



UNIVERSIDAD DE BUENOS AIRES  
FACULTAD DE CIENCIAS EXACTAS Y NATURALES  
DEPARTAMENTO DE COMPUTACIÓN

Tesis presentada para optar al título de  
Licenciado en Ciencias de la Computación

Anton Galitch

Director:

Codirector:

Buenos Aires, 2014



## Resumen

tes etst set set set set s

**Palabras claves:** test



## Índice general

1..	Introducción . . . . .	1
1.1.	Motivación . . . . .	1
1.2.	Objetivos Especificos . . . . .	1
1.2.1.	Extensión de Heterogenius . . . . .	1
1.2.2.	True Heterogeneity . . . . .	1
1.2.3.	Ampliación del árbol de análisis . . . . .	1
2..	Preliminares . . . . .	3
2.1.	Cálculo de Secuentes . . . . .	3
2.2.	Demostraciones Heterogeneas . . . . .	3
2.3.	Heterogenius . . . . .	3
2.4.	TPTP-FOF . . . . .	3
3..	Aportes . . . . .	5
3.1.	Heterogenius y el Árbol de Análisis . . . . .	5
3.1.1.	Limitaciones . . . . .	5
3.2.	Rediseño del Árbol de Análisis . . . . .	6
3.2.1.	Ramas alternativas . . . . .	6
3.2.2.	Nueva Clasificación de Acciones . . . . .	7
3.3.	Heterogeneidad Verdadera . . . . .	8
3.3.1.	Operaciones para el manejo de fórmulas . . . . .	8
3.4.	Extensión de Heterogenius con Herramientas de Lógica de Primer Orden . .	10
3.4.1.	Herramientas Usadas . . . . .	10
3.4.2.	Cálculo de secuentes . . . . .	10
3.5.	Detalles de Implementación . . . . .	12
3.5.1.	Extensión de las traducciones Rho . . . . .	12
3.5.2.	Demostradores de teoremas . . . . .	12
3.5.3.	Búsqueda de contraejemplos . . . . .	13
4..	Caso de Estudio . . . . .	15
5..	Conclusiones y Trabajos Futuros . . . . .	17
6..	Referencias . . . . .	19



# 1. INTRODUCCIÓN

## 1.1. Motivación

Analisis de software.  
Heterogeneidad.  
Tool support: lightweight y heavyweight.  
Usabilidad e interfaces.

## 1.2. Objetivos Especificos

### 1.2.1. Extensión de Heterogenius

El lenguaje de lógica de primer orden TPTP-FOF, al ser muy difundido en la comunidad de investigadores de demostradores automáticos de teoremas es soportado por numerosas herramientas. Entre ellas EProver y SPASS, demostradores automáticos de teoremas; EProver y Mace4, buscadores de modelos.

Para permitir el uso de todas éstas herramientas y otras, se decidió integrar TPTP-FOF con Heterogenius mediante la implementación de una  $\rho$ -translation desde el lenguaje *PDOCFA*.

### 1.2.2. True Heterogeneity

Heterogenius en su primera versión permitió realizar demostraciones heterogeneas mediante traducciones de secuentes. Cada secuente tenía fórmulas de un mismo lenguaje haciendo que los secuentes sean en realidad homogeneos.

Se decidió ampliar el concepto de heterogeneidad expandiendolo también a los secuentes. De esta forma se permitió tener mayor flexibilidad en las demostraciones al poder soportar fórmulas de distintos lenguajes en el mismo secuente.

### 1.2.3. Ampliación del árbol de análisis

Con el objetivo de permitir documentar todo el proceso de demostración (camino alternativos tomados, decisiones que no produjeron ningún resultado exitoso, etc), se decidió ampliar el concepto de árbol de analisis. Para esto se incluyó un nuevo tipo de ramificación, que permite soportar ramas de demostraciones alternativas y ramas de decisiones no exitosas.

[TODO: tal vez algun grafico mostrando las diferentes ramas.]





## 2. PRELIMINARES

### 2.1. Cálculo de Secuentes

### 2.2. Demostraciones Heterogeneas

### 2.3. Heterogenius

### 2.4. TPTP-FOF

*TPTP* (mil problemas para demostradores automáticos, por sus siglas en ingles) [1] es una biblioteca de problemas para demostradores automáticos de teoremas. La principal motivación para *TPTP* es permitir el testeo y evaluación de diferentes sistemas de demostradores automáticos de teoremas. Los problemas están en cuatro lenguajes: *THF*, *TFF*, *FOF* y *CNF*.

