

Verificando modelos, o en qué se parecen Pancho Dotto y los diseñadores de software crítico

Fernando Schapachnik¹

¹Dependable Systems Research Group,
Departamento de Computación, FCEyN,
Universidad de Buenos Aires, Buenos Aires, Argentina

8^{va} borrachera, Charla de Borrachos, DC, FCEN, UBA
Cosecha julio/2005

Todo los resultados que aquí se presentan son fruto del trabajo conjunto con mis directores de tesis, Alfredo Olivero y Víctor Braberman.

Ante todo...

Todo los resultados que aquí se presentan son fruto del trabajo conjunto con mis directores de tesis, Alfredo Olivero y Víctor Braberman.
Sí, soy un chupamedias.

¿En qué se parecen...?

Encuentre las 7 diferencias...



Ambos trabajan con modelos

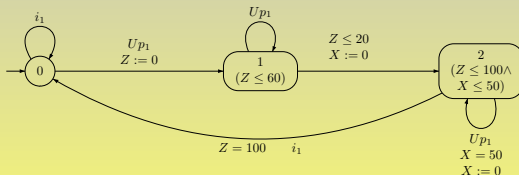


(a) Modelo

Ambos trabajan con modelos



(c) Modelo



(d) Modelo

Indistinguibles... como dos gotas de agua.

Indistinguibles...

Ambos son **exitosos**...

En la moda

En el software

Indistinguibles...

Ambos son **exitosos**...

En la moda	En el software
Giordano en Punta	Phillips Audio Protocol

Indistinguibles...

Ambos son exitosos...

En la moda	En el software
Giordano en Punta	Phillips Audio Protocol
Versace en Milán	SPARK (Ada)

Indistinguibles...

Ambos son **exitosos**...

En la moda	En el software
Giordano en Punta	Phillips Audio Protocol
Versace en Milán	SPARK (Ada)
Fashion Emergency	SPIN on Lucent's PathStar Access Server

Indistinguibles...

Ambos son **exitosos**...

En la moda	En el software
Giordano en Punta	Phillips Audio Protocol
Versace en Milán	SPARK (Ada)
Fashion Emergency	SPIN on Lucent's PathStar Access Server
Noche de los Óscares	Banf & Olufsen Audio/Video Protocol

Indistinguibles...

Ambos son **exitosos**...

En la moda	En el software
Giordano en Punta	Phillips Audio Protocol
Versace en Milán	SPARK (Ada)
Fashion Emergency	SPIN on Lucent's PathStar Access Server
Noche de los Óscares	Banf & Olufsen Audio/Video Protocol
Scouting Dotto Models 2004	Atelier B (Metro de Francia)

Indistinguibles...

Ambos son **exitosos**...

En la moda	En el software
Giordano en Punta	Phillips Audio Protocol
Versace en Milán	SPARK (Ada)
Fashion Emergency	SPIN on Lucent's PathStar Access Server
Noche de los Óscares	Banf & Olufsen Audio/Video Protocol
Scouting Dotto Models 2004	Atelier B (Metro de Francia)
	Raíz cuadrada en AMD-K5 (ACL.2 theorem prover)

Indistinguibles...

Ambos son **exitosos**...

En la moda	En el software
Giordano en Punta	Phillips Audio Protocol
Versace en Milán	SPARK (Ada)
Fashion Emergency	SPIN on Lucent's PathStar Access Server
Noche de los Óscares	Banf & Olufsen Audio/Video Protocol
Scouting Dotto Models 2004	Atelier B (Metro de Francia)
	Raíz cuadrada en AMD-K5 (ACL.2 theorem prover)
	Escher Technologies' Perfect Developer

Indistinguishables... (cont.)

Los modelos de ambos tienen **propiedades deseables...**

Indistinguibles... (cont.)

Los modelos de ambos tienen **propiedades deseables...**



- “El sistema nunca muestra datos demasiado viejos”
- “Si acciono el freno de emergencia, el tren frena”

Hablemos de modelos

- Abstracción

Hablemos de modelos

- Abstracción
- Temporizados vs. no temporizados

Hablemos de modelos

- Abstracción
- Temporizados vs. no temporizados
- Tiempo real vs. temporización relativa

Modelos temporizados

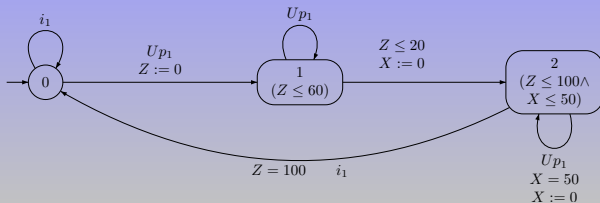


Figura: Un autómata temporizado

- Modelo:
 - Autómata temporizado: automata + relojes con valores en \mathbb{R} .
- I.e., grafo con **restricciones temporales**:
 - En las locaciones (circulitos): cuánto quedarse.
 - En las transiciones (flechas): condición de habilitación.

