CS 357 D

Lecture 2

Computational Model

http://cs357d.stanford.edu/

April 5, 2007

Computational Model

Behaviors: sequences of states

System description: state transition systems

compact first-order representation of all sequences of states that can be generated by a system

Programming language: SPL (simple programming language)

with well-defined semantics in terms of transition systems

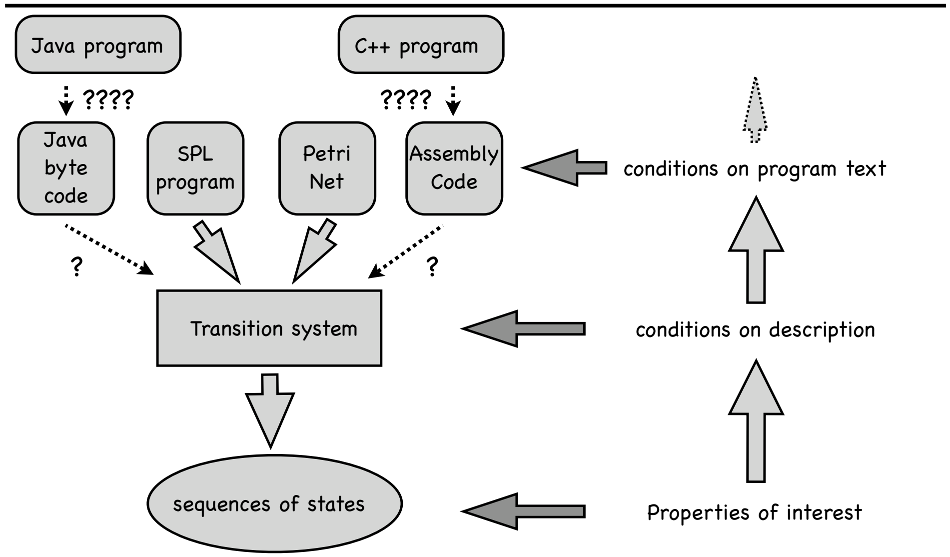
Reference:
Zohar Manna, Amir Pnueli, Temporal Verification of Reactive Systems: Safety, Springer-Verlag, 1995.

Properties of interest

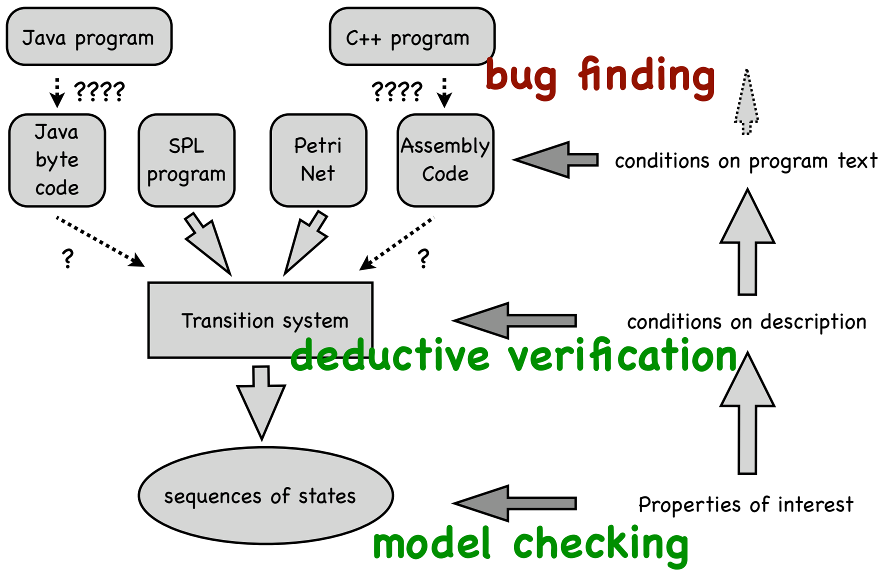
Invariants: overapproximation of the reachable state space

