QuickCheck를 사용한 neoESPA 입출력 데이터 생성 시스템

천준석[1], 김연어[2], 류샤오[3], 왕인서[4], 변석우[5], 우균[6]
부산대학교[12346], 경성대학교[5]
{jscheon, yeoneo, liuxiao, inseowang}@pusan.ac.kr, swbyun@ks.ac.kr, woogyun@pusan.ac.kr

neoESPA I/O Data Generating System Using QuickCheck

Junseok Cheon, Yeoneo Kim, Xiao Liu, Inseo Wang, Sugwoo Byun, Gyun Woo
Pusan National University, Kyungsung University

요 약

전세계적으로 창궐하고 있는 COVID-19로 인해 온라인 수업을 일상적으로 진행하게 되었다. 프로그래밍 수업의 경우에는 학생들의 프로그래밍 능력을 높이는 것이 가장 중요한 목표이므로 교사가 강의한 내용을 바탕으로 출제되는 프로그래밍 과제를 활용하고 있다. 학생이 제출한 소스 코드를 교사가 수동으로 채점하는 것은 시간이 오래 걸리므로 자동 채점 도구가 제안되었는데, neoESPA가 그 중 하나이다. neoESPA는 교사가 입력한 입력데이터와 출력데이터를 바탕으로 학생이 제출한 코드를 평가한다. 여기에 사용되는 입출력 데이터는 교사가 직접 생성하는 것이 일반적인데, 이는 시간이 오래 걸린다는 단점 외에도 잘못된 입출력 데이터를 생성할 수 있다는 위험성이 있다. 이 논문에서는 Haskell의 QuickCheck를 이용하여 neoESPA의 입출력 데이터를 자동으로 생성하는 시스템을 제안한다. 원의 넓이를 구하는 프로그램에 대한 입출력 데이터 생성 실험 결과, 백만 개의 입출력 데이터를 생성하는 데 160초 정도밖에 걸리지 않는 것을 확인하였다.

1. 서 론

2019년 12월 중국 우한시에서 발생한 COVID-19는 우리의 일상을 바꿔놓았다. 2021년 5월 현재 전세계적으로 1억 5천만 명 이상의 감염자가 발생하였고[1], 아직도 진정될 기미를 보이지 않고 있다. 우리는 현재까지도 마스크를 착용하며, 사회적 거리 두기를 실시하고 있다.

COVID-19의 창궐은 기존의 교육 방식에도 영향을 주었다. COVID-19의 확산 방지를 위해 오프라인 수업이 아닌 Zoom이나 e학습터 등의 시스템을 이용한 온라인 수업 위주로 진행되고 있다. 이러한 온라인 수업은 역병이 진정될 때까지 지속될 것으로 예상된다.

프로그래밍 수업의 경우에도 다른 수업과 마찬가지로 온라인 상에서 이루어지고 있다. 학생은 교사의 수업을 들은 후, 이를 바탕으로 과제를 해결하여 제출하게 된다. 학생의 평가는 학생이 제출한 코드를 실행하여 제대로 된 결과값이 나왔는지를 확인하는 것으로 이루어진다.

학생의 평가를 위해 교사가 수동으로 소스 코드를 하나하나 실행해보는 것은 비효율적인 작업이다. 이 문제를 해결하기 위해 과제 자동 평가 시스템인 neoESPA[2]가 제안되었다. neoESPA에서는 먼저 출제 과제에 대한 입출력 파일을 시스템 상에 업로드한다. 이후, 학생이 코드를 제출하면 여기에 입력 파일을 넣어서 실행하여 출력 파일이 올바로 나오는지를 확인하여 평가한다.

현재 neoESPA에서는 입출력 파일을 수동으로 만드는 한계점이 있다. 교사가 먼저 코드를 작성한 후, 여기에 임의의 입력값을 넣어서 출력값을 생성한다. 이 과정은 번거로울 뿐만 아니라, 교사가 실수하여 잘못된 입출력 데이터를 만드는 경우가 발생할 가능성도 있다.

이 논문에서 우리는 neoESPA의 입출력 파일을 자동으로 생성하는 시스템을 제안하려 한다. 우리는 입출력 파일 자동 생성을 위하여 Haskell[3]의 QuickCheck 패키지를 사용하였다. 이를 이용해 입출력 데이터를 대량으로 빠르게 생성할 수 있다.