Anualmente se realiza la competencia *CASC* (CADE ATP System Competition [2]), una competencia de demostradores automáticos de teoremas donde los sistemas participantes compiten para probar la mayor cantidad de problemas de *TPTP*. Ésta competencia resulta una muy buena prueba para evaluar el funcionamiento de las herramientas automáticas disponibles.

*FOF* [3] es un lenguaje de lógica de primer orden con igualdad. Su elección de *FOF* está motivada por su difundido uso y soporte que tiene de las herramientas automáticas y su expresividad a nivel de sintaxis. Una especificación de *TPTP-FOF* es una lista de fórmulas que además tienen un nombre y un tipo. Por ejemplo:

```
fof(john,axiom,(
    human(john) )).

fof(all_created_equal,axiom,(
    ! [H1,H2] :
        ( ( human(H1)
            & human(H2) )
        => created_equal(H1,H2) ) )).

fof(john_failed,axiom,(
    grade(john) = f )).

fof(someone_got_an_a,axiom,(
    ? [H] :
        ( human(H)
            & grade(H) = a ) )).

fof(distinct_grades,axiom,(
    a != f )).

fof(grades_not_human,axiom,(
```

```

! [G] : ~ human(grade(G)) ).

fof(someone_not_john,conjecture,(
  ? [H] :
    ( human(H)
      & H != john ) ) ).

```

Especificación típica en el lenguaje *TPTP-FOF*. La sintaxis contiene todos los elementos de la lógica de primer orden. **![X]** es el cuantificador universal sobre la variable  $X$ ; **?[X]** es el cuantificador existencial sobre la variable  $X$ . *human()*, *grade()*, *created\_equal()* son predicados; las variables empiezan con una letra mayúscula y las constantes con minúscula. Los operadores disponibles son  $\sim$ ,  $|$ ,  $\&$ ,  $=>$ ,  $<=>$ ,  $=$  y  $!=$  y corresponden con la negación, la disyunción, la conjunción, la implicación, la doble implicación, la igualdad y la desigualdad.

En el ejemplo anterior la última fórmula es de tipo *conjecture*. Ésto indica al demostrador que es la fórmula que se quiere probar. Las otras fórmulas, de tipo *axiom* se interpretan como axiomas.

### 3. APORTES

#### 3.1. Heterogenius y el Árbol de Análisis

El árbol de análisis de Heterogenius es el elemento principal de un proceso de demostración. Es donde se realizan todas las acciones y es donde se refleja el camino tomado para lograr una demostración exitosa. Cada nodo del árbol representa un seciente en algún lenguaje soportado por Heterogenius. Las aristas corresponden con las acciones ejecutadas. Dependiendo del lenguaje en el que esté el seciente se habilitan diferentes acciones, pero en general se las puede dividir en tres categorías:

- las de cálculo de secientes, son acciones que transforman un seciente en otro. Algunas pueden producir múltiples secientes (por ejemplo la acción *Case*) creando varias ramas que tienen que ser demostradas para lograr un resultado en el nodo raíz.
- las acciones de traducción, traducen un seciente de un lenguaje a otro. Dependiendo de la expresividad del lenguaje esta traducción puede ser con pérdida o no.
- las acciones de búsqueda de contraejemplos, implican el uso de herramientas externas para buscar contraejemplos en el seciente donde se aplique la acción. El resultado puede ser positivo, se encontró un contraejemplo o negativo, no se encontró nada pero esto no nos dice nada de la existencia del contraejemplo.

##### 3.1.1. Limitaciones

En su versión actual, Heterogenius presenta varios problemas y limitaciones. A continuación se detallarán algunas de éstas limitaciones que fueron tratadas en ésta tesis:

Lo primero que se puede notar es la falta de *heterogeneidad verdadera*. Si bien Heterogenius permite realizar una demostración en diferentes lenguajes, cada seciente se limita a tener un solo lenguaje. Debido a esto se puede decir que Heterogenius, en realidad permite tener demostraciones heterogeneas con secientes homogéneos.

