Identificación Automática de Relaciones Metamórficas en Sistemas de Búsqueda*

Juan C. Alonso, Alberto Martin-Lopez, Sergio Segura, and Antonio Ruiz-Cortés

Unidad de Excelencia Smart Computer Systems Research and Engineering (SCORE)
Intituto de Investigación en Ingeniería Informática (I3US)
Universidad de Sevilla, España
{javalenzuela, alberto.martin, sergiosegura, aruiz}@us.es

Resumen Proveer al usuario con opciones de búsqueda avanzada que faciliten el acceso a los datos que busca es una característica común de muchos sistemas software, tales como sistemas de información, sistemas de comercio electrónico y APIs web. La complejidad y magnitud de estos sistemas provocan que en la mayoría de los casos sea imposible determinar si la salida obtenida es correcta. Una solución para aliviar este problema es la aplicación de *pruebas metamórficas*, que explotan las relaciones existentes entre los parámetros de un sistema, conocidas como *relaciones metamórficas* (RMs). Sin embargo, la identificación manual de RMs exige tiempo y esfuerzo, lo que limita su aplicación. En este artículo, describimos nuestro trabajo en progreso para el desarrollo de un método para la identificación automática de RMs en el contexto de sistemas de búsqueda. Nuestro primer prototipo muestra la efectividad de nuestra propuesta para generar cientos de RMs en sistemas reales como YouTube, IMDb o SkyScanner en cuestión de segundos.

Keywords: Pruebas metamórficas · Pruebas automáticas · Problema del oráculo

1. Introducción

La mayoría de sistemas software actuales ofrecen opciones de búsqueda que permiten filtrar los resultados ya sea de forma textual o a través de una interfaz gráfica, nos referimos a esta característica como *sistemas de búsqueda*. Algunos ejemplos son las páginas de comercio electrónico como Amazon (ej. "Buscar smartphones de menos de 300$"), plataformas de proyectos software como GitHub (ej. "Buscar proyectos de Node.js con más de 10 commits") o servicios de streaming como Netflix (ej. "Ordenar series por popularidad").

* Este trabajo ha sido parcialmente financiado por la Comisión Europea (FEDER), la Junta de Andalucía bajo los proyectos APOLO (US-1264651) y EKIPMENT-PLUS (P18-FR-2895), el Gobierno de España bajo el proyecto Proyecto HORATIO (RTI2018-101204-B-C21), financiado por: FEDER/Ministerio de Ciencia e Innovación – Agencia Estatal de Investigación y el Programa de becas FPU, concedido por el Ministerio de Educación y Formación Profesional de España (FPU17/04077).

Probar estos sistemas constituye un desafío, ya que en la mayoría de las ocasiones no es posible conocer su salida esperada, esto es lo que se conoce como el *problema del oráculo* [1]. Por ejemplo, determinar si los 35.775 resultados obtenidos al buscar en IMDb documentales estrenados entre 1998 y 2005 son correctos analizando únicamente esta búsqueda sería extremadamente difícil.

Las *pruebas metamórficas* abordan el problema del oráculo desde una perspectiva original [3]. En lugar de analizar los resultados de cada prueba de forma aislada, este tipo de pruebas explotan *relaciones metamórficas* (RMs) entre las entradas y salidas de múltiples pruebas. Por ejemplo, supongamos que repetimos la búsqueda en IMDb descrita anteriormente limitando la búsqueda a documentales que pertenezcan al género "comedy". Los resultados del nuevo caso de prueba (en inglés, follow-up test case) deberían ser un subconjunto de los resultados de la prueba original (source test case). De no ser así habríamos detectado un fallo en el sistema. Recientemente, varios grupos de autores han propuesto el concepto de *patrón de relación metamórfica* (PRM) [2,4], como abstracciones que representan un conjunto (puede que infinito) de RMs. Por ejemplo, la relación anterior es un ejemplo del PRM *conjuntivo*, propuesto por Segura et al. [4] que representa a esas relaciones donde el conjunto de resultados de una búsqueda es restringido mediante un cambio en las entradas (ej. añadiendo un filtro). Este tipo de patrones facilita la identificación de relaciones metamórficas, sin embargo, éste sigue siendo un proceso manual y costoso, lo que supone un gran obstáculo para la aplicación de la técnica.

En este artículo, presentamos un método para la generación automática de RMs explotando el concepto de PRM. Experimentos iniciales con múltiples sistemas reales demuestran que es posible generar cientos de RMs a partir de una especificación del sistema y de un conjunto de pruebas inicial en segundos.

2. Propuesta

En esta sección presentamos el método propuesto para la identificación automática de RMs en el contexto de sistemas de búsqueda.

2.1. Especificación de parámetros

En primer lugar, nuestro método recibe como entrada una especificación de los parámetros del sistema en formato YAML (Código 1.1). En concreto, proponemos un sistema de clasificación de parámetros de acuerdo con su capacidad para identificar RMs que sean instancias del catálogo de PRMs propuesto por Segura et al. (Código 1.2) [2,4]. Las posibles categorías a las que puede pertenecer un parámetro son las siguientes:

- **Filtros conjuntivos.** Parámetros que permiten reducir los resultados devueltos por una query, resultando en un subconjunto de la prueba original. Por ejemplo, los resultados devueltos al incluir los parámetros `countries` y `languages` en una búsqueda en IMDb deben ser un subconjunto de los del source test case.