이 논문의 구성은 다음과 같다. 2절에서는 제안 시스템을 이해하는 데 필요한 관련 연구를 소개한다. 3절에서 제안 시스템을 설계한 후 구현한 내용을 소개한다. 4장에서는 제안 시스템을 실험하여 평가한 후, 5장에서 한계점을 고찰한다. 마지막으로 6장에서 결론을 맺는다.

2. 관련연구

2.1 neoESPA

neoESPA는 2015년에 개발된 온라인 과제 자동 평가 시스템이다. 교사는 출제할 문제에 대한 입출력 데이터를 미리 만들어서 neoESPA 상에 올려둔 후, 특정 기간 내에 학생이 코드를 제출하도록 한다. 학생이 코드를 제출하면, 입력 데이터를 이용하여 제대로 된 출력값이 나오는지를 평가하여 채점하게 된다. 학생들이 제출한 코드를 이용하여 표절 탐지를 수행하고, 코드의 품질을 측정하여 학생이 더 나은 코드를 작성하도록 돕는다.

우리는 neoESPA의 입출력 데이터를 자동으로 만드는 시스템을 구축하려 한다. 현재는 교사가 과제 코드와 입력값 10개를 작성한 후, 이를 이용하여 출력값 10개를 얻는다. 이 입출력값 20개를 neoESPA에 업로드하여 이를 바탕으로 학생의 코드를 채점하게 된다. 현재 neoESPA는 이 과정이 수동으로 이루어지며, 10개의 테스트 집합만

을 이용하는데, 이를 자동화하고 더 많은 테스트 집합을 이용하도록 변경하고자 한다.

2.2 Haskell과 QuickCheck

Haskell은 1990년에 발표된 순수 함수형 언어이다. 함수형 언어는 함수를 이용해 프로그램을 구성하는 프로그래밍 스타일을 기본으로 한다. 그중에서도 순수 함수형 언어는 '같은 식은 언제 계산해도 같은 값이 출력된다'는 참조 투명성(referencial transparency)이 있다.

QuickCheck 라이브러리는 1999년에 Koen Claessen과 John Hughes에 의해 제안된 Haskell용 테스트 라이브러리이다[4]. QuickCheck는 속성 테스팅(property testing) 개념에 기반하여 구축되어 있다. 즉, 함수가 갖추고 있는 property를 지정하면, QuickCheck가 그에 맞는 값을 자동으로 생성하여 테스트한다. 이를 통해 지정된 property를 그 함수가 만족하고 있는지를 검증한다.

우리는 Haskell의 QuickCheck를 neoESPA의 입출력 데이터를 만드는 데 사용하고자 한다. Haskell은 강력한 형 시스템을 지원하므로 입출력 파일을 만들 때 다른 언어에 비해 안전하게 만들 수 있기 때문이다. 또한, QuickCheck을 이용한 속성 테스팅 기법을 코드 채점에도 적용할 수 있을 것으로 생각하여 이를 개발하였다.

3. 시스템 설계 및 구현

이 절에서는 본 논문에서 제안하는 시스템의 설계에 대해 설명하려고 한다. 그림 1은 이 논문에서 제안하는 시스템의 전체 구조를 나타낸 것이다. 교사가 neoESPA용 입출력 데이터를 만들기 위해서는 문제에 맞는 Haskell 코드를 먼저 작성해야 한다. 그리고 문제의 입력 제약에 맞는 입력 데이터 명세 코드를 작성해야 한다. 이를 QuickCheck 라이브러리에 넣으면 제약 조건에 맞는 입출력 데이터 쌍이 만들어진다. 이를 neoESPA에 넣으면 문제가 생성된다.

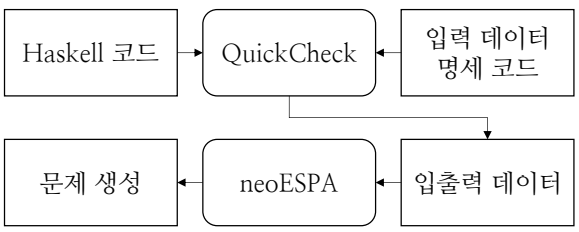

그림 1 시스템 전체 구조

QuickCheck를 이용하여 문제를 생성할 예제 코드는 Haskell로 작성하여야 한다. Haskell은 형에 엄격한 언어이기 때문에 문제에 대한 입출력 생성 시 유용하기 때문이다. 또한, QuickCheck가 여러 언어로 포팅되기는 하였지만, Haskell에서 파생된 것이므로 신뢰성이 있는 원본 버전을 사용하였다.

