Conceptos de Model Checking

Pedro R. D'Argenio

Universidad Nacional de Córdoba – CONICET (AR)

Saarland University (DE)

http://www.cs.famaf.unc.edu.ar/~dargenio/

CONICET

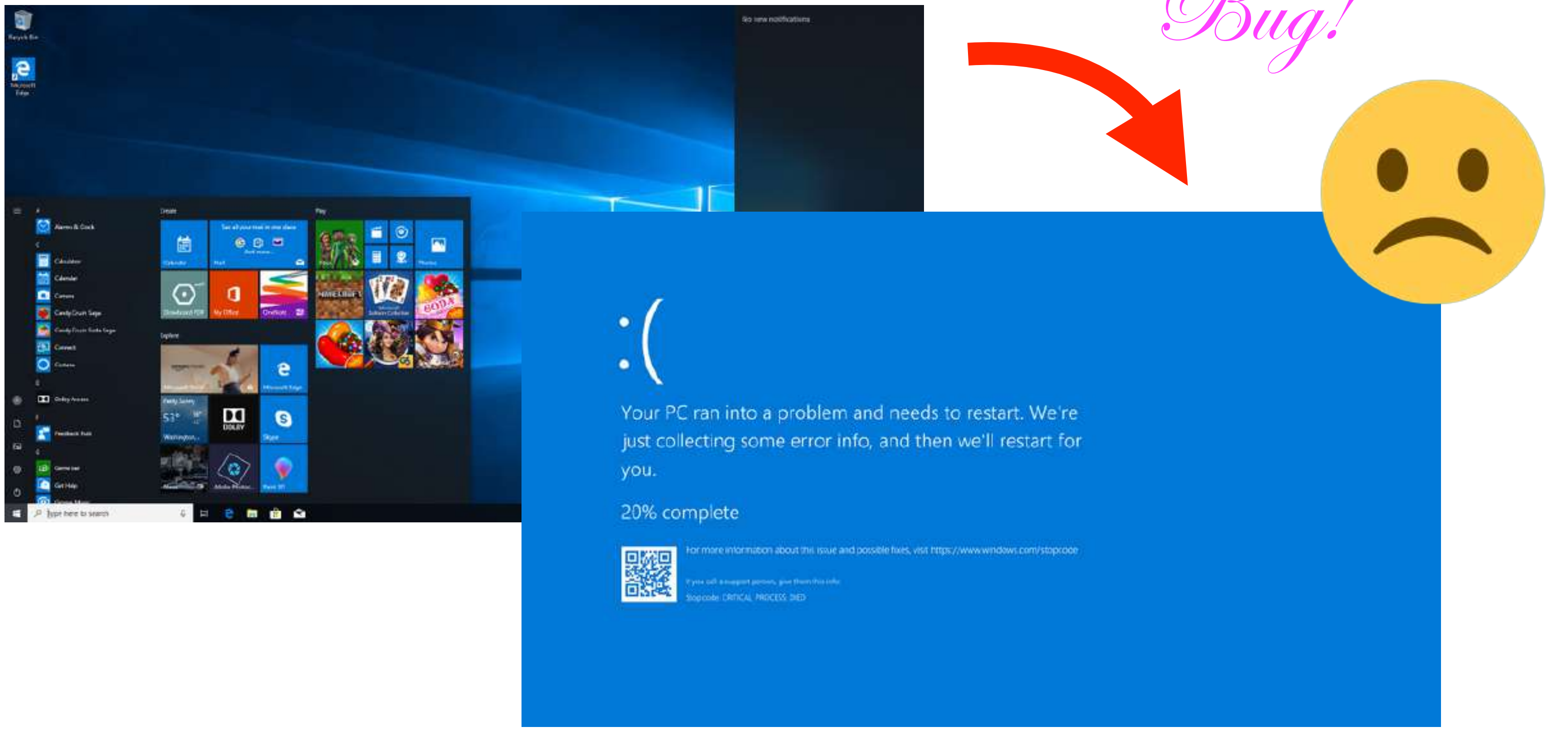
RIO 2019 - UNRC - Río Cuarto

UNC

La tolerancia al *Bug*

La tolerancia al *Bug*

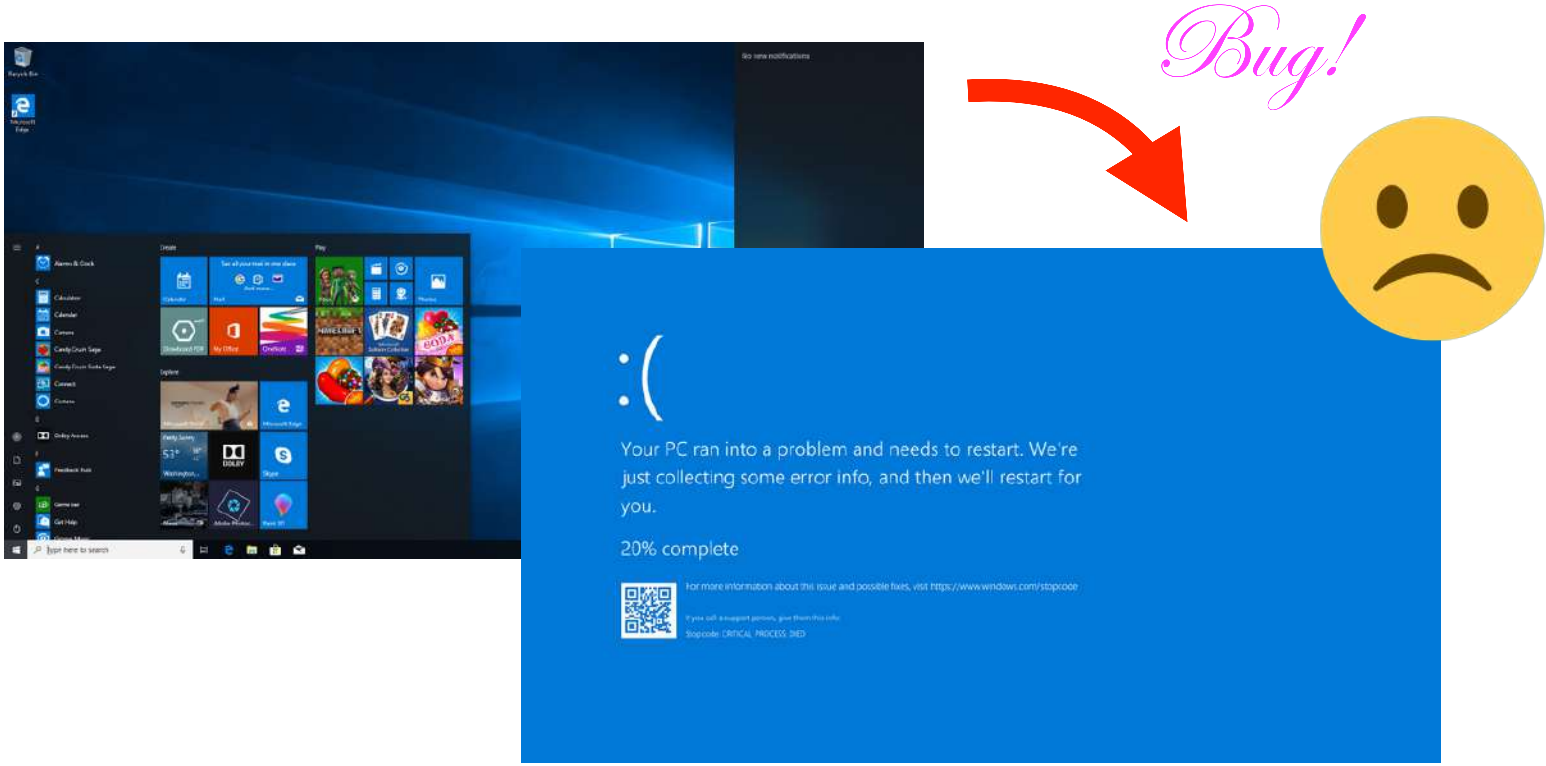
Bug!

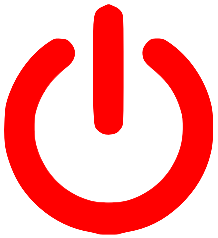


CONICET

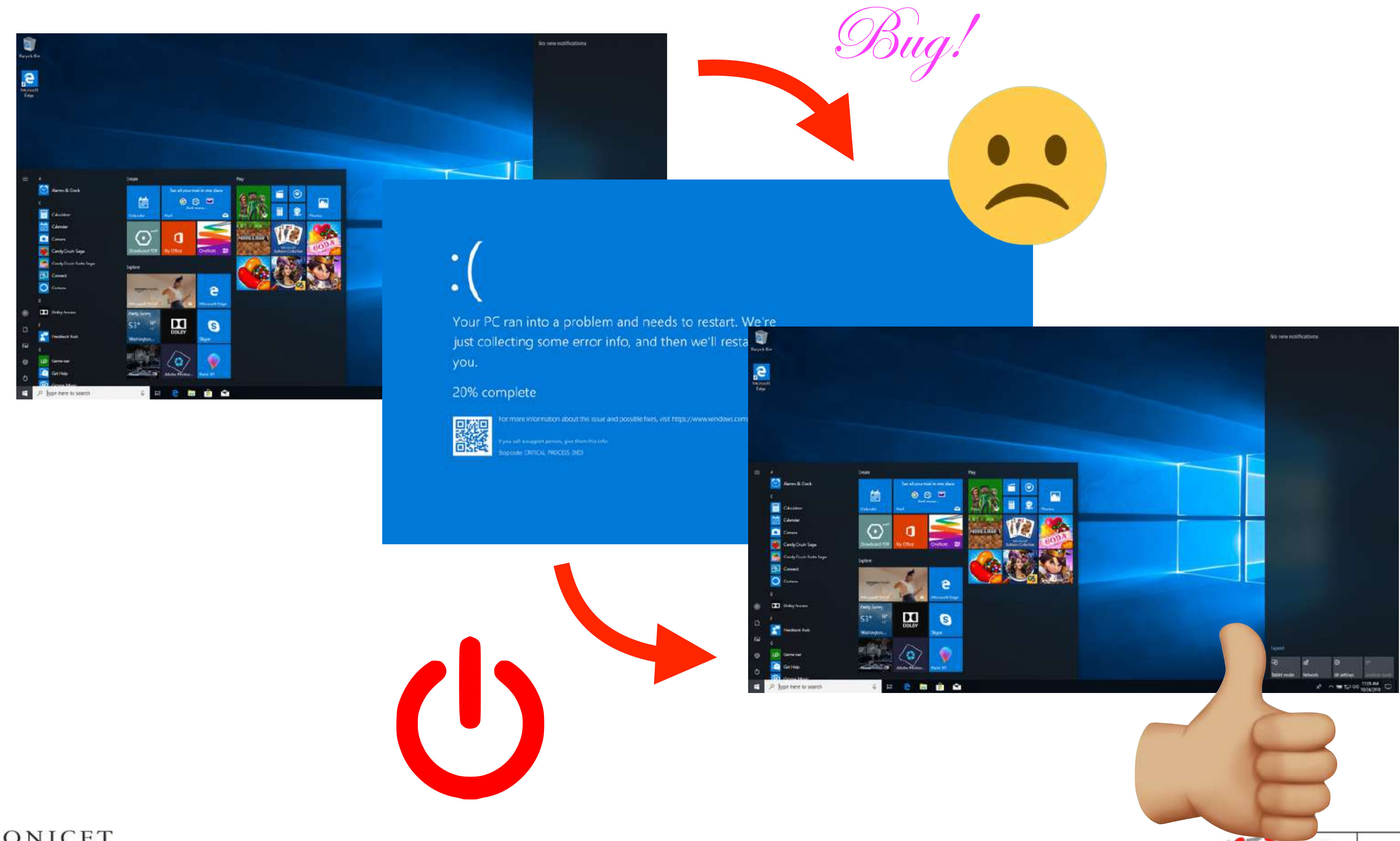
UNC

La tolerancia al *Bug*





La tolerancia al *Bug*

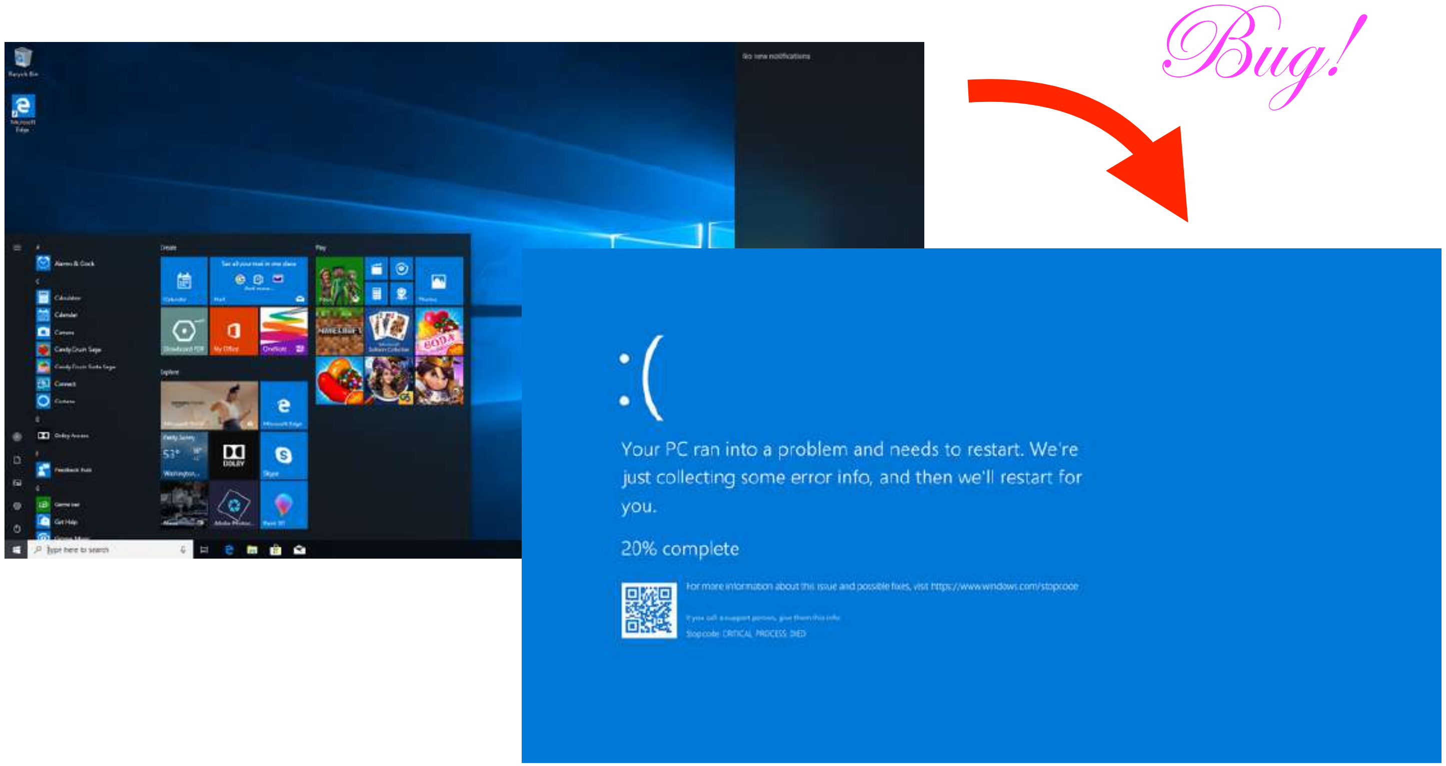


CONICET

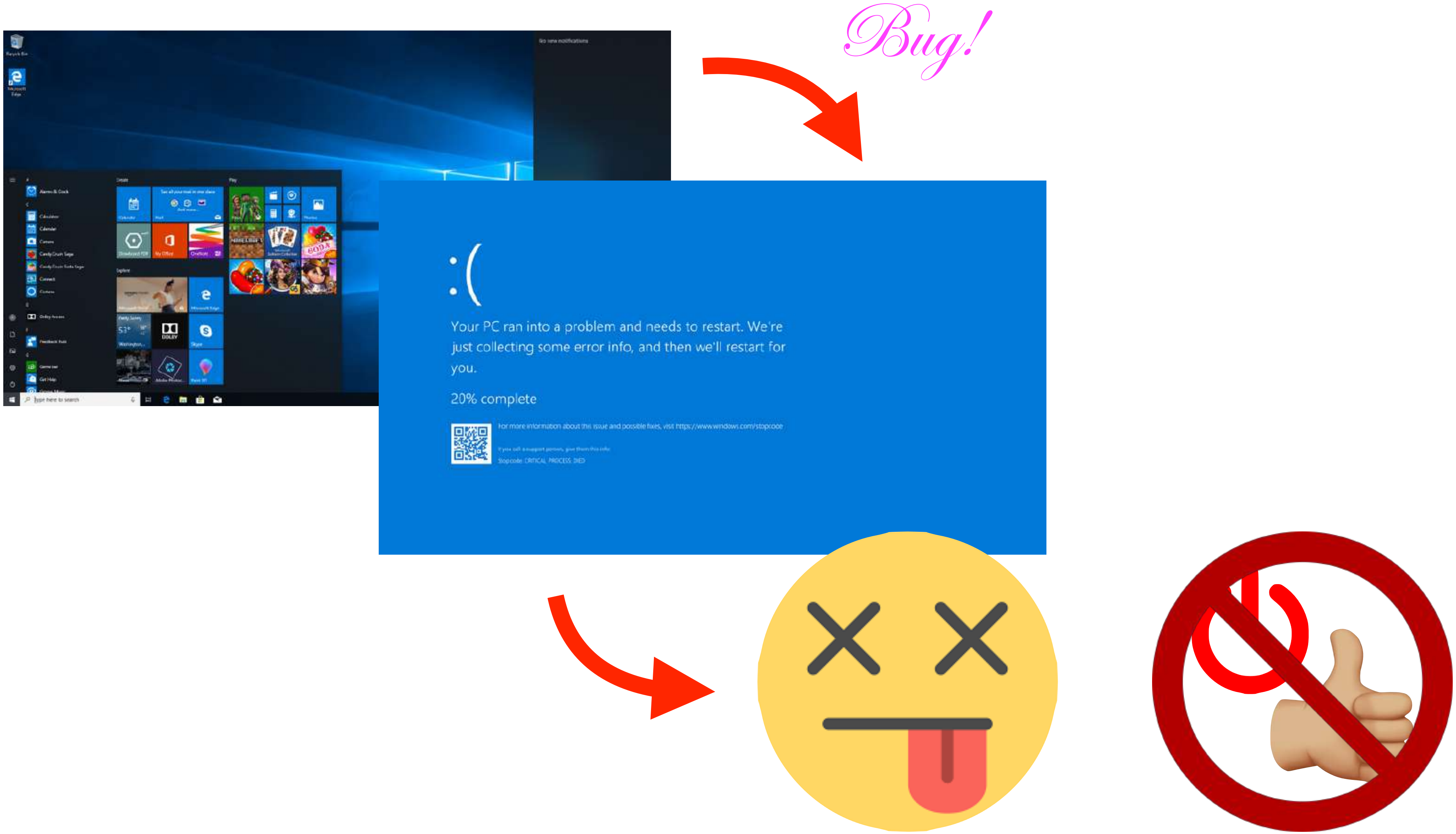
UNC

Sistema Crítico

Sistema Crítico



Sistema Crítico



Sistema Crítico

Es un sistema cuyo mal funcionamiento puede resultar en:

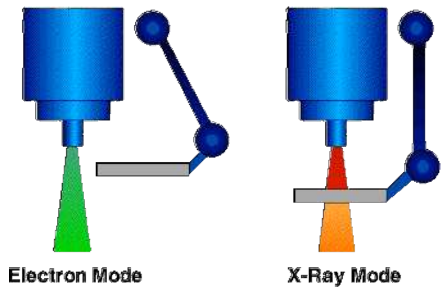
- Muerte o lesiones de individuos,
- Pérdida o daño de propiedad o equipamientos, o
- Daño ambiental

CONICET

UNC

Bugs famosos

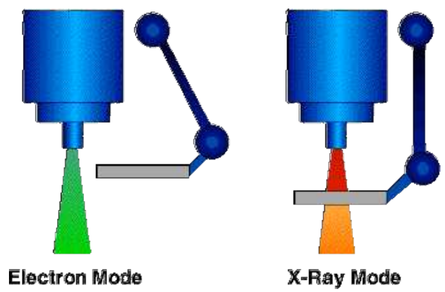
Bugs famosos



Therac-25:
Condición de carrera

Bugs famosos

Pentium:
FDIV

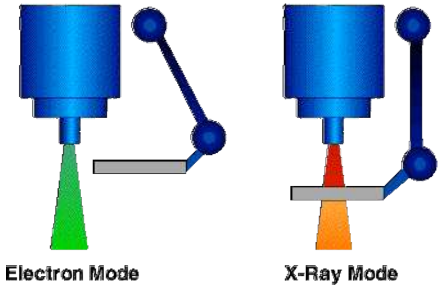


Therac-25:
Condición de carrera

Bugs famosos

Pentium:
FDIV

Ariane 5:
64 bits fp
vs 16 bits int

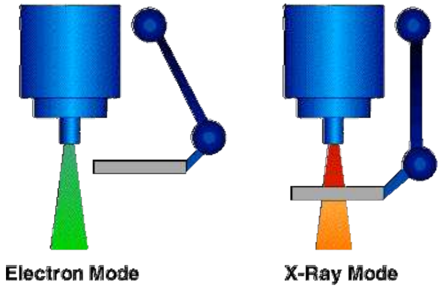


Therac-25:
Condición de
carrera

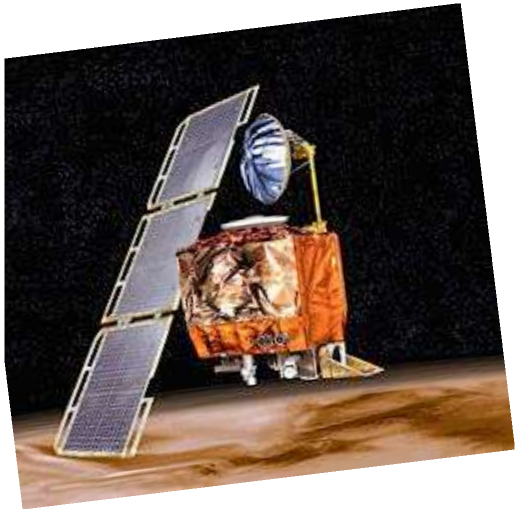
Bugs famosos

Pentium:
FDIV

Ariane 5:
64 bits fp
vs 16 bits int



Therac-25:
Condición de
carrera



Mars Climate
Orbiter:
Métrico vs Imperial

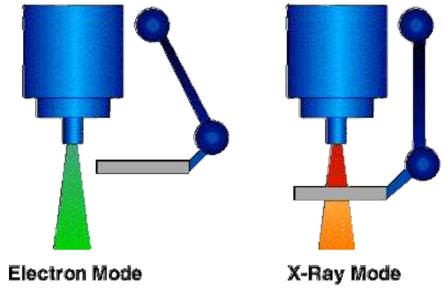
CONICET

UNC

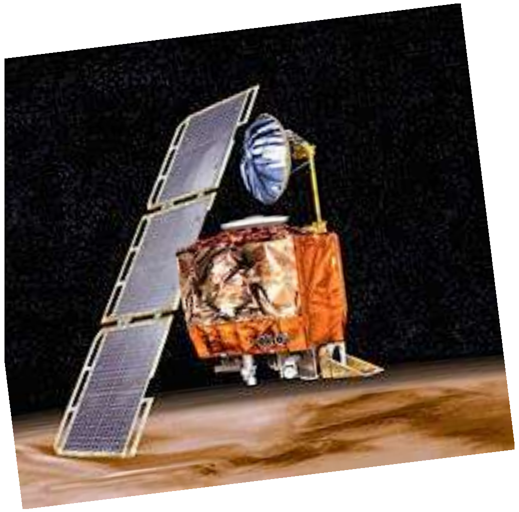
Bugs famosos

Pentium:
FDIV

Ariane 5:
64 bits fp
vs 16 bits int



Therac-25:
Condición de
carrera



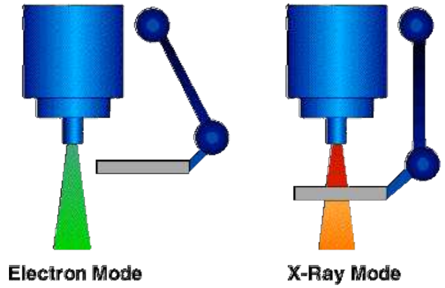
Mars Climate
Orbiter:
Métrico vs Imperial

Northeast blackout
en 2003:
Condición de
carrera

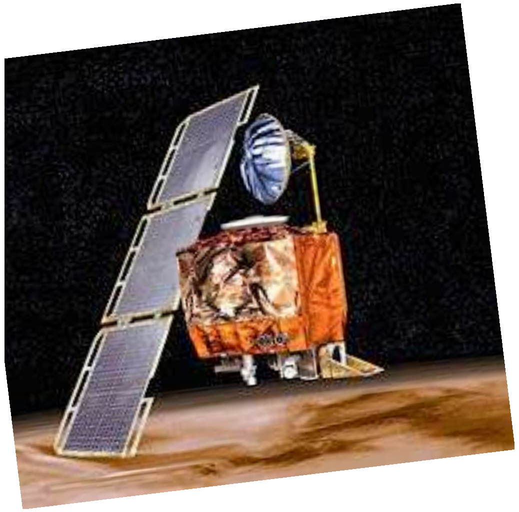
Bugs famosos

Pentium:
FDIV

Ariane 5:
64 bits fp
vs 16 bits int



Therac-25:
Condición de
carrera



Mars Climate
Orbiter:
Métrico vs Imperial

Northeast blackout
en 2003:
Condición de
carrera

Heartbleed:
Integridad/Confidencialidad

CONICET

UNC

Más *Bugs*

911 blackout:
MAX value
reached

Nest Thermostat:
Drenado de
batería

Nissan airbag:
Sensado
incorrecto

Lion Air 610
Boeing 737 MAX 8:
Sensado incorrecto

MELTDOWN SPECTRE

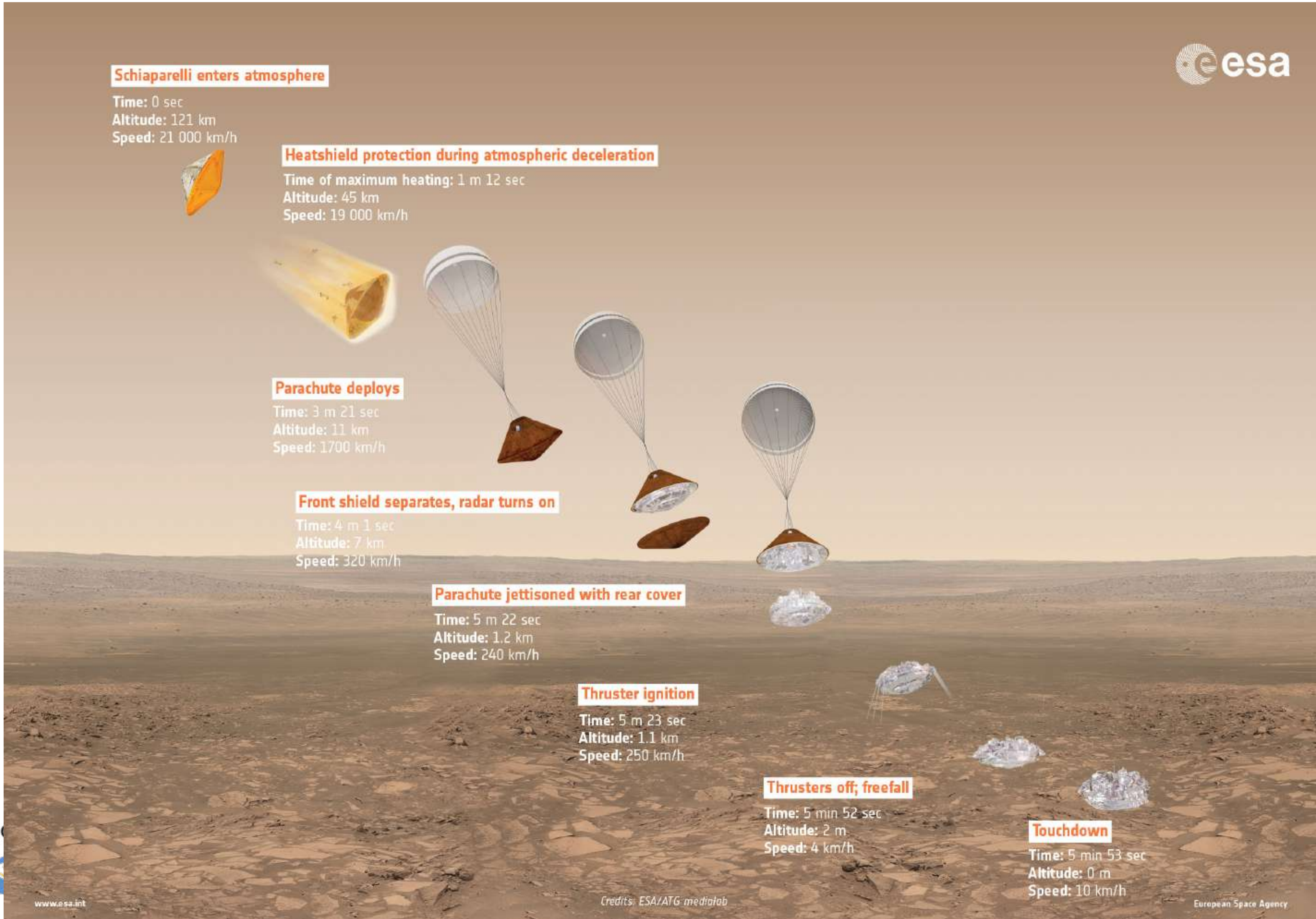
Intel: acceso a
memoria no
permitido

Tesla autopilot:
aprendizaje con
limitaciones??

CONICET

UNC

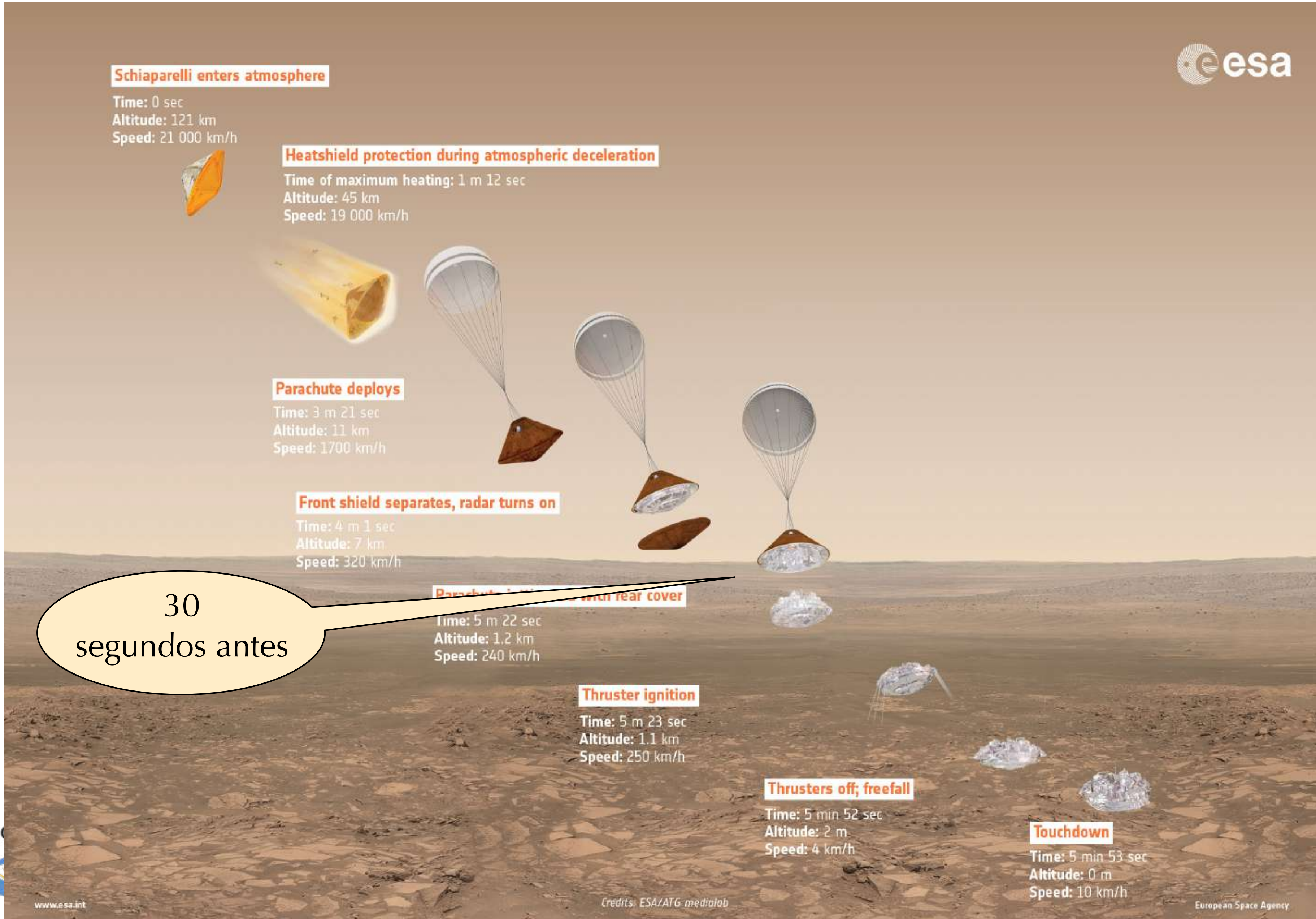
El problema de llamarlo *Bug*



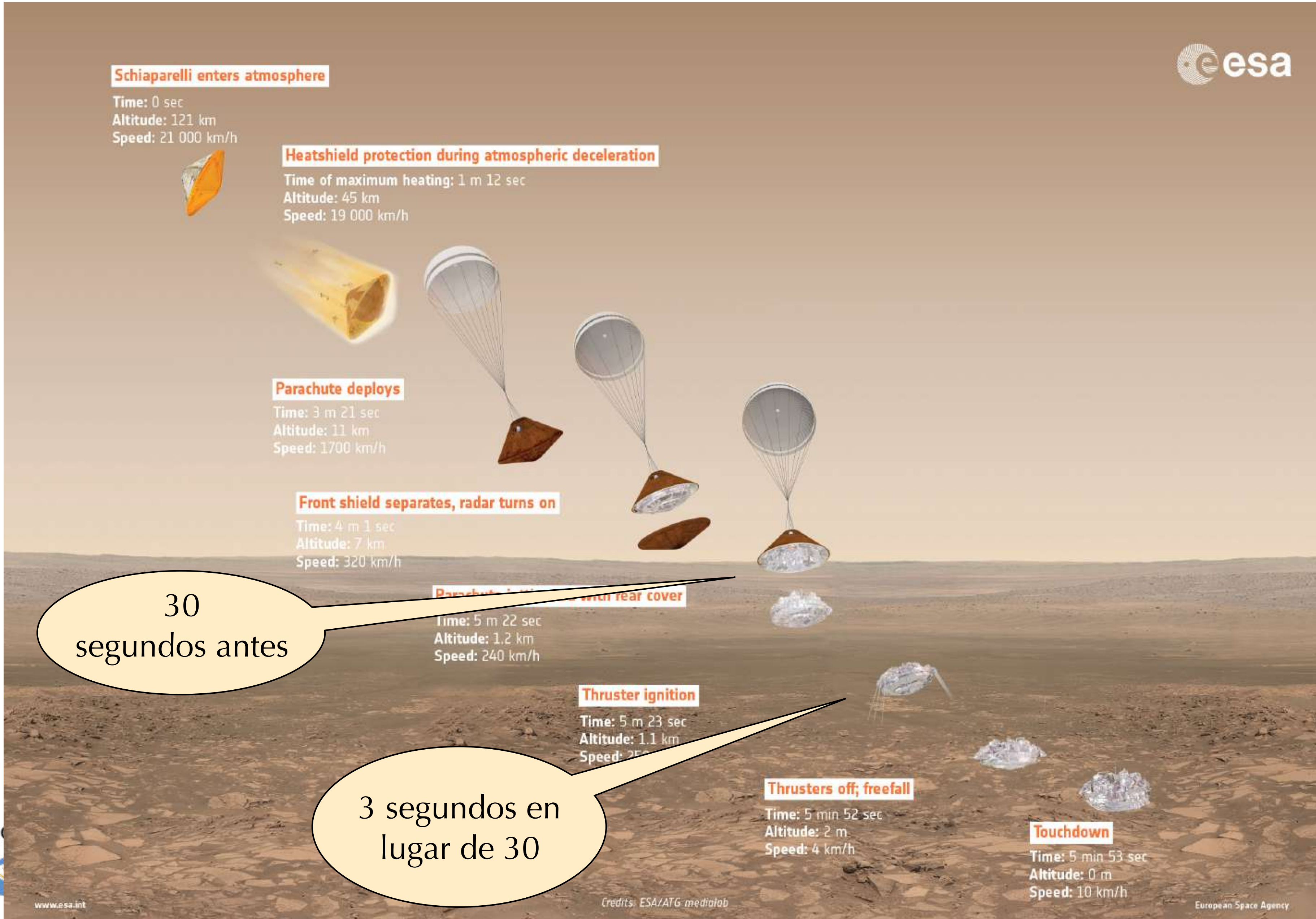
CONI

UNC

El problema de llamarlo *Bug*

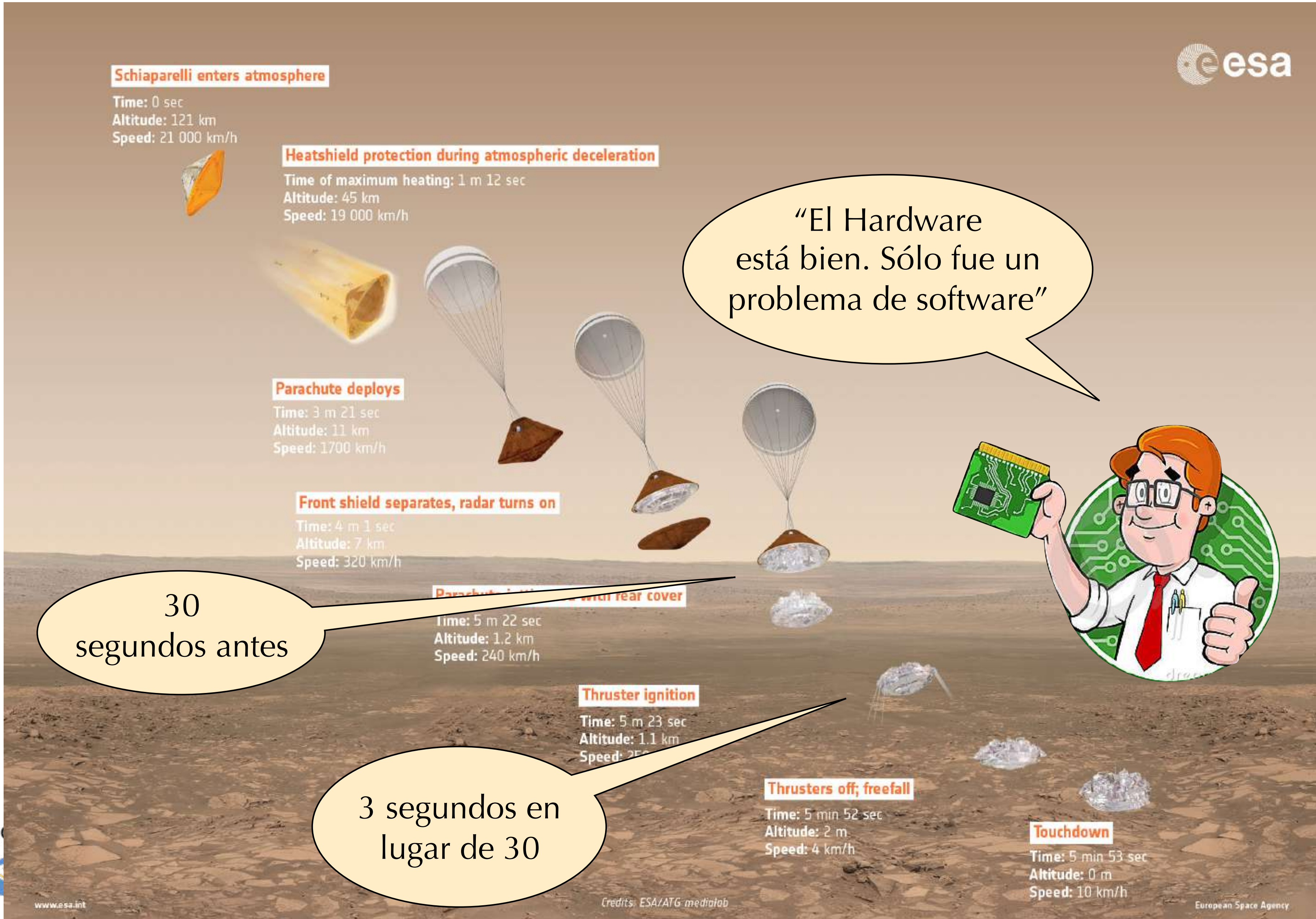


El problema de llamarlo *Bug*



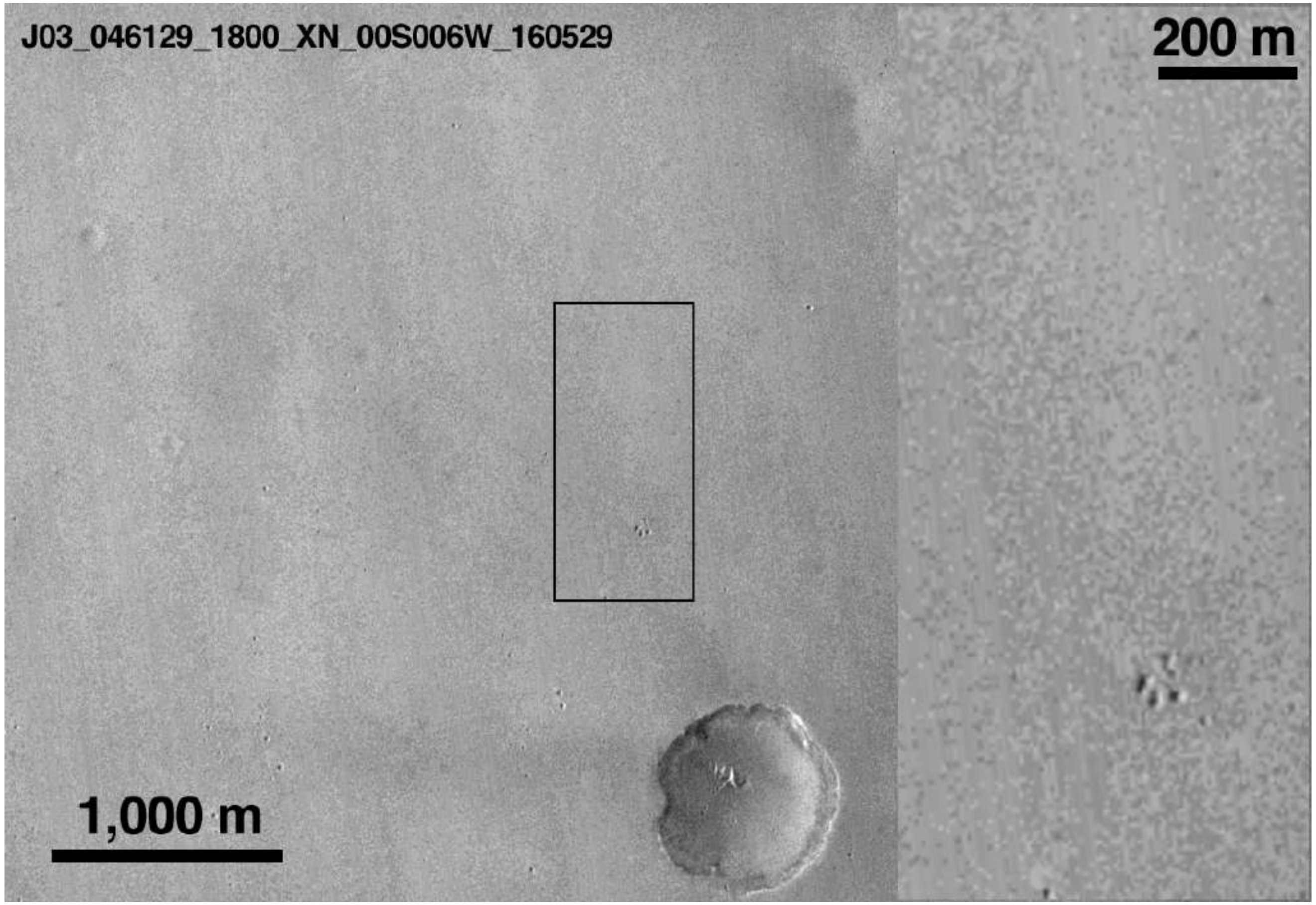
UNC

El problema de llamarlo *Bug*



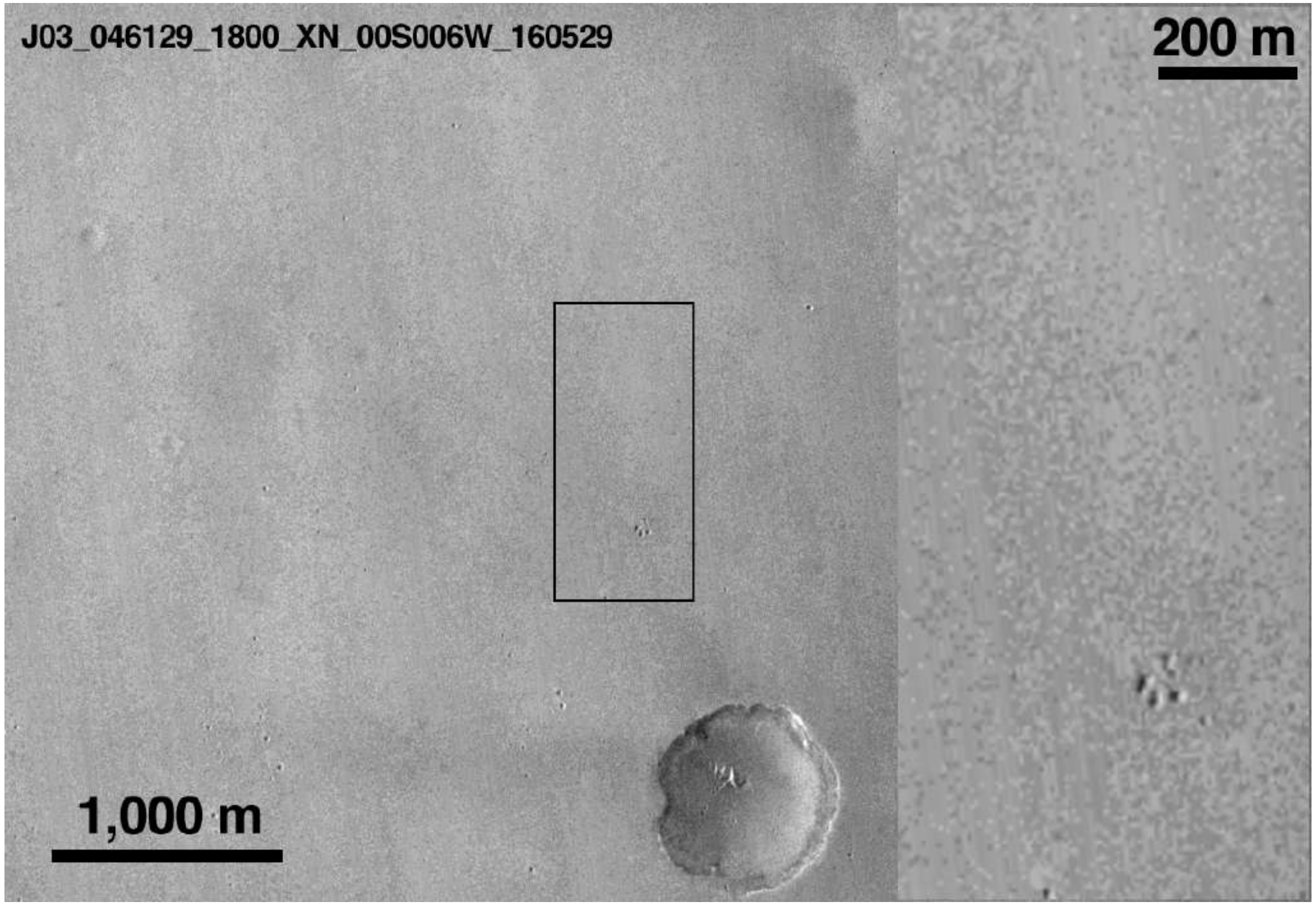
El problema de llamarlo *Bug*

El problema de llamarlo *Bug*



https://en.wikipedia.org/wiki/Schiaparelli_EDM_lander

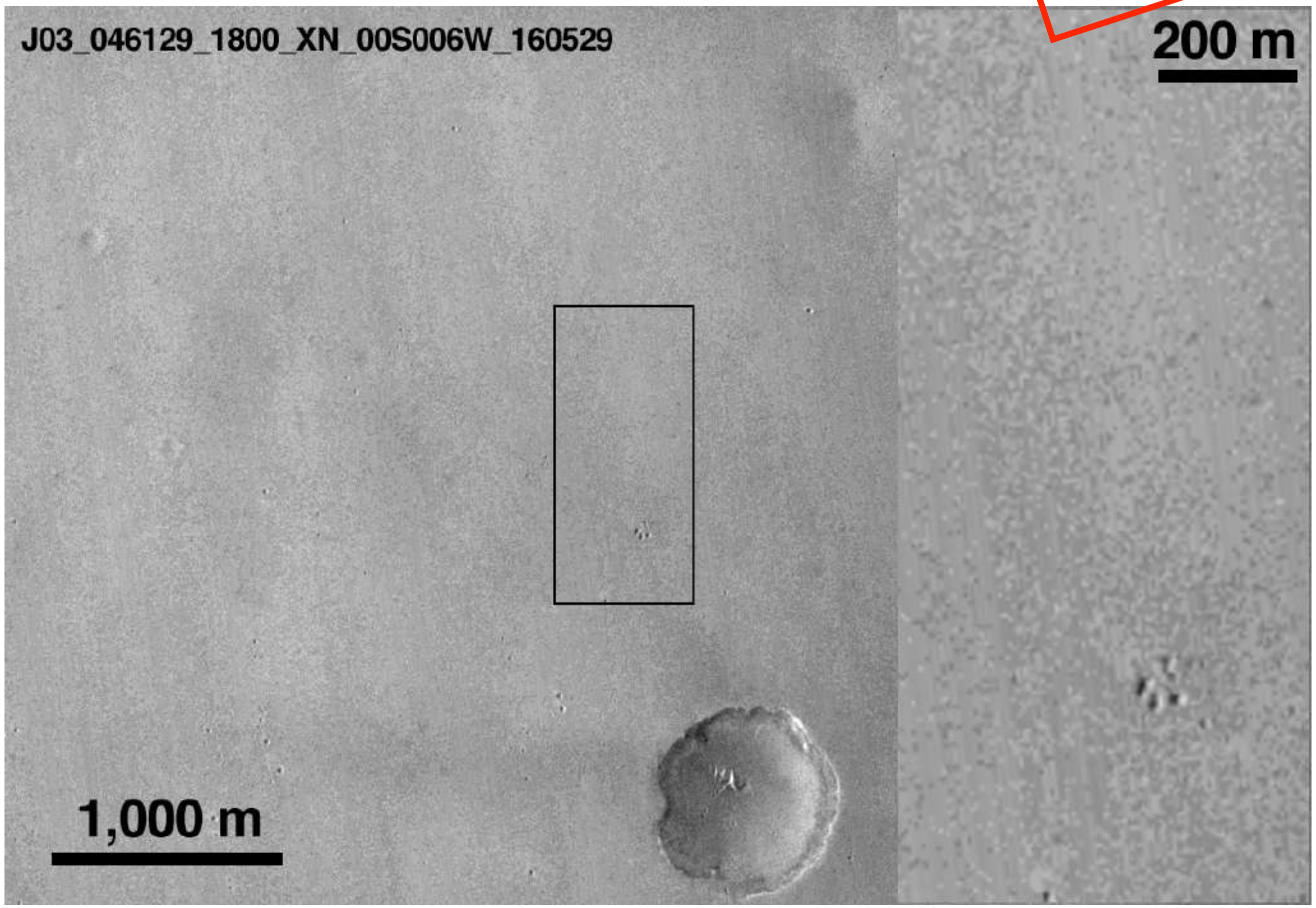
El problema de llamarlo *Bug*



https://en.wikipedia.org/wiki/Schiaparelli_EDM_lander

El problema de llamarlo Bug

MOCO!