Modelos temporizados

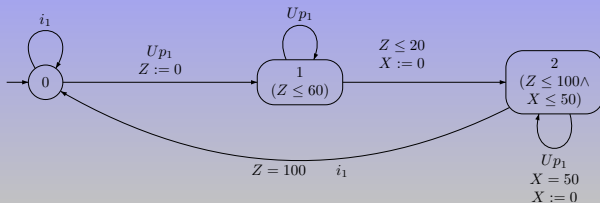
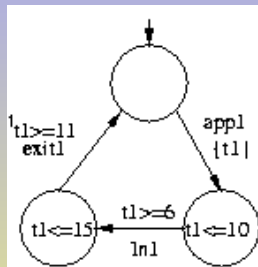


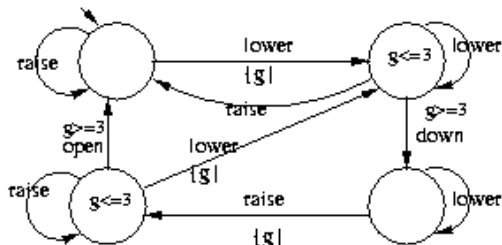
Figura: Un autómata temporizado

- Modelo:
 - Autómata temporizado: automata + relojes con valores en \mathbb{R} .
- I.e., grafo con **restricciones temporales**:
 - En las locaciones (circulitos): cuánto quedarse.
 - En las transiciones (flechas): condición de habilitación.
- Propiedades:
 - Las locaciones se “etiquetan” con variables booleanas.
 - Se utiliza una lógica (TCTL) para propiedades complejas.
 - La mayoría se resuelve por **alcanzabilidad**.