Otro gran problema con el que nos encontramos es la incapacidad del árbol de análisis de documentar el proceso de análisis completo. O sea, el árbol solamente muestra las acciones y pasos que llevan al éxito de una demostración. En el momento del análisis cuando una rama no lleva al resultado deseado y se quiere probar otro tipo de acciones, es necesario borrar la rama que no dió resultado. Esto lleva a que se pierdan partes del análisis que pueden ser útiles ya que documentan las acciones probadas (y que no dieron un resultado deseado) y no nos permite tener un historial de todo lo que se hizo en el análisis.

El árbol de análisis tiene solamente un tipo de ramas, que representan acciones obligatorias que se tienen que llevar a cabo para que el resultado se propague al nodo raíz. Esto limita la posibilidad de realizar demostraciones alternativas y experimentar en una demostración con diferentes tipos de herramientas, acciones, etc.

Debido a que las herramientas automáticas no siempre producen un resultado, decidimos que es necesario también documentar la aplicación de las acciones aún si no tienen un resultado definido. Todo esto aporta al objetivo general de documentar todas las acciones realizadas durante el análisis.

### 3.2. Rediseño del Árbol de Análisis

Para solucionar las limitaciones presentadas nos propusimos a rediseñar el árbol de análisis.

El primer cambio que realizamos es la implementación de *heterogeneidad verdadera*. Para lograr ésto flexibilizamos los secuentes permitiendo que también sean heterogeneos. Ésto nos permite manejar las demostraciones con mayor flexibilidad y abstraernos aún más del lenguaje usado para describir el problema. La implementación y fundamentos teóricos se explican con mayor detalle en la sección 3.3 .

En el árbol de análisis éste cambio se va a reflejar en la visualización de los nodos. Los nodos con secuentes heterogeneos van a estar marcados por una *H* y los nodos homogeneos quedarán sin ninguna marca.



Fig. 3.1: El nodo con H contiene un secuento heterogéneo. El otro nodo es homogéneo.

Para permitir la documentación y el historial de todas las acciones aplicadas así como la creación de caminos de análisis alternativos introducimos el concepto de *ramificación alternativa*:

#### 3.2.1. Ramas alternativas

Las ramas alternativas representan la existencia de multiples caminos en los que se subdivide el análisis para lograr un resultado.. En la interface de Heterogenius éste tipo de ramas, se representa con líneas punteadas y su significado semántico es el de un operador lógico “o”. Se corresponde con un camino alternativo en una demostración.

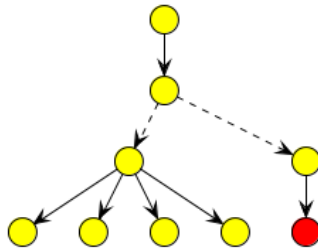


Fig. 3.2: La segunda rama alternativa presenta un contraejemplo. Ésto indica que existe un contraejemplo para el nodo del cual salen las ramas alternativas.

Un nodo con hijos conectados por las ramas alternativas, se entiende que vale si *alguna* de las ramas valen. Ésto es diferente de la ramificación normal (lineas continuas) que indica que el nodo padre vale si todos sus hijos valen.

Al aplicar una ramificación alternativa a un nodo del árbol de análisis, el nodo es copiado y agregado como sus propios hijos. Esto nos permite trabajar sobre las copias del nodo original. Cuando es necesario también se puede agregar ramas alternativas durante un análisis.

La principal ventaja de usar caminos alternativos es la de poder documentar todo el análisis que se hizo y las decisiones tomadas, incluso las decisiones que no llevaron al cumplimiento del objetivo. Por otro lado también nos permite experimentar con diferentes formas de probar lo mismo.

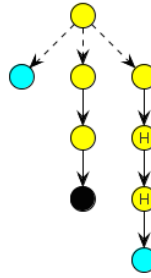


Fig. 3.3: Tres ramas alternativas: la primera y la última indican que no se encontró ningún contraejemplo. La segunda rama muestra que se pudo demostrar que el seciente vale, por lo cual el seciente del nodo raíz también vale.

### 3.2.2. Nueva Clasificación de Acciones

Debido a que la clasificación original de las acciones (3.1) es poco general, propusimos cambiar las categorías para permitir una mayor flexibilidad a la hora de agregar herramientas automáticas y acciones nuevas.

