Control de un Paso a Nivel de Tren

Proyecto: Modelado y Simulación de Sistemas Dinámicos con el Formalismo DEVS

Año: 2024

Descripción de Proyecto

El sistema consiste de tres subsistemas paralelos: un tren (train), un controlador (controller) y una barrera (gate) (TCG), los cuales están modelados con autómatas temporizados en la figura 1.

La variable de control st del tren varía sobre tres locaciones: $st = Far$ si el tren está lejos de cruzar el paso a nivel (no hay trenes cerca del cruce); $st = Near$ si el tren está próximo a cruzar; y, $st = Inside$ si está cruzando. Far es la locación del estado inicial. Cuando el tren está próximo a cruzar (es decir, se mueve de la locación Far a $Near$), envía una señal *Approach* al controlador, avisando sobre la proximidad de un tren. Esto ocurre n unidades de tiempo antes que el tren este cruzando el paso a nivel, con $n > kt1$. Cuando el tren termina de cruzar (es decir, se mueve de la locación $Inside$ a Far) envía una señal *Exit* al controlador, indicando el alejamiento del tren del paso a nivel. Esto ocurre antes que $kt2$ unidades de tiempo han pasado desde la señal *Approach*.

La variable de control sc del controlador varía sobre cuatro locaciones: $sc = Sc1$ si el controlador está esperando que el tren arribe; $sc = Sc2$ si la señal *Approach* ha sido recibida; $sc = Sc3$ si el controlador está esperando la señal *Exit*; y, $sc = Sc4$ si la señal *Exit* ha sido recibida. $Sc1$ es la locación del estado inicial. Cuando la señal *Approach* es recibida (es decir, el controlador se mueve de $Sc1$ a $Sc2$), el controlador envía a la barrera la señal *Lower*, exactamente $kc1$ unidades de tiempo después, indicando que ésta debe bajar. Cuando *Exit* es recibida (es decir, el controlador se mueve de $Sc3$ a $Sc4$), antes de $kc2$ unidades de tiempo el controlador envía a la barrera la señal *Raise*, para que la barrera comience a subir.

La variable de control sg de la barrera varía sobre cuatro locaciones: $sg = Open$ si la barrera está levantada y esperando la señal *Lower*; $sg = Lowering$ si la señal *Lower* ha sido recibida; $sg = Closed$ si la barrera está baja y esperando la señal *Raise*; y, $sg = Raising$ si la señal *Raise* ha sido recibida. *Open* es la locación del estado inicial. Cuando la señal *Lower* es recibida (es decir, la barrera se mueve de *Open* a *Lowering*), la barrera está baja antes de $kg1$ unidades de tiempo y cuando *Raise* llega, antes de $kg3$ y por lo menos $kg2$ unidades de tiempo después la barrera está levantada nuevamente.

Los valores de las constantes son los siguientes: $kt1 = 2$, $kt2 = 5$, $kc1 = 1$, $kc2 = 1$, $kg1 = 1$, $kg2 = 1$ y $kg3 = 2$.

Objetivos

Estamos interesados en analizar dos clases típicas de propiedades: *safety* y *liveness*. Las primeras se usan para especificar que "nada malo ocurrirá". Las propiedades de liveness permiten especificar que "algo bueno ocurrirá en el futuro".

Un requerimiento de *safety* asociado al sistema TCG es la propiedad "siempre que el tren está cruzando el paso a nivel, la barrera se encuentra baja". Un requerimiento de *liveness* asociado al sistema TCG es la propiedad "siempre que el tren envíe una señal *Approach* en el futuro enviará una señal *Exit*".

Se pide:

- Especificar el comportamiento del sistema utilizando el formalismo DEVS. Basarse en el lenguaje CML-DEVS. Descargar Tesis de Hollfmann desde los materiales del SIAT.

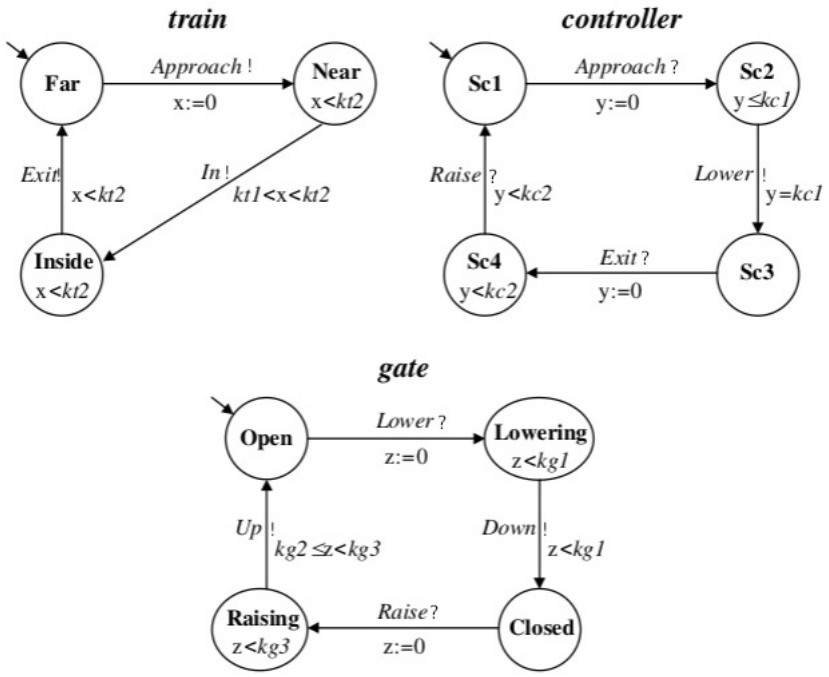

Figure 1: Autómata temporizado del sistema TCG.

- Implementar la especificación del TCG en PowerDevs. Diseñe los módulos lo más parametrizado posible, de tal manera que sea sencillo hacer cambios temporales.
- A través de simulaciones intentar detectar si alguna de las propiedades mencionadas no son válidas. Si esto sucede registrar los valores generados (esto se denomina *traza*) como prueba de cómo y cuando se dió esa situación. Proponer cambios en los parámetros del sistema de tal forma que con simulaciones "no pueden" violar ninguna de las propiedades.
- Graficar el comportamiento de los tres subsistemas (elegir el tipo de gráfico mas representativo) de tal forma que ayude a detectar visualmente la violación de las propiedades. Además sirve para comprender y ajustar ciertos valores temporales.
- Se desea conocer el tiempo promedio en que un tren entra al sistema (*Approach*) hasta que sale (*Exit*). Realizar un análisis de salidas mediante el método del Intervalo de Confianza (ver en "Analisis de Salidas").
- Presentar: (1) informe del proyecto que describa la especificación, análisis y proponga mejoras del sistema, (2) el proyecto PowerDevs.

Consideraciones:

- Utilice GNUPLOT para crear los gráficos.
- Para generar tiempos entre a y b utilizar una distribución uniforme(a,b).
- Por simplicidad asumir que no llega ningún tren mientras haya uno cruzando.