https://en.wikipedia.org/wiki/Schiaparelli_EDM_lander

El problema de la corrección

$$\mathit{Sistema} \models \mathit{Propiedad}$$

El problema de la corrección

$$\color{red}{\mathit{Sistema}} \models \color{blue}{\mathit{Propiedad}}$$

Usualmente una abstracción que describe su comportamiento

El problema de la corrección

$$\textcolor{red}{\mathit{Sistema}} \models \textcolor{blue}{\mathit{Propiedad}}$$

Usualmente una abstracción que describe su comportamiento

Describe lo que se espera del sistema (el criterio de corrección)

Testing

$$\textcolor{red}{\mathit{Sistema}} \models \textcolor{blue}{\mathit{Propiedad}}$$

Usualmente una abstracción que describe su comportamiento

Describe lo que se espera del sistema (el criterio de corrección)

Testing

$$\textit{Sistema} \models \textit{Propiedad}$$

Sistema en ejecución

Describe lo que se
espera del sistema
(el criterio de corrección)

Testing

$$\textit{Sistema} \models \textit{Propiedad}$$

Sistema en ejecución

Test suite

Testing

$$\textit{Sistema} \models \textit{Propiedad}$$

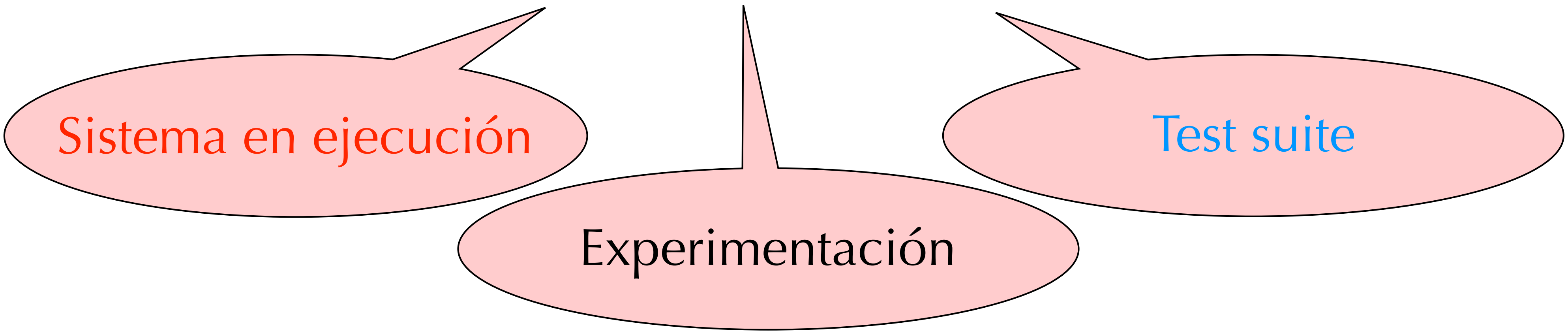
Sistema en ejecución

Experimentación

Test suite

Testing

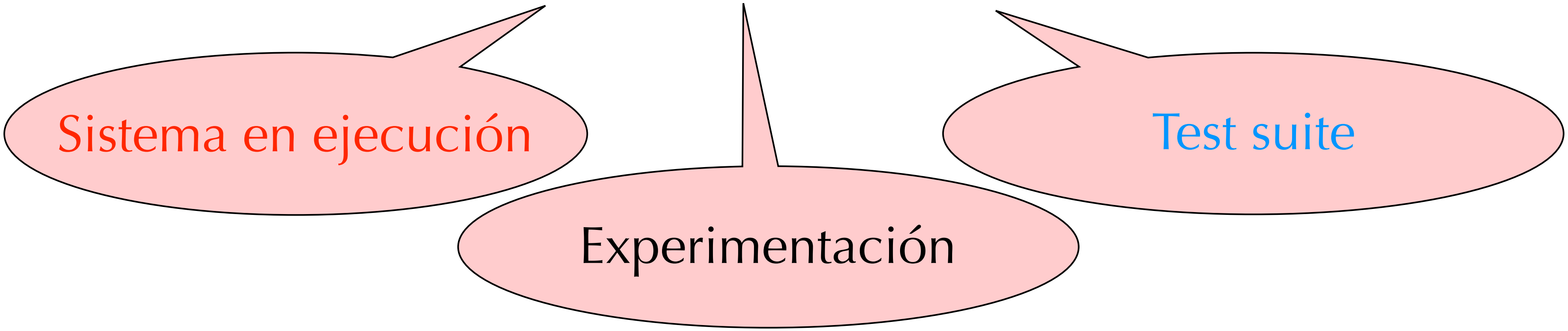
$$\textit{Sistema} \models \textit{Propiedad}$$



- Provee una serie de entradas al software, y estudia el comportamiento del mismo en esos casos.
- Llega tarde en el proceso de desarrollo
- Es muy incompleto (insuficiente como única forma de validación)

Testing

$$\textcolor{red}{\mathit{Sistema}} \vDash \textcolor{blue}{\mathit{Propiedad}}$$



- Provee una serie de entradas al software, y estudia el comportamiento del mismo en esos casos.
- Llega tarde en el proceso de desarrollo
- Es muy incompleto (insuficiente como única forma de validación)

Testing

$$\textit{Sistema} \models \textit{Propiedad}$$

Sistema en ejecución

Experimentación

Test suite

- Provee una serie de entradas al software, y estudia el comportamiento del mismo en esos casos.
- Llega tarde en el proceso de desarrollo
- Es muy incompleto (insuficiente como única forma de validación)

Verificación

Asercional (ej. Lógica de Hoare)

$$\mathit{Sistema} \vDash \mathit{Propiedad}$$

Asercional (ej. Lógica de Hoare)

$$\mathit{Sistema} \models \mathit{Propiedad}$$

$\{x = X \ \ \& \ \ y = Y\}$

```
aux := x;
x := y;
y := aux;
```

$\{x = Y \ \ \& \ \ y = X\}$

Asercional (ej. Lógica de Hoare)

Sistema $\models$ *Propiedad*

Programa

$\{x = X \;\; \& \;\; y = Y\}$

```
aux := x;
x := y;
y := aux;
```

$\{x = Y \;\; \& \;\; y = X\}$

Asercional (ej. Lógica de Hoare)

$\textit{Sistema} \models \textit{Propiedad}$

Programa

Pre / Post

$\{x = X \ \ \& \ \ y = Y\}$

```
aux := x;
x := y;
y := aux;
```

$\{x = Y \ \ \& \ \ y = X\}$

Asercional (ej. Lógica de Hoare)

$\textit{Sistema} \models \textit{Propiedad}$

Programa

Reglas de cálculo

Pre / Post

$\{x = X \;\;\&\;\; y = Y\}$

```
aux := x;
x := y;
y := aux;
```

$\{x = Y \;\;\&\;\; y = X\}$

Asercional (ej. Lógica de Hoare)

$\textcolor{red}{\textit{Sistema}} \models \textcolor{blue}{\textit{Propiedad}}$

Programa

Reglas de cálculo

Pre / Post

$\{x = X \ \ \& \ \ y = Y\}$

```
aux := x;
x  := y;
y  := aux;
```

$\{x = Y \ \ \& \ \ y = X\}$

- Solo programa secuenciales
- No es completamente automatizable
- Solo sistema funcional

CONICET

UNC

UNIVERSITÄT DES SAARLANDES

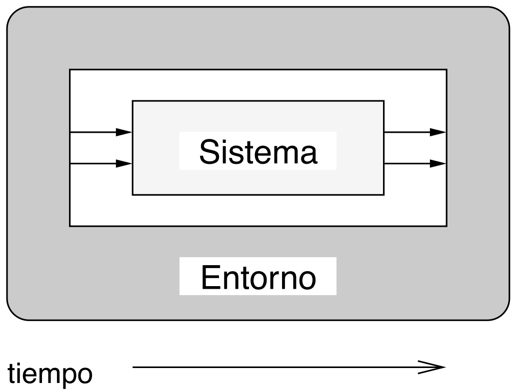
Asercional (ej. Lógica de Hoare)

$\mathit{Sistema} \models \mathit{Propiedad}$

Programa

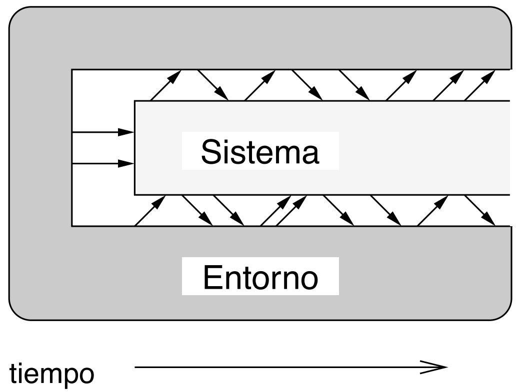
Reglas de cálculo

Pre / Post



Sistema funcional

vs.



Sistema reactivo

Asercional (ej. Lógica de Hoare)

Sistema $\models$ *Propiedad*

Programa

Reglas de cálculo

Pre / Post

$\{x = X \ \ \& \ \ y = Y\}$

```
aux := x;
x := y;
y := aux;
```

$\{x = Y \ \ \& \ \ y = X\}$

- Solo programa secuenciales
- No es completamente automatizable
- Solo sistema funcional

Asercional (ej. Lógica de Hoare)

$$\textit{Sistema} \models \textit{Propiedad}$$

Programa

Reglas de cálculo

Pre / Post

$\{x = X \;\; \& \;\; y = Y\}$

```
aux := x;
x  := y;
y  := aux;
```

$\{x = Y \;\; \& \;\; y = X\}$

- Solo programa secuenciales
- No es completamente automatizable
- Solo sistema funcional

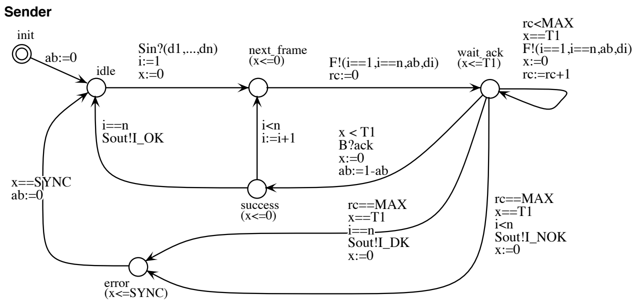
Model checking

$$\textit{Sistema} \models \textit{Propiedad}$$

Model checking

$$\textcolor{red}{\mathit{Sistema}} \models \textcolor{blue}{\mathit{Propiedad}}$$

Modelo del sistema

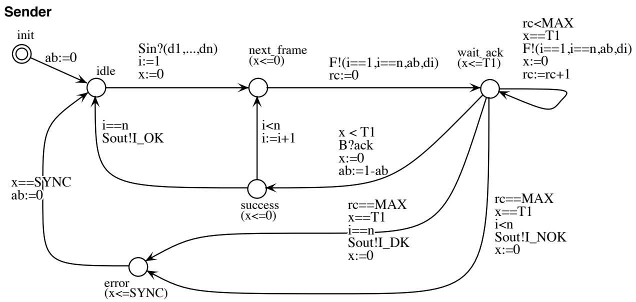


Model checking

$$\mathit{Sistema} \vDash \mathit{Propiedad}$$

Modelo del sistema

Lógica temporal



$$\Box\,(\,(\mathit{haz} = \text{on}) \Rightarrow (\mathit{filtro} = \text{activo})\,)$$

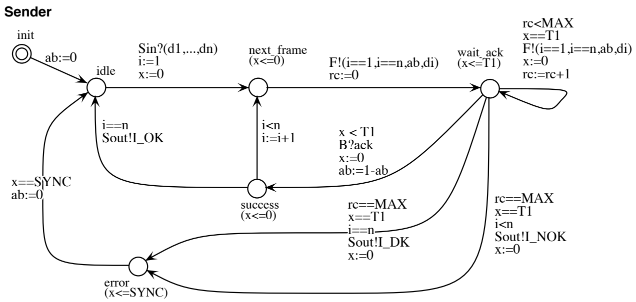
Model checking

$\textit{Sistema} \models \textit{Propiedad}$

Modelo del sistema

Algoritmo
basado en semántica

Lógica temporal



$\Box\,(\,(\textit{haz} = \text{on}) \Rightarrow (\textit{filtro} = \text{activo})\,)$

CONICET UNC UNIVERSITÄT DES SAARLANDES

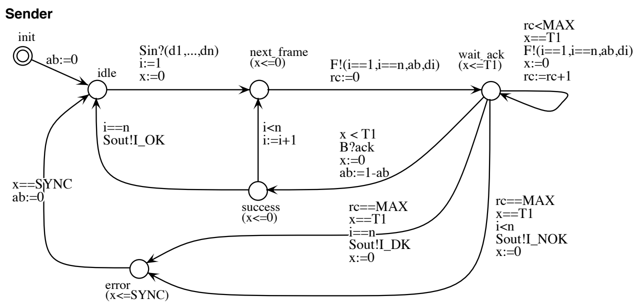
Model checking

$\textit{Sistema} \models \textit{Propiedad}$

Modelo del sistema

Algoritmo
basado en semántica

Lógica temporal



$\Box\,(\,(\textit{haz} = \text{on}) \Rightarrow (\textit{filtro} = \text{activo})\,)$

Dado un modelo de un sistema, verificar exhaustiva y automáticamente si éste satisface una especificación dada

Model checking

Pros:

- Generalidad
- Verificación parcial
- Acelera el proceso de diseño
- Fácil de usar
- Interés por parte de la industria
- Corrección asegurada matemáticamente

Contras:

- Se orienta al control (no a datos)
- Limitado por decibilidad / complejidad
- Verifica un modelo, no el sistema
- Requiere experiencia en modelado
- No es posible concluir generalizaciones
- (para dogmáticos) No da prueba explícita de las respuestas positivas

Concurrencia

- Programación de sistemas compuestos de varios procesos que se ejecutan de manera superpuesta en un período de tiempo e interactúan entre sí.

- Los programas concurrentes están compuestos por procesos (o threads, o componentes) que necesitan interactuar. Existen varios mecanismos de interacción entre procesos. Entre éstos se encuentran la memoria compartida y el pasaje de mensajes.

- Además, los programas concurrentes deben, en general, colaborar para llegar a un objetivo común, para lo cual la sincronización entre procesos es crucial.

Concurrencia: ejemplo

```
int y1 = 0 ;
int y2 = 0 ;
int in_critical = 0 ;
```

```
while true do
  y1 := y2 + 1 ;
  if
    [] ((y2 = 0) ∨ (y1 ≤ y2)) →
         in_critical ++ ;
         // región crítica
         in_critical -- ;
         y1 := 0
  fi
od
```

$\|$

```
while true do
  y2 := y1 + 1 ;
  if
    [] ((y1 = 0) ∨ (y2 < y1)) →
         in_critical ++ ;
         // región crítica
         in_critical -- ;
         y2 := 0
  fi
od
```

Concurrenc

El if es bloqueante:
espera hasta que alguna guarda
se haga verdadera

```
int y1 = 0 ;
int y2 = 0 ;
int in_critical = 0 ;
```

```
while true do                              ||   while true do
  y1 := y2 + 1 ;                           ||     y2 := y1 + 1 ;
  if                                       ||     if
     [] ((y2 = 0) ∨ (y1 ≤ y2)) →           ||        [] ((y1 = 0) ∨ (y2 < y1)) →
          in_critical ++ ;                 ||             in_critical ++ ;
          // región crítica                ||             // región crítica
          in_critical -- ;                 ||             in_critical -- ;
          y1 := 0                          ||             y2 := 0
  fi                                       ||     fi
od                                         ||   od
```

CONICET

UNC

Concurrencia: ejemplo

```
int y1 = 0 ;
int y2 = 0 ;
int in_critical = 0 ;
```

```
while true do
  y1 := y2 + 1 ;
  if
    [] ((y2 = 0) ∨ (y1 ≤ y2)) →
          in_critical ++ ;
          // región crítica
          in_critical -- ;
          y1 := 0
  fi
od
```

‖

```
while true do
  y2 := y1 + 1 ;
  if
    [] ((y1 = 0) ∨ (y2 < y1)) →
          in_critical ++ ;
          // región crítica
          in_critical -- ;
          y2 := 0
  fi
od
```

¿Qué hace este programa?

Concurrencia: ejemplo

```
int y1 = 0 ;
int y2 = 0 ;
int in_critical = 0 ;
```

```
while true do                            ||   while true do
  y1 := y2 + 1 ;                         ||     y2 := y1 + 1 ;
  if                                     ||     if
    [] ((y2 = 0) ∨ (y1 ≤ y2)) →          ||       [] ((y1 = 0) ∨ (y2 < y1)) →
          in_critical ++ ;               ||             in_critical ++ ;
          // región crítica              ||             // región crítica
          in_critical -- ;               ||             in_critical -- ;
          y1 := 0                        ||             y2 := 0
  fi                                     ||     fi
od                                       ||   od
```

Echar **MOCO** es fácil, encontrarlo es difícil

Para poder analizar el programa,
debemos comprender su semántica

CONICET

UNC

Estructura de Kripke

$$(S, s_0, \longrightarrow, L)$$

CONICET

UNC

Estructura de Kripke

$(S, s_0, \longrightarrow, L)$

Conjunto de estados

Estructura de Kripke

$$(S, s_0, \longrightarrow, L)$$

Estado inicial $s_0 \in S$

Conjunto de estados

Estructura de Kripke

$$(S, s_0, \longrightarrow, L)$$

Relación de transición

$\longrightarrow \subseteq S \times S$

Estado inicial $s_0 \in S$

Conjunto de estados

Estructura de Kripke

$(S, s_0, \longrightarrow, L)$

Función de etiquetado

$L : S \to \mathcal{P}(\mathsf{PA})$

Relación de transición

$\longrightarrow \subseteq S \times S$

Estado inicial $s_0 \in S$

Conjunto de estados

CONICET

UNC

Estructura de Kripke

$(S, s_0, \longrightarrow, L)$

$S = \{0,1\} \times \{0,1\}$

Función de etiquetado

$L : S \to \mathcal{P}(\mathsf{PA})$

Relación de transición

$\longrightarrow \subseteq S \times S$

Estado inicial $s_0 \in S$

Conjunto de estados

Estructura de Kripke

$$(S, s_0, \longrightarrow, L)$$

Función de etiquetado

$L : S \rightarrow \mathcal{P}(\mathsf{PA})$

Relación de transición

$\longrightarrow\ \subseteq S \times S$

Estado inicial $s_0 \in S$

Conjunto de estados

$S = \{0,1\} \times \{0,1\}$

$s_0 = (0,0)$

Estructura de Kripke

$(S, s_0, \longrightarrow, L)$

Función de etiquetado

$L : S \to \mathcal{P}(\mathsf{PA})$

Relación de transición

$\longrightarrow \subseteq S \times S$

Estado inicial $s_0 \in S$

Conjunto de estados

$S = \{0,1\} \times \{0,1\}$

$s_0 = (0,0)$

$(x,y) \longrightarrow (x,0)$ (escribir 0 en y)

$(x,y) \longrightarrow (x,1)$ (escribir 1 en y)

$(x,y) \longrightarrow (y,x)$ (swap)

Estructura de Kripke

$(S, s_0, \longrightarrow, L)$

Función de etiquetado
$L : S \to \mathcal{P}(\mathsf{PA})$

Relación de transición
$\longrightarrow \subseteq S \times S$

Estado inicial $s_0 \in S$

Conjunto de estados

$S = \{0,1\} \times \{0,1\}$

$s_0 = (0,0)$

$(x,y) \longrightarrow (x,0)$ (escribir 0 en y)

$(x,y) \longrightarrow (x,1)$ (escribir 1 en y)

$(x,y) \longrightarrow (y,x)$ (swap)

$L(0,0) = \{(x=0),(x\neq 1),(y=0),(y\neq 1),(x=y)\}$

$L(0,1) = \{(x=0),(x\neq 1),(y=1),(y\neq 0),(x\neq y)\}$

...etc.

CONICET

UNC

Estructura de Kripke

$(S, s_0, \longrightarrow, L)$

Función de etiquetado
$L : S \to \mathcal{P}(\text{PA})$

Relación de transición
$\longrightarrow \subseteq S \times S$

Estado inicial $s_0 \in S$

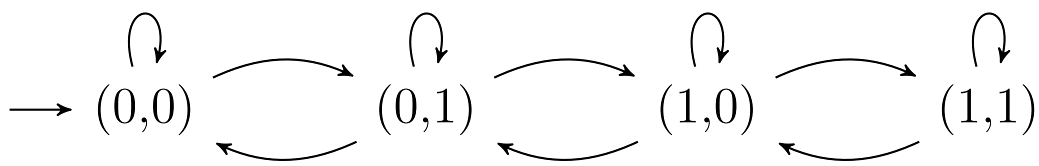
Conjunto de estados

$S = \{0,1\} \times \{0,1\}$

$s_0 = (0,0)$

$(x,y) \longrightarrow (x,0)$	(escribir 0 en y)
$(x,y) \longrightarrow (x,1)$	(escribir 1 en y)
$(x,y) \longrightarrow (y,x)$	(swap)

$L(0,0) = \{(x=0), (x \neq 1), (y=0), (y \neq 1), (x=y)\}$
$L(0,1) = \{(x=0), (x \neq 1), (y=1), (y \neq 0), (x \neq y)\}$
...etc.



Un simple lenguaje concurrente

$\texttt{x := } \mathit{expr}$	asignación
$P_1 \texttt{ ; } P_2$	secuencia
$\texttt{if } b_0 \rightarrow P_0 \; [\!] \ldots [\!] \; b_n \rightarrow P_n \texttt{ fi}$	condicional bloqueante
$\texttt{while } b \texttt{ do } P \texttt{ od}$	iteración
$P_1 \parallel P_2$	composición paralela

CONICET

UNC

Un simple lenguaje concurrente

expresión aritmética

$\texttt{x := } \mathit{expr}$ — asignación

$P_1 \; ; \; P_2$ — secuencia

$\texttt{if } b_0 \rightarrow P_0 \;[\!]\; \ldots \;[\!]\; b_n \rightarrow P_n \texttt{ fi}$ — condicional bloqueante

$\texttt{while } b \texttt{ do } P \texttt{ od}$ — iteración

$P_1 \mid\mid P_2$ — composición paralela

CONICET

UNC

Un simple lenguaje concurrente

expresión aritmética

$\texttt{x := } \mathit{expr}$ — asignación

expresión booleana

$P_1 \; ; \; P_2$ — secuencia

$\texttt{if } b_0 \rightarrow P_0 \; [\!] \ldots [\!] \; b_n \rightarrow P_n \texttt{ fi}$ — condicional bloqueante

$\texttt{while } b \texttt{ do } P \texttt{ od}$ — iteración

$P_1 \parallel P_2$ — composición paralela

CONICET

UNC

Un simple lenguaje concurrente

expresión aritmética

`x :=` $expr$	asignación
P_1 ; P_2	secuencia
`if` $b_0 \rightarrow P_0 \; [\!] \ldots [\!] \; b_n \rightarrow P_n$ `fi`	condicional bloqueante
`while` b `do` P `od`	iteración
$P_1 \; \|\| \; P_2$	composición paralela

expresión booleana

Para definir la semántica, consideramos también el indicador de terminación $\surd$

Semántica

Semántica

$S = \mathsf{Lang} \times \mathcal{M}$

Semántica

$S = \mathsf{Lang} \times \mathcal{M}$

Conjunto de memorias: $\mu : \mathsf{Var} \to \mathsf{Values}$

Conjunto de programas del lenguaje

Semántica

$$S = \mathsf{Lang} \times \mathcal{M}$$

Conjunto de memorias: $\mu : \mathsf{Var} \to \mathsf{Values}$

Conjunto de programas del lenguaje

$$s_0 = \langle P, \mu_{\mathsf{init}} \rangle$$

CONICET

UNC

Semántica

$$S = \mathsf{Lang} \times \mathcal{M}$$

Conjunto de memorias: $\mu : \mathsf{Var} \to \mathsf{Values}$

Conjunto de programas del lenguaje

$$s_0 = \langle P, \mu_{\mathsf{init}} \rangle$$

Memoria inicializada con valores por defecto

Programa de interés

Semántica

$S = \mathsf{Lang} \times \mathcal{M}$

Restringido a los alcanzables desde s_0

Conjunto de memorias: $\mu : \mathsf{Var} \to \mathsf{Values}$

Conjunto de programas del lenguaje

$s_0 = \langle P, \mu_{\mathsf{init}} \rangle$

Memoria inicializada con valores por defecto

Programa de interés

Semántica

$S = \mathsf{Lang} \times \mathcal{M}$

Restringido a los alcanzables desde s_0

Conjunto de memorias: $\mu : \mathsf{Var} \to \mathsf{Values}$

Conjunto de programas del lenguaje

$s_0 = \langle P, \mu_{\mathsf{init}} \rangle$

Memoria inicializada con valores por defecto

Programa de interés

$$L(\langle P, \mu \rangle) = \{a \in \mathsf{PA} \mid \mu(a) \text{ es verdadera}\}$$

CONICET

UNC

Semántica

Restringido a
los alcanzables
desde s_0

$$S = \mathsf{Lang} \times \mathcal{M}$$

Conjunto de memorias: $\mu : \mathsf{Var} \to \mathsf{Values}$

Conjunto de programas del lenguaje

$$s_0 = \langle P, \mu_{\mathsf{init}} \rangle$$

Memoria inicializada con valores por defecto

Programa de interés

$$L(\langle P, \mu \rangle) = \{a \in \mathsf{PA} \mid \mu(a) \text{ es verdadera}\}$$

Serán expresiones
como “`x=0`”, “`x+y≤10`”,
“`z*x≥y^2`”, etc.

Semántica

$$\langle x \texttt{ := } e, \mu \rangle \longrightarrow \langle \surd, \mu[x \mapsto \mu(e)] \rangle$$

Semántica

$$\langle x := e, \mu \rangle \longrightarrow \langle \surd, \mu[x \mapsto \mu(e)] \rangle$$

$$\downarrow$$

$$\langle y := 1, (0,0) \rangle$$

para simplificar suponemos $(\mu(x),\mu(y))$

Semántica

$$\langle x := e, \mu \rangle \longrightarrow \langle \surd, \mu[x \mapsto \mu(e)] \rangle$$

$$\begin{array}{c} \downarrow \\ \langle y := 1, (0,0) \rangle \\ \Big\downarrow \\ \langle \surd, (0,1) \rangle \end{array}$$

Semántica

$$\langle x \ := e, \mu\rangle \longrightarrow \langle \surd, \mu[x \mapsto \mu(e)]\rangle$$

$$\begin{tikzcd}
{} \arrow[d] & {} \arrow[d] \\
\langle y \ := 1, (0,0)\rangle \arrow[d] & \langle x \ := x + 1, (0,0)\rangle \\
\langle \surd, (0,1)\rangle &
\end{tikzcd}$$

CONICET

UNC

Semántica

$$\langle x := e, \mu \rangle \longrightarrow \langle \surd, \mu[x \mapsto \mu(e)] \rangle$$

$$\begin{tikzcd} {} \arrow[d] & {} \arrow[d] \\ \langle y := 1, (0,0) \rangle \arrow[d] & \langle x := x + 1, (0,0) \rangle \arrow[d] \\ \langle \surd, (0,1) \rangle & \langle \surd, (1,0) \rangle \end{tikzcd}$$

Semántica

$$\langle x := e, \mu\rangle \longrightarrow \langle \surd, \mu[x \mapsto \mu(e)]\rangle$$

$$\begin{array}{ccc} \downarrow & \downarrow & \downarrow \\ \langle y := 1, (0,0)\rangle & \langle x := x+1, (0,0)\rangle & \langle x := x-1, (1,0)\rangle \\ \downarrow & \downarrow & \\ \langle \surd, (0,1)\rangle & \langle \surd, (1,0)\rangle & \end{array}$$

Semántica

$$\langle x \;\texttt{:=}\; e, \mu\rangle \longrightarrow \langle \surd, \mu[x \mapsto \mu(e)]\rangle$$

$$\begin{array}{ccc}
\downarrow & \downarrow & \downarrow \\
\langle y \;\texttt{:=}\; 1, (0,0)\rangle & \langle x \;\texttt{:=}\; x+1, (0,0)\rangle & \langle x \;\texttt{:=}\; x-1, (1,0)\rangle \\
\Big\downarrow & \Big\downarrow & \Big\downarrow \\
\langle \surd, (0,1)\rangle & \langle \surd, (1,0)\rangle & \langle \surd, (0,0)\rangle
\end{array}$$

UNC

Semántica

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle \quad P_1' \neq \surd}{\langle P_1 \ ;\ P_2, \mu\rangle \longrightarrow \langle P_1' \ ;\ P_2, \mu'\rangle}$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle \surd, \mu'\rangle}{\langle P_1 \ ;\ P_2, \mu\rangle \longrightarrow \langle P_2, \mu'\rangle}$$

Semántica

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle \quad P_1' \neq \surd}{\langle P_1\ ;\ P_2, \mu\rangle \longrightarrow \langle P_1'\ ;\ P_2, \mu'\rangle}$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle \surd, \mu'\rangle}{\langle P_1\ ;\ P_2, \mu\rangle \longrightarrow \langle P_2, \mu'\rangle}$$

$$\begin{array}{ll} P_{\mathsf{s}}: & y := 1\ ; \\ & y := 0 \end{array}$$

Semántica

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle \quad P_1' \neq \surd}{\langle P_1 \ ; \ P_2, \mu\rangle \longrightarrow \langle P_1' \ ; \ P_2, \mu'\rangle}$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle \surd, \mu'\rangle}{\langle P_1 \ ; \ P_2, \mu\rangle \longrightarrow \langle P_2, \mu'\rangle}$$

$$P_{\mathsf{s}}: \quad y := 1 \ ; \\ \quad y := 0$$

$$\downarrow$$

$$\langle P_{\mathsf{s}}, (0,0)\rangle$$

CONICET

UNC

Semántica

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle \quad P_1' \neq \surd}{\langle P_1 \ ; \ P_2, \mu\rangle \longrightarrow \langle P_1' \ ; \ P_2, \mu'\rangle}$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle \surd, \mu'\rangle}{\langle P_1 \ ; \ P_2, \mu\rangle \longrightarrow \langle P_2, \mu'\rangle}$$

$$\Downarrow$$

$$\frac{\langle y \ := 1, (0,0)\rangle \longrightarrow \langle \surd, (0,1)\rangle}{\langle y \ := 1 \ ; \ y \ := 0, (0,0)\rangle \longrightarrow \langle y \ := 0, (0,1)\rangle}$$

$$\begin{array}{ll} P_{\mathsf{s}} : & y \ := 1 \ ; \\ & y \ := 0 \end{array}$$

$$\downarrow$$

$$\langle P_{\mathsf{s}}, (0,0)\rangle$$

Semántica

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle \quad P_1' \neq \surd}{\langle P_1 \ ; \ P_2, \mu\rangle \longrightarrow \langle P_1' \ ; \ P_2, \mu'\rangle}$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle \surd, \mu'\rangle}{\langle P_1 \ ; \ P_2, \mu\rangle \longrightarrow \langle P_2, \mu'\rangle}$$

$$\Downarrow$$

$$\frac{\langle y := 1, (0,0)\rangle \longrightarrow \langle \surd, (0,1)\rangle}{\langle y := 1 \ ; \ y := 0, (0,0)\rangle \longrightarrow \langle y := 0, (0,1)\rangle}$$

$$\begin{aligned} P_{\mathsf{s}} : \quad & y := 1 \ ; \\ & y := 0 \end{aligned}$$

$$\begin{array}{c} \downarrow \\ \langle P_{\mathsf{s}}, (0,0)\rangle \\ \downarrow \\ \langle y := 0, (0,1)\rangle \end{array}$$

CONICET

UNC

Semántica

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle \quad P_1' \neq \surd}{\langle P_1 \ ; \ P_2, \mu\rangle \longrightarrow \langle P_1' \ ; \ P_2, \mu'\rangle}$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle \surd, \mu'\rangle}{\langle P_1 \ ; \ P_2, \mu\rangle \longrightarrow \langle P_2, \mu'\rangle}$$

$$\Downarrow$$

$$\frac{\langle y := 1, (0,0)\rangle \longrightarrow \langle \surd, (0,1)\rangle}{\langle y := 1 \ ; \ y := 0, (0,0)\rangle \longrightarrow \langle y := 0, (0,1)\rangle}$$

$$P_{\mathsf{s}} : \quad \begin{array}{l} y := 1 \ ; \\ y := 0 \end{array}$$

$$\begin{array}{c} \downarrow \\ \langle P_{\mathsf{s}}, (0,0)\rangle \\ \downarrow \\ \langle y := 0, (0,1)\rangle \\ \downarrow \\ \langle \surd, (0,1)\rangle \end{array}$$

Semántica

$$\frac{\langle P_j, \mu\rangle \longrightarrow \langle P_j', \mu'\rangle \qquad \mu(b_j) \text{ holds}}{\langle \texttt{if } b_0 \to P_0 \ [\!]\ \ldots\ [\!]\ b_n \to P_n \texttt{ fi}, \mu\rangle \longrightarrow \langle P_j', \mu'\rangle}$$

CONICET

UNC

Semántica

$$\frac{\langle P_j, \mu\rangle \longrightarrow \langle P'_j, \mu'\rangle \qquad \mu(b_j) \text{ holds}}{\langle \texttt{if}\ b_0 \to P_0\ [\!]\ \ldots\ [\!]\ b_n \to P_n\ \texttt{fi}, \mu\rangle \longrightarrow \langle P'_j, \mu'\rangle}$$

$$
\begin{array}{ll}
P_{\mathsf{c}}: & \texttt{if} \\
& \quad [\!]\ (x > 0) \wedge (y = 0) \to \\
& \qquad x \ \texttt{:=}\ x - 1 \\
& \quad [\!]\ (x < 2) \wedge (y = 0) \to \\
& \qquad x \ \texttt{:=}\ x + 1 \\
& \texttt{fi}
\end{array}
$$

Semántica

$$\frac{\langle P_j, \mu\rangle \longrightarrow \langle P'_j, \mu'\rangle \qquad \mu(b_j) \text{ holds}}{\langle \texttt{if } b_0 \to P_0 \ [\!] \ldots [\!] \ b_n \to P_n \texttt{ fi}, \mu\rangle \longrightarrow \langle P'_j, \mu'\rangle}$$

P_{c} : `if`

$\quad [\!]\ (x > 0) \wedge (y = 0) \to$

$\qquad x \texttt{ := } x - 1$

$\quad [\!]\ (x < 2) \wedge (y = 0) \to$

$\qquad x \texttt{ := } x + 1$

`fi`

$$\downarrow$$
$$\langle P_{\mathsf{c}}, (0, 0)\rangle$$

Semántica

$$\frac{\langle P_j, \mu\rangle \longrightarrow \langle P_j', \mu'\rangle \qquad \mu(b_j) \text{ holds}}{\langle \texttt{if } b_0 \to P_0 \;[\!]\; \ldots \;[\!]\; b_n \to P_n \texttt{ fi}, \mu\rangle \longrightarrow \langle P_j', \mu'\rangle}$$

$P_{\mathsf{c}}:$ `if`

$\quad [\!]\ (x > 0) \wedge (y = 0) \to$

$\qquad x \mathrel{\texttt{:=}} x - 1$

$\quad [\!]\ (x < 2) \wedge (y = 0) \to$

$\qquad x \mathrel{\texttt{:=}} x + 1$

`fi`

$$\begin{array}{c} \downarrow \\ \langle P_{\mathsf{c}}, (0,0)\rangle \\ \downarrow \\ \langle \surd, (1,0)\rangle \end{array}$$

Semántica

$$\frac{\langle P_j, \mu\rangle \longrightarrow \langle P_j', \mu'\rangle \qquad \mu(b_j) \text{ holds}}{\langle \texttt{if } b_0 \rightarrow P_0 \ [\!]\ \ldots\ [\!]\ b_n \rightarrow P_n \texttt{ fi}, \mu\rangle \longrightarrow \langle P_j', \mu'\rangle}$$

$$
\begin{array}{lll}
P_{\mathsf{c}}: & \texttt{if} & \\
& \quad [\!]\ (x > 0) \wedge (y = 0) \rightarrow & \\
& \qquad\quad x \ \texttt{:=}\ x - 1 & \\
& \quad [\!]\ (x < 2) \wedge (y = 0) \rightarrow & \\
& \qquad\quad x \ \texttt{:=}\ x + 1 & \\
& \texttt{fi} &
\end{array}
$$

$$
\begin{array}{ccc}
\downarrow & \qquad & \downarrow \\
\langle P_{\mathsf{c}}, (0,0)\rangle & & \langle P_{\mathsf{c}}, (2,0)\rangle \\
\downarrow & & \\
\langle \surd, (1,0)\rangle & &
\end{array}
$$

Semántica

$$\frac{\langle P_j, \mu\rangle \longrightarrow \langle P'_j, \mu'\rangle \qquad \mu(b_j) \text{ holds}}{\langle \texttt{if } b_0 \to P_0 \ [\!]\ \ldots\ [\!]\ b_n \to P_n \texttt{ fi}, \mu\rangle \longrightarrow \langle P'_j, \mu'\rangle}$$

$$\begin{array}{ll} P_{\mathsf{c}}: & \texttt{if} \\ & \quad [\!]\ (x > 0) \wedge (y = 0) \to \\ & \qquad x \texttt{ := } x - 1 \\ & \quad [\!]\ (x < 2) \wedge (y = 0) \to \\ & \qquad x \texttt{ := } x + 1 \\ & \texttt{fi} \end{array}$$

$$\begin{array}{ccc} \downarrow & \qquad & \downarrow \\ \langle P_{\mathsf{c}}, (0,0)\rangle & & \langle P_{\mathsf{c}}, (2,0)\rangle \\ \downarrow & & \downarrow \\ \langle \surd, (1,0)\rangle & & \langle \surd, (1,0)\rangle \end{array}$$

Semántica

$$\frac{\langle P_j, \mu\rangle \longrightarrow \langle P'_j, \mu'\rangle \qquad \mu(b_j) \text{ holds}}{\langle \texttt{if } b_0 \rightarrow P_0 \;[\!]\; \ldots \;[\!]\; b_n \rightarrow P_n \texttt{ fi}, \mu\rangle \longrightarrow \langle P'_j, \mu'\rangle}$$

P_{c} : `if`

$[\!]\ (x > 0) \wedge (y = 0) \rightarrow$

$\quad x := x - 1$

$[\!]\ (x < 2) \wedge (y = 0) \rightarrow$

$\quad x := x + 1$

`fi`

$$\begin{array}{ccc} \downarrow & & \downarrow \\ \langle P_{\mathsf{c}}, (0,0)\rangle & & \langle P_{\mathsf{c}}, (2,0)\rangle \\ \downarrow & & \downarrow \\ \langle \surd, (1,0)\rangle & & \langle \surd, (1,0)\rangle \end{array}$$

$$\begin{array}{c} \downarrow \\ \langle P_{\mathsf{c}}, (1,0)\rangle \end{array}$$

Semántica

$$\frac{\langle P_j, \mu\rangle \longrightarrow \langle P'_j, \mu'\rangle \qquad \mu(b_j) \text{ holds}}{\langle \texttt{if } b_0 \rightarrow P_0 \ [\!]\ \ldots\ [\!]\ b_n \rightarrow P_n \texttt{ fi}, \mu\rangle \longrightarrow \langle P'_j, \mu'\rangle}$$

$$\begin{array}{ll}
P_{\mathsf{c}}: & \texttt{if} \\
& \quad [\!]\ (x > 0) \wedge (y = 0) \rightarrow \\
& \qquad\quad x \ \texttt{:=}\ x - 1 \\
& \quad [\!]\ (x < 2) \wedge (y = 0) \rightarrow \\
& \qquad\quad x \ \texttt{:=}\ x + 1 \\
& \texttt{fi}
\end{array}$$

$$\begin{tikzcd}
{} \arrow[d] & {} \arrow[d] \\
\langle P_{\mathsf{c}}, (0,0)\rangle \arrow[d] & \langle P_{\mathsf{c}}, (2,0)\rangle \arrow[d] \\
\langle \surd, (1,0)\rangle & \langle \surd, (1,0)\rangle
\end{tikzcd}$$

$$\begin{tikzcd}
& {} \arrow[d] & \\
& \langle P_{\mathsf{c}}, (1,0)\rangle \arrow[dl] \arrow[dr] & \\
\langle \surd, (0,0)\rangle & & \langle \surd, (2,0)\rangle
\end{tikzcd}$$

Semántica

$$\frac{\langle P_j, \mu\rangle \longrightarrow \langle P_j', \mu'\rangle \qquad \mu(b_j) \text{ holds}}{\langle \texttt{if } b_0 \to P_0 \;[\!]\; \ldots \;[\!]\; b_n \to P_n \texttt{ fi}, \mu\rangle \longrightarrow \langle P_j', \mu'\rangle}$$

$P_{\mathsf{c}}:$ `if`
$\quad [\!]\ (x > 0) \wedge (y = 0) \to$
$\qquad x \;\texttt{:=}\; x - 1$
$\quad [\!]\ (x < 2) \wedge (y = 0) \to$
$\qquad x \;\texttt{:=}\; x + 1$
`fi`

$$\begin{tikzcd} {} \arrow[d] & {} \arrow[d] \\ \langle P_{\mathsf{c}}, (0,0)\rangle \arrow[d] & \langle P_{\mathsf{c}}, (2,0)\rangle \arrow[d] \\ \langle \surd, (1,0)\rangle & \langle \surd, (1,0)\rangle \end{tikzcd}$$

$$\begin{tikzcd} & {} \arrow[d] & & {} \arrow[d] \\ & \langle P_{\mathsf{c}}, (1,0)\rangle \arrow[dl] \arrow[dr] & & \langle P_{\mathsf{c}}, (1,1)\rangle \\ \langle \surd, (0,0)\rangle & & \langle \surd, (2,0)\rangle & \end{tikzcd}$$

Semántica

$$\frac{\langle P; \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle \qquad \mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle}$$

$$\frac{\neg\mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle \surd, \mu\rangle}$$

CONICET

UNC

Semántica

$$\frac{\langle P; \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle \qquad \mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle}$$

$$\frac{\neg\mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle \surd, \mu\rangle}$$

$$
\begin{array}{ll}
P_{\mathsf{w}}: & \texttt{while}\ x < 2\ \ \texttt{do} \\
& \quad \texttt{if} \\
& \quad\quad [\!]\ (x > 0) \wedge (y = 0) \rightarrow \\
& \quad\quad\quad\quad x \ \texttt{:=}\ x - 1 \\
& \quad\quad [\!]\ (x < 2) \wedge (y = 0) \rightarrow \\
& \quad\quad\quad\quad x \ \texttt{:=}\ x + 1 \\
& \quad \texttt{fi} \\
& \texttt{od}
\end{array}
$$

Semántica

$$\frac{\langle P; \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle \qquad \mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle}$$

$$\frac{\neg\mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle \surd, \mu\rangle}$$

$$\downarrow$$

$$\langle P_{\mathsf{w}}, (0,0)\rangle$$

$P_{\mathsf{w}}:$ `while` $x < 2$ `do`
  `if`
    $[\!]\ (x > 0) \wedge (y = 0) \rightarrow$
      x `:=` $x - 1$
    $[\!]\ (x < 2) \wedge (y = 0) \rightarrow$
      x `:=` $x + 1$
  `fi`
`od`

Semántica

$$\frac{\langle P; \texttt{while } b \texttt{ do } P \texttt{ od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle \qquad \mu(b) \text{ holds}}{\langle \texttt{while } b \texttt{ do } P \texttt{ od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle}$$

$$\frac{\neg\mu(b) \text{ holds}}{\langle \texttt{while } b \texttt{ do } P \texttt{ od}, \mu\rangle \longrightarrow \langle \surd, \mu\rangle}$$

$$\downarrow$$

$$\langle P_{\mathsf{w}}, (0,0)\rangle$$

P_{w} : while $x < 2$ do
  if
    [] $(x > 0) \wedge (y = 0) \rightarrow$
      x := $x - 1$
    [] $(x < 2) \wedge (y = 0) \rightarrow$
      x := $x + 1$
  fi
od

$$\langle P_{\mathsf{w}}, (0,0)\rangle \longrightarrow$$

Semántica

$$\frac{\langle P; \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle \qquad \mu(b) \text{ holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle}$$

$$\frac{\neg\mu(b) \text{ holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle \surd, \mu\rangle}$$

$$\downarrow$$

$$\langle P_{\mathsf{w}}, (0,0)\rangle$$

P_{w} : while $x < 2$ do
if
[] $(x > 0) \wedge (y = 0) \rightarrow$
x := $x - 1$
[] $(x < 2) \wedge (y = 0) \rightarrow$
x := $x + 1$
fi
od

$$\frac{\langle P_{\mathsf{c}}\ ;\ P_{\mathsf{w}}, (0,0)\rangle \longrightarrow}{\langle P_{\mathsf{w}}, (0,0)\rangle \longrightarrow}$$

CONICET

UNC

Semántica

$$\frac{\langle P; \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle \qquad \mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle}$$

$$\frac{\neg\mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle \surd, \mu\rangle}$$

$$\downarrow$$

$$\langle P_{\mathsf{w}}, (0,0)\rangle$$

$P_{\mathsf{w}}:$ $\texttt{while}\ x < 2\ \texttt{do}$
$\quad\texttt{if}$
$\qquad[]\ (x > 0) \wedge (y = 0) \to$
$\qquad\quad x \mathrel{\texttt{:=}} x - 1$
$\qquad[]\ (x < 2) \wedge (y = 0) \to$
$\qquad\quad x \mathrel{\texttt{:=}} x + 1$
$\quad\texttt{fi}$
$\texttt{od}$

$$\frac{\dfrac{\langle P_{\mathsf{c}}, (0,0)\rangle \longrightarrow}{\langle P_{\mathsf{c}}\ ;\ P_{\mathsf{w}}, (0,0)\rangle \longrightarrow}}{\langle P_{\mathsf{w}}, (0,0)\rangle \longrightarrow}$$

Semántica

$$\frac{\langle P; \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle \qquad \mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle}$$

$$\frac{\neg\mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle \surd, \mu\rangle}$$

$$\downarrow$$

$$\langle P_{\mathsf{w}}, (0,0)\rangle$$

P_{w} : $\texttt{while}\ x < 2\ \texttt{do}$
  $\texttt{if}$
    $[\!]\ (x > 0) \wedge (y = 0) \rightarrow$
      $x\ \texttt{:=}\ x - 1$
    $[\!]\ (x < 2) \wedge (y = 0) \rightarrow$
      $x\ \texttt{:=}\ x + 1$
  $\texttt{fi}$
$\texttt{od}$

$$\cfrac{\cfrac{\cfrac{\langle x\ \texttt{:=}\ x+1, (0,0)\rangle \longrightarrow \langle \surd, (1,0)\rangle \qquad (0,0) \models (x<2) \wedge (y=0)}{\langle P_{\mathsf{c}}, (0,0)\rangle \longrightarrow}}{\langle P_{\mathsf{c}}\ ;\ P_{\mathsf{w}}, (0,0)\rangle \longrightarrow}}{\langle P_{\mathsf{w}}, (0,0)\rangle \longrightarrow}$$

