

GEM 1501 Problem Solving With Computers

Lecture 7:

Noncomputability and Undecidability

Martin Henz

Summary of Previous Lecture

- Growth of functions
- NP-complete problems
- Is P equal NP ?
- Even worse problems

Question

- Given a yes/no problem with an infinite set of inputs.
- Is there an algorithm (no matter how inefficient) that computes the correct answer for every input?
- Example: The set of inputs is the set of all graphs. The yes/no problem is the question whether the graph has a Hamiltonian path.

Example

- Tiles cannot be turned and the color of connected edges must match.
- Given a set of tile types.
- Can you use tiles of this type to cover any finite area of any size?

Example (continued)

- Is there a function in a given programming language, say Oz, to which you can pass a description of the tile types and which returns `true` if tiles of these types can cover any finite area, and `false` otherwise?
- ```
fun {Covering TileTypes}
 ...
end
{Browse {Covering [type(up:green down:blue left:green right:blue)
 type(up:red down:red left:blue right:green)]}}
```

# Answer: NO!

- There is no Oz program that does this job!
- There is no program in any language that does this job!
- We can prove that there is no program in any language on any computer of any speed in any amount of time that does this job!

# Snakes of Tiles

- If the snake can use only a finite space, the problem is clearly decidable.
- If the snake can use any space on the plane, the problem is decidable!
- If the snake can use only the upper half of the plane, the problem is undecidable!

# Word Correspondence

- Given two rows of words with the same number of words.
- Is there a sequence (of any length) of indices such taken the words in both rows in the sequence of indices spell out the same word?
- Undecidable!

# Program Behavior

- Many problems in computer science are undecidable.
- Examples:
  - Do programs terminate?
  - Do given program lines get executed?
  - Will given programs encounter a certain error (e.g. division by zero)?

# Proving Undecidability

- Consider the Halting Problem, i.e. the question whether a given program terminates on a given set of input or not.
- Assume that there is a program  $Q$  that decides the halting problem.
- Construct a new program  $S$  that applies  $Q$  to a given program using the program itself as input.  $S$  does not terminate if the answer is “yes” and it does terminate if the answer is “no”.
- Apply  $S$  to itself and derive a contradiction!