Lenguajes formales vs no tan formales
ó
James Bond vs The dude

¿Cuál es el mejor look para especificar requerimientos?

Charla de Borrachos – 30 de marzo de 2012

Lic. Fernando Asteasuain

Santo Grial

- Santo Grial En Software:

  Ausencia de errores y bugs

Testing

- Programadores invierten mucho tiempo en testear y debuguear software.
- Plétora de aplicaciones, herramientas, etc, enfocadas en el proceso de Testing.
- **No garantiza la ausencia de errores**.
- **Pueden todavía ocurrir fallas graves en los sistemas**.
- Se buscaron alternativas en métodos formales y verificación.

Verificación formal

- Dado un sistema ***S***, y su especificación ***s***, determinar si el comportamiento de ***S*** cumple lo pedido en ***s***.

Queremos ver si S se comporta como lo esperado

Los 80's

- Verificación formal: Pruebas formales manuales usando axiomas y reglas de inferencia

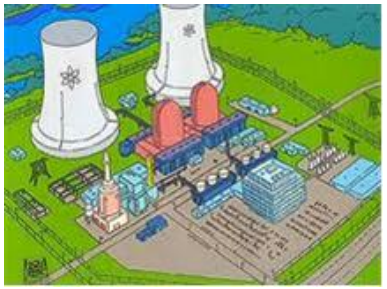
Model Checking

- *Proceso **automático** para comprobar el comportamiento de un modelo del sistema respecto de ciertas propiedades.*

- *Cada propiedad se enfoca en un aspecto en particular del comportamiento*

- *Tools: Bandera, Blast, LTSA, SPIN, UPPAAL,VinTime,Zeus, Kronos.*

Transferencia a la industria



- ¿Cuál es la mayor dificultad para que estas técnicas se adopten en la industria?

- **Escribir las Propiedades!**

Verificación

Herramienta Automática

¿Cómo escribir las propiedades en un lenguaje formal?

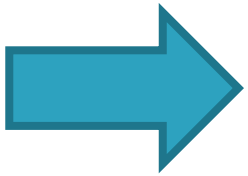
(Escrita en un lenguaje formal)

Escribiendo propiedades

- Del comportamiento expresado en lenguaje natural a la propiedad escrita en un lenguaje formal hay un largo trecho.
- Haciendo una analogía con la vida doctoral…

Lenguaje Natural

Lenguaje Formal



Lenguaje Natural

- ¿Es posible utilizarlo?

- Complica los procesamientos automáticos
- Especificaciones ambiguas, redundantes, etc.

- El lenguaje natural puede ser confuso:

Ya no soporto al gorila de tu tío

Escribir en lenguajes formales

- Se trata de algo más complejo que sólo escribir…

1. **Escribir la propiedad en el lenguaje formal**

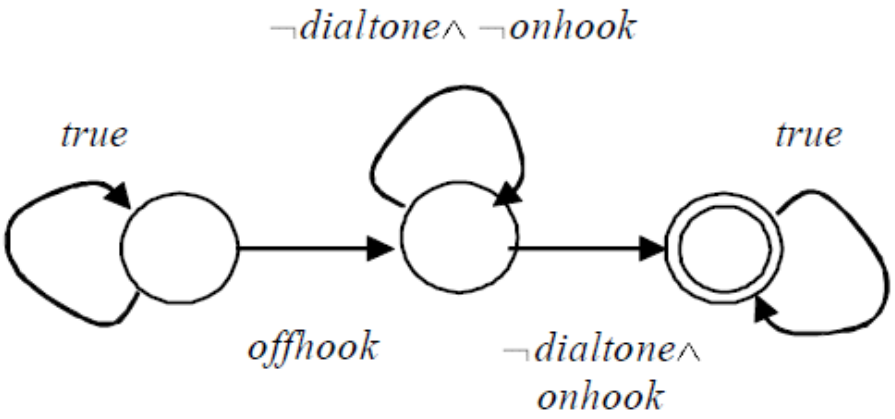
2. **Chequear que lo escrito se corresponda con lo que se desea expresar.**

Qué necesito

- Necesito **entender** la fórmula
- Especificaciones pequeñas, fáciles de comparar, modificar y manipular.
- Facilidades para razonar por comportamiento complementario

