Entscheidungsverfahren für die Software-Verifikation

Dr. Carsten Sinz

Verifikation:
Wie kann *garantiert* werden, dass ein Computerprogramm fehlerfrei arbeitet?

Fehlerfrei?
- Keine Programmabstürze
- Keine "falschen" Rechenergebnisse
- Keine Sicherheitslücken

Welche Programme?
Für Automobile, Flugzeuge, Großrechner, mobile Geräte, Internet-Infrastruktur, …

"Am 31. Dezember 2008 fielen weltweit die Zune-Geräte der ersten Generation aus. Ursache war ein interner Fehler bei der Handhabung von Schaltjahren"

Heise Online, 03.01.2009

Buggy smartphone software is "new real
RIM CEO Jim Balsillie über die fehlerhafte Software des Blackberry Storm (Jan. 2009)

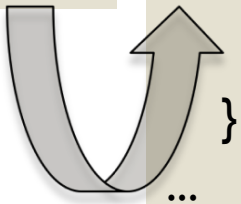
Z2K9: Source Code

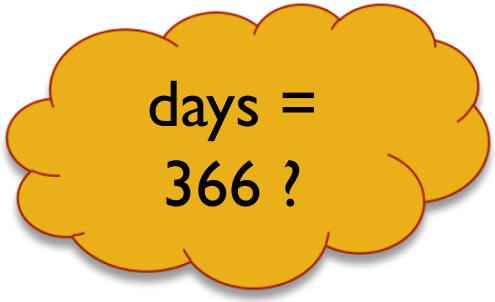
```
BOOL ConvertDays(UINT32 days,
    SYSTEMTIME* lpTime)
{
    int dayofweek, month, year;
    UINT8 *month_tab;

    //Calculate current day of the week
    dayofweek = GetDayOfWeek(days);

    year = ORIGINYEAR;
```

```
while (days > 365)
  {
      if (IsLeapYear(year))  {
          if (days > 366) {
              days -= 366;
              year += 1;
          }
      }
      else {
          days -= 365;
          year += 1;
      }
  }
...
```





Bedeutung Software-Verifikation

- *„(...) Software verification (...) has been the **Holy Grail** of **computer science**. (...) Now in some key areas (...) we are building tools that can do **actual proofs about** the **software** (...) in order to **guarantee** the **reliability**.“*

Bill Gates, WinHEC 2002

Entscheidungsverfahren

- Beispiele
  1. SAT: Äquivalenz von C-Ausdrücken
  2. Bit-Vektor-Logik: c32sat

GCC: Code Beispiel

```
if (mode1 == VOIDmode
   || GET_CODE (op0) == REG || GET_CODE (op0) == SUBREG
   || (modifier != EXPAND_CONST_ADDRESS
       && modifier != EXPAND_INITIALIZER
       && ((mode1 != BLKmode && ! direct_load[(int) mode1]
            && GET_MODE_CLASS (mode) != MODE_COMPLEX_INT
            && GET_MODE_CLASS (mode) != MODE_COMPLEX_FLOAT)
           /* If the field isn't aligned enough to fetch as a memref,
              fetch it as a bit field.  */
           || (mode1 != BLKmode
               && SLOW_UNALIGNED_ACCESS (mode1, alignment)
               && ((TYPE_ALIGN (TREE_TYPE (tem)) < GET_MODE_ALIGNMENT (mode))
                   || (bitpos % GET_MODE_ALIGNMENT (mode) != 0)))
           /* If the type and the field are a constant size and the
              size of the type isn't the same size as the bitfield,
              we must use bitfield operations.  */
           || ((bitsize >= 0
                && (TREE_CODE (TYPE_SIZE (TREE_TYPE (exp))) == INTEGER_CST)
                && 0 != compare_tree_int (TYPE_SIZE (TREE_TYPE (exp)), bitsize)))))
   || (modifier != EXPAND_CONST_ADDRESS
       && modifier != EXPAND_INITIALIZER
       && mode == BLKmode
       && SLOW_UNALIGNED_ACCESS (mode, alignment)
       && (TYPE_ALIGN (type) > alignment
           || bitpos % TYPE_ALIGN (type) != 0))) { ...}
```

Organisatorisches

- Webseite der Vorlesung:
  - http://verialg.iti.kit.edu/441.php
- Übungen:
  - Mi 11:30-13:00, 14-tägig, SR -109, Beginn wird noch bekanntgegeben
  - Teilnahme an Übungen nicht verpflichtend, aber empfohlen
  - http://verialg.iti.kit.edu/446.php
- Prüfung:
  - mündlich, 2 SWS, Vertiefungsgebiet Theoretische Grundlagen
- Literatur:
  - D. Kröning / O. Strichman: Decision Procedures – An Algorithmic Point of View (Springer, 2008)
    [Innerhalb des Uni-Netzes ist eine Online-Version im Volltext verfügbar.]