FlowX: 손쉬운 데이터 전처리를 위한 흐름 기반 데이터 전처리 프로그램

박지원*, 예창언**, 이우령***, 이현서****

FlowX: Data flow-based GUI editing program for easy data preprocessing

Park Jiwon*, Yea Changen**, Lee Woolyung***, Lee Hyunseo****

요 약

기존 데이터 전처리를 위한 프로그래밍 언어와 도구들, 일반인들을 대상으로 하는 서비스들은 각각 진입 장벽과 사용성 측면에서의 문제점이 존재한다. 이에 따라 본 논문에서는 데이터 전처리를 위한 FlowX 프로그램을 제안한다. 이 프로그램은 흐름 기반으로 직관적인 그래픽 인터페이스를 제공하며 예외 처리와 타입 시스템 측면에서의 runtime safe를 보장한다.

Abstract

In the field of data preprocessing, there are barriers to entry with existing programming languages and tools. Additionally, services targeting to non-experts encounter usability issues. So, in this paper, we propose 'FlowX,' a program for data preprocessing that is flow-based, provides an intuitive graphical interface, and guarantees runtime safety in terms of exception handling and the type system.

Key words

data preprocessing, visual programming, graph programming, runtime safe, error handling

Ⅰ. 서 론

본 논문은 데이터 과학의 발전과 함께 고품질의 모델, 정확도 높은 분석을 위해 데이터 전처리가 중요한 역할을 하게 된 시대적 배경을 바탕으로 한다. 기존의 전문가들이 사용하는 Python, R과 같은 프로그래밍 언어들과 그 언어들의 데이터 전처리 라이브러리들은 일반인들이 사용하기에 그 진입 장벽이 높다. 또한 일반인들의 데이터 전처리를 목적으로 하는 EUD(End User Development) 기반의 태블로나 Trifacta는 예외 처리나 타입 시스템이 약하기에 runtime 오류가 발생할 가능성이 높다는 문제점이 있다. 이에 따라 본 논문에서는 최종 사용자를 대상으로 예외와 타입 시스템에서 runtime safe한 언어를 포함하는 흐름 기반 데이터 전처리 프로그램을 제안한다. 이 논문은 데이터 처리 분야에서 사용성을 고려한 데이터 전처리 프로그램의 중요성을 강조하며, 해당 분야의 발전에 기여할 수 있다

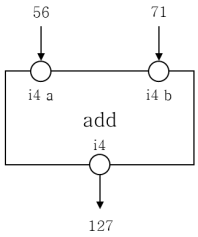
*경북대학교 학부생, raipendalk@gmail.com, **경북대학교 학부생, nsce9806q@gmail.com, ***경북대학교 학부생, vlfkdydtk@gmail.com, **** 경북대학교 학부생, heart20021010@gmail.com

는 의의를 지닌다.

Ⅱ. 프로그램 설계 목적

흐름 기반의 비주얼 프로그래밍

FlowX는 흐름 기반 비주얼 프로그래밍을 다룬다. 이는 텍스트 기반의 프로그래밍과 비교했을 때에 쉽고 직관적인 모델링을 가능케 하며, EUD의 측면에서 장점이 된다. [1]



<그림 1> 두 수를 더하는
FlowX의 기본 함수

<그림 1>은 FlowX의 기본 함수의 사용을 나타낸다. 그래프의 노드는 함수를 의미하며 방향 간선은 그 함수의 입력과 출력을 의미한다. 따라서, 프로그램은 함수와 함수의 연결로 구성되어 그래프 형태로 나타나 사용자가 데이터 흐름을 쉽게 파악하게 설계할 수 있다. [2] 여기서 그래프는 DAG (Directed Acyclic Graph)를 유지해야 하며 반복과 재귀 호출을 제공하지 않는다. 재귀호출과 반복문은 사용자가 데이터 흐름을 파악하기 어렵고 오류가 발생하기 쉽기 때문이다. [3]

병렬 처리 확장 가능성

FlowX 프로그램은 입력 데이터 전체에 대한 연산을 하나의 행에 대한 연산으로 추상화하여 적용한다. 이런 접근 방식은 스프레드시트에서 행 단위에 수식을 작성하면 모든 행에 그 연산이 적용되는 방식과 유사한데, 이는 데이터 전처리 과정을 직관적으로 표현할 수 있다는 장점을 지닌다. FlowX의 함수 노드가 부수 효과(side effect)가 없는 순수 함수 형태로 설계되었음에 따라 이러한 추상화된 행 연산은 멀티스레드 환경에서 발생할 수 있는 race condition과 deadlock과 같은 문제를 방지하면서, 병렬 처리의 이점을 최대화할 수 있다.[4]

병렬 처리가 불가능한 부분에서는 'Thread Joining' 기법을 활용해 데이터 전처리의 성능을 극대화할 수 있다. 파일로부터의 데이터 입력과 데이터 출력, 혹은 평균값 계산과 같은 연산은 순차적으로 처리되거나 내부적인 병렬 처리 기법으로 처리된다. 읽어온 데이터들은 순수 함수들의 조합으로 병렬적으로 처리되어 데이터 전처리를 효율적으로 완료하게 된다.