CONICET

UNC

Semántica

$$\frac{\langle P; \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle \qquad \mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle}$$

$$\frac{\neg\mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle \surd, \mu\rangle}$$

$$\downarrow$$

$$\langle P_{\mathsf{w}}, (0,0)\rangle$$

P_{w} : $\texttt{while}\ x < 2\ \texttt{do}$

$\quad\texttt{if}$

$\qquad [\!]\ (x > 0) \wedge (y = 0) \rightarrow$

$\qquad\quad x\ \texttt{:=}\ x - 1$

$\qquad [\!]\ (x < 2) \wedge (y = 0) \rightarrow$

$\qquad\quad x\ \texttt{:=}\ x + 1$

$\quad\texttt{fi}$

$\texttt{od}$

$$\dfrac{\dfrac{\dfrac{\langle x\ \texttt{:=}\ x+1, (0,0)\rangle \longrightarrow \langle \surd, (1,0)\rangle \qquad (0,0) \models (x<2) \wedge (y=0)}{\langle P_{\mathsf{c}}, (0,0)\rangle \longrightarrow \langle \surd, (1,0)\rangle}}{\langle P_{\mathsf{c}}\ ;\ P_{\mathsf{w}}, (0,0)\rangle \longrightarrow}}{\langle P_{\mathsf{w}}, (0,0)\rangle \longrightarrow}$$

Semántica

$$\frac{\langle P; \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle \qquad \mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle}$$

$$\frac{\neg\mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle \surd, \mu\rangle}$$

$$\downarrow$$

$$\langle P_{\mathsf{w}}, (0,0)\rangle$$

$P_{\mathsf{w}}:$ `while` $x < 2$ `do`
  `if`
    $[]\ (x > 0) \wedge (y = 0) \rightarrow$
      x `:=` $x - 1$
    $[]\ (x < 2) \wedge (y = 0) \rightarrow$
      x `:=` $x + 1$
  `fi`
`od`

$$\frac{\dfrac{\dfrac{\langle x \mathrel{\texttt{:=}} x+1, (0,0)\rangle \longrightarrow \langle \surd, (1,0)\rangle \qquad (0,0) \models (x < 2) \wedge (y = 0)}{\langle P_{\mathsf{c}}, (0,0)\rangle \longrightarrow \langle \surd, (1,0)\rangle}}{\langle P_{\mathsf{c}}\ ;\ P_{\mathsf{w}}, (0,0)\rangle \longrightarrow \langle P_{\mathsf{w}}, (1,0)\rangle}}{\langle P_{\mathsf{w}}, (0,0)\rangle \longrightarrow}$$

Semántica

$$\frac{\langle P; \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle \qquad \mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle}$$

$$\frac{\neg\mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle \surd, \mu\rangle}$$

$$\downarrow$$

$$\langle P_{\mathsf{w}}, (0,0)\rangle$$

$P_{\mathsf{w}}:$ $\texttt{while}\ x < 2\ \texttt{do}$
  $\texttt{if}$
    $[]\ (x > 0) \wedge (y = 0) \rightarrow$
      $x := x - 1$
    $[]\ (x < 2) \wedge (y = 0) \rightarrow$
      $x := x + 1$
  $\texttt{fi}$
$\texttt{od}$

$$\cfrac{\cfrac{\cfrac{\langle x := x+1, (0,0)\rangle \longrightarrow \langle \surd, (1,0)\rangle \qquad (0,0) \models (x < 2) \wedge (y = 0)}{\langle P_{\mathsf{c}}, (0,0)\rangle \longrightarrow \langle \surd, (1,0)\rangle}}{\langle P_{\mathsf{c}}\ ;\ P_{\mathsf{w}}, (0,0)\rangle \longrightarrow \langle P_{\mathsf{w}}, (1,0)\rangle}}{\langle P_{\mathsf{w}}, (0,0)\rangle \longrightarrow \langle P_{\mathsf{w}}, (1,0)\rangle}$$

Semántica

$$\frac{\langle P; \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle \qquad \mu(b) \text{ holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle}$$

$$\frac{\neg\mu(b) \text{ holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle \surd, \mu\rangle}$$

$$\begin{array}{c} \downarrow \\ \langle P_{\mathsf{w}}, (0,0)\rangle \\ \downarrow \\ \langle P_{\mathsf{w}}, (1,0)\rangle \end{array}$$

P_{w} : while $x < 2$ do
  if
    $[]\ (x > 0) \wedge (y = 0) \rightarrow$
      x := $x - 1$
    $[]\ (x < 2) \wedge (y = 0) \rightarrow$
      x := $x + 1$
  fi
od

Semántica

$$\frac{\langle P; \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle \qquad \mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle}$$

$$\frac{\neg\mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle \surd, \mu\rangle}$$

$$\begin{array}{ll}
P_{\mathsf{w}}: & \texttt{while}\ x < 2\ \ \texttt{do} \\
& \quad \texttt{if} \\
& \quad\quad [\!]\ (x > 0) \wedge (y = 0) \rightarrow \\
& \quad\quad\quad\quad x\ \texttt{:=}\ x - 1 \\
& \quad\quad [\!]\ (x < 2) \wedge (y = 0) \rightarrow \\
& \quad\quad\quad\quad x\ \texttt{:=}\ x + 1 \\
& \quad \texttt{fi} \\
& \texttt{od}
\end{array}$$

$$\begin{tikzcd}
{} \arrow[d] \\
\langle P_{\mathsf{w}}, (0,0)\rangle \arrow[d, bend left] \\
\langle P_{\mathsf{w}}, (1,0)\rangle \arrow[u, bend left]
\end{tikzcd}$$

Semántica

$$\frac{\langle P;\texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle \qquad \mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle}$$

$$\frac{\neg\mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle \surd, \mu\rangle}$$

$$\begin{array}{ll}
P_{\mathsf{w}}: & \texttt{while}\ x < 2\ \ \texttt{do} \\
& \quad \texttt{if} \\
& \quad\quad [\!]\ (x > 0) \wedge (y = 0) \rightarrow \\
& \quad\quad\quad\quad x \ \texttt{:=}\ x - 1 \\
& \quad\quad [\!]\ (x < 2) \wedge (y = 0) \rightarrow \\
& \quad\quad\quad\quad x \ \texttt{:=}\ x + 1 \\
& \quad \texttt{fi} \\
& \texttt{od}
\end{array}$$

$$\begin{tikzcd}
{} \arrow[d] \\
\langle P_{\mathsf{w}}, (0,0)\rangle \arrow[d, bend left] \\
\langle P_{\mathsf{w}}, (1,0)\rangle \arrow[u, bend left] \arrow[d] \\
\langle P_{\mathsf{w}}, (2,0)\rangle
\end{tikzcd}$$

Semántica

$$\frac{\langle P; \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle \qquad \mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle}$$

$$\frac{\neg\mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle \surd, \mu\rangle}$$

P_{w} :
```
while x < 2 do
  if
    [] (x > 0) ∧ (y = 0) →
         x := x − 1
    [] (x < 2) ∧ (y = 0) →
         x := x + 1
  fi
od
```

$$\begin{tikzcd}
{} \arrow[d] \\
\langle P_{\mathsf{w}}, (0,0)\rangle \arrow[d, bend left] \\
\langle P_{\mathsf{w}}, (1,0)\rangle \arrow[u, bend left] \arrow[d] \\
\langle P_{\mathsf{w}}, (2,0)\rangle \arrow[d] \\
\langle \surd, (2,0)\rangle
\end{tikzcd}$$

Semántica

$$\frac{\langle P; \mathtt{while}\ b\ \mathtt{do}\ P\ \mathtt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle \qquad \mu(b)\ \text{holds}}{\langle \mathtt{while}\ b\ \mathtt{do}\ P\ \mathtt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle}$$

$$\frac{\neg\mu(b)\ \text{holds}}{\langle \mathtt{while}\ b\ \mathtt{do}\ P\ \mathtt{od}, \mu\rangle \longrightarrow \langle \surd, \mu\rangle}$$

Semántica

$$\frac{\langle P; \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle \qquad \mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle}$$

$$\frac{\neg\mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle \surd, \mu\rangle}$$

P_1^∞ : `while true do`

```
    if
      [] (x > 0) ∧ (y = 0) →
           x := x − 1
      [] (x < 2) ∧ (y = 0) →
           x := x + 1
    fi
od
```

CONICET

UNC

Semántica

$$\frac{\langle P; \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle \qquad \mu(b) \text{ holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle}$$

$$\frac{\neg\mu(b) \text{ holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle \surd, \mu\rangle}$$

$$\begin{array}{l}
P_1^\infty : \texttt{while}\ \texttt{true}\ \texttt{do} \\
\qquad \texttt{if} \\
\qquad\quad [\!]\ (x > 0) \wedge (y = 0) \rightarrow \\
\qquad\qquad x := x - 1 \\
\qquad\quad [\!]\ (x < 2) \wedge (y = 0) \rightarrow \\
\qquad\qquad x := x + 1 \\
\qquad \texttt{fi} \\
\quad \texttt{od}
\end{array}$$

Semántica

$$\frac{\langle P; \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle \qquad \mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle}$$

$$\frac{\neg\mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle \surd, \mu\rangle}$$

$$
\begin{array}{l}
P_1^\infty : \texttt{while}\ \texttt{true}\ \texttt{do} \\
\qquad \texttt{if} \\
\qquad\quad [\!]\ (x > 0) \wedge (y = 0) \rightarrow \\
\qquad\qquad x \texttt{ := } x - 1 \\
\qquad\quad [\!]\ (x < 2) \wedge (y = 0) \rightarrow \\
\qquad\qquad x \texttt{ := } x + 1 \\
\qquad \texttt{fi} \\
\quad \texttt{od}
\end{array}
$$

$$
\begin{tikzcd}
{} \arrow[d] \\
\langle P_1^\infty, (0,0)\rangle \arrow[d, bend left] \\
\langle P_1^\infty, (1,0)\rangle \arrow[u, bend left] \arrow[d, bend left] \\
\langle P_1^\infty, (2,0)\rangle \arrow[u, bend left]
\end{tikzcd}
$$

Semántica

$$\frac{\langle P; \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle \qquad \mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle}$$

$$\frac{\neg\mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle \surd, \mu\rangle}$$

P_1^∞ : `while` `true` `do`
  `if`
   $[\!]\ (x > 0) \wedge (y = 0) \rightarrow$
     x `:=` $x - 1$
   $[\!]\ (x < 2) \wedge (y = 0) \rightarrow$
     x `:=` $x + 1$
  `fi`
 `od`

$$\begin{tikzcd}
{} \arrow[d] \\
\langle P_1^\infty, (0,0)\rangle \arrow[d, bend left] \\
\langle P_1^\infty, (1,0)\rangle \arrow[u, bend left] \arrow[d, bend left] \\
\langle P_1^\infty, (2,0)\rangle \arrow[u, bend left]
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
{} \arrow[d] \\
\langle P_1^\infty, (?,1)\rangle
\end{tikzcd}$$

Semántica

$$\frac{\langle P; \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle \qquad \mu(b) \text{ holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle}$$

$$\frac{\neg\mu(b) \text{ holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle \surd, \mu\rangle}$$

P_1^∞ : `while true do`

```
        if
          [] (x > 0) ∧ (y = 0) →
               x := x − 1
          [] (x < 2) ∧ (y = 0) →
               x := x + 1
        fi
    od
```

$$\begin{tikzcd}
\arrow[d] \\
\langle P_1^\infty, (0,0)\rangle \arrow[d, bend left] \\
\langle P_1^\infty, (1,0)\rangle \arrow[u, bend left] \arrow[d, bend left] \\
\langle P_1^\infty, (2,0)\rangle \arrow[u, bend left]
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
\arrow[d] \\
\langle P_1^\infty, (?, 1)\rangle
\end{tikzcd}$$

Semántica

$$\frac{\langle P; \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle \qquad \mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle}$$

$$\frac{\neg\mu(b)\ \text{holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle \surd, \mu\rangle}$$

$$\begin{array}{ll} P_2^\infty : & \texttt{while}\ \texttt{true}\ \ \texttt{do} \\ & \quad y\ \texttt{:=}\ 1\ ; \\ & \quad y\ \texttt{:=}\ 0 \\ & \texttt{od} \end{array}$$

$$\begin{tikzcd}
{} \arrow[d] \\
\langle P_2^\infty, (?, 0)\rangle \arrow[d, bend left] \\
\langle y\ \texttt{:=}\ 0\ ;\ P_1^\infty, (?, 1)\rangle \arrow[u, bend left]
\end{tikzcd}$$

Semántica

$$\frac{\langle P; \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle \qquad \mu(b) \text{ holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle}$$

$$\frac{\neg\mu(b) \text{ holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle \surd, \mu\rangle}$$

$$\begin{array}{ll} P_2^\infty : & \texttt{while}\ \texttt{true}\ \ \texttt{do} \\ & \quad y\ \texttt{:=}\ 1\ ; \\ & \quad y\ \texttt{:=}\ 0 \\ & \texttt{od} \end{array}$$

$$\begin{tikzcd}
{} \arrow[d] \\
\langle P_2^\infty, (?, 0)\rangle \arrow[d, bend left] \\
\langle y\ \texttt{:=}\ 0\ ;\ P_1^\infty, (?, 1)\rangle \arrow[u, bend left]
\end{tikzcd}$$

Semántica

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle \quad P_2 \neq \surd}{\langle P_1 \mid\mid P_2, \mu\rangle \longrightarrow \langle P_1' \mid\mid P_2, \mu'\rangle}$$

$$\frac{\langle P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle \quad P_1 \neq \surd}{\langle P_1 \mid\mid P_2, \mu\rangle \longrightarrow \langle P_1 \mid\mid P_2', \mu'\rangle}$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle}{\langle P_1 \mid\mid \surd, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle}$$

$$\frac{\langle P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle}{\langle \surd \mid\mid P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle}$$

CONICET

UNC

Semántica

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle \quad P_2 \neq \surd}{\langle P_1 \mid\mid P_2, \mu\rangle \longrightarrow \langle P_1' \mid\mid P_2, \mu'\rangle}$$

$$\frac{\langle P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle \quad P_1 \neq \surd}{\langle P_1 \mid\mid P_2, \mu\rangle \longrightarrow \langle P_1 \mid\mid P_2', \mu'\rangle}$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle}{\langle P_1 \mid\mid \surd, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle}$$

$$\frac{\langle P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle}{\langle \surd \mid\mid P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle}$$

```
while true  do
  if
    [] (x > 0) ∧ (y = 0) →
         x := x − 1
    [] (x < 2) ∧ (y = 0) →
         x := x + 1
  fi
od
```

||

```
while true  do
  y := 1 ;
  y := 0
od
```

CONICET

UNC

Semántica

$$\frac{\langle P_1,\mu\rangle \longrightarrow \langle P_1',\mu'\rangle \quad P_2 \neq \surd}{\langle P_1 \mid\mid P_2,\mu\rangle \longrightarrow \langle P_1' \mid\mid P_2,\mu'\rangle}$$

$$\frac{\langle P_2,\mu\rangle \longrightarrow \langle P_2',\mu'\rangle \quad P_1 \neq \surd}{\langle P_1 \mid\mid P_2,\mu\rangle \longrightarrow \langle P_1 \mid\mid P_2',\mu'\rangle}$$

$$\frac{\langle P_1,\mu\rangle \longrightarrow \langle P_1',\mu'\rangle}{\langle P_1 \mid\mid \surd,\mu\rangle \longrightarrow \langle P_1',\mu'\rangle}$$

$$\frac{\langle P_2,\mu\rangle \longrightarrow \langle P_2',\mu'\rangle}{\langle \surd \mid\mid P_2,\mu\rangle \longrightarrow \langle P_2',\mu'\rangle}$$

```
while true  do              ||    while true  do
  if                        ||      y := 1 ;
    [] (x > 0)∧(y = 0) →    ||      y := 0
         x := x − 1         ||    od
    [] (x < 2)∧(y = 0) →    ||
         x := x + 1         ||
  fi                        ||
od                          ||
```

$$\downarrow$$

$$\langle P_1^{\infty} \mid\mid P_2^{\infty},(0,0)\rangle$$

Semántica

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle \quad P_2 \neq \surd}{\langle P_1 \mid\mid P_2, \mu\rangle \longrightarrow \langle P_1' \mid\mid P_2, \mu'\rangle}$$

$$\frac{\langle P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle \quad P_1 \neq \surd}{\langle P_1 \mid\mid P_2, \mu\rangle \longrightarrow \langle P_1 \mid\mid P_2', \mu'\rangle}$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle}{\langle P_1 \mid\mid \surd, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle}$$

$$\frac{\langle P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle}{\langle \surd \mid\mid P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle}$$

```
while true  do                          ||   while true do
  if                                    ||     y := 1 ;
    [] (x > 0) ∧ (y = 0) →              ||     y := 0
          x := x − 1                    ||   od
    [] (x < 2) ∧ (y = 0) →              ||
          x := x + 1                    ||
  fi                                    ||
od                                      ||
```

$$\downarrow$$

$$\langle P_1^{\infty} \mid\mid P_2^{\infty}, (0,0)\rangle$$

$$\begin{tikzcd}
{} \arrow[d] & {} \arrow[d] & {} \arrow[d] \\
\langle P_1^{\infty}, (0,0)\rangle \arrow[d, bend left] & \langle P_1^{\infty}, (?,1)\rangle & \langle P_2^{\infty}, (?,0)\rangle \arrow[d, bend left] \\
\langle P_1^{\infty}, (1,0)\rangle \arrow[u, bend left] \arrow[d, bend left] & & \langle y := 0 \,;\, P_1^{\infty}, (?,1)\rangle \arrow[u, bend left] \\
\langle P_1^{\infty}, (2,0)\rangle \arrow[u, bend left] & &
\end{tikzcd}$$

Semántica

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle \quad P_2 \neq \surd}{\langle P_1 \mid\mid P_2, \mu\rangle \longrightarrow \langle P_1' \mid\mid P_2, \mu'\rangle}$$

$$\frac{\langle P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle \quad P_1 \neq \surd}{\langle P_1 \mid\mid P_2, \mu\rangle \longrightarrow \langle P_1 \mid\mid P_2', \mu'\rangle}$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle}{\langle P_1 \mid\mid \surd, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle}$$

$$\frac{\langle P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle}{\langle \surd \mid\mid P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle}$$

```
while true do                        ||   while true do
  if                                 ||     y := 1 ;
    [] (x > 0) ∧ (y = 0) →           ||     y := 0
         x := x − 1                  ||   od
    [] (x < 2) ∧ (y = 0) →           ||
         x := x + 1                  ||
  fi                                 ||
od                                   ||
```

$$\begin{tikzcd}
\ \arrow[d] & \\
\langle P_1^\infty \mid\mid P_2^\infty, (0,0)\rangle \arrow[r, bend left] \arrow[d, bend left] & \langle P_1^\infty \mid\mid (y := 0 \,;\, P_2^\infty), (0,1)\rangle \arrow[l, bend left] \\
\langle P_1^\infty \mid\mid P_2^\infty, (1,0)\rangle \arrow[r, bend left] \arrow[u, bend left] \arrow[d, bend left] & \langle P_1^\infty \mid\mid (y := 0 \,;\, P_2^\infty), (1,1)\rangle \arrow[l, bend left] \\
\langle P_1^\infty \mid\mid P_2^\infty, (2,0)\rangle \arrow[r, bend left] \arrow[u, bend left] & \langle P_1^\infty \mid\mid (y := 0 \,;\, P_2^\infty), (2,1)\rangle \arrow[l, bend left]
\end{tikzcd}$$

$$\begin{tikzcd}
\ \arrow[d] & \ \arrow[d] & \ \arrow[d] \\
\langle P_1^\infty, (0,0)\rangle \arrow[d, bend left] & \langle P_1^\infty, (?,1)\rangle & \langle P_2^\infty, (?,0)\rangle \arrow[d, bend left] \\
\langle P_1^\infty, (1,0)\rangle \arrow[u, bend left] \arrow[d, bend left] & & \langle y := 0 \,;\, P_1^\infty, (?,1)\rangle \arrow[u, bend left] \\
\langle P_1^\infty, (2,0)\rangle \arrow[u, bend left] & &
\end{tikzcd}$$

Ejecuciones

Una ejecución de $K = (S, s_0, \longrightarrow, L)$ es una función $\rho : \mathbb{N} \to S$ tal que:

1. $\rho(0) = s_0$, y
2. $\rho(i) \longrightarrow \rho(i+1)$, para todo $i \geq 0$.

Ejecuciones

supondremos
que siempre $s \longrightarrow s'$
para algún s'

Una ejecución de $K = (S, s_0, \longrightarrow, L)$ es una función $\rho : \mathbb{N} \to S$ tal que:

1. $\rho(0) = s_0$, y
2. $\rho(i) \longrightarrow \rho(i+1)$, para todo $i \geq 0$.

Ejecuciones

supondremos que siempre $s \longrightarrow s'$ para algún s'

Una ejecución de $K = (S, s_0, \longrightarrow, L)$ es una función $\rho : \mathbb{N} \to S$ tal que:

denotado por $\rho \in S^\omega$

1. $\rho(0) = s_0$, y
2. $\rho(i) \longrightarrow \rho(i+1)$, para todo $i \geq 0$.

Ejecuciones

supondremos que siempre $s \longrightarrow s'$ para algún s'

Una ejecución de $K = (S, s_0, \longrightarrow, L)$ es una función $\rho : \mathbb{N} \to S$ tal que:

denotado por $\rho \in S^\omega$

1. $\rho(0) = s_0$, y
2. $\rho(i) \longrightarrow \rho(i+1)$, para todo $i \geq 0$.

$$\begin{tikzcd}
 & {} \arrow[d] & & \\
\{b\} & s_0 \arrow[rr, bend left] \arrow[d] & & s_1 \arrow[ll, bend left] & \{b\} \\
\varnothing & s_2 \arrow[rr, bend left] & & s_3 \arrow[ll, bend left] \arrow[u] & \{a, b\}
\end{tikzcd}$$

Ejecuciones

supondremos que siempre $s \longrightarrow s'$ para algún s'

Una ejecución de $K = (S, s_0, \longrightarrow, L)$ es una función $\rho : \mathbb{N} \to S$ tal que:

denotado por $\rho \in S^\omega$

1. $\rho(0) = s_0$, y
2. $\rho(i) \longrightarrow \rho(i+1)$, para todo $i \geq 0$.

$$\begin{tikzcd}
 & {} \arrow[d] & & \\
\{b\} & s_0 \arrow[rr, bend left] \arrow[d] & & s_1 \arrow[ll, bend left] & \{b\} \\
\varnothing & s_2 \arrow[rr, bend left] & & s_3 \arrow[ll, bend left] \arrow[u] & \{a,b\}
\end{tikzcd}$$

$s_0\ s_1\ s_0\ s_1\ s_0\ s_2\ s_3\ s_1\ s_0\ s_2\ s_3\ s_2\ \ldots$

$s_1\ s_0\ s_1\ s_0\ s_2\ s_3\ s_1\ s_0\ s_2\ s_3\ s_2\ s_3\ \ldots$

$s_0\ s_1\ s_2\ s_1\ s_0\ s_2\ s_3\ s_0\ s_1\ s_0\ s_2\ s_2\ \ldots$

Ejecuciones

supondremos que siempre $s \longrightarrow s'$ para algún s'

Una ejecución de $K = (S, s_0, \longrightarrow, L)$ es una función $\rho : \mathbb{N} \to S$ tal que:

denotado por $\rho \in S^\omega$

1. $\rho(0) = s_0$, y
2. $\rho(i) \longrightarrow \rho(i+1)$, para todo $i \geq 0$.

$$
\begin{tikzcd}
 & {} \arrow[d] & & \\
\{b\} & s_0 \arrow[r, bend left] \arrow[d] & s_1 \arrow[l, bend left] & \{b\} \\
\varnothing & s_2 \arrow[r, bend left] & s_3 \arrow[l, bend left] \arrow[u] & \{a, b\}
\end{tikzcd}
$$

✓ $s_0\ s_1\ s_0\ s_1\ s_0\ s_2\ s_3\ s_1\ s_0\ s_2\ s_3\ s_2\ \ldots$

$s_1\ s_0\ s_1\ s_0\ s_2\ s_3\ s_1\ s_0\ s_2\ s_3\ s_2\ s_3\ \ldots$

$s_0\ s_1\ s_2\ s_1\ s_0\ s_2\ s_3\ s_0\ s_1\ s_0\ s_2\ s_2\ \ldots$

Ejecuciones

supondremos que siempre $s \longrightarrow s'$ para algún s'

Una ejecución de $K = (S, s_0, \longrightarrow, L)$ es una función $\rho : \mathbb{N} \to S$ tal que:

denotado por $\rho \in S^\omega$

1. $\rho(0) = s_0$, y
2. $\rho(i) \longrightarrow \rho(i+1)$, para todo $i \geq 0$.

$$\begin{tikzcd} \{b\}\ s_0 \arrow[r, bend left] \arrow[d] & s_1\ \{b\} \arrow[l, bend left] \\ \varnothing\ s_2 \arrow[r, bend left] & s_3\ \{a,b\} \arrow[l, bend left] \arrow[u] \end{tikzcd}$$

✓ $s_0\ s_1\ s_0\ s_1\ s_0\ s_2\ s_3\ s_1\ s_0\ s_2\ s_3\ s_2\ \ldots$

✗ $s_1\ s_0\ s_1\ s_0\ s_2\ s_3\ s_1\ s_0\ s_2\ s_3\ s_2\ s_3\ \ldots$

$s_0\ s_1\ s_2\ s_1\ s_0\ s_2\ s_3\ s_0\ s_1\ s_0\ s_2\ s_2\ \ldots$

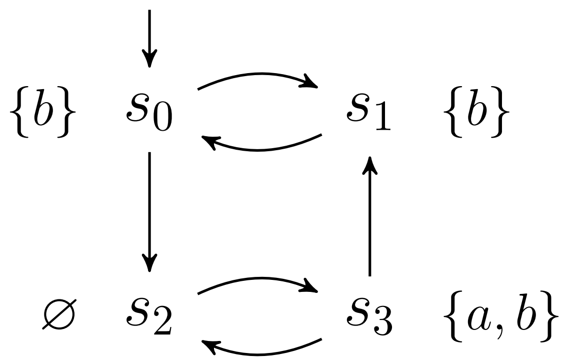
Ejecuciones

supondremos que siempre $s \longrightarrow s'$ para algún s'

Una ejecución de $K = (S, s_0, \longrightarrow, L)$ es una función $\rho : \mathbb{N} \to S$ tal que:

denotado por $\rho \in S^\omega$

1. $\rho(0) = s_0$, y
2. $\rho(i) \longrightarrow \rho(i+1)$, para todo $i \geq 0$.



✓ $s_0\ s_1\ s_0\ s_1\ s_0\ s_2\ s_3\ s_1\ s_0\ s_2\ s_3\ s_2\ \ldots$

✗ $s_1\ s_0\ s_1\ s_0\ s_2\ s_3\ s_1\ s_0\ s_2\ s_3\ s_2\ s_3\ \ldots$

✗ $s_0\ s_1\ s_2\ s_1\ s_0\ s_2\ s_3\ s_0\ s_1\ s_0\ s_2\ s_2\ \ldots$

Trazas observables

- Para hablar de las propiedades de un sistema debemos observar las propiedades de sus estados y su orden temporal en las ejecuciones.

CONICET

UNC

Trazas observables

- Para hablar de las propiedades de un sistema debemos observar las propiedades de sus estados y su orden temporal en las ejecuciones.

$$\mathcal{L}(K) = \{\sigma : \mathbb{N} \to \mathcal{P}(\mathsf{PA}) \mid \exists\ \rho \text{ ejecución de } K : \forall\ i \geq 0 : \sigma(i) = \mathcal{L}(\rho(i))\}$$

Trazas observables

- Para hablar de las propiedades de un sistema debemos observar las propiedades de sus estados y su orden temporal en las ejecuciones.

$$\mathcal{L}(K) = \{\sigma \in (\mathcal{P}(\mathsf{PA}))^\omega \quad | \; \exists\, \rho \text{ ejecución de } K : \forall\, i \geq 0 : \sigma(i) = \mathcal{L}(\rho(i))\}$$

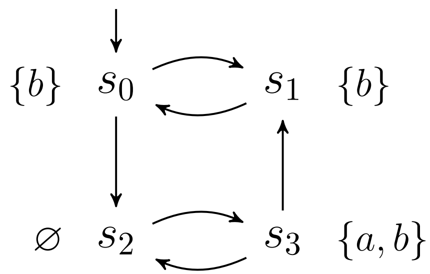
CONICET

UNC

Trazas observables

- Para hablar de las propiedades de un sistema debemos observar las propiedades de sus estados y su orden temporal en las ejecuciones.

$$\mathcal{L}(K) = \{\sigma \in (\mathcal{P}(\mathsf{PA}))^{\omega} \quad | \; \exists\, \rho \text{ ejecución de } K : \forall\, i \geq 0 : \sigma(i) = \mathcal{L}(\rho(i))\}$$



Trazas observables

- Para hablar de las propiedades de un sistema debemos observar las propiedades de sus estados y su orden temporal en las ejecuciones.

$$\mathcal{L}(K) = \{\sigma \in (\mathcal{P}(\mathsf{PA}))^{\omega} \quad \mid \exists\, \rho \text{ ejecución de } K : \forall\, i \geq 0 : \sigma(i) = \mathcal{L}(\rho(i))\}$$

$$\rho \;=\; s_0 \quad s_1 \quad s_0 \quad s_1 \quad s_0 \quad s_2 \quad s_3 \quad s_1 \quad s_0 \quad s_2 \quad s_3 \quad s_2 \quad \ldots$$

$$\sigma \;=$$

$$\begin{tikzcd}
 & {} \arrow[d] & & \\
\{b\} & s_0 \arrow[r, bend left] \arrow[d] & s_1 \arrow[l, bend left] & \{b\} \\
\varnothing & s_2 \arrow[r, bend left] & s_3 \arrow[l, bend left] \arrow[u] & \{a,b\}
\end{tikzcd}$$

Trazas observables

- Para hablar de las propiedades de un sistema debemos observar las propiedades de sus estados y su orden temporal en las ejecuciones.

$$\mathcal{L}(K) = \{\sigma \in (\mathcal{P}(\mathsf{PA}))^\omega \quad | \; \exists\, \rho \text{ ejecución de } K : \forall\, i \geq 0 : \sigma(i) = \mathcal{L}(\rho(i))\}$$

$$\rho \; = \; s_0 \quad s_1 \quad s_0 \quad s_1 \quad s_0 \quad s_2 \quad s_3 \quad s_1 \quad s_0 \quad s_2 \quad s_3 \quad s_2 \quad \ldots$$

$$\sigma \; = \; L(s_0)$$

$$\begin{tikzcd}
 & {} \arrow[d] & & \\
\{b\} & s_0 \arrow[rr, bend left] \arrow[d] & & s_1 \arrow[ll, bend left] & \{b\} \\
\varnothing & s_2 \arrow[rr, bend left] & & s_3 \arrow[ll, bend left] \arrow[u] & \{a,b\}
\end{tikzcd}$$

Trazas observables

- Para hablar de las propiedades de un sistema debemos observar las propiedades de sus estados y su orden temporal en las ejecuciones.

$$\mathcal{L}(K) = \{\sigma \in (\mathcal{P}(\mathsf{PA}))^\omega \mid \exists\, \rho \text{ ejecución de } K : \forall\, i \geq 0 : \sigma(i) = \mathcal{L}(\rho(i))\}$$

$$\rho \;=\; s_0 \quad s_1 \quad s_0 \quad s_1 \quad s_0 \quad s_2 \quad s_3 \quad s_1 \quad s_0 \quad s_2 \quad s_3 \quad s_2 \quad \ldots$$

$$\sigma \;=\; L(s_0) \quad L(s_1)$$

$$\begin{tikzcd}
\{b\} & s_0 \arrow[r, bend left] \arrow[d] & s_1 \arrow[l, bend left] & \{b\} \\
\varnothing & s_2 \arrow[r, bend left] & s_3 \arrow[l, bend left] \arrow[u] & \{a, b\}
\end{tikzcd}$$

Trazas observables

- Para hablar de las propiedades de un sistema debemos observar las propiedades de sus estados y su orden temporal en las ejecuciones.

$$\mathcal{L}(K) = \{\sigma \in (\mathcal{P}(\mathsf{PA}))^\omega \quad | \; \exists\, \rho \text{ ejecución de } K : \forall\, i \geq 0 : \sigma(i) = \mathcal{L}(\rho(i))\}$$

$$\begin{array}{cccccccccccccc}
\rho & = & s_0 & s_1 & s_0 & s_1 & s_0 & s_2 & s_3 & s_1 & s_0 & s_2 & s_3 & s_2 & \ldots \\
 & & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \\
\sigma & = & L(s_0) & L(s_1) & L(s_0) & L(s_1) & L(s_0) & L(s_2) & L(s_3) & L(s_1) & L(s_0) & L(s_2) & L(s_3) & L(s_2) & \ldots
\end{array}$$

$$\begin{tikzcd}
 & {} \arrow[d] & & \\
\{b\} & s_0 \arrow[r, bend left] \arrow[d] & s_1 \arrow[l, bend left] & \{b\} \\
\varnothing & s_2 \arrow[r, bend left] & s_3 \arrow[l, bend left] \arrow[u] & \{a,b\}
\end{tikzcd}$$

CONICET

UNC

Trazas observables

- Para hablar de las propiedades de un sistema debemos observar las propiedades de sus estados y su orden temporal en las ejecuciones.

$$\mathcal{L}(K) = \{\sigma \in (\mathcal{P}(\mathsf{PA}))^\omega \quad | \; \exists\, \rho \text{ ejecución de } K : \forall\, i \geq 0 : \sigma(i) = \mathcal{L}(\rho(i))\}$$

$$\begin{array}{ccccccccccccccc}
\rho & = & s_0 & s_1 & s_0 & s_1 & s_0 & s_2 & s_3 & s_1 & s_0 & s_2 & s_3 & s_2 & \ldots \\
 & & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \\
\sigma & = & \{b\} & \{b\} & \{b\} & \{b\} & \{b\} & \varnothing & \{a,b\} & \{b\} & \{b\} & \varnothing & \{a,b\} & \varnothing & \ldots
\end{array}$$

$$\begin{tikzcd}
 & {} \arrow[d] & & \\
\{b\} & s_0 \arrow[r, bend left] \arrow[d] & s_1 \arrow[l, bend left] & \{b\} \\
\varnothing & s_2 \arrow[r, bend left] & s_3 \arrow[l, bend left] \arrow[u] & \{a,b\}
\end{tikzcd}$$

Ejercicio 1: Considere el siguiente programa concurrente. Siguiendo la semántica, defina dos fragmentos de ejecución distintos que lleven a la violación de la exclusión mutua (es decir `in_critical=2`) desde el estado inicial $\langle P_0^1 \mid\mid P_0^2, (0,0,0)\rangle$. Suponga que la memoria esta representada como la terna $(\mu(\texttt{y1}), \mu(\texttt{y2}), \mu(\texttt{in\_critical}))$ y que cada variable solo puede tomar los valores 0, 1, y 2.

```
P0^1 [      while true  do
              y1 := y2 + 1 ;
 P1^1 [       if
                [] ((y2 = 0) ∨ (y1 ≤ y2)) →
                     in_critical ++ ;
                     // región crítica
  P2^1 [             in_critical -- ;
   P3^1 [            y1 := 0
              fi
            od
```

||

```
P0^2 [      while true  do
              y2 := y1 + 1 ;
 P1^2 [       if
                [] ((y1 = 0) ∨ (y2 < y1)) →
                     in_critical ++ ;
                     // región crítica
  P2^2 [             in_critical -- ;
   P3^2 [            y2 := 0
              fi
            od
```

La lógica proposicional como lenguaje de especificación

Ejemplo: "Siempre que llovió, paró".

La lógica proposicional como lenguaje de especificación

Ejemplo: "Siempre que llovió, paró".

- 1er. intento:

$$\mathsf{llueve} \Rightarrow \neg \mathsf{llueve}$$

La lógica proposicional como lenguaje de especificación

Ejemplo: "Siempre que llovió, paró".

- 1er. intento:

$$\text{llueve} \Rightarrow \neg\text{llueve}$$

¡Sólo es verdadera si no llueve!

La lógica proposicional como lenguaje de especificación

Ejemplo: "Siempre que llovió, paró".

¡Sólo es verdadera si no llueve!

- 1er. intento:

$$\text{llueve} \Rightarrow \neg\text{llueve}$$

- 2do. intento:

$$\text{primero\_llueve} \Rightarrow \text{luego\_para}$$

CONICET

UNC

La lógica proposicional como lenguaje de especificación

Ejemplo: "Siempre que llovió, paró".

- 1er. intento:

$$llueve \Rightarrow \neg llueve$$

¡Sólo es verdadera si no llueve!

- 2do. intento:

$$primero\_llueve \Rightarrow luego\_para$$

Disocia los conceptos de llover y dejar de llover.

¿Cómo diferenciamos que esto ocurre siempre o que esto ocurre sólo una vez?

CONICET

UNC

La lógica proposicional como lenguaje de especificación

Ejemplo: "Siempre que llovió, paró".

¡Sólo es verdadera si no llueve!

- 1er. intento:

$$\text{llueve} \Rightarrow \neg\text{llueve}$$

Disocia los conceptos de llover y dejar de llover.

- 2do. intento:

$$\text{primero\_llueve} \Rightarrow \text{luego\_para}$$

¿Cómo diferenciamos que esto ocurre siempre o que esto ocurre sólo una vez?

No posee suficiente expresividad:

No puede reflejar el cambio de valor de verdad de las proposiciones según transcurra el tiempo.

CONICET

UNC

La lógica de primer orden como lenguaje de especificación

Ejemplo: “Siempre que llovió, paró”:

CONICET

UNC

La lógica de primer orden como lenguaje de especificación

Ejemplo: "Siempre que llovió, paró":

$$\forall\, t \in \textsc{Tiempo} : \mathit{llueve}(t) \Rightarrow \exists\, t' \in \textsc{Tiempo} : (t \leq t') \wedge \neg \mathit{llueve}(t')$$

donde $\textsc{Tiempo}$ podría ser los reales no-negativos.

La lógica de primer orden como lenguaje de especificación

Ejemplo: "Siempre que llovió, paró":

$$\forall\, t \in \textsc{Tiempo} : \mathit{llueve}(t) \Rightarrow \exists\, t' \in \textsc{Tiempo} : (t \leq t') \wedge \neg \mathit{llueve}(t')$$

donde $\textsc{Tiempo}$ podría ser los reales no-negativos.

Sin embargo:

- lenguaje demasiado complejo para representar propiedades temporales.
- satisfactibilidad es indecidible

  $\Rightarrow$ no es posible un cálculo automático

CONICET

UNC

La lógica de primer orden como lenguaje de especificación

Ejemplo: "Siempre que llovió, paró":

$$\forall\, t \in \textsc{Tiempo} : \mathit{llueve}(t) \Rightarrow \exists\, t' \in \textsc{Tiempo} : (t \leq t') \wedge \neg \mathit{llueve}(t')$$

donde $\textsc{Tiempo}$ podría ser los reales no-negativos.

Sin embargo:

- lenguaje demasiado complejo para representar propiedades temporales.
- satisfactibilidad es indecidible

$\Rightarrow$ no es posible un cálculo automático

Demasiado expresiva

CONICET

LTL: Lógica Temporal Lineal

- Extiende la lógica proposicional con modalidades para expresar la relación temporal entre la validez de las fórmulas.

- Es lineal porque sólo permite expresar tal relación a lo largo de una sola ejecución

- No considera tiempo explícito, sino el orden relativo de los eventos a lo largo del tiempo

LTL: Lógica Temporal Lineal

- Extiende la lógica proposicional con modalidades para expresar la relación temporal entre la validez de las fórmulas.

  Ej.: $\mathtt{G}(\text{llueve} \Rightarrow \mathtt{F}\, \neg\text{llueve})$

- Es lineal porque sólo permite expresar tal relación a lo largo de una sola ejecución

- No considera tiempo explícito, sino el orden relativo de los eventos a lo largo del tiempo

CONICET

UNC

LTL: Lógica Temporal Lineal

- Extiende la lógica proposicional con modalidades para expresar la relación temporal entre la validez de las fórmulas.

  Ej.: $\mathtt{G}(\mathsf{llueve} \Rightarrow \mathtt{F}\, \neg \mathsf{llueve})$

- Es lineal porque sólo permite expresar tal relación a lo largo de una sola ejecución

  otras cuantifican la bifurcación de las ejecuciones

- No considera tiempo explícito, sino el orden relativo de los eventos a lo largo del tiempo

LTL: Lógica Temporal Lineal

- Extiende la lógica proposicional con modalidades para expresar la relación temporal entre la validez de las fórmulas.

  Ej.: $\mathtt{G}(\text{llueve} \Rightarrow \mathtt{F}\, \neg\text{llueve})$

- Es lineal porque sólo permite expresar tal relación a lo largo de una sola ejecución

  otras cuantifican la bifurcación de las ejecuciones

- No considera tiempo explícito, sino el orden relativo de los eventos a lo largo del tiempo

  otras expresan el tiempo preciso de ocurrencia (tiempo real)

CONICET

UNC

LTL: Sintaxis

$$
\begin{array}{llllllllllll}
\phi, \psi ::= \\
& p & | & \neg\phi & | & \phi \wedge \psi & | & \phi \vee \psi & | & \phi \Rightarrow \psi & | & \text{(op. proposicionales)} \\
& \mathtt{X}\,\phi & | & \mathtt{F}\,\phi & | & \mathtt{G}\,\phi & | & \phi\,\mathtt{U}\,\psi & | & \phi\,\mathtt{R}\,\psi & & \text{(op. temporales)}
\end{array}
$$

donde $p \in$ PA es cualquier variable proporsional (o proposición atómica)

Los operadores básicos son $\neg$, $\vee$, $\mathtt{X}$, y $\mathtt{U}$.

Los otros operadores se definen en término de estos.

LTL: Semántica

Algunos operadores modales que admite la lógica LTL.

$\mathtt{G}\,\phi$	siempre se satisface ϕ	(Globally)
$\mathtt{F}\,\phi$	en algún instante futuro se satisface ϕ	(Finally)
$\mathtt{X}\,\phi$	en el siguiente instante se satisface ϕ	(Next)
$\phi\,\mathtt{U}\,\psi$	ϕ se debe satisfacer hasta que se satisfaga ψ	(Until)
$\phi\,\mathtt{R}\,\psi$	la satisfacción de ϕ libera la satisfacción de ψ	(Release)

CONICET

UNC

LTL: Semántica

$\mathtt{G}\, p$

$\mathtt{F}\, p$

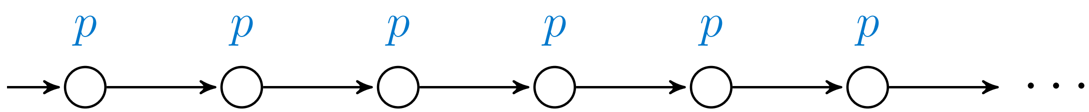
$\mathtt{X}\, p$

$p\ \mathtt{U}\ q$

$p\ \mathtt{R}\ q$

LTL: Semántica

$\mathtt{G}\,p$



$\mathtt{F}\,p$

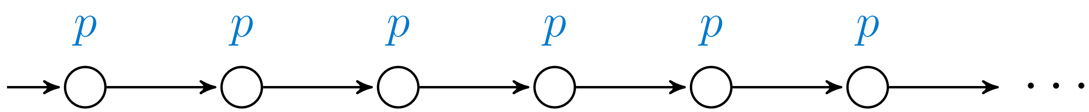
$\mathtt{X}\,p$

$p\;\mathtt{U}\;q$

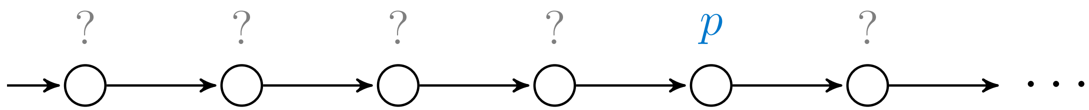
$p\;\mathtt{R}\;q$

LTL: Semántica

G p



F p



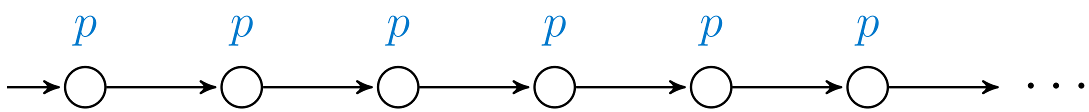
X p

p U q

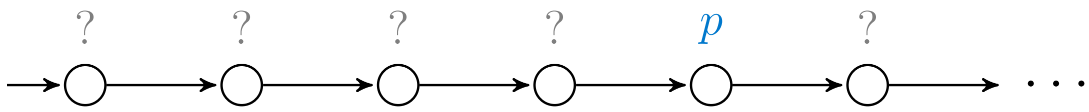
p R q

LTL: Semántica

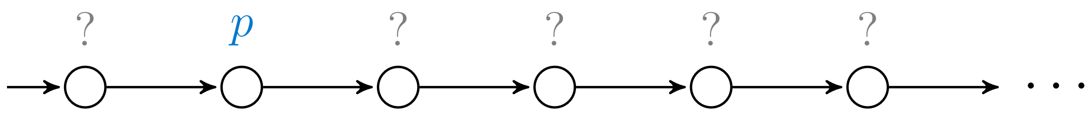
$\mathtt{G}\,p$



$\mathtt{F}\,p$



$\mathtt{X}\,p$

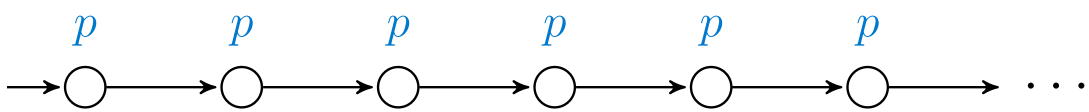


$p\,\mathtt{U}\,q$

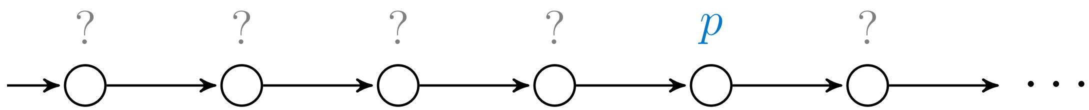
$p\,\mathtt{R}\,q$

LTL: Semántica

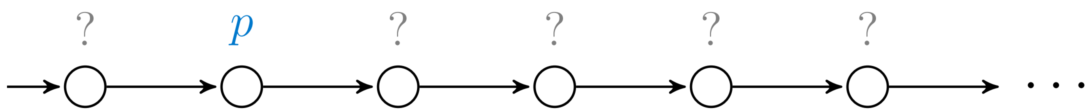
$\mathtt{G}\ p$



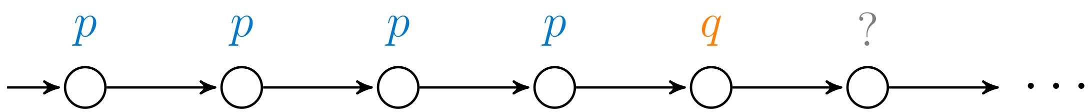
$\mathtt{F}\ p$



$\mathtt{X}\ p$



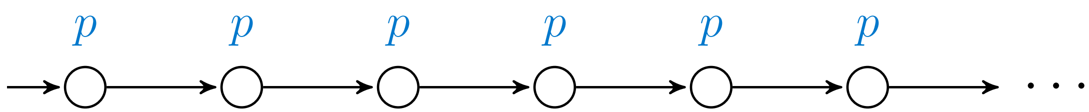
$p\ \mathtt{U}\ q$



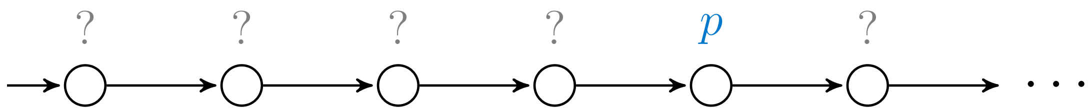
$p\ \mathtt{R}\ q$

LTL: Semántica

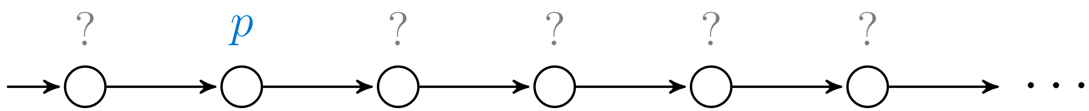
$\mathtt{G}\, p$



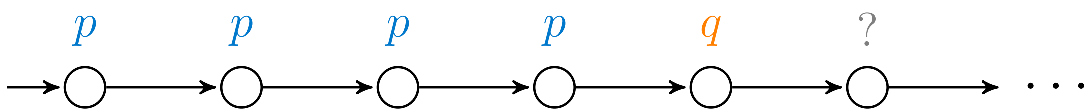
$\mathtt{F}\, p$

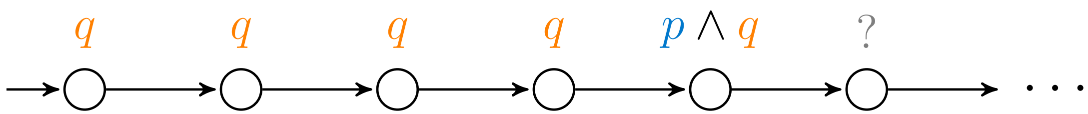


$\mathtt{X}\, p$



$p\ \mathtt{U}\ q$

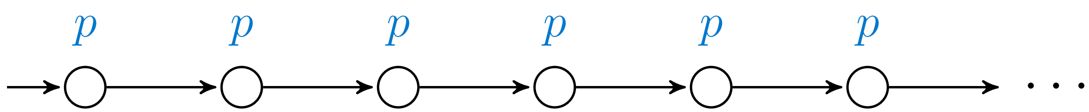




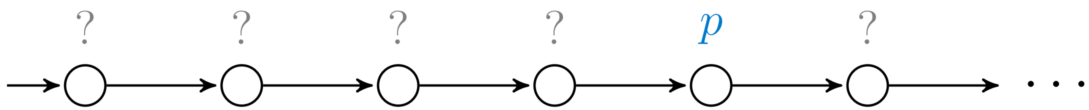
$p\ \mathtt{R}\ q$

LTL: Semántica

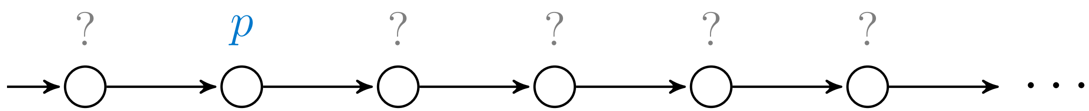
$\mathtt{G}\ p$



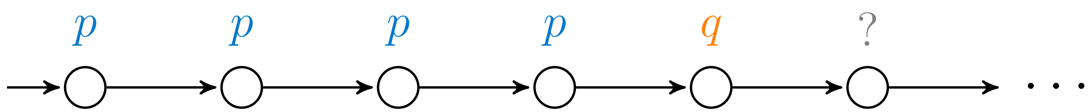
$\mathtt{F}\ p$



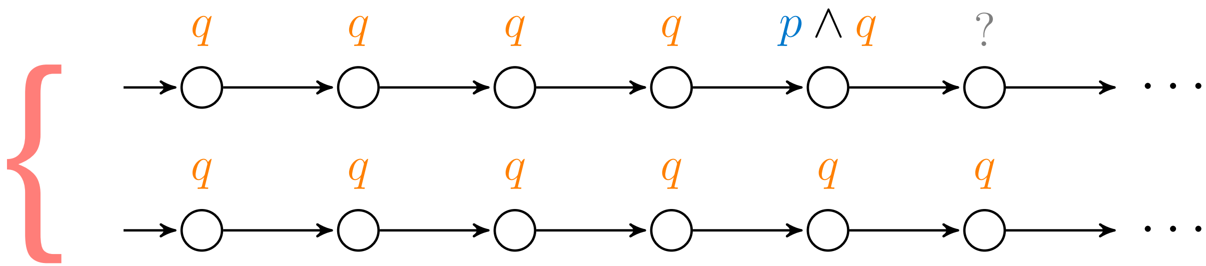
$\mathtt{X}\ p$



$p\ \mathtt{U}\ q$



$p\ \mathtt{R}\ q$



Parentesis:
Semántica de la lógica proposicional

Recordemos: Dado una fórmula proposicional, todo modelo de ésta puede verse como un subconjunto de proposiciones atómicas $A \subseteq \mathsf{PA}$ donde las fórmulas de ese conjunto se hacen verdaderos.