- **Filtros disyuntivos.** Parámetros que permiten aumentar los resultados devueltos por una query, resultando en un superconjunto de la prueba original. Por ejemplo, los resultados de una búsqueda en Skyscanner con el parámetro `Add nearby airports` deberían ser un superconjunto de esa misma prueba omitiendo dicho parámetro.
- **Filtros disjuntos.** Parámetros con un conjunto discreto de valores que permiten dividir los resultados de una query en particiones disjuntas , sin elementos comunes entre los follow-up test cases Por ejemplo, no deben existir elementos comunes entre búsquedas en IMDb de títulos que el usuario no ha visto y títulos que sí ha visto.
- **Filtros completos.** Parámetros con un conjunto discreto de valores que permiten dividir los resultados de una query en particiones completas (no necesariamente disjuntas), la unión de las salidas de los follow-up test cases debe ser equivalente a la salida del source test case. Por ejemplo, al filtrar la búsqueda de películas en IMDb por géneros, la unión de los resultados de buscar todos los géneros por separado debe ser igual al resultado obtenido si no se realiza este filtrado.
- **Parámetros de ordenación.** Parámetros utilizados para cambiar el orden en el que se mostrarán los resultados. Por ejemplo, los resultados obtenidos al buscar películas en IMDb al establecer el parámetro `sorted by` con el valor "Popularity Ascending" deben ser los mismos si se cambia el valor de dicho parámetro, pero en distinto orden.
- **Valores por defecto.** Parámetros opcionales con valores por defecto claramente definidos. Este tipo de parámetros permite detectar relaciones en las que las salidas del source test case y los follow-up test cases deben ser equivalentes (incluyendo el orden). Por ejemplo, los resultados obtenidos al realizar una búsqueda en iTunes con el valor por defecto del parámetro `country` ("US") deben ser los mismos que si no se incluye dicho parámetro.

Código 1.1. Parámetros.

```yaml
features:
- id: SearchTitle
  description: Advanced search
  # Parámetros
  parameters:
  - name: title_type
     description: 'Title type'
     type: string
     enum:
      - TV Series
      - Documentary
     required: false
  - name: release_date_start ...
  - name: release_date_end ...
  - name: genres ...
  - name: countries ...
  - name: languages ...
  - name: your_ratings ...
  - name: sorted_by
     description:'Sorting method'
     type: string
     enum:
      - Popularity Ascending
      - A-Z Descending
     required: true
```

Código 1.2. Especificación de RMs.

```yaml
# Clasificación de parámetros
  configuration:
     ordering_parameters:
     - sorted_by
     conjunctive_filters:
     - title_type
     - release_date_start
     - release_date_end
     - genres
     - countries
     - languages
     - your_ratings
     disjoint_filters:
     - parameter: your_ratings
       values:
        - Titles I have Seen
        - Titles I have not Seen
     - parameter: adult_titles
     complete_filters:
     - parameter: genres
       all_value: empty
     - parameter: countries
       all_value: empty
     - parameter: languages
       all_value: empty
```

2.2. Generación de relaciones metamórficas

Nuestra propuesta permite generar RMs partir de la especificación de los parámetros del sistema (Códigos 1.1 y 1.2) y un conjunto inicial de casos de prueba en formato CSV (conjunto de pares <parámetro,valor>). A cada caso de prueba (source test case), se le aplican una o varias transformaciones consistentes en añadir, eliminar o modificar aleatoriamente uno o varios de los parámetros de la prueba, resultando en uno o más follow-up test cases. Por ejemplo, utilizando como source test case una búsqueda en IMDb que contenga los parámetros `title_type`, `countries` y `genres`, nuestro método podría generar una RM conjuntiva creando un follow-up test case que no contenga el filtro conjuntivo `genres`, de manera que los resultados de esta nueva prueba deban ser un superconjunto de los devueltos por la prueba original. Como resultado, se genera una RM que contiene el source test case y los follow-up test cases, además de un oráculo que especifica la relación que debe existir entre estos.

3. Evaluación preliminar

La Tabla 1 muestra resultados preliminares en cuatro sistemas industriales: dos aplicaciones web y dos APIs web. El número total de RMs identificadas por sistema varía entre 17.8 (SkyScanner) y 1075.3 (YouTube), aumentando en función del número de parámetros y source test cases (Tabla 1). Limitando el número máximo de follow-up test cases a generar por RM a 6, el número medio de follow-up test cases por sistema (Media FUTC) varía entre 1.28 (SkyScanner) y 3.18 (IMDb), alcanzando el máximo de 6 en todos los sistemas.

Sistema	Parámetros	Source test cases	Media FUTC	RMs	Tiempo (s)
IMDb	32	20	3.18	1072.7	65.31
Prestashop	11	18	2.31	284.9	1.03
SkyScanner	14	12	1.28	17.8	0.01
YouTube	31	64	1.61	1075.3	35.62

Tabla 1. Relaciones metamórficas identificadas (Valor medio tras 10 ejecuciones).

4. Trabajo futuro

Nuestro trabajo futuro incluye llevar a cabo una evaluación más extensa con más sistemas y en términos de cobertura de código y capacidad de detección de errores, además de desarrollar un método que nos permita priorizar las RMs generadas.

Referencias

1. Barr, E.T., Harman, M., McMinn, P., Shahbaz, M., Yoo, S.: The oracle problem in software testing: A survey. IEEE transactions on software engineering **41**(5), 507–525 (2014)
2. Segura, S., Durán Toro, A., Troya, J., Ruiz-Cortés, A.: Metamorphic relation patterns for query-based systems (07 2019). https://doi.org/10.1109/MET.2019.00012
3. Segura, S., Fraser, G., Sanchez, A.B., Ruiz-Cortés, A.: A survey on metamorphic testing. IEEE Transactions on software engineering **42**(9), 805–824 (2016)
4. Segura, S., Parejo, J.A., Troya, J., Ruiz-Cortés, A.: Metamorphic testing of restful web apis. IEEE Transactions on Software Engineering **44**(11), 1083–1099 (2017)