Model Checking

Lecture 3 (April 30th)

TUM

Reachability computation

Let φ be a formula over V and let ρ be a formula over V and V'. We define a *post-condition* function *post* as follows.

$$post(\varphi, \rho) \ = \ \exists V'' : \varphi[V''/V] \wedge \rho[V''/V][V/V'] \tag{1}$$

An application $post(\varphi, \rho)$ computes the image of the set φ under the relation ρ. Furthermore, for a natural number n we define $post^n(\varphi, \rho)$ as follows.

$$post^n(\varphi, \rho) \ = \ \begin{cases} \varphi & \text{if } n = 0 \\ post(post^{n-1}(\varphi, \rho), \rho) & \text{otherwise} \end{cases} \tag{2}$$

By $post^n(\varphi, \rho)$ we represent the n-fold application of the *post* function to φ with respect to ρ. We observe the following useful property of the post-condition function.

$$\begin{aligned} &\forall \varphi\ \forall \rho_1\ \forall \rho_2 : post(\varphi, \rho_1 \vee \rho_2) = (post(\varphi, \rho_1) \vee post(\varphi, \rho_2)) \\ &\forall \varphi_1\ \forall \varphi_2\ \forall \rho : post(\varphi_1 \vee \varphi_2, \rho) = (post(\varphi_1, \rho) \vee post(\varphi_2, \rho)) \end{aligned} \tag{3}$$

This property states that the post-condition computation distributes over disjunction wrt. each argument.

Example 1. For example, given the transition relation ρ_2 and the program variables $V = (pc, x, y, z)$ from our example program, we compute the following post condition.

$$\begin{aligned} & post(at\_\ell_2 \wedge y \geq z, \rho_2) \\ &\quad = (\exists V'' : (at\_\ell_2 \wedge y \geq z)[V''/V] \wedge \rho_2[V''/V][V/V']) \\ &\quad = (\exists V'' : (pc'' = \ell_2 \wedge y'' \geq z'') \wedge \\ &\qquad\qquad (pc'' = \ell_2 \wedge pc' = \ell_2 \wedge x'' + 1 \leq y'' \wedge x' = x'' + 1 \wedge \\ &\qquad\qquad\ y' = y'' \wedge z' = z'')[V/V']) \\ &\quad = (\exists V'' : (pc'' = \ell_2 \wedge y'' \geq z'') \wedge \\ &\qquad\qquad (pc'' = \ell_2 \wedge pc = \ell_2 \wedge x'' + 1 \leq y'' \wedge x = x'' + 1 \wedge \\ &\qquad\qquad\ y = y'' \wedge z = z'')) \\ &\quad = (pc = \ell_2 \wedge y \geq z \wedge x \leq y) \end{aligned}$$

We compute the 2-fold application by reusing the above result.

$$
\begin{aligned}
& post^2(at\_\ell_2 \wedge y \geq z, \rho_2) \\
& \quad = post(post(at\_\ell_2 \wedge y \geq z, \rho_2), \rho_2) \\
& \quad = post(pc = \ell_2 \wedge y \geq z \wedge x \leq y, \rho_2) \\
& \quad = (\exists V'' : (pc'' = \ell_2 \wedge y'' \geq z'' \wedge x'' \leq y'') \wedge \\
& \qquad\qquad\qquad (pc'' = \ell_2 \wedge pc = \ell_2 \wedge x'' + 1 \leq y'' \wedge x = x'' + 1 \wedge \\
& \qquad\qquad\qquad\; y = y'' \wedge z = z'')) \\
& \quad = (pc = \ell_2 \wedge y \geq z \wedge x - 1 \leq y \wedge x \leq y) \\
& \quad = (pc = \ell_2 \wedge y \geq z \wedge x \leq y)
\end{aligned}
$$

□

We characterize φ_{reach} using *post* as follows.

$$
\begin{aligned}
\varphi_{reach} &= \varphi_{init} \vee post(\varphi_{init}, \rho_{\mathcal{R}}) \vee post(post(\varphi_{init}, \rho_{\mathcal{R}}), \rho_{\mathcal{R}}) \vee \ldots \\
&= \bigvee\nolimits_{i \geq 0} post^i(\varphi_{init}, \rho_{\mathcal{R}})
\end{aligned}
\tag{4}
$$

The above disjunction (over every number of applications of the post-condition function) ensures that all reachable states are taken into consideration.

