Lecture 19: The Logic of Circuits, II

1 Review: Sequential Circuits

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

Definition 1. *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 circuits, use a binary translation which is the transition relation: $E^{(2)}$

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

2 Properties

Now that we have a mathematical model, we want to describe *designer intent* as *formal requirements*. The most common requirement we have is *mutual exclusion*, e.g., do not allow two processes to write to the same register at the same time). This is a *safety property*, which says that "bad things should not happen". For example, we can assert: r_j and r_k cannot be high at the same time, which leads us to write:

$$(\forall x)[\neg(r_j(x) \wedge r_k(x))]$$

However, this is not correct as the universal quantifier ranges over too many sates. We say that $(\overline{i_k}, \overline{r_k})$ is *reachable* if it is reachable from an Initial state. Then we can write:

$$(\forall x)[reachable(x) \rightarrow \neg(r_j(x) \wedge r_k(x))]$$

However, *reachable* is not included in our vocabulary, so we need to express reachability in first-order logic. As we shall see, however, reachability is not expressible in first-order logic. So this way of modeling circuits is not conducive to expressing properties of computation of circuits.

3 Traces

Instead of modeling circuits as relational circuits, we can model traces.

Definition 2. *A* **trace** *of a sequential circuit* $C = \langle I, O, R, Out, Trans, Start\rangle$ *is a sequence* $(\bar{i_0}, \bar{r_0}, \bar{o_0})(\bar{i_1}, \bar{r_1}, \bar{o_1}), ldots, (\bar{i_k}, \bar{r_k}, \bar{o_k})$ *of elements in* $2^I \times 2^R \times 2^O$ *such that:*

1. Initiation*:* $\bar{r}_0 = start$ *(Circuit starts properly)*
2. Conseqution*:* $\bar{r}_{j+1} = Trans(\bar{i}_j, \bar{r}_j)$ *(Circuit transitions property)*
3. Output*:* $\bar{o}_j = Out(\bar{i}_j, \bar{r}_j)$ *(Circuit outputs properly)*

We now model traces by relational structures. Again we use I, R, O as unary relation symbols, but instead of using a transition relation E, we use $<$ to denote the passage of time. A *trace structure* is then a relational structure:

$$A = (N, <^A, i_1^A, \ldots, {r_1}^A, \ldots, {o_1}^A, \ldots,).$$

where $<^A$ is the standard order on the natural numbers, and the other unary relations are arbitrary. Such a structure describe an arbitrary trace.

We can now describe properties of traces. For example, we can assert: r_j and r_k cannot be high at the same time, which we express by:

$$(\forall x)[\neg(r_j(x) \wedge r_k(x))]$$

This is now correct, since we are talking about all points in the e, not all states of a cicruit.

Consider the property: r_1 must toggle in every cycle. We express it using the formula:

$$(\forall x)[r_j(x) \leftrightarrow \neg(r_j(x+1))]$$

Here we used $x+1$ as an abbreviation. Instead of saying $y = x+1$, we write $(y > x) \wedge (\forall z)(\neg(y > z \wedge z > x)$.

Consider the following property: r_j being high designates a request, which must be granted in the future, designated by r_k being high.

$$(\forall x)[r_j(x) \rightarrow (\exists y)(y > x \wedge r_k(y))]$$

Such a property is called a *liveness* property, which says that "good things must happen".

4 Decision Problems

Reasoning about traces raises two problems:

1. Satisfiability: Given a formula φ over traces, is it satisfiabile?
2. Model Checking: Given a sequential circuit C and formula φ over traces, does C satisfy φ? That is, does $Traces(C) \subseteq models(\varphi)$ hold?

Both problems are decidable in NonelementaryTime, which is TIME($f(n,n)$), where $f(0,n) = n$, and $f_{i+1}(n) = 2^{f(i,n)}$. This function describes an unbounded tower of exponentials, which grows faster than every bounded tower of exponential. This is both an upper bound and a lower bound for this problem. So this problem is decidable, but with a horrible time bound. We must remember, however, that these are worst-case complexity bounds, which may be too pessimistic.