Desarrollo de un sistema dinámico de verificación de software basado en un comprobador de satisfacibilidad lógica

Trabajo fin de máster del
Máster en Investigación en Ingeniería del Software
de la Universidad Nacional de Educación a Distancia

Ponente: Diego J. Romero López
Tutor: Elena Ruiz-Larrocha

Índice

1. Introducción

La complejidad del software llega a niveles en los que es imposible abarcar los estados del sistema.

Hay sistemas críticos que no deben fallar.

Si fallan, es una catástrofe.

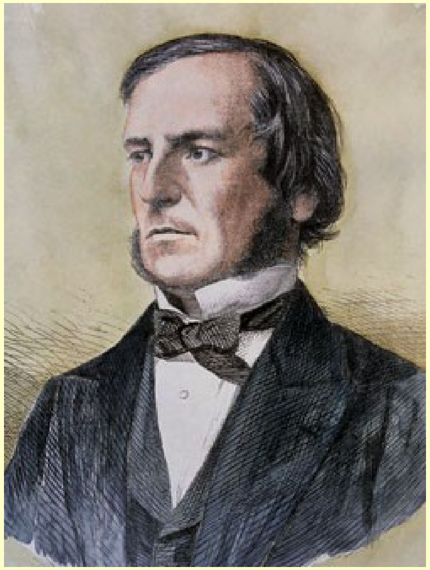
El coste de comprobar manualmente todos los estados de un sistema software es demasiado elevado.

¿Puede el sistema recuperarse si detecta un fallo?

Ejemplos de sistemas críticos:

- Aviónica.
- Transportes: vehículos, trenes...
- Sistemas médicos.
- Sistemas militares
- Banca
- Sistemas de posicionamiento geográfico

2. Lógica booleana



Definida por George Boole en el S. XIX.

Axiomatización del pensamiento filosófico.

Es la base de toda la lógica en computación.

Permite almacenar conocimiento con una expresión lógica.

Ej:

$$RESERVA_{estado\,=\,PENDIENTE} \wedge PAGA \Rightarrow RESERVA_{estado\,=\,PAGADO}$$

Definición

Una proposición lógica es:

- TRUE
- FALSE
- ¬ q, donde q es una proposición y ¬ el operador de negación.
- p OP q, con p y q proposiciones y OP un operador lógico.
- Operadores lógicos:
  - NOT
  - AND
  - OR
  - IMP

Leyes de la lógica proposicional

- Asociativas

$$p \vee (q \vee r) \Leftrightarrow (p \vee q) \vee r$$

$$p \wedge (q \wedge r) \Leftrightarrow (p \wedge q) \wedge r$$

- Distributivas

$$p \wedge (q \vee r) \Leftrightarrow (p \wedge q) \vee (p \wedge r)$$

$$p \vee (q \wedge r) \Leftrightarrow (p \vee q) \wedge (p \vee r)$$

- Leyes de De Morgan.

$$\neg(p \wedge q) \Leftrightarrow \neg p \vee \neg q$$

$$\neg(p \vee q) \Leftrightarrow \neg p \wedge \neg q$$

Satisfacibilidad

Una expresión lógica es *satisfacible* cuando se puede obtener el valor cierto (TRUE) para al menos una asignación de valores de verdad a sus proposiciones.

Satisfacibles

$$p \wedge (q \wedge r)$$

$$p \wedge (\neg p \vee r)$$

$$(p \wedge r) \vee \neg q$$

No satisfacibles

$$(p \wedge not\ r) \wedge (not\ p \wedge r)$$

$$p \wedge (\neg p \wedge r)$$

Nuestro objetivo es comprobar la satisfacibilidad de forma eficiente.

¿Cómo podemos hacerlo (de forma eficiente)?

- Analizador sintáctico.
- Procesamiento de texto.
- Resolutores SAT.
- Árboles Binarios de Decisión

3. Árboles Binarios de Decisión

Estructura de datos basada en la expansión de Boole-Shannon:

$$F = xF_{(x=1)} + \neg xF_{(x=0)}$$