5.2 Abstrakte Semantik zur Prädikatenabstraktion:

Sei $Abs(P)$ festgelegt.

Ziel: Bestimme $post^{\#}_{c(c')}$ mit

$$\alpha(post_{c(c')}(\gamma(q))) \leq post^{\#}_{c(c')}(q) \quad \text{für alle } q \in Abs(P).$$

Idee:
- Sei $x := a$ der Befehl, der von c zu c' führt.
- Die Menge der Zustände

$$post_{c(c')}(\gamma(q))$$

ist gegeben durch die stärkste Nachbedingung $sp(q, x:=a)$:

$$post_{c(c')}(\gamma(q)) = \{\sigma \in Stak \mid \sigma \models sp(q, x:=a)\}. \qquad (*)$$

- Dann ist

$$\alpha(post_{c(c')}(\gamma(q)))$$

$$(Mit\ *) = \alpha(\{\sigma \in Stak \mid \sigma \models sp(q, x:=a)\})$$

$$(Def.\ \alpha) = \bigsqcup\{q_\sigma \in Abs(P) \mid \sigma \models sp(q, x:=a)\}.$$

Trick:

$$\bigsqcup\{q_\sigma \in Abs(P) \mid \sigma \models sp(q, x:=a)\} \quad \models\mid \quad \overline{sp(q, x:=a)}.$$

- Mit dem Trick ist keine Konkretisierung über σ und q_σ notwendig
- Strongest Postconditions sind dennoch wegen der Nutzung von Quantoren nachteilig.

Definition (Hoare-Tripel (Sir Tony Hoare, Turing-Award 1980))
- Ein Hoare-Tripel $\{b\}\, x:=a\, \{p\}$ besteht aus zwei Booleschen Ausdrücken $b, p \in BExp$ und einem Programm, hier $x:=a$.
- Das Hoare-Tripel $\{b\}\, x:=a\, \{p\}$ heißt gültig, falls

$$\forall \sigma \in Stak \text{ mit } \sigma \models b \text{ und } \forall \sigma' \in Stak \text{ mit } (x:=a, \sigma) \to \sigma'$$

gilt: $\sigma' \models p$.

"Falls vor der Ausführung des Befehls b gilt, gilt nach der Ausführung p."

Definition (Stärkste Nachbedingung, schwächste Vorbedingung, Edsger Dijkstra, Turing-Award 1972)

- Gegeben $b \in BExp$ und $x := a$, bezeichnen wir mit
  $sp(b, x:=a)$
  die stärkste (bzgl. $\models$) Formel $p \in BExp$, für die
  $\{b\}\, x:=a\, \{p\}$ gültig ist.
  Man nennt $sp(b, x:=a)$ die stärkste Nachbedingung (strongest postcondition) von b und $x:=a$.
- Gegeben $p \in BExp$ und $x := a$, bezeichnen wir mit
  $wp(x:=a, p)$
  die schwächste Formel $b \in BExp$, für die
  $\{b\}\, x:=a\, \{p\}$ gültig ist.
  Man nennt $wp(x:=a, p)$ die schwächste Vorbedingung (weakest precondition) von $x:=a$ und p.

Satz (Dijkstra '76, aus FGDP bekannt)

(1) $sp(b, x:=a) \models \exists x' : (b\{x'/x\} \wedge x = (a\{x'/x\}))$

(2) $wp(x:=a, p) \models p\{a/x\}$.

Beobachtung:
- Beide Formeln existieren also und lassen sich berechnen.
- Stärkste Nachbedingungen benötigen allerdings Quantoren, die für Tools schwierig zu handhaben sind.
  (satisfiability modulo theories (SMT) solver).

