Abstrakte Interpretation

Alexander Pretschner, 14.11.00

Dieses geschieht:

- Abstrakte Interpretation ist Technik der statischen Programmanalyse (Optimierung)
  - klassische Anwendungen: Intervallgrenzen, Codeerreichbarkeit, Konstantenpropagation, Mode-Analyse in LP
  - nicht so klassisch: unendliche reaktive Systeme, abstraktes Constraint-Lösen
- „Rechnen mit Mengen von Zuständen“
  - vereinfachte Erläuterung der Konzepte mit drei Beispielen
- Abstraktionsfunktionen für Daten und Operatoren mit Galois-Verbindungen
- Anwendung: Prädikatenabstraktion
- Zusammenfassung+Kohäsionsförderung

Multiplikation falsifizieren

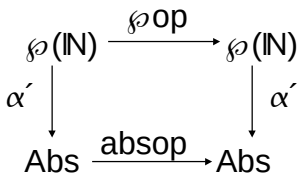
- Restklassenarithmetik: $x*y=z \Rightarrow [[x]_9 * [y]_9]=[z]_9$
- 140 * 256 = 35840: [5*4] = [2] ⇒ [2] = [2] ✔
  140 * 256 ≠ 2: [2] = [2] ✘
  140 * 256 ≠ 35841: [2] ≠ [3] ✔
- Wenn „Ungleichheit modulo 9“ gilt, dann war das Ergebnis der Multiplikation inkorrekt
- Wenn „Gleichheit modulo 9“ gilt, dann heißt das gar nichts!

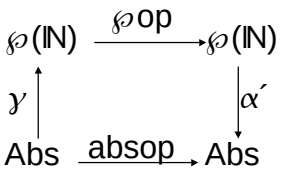
⇒ Abstraktion, die negative Eigenschaften erhält, positive aber nicht unbedingt

Was ist passiert?

- Syntax mit `mal`, `gleich` und natürlichen Zahlen
- Standardsemantik: $\|n_1\ \texttt{mal}\ n_2\| = n_1 * n_2$
  $\|n_1\ \texttt{gleich}\ n_2\| \in \{\text{true,false}\}$
- Datenabstraktion: $\alpha(n) = n \,\%\, 9 =: [n]$
- Abstrakte Operatoren: $\|n_1\ \texttt{mal}\ n_2\| = (n_1\%9 * n_2\%9)\%9$
  $\|n_1\ \texttt{gleich}\ n_2\| \in \{\text{fehler,weißnicht}\}$
- Hier: Effiziente Berechnung des abstrakten `mal`; eigentlich Rechnen mit **Mengen von Werten**:
  $\gamma([n]) = \{m \mid m\%9 \equiv n\}$
  $\|n_1\ \texttt{mal}\ n_2\| = \alpha(\ \gamma([n_1])\ \wp(*)\ \gamma([n_2])\ )$

Beobachtungen

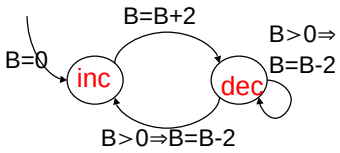
- Konkreter Datenraum wird abstrahiert:
  Natürliche Zahlen ⇒ endliche Menge von Restklassen
- Konkrete Operatoren werden abstrahiert:
  Effizientes Rechnen mit Mengen durch Rechnen mit Repräsentanten (funktioniert wg. Restklassenarithmetik)
- 



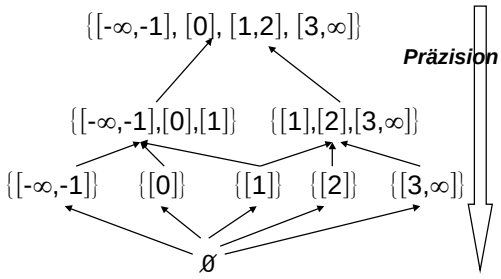
  absop ∘ α´ = α´ ∘ ℘op
  Definition „von Hand“

  α´ ∘ ℘op ∘ γ = absop
  „induzierte Operatoren“
- Eigenschaftserhaltung: A ⊨ P ⇒ C ⊨ P

