Program Synthesis

William Schultz

October 26, 2022

- The **verification problem**: given system M and spec φ, check that $M \models \varphi$.
- The **synthesis problem**: given spec φ, find M such that $M \models \varphi$.

Deductive Synthesis

The *deductive approach* [Manna and Waldinger, 1980] tries to synthesize an input/output program by extracting it from a realizability proof.

Temporal Synthesis

Temporal synthesis considers specifications given in the form of LTL (or CTL), for example. Initial approach was to use satisfiability of a temporal formula as a way to derive M [Clarke and Emerson, 1982]. See also [Manna and Wolper, 1984].

In [Clarke and Emerson, 1982] they consider concurrent systems consisting of a finite number of fixed processes $P_1, \ldots, P_m$ running in parallel. They treat parallelism in the usual sense i.e. non-deterministic interleaving of the sequential atomic actions of each process. They use CTL as a specification language, and consider the semantics of CTL with respect to a (Kripke) structure $M = (S, A_1, \ldots, A_k, L)$, where

- S: countable set of system states
- $A_i \subseteq S \times S$: transition relation of process i
- L: assignment of atomic propositions to each state

They use a decision procedure for satisfiability of CTL formulae (similar to one described in [Ben-Ari et al., 1981]) as part of their synthesis procedure. Given a CTL formula f_0, the procedure returns either "Yes, f_0 is satisfiable or "No, f_0 is unsatisfiable". If f_0 is satisfiable, then a finite model (structure) is also constructed.

So, their overall synthesis algorithm consists of the following high level steps:

1. Specify the desired behavior of the concurrent system using a CTL formula φ.
2. Apply the decision procedure to the formula φ to obtain a finite model fo the formula.
3. Factor out the synchroniztion skeletons of the individual processes from the global system flowgraph defined by the model.

They demonstrate this procedure on a simple, 2 process mutual exclusion example. Below is shown the description of the abstract states of each process, NCS_i, TRY_i, CS_i:

We illustrate the method by solving a mutual exclusion problem for processes P_1 and P_2. Each process is always in one of three regions of code:

NCS_i	the NonCritical Section
TRY_i	the TRYing Section
CS_i	the Critical Section

which it moves through as suggested in Fig. 6.1.

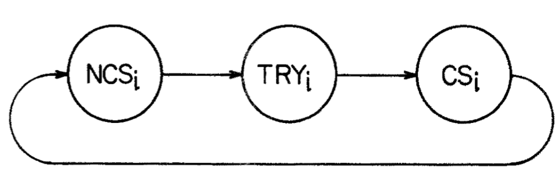

Figure 6.1

and they give the specification of the mutual exclusion problem in CTL as follows:

1. start state

$$NCS_1 \wedge NCS_2$$

2. mutual exclusion

$$AG(\sim(CS_1 \wedge CS_2))$$

3. absence of starvation for P_i

$$AG(TRY_i \rightarrow AF\ CS_i)$$

4. each process P_i is always in exactly one of the three code regions

$$AG(NCS_i \vee TRY_i \vee CS_i)$$
$$AG(NCS_i \rightarrow \sim(TRY_i \vee CS_i))$$
$$AG(TRY_i \rightarrow \sim(NCS_i \vee CS_i))$$
$$AG(CS_i \rightarrow \sim(NCS_i \vee TRY_i))$$

5. it is always possible for P_i to enter its trying region from its non-critical region

$$AG(NCS_i \rightarrow EX_i TRY_i)$$

6. it is always the case that any move P_i makes from its trying region is into the critical region

$$AG(TRY_i \wedge EX_i True \rightarrow AX_i CS_i)$$

7. it is always possible for P_i to re-enter its noncritical region from its critical region

$$AG(CS_i \rightarrow EX_i NCS_i)$$

8. a transition by one process cannot cause a move by the other

$$AG(NCS_i \rightarrow AX_j NCS_i)$$
$$AG(TRY_i \rightarrow AX_j TRY_i)$$
$$AG(CS_i \rightarrow AX_j CS_i)$$

9. some process can always move

$$AG(EX\ true)$$

References

[Ben-Ari et al., 1981] Ben-Ari, M., Manna, Z., and Pnueli, A. (1981). The temporal logic of branching time. In *Proceedings of the 8th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '81, page 164–176, New York, NY, USA. Association for Computing Machinery.

[Clarke and Emerson, 1982] Clarke, E. M. and Emerson, E. A. (1982). Design and synthesis of synchronization skeletons using branching time temporal logic. In Kozen, D., editor, *Logics of Programs*, pages 52–71, Berlin, Heidelberg. Springer Berlin Heidelberg.

[Manna and Waldinger, 1980] Manna, Z. and Waldinger, R. (1980). A deductive approach to program synthesis. *ACM Transactions on Programming Languages and Systems (TOPLAS)*, 2(1):90–121.

[Manna and Wolper, 1984] Manna, Z. and Wolper, P. (1984). Synthesis of Communicating Processes from Temporal Logic Specifications. *ACM Trans. Program. Lang. Syst.*, 6(1):68–93.