Beispiele:

(1) $sp(y>5, x := y+3) \models \exists x' : (y>5\{x'/x\} \wedge x = (y+3\{x'/x\}))$
$\models \exists x' : (y>5 \wedge x = y+3)$
$\models y>5 \wedge x = y+3.$

(2) $sp(x>5 \wedge y>3,\ x := x+y) \models \exists x' : \big((x>5 \wedge y>3)\{x'/x\} \wedge x = (x+y\{x'/x\})\big)$

$\models \exists x' : (x'>5 \wedge y>3 \wedge x = x'+y)$

(3) $wp(x := y+7,\ x>5) \models x>5\{y+7/x\}$

$\models y+7>5 \quad \models y>-2.$

Satz (Dijkstra '76):

$sp(b, x:=a) \models p$ gdw. $b \models wp(x:=a, p)$.

Zurück zur abstrakten Transitionsrelation $\Rightarrow \subseteq (Prog \times \overline{Obs(P)}) \times ((Prog \times \overline{Obs(P)}) \cup \overline{Obs(P)})$ definiert durch:

$$\frac{}{(x:=a, q) \Rightarrow \overline{sp(q, x:=a)}} \qquad \frac{}{(skip, q) \Rightarrow q}$$

$$\frac{(c_1, q) \Rightarrow (c_1', q')}{(c_1; c_2, q) \Rightarrow (c_1'; c_2, q')} \qquad \frac{(c_1, q) \Rightarrow q'}{(c_1; c_2, q) \Rightarrow (c_2, q')}$$

$$\frac{}{(\underline{\text{if}}\ b\ \underline{\text{then}}\ c_1\ \underline{\text{else}}\ c_2\ \underline{\text{fi}}, q) \Rightarrow (c_1, \overline{q \wedge b})}$$

$$\frac{}{(\underline{\text{if}}\ b\ \underline{\text{then}}\ c_1\ \underline{\text{else}}\ c_2\ \underline{\text{fi}}, q) \Rightarrow (c_2, \overline{q \wedge \neg b})}$$

$$\frac{}{(\underline{\text{while}}\ b\ \underline{\text{do}}\ c\ \underline{\text{od}}, q) \Rightarrow (c; \underline{\text{while}}\ b\ \underline{\text{do}}\ c\ \underline{\text{od}}, \overline{q \wedge b})}$$

$$\frac{}{(\underline{\text{while}}\ b\ \underline{\text{do}}\ c\ \underline{\text{od}}, q) \Rightarrow \overline{q \wedge \neg b}}$$

Es lässt sich zeigen, dass diese abstrakte Transitionsrelation die genaueste abstrakte Semantik induziert.

Definiere dazu

$$post^{\#}_{c(c')}(q) := \begin{cases} q', & \text{falls } (c,q) \Rightarrow (c',q') \text{ bzw. } (c,q) \Rightarrow q' \\ false, & \text{sonst.} \end{cases}$$

Satz:

Die Familie von Funktionen $post^{\#}_{c(c')}$ ist die genaueste abstrakte Semantik.

Beweis:

Per Definition der genauesten sicheren Approximation ist zu zeigen

$$\alpha(post_{C(C')}(\gamma(q))) \models post^{\#}_{C(C')}(q) \text{ für alle } q \in Abs(P).$$

Betrachte den Basisfall $C = x := a$.

$$\alpha(post_{x:=a}(\gamma(q)))$$

$$(\text{Lemma 1}) = \alpha(\{\sigma \in Stak \mid \sigma \models sp(q, x:=a)\})$$

$$(\text{Def. } \alpha) = \bigsqcup \{q_\sigma \mid \sigma \models sp(q, x:=a)\}$$

$$(\text{Satz 2}) \models \overline{sp(q, x:=a)}$$

$$(\text{Def } \Rightarrow) = post^{\#}_{x:=a}(q).$$

□

Zeige die benötigten Lemmata und Sätze.

