Proofs

Mark Greenstreet, CpSc 421, Term 1, 2006/07

- Proposistional Logic
- Number Theory
  - Proofs are decidable.
  - Theorems are not.

Proof Rules for Proposistions

	Rule	Name
$0.$	$\dfrac{\text{Given } p}{\therefore \quad p}$	Hypothesis
$1.$	$\dfrac{p}{\therefore \quad p \vee q}$	Disjunctive Addition
$2.$	$\dfrac{p \vee q}{\therefore \quad q \vee v}$	Commutativity of Disjunction
$3.$	$\dfrac{p \vee q, \qquad \neg p}{\therefore \quad q}$	Disjunctive Simplification
$4.$	$\dfrac{p \wedge q}{\therefore \quad p}$	Conjunctive Simplification
$5.$	$\dfrac{p \wedge q}{\therefore \quad q \wedge p}$	Commutativity of Conjunction

More Proof Rules

	Rule	Name
6.	$p, \quad q$ $\therefore \quad p$	Conjunction
7.	$p, \quad p \Rightarrow q$ $\therefore \quad q$	Modus Ponens
8.	$\neg q, \quad p \Rightarrow q$ $\therefore \quad \neg p$	Modus Tollens
9.	$p \Rightarrow q, \quad q \Rightarrow r$ $\therefore \quad p \Rightarrow r$	Transitivity of Implication
10.	$p \vee qigap \not{p} \vee r$ $\therefore \quad q \vee r$	Resolution

A Simple Proof

- Given: $\neg p \wedge q$, $r \Rightarrow p$, $\neg r \Rightarrow s$, $s \Rightarrow t$.
- Prove: t.

Step		Justification	Stringification
1.	$\neg p \wedge q$	Hypothesis 1	$\#\neg p \wedge q, 0, 1$
2.	$\neg p$	Conjunctive Simplification	$\#\neg p, 4, 1$
3.	$r \Rightarrow p$	Hypothesis 2	$\#r \Rightarrow p, 0, 2$
4.	$\neg r$	Modus tollens, steps 2 & 3	$\#\neg r, 8, 2, 3$
5.	$\neg r \Rightarrow s$	Hypothesis 3	$\#\neg r \Rightarrow s, 0, 3$
6.	s	Modus ponens, steps 4 & 5	$\#s, 7, 4, 5$
7.	$s \Rightarrow t$	Hypothesis 4	$\#s \Rightarrow t, 0, 4$
8.	t	Modus ponens, steps 6 & 7	$\#t, 7, 6, 7$

A Grammar for Proofs

- Let $\Sigma = \{\texttt{0}, \texttt{1}, \texttt{v}, \texttt{\#}, \texttt{,}, \wedge, \vee, \neg, \Rightarrow, \texttt{(}, \texttt{)}\}$.
- The grammar:

$$
\begin{array}{rcl}
\mathit{Proof} & \rightarrow & \mathit{Hypotheses}\ \texttt{\#}\ \mathit{ProofSteps}\ \texttt{\#}\ \mathit{Conclusion} \\
\mathit{Hypotheses} & \rightarrow & \epsilon \mid \mathit{HypothesisList} \\
\mathit{HypothesisList} & \rightarrow & \mathit{Hypothesis} \mid \mathit{HypothesisList}\ \texttt{,}\ \mathit{Hypothesis} \\
\mathit{Hypothesis} & \rightarrow & \mathit{Proposition} \\
\mathit{Proposition} & \rightarrow & \mathit{Prop1} \mid \mathit{Prop1} \Rightarrow \mathit{Proposition} \\
\mathit{Prop1} & \rightarrow & \mathit{Prop2} \mid \mathit{Prop1} \vee \mathit{Prop2} \\
\mathit{Prop2} & \rightarrow & \mathit{Prop3} \mid \mathit{Prop2} \wedge \mathit{Prop3} \\
\mathit{Prop3} & \rightarrow & \mathit{Prop4} \mid \neg\ \mathit{Prop3} \\
\mathit{Prop4} & \rightarrow & \mathit{Variable} \mid \texttt{(}\ \mathit{Proposition}\ \texttt{)} \\
\mathit{Variable} & \rightarrow & \texttt{v}\ \mathit{Number} \\
\mathit{Number} & \rightarrow & \mathit{Digit} \mid \mathit{Number}\ \mathit{Digit} \\
\mathit{Digit} & \rightarrow & \texttt{0} \mid \texttt{1}
\end{array}
$$

