Verificación de Sistemas Reactivos
Casos de estudio

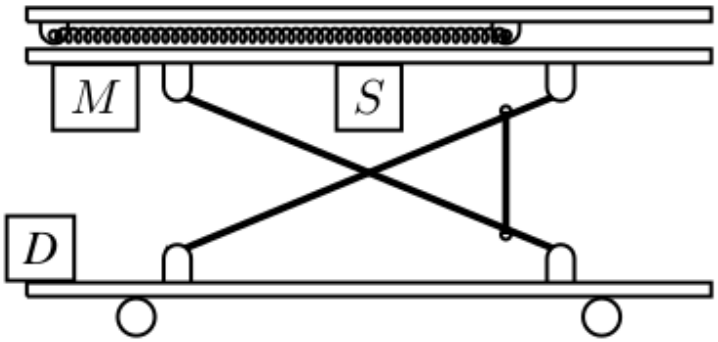
Alejandro Sánchez

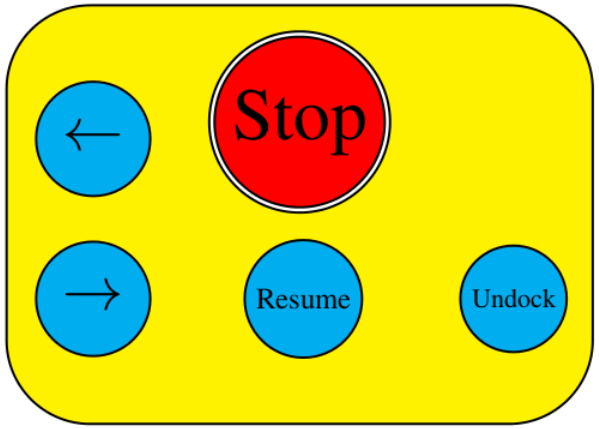
Departamento de Informática
Universidad Nacional de San Luis

Maestría en Ingeniería de Software
Especialización en Ingeniería de Software
30 de Octubre 2013

Motivación

Verificar el modelo de un caso de estudio (sistema embebido)





Plataforma móvil para scanner de resonancia magnética (versión simplificada)

Modelling and Analysis of Communicating Systems (draft)
Jan Friso Groote and Mohammad Reza Mousavi, Aug 2012

Estructura de la presentación

1. Fundamentos de model checking
2. Enunciado del caso de estudio
3. Desarrollo del caso de estudio
   - Modelar interacción controlador-dispositivo
   - Especificar modelo
   - Especificar y verificar propiedades
4. Conclusión

Estructura de la presentación

Enfoques para la verificación de sistemas

- Testing
- Inspección de código
- Métodos deductivos
- Model checking

Model checking

Definición

Model checking es una técnica automatizada que,
dado un modelo de estados finito de un sistema
y una propiedad formal,
sistemáticamente verifica si la propiedad es válida
en (un estado dado en) el modelo.

Model checking - Ventajas

- Enfoque de verificación general aplicable a un amplio rango de sistemas
- Soporta verificación parcial (alguna propiedad)
- No es vulnerable a la posibilidad de no exponer un error
- Provee información de diagnóstico
- Es una tecnología potencialmente de “apretar un botón”
- Captura un interés creciente en la industria
- Es facilmente integrado en el ciclo de desarrollo de la industria
- Tiene fundamentos matemáticos sólidos

Model checking - Desventajas

- Principalmente apropiado para aplicaciones de control (menos apropiado para aplicaciones con uso intensivo de datos)
- Aplicación sujeta a cuestiones de decidibilidad
- Verifica un modelo y no el sistema
- Chequea solo los requerimientos formulados
- Sufre del problema de la explosión de estados
- Requiere experiencia en encontrar abstracciones apropiadas
- Es software que puede contener defectos a su vez
- No permite generalizaciones (se verifican instancias)

Verificación vs. validación

En model checking desarrollamos:

- Modelo
- Propiedades

Se deben verificar y validar

- Verificación: chequear que el diseño satisface los requerimientos ("check that we are building the thing right")
- Validación: chequear que el modelo formal es consistente con el concepto informal del diseño
  ("check that we are building the right thing")

Modelos

Sistemas de transición de estados

- simples
- con representación de datos
- con representación de concurrencia
  - variables compartidas
  - pasaje de mensajes (síncrono/asíncrono-canales)

Modelos

Sistemas de transición de estados

- simples
- con representación de datos
- con representación de concurrencia
  - variables compartidas
  - pasaje de mensajes (síncrono/asíncrono-canales)

Explosión de estados

Propiedades - tipos

- Reachability. Una situación particular será alcanzada.
- Safety. Bajo ciertas condiciones, un evento no ocurrirá nunca.
- Liveness. Bajo ciertas condiciones, un evento eventualmente ocurrirá.
- Fairness. Bajo ciertas condiciones, un evento ocurrirá infinitamente a menudo.
- Deadlock freeness.

Propiedades - lógicas temporales

Se basan en el estudio de

- Estados, y/o
- Transiciones

Diferentes en su expresividad

- Linear Temporal Logic (LTL): X,G,F
- Computational Tree Logic (CTL): A,E,X,G,F restricto
- Computational Tree Logic* (CTL*): A,E,X,G,F
- Modal μ-calculus + Hennessy-Milner logic

Modal μ-calculus + Hennessy-Milner logic

Incorpora puntos fijos

- Mínimo (mu μ).
- Máximo (nu ν).

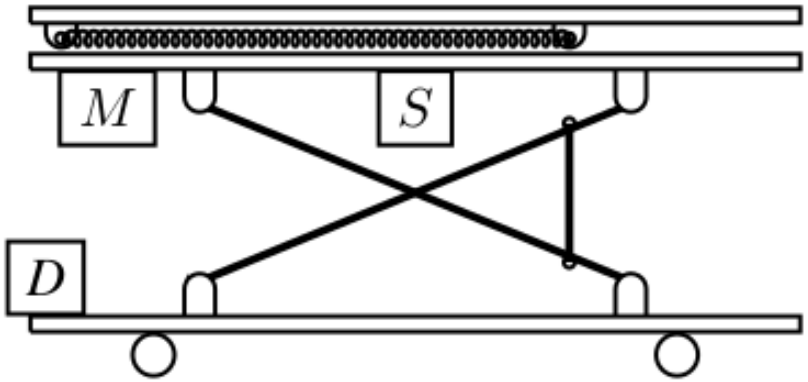
Intuición:
La formula es un grafo donde
las variables son estados y las modalidades son transiciones.
Una formula es verdadera si
se pasa un número finito de veces sobre los puntos fijos mínimos,
y está permitido pasar un número infinito por los puntos fijos máximos.