Lemma 1:

$$post_{x:=a}(\gamma(q)) = \{\sigma \in Stak \mid \sigma \models sp(q, x:=a)\}.$$

Beweis:

$$\sigma \in post_{x:=a}(\gamma(q))$$

(Def. post) gdw. $\exists \sigma' \in Stak: \sigma' \in \gamma(q)$ und $(x:=a, \sigma') \to \sigma$

(Def. γ) gdw. $\exists \sigma' \in Stak: \sigma' \models q$ und $(x:=a, \sigma') \to \sigma$

(zu zeigen) gdw. $\sigma \models sp(q, x:=a)$.

- In der letzten Äquivalenz gilt die Implikation von oben nach unten unmittelbar mit der Forderung, dass $\{q\}\ x:=a\ \{sp(q, x:=a)\}$ ein gültiges Hoare-Tripel ist.
- Für die Umkehrung sei angenommen, dass $\sigma \models sp(q, x:=a)$. Da mit dem Satz von Dijkstra

$$sp(q, x:=a) \models \exists x': (q\{x'/x\} \wedge x = (a\{x'/x\})),$$

folgt, dass es $d \in \mathbb{Z}$ gibt mit

$$\sigma[x' := d] \models \varphi\{x'/x\} \qquad (*)$$

und $\sigma[x' := d] \models x = (a\{x'/x\})$. $(**)$

Aus $(**)$ folgt die Gleichung

$$\begin{aligned}\sigma(x) &= (\sigma[x' := d])(x) \\ &= \mathcal{S}[\![a\{x'/x\}]\!](\sigma[x' := d]) \\ &= \mathcal{S}[\![a]\!](\sigma[x := d]) \qquad (***)\end{aligned}$$

Mit dem Zustand/Belegung

$\sigma[x := d]$

lässt sich zeigen

(1) $\sigma[x := d] \models \varphi$ und

(2) $(x := a, \sigma[x := d]) \longrightarrow \sigma$.

<u>Zu (1):</u>

Da $\sigma[x' := d] \models \varphi\{x'/x\}$ mit $(*)$,

folgt

$\sigma[x := d] \models \varphi$.

<u>Zu (2):</u>

$(x := a, \sigma[x := d]) \longrightarrow (\sigma[x := d])[x := \mathcal{S}[\![a]\!](\sigma[x := d])]$

Für die resultierende Belegung gilt:

$(\sigma[x := d])[x := \mathcal{S}[\![a]\!](\sigma[x := d])]$

(Def. Modifikation) $= \sigma[x := \mathcal{S}[\![a]\!](\sigma[x := d])]$

$(***) = \sigma[x := \sigma(x)]$

$= \sigma$. □

<u>Satz 2:</u>

$sp(b, x := a) \models\models \bigcup\{q_\sigma \in Abs(P) \mid \sigma \models sp(b, x := a)\}$.

Da Satz benötigt einige Lemmata.

Für das folgende Lemma 3 erweitern wir die Syntax von Disjunktionen auf <u>endliche</u> Mengen.

Lemma 3:

(1) $sp(b, x:=a) \models \bigvee \{ q_\sigma \in Abs(P) \mid \sigma \models sp(b, x:=a) \}$

(2) $\bigvee \{ q_\sigma \in Abs(P) \mid \sigma \models sp(b, x:=a) \} \models \overline{sp(b, x:=a)}$.

Satz 2 folgt unmittelbar aus Lemma 3 unter Verwendung folgender Eigenschaften.

Lemma 4:

(1) Aus $b_1 \models b_2$ folgt $\overline{b_1} \models \overline{b_2}$.

(2) $\overline{\overline{b}} \models\!\models b$

(3) $\sigma \models q_\sigma$.

