1. Übungsblatt zur Vorlesung “Formale Methoden im Software Entwurf”

TECHNISCHE UNIVERSITÄT DARMSTADT

Einführung in PROMELA
Diskussion der Aufgaben und Lösungen in den Tutorien: Kalenderwoche 44

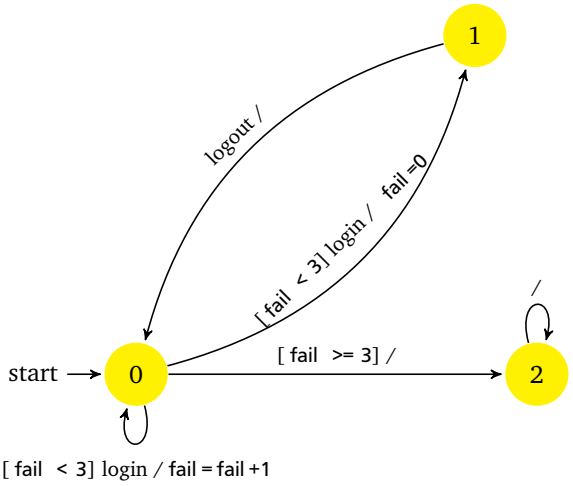
Aufgabe 1 Installation des Modelcheckers Spin

Installieren Sie zunächst den Modelchecker Spin sowie dessen graphische Benutzeroberfläche iSpin. Verweise zu den Installationsanleitungen und kleinere Hinweise finden Sie unter

https://moodle.informatik.tu-darmstadt.de/mod/page/view.php?id=16001

Aufgabe 2 Nichtdeterministische Bedingte Anweisung (non-deterministic conditional statement)

Die Authentifizierungskomponente einer sicherheitskritischen Anwendung wurde in der Modellierungsphase zunächst mit Hilfe des unten stehenden abstrakten endlichen Automaten spezifiziert:



Legende:

- Kreis mit Markierung x repräsentiert den Zustand x (Intuition: 0:Init, 1:Authenticated, 2:Locked)
- Beschriftung der Transitionen (Kanten): ’[’Bedingung’]’? (Ereignis)? ’/’ (Aktion)? (alle drei Teile sind optional)
  - Ereignis: Externes Ereignis der Art login, logout
  - Bedingung: Boolsche Bedingung (hier z.B. über den Wert der Zählvariablen fail)
  - Aktion: Anweisung, z.B. Zuweisung eines Wertes an eine Variable (hier: an die Variable fail)
- Eine Transition ist aktiviert/enabled (kann genommen werden) gdw. das spezifizierte Ereignis vorliegt *und* die Bedingung erfüllt ist. Ist kein Ereignis spezifiziert, dann ist die Transaktion aktiviert gdw. die Auswertung der Bedingung **true** ergibt. Eine fehlende Bedingung ist gleichbedeutend mit der Bedingung **true**.
- Im Falle eines nicht erwarteten Ereignisses (also z.B. login im Zustand 1) blockiert der endliche Automat.

Fortsetzung auf der nächsten Seite

Aufgabe 2.1 Implementierung in PROMELA

Realisieren Sie diesen Automaten in PROMELA. Vervollständigen Sie dazu das Rahmenprogramm in Listing 1 wie folgt:

a) Implementieren Sie das Inline-Makro selectEvent gemäß dem im Quellcode angegebenen Kommentar.

b) Vervollständigen Sie die Implementierung des Prozesses Authentication.

- Ersetzen Sie die Ausdrücke <T1> und <T2> mit geeigneten Typen. Begründen Sie Ihre Wahl.
- Implementieren Sie den Automaten, so dass die Variable currentState den aktuellen Zustand kodiert und fail die Fehlschläge beim Login zählt. Die Variable ev soll jeweils nach Betreten des neuen Zustands ein Ereignis mit Hilfe des Inline-Makros selectEvent zugewiesen, welches dann für den nächsten Zustandsübergang herangezogen wird.

