Документация

По проект „Симулатор на вселената“ към курс “Функционално програмиране” зимен семестър 2021/2022 г.

*Изготвил: Виктор Василев*

*Спец. Информатика*

*3-ти курс, група 3, ф.н. 45676*

1. **Кратко описание на решението**

Проекта е реализиран на Haskell.

Казано накратко, целта е да се генерират редици от елементарни частици като на всеки такт, някои от тях могат да се разпаднат недетерминистично по няколко начина. Поради нуждата да се спазят точните проценти шанс една частица да се разпадне (и на какви частици) използвам библиотека за генериране на произволни вариации за дискретни разпределения.

1. **Кратък преглед на структурата на проекта, основни функции и начин на използването им**

Проекта се състои от 2 файла – Particle.hs и Main.hs.

В Particle.hs е описан типът данни Particle.

Той се състои от:

chance – число с плаваща запетая от 0 до 1 – вероятността да възникне разпад на частицата, symbol – низ – текстовото представяне на частицата, decay – списък от списъци от частици – всички възможни начини по които тази частица може да се разпадне на други частици, weights – вектор от числа с плаваща запетая - съответните проценти шанс за всеки възможен разпад.

Дефинирани са и съответните селектори за горе споменатите данни на типа, както и всички частици следвайки условието.

Функциите decide и decideFrom използват функции от библиотеката mwc-random който използва алгоритъма на Marsaglia за генериране на псевдослучайни числа. Функцията decide връща число на базата на бернулиево разпределение, а decideFrom

приема вектор от стойности (проценти) за категорично разпределение и на тяхна база връща число индекс. Тъй като това е типичен пример за програмиране със странични ефекти, повечето функции връщат IO типове.

Функцията getDecay приема частица p и връща частиците на които се е разпаднала в този такт.

Функцията decideDecay извършва или този разпад на частицита или я остава същата, на база на булева стойност.

Функцията particleDecay извиква decideDecay със стойността върната от decide.

Накратко, процесът е следният:

На всеки ход за всяка частица в списъка се прави разпад (getAllParticleDecay), така този списък вече е списък от списъци, след което сливаме списъците в един списък чрез функцията foldParticleLists.

1. **Кратки инструкции за работа**

Main.hs импортва Particles.hs.

За да се наблюдава симулацията извикваме функцията start или startWith ако искаме да стартираме със собствени условия.

Използвани материали: 2021 Jared Tobin - Randomness in Haskell

https://jtobin.io/randomness-in-haskell?fbclid=IwAR0ghLEf4pEGq1VKw53bLAy0HdEsA7NyLMfMVVua\_KfXM1By75hc275GLbA