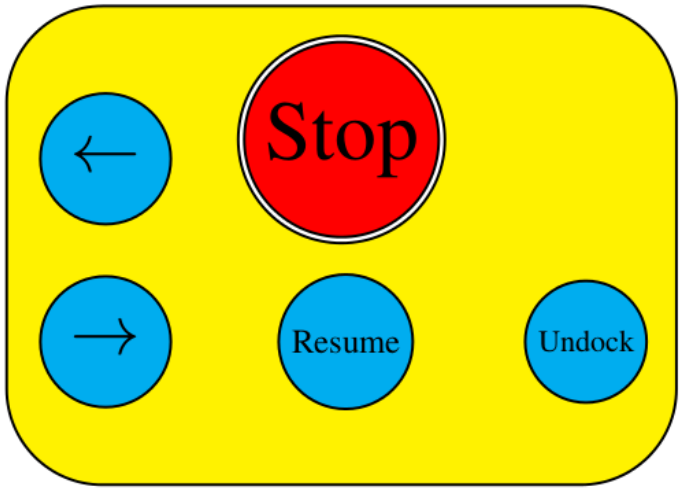
Ejemplos

- $\mu X([\overline{a}]X \wedge \langle true \rangle true)$ inevitabilidad de a
- $\nu X([a]X \wedge \varphi) = [a*]\varphi$

Estructura de la presentación

Descripción del dispositivo



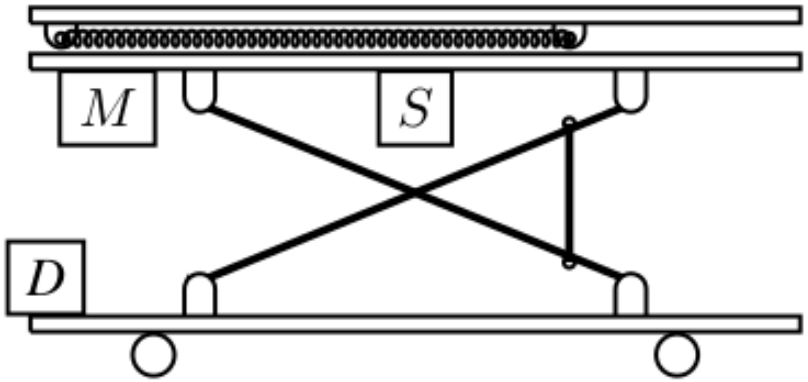


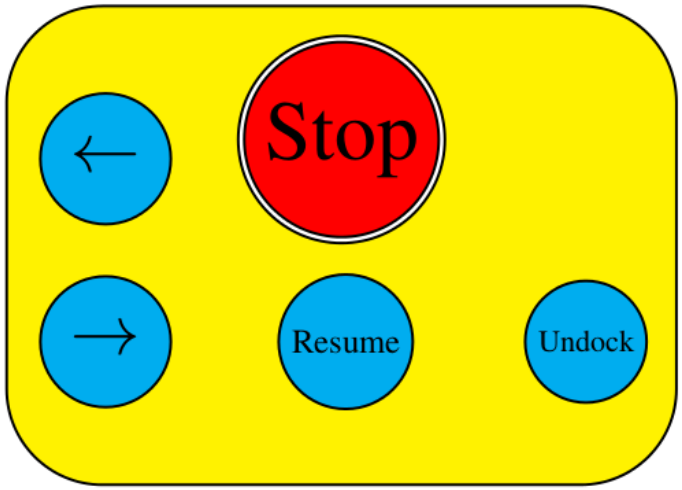
Plataforma móvil para scanner de resonancia magnética

Objetivo: desarrollar su controlador

Modelling and Analysis of Communicating Systems (draft)
Jan Friso Groote and Mohammad Reza Mousavi, Aug 2012

Descripción del dispositivo

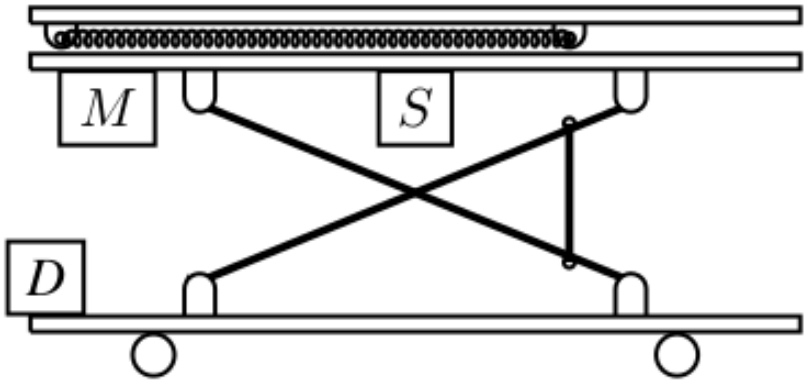


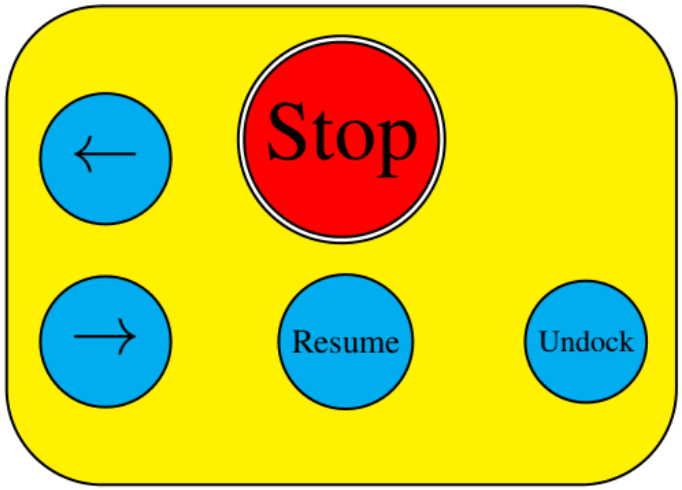


Trolley

- Ingresa y egresa al paciente del scanner
- Se fija al scanner (*docked*)

