Verification

Lecture 26

Bernd Finkbeiner
Peter Faymonville
Michael Gerke

UNIVERSITÄT
DES
SAARLANDES

Coming up next week...

End-of-term exam will take place
on Feb 9, 2pm-5pm, in HS 002
Open Book

Final problem set will be discussed in lecture on Tuesday.

REVIEW: Proving Invariance

For assertion q,

$$\begin{array}{ll} \text{B1.} & P \Vdash \Theta \rightarrow q \\ \text{B2.} & P \Vdash \{q\}\ \mathcal{T}\ \{q\} \\ \hline & P \vDash \Box q \end{array} \quad \text{B-INV}$$

- for $\tau \in \mathcal{T}$ in P

  $\{\varphi\}\tau\{\psi\}$: $\rho_\tau \wedge \varphi \rightarrow \psi'$

  "τ leads from φ to ψ in P"

- for $\mathcal{T}$ in P

  $\{\varphi\}\mathcal{T}\{\psi\}$: $\{\varphi\}\tau\{\psi\}$ for every $\tau \in \mathcal{T}$

  "$\mathcal{T}$ leads from φ to ψ in P"

Completeness of Rule INV

For assertions q, φ

$$\begin{array}{ll} \text{I1.} & P \Vdash \varphi \rightarrow q \\ \text{I2.} & P \Vdash \Theta \rightarrow \varphi \\ \text{I3.} & P \Vdash \{\varphi\}\ \mathcal{T}\ \{\varphi\} \\ \hline & P \vDash \square\, q \end{array} \quad \text{INV}$$

For every assertion q such that

$\square\, q$ is P-valid

there exists an assertion φ such that I1 − I3 are provable from state validities

Note: We actually show *completeness relative to first-order reasoning* taking all state-valid assertions as axioms.

Proof Outline

Given FTS P with system variables

$$\overline{y} = (y_1, \ldots, y_m)$$

- Assume $\square q$ is P-valid, i.e.,
  (†) q holds over every P-accessible state
- Construct (to be shown) accessibility assertion

  $$acc_P(\overline{y})$$

  such that for any state s,
  (*) s is P-accessible state iff $s \Vdash acc_P$
- Take $\varphi = acc_P$

We have to show :
1. acc_P satisfies I1 − I3
2. acc_P can be constructed

Proof

1. acc_P **satisfies I1 – I3**

- Premise I1: $\underbrace{acc_P}_{\varphi} \rightarrow q$

$$s \Vdash acc_P \overset{(*)}{\Rightarrow} s \text{ is } P\text{-accessible state}$$

$$\overset{(\dagger)}{\Rightarrow} s \Vdash q$$

Thus

$$\underbrace{acc_P}_{\varphi} \rightarrow q$$

is state-valid

Proof (cont‘d)

- Premise I2: $\Theta \rightarrow \underbrace{acc_P}_{\varphi}$

$$s \Vdash \Theta \quad \Rightarrow \quad s \text{ is } P\text{-accessible}$$

$$\stackrel{(*)}{\Rightarrow} \quad s \Vdash \underbrace{acc_P}_{\varphi}$$

Thus

$$\Theta \rightarrow \underbrace{acc_P}_{\varphi}$$

is state-valid

Proof (cont'd)

- Premise I3: for every $\tau \in \mathcal{T}$,

  $$\rho_\tau \wedge acc_P \rightarrow acc'_P$$

  where $acc'_P = acc_P(\overline{y}')$.

  Take s' to be a $\overline{y}$-variant of s (s agrees with s' on all variables other than $\overline{y}$) and for each y_i take

  $$s'[y_i] = s[y'_i]$$

  Then

  $$\left.\begin{array}{lcl} s \Vdash \rho_\tau & \Rightarrow & s' \text{ is a } \tau\text{-successor of } s \\ s \Vdash acc_P & \stackrel{(*)}{\Rightarrow} & s \text{ is } P\text{-accessible} \end{array}\right\} \Rightarrow$$

  $$\begin{array}{ll} \Rightarrow & s' \text{ is } P\text{-accessible} \\ \stackrel{(*)}{\Rightarrow} & s' \Vdash acc_P \\ \Rightarrow & s \Vdash acc'_P \end{array}$$

Proof (cont'd)

2. Construction of acc_P

Assume assertion language includes
dynamic array $\underline{a}$ over D

Array $\underline{a}$ is viewed as function,

$$a\colon\ [1..n] \mapsto D$$

where n is the size of the array

Assumption is not essential.
E.g., use encoding

$$(n_1, \ldots, n_k) \quad \rightarrow \quad n = p_1^{n_1} \cdots p_k^{n_k}$$

where p_i is the ith prime number

Proof (cont'd)