교사가 이 시스템을 사용해서 지름이 주어졌을 때 원의 넓이를 구하는 문제를 낸다고 가정하면, 코드 1과 같은 Haskell 코드를 작성해야 한다. 코드 1에는 지름이 주어졌을 때, 원의 넓이를 구하는 코드와 입력 데이터에 대한 스펙 코드가 같이 담겨 있다. 먼저 QuickCheck와 관련 함수를 사용하기 위해 1~2번째 줄과 같이 모듈을 import한다[5]. 다음으로, 입력 데이터에 대한 제약을 주기 위해 4번째 줄과 같이 **Diam**이라는 데이터 형을 새로 생성하였다. 다음으로 6~9번째 줄에서 원의 넓이를 계산한다. 이 때, 입력값이 양수인 경우에만 계산을 하고 0이나 음수가 입력되었을 때는 각각 zero와 negative가 출력되도록 하였다.

```
1 import Test.QuickCheck
2 import Test.QuickCheck.Property
3
4 data Diam = MkDiam Double deriving (Eq, Show)
5
6 circleArea :: Diam -> String
7 circleArea (MkDiam n) = if n > 0 then show ((n/2) * pi)
8                                  else if n == 0 then "zero"
9                                  else "negative"
10
11 instance Arbitrary Diam where
12   arbitrary = do
13     x <- choose (-10000.0, 10000.0)
14     return (MkDiam x)
15
16 prop_Invariant :: Diam -> Property
17 prop_Invariant text =
18   circleArea text  === circleArea text
19
20 main :: IO ()
21 main = do
22   verboseCheckWith stdArgs {maxSuccess = 1000} prop_Area
23   putStrLn "done"
```

코드 1 원 넓이를 구하는 문제에 대한 Haskell 코드

11~14번째 줄에서는 **Diam**을 QuickCheck에서 제공하는 형인 **Arbitrary**의 인스턴스로 만들어서 QuickCheck를 사용할 수 있게 하였다. 그리고 입력값을 -10,000~10000 사이의 무작위 값이 올 수 있도록 하였다. 16~18번째 줄에서는 실제로 QuickCheck가 실행되는 곳이다. 마지막으로 20~23번째 줄은 **main** 함수로 입출력값을 1,000번 생성하도록 하였다.

교사가 새로운 문제를 생성할 때에는 먼저 입출력 데이터에 맞게 새로운 형을 생성해야 한다(4번째 줄). 다음으로 이 형을 **Arbitrary**의 인스턴스로 만든 후, 입력값의 범위를 지정해야 한다(11~14번째 줄). 그리고 문제에 대한 코딩을 한 후(6~9번째 줄), QuickCheck를 실행한 후(16~18번째 줄), 마지막으로 main에서 반복 횟수를 설정한다(20~23번째 줄).

4. 시스템 평가

이 절에서는 이 논문에서 제안한 시스템이 얼마나 빠른지에 대한 실험을 수행하였다. 시험에 사용된 코드는 코드 1에서 소개한 코드이다. 이 코드에 대하여 입출력값 생성 개수를 다르게 하면 얼마나 걸리는지 확인하였

다. 이를 위해서 리눅스의 time 명령어를 사용하였다[6]. 표 1은 입출력값을 생성 개수를 변경하였을 때 소요되는 시간을 나타내었다.

표 1 입출력값 생성 개수에 따른 입출력 생성 시간(초)

구 분	10^3	10^4	10^5	10^6
real	0.221	1.465	15.661	160.352
user	0.089	0.053	5.180	49.561
sys	0.067	0.754	7.934	150.372
real ÷ 생성 수 (ms)	0.221	0.147	0.157	16.035

표 1의 항목 중 real의 경우 프로그램 호출부터 종료까지 걸린 시간을 나타낸 것이다. user는 프로그램 자체의 처리 시간이다. sys는 프로그램을 처리하기 위해 OS가 처리를 한 시간이다. 실제 입출력 값을 만들기 위해서는 real 값이 중요하다. 이 값을 기준으로 살펴보면, 모든 항목에서 사람이 만들어내는 것보다 훨씬 빠른 시간 내에 생성해내는 것을 확인할 수 있다. 입출력값을 백만 개 생성하는 경우에는 나머지 세 경우보다 속도가 느려졌지만, 사람보다는 훨씬 빠르게 생성하는 것을 확인할 수 있다.

