Universidad de Buenos Aires

Facultad de Ciencias Exactas y Naturales

Departamento de Computación

Síntesis de Controladores Actualizables Dinámicamente: Herramienta MTSA

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

Leandro Ezequiel Nahabedian

Director: Nicolás Roque D'Ippolito

Buenos Aires, Argentina 2014

SÍNTESIS DE CONTROLADORES ACTUALIZABLES DINÁMICAMENTE

Es esperado que muchos sistemas corran continuamente mientras el ambiente cambia y los requerimientos evolucionan, por lo tanto las implementaciones de dichos sistemas deben ser actualizados dinámicamente para satisfacer los cambios de requerimientos, respetando los cambios del ambiente. Lo complejo de este paso, es poder determinar en que puntos de la ejecución previa es seguro hacer la actualización, y si es seguro, como deberá seguir ejecutando el nuevo sistema. Tanto la máquina, como el ambiente y los requerimientos, pueden ser interpretados por modelos de comportamiento que son estructuras formales que definen acciones que pueden suceder. Mediante la síntesis de controladores podremos obtener modelos de forma correcta debido a que son obtenidos mediante construcciones.

El enfoque de esta tesis es definir y plantear formalmente mediante síntesis de controladores el problema de la actualización dinámica, detallando un conjunto de inputs necesarios para la solución del mismo. A su vez, desarrollaremos varios casos de test utilizando la herramienta MTSA (Modal Transition System Analyser) dejando constancia de que los inputs definidos son suficientes para obtener el controlador buscado.

Palabras claves: *Ingeniería de requerimientos; Síntesis de controladores; Actualización dianámica; Especificación basada en eventos; MTSA framework; Concurrencia, LTS; Fluent; LTL.*

AGRADECIMIENTOS

Quiero aprovechar este espacio para agradecer

A mi familia con todo mi amor.

Índice general

1.. Introducción 1
1.1. Motivación 1
1.2. Resumen de la contribución 2
1.3. Esquema de tesis 3

2.. Fundamentos teórico 5
2.1. El Mundo y la Máquina 5
2.2. Sistema de Transición Etiquetados (Labelled Transition System) 6
2.3. Lógica Lineal Temporal de Flujos (Fluent Linear Temporal Logic) 6
2.4. Problemas de síntesis de controladores 6
2.5. Juegos de dos jugadores 6
2.6. Procesos de estados Finitos (Finite State Process) 6

3.. MTSA como herramienta de modelado y síntesis 7

4.. Problema de actualización de controladores dinámicamente 9

5.. Validando los algoritmos 11
5.1. Casos de Estudio 11
5.2. Resultados 11

6.. Conclusiones y trabajos futuros 13
6.1. Conclusiones 13
6.2. Trabajo futuro 13

1. INTRODUCCIÓN

1.1. Motivación

Cuando trabajamos en ingeniería de requerimientos, nuestra tarea es relevar y documentar los objetivos y el funcionamiento del ambiente, para elaborar un conjunto de requerimientos cuya máquina a construir debe cumplir. Teniendo estas tres componentes, podemos formular la siguiente ecuación: R, D $\vDash$ G. Donde R son los Requerimientos de la máquina, D las asunciones del dominio y G los objetivos que la máquina deberá satisfacer.

Entonces, un problema clave de la ingeniería de requerimientos puede ser visto como un problema de síntesis, tal que dado un modelo de la máquina, un modelo del mundo y un conjunto de objetivos del sistema, permite construir un modelo operacional el cual satisface los objetivos. La técnica que resuelve esta ecuación mediante los modelos mencionados es llamada síntesis de controladores y esta siendo estudiada exhaustivamente en varios aspectos de la ingeniería de los requerimientos.

