descargalo de
www.dc.uba.ar/dynamite

Dynamite

un demostrador de teoremas
para el ingeniero de software

Permite
asegurar la validez
de propiedades **de una especificación**

basado en
PVS

demostrador interactivo de alto orden (SRI)

Cálculo de secuentes

fórmula
$\bigwedge a_i \Rightarrow \bigvee c_i$

$\Longleftrightarrow$

secuente
$a_1 \dots a_n \vdash c_1 \dots c_m$
antecedente — consecuente

Para demostrar f comenzamos con el secuente $\vdash f$
Lo transformamos aplicándole reglas

Proposicionales

$$\frac{A, B, \Gamma \vdash \Delta}{A \wedge B, \Gamma \vdash \Delta}\ {}^{\wedge\vdash} \qquad \frac{\Gamma \vdash A, \Delta \quad \Gamma \vdash B, \Delta}{\Gamma \vdash A \wedge B, \Delta}\ {}^{\vdash\wedge}$$

$$\frac{A, \Gamma \vdash \Delta \quad B, \Gamma \vdash \Delta}{A \vee B, \Gamma \vdash \Delta}\ {}^{\vee\vdash} \qquad \frac{\Gamma \vdash A, B, \Delta}{\Gamma \vdash A \vee B, \Delta}\ {}^{\vdash\vee}$$

$$\frac{B, \Gamma \vdash \Delta \quad \Gamma \vdash A, \Delta}{A \supset B, \Gamma \vdash \Delta}\ {}^{\supset\vdash} \qquad \frac{\Gamma, A \vdash B, \Delta}{\Gamma \vdash A \supset B, \Delta}\ {}^{\vdash\supset}$$

$$\frac{\Gamma \vdash A, \Delta}{\Gamma, \neg A \vdash \Delta}\ {}^{\neg\vdash} \qquad \frac{\Gamma, A \vdash \Delta}{\Gamma \vdash \neg A, \Delta}\ {}^{\vdash\neg}$$

Cuantificadores

$$\frac{\Gamma, A\{x \leftarrow t\} \vdash \Delta}{\Gamma, (\forall x{:}\, A) \vdash \Delta}\ {}^{\forall\vdash} \qquad \frac{\Gamma \vdash A\{x \leftarrow a\}, \Delta}{\Gamma \vdash (\forall x{:}\, A), \Delta}\ {}^{\vdash\forall}$$

$$\frac{\Gamma, A\{x \leftarrow a\} \vdash \Delta}{\Gamma, (\exists x{:}\, A) \vdash \Delta}\ {}^{\exists\vdash} \qquad \frac{\Gamma \vdash A\{x \leftarrow t\}, \Delta}{\Gamma \vdash (\exists x{:}\, A), \Delta}\ {}^{\vdash\exists}$$

Regla de corte

$$\frac{\Gamma, A \vdash \Delta \quad \Gamma \vdash A, \Delta}{\Gamma \vdash \Delta}\ {}^{Cut}$$

Trivialidad

$$\frac{}{\Gamma, A \vdash B, \Delta}\ {}^{Ax}$$

con A y B sintácticamente equivalentes

cierra la rama a la quese aplica

La demostración se completa cuando el árbol se cierra

Usa un
lenguaje fácil
de aprender y manejar

Su idea subyacente
puede aplicarse a otros formalismos

brinda
ayuda al usuario
mediante
técnicas automáticas

Alloy (MIT)

```
sig Address {}
sig Agent { attachments: set Domain }
sig Server extends Agent {}
sig Client extends Agent {
  knownAt: Address  > Domain
}
sig Domain {
  space: set Address,
  map: space  > Agent
}

fact { all d: Domain, g: Agent |
  g in Address.(d.map) => d in g.attachments
}
```

Define una relación *unitaria* cuyos elementos son atómicos

El campo attachments denota una relación en Agent x Domain

La signatura Client distingue ciertos agentes como clientes y define un nuevo campo knownAt

Los axiomas se agregan mediante facts

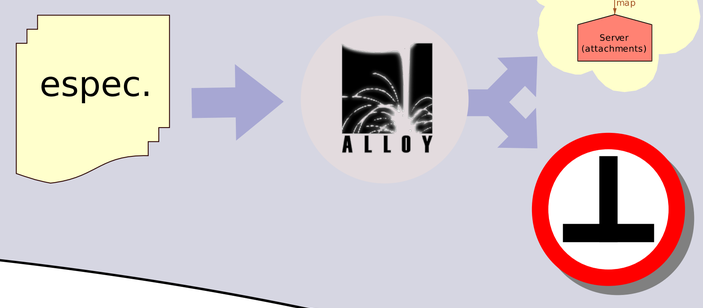
constantes relacionales y operaciones usuales: identidad (iden), vacío (zero), intersección (&), unión (+), composición (.), etc.

