETE MODELS OF COMPUTATION.

CAN RECAST AS FRS

W/O ORDERING $\equiv$ PETRI NET

W ORDERING $\equiv$ PETRI NET WITH PRIORITIES

Reduction Concepts

- Proofs by contradiction are tedious after you've seen a few. We really would like proofs that build on known unsolvable problems to show other, open problems are unsolvable. The technique commonly used is called reduction. It starts with some known unsolvable problem and then shows that this problem is no harder than some open problem in which we are interested.

PROBLEM CATEGORIES

RECURSIVE (SOLVABLE)

LOTS OF EXAMPLES

RE, NON-RECURSIVE (UNDEC BUT SEMI DEC)

$HALT = \{ \langle f, x \rangle \mid \varphi_f(x)\downarrow \}$

SHOWN BY DIAGONALIZATION

NON-RE (CANNOT EVEN SEMI-DECIDE)

$TOTAL = \{ f \mid \forall x \; \varphi_f(x)\downarrow \}$

PROBLEM CATEGORIES

RECURSIVE (SOLVABLE)

LOTS OF EXAMPLES

RE, NON-RECURSIVE (UNDEC BUT SEMI DEC)

$HALT = \{\langle f,x \rangle \mid \varphi_f(x)\downarrow\}$

SHOWN BY DIAGONALIZATION

NON-RE (CANNOT EVEN SEMI-DECIDE)

$TOTAL = \{ f \mid \forall x\ \varphi_f(x)\downarrow\}$

INTRO TO REDUCTION

$A \leq_m B$ IF THERE EXISTS SOME COMPUTABLE ALGORITHM $f \ni$

$x \in A \Leftrightarrow f(x) \in B$

IF B IS EASY TO SOLVE THEN SO IS A IF f DOES NOT ADD TO COMPUTATIONAL COMPLEXITY

HOWEVER, IF A IS KNOWN TO BE HARD (OR EVEN UNSOLVABLE) AND f DOES NOT CHANGE THE COMPLEXITY LANDSCAPE, THEN B MUST BE HARD AT LEAST WITHIN THE ORDER OR f's AND A's COMPLEXITY, IF A IS UNSOLVABLE THEN SO IS B.

SHOWING COMPLEXITY OF NEW PROBLEMS

FIRST TECHNIQUE IS REDUCTION

LET B BE SOME SET OF UNKOWN COMPLEXITY

LET A BE SOME SET OF KNOWN COMPLEXITY

LET f BE A COMPUTABLE M-1 FUNCTION (TOTAL)

$$A \le_m B \text{ OR JUST } A \le B$$

VIA f IF

$x \in A$ IFF $f(x) \in B$

IF A IS RE, NON-REC. THEN B IS NON-REC., BUT NOT NEC. RE

IF A IS NON-RE THEN B IS NON-RE AND, OF COURSE, NON-REC.

REDUCTION EXAMPLE #1

SHOW HALT $\leq$ TOTAL

LET f, x BE ARBITRARY NAT. NUMBERS

$\langle f, x \rangle \in$ HALT IFF $\varphi_f(x)\downarrow$

DEFINE F_x BY

$\forall y\, F_x(y) = \varphi_f(x)$ // IGNORES INPUT

NOW

$\langle f, x \rangle \in$ HALT IFF $F_x \in$ TOTAL

THUS, HALT $\leq_m$ TOTAL

BY THIS, TOTAL IS NON-REC. BUT

WE DO NOT KNOW IF IT'S RE

(WELL, WE DO, AND IT'S NOT)

NOTE: WE CAN LEAVE OUT φ

AND JUST SAY

$\forall y\, F_x(y) = f(x)$ // OVERLOAD

FOR CONVENIENCE

EXAMPLE #2

HASZERO = $\{f \mid \exists x\ f(x) = 0\}$ //SKIP $\emptyset$

SHOW HASZERO IS NON-REC.