Descripción del dispositivo

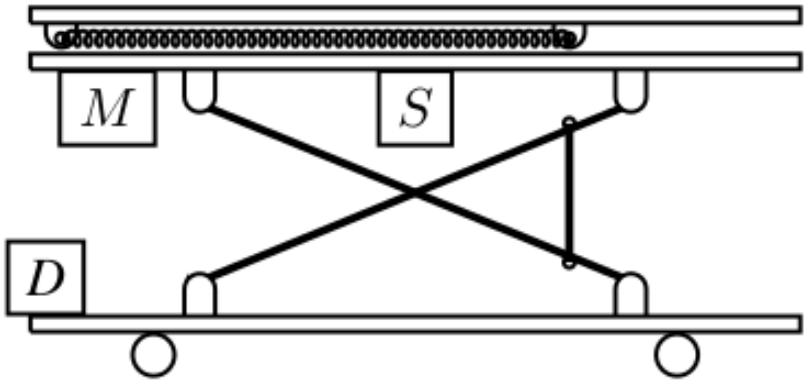


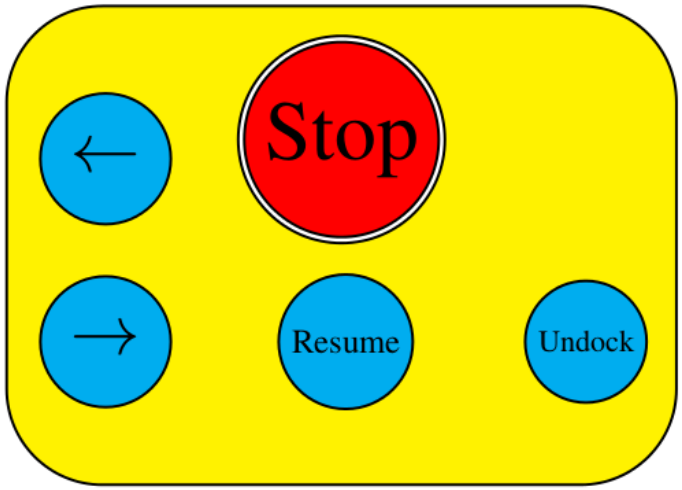


Trolley

- Platform
  - donde se ubica el paciente
  - se desliza a izquierda y derecha

Descripción del dispositivo

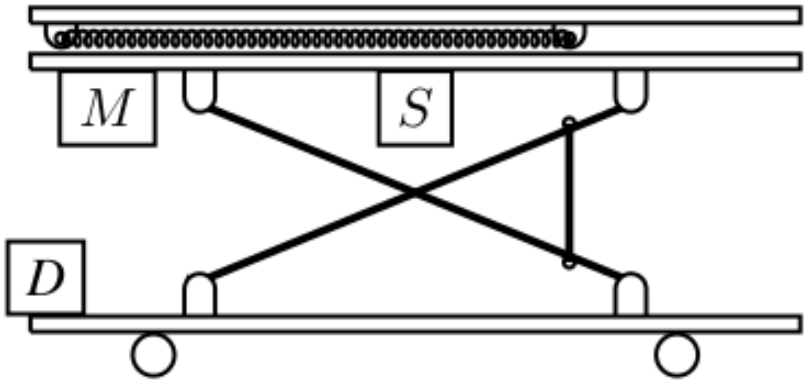


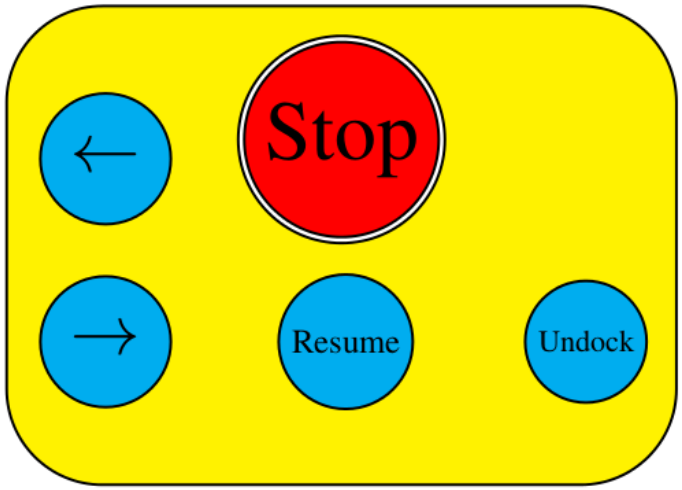


Trolley

- Motor M
  - mueve la plataforma a derecha o izquierda (dentro/fuera scanner)
  - freno aplicado cuando no esta fijado (*docked*) al scanner