Example 2. We compute φ_{reach} for our example program. We first obtain the post-condition after applying the transition relation of the program once.

$$
\begin{aligned}
& post(at\_\ell_1, \rho_{\mathcal{R}}) \\
& \quad = (post(at\_\ell_1, \rho_1) \vee post(at\_\ell_1, \rho_2) \vee post(at\_\ell_1, \rho_3) \vee \\
& \qquad\; post(at\_\ell_1, \rho_4) \vee post(at\_\ell_1, \rho_5)) \\
& \quad = post(at\_\ell_1, \rho_1) \\
& \quad = (at\_\ell_2 \wedge y \geq z)
\end{aligned}
$$

Next, we obtain the post-condition for one more application.

$$
\begin{aligned}
& post(at\_\ell_2 \wedge y \geq z, \rho_{\mathcal{R}}) \\
& \quad = (post(at\_\ell_2 \wedge y \geq z, \rho_2) \vee post(at\_\ell_2 \wedge y \geq z, \rho_3)) \\
& \quad = (at\_\ell_2 \wedge y \geq z \wedge x \leq y \vee at\_\ell_3 \wedge y \geq z \wedge x \geq y)
\end{aligned}
$$

We repeat the application step once again.

$$
\begin{aligned}
& post(at\_\ell_2 \wedge y \geq z \wedge x \leq y \vee at\_\ell_3 \wedge y \geq z \wedge x \geq y, \rho_{\mathcal{R}}) \\
& \quad = (post(at\_\ell_2 \wedge y \geq z \wedge x \leq y, \rho_{\mathcal{R}}) \vee post(at\_\ell_3 \wedge y \geq z \wedge x \geq y, \rho_{\mathcal{R}})) \\
& \quad = (post(at\_\ell_2 \wedge y \geq z \wedge x \leq y, \rho_2) \vee post(at\_\ell_2 \wedge y \geq z \wedge x \leq y, \rho_3) \vee \\
& \qquad\; post(at\_\ell_3 \wedge y \geq z \wedge x \geq y, \rho_4) \vee post(at\_\ell_3 \wedge y \geq z \wedge x \geq y, \rho_5)) \\
& \quad = (at\_\ell_2 \wedge y \geq z \wedge x \leq y \vee at\_\ell_3 \wedge y \geq z \wedge x = y \vee \\
& \qquad\; at\_\ell_4 \wedge y \geq z \wedge x \geq y)
\end{aligned}
$$

So far, by iteratively applying the post-condition function to φ_{init} we obtained the following disjunction.

$$
\begin{aligned}
& at\_\ell_1 \lor \\
& at\_\ell_2 \land y \geq z \lor \\
& at\_\ell_2 \land y \geq z \land x \leq y \lor at\_\ell_3 \land y \geq z \land x \geq y \lor \\
& at\_\ell_2 \land y \geq z \land x \leq y \lor at\_\ell_3 \land y \geq z \land x = y \lor \\
& at\_\ell_4 \land y \geq z \land x \geq y
\end{aligned}
$$

We present this disjunction in a logically equivalent, simplified form as follows.

$$
\begin{aligned}
& at\_\ell_1 \lor \\
& at\_\ell_2 \land y \geq z \lor \\
& at\_\ell_3 \land y \geq z \land x \geq y \lor \\
& at\_\ell_4 \land y \geq z \land x \geq y
\end{aligned}
$$

Any further application of the post-condition function does not produce any additional disjuncts. Hence, φ_{reach} is the above disjunction. □

Inductive Safety Arguments

An *inductive invariant* φ contains the intial states and is closed under successors. Formally, an inductive invariant is a formula over the program variables that represents a superset of the initial program states and is closed under the application of the *post* function wrt. the relation $\rho_{\mathcal{R}}$, i.e.,

$$\varphi_{init} \models \varphi \quad \text{and} \quad post(\varphi, \rho_{\mathcal{R}}) \models \varphi \ .$$

A program is safe if there exists an inductive invariant φ that does not contain any error states, i.e., $\varphi \land \varphi_{err} \models false$.