Intervallabstraktionen



System

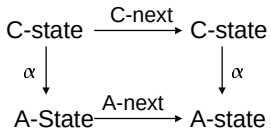


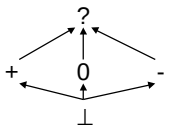
Abstrakter Verband A

- Potentiell zustandsunendlich, Abstraktion endlich
- Vervollständigung der ursprünglichen Abstraktion
- α(Z) ist kleinstes Element in A, das alle aus Z enthält
- Induzierter Operator +
  - „{[-∞,-1]} + 2 = {[-∞,1]} = {[-∞,-1], [0] ,[1]}“
  - „ {[0]} - 2 = {[-2]} ⊆ {[-∞,-1]}“

Vorzeichenabstraktion

- Abstraktionsfkt. müssen keine Homomorphismen sein





- p: Y:=X+Y; goto q, aktueller Zustand σ=(p,[X/1,Y/-2])
  - α(c-next(σ)) = α((q, [X/1,Y/-1])) = (q, [X/+,Y/-])
  - a-next(α(σ)) = a-next((p, [X/+,Y/-])) = (q, [X/+,Y/?])

- Deshalb obere, „sichere“, Approximation:
  a-next ∘ α ≥ α ∘ c-next (*) bzw. a-next ≥ α ∘ c-next ∘ γ
  - induzierte Operatoren sind die präzisesten mit (*)

- p: if X>Y then goto q else goto r
  - a-next((p, [X/+,Y/+]))=({**q,r**},[X/+,Y/+]): Rechnen mit **Zustandsmengen**

Korrektheit der Abstraktion

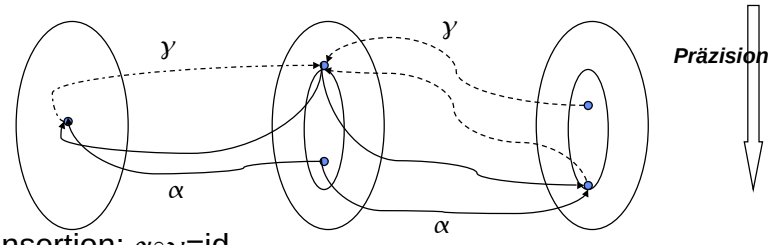
- Konkreter Bereich C
  - z.B. Potenzmengenverband des Datentyps
- Abstrakter Bereich A
  - zweckmäßigerweise (un-)endlicher vollständiger Verband
- Abstraktion α:A→C, Konkretisierung γ:C→A
  - total und monoton
  - konkreter Operator op monoton(bzgl. Mengeninklusion)
- Wenn α und γ eine Galois-Verbindung bilden, dann ist eine Abstraktion mit α ∘ op ∘ γ ≤ absop korrekt:

  lfp(op) ≤ γ**(lfp(absop))** bzw. α**(lfp(op))** ≤ **lfp(absop)**

- d.h.: AI berechnet obere Fixpunktapproximation!

Galois-Verbindungen

- α, γ total, monoton mit $\alpha \circ \gamma \leq id$ und $\gamma \circ \alpha \geq id$



- Insertion: $\alpha \circ \gamma = id$, falls γ injektiv

- α und γ legen einander eindeutig fest: $\gamma(a) = lub\{c \mid \alpha(c) \leq a\}$
- $\alpha(c) \leq a \Leftrightarrow \gamma(a) \leq c$ für totale Funktionen
- „sequentielle“ Komposition zweier GC ergibt GC
- „Komponentenweise“ Komposition ergibt GC
- $\alpha \circ \gamma \circ \alpha = \alpha$, $\gamma \circ \alpha \circ \gamma = \gamma$, ...