Lenguajes Formales

- Notaciones Operacionales: Autómatas



- **Notaciones Declarativas: Lógicas Temporales:**
  - **Ejemplo Fórmula LTL: □(S → ◊R)**

Ejemplos

- "Entre que se llama al ascensor en un piso, y el ascensor llega, el mismo podrá pasar como máximo dos veces por ese piso"
- *En LTL:*

$$\Box((\text{call} \wedge \Diamond\text{open}) \rightarrow ((\neg\text{atfloor} \wedge \neg\text{open}) \; \mathcal{U} \; (\text{open} \vee ((\text{atfloor} \wedge \neg\text{open}) \; \mathcal{U} \; (\text{open} \vee ((\neg\text{atfloor} \wedge \neg\text{open}) \; \mathcal{U} \; (\text{open} \vee ((\text{atfloor} \wedge \neg\text{open}) \; \mathcal{U} \; (\text{open} \vee (\neg\text{atfloor} \; \mathcal{U} \; \text{open})))))))))$$

Difícil de escribir, de leer, de entender y de manipular

Ejemplo 2

- "When the subscriber picks up the phone, dial-tone is always generated"
- En LTL

$$\Box\,(offhook \rightarrow \Diamond\ dialtone)$$

- ¿Qué pasa si el usuario cuelga y vuelve a levantar el teléfono?
- Reescribo la propiedad:

$$\Diamond\ (\ offhook \rightarrow (\ \neg\ dialtone\ \mathbf{U}\ onhook\ )).$$

Entonces…?

- ¿Cumple $\Diamond$ (*offhook* $\rightarrow$ ($\neg$ *dialtone* **U** *onhook*)).
  lo esperado?

- La implicación lógica no es un operador temporal

$$(p \rightarrow q) \equiv (\neg p \vee q)$$

- Luego, la fórmula anterior puede satisfacerse en una ejecución sin *offhook (ie, offhook es false)* o también si en una ejecución ocurre el evento *onhook*.

FINALMENTE

- Y así hasta:

$$\Diamond\ (\ offhook \wedge \mathbf{X}\ (\ (\neg\ dialtone \wedge \neg\ onhook)\ \backslash$$
$$\mathbf{U}\ (\neg\ dialtone \wedge onhook\ )).$$

Atacando el problema

- LTL requiere usuarios expertos
- El problema es escribir la propiedad…

- **Usemos patrones!**
- Existen patrones de especificación que ayudan a escribir la propiedad

Patrones de especificación

- Es un conjunto de soluciones generales
- Se describe el patrón, la intención del mismo, y viene definida su traducción a lenguajes formales.

- Ejemplo: patrón Response:

  "Cuando la ocurrencia de un evento **estímulo** provoca la ocurrencia de otro evento **acción**"
- En LTL

$$\square\,(Estímulo \rightarrow \Diamond\ Acción)$$

Patrones bajo la Lupa

- ¿En cuánto ayudan los patrones?
- ¿El usuario realmente entiende la formula resultado?
- ¿Se obtienen especificaciones pequeñas?
- ¿Es fácil comparar las fórmulas, cambiarlas?
- ¿Es fácil pensar en cómo No se cumple la propiedad?

Especificaciones pequeñas

- Patrón Response, caso más simple:

$$\Box(S \rightarrow \Diamond R)$$

- Lo mismo, pero entre P y Q :

$$((P \wedge \neg Q \wedge \Diamond Q) \rightarrow (S \rightarrow (\neg Q \; U \; (R \wedge \neg Q))) \; U \; Q)$$

- Hay un crecimiento significativo
- Se repite para otros patrones....

- Se vuelven cada vez más difíciles de manejar

Modificando Propiedades

- "Entre los eventos P y Q, cuando sucede S, luego tiene que ocurrir R1 y R2"
- La traducción a LTL es:
- $\Box((P \wedge \Diamond Q) \rightarrow (S \rightarrow (\neg Q U (R1 \wedge \neg Q \wedge X(\neg Q \ U R2))))UQ)$

- **Interrogantes:**
  - **¿Que pasa si las respuestas también pueden ocurrir después de Q?**
  - **¿Quién modifica la formula LTL de arriba?**