Loop termination: demonstrated by the existence of a ranking function

Semantics



Semantics



States

V: Vocabulary -- set of typed variables {x,y: integer, b: boolean}

expression over V x+y

assertion over V x>y

s: state -- interpretation of all variables {x:2,y:3,b:true}

s[x]=2,s[y]=3,s[b]=true

extends to expressions and assertions s[x+y]=5 s[x>y]=false

Σ: set of all states Z × Z × {true,false}

System Description: Transition systems

Set of typed variables

Example: {x:int, y:int}

$\Phi: \langle V, \Theta, \mathcal{T} \rangle$

Initial condition:
first-order formula

Example: $x=0 \wedge y=0$

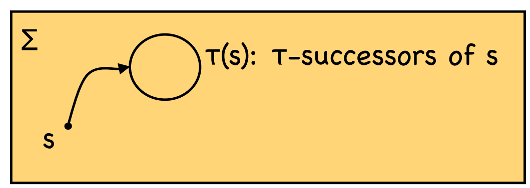
Set of transitions

Compact first-order representation of all sequences of states that can be generated by a system

Transitions

$\mathcal{T}$: finite set of transitions

$\tau \in \mathcal{T}: \Sigma \rightarrow 2^{\Sigma}$



Example:

⋮

τ(<x:2>) = {<x:3>,<x:4>}
τ(<x:3>) = {<x:4>,<x:5>}
τ(<x:4>) = {<x:5>,<x:6>}
τ(<x:5>) = {<x:6>,<x:7>}
τ(<x:6>) = {<x:7>,<x:8>}

⋮

represented by a transition relation $\rho_\tau(V,V')$

V : values of variables in the current state
V': values of variables in the next state

$\rho_\tau: x'=x+1 \vee x'=x+2$

Transitions

A transition τ is **enabled** in a state s if: $\tau(s) \neq \emptyset$

A transition τ is **disabled** in a state s if: $\tau(s) = \emptyset$

Example:

Transition τ with ρ_τ: $(x = 0 \vee x = 1) \wedge (x' = x + 1 \vee x' = x + 2)$

$\tau(\langle x{:}0\rangle) = \{\langle x{:}1\rangle, \langle x{:}2\rangle\}$ $\tau(\langle x{:}2\rangle) = \emptyset$

$\tau(\langle x{:}1\rangle) = \{\langle x{:}2\rangle, \langle x{:}3\rangle\}$ $\tau(\langle x{:}3\rangle) = \emptyset$

Runs

Infinite sequence of states

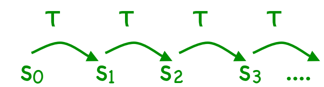
σ: $s_0\ s_1\ s_2\ s_3\ s_4$ ....

is a **run** of Φ if

- **Initiality:** $s_0 \models \Theta$ (s$_0$ is an initial state)
- **Consecution:** for all $i > 0$

  s_{i+1} is a τ-successor of s_i

  for some $\tau \in \mathcal{T}$



Runs: Example

V: {x:integer}

Θ: x=0

$\mathcal{T}$: $\{\tau_1, \tau_2, \tau_3\}$ with

$$\begin{cases} \rho_{\tau 1} : x'=x+1 \vee x'=x+3 \\ \rho_{\tau 2} : x'=x+2 \vee x'=2x \\ \rho_{\tau 3} : x'=x \end{cases}$$

σ_1: 0, 1, 2, 3, 4, 5, 6, 7, ...

σ_2: 0, 0, 0, 0, 0, 0, 0, 0 ....

σ_3: 0, 2, 4, 8, 16, 19, ...

σ_4: 0, 1, 1, 3, 3, 5, 5, 7, 7, ...

σ_5: 0, 1, 2, 3, 5, 6, 8, 9, 18, ...

Runs: Example

V: {x:integer}

Θ: x=0

$\mathcal{T}$: $\{\tau_1, \tau_2, \tau_3\}$ with

$$\begin{cases} \rho_{\tau 1} \ (x=0 \vee x=1) \wedge (x'=x+1 \vee x'=x+3) \\ \rho_{\tau 2} : x'=x+2 \vee x'=2x \\ \rho_{\tau 3} : x'=x \end{cases}$$

σ_1: 0, 1, 2, 3, 4, 5, 6, 7, ... not a run!

σ_2: 0, 0, 0, 0, 0, 0, 0, 0 ....

σ_3: 0, 2, 4, 8, 16, 32, ...

σ_4: 0, 1, 1, 3, 3, 5, 5, 7, 7, ...

σ_5: 0, 1, 2, 3, 5, 6, 8, 9, 18, ... not a run!