5. 고 찰

본 논문에서 제안하는 시스템은 입출력 데이터를 만들기 위해서는 Haskell 프로그래밍이 필요하다. 하지만 Haskell 프로그래밍은 어렵게 느껴지는 것이 사실이다. 왜냐하면 우리는 명령형 프로그래밍 스타일에 익숙해져 있지만, Haskell은 함수형 프로그래밍 스타일로 프로그래밍해야 하기 때문이다. 하지만 강력한 형 시스템을 지원하는 언어가 바로 Haskell이기 때문에 과제 채점에 있어서 적합한 언어라고 생각한다. 따라서, Haskell 코드보다 익숙한 언어로 코딩할 수 있도록 새로운 도메인 특화 언어(domain-specific language)를 개발할 필요가 있다.

한편 프로그래밍을 통해 원하는 입출력값을 생성하기 어려운 경우가 있다. 앞에서 소개한 원의 지름을 통해 원의 넓이를 구하는 문제는 단순한 입력값만 들어가므로 문제가 되지 않는다. 하지만 회문(palindrome)과 같이 특별한 입력값이 필요한 경우에는 정답을 생성하기 위해 많은 노력이 필요하다. 이러한 특수한 패턴을 생성하기 위해서는 정규식을 이용해야 하며, QuickCheck에서도 이 기능을 지원한다. 그리고 정답 집합과 오답 집합 비율을 동등하게 하는 등 여러 상황이 존재하는데, 이를 해결하기 위한 연구도 필요하다.

6. 결 론

이 논문에서는 코드 평가 시스템인 neoESPA의 입출력 데이터를 자동으로 생성하기 위한 시스템을 제안하였다. 기존 neoESPA에서는 입력 데이터를 만들기 위해 교사가 문제에 대한 코드를 만들고, 여기에 입력 데이터를 수동으로 넣어서 출력 데이터를 생성해낸다. 이러한 방법은 시간이 많이 걸리고, 프로그램의 버그로 인한 잘못된 입출력 데이터가 생성될 가능성이 있는 문제가 있다.

우리는 Haskell의 QuickCheck 라이브러리를 사용하여 입출력 데이터를 자동으로 생성하게 하였다. 교사는 Haskell을 이용해서 과제 프로그램을 만들고, 입력 데이터에 대한 자세한 스펙을 지정한다. 이후, 입출력 데이터 개수를 지정하고 프로그램을 실행하면 스펙에 맞는 입출력 데이터가 자동으로 생성된다. 생성 속도를 확인하기 위해 입출력 데이터 개수를 늘려가면서 실험하였는데, 백만 개를 생성하여도 3분이 채 걸리지 않는 시간 내에 완료되는 것을 확인하였다. 하지만 복잡한 형태의 입출력 값이 필요한 경우에는 고도의 Haskell 프로그래밍이 필요한 문제가 있다.

추후에는 일반적인 프로그래머가 익숙한 명령형 프로그래밍 스타일의 프로그래밍을 지원하도록 연구할 예정이다. 익숙하지 않은 Haskell을 이용해서 복잡한 입출력을 만들기는 상당히 어렵기 때문에 여기에 드는 부하를 줄이기 위해서 다른 도메인 특화 언어를 연구하고자 한다.

ACKNOWLEDGEMENT

*이 논문은 2021년도 정부(과학기술정보통신부)의 재원으로 정보통신기획평가원의 지원을 받아 수행된 연구임 (No.2014-3-00035, 매니코어 기반 초고성능 스케일러블 OS 기초연구 (차세대 OS 기초연구센터)).
*교신 저자: 우균(부산대학교, woogyun@pusan.ac.kr).

참고문헌

[1] Google, 코로나바이러스(코로나19) [Online]. Available: https://news.google.com/covid19/map?hl=ko&gl=KR&ceid=KR%3Ako (downloaded 2021, May 7)

[2] Xiao Liu, Yeoneo Kim, Hwan-Gue Cho, and Gyun Woo, “NeoESPA - A New Evaluation System for Programming Assignments,” Proceedings of the 14th International Conference on Electronics, Information and Communication (ICEIC 2015), 2015.

[3] haskell.org, Haskell Language [Online]. Available: https://www.haskell.org/ (downloaded 2021, May 7)

[4] Koen Claessen and John Hughes, “QuickCheck: a lightweight tool for random testing of Haskell programs,” Proceedings of the fifth ACM SIGPLAN international conference on Functional programming, pp. 268-279, 2000.

[5] hackage, Test.QuickCheck.Property [Online]. Available: https://hackage.haskell.org/package/QuickCheck-2.11.3/docs/Test-QuickCheck-Property.html (downloaded 2021, May 7)

[6] man7.org, time(1) — Linux manual page [Online]. Available: https://man7.org/linux/man-pages/man1/time.1.html (downloaded 2021, May 7)