cs3102: Theory of Computation

Class 20: Busy Beavers

Exam 2

• Out Thursday, due next Tuesday (2:01pm)
• Should not take more than two hours to complete (but don’t wait until 12:01pm Tuesday to start it!)
• Prepare (before class Thursday) one page of notes: you may use this as much as you want during the exam
• You may also use: course book, course handouts, your notes, other resources: but only for up to 30 minutes during exam

Exam 2: Main Topics

• Countable and uncountable infinite sets
• Turing machines:
  – Formal definition
  – Computing model
  – Robustness of TM model
• Church-Turing thesis and its implications
• Turing-recognizable, Turing-decidable classes and their properties
• Using reduction to prove a language is undecidable
• Everything from Exam 1 also

“Game Theory” Experiment

Number of “Exempt Me” messages received: 14

No exemptions on Exam 2. (Although some more opportunities in today’s class.)

Reduction Proofs

A reduces to B

means

Y can be used to make X that can decide A

If we already know A is undecidable, this proves B is undecidable.

Singleton Language

\[ \text{SINGLETON}_{TM} = \{ \langle M \rangle | M \text{ is a description of a TM and } |L(M)| = 1 \} \]
Proving Undecidability

1. Assume \( \text{SINGLETON}_{TM} \) is decidable.
2. Then, there exists a TM that decides it. We'll call it \( M_{\text{SINGLETON}} \).
3. Show how to use \( M_{\text{SINGLETON}} \) to build a machine that decides \( \text{HALTS}_{TM} \).
4. Contradiction: we know \( \text{HALTS}_{TM} \) is undecidable, so if we could use \( M_{\text{SINGLETON}} \) to construct a machine that decides it, \( M_{\text{SINGLETON}} \) must not exist.

Reducing \( \text{HALTS}_{TM} \) to \( \text{SINGLETON}_{TM} \)

\[ M_H((M, w)) = \text{A TM that constructs } M', \text{ a Turing Machine that:} \]

1. Checks if the input is 0101. If not, reject.
2. Erases the input and write \( w \) on the tape.
3. Simulates \( M \) except with all \( q_{\text{reject}} \)'s in the output of \( M \)'s \( \delta \) function replaced by \( q_{\text{accept}} \).

Then, it simulates \( M_{\text{SINGLETON}}(\langle M' \rangle) \) and outputs the result.

Reducing \( \text{HALTS}_{TM} \) to \( \text{WELL_TYPED}_{\text{Python}} \)

\[ M_H((M, w)) = \text{A TM that constructs } P, \text{ a Python program:} \]

\[
\text{simulate}\_\text{Turing}\_\text{machine}(M, w) \\
"\text{"hello" + 3}"
\]

Then, it simulates \( M_{\text{WELL_TYPED}_{\text{Python}}}(\langle P \rangle) \)
and outputs the result: the opposite.

Type Safety

\[ \text{WELL_TYPED}_{\text{Python}}(\langle P \rangle) = \{ P \text{ is a Python program that cannot produce a type error} \} \]

\[ \alpha() \quad \{"hello" + 3 \}
\]

Traceback (most recent call last):
  File "<pyshell#1>", line 1, in <module>
    "hello" + 3
TypeError: Can't convert 'int' object to str implicitly

Reducing \( \text{HALTS}_{TM} \) to \( \text{WELL_TYPED}_{\text{Python}} \)

\[ M_H((M, w)) = \text{A TM that constructs } P, \text{ a Python program:} \]

\[
\text{simulate}_\text{Turing}_\text{machine}(M, w) \\
"Hello" + 3
\]

Then, it simulates \( M_{\text{WELL_TYPED}_{\text{Python}}}(\langle P \rangle) \)
and outputs the result.
Well-Typed Java?

```java
public class Test {
    static public void main(String args[]) {
        System.out.println("Hello" + 3);
    }
}
```

```
> javac Test.java  
> java Test  
> Test.java:3: operator - cannot be applied to java.lang.String, int
```

Type Safety

\[ \text{WELLTYPED}_{\text{java}} (<P>) = \{ P \text{ is a Java program that does not use type casts or array assignments and } P \text{ never produces a run-time type error. } \} \]

This is decidable: your Java compiler should do this (and should always terminate)!

Reducing \( \text{HALTS}_{TM} \) to \( \text{WELLTYPED}_{\text{java}} \)?

\[ M_H((M, w)) = \text{TM that constructs } , \text{a Java program:} \]

```java
public class Test {
    static public void main(String args[]) {
        simulateTM(M, w);
        System.out.println("Hello" - 3);
    }
}
```

Then, it simulates \( M_{\text{WELLTYPED}_{\text{java}}}((P)) \) and outputs the result.