Descripción del dispositivo

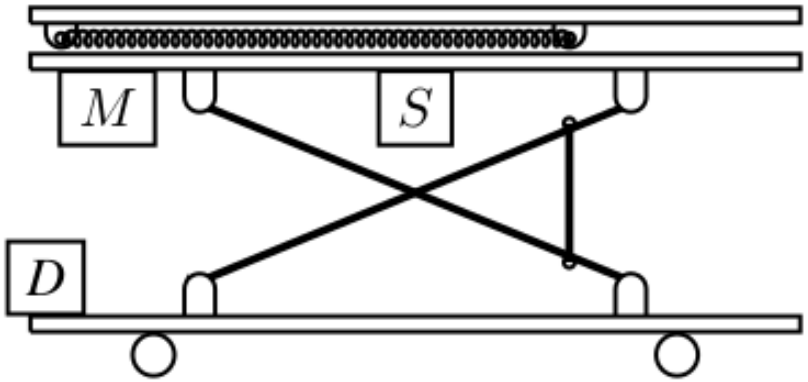


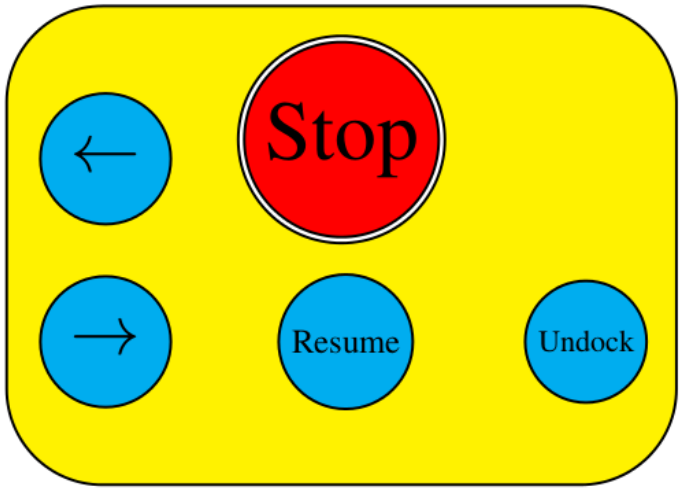


Trolley

- Sensor S
  - Detecta la posición de la plataforma con respecto al trolley

Descripción del dispositivo

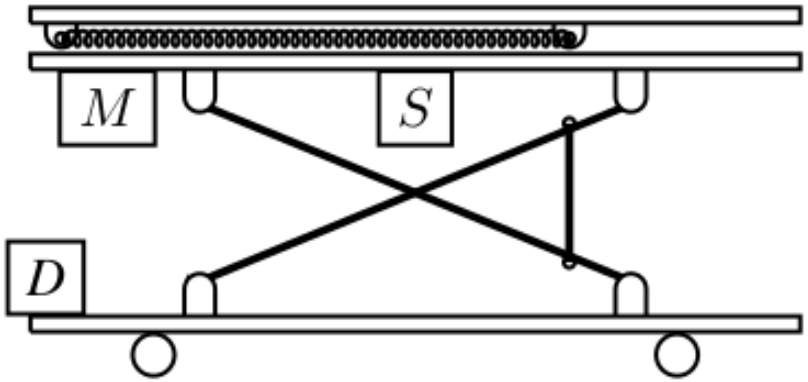


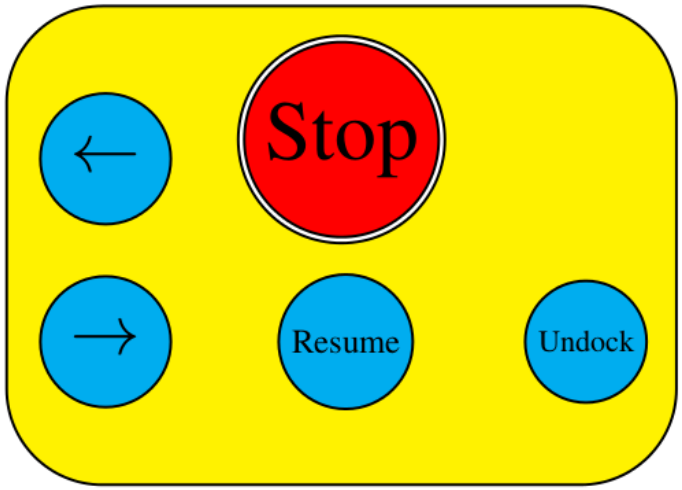


Trolley

- Docking station D
  - Tiene un cierre automático que fija el trolley al scanner
  - Detecta si el trolley esta fijo en el scanner
  - El cierre puede liberarse con una señal

Descripción del dispositivo

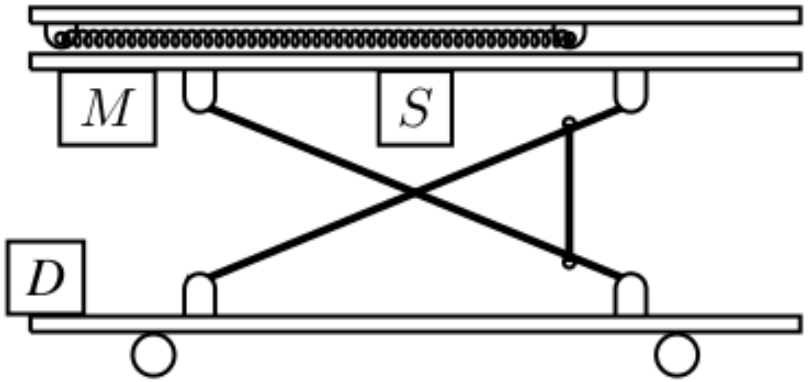


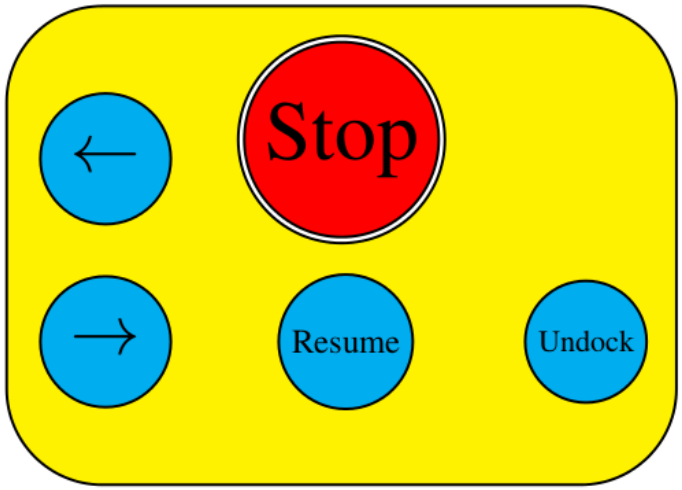


Panel

- Stop: inicia modo de emergencia
  - detiene todo movimiento de la plataforma
  - trolley fijo al scanner: liberar freno, sacar paciente manualmente
  - trolley libre: aplicar frenos
- Resume: vuelve a modo normal