LET f, x BE ARB.

DEFINE F_x BY

$\forall y\ F_x(y) = f(x) - f(x)$

CLEARLY, $\forall y F_x(y) = 0$ IF $f(x)\downarrow$

ELSE $\forall y\ F_x(y)\uparrow$

SO

$\langle f, x\rangle \in$ HALT $\Leftrightarrow$ $F_x \in$ HASZERO

THUS, HASZERO IS NON-REC. SINCE

HALT $\leq_m$ HASZERO

BUT IS HASZERO RE?

WELL IT IS AND WE WILL SHOW

THAT LATER

EXAMPLE #3

$$\text{ZERO} = \{ f \mid \forall x\, f(x) = 0 \}$$

SHOW ZERO IS NON-RE

NOTE PRIOR EXAMPLE SHOWED NON-REC !!

LET f BE ARB.

DEFINE

$$\forall x\; h(x) = f(x) - f(x)$$

NOW

$f \in \text{TOTAL}$ IFF $\forall x\, f(x)\downarrow$

IFF $\forall x\; h(x) = 0$

IFF $h \in \text{ZERO}$

THUS,

$$\text{TOTAL} \le_m \text{ZERO}$$

AND SO ZERO IS NON-RE

EXAMPLE #4

$\text{IDENTITY} = \{ f \mid \forall x\, f(x) = x \}$

LET f BE AN ARBITRARY INDEX

DEFINE

$$\forall x \; g_f(x) = f(x) - f(x) + x$$

NOW

$$
\begin{aligned}
f \in \text{TOTAL} \quad & \text{IFF} \quad \forall x \; f(x)\downarrow \\
& \text{IFF} \quad \forall x \; g_f(x) = x \\
& \text{IFF} \quad g_f \in \text{IDENTITY}
\end{aligned}
$$

THUS,

$$\text{TOTAL} \leq_m \text{IDENTITY}$$

AND SO IDENTITY IS NOT EVEN RE

TYPES OF REDUCTION

M-1 $\leq_m$

1-1 $\leq_1$

TURING (AKA ORACLE)

$\leq_t$

DEGREES ARE EQUIV. CLASSES

$\equiv_1$

$\equiv_m$

$\equiv_t$

ONE CLASS WE CARE ABOUT IS COMPLETE DEGREE (HIGHEST) AMONG RE SETS

RE COMPLETE

S IS RE-COMPLETE IFF

(1) S IS RE

(2) IF T IS RE THEN $T \leq S$

HALT (AKA K_0) IS RE-COMPLETE

LET A BE ARB RE SET THEN

$A = \mathrm{Dom}\,\varphi_a$ FOR SOME INDEX a

HERE $A = W_a$ (ENUMERATION THEOREM)

$$x \in A \iff x \in \mathrm{Dom}(\varphi_a) \iff \varphi_a(x)\downarrow \iff \langle a, x\rangle \in K_0$$

THUS

$A \leq K_0$ (REALLY $A \leq_1 K_0$)

K IS ALSO RE-COMPLETE

$K = \{ f \mid \varphi_f(f)\downarrow \}$

LET f, x BE ARB.

DEFINE $\forall y\ F_x(y) = f(x)$

$\langle f, x\rangle \in \text{HALT}(K_0) \Leftrightarrow$

$x \in \text{DOM}(f) \Leftrightarrow$

$\forall y\ F_x(y)\downarrow$

$\Rightarrow F_x \in K$

$\langle f, x\rangle \notin \text{HALT}(K_0) \Leftrightarrow$

$x \notin \text{DOM}(f) \Leftrightarrow$

$\forall y\ F_x(y)\uparrow$

$\Rightarrow F_x \notin K$

THUS,

$K_0 \leq K$ (ACTUALLY $K \equiv K_0$)

BUT K IS OBVIOUSLY RE

AND SO K IS RE-COMPLETE