BIG IDEA: System specification is just a temporal predicate!

- What is a predicate?

Formula containing variables that evaluates to true or false given a state (values to variables)

eg: True / $2+2=3$ / $x+y=2$ /

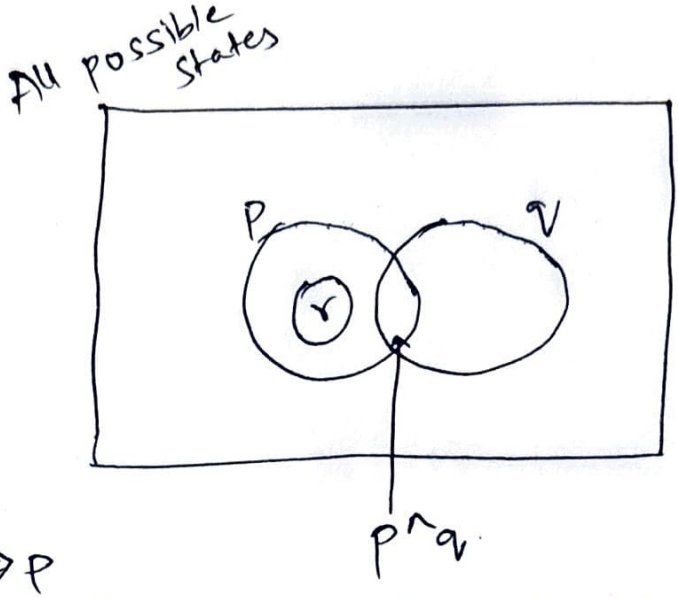
$\forall s, t \in Servers : \neg(rmState[s] = \text{"committed"} \wedge rmState[t] = \text{"aborted"})$

- Given a state s, we can judge whether s SATISFIES the predicate p

i.e, p ACCEPTS the state s

Ex. p: $x+y=z$ ACCEPTS s_0: $x=3$, $y=4$, $z=7$ i.e, $p(s_0)$

BUT NOT s_1: $x=4$, $y=0$, $z=5$, i.e, $\neg p(s_1)$



$r \Rightarrow p$

We can INFER p if r is TRUE

- Logic gives us INFERENCE RULES for reasoning

$\frac{p}{p \vee q}$ Disjunction introduction

$\frac{p \wedge q}{p}$ Conjunction elimination.

$\frac{p \Rightarrow q, p}{q}$ modus pones

Inference rules can also be written with implies. eg. $p \Rightarrow (p \vee q)$

e.g. propositional logic

- If $p \rightarrow q$ and $q \rightarrow p$ then p and q are equivalent. $p \equiv q$

$\neg(\neg p) \equiv p$ $\quad (p \Rightarrow q) \equiv (\neg q \Rightarrow \neg p)$

$\neg \forall x P(x) \equiv \exists x \neg P(x)$

eg: predicate logic

$\forall x P(x) \wedge \forall x Q(x) \equiv \forall x [P(x) \vee Q(x)]$

$\frac{\forall x A(x) \wedge \exists x B(x)}{\exists x A(x) \wedge B(x)}$

- Inference and equivalence rules can help prove statements.

Given $[\forall x\ P(x) \Rightarrow Q(x)] \wedge [\neg \forall x\ Q(x)]$ Prove $[\exists x \neg P(x)]$

$[\forall x\ P(x) \Rightarrow Q(x)] \wedge [\exists x \neg Q(x)]$

$\exists x (P(x) \Rightarrow Q(x) \wedge \neg Q(x)$

$\exists x \neg P(x)$

- Now, system specification is just a temporal predicate.

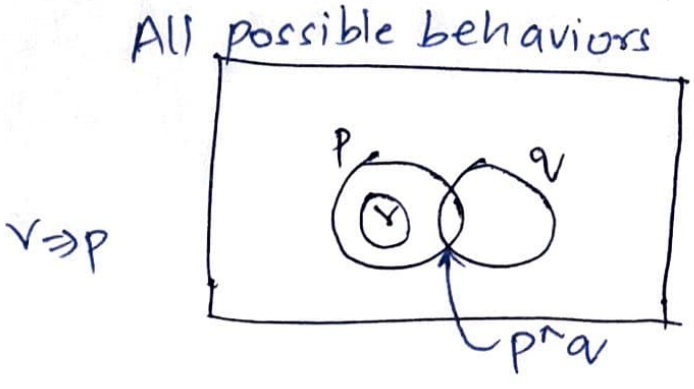
Temporal formula that evaluates to True/false given a sequence of states, behaviors.

$S_1 \rightarrow S_2 \rightarrow S_3 \rightarrow S_4 \cdots$

$S_4 \rightarrow S_2 \rightarrow S_1 \rightarrow S_3 \cdots$

$t_1 \rightarrow t_2 \rightarrow S_3 \rightarrow S_1 \cdots$

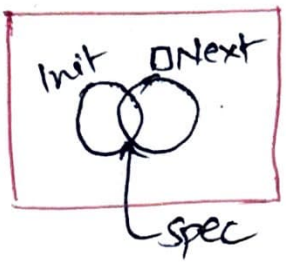
- If a behavior b SATISFIES a temporal predicate p, we say p ACCEPT behavior b



- BIG IDEA: Specification is just a temporal predicate that accepts some behaviors out of all possible behaviors.