자료형과 에러처리

각 함수 노드의 입력과 출력은 컴파일 타임에 그 자료형이 결정되며, 제네릭을 사용하는 경우 클래스를 필수적으로 사용해야 하기에 타입 시스템 측면에서 안전하다.

FlowX은 에러 발생 가능성이 있는 함수에 대해선 그 에러를 처리하는 핸들러가 바로 뒤에 위치하도록 강제한다. 이는 에러의 발생 가능성을 두 함수 이상 거치도록 하지 않게 함으로써 신뢰성 높은 데이터 전처리 모델을 구축할 수 있게 된다.

Ⅲ. 프로그램 명세

3.1. 자료형

기본 자료형과 사용자 정의 자료형(구조체)으로 나뉜다.

3.1.1. 기본 자료형

1. 값의 종류와 바이트 수의 순서쌍, 혹은 함수의 원형이다.

2. 정수 자료형은 4, 8바이트의 부호가 있는 정수이다. (i4, i8)

3. 실수 자료형은 4, 8바이트의 IEEE 754 부동소수점 방식의 실수이다. (f4, f8)

4. 문자열 자료형은 $2^n (1 \leq n \leq 16)$ 바이트의 유니코드 문자열이다. (s2, … , s65536)

5. 불리언 자료형은 true, false를 나타내는 자료형이다. (b1)

3.1.2 구조체

1. 구조체를 이루는 멤버들의 자료형과 이름으로 구성된다.

2. 구조체가 중첩된다면 그 중첩 관계가 순환되어선 안 된다.

3. 구조체의 멤버는 errorable이 아니어야 한다.

<그림 2>는 구조체를 나타내는 몇 가지 예시들이다.

vec3	
f4	x
f4	y
f4	z

mat3	
vec3	a
vec3	b
vec3	c

person	
s64	name
i4	age
f4	weight
f4	height

<그림 2> 구조체를 나타내는 예시들

3.1.3. nullable과 errorable

1. 모든 자료형은 nullable과 errorable의 형태가 있다.

2. nullable은 값이 없을 수 있는 상태를 의미하며 T?로 표기한다.

3. errorable은 오류가 발생했을 수 있는 상태를 의미하며 T!로 표기한다.

4. nullable하며 errorable한 자료형(T?!)은 존재하지만 그 외의 중첩된 자료형은 존재하지 않는다.

3.2. 클래스

특정 형태의 함수를 만족하는 자료형들의 집합이다.

3.2.1 제네릭 함수를 제한하는 클래스

1. 제네릭 함수를 정의할 때 입력의 자료형을 하나 이상의 클래스로 지정해야 한다.

2. 제네릭 함수에 사용할 수 있는 자료형은 그 클래스의 교집합의 원소로 제한된다.

<그림 3>은 제네릭 함수를 제한하는 클래스의 예시이다.

```
addable = { t ¦ add(t, t):t ∈ F }
comparable = { t ¦ t ∈ T, x(t, t):bool ∈ F,
                    x ∈ {<, >, ==} }
F는 함수 집합, T는 자료형 집합
```

<그림 3> 제네릭 함수를 제한하는 클래스

3.3. 함수

1. 기본 함수와 사용자 정의 함수로 나뉜다.

2. 함수는 0개 이상의 입력과 1개 이상의 출력을 가진다.

3.3.1. 기본 함수

다음은 FlowX에서 제공하는 몇 가지 기본 함수들이다.

$$\mathrm{errorToValue}(T!\,x, T\,v) : T = \begin{cases} x : x\,\mathrm{isn't\,error} \\ v : \mathrm{else} \end{cases}$$

$$\mathrm{nullToValue}(T?\,x, T\,v) : T = \begin{cases} x : x\,\mathrm{isn't\,null} \\ v : \mathrm{else} \end{cases}$$

$$\mathrm{panic}(T!\,x) : T = \begin{cases} x & : x\,\mathrm{isn't\,error} \\ \mathrm{runtime\,error} & : \mathrm{else} \end{cases}$$

$$\mathrm{member}(T : \mathrm{structure}\,x) : (T\mathrm{'s\,members}) = (x\mathrm{'s\,members})$$

$$\mathrm{struct}(T\mathrm{'s\,members}) : T = T(T\mathrm{'s\,members})$$

$$\mathrm{assert}(T\,x, \mathrm{f}(T) : \mathrm{bool}) : T! = \begin{cases} x & : \mathrm{f}(x)\,\mathrm{is\,true} \\ \mathrm{error} & : \mathrm{else} \end{cases}$$