Listing 1: "Rahmenprogramm Authentication (Datei: Authentication.pml)"

```
mtype = {login, logout}

// selects arbitraily one of the events 'login' or 'logout' and
// assigns the selected event to parameter 'event'
inline selectEvent(event) {
   // TODO: IMPLEMENT
}

active proctype Authentication() {
// counts failed authentication tries
// replace <T1> by a suitable type
<T1> fail = 0;

// current state
// replace <T2> by a suitable type
<T2> currentState = 0;

// received event
mtype ev;

// TODO: IMPLEMENT

}
```

Aufgabe 2.2 Simulation

a) Laden Sie Ihr Programm mit iSpin (oder verwenden Sie die Kommandozeile). Führen Sie dann die folgenden Simulationsläufe *interaktiv* aus:

1. Erfolgreicher Log-in nach einmaligem Versuch und anschließendes Ausloggen.
2. Gesperrter Account nach drei Fehlversuchen
3. Blockierter Automat bei Empfang des Ereignisses logout im initialem Zustand (Zustand: 0)

b) Führen Sie nun zwei verschiedene, zufällige Simulationsläufe automatisch aus (Achtung: bei iSpin müssen Sie unterschiedliche *random seeds* eingeben, wenn Sie unterschiedliche Testläufe bekommen wollen). Erklären Sie die in iSpin dargestellten Ablauftraces.

Aufgabe 2.3 Alternative Lösungen

Implementieren Sie den Automaten aus der ersten Teilaufgabe ohne Verwendung einer Variablen wie currentState (Hinweis: Labels). Können Sie auch das Inline-Makro vermeiden (ohne es von Hand aufzulösen)?

Aufgabe 3 Knapp vor der Deadline

P. kommt bei Ihnen 18 Minuten vor Abgabe des Übungsblattes mit den Nerven am Ende vorbei und bittet Sie um Hilfe, da das PROMELA Programm (Listing 2) sich nicht wie erwartet verhält. Können Sie es noch rechtzeitig für P. zum Laufen bringen? Bedenken Sie dabei, es soll Ps Lösung bleiben, d.h. nicht komplett umschreiben, sondern nur fehlerbereinigen.

Kontext: Das PROMELA Modell soll ein einfaches Würfelspiel modellieren: Das Spiel ist gewonnen, wenn, innerhalb von zwei Versuchen, die Augenzahl zweier Würfel (bzw. die Augensumme eines zweimal geworfenen Würfels) die Zahl 12 ergibt.

Listing 2: "Ps Lösung (Datei: PsProgram.pml)"

```
inline rollDiceTwice ( result ) {
    result = 0; // stores sum of two dice throws
    times = 0;  // loop counter
    do
      :: times <= 1 ->
           if
             :: result = result + 1;
             :: result = result + 2;
             :: result = result + 3;
             :: result = result + 4;
             :: result = result + 5;
             :: else -> result = result + 6;
           fi ;
           times = times + 1
      :: times > 1 -> goto loopEnd
    od;
    loopEnd:
      printf ("Result␣of␣two␣throws:␣%d\n", result)

}

proctype P() {
    byte dice = 0;
    byte times = 0;
    rollDiceTwice (dice );
    do
      :: dice == 12 -> printf("Won\n"); goto gameEnd
      :: dice != 12 && times < 1 -> rollDiceTwice(dice); times = times + 1
      :: true -> printf("Lost\n"); goto gameEnd
   od;
   gameEnd:
}
```

Aufgabe 4 Arrays

Das PROMELA Modell in Listing 3 soll die Elemente des Integer-Arrays a aufmultiplizieren.

Listing 3: "Product (Datei: Array.pml)"

```
#define N 5

active proctype ARRAY() {
  int a[N];
  int prod;

  /* TODO: IMPLEMENT */

  printf("The␣product␣is:␣%d\n", prod)
}
```

Erweitern Sie das obige Modell wie folgt:

a) alle Arrayelemente werden mit willkürlich gewählten Werten von 1 bis 5 initialisiert, danach wird

b) das Produkt aller Arrayelemente berechnet, in der Variablen prod abgelegt und auf dem Bildschirm ausgegeben.

Führen Sie mehrere zufällige Testdurchläufe aus (bei iSpin nicht vergessen den *random seed* zu ändern, sonst bleibt der Testlauf immer derselbe).

Aufgabe 5 Interleaving

Betrachten Sie das folgende PROMELA Modell (Datei: `Interleave.pml`):

```
int x = 3;
int y = 5;
int z = 0;

active proctype P() {
  if
    :: x > y -> z = x
    :: else -> z = y
  fi
}

active proctype Q() {
  x++
}

active proctype R() {
  y = y - x
}
```

a) Wieviele verschiedene Interleavings gibt es in dem Modell?

b) Geben Sie die Endergebnisse von x, y und z für jede der Interleavingmöglichkeiten an.