Ejemplo



TRAIN



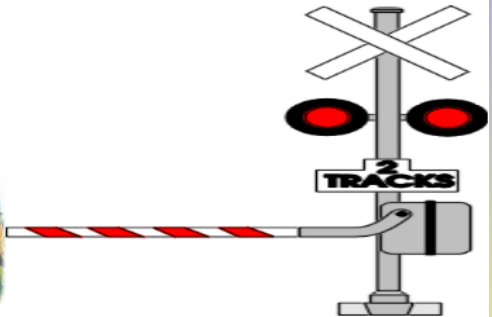
GATE

```
1  # Importar las librerías necesarias
2  import numpy as np
3  import pandas as pd
4  import matplotlib.pyplot as plt
5  import seaborn as sns
6  import sklearn.preprocessing as preprocessing
7  import sklearn.metrics as metrics
8  import sklearn.model_selection as model_selection
9  import sklearn.linear_model as linear_model
10 import sklearn.svm as svm
11 import sklearn.tree as tree
12 import sklearn.ensemble as ensemble
13 import sklearn.neighbors as neighbors
14 import sklearn.cluster as cluster
15 import sklearn.decomposition as decomposition
16 import sklearn.pipeline as pipeline
17 import sklearn.metrics.pairwise as pairwise
18 import sklearn.metrics.cluster as cluster_metrics
19 import sklearn.metrics.classification as classification_metrics
20 import sklearn.metrics.regression as regression_metrics
21 import sklearn.metrics.text_similarity as text_similarity
22 import sklearn.metrics.image_similarity as image_similarity
23 import sklearn.metrics.audio_similarity as audio_similarity
24 import sklearn.metrics.video_similarity as video_similarity
25 import sklearn.metrics.location_similarity as location_similarity
26 import sklearn.metrics.time_similarity as time_similarity
27 import sklearn.metrics.temperature_similarity as temperature_similarity
28 import sklearn.metrics.humidity_similarity as humidity_similarity
29 import sklearn.metrics.pressure_similarity as pressure_similarity
30 import sklearn.metrics.wind_similarity as wind_similarity
31 import sklearn.metrics.cloud_similarity as cloud_similarity
32 import sklearn.metrics.visibility_similarity as visibility_similarity
33 import sklearn.metrics.precipitation_similarity as precipitation_similarity
34 import sklearn.metrics.solar_radiation_similarity as solar_radiation_similarity
35 import sklearn.metrics.air_quality_similarity as air_quality_similarity
36 import sklearn.metrics.noise_similarity as noise_similarity
37 import sklearn.metrics.light_similarity as light_similarity
38 import sklearn.metrics.magnetic_field_similarity as magnetic_field_similarity
39 import sklearn.metrics.electromagnetic_interference_similarity as electromagnetic_interference_similarity
40 import sklearn.metrics.vibration_similarity as vibration_similarity
41 import sklearn.metrics.acceleration_similarity as acceleration_similarity
42 import sklearn.metrics.rotation_similarity as rotation_similarity
43 import sklearn.metrics.orientation_similarity as orientation_similarity
44 import sklearn.metrics.location_similarity as location_similarity
45 import sklearn.metrics.time_similarity as time_similarity
46 import sklearn.metrics.temperature_similarity as temperature_similarity
47 import sklearn.metrics.humidity_similarity as humidity_similarity
48 import sklearn.metrics.pressure_similarity as pressure_similarity
49 import sklearn.metrics.wind_similarity as wind_similarity
50 import sklearn.metrics.cloud_similarity as cloud_similarity
51 import sklearn.metrics.visibility_similarity as visibility_similarity
52 import sklearn.metrics.precipitation_similarity as precipitation_similarity
53 import sklearn.metrics.solar_radiation_similarity as solar_radiation_similarity
54 import sklearn.metrics.air_quality_similarity as air_quality_similarity
55 import sklearn.metrics.noise_similarity as noise_similarity
56 import sklearn.metrics.light_similarity as light_similarity
57 import sklearn.metrics.magnetic_field_similarity as magnetic_field_similarity
58 import sklearn.metrics.electromagnetic_interference_similarity as electromagnetic_interference_similarity
59 import sklearn.metrics.vibration_similarity as vibration_similarity
60 import sklearn.metrics.acceleration_similarity as acceleration_similarity
61 import sklearn.metrics.rotation_similarity as rotation_similarity
62 import sklearn.metrics.orientation_similarity as orientation_similarity
63 import sklearn.metrics.location_similarity as location_similarity
64 import sklearn.metrics.time_similarity as time_similarity
65 import sklearn.metrics.temperature_similarity as temperature_similarity
66 import sklearn.metrics.humidity_similarity as humidity_similarity
67 import sklearn.metrics.pressure_similarity as pressure_similarity
68 import sklearn.metrics.wind_similarity as wind_similarity
69 import sklearn.metrics.cloud_similarity as cloud_similarity
70 import sklearn.metrics.visibility_similarity as visibility_similarity
71 import sklearn.metrics.precipitation_similarity as precipitation_similarity
72 import sklearn.metrics.solar_radiation_similarity as solar_radiation_similarity
73 import sklearn.metrics.air_quality_similarity as air_quality_similarity
74 import sklearn.metrics.noise_similarity as noise_similarity
75 import sklearn.metrics.light_similarity as light_similarity
76 import sklearn.metrics.magnetic_field_similarity as magnetic_field_similarity
77 import sklearn.metrics.electromagnetic_interference_similarity as electromagnetic_interference_similarity
78 import sklearn.metrics.vibration_similarity as vibration_similarity
79 import sklearn.metrics.acceleration_similarity as acceleration_similarity
80 import sklearn.metrics.rotation_similarity as rotation_similarity
81 import sklearn.metrics.orientation_similarity as orientation_similarity
82 import sklearn.metrics.location_similarity as location_similarity
83 import sklearn.metrics.time_similarity as time_similarity
84 import sklearn.metrics.temperature_similarity as temperature_similarity
85 import sklearn.metrics.humidity_similarity as humidity_similarity
86 import sklearn.metrics.pressure_similarity as pressure_similarity
87 import sklearn.metrics.wind_similarity as wind_similarity
88 import sklearn.metrics.cloud_similarity as cloud_similarity
89 import sklearn.metrics.visibility_similarity as visibility_similarity
90 import sklearn.metrics.precipitation_similarity as precipitation_similarity
91 import sklearn.metrics.solar_radiation_similarity as solar_radiation_similarity
92 import sklearn.metrics.air_quality_similarity as air_quality_similarity
93 import sklearn.metrics.noise_similarity as noise_similarity
94 import sklearn.metrics.light_similarity as light_similarity
95 import sklearn.metrics.magnetic_field_similarity as magnetic_field_similarity
96 import sklearn.metrics.electromagnetic_interference_similarity as electromagnetic_interference_similarity
97 import sklearn.metrics.vibration_similarity as vibration_similarity
98 import sklearn.metrics.acceleration_similarity as acceleration_similarity
99 import sklearn.metrics.rotation_similarity as rotation_similarity
100 import sklearn.metrics.orientation_similarity as orientation_similarity
```

Ejemplo



TRAIN



GATE

Model checking

¿Qué modelamos?

- Sistemas de tiempo real.

Model checking

¿Qué modelamos?

- Sistemas de tiempo real.
- En general, críticos.

Model checking

¿Qué modelamos?

- Sistemas de tiempo real.
- En general, críticos.
- Interacciones complejas –y temporizadas– entre los componentes.

Model checking

¿Qué modelamos?

