Testen und abstrakte Interpretation

Elena Vorobev
Universität Bremen
Fachbereich 3, Mathematik und Informatik

Universität Bremen, Cartesium, 13.10.09

Motivation

SAMS

- Bereits vorgestellt: formale Verifikation mit Isabelle
  - → Nachweis funktionaler Korrektheit
- Aber: im zugrundeliegenden mathematischen Modell werden Probleme der Computerarithmetik nicht berücksichtigt
  - → auftretende Rundungsfehler
  - → Möglichkeit arithmetischer Unter-/Überläufe
- Daher: zusätzliche Prüfung dieser Eigenschaften durch
  - → Testen
  - → abstrakte Interpretation

Testen und abstrakte Interpretation sind zwei grundsätzlich verschiedene Methoden!

Testen vs. abstrakte Interpretation

SAMS

Testen ...	Abstrakte Interpretation ...
• ist ein dynamisches Verfahren • prüft Testlingverhalten nur für einzelne Werte • i.d.R. Prüfung nur einer Untermenge des Testlingverhaltens realistisch	• zählt zur statischen Analyse • kann Testlingverhalten für ganze Wertebereiche prüfen • Prüfung des gesamten Testlingverhaltens möglich (→ Verifikation)

- Wozu zwei unterschiedliche Verfahren?
  - → abstrakter Interpreter noch im Entwicklungsstadium
  - → für die Zertifizierung bilden Tests die Bewertungsgrundlage
  - → abstrakte Interpretation als optionale Zusatzprüfung

Tests

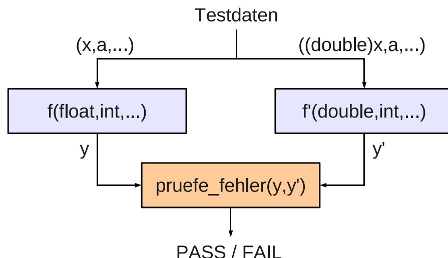
Rundungsfehlertests

SAMS

- Realisierung als Bottom-Up-Integrationstests
  - $\longrightarrow$ Berücksichtigung von Fehlerfortpflanzung und -akkumulation
  - $\longrightarrow$ Funktion - Unterfunktion nicht unabhängig voneinander
- Definition von Fehlerschranken je Größe
  - $\longrightarrow$ relative Fehlerschranke für Längen in mm: 10^{-3}, ...
- Testdatenauswahl anhand typischer Szenarien und Grenzwerte
  - $\longrightarrow$ Geradeausfahrt vorwärts / rückwärts, ...
  - $\longrightarrow$ Fahrt mit Maximalgeschwindigkeit, ...

Rundungsfehlertests (Forts.)

- Testling `f` verwendet Fließkommaarithmetik einfacher Genauigkeit (`float`)
- Referenz `f'` (Testlingkopie) verwendet Fließkommaarithmetik doppelter Genauigkeit (`double`)
- Rundungsfehler von `f` wird relativ zum Ergebnis von `f'` berechnet



Unter-/Überlauftests

- Realisierung als Modultests
- Testdatenauswahl richtet sich am Ziel einen Unter-/Überlauf in (Zwischen-)Ergebnissen zu provozieren
- Unterfunktionen werden durch Stubs ersetzt, die geeignete Werte zurückgeben

Abstrakte Interpretation

Datentypabstraktion

SAMS

Grundidee: Interpretation eines Programms „mit allen Werten gleichzeitig“

- Abstraktion numerischer Datentypen `T` durch Intervallverbände `Interval<T>`
- Abstraktion des Datentyps `bool` durch Boolverband
- Abstraktion komplexerer Datentypen entsprechend ihrer Definition

  aus `struct Vektor2D { float x; float y; }`

  wird `struct Vektor2D { Interval<float> x; Interval<float> y; }`
- jede Operation muss auf den abstrakten Datentyp geliftet werden

  → Beispiel: Addition $x + y$

  $[x_1, x_2][+][y_1, y_2] = [\{x + y | x \in [x_1, x_2], y \in [y_1, y_2]\}] = [x_1 + y_1, x_2 + y_2]$

  → Ergebnisintervall enthält alle möglichen Resultate

Rundungsfehlerberechnung, Unter-/Überlaufprüfung

SAMS

- Rundungsfehlerberechnung und Unter-/Überlaufprüfung erfolgen in der Intervallbibliothek
- Beispiel: Addition $[x_1, x_2][+][y_1, y_2] = [x_1 + y_1, x_2 + y_2]$

```
Interval<T> operator+(Interval<T> i1, Interval<T> i2) {
   Interval<T> iOut = Interval<T>();
   ...
   iOut.calcErrAdd(i1, i2); // Rundungsfehlerberechnung
   iOut.checkAdd(i1, i2); // Unter-/Überlaufprüfung
   ...
   T lb = addDown(i1.lower(), i2.lower()); // Rundungsmodus → -inf
   T ub = addUp(i1.upper(), i2.upper()); // Rundungsmodus → +inf
   iOut.assign(lb, ub);
   return iOut;
}
```

