Lecture 17

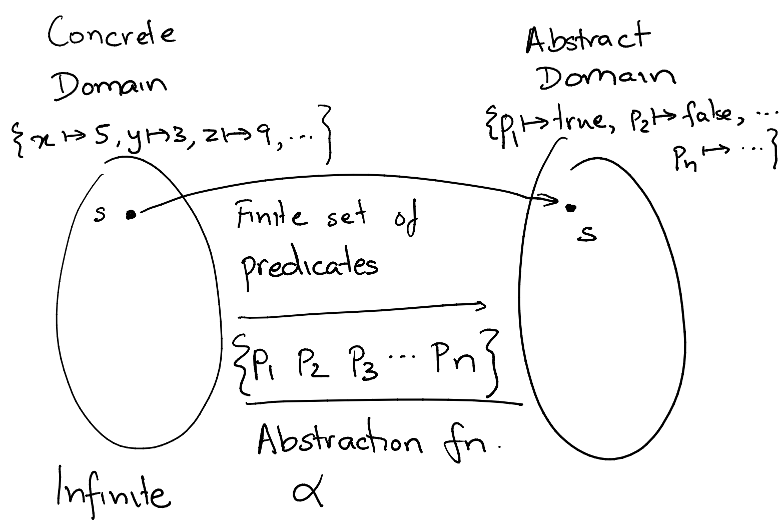
Wednesday, March 25, 2020 4:24 PM

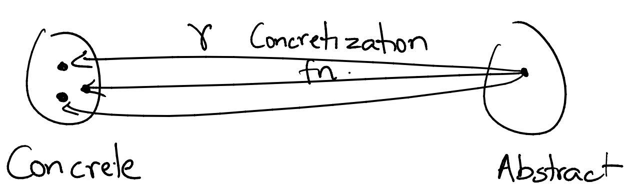
- Hello!
- Please use the link to the shared Google Doc posted on the website

https://docs.google.com/document/d/1cKkrptYBdWS6e-3PqqIblf5amlMcACjF705rxlJgVpo/edit?usp=sharing

---

- Last class: Predicate abstraction. Introduction
- Today: Predicate abstraction, cont'd.
  Counterexample Guided Abstraction Refinement.



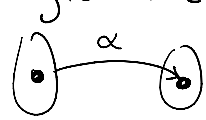


Sanity check theorem:

$$\alpha(\underbrace{\gamma(\hat{s})}) = \hat{s}$$

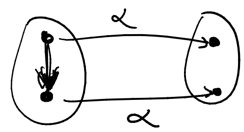
$$\forall s \in \gamma(\hat{s}),\ \alpha(s) = \hat{s}.$$

Abstracting a single state



Edge drawing rule

Abstracting transitions



Given $\hat{s}$ $\hat{s}'$, if there is an edge between the corresponding concrete states, then draw $\hat{s} \longrightarrow \hat{s}'$.

Given $\hat{s}$ $\hat{s}'$, if there is an edge between any of their concrete states, then draw an edge between $\hat{s}$ & $\hat{s}'$.

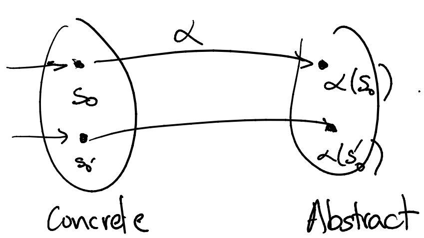
$$\boxed{\exists s \; s' . \quad s \longrightarrow s' \quad \text{and} \quad \alpha(s) = \hat{s} \quad \text{and} \quad \alpha(s') = \hat{s}'}$$

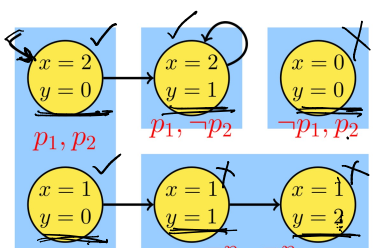
Definition (Minimal Existential Abstraction)

A model $\hat{M} = (\hat{S}, \hat{S}_0, \hat{T})$ is the *minimal existential abstraction* of $M = (S, S_0, T)$ with respect to $\alpha : S \to \hat{S}$ iff

- $\exists s \in S_0 . \alpha(s) = \hat{s} \iff \hat{s} \in \hat{S}_0$ and ← Starting state
- $\exists (s, s') \in T . \alpha(s) = \hat{s} \wedge \alpha(s') = \hat{s}' \iff (\hat{s}, \hat{s}') \in \hat{T}$.