- Sistemas de tiempo real.
- En general, críticos.
- Interacciones complejas –y temporizadas– entre los componentes.

¿Qué nos interesa?

- Buscamos responder de manera exacta: ¿el sistema S cumple la propiedad P ?

Model checking

¿Qué modelamos?

- Sistemas de tiempo real.
- En general, críticos.
- Interacciones complejas –y temporizadas– entre los componentes.

¿Qué nos interesa?

- Buscamos responder de manera exacta: ¿el sistema S cumple la propiedad P ?
- Si no, queremos obtener un *contraejemplo*.

Model checking

¿Qué modelamos?

- Sistemas de tiempo real.
- En general, críticos.
- Interacciones complejas –y temporizadas– entre los componentes.

¿Qué nos interesa?

- Buscamos responder de manera exacta: ¿el sistema S cumple la propiedad P ?
- Si no, queremos obtener un *contraejemplo*.

¿Cómo lo hacemos?

- Recorremos todo el espacio de estados.

Model checking

¿Qué modelamos?

- Sistemas de tiempo real.
- En general, críticos.
- Interacciones complejas –y temporizadas– entre los componentes.

¿Qué nos interesa?

- Buscamos responder de manera exacta: ¿el sistema S cumple la propiedad P ?
- Si no, queremos obtener un *contraejemplo*.

¿Cómo lo hacemos?

- Recorremos todo el espacio de estados.
- Muy difícil hacerlo “a mano”.

Model checking

¿Qué modelamos?

- Sistemas de tiempo real.
- En general, críticos.
- Interacciones complejas –y temporizadas– entre los componentes.

¿Qué nos interesa?

- Buscamos responder de manera exacta: *¿el sistema S cumple la propiedad P ?*
- Si no, queremos obtener un *contraejemplo*.

¿Cómo lo hacemos?

- Recorremos todo el espacio de estados.
- Muy difícil hacerlo “a mano”.
- Para las máquinas, conceptualmente simple...

Model checking

¿Qué modelamos?

- Sistemas de tiempo real.
- En general, críticos.
- Interacciones complejas –y temporizadas– entre los componentes.

¿Qué nos interesa?

- Buscamos responder de manera exacta: ¿el sistema S cumple la propiedad P ?
- Si no, queremos obtener un *contraejemplo*.

¿Cómo lo hacemos?

- Recorremos todo el espacio de estados.
- Muy difícil hacerlo “a mano”.
- Para las máquinas, conceptualmente simple...
- ...si tuviésemos tiempo infinito: problema de la explosión combinatoria de la cantidad de estados

Model checking

¿Qué modelamos?

- Sistemas de tiempo real.
- En general, críticos.
- Interacciones complejas –y temporizadas– entre los componentes.

¿Qué nos interesa?

- Buscamos responder de manera exacta: *¿el sistema S cumple la propiedad P ?*
- Si no, queremos obtener un *contraejemplo*.

¿Cómo lo hacemos?

- Recorremos todo el espacio de estados.
- Muy difícil hacerlo “a mano”.
- Para las máquinas, conceptualmente simple...
- ...si tuviésemos tiempo infinito: problema de la *explosión combinatoria de la cantidad de estados*

¿Qué modela?

Model checking

¿Qué modelamos?

- Sistemas de tiempo real.
- En general, críticos.
- Interacciones complejas –y temporizadas– entre los componentes.

¿Qué nos interesa?

- Buscamos responder de manera exacta: ¿el sistema S cumple la propiedad P ?
- Si no, queremos obtener un *contraejemplo*.

¿Cómo lo hacemos?

- Recorremos todo el espacio de estados.
- Muy difícil hacerlo “a mano”.
- Para las máquinas, conceptualmente simple...
- ...si tuviésemos tiempo infinito: problema de la explosión combinatoria de la cantidad de estados

¿Qué modela?

- Ropa, de tamaño inversamente proporcional al precio.

Model checking

¿Qué modelamos?

- Sistemas de tiempo real.
- En general, críticos.
- Interacciones complejas –y temporizadas– entre los componentes.

¿Qué nos interesa?

- Buscamos responder de manera exacta: *¿el sistema S cumple la propiedad P ?*
- Si no, queremos obtener un *contraejemplo*.

¿Cómo lo hacemos?

- Recorremos todo el espacio de estados.
- Muy difícil hacerlo “a mano”.
- Para las máquinas, conceptualmente simple...
- ...si tuviésemos tiempo infinito: problema de la *explosión combinatoria de la cantidad de estados*

¿Qué modela?

- Ropa, de tamaño inversamente proporcional al precio.

¿Qué le interesa?

Model checking

¿Qué modelamos?

- Sistemas de tiempo real.
- En general, críticos.
- Interacciones complejas –y temporizadas– entre los componentes.

¿Qué nos interesa?

