English for Computer Science

Mohammad Farshi

Department of Computer Science,
Yazd University

1388-1389

Azmoone 1389(CS)

Passage 1

The Pigeonhole Principle

In Example 2.13 we used an important reasoning technique called the pigeonhole principle. Colloquially, if you have more pigeons than pigeonholes, and each pigeon flies into some pigeonhole, then there must be at least one hole that has more than one pigeon. In our example, the "pigeons" are the sequences of n bits, and the "pigeonholes" are the states. Since there are fewer states than sequences, one state must be assigned two sequences.

The pigeonhole principle may appear obvious, but it actually depends on the number of pigeonholes being finite. Thus it works for finite-state automata, with the states as pigeonholes, but does not apply to other kinds of automata that have an infinite number of states.

Passage 1

The Pigeonhole Principle

To see why the finiteness of the number of pigeonholes is essential, consider the infinite situation where the pigeonholes correspond to integers $1, 2, \ldots$. Number the pigeons $0, 1, 2, \ldots$, so there is one more pigeon than there are pigeonholes. However, we can send pigeon i to hole $i + 1$ for all $i \geq 0$. Then each of the infinite number of pigeons gets a pigeonhole, and no two pigeons have to share a pigeonhole.

Passage 1

16-Since the number of sequences is more than the number of states, then the pigeonhole principle is . . ..

1) applicable 2) irrelevant 3) violated 4) unreasonable

17- The pigeonhole principle is used · · · .

1) for inductive proofs 2) to refute a known claim
3) to establish a contradiction 4) as a means to conclude a known result

18- When the number of pigeons is not finite, then the pigeonhole principle · · · .

1) is invalid 2) is contradictory
3) applies more strongly 4) is valid inductively

Passage 1

16-Since the number of sequences is more than the number of states, then the pigeonhole principle is . . ..

1) applicable 2) irrelevant 3) violated 4) unreasonable

17- The pigeonhole principle is used · · · .

1) for inductive proofs
2) to refute a known claim
3) to establish a contradiction
4) as a means to conclude a known result

18- When the number of pigeons is not finite, then the pigeonhole principle · · · .

1) is invalid
2) is contradictory
3) applies more strongly
4) is valid inductively

Passage 1

16-Since the number of sequences is more than the number of states, then the pigeonhole principle is . . ..

1) applicable 2) irrelevant 3) violated 4) unreasonable

17- The pigeonhole principle is used · · · .

1) for inductive proofs
2) to refute a known claim
3) to establish a contradiction
4) as a means to conclude a known result

18- When the number of pigeons is not finite, then the pigeonhole principle · · · .

1) is invalid
2) is contradictory
3) applies more strongly
4) is valid inductively

Passage 1

19- For automata having an infinite number of states, the pigeonhole principle · · · .

1) considers the sequences as states.
2) is applicable if we do not consider the states as pigeonholes
3) is not applicable
4) reduces the states to be finite

20- There · · · in the pigeonhole principle.

1) are exactly two pigeons in a pigeonhole
2) is exactly one pigeon for each pigeonhole
3) are more pigeonholes than pigeons
4) are two interpretations of finite and infinite cases

Passage 1

19- For automata having an infinite number of states, the pigeonhole principle · · · .

1) considers the sequences as states.
2) is applicable if we do not consider the states as pigeonholes
3) is not applicable
4) reduces the states to be finite

20- There · · · in the pigeonhole principle.

1) are exactly two pigeons in a pigeonhole
2) is exactly one pigeon for each pigeonhole
3) are more pigeonholes than pigeons
4) are two interpretations of finite and infinite cases

Passage 2

Why Undecidable Problems Must Exist

While it is tricky to prove that a specific such as “hello world problem” discussed here, must be undecidable, it is quite easy to see why almost all problems must be undecidable by any system that involves programming. Recall that a “problem” is rally membership of a string in a language. The number of different languages over any alphabet of more than one symbol is not countable. That is there is no way to assign integers to the languages such that every language has an integer, and every integer is assigned to one language.
On the other hand programs, being finite strings over a finite alphabet (typically a subset of the ASCII alphabet). are countable. That is, we can order them by length, and for programs of the same length, order them lexicographically. Thus, we can speak of the first program, the second program, and in general, the ith program for any integer i.

Passage 2

Why Undecidable Problems Must Exist

As a result, we know there are infinitely fewer programs than there are problems. If we picked a language at random, almost certainly it would be an undecidable problem. The only reason that most problems appear to be decidable is that we rarely are interested in random problems. Rather, we tend to look at fairly simple well-structured problems, and indeed these are often decidable. However, even among the problems we are interested in and can state clearly and succinctly, we find many that are undecidable; the hello-world problem is a case in point.

Passage 2

21- The “hello world problem” ....

1) is decidable, but its being declared undecidable is due to a trick
2) is undecidable even though it is not a random problem
3) is decidable because it is not a random problem
4) can be decidable or undecidable depending on how it is considered as an input to a program

22- The number of different languages is uncountable only if the number of the symbols of the alphabet being used is · · · .

1) unknown 2) finite 3) not finite 4) more than one

23- The programs are countable because we can assign · · · .

1) integers to distinguish programs
2) every integer to one program and every program to one integer
3) exactly one integer to programs of the same length
4) programs to integers after they are executed

Passage 2

21- The “hello world problem” ....

