TLA+

Danilo Capkob, Rocio Carnavale, Juan Costamagna

FaMAF Universidad Nacional de Cordoba, Cordoba, Argentina

Abstract. TLA+ es un lenguaje de especificación formal, este es utilizado para diseñar, modelar, documentar, y verificar sistemas concurrentes. Las especificaciones son escritas con expresiones logicas y matematicas, para ello utiliza logica temporal. Esta herramienta posee un entorno de desarrollo integrado llamado TLA+ toolbox, desarrollado en java.

Keywords: TLA+ · Logica temporal · Verificación de sistemas concurrentes.

1 Introducción

Especificamos un sistema describiendo sus comportamientos permitidos, es decir, que es lo que debe hacer en el curso de la ejecución. En 1977, Amir Pnueli introdujo el uso de la logica temporal para describir el comportamineto de sistemas. En principio, un sistema podria ser descripto por una sola formula de logica temporal. En la practica, no es posible. La logica temporal de Pnueli fue ideal para describir algunas propiedades de sistemas, pero mala para otros.

Leslie Lamport se interesó en el problema de la Lógica Temporal Lineal (LTL) después de que una revisión por pares encontró un error en un documento que presentó sobre la exclusión mutua.Lamport se encontró por primera vez con la LTL de Pnueli durante un seminario de 1978 en Stanford organizado por Susan Owicki. Según Lamport, "estaba seguro de que la lógica temporal era una especie de absurdo abstracto que nunca tendría ninguna aplicación práctica, pero me pareció divertido, así que asistí". En 1980 publicó "Sometime" is sometimes "Not never" ("A veces" algunas veces es "no nunca"), que se convirtió en uno de los artículos más citados en la literatura de la lógica temporal. Lamport trabajó en escribir especificaciones de lógica temporal durante su tiempo en SRI (instituto y organización estadounidense de investigación científica sin fines de lucro con sede en Menlo Park, California), pero encontró que el enfoque no era práctico.

Hacia fines de la decada de los 80, Leslie Lamport inventó TLA, la logica temporal de las acciones, que es una simple variacion de la logica original de Pnueli. TLA es más practica para describir un sistema con una sola formula. Ademas la mayoria de las especificaciones TLA consisten de matematica ordinaria y no temporal. La lógica temporal juega un papel significativo solo en la descripción de las propiedades que es buena para describir. Tambien provee una buena forma de formalizar el estilo de razonamiento sobre sistemas que se ha probado que es mas efectivo en la practica, el estilo conocido como razonamiento

asercional. TLA permitió el uso de acciones en fórmulas temporales, lo que, según Lamport, "proporciona una manera elegante de formalizar y sistematizar todo el razonamiento utilizado en la verificación de sistemas concurrentes". TLA proporcionó una base matemática para el lenguaje de especificación TLA+. Más tarde, ese mismo año, Yuan Yu escribió el verificador de modelos TLC para las especificaciones TLA+; TLC se utilizó para encontrar errores en el protocolo de coherencia de caché para un multiprocesador Compaq. PlusCal se introdujo en 2009, y el sistema de prueba TLA+ (TLAPS) en 2012. TLA+2 se anunció en 2014, agregando algunas construcciones de lenguaje adicionales, así como un apoyo en el idioma mucho mayor para el sistema de pruebas.

TLA+ es muy bueno para especificar una gran cantidad de sistemas, desde interfaces de programas (APIs) hasta sistemas distribuidos. En Microsoft, se descubrió un error crítico en el módulo de memoria de 61 Xbox 360 durante el proceso de escritura de una especificación. Tambien se usó para escribir pruebas formales de corrección para los "Paxos bizantinos" (Paxos es una familia de protocolos para resolver el consenso en una red de procesadores no confiables, es decir, procesadores que pueden fallar), y en los "componentes de la tabla Pastry de hash distribuida", (Pastry es una red de superposición y una red de enrutamiento para la implementación de una tabla hash distribuida (DHT) similar a Chord). Amazon Web Services ha utilizado TLA+ desde 2011. El modelo verifica errores descubiertos en DynamoDB, S3, EBS y un administrador de bloqueo distribuido interno; algunos errores requerían rastros de estado de 35 pasos. La verificación de modelos también se usó para verificar optimizaciones agresivas. Además, se encontró que las especificaciones de TLA+ tienen valor como documentación y ayudas de diseño. Microsoft Azure usó TLA+ para diseñar CosmosDB, una base de datos distribuida globalmente con cinco modelos de consistencia diferentes.