System Description: Summary

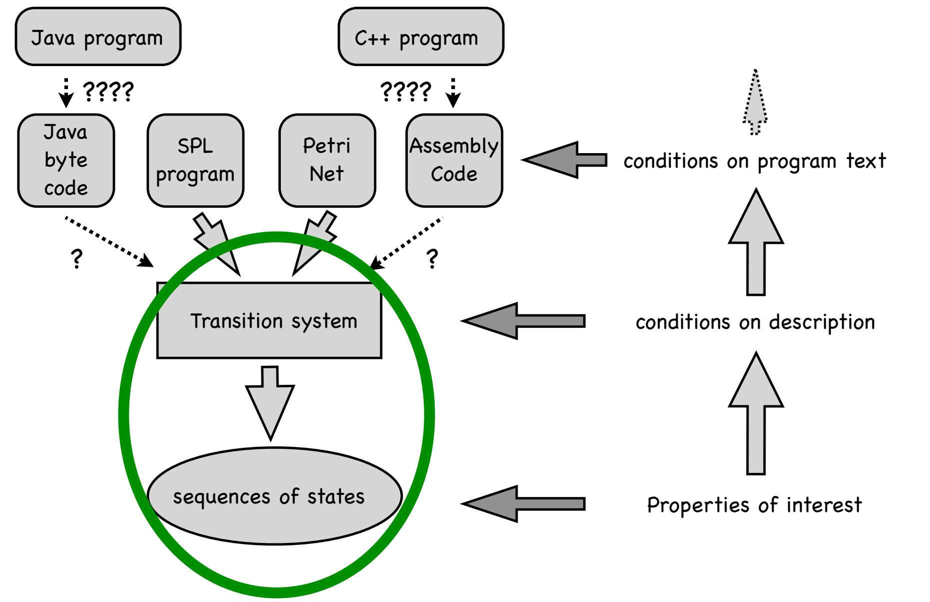
Transition system: Φ: < V , Θ , $\mathcal{T}$ >

Run: Initiality + Consecution

$\mathcal{L}(\Phi)$: all runs of Φ "Behavior of the program"

(all sequences of states that satisfy Initiality and Consecution)

Semantics



SPL

Simple programming language with constructs (a.o.):

- assignment
- conditional (if - then - else)
- concatenation
- selection
- while

Static global variable initialization

Statements are labeled

☛ define program locations (equivalence relation on labels)

SPL

Given an SPL program P we can construct the corresponding transition system Φ: < V , Θ , $\mathcal{T}$ >.

- each program statement corresponds to a transition

  no sequential structure in transition systems, therefore control is modeled explicitly by a control variable π that ranges over program locations

- V: program variables $\cup$ $\{\pi\}$
- Θ: program initial condition

SPL statements

assignment statement l_1: x := e; l_2

translates into transition τ with transition relation (y′=y for all other variables in V)

$$\rho_\tau: \quad \pi = l_1 \wedge x' = e \wedge \pi' = l_2 \wedge pres(V - \{x, \pi\})$$

conditional statement l_1: **if** c **then** l_2: S_1 **else** l_3: S_2

translates into transition τ with transition relation

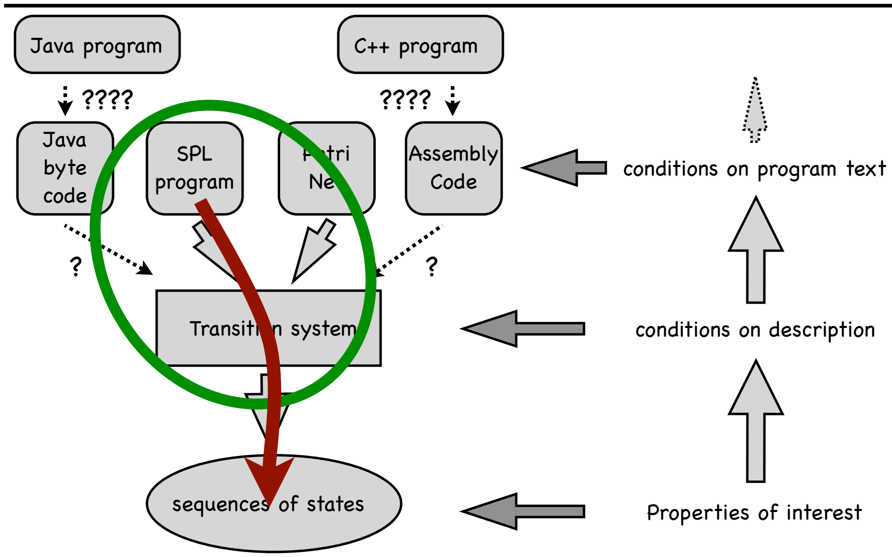
$$\rho_\tau: \quad \pi = l_1 \wedge ((c \wedge \pi' = l_2) \vee (\neg c \wedge \pi' = l_3)) \wedge pres(V - \{\pi\})$$

SPL semantics

Full semantics of SPL in

Zohar Manna, Amir Pnueli, Temporal Verification of Reactive Systems: Safety. Springer-Verlag 1995. pp 18-36.