Beweis (von Satz 2):

„$\models$" Mit Lemma 3.(1) gilt

$$sp(b, x:=a) \models \bigvee \{ q_\sigma \in Abs(P) \mid \sigma \models sp(b, x:=a) \}.$$

Wendet man auf diese Folgerung Lemma 4.(1) an, erhält man

$$\overline{sp(b, x:=a)} \models \overline{\bigvee \{ q_\sigma \in Abs(P) \mid \sigma \models sp(b, x:=a) \}}.$$

Es gilt also per Definition von $\sqcup$:

$$\overline{\bigvee \{ q_\sigma \in Abs(P) \mid \sigma \models sp(b, x:=a) \}} \models\!\models \bigsqcup \{ q_\sigma \in Abs(P) \mid \sigma \models sp(b, x:=a) \},$$

womit

$$\overline{sp(b, x:=a)} \models \bigsqcup \{ q_\sigma \in Abs(P) \mid \sigma \models sp(b, x:=a) \}$$

folgt.

„$\models$" Für die Rückrichtung liefert Lemma 3.(2):

$$\bigvee \{ q_\sigma \in Abs(P) \mid \sigma \models sp(b, x:=a) \} \models \overline{sp(b, x:=a)}.$$

Mit Lemma 4.(1) folgt

$$\overline{\bigvee \{ q_\sigma \in Abs(P) \mid \sigma \models sp(b, x:=a) \}} \models \overline{\overline{sp(b, x:=a)}}.$$

Da wiederum

$$\overline{V\{q_\sigma \in Abs(P) \mid \sigma \models sp(b, x:=a)\}} \models\!\models \bigwedge\{q_\sigma \in Abs(P) \mid \sigma \models sp(b, x:=a)\}$$

und andererseits

$$\overline{\overline{sp(b, x:=a)}} \models\!\models \overline{sp(b, x:=a)}$$

mit Lemma 4.(2),

folgt

$$\bigwedge\{q_\sigma \in Abs(P) \mid \sigma \models sp(b, x:=a)\} \models \overline{sp(b, x:=a)}$$

wie gefordert. □

Es bleibt Lemma 3 zu zeigen (Lemma 4 ist eine Übungsaufgabe).

Beweis (von Lemma 3):

Zu (1): Gelte $\sigma \models sp(b, x:=a)$.

Da $\sigma \models q_\sigma$ mit Lemma 4.(3),

folgt unmittelbar die Disjunktion:

$$\sigma \models V\{q_\sigma \in Abs(P) \mid \sigma \models sp(b, x:=a)\}.$$

Zu (2): Gelte $\sigma \models V\{q_\sigma \in Abs(P) \mid \sigma \models sp(b, x:=a)\}$.

Es gilt $\overline{sp(b, x:=a)} \models\!\models \bigwedge\{l \in P_U \cup \neg P \mid sp(b, x:=a) \models l\}$.

Um $\overline{sp(b, x:=a)}$ aus der Disjunktion zu folgern,

genügt es also, jedes $l \in P_U \cup \neg P$ zu folgern,

für das

$$sp(b, x:=a) \models l \quad \text{gilt.}$$

Betrachte also so ein l.

Da $\sigma \models V\{q_\sigma \in Abs(P) \mid \sigma \models sp(b, x:=a)\}$,

folgt $\sigma \models q_{\sigma'}$ für ein $\sigma' \in Stat$ mit

$$\sigma' \models sp(b, x:=a).$$

Da $\sigma' \models sp(b, x:=a)$ und da $sp(b, x:=a) \models l$,

folgt $\sigma' \models l$.

Damit ist

$$l \in \{ l' \in P_\cup \neg P \mid \sigma' \models l' \}.$$

Diese Menge definiert wiederum $q_{\sigma'}$:

$$q_{\sigma'} = \bigwedge \{ l' \in P_\cup \neg P \mid \sigma' \models l' \}.$$

Dann

$$\sigma \models q_{\sigma'},$$

heißt also insbesondere

$$\sigma \models l.$$

□

Beobachtungen (zu abstrakten Transitionsrelation):

- Eigentlich sind auch $q \wedge b$ sowie $q \wedge \neg b$ stärkste Nachbedingungen bei Conditionals. Insbesondere werden in diesen Fällen die Guards zum Prädikat hinzugefügt.
- Abstrakte Konfigurationen der Form $(c, false)$ repräsentieren keine erreichbaren Konfigurationen der konkreten Semantik, denn $\sigma \models false$ gilt für keinen Zustand.