- Rundungsfehlerberechnung hängt von der Operation und den Rundungsfehlern der Operanden ab
- Rundungsfehler und Unter-/Überlaufflags sind Objektattribute von `Interval<T>` und können anschließend abgefragt werden

Beispiel für abstrakte Interpretation

SAMS

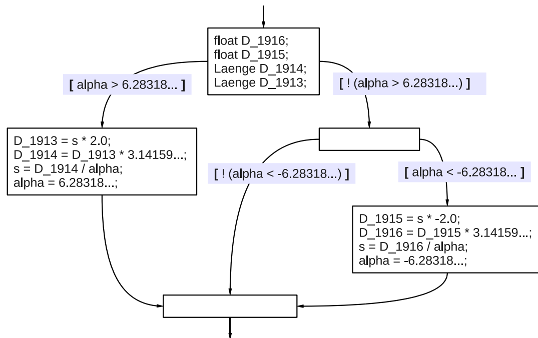
- Begrenzung einer Konfiguration (s,alpha)
  - → Winkel `alpha` wird auf eine Kreisumdrehung $+2\pi$ bzw. -2π begrenzt
  - → Bogenlänge `s` wird proportional angepasst
- Datentypen `WinkelRad` und `Laenge` sind `typedefs` auf `float`
- Konstante `sams_pi` ist definiert als `3.14159265358979323846`

```
void alpha_begrenzen( WinkelRad alpha, Laenge s ) {
   if ( alpha > 2.0f * sams_pi ) {
      s = s * 2.0f * sams_pi / alpha;
      alpha = 2.0f * sams_pi;
   } else if ( alpha < -2.0f * sams_pi ) {
      s = -s * 2.0f * sams_pi / alpha;
      alpha = -2.0f * sams_pi;
   } else {
      /* keine Änderung */
   }
}
```

Maximaler Rundungsfehler von `s` und `alpha` nach der Ausführung?

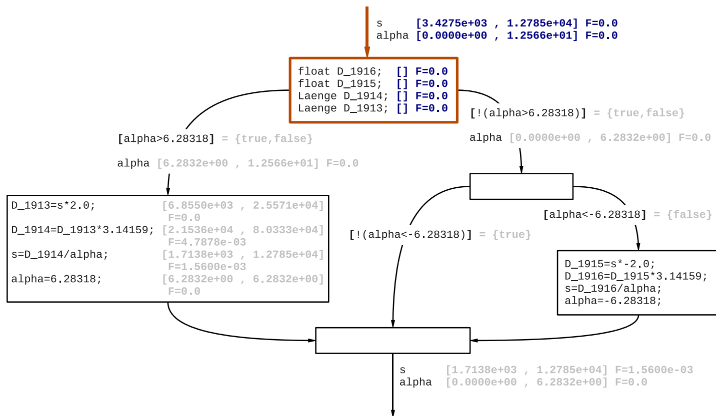
Beispiel für abstrakte Interpretation (Forts.)

- Vorbereitung: Transformation in die Sprache GIMPLE
  - → 3-Adress-Code Sprache
  - → wird vom GCC zur internen Funktionsdarstellung verwendet
  - → semantisch äquivalente Übersetzung unter Verwendung von Hilfsvariablen



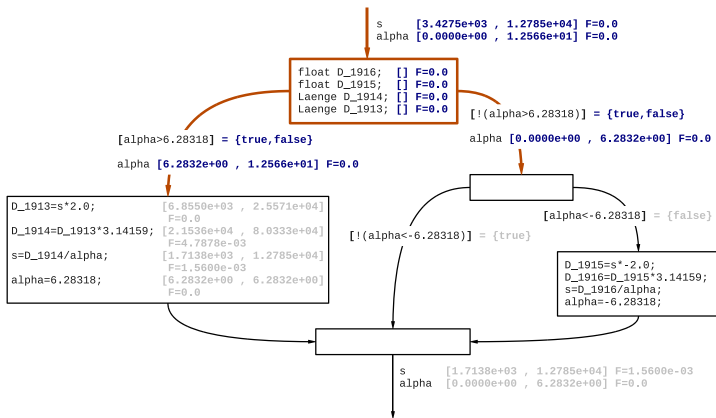
Beispiel für abstrakte Interpretation (Forts.)

- Initialisierung der Eingabeparameter mit gewünschten Wertebereichen
- Initialisierung lokaler Variablen mit [] (undefiniert)



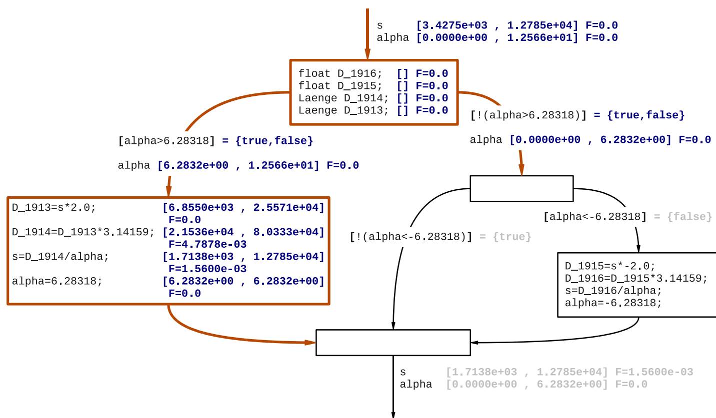
Beispiel für abstrakte Interpretation (Forts.)