- Buscamos responder de manera exacta: ¿el sistema S cumple la propiedad P ?
- Si no, queremos obtener un *contraejemplo*.

¿Cómo lo hacemos?

- Recorremos todo el espacio de estados.
- Muy difícil hacerlo “a mano”.
- Para las máquinas, conceptualmente simple...
- ...si tuviésemos tiempo infinito: problema de la explosión combinatoria de la cantidad de estados

¿Qué modela?

- Ropa, de tamaño inversamente proporcional al precio.

¿Qué le interesa?

- “Conocer” a sus modelos...

Model checking

¿Qué modelamos?

- Sistemas de tiempo real.
- En general, críticos.
- Interacciones complejas –y temporizadas– entre los componentes.

¿Qué nos interesa?

- Buscamos responder de manera exacta: *¿el sistema S cumple la propiedad P ?*
- Si no, queremos obtener un *contraejemplo*.

¿Cómo lo hacemos?

- Recorremos todo el espacio de estados.
- Muy difícil hacerlo “a mano”.
- Para las máquinas, conceptualmente simple...
- ...si tuviésemos tiempo infinito: problema de la *explosión combinatoria de la cantidad de estados*

¿Qué modela?

- Ropa, de tamaño inversamente proporcional al precio.

¿Qué le interesa?

- “Conocer” a sus modelos...

¿Cómo lo hace?

Model checking

¿Qué modelamos?

- Sistemas de tiempo real.
- En general, críticos.
- Interacciones complejas –y temporizadas– entre los componentes.

¿Qué nos interesa?

- Buscamos responder de manera exacta: ¿el sistema S cumple la propiedad P ?
- Si no, queremos obtener un *contraejemplo*.

¿Cómo lo hacemos?

- Recorremos todo el espacio de estados.
- Muy difícil hacerlo “a mano”.
- Para las máquinas, conceptualmente simple...
- ...si tuviésemos tiempo infinito: problema de la explosión combinatoria de la cantidad de estados

¿Qué modela?

- Ropa, de tamaño inversamente proporcional al precio.

¿Qué le interesa?

- “Conocer” a sus modelos...

¿Cómo lo hace?

- Preferible hacerlo sin la asistencia de máquinas...

La noche de Pancho

Modelemos las expectativas de Pancho a lo largo de la noche...

Comienza la noche



Claudia

La noche de Pancho

Modelemos las expectativas de Pancho a lo largo de la noche...

Comienza la noche



Claudia
($X \leq 2\text{AM}$)

La noche de Pancho

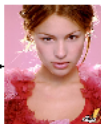
Modelemos las expectativas de Pancho a lo largo de la noche...

Comienza la noche



Claudia
($X \leq 2\text{AM}$)

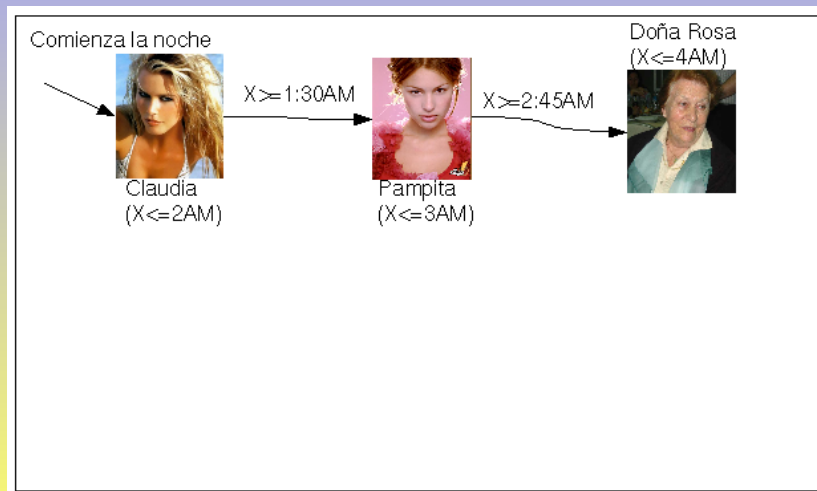
$X \geq 1:30\text{AM}$



Pampita
($X \leq 3\text{AM}$)