2 Objetivo de la herramienta

TLA es un acrónimo de Lógica Temporal de Acciones, es un lenguaje para modelado de software por encima del nivel de código y de modelado de hardware por encima del nivel de circuito. Tiene un IDE (Entorno de Desarrollo Integrado) para escribir modelos y ejecutar herramientas para verificarlos. La herramienta más utilizada por los ingenieros es el verificador de modelos TLC.

TLA+ comenzó como una notación, y el verificador de modelos TLC, solo salió 5 años después. Como nunca fue pensado para ejecutarse, hay algunas suposiciones de que es un documento leído en lugar de ser un código ejecutable. Los modelos TLA+ suelen denominarse especificaciones.

PlusCal salió quince años después de TLA+. Se supone que TLC debe seguir perfectamente la semántica de TLA+, y como PlusCal es un estilo completamente diferente, no se puede ajustar al mismo esquema. Este es un lenguaje para escribir algoritmos, especialmente los concurrentes y distribuidos. Está destinado a reemplazar el pseudocódigo con un código preciso y comprobable. PlusCal parece un lenguaje de programación simple, pero con construcciones para

describir la concurrencia y la no determinación. Es mucho más expresivo que cualquier lenguaje de programación porque cualquier fórmula matemática puede usarse como una expresión de PlusCal. Un algoritmo PlusCal se traduce en un modelo TLA+ que puede verificarse con las herramientas de TLA+. Debido a que parece un lenguaje de programación, la mayoría de los ingenieros consideran que PlusCal es más fácil de aprender que TLA+. Pero debido a que parece un lenguaje de programación, PlusCal no puede estructurar modelos complejos tan bien como TLA+.

Las computadoras y las redes de computadoras son objetos físicos cuyos comportamientos están descritos por leyes físicas continuas. Se diferencian de la mayoría de los otros tipos de objetos físicos en que sus comportamientos se modelan naturalmente como conjuntos de eventos discretos. La programación, la ingeniería de software y la mayoría de las ciencias de la computación se ocupan de los modelos en los que el comportamiento de un sistema se describe como un conjunto de eventos discretos. Ningún modelo es una descripción completamente precisa de un sistema real. Un modelo es una descripción de algún aspecto del sistema, escrito para algún propósito.

TLA+ se basa en estado, lo que significa que modela una ejecución de un sistema como una secuencia de estados, donde un evento está representado por un par de estados consecutivos. Llamamos a una secuencia de estados un comportamiento; y llamamos a un par de estados consecutivos un paso en lugar de un evento. Un sistema se modela como el conjunto de comportamientos que describen todas sus posibles ejecuciones.

Como muchos métodos basados el estado, TLA+ describe un conjunto de comportamientos con dos cosas:

* Una condición inicial que especifica los posibles estados de inicio.
* Una relación de estado siguiente que especifica los pasos posibles (pares de estados sucesivos).

Especifican el conjunto de comportamientos cuyo primer estado satisface la condición inicial y cada paso satisface la relación del siguiente estado.

Este tipo de modelo a menudo se llama una máquina de estado, que es una máquina con un conjunto finito de estados posibles. Estas no son tan útiles como las máquinas de estados generales. Una máquina de Turing es un ejemplo de una máquina de estados. En una máquina de Turing determinista, la relación de estado siguiente permite a lo sumo un estado siguiente para cualquier estado, y no permite el estado siguiente para un estado de terminación.

El método más simple y práctico de describir con precisión la semántica de un lenguaje de programación, llamado semántica operacional, consiste esencialmente en mostrar cómo compilar un programa en el lenguaje en una máquina de estados. Dada una semántica operacional, cualquier programa en el lenguaje puede verse como una máquina de estado. La acción del siguiente estado especifica qué pasos pueden suceder; no especifica qué pasos, si los hay, deben ocurrir. Eso requiere una condición adicional, llamada propiedad de equidad . Una máquina de estado que modela un programa secuencial generalmente incluye la propiedad de imparcialidad de que se debe dar algún paso (el comportamiento

no debe detenerse) si la relación del estado siguiente permite que se tome un paso. Los modelos de programas concurrentes y distribuidos a menudo tienen propiedades de equidad más complicadas.