Descripción del dispositivo

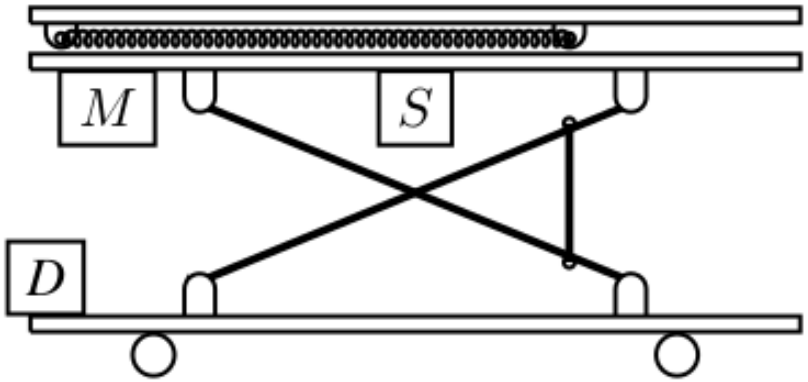


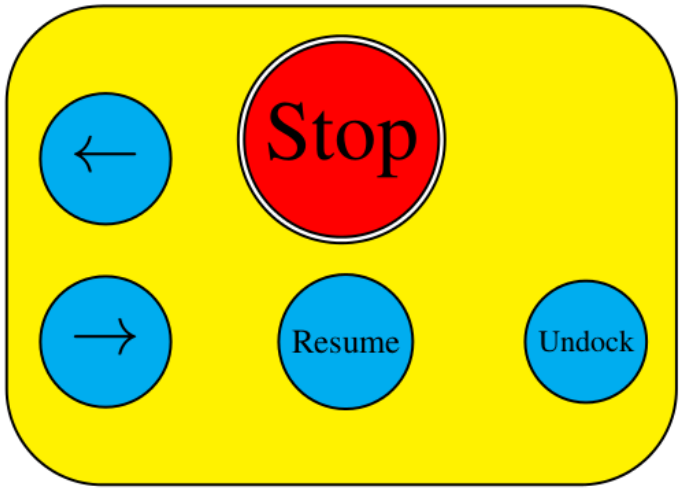


Panel

- Right, Left: si el trolley esta fijo, mueven la plataforma

Descripción del dispositivo

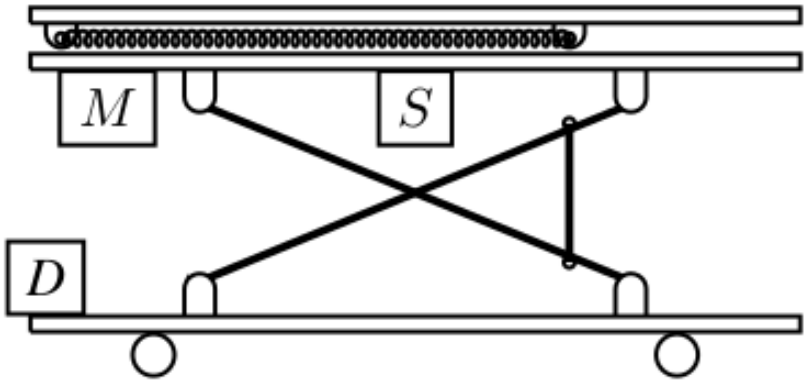


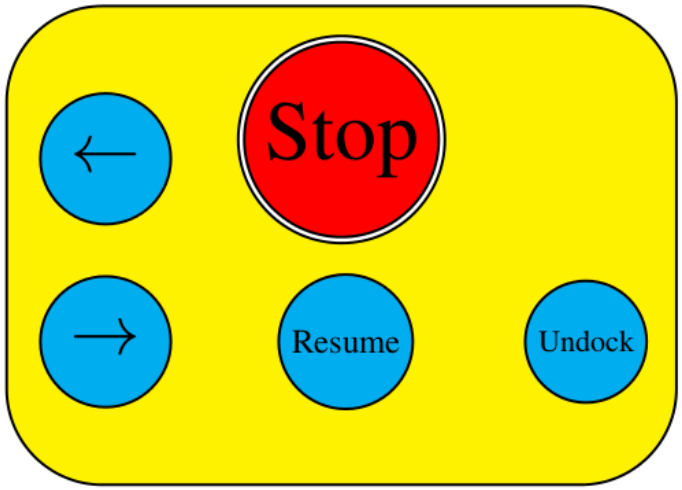


Panel

- Undock: libera el trolley (*docking unit*) del scanner
  - requiere que la plataforma completa esta sobre el trolley
  - se aplican los frenos de la cama al liberarse

Requerimientos del controlador



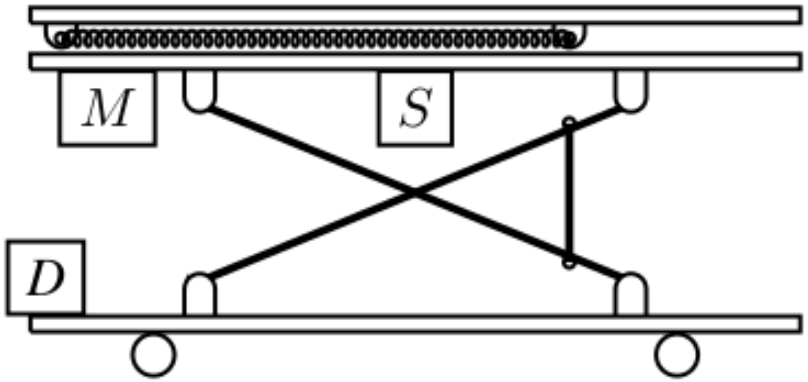


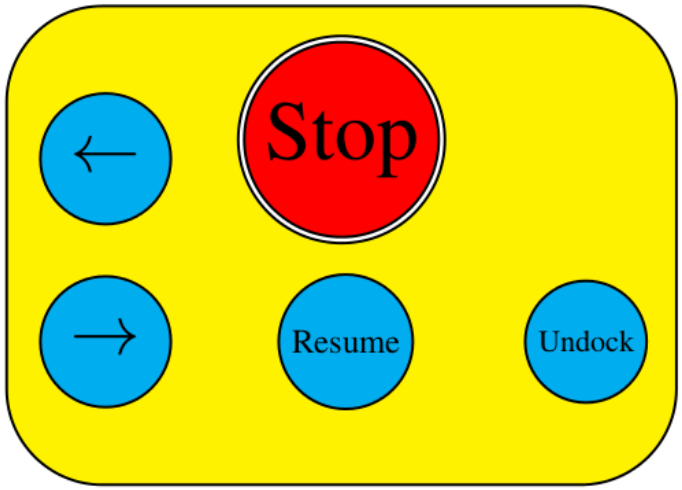
Safety properties

Prop 1

La plataforma no debe tumbarse:
cuando el trolley no esta fijo al scanner,
la plataforma completa debe estar sobre el trolley,
los frenos aplicados y el motor apagado.

Requerimientos del controlador



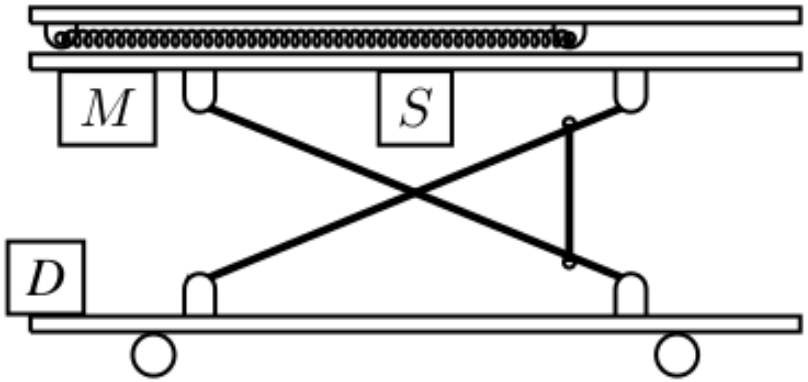


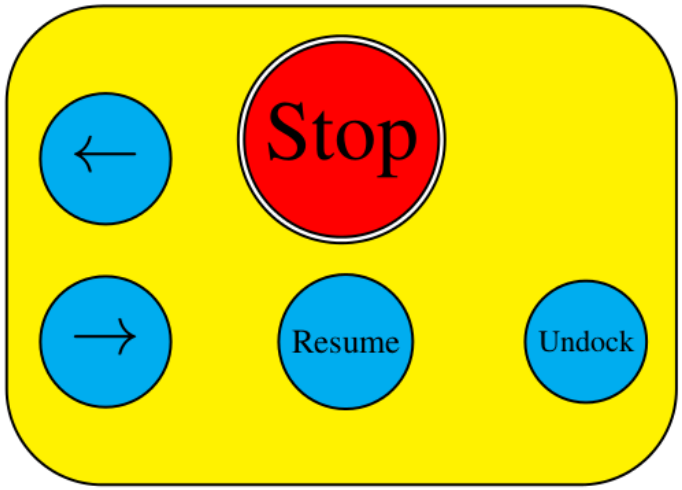
Safety properties

Prop 2

Al apretar “Stop”, un paciente puede ser sacado manualmente del scanner y de la habitación.