Semantics



Reachable state space

state s is Φ-reachable if it appears in some Φ-run

σ: s_0 s_1 s_2 s_3 s_4 ....

system Φ is finite-state if the set of Φ-reachable states is finite

Notation: Σ : state space

$\Sigma_{\Phi\triangleright}$: Φ-reachable state space

Example:
V: $\{b_1, b_2\}$
Θ: $b_1 \wedge b_2$
$\mathcal{T}$: {τ} with ρ_τ: $b_1' = \neg b_1 \wedge b_2' = \neg b_2$

Σ = {<t,t>,<t,f>,<f,t>,<f,f>}

$\Sigma_{\Phi\triangleright}$ = {<t,t>,<f,f>}

Reachable state space

state s is Φ-reachable if it appears in some Φ-run

σ: $s_0\ s_1\ s_2\ s_3\ s_4$ ....

system Φ is finite-state if the set of Φ-reachable states is finite

Notation: Σ : state space
$\Sigma_{\Phi\triangleright}$: Φ-reachable state space

Example:
V: {x:int}
Θ: x=0
$\mathcal{T}$: {τ} with ρ_τ: $x=0 \wedge x'=x+1$

$\Sigma = N$

$\Sigma_{\Phi\triangleright} = \{x{:}0,\ x{:}1\}$

Reachable state space

state s is Φ-reachable if it appears in some Φ-run

σ: $s_0\ s_1\ s_2\ s_3\ s_4$ ....

system Φ is finite-state if the set of Φ-reachable states is finite

Notation: Σ : state space
$\Sigma_{\Phi\triangleright}$: Φ-reachable state space

Example:
V: {x:int}
Θ: $0 \le x \le M$
$\mathcal{T}$: $\{\tau_1,\tau_2\}$ with
$\rho_{\tau 1}$: $odd(x) \wedge x'=3x+1$
$\rho_{\tau 2}$: $even(x) \wedge x'=x/2$

$\Sigma = N$
$\Sigma_{\Phi\triangleright} = ?$

a.k.a. Collatz problem
or 3n+1 problem

Reachable state space vs runs

System Φ may have any combination of

finite state space	$\otimes$	finite # of runs
infinite state space		infinite # of runs

Invariants

An **invariant** q of program P is

- a superset of the reachable state space of P
- q is an **assertion** (first-order formula)
- also written:

$P \Vdash q$ all reachable states of P satisfy q

$P \vDash \Box q$ all states of all runs of P satisfy q

Properties of program behaviors

First-order logic

models are states

$\langle x:3, y:1 \rangle \Vdash x > y$

assertion p
represents
the set of states
for which p is true

Temporal logic

models are sequences of states

$\langle s_0\ s_1\ s_2\ s_3\ \ldots \rangle \vDash \varphi$

temporal formula φ
represents
the set of sequences of states
for which φ is true

Specification: underlying assertion language

Assertion language $\mathcal{A}$:

first-order language over system variables
(+ theories for their domains)

Formulas in $\mathcal{A}$: state formulas (aka assertions)

evaluated over a single state

$s \Vdash p$ iff $s[p] = \text{true}$

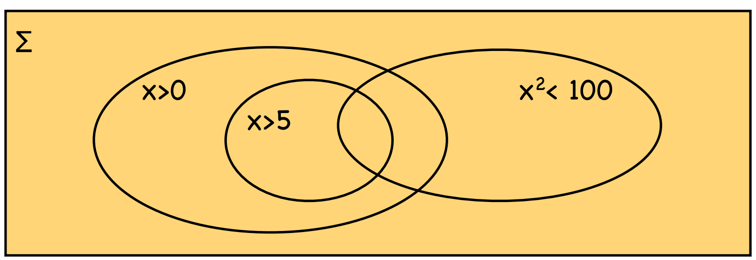
p holds at s
s satisfies s
s is a p-state

