An Overview of Generate & Check Method in CafeOBJ

FUTATSUGI,Kokichi
二木 厚吉
JAIST

Specification Verification

- Constructing specifications and verifying them in the upstream of system/software development are still one of the most important challenges in system/software development and engineering. It is because many critical defects are caused at the phases of domain, requirement, and design specifications.
- Proof scores are intended to meet this challenge.

Verification with Proof Scores (1)

- For verifying a system, a model of the system should be formalized and described as system specifications that are formal specifications of the behavior of the system. System specifications are formalized in equations and transition rules.
- In conjunction with the system specifications, functions and predicates that are necessary for expressing the system's supposed properties are formalized and described in equations as property specifications.
- Proof scores are developed to verify that the system's supposed properties are deduced from the system and property specifications.

Verification with Proof Scores (2)

- Proof scores are described in equations, and the deduction is done only by reduction (i.e. rewriting from left to wright) with the equations.
- Proof scores are written uniformly with specifications in an executable algebraic specification language, and can enjoy a transparent, simple, executable and efficient logical foundation based on the equational and rewriting logics.

Generate & Check Method (1)

- For a sort Srt and a predicate p on Srt we get $((p(X{:}Srt) \to_E^* \texttt{true})\ \texttt{implies}\ (\forall t \in (T_\Sigma)_{Srt})(p(t) =_E \texttt{true}))$ and $(p(X{:}Srt) \to_E^* \texttt{true})$ is a sufficient condition to prove $(\forall t)p(t)$.
- However, usually p is not simple enough to obtain $(p(X{:}Srt) \to_E^* \texttt{true})$ directly, and we need to analyze the structure of terms in $(T_\Sigma)_{Srt}$ and E for (1) **generating** a set of terms $\{t_1, \cdots, t_m\} \subseteq T_\Sigma(Y)_{Srt}$ that covers all possible cases of $(T_\Sigma)_{Srt}$, and (2) **checking** $(p(t_i) \to_E^* \texttt{true})$ for each $i \in \{1, \cdots, m\}$.
- **Induction** is another technique for proving $(p(X{:}Srt) \to_E^* \texttt{true})$ for a constrained sort Srt.

Generate & Check Method (2)

- The generation & checking can be a theorem proving method for transition systems based on
  (1) generation of finite state patters that cover all possible infinite states, and
  (2) checking the validities of verification conditions for each of the finite state patterns.
- The state space of a transition system is formalized as a quotient set (i.e. a set of equivalence classes) of terms of a topmost sort `State`, and the transitions are specified with conditional transition rules (rewrite rules) over the quotient set.

Generate & Check Method (3)

A property to be verified is either

- an invariant (i.e. a state predicate that is valid for all reachable states), or
- a (p leads-to q) property for two state predicates p and q ((p leads-to q) means that from any reachable state s with $(\mathrm{p}(s) = \mathtt{true})$ the system will get to a state t with $(\mathrm{q}(t) = \mathtt{true})$ no matter what transition sequence is taken).

- A **transition system** is defined as a three tuple (St, Tr, In).
- St is a set of states, $Tr \subseteq St \times St$ is a set of transitions on the states, and $In \subseteq St$ is a set of initial states.
- A sequence of states $s_1 s_2 \cdots s_n$ with $(s_i, s_{i+1}) \in Tr$ for each $i \in \{1, \cdots, n-1\}$ is defined to be a **transition sequence**.
- A state $s^r \in St$ is defined to be **reachable** if there exists a transition sequence $s_1 s_2 \cdots s_n$ with $s_n = s^r$ for $n \in \{1, 2, \cdots\}$ such that $s_1 \in In$.
- A state predicate p (i.e. a function from St to `Bool`) is defined to be an **invariant** (or an invariant property) if ($p(s^r)$ $=$ `true`) for any reachable state s^r.