La nueva clasificación propuesta es:

- Acciones de cálculo de secuentes: son todas las acciones que toman un seciente como su entrada y devuelven uno o mas secuentes. Algunas de éstas acciones son: *Use*, *Lemma*, *Case*, *Skolem*, etc.
- Acciones de herramientas estructurales: son acciones que trabajan directamente sobre la estructura del árbol de análisis. Los traductores- $\rho$  forman parte de éste grupo, así como las nuevas acciones introducidas: *carga de antecedentes externos*, *traducción- $\rho$  de fórmulas* y *proyección de fórmulas*.
- Acciones de herramientas automáticas: éste grupo representa a las acciones para las que se usan herramientas automáticas como lo son los demostradores automáticos de teoremas y los buscadores de contraejemplos.

Ésta clasificación se refleja en la arquitectura de Heterogenius y permite guiar las futuras extensiones y funcionalidades adicionales.

### 3.3. Heterogeneidad Verdadera

Para solucionar otra de las limitaciones presentes en Heterogenius extendimos el concepto de heterogeneidad para lograr tener demostraciones verdaderamente heterogeneas en lugar de demostraciones homogeneas en un árbol de análisis heterogeneo.

La diferencia principal radica en que con la nueva implementación, los secuentes pueden soportar fórmulas de diferentes lenguajes. Así un secuyente puede ser de tipo homoganeo o heterogeneo. En el primer caso todas las fórmulas del secuyente usan el mismo lenguaje; en el segundo las fórmulas son de lenguajes distintos.

La ventaja de los secuentes heterogeneos es que se puede combinar fórmulas (lemmas, propiedades, teoremas) provenientes de distintas especificaciones escritas en lenguajes diferentes. De éste forma nos podemos abstraer del lenguaje en el que están escritas las fórmulas y concentrarnos en el análisis.

La principal limitación de los secuentes heterogeneos es que las herramientas (calculadores de secuentes, buscadores de contraejemplos, demostradores automáticos) trabajan con secuentes escritos en un solo lenguaje, o sea secuentes homogeneos. Debido a ésto se proveen nuevas operaciones para el manejo de fórmulas dentro de un secuyente:

#### 3.3.1. Operaciones para el manejo de fórmulas

Cada una de las siguientes operaciones puede cambiar o no la heterogeneidad de un secuyente. Dependiendo de los lenguajes de las fórmulas del resultado, el secuyente puede pasar a ser heterogeneo, homoganeo o mantener su tipo.

##### Proyección

Dado un secuyente, se selecciona un subconjunto de las fórmulas que se quiere proyectar y el nuevo secuyente se forma a partir de las fórmulas seleccionadas.

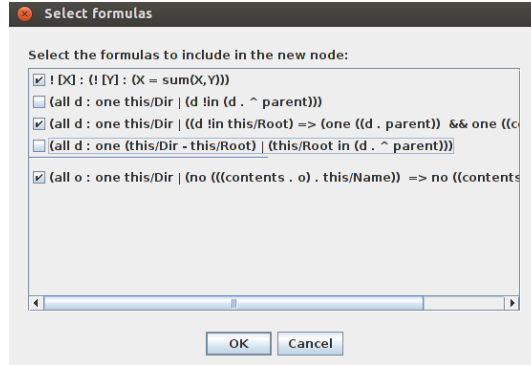


Fig. 3.4: Se marcan las fórmulas que se quiere proyectar al nuevo secuyente.

Dado un secuyente:

$$\frac{\alpha_1, \dots, \alpha_n}{\alpha_{n+1}, \dots, \alpha_m}$$

y un subconjunto  $\mathcal{C} \subseteq \{1 \dots m\}$ , el secuyente resultante:

$$\frac{\alpha_i \text{ con } i = 1 \dots n \text{ y } i \in \mathcal{C}}{\alpha_j \text{ con } j = n + 1 \dots m \text{ y } j \in \mathcal{C}}$$

Introducción de antecedentes desde una fuente externa

Ésta operación permite cargar desde un archivo de especificación, ya sea *Alloy* o *FOF* axiomas e introducirlas como antecedentes del seciente analizado.

Dado un seciente:

$$\frac{\alpha_1, \dots, \alpha_n}{\alpha_{n+1}, \dots, \alpha_m}$$

y un conjunto de fórmulas nuevas  $\{\beta_1 \dots \beta_k\}$ . El nuevo seciente es:

$$\frac{\alpha_1, \dots, \alpha_n, \beta_1, \dots, \beta_k}{\alpha_{n+1}, \dots, \alpha_m}$$

### Traducción

Se extendió el concepto de traducciones  $\rho$  para que se puedan traducir fórmulas por separado. El seciente resultante contendrá las fórmulas del seciente analizado en el lenguaje seleccionado.

