

RPCTL : Una Lógica para Tolerancia a Fallas

Pablo Castro, Cecilia Kilmurray y Nir Piterman

{pcastro, ckilmurray} AT dc.exa.unrc.edu.ar
{nir.piterman} AT leicester.ac.uk



Jornadas de Doctorandos de Computación - 2013

Cecilia Kilmurray

- Licenciada en Ciencias de la Computación .
- Docente del Departamento de computación de la UNRC desde Abril de 2010.
- Estudiante de Doctorado desde Junio de 2010. Bajo la dirección de *Pablo Castro* en la UNRC.
- Trabajo en el área de Sistemas Tolerantes a Fallas, principalmente en la especificación y verificación de este tipo de sistemas (lógicas modales, model checking, elems. de probabilidades).

Qué es un Sistema Tolerante a Fallos?

Es la capacidad de un sistema de proveer cierto comportamiento esperado aún ante la ocurrencia ocasional de fallas.



Qué propiedades nos Interesan

Sistemas que ante la ocurrencia de fallas:

- Permanecen en una región “normal” del sistema.

Qué propiedades nos Interesan

Sistemas que ante la ocurrencia de fallas:

- Permanecen en una región “normal” del sistema.
- Masking: (**Safety + Liveness**)

Qué propiedades nos Interesan

Sistemas que ante la ocurrencia de fallas:

- Permanecen en una región “normal” del sistema.
- Masking: (**Safety + Liveness**)
- Revisitan una región “normal” del sistema con infinita frecuencia.

Qué propiedades nos Interesan

Sistemas que ante la ocurrencia de fallas:

- Permanecen en una región “normal” del sistema.
- Revisitan una región “normal” del sistema con infinita frecuencia.
- Masking: (Safety + Liveness)
- No-Masking: (Liveness)

Qué propiedades nos Interesan

Sistemas que ante la ocurrencia de fallas:

- Permanecen en una región “normal” del sistema.
- Revisitan una región “normal” del sistema con infinita frecuencia.
- Permanecen en una región “segura” del sistema.
- Masking: (Safety + Liveness)
- No-Masking: (Liveness)

Qué propiedades nos Interesan

Sistemas que ante la ocurrencia de fallas:

- Permanecen en una región “normal” del sistema.
- Revisitan una región “normal” del sistema con infinita frecuencia.
- Permanecen en una región “segura” del sistema.
- Masking: (**Safety + Liveness**)
- No-Masking: (**Liveness**)
- Failsafe: (**Safety**)

Una Lógica Temporal con probabilidades

PCTL

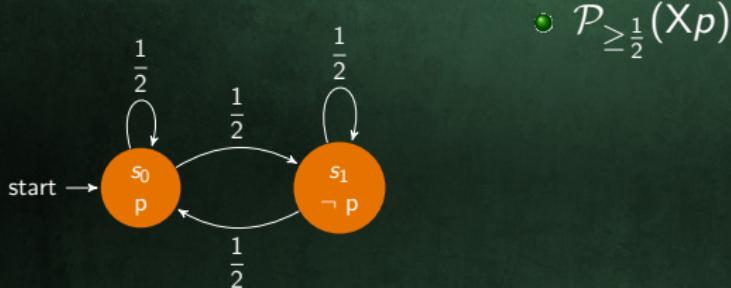
- $(\text{CTL}_{-\{\exists, \forall\}}) + \mathcal{P}_J(\Psi)$ (Ψ es una path fórmula).
- Permite medir de manera cuantitativa propiedades sobre trazas.

Una Lógica Temporal con probabilidades

PCTL

- $(\text{CTL}_{-\{\exists, \forall\}}) + \mathcal{P}_J(\Psi)$ (Ψ es una path fórmula).
- Permite medir de manera cuantitativa propiedades sobre trazas.

Las fórmulas en PCTL son interpretadas sobre *Cadenas de Markov* discretas.

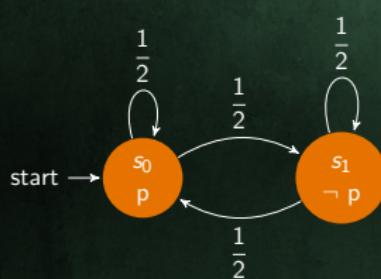


Una Lógica Temporal con probabilidades

PCTL

- $(\text{CTL}_{-\{\text{E}, \text{A}\}}) + \mathcal{P}_J(\psi)$ (ψ es una path fórmula).
- Permite medir de manera cuantitativa propiedades sobre trazas.

