Lecture 18: The Logic of Circuits - I

1 Reachability and Expressibility

Imagine a dedicated traveler wants to ask the Travelocity database if it's possible to travel from Houston to Timbuktu. This is a reachability query on a graph.

Given a Graph $G = (V, E)$ we define paths and reachability as follows:

Definition 1. *A* path *from $u \in V$ to $v \in V$ is a sequence $u_0, \ldots, u_k$ of $k \geq 0$ nodes such that $u_0 = u$, $u_k = v$, and $(u_i, u_{i+1}) \in E$ for $0 \leq i \leq k-1$.*

Definition 2. *A vertex x is* reachable *from a vertex y if there exists a path from x to y. We write $reach(x, y)$.*

We would like to express $reach(x, y)$ in first-order logic. Consider:

- $Reach_0(x, y) = E(x, y)$
- $Reach_{i+1}(x, y) = (\exists z)(E(x, z) \wedge Reach_i(z, y))$

So we can define the query $Reach_i(x, y)$ for all $i \geq 0$, but can we define it without fixing i? This requires us to study the expressive power of first-order logic. In fact, we will show that reachability is not expressible in first-order logic.

2 Transducers and Circuits

Logic has emerged as an important field because of its great modelling power. We now see one example: modelling sequential circuit. First, we define what it is:

2.1 Transducers

If we bound the amount of memory Turing machines have (no infinite tape), and focus on interaction, we get the universal model for interactive computation.

Definition 3. *A* **transducer** *is a tuple $T = \langle \Sigma, \Delta, S, s_0, \rho, \alpha \rangle$, where:*

- *Σ is a finite input alphabet,*
- *Δ is a finite output alphabet,*

- *S is a finite state set,*
- *$s_0 \in S$ is a start state,*
- *$\rho : S \times \Sigma \to S$ is a tarnsition function, and*
- *$\alpha : S \to \Delta$ is an output function.*

When given an infinite input stream $a_0, a_1, \ldots$, the transducer produces an infinite run $s_0, s_1, \ldots$, where $s_{i+1} = \rho(s_i, a_i)$, and an infinite output stream $c_0, c_1, \ldots$, where $c_i = \alpha(s_i)$.

A transducer can be viewed as a generalization of a DFA (deterministic finite automaton). A DFA $T = \langle \Sigma, S, s_0, \rho, F \rangle$ can be viewed as a trransducer where the output alphabet is $\Delta = \{0, 1\}$, and the output function is simply the characteristic function of F.

2.2 Sequential Circuits

A *sequential circuit* is an implementation of a transducer in terms of Boolean elements.

Definition 4. *A* **sequential circuit** *is a tuple $C = \langle I, O, R, Out, Trans, Start \rangle$, where:*

- *I is a finite set of Boolean input signals,*
- *O is a finite set of Boolean output signals*
- *R is a finite set of Boolean registers,*
- *$Trans : 2^I \times 2^R \to 2^R$ is the transition function,*
- *$Out : 2^R \to 2^O$ is the output function*
- *$Start \in 2^R$ is the starting state*

The corresponding transducer then is $T_C = \langle 2^I, 2^O, 2^R, Start, Trans, Out \rangle$.

Note that we can describe $Trans$ (transition function) as a sequence of functions $Trans_i \in 2^I \times 2^R \to \{0, 1\}$. Each such function is the transition function for one register. Similarly, we can describe Out as a sequence of functions $Out_i \in 2^I \times 2^R \to \{0, 1\}$.

We now want to model sequential circuits as relational models. The first step to modelling sequential circuits as relational structures is to pick the domain and vocabulary. Here we choose to pick our domain to be the set of states, that is $D = 2^I \times 2^R$. We also define the vocabulary: I, R, O as unary relations (for $i \in I$, $i(x) = true$ if high or $false$ if low). For the behavior of the state, define a binary translation which is a transition relation: $E^{(x)}$

$$
\begin{gathered}
A_c = (2^R \times 2^I, E^A, i_1^A, \ldots, {r_1}^A, \ldots, {o_1}^A, \ldots,) \\
E^A((\overline{r}, \overline{i}), (\overline{r}', \overline{i}')) \leftrightarrow Trans(\overline{i}, \overline{r}) = \overline{r}' \\
{i_j}^A(\overline{r}, \overline{i}) \leftrightarrow i_j = 1 \\
{r_j}^A(\overline{r}, \overline{i}) \leftrightarrow r_j = 1 \\
{o_j}^A(\overline{r}, \overline{i}) \leftrightarrow out(\overline{i}, \overline{r})|_j = 1
\end{gathered}
$$