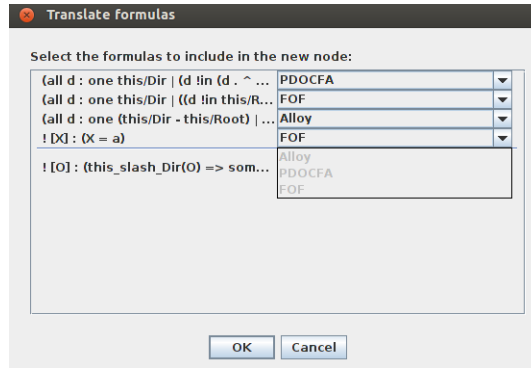


Fig. 3.5: Para cada fórmula se puede seleccionar el lenguaje al que se quiere traducir.

Dado un seciente  $S$

$$\frac{\alpha_1, \dots, \alpha_n}{\alpha_{n+1}, \dots, \alpha_m}$$

y una relación  $\mathcal{T} : \text{Formula} \times \text{Lenguaje}$  que indica el lenguaje seleccionado para cada fórmula del seciente  $S$ , el seciente resultante  $S'$  es:

$$\frac{\beta_1, \dots, \beta_n}{\beta_{n+1}, \dots, \beta_m}$$

donde  $\beta_i = \rho(\alpha_i, \mathcal{T}(\alpha_i))$ .

con  $\rho(\alpha, l) : \text{Formula} \times \text{Lenguaje} \rightarrow \text{Formula}$ : función de traducción para la fórmula  $\alpha$  al lenguaje  $l$ .

### 3.4. Extensión de Heterogenius con Herramientas de Lógica de Primer Orden

Existen numerosas herramientas que funcionan con el lenguaje de lógica de primer orden *TPTP-FOF*. Para permitir la integración de estas herramientas con Heterogenius y abrir el camino para la interacción con las futuras tecnologías basadas en éste lenguaje, se agregó *TPTP-FOF* al motor de Heterogenius como un lenguaje de análisis.

Junto a la integración de *TPTP-FOF*, se incorporaron los siguientes mecanismos para poder usar las herramientas correspondientes:

- Se permitió la carga de especificaciones escritas puramente en *TPTP-FOF*.
- Se agregó una  $\rho$ -translation desde las formulas *PDOCFA* a *TPTP-FOF*. [TODO: referenciar la seccion que explica esto en detalle].

Teniendo el soporte de *TPTP-FOF* por parte del motor de cálculo de secuentes de Heterogenius, integramos algunas de las herramientas mas difundidas en el ámbito de demostradores automáticos de teoremas para el lenguaje *TPTP-FOF*: *E-Prover* y *SPASS* como calculadores de secuentes; *E-Prover* y *Mace4* como buscadores de contraejemplos.

#### 3.4.1. Herramientas Usadas

##### E-Prover

Es un demostrador automático de teoremas de lógica de primer orden basado en el calculo por superposición. Además realiza búsquedas de modelos por lo cual también se usa como un buscador de contraejemplos.

##### SPASS

Es un demostrador automático de teoremas de lógica de primer orden con igualdad desarrollado por el Instituto Max Planck.

Desde el 2000, tanto *SPASS* como *E-Prover* ocupan los primeros lugares en la competencia anual de demostradores de teoremas *CASC* (CADE ATP System Competition).

##### Mace4

Es un buscador de modelos finitos y contraejemplos para lógica de primer orden. Se seleccionó debido a su notoriedad en el ámbito de herramientas automáticas de lógica de primer orden.

#### 3.4.2. Cálculo de secuentes

Al introducir el soporte para lenguajes de primer orden y herramientas automáticas fue necesario extender el cálculo de secuentes agregando algunas reglas nuevas.

Sea  $\Gamma \vdash \alpha$  el secuente que se quiere analizar, se introducen las siguientes reglas:

$$\textbf{Regla 1: } \frac{\Gamma \vdash \alpha}{\top} \text{ (si vale } \Gamma \vdash^{fof} \alpha \text{)}$$



Ésta regla indica que si se logra encontrar una demostración del seciente  $\Gamma \vdash \alpha$  con un demostrador automático *TPTP-FOF*, el resultado es  $\top$  y se termina la demostración.