- Let $\Sigma = (S, \leq, F)$ be a regular order-sorted signature with a set of sorts S, and let $X = \{X_s\}_{s \in S}$ be an S-sorted set of variables.
- Let $T_\Sigma(X)$ be S-sorted set of $\Sigma(X)$-terms, let $T_\Sigma(X)_s$ be a set of $\Sigma(X)$-terms of sort s, let E be a set of $\Sigma(X)$-equations, and let (Σ, E) be an equational specification with unique sort `State`.
- Let $\theta \in T_\Sigma(Y)^X$ be a substitution (i.e. a map) from X to $T_\Sigma(Y)$ for disjoint X and Y then θ extends to the morphism from $T_\Sigma(X)$ to $T_\Sigma(Y)$, and $t\,\theta$ is the term obtained by substituting $x \in X$ in t with $x\,\theta$.

- Let $tr = (\forall X)(l \to r \; \texttt{if} \; c)$ be a rewrite rule with $l, r \in T_\Sigma(X)_{\texttt{State}}$ and $c \in T_\Sigma(X)_{\texttt{Bool}}$, then tr is called a transition rule and defines the one step transition relation $\to_{tr} \in T_\Sigma(Y)_{\texttt{State}} \times T_\Sigma(Y)_{\texttt{State}}$ for Y being disjoint from X as follows.
- Note that $=_E$ is understood to be defined with $((\Sigma \cup Y), E)$ by considering $y \in Y$ as a fresh constant if Y is not empty.

$$(s \to_{tr} s') \stackrel{\text{def}}{=} (\exists \theta \in T_\Sigma(Y)^X)((s =_E l\,\theta) \text{ and } (s' =_E r\,\theta) \text{ and } (c\,\theta =_E \texttt{true}))$$

- Let $TR = \{tr_1, \cdots, tr_m\}$ be a set of transition rules, let $\rightarrow_{TR} \stackrel{\mathrm{def}}{=} \bigcup_{i=1}^{m} \rightarrow_{tr_i}$, and let $In \subseteq (T_\Sigma/{=_E})_{\mathtt{State}}$. In is assumed to be defined via a state predicate *init* that is defined with E, i.e. $(s \in In)$ `iff` $(init(s) =_E \mathtt{true})$.
- Then a transition specification (Σ, E, TR) defines a transition system $((T_\Sigma/{=_E})_{\mathtt{State}}, \rightarrow_{TR}, In)$.

- Given a transition system $TS = (St, Tr, In)$, let $p_1, p_2, \cdots, p_n$ $(n \in \{1, 2, \cdots\})$ be state predicates of TS, and $inv(s) \stackrel{\text{def}}{=} (p_1(s) \texttt{ and } p_2(s) \texttt{ and } \cdots \texttt{ and } p_n(s))$ for $s \in St$.
- The following three conditions are sufficient for a state predicate p^t to be an invariant.

$$\begin{array}{ll} (1) & (\forall s \in St)(inv(s) \texttt{ implies } p^t(s)) \\ (2) & (\forall s \in St)(init(s) \texttt{ implies } inv(s)) \\ (3) & (\forall (s, s') \in Tr)(inv(s) \texttt{ implies } inv(s')) \end{array}$$

- A predicate that satisfies the conditions (2) and (3) like *inv* is called an **inductive invariant**. If p^t itself is an inductive invariant then taking $p_1 = p^t$ and $n = 1$ is enough. However, $p_1, p_2, \cdots, p_n$ $(n > 1)$ are almost always needed to be found for getting an inductive invariant, and to find them is a most difficult part of the invariant verification.

It is worthwhile to note that there are following two contrasting approaches for formalizing $p_1, p_2, \cdots, p_n$ for a transition system and its property p^t.

- Make $p_1, p_2, \cdots, p_n$ as minimal as possible to imply the target property p^t;
  - usually done by lemma finding in interactive theorem proving,
  - it is difficult to find lemmas without some comprehensive understanding of the system.
- Make $p_1, p_2, \cdots, p_n$ as comprehensive as possible to characterize the system;
  - usually done by specifying elemental properties of the system as much as possible in formal specification development,
  - it is difficult to identify the elemental properties without focusing on the property to be proved (i.e. p^t).