We claimed on the last slide that it is decidable! Something must be broken with this "proof".

Busy Beavers

The "Busy Beaver" Game

Design a (2-way infinite tape) Turing Machine that:
- Uses \( k \) symbols (e.g., "0" and "1")
- Starts with a tape of all "0"s (no blanks)
- Eventually halts (can’t run forever)
- Has \( n \) states (not counting \( q_{\text{Accept}} \) and \( q_{\text{Reject}} \))

![Tibor Radó, 1895-1965](image)

Goal is to run for as many steps as possible before eventually halting

Busy Beaver: \( N = 1 \)

\( BB(1, 2) = 1 \)

Most steps a 1-state machine that halts can make
$BB(2, 2) = ?$

Step 0

Step 1

Step 2

Step 3

Step 4

Step 5
Busy Beaver Numbers

\[ BB(1, 2) = 1 \]
\[ BB(2, 2) = 6 \]
\[ BB(3, 2) = 21 \quad \text{Proved by Lin and Rado, 1965} \]
\[ BB(4, 2) = 107 \]
\[ BB(5, 2) = \text{Unknown!} \]
\[ BB(6, 2) = \text{Unknown!} \]

Best so far is \( BB(5, 2) = 47,176,870 \)

Why is \( BB(5, 2) \) unknown?

Busy Bunny?

6-state machine found by Buntrock and Marxen, 2001

What is \( BB(6, 2) \)?

Busy Bunny?

Step 6:
Halted

Actually, it is known that \( BB(2, 2) = 6 \)

\[ BB(2, 2) \geq 6 \]

What is \( BB(6, 2) \)?

Busy Bunny?

6-state machine found by Buntrock and Marxen, 2001

Best found before 2001, only 925 digits!

http://drb9.drb.insel.de/~heiner/BB/index.html

(1730 digits)

Why is \( BB(5, 2) \) unknown?

Busy Bunny?

flickr: climbnh2003

Previous best: \( > 10^{6292} \) (discovered in 2007)
Previous best: \( > 10^{1730} \) (discovered in 2001, previous slide)
Previous best: \( > 10^{925} \) (discovered in August 2000)
How many (5, 2) TMs are there?

![Diagram of a Turing machine with states A, B, C, D, and E, and transitions between states.]

Why BB(5, 2) is Unknown

- Best found so far: 47,176,870
- There are about 40 (5,2) TMs that run for more than 47,176,870 steps without obviously repeating (but haven’t halted yet)
  - Probably, these machines never halt and \( BB(5,2) = 47176870 \)
  - But, its possible one of these machines eventually accepts!

What does this have to do with reviewing reduction proofs? (or: is BB a language?)

\[
L_{BB} = \{ (n, k, s) | n, k, s \in \mathcal{N} \text{ and } s \text{ is the maximum number of steps a Turing machine with } n \text{ non-final states and } k \text{ tape symbols can run before eventually halting.} \}
\]

\[<2, 2, 6> \in L_{BB} \]

Is \( L_{BB} \) Decidable?
Do we know this terminates?

Someone pointed out after class today that this is not quite correct! The problem is we’re simulating $M$ starting with $w$ on the tape, but the BB starts with a blank tape. To fix this, we need to create $M'$ that first writes $w$ on the tape and then simulates $M$.

$L_{BB}$ is undecidable. Proof by reducing $HALT_{TM}$ to $L_{BB}$. Assume $L_{BB}$ is decidable. Then, there is a TM $M_{BB}$ that decides $L_{BB}$. We can use $M_{BB}$ to construct $M_H$ that decides $HALT_{TM}$:

$$M_H(\langle M, w \rangle) =$$

1. Let $n = \text{the number of states in } M$.
2. Let $k = \text{the number of stack symbols in } M$.
3. Try $s = 1, 2, 3, \ldots$ until simulating $M_{BB}(n, k, s)$ accepts. Do we know this terminates?
4. Simulate $M$ on $w$ for up to $s$ steps. If it halts, accept. Otherwise, reject.

Busy Beaver Challenges

- Determine $BB(5, 2)$
- The standard Busy Beaver problem is defined for a doubly-infinite tape TM. For the one-way infinite tape TM, what is $BB(4, 2)$?

Charge

- Exam 2 will be handed out at end of class Thursday, due on Tuesday
- Prepare your Exam 2 note sheet before class Thursday
- Thursday’s office hours: 11am-1pm (not normal time)