Formalmente, diremos que

> una fórmula proposicional ϕ se satisface en $A \subseteq \mathsf{PA}$ si $A \models \phi$.

Esta última noción se define inductivamente como sigue:

$$
\begin{array}{lll}
A \models p & \text{sii} & p \in A \\
A \models \neg\phi & \text{sii} & A \not\models \phi \\
A \models \phi \wedge \psi & \text{sii} & A \models \phi \text{ y } A \models \psi
\end{array}
$$

LTL: Semántica

LTL especifica el cambio de la validez de las proposiciones acorde avanza el tiempo

Entonces: Un fórmula LTL se satisfará en una secuencia infinita de modelos de formulas proposicionales, es decir, en una secuencia de subconjuntos de PA, denominado traza.

Es decir:

> una fórmula LTL ϕ se satisface en una traza $\sigma \in (\mathcal{P}(\mathsf{PA}))^\omega$ por $\sigma \models \phi$.

Esta última noción se define inductivamente como sigue:

LTL: Semántica

$\sigma \models p$

$\sigma \models \neg\phi$

$\sigma \models \phi \wedge \psi$

$\sigma \models \mathtt{X}\,\phi$

$\sigma \models \phi\,\mathtt{U}\,\psi$

CONICET

UNC

LTL: Semántica

$\sigma \models p$ sii $p \in \sigma(0)$ para todo $p \in$ PA

$\sigma \models \neg\phi$

$\sigma \models \phi \wedge \psi$

$\sigma \models \mathtt{X}\,\phi$

$\sigma \models \phi\,\mathtt{U}\,\psi$

LTL: Semántica

$\sigma \models p$	sii	$p \in \sigma(0)$	para todo $p \in$ PA
$\sigma \models \neg\phi$	sii	$\sigma \not\models \phi$	
$\sigma \models \phi \wedge \psi$			
$\sigma \models \mathtt{X}\,\phi$			
$\sigma \models \phi\,\mathtt{U}\,\psi$			

LTL: Semántica

$\sigma \models p$	sii	$p \in \sigma(0)$ para todo $p \in$ PA
$\sigma \models \neg\phi$	sii	$\sigma \not\models \phi$
$\sigma \models \phi \wedge \psi$	sii	$\sigma \models \phi$ y $\sigma \models \psi$
$\sigma \models \mathtt{X}\,\phi$		
$\sigma \models \phi\,\mathtt{U}\,\psi$		

LTL: Semántica

$\sigma \models p$ sii $p \in \sigma(0)$ para todo $p \in$ PA

$\sigma \models \neg\phi$ sii $\sigma \not\models \phi$

$\sigma \models \phi \wedge \psi$ sii $\sigma \models \phi$ y $\sigma \models \psi$

$\sigma \models \mathtt{X}\,\phi$ sii $\sigma[1..] \models \phi$

$\sigma \models \phi\,\mathtt{U}\,\psi$

$\sigma[i..]$ denota el prefijo i-ésimo de σ.

LTL: Semántica

$\sigma \models p$	sii	$p \in \sigma(0)$ para todo $p \in$ PA
$\sigma \models \neg\phi$	sii	$\sigma \not\models \phi$
$\sigma \models \phi \wedge \psi$	sii	$\sigma \models \phi$ y $\sigma \models \psi$
$\sigma \models \mathtt{X}\,\phi$	sii	$\sigma[1..] \models \phi$
$\sigma \models \phi\,\mathtt{U}\,\psi$	sii	$\exists\, i : i \geq 0 : \sigma[i..] \models \psi$ y $\forall\, j : 0 \leq j < i : \sigma[j..] \models \phi$

$\sigma[i..]$ denota el prefijo i-ésimo de σ.

CONICET

UNC

LTL: Semántica

$\sigma \models \mathtt{F}\, \phi$

$\sigma \models \mathtt{G}\, \phi$

$\sigma \models \phi \,\mathtt{R}\, \psi$

LTL: Semántica

$\sigma \models \mathtt{F}\,\phi$ sii $\exists\, i : i \geq 0 : \sigma[i..] \models \phi$

$\sigma \models \mathtt{G}\,\phi$

$\sigma \models \phi\,\mathtt{R}\,\psi$

CONICET

UNC

LTL: Semántica

$\sigma \models \mathtt{F}\,\phi$ sii $\exists\, i : i \geq 0 : \sigma[i..] \models \phi$

$\sigma \models \mathtt{G}\,\phi$ sii $\forall\, i : i \geq 0 : \sigma[i..] \models \phi$

$\sigma \models \phi\,\mathtt{R}\,\psi$

LTL: Semántica

$\sigma \models \mathtt{F}\,\phi$ sii $\exists\, i : i \geq 0 : \sigma[i..] \models \phi$

$\sigma \models \mathtt{G}\,\phi$ sii $\forall\, i : i \geq 0 : \sigma[i..] \models \phi$

$\sigma \models \phi\,\mathtt{R}\,\psi$ sii $\forall\, i : i \geq 0 : \sigma[i..] \models \psi$ o
$\exists\, j : 0 \leq j < i : \sigma[j..] \models \phi$

LTL: Semántica

$\sigma \models \mathtt{F}\,\phi$ sii $\exists\, i : i \geq 0 : \sigma[i..] \models \phi$

$\sigma \models \mathtt{G}\,\phi$ sii $\forall\, i : i \geq 0 : \sigma[i..] \models \phi$

$\sigma \models \phi\,\mathtt{R}\,\psi$ sii $\forall\, i : i \geq 0 : \sigma[i..] \models \psi$ o
$\exists\, j : 0 \leq j < i : \sigma[j..] \models \phi$

sii $\forall\, i : i \geq 0 : \sigma[i..] \not\models \psi$ implica
$\exists\, j : 0 \leq j < i : \sigma[j..] \models \phi$

LTL: Semántica

$\sigma \models \mathtt{F}\,\phi$ sii $\exists\, i : i \geq 0 : \sigma[i..] \models \phi$ $\qquad \mathtt{F}\,\phi \;=\; \mathit{true}\,\mathtt{U}\,\phi$

$\sigma \models \mathtt{G}\,\phi$ sii $\forall\, i : i \geq 0 : \sigma[i..] \models \phi$

$\sigma \models \phi\,\mathtt{R}\,\psi$ sii $\forall\, i : i \geq 0 : \sigma[i..] \models \psi$ o
$\exists\, j : 0 \leq j < i : \sigma[j..] \models \phi$

sii $\forall\, i : i \geq 0 : \sigma[i..] \not\models \psi$ implica
$\exists\, j : 0 \leq j < i : \sigma[j..] \models \phi$

CONICET

UNC

LTL: Semántica

$\sigma \models \mathtt{F}\,\phi$ sii $\exists\, i : i \geq 0 : \sigma[i..] \models \phi$ $\qquad \mathtt{F}\,\phi \;=\; \mathit{true}\,\mathtt{U}\,\phi$

$\sigma \models \mathtt{G}\,\phi$ sii $\forall\, i : i \geq 0 : \sigma[i..] \models \phi$ $\qquad \mathtt{G}\,\phi \;=\; \neg\,\mathtt{F}\,\neg\phi$

$\sigma \models \phi\,\mathtt{R}\,\psi$ sii $\forall\, i : i \geq 0 : \sigma[i..] \models \psi$ o
$\exists\, j : 0 \leq j < i : \sigma[j..] \models \phi$

sii $\forall\, i : i \geq 0 : \sigma[i..] \not\models \psi$ implica
$\exists\, j : 0 \leq j < i : \sigma[j..] \models \phi$

CONICET

UNC

LTL: Semántica

$\sigma \models \mathtt{F}\,\phi$ sii $\exists\, i : i \geq 0 : \sigma[i..] \models \phi$ $\quad$ $\mathtt{F}\,\phi \;=\; \mathit{true}\,\mathtt{U}\,\phi$

$\sigma \models \mathtt{G}\,\phi$ sii $\forall\, i : i \geq 0 : \sigma[i..] \models \phi$ $\quad$ $\mathtt{G}\,\phi \;=\; \neg\,\mathtt{F}\,\neg\phi$

$\sigma \models \phi\,\mathtt{R}\,\psi$ sii $\forall\, i : i \geq 0 : \sigma[i..] \models \psi$ o $\exists\, j : 0 \leq j < i : \sigma[j..] \models \phi$ $\quad$ $\phi\,\mathtt{R}\,\psi \;=\; \neg(\neg\phi\,\mathtt{U}\,\neg\psi)$

sii $\forall\, i : i \geq 0 : \sigma[i..] \not\models \psi$ implica $\exists\, j : 0 \leq j < i : \sigma[j..] \models \phi$

CONICET

UNC

LTL: Semántica

Un modelo de una fórmula LTL es un conjunto de trazas que satisfacen dicha fórmula

Es decir, $M \subseteq (\mathcal{P}(\mathsf{PA}))^\omega$ es modelo de ϕ, denotado con $M \models \phi$, si para todo $\sigma \in (\mathcal{P}(\mathsf{PA}))^\omega$,

$$\sigma \in M \ \text{ implies } \ \sigma \models \phi$$

El lenguaje de ϕ es su modelo más grande. Formalmente:

$$\mathcal{L}(\phi) = \{\sigma \in (\mathcal{P}(\mathsf{PA}))^\omega \mid \sigma \models \phi\}$$

Por consiguiente, M es modelo de ϕ si

CONICET

UNC

LTL: Semántica

Un modelo de una fórmula LTL es un conjunto de trazas que satisfacen dicha fórmula

Es decir, $M \subseteq (\mathcal{P}(\mathsf{PA}))^\omega$ es modelo de ϕ, denotado con $M \models \phi$, si para todo $\sigma \in (\mathcal{P}(\mathsf{PA}))^\omega$,

$$\sigma \in M \text{ implies } \sigma \models \phi$$

El lenguaje de ϕ es su modelo más grande. Formalmente:

$$\mathcal{L}(\phi) = \{\sigma \in (\mathcal{P}(\mathsf{PA}))^\omega \mid \sigma \models \phi\}$$

Por consiguiente, M es modelo de ϕ si $M \subseteq \mathcal{L}(\phi)$.

CONICET UNC

Algunas leyes

dualidad

$$\neg\,\mathtt{X}\,\phi \equiv \mathtt{X}\,\neg\phi$$

$$\neg\,\mathtt{F}\,\phi \equiv \mathtt{G}\,\neg\phi$$

$$\neg\,\mathtt{G}\,\phi \equiv \mathtt{F}\,\neg\phi$$

absorción

$$\mathtt{F}\,\mathtt{G}\,\mathtt{F}\,\phi \equiv \mathtt{G}\,\mathtt{F}\,\phi$$

$$\mathtt{G}\,\mathtt{F}\,\mathtt{G}\,\phi \equiv \mathtt{F}\,\mathtt{G}\,\phi$$

distributividad

$$\mathtt{F}\,\phi \vee \mathtt{F}\,\psi \equiv \mathtt{F}(\phi \vee \psi)$$

$$\mathtt{G}\,\phi \wedge \mathtt{G}\,\psi \equiv \mathtt{G}(\phi \wedge \psi)$$

$$\mathtt{X}\,\phi\,\mathtt{U}\,\mathtt{X}\,\psi \equiv \mathtt{X}(\phi\,\mathtt{U}\,\psi)$$

idempotencia

$$\mathtt{F}\,\mathtt{F}\,\phi \equiv \mathtt{F}\,\phi$$

$$\mathtt{G}\,\mathtt{G}\,\phi \equiv \mathtt{G}\,\phi$$

$$\phi\,\mathtt{U}\,(\phi\,\mathtt{U}\,\psi) \equiv \phi\,\mathtt{U}\,\psi$$

$$(\phi\,\mathtt{U}\,\psi)\,\mathtt{U}\,\psi \equiv \phi\,\mathtt{U}\,\psi$$

expansión

$$\mathtt{F}\,\phi \equiv \phi \vee \mathtt{X}\,\mathtt{F}\,\phi$$

$$\mathtt{G}\,\phi \equiv \phi \wedge \mathtt{X}\,\mathtt{G}\,\phi$$

$$\phi\,\mathtt{U}\,\psi \equiv \psi \vee (\phi \wedge \mathtt{X}(\phi\,\mathtt{U}\,\psi))$$

Especificación de propiedades

Safety: "Nada malo va a pasar"

- Son las propiedades cuya violación es evidenciada por un testigo finito
- "Si algo malo pasó en una ejecución infinita entonces seguro pasó en algún fragmento finito de esta"

Liveness: "Algo bueno va a pasar"

- Son las propiedades cuya violación no puede ser evidenciada por un testigo finito
- "No importa que haya pasado en un fragmento finito de ejecución, siempre es posible que lo bueno venga después"

Especificación de propiedades

Safety: "Nada malo va a pasar"

- Son las propiedades cuya violación es evidenciada por un testigo finito
- "Si algo malo pasó en una ejecución infinita entonces seguro pasó en algún fragmento finito de esta"

Liveness: "Algo bueno va a pasar"

- Son las propiedades cuya violación no puede ser evidenciada por un testigo finito
- "No importa que haya pasado en un fragmento finito de ejecución, siempre es posible que lo bueno venga después"

Toda propiedad se puede expresar como la conjunción de una propiedad de safety y otra de liveness

Especificación de propiedades
Ejemplos

Deadlock:

- El sistema no tiene deadlock

Terminación:

- Está garantizado que el programa termine

Exclusión mutua:

- No puede ocurrir que dos procesos estén en la región crítica a la vez

Respuesta:

- Todo mensaje enviado se recibe en algún momento

Especificación de propiedades
Ejemplos

Deadlock:

- El sistema no tiene deadlock

$\mathtt{G}\,\neg deadlock$

Terminación:

- Está garantizado que el programa termine

Exclusión mutua:

- No puede ocurrir que dos procesos estén en la región crítica a la vez

Respuesta:

- Todo mensaje enviado se recibe en algún momento

Especificación de propiedades
Ejemplos

Deadlock:

- El sistema no tiene deadlock

SAFETY

Terminación:

- Está garantizado que el programa termine

Exclusión mutua:

- No puede ocurrir que dos procesos estén en la región crítica a la vez

Respuesta:

- Todo mensaje enviado se recibe en algún momento

$\mathtt{G}\,\neg \mathit{deadlock}$

Especificación de propiedades
Ejemplos

Deadlock:

- El sistema no tiene deadlock — $\mathtt{G}\,\neg\mathit{deadlock}$

SAFETY

Terminación:

- Está garantizado que el programa termine — $\mathtt{F}\,\mathit{end}$

Exclusión mutua:

- No puede ocurrir que dos procesos estén en la región crítica a la vez

Respuesta:

- Todo mensaje enviado se recibe en algún momento

CONICET

UNC

Especificación de propiedades
Ejemplos

Deadlock:

- El sistema no tiene deadlock — $\mathtt{G}\,\neg deadlock$

SAFETY

Terminación:

- Está garantizado que el programa termine — $\mathtt{F}\,end$

LIVENESS

Exclusión mutua:

- No puede ocurrir que dos procesos estén en la región crítica a la vez

Respuesta:

- Todo mensaje enviado se recibe en algún momento

CONICET

UNC

Especificación de propiedades
Ejemplos

Deadlock:

- El sistema no tiene deadlock

$\mathtt{G}\,\neg \mathit{deadlock}$

SAFETY

Terminación:

- Está garantizado que el programa termine

$\mathtt{F}\;\mathit{end}$

LIVENESS

Exclusión mutua:

- No puede ocurrir que dos procesos estén en la región crítica a la vez

$\mathtt{G}\,\neg(\mathit{crit}_1 \wedge \mathit{crit}_2)$

Respuesta:

- Todo mensaje enviado se recibe en algún momento

Especificación de propiedades
Ejemplos

Deadlock:

- El sistema no tiene deadlock — $\mathtt{G}\,\neg deadlock$

SAFETY

Terminación:

- Está garantizado que el programa termine — $\mathtt{F}\ end$

LIVENESS

Exclusión mutua:

- No puede ocurrir que dos procesos estén en la región crítica a la vez — $\mathtt{G}\,\neg(crit_1 \wedge crit_2)$

SAFETY

Respuesta:

- Todo mensaje enviado se recibe en algún momento

Especificación de propiedades
Ejemplos

Deadlock:

- El sistema no tiene deadlock — SAFETY — $\mathtt{G}\,\neg deadlock$

Terminación:

- Está garantizado que el programa termine — LIVENESS — $\mathtt{F}\,end$

Exclusión mutua:

- No puede ocurrir que dos procesos estén en la región crítica a la vez — SAFETY — $\mathtt{G}\,\neg(crit_1 \wedge crit_2)$

Respuesta:

- Todo mensaje enviado se recibe en algún momento — $\mathtt{G}(snd_m \Rightarrow \mathtt{F}\,rcv_m)$

CONICET

UNC

Especificación de propiedades
Ejemplos

Deadlock:

- El sistema no tiene deadlock — $\mathtt{G}\,\neg deadlock$

SAFETY

Terminación:

- Está garantizado que el programa termine — $\mathtt{F}\,end$

LIVENESS

Exclusión mutua:

- No puede ocurrir que dos procesos estén en la región crítica a la vez — $\mathtt{G}\,\neg(crit_1 \wedge crit_2)$

SAFETY

Respuesta:

- Todo mensaje enviado se recibe en algún momento — $\mathtt{G}(snd_m \Rightarrow \mathtt{F}\,rcv_m)$

LIVENESS

$$
P_1 : \texttt{x} := 1 \quad \Big|\Big| \quad P_2 : \begin{array}{l} \texttt{while x} = 0 \\ \texttt{do} \\ \quad \texttt{y} := 0 \\ \texttt{od ;} \\ \texttt{y} := 1 \end{array}
$$

$$\textcolor{red}{P_1} : \texttt{x := } 1 \quad \Big\| \quad \textcolor{red}{P_2} : \begin{array}{l} \texttt{while x } = \ 0 \\ \texttt{do} \\ \quad \texttt{y := } 0 \\ \texttt{od ;} \\ \texttt{y := } 1 \end{array}$$

¿Satisface $\texttt{F}(\mathrm{y} = 1)$?

$$
\begin{array}{l|l}
P_1 : \texttt{x := } 1 & P_2 : \texttt{while x} = 0 \\
 & \quad \texttt{do} \\
 & \qquad \texttt{y := } 0 \\
 & \quad \texttt{od ;} \\
 & \quad \texttt{y := } 1
\end{array}
$$

¿Satisface $\texttt{F}(\texttt{y} = 1)$?

$$
\begin{tikzcd}
{} \arrow[d] \\
\langle P_1 \parallel P_2, (0,0) \rangle \arrow[loop right] \arrow[d] \\
\langle P_2, (1,0) \rangle \arrow[d] \\
\langle \texttt{y := } 1, (1,0) \rangle \arrow[d] \\
\langle \surd, (1,1) \rangle
\end{tikzcd}
$$

```
P1 : x := 1  ||  P2 : while x = 0
                      do
                        y := 0
                      od ;
                      y := 1
```

¿Satisface $\mathtt{F}(\mathtt{y} = 1)$?

$$
\begin{tikzcd}
{} \arrow[d] \\
\langle P_1 \mid\mid P_2, (0,0)\rangle \arrow[loop right] \arrow[d] \\
\langle P_2, (1,0)\rangle \arrow[d] \\
\langle \mathtt{y := 1}, (1,0)\rangle \arrow[d] \\
\langle \surd, (1,1)\rangle
\end{tikzcd}
$$

CONICET

UNC

$\color{red}{P_1}$: `x := 1` $\quad || \quad$ $\color{red}{P_2}$:

```
while x = 0
do
  y := 0
od ;
y := 1
```

$$
\begin{array}{c}
\downarrow \\
\langle \color{red}{P_1} \mid\mid \color{red}{P_2}, (0,0) \rangle \circlearrowleft \\
\downarrow \\
\langle \color{red}{P_2}, (1,0) \rangle \\
\downarrow \\
\langle \mathtt{y := 1}, (1,0) \rangle \\
\downarrow \\
\langle \surd, (1,1) \rangle
\end{array}
$$

¿Satisface $\mathtt{F}(\mathtt{y} = 1)$?

$$\langle \color{red}{P_1} \mid\mid \color{red}{P_2}, (0,0) \rangle \;\; \langle \color{red}{P_1} \mid\mid \color{red}{P_2}, (0,0) \rangle \;\; \langle \color{red}{P_1} \mid\mid \color{red}{P_2}, (0,0) \rangle \;\; \langle \color{red}{P_1} \mid\mid \color{red}{P_2}, (0,0) \rangle \;\; \cdots$$

P_1 : `x := 1` $\|$ P_2 : `while x = 0`
```
do
  y := 0
od ;
y := 1
```

$$
\begin{array}{c}
\downarrow \\
\langle P_1 \parallel P_2, (0,0) \rangle \circlearrowleft \\
\downarrow \\
\langle P_2, (1,0) \rangle \\
\downarrow \\
\langle \texttt{y := 1}, (1,0) \rangle \\
\downarrow \\
\langle \surd, (1,1) \rangle
\end{array}
$$

¿Satisface $\texttt{F}(\texttt{y} = 1)$?

$$\langle P_1 \parallel P_2, (0,0) \rangle \;\; \langle P_1 \parallel P_2, (0,0) \rangle \;\; \langle P_1 \parallel P_2, (0,0) \rangle \;\; \langle P_1 \parallel P_2, (0,0) \rangle \;\; \cdots$$

Necesidad de Fairness

Fairness

"Bajo cierta condición, el evento ocurrirá de manera frecuente"

Fairness

LIVENESS

"Bajo cierta condición, el evento ocurrirá de manera frecuente"

Fairness

LIVENESS

"Bajo cierta condición, el evento ocurrirá de manera frecuente"

Progreso (Fairness incondicional):

"El evento ocurre de manera frecuente"

Weak fairness:

"Si la condición permanece verdadera, el evento ocurrirá de manera frecuente"

Strong fairness:

"Si la condición es frecuentemente verdadera, el evento ocurrirá de manera frecuente"

Fairness

LIVENESS

"Bajo cierta condición, el evento ocurrirá de manera frecuente"

Progreso (Fairness incondicional):

"El evento ocurre de manera frecuente"

$\mathtt{G}\,\mathtt{F}\,evn$

Weak fairness:

"Si la condición permanece verdadera, el evento ocurrirá de manera frecuente"

Strong fairness:

"Si la condición es frecuentemente verdadera, el evento ocurrirá de manera frecuente"

CONICET

UNC

Fairness LIVENESS

"Bajo cierta condición, el evento ocurrirá de manera frecuente"

Progreso (Fairness incondicional):

"El evento ocurre de manera frecuente"

$$\mathtt{G}\,\mathtt{F}\,evn$$

Weak fairness:

"Si la condición <u>permanece</u> verdadera, el evento ocurrirá de manera frecuente"

$$\mathtt{F}\,\mathtt{G}\,cnd \Rightarrow \mathtt{G}\,\mathtt{F}\,evn$$

Strong fairness:

"Si la condición es <u>frecuentemente</u> verdadera, el evento ocurrirá de manera frecuente"

CONICET UNC

Fairness

LIVENESS

"Bajo cierta condición, el evento ocurrirá de manera frecuente"

Progreso (Fairness incondicional): $\mathtt{G}\,\mathtt{F}\,evn$

"El evento ocurre de manera frecuente"

Weak fairness: $\mathtt{F}\,\mathtt{G}\,cnd \Rightarrow \mathtt{G}\,\mathtt{F}\,evn$

"Si la condición permanece verdadera, el evento ocurrirá de manera frecuente"

Strong fairness: $\mathtt{G}\,\mathtt{F}\,cnd \Rightarrow \mathtt{G}\,\mathtt{F}\,evn$

"Si la condición es frecuentemente verdadera, el evento ocurrirá de manera frecuente"

CONICET

UNC

Fairness

P_1 : `x :=` 1 $\parallel$ P_2 : `while x` $=$ 0

```
do
  y := 0
od ;
y := 1
```

Fairness

P_1 : `x := 1` $\|$ P_2 :

```
while x = 0
do
  y := 0
od ;
y := 1
```

$$\downarrow$$
$$\langle P_1 \;||\; P_2, (0,0)\rangle$$
$$\downarrow$$
$$\langle P_2, (1,0)\rangle$$
$$\downarrow$$
$$\langle \texttt{y := } 1, (1,0)\rangle$$
$$\downarrow$$
$$\langle \surd, (1,1)\rangle$$

Fairness

P_1 : `x := 1` $\quad \| \quad$ P_2 :
```
while x = 0
do
  y := 0
od ;
y := 1
```

$$
\begin{tikzcd}
{} \arrow[d] \\
\langle P_1 \;\|\; P_2, (0,0)\rangle \arrow[loop right] \arrow[d] \\
\langle P_2, (1,0)\rangle \arrow[d] \\
\langle \mathtt{y} := 1, (1,0)\rangle \arrow[d] \\
\langle \surd, (1,1)\rangle
\end{tikzcd}
$$

¿Satisface $(\mathtt{F}\,\mathtt{G}\,\mathit{habilitada} \Rightarrow \mathtt{G}\,\mathtt{F}\,\mathit{ejecutó}) \Rightarrow \mathtt{F}(\mathtt{y} = 1)$?

CONICET

UNC

Fairness

P_1 : `x := 1` $\;\|\;$ P_2 :

```
while x = 0
do
  y := 0
od ;
y := 1
```

$$
\begin{array}{rc}
 & \downarrow \\
\textit{habilitada} & \langle P_1 \;||\; P_2, (0,0)\rangle \\
 & \downarrow \\
\textit{ejecutó} & \langle P_2, (1,0)\rangle \\
 & \downarrow \\
 & \langle \mathtt{y := 1}, (1,0)\rangle \\
 & \downarrow \\
 & \langle \surd, (1,1)\rangle
\end{array}
$$

¿Satisface $(\mathtt{F}\,\mathtt{G}\,\textit{habilitada} \Rightarrow \mathtt{G}\,\mathtt{F}\,\textit{ejecutó}) \Rightarrow \mathtt{F}(\mathtt{y} = 1)$?

Fairness

P_1 : `x := 1` $\quad || \quad$ P_2 :

```
while x = 0
do
  y := 0
od ;
y := 1
```

$$
\begin{tikzcd}
 & {} \arrow[d] \\
\textit{habilitada} & \langle P_1 \mid\mid P_2, (0,0) \rangle \arrow[loop right] \arrow[d] \\
\textit{ejecutó} & \langle P_2, (1,0) \rangle \arrow[d] \\
 & \langle \texttt{y := 1}, (1,0) \rangle \arrow[d] \\
 & \langle \surd, (1,1) \rangle
\end{tikzcd}
$$

¿Satisface $\underbrace{(\mathtt{F}\,\mathtt{G}\,\textit{habilitada} \Rightarrow \mathtt{G}\,\mathtt{F}\,\textit{ejecutó})}_{\text{Weak Fairness}} \Rightarrow \mathtt{F}(\mathtt{y} = 1)$?

Fairness

P_1 : `x := 1` $\;\Big|\Big|\;$ P_2 :

```
while x = 0
do
  y := 0
od ;
y := 1
```

$$
\begin{array}{rc}
 & \downarrow \\
\textit{habilitada} & \langle P_1 \mid\mid P_2, (0,0)\rangle \circlearrowleft \\
 & \downarrow \\
\textit{ejecutó} & \langle P_2, (1,0)\rangle \\
 & \downarrow \\
 & \langle \mathtt{y} := 1, (1,0)\rangle \\
 & \downarrow \\
 & \langle \surd, (1,1)\rangle
\end{array}
$$

¿Satisface $\underbrace{(\mathtt{F}\,\mathtt{G}\,\textit{habilitada} \Rightarrow \mathtt{G}\,\mathtt{F}\,\textit{ejecutó})}_{\text{Weak Fairness}} \Rightarrow \mathtt{F}(\mathtt{y} = 1)$?

CONICET

UNC

Ejercicio 2: Demuestre usando la semántica de LTL que

$$\mathtt{G}(\mathit{llueve} \Rightarrow \mathtt{F}\,\neg \mathit{llueve}) \;=\; \mathtt{G}\,\mathtt{F}\,\neg \mathit{llueve}.$$

Ejercicio 3: Sabemos que $\phi\,\mathtt{R}\,\psi \;=\; \neg(\neg\phi\,\mathtt{U}\,\neg\psi)$. Demuestre usando las leyes que

$$\phi\,\mathtt{R}\,\psi \equiv \psi \wedge (\phi \vee \mathtt{X}(\phi\,\mathtt{R}\,\psi)).$$

*** Ejercicio 4:** Demuestre que Strong Fairness implica Weak Fairness.

** Ejercicio estrella: Suma pero no resta* 🙂

CONICET

UNC

Un simple lenguaje concurrente

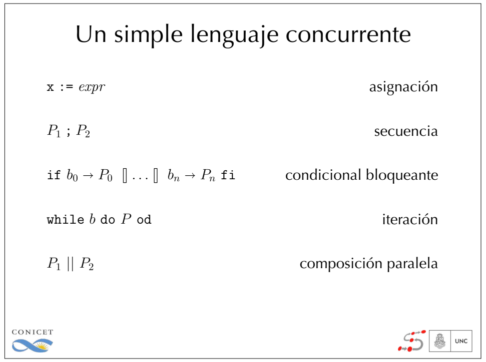
$\texttt{x := } expr$	asignación
$P_1 \ ; \ P_2$	secuencia
$\texttt{if } b_0 \to P_0 \ [\!] \ldots [\!] \ b_n \to P_n \texttt{ fi}$	condicional bloqueante
$\texttt{while } b \texttt{ do } P \texttt{ od}$	iteración
$P_1 \ \| \ P_2$	composición paralela

CONICET

UNC

CONICET

UNC



$$
\begin{array}{l}
\texttt{while true do} \\
\quad \texttt{if} \\
\qquad [\!]\ (x > 0) \wedge (y = 0) \rightarrow \\
\qquad\qquad x \mathrel{\texttt{:=}} x - 1 \\
\qquad [\!]\ (x < 2) \wedge (y = 0) \rightarrow \\
\qquad\qquad x \mathrel{\texttt{:=}} x + 1 \\
\quad \texttt{fi} \\
\texttt{od}
\end{array}
\quad \Bigg\| \quad
\begin{array}{l}
\texttt{while true do} \\
\quad y \mathrel{\texttt{:=}} 1\ ; \\
\quad y \mathrel{\texttt{:=}} 0 \\
\texttt{od}
\end{array}
$$

Un simple lenguaje concurrente

x := $expr$	asignación
P_1 ; P_2	secuencia
if $b_0 \to P_0$ $[\!]$ ... $[\!]$ $b_n \to P_n$ fi	condicional bloqueante
while b do P od	iteración
$P_1 \parallel P_2$	composición paralela

CONICET

$$P$$

CONICET

UNC

Un simple lenguaje concurrente

$\mathtt{x} := expr$

$P_1 \; ; \; P_2$

$\mathtt{if}\ b_0 \to P_0\ [] \ldots []\ b_n$

$\mathtt{while}\ b\ \mathtt{do}\ P\ \mathtt{od}$

$P_1 \parallel P_2$

LTL: Sintaxis

$\phi, \psi ::=$

$$p \quad | \quad \neg\phi \quad | \quad \phi \wedge \psi \quad | \quad \phi \vee \psi \quad | \quad \phi \Rightarrow \psi \quad | \qquad \text{(op. proposicionales)}$$

$$\mathtt{X}\,\phi \quad | \quad \mathtt{F}\,\phi \quad | \quad \mathtt{G}\,\phi \quad | \quad \phi\,\mathtt{U}\,\psi \quad | \quad \phi\,\mathtt{R}\,\psi \qquad \text{(op. temporales)}$$

donde $p \in$ PA es cualquier variable proporsional (o proposición atómica)

Los operadores básicos son $\neg$, $\vee$, $\mathtt{X}$, y $\mathtt{U}$.

Los otros operadores se definen en término de estos.

CONICET

UNC

CONICET

UNC

Un simple lenguaje concurrente

$\texttt{x := } expr$	asignación
$P_1\ ;\ P_2$	secuencia
$\texttt{if } b_0 \to P_0\ [\!]\ \dots\ [\!]\ b_n \to P_n \texttt{ fi}$	condicional bloqueante
$\texttt{while } b \texttt{ do } P \texttt{ od}$	iteración
$P_1 \parallel P_2$	composición paralela

$$P$$

LTL: Sintaxis

$\phi, \psi ::=$

$p \ \mid\ \neg\phi \ \mid\ \phi \wedge \psi \ \mid\ \phi \vee \psi \ \mid\ \phi \Rightarrow \psi \ \mid$ (op. proposicionales)

$\texttt{X}\,\phi \ \mid\ \texttt{F}\,\phi \ \mid\ \texttt{G}\,\phi \ \mid\ \phi\,\texttt{U}\,\psi \ \mid\ \phi\,\texttt{R}\,\psi$ (op. temporales)

donde $p \in \mathrm{PA}$ es cualquier variable proporsional (o proposición atómica)

Los operadores básicos son $\neg$, $\vee$, $\texttt{X}$, y $\texttt{U}$.

Los otros operadores se definen en término de estos.

$$\texttt{G}((y = 1) \Rightarrow \texttt{X}(y = 0))$$

Un simple lenguaje concurrente

`x` := *expr*	asignación
P_1 ; P_2	secuencia
`if` $b_0 \to P_0$ $[\!]$ ... $[\!]$ $b_n \to P_n$ `fi`	condicional bloqueante
`while` b `do` P `od`	iteración
$P_1 \parallel P_2$	composición paralela

$P \qquad \phi$

LTL: Sintaxis

$$\begin{array}{llllllllll} \phi, \psi ::= \\ \quad p & \mid & \neg\phi & \mid & \phi \wedge \psi & \mid & \phi \vee \psi & \mid & \phi \Rightarrow \psi & \mid \quad \text{(op. proposicionales)} \\ \quad \mathtt{X}\,\phi & \mid & \mathtt{F}\,\phi & \mid & \mathtt{G}\,\phi & \mid & \phi\,\mathtt{U}\,\psi & \mid & \phi\,\mathtt{R}\,\psi & \quad \text{(op. temporales)} \end{array}$$

donde $p \in$ PA es cualquier variable proporsional (o proposición atómica)

Los operadores básicos son $\neg$, $\vee$, `X`, y `U`.

Los otros operadores se definen en término de estos.

Un simple lenguaje concurrente

`x :=` $expr$	asignación
P_1 ; P_2	secuencia
`if` $b_0 \to P_0 \; [\!] \ldots [\!] \; b_n \to P_n$ `fi`	condicional bloqueante
`while` b `do` P `od`	iteración
$P_1 \parallel P_2$	composición paralela

$$P \overset{?}{\models} \phi$$

LTL: Sintaxis

$\phi, \psi ::=$

$p \;|\; \neg\phi \;|\; \phi \wedge \psi \;|\; \phi \vee \psi \;|\; \phi \Rightarrow \psi \;|$ (op. proposicionales)

$\mathtt{X}\,\phi \;|\; \mathtt{F}\,\phi \;|\; \mathtt{G}\,\phi \;|\; \phi\,\mathtt{U}\,\psi \;|\; \phi\,\mathtt{R}\,\psi$ (op. temporales)

donde $p \in$ PA es cualquier variable proporsional (o proposición atómica)

Los operadores básicos son $\neg$, $\vee$, X, y U.

Los otros operadores se definen en término de estos.

Un simple lenguaje concurrente

$\mathtt{x} := expr$

$P_1 \ ; \ P_2$

$\mathtt{if}\ b_0 \to P_0\ [\!]\ \ldots\ [\!]\ b_n$

$\mathtt{while}\ b\ \mathtt{do}\ P\ \mathtt{od}$

$P_1 \ || \ P_2$

LTL: Sintaxis

(op. proposicionales)

(op. temporales)

(o proposición atómica)

os.

Semántica del lenguaje de modelado

$$\langle x \mathrel{:=} e, \mu \rangle \longrightarrow \langle \surd, \mu[x \mapsto \mu(e)] \rangle$$

$$\frac{\langle P_1, \mu \rangle \longrightarrow \langle P_1', \mu' \rangle \qquad P_1' \neq \surd}{\langle P_1 \ ; \ P_2, \mu \rangle \longrightarrow \langle P_1' \ ; \ P_2, \mu' \rangle}$$

$$\frac{\langle P_1, \mu \rangle \longrightarrow \langle P_1', \mu' \rangle \qquad P_2 \neq \surd}{\langle P_1 \ || \ P_2, \mu \rangle \longrightarrow \langle P_1' \ || \ P_2, \mu' \rangle}$$

$$\frac{\langle P_1, \mu \rangle \longrightarrow \langle \surd, \mu' \rangle}{\langle P_1 \ ; \ P_2, \mu \rangle \longrightarrow \langle P_2, \mu' \rangle}$$

$$\frac{\langle P_2, \mu \rangle \longrightarrow \langle P_2', \mu' \rangle \qquad P_1 \neq \surd}{\langle P_1 \ || \ P_2, \mu \rangle \longrightarrow \langle P_1 \ || \ P_2', \mu' \rangle}$$

$$\frac{\langle P_j, \mu \rangle \longrightarrow \langle P_j', \mu' \rangle \qquad \mu(b_j) \text{ holds}}{\langle \mathtt{if}\ b_0 \to P_0\ [\!]\ \ldots\ [\!]\ b_n \to P_n\ \mathtt{fi}, \mu \rangle \longrightarrow \langle P_j', \mu' \rangle}$$

$$\frac{\langle P_1, \mu \rangle \longrightarrow \langle P_1', \mu' \rangle}{\langle P_1 \ || \ \surd, \mu \rangle \longrightarrow \langle P_1', \mu' \rangle}$$

$$\frac{\langle P; \mathtt{while}\ b\ \mathtt{do}\ P\ \mathtt{od}, \mu \rangle \longrightarrow \langle P', \mu' \rangle \qquad \mu(b) \text{ holds}}{\langle \mathtt{while}\ b\ \mathtt{do}\ P\ \mathtt{od}, \mu \rangle \longrightarrow \langle P', \mu' \rangle}$$

$$\frac{\langle P_2, \mu \rangle \longrightarrow \langle P_2', \mu' \rangle}{\langle \surd \ || \ P_2, \mu \rangle \longrightarrow \langle P_2', \mu' \rangle}$$

$$\frac{\neg \mu(b) \text{ holds}}{\langle \mathtt{while}\ b\ \mathtt{do}\ P\ \mathtt{od}, \mu \rangle \longrightarrow \langle \surd, \mu \rangle}$$

CONICET

UNC

Un simple lenguaje concurrente

`x := `$expr$	asignación
P_1 ; P_2	secuencia
`if` $b_0 \to P_0$ $[\!]\ldots[\!]$ $b_n \to P_n$ `fi`	condicional bloqueante
`while` b `do` P `od`	iteración
$P_1 \mid\mid P_2$	composición paralela

LTL: Sintaxis

$\phi, \psi ::=$

$p \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \phi \Rightarrow \psi \mid$ (op. proposicionales)

$\mathtt{X}\,\phi \mid \mathtt{F}\,\phi \mid \mathtt{G}\,\phi \mid \phi\,\mathtt{U}\,\psi \mid \phi\,\mathtt{R}\,\psi$ (op. temporales)

donde $p \in \mathrm{PA}$ es cualquier variable proporsional (o proposición atómica)

Los operadores básicos son $\neg$, $\vee$, $\mathtt{X}$, y $\mathtt{U}$.

Los otros operadores se definen en término de estos.

Semántica del lenguaje de modelado

$$\langle x := e, \mu\rangle \longrightarrow \langle\surd, \mu[x \mapsto \mu(e)]\rangle$$

$$\frac{\langle P_1,\mu\rangle \longrightarrow \langle P_1',\mu'\rangle \quad P_2 \neq \surd}{\langle P_1 \mid\mid P_2,\mu\rangle \longrightarrow \langle P_1' \mid\mid P_2,\mu'\rangle}$$

$$\frac{\langle P_2,\mu\rangle \longrightarrow \langle P_2',\mu'\rangle \quad P_1 \neq \surd}{\langle P_1 \mid\mid P_2,\mu\rangle \longrightarrow \langle P_1 \mid\mid P_2',\mu'\rangle}$$

$$\frac{\langle P_1,\mu\rangle \longrightarrow \langle P_1',\mu'\rangle}{\langle P_1 \mid\mid \surd,\mu\rangle \longrightarrow \langle P_1',\mu'\rangle}$$

$$\frac{\langle P_2,\mu\rangle \longrightarrow \langle P_2',\mu'\rangle}{\langle \surd \mid\mid P_2,\mu\rangle \longrightarrow \langle P_2',\mu'\rangle}$$

$$\frac{\langle P_1,\mu\rangle \longrightarrow \langle P_1',\mu'\rangle \quad P_1' \neq \surd}{\langle P_1 \,;\, P_2,\mu\rangle \longrightarrow \langle P_1' \,;\, P_2,\mu'\rangle}$$

$$\frac{\langle P_1,\mu\rangle \longrightarrow \langle \surd,\mu'\rangle}{\langle P_1 \,;\, P_2,\mu\rangle \longrightarrow \langle P_2,\mu'\rangle}$$

$$\frac{\langle P_j,\mu\rangle \longrightarrow \langle P_j',\mu'\rangle \qquad \mu(b_j) \text{ holds}}{\langle \mathtt{if}\ b_0 \to P_0\ [\!]\ldots[\!]\ b_n \to P_n\ \mathtt{fi},\mu\rangle \longrightarrow \langle P_j',\mu'\rangle}$$

$$\frac{\langle P; \mathtt{while}\ b\ \mathtt{do}\ P\ \mathtt{od},\mu\rangle \longrightarrow \langle P',\mu'\rangle \qquad \mu(b) \text{ holds}}{\langle \mathtt{while}\ b\ \mathtt{do}\ P\ \mathtt{od},\mu\rangle \longrightarrow \langle P',\mu'\rangle}$$

$$\frac{\neg\mu(b) \text{ holds}}{\langle \mathtt{while}\ b\ \mathtt{do}\ P\ \mathtt{od},\mu\rangle \longrightarrow \langle \surd,\mu\rangle}$$

$$P \overset{?}{\models} \phi$$

$$\begin{tikzcd}
\arrow[d] & \\
\langle P_1^\infty \mid\mid P_2^\infty, (0,0)\rangle \arrow[r, bend left] \arrow[d, bend left] & \langle P_1^\infty \mid\mid (y := 0 \,;\, P_2^\infty), (0,1)\rangle \arrow[l, bend left] \\
\langle P_1^\infty \mid\mid P_2^\infty, (1,0)\rangle \arrow[r, bend left] \arrow[u, bend left] \arrow[d, bend left] & \langle P_1^\infty \mid\mid (y := 0 \,;\, P_2^\infty), (1,1)\rangle \arrow[l, bend left] \\
\langle P_1^\infty \mid\mid P_2^\infty, (2,0)\rangle \arrow[r, bend left] \arrow[u, bend left] & \langle P_1^\infty \mid\mid (y := 0 \,;\, P_2^\infty), (2,1)\rangle \arrow[l, bend left]
\end{tikzcd}$$

Un simple lenguaje concurrente

`x :=` $expr$	asignación
P_1 ; P_2	secuencia
`if` $b_0 \rightarrow P_0$ $[\!]$ $\ldots$ $[\!]$ $b_n \rightarrow P_n$ `fi`	condicional bloqueante
`while` b `do` P `od`	iteración
$P_1 \parallel P_2$	composición paralela

Semántica del lenguaje de modelado

$$\langle x := e, \mu\rangle \longrightarrow \langle \surd, \mu[x \mapsto \mu(e)]\rangle$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle \quad P_2 \neq \surd}{\langle P_1 \parallel P_2, \mu\rangle \longrightarrow \langle P_1' \parallel P_2, \mu'\rangle}$$

$$\frac{\langle P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle \quad P_1 \neq \surd}{\langle P_1 \parallel P_2, \mu\rangle \longrightarrow \langle P_1 \parallel P_2', \mu'\rangle}$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle}{\langle P_1 \parallel \surd, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle}$$

$$\frac{\langle P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle}{\langle \surd \parallel P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle}$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle \quad P_1' \neq \surd}{\langle P_1 \,;\, P_2, \mu\rangle \longrightarrow \langle P_1' \,;\, P_2, \mu'\rangle}$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle \surd, \mu'\rangle}{\langle P_1 \,;\, P_2, \mu\rangle \longrightarrow \langle P_2, \mu'\rangle}$$

$$\frac{\langle P_j, \mu\rangle \longrightarrow \langle P_j', \mu'\rangle \qquad \mu(b_j) \text{ holds}}{\langle \texttt{if } b_0 \rightarrow P_0 \;[\!]\; \ldots \;[\!]\; b_n \rightarrow P_n \texttt{ fi}, \mu\rangle \longrightarrow \langle P_j', \mu'\rangle}$$

$$\frac{\langle P; \texttt{while } b \texttt{ do } P \texttt{ od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle \qquad \mu(b) \text{ holds}}{\langle \texttt{while } b \texttt{ do } P \texttt{ od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle}$$

$$\frac{\neg\mu(b) \text{ holds}}{\langle \texttt{while } b \texttt{ do } P \texttt{ od}, \mu\rangle \longrightarrow \langle \surd, \mu\rangle}$$

$$P \overset{?}{\models} \phi$$

$$\Big\downarrow$$

$$K(P)$$

LTL: Sintaxis

$\phi, \psi ::=$

p	$\mid \neg\phi$	$\mid \phi \wedge \psi$	$\mid \phi \vee \psi$	$\mid \phi \Rightarrow \psi$ $\mid$	(op. proposicionales)
$\mathtt{X}\,\phi$	$\mid \mathtt{F}\,\phi$	$\mid \mathtt{G}\,\phi$	$\mid \phi\,\mathtt{U}\,\psi$	$\mid \phi\,\mathtt{R}\,\psi$	(op. temporales)

donde $p \in$ PA es cualquier variable proporsional (o proposición atómica)

Los operadores básicos son $\neg$, $\vee$, `X`, y `U`.

Los otros operadores se definen en término de estos.

Un simple lenguaje concurrente

`x := `$expr$	asignación
P_1 ; P_2	secuencia
`if` $b_0 \to P_0 \;[]\; \ldots \;[]\; b_n \to P_n$ `fi`	condicional bloqueante
`while` b `do` P `od`	iteración
$P_1 \parallel P_2$	composición paralela

Semántica del lenguaje de modelado

$$\langle x := e, \mu\rangle \longrightarrow \langle \surd, \mu[x \mapsto \mu(e)]\rangle$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle \quad P_2 \neq \surd}{\langle P_1 \parallel P_2, \mu\rangle \longrightarrow \langle P_1' \parallel P_2, \mu'\rangle}$$

$$\frac{\langle P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle \quad P_1 \neq \surd}{\langle P_1 \parallel P_2, \mu\rangle \longrightarrow \langle P_1 \parallel P_2', \mu'\rangle}$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle}{\langle P_1 \parallel \surd, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle}$$

$$\frac{\langle P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle}{\langle \surd \parallel P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle}$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle \quad P_1' \neq \surd}{\langle P_1 \,;\, P_2, \mu\rangle \longrightarrow \langle P_1' \,;\, P_2, \mu'\rangle}$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle \surd, \mu'\rangle}{\langle P_1 \,;\, P_2, \mu\rangle \longrightarrow \langle P_2, \mu'\rangle}$$

$$\frac{\langle P_j, \mu\rangle \longrightarrow \langle P_j', \mu'\rangle \qquad \mu(b_j) \text{ holds}}{\langle \texttt{if } b_0 \to P_0 \;[]\; \ldots \;[]\; b_n \to P_n \texttt{ fi}, \mu\rangle \longrightarrow \langle P_j', \mu'\rangle}$$

$$\frac{\langle P; \texttt{while } b \texttt{ do } P \texttt{ od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle \qquad \mu(b) \text{ holds}}{\langle \texttt{while } b \texttt{ do } P \texttt{ od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle}$$

$$\frac{\neg\mu(b) \text{ holds}}{\langle \texttt{while } b \texttt{ do } P \texttt{ od}, \mu\rangle \longrightarrow \langle \surd, \mu\rangle}$$

$$P \overset{?}{\models} \phi$$

$$\Big\downarrow$$

$$K(P) \overset{?}{\models} \phi$$

LTL: Sintaxis

$\phi, \psi ::=$

$p \;|\; \neg\phi \;|\; \phi \wedge \psi \;|\; \phi \vee \psi \;|\; \phi \Rightarrow \psi \;|$ (op. proposicionales)

$\mathtt{X}\,\phi \;|\; \mathtt{F}\,\phi \;|\; \mathtt{G}\,\phi \;|\; \phi\,\mathtt{U}\,\psi \;|\; \phi\,\mathtt{R}\,\psi$ (op. temporales)

donde $p \in$ PA es cualquier variable proporsional (o proposición atómica)

Los operadores básicos son $\neg$, $\vee$, `X`, y `U`.