La construcción de dichos modelos son una de las principales herramientas en el diseño de sistemas concurrentes, debido a que nos facilita la detección de errores de diseño en las primeras etapas de desarrollo. Por otro lado, estos modelos de comportamiento pueden resultar complejos de construir. Utilizar la técnica de síntesis anteriormente mencionada facilita la construcción tomando modelos pequeños y propiedades lógicas, que suelen ser más sencillas de especificar y de validar.

Por lo tanto, un problema de control consiste en generar automáticamente una máquina que restrinja la ocurrencia de eventos controlables basado en los eventos del mundo que se producen. Es decir, teniendo la especificación de un ambiente, asunciones, objetivos y un conjunto de acciones controlables, podemos definir una máquina cuyo comportamiento concurrente con el ambiente satisfaga las asunciones y los objetivos del sistema.

Este problema está siendo ampliamente estudiado en diferentes contextos, desde sistemas autoadaptativos hasta composición automática de web-services. Para diseñar softwa-

res autoadaptativos los más evolucionados fueron los basados en arquitectura y los basados en requerimientos. El primero de estos utiliza reglas de adaptación que serán utilizadas para monitorear condiciones del conjunto de operaciones del sistema y definir acciones en run-time si las condiciones son desfavorables. Por otro lado, los basados en requerimientos extienden las técnicas de ingeniería de los requerimientos con el fin de representar a los requisitos de adaptación y/o la incertidumbre inherente del medio ambiente en el que opera el sistema. El modelo más utilizado en la comunidad, hoy en día, para definir sistemas autoadaptables es el modelo arquitectural.

Actualmente, trabajos recientes en el área de sistemas auto adaptativos definen una técnica para lograr encontrar estados de la ejecución del controlador en los cuales es seguro realizar una actualización y seguir ejecutando el nuevo controlador sin tener que apagar y prender el primero. Sin embargo, dicho problema no ha sido estudiado en el marco de síntesis de controladores.

1.2. Resumen de la contribución

Trabajos previos en el área de ingeniería de los requerimientos han estudiado la utilidad de diversas técnicas de sistemas adaptativos, debido a la necesidad de cambiar el ambiente o los requerimientos de un sistema sin frenar o interrumpir su ejecución.

La contribución central de esta tesis es la presentación de una técnica que produce la actualización automática de sistemas. Dicha técnica, propone mejorar ciertos problemas que las anteriores estrategias padecían, como la necesidad de conocer estados de la ejecución en los cuales es "seguro" actualizar, y si lo es, detectaremos cual es el estado equivalente en el sistema actualizado. Detallaremos en el capitulo BLA por qué y cómo resolvemos dichas complicaciones.

Por otro lado, lo novedoso del trabajo presentado es el enfoque usado. La actualización automática de sistemas ha sido estudiado en diversos enfoques pero no bajo el marco de síntesis de controladores. Al presentar dicha técnica bajo un marco no explorado abrimos a la comunidad científica un nuevo paradigma que podría potencialmente solucionar problemas en otras áreas relacionadas con la ingeniería de los requerimientos.

1.3. Esquema de tesis

2. FUNDAMENTOS TEÓRICO

2.1. El Mundo y la Máquina

Los primeros conceptos que detallaré estarán involucrados con la ingeniería de los requerimientos. Los puntos de vista mas relevantes son los de Zave y Jackson ([5, 1, 2]) por un lado, y los de Letier y Van Lamsweerde ([4, 3]) por el otro. Ambos puntos de vista distinguen a los problemas del *Mundo* y las soluciones de la *Máquina* como fundamentales para reconocer si las operaciones de la máquina soluciona los problemas planteados en el mundo. De hecho, el efecto de la máquina en el mundo y las suposiciones que hacemos acerca de este mundo son fundamentales para el proceso de toma de requerimientos. En el lado del mundo definimos una serie de problemas que existen en el mundo real que serán solucionados al construir una máquina. Fácilmente podremos notar que existen componentes en la máquina que interactúan directamente con el mundo siguiendo normas y procesos conocidos. Estas, forman parte de la intersección entre el mundo y la máquina. Por ejemplo un taladro, un brazo robótico o las reglas de procesamiento para cada elemento que entra en una linea de producción (véase la Fig. 2.1).