Las fórmulas en PCTL son interpretadas sobre *Cadenas de Markov* discretas.



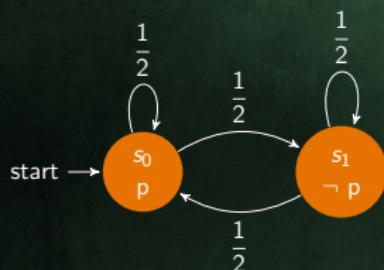
- $\mathcal{P}_{\geq \frac{1}{2}}(\mathsf{X}p)$
- $p \rightarrow \mathcal{P}_{\geq 1}(\mathsf{F}(\neg p))$

Una Lógica Temporal con probabilidades

PCTL

- $(\text{CTL}_{-\{\exists, \forall\}}) + \mathcal{P}_J(\Psi)$ (Ψ es una path fórmula).
- Permite medir de manera cuantitativa propiedades sobre trazas.

Las fórmulas en PCTL son interpretadas sobre *Cadenas de Markov* discretas.



- $\mathcal{P}_{\geq \frac{1}{2}}(\mathsf{X}p)$
 - $p \rightarrow \mathcal{P}_{\geq 1}(\mathsf{F}(\neg p))$
- | | | |
|-------------------------------------|---------|-----|
| $\mathcal{P}(s_0, s_1)$ | = 0,5 | + |
| $\mathcal{P}(s_0, s_0, s_1)$ | = 0,25 | + |
| $\mathcal{P}(s_0, s_0, s_0, s_1)$ | = 0,125 | + |
| $\mathcal{P}(s_0, s_0, \dots, s_1)$ | = ... | = 1 |

Una nueva clase de Autómatas probabilísticos:

p-Autómatas

Un p-Autómata A es una 5-upla, de la forma $\langle \Sigma, Q, \delta, \varphi^{in}, \alpha \rangle$,

- El lenguaje aceptado por A son todas aquellas cadenas de Markov, y sus bisimilares, donde se cumple la condición de aceptación $\alpha \subseteq Q$.
- $\delta : Q \times \Sigma \rightarrow B^+(Q \cup \llbracket Q \rrbracket)$, es la función de transición.

Una nueva clase de Autómatas probabilísticos:

p-Autómatas

Un p-Autómata A es una 5-upla, de la forma $\langle \Sigma, Q, \delta, \varphi^{in}, \alpha \rangle$,

- El lenguaje aceptado por A son todas aquellas cadenas de Markov, y sus **bisimilares**, donde se cumple la condición de aceptación $\alpha \subseteq Q$.
- $\delta : Q \times \Sigma \rightarrow B^+(Q \cup \llbracket Q \rrbracket)$, es la función de transición.

Una nueva clase de Autómatas probabilísticos:

p-Autómatas

Un p-Autómata A es una 5-upla, de la forma $\langle \Sigma, Q, \delta, \varphi^{in}, \alpha \rangle$,

- El lenguaje aceptado por A son todas aquellas cadenas de Markov, y sus **bisimilares**, donde se cumple la condición de aceptación $\alpha \subseteq Q$.
- $\delta : Q \times \Sigma \rightarrow B^+(Q \cup \llbracket Q \rrbracket)$, es la función de transición.

La intuición atrás de la función de transición:

- $\delta(q_1, \{a\}) = q_1 \wedge \llbracket q_1 \rrbracket_{\geq \frac{1}{2}}$

Una nueva clase de Autómatas probabilísticos:

p-Autómatas

Un p-Autómata A es una 5-upla, de la forma $\langle \Sigma, Q, \delta, \varphi^{in}, \alpha \rangle$,

- El lenguaje aceptado por A son todas aquellas cadenas de Markov, y sus **bisimilares**, donde se cumple la condición de aceptación $\alpha \subseteq Q$.
- $\delta : Q \times \Sigma \rightarrow B^+(Q \cup \llbracket Q \rrbracket)$, es la función de transición.

La intuición atrás de la función de transición:

- $\delta(q_1, \{a\}) = q_1 \wedge \llbracket q_1 \rrbracket_{\geq \frac{1}{2}}$
- “a” se cumple en q_1 y más de $\frac{1}{2}$ de los sucesores de los estados leídos por q_1 van a cumplir con la propiedad de que valga “a”.