Example 3. For our example program, the weakest inductive invariant consists of the set of all states and is represented by the formula *true*. The strongest inductive invariant was obtained in Example 2 and is shown below.

$$at\_\ell_1 \lor (at\_\ell_2 \land y \geq z) \lor (at\_\ell_3 \land y \geq z \land x \geq y) \lor (at\_\ell_4 \land y \geq z \land x \geq y)$$

The strongest inductive invariant does not contain any error states. We observe that a slightly weaker inductive invariant below also proves the safety of our examples.

$$at\_\ell_1 \lor (at\_\ell_2 \land y \geq z) \lor (at\_\ell_3 \land y \geq z \land x \geq y) \lor at\_\ell_4$$

□

Computation of reachable program states requires iterative application of the post-condition function on the initial program states, see Equation (4). The iteration finishes when no new program states are discovered. Unfortunately, such an iteration process does not terminate in finite time.

Example 4. For example, we consider the iterative computation of the set of states that is reachable from $at\_\ell_2 \wedge x \leq z$ by applying the transition ρ_2 of our example program. We obtain the following sequence of post-conditions (where $V = (pc, x, y, z)$).

$$\begin{aligned}
post(at_\ell_2 \wedge x \leq z, \rho_2) &= (\exists V'' : (pc'' = \ell_2 \wedge x'' \leq z'') \wedge \\
&\qquad\qquad\qquad (pc'' = \ell_2 \wedge pc = \ell_2 \wedge x'' + 1 \leq y'' \wedge \\
&\qquad\qquad\qquad\ x = x'' + 1 \wedge y = y'' \wedge z = z'')) \\
&= (at_\ell_2 \wedge x - 1 \leq z \wedge x \leq y) \\
post^2(at_\ell_2 \wedge x \leq z, \rho_2) &= (at_\ell_2 \wedge x - 2 \leq z \wedge x \leq y) \\
post^3(at_\ell_2 \wedge x \leq z, \rho_2) &= (at_\ell_2 \wedge x - 3 \leq z \wedge x \leq y) \\
&\dots \\
post^n(at_\ell_2 \wedge x \leq z, \rho_2) &= (at_\ell_2 \wedge x - n \leq z \wedge x \leq y)
\end{aligned}$$

In this sequence, we observe that at each iteration yields a set of states that contains states not discovered before. For example, the set of states reachable after applying the post-condition function once is not included in the original set, i.e.,

$$(at\_\ell_2 \wedge x - 1 \leq z \wedge x \leq y) \not\models (at\_\ell_2 \wedge x \leq z) \ .$$

The set of states reachable after applying the post-condition function twice is not included in the union of the above two sets, i.e.,

$$(at\_\ell_2 \wedge x - 2 \leq z \wedge x \leq y) \not\models (at\_\ell_2 \wedge x - 1 \leq z \wedge x \leq y \vee at\_\ell_2 \wedge x \leq z) \ .$$

Furthermore, we observe that the set of states reachable after n-fold application of *post*, where $n \geq 1$, still contains previously unreached states, i.e.,

$$\begin{aligned}
&\forall n \geq 1 : (at_\ell_2 \wedge x - n \leq z \wedge x \leq y) \\
&\qquad\qquad \not\models (at_\ell_2 \wedge x \leq z \vee \textstyle\bigvee_{1 \leq i < n}(at_\ell_2 \wedge x - i \leq z \wedge x \leq y)) \ .
\end{aligned}$$

$\square$