Los otros operadores se definen en término de estos.

Trazas observables

❖ Para hablar de las propiedades de un sistema debemos observar las propiedades de sus estados y su orden temporal en las ejecuciones.

$$\mathcal{L}(K) = \{\sigma \in (\mathcal{P}(\mathsf{PA}))^{\omega} \quad | \; \exists\, \rho \text{ ejecución de } K : \forall\, i \geq 0 : \sigma(i) = \mathcal{L}(\rho(i))\}$$

$$\begin{array}{cccccccccccccc}
\rho & = & s_0 & s_1 & s_0 & s_1 & s_2 & s_3 & s_1 & s_0 & s_2 & s_3 & s_2 & s_3 & \ldots \\
 & & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \\
\sigma & = & \{b\} & \{b\} & \{b\} & \{b\} & \varnothing & \{a,b\} & \{b\} & \{b\} & \varnothing & \{a,b\} & \varnothing & \{a,b\} & \ldots
\end{array}$$

$$\begin{tikzcd}
 & \arrow[d] & & \\
\{b\} & s_0 \arrow[r, bend left] \arrow[d] & s_1 \arrow[l, bend left] & \{b\} \\
\varnothing & s_2 \arrow[r, bend left] & s_3 \arrow[l, bend left] \arrow[u] & \{a,b\}
\end{tikzcd}$$

CONICET

UNC

Un simple lenguaje concurrente

`x :=` $expr$	asignación
P_1 ; P_2	secuencia
`if` $b_0 \to P_0$ $[\!]$ … $[\!]$ $b_n \to P_n$ `fi`	condicional bloqueante
`while` b `do` P `od`	iteración
$P_1 \parallel P_2$	composición paralela

Semántica del lenguaje de modelado

$$\langle x := e, \mu\rangle \longrightarrow \langle \surd, \mu[x \mapsto \mu(e)]\rangle$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle \quad P_2 \neq \surd}{\langle P_1 \parallel P_2, \mu\rangle \longrightarrow \langle P_1' \parallel P_2, \mu'\rangle}$$

$$\frac{\langle P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle \quad P_1 \neq \surd}{\langle P_1 \parallel P_2, \mu\rangle \longrightarrow \langle P_1 \parallel P_2', \mu'\rangle}$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle}{\langle P_1 \parallel \surd, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle}$$

$$\frac{\langle P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle}{\langle \surd \parallel P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle}$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle \quad P_1' \neq \surd}{\langle P_1 \,;\, P_2, \mu\rangle \longrightarrow \langle P_1' \,;\, P_2, \mu'\rangle}$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle \surd, \mu'\rangle}{\langle P_1 \,;\, P_2, \mu\rangle \longrightarrow \langle P_2, \mu'\rangle}$$

$$\frac{\langle P_j, \mu\rangle \longrightarrow \langle P_j', \mu'\rangle \qquad \mu(b_j) \text{ holds}}{\langle \texttt{if } b_0 \to P_0 \ [\!] \ldots [\!] \ b_n \to P_n \ \texttt{fi}, \mu\rangle \longrightarrow \langle P_j', \mu'\rangle}$$

$$\frac{\langle P; \texttt{while } b \texttt{ do } P \texttt{ od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle \qquad \mu(b) \text{ holds}}{\langle \texttt{while } b \texttt{ do } P \texttt{ od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle}$$

$$\frac{\neg\mu(b) \text{ holds}}{\langle \texttt{while } b \texttt{ do } P \texttt{ od}, \mu\rangle \longrightarrow \langle \surd, \mu\rangle}$$

Trazas observables

- Para hablar de las propiedades de un sistema debemos observar las propiedades de sus estados y su orden temporal en las ejecuciones.

$$\mathcal{L}(K) = \{\sigma \in (\mathcal{P}(\mathsf{PA}))^\omega \quad | \ \exists\, \rho \text{ ejecución de } K : \forall\, i \geq 0 : \sigma(i) = \mathcal{L}(\rho(i))\}$$

$$\begin{array}{cccccccccccccc} \rho & = & s_0 & s_1 & s_0 & s_1 & s_2 & s_3 & s_1 & s_0 & s_2 & s_3 & s_2 & s_3 & \ldots \\ & & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \\ \sigma & = & \{b\} & \{b\} & \{b\} & \{b\} & \varnothing & \{a,b\} & \{b\} & \{b\} & \varnothing & \{a,b\} & \varnothing & \{a,b\} & \ldots \end{array}$$

$$\begin{tikzcd} \{b\} & s_0 \arrow[r, bend left] \arrow[d] & s_1 \arrow[l, bend left] & \{b\} \\ \varnothing & s_2 \arrow[r, bend left] & s_3 \arrow[l, bend left] \arrow[u] & \{a,b\} \end{tikzcd}$$

$$P \overset{?}{\models} \phi$$

$$\Big\downarrow$$

$$K(P) \overset{?}{\models} \phi$$

$$\Big\downarrow$$

$$\mathcal{L}(K(P))$$

LTL: Sintaxis

$$\begin{array}{lllllllll} \phi, \psi ::= & & & & & & & & \\ & p & | & \neg\phi & | & \phi \wedge \psi & | & \phi \vee \psi & | \quad \phi \Rightarrow \psi \quad | \qquad \text{(op. proposicionales)} \\ & \mathtt{X}\,\phi & | & \mathtt{F}\,\phi & | & \mathtt{G}\,\phi & | & \phi\,\mathtt{U}\,\psi & | \quad \phi\,\mathtt{R}\,\psi \qquad\qquad \text{(op. temporales)} \end{array}$$

donde $p \in \mathsf{PA}$ es cualquier variable proporsional (o proposición atómica)

Los operadores básicos son $\neg$, $\vee$, `X`, y `U`.

Los otros operadores se definen en término de estos.

Un simple lenguaje concurrente

`x :=` $expr$	asignación
P_1 ; P_2	secuencia
`if` $b_0 \to P_0$ $[\!]\ \ldots\ [\!]$ $b_n \to P_n$ `fi`	condicional bloqueante
`while` b `do` P `od`	iteración
$P_1 \parallel P_2$	composición paralela

Semántica del lenguaje de modelado

$$\langle x := e, \mu\rangle \longrightarrow \langle \surd, \mu[x \mapsto \mu(e)]\rangle$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle \quad P_2 \neq \surd}{\langle P_1 \parallel P_2, \mu\rangle \longrightarrow \langle P_1' \parallel P_2, \mu'\rangle}$$

$$\frac{\langle P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle \quad P_1 \neq \surd}{\langle P_1 \parallel P_2, \mu\rangle \longrightarrow \langle P_1 \parallel P_2', \mu'\rangle}$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle}{\langle P_1 \parallel \surd, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle}$$

$$\frac{\langle P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle}{\langle \surd \parallel P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle}$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle \quad P_1' \neq \surd}{\langle P_1 \ ; P_2, \mu\rangle \longrightarrow \langle P_1' \ ; P_2, \mu'\rangle}$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle \surd, \mu'\rangle}{\langle P_1 \ ; P_2, \mu\rangle \longrightarrow \langle P_2, \mu'\rangle}$$

$$\frac{\langle P_j, \mu\rangle \longrightarrow \langle P_j', \mu'\rangle \qquad \mu(b_j) \text{ holds}}{\langle \texttt{if } b_0 \to P_0 \ [\!]\ \ldots\ [\!]\ b_n \to P_n \texttt{ fi}, \mu\rangle \longrightarrow \langle P_j', \mu'\rangle}$$

$$\frac{\langle P; \texttt{while } b \texttt{ do } P \texttt{ od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle \qquad \mu(b) \text{ holds}}{\langle \texttt{while } b \texttt{ do } P \texttt{ od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle}$$

$$\frac{\neg\mu(b) \text{ holds}}{\langle \texttt{while } b \texttt{ do } P \texttt{ od}, \mu\rangle \longrightarrow \langle \surd, \mu\rangle}$$

Trazas observables

- Para hablar de las propiedades de un sistema debemos observar las propiedades de sus estados y su orden temporal en las ejecuciones.

$$\mathcal{L}(K) = \{\sigma \in (\mathcal{P}(\mathsf{PA}))^\omega \quad | \ \exists\, \rho \text{ ejecución de } K : \forall\, i \geq 0 : \sigma(i) = \mathcal{L}(\rho(i))\}$$

$$\begin{array}{cccccccccccccc} \rho & = & s_0 & s_1 & s_0 & s_1 & s_2 & s_3 & s_1 & s_0 & s_2 & s_3 & s_2 & s_3 & \ldots \\ & & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \\ \sigma & = & \{b\} & \{b\} & \{b\} & \{b\} & \varnothing & \{a,b\} & \{b\} & \{b\} & \varnothing & \{a,b\} & \varnothing & \{a,b\} & \ldots \end{array}$$

$$\begin{tikzcd} \{b\}\ s_0 \arrow[r, bend left] \arrow[d] & s_1\ \{b\} \arrow[l, bend left] \\ \varnothing\ s_2 \arrow[r, bend left] & s_3\ \{a,b\} \arrow[l, bend left] \arrow[u] \end{tikzcd}$$

$$P \overset{?}{\models} \phi$$

$$\Big\downarrow$$

$$K(P) \overset{?}{\models} \phi$$

$$\Big\downarrow$$

$$\mathcal{L}(K(P)) \overset{?}{\models} \phi$$

LTL: Sintaxis

$\phi, \psi ::=$

p	$\mid\ \neg\phi$	$\mid\ \phi \wedge \psi$	$\mid\ \phi \vee \psi$	$\mid\ \phi \Rightarrow \psi$ $\mid$	(op. proposicionales)
$\mathtt{X}\,\phi$	$\mid\ \mathtt{F}\,\phi$	$\mid\ \mathtt{G}\,\phi$	$\mid\ \phi\,\mathtt{U}\,\psi$	$\mid\ \phi\,\mathtt{R}\,\psi$	(op. temporales)

donde $p \in \mathsf{PA}$ es cualquier variable proporsional (o proposición atómica)

Los operadores básicos son $\neg$, $\vee$, `X`, y `U`.

Los otros operadores se definen en término de estos.

CONICET

UNC

LTL: Semántica

Un modelo de una fórmula LTL es un conjunto de trazas que satisfacen dicha fórmula

Es decir, $M \subseteq (\mathcal{P}(\mathsf{PA}))^\omega$ es modelo de ϕ, denotado con $M \models \phi$, si para todo $\sigma \in (\mathcal{P}(\mathsf{PA}))^\omega$,

$$\sigma \in M \ \text{ implies } \ \sigma \models \phi$$

El lenguaje de ϕ es su modelo más grande. Formalmente:

$$\mathcal{L}(\phi) = \{\sigma \in (\mathcal{P}(\mathsf{PA}))^\omega \mid \sigma \models \phi\}$$

Por consiguiente, M es modelo de ϕ si $M \subseteq \mathcal{L}(\phi)$.

Un simple lenguaje concurrente

`x :=` *expr*	asignación
P_1 ; P_2	secuencia
`if` $b_0 \to P_0$ $[\!]\ \ldots\ [\!]$ $b_n \to P_n$ `fi`	condicional bloqueante
`while` b `do` P `od`	iteración
$P_1 \parallel P_2$	composición paralela

Semántica del lenguaje de modelado

$$\langle x := e, \mu\rangle \longrightarrow \langle \surd, \mu[x \mapsto \mu(e)]\rangle$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle \quad P_2 \neq \surd}{\langle P_1 \parallel P_2, \mu\rangle \longrightarrow \langle P_1' \parallel P_2, \mu'\rangle}$$

$$\frac{\langle P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle \quad P_1 \neq \surd}{\langle P_1 \parallel P_2, \mu\rangle \longrightarrow \langle P_1 \parallel P_2', \mu'\rangle}$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle}{\langle P_1 \parallel \surd, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle}$$

$$\frac{\langle P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle}{\langle \surd \parallel P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle}$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle \quad P_1' \neq \surd}{\langle P_1 \,;\, P_2, \mu\rangle \longrightarrow \langle P_1' \,;\, P_2, \mu'\rangle}$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle \surd, \mu'\rangle}{\langle P_1 \,;\, P_2, \mu\rangle \longrightarrow \langle P_2, \mu'\rangle}$$

$$\frac{\langle P_j, \mu\rangle \longrightarrow \langle P_j', \mu'\rangle \qquad \mu(b_j) \text{ holds}}{\langle \texttt{if } b_0 \to P_0\ [\!]\ \ldots\ [\!]\ b_n \to P_n \texttt{ fi}, \mu\rangle \longrightarrow \langle P_j', \mu'\rangle}$$

$$\frac{\langle P; \texttt{while } b \texttt{ do } P \texttt{ od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle \qquad \mu(b) \text{ holds}}{\langle \texttt{while } b \texttt{ do } P \texttt{ od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle}$$

$$\frac{\neg\mu(b) \text{ holds}}{\langle \texttt{while } b \texttt{ do } P \texttt{ od}, \mu\rangle \longrightarrow \langle \surd, \mu\rangle}$$

Trazas observables

- Para hablar de las propiedades de un sistema debemos observar las propiedades de sus estados y su orden temporal en las ejecuciones.

$$\mathcal{L}(K) = \{\sigma \in (\mathcal{P}(\mathsf{PA}))^\omega \;\mid \exists\, \rho \text{ ejecución de } K : \forall\, i \geq 0 : \sigma(i) = \mathcal{L}(\rho(i))\}$$

$$\begin{array}{ccccccccccccccc}
\rho & = & s_0 & s_1 & s_0 & s_1 & s_2 & s_3 & s_1 & s_0 & s_2 & s_3 & s_2 & s_3 & \ldots \\
 & & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \\
\sigma & = & \{b\} & \{b\} & \{b\} & \{b\} & \varnothing & \{a,b\} & \{b\} & \{b\} & \varnothing & \{a,b\} & \varnothing & \{a,b\} & \ldots
\end{array}$$

$$\begin{tikzcd}
\{b\}\ s_0 \arrow[r, bend left] \arrow[d] & s_1\ \{b\} \arrow[l, bend left] \\
\varnothing\ s_2 \arrow[r, bend left] & s_3\ \{a,b\} \arrow[l, bend left] \arrow[u]
\end{tikzcd}$$

$$P \overset{?}{\models} \phi$$

$$\Big\downarrow$$

$$K(P) \overset{?}{\models} \phi$$

$$\Big\downarrow$$

$$\mathcal{L}(K(P)) \overset{?}{\models} \phi$$

$$\Big\downarrow$$

$$\mathcal{L}(K(P)) \overset{?}{\subseteq} \mathcal{L}(\phi)$$

LTL: Sintaxis

$\phi, \psi ::=$

$p \;\mid\; \neg\phi \;\mid\; \phi \wedge \psi \;\mid\; \phi \vee \psi \;\mid\; \phi \Rightarrow \psi \;\mid$ (op. proposicionales)

$\mathtt{X}\,\phi \;\mid\; \mathtt{F}\,\phi \;\mid\; \mathtt{G}\,\phi \;\mid\; \phi\,\mathtt{U}\,\psi \;\mid\; \phi\,\mathtt{R}\,\psi$ (op. temporales)

donde $p \in \mathsf{PA}$ es cualquier variable proporsional (o proposición atómica)

Los operadores básicos son $\neg$, $\vee$, $\mathtt{X}$, y $\mathtt{U}$.

Los otros operadores se definen en término de estos.

LTL: Semántica

Un modelo de una fórmula LTL es un conjunto de trazas que satisfacen dicha fórmula

Es decir, $M \subseteq (\mathcal{P}(\mathsf{PA}))^\omega$ es modelo de ϕ, denotado con $M \models \phi$, si para todo $\sigma \in (\mathcal{P}(\mathsf{PA}))^\omega$,

$$\sigma \in M \;\text{ implies }\; \sigma \models \phi$$

El lenguaje de ϕ es su modelo más grande. Formalmente:

$$\mathcal{L}(\phi) = \{\sigma \in (\mathcal{P}(\mathsf{PA}))^\omega \mid \sigma \models \phi\}$$

Por consiguiente, M es modelo de ϕ si $M \subseteq \mathcal{L}(\phi)$.

Un simple lenguaje concurrente

`x := ` $expr$	asignación
P_1 ; P_2	secuencia
`if` $b_0 \to P_0$ [] … [] $b_n \to P_n$ `fi`	condicional bloqueante
`while` b `do` P `od`	iteración
$P_1 \parallel P_2$	composición paralela

Semántica del lenguaje de modelado

$$\langle x := e, \mu\rangle \longrightarrow \langle \surd, \mu[x \mapsto \mu(e)]\rangle$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle \quad P_2 \neq \surd}{\langle P_1 \parallel P_2, \mu\rangle \longrightarrow \langle P_1' \parallel P_2, \mu'\rangle}$$

$$\frac{\langle P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle \quad P_1 \neq \surd}{\langle P_1 \parallel P_2, \mu\rangle \longrightarrow \langle P_1 \parallel P_2', \mu'\rangle}$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle}{\langle P_1 \parallel \surd, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle}$$

$$\frac{\langle P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle}{\langle \surd \parallel P_2, \mu\rangle \longrightarrow \langle P_2', \mu'\rangle}$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle P_1', \mu'\rangle \quad P_1' \neq \surd}{\langle P_1 \,;\, P_2, \mu\rangle \longrightarrow \langle P_1' \,;\, P_2, \mu'\rangle}$$

$$\frac{\langle P_1, \mu\rangle \longrightarrow \langle \surd, \mu'\rangle}{\langle P_1 \,;\, P_2, \mu\rangle \longrightarrow \langle P_2, \mu'\rangle}$$

$$\frac{\langle P_j, \mu\rangle \longrightarrow \langle P_j', \mu'\rangle \qquad \mu(b_j) \text{ holds}}{\langle \texttt{if}\ b_0 \to P_0\ [] \ldots []\ b_n \to P_n\ \texttt{fi}, \mu\rangle \longrightarrow \langle P_j', \mu'\rangle}$$

$$\frac{\langle P; \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle \qquad \mu(b) \text{ holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle P', \mu'\rangle}$$

$$\frac{\neg\mu(b) \text{ holds}}{\langle \texttt{while}\ b\ \texttt{do}\ P\ \texttt{od}, \mu\rangle \longrightarrow \langle \surd, \mu\rangle}$$

Trazas observables

- Para hablar de las propiedades de un sistema debemos observar las propiedades de sus estados y su orden temporal en las ejecuciones.

$$\mathcal{L}(K) = \{\sigma \in (\mathcal{P}(\mathsf{PA}))^\omega \mid \exists\, \rho \text{ ejecución de } K : \forall\, i \geq 0 : \sigma(i) = \mathcal{L}(\rho(i))\}$$

$$\begin{array}{cccccccccccccc}
\rho & = & s_0 & s_1 & s_0 & s_1 & s_2 & s_3 & s_1 & s_0 & s_2 & s_3 & s_2 & s_3 & \ldots \\
 & & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \downarrow & \\
\sigma & = & \{b\} & \{b\} & \{b\} & \{b\} & \varnothing & \{a,b\} & \{b\} & \{b\} & \varnothing & \{a,b\} & \varnothing & \{a,b\} & \ldots
\end{array}$$

$$\begin{tikzcd}
\{b\}\ s_0 \arrow[r, bend left] \arrow[d] & s_1\ \{b\} \arrow[l, bend left] \\
\varnothing\ s_2 \arrow[r, bend left] & s_3\ \{a,b\} \arrow[l, bend left] \arrow[u]
\end{tikzcd}$$

$$P \overset{?}{\models} \phi$$

$$\Downarrow$$

$$K(P) \overset{?}{\models} \phi$$

$$\Downarrow$$

$$\mathcal{L}(K(P)) \overset{?}{\models} \phi$$

$$\Downarrow$$

$$\mathcal{L}(K(P)) \overset{?}{\subseteq} \mathcal{L}(\phi)$$

El problema de model checking se reduce a verificar esta inclusión de lenguajes ω-regulares

LTL: Sintaxis

$\phi, \psi ::=$

$p \mid \neg\phi \mid \phi \wedge \psi \mid \phi \vee \psi \mid \phi \Rightarrow \psi \mid$ (op. proposicionales)

$\mathtt{X}\,\phi \mid \mathtt{F}\,\phi \mid \mathtt{G}\,\phi \mid \phi\,\mathtt{U}\,\psi \mid \phi\,\mathtt{R}\,\psi$ (op. temporales)

donde $p \in$ PA es cualquier variable proporsional (o proposición atómica)

Los operadores básicos son $\neg$, $\vee$, `X`, y `U`.

Los otros operadores se definen en término de estos.

LTL: Semántica

Un modelo de una fórmula LTL es un conjunto de trazas que satisfacen dicha fórmula

Es decir, $M \subseteq (\mathcal{P}(\mathsf{PA}))^\omega$ es modelo de ϕ, denotado con $M \models \phi$, si para todo $\sigma \in (\mathcal{P}(\mathsf{PA}))^\omega$,

$$\sigma \in M \ \text{ implies } \ \sigma \models \phi$$

El lenguaje de ϕ es su modelo más grande. Formalmente:

$$\mathcal{L}(\phi) = \{\sigma \in (\mathcal{P}(\mathsf{PA}))^\omega \mid \sigma \models \phi\}$$

Por consiguiente, M es modelo de ϕ si $M \subseteq \mathcal{L}(\phi)$.

Autómata de Büchi

$\mathcal{A} = (\Sigma, S, s_0, \delta, A)$

Autómata de Büchi

$$\mathcal{A} = (\Sigma, S, s_0, \delta, A)$$

Alfabeto (conjunto de símbolos)

Autómata de Büchi

$$\mathcal{A} = (\Sigma, S, s_0, \delta, A)$$

Conjunto de estados

Alfabeto (conjunto de símbolos)

Autómata de Büchi

$$\mathcal{A} = (\Sigma, S, s_0, \delta, A)$$

Estado inicial

$s_0 \in S$

Conjunto de estados

Alfabeto (conjunto de símbolos)

Autómata de Büchi

$$\mathcal{A} = (\Sigma, S, s_0, \delta, A)$$

Función de transición

$\delta : \Sigma \times S \to \mathcal{P}(S)$

Estado inicial

$s_0 \in S$

Conjunto de estados

Alfabeto (conjunto de símbolos)

Autómata de Büchi

$$\mathcal{A} = (\Sigma, S, s_0, \delta, A)$$

Conjunto de
estados de aceptación
$A \subseteq S$

Función de transición
$\delta : \Sigma \times S \to \mathcal{P}(S)$

Estado inicial
$s_0 \in S$

Conjunto de estados

Alfabeto (conjunto de símbolos)

Autómata de Büchi

$\mathcal{A} = (\Sigma, S, s_0, \delta, A)$

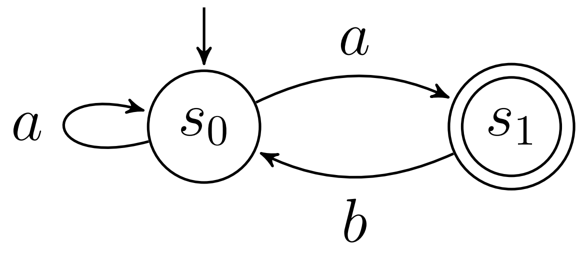
Conjunto de estados de aceptación $A \subseteq S$

Función de transición $\delta : \Sigma \times S \to \mathcal{P}(S)$

Estado inicial $s_0 \in S$

Conjunto de estados

Alfabeto (conjunto de símbolos)



$$\Sigma = \{a, b\}$$

$$S = \{s_0, s_1\}$$

$$A = \{s_1\}$$

$$\delta(a, s_0) = \{s_0, s_1\}$$

$$\delta(b, s_1) = \{s_0\}$$

$$\delta(b, s_0) = \delta(a, s_1) = \varnothing$$

Lenguaje definido por un Autómata de Büchi

Una traza $\sigma \in \Sigma^\omega$ es aceptada por $\mathcal{A}$ si existe una ejecución $\rho \in S^\omega$ tal que

1. $\rho(0) = s_0$,
2. $\rho(i+1) \in \delta(\sigma(i), \rho(i))$ para todo $i \geq 0$, y
3. el conjunto $\{i \geq 0 \mid \rho(i) \in A\}$ es infinito.

CONICET

UNC

Lenguaje definido por un Autómata de Büchi

Una traza $\sigma \in \Sigma^\omega$ es aceptada por $\mathcal{A}$ si existe una ejecución $\rho \in S^\omega$ tal que

1. $\rho(0) = s_0$,
2. $\rho(i+1) \in \delta(\sigma(i), \rho(i))$ para todo $i \geq 0$, y
3. el conjunto $\{i \geq 0 \mid \rho(i) \in A\}$ es infinito.

es decir,
algunos estados de
aceptación deben aparecer
infinitas veces

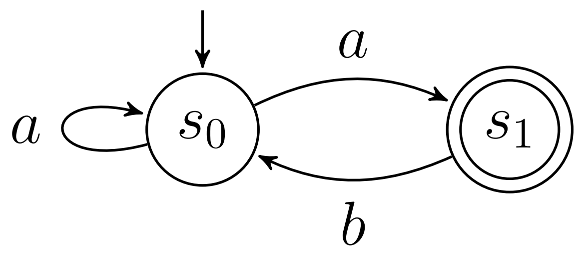
Lenguaje definido por un Autómata de Büchi

Una traza $\sigma \in \Sigma^\omega$ es aceptada por $\mathcal{A}$ si existe una ejecución $\rho \in S^\omega$ tal que

1. $\rho(0) = s_0$,
2. $\rho(i+1) \in \delta(\sigma(i), \rho(i))$ para todo $i \geq 0$, y
3. el conjunto $\{i \geq 0 \mid \rho(i) \in A\}$ es infinito.

es decir,
algunos estados de
aceptación deben aparecer
infinitas veces

$\mathcal{L}(\mathcal{A})$ denota al lenguaje aceptado por $\mathcal{A}$, es decir, al conjunto de todas las trazas aceptadas por $\mathcal{A}$.



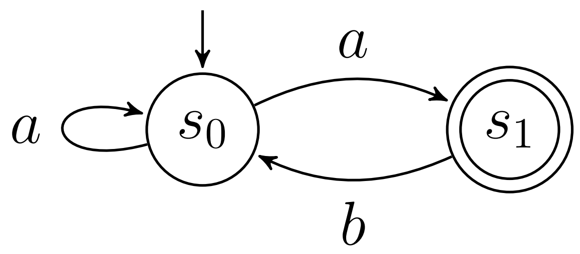
Lenguaje definido por un Autómata de Büchi

Una traza $\sigma \in \Sigma^\omega$ es aceptada por $\mathcal{A}$ si existe una ejecución $\rho \in S^\omega$ tal que

1. $\rho(0) = s_0$,
2. $\rho(i+1) \in \delta(\sigma(i), \rho(i))$ para todo $i \geq 0$, y
3. el conjunto $\{i \geq 0 \mid \rho(i) \in A\}$ es infinito.

es decir,
algunos estados de
aceptación deben aparecer
infinitas veces

$\mathcal{L}(\mathcal{A})$ denota al lenguaje aceptado por $\mathcal{A}$, es decir, al conjunto de todas las trazas aceptadas por $\mathcal{A}$.



$\sigma \in \mathcal{L}(\mathcal{A})$ si $\sigma \in \{a, b\}^\omega$ y

- $\sigma(0) = a$,
- para todo $i \geq 0$, $\sigma(i) = b$ implica que $\sigma(i+1) = a$,
- para todo $i \geq 0$, existe $j \geq i$ tal que $\sigma(i) = b$.

Autómatas de Büchi

- Los autómatas de Büchi aceptan exactamente todos los lenguajes ω-regulares

Tienen la pinta $\bigcup_{i=1}^{n} E_i \cdot F_i^{\omega}$ con E_i y F_i lenguajes regulares, para todo $1 \geq i \geq n$.

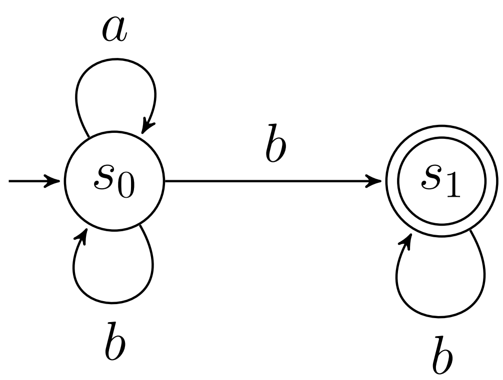
Autómatas de Büchi

- Los autómatas de Büchi aceptan exactamente todos los lenguajes ω-regulares

  Tienen la pinta $\bigcup_{i=1}^{n} E_i \cdot F_i^{\omega}$ con E_i y F_i lenguajes regulares, para todo $1 \geq i \geq n$.

- Los autómatas de Büchi no son determinizables

  Ej.:



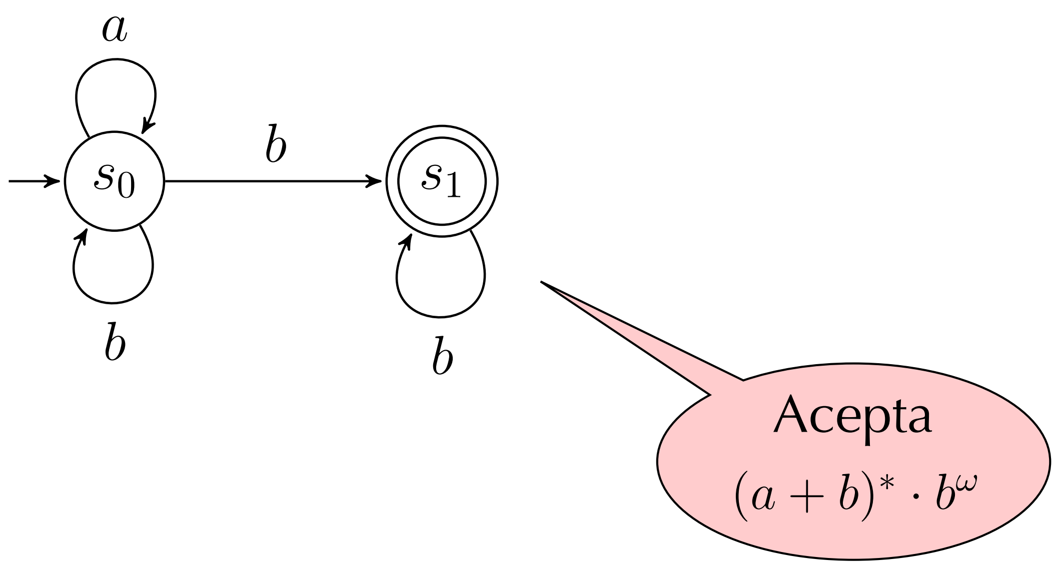
Autómatas de Büchi

- Los autómatas de Büchi aceptan exactamente todos los lenguajes ω-regulares

  Tienen la pinta $\bigcup_{i=1}^{n} E_i \cdot F_i^{\omega}$ con E_i y F_i lenguajes regulares, para todo $1 \geq i \geq n$.

- Los autómatas de Büchi no son determinizables

  Ej.:



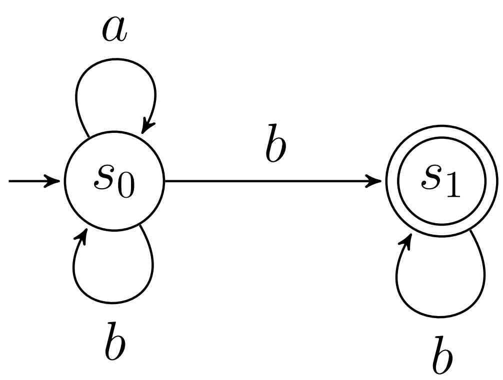
Autómatas de Büchi

- Los autómatas de Büchi aceptan exactamente todos los lenguajes ω-regulares

  Tienen la pinta $\bigcup_{i=1}^{n} E_i \cdot F_i^{\omega}$ con E_i y F_i lenguajes regulares, para todo $1 \geq i \geq n$.

- Los autómatas de Büchi no son determinizables

  Ej.:



Acepta
$(a + b)^* \cdot b^{\omega}$

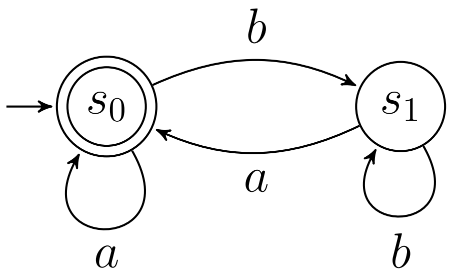
Existen autómatas ω-regulares determinizables (Rabin, Streett)

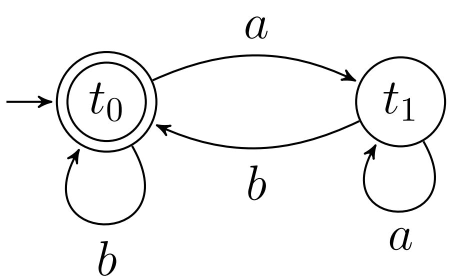
CONICET

UNC

Intersección

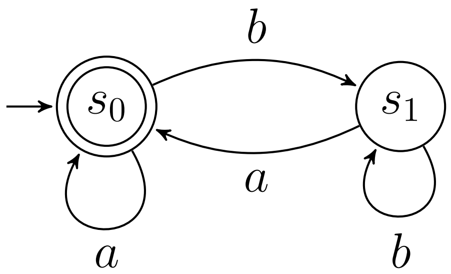
Teorema: Dado dos autómatas de Büchi $\mathcal{A}_1$ y $\mathcal{A}_2$, se puede construir otro autómata $\mathcal{A}_1 \cap \mathcal{A}_2$ tal que $\mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2) = \mathcal{L}(\mathcal{A}_1 \cap \mathcal{A}_2)$.



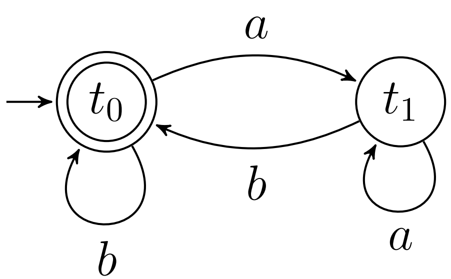


Intersección

Teorema: Dado dos autómatas de Büchi $\mathcal{A}_1$ y $\mathcal{A}_2$, se puede construir otro autómata $\mathcal{A}_1 \cap \mathcal{A}_2$ tal que $\mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2) = \mathcal{L}(\mathcal{A}_1 \cap \mathcal{A}_2)$.



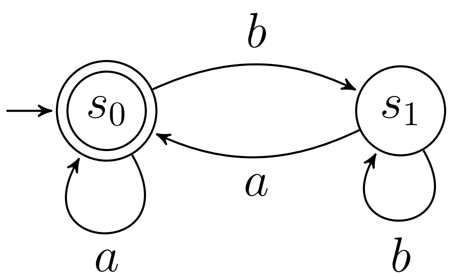
$\sigma \in \{a, b\}^\omega$ y no ocurren infinitas b consecutivas

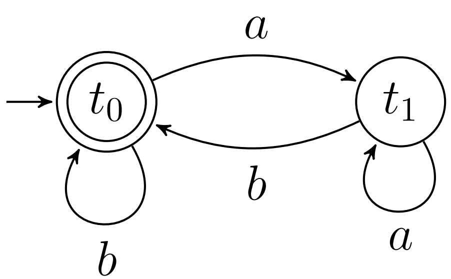


$\sigma \in \{a, b\}^\omega$ y no ocurren infinitas a consecutivas

Intersección

Teorema: Dado dos autómatas de Büchi $\mathcal{A}_1$ y $\mathcal{A}_2$, se puede construir otro autómata $\mathcal{A}_1 \cap \mathcal{A}_2$ tal que $\mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2) = \mathcal{L}(\mathcal{A}_1 \cap \mathcal{A}_2)$.





Si $\mathcal{A}_i = (\Sigma, S_i, s_0^i, \delta_i, A_i)$, con $i = 1, 2$, $\mathcal{A}_1 \cap \mathcal{A}_2 = (\Sigma, S, s_0, \delta, A)$, donde

$$S = S_1 \times S_2 \times \{0, 1, 2\}$$

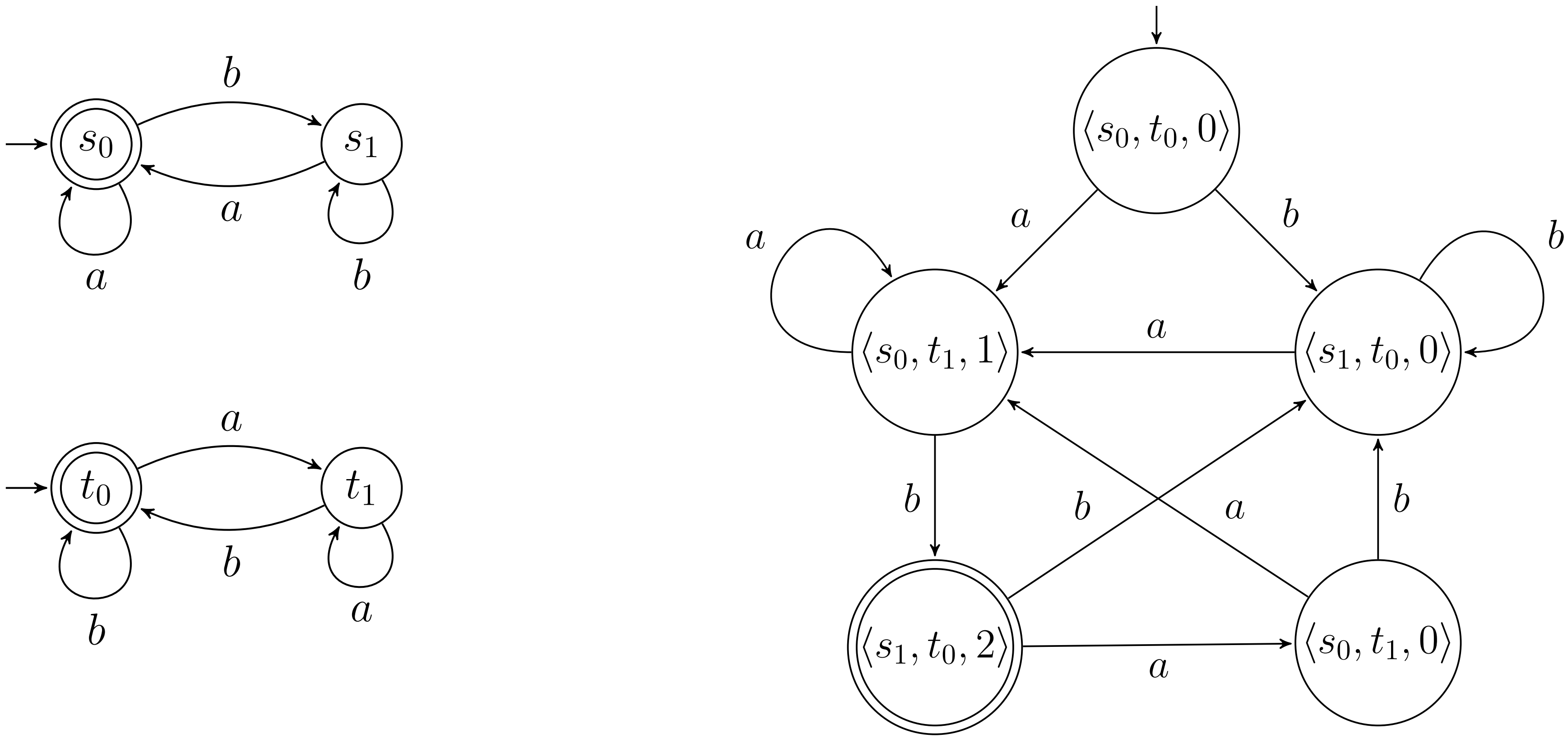
$$s_0 = \langle s_0^1, s_0^2, 0\rangle$$

$$A = S_1 \times S_2 \times \{2\}$$

$$\langle s_1', s_2', i\rangle \in \delta(a, \langle s_1, s_2, j\rangle) \text{ sii } \begin{cases} s_1' \in \delta_1(a, s_1), s_2' \in \delta_2(a, s_2), \text{ y} \\ i = \begin{cases} 0 & \text{si } j = 2 \\ 1 & \text{si } j = 0 \text{ y } s_1' \in A_1 \\ 2 & \text{si } j = 1 \text{ y } s_2' \in A_2 \\ j & \text{en culaquier otro caso} \end{cases} \end{cases}$$

Intersección

Teorema: Dado dos autómatas de Büchi $\mathcal{A}_1$ y $\mathcal{A}_2$, se puede construir otro autómata $\mathcal{A}_1 \cap \mathcal{A}_2$ tal que $\mathcal{L}(\mathcal{A}_1) \cap \mathcal{L}(\mathcal{A}_2) = \mathcal{L}(\mathcal{A}_1 \cap \mathcal{A}_2)$.



$\sigma \in \{a, b\}^\omega$ conteniendo infinitas a y b

Complementación

Teorema: Dado el autómatas de Büchi $\mathcal{A}$ se puede construir un autómata $\mathcal{A}^{\mathsf{c}}$ tal que $\mathcal{L}(\mathcal{A})^{\mathsf{c}} = \mathcal{L}(\mathcal{A}^{\mathsf{c}})$.

Complementación

Teorema: Dado el autómatas de Büchi $\mathcal{A}$ se puede construir un autómata $\mathcal{A}^{\mathsf{c}}$ tal que $\mathcal{L}(\mathcal{A})^{\mathsf{c}} = \mathcal{L}(\mathcal{A}^{\mathsf{c}})$.

- $\mathcal{A}^{\mathsf{c}}$ crece exponencialmente.
- El mejor algoritmo da la cota dura de $O\left((0.76 \cdot |S|)^{|S|}\right)$.

Complementación

Teorema: Dado el autómatas de Büchi $\mathcal{A}$ se puede construir un autómata $\mathcal{A}^c$ tal que $\mathcal{L}(\mathcal{A})^c = \mathcal{L}(\mathcal{A}^c)$.

- $\mathcal{A}^c$ crece exponencialmente.
- El mejor algoritmo da la cota dura de $O\left((0.76 \cdot |S|)^{|S|}\right)$.