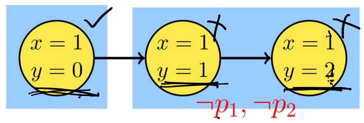
$\forall \hat{s} \; \hat{s}'$

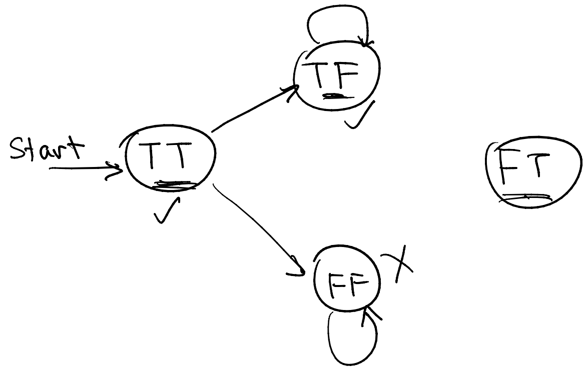




$P_1 = (x > y)$

$P_2 = (y = 0)$





Why is predicate abstraction hard?

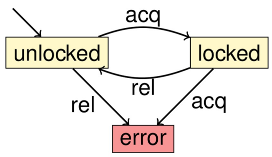
- If there are n predicates, there are 2^n abstract states.
- The naive approach makes $2^n \cdot 2^n = 2^{2n}$ requests to the SMT solver.

Claim: If every reachable abstract state satisfies the assertion, then every reachable concrete state also satisfies the assertion.

~~Converse: If every reachable concrete state satisfies the assertion, then every reachable abstract state also satisfies the assertion.~~

This is false

Assertion → Temporal safety properties.



assert(φ)

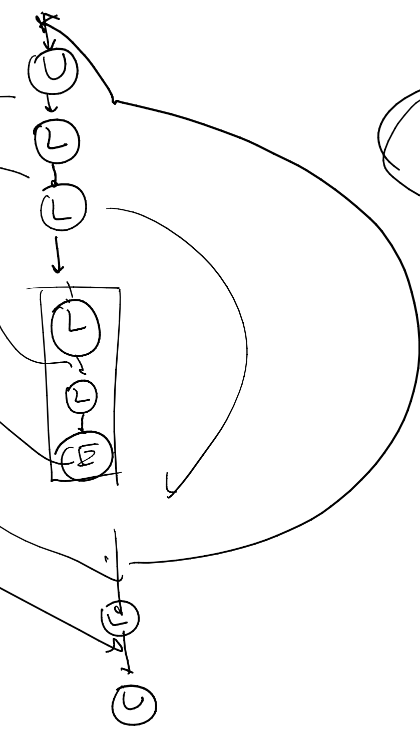
```
var state ∈ {Unlocked,
             Locked,
             Error} = Unlocked

void acq() {
    if (state = Locked) {
        assert(false);
    }
    state := Locked
}

void release() {
    if (state = Unlocked)
        assert(false)
    state := Unlocked
}
```

```
do {
    KeAcquireSpinLock();
    nPacketsOld = nPackets;
    if(request){
        request = request->Next;
        KeReleaseSpinLock();
        nPackets++;
    }
} while(nPackets != nPacketsOld);

KeReleaseSpinLock();
```



npackets++

npackets == npacketsOld

```
if (b) {
    b := false;
} else {
    /* Maybe np = npOld+1;
       In which case,
       b := true.
    Or maybe,
       np = npOld - 2.
       b := false. */
```

? $b := *$

$s := U$

$s := L$

$npo_0 := np_0$

$s := U$

$np_1 := np_0 + 1$

$\boxed{\text{assume}}\ (np_1 == npo_0)$

$\boxed{s := U}$

$\exists\ npo_0\ np_0\ np_1$

s.t.

$npo_0 = np_0$ and

$np_1 = np_0 + 1$ and

$np_1 = npo_0$.

$np_0 + 1 = npo_0$

$= npo_0 + 1 = npo_0$

Components of the abstract state.

Line of code being executed — $l_1, l_2 \cdots l_7$

State of lock: L, U

Value of predicate $npo = np$: T F.

$7 \times 2 \times 2 = 28$ abstract states.

Claim 1: If CEGAR reports OK, then the program is safe.

Claim 2: If CEGAR produces counterexample, then the program is unsafe.

$$\begin{tikzcd}
\text{Abstract}: & \hat{s}_0 \arrow[r] & \hat{s}_1 \arrow[r] & \hat{s}_2 \arrow[r] & \cdots \arrow[r] & \hat{s}_n \; \text{Bad} \\
\text{Concrete}: & s_0 \arrow[r] \arrow[u, "\alpha"] & s_1 \arrow[r] \arrow[u, "\alpha"] & s_2 \arrow[r] \arrow[u, "\alpha"] & \cdots \arrow[r] & s_n \; \text{Bad} \arrow[u, "\alpha"]
\end{tikzcd}$$

Claim 3: Verification of programs with loops is undecidable.