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

TECHNISCHE UNIVERSITÄT DARMSTADT

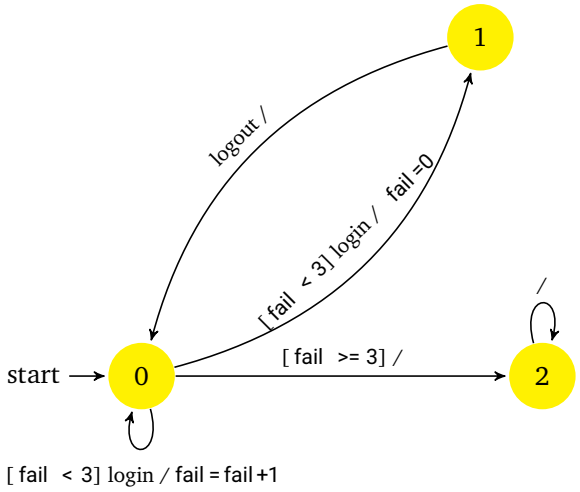
Einführung in PROMELA
Diskussion der Aufgaben und Lösungen in den Tutorien: 30.4.2024 - 3.5.2024

1 Installation des Modelcheckers Spin

Installieren Sie zunächst den Modelchecker Spin sowie dessen graphische Benutzeroberfläche iSpin. Verweise auf Installationsanleitungen und kleinere Hinweise finden Sie im Moodle unter „Installationshinweise für den Spin Modelchecker“.

2 Nichtdeterministisch bedingte Anweisung (non-deterministic conditional statement)

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



Definition der Funktionsweise:

- Knoten sind Zustände, Startzustand markiert mit ‘start’ (hier sind die Zustände $0, 1, 2$ mit Startzustand 0)
  (Intuition: 0:Unauthenticated, 1:Authenticated, 2:Locked)
- Beschriftung der Transitionen (Kanten):
  ‘[’ *Bedingung* ‘]’ ? *Ereignis*? ‘/’ *Aktion*?
  (alle drei Teile sind optional)
  - *Ereignis*: Externes Ereignis der Art ‘login’, ‘logout’
  - *Bedingung*: Boolesche Aussage (hier 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*, wenn das spezifizierte Ereignis vorliegt und die Bedingung erfüllt ist. Ist kein Ereignis spezifiziert, dann ist die Transaktion aktiviert, wenn die Bedingung erfüllt ist. Eine fehlende Bedingung ist implizit erfüllt, also wie [**true**].
- Falls eine Transition aktiviert ist, dann wird die Transition ausgeführt. Falls mehrere Transitionen aktiviert sind, dann wird eine davon zufällig *ausgewählt* und ausgeführt.
- Die Ausführung des Automaten stoppt, falls keine Transition aktiviert ist (also z.B. bei ‘login’ im Zustand 1).

[[*Zusatzaufgabe:* Geben Sie eine PROMELA-Implementierung für *jeden* Automaten dieses Typs *ohne* Ereignisse.]]

Fortsetzung auf der nächsten Seite

2.1 Implementierung in PROMELA

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

1. Implementieren Sie das Inline-Makro selectEvent gemäß dem im Quellcode angegebenen Kommentar.
2. 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() {
    // replace <T1> by a suitable type
    <T1> fail = 0;

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

    // received event
    mtype ev;

    // TODO: IMPLEMENT
}
```

2.2 Simulation

1. Laden Sie Ihr Programm mit iSpin (oder verwenden Sie die Kommandozeile). Führen Sie dann die folgenden Simulationsläufe *interaktiv* aus:
   a) Erfolgreicher Log-in nach einmaligem Versuch und anschließendes Ausloggen.
   b) Gesperrter Account nach drei Fehlversuchen.
   c) Blockierter (gestoppter) Automat bei Empfang des Ereignisses logout in Zustand 0.
2. 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.

2.3 Alternative Lösungen

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

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 P.s 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* die Zahl 12 ergibt.

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

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

proctype P() {
    byte sum = 0;
    byte counter = 0; // try counter
  again:
    throwTwoDice(sum);
    if
      :: sum == 12 ->
           printf ("Won\n");
      :: sum != 12 && counter < 1 ->
           goto again;
      :: true ->
           printf ("Lost\n");
    fi ;
}
```

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:

1. alle Arrayelemente werden mit willkürlich gewählten Werten von 1 bis 5 initialisiert, danach wird
2. 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).

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
}
```

1. Wieviele verschiedene Interleavings gibt es in dem Modell?
   Hinweis: Zeichne einen Ausführungsgraphen.
2. Geben Sie die Endergebnisse von x, y und z für jede der Interleavingmöglichkeiten an.