Una nueva clase de Autómatas probabilísticos:

p-Autómatas

Un p-Autómata A es una 5-upla, de la forma $\langle \Sigma, Q, \delta, \varphi^{in}, \alpha \rangle$,

- El lenguaje aceptado por A son todas aquellas cadenas de Markov, y sus **bisimilares**, donde se cumple la condición de aceptación $\alpha \subseteq Q$.
- $\delta : Q \times \Sigma \rightarrow B^+(Q \cup \llbracket Q \rrbracket)$, es la función de transición.

La intuición atrás de la función de transición:

- $\delta(q_1, \{a\}) = q_1 \wedge \llbracket q_1 \rrbracket_{\geq \frac{1}{2}}$
- “a” se cumple en q_1 y más de $\frac{1}{2}$ de los sucesores de los estados leídos por q_1 van a cumplir con la propiedad de que valga “a”.

Una nueva clase de Autómatas probabilísticos:

p-Autómatas

Un p-Autómata A es una 5-upla, de la forma $\langle \Sigma, Q, \delta, \varphi^{in}, \alpha \rangle$,

- El lenguaje aceptado por A son todas aquellas cadenas de Markov, y sus **bisimilares**, donde se cumple la condición de aceptación $\alpha \subseteq Q$.
- $\delta : Q \times \Sigma \rightarrow B^+(Q \cup \llbracket Q \rrbracket)$, es la función de transición.

La intuición atrás de la función de transición:

- $\delta(q_1, \{a\}) = q_1 \wedge \llbracket q_1 \rrbracket_{\geq \frac{1}{2}}$
- “a” se cumple en q_1 y más de $\frac{1}{2}$ de los sucesores de los estados leídos por q_1 van a cumplir con la propiedad de que valga “a”.

Permiten caracterizar subconjuntos de trazas y definir probabilidades locales sobre estos.

Nuestra propuesta

RPCTL

Es una extensión de PCTL con fórmulas recursivas.

Sintácticamente:

- Permitimos que una fórmula contenga una llamada recursiva a través de los operadores *call* y *rec*.

$$\begin{aligned} J &::= \{>, \geq\} \times [0, 1] \\ \Phi &::= \dots \mid call_i \mid \mathcal{P}_J(\Psi) \mid rec(\Psi) \end{aligned}$$

Nuestra propuesta

RPCTL

Es una extensión de PCTL con fórmulas recursivas.

Sintácticamente:

- Permitimos que una fórmula contenga una llamada recursiva a través de los operadores *call* y *rec*.

$$\begin{aligned} J &::= \{>, \geq\} \times [0, 1] \\ \Phi &::= \dots \mid \textcolor{blue}{call}_i \mid \mathcal{P}_J(\Psi) \mid \textcolor{blue}{rec}(\Psi) \end{aligned}$$

Nuestra propuesta

RPCTL

Es una extensión de PCTL con fórmulas recursivas.

Sintácticamente:

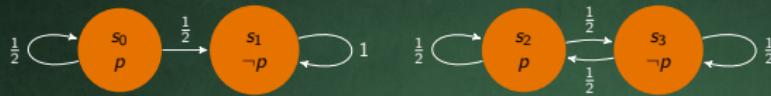
- Permitimos que una fórmula contenga una llamada recursiva a través de los operadores *call* y *rec*.

$$\begin{aligned} J &::= \{>, \geq\} \times [0, 1] \\ \Phi &::= \dots \mid \textcolor{blue}{call}_i \mid \mathcal{P}_J(\Psi) \mid \textcolor{blue}{rec}(\Psi) \end{aligned}$$

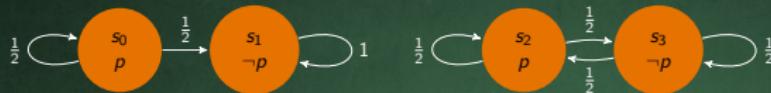
Semánticamente:

- Cada fórmula φ es traducida a un p-autómata A_φ que acepta todas las *Cadenas de Markov* que satisfacen φ .

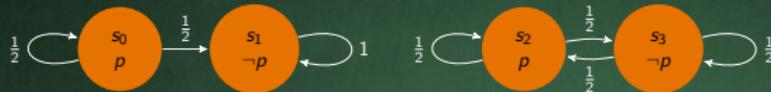
$$M, s \models \varphi \iff A_\varphi \text{ accepts } M_s$$



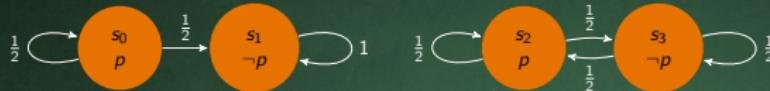
- $M, s \models \text{rec}(p \wedge \mathcal{P}_{>0}(\text{X call}_1))$



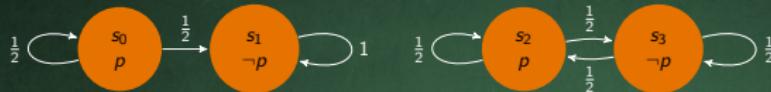
- $M, s \models \text{rec}(p \wedge \mathcal{P}_{>0}(\text{X call}_1))$
- “p” vale en s y s tiene un sucesor donde vale “p”, el cual a su vez tiene un sucesor donde también vale ... y así sucesivamente.