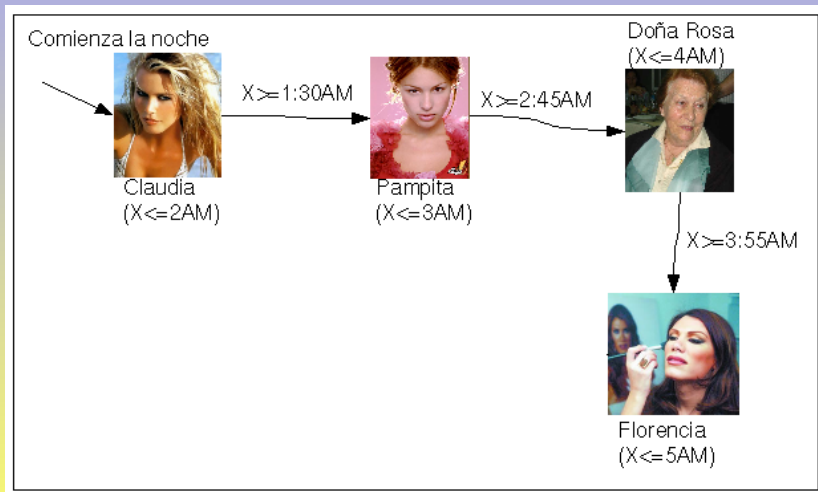
La noche de Pancho

Modelemos las expectativas de Pancho a lo largo de la noche...



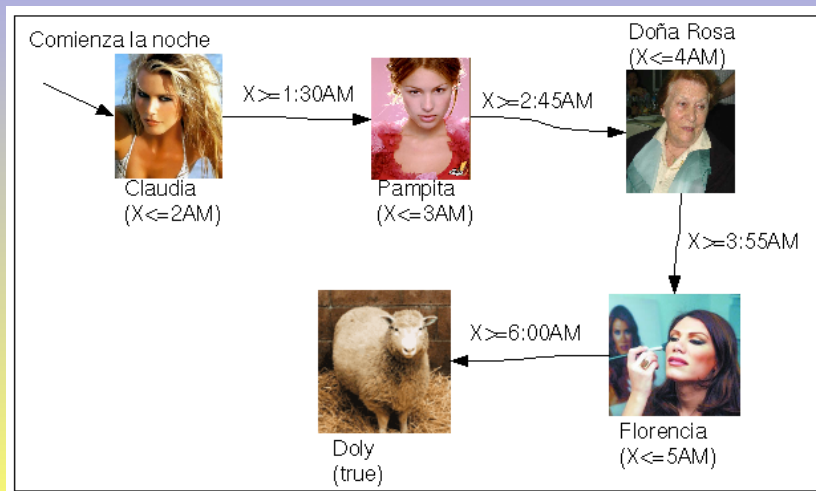
La noche de Pancho

Modelemos las expectativas de Pancho a lo largo de la noche...



La noche de Pancho

Modelemos las expectativas de Pancho a lo largo de la noche...



Representación simbólica de los estados

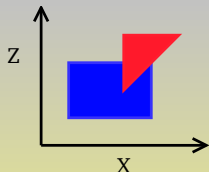
Estructura de datos: DBM, que representa un fragmento del espacio de tiempo.

Representación simbólica de los estados

Estructura de datos: DBM, que representa un fragmento del espacio de tiempo.

$$(1 \leq X \leq 4 \wedge 1 \leq Z \leq 3) \vee (3 \leq X \wedge Z \leq 4 \wedge X - Z \leq 1)$$

(a) Ecuaciones.



(b) Representación gráfica.

	0	X	Z
0	0	-1	-1
X	4	0	3
Z	3	2	0

	0	X	Z
0	0	-3	-2
X	5	0	1
Z	4	1	0

(c) Representación como DBM.

Figura: Ejemplo de representación simbólica de estados

Toma mucho tiempo...

Toma mucho tiempo...

Complejidad: $O(n!2^n C^n)$, donde n es la cantidad de relojes del sistema y C la cte. máxima del sistema.

Autómata	Loc.	Trans.	Rel.	Tiempo
Cada tren	3	3	1	
Barrera	4	10	1	
Control (5 trenes)	18	205	1	
Control (6 trenes)	21	280	1	
Control (7 trenes)	24	368	1	
Observador	5	22	1	
Sistema (5 trenes)	4764	27850	8	248'
Sistema (6 trenes)	14388	90262	9	∞ (500' \times 3)
Sistema (7 trenes)	43356	339874	10	∞

Solución obvia (¿?)

- **Computación distribuida:** Zeus 2002 (asincrónico)

Solución obvia (¿?)

- **Computación distribuida:** Zeus 2002 (asincrónico)
- Algunos resultados esperanzadores

Solución obvia (¿?)

- **Computación distribuida:** Zeus 2002 (asincrónico)
- Algunos resultados esperanzadores
- Otros no tanto...

