Towards a Mechanized Theory of Computation for Education

Tiago Cogumbreiro and Yannick Forster

20–23 June, 2022

University of Nantes, TYPES22

Motivation

- ***Context:*** Formal languages and automata (FLA) for undergraduate course
- ***Goal:*** help with math anxiety; rethink FLA to computer scientists

Contributions

- A simple and expressive calculus for decidability/computability
- Used in 3 editions of a course on FLA at UMass Boston
- Formalized multiple textbook results (e.g., halting problem)

Our technique

- Assumptions
- Base calculus
- Results

Math anxiety & Proofs

As a student:

- How do I know which theorems are available to use in a proof?
- How do I know if my steps are correct?
- How can I get more details about a particular proof?
- How can I study autonomously?

Context

- Undergraduate students (3rd, 4th year)
- Compulsory course on FLA/computability/decidability
- **No experience** with proof assistants

UMass Boston

Proof assistants in education

- (Math anxiety) Interactive mechanism allows students to step through a proof autonomously (independent study)
- (Math anxiety) Proof assistant turns a logic assignment into a programming assignment (great for computer science students)
- (Courseware) Machine checked proof scripts help automate grading

UMB-SVL Turing

- Open source software (MIT License)
- Regular languages results (eg, pumping lemma)
- Decidability, undecidability, recognizability results

https://gitlab.com/umb-svl/turing/

UMass Boston

Mechanization goals

Develop supplementary material to Michael Sipser's ***Introduction to the theory of computation***

- Coq formalism should be **similar to the textbook**
- **Simple proofs and techniques**, expect rudimentary knowledge of Coq (case analysis, induction, polymorphism, and logical connectives).
- **Include alternative proofs**, when there is pedagogical benefit

Use case: Theorem 4.11 A_{TM} is undecidable

AN UNDECIDABLE LANGUAGE

Now we are ready to prove Theorem 4.11, the undecidability of the language

$$A_{\mathsf{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}.$$

PROOF We assume that A_{TM} is decidable and obtain a contradiction. Suppose that H is a decider for A_{TM}. On input $\langle M, w\rangle$, where M is a TM and w is a string, H halts and accepts if M accepts w. Furthermore, H halts and rejects if M fails to accept w. In other words, we assume that H is a TM, where

$$H\big(\langle M, w\rangle\big) = \begin{cases} \mathit{accept} & \text{if } M \text{ accepts } w \\ \mathit{reject} & \text{if } M \text{ does not accept } w. \end{cases}$$

Now we construct a new Turing machine D with H as a subroutine. This new TM calls H to determine what M does when the input to M is its own description $\langle M\rangle$. Once D has determined this information, it does the opposite. That is, it rejects if M accepts and accepts if M does not accept. The following is a description of D.

D = "On input $\langle M\rangle$, where M is a TM:

1. Run H on input $\langle M, \langle M\rangle\rangle$.
2. Output the opposite of what H outputs. That is, if H accepts, *reject*; and if H rejects, *accept*."

UMass Boston

Formalizing the language

$$A_{\mathsf{TM}} = \{\langle M, w\rangle \mid M \text{ is a TM and } M \text{ accepts } w\}$$

```
Definition A_tm : input → Prop :=
  fun i ⇒
    let (M, w) := decode_mach_input i in
    Run (Call M w) true.
```

- A ***language*** is `input → Prop`, a function from an input to a proposition.
- `Run : prog → bool → Prop` runs `prog`ram (our calculus) and returns acceptance upon termination
- `decode_match_input` deconstruct an input as a pair `M` (Turing machine) and input `w`
- For any function `f:input → prog` there exist a machine `M` that computes `f` (**Axiom**)

Formalizing "high-level descriptions"

A calculus to invoke and compose abstract Turing machines

$D =$ "On input $\langle M \rangle$, where M is a TM:

1. Run H on input $\langle M, \langle M \rangle \rangle$.
2. Output the opposite of what H outputs. That is, if H accepts, *reject*; and if H rejects, *accept*."

```
Definition D (H:input → prog) : input → prog :=
  fun (w:input) ⇒ (* w = <M> and decode_mach w = M *)
    mlet b ← H <[decode_mach w, w]> in (* Run H on input <M, <M>> *)
    if b then Ret false   (* If H accepts, reject *)
         else Ret true    (* If H rejects, accept *)
  .
```

UMass Boston

Syntax

$$p \ ::= \ \texttt{mlet}\ x = p\ \texttt{in}\ p \mid \texttt{call}\ M\ i \mid \texttt{return}\ b \qquad \text{where } b \in \{\top, \bot\}$$

Semantics

$$\frac{}{\texttt{return}\ b \Downarrow b} \qquad \frac{M \text{ accepts } i}{\texttt{call}\ M\ i \Downarrow \top}$$

$$\frac{M \text{ rejects } i}{\texttt{call}\ M\ i \Downarrow \bot} \qquad \frac{p \Downarrow b \qquad p'[x := b] \Downarrow b'}{\texttt{mlet}\ x = p\ \texttt{in}\ p' \Downarrow b'}$$

We embed Coq functions in our High-level descriptions

UMass Boston

Results

- $A_{TM}, HALT_{TM} = \{\langle M \rangle \mid M \text{ halts}\}$, and $EQ_{TM} = \{\langle M_1, M_2 \rangle \mid \forall i, M_1 \text{ accepts } i \iff M_2 \text{ accepts } i\}$ are undecidable
- A is decidable iff A is recognizable and co-recognizable
- EQ_{TM} is neither recognizable nor co-recognizable
- Rice's Theorem (proved by Kleopatra Gjini, ***undergrad research project***)
- Results include direct proofs and proofs using map-reducibility

Chapter 1: `Regular.v`

- [x] Theorem 1.70: Pumping lemma for regular languages
- [x] Example 1.73: $\{0^n1^n \mid n \geq 0\}$ is not regular

Chapter 4: `LangDec.v`

- [x] Theorem 4.11: A_{TM} is undecidable.
- [x] Corollary 4.18: Some languages are not recognizable.
- [x] Theorem 4.22: L is decidable iff L is recognizable and co-recognizable
- [x] Theorem 4.23: $\overline{A}_{\text{TM}}$ is not recognizable.

Chapter 5: `LangRed.v`

- [x] Theorem 5.1: $HALT_{\text{TM}}$ is undecidable.
- [x] Theorem 5.2: E_{TM} is undecidable.
- [x] Theorem 5.4: EQ_{TM} is undedicable.
- [x] Theorem 5.22: If $A \leq_{\text{m}} B$ and A decidable, then B decidable.
- [x] Theorem 5.28: If $A \leq_{\text{m}} B$ and A recognizable, then B recognizable.
- [x] Corollary 5.29: If $A \leq_{\text{m}} B$ and B is undecidable, then A is undecidable.
- [x] Corollary 5.30: EQ_{TM} unrecognizable and co-unrecognizable.

Future Work

- **Consistency of axioms:** instantiate our theory with one of the models of the ***Coq library of undecidable problems*** [CoqPL'20]
- **Report on education insights:** How to teach effectively with Proof assistants
  - step-by-step evaluation in proofs (understanding `simpl`)
  - induction principles and recursive types
  - brute forcing solutions

Assumptions

- Theory parameterized by input type, Turing machine type, and Turing machine ***deterministic*** semantics
- Any Turing machine either accepts, rejects, or neither (eg, loops)
- For any Turing function f there exists a machine M that computes f (definable Coq functions are computable, Church's Thesis)

UMass Boston