Case: single-variable y

Define

$$acc_P(y)\colon\ (\exists n > 0)\ \ (\exists a \in [1..n] \mapsto D)\,.\, init \wedge last \wedge evolve$$

where

$$init\colon \quad \Theta(a[1])$$

$$last\colon \quad a[n] \ = \ y$$

$$evolve\colon \quad \forall i\,.\, 1 \leq i < n\,.\, \bigvee_{\tau \in \mathcal{T}} \rho_\tau(a[i], a[i+1])$$

Proof (cont'd)

array a represents a prefix

$$s_1, \ldots, s_n$$

of a computation where $a[i]$ stands for

the value of y at state s_i

<u>Claim</u>:

For any value $d \in D$,

$acc_P(d) = \mathrm{T}$

iff

d is a possible value of y in a P-accessible state

Proof (cont'd)

Multivariable $\overline{y} = (y_1, \ldots, y_m)$ case

Use 2-dimensional array a

$$
\begin{array}{ccccccc}
\underline{y_1} & & & & & & \underline{y_m} \\
a[1,1] & . & . & . & . & & a[1,m] \\
a[2,1] & . & . & . & . & & a[2,m] \\
. & & & & & & . \\
. & & & & & & . \\
. & & & & & & .
\end{array}
$$

Example

V: $\{y\}$	ranges over $\mathbb{Z}$ (the integers)
Θ: $y = 0$	
ρ_τ: $y' = y + 2$	

acc_P: $(\exists n > 0)(\exists a \in [1..n] \mapsto \mathbb{Z})\,.$

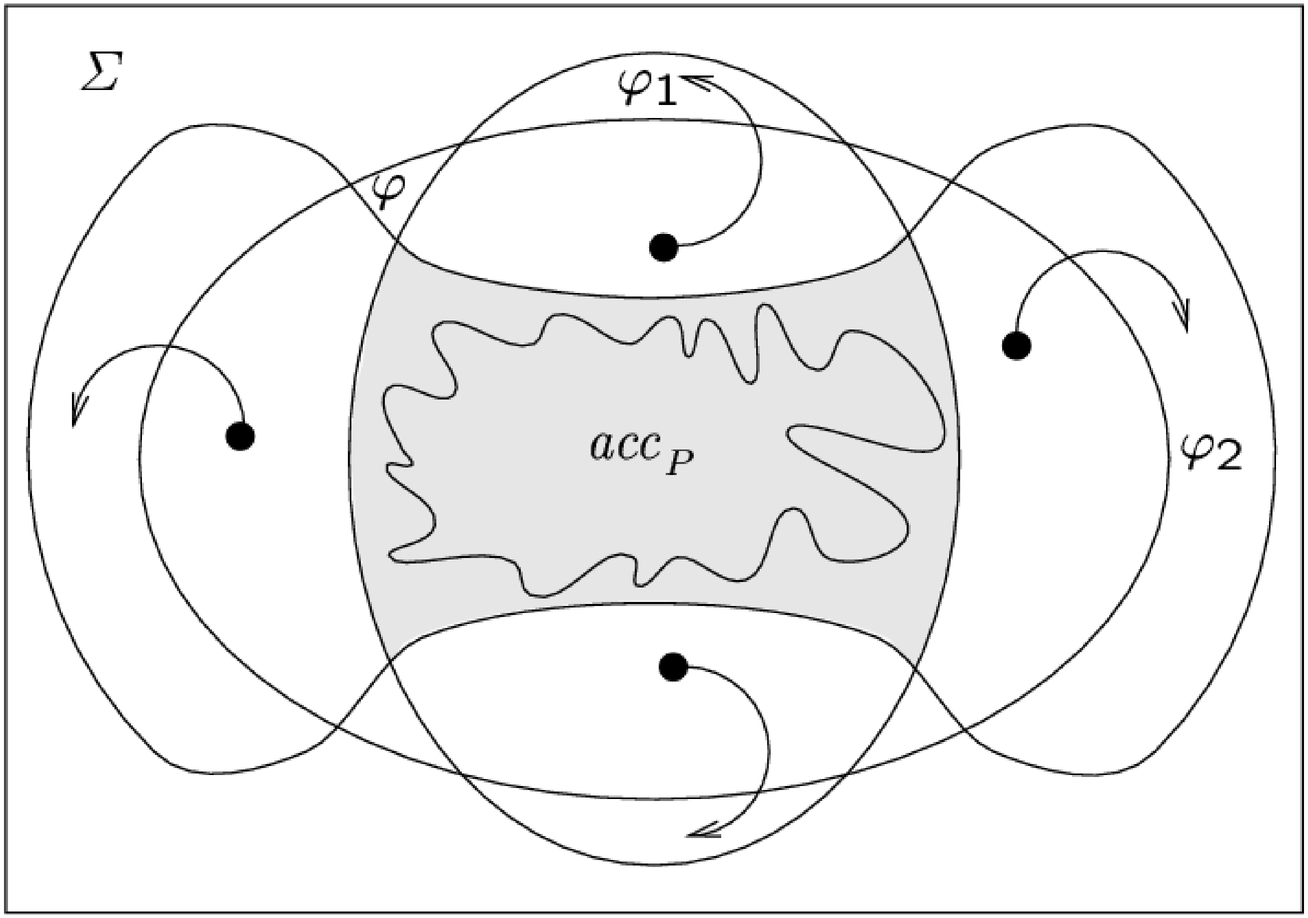
$$\begin{pmatrix} a[1] = 0 \ \wedge \ a[n] = y \ \wedge \\ \forall i\,.\,1 \le i < n\,.\,a[i+1] = a[i] + 2 \end{pmatrix}$$