Requerimientos del controlador



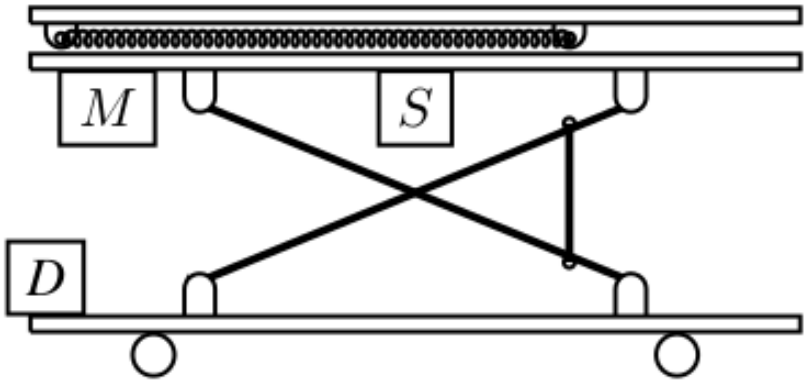


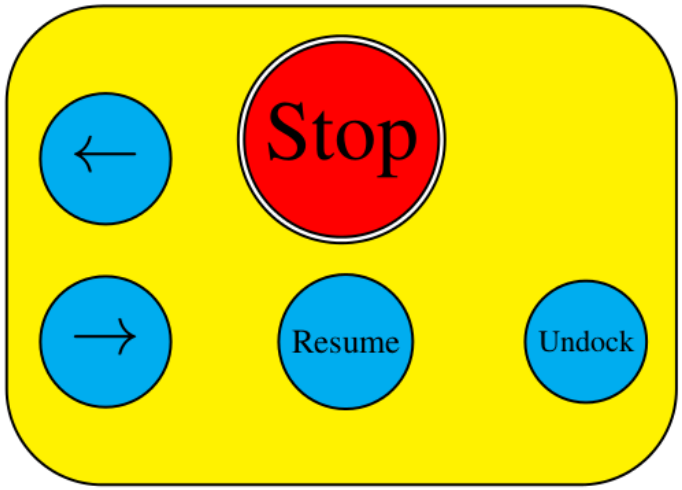
Safety properties

Prop 3

Para proteger el motor,
este no debe funcionar más allá
de la posición más interna o externa de la plataforma.

Requerimientos del controlador



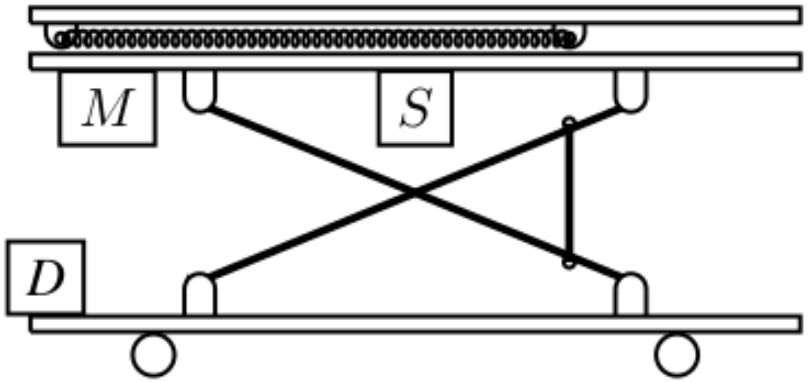


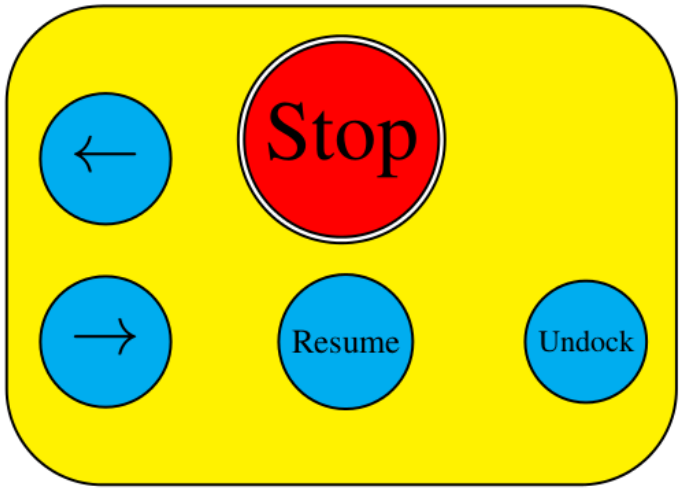
Liveness properties

Prop 4

Si se presiona “Left”,
el trolley esta fijo al scanner, no se está en modo emergencia,
la plataforma esta completamente en el scanner
y el motor no está en movimiento hacia la izquierda,
el motor se enciende hacia la derecha.

Requerimientos del controlador





Liveness properties

Prop 5

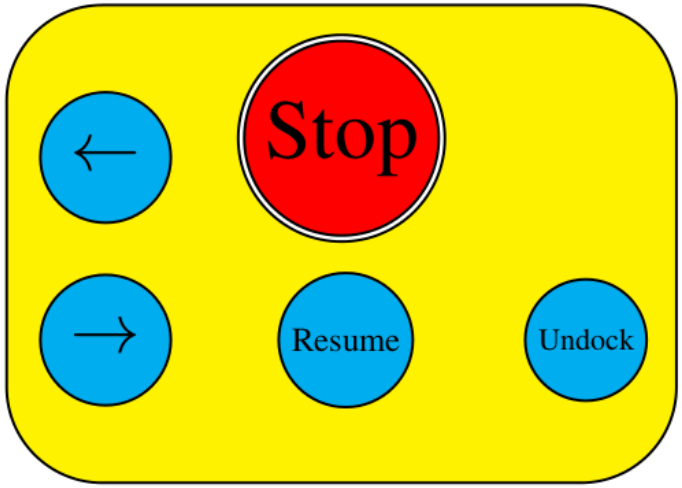
Si se presiona “Right”,
el trolley esta fijo al scanner, no se está en modo emergencia,
la plataforma esta completamente sobre el trolley
y el motor no está en movimiento hacia la derecha,
el motor se enciende hacia la derecha.

Estructura de la presentación

Pasos para el desarrollo

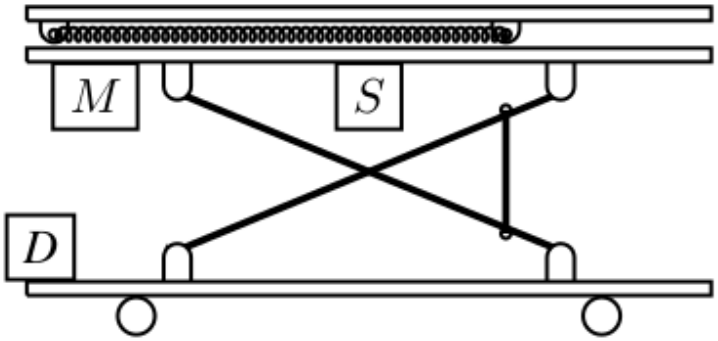
- Definir interacción del controlador con dispositivo
- Especificar el modelo
- Especificar los requerimientos como propiedades
- Verificar si el modelo cumple los requerimientos