1) is decidable, but its being declared undecidable is due to a trick
2) is undecidable even though it is not a random problem
3) is decidable because it is not a random problem
4) can be decidable or undecidable depending on how it is considered as an input to a program

22- The number of different languages is uncountable only if the number of the symbols of the alphabet being used is · · · .

1) unknown 2) finite 3) not finite 4) more than one

23- The programs are countable because we can assign · · · .

1) integers to distinguish programs
2) every integer to one program and every program to one integer
3) exactly one integer to programs of the same length
4) programs to integers after they are executed

Passage 2

21- The “hello world problem” ....

1) is decidable, but its being declared undecidable is due to a trick
2) is undecidable even though it is not a random problem
3) is decidable because it is not a random problem
4) can be decidable or undecidable depending on how it is considered as an input to a program

22- The number of different languages is uncountable only if the number of the symbols of the alphabet being used is · · · .

1) unknown 2) finite 3) not finite 4) more than one

23- The programs are countable because we can assign · · · .

1) integers to distinguish programs
2) every integer to one program and every program to one integer
3) exactly one integer to programs of the same length
4) programs to integers after they are executed

Passage 2

24- A language being picked randomly · · · .

1) cannot be undecidable
2) is decidable if programs in that language are countable
3) is often undecidable because it is often complicated
4) is undecidable if it contains uncountable programs

25- Select the correct statement.

1) Only simple programs are composed of finite strings.
2) Both simple and complicated programs are composed of finite strings over a finite alphabet.
3) Uncountable programs are undecidable problems.
4) Programs are countable when they are decidable problems.

Passage 2

24- A language being picked randomly · · · .

1) cannot be undecidable
2) is decidable if programs in that language are countable
3) is often undecidable because it is often complicated
4) is undecidable if it contains uncountable programs

25- Select the correct statement.

1) Only simple programs are composed of finite strings.
2) Both simple and complicated programs are composed of finite strings over a finite alphabet.
3) Uncountable programs are undecidable problems.
4) Programs are countable when they are decidable problems.

Passage 3

Why “Recursive”

Programmers today are familiar with recursive functions. Yet these recursive functions don’t seem to have anything to do with Turing machines that always halt. Worse, the opposite–nonrecursive or undecidable– refers to languages that cannot be recognized by any algorithm, yet we are accustomed to thinking of “nonrecursive” as referring to computations that are so simple there is no need for recursive function calls.

The term “recursive,” as a synonym for “decidable,” goes back to Mathematics as it existed prior to computers. Then formalisms for computation based on recursion (but not iteration or loops) were commonly used as a notion of computation. These notations, which we shall not cover here, had some of the flavor of computations in functional programming languages such as LISP or ML.

Passage 3

In that sense, to say a problem was “recursive” had the positive sense of “it is sufficiently simple that I can write a recursive function to solve it, and the function always finishes.” That is exactly the meaning carried by the term today, in connection with Turing machines.

The term “recursively enumerable” harks back to the same family of concepts. A function could list all the members of a language, in some order; that is, it could “enumerate” them. The languages that can have their members listed in some order are the same as the languages that are accepted by some TM, although that TM might run forever on inputs that it does not accept.

Passage 3

26- Choose the correct statement about “nonrecursive”.

1) Languages that are undecidable.
2) Any program that is not recursive.
3) Languages that cannot produce algorithms.
4) Simple algorithms that need not be stated recursively.

27- Before the invention of computers, mathematicians · · · .

1) used recursive functions formally
2) knew that recursive functions where decidable
3) devised recursive programs for undecidable problems
4) invented programming languages such as LISP

28- Any problem that can be · · · is called decidable.

1) stated formally
2) solved by terminating recursive function
3) considered as a language
4) given as an input to a Turing machine

Passage 3

26- Choose the correct statement about “nonrecursive”.

1) Languages that are undecidable.
2) Any program that is not recursive.
3) Languages that cannot produce algorithms.
4) Simple algorithms that need not be stated recursively.

27- Before the invention of computers, mathematicians · · · .

1) used recursive functions formally
2) knew that recursive functions where decidable
3) devised recursive programs for undecidable problems
4) invented programming languages such as LISP

28- Any problem that can be · · · is called decidable.

1) stated formally
2) solved by terminating recursive function
3) considered as a language
4) given as an input to a Turing machine

Passage 3

26- Choose the correct statement about “nonrecursive”.

1) Languages that are undecidable.
2) Any program that is not recursive.
3) Languages that cannot produce algorithms.
4) Simple algorithms that need not be stated recursively.

27- Before the invention of computers, mathematicians · · · .

1) used recursive functions formally
2) knew that recursive functions where decidable
3) devised recursive programs for undecidable problems
4) invented programming languages such as LISP

28- Any problem that can be · · · is called decidable.

1) stated formally
2) solved by terminating recursive function
3) considered as a language
4) given as an input to a Turing machine

Passage 3

29- Enumerable languages · · · .

1) are undecidable problems
2) are accepted by Turing machine
3) are decidable or undecidable
4) may not be accepted by a Turing m

30- The term decidable/undecidable is used for languages that · · · .

1) are enumerated / recursive
2) are nonrecursive / recursive
3) have formal / informal definitions
4) can be recognized / not recognized by an algorithm

Passage 3

29- Enumerable languages · · · .

1) are undecidable problems
2) are accepted by Turing machine
3) are decidable or undecidable
4) may not be accepted by a Turing m

30- The term decidable/undecidable is used for languages that · · · .

1) are enumerated / recursive
2) are nonrecursive / recursive
3) have formal / informal definitions
4) can be recognized / not recognized by an algorithm

END.