No lo vamos a ver en el
curso pero googleen si les interesa.
Hay mucho al respecto

Verificación de vacuidad

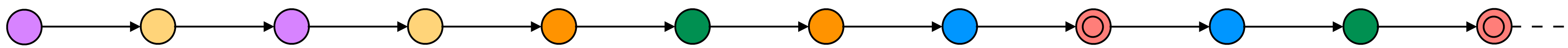
Si $L(\mathcal{A}) \neq \varnothing$

Verificación de vacuidad

Si $L(\mathcal{A}) \neq \emptyset$

$\Rightarrow$ hay una ejecución ρ aceptada por $\mathcal{A}$

ρ



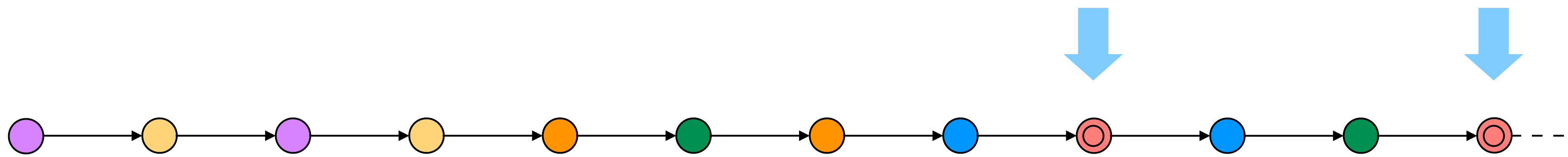
Verificación de vacuidad

Si $\mathcal{L}(\mathcal{A}) \neq \varnothing$

$\Rightarrow$ hay una ejecución ρ aceptada por $\mathcal{A}$

$\Rightarrow$ ρ contiene infinitas ocurrencias de algunos estados de aceptación

ρ



CONICET

UNC

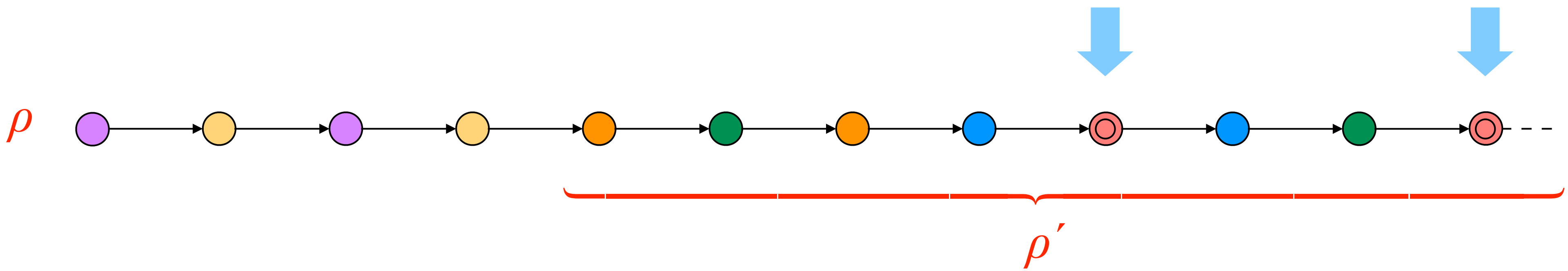
Verificación de vacuidad

Si $L(\mathcal{A}) \neq \varnothing$

$\Rightarrow$ hay una ejecución ρ aceptada por $\mathcal{A}$

$\Rightarrow$ ρ contiene infinitas ocurrencias de algunos estados de aceptación

$\Rightarrow$ existe un sufijo ρ' de ρ tal que todo estado aparece infinitas veces (dado que es $\mathcal{A}$ finito)



Verificación de vacuidad

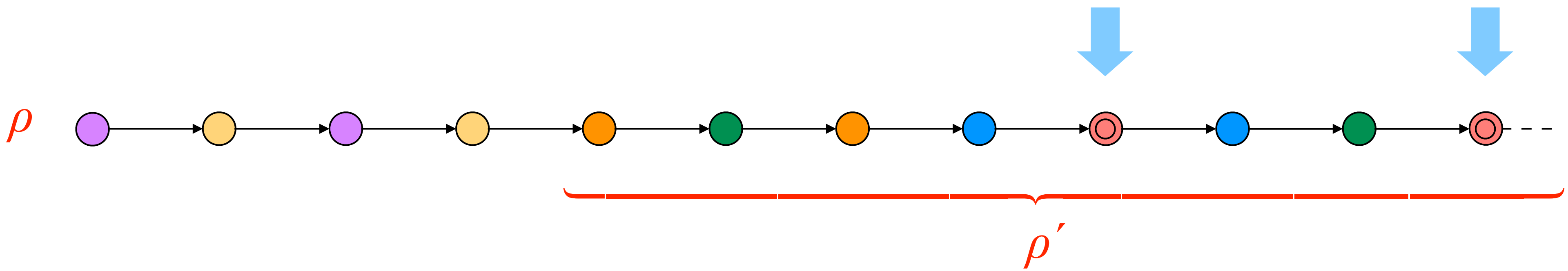
Si $L(\mathcal{A}) \neq \varnothing$

$\Rightarrow$ hay una ejecución ρ aceptada por $\mathcal{A}$

$\Rightarrow$ ρ contiene infinitas ocurrencias de algunos estados de aceptación

$\Rightarrow$ existe un sufijo ρ' de ρ tal que todo estado aparece infinitas veces (dado que es $\mathcal{A}$ finito)

$\Rightarrow$ todo estado de ρ' es alcanzado por cualquier otro estado de ρ'



Verificación de vacuidad

Si $L(\mathcal{A}) \neq \varnothing$

$\Rightarrow$ hay una ejecución ρ aceptada por $\mathcal{A}$

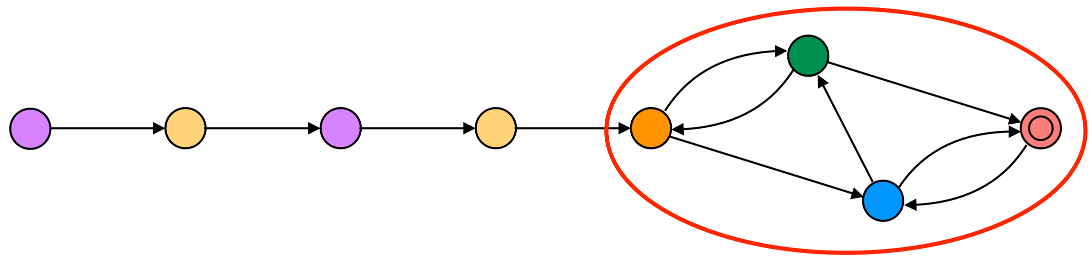
$\Rightarrow$ ρ contiene infinitas ocurrencias de algunos estados de aceptación

$\Rightarrow$ existe un sufijo ρ' de ρ tal que todo estado aparece infinitas veces (dado que es $\mathcal{A}$ finito)

$\Rightarrow$ todo estado de ρ' es alcanzado por cualquier otro estado de ρ'

$\Rightarrow$ los estados de ρ' forman una componente fuertemente conexa en $\mathcal{A}$ y alguno de ellos es de aceptación

ρ



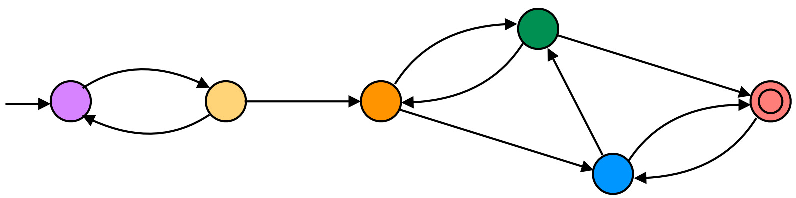
estados de ρ'

CONICET

UNC

Verificación de vacuidad

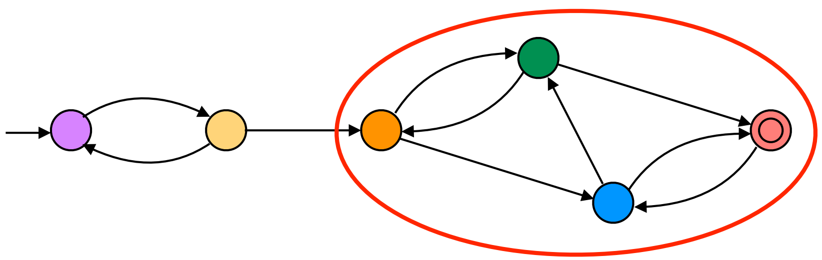
Reciprocamente:



Verificación de vacuidad

Reciprocamente:

Si $\mathcal{A}$ contiene una componente fuertemente conexa con un estado de aceptación en ella alcanzable del estado inicial

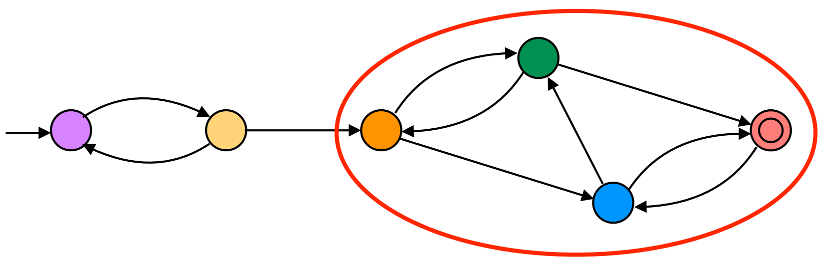


Verificación de vacuidad

Reciprocamente:

Si $\mathcal{A}$ contiene una componente fuertemente conexa con un estado de aceptación en ella alcanzable del estado inicial

$\Rightarrow$ $\mathcal{A}$ tiene una ejecución donde un estado de aceptación aparece infinitas veces

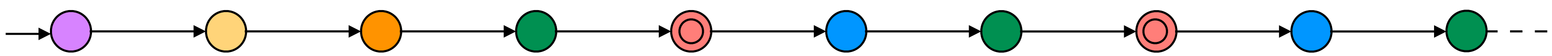


Verificación de vacuidad

Reciprocamente:

Si $\mathcal{A}$ contiene una componente fuertemente conexa con un estado de aceptación en ella alcanzable del estado inicial

$\Rightarrow$ $\mathcal{A}$ tiene una ejecución donde un estado de aceptación aparece infinitas veces



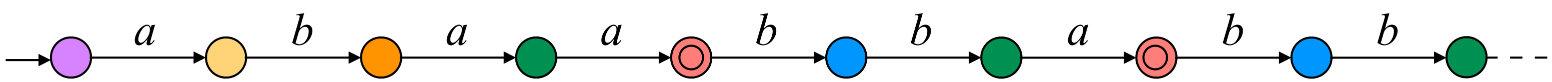
Verificación de vacuidad

Reciprocamente:

Si $\mathcal{A}$ contiene una componente fuertemente conexa con un estado de aceptación en ella alcanzable del estado inicial

$\Rightarrow$ $\mathcal{A}$ tiene una ejecución donde un estado de aceptación aparece infinitas veces

$\Rightarrow$ $\mathcal{A}$ acepta una traza



CONICET

UNC

Verificación de vacuidad

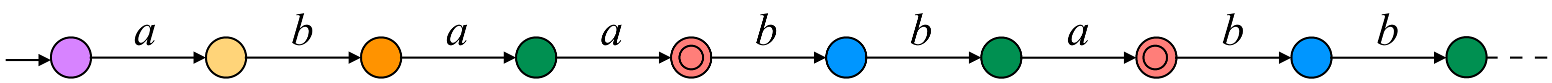
Reciprocamente:

Si $\mathcal{A}$ contiene una componente fuertemente conexa con un estado de aceptación en ella alcanzable del estado inicial

$\Rightarrow$ $\mathcal{A}$ tiene una ejecución donde un estado de aceptación aparece infinitas veces

$\Rightarrow$ $\mathcal{A}$ acepta una traza

$\Rightarrow$ $\mathcal{L}(\mathcal{A}) \neq \varnothing$



CONICET

UNC

Verificación de vacuidad

Reciprocamente:

Si $\mathcal{A}$ contiene una componente fuertemente conexa con un estado de aceptación en ella alcanzable del estado inicial

$\Rightarrow$ $\mathcal{A}$ tiene una ejecución donde un estado de aceptación aparece infinitas veces

$\Rightarrow$ $\mathcal{A}$ acepta una traza

$\Rightarrow$ $\mathcal{L}(\mathcal{A}) \neq \varnothing$

Por lo tanto: verificar vacuidad equivale a verificar que del estado inicial no se alcanza un ciclo que contenga un estado de aceptación

Verificación de vacuidad

Reciprocamente:

Si $\mathcal{A}$ contiene una componente fuertemente conexa con un estado de aceptación en ella alcanzable del estado inicial

$\Rightarrow$ $\mathcal{A}$ tiene una ejecución donde un estado de aceptación aparece infinitas veces

$\Rightarrow$ $\mathcal{A}$ acepta una traza

$\Rightarrow$ $\mathcal{L}(\mathcal{A}) \neq \emptyset$

Por lo tanto: verificar vacuidad equivale a verificar que del estado inicial no se alcanza un ciclo que contenga un estado de aceptación

Se realiza usando un doble DFS

CONICET

UNC

Verificación de vacuidad

Reciprocamente:

Si $\mathcal{A}$ contiene una componente fuertemente conexa con un estado de aceptación en ella alcanzable del estado inicial

$\Rightarrow$ $\mathcal{A}$ tiene una ejecución donde un estado de aceptación aparece infinitas veces

$\Rightarrow$ $\mathcal{A}$ acepta una traza

$\Rightarrow$ $\mathcal{L}(\mathcal{A}) \neq \varnothing$

Si encuentra ciclo, la traza testigo se encuentra en la pila de la llamada recursiva

Por lo tanto: verificar vacuidad equivale a verificar que del estado inicial no se alcanza un ciclo que contenga un estado de aceptación

Se realiza usando un doble DFS

CONICET UNC

Verificación de inclusión

$\mathcal{L}(A_1) \subseteq \mathcal{L}(A_2)$

Verificación de inclusión

$$\mathcal{L}(A_1) \subseteq \mathcal{L}(A_2)$$

$$\Updownarrow$$

$$\mathcal{L}(A_1) \cap \mathcal{L}(A_2)^{\mathsf{c}} = \varnothing$$

Verificación de inclusión

$$\mathcal{L}(A_1) \subseteq \mathcal{L}(A_2)$$

$$\Updownarrow$$

$$\mathcal{L}(A_1) \cap \mathcal{L}(A_2)^{\mathsf{c}} = \varnothing$$

$$\Updownarrow$$

Verificar que

$\mathcal{L}(A_1 \cap A_2^{\mathsf{c}})$ es vacío

Verificación de inclusión

$$\mathcal{L}(A_1) \subseteq \mathcal{L}(A_2)$$

$$\Updownarrow$$

$$\mathcal{L}(A_1) \cap \mathcal{L}(A_2)^{\mathsf{c}} = \varnothing$$

$$\Updownarrow$$

Verificar que

$\mathcal{L}(A_1 \cap A_2^{\mathsf{c}})$ es vacío

$$P \overset{?}{\models} \phi$$

$$\downarrow$$

$$K(P) \overset{?}{\models} \phi$$

$$\downarrow$$

$$\mathcal{L}(K(P)) \overset{?}{\models} \phi$$

$$\downarrow$$

$$\mathcal{L}(K(P)) \overset{?}{\subseteq} \mathcal{L}(\phi)$$

CONICET

UNC

Verificación de inclusión

$\mathcal{L}(A_1) \subseteq \mathcal{L}(A_2)$

$\Updownarrow$

$\mathcal{L}(A_1) \cap \mathcal{L}(A_2)^{\mathsf{c}} = \varnothing$

$\Updownarrow$

Verificar que

$\mathcal{L}(A_1 \cap A_2^{\mathsf{c}})$ es vacío

$P \overset{?}{\models} \phi$

$\downarrow$

$K(P) \overset{?}{\models} \phi$

$\downarrow$

$\mathcal{L}(K(P)) \overset{?}{\models} \phi$

$\downarrow$

$\mathcal{L}(K(P)) \overset{?}{\subseteq} \mathcal{L}(\phi)$

Si solo pudieramos transformarlos a autómatas de Büchi…

CONICET

UNC

De estructura de Kripke a Autómata de Büchi

Consideremos $K = (S, s_0, \longrightarrow, L)$.

Definimos $\mathcal{A}(K) = (\Sigma, S, s_0, \delta, A)$, donde

- $\Sigma =$
- $A =$
- $s' \in \delta(P, s)$ sii

De estructura de Kripke a Autómata de Büchi

Consideremos $K = (S, s_0, \longrightarrow, L)$.

Definimos $\mathcal{A}(K) = (\Sigma, S, s_0, \delta, A)$, donde

- $\Sigma = \mathcal{P}(\mathsf{PA})$
- $A =$
- $s' \in \delta(P, s)$ sii

De estructura de Kripke a Autómata de Büchi

Consideremos $K = (S, s_0, \longrightarrow, L)$.

Definimos $\mathcal{A}(K) = (\Sigma, S, s_0, \delta, A)$, donde

- $\Sigma = \mathcal{P}(\mathsf{PA})$
- $A = S$
- $s' \in \delta(P, s)$ sii

Notar que todos los estados son estados de aceptación

Esto es así porque nos interesan todas las ejecuciones posibles del sistema

De estructura de Kripke a Autómata de Büchi

Consideremos $K = (S, s_0, \longrightarrow, L)$.

Definimos $\mathcal{A}(K) = (\Sigma, S, s_0, \delta, A)$, donde

- $\Sigma = \mathcal{P}(\mathsf{PA})$
- $A = S$
- $s' \in \delta(P, s)$ sii $s \longrightarrow s'$ y $L(s) = P$

Notar que todos los estados son estados de aceptación

Esto es así porque nos interesan todas las ejecuciones posibles del sistema

De estructura de Kripke a Autómata de Büchi

Consideremos $K = (S, s_0, \longrightarrow, L)$.

Definimos $\mathcal{A}(K) = (\Sigma, S, s_0, \delta, A)$, donde

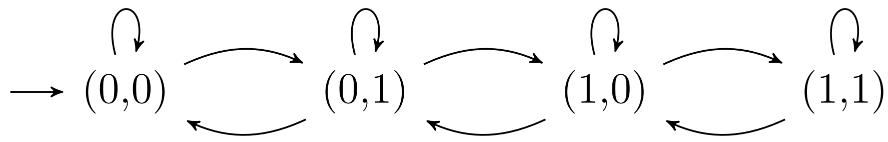
- $\Sigma = \mathcal{P}(\mathsf{PA})$
- $A = S$
- $s' \in \delta(P, s)$ sii $s \longrightarrow s'$ y $L(s) = P$

Teorema: $\mathcal{L}(K) = \mathcal{L}(\mathcal{A}(K))$

Notar que todos los estados son estados de aceptación

Esto es así porque nos interesan todas las ejecuciones posibles del sistema

De estructura de Kripke a Autómata de Büchi



$$L(0,0) = \{(x=0),(x\neq 1),(y=0),(y\neq 1),(x=y)\}$$
$$L(0,1) = \{(x=0),(x\neq 1),(y=1),(y\neq 0),(x\neq y)\}$$
$$L(1,0) = \{(x=1),(x\neq 0),(y=0),(y\neq 1),(x\neq y)\}$$
$$L(1,1) = \{(x=1),(x\neq 0),(y=1),(y\neq 0),(x=y)\}$$

Consideremos $K = (S, s_0, \longrightarrow, L)$.

Definimos $\mathcal{A}(K) = (\Sigma, S, s_0, \delta, A)$, donde

- $\Sigma = \mathcal{P}(\mathsf{PA})$
- $A = S$
- $s' \in \delta(P, s)$ sii $s \longrightarrow s'$ y $L(s) = P$

Teorema: $\mathcal{L}(K) = \mathcal{L}(\mathcal{A}(K))$

CONICET

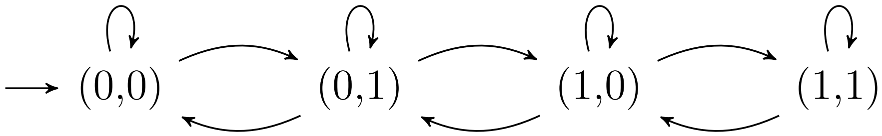
UNC

De estructura de Kripke a Autómata de Büchi

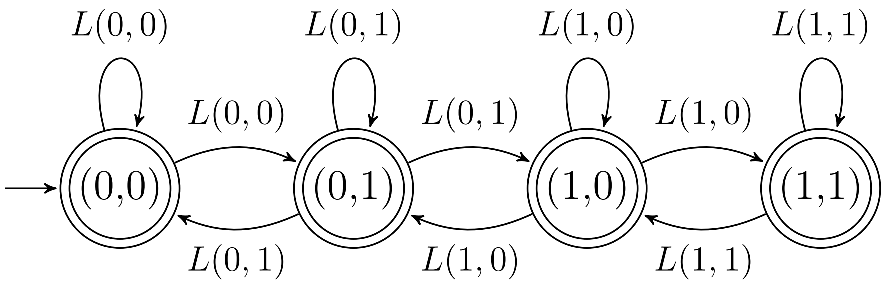
Consideremos $K = (S, s_0, \longrightarrow, L)$.

Definimos $\mathcal{A}(K) = (\Sigma, S, s_0, \delta, A)$, donde

- $\Sigma = \mathcal{P}(\mathsf{PA})$
- $A = S$
- $s' \in \delta(P, s)$ sii $s \longrightarrow s'$ y $L(s) = P$



$$L(0,0) = \{(x = 0), (x \neq 1), (y = 0), (y \neq 1), (x = y)\}$$
$$L(0,1) = \{(x = 0), (x \neq 1), (y = 1), (y \neq 0), (x \neq y)\}$$
$$L(1,0) = \{(x = 1), (x \neq 0), (y = 0), (y \neq 1), (x \neq y)\}$$
$$L(1,1) = \{(x = 1), (x \neq 0), (y = 1), (y \neq 0), (x = y)\}$$



Teorema: $\mathcal{L}(K) = \mathcal{L}(\mathcal{A}(K))$

De LTL a Autómata de Büchi

Teorema: Para toda fórmula LTL ϕ, existe un autómata de Büchi $\mathcal{A}(\phi)$ tal que $\mathcal{L}(\phi) = \mathcal{L}(\mathcal{A}(\phi))$.

$\mathcal{A}(\phi)$ crece exponencialmente y tiene una cota inferior de $2^{O(|\phi|)}$.

De LTL a Autómata de Büchi

Teorema: Para toda fórmula LTL ϕ, existe un autómata de Büchi $\mathcal{A}(\phi)$ tal que $\mathcal{L}(\phi) = \mathcal{L}(\mathcal{A}(\phi))$.

$\mathcal{A}(\phi)$ crece exponencialmente y tiene una cota inferior de $2^{O(|\phi|)}$.

La demostración de este teorema es compleja

Ver, por ejemplo,

- C. Baier & J.-P. Katoen. “Concepts of Model Checking”. MIT Press, 2008. (Capítulo 5)
- E.M. Clarke Jr., O. Grumberg, D.A. Peled. “Model Checking”. MIT Press, 1999. (Capítulo 9)

CONICET

UNC

De LTL a Autómata de Büchi

Ejemplos

$\mathtt{F}\,p$

$\mathtt{G}\,p$

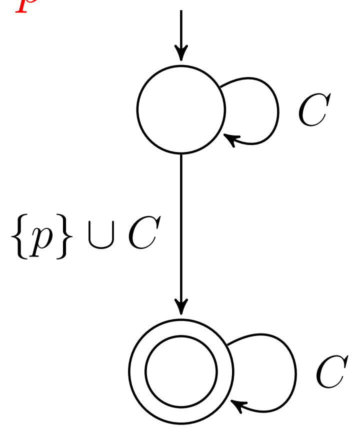
$p\,\mathtt{U}\,q$

$p\,\mathtt{R}\,q$

De LTL a Autómata de Büchi

Ejemplos

$\mathtt{F}\, p$



$\mathtt{G}\, p$

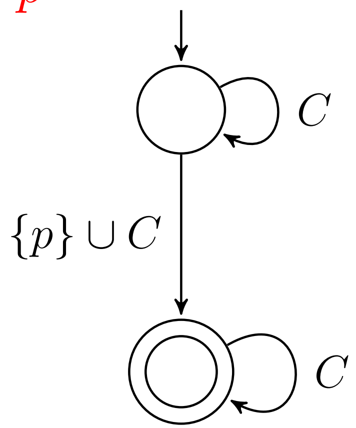
$p\, \mathtt{U}\, q$

$p\, \mathtt{R}\, q$

De LTL a Autómata de Büchi

Ejemplos

$\mathtt{F}\,p$



$\mathtt{G}\,p$

C es cualquier subconjunto de PA.
Es decir, cada flecha representa muchas transiciones a la vez.

$p\,\mathtt{U}\,q$

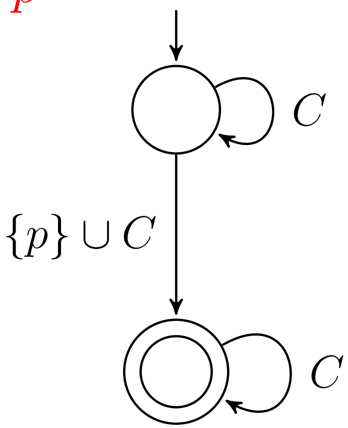
CONICET

UNC

De LTL a Autómata de Büchi

Ejemplos

$\mathtt{F}\, p$



$\mathtt{G}\, p$

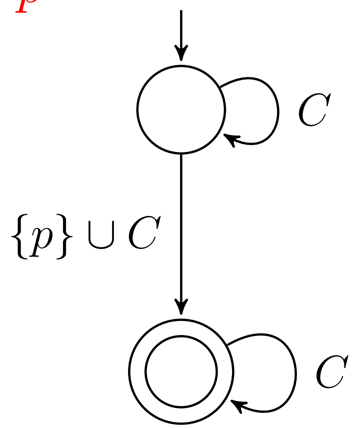
$p\, \mathtt{U}\, q$

$p\, \mathtt{R}\, q$

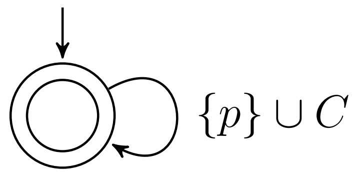
De LTL a Autómata de Büchi

Ejemplos

$\mathtt{F}\,p$



$\mathtt{G}\,p$



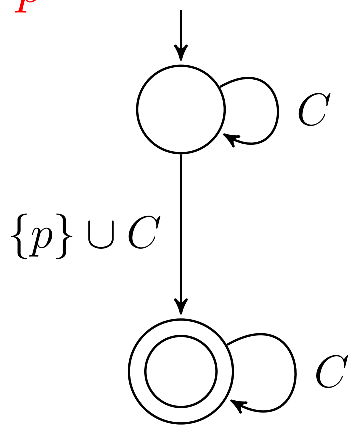
$p\,\mathtt{U}\,q$

$p\,\mathtt{R}\,q$

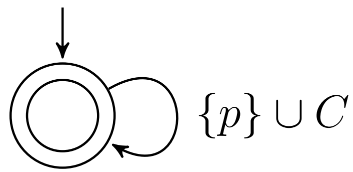
De LTL a Autómata de Büchi

Ejemplos

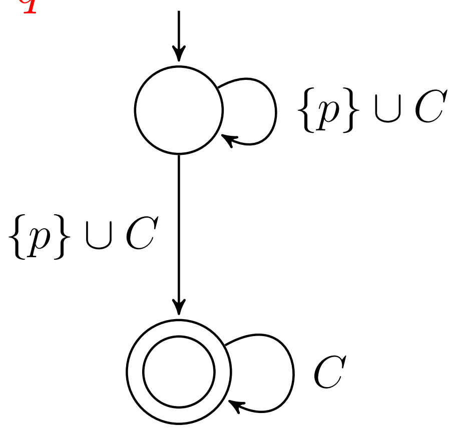
$\mathtt{F}\,p$



$\mathtt{G}\,p$



$p\,\mathtt{U}\,q$

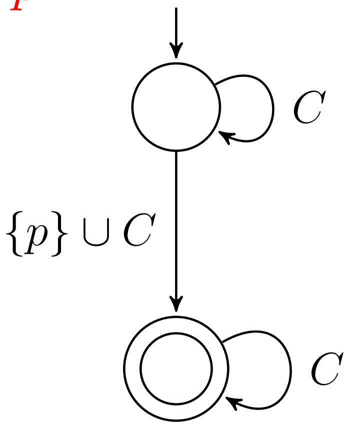


$p\,\mathtt{R}\,q$

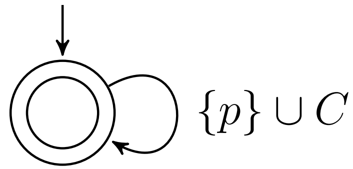
De LTL a Autómata de Büchi

Ejemplos

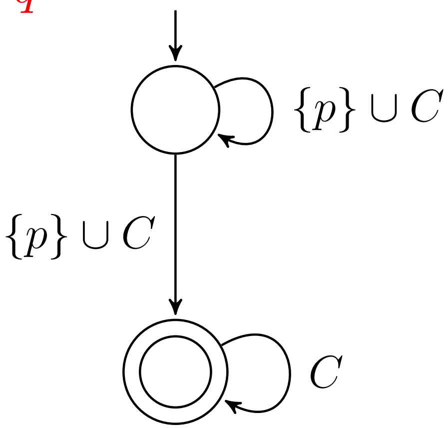
$\mathtt{F}\,p$



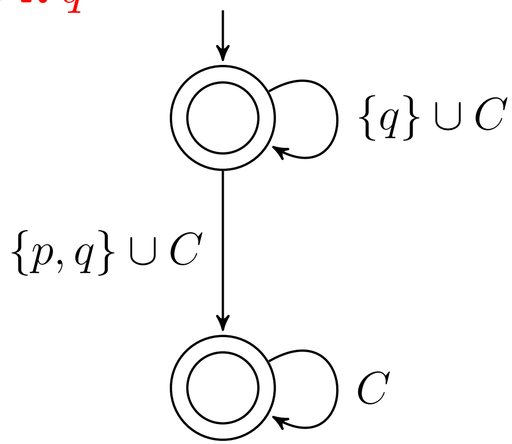
$\mathtt{G}\,p$



$p\,\mathtt{U}\,q$



$p\,\mathtt{R}\,q$



P

```
while true do                          ||   while true do
  if                                   ||     y := 1 ;
    [] (x > 0) ∧ (y = 0) →             ||     y := 0
         x := x − 1                    ||   od
    [] (x < 2) ∧ (y = 0) →             ||
         x := x + 1                    ||
  fi                                   ||
od                                     ||
```

$\models$

$\mathtt{G}((\mathtt{y} = 1) \Rightarrow \mathtt{X}(\mathtt{y} = 0))$

ϕ

CONICET

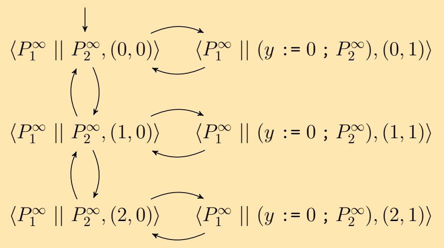
UNC

P

while true **do**
  if
    $[]\ (x > 0) \wedge (y = 0) \rightarrow$
      $x := x - 1$
    $[]\ (x < 2) \wedge (y = 0) \rightarrow$
      $x := x + 1$
  fi
od
	while true **do**
	  $y := 1$;
	  $y := 0$
	od

$\models$ $\mathtt{G}((y = 1) \Rightarrow \mathtt{X}(y = 0))$ ϕ

$K(P)$

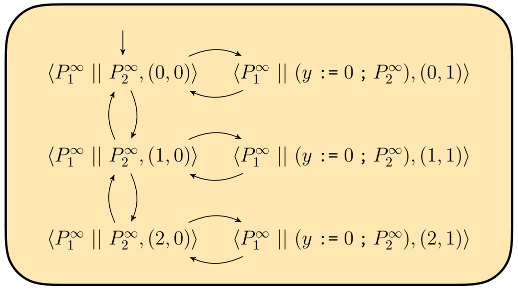


P

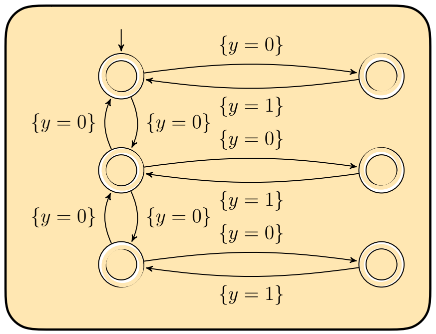
```
while true do                        ||   while true do
  if                                 ||     y := 1 ;
     [] (x > 0) ∧ (y = 0) →          ||     y := 0
          x := x − 1                 ||   od
     [] (x < 2) ∧ (y = 0) →          ||
          x := x + 1                 ||
  fi                                 ||
od                                   ||
```

$\models$ $\mathtt{G}((y = 1) \Rightarrow \mathtt{X}(y = 0))$ ϕ

$K(P)$



$\mathcal{A}(K(P))$

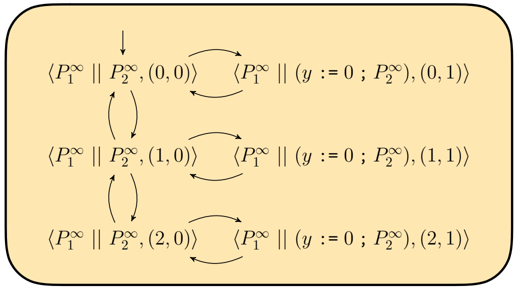


P

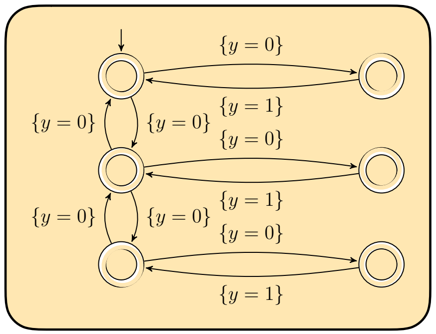
```
while true do
  if
    [] (x > 0)∧(y = 0) →
        x := x − 1
    [] (x < 2)∧(y = 0) →
        x := x + 1
  fi
od
                              ||
while true do
  y := 1 ;
  y := 0
od
```

$\models$ $\mathtt{G}((\mathtt{y}=1) \Rightarrow \mathtt{X}(\mathtt{y}=0))$ ϕ

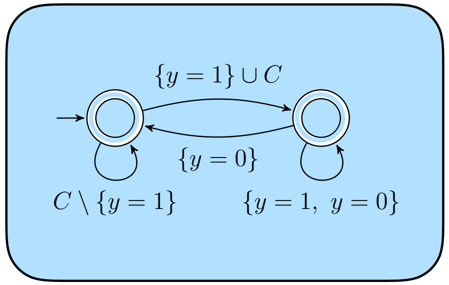
$K(P)$

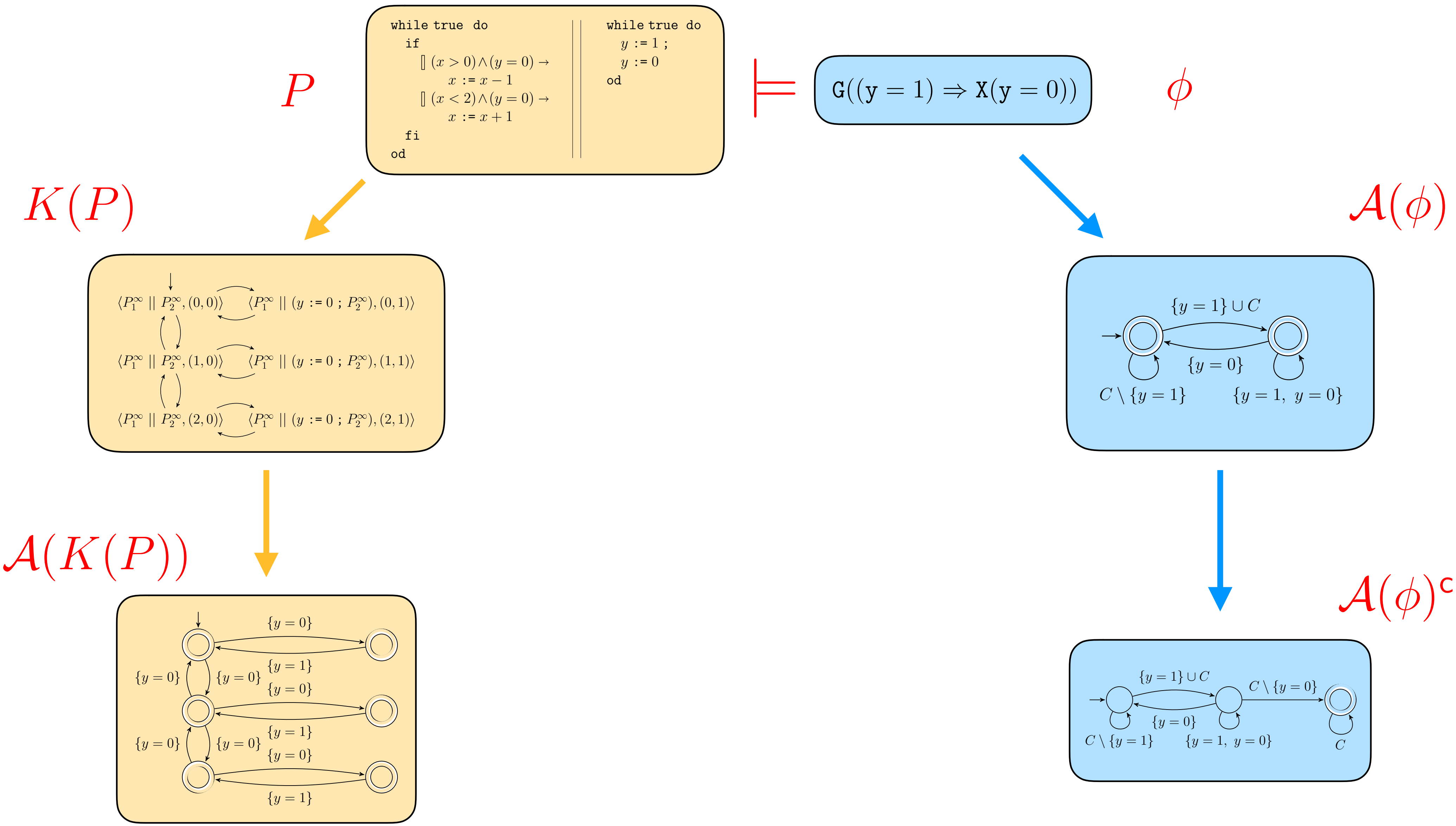


$\mathcal{A}(K(P))$



$\mathcal{A}(\phi)$

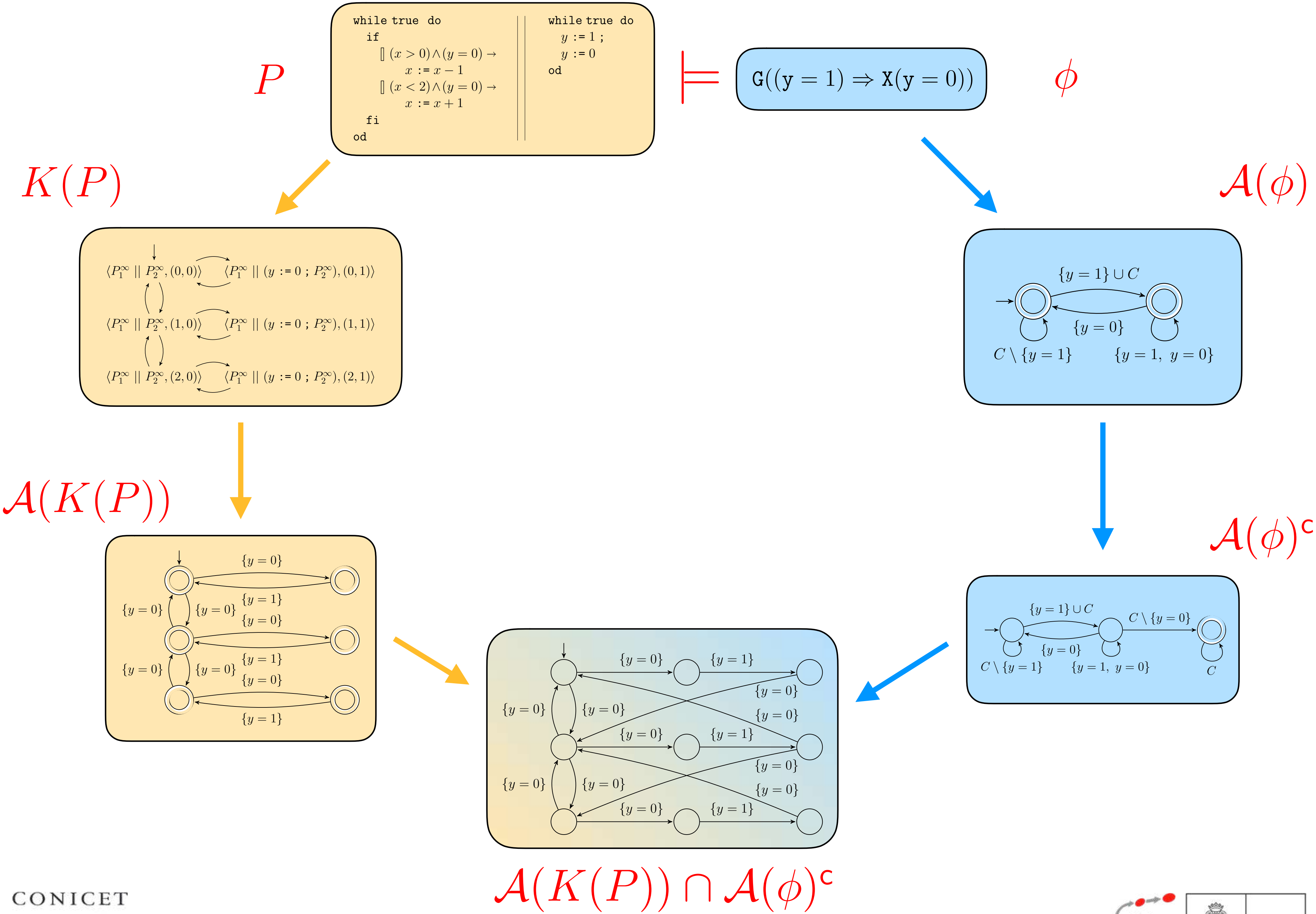




P
while true do
if
[] (x > 0) ∧ (y = 0) →
x := x − 1
[] (x < 2) ∧ (y = 0) →
x := x + 1
fi
od
while true do
y := 1 ;
y := 0
od
⊨
G((y = 1) ⇒ X(y = 0))
φ
K(P)
⟨P1∞ || P2∞, (0, 0)⟩
⟨P1∞ || (y := 0 ; P2∞), (0, 1)⟩
⟨P1∞ || P2∞, (1, 0)⟩
⟨P1∞ || (y := 0 ; P2∞), (1, 1)⟩
⟨P1∞ || P2∞, (2, 0)⟩
⟨P1∞ || (y := 0 ; P2∞), (2, 1)⟩
A(K(P))
{y = 0}
{y = 1}
A(φ)
{y = 1} ∪ C
{y = 0}
C \ {y = 1}
{y = 1, y = 0}
A(φ)c
C \ {y = 0}
C

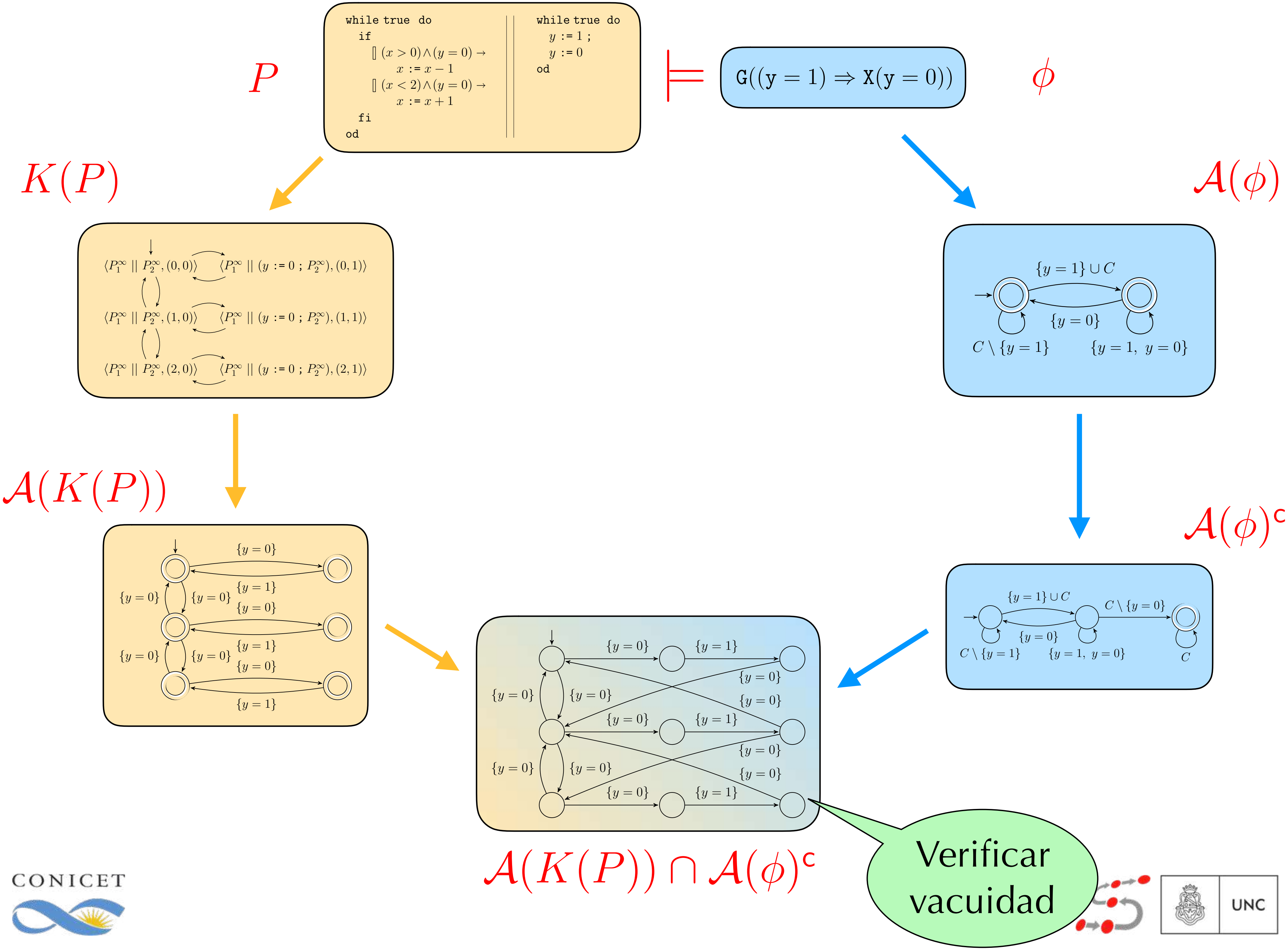
CONICET

UNC

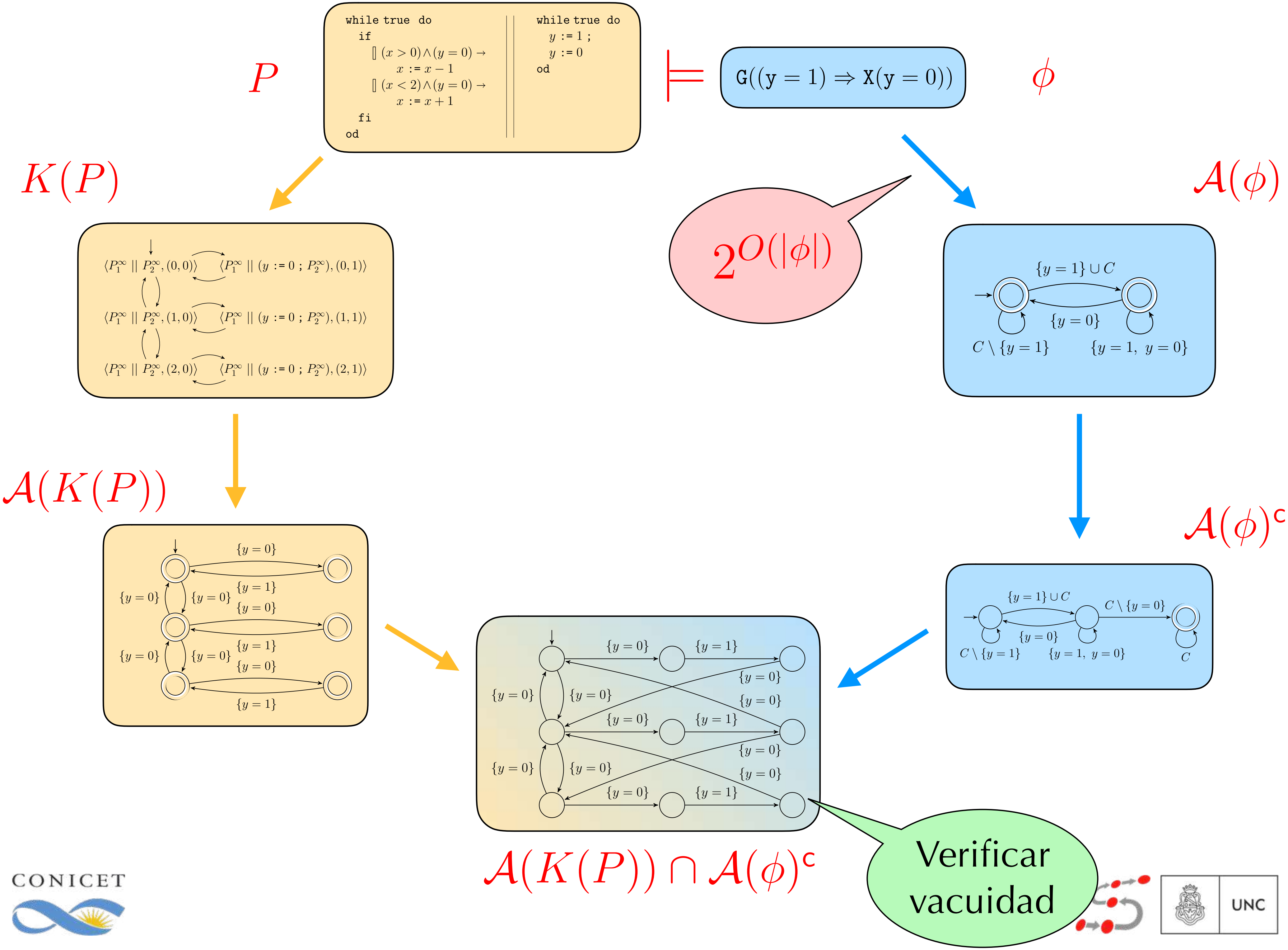

P
while true do
if
[] (x > 0) ∧ (y = 0) →
x := x − 1
[] (x < 2) ∧ (y = 0) →
x := x + 1
fi
od
while true do
y := 1 ;
y := 0
od
⊨
G((y = 1) ⇒ X(y = 0))
ϕ
K(P)
A(ϕ)
{y = 1} ∪ C
{y = 0}
C \ {y = 1}
{y = 1, y = 0}
A(K(P))
{y = 0}
{y = 1}
A(ϕ)^c
C \ {y = 0}
C
A(K(P)) ∩ A(ϕ)^c

CONICET

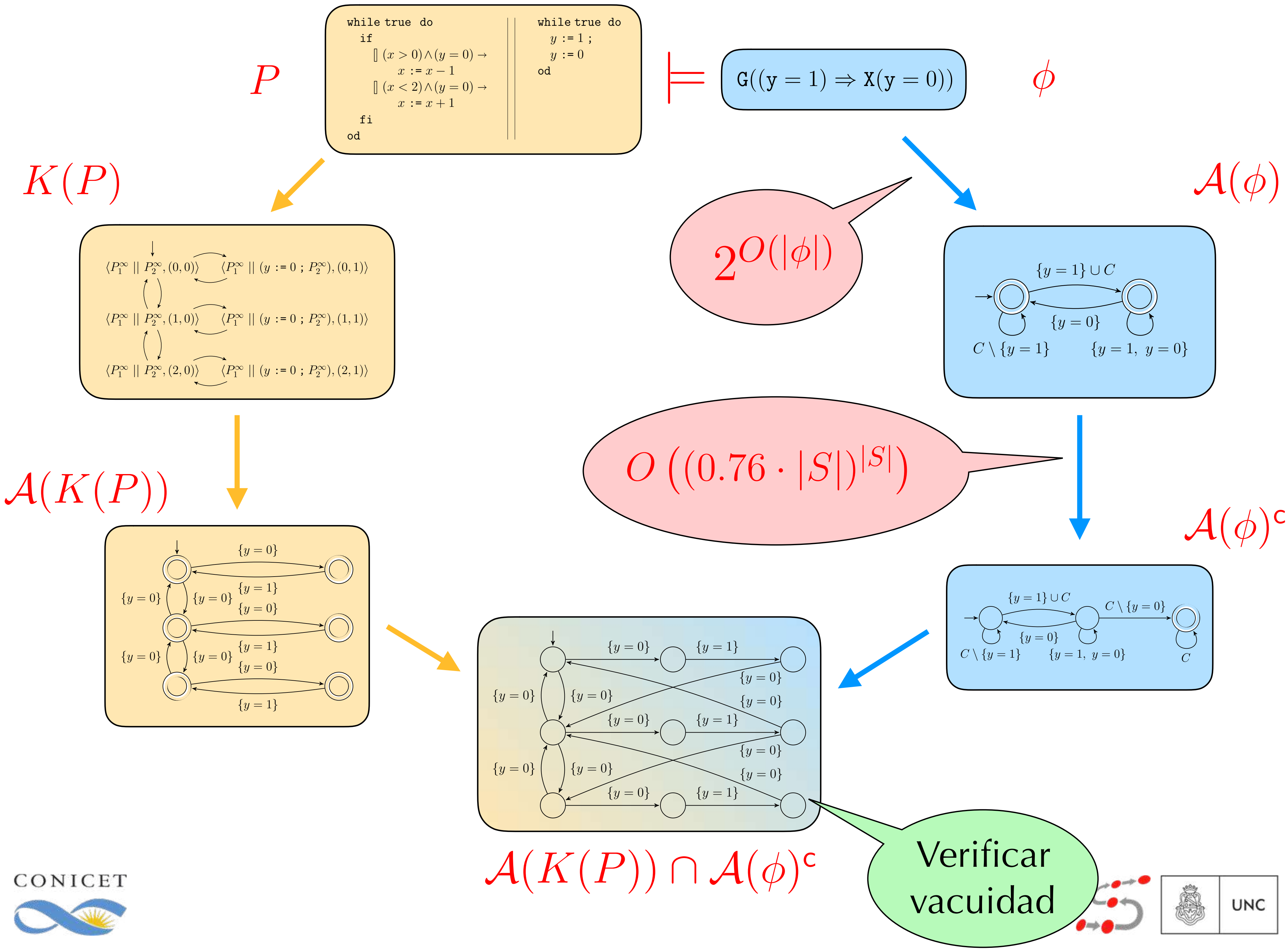
UNC



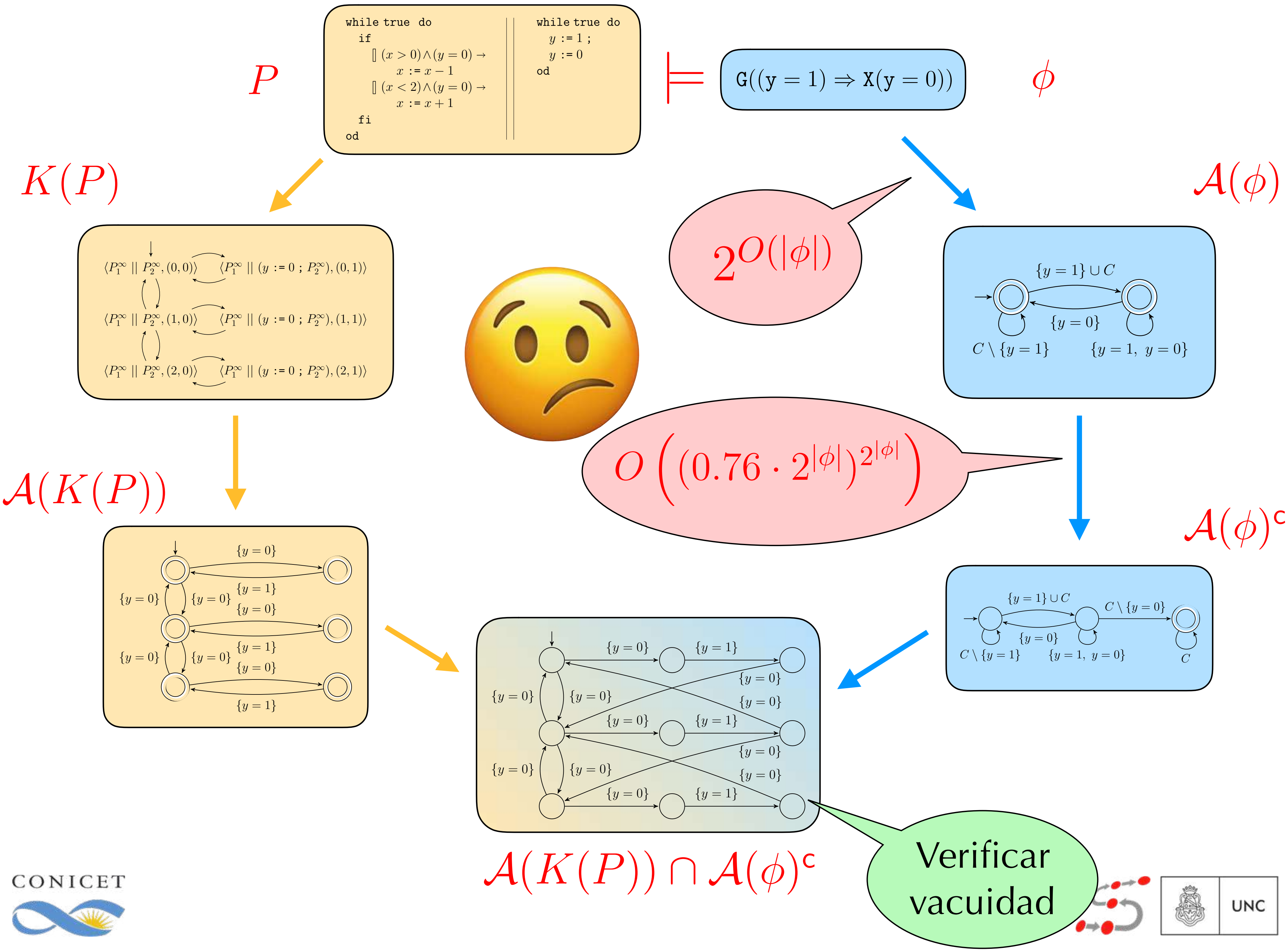
P
while true do
if
[] (x > 0)∧(y = 0) →
x := x − 1
[] (x < 2)∧(y = 0) →
x := x + 1
fi
od
while true do
y := 1 ;
y := 0
od
⊨
G((y = 1) ⇒ X(y = 0))
ϕ
K(P)
A(ϕ)
A(K(P))
A(ϕ)^c
A(K(P)) ∩ A(ϕ)^c
Verificar vacuidad
CONICET
UNC