Diese abstrakten Konfigurationen $(c, false)$ kann man weglassen.

Beispiel:

```
if [x>y]^1 then
  while [y≠0]^2 do
    [x:=x-1]^3;
    [y:=y-1]^4;
  od
  if [x>y]^5 then
    [skip]^6;
  else
    [skip]^7;
  fi
else
  [skip]^8;
```

↳ Die while-Schleife wird nur betreten, wenn $x > y$ gilt.

↳ In der Schleife werden x und y gleichmäßig dekrementiert.

↳ Damit ist $x > y$ eine Invariante der Schleife, die beim Verlassen gilt.

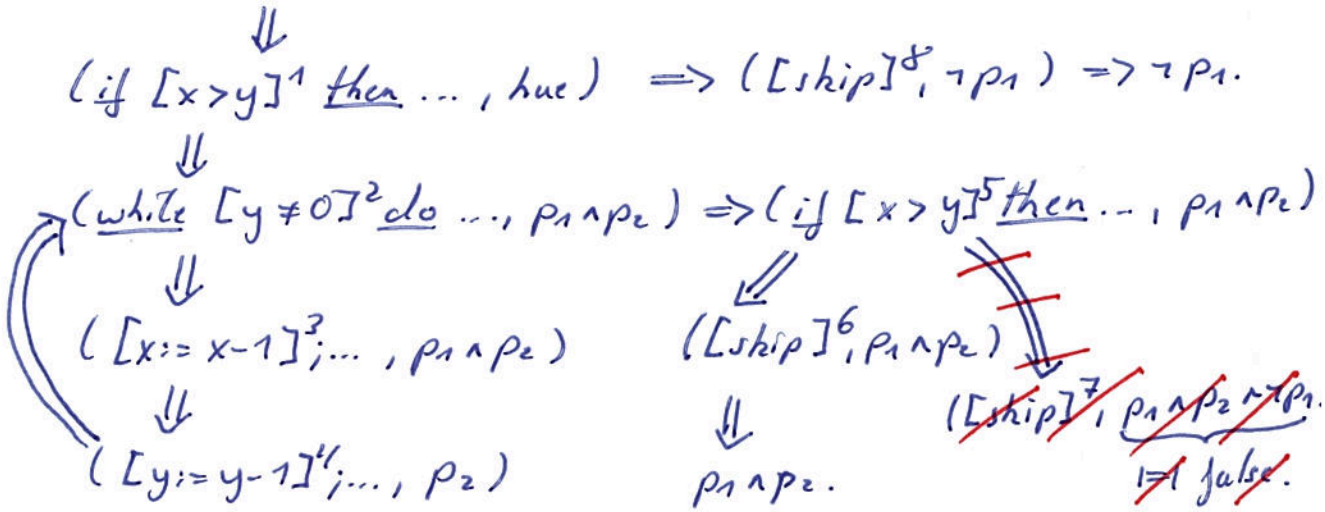
↳ Weil die Invariante in Block 5 erfüllt ist, wird Block 7 nie betreten.

Ziel: · Beweise dieses Verhalten automatisch mit Prädikatenabstraktion.

· Verwende

$p_1 := x > y$ und $p_2 := x \geq y$.

↳ Das abstrakte Transitionssystem ist



↳ Abstrakte Konfigurationen der Form (c, false) werden weggelassen, zum Beispiel $([skip]^7, false)$. Damit ist abzulesen, dass Block 7 nicht erreichbar ist.

↳ Prädikatenabstraktion mit $p_1 = x > y$ alleine wäre nicht erfolgreich gewesen: unklar, ob p_1 nach Block 3 noch gilt.

Um Prädikatenabstraktion zu implementieren, ist es nötig, $\overline{sp(q, x:=a)}$ zu berechnen.