Panel



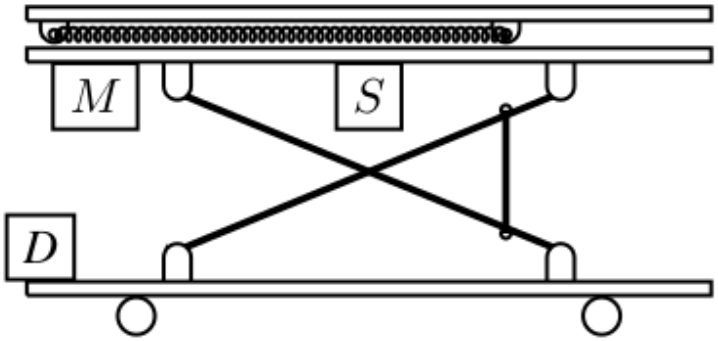
pressStop	botón “Stop” presionado
pressResume	botón “Resume” presionado
pressUndock	botón “Undock” presionado
pressLeft	botón “←” presionado
pressRight	botón “→” presionado

Motor M



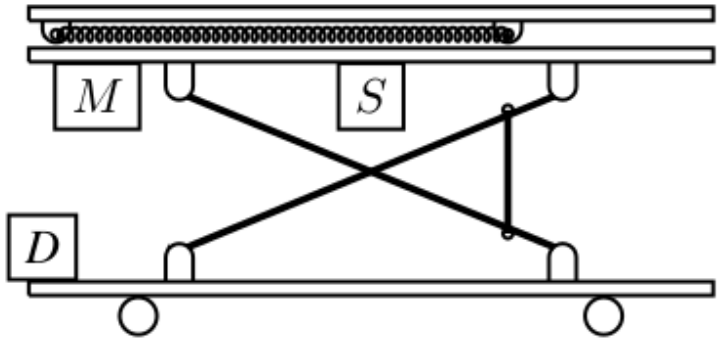
motorLeft	se enciende el motor hacia la izquierda
motorRight	se enciende el motor hacia la derecha
motorOff	el motor se apaga
applyBrake	aplicar los frenos del motor
releaseBrake	liberar los frenos del motor

Docking unit D



idDocked	indica que el trolley fue fijado al scanner
unlockDock	liberar el cierre del docking unit

Sensor S



atInnermost	indica que la plataforma está dentro del scanner
atOutermost	indica que la plataforma está completamente sobre el trolley

Acciones

```
act
    pressStop, pressResume,
    pressUndock, pressLeft, pressRight,
    motorLeft, motorRight, motorOff,
    applyBrake, releaseBrake,
    isDocked, unlockDock,
    atInnermost, atOutermost;
```

Proceso - Estado

```
sort
  Mode = struct Normal | Emergency ;
  MotorStatus = struct moveleft | moveright | stopped ;

proc Controller ( m:Mode,
  docked, rightmost, leftmost:Bool,
  ms:MotorStatus ) = ...

init Controller(Normal,true,false,false,stopped);
```

Proceso - Interacciones Modo

```
proc Controller ( m:Mode,
  docked, rightmost, leftmost:Bool,
  ms:MotorStatus ) =
pressStop.releaseBrake.motorOff.
  Controller(Emergency,docked,rightmost,leftmost,ms)
+
pressResume.
  Controller(Normal,docked,rightmost,leftmost,ms)
+
...
```

Proceso - Interacciones Docking

```
proc Controller ( m:Mode,
  docked, rightmost, leftmost:Bool,
  ms:MotorStatus ) =
...
+
pressUndock.
  (docked && rightmost)
  -> applyBrake.unlockDock.
     Controller(m,false,rightmost,leftmost,ms)
  <> Controller(m,docked,rightmost,leftmost,ms)
+
isDocked.
  Controller(m,true,rightmost,leftmost,ms)
+
...
```

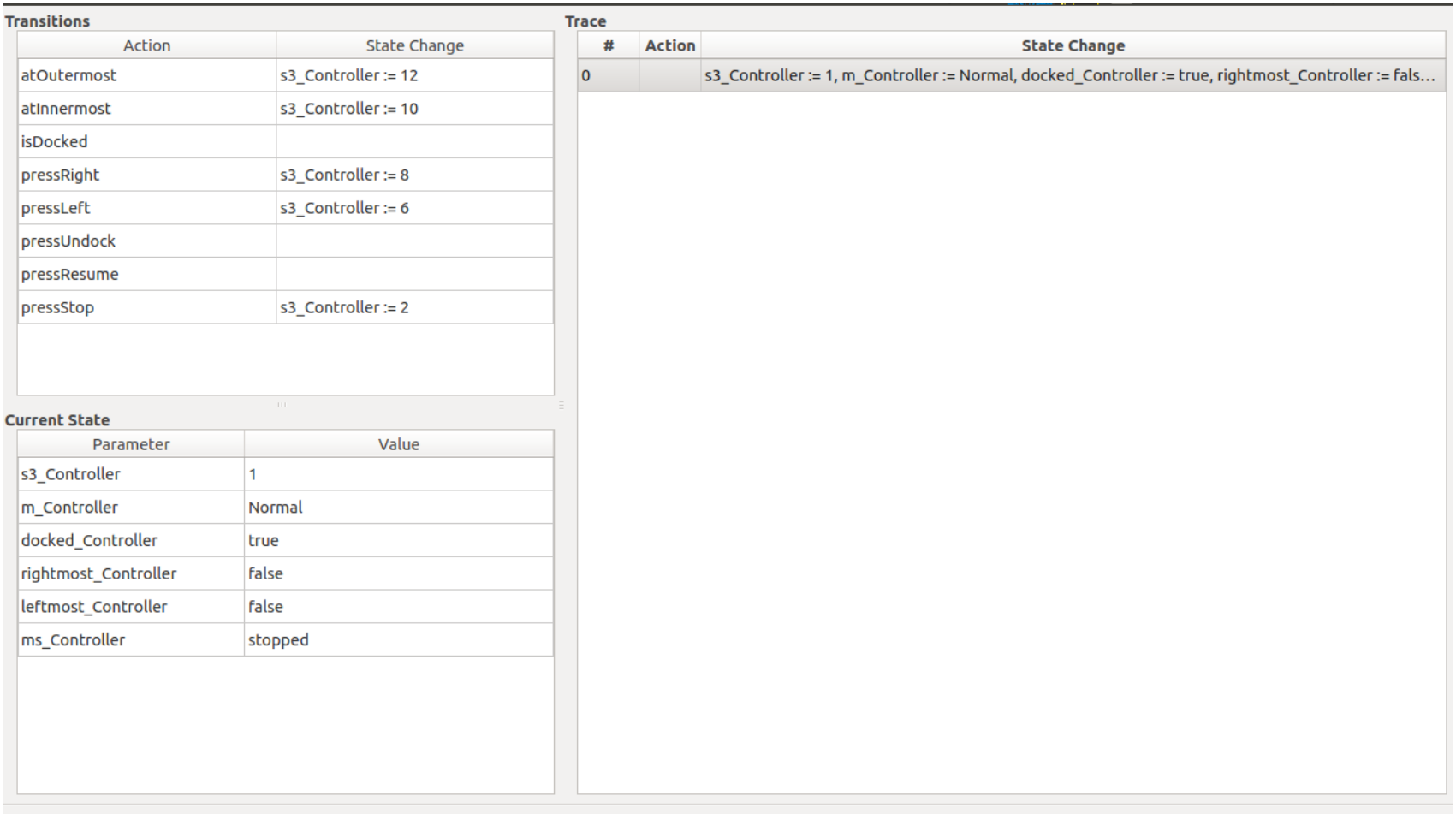
Proceso - Interacciones Motor

```
proc Controller ( m:Mode,
  docked, rightmost, leftmost:Bool,
  ms:MotorStatus ) =
...
+
pressLeft.
  (docked && ms!=moveleft && !leftmost && m==Normal)
  -> releaseBrake.motorLeft.
     Controller(m,docked,false,leftmost,moveleft)
  <> Controller(m,docked,rightmost,leftmost,ms)
+
pressRight.
  (docked && ms!=moveright && !rightmost && m==Normal)
  -> releaseBrake.motorRight.
     Controller(m,docked,rightmost,false,moveright)
  <> Controller(m,docked,rightmost,leftmost,ms)
+
...
```