**Regla 2:**  $\frac{\Gamma \vdash \alpha}{\perp}$  (si existe  $\mathcal{M} \in Mod^{fof}(\Gamma)$  y  $\mathcal{M} \not\models^{fof} \alpha$  )

Si de lo contrario, se logra encontrar un modelo de  $\Gamma$  que no satisface  $\alpha$ , éste modelo es un contraejemplo y el resultado del análisis es  $\perp$ .

**Regla 3:**  $\frac{\Gamma \vdash \alpha}{\Gamma \vdash \alpha}$  (si no)

Por último, debido a que la lógica de primer orden no es completa y la ejecución de las herramientas automáticas está limitada por un timeout, un caso posible es el de no encontrar ni una demostración ni un contraejemplo. En éste caso no se sabe nada y el resultado es el mismo seciente.

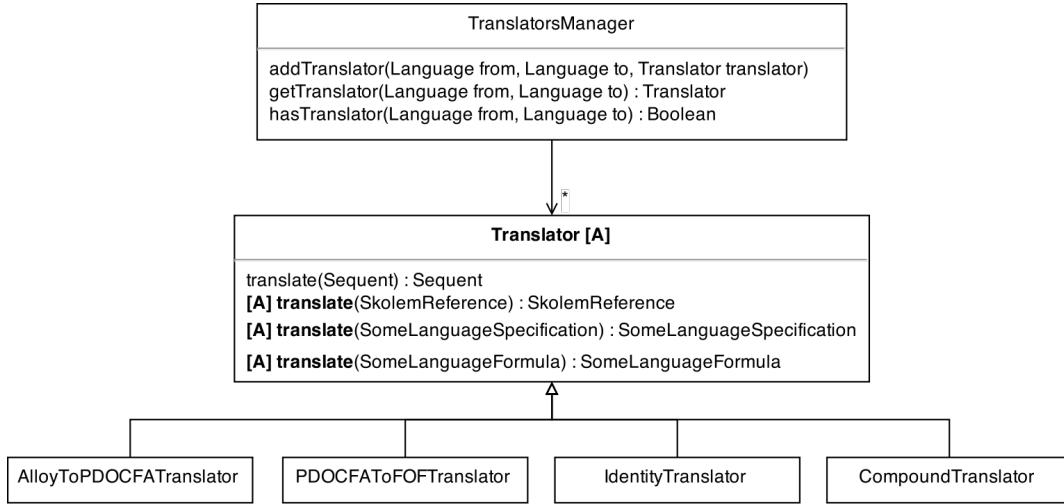
Al usar algún demostrador automático de teoremas se aplican las reglas 1 y 3. En cambio cuando se realiza una búsqueda de contraejemplo, las reglas usadas son 2 y 3.

### 3.5. Detalles de Implementación

#### 3.5.1. Extensión de las traducciones Rho

Debido a los cambios introducidos fue necesario refactorizar el diseño de la infraestructura de las traducciones  $\rho$ . Lo primero que se hizo fue agregar un *TranslationsManager*, un objeto encargado de manejar todas las traducciones soportadas por el sistema.

Por otro lado los traductores (subclases de *Translator*) deben implementar los tres métodos abstractos definidos en la clase padre. Cada uno de éstos métodos permite un control más fino de las traducciones al separar el seciente en sus partes, que son: una referencia de skolemización, una especificación y la fórmula analizada.



Se proveen los traductores de *Alloy* a *PDOCFAT*, de *PDOCFAT* a *TPTP-FOF* así como el *CompoundTranslator* que permite componer los traductores para lograr traducciones transitivas, por ejemplo de *Alloy* a *TPTP-FOF*.

#### 3.5.2. Demostradores de teoremas

##### Preparación del seciente

El formato *TPTP-FOF* que usan las herramientas automáticas agregadas presenta un archivo de texto con una lista de fórmulas escritas en el lenguaje *TPTP-FOF*. Cada una de éstas fórmulas debe tener un tipo que puede ser: *axiom* o *conjecture* (existen mas tipos pero son irrelevantes en éste caso).

Una fórmula de tipo *axiom* se considera verdadera y se la toma como parte de la especificación. Las fórmulas de tipo *conjecture* son las que el demostrador automático va a tratar de probar.

Para convertir el seciente analizado al formato *TPTP-FOF* se agregan todas las fórmulas del antecedente con el tipo *axiom* y las fórmulas del consecuente con el tipo *conjecture*.

Al ejecutar un demostrador automático con la entrada preparada de éste modo, se va a tratar de probar las fórmulas del consecuente usando que las fórmulas del antecedente son verdaderas.

