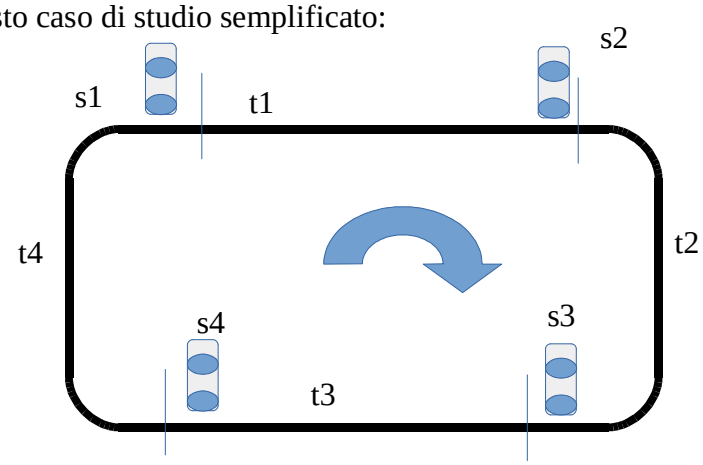
# INTERLOCKING CONTROL SYSTEM

Prendiamo un versione semplificata dell'interlocking tables per un ferrovia.

*In railway signalling, an interlocking is an arrangement of signal apparatus that prevents conflicting movements through an arrangement of tracks such as junctions or crossings. The signalling appliances and tracks are sometimes collectively referred to as an interlocking plant. An interlocking is designed so that it is impossible to display a signal to proceed unless the route to be used is proven safe.*

Lo proviamo per questo caso di studio semplificato:




E' un circuito composto da 4 tracks sections (t1, t2, t3, t4), ognuna preceduta da un signal (semaforo). Consideriamo solo la circolazione in senso orario e con due soli treni TR1 inizilamente in t1 e TR2 inizilamente in t3. I semafori possono essere verdi, gialli o rossi. I treni possono andare in una track solo se il semaforo che la precede non è rosso (cioè verde o giallo). Se il semaforo è rosso il treno sta nella sua track.

Il sistema deve controllare i semafori in modo da grantire la circolazione sicura dei treni. Inoltre deve avvisare che il prossimo semaforo è rosso mediante un semaforo giallo (ad esempio se s4 è rosso, s3 sarà giallo). I requisiti sono che:

- No collision: It never happens that two trains in the system collide. That is, two trains are never running on the same track at the same time.
- Liveness: I treni possono muoversi (cioè il sistema non è bloccato).

# Compito

In un file word annota tutto quello che hai fatto. Sviluppa un progetto in eclipse per ognuno delle domande. Puoi anche andare in parallelo (ad esempio modello NUSMV e codice Java).

## 1. Testing basato su programmi

Scrivi l'implementazione in Java del Interlocking Control System. Se fai delle assunzioni spiegale. Cerca di scrivere la classi in modo che possano essere riusate per un'altra topologia di ferrovia (cioè che sia facile cambiare la configurazione del circuito di esempio).

Scrivi i test Junit per la copertura istruzioni, condizioni, decisioni, MCDC. Eseguili e valuta la copertura.

Commenta nei test Junit cosa fanno i diversi test e che copertura ti permettono di ottenere.

Prova a generare casi di test on con Randoop  e valuta i casi di test generati.

# 2. DBC/JML

Copia le classi per l'Interlocking control System e aggiungi i contratti JML (almeno per una classe).

Prova con i contratti JML con una classe main in cui chiami i diversi metodi.

Prova anche a modificare il codice e controlla che i contratti siano violati. Documenta bene le violazioni e le loro cause in commenti o nel file di documento.

# 2b  verifica dei programmi (key hoare)

Prova a verificare qualche contratto.

# 3. NUMSV.

Scrivi il modello NuSMV per l'Interlocking control System. Consiglio: usa i moduli

Scrivi un po' di proprietà – seguendo il testo dei requisiti.

Aggiungi proprietà che ti sembrano interessanti, spiegale bene, e provale.

Trova anche qualche proprietà giustamente falsa e commenta lo scenario in cui viene falsificata.

# 4a MBT con le FSM

Prova a scrivere una FSM (con disegno) per il semaforo e anche la sua implementazione in Java.

Testala in modalità offline (genera i casi di test a mano). Implementala in modelJunit e caricala. Salva il disegno della macchina come immagine.

Genera i casi di test con modelJunit e salvali con TXT

# 4b Input domain modelling

- Applica IDM ad un metodo a tua scelta
- Prova a considerare il sistema delle 4 tracks (in prima istanza ignora i semafori) e applica il combinatorial testing per vedere quali sono le configurazioni possibili delle track (usa i constraints)