simplifies to $(\exists n > 0)(\exists a \in [1..n] \mapsto \mathbb{Z})\,.$

$$\begin{pmatrix} a[n] = y \ \wedge \\ \forall i\,.\,1 \le i \le n\,.\,a[i] = 2 \cdot (i - 1) \end{pmatrix}$$

simplifies to $\boxed{y \ge 0 \ \wedge \ even(y)}$

Discussion



Although the assertion acc_P is inductive and strengthens any P-invariant, it is not very useful in practice.

Induction-based Model Checking

IC3

- incremental construction of
- inductive clauses for
- indubitable correctness

IC3

Goal: decide whether an assertion P is S-invariant for some transition system S.

Core data structure:
Sequence of formulas $F_0 = \Theta, F_1, F_2, \ldots, F_k$
that are overapproximations of the sets of states reachable in at most $1, \ldots, k$ steps.

Approach: Refine sequence such that
if P is S-invariant, some F_i will eventually become inductive.

IC3 Invariants

- $\Theta \Rightarrow F_0$
- $F_i \Rightarrow F_{i+1}$ for all $0 \leq i < k$
- $F_i \Rightarrow P$ for all $0 \leq i \leq k$
- $F_i \wedge \rho \Rightarrow F'_{i+1}$ for all $0 \leq i < k$

Initially, $k = 1$ and $F_0 = \Theta$, $F_1 = P$.
IC3 invariants initially established by checking for counterexamples of length 0 and 1.

k is increased whenever it is proven that there are no counterexamples of length k.

Main Algorithm

if $(\Theta \not\Rightarrow P$ or $\Theta \wedge \rho \not\Rightarrow P')$ **return** $\bot$;
$F_0 := \Theta; F_1 := P; k := 1;$
repeat $\{$
  while (there are CTIs in F_k) $\{$
    refine $F_1, \ldots F_k$
    if (counterexample found) **return** $\bot$
  $\}$;
  $k++$;
  $F_k := P$;
  propagate clauses
  if $(F_i = F_{i+1}$ for some $0 \leq i < k)$ **return** $\top$
$\}$

Counterexample-to-induction (CTI)

A counterexample to induction (CTI) is a state s that is

- reachable in k steps and
- that has an outgoing transition to a $\neg P$ state.

To find a CTI, check whether

$$F_k \wedge \rho \Rightarrow P$$

holds.

Refine $F_0, \ldots F_k$ (part 1)

- Suppose a CTI s exists
- If P is an invariant, then $\neg s$ is inductive relative to (at least) F_0.
  We say G is inductive relative to H iff (1) $\Theta \Rightarrow G$ and
  (2) $H \wedge G \wedge \rho \Rightarrow G'$.
  If $\neg s$ is not even inductive relative to F_0 then P is not an invariant ($\rightarrow$ counterexample).
- Pick the greatest i such that $\neg s$ is inductive relative to F_i.
- Exclude $\neg s$ from F_{i+1}.
  In principle, this could be done by setting F_{i+1} to $F_{i+1} \wedge \neg s$.
  Better: generalize $\neg s$ by dropping literals such that the subclause is still inductive relative to F_i

Propagate clauses

For any clause c of F_i

- such that $F_i \wedge c \Rightarrow c'$,
- we add c to F_{i+1},
  i.e., $F_{i+1} := F_{i+1} \wedge c$.

Refine $F_0, \ldots F_k$ (part 2)

- Previously: We excluded (the generalization of) $\neg s$ from F_{i+1}.
- This does not necessarily rule out the CTI s, if $i < k - 1$.
- In this case: Find predecessor t in $F_{i+1} \setminus F_i$
- Recur on t: eliminate t in F_{i+1}