# Computability (Section 12.3)

# Computability

- Some problems cannot be solved by any machine/algorithm. To prove such statements we need to effectively describe all possible algorithms.
- *Example (Turing Machines):* Associate a Turing machine with each $n \in \mathbb{N}$ as follows:

$$\begin{aligned} n &\leftrightarrow b(n) \text{ (the binary representation of } n) \\ &\leftrightarrow a(b(n)) \text{ } (b(n) \text{ split into 7-bit ASCII blocks, w/ leading 0's} \\ &\leftrightarrow \textbf{if } a(b(n)) \text{ is the syntax of a TM} \\ &\qquad \textbf{then } a(b(n)) \textbf{ else } (0, a, a, S, \text{halt}) \textbf{ fi} \end{aligned}$$

- So we can effectively describe all possible Turing machines: $T_0, T_1, T_2, \ldots$

# Continued

- Of course, we could use the same technique to list all possible instances of any computational model. For example, we can effectively list all possible Simple programs and we can effectively list all possible partial recursive functions.
- If we want to use the Church-Turing thesis, then we can effectively list all possible solutions (e.g., Turing machines) to every intuitively computable problem.

# Decision Problems

- A decision problem is a problem that can be phrased as a yes/no question. Such a problem is decidable if an algorithm exists to answer yes or no to each instance of the problem. Otherwise it is undecidable. A decision problem is partially decidable if an algorithm exists to halt with the answer yes to yes-instances of the problem, but may run forever if the answer is no.
- Examples:
  - Is an arbitrary propositional wff a tautology?

# Decision Problems

- A decision problem is a problem that can be phrased as a yes/no question. Such a problem is decidable if an algorithm exists to answer yes or no to each instance of the problem. Otherwise it is undecidable. A decision problem is partially decidable if an algorithm exists to halt with the answer yes to yes-instances of the problem, but may run forever if the answer is no.
- Examples:
  - Is an arbitrary propositional wff a tautology? Decidable.
  - Is an arbitrary first-order wff valid?

# Decision Problems

- A decision problem is a problem that can be phrased as a yes/no question. Such a problem is decidable if an algorithm exists to answer yes or no to each instance of the problem. Otherwise it is undecidable. A decision problem is partially decidable if an algorithm exists to halt with the answer yes to yes-instances of the problem, but may run forever if the answer is no.
- Examples:
  - Is an arbitrary propositional wff a tautology? Decidable.
  - Is an arbitrary first-order wff valid? Undecidable and partially decidable.
  - Does a DFA accept infinitely many strings?

# Decision Problems

- A decision problem is a problem that can be phrased as a yes/no question. Such a problem is decidable if an algorithm exists to answer yes or no to each instance of the problem. Otherwise it is undecidable. A decision problem is partially decidable if an algorithm exists to halt with the answer yes to yes-instances of the problem, but may run forever if the answer is no.
- Examples:
  - Is an arbitrary propositional wff a tautology? Decidable.
  - Is an arbitrary first-order wff valid? Undecidable and partially decidable.
  - Does a DFA accept infinitely many strings? Decidable
  - Does a PDA accept a string $s$?

# Decision Problems

- A decision problem is a problem that can be phrased as a yes/no question. Such a problem is decidable if an algorithm exists to answer yes or no to each instance of the problem. Otherwise it is undecidable. A decision problem is partially decidable if an algorithm exists to halt with the answer yes to yes-instances of the problem, but may run forever if the answer is no.
- Examples:
  - Is an arbitrary propositional wff a tautology? Decidable.
  - Is an arbitrary first-order wff valid? Undecidable and partially decidable.
  - Does a DFA accept infinitely many strings? Decidable
  - Does a PDA accept a string $s$? Decidable

# The Halting Problem

- A total classic.
- The halting problem asks the following question:
  - Does an arbitrary program halt on an arbitrary input?
- The halting problem is partially decidable: Just execute the program on the input. If the program halts, output yes.

# Halting problem is undecidable

- The halting problem is undecidable.
- To prove this, we'll prove the following form for computable functions:
  - Is there a computable function that can decide whether an arbitrary computable function halts on an arbitrary input?
- *Proof:* Assume, BWOC, that the halting problem is decidable. We'll use the following effective listing of all computable functions of a single variable:
  - $f_0, f_1, f_2, \ldots$
- Define the partial function $g$ as:
  - $g(n) =$ **if** $f_n(n)$ halts **then** loop forever **else** 0 **fi**
- Observe that $g$ is computable, because $f_n(n)$ halts is computable by assumption. So there is some $k$ such that $g = f_k$. Now apply $g$ to $k$ to obtain a contradiction: $g(k)$ halts iff $g(k)$ does not halt. Therefore the halting problem is undecidable.