Las fórmulas se construyen con operadores de **Primer Orden**

- Validación de secuentes
- Introducción de hipótesis
- Separación en casos
- Descubrimiento automático de **fórmulas relevantes**

Alloy Analyzer (MIT)

dada una especificación construye un modelo (si es consistente)



Mariano M. Moscato — mmoscato@dc.uba.ar
Carlos G. Lopez Pombo — clpombo@dc.uba.ar
Marcelo F. Frias — mfrias@itba.edu.ar

generación automática de candidatos

para instanciar cuantificadores existenciales

Para demostrar una fórmula como

some a: A | P[a]

Es necesario exhibir un término de tipo A que cumpla con P para toda valuación.

La idea es utilizar al AlloyAnalyzer para generar estos términos.

Principal Problema

Cantidad de elementos sintácticos respecto de los elementos que denotan.

En los modelos de las T' deben representarse ambos tipos de elementos: sintácticos y semánticos.

1. Se traduce la teoría original a una que codifica las construcciones sintácticas propias de Alloy y aquellas que pueden crearse usando las definiciones dadas por el usuario.

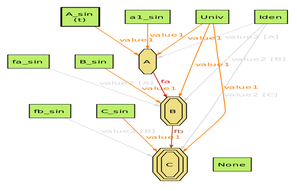
Teoría T
declaraciones
+axiomas

Sentencia
all x, y: A | some a: A | P[a]

2. También se genera un predicado P' que se satisface cuando el término correspondiente a t en la teoría original, satisface P.

Teoría T'
basada en la original
P'[t]

3. Se usa el Alloy Analyzer para generar un modelo de la teoría T', en el cual t satisfaga a P'



4. Se construye la expresión Alloy determinada por el elemento semántico correspondiente a t'

F[x, y, c, ...]

5. Se buscan contraejemplos para P[x+y]

6. Si hay contraejemplos, se modifica P' para que no vuelva a proponer el mismo candidato

7. Si no hay contra-ejemplos, se propone ese término como candidato a utilizar en la demostración

Bibliografía

Mariano M. Moscato, Carlos G. López Pombo, Marcelo F. Frias; **Dynamite 2.0: New Features Based on UnSAT-Core Extraction to Improve Verification of Software Requirements**; In A. Cavalcanti, D. Deharbe, M.-C. Gaudel, J. Woodcock, eds.: Proceedings of the 7th International Colloquium on Theoretical Aspects of Computing, Natal, Rio Grande do Norte, Brazil, Sept., 2010. Vol. 6255 of LNCS, Springer-Verlag.

Mariano M. Moscato, Carlos G. Lopez Pombo, Marcelo Frias; **Lessons Learnt on the Verification of Models Using Dynamite**; Symposium on Automatic Program Verification APV09; 15 de febrero de 2009, Río Cuarto, Argentina.

Marcelo F. Frias, Carlos G. Lopez Pombo, Mariano M. Moscato; **Alloy Analyzer + PVS in the Analysis and Verification of Alloy Specifications**; Thirteenth International Conference on Tools and Algorithms for the Construction and Analysis of Systems, TACAS'07, evento miembro de la European Joint Conferences on Theory And Practice of Software (ETAPS'07); 24 Marzo - 1 Abril, 2007, Braga, Portugal.

Marcelo F. Frias, Carlos G. Lopez Pombo, Mariano M. Moscato; **Dynamite: Alloy Analyzer + PVS in the Analysis and Verification of Alloy Specifications**; First Alloy Workshop (collocated with the Fourteenth ACM SIGSOFT Symposium on Foundations of Software Engineering); 6 de Noviembre, 2006; Portland, Oregon, EEUU.