## Integración con Heterogenius

Para integrar las herramientas automáticas, tanto los demostradores de teoremas como los buscadores de contraejemplos fue necesario refactorizar y extender el diseño de algunas partes de la arquitectura de Heterogenius. Algunos de los objetivos y criterios que se usaron durante el rediseño fueron lograr abstraer las herramientas usadas y hacer un diseño lo suficientemente extensible y abierto para la integración con nuevas herramientas en el futuro.

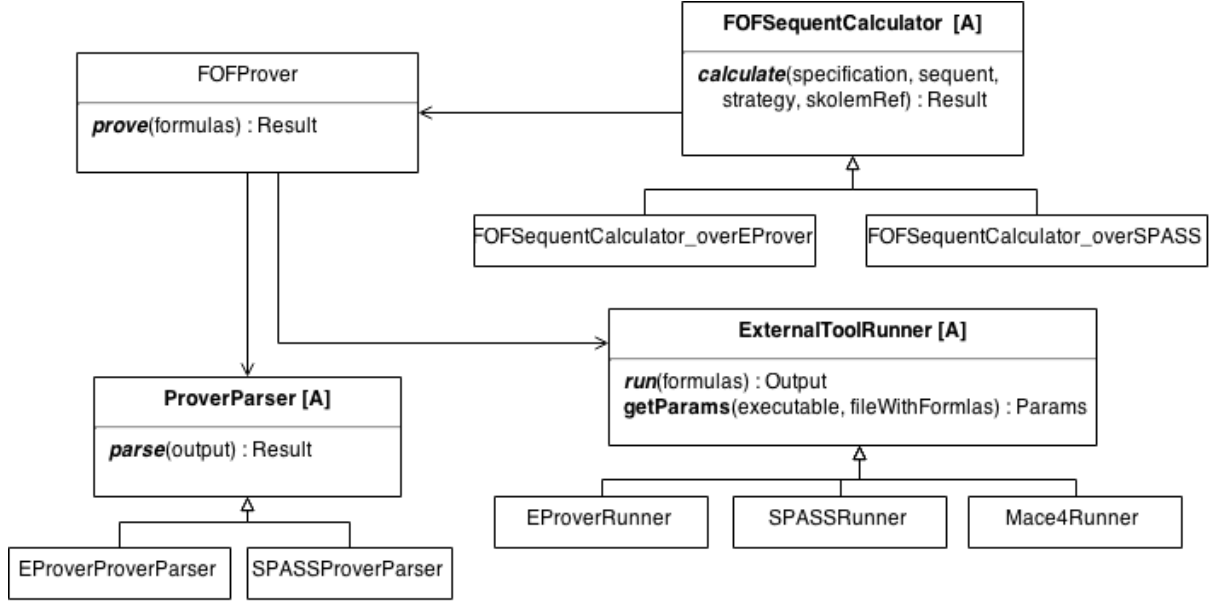


Fig. 3.6: Demostradores automáticos usados como calculadores de secuentes.

Un calculador de secuentes basado en un demostrador automático del lenguaje *TPTP-FOF* se crea subclasificando la clase abstracta **FOFSequentCalculator**. La nueva clase va a contener un objeto *FOFProver* compuesto con los objetos de las subclases de **ProverParser** y de **ExternalToolRunner** correspondientes.

Para agregar una herramienta nueva primero se debe subclasificar la clase abstracta **ExternalToolRunner**. Ésta clase abstrae las particularidades de ejecución de la herramienta especificando los parametros necesarios. Luego se extiende la clase **ProverParser** que se encarga de procesar el texto correspondiente a la salida de la herramienta.

### 3.5.3. Búsqueda de contraejemplos

#### Preparación del seciente

Tanto *Mace4* como *E-Prover* se usan para buscar contraejemplos de los secuentes *TPTP-FOF*. Como las dos herramientas son buscadores de modelos, para preparar el seciente lo que se hace es armar una lista de fórmulas de tipo *axiom* a partir de las fórmulas del antecedente y del consecuente del seciente analizado.

Cuando todas las fórmulas que se envían como entrada a los buscadores de modelos son de tipo *axiom*. Las herramientas tratan de buscar un modelo para el conjunto de los axiomas recibidos.