Example:
s: $\langle x:4, y:1 \rangle$

$s \Vdash x=0 \vee y=1$
$s \Vdash x > y$
$s \Vdash x = y+3$
$s \nVdash odd(x)$

Specification: underlying assertion language

Assertions represent sets of states



Specification: underlying assertion language

assertion p is state-satisfiable if $s \Vdash p$ for some state $s \in \Sigma$
Example: $x>0$

assertion p is state-valid $s \Vdash p$ for all states $s \in \Sigma$
Example: $x>y \rightarrow x+1 > y$

State validity and system state validity

Given a system Φ

- for a state formula q

  $\Vdash q$

  q holds in all states
  q is state-valid

  Example: $\Vdash x=1 \rightarrow x>0$

- for a state formula q

  $\Phi \Vdash q$

  q holds in all Φ-reachable states
  q is Φ-state-valid

  Example: $\Phi = \langle V, \Theta, \mathcal{T} \rangle$

  V: {x}
  Θ: x=0
  $\mathcal{T}$: {τ} with ρ_τ: $x' = x+2$

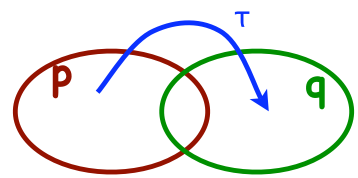
  $\Phi \Vdash x \geq 0 \wedge even(x)$

Verification condition

$$p \wedge \rho_\tau \rightarrow q'$$

Starting from a state that satisfies p, transition τ leads to a state that satisfies q

aka "Hoare triple" {p} τ {q}



Verification conditions: examples

$$p \wedge \rho_\tau \rightarrow q'$$

{p} τ {q}

{x>0} x′=x+1 {x>1} $\quad x>0 \wedge x'=x+1 \rightarrow x'>1$

substitute x+1 for x′: $x>0 \rightarrow (x+1) > 1$

{x>0} x′=x+1 {true} $\quad x>0 \wedge x'=x+1 \rightarrow true$

{x≥0} x>0 ∧ x′=x−1 {x≥0}

{true} x>0 ∧ x′=x−1 {x≥0}

Proving invariance properties

Invariant: $\Box p$ for state formula p

We want to prove $\Phi \vDash \Box p$

every state of every run of Φ satisfies p

Recall: A sequence of states σ: $s_0, s_1, s_2 \ldots$ is a run of Φ: $\langle V, \Theta, \mathcal{T} \rangle$ if

- ☛ Initiality: $s_0 \vDash \Theta$
- ☛ Consecution: for each $j \geq 0$, s_{j+1} is a τ-successor of s_j, for some $\tau \in \mathcal{T}$

Proving invariance properties

Proving $\Phi \vDash \Box p$

means proving that every state of every sequence of states that satisfies

- Initiality: $s_0 \vDash \Theta$
- Consecution: for each $j \geq 0$, s_{j+1} is a τ-successor of s_j, for some $\tau \in \mathcal{T}$

also satisfies p

Proof by induction:

Base case: $\Theta \rightarrow p$ ensures that every initial state satisfies p

Inductive step: $p \wedge \rho_\tau \rightarrow p'$ for every $\tau \in \mathcal{T}$

ensures that p is preserved by all transitions

Verification rule B-INV (basic invariance)