Se puede usar un modelo de máquina de estado sin una condición de imparcialidad para detectar errores de comisión, en los que el sistema hace algo incorrecto. No se puede utilizar para detectar errores de omisión, en los que el sistema no puede hacer algo. En la práctica, los errores de comisión son más numerosos y difíciles de encontrar que los errores de omisión. A menudo, los ingenieros no se molestan en agregar condiciones de equidad. Por lo tanto, primero debe aprender a escribir la condición inicial y la relación del siguiente estado en sus modelos TLA+. Más tarde, puedes aprender a escribir condiciones de justicia.

Una de las razones para modelar un sistema es verificar si hace lo que queremos. Lo hacemos verificando si el modelo satisface las propiedades que creemos que afirman que el sistema hace lo que debe. TLA+ puede afirmar, y sus herramientas pueden verificar, solo que alguna propiedad de un comportamiento individual es cierta para cada comportamiento posible del modelo. Por lo tanto, TLA+ no puede afirmar que el 99% de todos los comportamientos posibles terminen en un estado correcto. Sin embargo, puede afirmar (y sus herramientas pueden verificar) que cada comportamiento posible termina en un estado correcto si su estado inicial pertenece a un conjunto particular que contiene el 99% de todos los estados iniciales posibles.

El tipo de propiedad más útil para verificar es una propiedad de invariancia, que afirma que algo es cierto para cada estado de cada comportamiento posible. A menudo, un ingeniero verificará solo las propiedades de invariancia de un modelo.

Para un modelo que contiene una condición de imparcialidad, también debe verificar las propiedades simples que aseguran que algo suceda eventualmente, por ejemplo, que cada ejecución finalmente se detiene. Esas propiedades, llamadas propiedades de vitalidad , se expresan fácilmente en TLA+.

La gran variedad de propiedades que queremos comprobar para los sistemas concurrentes no se puede expresar como propiedades de invariancia y vida simple. Se pueden expresar como máquinas de estado (posiblemente con condiciones de imparcialidad). Una máquina de estado se puede ver como la propiedad que se satisface con los posibles comportamientos de la máquina de estado. Podemos comprobar si otra máquina de estado satisface esta propiedad. Si lo hace, decimos que la otra máquina de estado implementa la máquina de estado.

En TLA+ no hay una distinción formal entre una máquina de estado y una propiedad. Ambos se describen mediante fórmulas matemáticas. Una máquina de estado es una fórmula que tiene una forma particular, diferente de la forma de una propiedad de invariancia o vida. Tanto la satisfacción de una propiedad y la implementación de una máquina de estado significa que una fórmula implica otra.

Hoy en día, la mayoría de los ingenieros verifican solo las propiedades de invariancia y las propiedades de vida simple. Sin embargo, incluso si nunca lo

hace, saber cómo se hace explica lo que significa que un programa implemente un modelo, lo que puede ayudarlo a evitar cometer errores en su código.

3 Descripción de la herramienta

TLA+ posee un entorno de desarrollo denominado TLA+ toolbox, esta desarrollada en java y se puede instalar en todos los sitemas operativos que cuenten con una version de java para poder correr esta herramienta. Una vez que se desarrolla el modelo o se carga a traves de un archivo con extensión .tla, se tiene la posibilidad de probar si este tiene problemas de concurrencia como deadlock. Esto se lleva a cabo creando un grafo a partir de la especificacion, para luego realizar BFS o DFS y reconocer los errores que nuestro modelo posee. El programa da como salida los trazas en donde se encontraron problemas. Existe la posibilidad de chequear una propiedad particular de nuestro modelo a traves de la evaluacion de una expresión.

Trabajos relacionados The contribution should contain no more than four levels of headings.