[Subsume] A term $t' \in T_\Sigma(Y)$ is defined to be an **instance** of a term $t \in T_\Sigma(X)$ iff there exits a substitution $\theta \in T_\Sigma(Y)^X$ such that $t' = t\,\theta$. A finite set of terms $C \subseteq T_\Sigma(X)$ is defined to **subsume** a (may be infinite) set of ground terms (i.e. terms without variables) $G \subseteq T_\Sigma$ iff for any $t' \in G$ there exits $t \in C$ such that t' is an instance of t.
[Generate&Check-S] Let $(T_\Sigma/{=_E})_{\mathtt{State}}, \rightarrow_{TR}, In)$ be a transition system defined by a transition specification (Σ, E, TR). Then, for a state predicate p_{st}, doing the following **Generate** and **Check** are sufficient for verifying

$$(\forall t \in (T_\Sigma)_{\mathtt{State}})(p_{st}(t) =_E \mathtt{true}).$$

Generate a finite set of state terms $C \subseteq T_\Sigma(X)_{\mathtt{State}}$ that subsumes $(T_\Sigma)_{\mathtt{State}}$.

Check $(p_{st}(s) \twoheadrightarrow_E^* \mathtt{true})$ for each $s \in C$. □

Let `q` be a predicate with arity “`State State`” for stating some relation of the current state and the next state, like ($inv(s)$ `implies` $inv(s')$). Let the function `valid-q` be defined using the CafeOBJ 's built-in search predicate

`pred _=(*,1)=>+_if_suchThat_{_} : State %State %Bool Bool Info`

as follows.

```
-- predicate to be checked for a State
pred valid-q : State .
eq valid-q(S:State) =
   not(S =(*,1)=>+ SS:State if CC:Bool
       suchThat not((CC implies q(S,SS)) == true)
       {(ifm S SS CC q(S,SS))}) .
```

For a state term $s \in T_\Sigma(Y)_{\mathtt{State}}$, the reduction of the Boolean term: `valid-q(s)` with $\twoheadrightarrow^*_E \cup \rightarrow_{TR}$ behaves as follows based on the definition of the behavior of the built-in search predicate.

1. Search for evey pair (tr_j, θ) of a transition rule $tr_j = (\forall X)(l_j \rightarrow r_j \ \mathtt{if}\ c_j)$ in Tr and a substitution $\theta \in T_\Sigma(Y)^X$ such that $\mathtt{s} = l_j\,\theta$.
2. For each found (tr_j, θ), let $(\mathtt{SS} = r_j\,\theta)$ and $(\mathtt{CC} = c_j\,\theta)$ and print out `(ifm s SS CC q(s,SS))` and tr_j if $\big(\mathtt{not((CC\ implies\ q(s,SS))\ ==\ true)} \twoheadrightarrow^*_E \mathtt{true}\big)$.
3. Returns `false` if any print out exits, and returns `true` otherwise.

[Cover] Let $C \subseteq T_\Sigma(Y)$ and $C' \subseteq T_\Sigma(X)$ be finite sets. C is defined to **cover** C' iff for any ground instance $t'_g \in T_\Sigma$ of any $t' \in C'$, there exits $t \in C$ such that t'_g is an instance of t and t is an instance of t'.

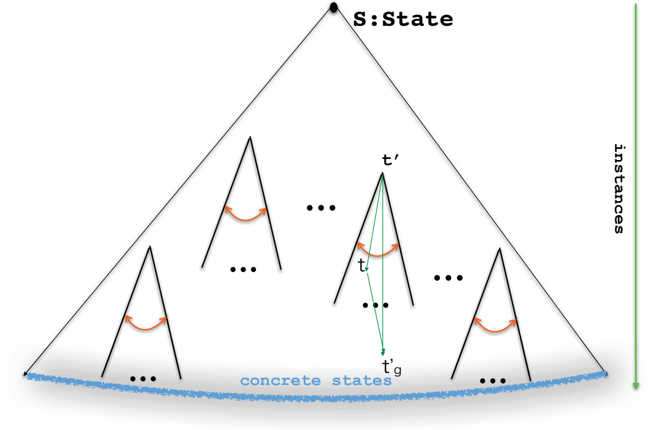
[Generate&Check-T1] Let $((T_\Sigma/{=_E})_{\mathtt{State}}, \rightarrow_{TR}, In)$ be a transition system, and let $C' \subseteq T_\Sigma(X)$ be the set of all the left-hand sides of the transition rules in TR. Then doing the following **Generate** and **Check** are sufficient for verifying

$$(\forall (s, s') \in ((T_\Sigma \times T_\Sigma) \cap \rightarrow_{TR}))(\mathtt{q_{tr}}(s, s') =_E \mathtt{true})$$

for a predicate "$\mathtt{pred\ q_{tr}\ :\ \ State\ State}$".

