

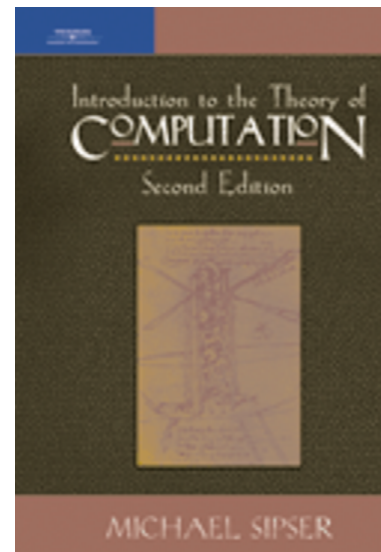
# Welcome - CSC 544

## *Theory of Computation*

Dr. Lutz Hamel

hamel@cs.uri.edu

Tyler Hall, Rm 251





# Optional Reference Material

---

Other books you might find useful/interesting to consult:

- *The Annotated Turing: A Guided Tour Through Alan Turing's Historic Paper on Computability and the Turing Machine*, Charles Petzold, Wiley, 2008.
- *Alan Turing: Life and Legacy of a Great Thinker*, Christof Teuscher (Editor), Springer, 2006.
- *The Universal Computer: The Road from Leibniz to Turing*, Martin Davis, W. W. Norton & Company, 2000.



# The *What* and the *Why*

**What:** The *Theory of Computation* is the formal (mathematical) investigation of models of computation.

**Why:** We investigate models of computation in order to understand the nature of computation without having to worry about specific hardware.

However, if the models reflect the major characteristics of the hardware implementation, then the laws of computation discovered in the models are also applicable to the actual hardware! Thus, the theoretical results are directly applicable to software design.

The nature of computing does not only involve limits of computability, but also problem complexity. We could be faced with a perfectly computable function (whatever that means at this point) but the time complexity of this function is such that the computation would never complete in a reasonable amount of time (whatever that means at this point).



# Models of Computation

---

1. Finite Automata
2. Turing Machines
3.  $\lambda$ -Calculus – invented by Alonzo Church to investigate the nature of computation
4. Recursive Functions – a term coined by Rózsa Péter a Hungarian mathematician and used by Kurt Gödel in some of his famous proofs
5. String Rewriting Systems – also called semi-Thue systems after Axel Thue a Norwegian mathematician

It is interesting to note that

1. all models of computation except the first one were invented separately by logicians in the early 1900's and have been shown to be equivalent, they each can simulate the other
2. Turing machines can simulate finite automata in a straightforward manner (but not vice-versa)



