**Noncomputable Sets**

Computable strings in \( \{0,1\}^\omega \)

An infinite string \( s \) in \( \{0,1\}^\omega \) is computable iff some procedure computes its digits.

(Procedure applied to argument \( n \) returns \( n \)th digit of \( s \).)

\{ASCII\}* is countable

Only countably many finite ASCII strings. (List them in order of length.)

Procedures can be expressed in ASCII, so only countably many procedures.

Noncomputable strings in \( \{0,1\}^\omega \)

So only countably many computable infinite binary strings.

But \( \{0,1\}^\omega \) is uncountable, so there must be noncomputable strings in \( \{0,1\}^\omega \) —in fact, uncountably many!
The Halting Problem

There is no test procedure for halting of arbitrary procedures. The Halting Problem is not decidable by computational procedures.

String procedure $P$ takes a String argument:
- $P("no")$ returns 2
- $P("albert")$ returns "meyer"
- $P("&&%99!!")$ causes an error
- $P("what now?")$ runs forever.

Let $s$ be an ASCII string defining $P_s$. Say $s$ HALTS iff $P_s(s)$ returns something.

Suppose there was a procedure $Q$ that decided HALTS:
- $Q(s)$ returns "yes" if $s$ HALTS
- returns "no" otherwise
The Halting Problem

Modify \( Q \) to \( Q' \):
- \( Q'(s) \) returns "yes" if \( Q(s) \) returns "no"
- \( Q'(s) \) returns nothing if \( Q(s) \) returns "yes"

So
- \( s \) HALTS iff \( Q'(s) \) returns nothing

Let \( \dagger \) be the text for \( Q' \)
- So by def of HALTS:
  - \( \dagger \) HALTS iff \( Q'(\dagger) \) returns
- and by def of \( Q' \):
  - \( Q'(\dagger) \) returns iff \( \text{NOT}(\dagger \text{ HALTS}) \)

CONTRADICTION:
- \( \dagger \) HALTS iff \( \text{NOT}(\dagger \text{ HALTS}) \)
- There can't be such a \( Q \): it is impossible to write a procedure that decides whether strings HALT
The Type-checking Problem

There is no string procedure that type-checks perfectly, because:

Suppose \( C \) was a type-checking procedure: for program text \( s \)

\( C(s) \) returns “yes” if \( s \) would cause a run-time type error

returns “no” otherwise.

Use \( C \) to get a HALTS Tester \( H \):
to compute \( H(s) \), construct a new program text, \( s' \), that acts like a slightly modified interpreter for \( s \). Namely:

- \( s' \) skips any command that would cause \( s \) to make a run-time type error.
- \( s' \) purposely makes a type-error when it finds that \( s \) HALTS.

Then compute \( C(s') \) and return the same value.

So \( s \) HALTS

iff \( s' \) makes run-time type error

iff \( C(s') = “yes” \)

iff \( H(s) = “yes” \)
The Type-checking Problem

H solves the Halting Problem, a contradiction. So C must not error check correctly.

No run-time properties are decidable

The same reasoning shows that there is no perfect checker for essentially any property of procedure outcomes.