Spec ≡ Init ∧ □ Next

Temporal formula



- Invariants are also temporal predicate

Safety ≡ □ TypeOK

Spec ⇒ Safety

Can model check. Or can directly prove using INFERENCE AND EQUIVALENCE RULES

- Example: Transaction Commit

TC Spec ≡ TC Init ∧ □ TC Next ⇒ □ TC TypeOk

TC Init = ∀ s : rm state [s] = "working"

TC TypeOk = rm State[s] ∈ {"working", ...}

They do not have var', so they are state predicates

- A state predicate is a temporal predicate that accepts behaviors where the first state satisfies the state predicate.

If TCInit(s_1) then

$s_1 \to s_2 \to s_3 \to$ are all ACCEPTED by
$s_1 \to s_3 \to s_1 \to s_2$ TCInit
$s_1 \to s_1 \to s_1 \to s_1$

- $\Box$ TypeOk (always) is a temporal formula that accepts behaviors where EVERY state satisfies the state predicate.

$s_1 \to s_2 \to s_3 \to \ldots$ is ACCEPTED by $\Box$ TCTypeOk

iff TCTyok(s_1) $\wedge$ TCTypeOk(s_2) $\wedge$ TCTypeOk(s_3 ... .

- TCNext = $\wedge$ rm state = ...
  $\wedge$ rm state' = ...

is an ACTION predicate

An action predicate is a temporal predicate that accepts behaviors where the first two states satisf satisfy the action predicate.

If TCNext(s_1, s_2) then $s_1 \to s_2 \to s_1 \to s_3$
$s_1 \to s_2 \to s_2 \to s_3$
:

are all ACCEPTED by TCNext

- $\Box$ TCNext is a temporal formula that accepts behaviors where EVERY pair of states satisfy the action predicate

  ($\Box$: always)

$s_1 \to s_2 \to s_3 \cdots$ is ACCEPTED by $\Box$ TCNext iff

$TCNext(s_1, s_2) \wedge TCNext(s_2, s_3) \wedge \cdots$

- TCSpec $\equiv$ TCInit $\wedge \Box$ TCNext ACCEPTS

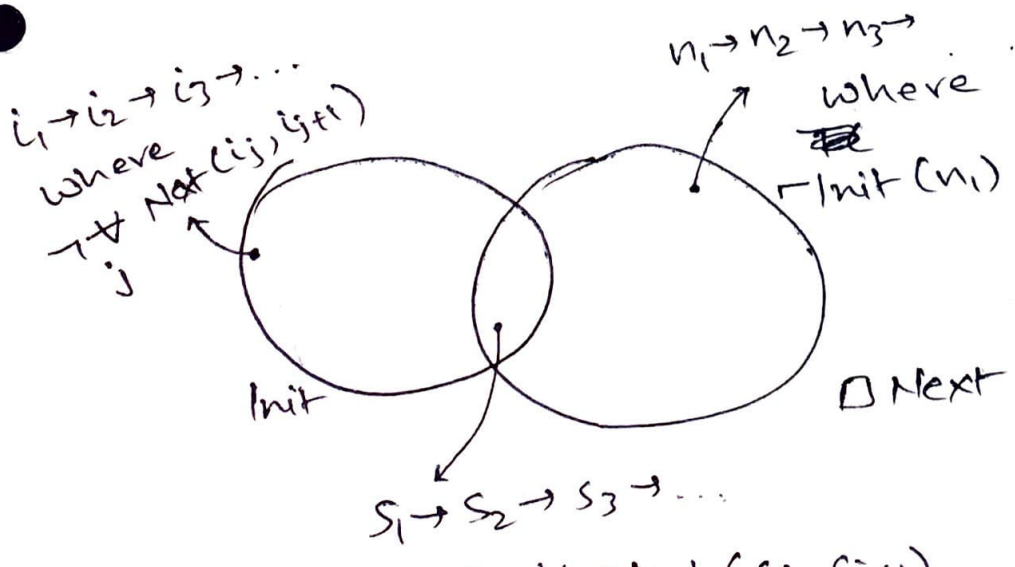
$b = s_1 \to s_2 \to s_3 \cdots$ iff

(Interpret as temporal predicates)

$TCSpec(b) \equiv TCInit(b) \wedge \Box TCNext(b)$

$= TCInit(s_1) \wedge \forall_i TCNext(s_i, s_{i+1})$

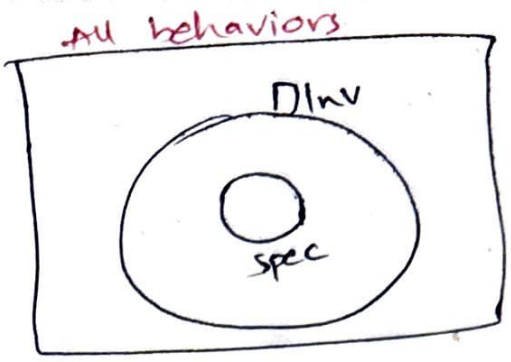
(Interpret as state predicate; Action predicate)

- 

$Next \equiv Init(s_1) \wedge \forall_i Next(s_i, s_{i+1})$

- Proving Invariants

Spec $\Rightarrow$ □Inv



$$Init(s_1) \wedge \forall_i Next(s_i, s_{i+1}) \Rightarrow \forall_i Inv(s_i)$$

- Induction.

$Init(s_1) \Rightarrow Inv(s_1)$ Base Case

$Inv(s_i) \wedge Next(s_i, s_{i+1}) \Rightarrow Inv(s_{i+1})$ Inductive case

i.e,

$Init \Rightarrow Inv$

$Inv \wedge Next \Rightarrow Inv'$

↙ State predicate on this state

↓ Action predicate

↘ state predicate on next state