Proof Grammar (cont)

- The rest of the grammar

$$\begin{array}{rcl} \mathit{ProofSteps} & \rightarrow & \mathit{ProofStep} \mid \mathit{ProofSteps} \mid \#\mathit{ProofStep} \\ \mathit{ProofStep} & \rightarrow & \mathit{Proposition}\ \texttt{,}\ \mathit{ProofRule}\ \mathit{RuleArgs} \\ \mathit{ProofRule} & \rightarrow & \mathit{Number} \\ \mathit{RuleArgs} & \rightarrow & \epsilon \mid \texttt{,}\ \mathit{Number}\ \mathit{RuleArgs} \\ \mathit{Conclusion} & \rightarrow & \mathit{Proposition} \end{array}$$

- Example, the proof from slide 4. Let $p \rightarrow \texttt{v000}$, $q \rightarrow \texttt{v001}$, $r \rightarrow \texttt{v010}$, $s \rightarrow \texttt{v011}$, $t \rightarrow \texttt{v100}$. The string corresponding to the proof from slide 4 is:

  $\neg\texttt{v000} \wedge \texttt{v001}\texttt{,}\texttt{v010} \Rightarrow \texttt{v000}, \neg\texttt{v010} \Rightarrow \texttt{v011}, \texttt{v011} \Rightarrow \texttt{v100}$
  $\#\neg\texttt{v000} \wedge \texttt{v001}, 0, 1\#\neg\texttt{v000}, 4, 1\#\texttt{v010} \Rightarrow \texttt{v000}, 0, 2\#\neg\texttt{v010}, 8, 2, 3$
  $\#\neg\texttt{v010} \Rightarrow \texttt{v011}, 0, 3\#\texttt{v011}, 7, 4, 5\#\texttt{v011} \Rightarrow \texttt{v100}, 0, 4\#\texttt{v100}, 7, 6, 7$
  $\#\texttt{v100}$

A Language for Proofs

- Let $P = \{w \mid w \text{ in a valid proof}\}$.
- P is Turing-decidable.
  Proof: construct a Turing machine, M_P that on input w:
  1. M_P first makes sure that w is generated by the grammar given on slides 5 and 6.
  2. For each $ProofStep$, $Proposition$, $ProofRule\ RuleArgs$, in w:
     A. M_P makes sure that each argument to the rule refers to a hypothesis or a previous proof-step.
     B. M_P applies the proof rule with the given arguments and makes sure that the result matches the proposition give from the proof step.
  3. M_P makes sure that the $Conclusion$ matches the $Proposition$ of the final $ProofStep$.

A Language for Theorems

- Let
  $T = \{Hypotheses\#Conclusion \mid \exists u.\ Hypotheses\#u\#Conclusion \in P\}$.
- T is Turing-Decidable.
  - Each propositional variable can be either true or false.
  - Just try all combinations and make sure that the claim holds
- This was easy because our language was so simple.
- We can add universal and existential quantifiers, and the resulting theory is still decidable. Again, for any given formula, there are only a finite number of combinations that the decider Turing machine needs to check.

Another Decidable Theory

- Add variables that are quantified over the natural numbers, +, <, =, and >.
- We can define rules for proofs and a new language of proofs, $P_{\mathbb{N},+}$ and a new language of theorems, $T_{\mathbb{N},+}$.
- $P_{\mathbb{N},+}$ is decidable. There are a few more proof rules, but the basic approach remains the same.
- $T_{\mathbb{N},+}$ is decidable.
  - We can show this by building a clever DFA for addition (remember the DFAs that check binary addition?).
  - We use an NFA to verify existentially quantified variables.
  - We use language complement and an NFA to verify universally quantified variables.
  - The details are in Sipser (see theorem 6.12).

Natural Numbers with $+$ and $*$

- Add $*$.
- Note that we can get subtraction, division, and mod using quantifiers:
  - $\exists q.\ (q * x \leq y) \wedge ((q+1) * x > y) \wedge \varphi$ sets q to $\text{div}(y, x) = \lfloor y/x \rfloor$ in formula φ.
  - $\exists r.\ (y = x * \text{div}(y, x) + r) \wedge \varphi$ sets r to $\text{mod}(y, x)$ in formula φ.
  - $\exists d.\ (y = x + d) \wedge \varphi$ sets d to $y - x$ in formula φ.
- We can define rules for proofs and a new language of proofs, $P_{\mathbb{N},+,*}$ and a new language of theorems, $T_{\mathbb{N},+,*}$.
- $P_{\mathbb{N},+,*}$ is decidable. There are a few more proof rules, but the basic approach remains the same.
- $T_{\mathbb{N},+,*}$ is NOT decidable.