Negar puede ser una solución

- Dada la siguiente propiedad en LTL:
- $((P \wedge \neg Q \wedge \Diamond Q) \rightarrow (S \rightarrow (\neg Q \; U \; (R \wedge \neg Q))) \; U \; Q)$

- ***¿Cómo se puede razonar sobre el comportamiento complementario?***

- ***¿Qué situaciones llevan a no cumplir la propiedad?***

Problema a atacar

- Determinar si el comportamiento de un sistema es el esperado.
- Herramientas automáticas:
  - Model Checking
    - Ver si un modelo del sistema cumple una determinada propiedad.
- La pregunta:
  - 2 décadas con especificaciones formales y herramientas automáticas….
    - *¿Porqué no se terminan de adoptar estas técnicas?*

La pregunta, el problema

- Los usuarios deben escribir la propiedad en un lenguaje de especificación:
  - LTL
  - Formalismos basados en autómatas
- No es una tarea trivial:
  - Escribir la propiedad deseada
  - Controlar que esa propiedad describa el comportamiento buscado
  - Patrones de especificaciones no ayudan del todo

En otras palabras

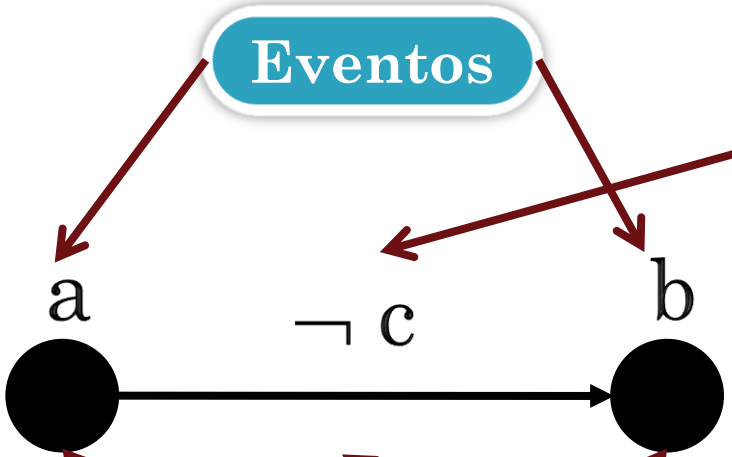
- Necesitamos formalidad. Pero queremos escribir y entender fácilmente las propiedades
- Sería bueno algo entre  y 
- Un look canchero, moderno. Algo como 

FVS: Feather Weight visual Scenarios

Una aproximación para la validación de especificaciones de software

FVS: Feather Weight Visual Scenarios

- Lenguaje de Especificación de Trazas
- Basado en Escenarios:



- *Ocurre a, luego b, y ninguna c en el medio*

Reglas FVS

- Implicación: Antecedente + Consecuente (s)

Intuición: "pattern matching sobre trazas"

Siempre que en una traza "encuentro" el antecedente entonces también debo encontrar al menos uno de los consecuentes.

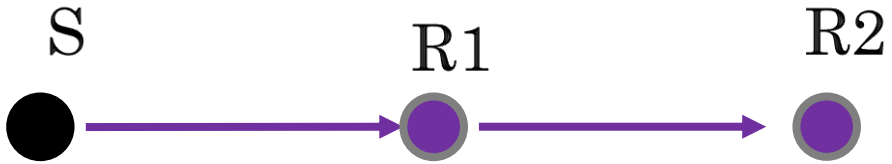
- Ejemplo: Asignación de Recursos

(Antecedente: En negro, Consecuente: En Violeta)

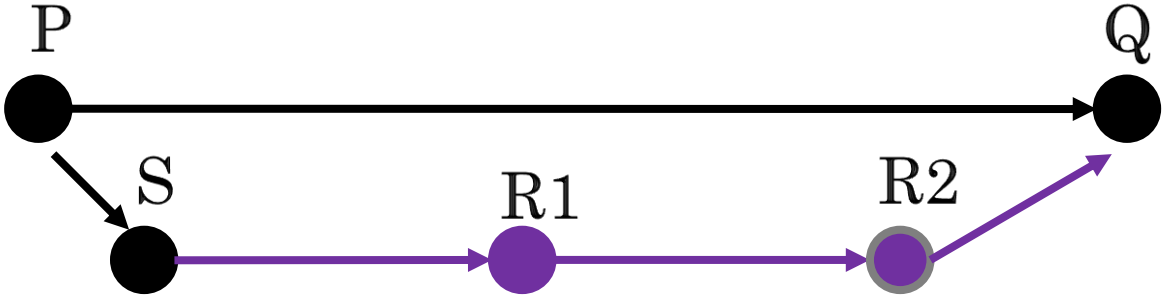
Patrones en FVS

- Response:

- Response Chain:



- Response Chain entre P y Q:



Comparaciones

- Comparamos FVS contra otros formalismos
- Consideramos los siguientes aspectos

1. **Especificaciones pequeñas**

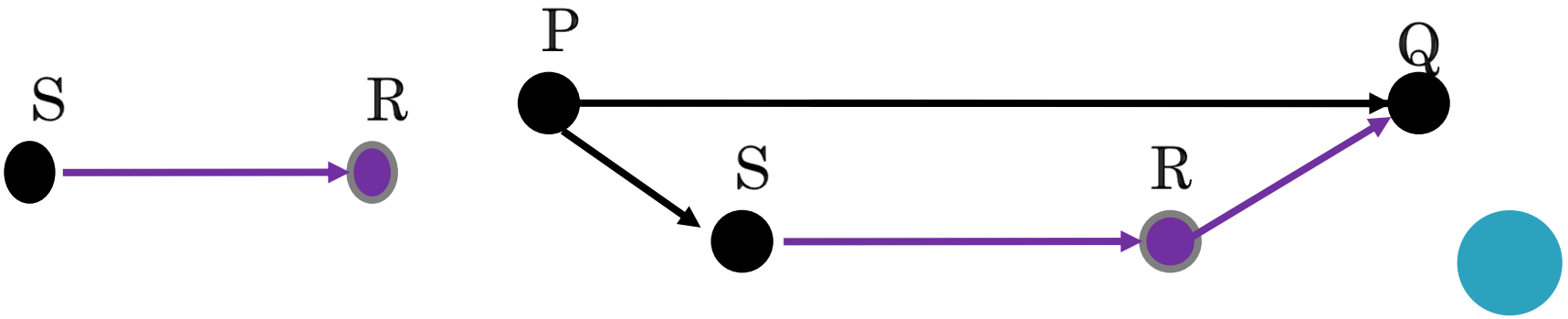
2. **Facilidad de Validar**
   a) Comparar propiedades
   b) Comportamiento complementario

3. **Modificabilidad**

Especificaciones pequeñas

- Crecimiento de las fórmulas LTL
- Pasaba de $\Box(S \rightarrow \Diamond R)$ a
- $((P \wedge \neg Q \wedge \Diamond Q) \rightarrow (S \rightarrow (\neg Q \; U \; (R \wedge \neg Q))) \; U \; Q)$

- En FVS



Modificabilidad

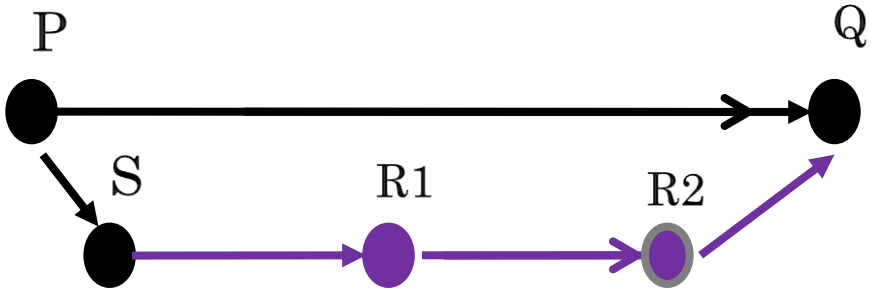
- Teníamos este caso

$$\Box((P \wedge \Diamond Q) \rightarrow (S1 \rightarrow (\neg Q U (R1 \wedge \neg Q \wedge X(\neg Q \ U R2))))UQ)$$

