COMP340-08B Reasoning About Programs

36. Undecidability

Robi Malik

THE UNIVERSITY OF WAIKATO
DEPARTMENT OF COMPUTER SCIENCE
TARI ROROHIKO

Definition of Turing Machines

A **Turing machine**

$$T = (\Sigma, Q, \delta, q_i, q_m)$$

consists of

- tape alphabet Σ
- finite set of states Q
- transition function $\delta: Q \times \Sigma \to Q \times (\Sigma \cup \{\mathbf{L}, \mathbf{R}\})$
- initial state $q_i \in Q$
- accepting state $q_m \in Q$

In this Lecture …

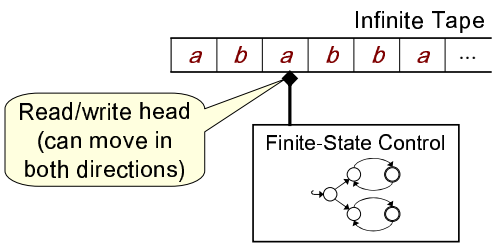
- Predicate logic formulas can be used to describe computation.
- In particular, every Turing Machine can be described using an appropriate predicate logic formula.
- This has the consequence that predicate logic cannot be decidable.

Executing Turing Machines

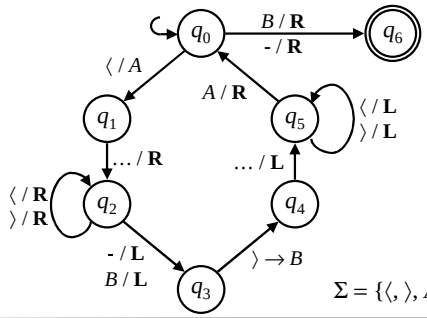
$$T = (\Sigma, Q, \delta, q_i, q_m)$$

- A Turing machine starts in q_i, with the first symbol on the tape as input.
- $\delta(q, \alpha) = (p, \beta)$ means:
  "In state q, with input α, change to state p, and replace α with β."
- $\delta(q, \alpha) = (p, \mathbf{R})$ means:
  "In state q, with input α, change to state p and move the read/write head right."

Recall: Turing Machines



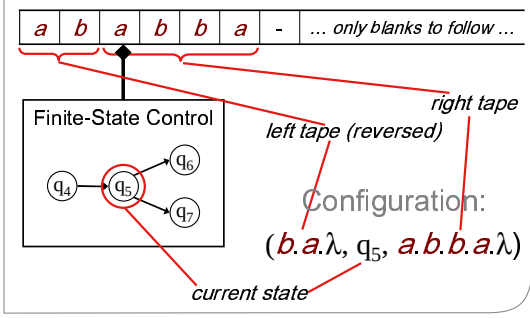
An Example Turing Machine



The Halting Problem

There is no effective algorithm that, given a Turing Machine **TM** and input tape, can determine in a finite number of computation steps whether the computation of **TM** with the given input tape will terminate or not.

Configuration Example



Describing Tape Contents in Logic

Tape contents can be described using list terms built up from:

- Constant symbol λ representing an empty list of symbols
- Binary concatenation operator "."

Examples:

- empty list: λ
- a one-element list: α.λ
- a two-element list: α.(β.λ) or α.β.λ

Turing Machine States in Logic

Define the 3-ary predicate symbol reach with the intended meaning that

$$\text{reach}(s, q, t)$$

is **true** if the configuration

$$(s, q, t)$$

is **reachable** in the Turing Machine under consideration.

Configurations

A state or **configuration** of a Turing Machine is conveniently represented as a triple

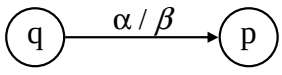
$$(s, q, t)$$

where

- s is a list representing the contents of the tape left of the read/write head, in reversed order;
- q is the current control state;
- t is a list representing the contents of the tape at the read/write head and to the right of it.

Translating Transitions into Logic

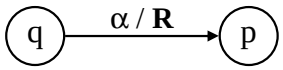
For each transition like this …



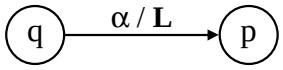
create a formula like this …

$$\forall s\ \forall t\ (\text{reach}(s, \text{q}, \alpha.t) \rightarrow \text{reach}(s, \text{p}, \beta.t))$$

Translating the other Transitions



$\forall s\, \forall t\, (\text{reach}(s, q, \alpha.t) \rightarrow \text{reach}(\alpha.s, p, t))$



$\forall s\, \forall t\, \forall u\, (\text{reach}(u.s, q, \alpha.t) \rightarrow \text{reach}(s, p, u.\alpha.t))$

Some more Pieces

Initial state …

$\text{reach}(\lambda, q_i, \text{tape}_i)$

q_i and tape_i are the initial state and initial tape contents.

Adding blanks at the end of the tape …

$\forall s\, \forall x\, (\text{reach}(s, x, \lambda) \rightarrow \text{reach}(s, x, \text{-}.\lambda))$

The ability to terminate …

$\exists t\, \text{reach}(\lambda, q_m, t)$

Putting it all Together

Given Turing Machine **TM** construct the following formula φ_{TM}:

$\text{reach}(\lambda, q_i, \text{tape}_i) \wedge$
$\forall s\, \forall t\, (\text{reach}(s, q, \alpha.t) \rightarrow \text{reach}(s, p, \beta.t)) \wedge$
$\ldots \wedge$
$\ldots \wedge$
} *… formulas for all other transitions …*
$\forall s\, \forall x\, (\text{reach}(s, x, \lambda) \rightarrow \text{reach}(s, x, \text{-}.\lambda)) \rightarrow$
$\exists t\, \text{reach}(\lambda, q_m, t)$

Observation

The formula φ_{TM} is **valid** if and only if the Turing Machine **TM** halts on the given input tape tape_i.

Why Predicate Logic is not Decidable

Assume there exists an algorithm **Decide** that, given a predicate logic formula φ, can effectively decide whether φ is valid or not.

This algorithm could be used to solve the halting problem for Turing Machines by the following algorithm:

Given a Turing Machine **TM** and input tape, construct the formula φ_{TM} and use algorithm **Decide** to determine whether it is valid. If φ_{TM} is valid then **TM** halts on the given input tape, otherwise it does not.

Since the halting problem for Turing Machines is known to be undecidable, we must conclude that the algorithm **Decide** cannot exist. □

Conclusion

- **Predicate logic is not decidable.**
  There is no algorithm that can effectively determine whether a given formula is valid or not.

But:

- **Predicate logic is semi-decidable.**
  There are sound and complete deductive systems that will find a proof for any given valid formula.