Análisis de la herramienta TLA+ Proof System

Pablo Celayes, Giovanni Rescia, Ariel Wolfmann

Facultad de Matemática, Astronomía y Física
Universidad Nacional de Córdoba

Resumen Agente: Cuando yo le diga hola señor Thompson, usted dice, hola
Homero: ¡Bien!
Agente: ¡¡Hola señor Thompson!!
Homero: ... ...
Agente: ¡¡Recuerde!! ¡su nombre ahora es Homero Thompson!
Homero: ¡¡Enterado!!
Agente: ¡¡Hola señor Thompson!!
Homero: (al otro agente)¡¡Creo que le habla a usted!!!. ...

Keywords: model checking, proof system, recibirse

1. Contexto de creación de la herramienta

En 1977, Amir Pnueli introdujo el uso de lógica temporal para describir el comportamiento de un sistema. En principio, un sistema se podía describir con una sola fórmula de lógica temporal. En la práctica no era así. La lógica temporal de Pnueli era ideal para describir ciertas propiedades de los sistemas, pero poco adecuada para muchas otras. Es por ello que normalmente se la combinaba con maneras más tradicionales de describir sistemas. [1]

A fines de los 80's Leslie Lamport inventó el lenguaje TLA (Temporal Logic of Actions, una variante simple de la lógica temporal de Pnueli). TLA facilita la tarea de describir un sistema completo en una sola fórmula. La mayor parte de una especificación TLA se compone de Matemática ordinaria, no temporal. La lógica temporal sólo juega un rol significativo en la descripción de aquellas propiedades para las que realmente es buena. [2]

En 2001, Lamport comenzó a trabajar en el centro *Microsoft Research* de Mountain View, Estados Unidos. De esta etapa surge el proyecto TLA+ Proof System (TLAPS) que actualmente se desarrolla como parte del proyecto *Tools for Proofs* en el centro conjunto de investigación *Microsoft Research - INRIA* de Palaiseau, Francia. El proyecto cuenta con una activa lista de mail para discusiones relativas a su uso y un sistema público de seguimiento de *bugs*.

2. Objetivo de la herramienta

TLAPS es una herramienta que verifica mecánicamente la correctitud de pruebas escritas en TLA+. Éste es un lenguaje de especificación de propósito general, orientado a sistemas concurrentes y distribuidos.

En general, una prueba de TLA+ es una colección de sentencias estructuradas jerárquicamente, donde cada sentencia tiene una afirmación, injustificada o justificada por una colección de hechos citados.

El propósito de TLAPS es verificar las pruebas de teoremas propuestas por el usuario, es decir, que la jerarquía de las sentencias de hecho establecen la veracidad del teorema si las afirmaciones fueran ciertas, y luego verificar que la afirmación de cada sentencia justificada es implicada por los hechos citados.

Si un teorema de TLA+ tiene una prueba con todas sus afirmaciones justificadas, entonces, como resultado de comprobar la prueba, TLAPS verifica que el teorema es cierto.

3. Descripción de la herramienta del lado del usuario

TLA+ es una herramienta de alto nivel para la descripción de los sistemas, especialmente sistemas concurrentes asíncronos y distribuidos. Fue diseñado para ser simple, muy expresivo, y permitir una formalización directa del razonamiento de aserciones tradicional. Se fundamenta en la idea de que la mejor manera de describir formalmente es con matemática simple, y que un lenguaje de especificación debe contener lo menos posible más allá de lo que se necesita para escribir matemática simple con precisión.

Para cumplir con el objetivo de formalizar el razonamiento de aserciones, TLA+ está basado en TLA (*Temporal Logic of Actions*), una variante simple de la lógica temporal lineal.

TLA+ utiliza a PlusCal como lenguaje de especificación. Para el modelado se basa en el modelo Standard. TLC es quien se encarga de verificar la corrección de los modelos. Finalmente, TLA+ Proof System es un asistente de pruebas, para verificar las especificaciones dadas en TLA+. En esta herramienta específica es donde nos enfocaremos el resto del documento.

Cabe aclarar que TLA+ posee una interfaz gráfica donde el usuario puede utilizar todas estas herramientas de manera sencilla e integrada.

3.1. PlusCal

TLA+ utiliza PlusCal como lenguaje en el cual se debe realizar la especificación. PlusCal es un lenguaje algorítmico que, a primera vista, parece un lenguaje de programación imperativo pequeño. Sin embargo, una expresión PlusCal puede ser cualquier expresión de TLA+, lo que significa cualquier cosa que pueda ser expresada con matemática. Esto hace a PlusCal mucho más expresivo que cualquier lenguaje de programación. Un algoritmo PlusCal se traduce (compila) en una especificación TLA+, la cual puede ser chequeada con las herramientas de TLA+. Fue diseñado para reemplazar al pseudocódigo, conservando su sencillez y proporcionando al mismo tiempo un lenguaje formalmente definido y verificable, en el cual las personas sin tantos conocimientos profundos en matemática pueden definir su especificacion de forma sencilla y facil de entender, para luego traducirla a TLA+ usando la herramienta.