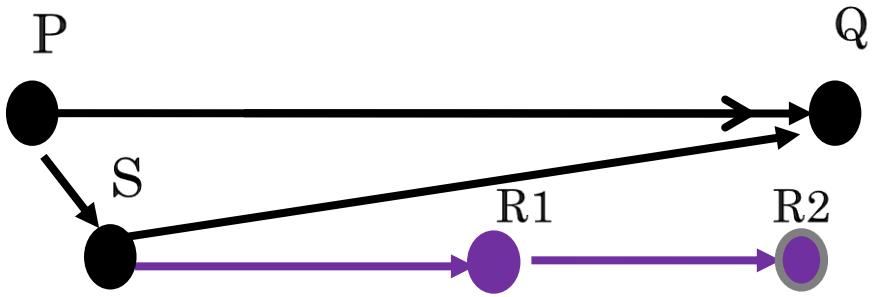
- Qué ocurre si se desea especificar que las respuesta pueden venir después de Q?

Modificabilidad en FVS

- FVS :



- Si R1 y R2 pueden ocurrir luego de Q, simplemente se cambia la relación de precedencia.



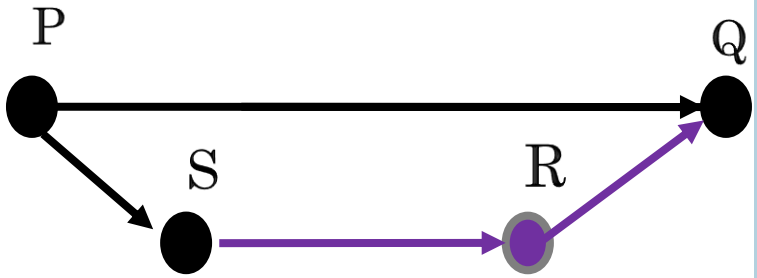
Razonando la negación

- Con la siguiente fórmula
- $\square((P \wedge \neg Q \wedge \Diamond Q) \rightarrow (S \rightarrow (\neg Q \; U \; (R \wedge \neg Q))) \; U \; Q)$

- Preguntas posibles:
- ¿Es esta la propiedad que realmente deseo?
- ¿Que escenarios me llevan a violar esta fórmula?

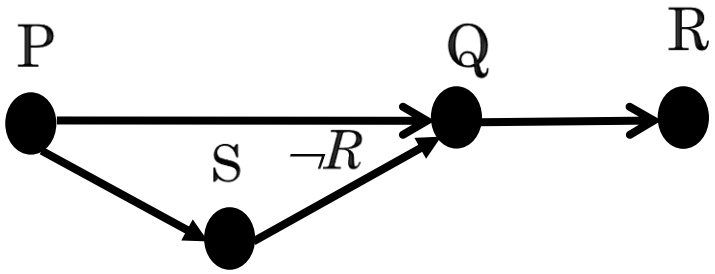
Anti-Scenarios en FVS

- En FVS, dada una regla, se pueden obtener escenarios que representan situaciones que violan la propiedad.
- Regla FVS para el patrón Response

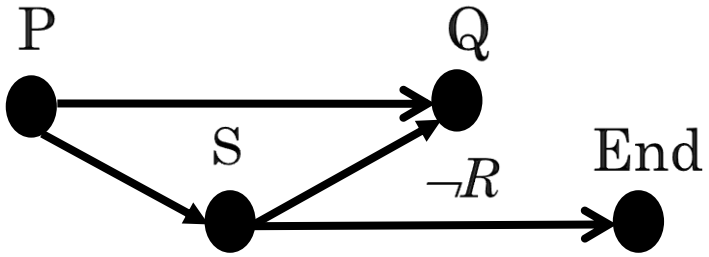


- Anti-Escenarios:

La respuesta llega tarde



La respuesta nunca llega



Conclusiones

- Validar Propiedades es clave.
- Notaciones formales son necesarias, y deben manejar mecanismos de validación.
- Las propiedades deben ser fáciles de comparar, complementar, modificar.
- Los patrones de especificación ayudan, pero todavía persisten problemas graves.
- FVS puede ser un buen paso inicial
  - Se modelaron en FVS todos los patrones de especificación.
  - Cuenta con mecanismos deseables de validación de propiedades

Acerca de…

- **LaFHIS: La**boratorio de **F**undamentos y **H**erramientas para la **I**ngeniería de **S**oftware.
- Somos:

Temas que se investigan en Lafhis

Nueva Optativa:

Verificación y Validación Automatizada

Técnicas de análisis automático de modelos
y de código


FIN