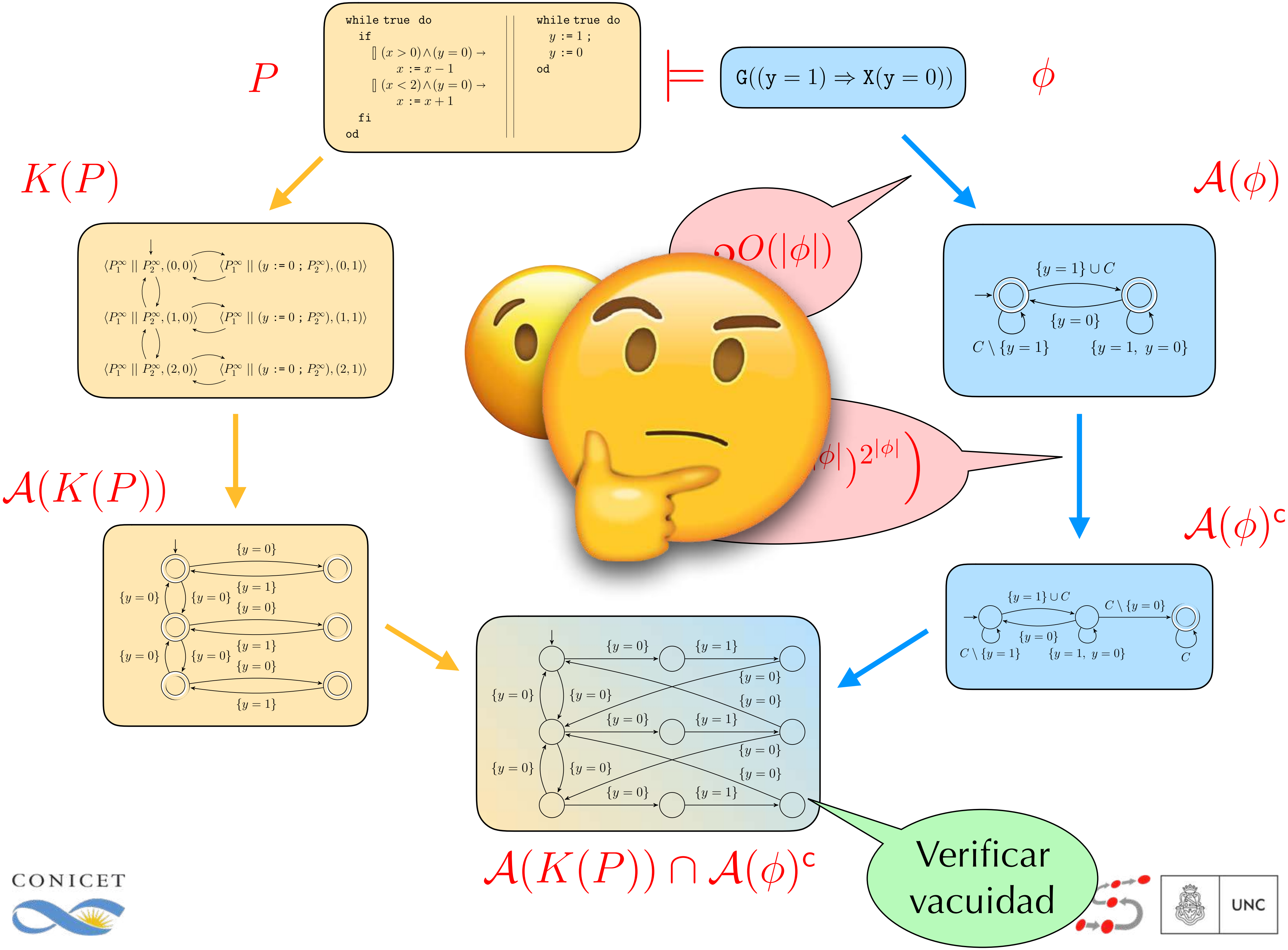
while true do
if
[] (x > 0)∧(y = 0) →
x := x − 1
[] (x < 2)∧(y = 0) →
x := x + 1
fi
od
while true do
y := 1 ;
y := 0
od
P
⊨
G((y = 1) ⇒ X(y = 0))
ϕ
K(P)
A(ϕ)
2^{O(|ϕ|)}
A(K(P))
A(ϕ)^c
A(K(P)) ∩ A(ϕ)^c
Verificar vacuidad
CONICET
UNC



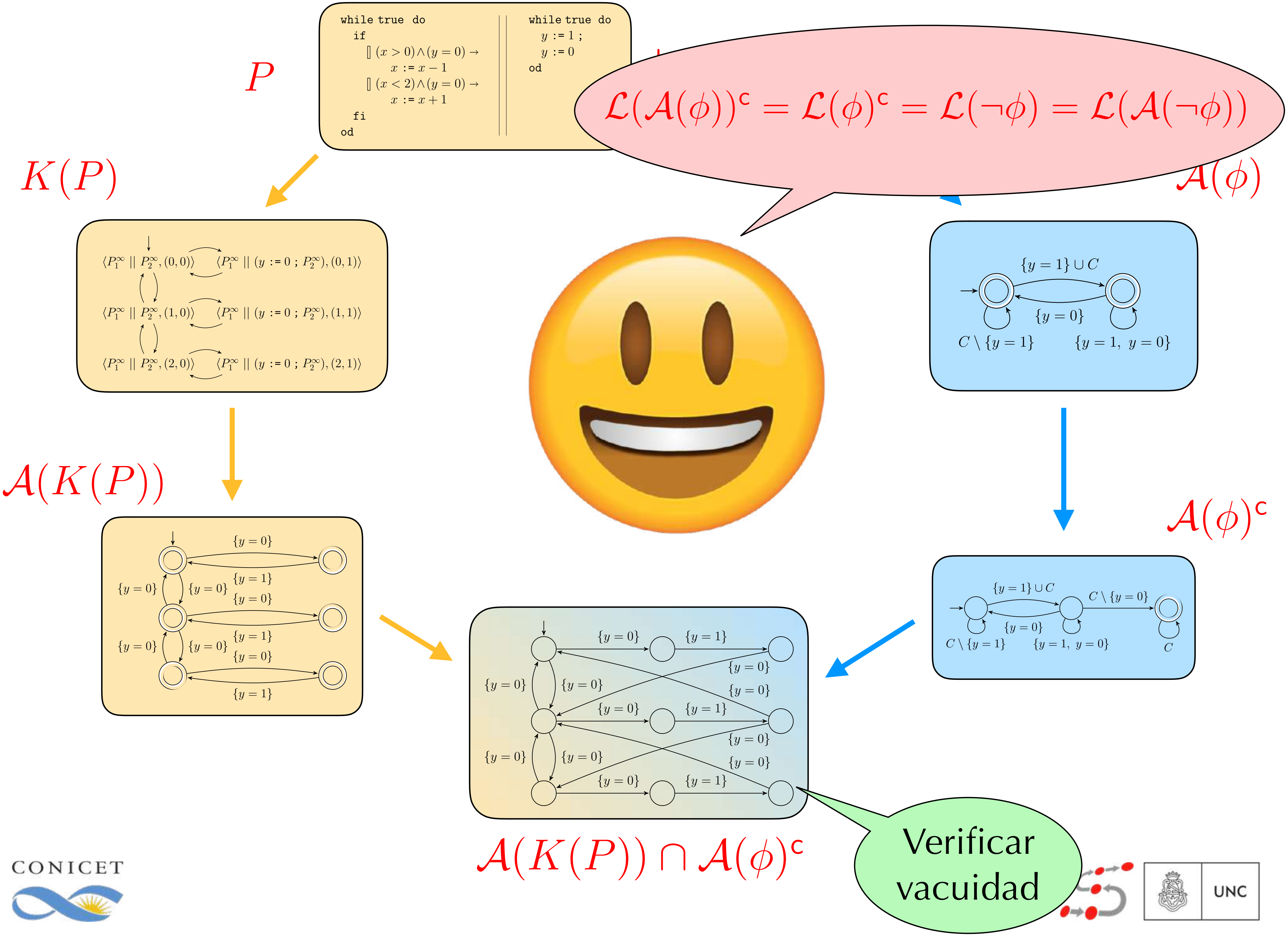
P
while true do
if
[] (x > 0) ∧ (y = 0) →
x := x − 1
[] (x < 2) ∧ (y = 0) →
x := x + 1
fi
od
||
while true do
y := 1 ;
y := 0
od
⊨
G((y = 1) ⇒ X(y = 0))
φ
K(P)
⟨P₁^∞ || P₂^∞, (0,0)⟩
⟨P₁^∞ || (y := 0 ; P₂^∞), (0,1)⟩
⟨P₁^∞ || P₂^∞, (1,0)⟩
⟨P₁^∞ || (y := 0 ; P₂^∞), (1,1)⟩
⟨P₁^∞ || P₂^∞, (2,0)⟩
⟨P₁^∞ || (y := 0 ; P₂^∞), (2,1)⟩
A(φ)
2^{O(|φ|)}
{y = 1} ∪ C
{y = 0}
C \ {y = 1}
{y = 1, y = 0}
O((0.76 · |S|)^{|S|})
A(K(P))
{y = 0}
{y = 1}
A(φ)^c
{y = 1} ∪ C
C \ {y = 0}
{y = 0}
C \ {y = 1}
{y = 1, y = 0}
C
A(K(P)) ∩ A(φ)^c
Verificar vacuidad
CONICET
UNC



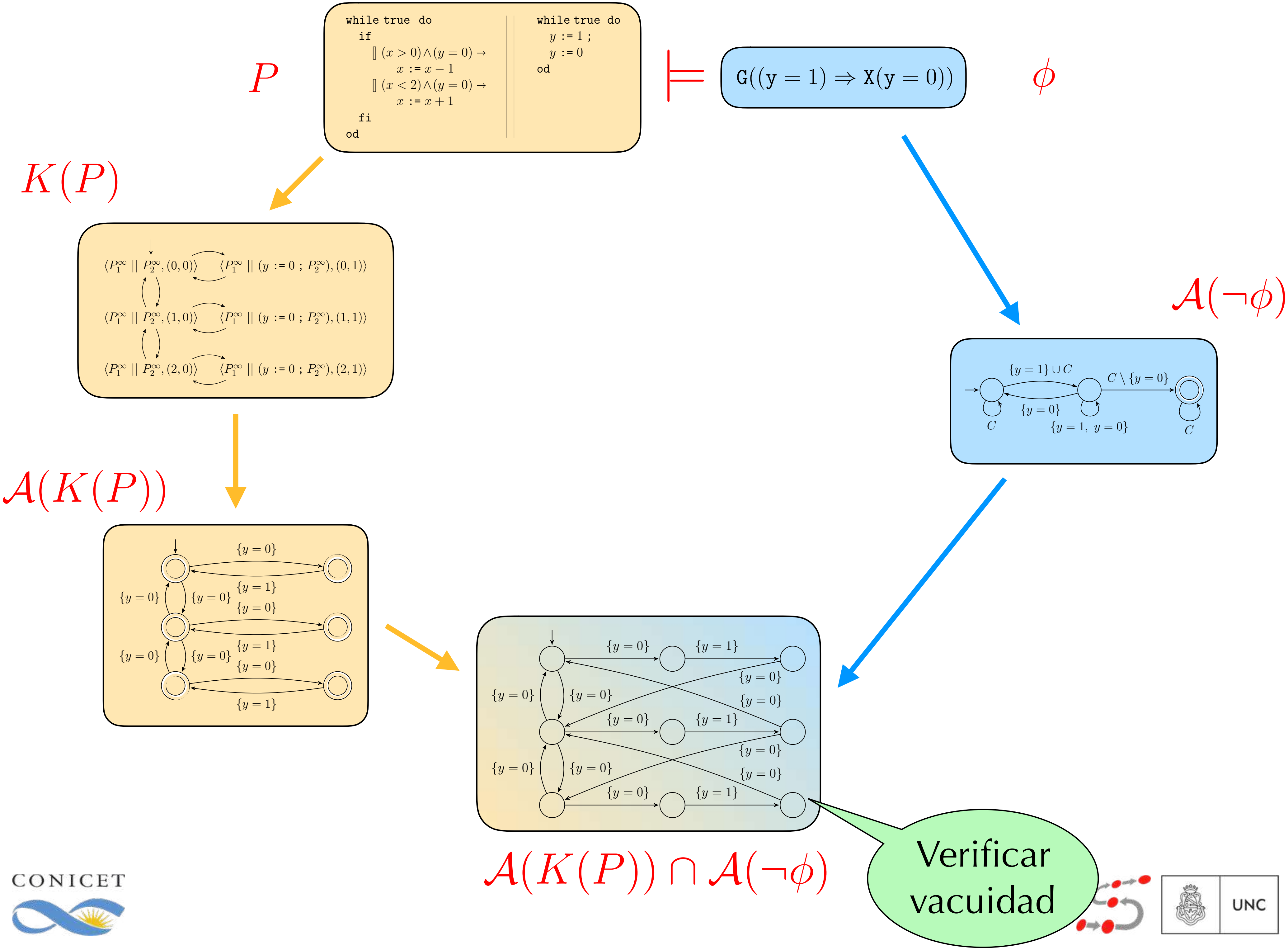
while true do
if
[] $(x > 0) \wedge (y = 0) \rightarrow$
$x := x - 1$
[] $(x < 2) \wedge (y = 0) \rightarrow$
$x := x + 1$
fi
od
while true do
$y := 1$;
$y := 0$
od
P
$\models$
$\mathtt{G}((\mathtt{y}=1) \Rightarrow \mathtt{X}(\mathtt{y}=0))$
ϕ
$K(P)$
$\mathcal{A}(\phi)$
$2^{O(|\phi|)}$
$\langle P_1^\infty \parallel P_2^\infty, (0,0)\rangle$
$\langle P_1^\infty \parallel (y := 0 \,;\, P_2^\infty), (0,1)\rangle$
$\langle P_1^\infty \parallel P_2^\infty, (1,0)\rangle$
$\langle P_1^\infty \parallel (y := 0 \,;\, P_2^\infty), (1,1)\rangle$
$\langle P_1^\infty \parallel P_2^\infty, (2,0)\rangle$
$\langle P_1^\infty \parallel (y := 0 \,;\, P_2^\infty), (2,1)\rangle$
$\{y=1\} \cup C$
$\{y=0\}$
$C \setminus \{y=1\}$
$\{y=1,\ y=0\}$
$\mathcal{A}(K(P))$
$O\left((0.76 \cdot 2^{|\phi|})^{2^{|\phi|}}\right)$
$\mathcal{A}(\phi)^{\mathsf{c}}$
$\{y=0\}$
$\{y=1\}$
$\{y=1\} \cup C$
$C \setminus \{y=0\}$
$\{y=0\}$
$C \setminus \{y=1\}$
$\{y=1,\ y=0\}$
C
$\mathcal{A}(K(P)) \cap \mathcal{A}(\phi)^{\mathsf{c}}$
Verificar vacuidad
CONICET
UNC



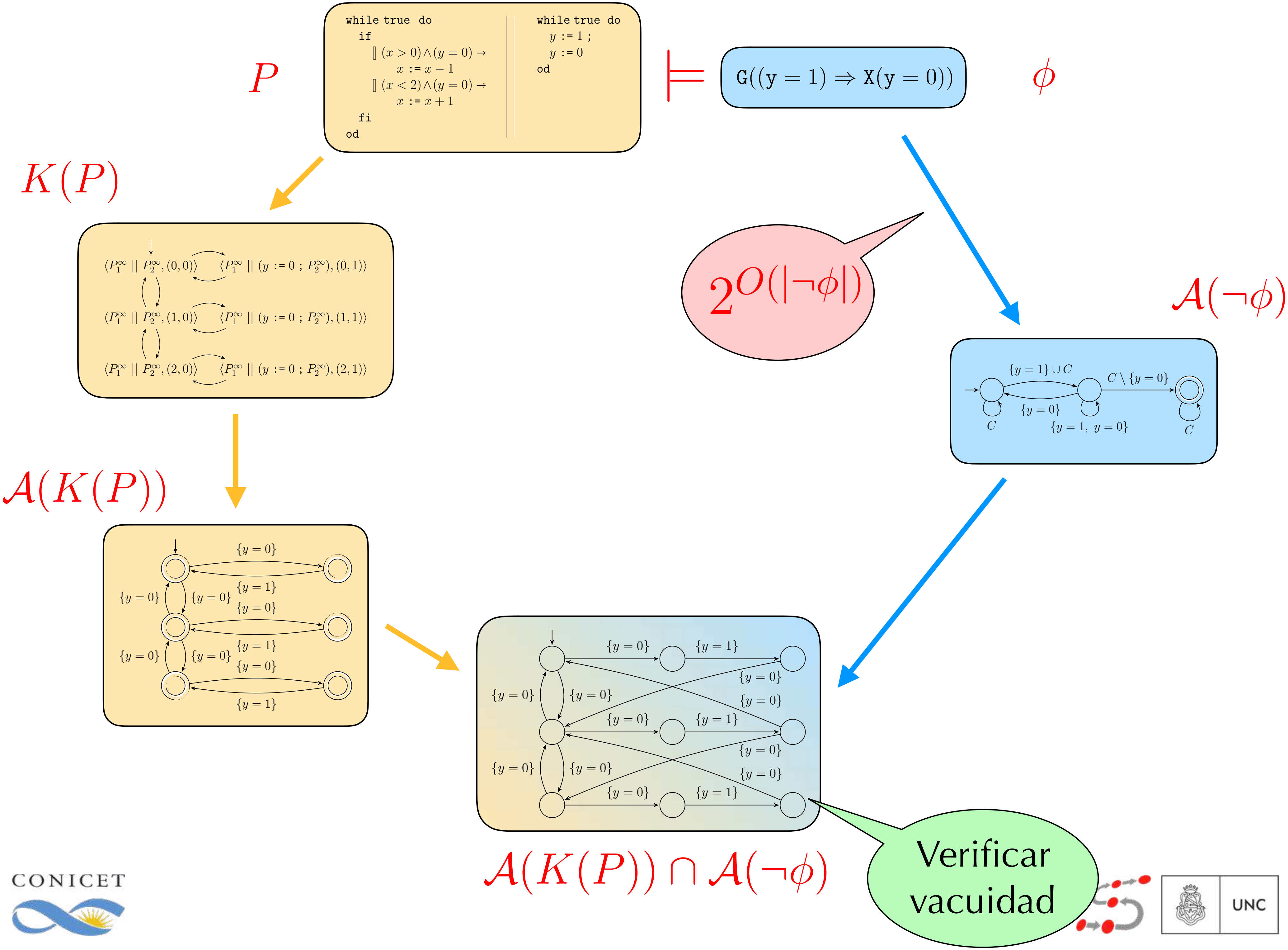
P
while true do
if
$[]\ (x>0) \wedge (y=0) \rightarrow$
$x := x-1$
$[]\ (x<2) \wedge (y=0) \rightarrow$
$x := x+1$
fi
od
while true do
$y := 1$;
$y := 0$
od
$\models$
$\mathtt{G}((\mathtt{y}=1) \Rightarrow \mathtt{X}(\mathtt{y}=0))$
ϕ
$K(P)$
$\langle P_1^\infty \parallel P_2^\infty, (0,0)\rangle$
$\langle P_1^\infty \parallel (y := 0\ ;\ P_2^\infty), (0,1)\rangle$
$\langle P_1^\infty \parallel P_2^\infty, (1,0)\rangle$
$\langle P_1^\infty \parallel (y := 0\ ;\ P_2^\infty), (1,1)\rangle$
$\langle P_1^\infty \parallel P_2^\infty, (2,0)\rangle$
$\langle P_1^\infty \parallel (y := 0\ ;\ P_2^\infty), (2,1)\rangle$
$\mathcal{A}(\phi)$
$\{y=1\} \cup C$
$\{y=0\}$
$C \setminus \{y=1\}$
$\{y=1,\ y=0\}$
$2^{O(|\phi|)}$
$|\phi|)2^{|\phi|})$
$\mathcal{A}(K(P))$
$\{y=0\}$
$\{y=1\}$
$\mathcal{A}(\phi)^{\mathsf{c}}$
$\{y=1\} \cup C$
$C \setminus \{y=0\}$
$\{y=0\}$
$C \setminus \{y=1\}$
$\{y=1,\ y=0\}$
C
$\{y=0\}$
$\{y=1\}$
$\mathcal{A}(K(P)) \cap \mathcal{A}(\phi)^{\mathsf{c}}$
Verificar vacuidad
CONICET
UNC



P
while true do
if
$\| \; (x > 0) \wedge (y = 0) \rightarrow$
$x := x - 1$
$\| \; (x < 2) \wedge (y = 0) \rightarrow$
$x := x + 1$
fi
od
while true do
$y := 1$;
$y := 0$
od
$\mathcal{L}(\mathcal{A}(\phi))^{\mathsf{c}} = \mathcal{L}(\phi)^{\mathsf{c}} = \mathcal{L}(\neg\phi) = \mathcal{L}(\mathcal{A}(\neg\phi))$
$K(P)$
$\langle P_1^\infty \parallel P_2^\infty, (0,0)\rangle$
$\langle P_1^\infty \parallel (y := 0 \; ; \; P_2^\infty), (0,1)\rangle$
$\langle P_1^\infty \parallel P_2^\infty, (1,0)\rangle$
$\langle P_1^\infty \parallel (y := 0 \; ; \; P_2^\infty), (1,1)\rangle$
$\langle P_1^\infty \parallel P_2^\infty, (2,0)\rangle$
$\langle P_1^\infty \parallel (y := 0 \; ; \; P_2^\infty), (2,1)\rangle$
$\mathcal{A}(\phi)$
$\{y = 1\} \cup C$
$\{y = 0\}$
$C \setminus \{y = 1\}$
$\{y = 1, \; y = 0\}$
$\mathcal{A}(K(P))$
$\{y = 0\}$
$\{y = 1\}$
$\mathcal{A}(\phi)^{\mathsf{c}}$
$\{y = 1\} \cup C$
$C \setminus \{y = 0\}$
$\{y = 0\}$
$C \setminus \{y = 1\}$
$\{y = 1, \; y = 0\}$
C
$\mathcal{A}(K(P)) \cap \mathcal{A}(\phi)^{\mathsf{c}}$
Verificar vacuidad
CONICET
UNC



P
while true do
if
[] (x > 0)∧(y = 0) → x := x − 1
[] (x < 2)∧(y = 0) → x := x + 1
fi
od
while true do
y := 1 ;
y := 0
od
⊨
G((y = 1) ⇒ X(y = 0))
ϕ
K(P)
⟨P1∞ || P2∞, (0, 0)⟩
⟨P1∞ || (y := 0 ; P2∞), (0, 1)⟩
⟨P1∞ || P2∞, (1, 0)⟩
⟨P1∞ || (y := 0 ; P2∞), (1, 1)⟩
⟨P1∞ || P2∞, (2, 0)⟩
⟨P1∞ || (y := 0 ; P2∞), (2, 1)⟩
A(¬ϕ)
{y = 1} ∪ C
C \ {y = 0}
{y = 0}
{y = 1, y = 0}
C
A(K(P))
{y = 0}
{y = 1}
A(K(P)) ∩ A(¬ϕ)
Verificar vacuidad
CONICET
UNC



P
while true do
if
[] $(x > 0) \wedge (y = 0) \rightarrow$
$x := x - 1$
[] $(x < 2) \wedge (y = 0) \rightarrow$
$x := x + 1$
fi
od
while true do
$y := 1$;
$y := 0$
od
$\models$
$\mathtt{G}((\mathtt{y} = 1) \Rightarrow \mathtt{X}(\mathtt{y} = 0))$
ϕ
$K(P)$
$\langle P_1^\infty \parallel P_2^\infty, (0,0)\rangle$
$\langle P_1^\infty \parallel (y := 0 \,;\, P_2^\infty), (0,1)\rangle$
$\langle P_1^\infty \parallel P_2^\infty, (1,0)\rangle$
$\langle P_1^\infty \parallel (y := 0 \,;\, P_2^\infty), (1,1)\rangle$
$\langle P_1^\infty \parallel P_2^\infty, (2,0)\rangle$
$\langle P_1^\infty \parallel (y := 0 \,;\, P_2^\infty), (2,1)\rangle$
$2^{O(|\neg\phi|)}$
$\mathcal{A}(\neg\phi)$
$\{y = 1\} \cup C$
$C \setminus \{y = 0\}$
$\{y = 0\}$
$\{y = 1,\ y = 0\}$
C
$\mathcal{A}(K(P))$
$\{y = 0\}$
$\{y = 1\}$
$\mathcal{A}(K(P)) \cap \mathcal{A}(\neg\phi)$
Verificar vacuidad

CONICET

UNC

Model Checking: características

- Además de determinar si una propiedad se cumple, dan contraejemplos en caso de que no se cumpla.
- El algoritmo básico se basa en "fuerza bruta": recorre todo el grafo subyacente.
- Esto se agrava con el problema de la explosión de estados.
    - Se agranda exponencialmente con cada variable y cada proceso.
- El grafo subyacente usualmente necesita ser finito.

Model Checking: caracte

¡Importante!
Justificación del error

- Además de determinar si una propiedad se cumple, dan contraejemplos en caso de que no se cumpla.
- El algoritmo básico se basa en "fuerza bruta": recorre todo el grafo subyacente.
- Esto se agrava con el problema de la explosión de estados.
  - Se agranda exponencialmente con cada variable y cada proceso.
- El grafo subyacente usualmente necesita ser finito.

Ejercicio 5: Dé el autómata de Büchi que acepta exactamente todas las palabras que satisfacen $\mathtt{G}\,\mathtt{F}\,\neg \mathit{llueve}$.

Ejercicio 6: Una fórmula LTL ϕ es satisfactible si existe $\sigma \in (\mathcal{P}(\mathsf{PA}))^\omega$ tal que $\sigma \models \phi$. Conisderando las herramientas dadas en el curso, dé un algoritmo para determinar si una fórmula LTL ϕ es satisfactible.

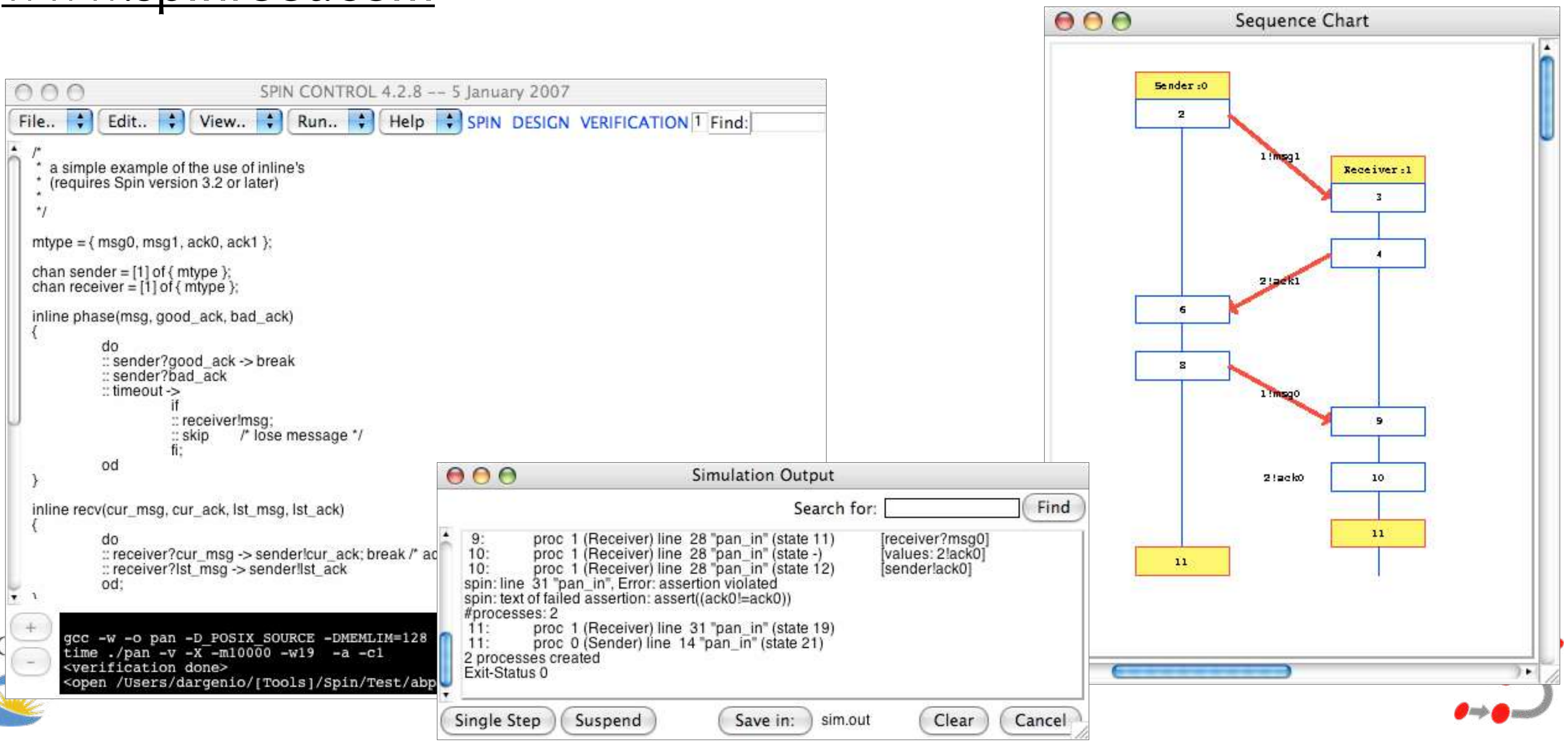
Herramientas de Model Checking

CONICET

UNC

El model checker SPIN

- Desarrollado en AT&T / Bell Labs.
- Principalmente desarrollado por Gerard Holzmann
- Bibliografía:
  - G. Holzmann. The Spin Model Checker. Addisson Wesley. 2004.
- www.spinroot.com

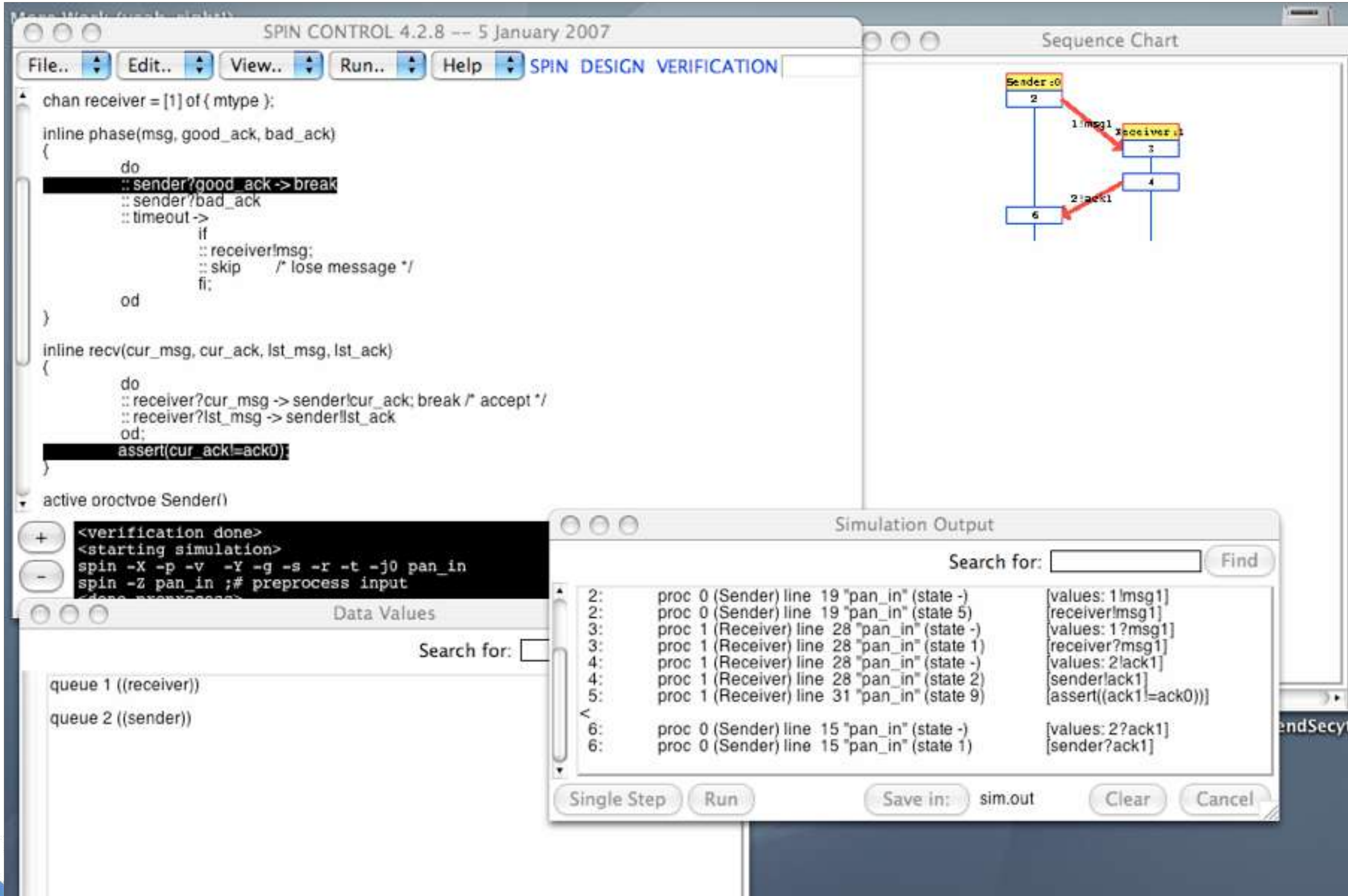


- La descripción de los modelos se realiza en PROMELA
- Promela se asemeja a C y agrega primitivas para manejar concurrencia, canales, atomicidad, no determinismo, ...

```
/*
 *  The alternating bit protocol.
 *  A simple example of the use of inline's
 */

mtype = { msg0, msg1, ack0, ack1 };

chan sender = [1] of { mtype };
chan receiver = [1] of { mtype };

inline phase(msg, good_ack, bad_ack)
{
     do
     :: sender?good_ack -> break
     :: sender?bad_ack
     :: timeout ->
          if
          :: receiver!msg;
          :: skip   /* lose message */
          fi;
     od
}
```

```
inline recv(cur_msg, cur_ack, lst_msg, lst_ack)
{
     do
     :: receiver?cur_msg -> sender!cur_ack;
                             break /* accept */
     :: receiver?lst_msg -> sender!lst_ack
     od;
}

active proctype Sender()
{
     do
     :: phase(msg1, ack1, ack0);
        phase(msg0, ack0, ack1)
     od
}

active proctype Receiver()
{
     do
     :: recv(msg1, ack1, msg0, ack0);
        recv(msg0, ack0, msg1, ack1)
     od
}
```

Permite realizar simulaciones guiadas, aleatorias, sobre una traza específica (ej: contraejemplo de una propiedad).



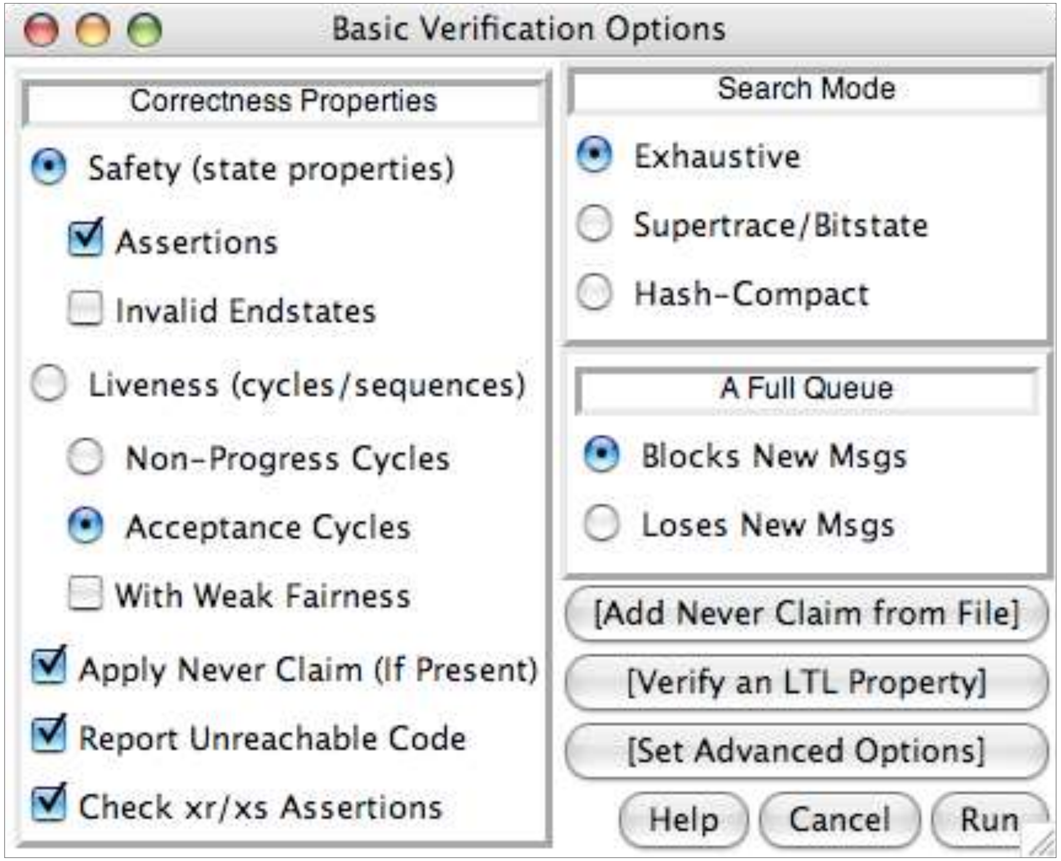
CON

UNC

El model checker SPIN

Permite distintos tipos de verificaciones:

- Propiedades en LTL
- Aserciones dentro del modelo
- Deadlocks
- Progreso
- Permite verificar bajo weak fairness



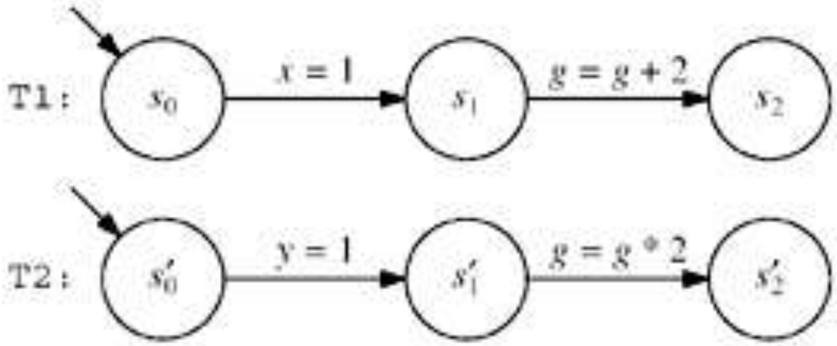
CONICET

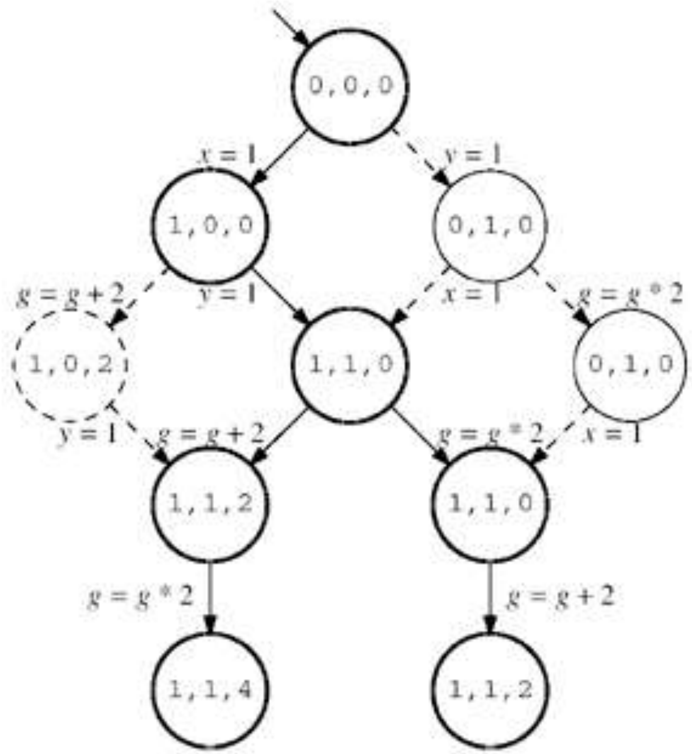
UNC

El model checker SPIN

Técnicas de optimización

- Bitstate hashing: los visitados en el DFS se marcan usando una tabla hash con imagen en {0,1}.
- Reducción por orden parcial: aprovecha la simetría introducida por el “interleaving”:





El model checker SPIN

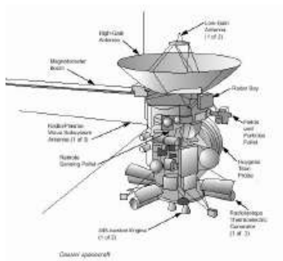
Usos

- Spin se ha utilizado en múltiples ocasiones, y en particular, directamente en la industria (¡se implementó en la industrial!).
- Además es utilizado en la academia para aplicaciones reales (subcontratos/proyectos por parte de empresas).

Ejemplos:

- Verificación de protocolos embebidos en automotores (Bosch)
- Verificación del dique de emergencia climática en Rotterdam.
- Software para el procesamiento de llamadas (Lucent Tech.)
- Diversos algoritmos en proyectos de la NASA como Deep Space 1, Cassini, Mars Exploration Rovers, Deep Impact, etc.

CONICET



UNC

El model checker SMV

- SMV fue originalmente desarrollado por Ken McMillan / Edmund Clarke en Carnegie-Mellon University.
- El SMV original derivó en múltiples versiones:
  - SMV CMU (www.cs.cmu.edu/~modelcheck/smv.html)
  - SMV Cadence (www.kenmcmil.com/smv.html)
  - NuSMV (nusmv.fbk.eu) / nuXmv (nuxmv.fbk.eu)
    - => Elegir éste! (LGPL, más nuevo, mejor mantenido)
- Originalmente destinado a la verificación de hardware.
- Uso en línea de comandos :-(
- Manipula espacio de estados enormes.

- El lenguaje de SMV es bastante básico.
- Describe redes de autómatas (con composición sincrónica o asincrónica según se especifique).
- Descripción de cada autómata bastante declarativa, usando variables y un predicado “next” que permite hablar del valor de las variables en el siguiente estado.
- Simplemente de esa manera se definen las transiciones.

```
MODULE main
VAR
  semaphore : boolean;
  proc1 : process user(semaphore);
  proc2 : process user(semaphore);
ASSIGN
  init(semaphore) := 0;
SPEC
  AG (proc1.state = entering
          -> AF proc1.state = critical)

MODULE user(semaphore)
VAR
  state : {idle,entering,critical,exiting};
ASSIGN
  init(state) := idle;
  next(state) :=
    case
      state = idle : {idle,entering};
      state = entering & !semaphore : critical;
      state = critical : {critical,exiting};
      state = exiting : idle;
      1 : state;
    esac;
  next(semaphore) :=
    case
      state = entering : 1;
      state = exiting : 0;
      1 : semaphore;
    esac;
FAIRNESS
  running
```

CONICET

UNC

Simulación es posible pero a través de la línea de comandos

```
MODULE main
VAR
  semaphore : boolean;
  proc1 : process user(semaphore);
  proc2 : process user(semaphore);
ASSIGN
  init(semaphore) := 0;
SPEC
  AG (proc1.state = entering
           -> AF proc1.state = critical)

MODULE user(semaphore)
VAR
  state : {idle,entering,critical,exiting};
ASSIGN
  init(state) := idle;
  next(state) :=
    case
      state = idle : {idle,entering};
      state = entering & !semaphore : critical;
      state = critical : {critical,exiting};
      state = exiting : idle;
      1 : state;
    esac;
  next(semaphore) :=
    case
      state = entering : 1;
      state = exiting : 0;
      1 : semaphore;
    esac;
FAIRNESS
  running
```

```
MODULE main
VAR
  semaphore : boolean;
  proc1 : process user(semaphore);
  proc2 : process user(semaphore);
ASSIGN
  init(semaphore) := 0;
SPEC
  AG (proc1.state = entering
           -> AF proc1.state = critical)

MODULE user(semaphore)
VAR
  state : {idle,entering,critical,exiting};
ASSIGN
  init(state) := idle;
  next(state) :=
    case
      state = idle : {idle,entering};
      state = entering & !semaphore : critical;
      state = critical : {critical,exiting};
      state = exiting : idle;
      1 : state;
    esac;
  next(semaphore) :=
    case
      state = entering : 1;
      state = exiting : 0;
      1 : semaphore;
    esac;
FAIRNESS
  running
```

Simulación es
de la línea

Las propiedades se
expresan usando las lógicas
CTL, LTL, o PSL

```
MODULE main
VAR
  semaphore : boolean;
  proc1 : process user(semaphore);
  proc2 : process user(semaphore);
ASSIGN
  init(semaphore) := 0;
SPEC
  AG (proc1.state = entering
          -> AF proc1.state = critical)

MODULE user(semaphore)
VAR
  state : {idle,entering,critical,exiting};
ASSIGN
  init(state) := idle;
  next(state) :=
    case
      state = idle : {idle,entering};
      state = entering & !semaphore : critical;
      state = critical : {critical,exiting};
      state = exiting : idle;
      1 : state;
    esac;
  next(semaphore) :=
    case
      state = entering : 1;
      state = exiting : 0;
      1 : semaphore;
    esac;
FAIRNESS
  running
```

Simulación es

la línea

Las propiedades se expresan usando las lógicas CTL, LTL, o PSL

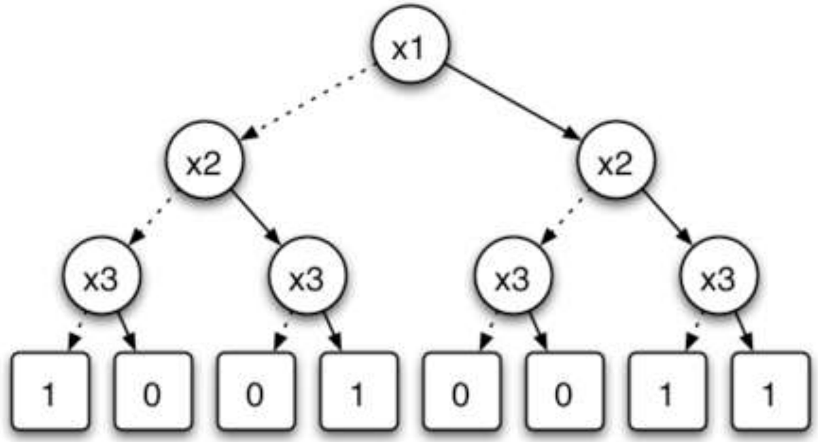
Es posible especificar que se desea hacer la verificación bajo la suposición de fairness

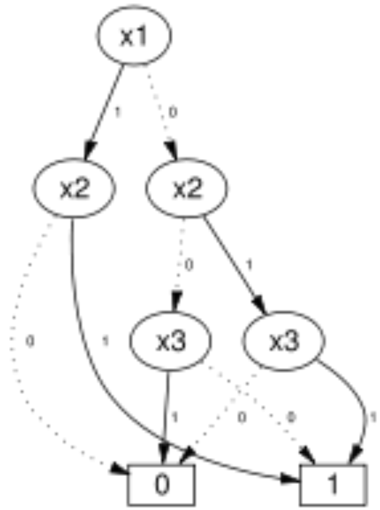
El model checker (Nu)SMV

Técnicas de optimización

Representación del espacio de estado usando BDDs

x1	x2	x3	*f*
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1





proposición a representar $f \equiv (x_2 \Leftrightarrow (x_1 \vee x_3))$

Binary decision tree

Binary decision diagram

Además: Bounded model checking utilizando SAT solvers.