3.2. Modelo STANDARD

En un modelo *Standard*, un sistema abstracto se describe como un conjunto de comportamientos, cada uno representando una posible ejecución del sistema, donde un comportamiento es una secuencia de estados y un estado es una asignación de valores a las variables. En este modelo, un evento, también llamado un paso, es la transición de un estado a otro en un comportamiento.

3.3. TLC

El chequeador de modelos TLC construye un modelo de estado finito de las especificaciones TLA+ para el control de las propiedades de invariancia. Genera un conjunto de estados iniciales que satisfacen la especificación y a continuación realiza una búsqueda en amplitud (Breadth-First Search) sobre todas las transiciones de estado definidas. La ejecución se detiene cuando todas las transiciones de estado conducen a estados que ya han sido descubiertos. Si TLC descubre un estado que viola un invariante del sistema, se detiene y ofrece una traza infractora. En caso de que haya alcanzado un estado que no tenga posibles acciones habilitadas, reporta un mensaje de error explicitando el posible *deadlock* (opcional).

TLC ofrece un método de declarar simetrías del modelo para evitar el fenómeno de *explosión combinatoria*. También paraleliza el paso de la exploración del estado, y se puede ejecutar en modo distribuido para repartir la carga de trabajo a través de un gran número de computadoras.

TLA+ es un ejemplo de lenguaje de mucha expresividad, puede ser fácilmente utilizado para especificar un programa que acepte una máquina de Turing arbitraria como entrada y puede determinar si se detendrá o no. Ningún chequeador de modelos puede manejar todas las especificaciones TLA+. TLC maneja un subconjunto de TLA+ que intenta incluir la mayoría de las especificaciones algorítmicas y propiedades de corrección, así como todas las especificaciones de diseño de protocolos y sistemas.

3.4. TLA+ Proof System

TLAPS, el sistema de prueba de TLA+, es una plataforma que extiende a TLA+, para el desarrollo y verificación mecánica de demostraciones.

El lenguaje de pruebas de TLA+ es declarativo y requiere cierto conocimiento previo de matemática elemental. Soporta desarrollo incremental y verifica la estructura jerárquica de la demostracion.

La herramienta traduce una demostración en un conjunto de pruebas independientes, y llama a una colección de verificadores que se encargan de chequearlas, que incluyen demostradores de teoremas, asistentes de pruebas, chequeadores de satisfacibilidad y procedimientos de decisión.

La versión actualmente disponible TLAPS maneja casi toda la parte no temporal de TLA+, como así también el razonamiento temporal necesario para

probar propiedades de seguridad (*safety*) estándar, en particular invariantes y simulación de transiciones, pero no para propiedades de vitalidad (*liveness*).

Desarrollar demostraciones es complejo y propenso a errores, por lo que previo a verificar la correctitud de una especificación, se recomienda chequear instancias finitas con TLC. Esto usualmente detecta varios errores sencilla y rápidamente. Una vez que TLC no encuentra errores, se puede intentar probar la correctitud de una demostración.

4. Aspectos técnicos de la herramienta

Las partes centrales de TLAPS son: el administrador de pruebas de LTA (*LTAPM*) y tres *back-ends* que éste invoca: ISABELLE, ZENON y SMT (Satisfiability Modulo Theories).

4.1. LTAPM

Una especificación en TLA+ consiste de un módulo raíz que puede (transitivamente), importar otros módulos por extensión e instanciación con sus respectivos parámetros. Cada módulo consiste en:

- parámetros (variables, o estados, y constantes sin interpretar)
- definiciones
- teoremas, que pueden tener su prueba

Un parámetro de un módulo puede ser una variable (estado), o una constante; en las fórmulas de TLA+ no se hace referencia explícita a las variables, en cambio, se usan dos copias de una variable v: v y v', que hacen referencia a un valor antes, y después de la transición. TLAPS se ejecuta invocando a TLAPM en el módulo raíz e indicando qué pruebas chequear.

(Verifying Safety Properties With the TLA+ Proof System).

El siguiente paso del LTAPM, es generar *obligaciones de prueba* para cada prueba terminal, es decir, las cosas que se tienen que demostrar, o corroborar, para probar la correctitud del teorema. Una vez generadas las obligaciones de prueba, organiza la ejecución de los backends para que las verifiquen.

Cada obligación de prueba es independiente y puede ser probada por separado. Cuando los backends no pueden encontrar una prueba en un límite razonable de tiempo, el sistema informará cuál obligación es la que falló, junto con su contexto (falló, cancelada por el usuario, omitida) y objetivo de la prueba. El usuario debe entonces determinar si falló debido a que depende de hechos o definiciones ocultas, o si el objetivo es demasiado complejo y necesita ser refinado con otro nivel de prueba. Esto indica que pueden haber obligaciones que son ciertas, y que por una cuestión de *timeout*, el backend las verifica como falsas.

