Übungen zur Vorlesung Safety Critical Systems I Wintersemester 2004/2005	AGBS Jan Peleska Tobias Hartmann

Serie 1

Fehlerbaumanalyse

Aufgabe 1:

Abbildung 1 beschreibt ein Bahngleisnetz mit Gleisabschnitten ($sa, sb, \ldots, s11, s12, \ldots$) und Weichen ($w11, w12, \ldots$), auf dem noch keine Signale eingetragen sind. Führen Sie für die Gleisabschnitte sa, sb, sc, $s11$, $s22$, $s32$, $s23$ jeweils eine Fehlerbaumanalyse durch, welche den Hazard *'Zug befindet sich auf Abschnitt s und wird von anderem Zug angefahren'* untersucht. Dabei nutzen Sie die im Anhang beschriebene Java-Notation und die in der Vorlesung eingeführte ASCII-Notation für Fehlerbäume. Grundlage für die Analyse sind folgende als gültig angenommene Regeln; hier zunächst informal spezifiziert:

1. Gleisabschnitte können LONG oder SHORT sein. LONG sind $s11, s12, \ldots$, SHORT sind $sa, sb, \ldots$.
2. Ein Zug bewegt sich immer in Richtung der Weichenstellungen.
3. Ein Zug befindet sich auf mindestens einem Gleisabschnitt, maximal auf
   - zwei LONG Gleisabschnitten mit allen dazwischen liegenden SHORT-Abschnitten oder
   - auf einem LONG Gleisabschnitt mit allen vorausgehenden SHORT Abschnitten oder
   - auf einem LONG Gleisabschnitt mit allen nachfolgenden SHORT Abschnitten .
4. Ein Zug im Zustand DRIVING erreicht den seiner Richtung entsprechenden nächsten Gleisabschnitt (rekursiv anzuwendende Regel!).
5. Ein Zug im Zustand BRAKING erreicht noch den nächsten LONG Gleisabschnitt in seiner Richtung, überfährt dabei alle vorausgehenden SHORT Gleisabschnitte. Danach hat er den Zustand STOPPED.
6. Ein Zug im Zustand STOPPED bleibt auf seinem Gleisabschnitt, bis er wieder in den Zustand DRIVING übergeht.

Die formale Semantik der Zugbewegungen ist der Java-Klasse `Train` zu entnehmen, die ebenfalls im WWW verfügbar ist.
Beispiel: Fehlerbaumanalyse für $s11$:

```
HAZARD(s11)
  + THREAT(s12)
      * s12.getTrainOnSegment() != null
      * s12.getTrainOnSegment().getDirection() == LEFT
      * s12.getTrainOnSegment().getSpeed() != STOPPED
      * s12.getTrainOnSegment().getSpeed() != EMERGENCY_STOPPED
  + THREAT(G2)
      + THREAT(sb)
          * sb.getTrainOnSegment() != null
```

```
    * sb.getTrainOnSegment().getDirection() == LEFT
    * sb.getTrainOnSegment().getSpeed() != STOPPED
    * sb.getTrainOnSegment().getSpeed() != EMERGENCY_STOPPED
+ THREAT(sd)
     ...
+ THREAT(se)
     ...
+ THREAT(s23)
     * sd.getNeighbour(LEFT) == sb
     * s23.getTrainOnSegment() != null
     * s23.getTrainOnSegment().getDirection() == LEFT
     * s23.getTrainOnSegment().getSpeed() != STOPPED
     * s23.getTrainOnSegment().getSpeed() != EMERGENCY_STOPPED
+ THREAT(s14)
    ...
```

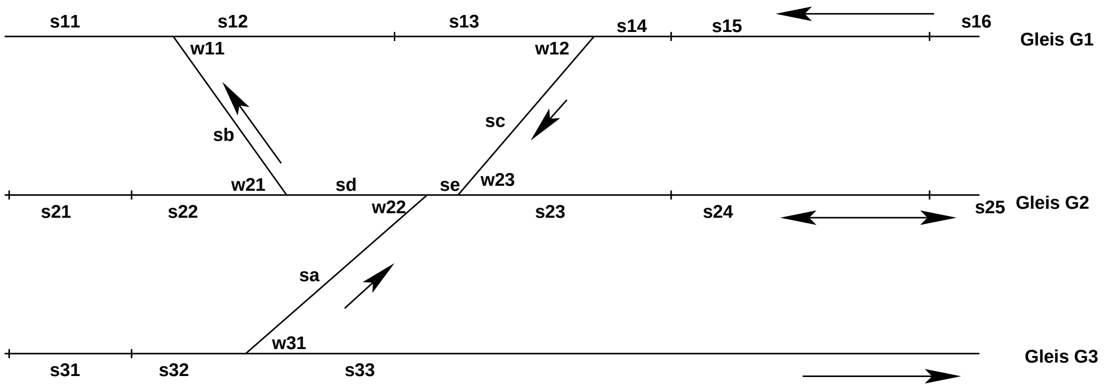

Abbildung 1: Bahngleisnetz, auf dem eine Fehlerbaumanalyse durchgeführt werden soll.

Aufgabe 2:

Führen Sie in der folgenden Skizze eines Bahngleisnetzes die minimal erforderlichen Signale ein, so dass die in Aufgabe 1 identifizierten Hazards vermieden werden können. Transformieren Sie die Spezifikationen der Hazards in Sicherheitsbedingungen über Signalzustände und Belegungen der Segmente.

Aufgabe 3:

Die im WWW mit dem Übungsblatt verfügbar gemachten Java-Quellen stellen eine Simulation des Gleisnetzes mit Zügen und Weichenbewegungen dar, ohne Sicherheitsbedingungen zu beachten – dementsprechend kommt es bei der Ausführung zu Exceptions, wenn zwei Züge auf einem Gleisabschnitt zusammentreffen. Erweitern Sie die Java-Programme um Signal-Objekte und Weichensteuerungen nach folgenden Regeln:

- Ist das in Fahrtrichtung zeigende Signal am Ende des Segmentes, auf dem der Zugkopf steht, in Stellung HALT, geht der Zug in Zustand BRAKING über.
- Ein Signal wird nur auf FAHRT gestellt, wenn hierdurch keine Kollisionen entstehen können.

- Eine Weiche wird nur gestellt, wenn auf allen angrenzenden Gleisabschnitten keine Züge stehen (Hier braucht man eine Synchronisationskonstruktion für die Zug- und Weichenthreads!)

Die resultierende Simulation soll es nun nicht mehr zu Kollisionen kommen lassen!
ABGABETERMIN: Die ausgedruckten Lösungen sollen bis zum Donnerstag (9.12.04) - spätestens in der Übung - abgegeben werden. Ebenfalls bis dahin sollten die Java-Quellen als tar.gz an `hartmann@tzi.de` gemailt worden sein.