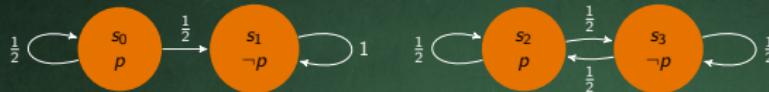
- $M, s \models \text{rec}(p \wedge \mathcal{P}_{>0}(\text{X call}_1))$
- “p” vale en s y s tiene un sucesor donde vale “p”, el cual a su vez tiene un sucesor donde también vale ... y así sucesivamente.
- Vale para $s = s_0$ y $s = s_2$



- $M, s \models \text{rec}(p \wedge \mathcal{P}_{>0}(\text{X call}_1))$
 - “p” vale en s y s tiene un sucesor donde vale “p”, el cual a su vez tiene un sucesor donde también vale ... y así sucesivamente.
 - Vale para $s = s_0$ y $s = s_2$
- $M, s \models \text{rec}(\mathcal{P}_{\geq \frac{1}{2}}(\text{X (call}_1 \wedge p)) \wedge \mathcal{P}_{\geq \frac{1}{2}}(\text{X } \neg p))$

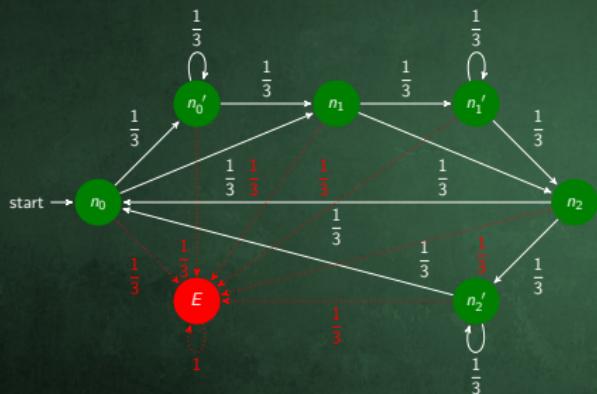


- $M, s \models \text{rec}(p \wedge \mathcal{P}_{>0}(\text{X call}_1))$
 - “p” vale en s y s tiene un sucesor donde vale “p”, el cual a su vez tiene un sucesor donde también vale ... y así sucesivamente.
 - Vale para $s = s_0$ y $s = s_2$
- $M, s \models \text{rec}(\mathcal{P}_{\geq \frac{1}{2}}(\text{X (call}_1 \wedge p)) \wedge \mathcal{P}_{\geq \frac{1}{2}}(\text{X } \neg p))$
 - Esta fórmula vale en s si s tiene $\frac{1}{2}$ de sus sucesores que satisfacen “p”, y ellos a su vez recursivamente cumplen con esta propiedad. Y el $\frac{1}{2}$ restante de los sucesores de s , satisfacen “ $\neg p$ ”.