Así mismo, es esperado que la máquina proponga una solución al problema. Por ejemplo, en la Fig. 2.1 podemos ver que la célula de producción debe procesar cada producto, solo si están disponible en la bandeja de entrada en ese momento. Con la sentencia $EnBandeja[p] \rightarrow get.Bandeja[p]$ muestro que se espera que el brazo del robot solo podrá tomar los productos de la bandeja cuando estén listos. Finalizando, los fenómenos compartidos entre el mundo y máquina, es decir, los que se encuentran en la intersección, representa a la *interfaz*, donde la máquina interactúa con el mundo. También, podemos definir a los fenómenos del mundo como el *modelo del entorno* ya que el conjunto de estos describen los eventos que suceden en el mundo real.

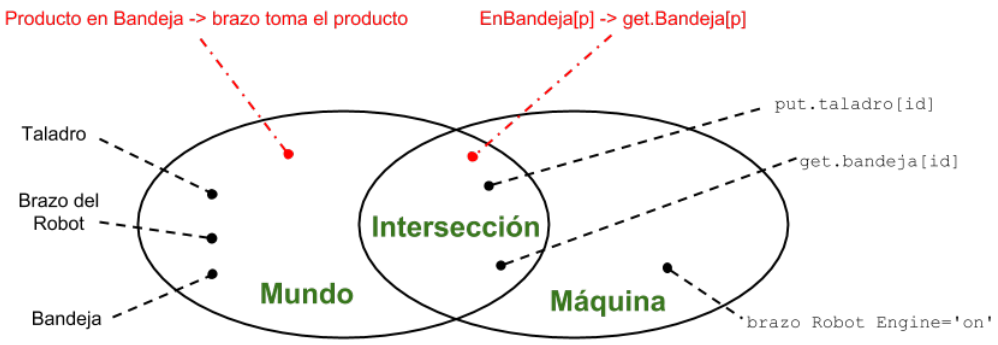

Fig. 2.1: El Mundo y la Máquina

2.2. Sistema de Transición Etiquetados (Labelled Transition System)

2.3. Lógica Lineal Temporal de Flujos (Fluent Linear Temporal Logic)

2.4. Problemas de síntesis de controladores

2.5. Juegos de dos jugadores

2.6. Procesos de estados Finitos (Finite State Process)

3. MTSA COMO HERRAMIENTA DE MODELADO Y SÍNTESIS

4. PROBLEMA DE ACTUALIZACIÓN DE CONTROLADORES DINÁMICAMENTE

5. VALIDANDO LOS ALGORITMOS

5.1. Casos de Estudio

5.2. Resultados

6. CONCLUSIONES Y TRABAJOS FUTUROS

6.1. Conclusiones

6.2. Trabajo futuro

Bibliografía

[1] M. Jackson. *Software Requirements &Amp; Specifications: A Lexicon of Practice, Principles and Prejudices*. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA, 1995.

[2] M. Jackson. The world and the machine. In *Software Engineering, 1995. ICSE 1995. 17th International Conference on*, pages 283–283, April 1995.

[3] A. Van Lamsweerde. Goal-oriented requirements engineering: A guided tour. In *Proceedings of the Fifth IEEE International Symposium on Requirements Engineering*, RE '01, pages 249–, Washington, DC, USA, 2001. IEEE Computer Society.

[4] A. Van Lamsweerde and E. Letier. Handling obstacles in goal-oriented requirements engineering. *Software Engineering, IEEE Transactions on*, 26(10):978–1005, Oct 2000.

[5] P. Zave and M. Jackson. Four dark corners of requirements engineering. *ACM Transactions on Software Engineering and Methodology*, 6:1–30, 1997.