1 Validity-Guided Synthesis from Assume-Guarantee Contracts

1.1 Preliminaries

(Text from TACAS submission) We describe a system using the disjoint sets *state* and *inputs*. Formally, an *implementation* is a *transition system* described by an initial state predicate $I(s)$ of type $state \rightarrow bool$ and by a transition relation $T(s, i, s')$ of type $state \rightarrow inputs \rightarrow state \rightarrow bool$.

An Assume-Guarantee (AG) contract can be formally defined by a set of *assumptions* and a set of *guarantees*. The *assumptions*, $A : state \rightarrow inputs \rightarrow bool$, impose constraints over the inputs which may be modal in terms of the previous state. The *guarantees* G consist of two separate subsets $G_I : state \rightarrow bool$ and $G_T : state \rightarrow inputs \rightarrow state \rightarrow bool$, where G_I defines the set of valid initial states, and G_T specifies the properties that need to be met during each new transition between two states. Note that we do not necessarily expect that a contract would be defined over all variables in the transition system, but we do not make any distinction between internal state variables and outputs in the formalism. This way, we can use state variables to (in some cases) simplify specification of guarantees.

1.2 Approach

The main idea of the algorithm is to use over- and underapproximations to create a set of states, say $F(s)$ in which the system can safely use the transition relation without violating the contract. Due to AE-VAL's machinery, we are able to finally extract a Skolem Function S_k that describes an implementation. The algorithm is split into two different phases. The first is focused on finding "good" initial states.

Initially, $F(s) = true$. We then ask AE-VAL whether the formula $\forall s, i.F(s) \wedge A(s, i) \wedge G_I(s) \Rightarrow \exists s'.G_T(s, i, s') \wedge F(s')$. If the formula is valid, any state can be initial with the given contract. In this case, we proceed with extracting a Skolem function that describes a transition to a new "good" state.

Alternatively, AE-VAL will return with a "non-valid" answer. In addition to this, a subset, named $Q(s, i)$ is also generated, for which the formula $\forall s, i.Q(s, i) \Rightarrow \exists s'.G_T(s, i, s')$ is valid. We can use this subset to refine $F(s)$ in the following way:

- Define $R(s) = true$. Examine the validity of $\forall s.R(s) \Rightarrow \exists i.\neg Q(s, i)$. If the formula is valid, the contract cannot be realizable, as for every state s, there are inputs for which the assumptions are violated. On the other hand, if AE-VAL returns "non-valid" we receive a valid subset of $R(s)$, namely $W(s)$.
- By definition, we now have that $\forall s.W(s) \Rightarrow \exists i.\neg Q(s, i)$. The set $W(s)$ effectively describes a region of states for which the contract is violated,

and therefore will have to be blocked from the resulting set that describes the system's "good" states. Thus, we refine $F(s) \equiv F(s) \wedge \neg W(s)$.

- We plug the refined $F(s)$ in the original formula, and reiterate the process. Eventually, a point is reached where we get a valid answer and extract a Skolem relation that precisely describes $F(s) \wedge A(s,i) \wedge G_I s$.

The second phase of the algorithm is identical to the first, but does not have the requirement that a state satisfies $G_I s$. Thus, we begin by checking the validity of $\forall s, i.F(s) \wedge A(s,i) \Rightarrow \exists s' G_T(s,i,s') \wedge F(s')$. The process to refine $F(s)$ is identical to the previous phase. Again, we eventually either decide that the contract is unrealizable, or are able to extract a Skolem function that describes a transition between states that comply to the contract.

1.3 Notes and Comments

A possible way to optimize the algorithm's convergence described above, would be to replace $F(s) = true$ with a refined set $F'(s)$ resulting from the recursive blocking of bad states, starting from the formula $\forall s, i.A(s,i) \wedge G_I(s) \Rightarrow \exists s'.G_T(s,i,s')$. Then, the algorithm would execute the first phase by checking the validity of $\forall s, i.F'(s) \Rightarrow existss'.G_T(s,i,s') \wedge F'(s')$. A similar approach can be also applied to the second phase.