Programming Embedded Systems

Lecture 6 (cont)

Real-valued data in embedded software

Wednesday Feb 8, 2012

Philipp Rümmer
Uppsala University
`Philipp.Ruemmer@it.uu.se`

Interval methods

Motivation

- Is it possible to draw reliable conclusions from computations done with limited precision?

Motivation (2)

- ***Yes*** ... if we can control the rounding!
- E.g.

Safe if '+' rounds down

```
if (a + b >= 0) {
    // do something dangerous that
    // that only works if a+b >= 0
}
```

Interval methods

- Common in scientific computing, numerical mathematics; rich body of research
- Often an extremely efficient method to obtain estimates on the precise result of computations
- Are sound in the sense that no wrong answers are given, but can fail since error bounds might be too large to be useful

Idealised interval arithmetic

- Domain of real intervals:

$$\mathbb{IR} = \{\boldsymbol{x} = [\underline{x}, \overline{x}] \mid \underline{x} \in \mathbb{R} \cup \{-\infty\}, \overline{x} \in \mathbb{R} \cup \{+\infty\}, \underline{x} \leq \overline{x}\}$$

- Mathematical operations are lifted to intervals, e.g.:

$$\begin{aligned} \boldsymbol{x} + \boldsymbol{y} &= [\underline{x} + \underline{y},\ \overline{x} + \overline{y}] \\ \boldsymbol{x} - \boldsymbol{y} &= [\underline{x} - \overline{y},\ \overline{x} - \underline{y}] \end{aligned}$$

- Generally defined such that:

$$\boldsymbol{x} \text{ op } \boldsymbol{y} \supseteq \{x \text{ op } y \mid x \in \boldsymbol{x}, y \in \boldsymbol{y}\}$$

Machine interval arithmetic

- Intervals represented using two machine numbers (floating-point or fixed-point):

$$\mathbb{I}D = \{\boldsymbol{x} = [\underline{x}, \overline{x}] \mid \underline{x} \in D \cup \{-\infty\}, \overline{x} \in D \cup \{+\infty\}, \underline{x} \leq \overline{x}\}$$

- Operations are used *rounding outward:*

$$\boldsymbol{x} \mathbin{\tilde{+}} \boldsymbol{y} \quad = \quad [round_{\downarrow}(\underline{x} + \underline{y}),\ round_{\uparrow}(\overline{x} + \overline{y})]$$

- This ensures that the precise result is always within the computed machine interval

Example

- Fixed-point arithmetic with rounding:

$$\begin{aligned} 0.5 \,\tilde{\cdot}\, 1.5 &= 0.5 \\ (1 \cdot 2^{-1}) \,\tilde{\cdot}\, (3 \cdot 2^{-1}) &= (1 \cdot 3 \gg 1) \cdot 2^{-1} \end{aligned}$$

- In interval arithmetic:

$$\begin{aligned} [0.5, 0.5] \,\tilde{\cdot}\, [1.5, 1.5] &= [round_{\downarrow}(0.75),\ round_{\uparrow}(0.75)] \\ &= [0.5, 1.0] \end{aligned}$$

Inputs are singleton intervals

Output bounds the precise result

Further reading

- Library on interval methods:
  http://www.cs.utep.edu/interval-comp/