Si un backend encuentra una prueba de una obligación, es decir si la puede verificar, genera una traza de la prueba realizada, llamada *certificación de prueba*. LTAPM mediará la certificación de los scripts de prueba en un entorno lógico confiable, que en el diseño actual es Isabelle/TLA+.

Finalmente, luego de certificar todas las obligaciones de prueba posibles generadas por las pruebas terminales, se procede a certificar el teorema en sí, en un proceso de dos pasos.

Primero, LTAPM la estructura de un lema (y sus pruebas en Isabelle/TLA+) que establece simplemente que la colección de obligaciones de pruebas terminales implican el teorema.

Luego, LTAPM genera una prueba del teorema usando la estructura del lema y las obligaciones ya certificadas. Si Isabelle acepta la prueba, estamos seguros de que la versión traducida del teorema es verdadera en Isabelle/TLA+.

4.2. Backends

4.3. Isabelle

5. Casos de aplicación de la herramienta

5.1. Amazon

Amazon Web Services utiliza TLA+ desde 2011. Haciendo *model checking* sobre TLA+ se han descubierto bugs en DynamoDB, S3, EBS, y en un *lock manager* interno distribuído.

Algunos bugs requirieron trazas de 35 pasos para ser detectados. También se ha usado model checking para verificar optimizaciones agresivas.

Además, las especificaciones TLA+ han resultado valiosas como documentación y ayudas de diseño. [3]

5.2. XBOX 360

Microsoft usó TLA+ para encontrar y resolver un *bug* en el sistema de memoria de la consola *XBOX 360*. [4]

6. Comparación con otras herramientas

Para realizar la comparación con otras herramientas decidimos primero comparar TLA+ con otra herramienta ya conocida como Alloy, y luego abocarnos específicamente a TLAPS.

6.1. TLA+

TLA+ y Alloy poseen un concepto de modelado similar, pero TLA+ es mucho mas expresivo que Alloy. La expresividad limitada de Alloy aparece como consecuencia del enfoque particular que toma su herramienta analizadora a la hora de realizar el análisis. Alloy es limitado a las relaciones sobre los identificadores, no tiene estructuras de datos ni recursión, esto lo hace muy eficiente para verificar modelos pequeños, pero para modelos más realistas presenta problemas.

Esta diferencia toma importancia en la práctica, ya que muchas especificaciones reales escritas en TLA+ son casi imposibles de escribir en Alloy.

6.2. TLAPS

Con respecto a TLAPS específicamente, las herramientas que encontramos como comparables son Isar, Focal y Coq.

- **Isar** es un lenguaje de prueba declarativo, que corre sobre Isabelle, pero tiene diferencias significativas, que llevan a un estilo de desarrollo de pruebas diferente. Por ejemplo, provee un acumulador para evitar referencias explícitas a los pasos de la prueba, lo cual es bueno para pruebas cortas, pero no tanto para demostraciones largas, que son típicas a la hora de verificar algoritmos. Además como Isabelle está diseñada para uso interactivo, los efectos de los comandos de Isar para demostraciones no son fácilmente predecibles, lo que estimula pruebas lineales en lugar de hacerlas jerárquicamente.
- El lenguaje **Focal** provee un conjunto de características funcionales y orientadas a objetos, que permiten expresar formalmente una especificación, avanzando incrementalmente, comprobando que la implementacion concuerda con la especificación y el diseño con los requerimientos. Es esencialmente un subconjunto de TLA+, que incluye el desarrollo de demostraciones jerárquicamente.
- **Coq** es un sistema asistente de demostraciones formales, que provee lenguaje formal para escribir definiciones matemáticas, ejecutar algoritmos y teoremas en conjunto, con un ambiente para el desarrollo semi interactivo de demostraciones verificadas automáticamente.

Básicamente, TLAPS realiza una integración entre las funcionalidades de Isar y Coq, ya que utiliza Isabelle y Zenon como backends.

7. Caso de estudio elegido

Ilustraremos la forma en que se trabaja con TLAPS a través de un ejemplo sencillo: El desarrollo de una prueba de correctitud del clásico *Algoritmo de Euclides* para el cálculo del *máximo común divisor* entre pares de números naturales.

8. Conclusiones particulares

Referencias

1. Pnueli, Amir
   The temporal logic of programs
   Proceedings of the 18th IEEE Symposium on Foundation of Computer Science, 1977
2. Lamport, Leslie
   Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers
   Addison-Wesley. ISBN 0-321-14306-X, 2002
3. Chris Newcombe, Tim Rath, Fan Zhang, Bogdan Munteanu, Marc Brooker, Michael Deardeuff
   How Amazon Web Services Uses Formal Methods
   Communications of the ACM, Vol. 58 No. 4, Pages 66-73
4. Lamport, Leslie
   Thinking for Programmers (at 21m46s) (Grabación de charla técnica).
   `http://channel9.msdn.com/Events/Build/2014/3-642#time=21m46s`
   San Francisco: Microsoft.