> **Generate** a finite set of state terms $C \subseteq T_\Sigma(Y)_{\mathtt{State}}$ that covers C'.
>
> **Check** $(\mathtt{valid\text{-}q_{tr}(t)} \twoheadrightarrow^*_E \cup \rightarrow_{TR} \mathtt{true})$ for each $\mathtt{t} \in C$. □

Covering



[Generate&Check-T2] Let $TR = \{tr_1, \cdots, tr_m\}$ be a set of transition rules, and let $tr_i = (\forall X)(l_i \rightarrow r_i \texttt{ if } c_i)$ for $i \in \{1, \cdots, m\}$. Then doing the following **Generate** and **Check** for all of $i \in \{1, \cdots, m\}$ is sufficient for verifying

$$(\forall(s, s') \in ((T_\Sigma \times T_\Sigma) \cap \rightarrow_{TR}))(\mathtt{q_{tr}}(s, s') =_E \mathtt{true})$$

for a predicate "$\mathtt{pred\ q_{tr}\ :\ \ State\ State}$".

> **Generate** a finite set of state terms $C_i \subseteq T_\Sigma(Y)_{\mathtt{State}}$ that covers $\{l_i\}$.
>
> **Check** $(\mathtt{valid\text{-}q_{tr}(t)} \twoheadrightarrow^*_E \cup \rightarrow_{tr_i} \mathtt{true})$ for each $\mathtt{t} \in C$. $\square$

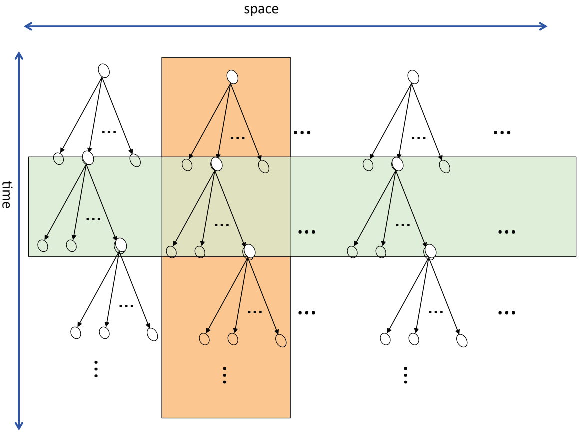
The conditions (1) and (2) for invariant properties can be verified by using Generate&Check-S with $\mathtt{p}_{\mathtt{st-1}}(s)$ and $\mathtt{p}_{\mathtt{st-2}}(s)$ defined as follows respectively.

$$
\begin{array}{ll}
(1) & \mathtt{p}_{\mathtt{st-1}}(s) = (inv(s)\ \mathtt{implies}\ p^t(s)) \\
(2) & \mathtt{p}_{\mathtt{st-2}}(s) = (init(s)\ \mathtt{implies}\ inv(s))
\end{array}
$$

Note that, if $inv \stackrel{\text{def}}{=} (p_1\ \mathtt{and} \cdots \mathtt{and}\ p_n)$ and $p^t = (p_{i_1}\ \mathtt{and} \cdots \mathtt{and}\ p_{i_m})$ for $\{i_1, \cdots, i_m\} \subseteq \{1, \cdots, n\}$, then condition (1) is directly obtained.

The condition (3) for invariant properties can be verified by using Generate&Check-T1 or T2 with $\mathtt{q}_{\mathtt{tr-3}}(s, s')$ defined as follows.

$$(3)\quad \mathtt{q}_{\mathtt{tr-3}}(s, s') = (\mathit{inv}(s)\ \mathtt{implies}\ \mathit{inv}(s'))$$



Searches on Time versus Space