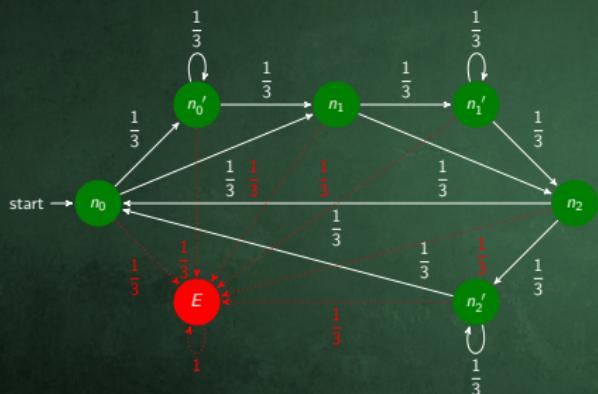


- $M, \textcolor{blue}{s} \models \text{rec}(\textcolor{blue}{p} \wedge \mathcal{P}_{>0}(\text{X call}_1))$
 - “p” vale en $\textcolor{blue}{s}$ y s tiene un **sucesor** donde vale “p”, el cual a su vez tiene un sucesor donde también vale ... y así **sucesivamente**.
 - Vale para $s = s_0$ y $s = s_2$
- $M, \textcolor{blue}{s} \models \text{rec}(\mathcal{P}_{\geq \frac{1}{2}}(\text{X (call}_1 \wedge \textcolor{blue}{p})) \wedge \mathcal{P}_{\geq \frac{1}{2}}(\text{X } \neg \textcolor{blue}{p}))$
 - Esta fórmula vale en s si $\textcolor{blue}{s}$ tiene $\frac{1}{2}$ de sus **sucesores** que satisfacen “p”, y ellos a su vez recursivamente cumplen con esta propiedad. Y el $\frac{1}{2}$ restante de los sucesores de s , satisfacen “ $\neg p$ ”.
 - En particular esta fórmula vale para s_0, s_2 y s_3 .

Token Ring



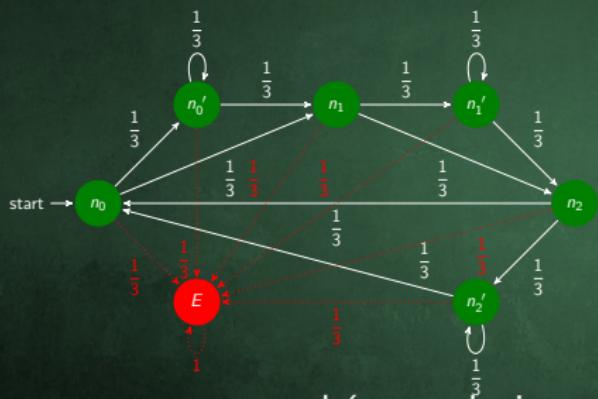
Token Ring



$$\text{rec} \left[\begin{array}{l} (n_0 \rightarrow \mathcal{P}_{\geq \frac{1}{2}}(\mathbf{F}(n_1 \wedge \text{call}_1))) \wedge \\ (n_1 \rightarrow \mathcal{P}_{\geq \frac{1}{2}}(\mathbf{F}(n_2 \wedge \text{call}_2))) \wedge \\ (n_2 \rightarrow \mathcal{P}_{\geq \frac{1}{2}}(\mathbf{F}(n_0 \wedge \text{call}_3))) \end{array} \right]$$

“Sin fallas permanece estable”.

Token Ring



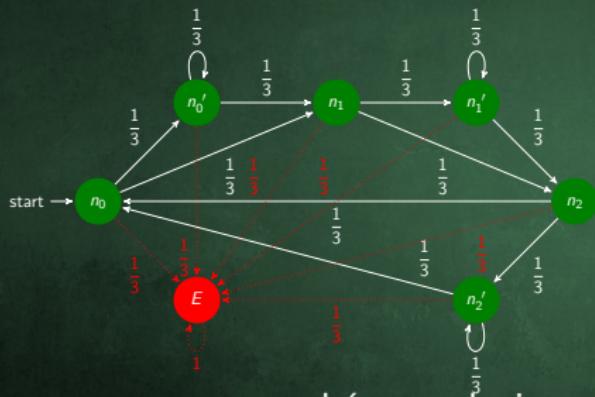
$$\text{rec} \left[\begin{array}{l} (n_0 \rightarrow \mathcal{P}_{\geq \frac{1}{2}}(\mathbf{F}(n_1 \wedge \text{call}_1))) \wedge \\ (n_1 \rightarrow \mathcal{P}_{\geq \frac{1}{2}}(\mathbf{F}(n_2 \wedge \text{call}_2))) \wedge \\ (n_2 \rightarrow \mathcal{P}_{\geq \frac{1}{2}}(\mathbf{F}(n_0 \wedge \text{call}_3))) \end{array} \right]$$

“Sin fallas permanece estable”.

y en PCTL , podríamos decir esto?

$$\varphi = \left[\begin{array}{l} (n_0 \rightarrow \mathcal{P}_{\geq \frac{1}{2}}(\mathbf{F}(n_1))) \wedge \\ (n_1 \rightarrow \mathcal{P}_{\geq \frac{1}{2}}(\mathbf{F}(n_2))) \wedge \\ (n_2 \rightarrow \mathcal{P}_{\geq \frac{1}{2}}(\mathbf{F}(n_0))) \end{array} \right]$$