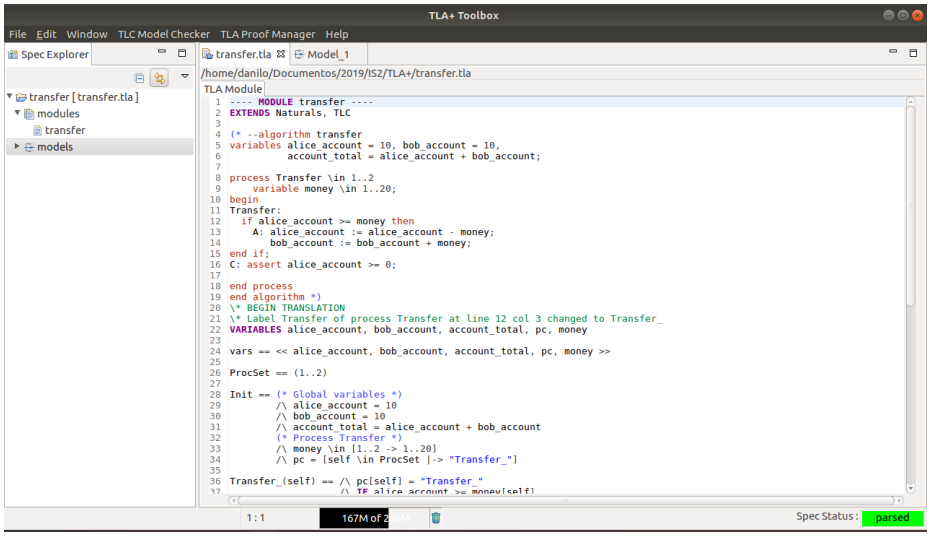

Fig. 1. Ventana principal de la IDE TLA+ toolbox

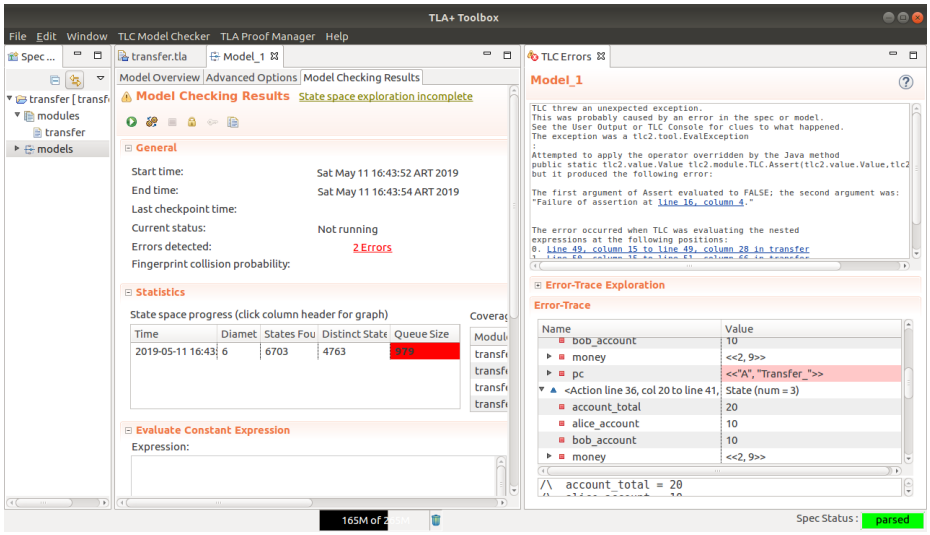

Fig. 2. Ventana de ejecución TLC model checker luego de encontrar errores en el modelo

4 Conclusión

References

1. Author, F.: Article title. Journal **2**(5), 99–110 (2016)
2. Author, F., Author, S.: Title of a proceedings paper. In: Editor, F., Editor, S. (eds.) CONFERENCE 2016, LNCS, vol. 9999, pp. 1–13. Springer, Heidelberg (2016). https://doi.org/10.10007/1234567890
3. Author, F., Author, S., Author, T.: Book title. 2nd edn. Publisher, Location (1999)
4. Author, A.-B.: Contribution title. In: 9th International Proceedings on Proceedings, pp. 1–2. Publisher, Location (2010)
5. LNCS Homepage, http://www.springer.com/lncs. Last accessed 4 Oct 2017