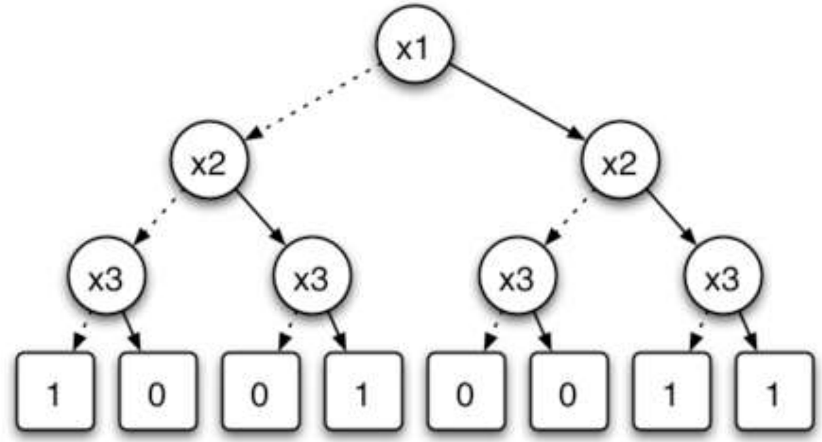
El model checker (Nu)SMV

Técnicas de optimización

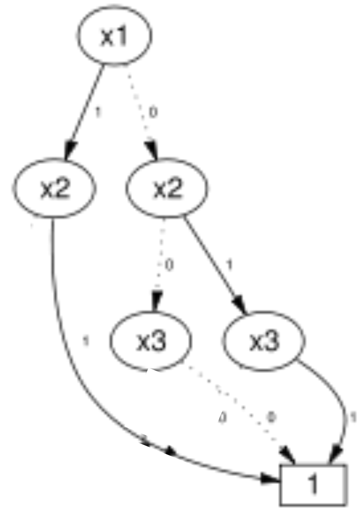
Representación del espacio de estado usando BDDs

x1	x2	x3	*f*
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	0
1	1	0	1
1	1	1	1

proposición a representar
$f \equiv (x_2 \Leftrightarrow (x_1 \vee x_3))$



Binary decision tree



Binary decision diagram

Además: Bounded model checking utilizando SAT solvers.

El model checker (Nu)SMV

Usos

- Desde CMU se proveen servicios a:

  National Science Foundation (NSF); Gigascale Systems Research Center (GSRC); Office of Naval Research (ONR); Army Research Office (ARO); Semiconductor Research Corporation (SRC); General Motors (GM).

- SMV se ha utilizado en múltiples ocaciones pero siempre desde la academia como servicio a la industria. Ej.:
  - Diversos protocolos para coherencia de cache (Gigamax, Futurebus+, etc.)
  - Diversos circuitos lógicos, componentes de procesadores y protocolos
  - Desafortunadamente no se reporta mucho en la literatura.

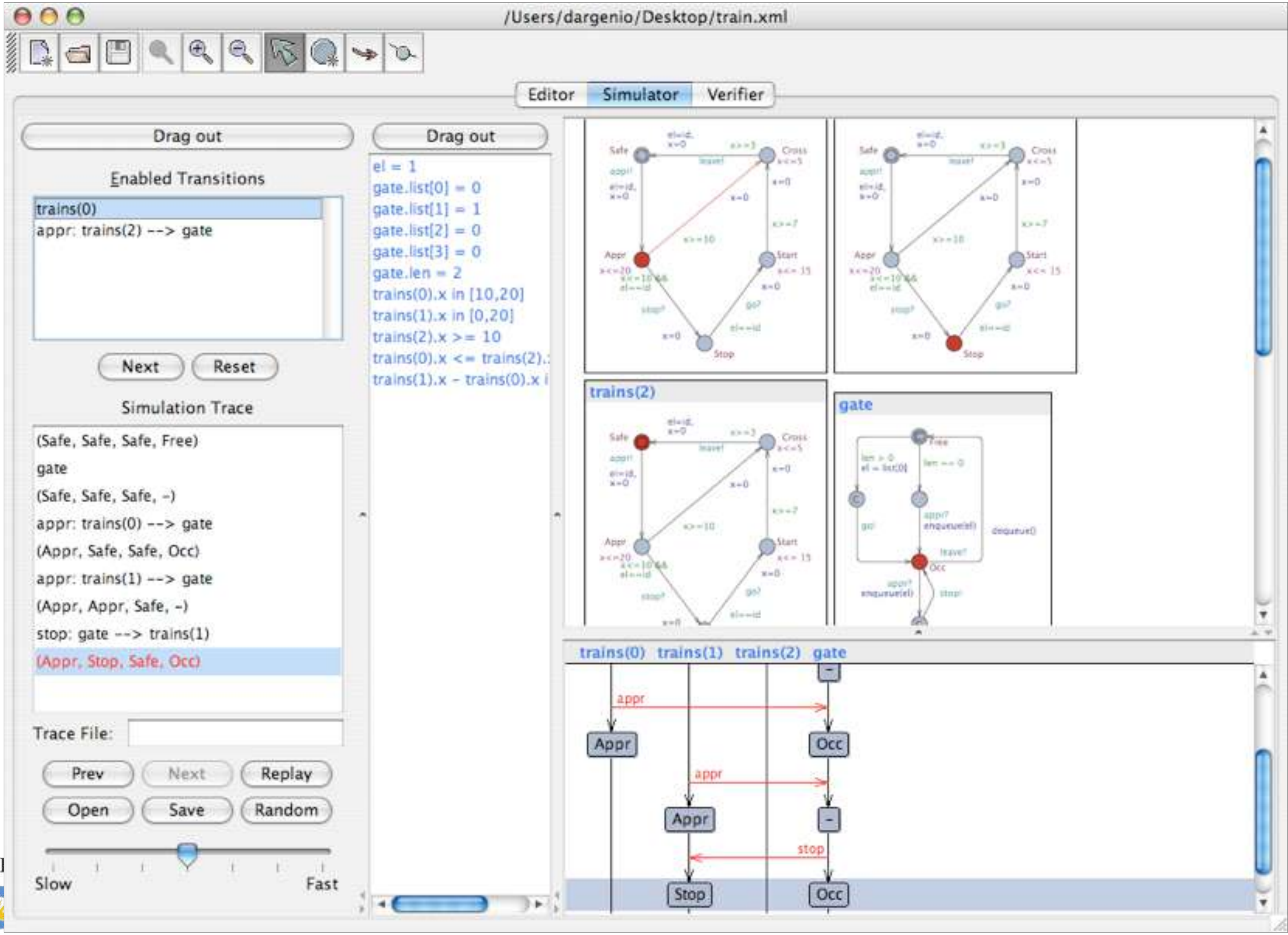
CONICET

UNC

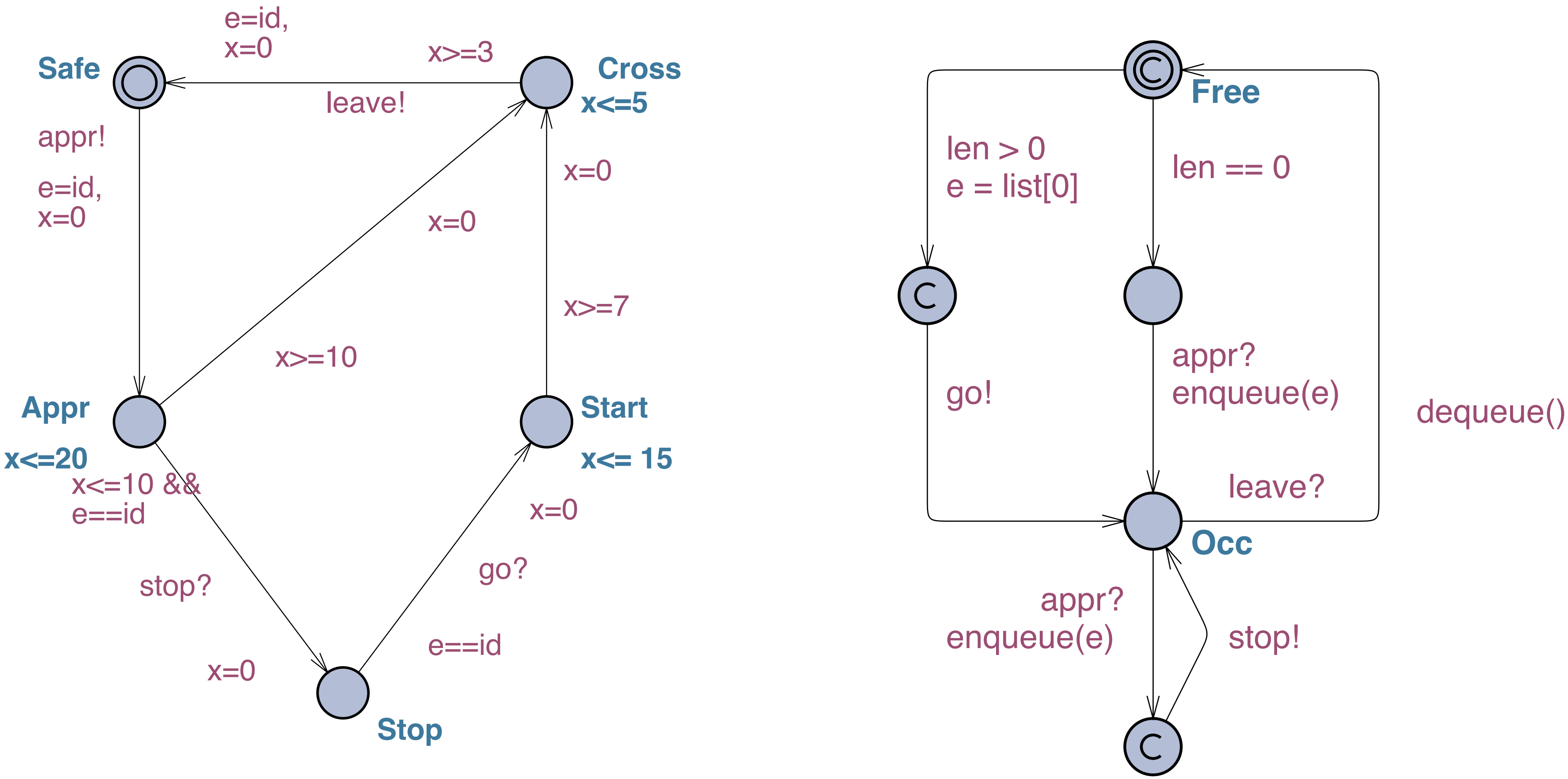
El model checker Uppaal

- Desarrollado en las universidades de Uppsala (SE) y Aalborg (DK).
- Mucha gente intervino en el desarrollo de esta herramienta. Principalmente: Kim Larsen, Wang Yi, Gerd Behrmann, Paul Petterson.
- www.uppaal.org
- Uppaal es un entorno integrado de herramientas para el modelado, simulación y verificación de sistemas de tiempo real.
- Como en SMV, los sistemas se modelan como redes de autómatas (en este caso, temporizados)
- Afortunadamente presenta un entorno gráfico muy amigable.

El model checker Uppaal



El model checker Uppaal

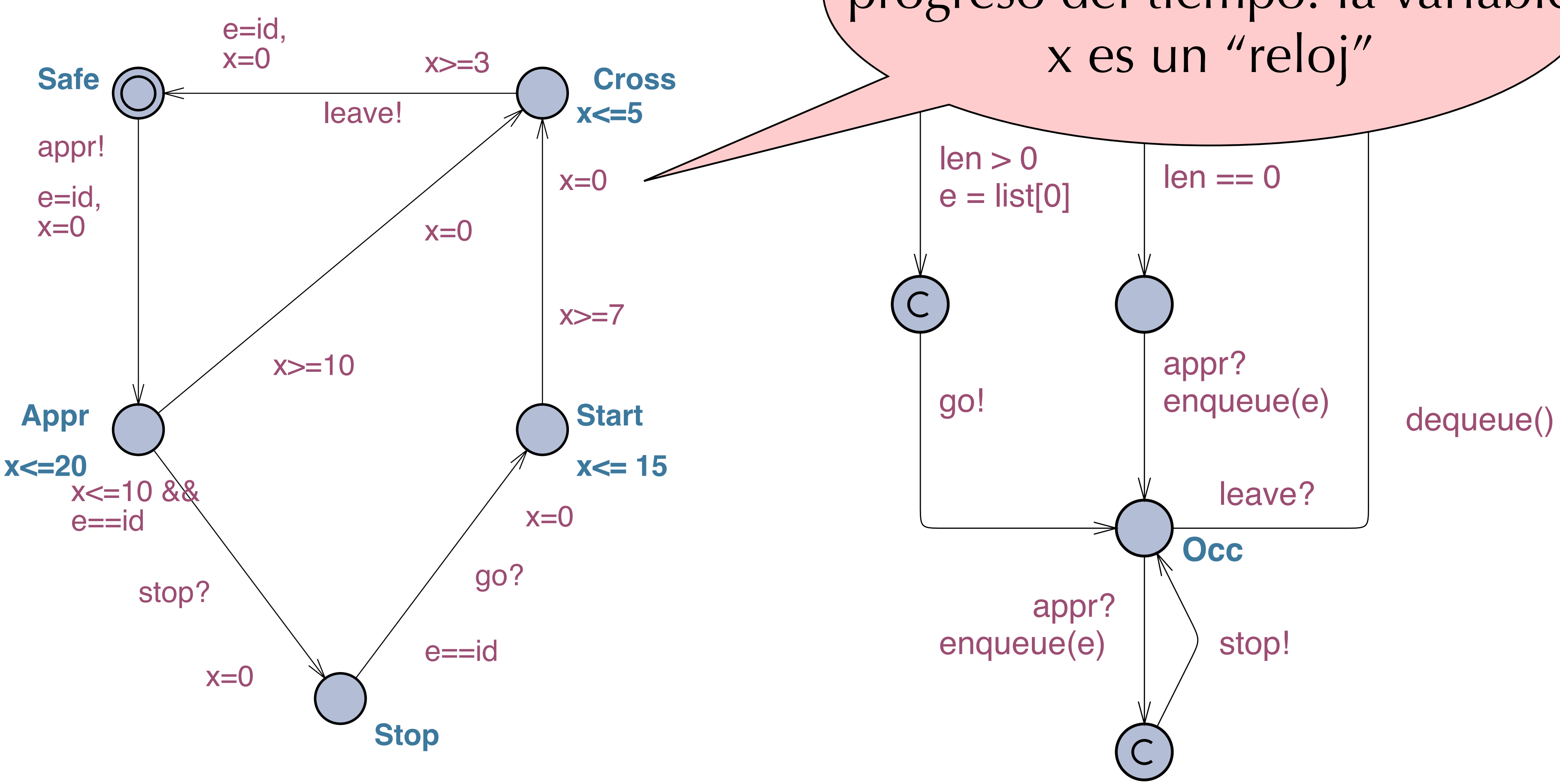


CONICET

UNC

El model checker Uppaal

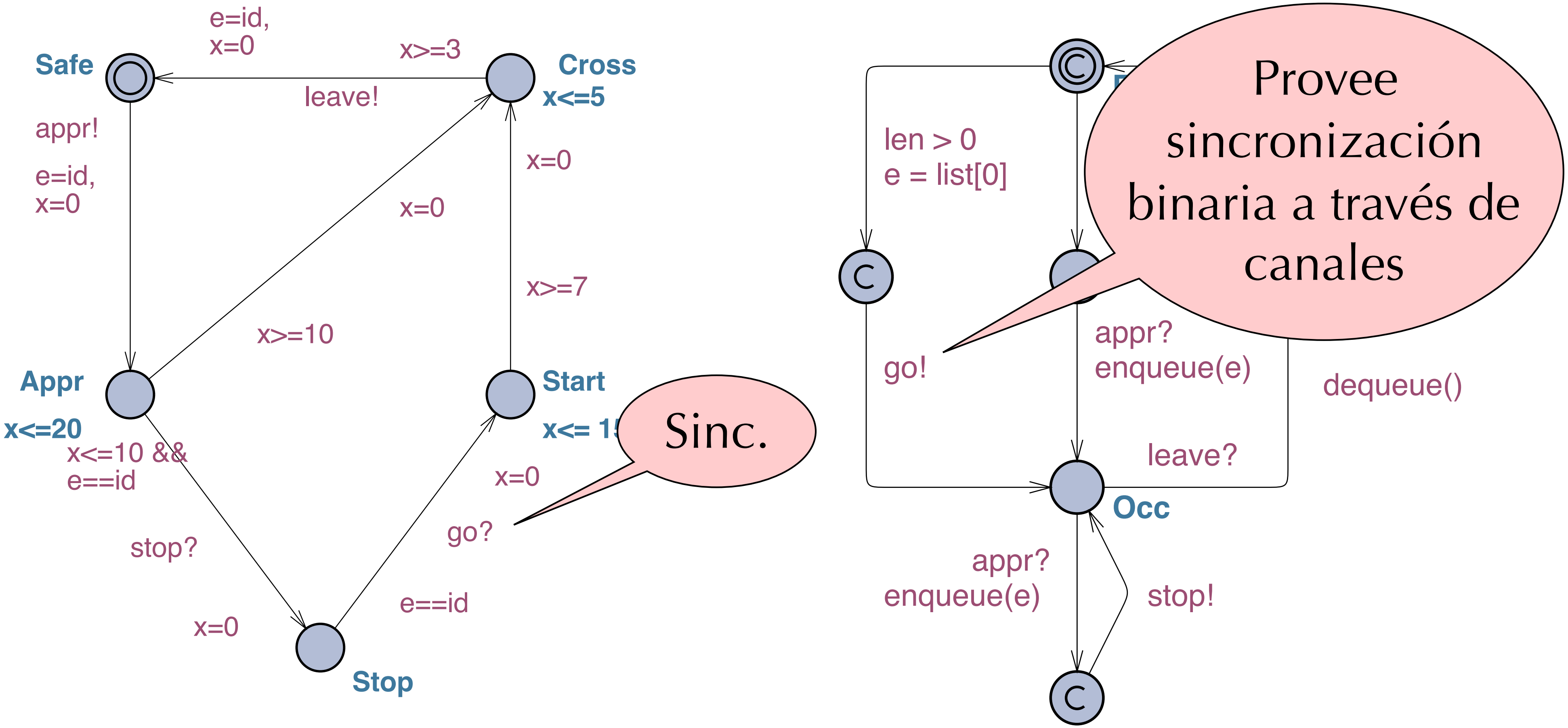
Permite expresar progreso del tiempo: la variable x es un “reloj”



CONICET

UNC

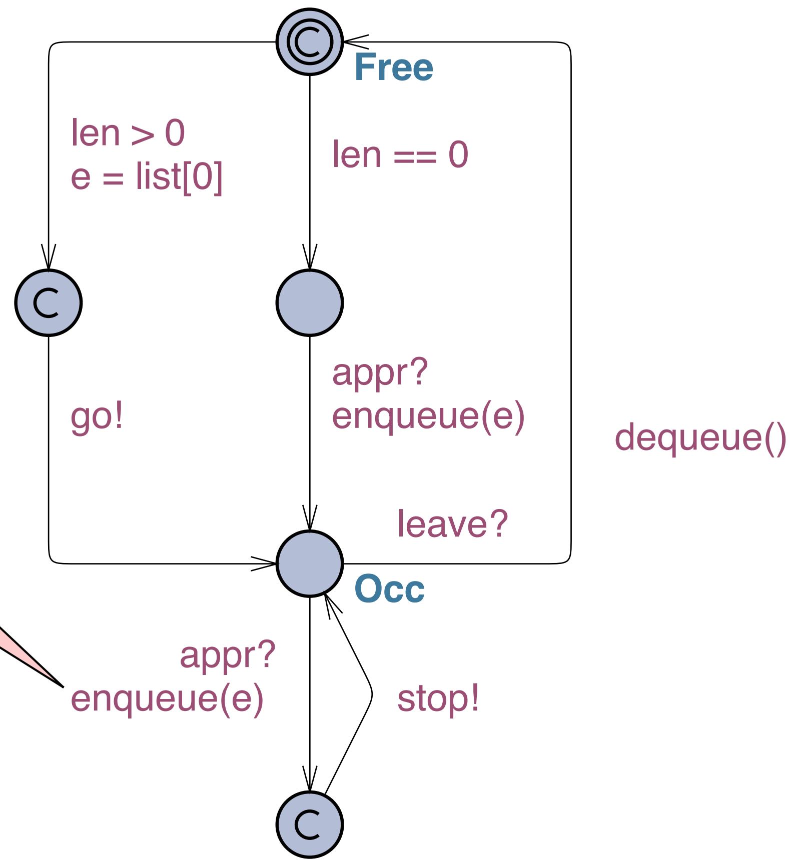
El model checker Uppaal



checker Uppaal

```
id_t list[N+1];
int[0,N] len;

void enqueue(id_t element)
{
        list[len++] = element;
}

void dequeue()
{
        int i = 0;
        len -= 1;
        while (i < len)
        {
                list[i] = list[i + 1];
                i++;
        }
        list[i] = 0;
        i = 0;
}
```

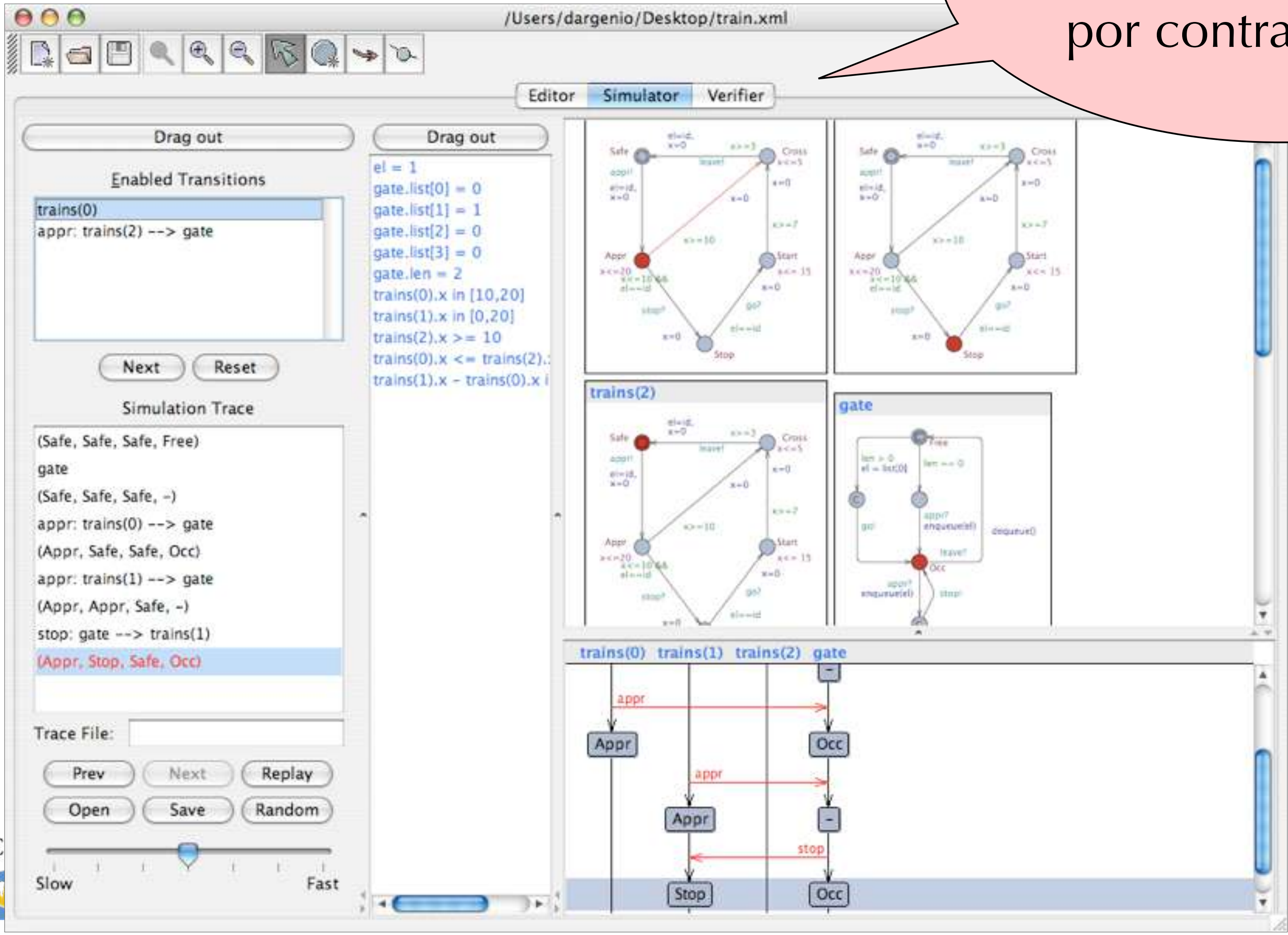
Stop



El model checker Uppaal

La simulación es un chiche imperdible:

Permite simulación aleatoria, asistida o guiada por contraejemplos



CONIC

UNC

El model checker Uppaal

Las propiedades que puede verificar son:

- Invarianza (`A[] prop`) y alcanzabilidad (`E<> prop`)
- Alcanzabilidad inevitable (`A<> prop`) y posiblemente siempre (`E[] prop`)
- Respuesta (`prop1 --> prop2`)

donde `prop` son formulas proposicionales que pueden utilizar relojes.

Las técnicas y algoritmos de optimización son específicos para autómatas temporizados (DBMs, CDDs, uso de zonas, algoritmos de recorridos,...)

El model checker Uppaal

Usos

- Los casos de usos industriales se han aplicado en la academia a través de servicios o con proyectos subsidiados por empresas.
- Estos se han realizado, además, por otras universidades distintas de las involucradas en el desarrollo.
- Ejemplos:
  - Protocolos para audio (Philips).
  - Protocolos de multimedia
  - Controladores de caja de cambios (Mecel AB)
  - Protocolos para bus de campo (ABB)
  - Protocolos para audio y video (Bang & Olufsen)
- Uppaal dio lugar a un enorme estudio posterior con otras herramientas relacionadas: Times, Uppaal Cora, Uppaal Tron, etc.

PRISM Model Checker

- Desarrollado originalmente en la Universidad de Birmingham, luego también las de Oxford y Glasgow
- Fundamentalmente desarrollado por David Parker, Gethin Norman, y Marta Kwiatkowska.
- www.prismmodelchecker.org
- PRISM es un model checker centrado en análisis cuantitativo dirigido a múltiples tipos de modelos
- Los modelos también se describen como redes de autómatas (según el tipo de modelo)
- Se puede utilizar desde línea de comando o desde un entorno gráfico

PRISM Model Checker

- Desarrollado originalmente en la Universidad de Birmingham, luego también las de Oxford y Glasgow
- Fundamentalmente desarrollado por David Parker, Gethin Norman, y Marta Kwiatkowska.
- www.prismmodelchecker.org
- PRISM es un model checker centrado en análisis cuantitativo dirigido a múltiples tipos de modelos
- Los modelos también se describen como redes de autómatas (según el tipo de modelo)
- Se puede utilizar desde línea de comando o desde un entorno gráfico

DTMC, CTMC, MDP, PTA

CONICET

UNC

```
mdp

const int N=8;
const int K;
const int range = 2*(K+1)*N;
const int counter_init = (K+1)*N;
const int left = N;
const int right = 2*(K+1)*N - N;

global counter : [0..range] init counter_init;

module process1
    pc1 : [0..3];
    coin1 : [0..1];
    [] (pc1=0)  -> 0.5 : (coin1'=0) & (pc1'=1) + 0.5 : (coin1'=1) & (pc1'=1);
    [] (pc1=1) & (coin1=0) & (counter>0) -> (counter'=counter-1) & (pc1'=2) & (coin1'=0);
    [] (pc1=1) & (coin1=1) & (counter<range) -> (counter'=counter+1) & (pc1'=2) & (coin1'=0);
    [] (pc1=2) & (counter<=left) -> (pc1'=3) & (coin1'=0);
    [] (pc1=2) & (counter>=right) -> (pc1'=3) & (coin1'=1);
    [] (pc1=2) & (counter>left) & (counter<right) -> (pc1'=0);
    [done] (pc1=3) -> (pc1'=3);
endmodule

module process2 = process1[pc1=pc2,coin1=coin2] endmodule
module process3 = process1[pc1=pc3,coin1=coin3] endmodule
module process4 = process1[pc1=pc4,coin1=coin4] endmodule
module process5 = process1[pc1=pc5,coin1=coin5] endmodule
module process6 = process1[pc1=pc6,coin1=coin6] endmodule
module process7 = process1[pc1=pc7,coin1=coin7] endmodule
module process8 = process1[pc1=pc8,coin1=coin8] endmodule

label "finished" = pc1=3 & pc2=3 & pc3=3 & pc4=3 & pc5=3 & pc6=3 & pc7=3 & pc8=3 ;
label "all_coins_equal_0" = coin1=0 & coin2=0 & coin3=0 & coin4=0 & coin5=0 & coin6=0 & coin7=0 & coin8=0 ;
label "all_coins_equal_1" = coin1=1 & coin2=1 & coin3=1 & coin4=1 & coin5=1 & coin6=1 & coin7=1 & coin8=1 ;
label "agree" = coin1=coin2 & coin2=coin3 & coin3=coin4 & coin4=coin5 & coin5=coin6 & coin6=coin7 &
coin7=coin8 ;

rewards "steps"
    true : 1;
endrewards
```

CONICET

UNC

```
mdp

const int N=8;
const int K;
const int range = 2*(K+1)*N;
const int counter_init = (K+1)*N;
const int left = N;
const int right = 2*(K+1)*N - N;

global counter : [0..range] init counter_init;

module process1
    pc1 : [0..3];
    coin1 : [0..1];
    [] (pc1=0)  -> 0.5 : (coin1'=0) & (pc1'=1) + 0.5 : (coin1'=1) & (pc1'=1);
    [] (pc1=1) & (coin1=0) & (counter>0) -> (counter'=counter-1) & (pc1'=2) & (coin1'=0);
    [] (pc1=1) & (coin1=1) & (counter<range) -> (counter'=counter+1) & (pc1'=2) & (coin1'=0);
    [] (pc1=2) & (counter<=left) -> (pc1'=3) & (coin1'=0);
    [] (pc1=2) & (counter>=right) -> (pc1'=3) & (coin1'=1);
    [] (pc1=2) & (counter>left) & (counter<right) -> (pc1'=0);
    [done] (pc1=3) -> (pc1'=3);
endmodule

module process2 = process1[pc1=pc2,coin1=coin2] endmodule
module process3 = process1[pc1=pc3,coin1=coin3] endmodule
module process4 = process1[pc1=pc4,coin1=coin4] endmodule
module process5 = process1[pc1=pc5,coin1=coin5] endmodule
module process6 = process1[pc1=pc6,coin1=coin6] endmodule
module process7 = process1[pc1=pc7,coin1=coin7] endmodule
module process8 = process1[pc1=pc8,coin1=coin8] endmodule

label "finished" = pc1=3 & pc2=3 & pc3=3 & pc4=3 & pc5=3 & pc6=3 & pc7=3 & pc8=3 ;
label "all_coins_equal_0" = coin1=0 & coin2=0 & coin3=0 & coin4=0 & coin5=0 & coin6=0 & coin7=0 & coin8=0 ;
label "all_coins_equal_1" = coin1=1 & coin2=1 & coin3=1 & coin4=1 & coin5=1 & coin6=1 & coin7=1 & coin8=1 ;
label "agree" = coin1=coin2 & coin2=coin3 & coin3=coin4 & coin4=coin5 & coin5=coin6 & coin6=coin7 &
coin7=coin8 ;

rewards "steps"
    true : 1;
endrewards
```

Lenguaje textual, pero simple

CONICET

UNC

```
mdp

const int N=8;
const int K;
const int range = 2*(K+1)*N;
const int counter_init = (K+1)*N;
const int left = N;
const int right = 2*(K+1)*N - N;

global counter : [0..range] init counter_init;

module process1
    pc1 : [0..3];
    coin1 : [0..1];
    [] (pc1=0)  -> 0.5 : (coin1'=0) & (pc1'=1) + 0.5 : (coin1'=1) & (pc1'=1);
    [] (pc1=1) & (coin1=0) & (counter>0) -> (counter'=counter-1) & (pc1'=2) & (coin1'=0);
    [] (pc1=1) & (coin1=1) & (counter<range) -> (counter'=counter+1) & (pc1'=2) & (coin1'=0);
    [] (pc1=2) & (counter<=left) -> (pc1'=3) & (coin1'=0);
    [] (pc1=2) & (counter>=right) -> (pc1'=3) & (coin1'=1);
    [] (pc1=2) & (counter>left) & (counter<right) -> (pc1'=0);
    [done] (pc1=3) -> (pc1'=3);
endmodule

module process2 = process1[pc1=pc2,coin1=coin2] endmodule
module process3 = process1[pc1=pc3,coin1=coin3] endmodule
module process4 = process1[pc1=pc4,coin1=coin4] endmodule
module process5 = process1[pc1=pc5,coin1=coin5] endmodule
module process6 = process1[pc1=pc6,coin1=coin6] endmodule
module process7 = process1[pc1=pc7,coin1=coin7] endmodule
module process8 = process1[pc1=pc8,coin1=coin8] endmodule

label "finished" = pc1=3 & pc2=3 & pc3=3 & pc4=3 & pc5=3 & pc6=3 & pc7=3 & pc8=3 ;
label "all_coins_equal_0" = coin1=0 & coin2=0 & coin3=0 & coin4=0 & coin5=0 & coin6=0 & coin7=0 & coin8=0 ;
label "all_coins_equal_1" = coin1=1 & coin2=1 & coin3=1 & coin4=1 & coin5=1 & coin6=1 & coin7=1 & coin8=1 ;
label "agree" = coin1=coin2 & coin2=coin3 & coin3=coin4 & coin4=coin5 & coin5=coin6 & coin6=coin7 &
coin7=coin8 ;

rewards "steps"
    true : 1;
endrewards
```

Lenguaje textual, pero simple

Permite modelar recompensas o costos

CONICET

UNC

PRISM Model Checker

Propiedades

LTL, PCTL, CSL, PCTL* y extensiones para manipular recompensas y costos

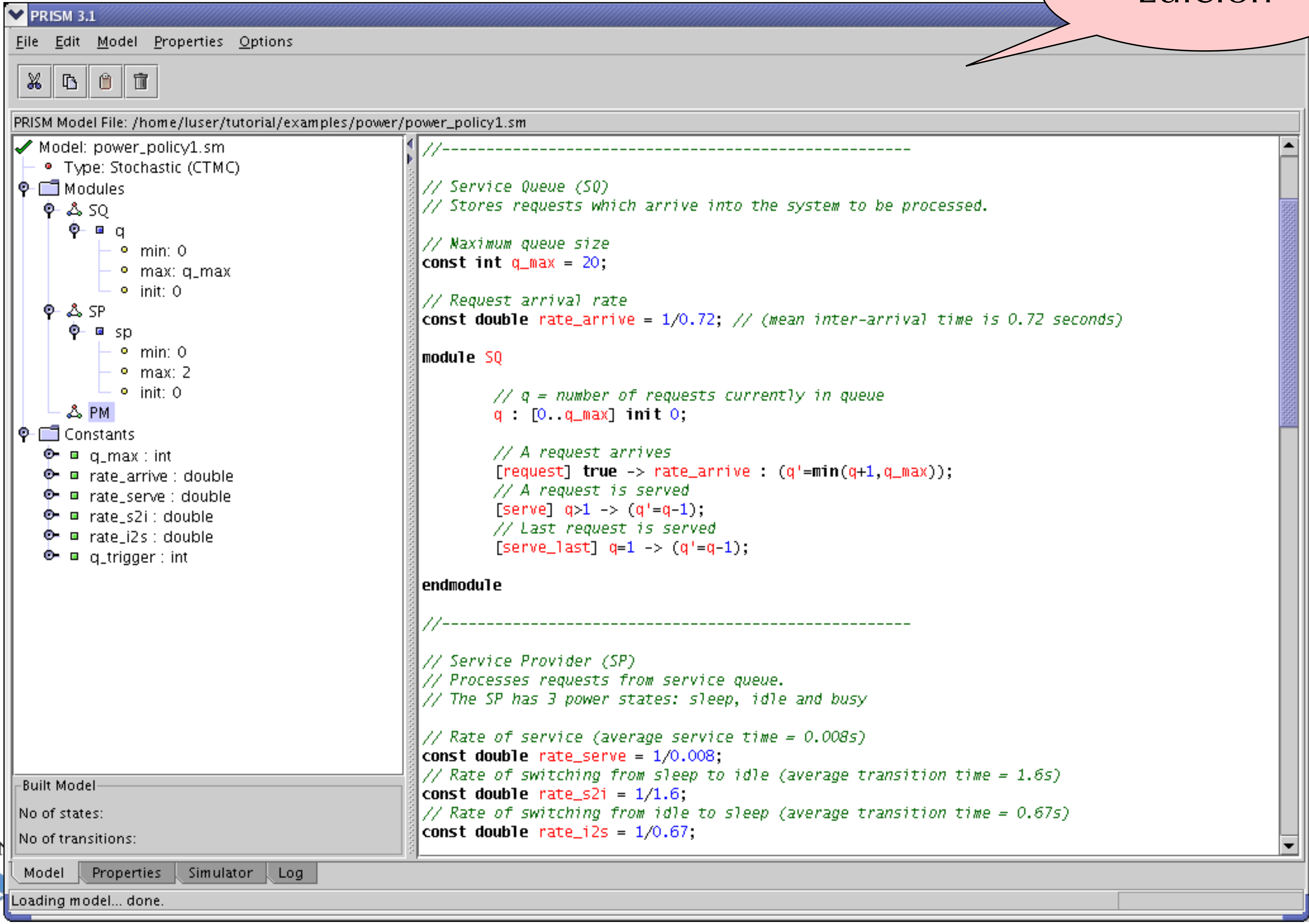
```
P>=1 [ F "terminate" ]

P<0.1 [ F<=100 (num_errors > 5) ]

S<0.01 [ num_sensors < min_sensors ]

P=? [ !exit U error ]

Pmax=? [ F<=T (messages_lost > 10) ]

S=? [ queue_size / max_size > 0.75 ]

R{"steps"}min=? [ F "finished" ]
```

CONICET

UNC

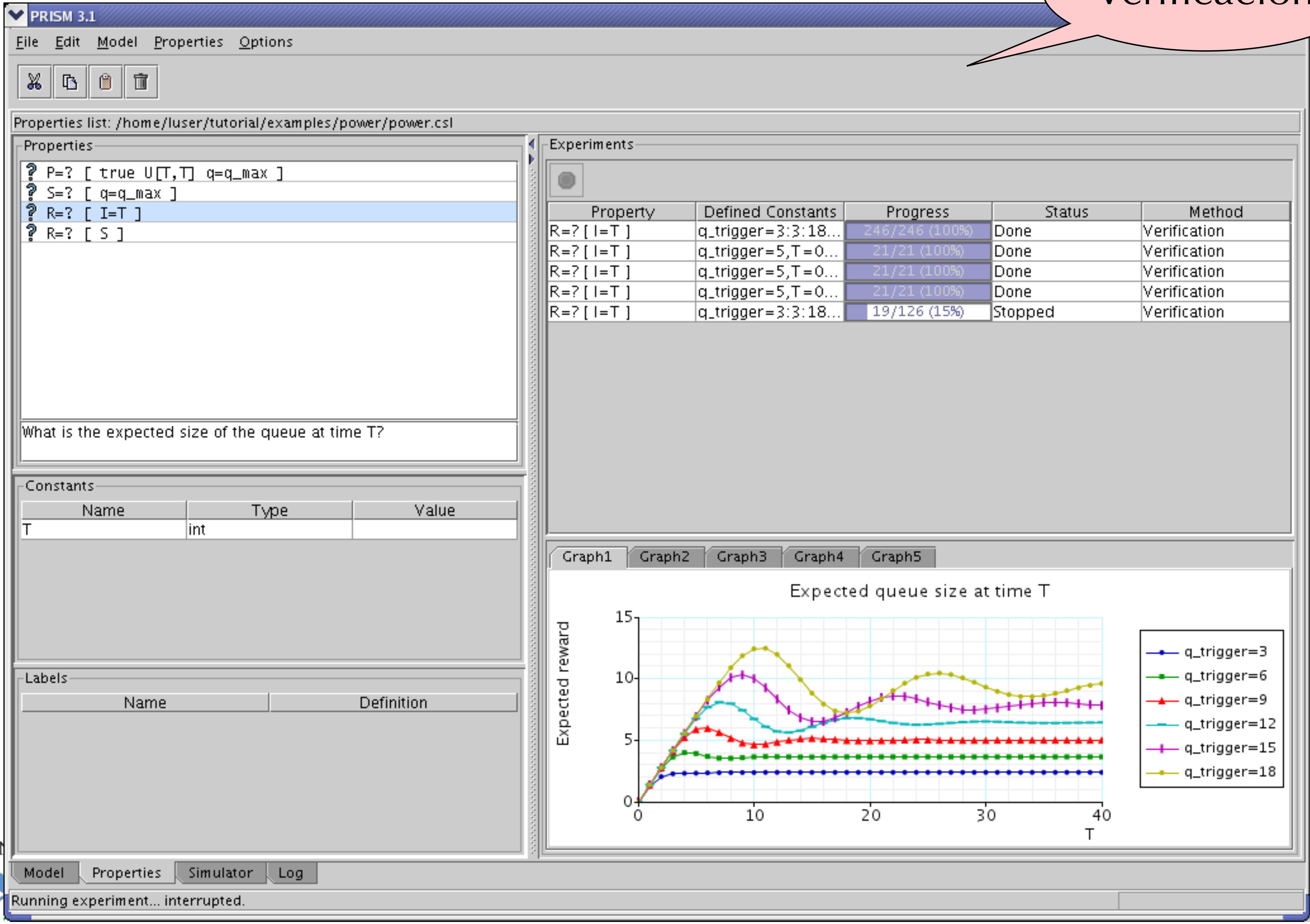
PRISM Model Checker

Edición



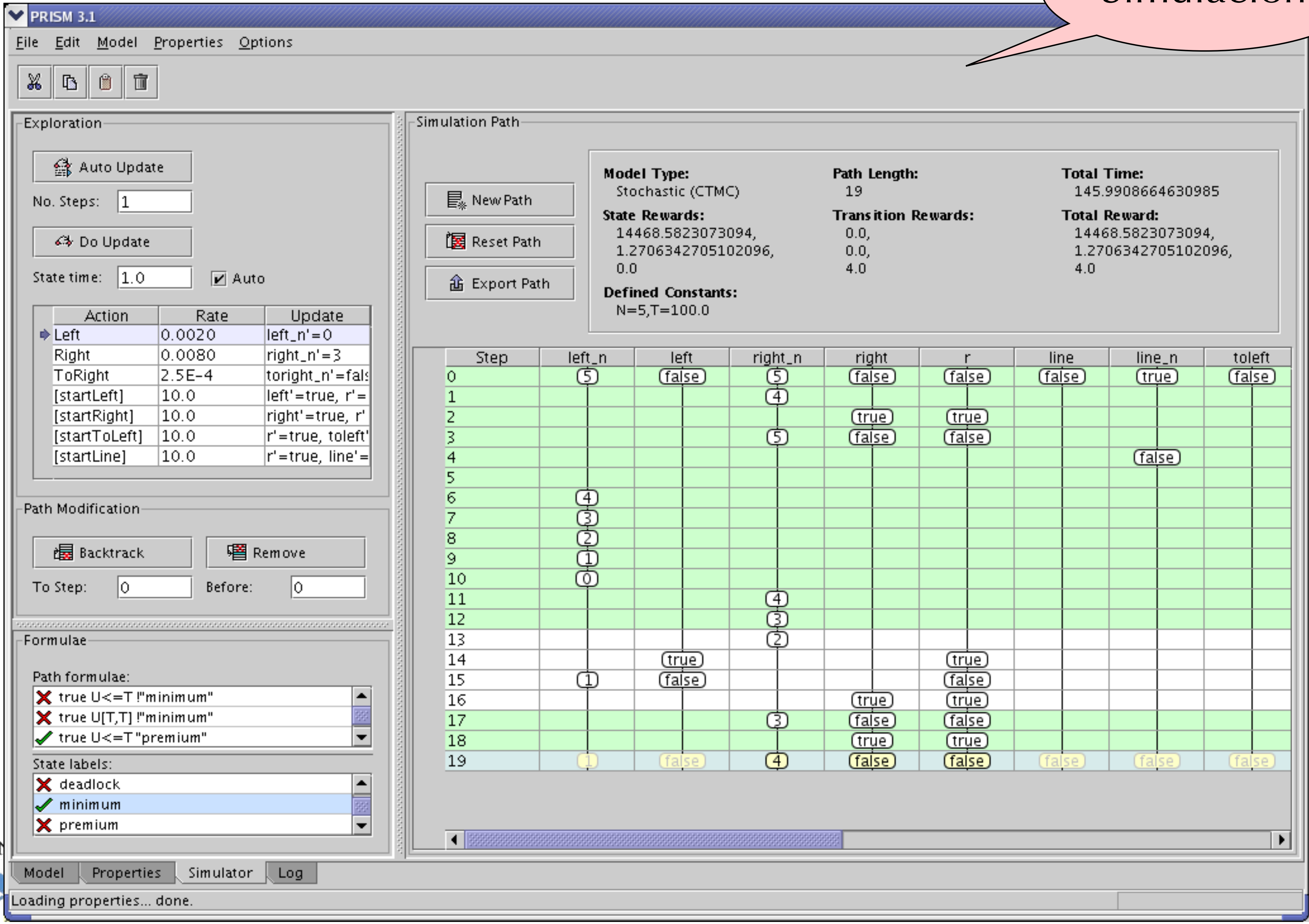
PRISM Model Checker

Verificación



PRISM Model Checker

Simulación



PRISM Model Checker

Algoritmos

- Análisis de grafo
- Análisis numérico
- Model checking estadístico (simulación por evento discreto)
  - sólo para modelos sin no-determinismo (DTMC y CTMC)
- Estructuras de datos:
  - Explícito
  - Matrices ralas
  - MTBDD
  - Híbrido

PRISM Model Checker

Uso

- No tengo información formal sobre casos industriales
  - pero sé que se han hecho (ej: ESA)
- Casos de estudios documentados:
  - www.prismmodelchecker.org/casestudies/
  - Numerosos: incluyen toy examples y casos reales
  - Ej:
    - Root contention en IEEE 1394 “Firewire”
    - Backoff en IEEE 802.3 CSMA/CD
    - etc.

Otros models checkers

- Software model checkers:
  - SLAM (C) - Terminator (C) - Space Invaders (C) - Blast (C) - CBMC (C y C++) - Java PathFinder - Bandera/Bogor (Java) - MoonWalker (.Net)
- Más:
  - CADP - mCRL2 - …
  - Storm - MRMC - E⊢MC² - Rapture - Liquor ...
  - HyTech - Kronos - Zeus - …
- en.wikipedia.org/wiki/List_of_model_checking_tools (está desorganizada pero hay un montón de links)

CONICET

UNC

Otros models checkers

Verificación automática de device drivers de Windows

- Software model checkers:
  - SLAM (C) - Terminator (C) - Space Invaders (C) - Blast (C) - CBMC (C y C++) - Java PathFinder - Bandera/Bogor (Java) - MoonWalker (.Net)
- Más:
  - CADP - mCRL2 - …
  - Storm - MRMC - E⊢MC^2 - Rapture - Liquor …
  - HyTech - Kronos - Zeus - …
- en.wikipedia.org/wiki/List_of_model_checking_tools (está desorganizada pero hay un montón de links)

Otros models ch

Verificación device drivers

Lograron verificar completamente de manera automática más de la mitad del kernel de Linux

- Software model checkers:
  - SLAM (C) - Terminator (C) - Space Invaders (C) - Blast (C) - CBMC (C y C++) - Java PathFinder - Bandera/Bogor (Java) - MoonWalker (.Net)
- Más:
  - CADP - mCRL2 - …
  - Storm - MRMC - $\mathrm{E}\vdash\mathrm{MC}^2$ - Rapture - Liquor …
  - HyTech - Kronos - Zeus - …
- en.wikipedia.org/wiki/List_of_model_checking_tools (está desorganizada pero hay un montón de links)

Ejercicio 1: Considere el siguiente programa concurrente. Siguiendo la semántica, defina dos fragmentos de ejecución distintos que lleven a la violación de la exclusión mutua (es decir `in_critical=2`) desde el estado inicial $\langle P_0^1 \parallel P_0^2, (0,0,0)\rangle$. Suponga que la memoria esta representada como la terna $(\mu(\mathtt{y1}), \mu(\mathtt{y2}), \mu(\mathtt{in\_critical}))$ y que cada variable solo puede tomar los valores 0, 1, y 2.

```
P0¹ [      while true do
              y1 := y2 + 1 ;
P1¹  [        if
                ⫿ ((y2 = 0) ∨ (y1 ≤ y2)) →
                      in_critical ++ ;
                      // región crítica
P2¹    [              in_critical -- ;
P3¹      [            y1 := 0
              fi
           od
```

$\|$

```
P0² [      while true do
              y2 := y1 + 1 ;
P1²  [        if
                ⫿ ((y1 = 0) ∨ (y2 < y1)) →
                      in_critical ++ ;
                      // región crítica
P2²    [              in_critical -- ;
P3²      [            y2 := 0
              fi
           od
```

Ejercicio 2: Demuestre usando la semántica de LTL que

$$\mathtt{G}(llueve \Rightarrow \mathtt{F}\,\neg llueve) \;=\; \mathtt{G}\,\mathtt{F}\,\neg llueve.$$

Ejercicio 3: Sabemos que $\phi\,\mathtt{R}\,\psi \;=\; \neg(\neg\phi\,\mathtt{U}\,\neg\psi)$. Demuestre usando las leyes que

$$\phi\,\mathtt{R}\,\psi \equiv \psi \wedge (\phi \vee \mathtt{X}(\phi\,\mathtt{R}\,\psi)).$$

* Ejercicio 4: Demuestre que Strong Fairness implica Weak Fairness.

** Ejercicio estrella: Suma pero no resta*

CONICET

UNC

Ejercicio 5: Dé el autómata de Büchi que acepta exactamente todas las palabras que satisfacen $\mathtt{G}\,\mathtt{F}\,\neg llueve$.

Ejercicio 6: Una fórmula LTL ϕ es satisfactible si existe $\sigma \in (\mathcal{P}(\mathsf{PA}))^{\omega}$ tal que $\sigma \models \phi$. Conisderando las herramientas dadas en el curso, dé un algoritmo para determinar si una fórmula LTL ϕ es satisfactible.

CONICET

UNC