Token Ring

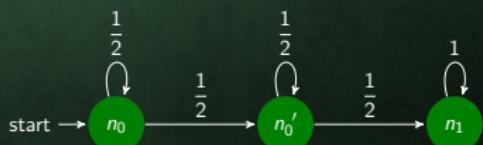


$$\text{rec} \left[\begin{array}{l} (n_0 \rightarrow \mathcal{P}_{\geq \frac{1}{2}}(\mathbf{F}(n_1 \wedge \text{call}_1))) \wedge \\ (n_1 \rightarrow \mathcal{P}_{\geq \frac{1}{2}}(\mathbf{F}(n_2 \wedge \text{call}_2))) \wedge \\ (n_2 \rightarrow \mathcal{P}_{\geq \frac{1}{2}}(\mathbf{F}(n_0 \wedge \text{call}_3))) \end{array} \right]$$

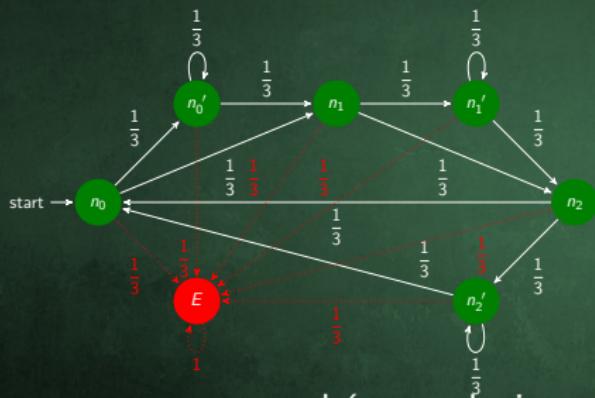
“Sin fallas permanece estable”.

y en PCTL , podríamos decir esto?

$$\varphi = \left[\begin{array}{l} (n_0 \rightarrow \mathcal{P}_{\geq \frac{1}{2}}(\mathbf{F}(n_1))) \wedge \\ (n_1 \rightarrow \mathcal{P}_{\geq \frac{1}{2}}(\mathbf{F}(n_2))) \wedge \\ (n_2 \rightarrow \mathcal{P}_{\geq \frac{1}{2}}(\mathbf{F}(n_0))) \end{array} \right]$$



Token Ring

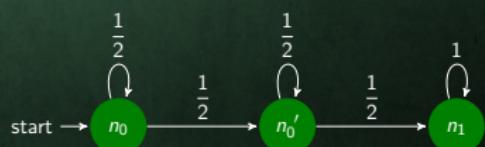


$$\text{rec} \left[\begin{array}{l} (n_0 \rightarrow \mathcal{P}_{\geq \frac{1}{2}}(\mathbf{F}(n_1 \wedge \text{call}_1))) \wedge \\ (n_1 \rightarrow \mathcal{P}_{\geq \frac{1}{2}}(\mathbf{F}(n_2 \wedge \text{call}_2))) \wedge \\ (n_2 \rightarrow \mathcal{P}_{\geq \frac{1}{2}}(\mathbf{F}(n_0 \wedge \text{call}_3))) \end{array} \right]$$

“Sin fallas permanece estable”.

y en PCTL , podríamos decir esto?

$$\varphi = \left[\begin{array}{l} (n_0 \rightarrow \mathcal{P}_{\geq \frac{1}{2}}(\mathbf{F}(n_1))) \wedge \\ (n_1 \rightarrow \mathcal{P}_{\geq \frac{1}{2}}(\mathbf{F}(n_2))) \wedge \\ (n_2 \rightarrow \mathcal{P}_{\geq \frac{1}{2}}(\mathbf{F}(n_0))) \end{array} \right]$$



$$\mathcal{P}_{>0}(\mathbf{G}\varphi) = 0!$$

Algunas características interesantes de la misma:

- Es adecuada para describir y analizar muchas de las propiedades de sistemas tolerantes a fallas:
 - Permite capturar patrones de repetición probabilísticamente.
- Demostramos que RPCTL es más expresiva que CTL y PCTL .
- La complejidad en tiempo del model checking de esta lógica es polinomial.

Algunos trabajos futuros...

- a corto plazo:
 - Implementar el algoritmo de model checking que tenemos para RPCTL .

Algunos trabajos futuros...

- a corto plazo:
 - Implementar el algoritmo de model checking que tenemos para RPCTL .
- un poco mas allá :
 - Investigar la relación que hay entre μ -calculo probabilista y p-autómatas.
 - Investigar *Specification Logic*, para la especificación de circuitos en el área de robótica.