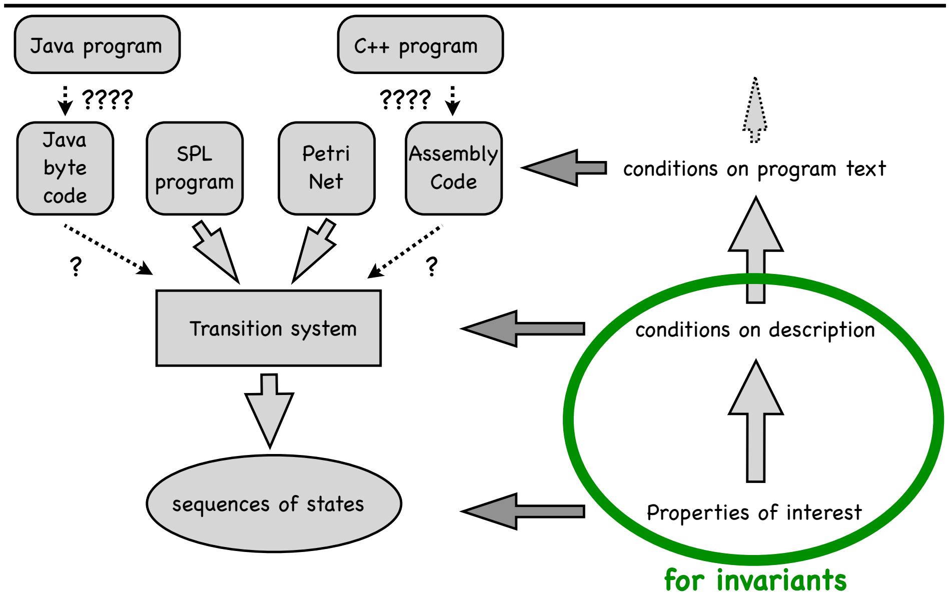
For assertion q

$$\begin{array}{ll} \text{B1.} & \Phi \Vdash \Theta \rightarrow q \\ \text{B2.} & \Phi \Vdash \{q\}\ \mathcal{T}\ \{q\} \\ \hline & \Phi \vDash \Box q \end{array}$$

$\{q\}\ \mathcal{T}\ \{q\}$ stands for $\{q\}\ \tau\ \{q\}$ for all $\tau \in \mathcal{T}$

B-INV reduces the proof of an invariant to checking the validity of $|\mathcal{T}| + 1$ first-order verification conditions in the underlying assertion language.

Semantics



B-INV : example

Φ:

V: $\{x\}$

Θ: $x=0$

$\mathcal{T}$: $\{\tau_1, \tau_2\}$ with $\rho_{\tau 1}$: $x'=x+1$

$\rho_{\tau 2}$: $x>0 \wedge x'=x-1$

to prove $\Phi \vDash \Box(x \geq 0)$

B1: $x=0 \rightarrow x \geq 0$ ✓ — B1. $\Phi \Vdash \Theta \rightarrow q$

B2: $x \geq 0 \wedge x'=x+1 \rightarrow x' \geq 0$ ✓ — B2. $\Phi \Vdash \{q\}\ \tau_1\ \{q\}$

$x \geq 0 \wedge x>0 \wedge x'=x-1 \rightarrow x' \geq 0$ ✓ — B2. $\Phi \Vdash \{q\}\ \tau_2\ \{q\}$

B-INV : example

Φ:

V: {x,y}

Θ: $x=0 \wedge y=0$

$\mathcal{T}$: $\{\tau_1,\tau_2\}$ with $\rho_{\tau 1}$: $x'=x+y \wedge y'=y+1$

$\rho_{\tau 2}$: $x>0 \wedge x'=x-1$

to prove $\Phi \vDash \square(x \geq 0)$

B1: $x=0 \wedge y=0 \rightarrow x \geq 0$ ✓

B2: $x \geq 0 \wedge x'=x+y \wedge y'=y+1 \rightarrow x' \geq 0$ ×

$x \geq 0 \wedge x>0 \wedge x'=x-1 \rightarrow x' \geq 0$ ✓

$x \geq 0$ is an invariant, but it is not inductive

Verification rule B-INV (basic invariance)

