Lecture 11

The Halting Problem

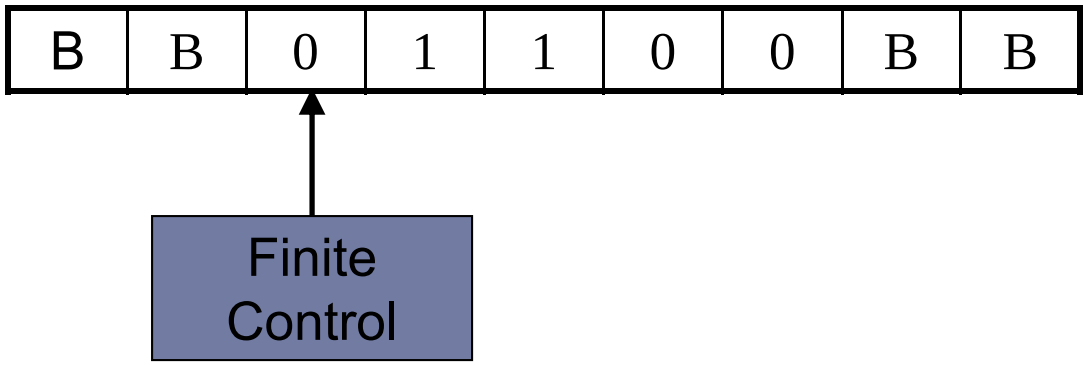
Zvonimir Rakamarić
University of Utah

Announcements

- Homework assignment due on Wed
- Any questions about the assignment?

Last Time

- Turing machine
  - Theoretical model of a computing machine
  - As powerful as any other computing device
  - http://www.youtube.com/watch?v=E3keLeMwfHY



This Time

- The Halting Problem

Limits of Turing Machines

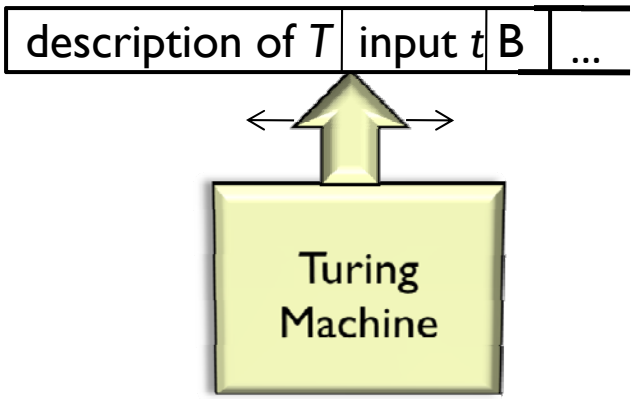
- There are limits to the power of TMs
  - A TM continues until it reaches accept state or reject state where it will halt
  - If it never reaches one, then it continues computing forever
- There exist problems that TMs cannot solve
  - These problems contain no effective procedure and no recursive computation exists
- The problems unsolvable by TMs are also unsolvable by any equivalent formal programming systems

Intro to the Halting Problem

- The best known problem that is unsolvable by a TM is the Halting Problem:
  - “Given an arbitrary Turing Machine T as input and equally arbitrary tape t, decide whether T halts on t.”
- Basically TM that takes a TM T as its input, and simulates T running on input t, and returns or decides whether or not T halts on t
- Can a TM accept a TM as input?
  - Universal Turing Machine

Universal Turing Machine

- Represents the set of all possible TMs, and all possible effective procedures
  - Takes input in the form (dT, t)
  - Mimics the actions of an arbitrary TM T by reading its description dT off the tape, and simulates its behavior on t
  - Produces the same result as T
- Simple TMs can also take descriptions of other TM as input



Proof of Undecidability

1. Prove that the following is undecidable
   - The problem of testing whether a Turing Machine M accepts a given input string w
   - Language ATM:
     {(M,w) | M is a TM and M accepts w}
2. Use the previous result to prove that the Halting Problem is undecidable

Recognizable vs Decidable

- Recognizable
  - If there is a TM M such that M halts in an accept state for an input x iff x is in the language
- Decidable
  - If there is a TM M such that M halts in an accept state for an input x iff x is in the language, and such that M halts in a reject state for an input x iff x is not in the language

Recognizable vs Decidable

- ATM is recognizable using the following TM
  - Simulate M on input w
  - If M enters its accept state, then accept; if M enters its reject state, then reject
- Does not decide since M might not halt on w

Step 1 of the Proof

- Prove that the following is undecidable
  - The problem of testing whether a Turing Machine accepts a given input string
  - Language ATM:
    {(M,w) | M is a TM and M accepts w}

Step 2 of the Proof

- Use the previous result to prove that the Halting Problem is undecidable

Halting Problem in a Programming Lang.

- Assume halting problem is decidable
  - We have a function `halts(f, i)` that returns true if function `f` halts on input `i`, and false otherwise
- Then, we define a function

```
test(f) {
  if (halts(f, f))
    while (true);
  else
    return;
}
```

- Invoke `test(test)`

Halting Problem in a Programming Lang.

```
if (halts(test, test))
  while (true);
else
  return;
```

- Two cases:
  - `halts(test, test)` returns true but test loops forever – contradiction
  - `halts(test, test)` returns false but test returns – contradiction
- Hence halting problem is undecidable

Consequences: Rice's Theorem

- Henry Gordon Rice, PhD thesis [1951]
- Essence
  - Any interesting question about the behavior of a program is undecidable
- Examples
  - Does this assertion always hold?
  - Is x always positive?
  - Which locations can pointer p point to?
  - What is a strong enough loop invariant?
  - …

Proof by Reduction from Halting Problem

- Assume `always_pos(f,x)` is decidable
- Reduce solving `halts` to solving `always_pos`

```
function solve_halts() {
   int x := 1;
   f(i);
   x := -1;
}
```

- Invoke `always_pos(solve_halts, x)`
  - If returns true then `f(i)` loops forever, otherwise `f(i)` halts
- Hence `always_pos(f, x)` is undecidable

Next Time

- Complexity