Proceso - Interacciones Sensor

```
proc Controller ( m:Mode,
  docked, rightmost, leftmost:Bool,
  ms:MotorStatus ) =
...
+
atInnermost.motorOff.applyBrake.
  Controller(m,docked,true,false,stopped)
+
atOutermost.motorOff.applyBrake.
  Controller(m,docked,false,true,stopped);
```

Simulación



Transitions

Action	State Change
atOutermost	s3_Controller := 12
atInnermost	s3_Controller := 10
isDocked	
pressRight	s3_Controller := 8
pressLeft	s3_Controller := 6
pressUndock	
pressResume	
pressStop	s3_Controller := 2

Current State

Parameter	Value
s3_Controller	1
m_Controller	Normal
docked_Controller	true
rightmost_Controller	false
leftmost_Controller	false
ms_Controller	stopped

Trace

#	Action	State Change
0		s3_Controller := 1, m_Controller := Normal, docked_Controller := true, rightmost_Controller := fals…

Safety properties - Prop 1

La plataforma no debe tumbarse:
cuando el trolley no esta fijo al scanner,
la plataforma completa debe estar sobre el trolley,
los frenos aplicados y el motor apagado.

```
[!applyBrake*.unlockDock] false &&
[!atOutermost*.unlockDock] false &&
[!motorOff*.unlockDock] false &&
[true*.releaseBrake.!applyBrake*.unlockDock] false &&
[true*.motorLeft.!atOutermost*.unlockDock] false &&
[true*.(motorLeft+motorRight).!motorOff*.unlockDock] false
```

Corrección Docking

```
proc Controller ( m:Mode,
  docked, rightmost, leftmost:Bool,
  ms:MotorStatus ) =
...
+
pressUndock.
 ( (docked && leftmost) || ms==Emergency)
  -> applyBrake.unlockDock.
     Controller(m,false,rightmost,leftmost,ms)
  <> Controller(m,docked,rightmost,leftmost,ms)
+
isDocked.
  Controller(m,true,rightmost,leftmost,ms)
+
...
```

Safety properties - Prop 2

Al apretar “Stop”, un paciente puede ser sacado manualmente del scanner y de la habitación.

```
([true*.pressStop]mu X.[!motorOff]X) &&
([true*.pressStop]mu X.[!releaseBrake]X) &&
([true*.pressStop]mu X.[!unlockDock]X)
```

Relaxing property 2

Al apretar “Stop”, un paciente puede ser sacado manualmente del scanner y de la habitación.

```
([true*.pressStop]mu X.[!motorOff]X) &&
([true*.pressStop]mu X.[!releaseBrake]X) &&
([true*.pressStop] <true*.unlockDock>)
```

Safety properties - Prop 3

Para proteger el motor,
este no debe funcionar más allá
de la posición más interna o externa de la plataforma.

```
[true*.(atOutermost+atInnermost)]mu X.[!motorOff]X
```

Liveness properties - Prop 4

Si se presiona “Left”, el trolley esta fijo al scanner ($b1$),
no se está en modo emergencia ($b2$),
la plataforma esta completamente dentro del scanner ($b3$)
y el motor no está en movimiento hacia la izquierda ($b4$),
el motor se enciende hacia la derecha.

Liveness properties - Prop 4

```
nu X(b1:Bool=false,b2:Bool=true,b3:Bool=true,b4:Bool=true).
(
 [isDocked] X(true,b2,b3,b4) &&
 [unlockDock] X(false,b2,b3,b4) &&
 [pressResume] X(b1,true,b3,b4) &&
 [pressStop]X(b1,false,b3,b4) &&
 [atInnermost] X(b1,b2,true,b4) &&
 [motorRight] X(b1,b2,false,b4) &&
 [motorOff+motorRight] X(b1,b2,b3,true)&&
 [motorLeft] X(b1,b2,b3,false) &&
 ( val(b1 && b2 && b3 && b4)
   => [pressLeft]
      mu Y.[!motorLeft &&
        !unlockDock && !pressStop && !atInnermost]Y))
```

Liveness properties - Prop 5

Si se presiona “Right”,
el trolley esta fijo al scanner, no se está en modo emergencia,
la plataforma no esta completamente en sobre el trolley
y el motor no está en movimiento hacia la derecha,
el motor se enciende hacia la derecha.

Liveness properties - Prop 5

```
nu X(b1:Bool=false, b2:Bool=true,b3:Bool=true,b4:Bool=true).
(
 [isDocked] X(true,b2,b3,b4) &&
 [unlockDock] X(false,b2,b3,b4) &&
 [pressResume] X(b1,true,b3,b4) &&
 [pressStop] X(b1,false,b3,b4) &&
 [atOutermost] X(b1,b2,true,b4)&&
 [motorLeft] X(b1,b2,false,b4) &&
 [motorOff+motorLeft] X(b1,b2,b3,true)&&
 [motorRight] X(b1,b2,b3,false) &&
 ( val(b1 && b2 && b3 && b4)
   => [pressRight]
      mu Y.[!motorRight &&
        !unlockDock && !pressStop && !atOutermost]Y))
```

Estructura de la presentación

Líneas de investigación

- Análisis y verificación de hardware
- Análisis y verificación de software
- Estudio de relaciones (ej. compatibilidad) entre componentes/servicios
- Síntesis de comportamiento en base a escenarios
- Formalización y verificación de modelos dinámicos

Trabajo

Explicar en un texto de 1200-1500 palabras
como aplicaría a un problema de un dominio de su conocimiento
las técnicas de análisis y verificación de sistemas reactivos.

¡Muchas gracias por su atención!