Der folgende Satz reduziert das Problem auf die Prüfung von Implikationen auf Gültigkeit.

Satz (Graf, Saidi 1997):

$$\overline{sp(q, x:=a)} \vdash\!\dashv \bigwedge \{ \ell \in P \cup \neg P \mid q \rightarrow wp(x:=a, \ell) \}.$$

Beweis:

$$\overline{sp(q, x := a)}$$

(Lehte VL) $\models l \wedge \{ l \in P \cup \neg P \mid sp(q, x := a) \models l \}$

(Satz von Dijkstra) $\models l \wedge \{ l \in P \cup \neg P \mid q \models wp(x := a, l) \}$

(Logik) $\models l \wedge \{ l \in P \cup \neg P \mid \models q \rightarrow wp(x := a, l) \}$. □

- Im Allgemeinen ist $\overline{sp(q, x := a)}$ dennoch nicht berechenbar (wegen Unentscheidbarkeiten in First-Order Logik).
- Um zu prüfen, ob

  $\models q \rightarrow wp(x := a, l)$,

  nutze SMT-Solver oder automatische Theorembeweiser.
- Damit der SMT-Solver terminiert, sollte die Formel in einem entscheidbaren logischen Fragment liegen.

Beispiele:

Theorie	Bedeutung	Entscheidbarkeit: voll	quantorenfrei
T_E	Gleichheit	✗	✓
T_{PA}	Peano-Arithmetik (mit +, ·)	✗	✗
$T_{\mathbb{N}}$	Presburger-Arithmetik (mit +)	✓	✓
$T_{\mathbb{Z}}$	ganze Zahlen (mit +)	✓	✓
$T_{\mathbb{Q}}$	rationale Zahlen (mit +)	✓	✓
$T_{\mathbb{R}}$	reelle Zahlen (mit +, ·)	✓	✓
T_{RDS}	rekursive Datenstrukturen (z.B. Listen)	✗	✓
T_{RDS}^{+}	azyklische rekursive Datenstrukturen	✓	✓
$T_{A}^{(=)}$	Arrays (mit Extensionality-Axiom)	✗	✓

Siehe auch (Table 3.1, Seite 90, Bradley & Manna):

↳ A. Bradley, Z. Manna: The Calculus of Computation, Springer 2007.

↳ D. Kroening, O. Strichman: Decision Procedures, Springer 2008.

- Neben Terminierung ist ein weiteres Problem, dass die wp-Berechnung das logische Fragment verlassen kann.

Zum Beispiel gilt:

$y > z$ ist in Presburger-Arithmetik,

aber

$wp(y := x \cdot x,\ y > z)$

$\vdash\!\!\!|\ x \cdot x > z$ ist nicht in Presburger-Arithmetik ausdrückbar.

Lösung 1: Überapproximation

- Verwende schwächere Nachbedingungen als

$$sp(q, x := a) \vdash\!\!\!|\ \bigwedge \{ l \in P \cup \neg P \mid \models q \rightarrow wp(x := a, l) \}$$

- Die schwächere Nachbedingung enthält nur diejenigen $l \in P \cup \neg P$, die sich mit einem Theorembeweiser (Resolution) oder SMT-Solver zeigen lassen.
- Um Terminierung zu erzwingen, wird ein Timeout gesetzt. Lässt sich l bis zu dem Timeout nicht zeigen, wird l nicht zur Konjunktion hinzugefügt.
- Mit dieser Methode lassen sich auch Theorien verwenden, die im Allgemeinen unentscheidbar sind.

Lösung 2: Eingeschränkte Programmklassen

- Verwende nur Klassen von Programmen, bei denen die wp-Berechnung nicht aus dem logischen Fragment hinausläuft.

  Zum Beispiel:

  Verwendet das Programm nur Addition und werden Prädikate der Presburger-Arithmetik gemacht, dann ist die wp-Berechnung unkritisch.
- Programme über endlichen Datenbereichen.