For assertion q

B1. $\Phi \Vdash \Theta \rightarrow q$

B2. $\Phi \Vdash \{q\}\ \mathcal{T}\ \{q\}$

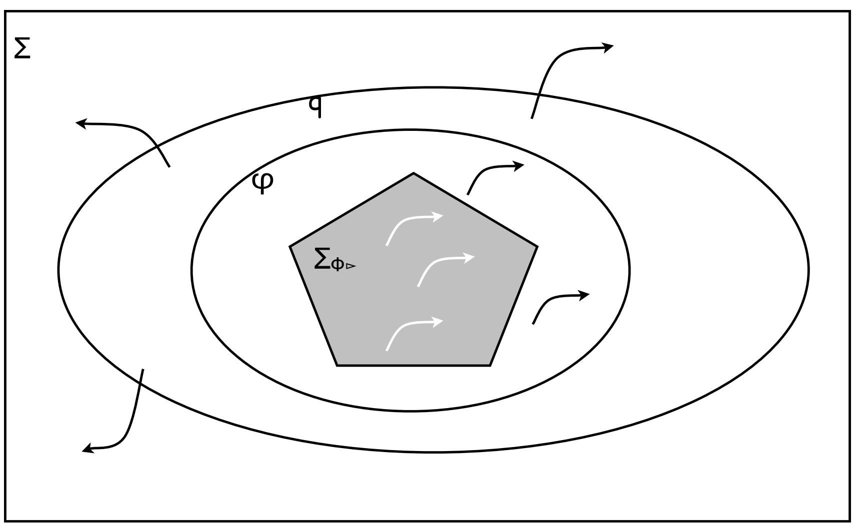
$\Phi \vDash \square q$

if B1 and B2 are (state) valid then q is inductive

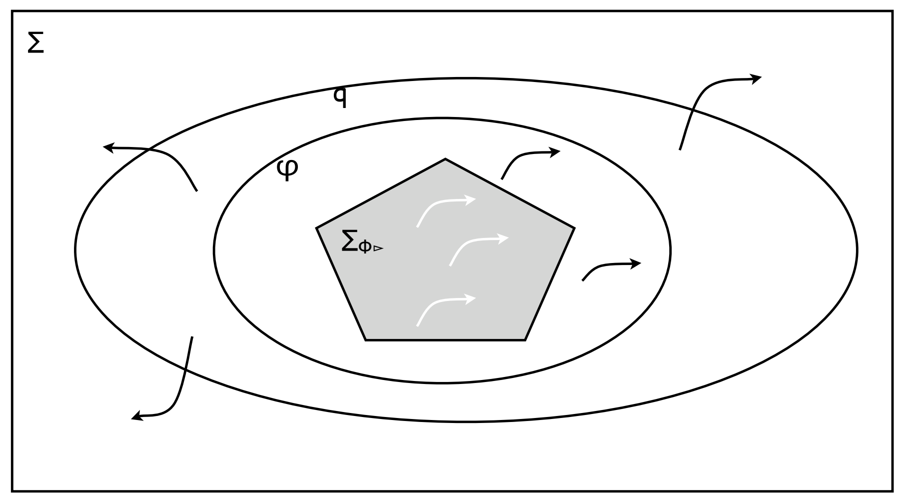
every inductive assertion is an invariant

the converse is not true: not every invariant is inductive

Non-inductive invariants



Non-inductive invariants



Strategy: strengthen q until it is inductive

Strategy 1: Strengthening

Φ:

V: {x,y}

Θ: $x=0 \land y=0$

$\mathcal{T}$: $\{\tau_1, \tau_2\}$ with $\rho_{\tau 1}$: $x'=x+y \land y'=y+1$

$\rho_{\tau 2}$: $x>0 \land x'=x-1 \land y'=y$

to prove $\Phi \vDash \Box(x \geq 0)$

strengthen it to

$\Phi \vDash \Box(x \geq 0 \land y \geq 0)$

B1: $x=0 \land y=0 \rightarrow x \geq 0 \land y \geq 0$ ✓

B2: $x \geq 0 \land y \geq 0 \land x'=x+y \land y'=y+1 \rightarrow x' \geq 0 \land y' \geq 0$ ✓

$x \geq 0 \land x>0 \land x'=x-1 \land y'=y \rightarrow x' \geq 0 \land y' \geq 0$ ✓

$x \geq 0 \land y \geq 0$ is an invariant and is inductive

Strategy 2: Incremental Proof

Φ:

V: {x,y}

Θ: $x=0 \land y=0$

$\mathcal{T}$: $\{\tau_1, \tau_2\}$ with $\rho_{\tau 1}$: $x'=x+y \land y'=y+1$

$\rho_{\tau 2}$: $x>0 \land x'=x-1 \land y'=y$

to prove $\Phi \vDash \Box(x \geq 0)$

first prove $\Phi \vDash \Box(y \geq 0)$

and then prove

$\Phi \vDash \Box(x \geq 0)$

relative to $\Box(y \geq 0)$

B1: $x=0 \land y=0 \rightarrow x \geq 0$ ✓

B2: $x \geq 0 \land y \geq 0 \land x'=x+y \land y'=y+1 \rightarrow x' \geq 0$ ✓

$x \geq 0 \land x>0 \land x'=x-1 \land y'=y \rightarrow x' \geq 0$ ✓

$x \geq 0$ is an invariant and is inductive relative to $y \geq 0$