Lo que se quiere lograr es encontrar un contraejemplo, entonces se hace necesario negar el seciente y buscar un modelo para la negación:

Sea  $\{\alpha_1 \dots \alpha_n\} \Rightarrow \{\beta_1 \dots \beta_m\}$  el seciente bajo análisis. La negación se puede escribir como:

$$\bigwedge_{i=1}^n \alpha_i \wedge \bigwedge_{j=1}^m \neg \beta_j \quad (3.1)$$

Con lo cual para armar la lista de fórmulas de entrada, para cada  $\alpha_i$  del antecedente se agrega  $\alpha_i$  como axioma y para cada  $\beta_i$  del consecuente se agrega  $\neg \beta_i$  también como axioma.

Encontrar un modelo para una especificación armada de éste modo implica la existencia de un contraejemplo para el seciente procesado.

### Integración con Heterogenius

También como en el caso de los demostradores automáticos fue necesario refactorizar el diseño de los buscadores de contraejemplos para permitir una mayor flexibilidad a la hora de agregar nuevas herramientas con soporte de *TPTP-FOF*.

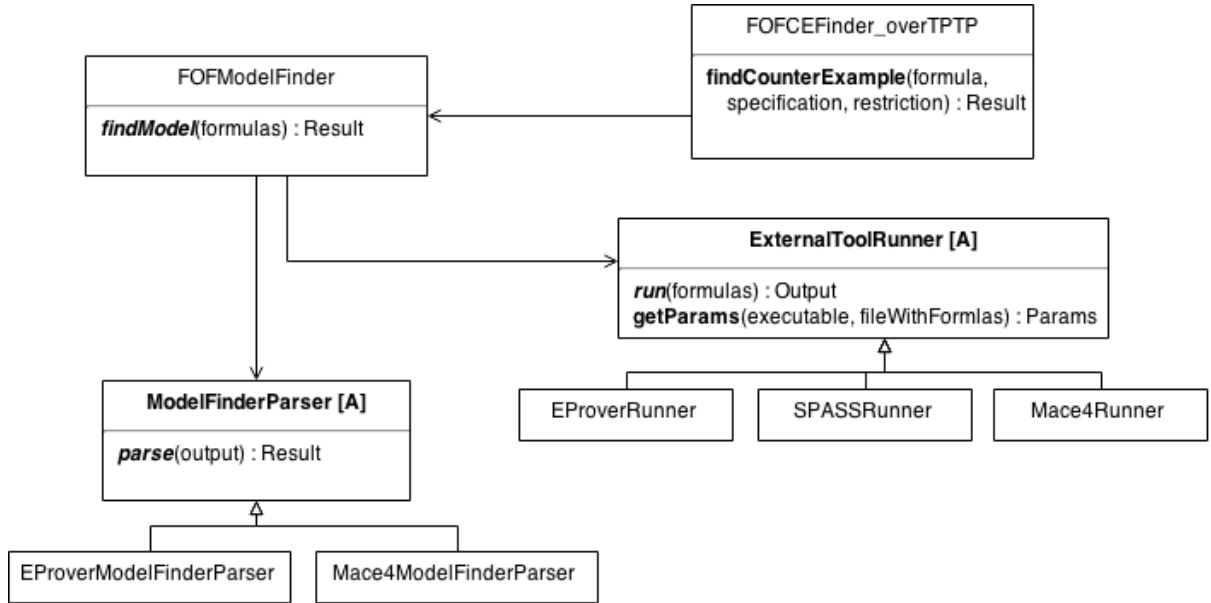


Fig. 3.7: Buscadores de contraejemplos.

El diseño de los buscadores de contraejemplos basados en *TPTP-FOF* es análogo a los calculadores de secientes. Se usa la misma clase **ExternalToolRunner** que en el caso anterior para la abstracción de la ejecución de la herramienta en sí. Pero en lugar de subclasificar **ProverParser** se subclasifica la clase abstracta **ModelFinderParser** que tiene un comportamiento análogo.

#### **4. CASO DE ESTUDIO**



## 5. CONCLUSIONES Y TRABAJOS FUTUROS

.....





## **6. REFERENCIAS**



## Bibliografía

- [1] Thousands of Problems for Theorem Provers <http://www.cs.miami.edu/~tptp/>
- [2] The CADE ATP System Competition <http://www.cs.miami.edu/~tptp/CASC/>
- [3] FOF: First Orden Formula <http://www.cs.miami.edu/~tptp/TPTP/SyntaxBNF.html>