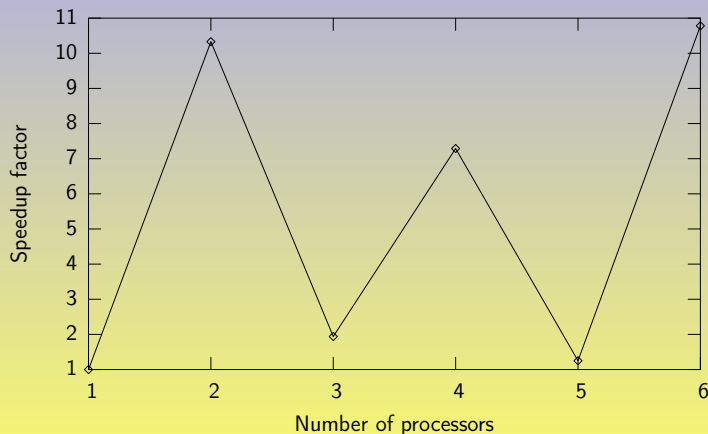


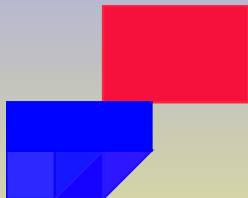
Figura: Aceleración versión sincrónica (RCS5)

¿Por qué?

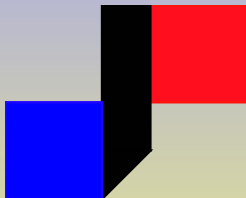
Conozcan al enemigo número 1:

¿Por qué?

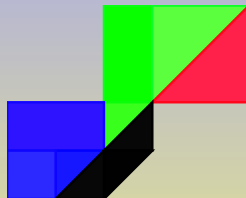
Conozcan al enemigo número 1: **fragmentación**



(a) Dos convexos.



(b) Tres convexos.



(c) Cuatro convexos.

Figura: Mismo espacio temporal, distintas representaciones.

¿Le gustaría
saber más al
respecto?



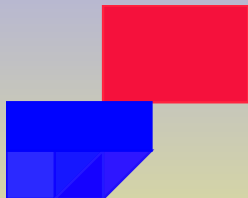
Dr. Víctor Braberman
Director del Grupo

Curse ART

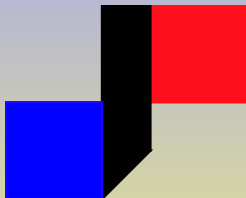
- Materia optativa para la Licenciatura y el Doctorado.
- 3 maravillosos puntos.
- ¡Inscríbase ya!

¿Por qué?

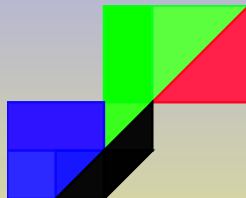
Conozcan al enemigo número 1: **fragmentación**



(a) Dos convexos.



(b) Tres convexos.



(c) Cuatro convexos.

Figura: Mismo espacio temporal, distintas representaciones.

Fragmentación

- Incrementa tanto los requerimientos de memoria como el número de operaciones.
- Solución (Zeus 2003): versión **sincrónica**...

Fragmentación

- Incrementa tanto los requerimientos de memoria como el número de operaciones.
- Solución (Zeus 2003): versión **sincrónica**... para respetar el orden de evaluación del monoprocesador.
- Resultado: no más fragmentación extra, pero...

Resultados sincrónicos

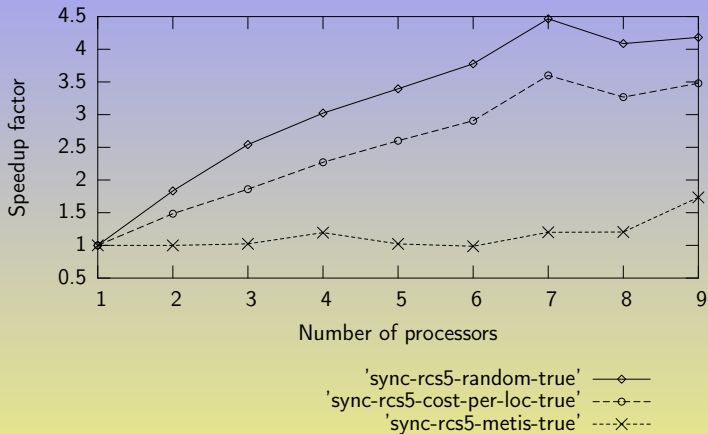
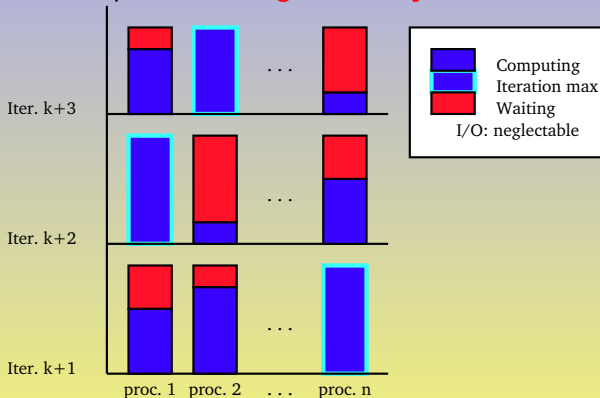


Figura: Ganancia de la versión sincrónica (RCS5).