3.3.2. 사용자 정의 함수

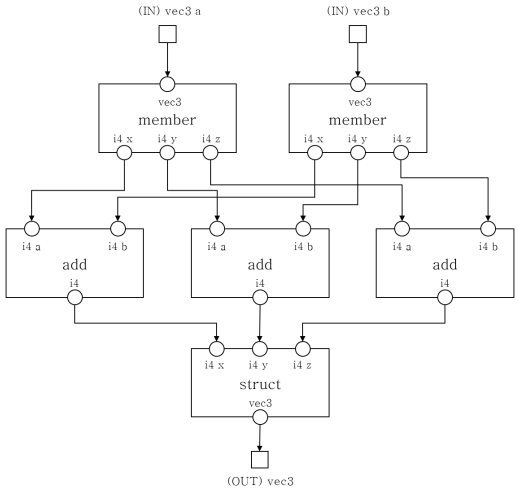
1. Directed Acyclic Graph(DAG)로 표현된다.

2. 그래프의 정점은 그 함수를 이루는 함수 또는 함수의 입력이나 출력이다.

3. 간선은 함수의 입력과 출력의 연결이다.

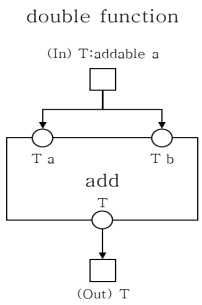
4. 만약 한 함수가 다른 함수를 호출할 가능성이 있다면 그 반대의 호출은 불가하다.

5. 모든 사용자 정의 함수의 입력은 errorable이어선 안 된다.

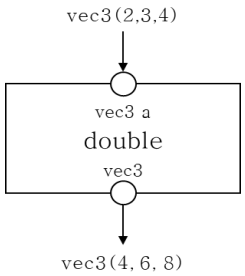


<그림 4> vec3가 addable 클래스가 되기 위한 사용자 정의 add 함수

<그림 4>는 vec3가 addable이기 위해 필요한 vec3의 add 사용자 정의 함수를 나타낸다. <그림 5-1>과 <그림 5-2>는 각각 사용자 정의 함수인 double 함수를 정의하고 사용하는 예시이다. vec3는 addable이므로 double(vec3 a): vec3 = add(a, a)가 정의된다.



<그림 5-1> 사용자 정의 함수 정의 예시



<그림 5-2> 사용자 정의 함수 사용 예시

3.3.3. 프로그램(메인 함수)

1. 프로그램은 하나만 존재할 수 있는 함수로 FlowX 프로그램의 시작과 끝을 의미한다.

2. 프로그램의 입력과 출력은 파일이다. 이 파일은 데이터를 구분할 수 있는 CSV, TSV, JSON 등의 파일이어야 한다.

3. 다른 함수들은 프로그램의 연산을 위한 과정으로 활용된다.

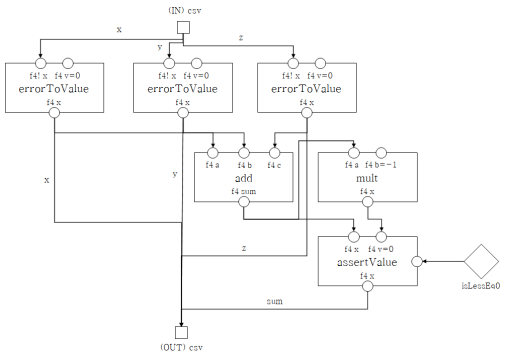
Ⅳ. 평가

데이터 처리 과정의 직관성

```
import pandas as pd

def proc(value):
    try:
        return float(value)
    except:
        return 0
def proc2(value):
    if value < 0:
        return -value
    return value
df = pd.read_csv('example.csv')
for x in ('x', 'y', 'z'):
    df[x] = df[x].apply(proc)
df['sum'] = df['x'] + df['y'] + df['z']
df['sum'] = df['sum'].apply(proc2)
df.to_csv('result.csv')
```

<프로그램 6-1> Python으로 작성된 데이터 전처리 프로그램의 예시



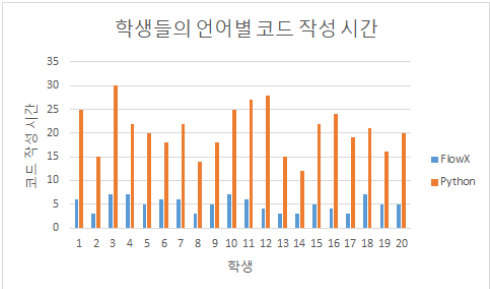
<그림 6-2> FlowX로 작성한 프로그램 7-1의 코드

동일한 기능을 하는 Python 프로그램 <프로그램 6-1>과 FlowX 프로그램 <그림 6-2>을 비교하면, 데이터를 불러오고 반복문을 통해 각 행의 데이터를 처리하고 저장하는 모든 과정을 코드로 작성해야 하는 Python 프로그램보다 FlowX 프로그램은 필요한 각 데이터에 집중할 수 있다는 장점이 있다.

