CS 245

Suppose P is a program.

Let Σ be some finite input alphabet for P.

The inputs to P are the finite strings of letters in Σ.

We use Σ^* to denote the finite strings over alphabet Σ.

We can consider programs that do some computation on their input, and then, if they halt, the programs may 'accept' the input.

Program P accepts $w \in \Sigma^*$ if P starts computation on input w and eventually halts computation in an accept state.

Some state of P are halting.

If P enters a halting state, no further computation is possible.

Some halting states are accept states.

Halting states that are not accept states are reject states.

Let $L(P)$ be the language of P, that is, the set of $w \in \Sigma^*$ for which P accepts w when P begins computation on input w.

$L \subseteq \Sigma^*$ is recognizable if there is some program P such that $L(P) = L$.

Program P is a decider if for all $w \in \Sigma^*$ it is the case that P halts on input w.

If P is a decider then

P decides $L(P)$.

A language $L \subseteq \Sigma^*$ is decidable

if there is some decider, P,

such that $L(P) = L$.

For example, if L is the language a finite state machine then L is decidable.

Consider $L_M = \{ \langle P, w \rangle \mid P$ is a program and P accepts $w \in \Sigma^* \}$.

Fact: L_M is not decidable.

That is, there is no program, $\hat{P}$, such that $\hat{P}$ is a decider and $L(\hat{P}) = L_M$.

Notice that L_M is recognizable.

We can create a program, U, such that on input $\langle P, w \rangle$ the program U mimics the behavior of P on w. In this way if P on input w halts in an accept state then U will halt in an accept state. But if P on w does not halt then U on input $\langle P, w \rangle$ will not halt.

So U recognizes L_M.

Suppose, to the contrary, that L_M is decidable and that H decides L_M.

So, by assumption, H is a program that halts on all inputs. Further, on input $\langle P, w \rangle$, H halts and accepts if P accepts w, otherwise H halts and rejects on input $\langle P, w \rangle$.

Construct program D that uses H as a subroutine.

Let P be some program.
P is itself a string over Σ^*.

D then calls H on input $\langle P, P \rangle$ to see if the program P accepts the input string P.

After D has determined, by using H, if program P accepts the input string P, program D then does the opposite.

D: On input string P, program

D calls H on $\langle P, P \rangle$.

If H accepts $\langle P, P \rangle$ then

D halts and rejects.

If H rejects $\langle P, P \rangle$ then

D halts and accepts.

Summarizing: D(⟨P⟩) halts and accepts if P does not halt and accept input P.

: D(⟨P⟩) halts and rejects if P does halt and accept input P.

Consider program D on input program D.

D on input D halts and accepts if D on input D rejects.

D on input D rejects if D on input D accepts.

So D, on input D must do the opposite of what D on input D does. We can conclude that there is no such D, and therefore, no such H.

Conclusion: L_M is not decidable.

The proof technique is known as a diagonalization proof.

Programs	Inputs P_0	P_1	P_2
P_0	accept		
P_1			accept
P_2	accept		
⋮			

where is D on D?

Recall that for formula ϕ of propositional logic we can decide if ϕ is a validity.

If ϕ has n propositional atoms, evaluate ϕ on each of 2^n valuations to the atoms.

If ϕ evaluates to 1 on all valuations then ϕ is valid.

Otherwise ϕ is not a validity.

Fact: The decision problem of validity in first order logic is not decidable. That is, there is no program P which given arbitrary input formula ϕ of first-order logic halts and accepts if ϕ is a validity and otherwise halts and rejects if ϕ is not a validity.

One way to prove this fact is to take a problem that is known to be undecidable and show that if the validity problem for first order logic was decidable then the known undecidable problem would also be decidable, a contradiction.

The Post correspondence problem is another example of a problem known to be undecidable.

Post Correspondence: Given a finite sequence of pairs

$$(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)$$

such that s_i and t_i are each binary strings of positive length, is there a sequence of indices $i_1, i_2, \ldots, i_n$ with $n \geq 1$ such that

$$s_{i_1} s_{i_2} \cdots s_{i_n} = t_{i_1} t_{i_2} \cdots t_{i_n}$$

Given a correspondence problem it can be turned into a formula ϕ such that ϕ is a validity if and only if the correspondence problem has a solution.

Since the Post correspondence problem is known to be undecidable then the validity problem for First order logic is also undecidable.