Mejor, pero todavía mucho por hacer...

Resultados sincrónicos (cont.)

Otro problema: **carga de trabajo fluctuante**



- Zeus 2004: balanceo de carga + predicción.

- Zeus 2004: balanceo de carga + predicción.
- Zeus 2005: nuevo algoritmo de verificación (en progreso).

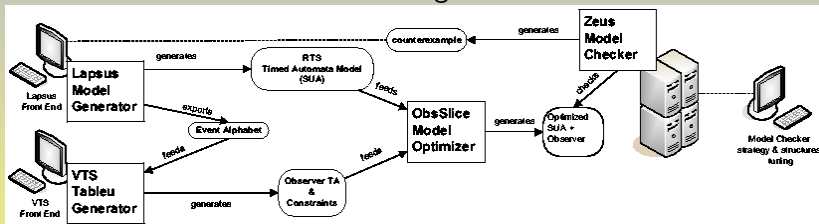
- Zeus 2004: balanceo de carga + predicción.
- Zeus 2005: nuevo algoritmo de verificación (en progreso).
- Nuevas estructuras de datos (en parte, tesis de Lic. en curso de Esteban Pavese).

- Zeus 2004: balanceo de carga + predicción.
- Zeus 2005: nuevo algoritmo de verificación (en progreso).
- Nuevas estructuras de datos (en parte, tesis de Lic. en curso de Esteban Pavese).
- Técnicas de reducción.

Zeus no está solo

- Lapsus
- VTS
- ObsSlice

VInTiMe: Verifier of INtegrated TImed ModELs



IBM Eclipse Innovation Grant 2005

Java - CircuitEditPart.java - Eclipse Platform

File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer

- demo
 - org.eclipse.jface.examples.logic
 - org.eclipse.jface.examples.logic.designer
 - actions
 - DecrementRangeAction.java
 - IncrementDecrementAction.java
 - IncrementRangeAction.java
 - LogicActionBarContributor.java
 - LogicPasteTemplateAction.java
 - PasteTemplateAction.java
 - dnd
 - edit
 - CircuitEditPart.java
 - ContainerHighLightPolicy.java
 - GateEditPart.java
 - GraphicalPartFactory.java
 - LabelCellEditorLocator.java
 - LabelDirectEditPolicy.java
 - LEDEditPart.java
 - LEDEditPolicy.java
 - LogicContainerEditPart.java
 - LogicContainerEditPolicy.java
 - LogicContainerTreeEditPart.java
 - LogicDiagramEditPart.java
 - LogicEditPart.java

Outline

```

1 state: 0 /* Open */
2 prop: Wait
3 invar: ( true )
4 trans:
5 true => I:GoDown; reset ( g ); goto 1
6 true => I:GoUp; reset ( ); goto 0
7
8 state: 1 /* near */
9 prop: Finish
10 invar: ( g <= 3 )
11 trans:
12 g >= 3 => O:Down; reset ( ); goto 2
13 true => I:GoDown; reset ( ); goto 1
14 true => I:GoUp; reset ( ); goto 0
15
16 state: 2 /* close */
17 prop: Retry
18 invar: ( true )
19 trans:
20 true => I:GoUp; reset ( g ); goto 3
21 true => I:GoDown; reset ( ); goto 2
22
23
24
25

```

UML State Machine Diagram:

- State 0 (Start): Wait. Transitions: True/!begin/x' to State 3; True/!x'=0 to State 1; True/!busy/x'=0 to State 1; True/?CD/x'=0 to State 2.
- State 1 (near): Finish. Transitions: True/!x'=0 to State 0; True/?CD/x'=0 to State 2.
- State 2 (close): Retry. Transitions: True/!x'=0 to State 0; True/?CD/x'=0 to State 1.
- State 3 (Finish): No outgoing transitions.

Tasks (7 items)

✓	!	Description	Resource	In Folder
✗		The project was not built due to classpath errors (incomplete or involved in cycle).	org.eclipse.jface	
✗		Missing required Java project resources.	org.eclipse.jface	
✗		Missing required Java project resources.	org.eclipse.jface	

Package Explorer Hierarchy

7 items: 0 tasks, 3 errors, 4 warnings, 0 infos

Nuestro trabajo involucra los siguientes temas:

- Lógica.
- Algorítmica.
- Programación.
- Computación distribuida.
- Estructuras de datos.
- Ingeniería del Software (¡de la buena!)

- <http://dependex.dc.uba.ar>
- <http://dependex.dc.uba.ar/vintime>
- <http://dependex.dc.uba.ar/~fpscha>

Gracias