- Bedingung wertet zu {`true`,`false`} aus; beide Zweige müssen interpretiert werden
- Wertebereich von `alpha` kann eingeschränkt werden



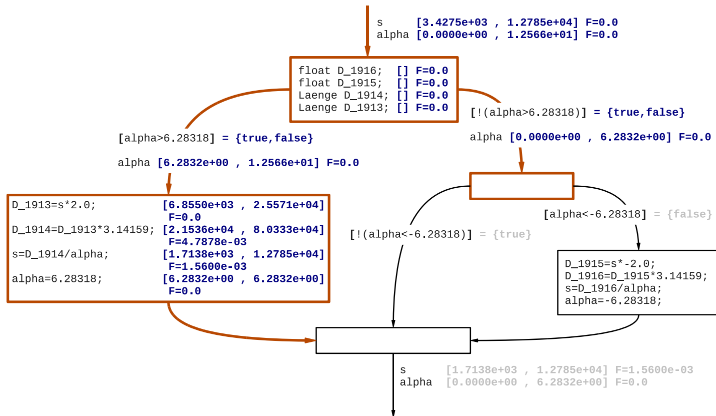
Beispiel für abstrakte Interpretation (Forts.)

- Interpretation des linken Zweiges



Beispiel für abstrakte Interpretation (Forts.)

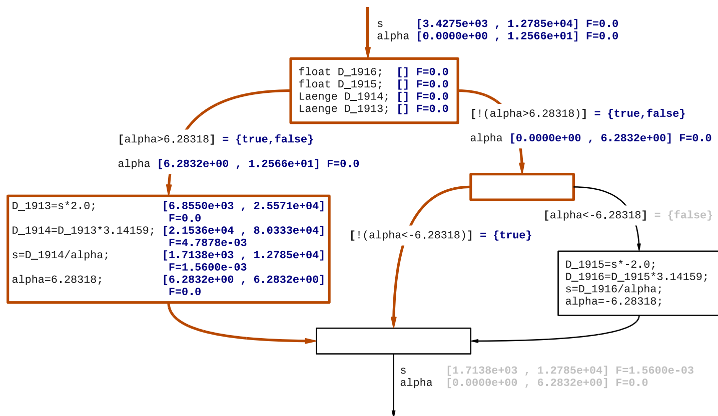
- Interpretation des rechten Zweiges



Beispiel für abstrakte Interpretation (Forts.)

SAMS

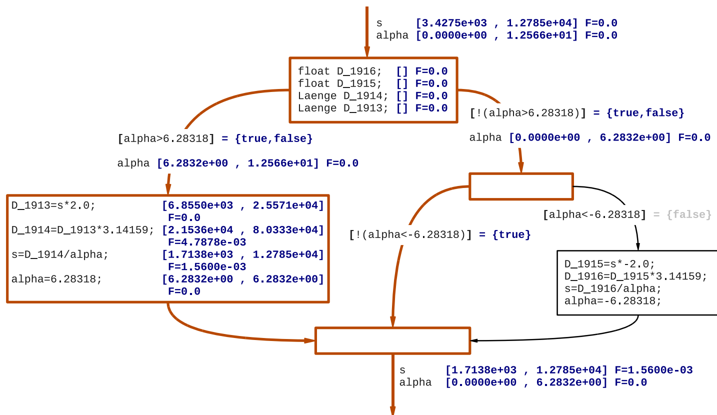
- Bedingung wertet zu `TRUE` aus; nur der linke Zweig muss interpretiert werden



Beispiel für abstrakte Interpretation (Forts.)

SAMS

- Evaluationen beider Zweige werden im ersten Block hinter der Verzweigung vereinigt



Eigenschaften abstrakter Interpretation

SAMS

- tatsächliches Programmverhalten wird überapproximiert
- i.A. sind auch solche Ausführungen enthalten, die mit konkreten Werten nicht möglich sind
- für das Ergebnis abstrakter Interpretation gilt folgende Beziehung:

Wenn eine Eigenschaft in der abstrakten Ausführung gilt,
dann gilt sie auch in der konkreten Ausführung

- wenn sie hingegen in der abstrakten Ausführung nicht gilt
  - → kann sie in der konkreten Ausführung dennoch gelten
  - → oder auch nicht ...
- abstrakte Interpretation kann nicht immer „verwertbare“ Resultate liefern

Vielen Dank für Ihre Aufmerksamkeit!