프로그램 작성의 편리함

> [이름, 전화번호, 성별, 나이] 데이터가 순서대로 작성된 CSV 파일에서 모든 데이터가 정상적으로 입력된(각 항목의 자료형이 일치하고, 전화번호 양식에 맞게 입력되었으며, 나이가 양수이고, 성별이 “F” 또는 “M”인) 데이터만 추출하여 새로운 CSV 파일에 저장하라.

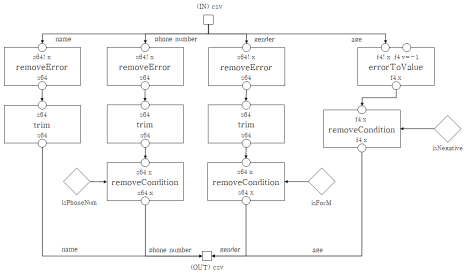
위와 같은 Task를 수행하는 프로그램을 경북대학교 컴퓨터학부 학부생 20명이 FlowX와 Python 각 언어로 작성하였을 때, 코드 작성에 걸린 시간은 <그림 7>과 같다.



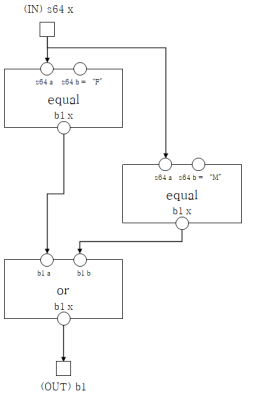
<그림 7> 언어별 동일 프로그램 작성 시간

Python은 평균 약 20분이 소요됐지만, FlowX로 작성한 경우는 평균 5분이 소요될 정도로 데이터 전처리 분야에서의 생산성이 뛰어남을 알 수 있다.

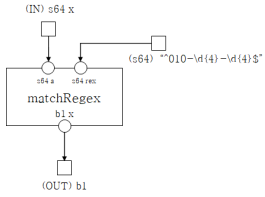
위 Task를 수행하는 FlowX 코드인 <그림 8-1>은 프로그램(메인 함수)이고, <그림 8-2>와 <그림 8-3>은 프로그램에서 사용하는 사용자 정의 함수이다.



<그림 8-1> 프로그램(메인 함수)



<그림 8-2>
isForM 함수



<그림 8-3>
isPhoneNumber 함수

V. 결론 및 향후 연구 방향

FlowX는 데이터 과학 분야에서 비전문가들이 쉽게 데이터 전처리를 수행할 수 있도록 하는 직관적인 흐름 기반 데이터 전처리 프로그램이다. FlowX는 컴파일 타임에 발생 가능한 모든 오류를 검사하도록 하여 runtime safety를 보장하며, GUI 기반의 플로우 시스템을 제공하여 사용자 친화적인 환경을 제공한다. 이러한 특징들은 비전문가들이 데이터 전처리 작업을 쉽게 수행할 수 있도록 돕는다. 이 논문은 FlowX의 설계를 통해, 전문가가 아닌 사람들도 데이터 처리를 쉽게 할 수 있는 프로그램의 필요성을 강조하며, 이러한 프로그램이 데이터 처리 분야의 발전에 기여할 수 있음을 보여준다.

FlowX와 같은 그래프 기반의 프로그래밍은 폰 노이만 구조의 컴퓨터에서 성능과 구현의 문제점이 존재한다. 따라서 효율적인 처리를 위한 그래프 시뮬레이팅과 전처리 모델 최적화 측면에서의 후속 연구가 필요하다.

사사

“본 연구는 과학기술정보통신부 및 정보통신기획평가원의 SW중심대학사업의 연구결과로 수행되었음“ (2021-0-01082)

참 고 문 헌

[1] Wallace, B. C., Dahabreh, I. J., Trikalinos, T. A., Lau, J., Trow, P., & Schmid, C. H. (2012). Closing the Gap between Methodologists and End-Users: R as a Computational Back-End. Journal of Statistical Software, 49(5), 1-15. https://doi.org/10.18637/jss.v049.i05

[2] 남영호, 김성조.(1994).함수형 계산 모델을 위한 그래프 프로그래밍 환경.(구)정보과학회논문지,21(1),182-195.

[3] 최현종.(2016).순서도를 활용한 프로그래밍 제어 구조 학습에 나타난 오류 유형 분석.컴퓨터교육학회 논문지,19(1),101-109.

[4] 김연어, 변석우, 우균.(2017).Haskell을 이용한 병렬 프로그래밍.정보과학회지,35(3),22-29.