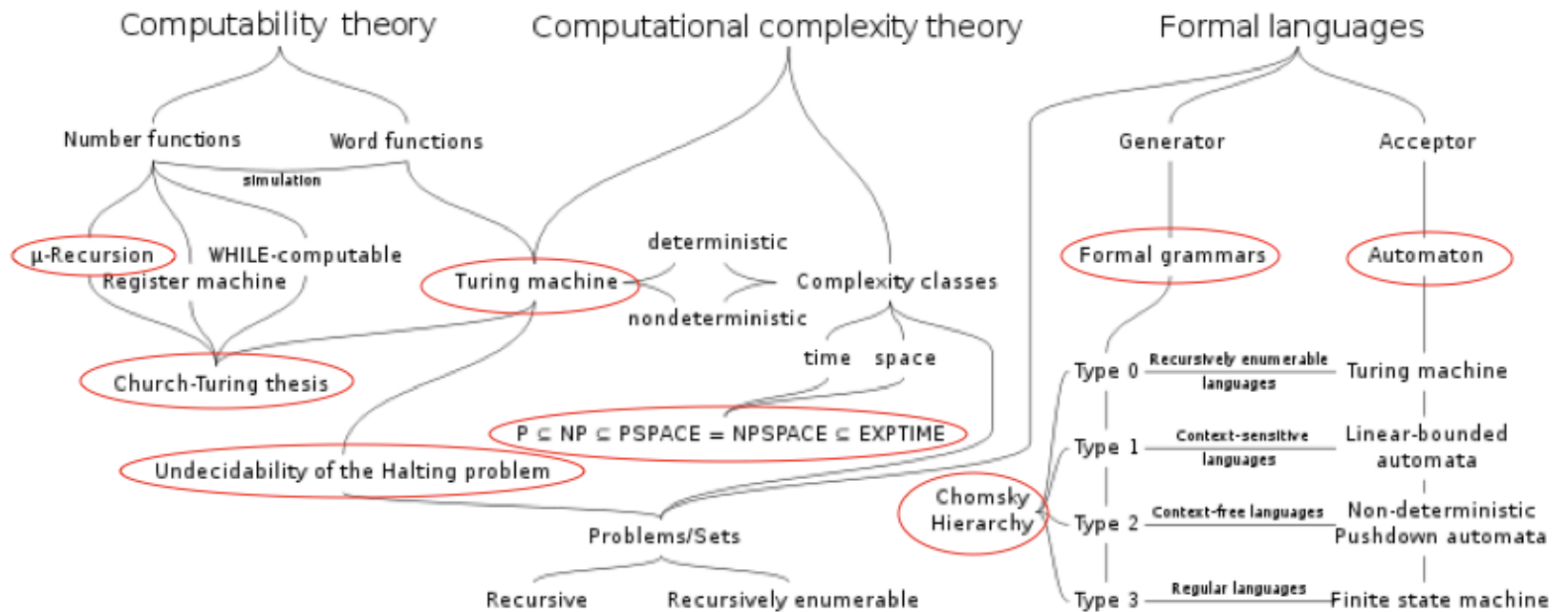
# Models of Computation

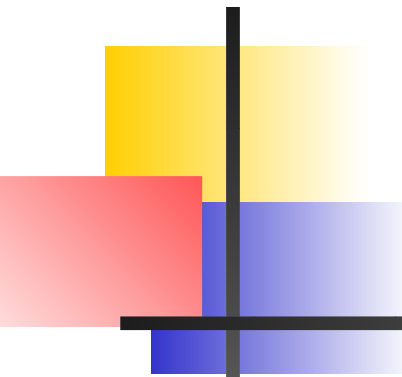
---

In this course we will start with a brief overview of results in automata theory and we will then continue to study the majority of computability and complexity theoretical results in the context of Turing machines.

We will also show the equivalence between the various models.

# Course Road Map





# Limits of Computer Science - The Halting Problem

Every discipline has its limits. Consider:

**Physics - The Perpetual Motion Machine** - construct a machine that once set in motion will produce useful work indefinitely without consuming any energy – Impossible, violates the second law of thermodynamics – entropy always increases.

**Mathematics - Squaring of the Circle** - with just a compass and a straightedge construct a square with the same area as a given circle – Impossible, given that the area of a circle is  $r^2\pi$  we need to construct a square with sides  $r\sqrt{\pi}$ , however,  $\pi$  is a transcendental number and therefore not constructable.

**Computer Science - The Halting Problem** - write a program that decides, given any program, whether that program will halt for all inputs – Impossible, such a general program cannot be constructed!



# Fundamental Concepts

---

**Algorithm** - What does it mean to “compute” and exactly “what can be computed”?

**Decidability** - Which problems can be solved using algorithms and which cannot?

**Complexity** - How much time and space does a particular problem solution require?

**Complexity Theory** - Can we organize problems according to the complexities?





# Mathematical Discipline

---

The theory of computation is ultimately a mathematical discipline. We rely on *proofs* to investigate truths and consequences of our model choices.

A proof is a *logical argument* that establishes the truth of a statement beyond any doubt. Here are common terms associated with proofs:

**Statement:** a sentence that is either true or false.

**Axiom:** a statement that is assumed to be universally true.

**Assumption:** a statement that is assumed to be true for the current proof at hand.

**Definition:** an unambiguous statement of the precise meaning of a word, symbol, phrase, or concept.

**Theorem:** a universal statement whose truth can be established using a chain of logical reasoning on the basis of certain assumptions that are explicitly given or implicit in the statement. Such a construction is called a *proof of a theorem*.



# Mathematical Discipline

**Lemma:** a statement in the context of the current proof whose truth can be established using a chain of logical reasoning (see theorem). A non-universal/auxiliary theorem to be used in the proof of another theorem.

**Corollary:** a theorem that follows *logically and easily* from another theorem already proved.



# The Structure of a Proof

---

Lemmas with their proofs.

Statement of the Theorem.

Proof of the theorem:

- axioms
- assumptions
- logical reasoning steps
- conclusion

# Proof Test Drive

**Definition:**  $q \in A \cap B$  iff  $q \in A$  and  $q \in B$ .

**Definition:**  $q \in A \cup B$  iff  $q \in A$  or  $q \in B$ .

**Theorem:** Given two sets  $A$  and  $B$ , then  $\overline{A \cup B} = \overline{A} \cap \overline{B}$ .

**Proof:** In order to show that the equivalence  $\overline{A \cup B} = \overline{A} \cap \overline{B}$  holds, we need to show that  $\overline{A \cup B} \subseteq \overline{A} \cap \overline{B}$  and  $\overline{A} \cap \overline{B} \subseteq \overline{A \cup B}$  holds.

(1)  $\overline{A \cup B} \subseteq \overline{A} \cap \overline{B}$ . Let some  $x \in \overline{A \cup B}$ , this implies that  $x \notin A \cup B$ . Therefore,  $x \notin A$  and  $x \notin B$ . From this it follows that  $x \in \overline{A}$  and  $x \in \overline{B}$ . Given the definition of intersection of sets, it follows that  $x \in \overline{A} \cap \overline{B}$ .

(2)  $\overline{A} \cap \overline{B} \subseteq \overline{A \cup B}$ . Let some  $x \in \overline{A} \cap \overline{B}$ , this implies  $x \in \overline{A}$  and  $x \in \overline{B}$  or  $x \notin A$  and  $x \notin B$ . It follows that  $x \notin A \cup B$  or expressed as the complement  $x \in \overline{A \cup B}$ .

This concludes the proof.  $\square$ .



# Types of Proofs

---

**Proof by Construction:** Many theorems posit that certain objects exist. One way to prove the existence of these objects is to construct them.

**Direct Proof:** A straight forward application of the modus ponens,

Given,  $A$  implies  $B$ .

Assume  $A$ .

Show that  $B$  holds.

**Proof by Contradiction:** (also called indirect proof),

Assume  $A$ .

Assume  $\neg B$ .

Show that  $\neg B$  implies  $\neg A$  (contradiction!)

Therefore,  $B$  has to hold.

**Proof by Induction:** Example, assuming a well-founded relation on some domain, such as the successor relation  $+1$  on the positive integers  $\mathcal{I}$ , and given some predicate  $P$ , then we can show that  $\forall i \in \mathcal{I}. P(i)$  iff  $P(1)$  and  $P(n+1)$  assuming that  $P(n)$  holds.