Zwischenergebnis

- Konkreter Potenzmengen-, abstrakter Datenverband
  - Vollständigkeit wegen Existenz der lubs
  - nicht unbedingt endlich
- Duale Abstraktions- und Konkretisierungsfunktion für die Datentypen bilden Galois-Verbindung
- Operatoren werden induziert oder (meist) approximiert
- Dann sind abstrakte Fixpunkte obere Approximation der konkreten
- Prinzip: „Sammelsemantik“ - abstrakte Beschreibungen von Mengen von Werten oder Kontrollzuständen
- „Abstraktion = mehr Nichtdeterminismus“: if nicht eindeutig

Anwendung: Prädikatenabstraktion (1)

- Programmsemantik $post[t^*]I = lfp^{\Leftarrow}(\lambda\phi.\ I \vee post[t]\phi)$ für Postconditions $post[t](\phi) = \exists q'.t(q',q) \wedge \phi(q')$
  - Programmanalyse ist Lösen rek. GLS, Model Checking ggf. auch
  - „Sammelsemantik“ - Beginne in I, nimm alle Transitionen, iteriere Verfahren.
- Korrektheit einer Abstraktion:

$$\begin{array}{ccc} q_{Abs} & \xrightarrow{t_{Abs}} & t_{Abs}(q_{Abs}) \\ \gamma \downarrow & & \downarrow \gamma \\ & & \geq \\ \gamma(q_{Abs}) & \xrightarrow{post[t]} & post[t](\gamma(q_{Abs})) \end{array}$$

- Für guarded commands $g_i(\vec{x}) \Rightarrow ass_i(\vec{x})$ sind Quantoren eliminierbar

Prädikatenabstraktion (2)

- Abstrakter Zustandsraum induziert durch Menge von Prädikaten ψ_i: Konjunktionen von (abstrakten) Literalen
  - $\pi_1\pi_2, \neg\pi_1\pi_2, \pi_1\neg\pi_2, \neg\pi_1\neg\pi_2, \pi_1, \pi_2, \neg\pi_1, \neg\pi_2$ mit Implikationsordnung, lub schwächer als $\vee$: Monomverband M
- Konkretisierung: $\gamma(\pi)=\pi[\psi_i/\pi_i]$ - Ersetzung
- Konkrete Transitionen $t_i = g_i \Rightarrow ass_i$, abstrakte a_i, $m \in M$
  - $a_i(m)$ = false, falls $\gamma(m) \Rightarrow \neg g_i$
  - $= \bigwedge m_k$ mit $m_k =$ π_k, falls $post[t_i](\gamma(m)) \Rightarrow \psi_k$
  - $\neg\pi_k$ falls $post[t_i](\gamma(m)) \Rightarrow \neg\psi_k$
  - true sonst
- Abstraktion korrekt für „pfad“universelle Eigenschaften
  - Fixpunkte im Abstrakten approximieren Fixpunkte im Konkreten
- Rechnen mit Mengen von Zuständen: Implikationen beweisen

Fixpunktapproximation

- Kleene-Sequenz erreicht lfp für stetige Funktionen auf cpos/vollst. Verbänden - aber wann? Nichtstetigkeit?
- Beschleunigung durch „Widening“
  - Supremum-Operator sup (eine obere Schranke)
  - Alle mit sup gebildeten aufsteigenden Ketten stabilisieren
  - also: statt $\mathbf{f_n = f(f_{n-1})}$ berechnet man $\mathbf{f_n = f_{n-1}\ sup\ f(f_{n-1})}$
- Dadurch „Überschießen“ des kleinsten Fixpunkts - sichere Approximation!
  - Verwandte Technik, „Narrowing“ kehrt dann wieder um (dual für gfp)
- Beschleunigung allein durch Modifikation der Operatoren
  - Problem: Widening-Operator geschickt definieren
- Anwendung z.B. in HyTech (AI von linearen GLS)
  - Widening: konvexe Hülle

Zusammenfassung

- AI: statische Programmanalyse
  - Fixpunktapproximation durch Rechnen mit Zustandsmengen
  - Abstraktionsbeziehung von Verbänden mit Galois-Verbindungen
  - Approximation der Operatoren
- Zusammenhang mit (Bi-)Simulation, Homomorphismen
- Abstraktion: „mehr Nichtdeterminismus“
- Erhalt von Eigenschaften?
  - Pfadquantifizierung universell vs. existentiell
  - Möglichkeit: verschiedene abstrakte Transitionsrelationen
- Problem: Wo kommen Abstraktionen her?