CS 103: Mathematical Foundations of Computing
Problem Set #9

Quang Nguyen

June 2, 2022

Due Friday, December 3 at 2:30 pm Pacific

Do not put your answer to Problem 5 in this file. You'll submit that separately. If you complete the Optional Fun Problem, you'll similarly submit that separately.

Some notation you might want for this problem set:

- The encoding of an object can be denoted $\langle M \rangle$.
- The complement of a language is denoted $\overline{L}$.

Problem One: Isn't Everything Undecidable?

i.

The decider does not take the function as an argument and the function is not self-referential.

ii.

The proof assumes, in the line "`return !willAccept(me, w);`", that the language which the function accepts and the one which the decider accepts are the same, which is not the case. In contrast, we, in context, have to set up a contradiction where the program has the property given by the decider if and only if it doesn't have the property given by the decider.

iii.

The proof assumes that the trickster still runs in case of syntax errors, which is not the case. If the trickster contains a syntax error, this leads to a compilation error, so we can't even run the given trickster, only can run the decider.

Problem Two: An Undecidable Problem

i. Fill in the blanks to Problem Two, part i. below.

- If `willReject(function, input)` returns true, then M rejects `w`.
- If `willReject(function, input)` returns false, then M does not reject `2`.

ii. Fill in the blank to Problem Two, part ii. below.

```
bool trickster(string input) {
    string me = /* source code of trickster */;
    return !willReject(me, input);
}
```

iii.

Theorem: $R_{TM} \notin R$

Proof: Assume for the sake of contradiction that $R_{TM} \in R$. Then there is a decider D for R_{TM}. We can represent D as a function

```
bool willReject(string function, string input);
```

that takes in the source code of a function `function` and a string `w`, then returns true if `function(w)` returns true and returns false otherwise.

Given this, consider the function `trickster`

```
bool trickster(string input) {
    string me = /* source code of trickster */;
    return !willReject(me, input);
}
```

Choose a string `w`. We consider two cases:

Case 1: `willReject(me, input)` returns true. Since `willReject` decides R_{TM}, this means that `trickster(w)` returns true. However, given how `trickster` is written, in this case `trickster(w)` returns false.

Case 2: `willReject(me, input)` returns false. Since `willReject` decides R_{TM}, this means that `trickster(w)` returns false. However, given how `trickster` is written, in this case `trickster(w)` returns true.

In both cases we reach a contradiction, so our assumption must have been wrong. Therefore, $R_{TM} \notin R$. ■

Problem Three: Password Checking

Theorem: The password checker problem is undecidable.
Proof: Assume for the sake of contradiction that the password checker problem is decidable. Then there is a decider D for the password checker problem. As the requirement, we can represent D as a function

```
bool isSecurePasswordChecker(string function);
```

that takes in the source code of a password checker function `function`, then returns true if `function` is secure and returns false otherwise.
Given this, consider the function `trickster`:

```
bool trickster(string input) {
    string me = /* source code of trickster */;

    if(isSecurePasswordChecker(me)) {
       return true;
    } else {
       return input == "iheartquokkas";
    }
}
```

We consider two cases:
Case 1: `isSecurePasswordChecker(me)` returns true. Since `isSecurePasswordChecker` is the decider, this means that `trickster` is secure. However, given how `trickster` is implemented, in this case, `trickster` vacuously returns true no matter what the input is, which is not secure.
Case 2: `isSecurePasswordChecker(me)` returns false. Since `isSecurePasswordChecker` is the decider, this means that `trickster` is not secure. However, given how `trickster` is implemented, in this case, `trickster` returns true only if the input is `iheartquokkas`, which is completely secure.
In both cases, we reached a contradiction, so our assumption must have been wrong. Therefore, the password checker problem is undecidable. This means that it's not possible to implement the `isSecurePasswordChecker` function so that it meets the above requirements, as required. ■

Problem Four: Double Verification

```
bool isInL(string w) {
    bool inL = false;
    for (string c: Σ*) {
        if (checkInL(w, c)) {
            inL = true;
            break;
        }
    }

    bool inLBar = false;
    for (string c: Σ*) {
        if (checkInLBar(w, c)) {
            inLBar = true;
            break;
        }
    }
    return inL && inLBar;
}
```

Congratulations on finishing the final Problem Set of the quarter!