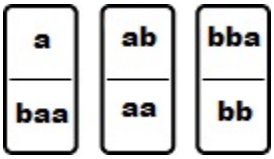
CSCI 270 Lecture 30: Extra Topics

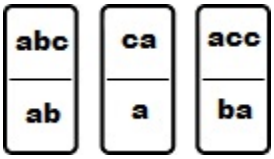
The Post Correspondence Problem

We are given a collection of dominoes, such as:



The goal is to make a list of these dominoes (repetitions permitted) so that the string on the top exactly matches the string on the bottom.

Is there a solution for the above example?



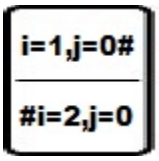
- Is there a solution for the above example?
- Can you design a brute-force algorithm to solve this problem?
- Are there any problems with your algorithm?

The proof that PCP is undecidable, like the Cook-Levin theorem, is outside the scope of this course. The high level idea of the proof, however, will follow.

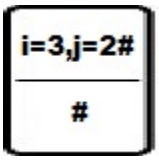
Halting Problem $\leq PCP$

You have a starting domino which has a starting symbol # on the top, and the bottom encapsulates the state of the algorithm at launch, including what line of code we're on, and what values the variables have. For example:

We then have other dominos that form an if-then statement. The top of the domino is the **if**: if we are on a certain line and the values of variables are certain things. The bottom of the domino is the **then**: then we go to a certain line and the values of the variables possibly change. For example:



There are dominos which allow you to catch up and finish whenever you reach a **return** statement which causes the program to terminate.



If there is a sequence of dominos where the top equals the bottom, we have effectively shown a path through our algorithm that causes termination. That is, our algorithm halts. If there is no such sequence of dominos, then our algorithm will never halt.

PSPACE

PSPACE is the set of problems which can be solved with a polynomial amount of space. NPSPACE is the set of problems which can be verified with a polynomial amount of space.

Here is what we know about the various complexity classes:

- P $\subseteq$ NP
- NP $\subseteq$ PSPACE
- PSPACE = NPSPACE
- PSPACE $\subseteq$ EXPTIME
- P $\neq$ EXPTIME

Most researchers believe: P $\subsetneq$ NP $\subsetneq$ PSPACE $\subsetneq$ EXPTIME.

A language B is **PSPACE-complete** when it satisfies both conditions:

- $B \in$ PSPACE
- $\forall A \in$ PSPACE, $A \leq_P B$

A **fully quantified Boolean formula** is a Boolean formula (such as $(x \vee y) \wedge (\overline{x} \vee \overline{y})$) which is preceded by quantifiers for all the variables.

An example quantified Boolean formula: $\phi = \forall x \exists y[(x \vee y) \wedge (\overline{x} \vee \overline{y})]$.

Is this formula true?

Determining whether a fully quantified Boolean formula is true is PSPACE-complete!