Simulating a Stack with $\mathbb{N}$, $+$, and $-$

- Let $K = |\Gamma| + 1$ where Γ is the stack alphabet.
- $S' \leftarrow \text{push}(S, c)$ becomes $\exists S'.\ (S' = K * S + c)$.
- $(S', c) \leftarrow \text{pop}(S)$ becomes $(S' = \text{div}(S, K)) \wedge (c = \text{mod}(S, K))$.
- Examples:
  - Let $\Gamma = \{1, 2, 3, 4, 5, 6, 7, 8, 9\}$.
  - Let

$$\begin{array}{rclcr}
S_0 & = & 0, & & \\
S_1 & = & \text{push}(S_0, 3) & = & 3 \\
S_2 & = & \text{push}(S_1, 8) & = & 38 \\
S_3 & = & \text{push}(S_2, 4) & = & 382 \\
(S_4, c) & = & \text{pop}(S_3) & = & (38, 2) \\
\ldots & & & &
\end{array}$$

an empty initial stack

- Note that we can represent strings with integers and manipulate them using $*$, div, and mod in the same way as we represented a stack.

An Undecidable Formula

- Given M, the integer corresponding to the string that describes some Turing machine, and w, the integer corresponding to a string that is the input for that Turing machine, we'll write define function $H_M(w, n)$ which returns 1 if M halts after at most n moves, and 0 otherwise.
- We do this by simulating M using a 2PDA
  - We implement the stacks for the PDAs using integers as described above.
  - We simluate M with a recursive function that returns 1 if it is in state q_{accept} or q_{reject} and returns the result of a recursive call with arguments corresponding to the next state of M otherwise.
- M halts on input w iff $\exists n.\ H_M(w, n) = 1$. Thus, to decide if M halts on input w, ask if $\exists n.\ H_M(w, n) = 1$ is a theorem.

The Details

- Defining the H_M and F_M functions:

$$
\begin{aligned}
&F_M(\mathit{left}, \mathit{right}, q, n) = \\
&\quad \texttt{if}((q = q_{accept}) \vee (q = q_{reject}))\ \texttt{then}\ 1 \\
&\quad \texttt{else if}(n == 0)\ \texttt{then}\ 0 \\
&\quad \texttt{else let}\ c = \text{mod}(\mathit{right}, K)\ \texttt{in} \\
&\qquad \texttt{let}\ (q', c', \mathit{dir}) = \delta_M(q, c)\ \texttt{in} \\
&\qquad\quad \texttt{if}(\mathit{dir} = L)\ F_M(\text{div}(\mathit{left}, K), K * (\mathit{right} - c + c') + \text{mod}(\mathit{left}, K), q', n - 1) \\
&\qquad\quad \texttt{else}\ F_M(K * \mathit{left} + c', \text{div}(\mathit{right}, K), q', n - 1) \\
&\quad H_M(w, n) = F_M(0, w, q_0, n)
\end{aligned}
$$

- Note: I'm assuming that the blank symbol is encoded with 0. This allows there to be an infinite number of blanks to the right of w. This simulation corresponds to a machine with a two-way infinite tape – that was easier.

A Final Remark

- $T_{\mathbb{N},+,*}$ is Turing-recognizable.
- Proof (sketch):
  - A TM can enumerate all strings in lexigraphical order and check each one to see if it is a proof for the proposed theorem.
  - If the proposed theorem is a theorem, then it has a proof, and there is some shortest proof. When the TM encounters this proof, it accepts.
  - If the proposed theorem is not a theorem, then it has a no proof, and this TM will loop.
  - $\therefore T_{\mathbb{N},+,*}$ is Turing-recognizable as claimed.

Reading List:

- Today: Sipser, 6.1
- Nov. 24: Sipser, 6.2
- Nov. 27: Everything else about complexity theory.
- Nov. 29: The GHz race is over, and what it means for you.
- Dec. 1: Surprise (?)