Симулатор на Вселената

Документация на Проект по Функционално Програмиране

Автор: Ася Русанова
ФН: 45399, Група 1
Специалност: Информатика

Глава 1. Увод

1.1. Описание и идея на проекта

Настоящият проект цели да представи примерна реализация на недетерминистичен разпад на частицата Хигс Бозон (*англ. пр. Higgs Boson*). Задачата в последствие е сведена до реализация на недетерминистичен разпад на крайно множество от частици, въведени от потребителя посредством интерфейса на програмата. Проектът се възползва от предоставените в условието на задачата процентни съотношения за вероятността за разпад на частиците.

1.2. Език за програмиране и използвани библиотеки

Избраният език е Хаскел. Програмата се състои от 4 файла - *Particle.hs* (базова имплементация на частиците. В него са дефинирани всички частици - стабилни и нестабилни), *ParticleDecay.hs* (съдържащ функциите за разпад на всяка една от нестабилните частици. Спазени са дадените вероятности.), *Simulator.hs* (съдържащ функциите за генериране на произволни числа и основния симулатор на разпад) и *Main.hs* (включва възможност за вход от потребителя на началното състояние на вселената ни).
За целите на проекта е включен пакетът *random*, а по-конкретно библиотеката *System.Random*, чрез които са създадени методи за генериране на числа с плаваща запетая - похват използван за илюстриране на недетерминизма на разпада.

Глава 2. Дефиниране на проблеми и избрани решения

1.1. Вероятности за разпад и тяхното представяне

Наличието на различна точност за разпад на частиците в условието на задачата е малък, но същевременно и важен проблем. Тъй като предоставените проценти са с точност до максимум четири знака след запетаята, именно тази точност бе подадена на функциите за разпад на конкретна частица, както и на тези за генериране на произволен процент.

Вероятностите за разпад на дадена частица са съвместно събрани в интервала от *0%* до *100%*. Интервалът *0%-x%* отговаря на първата посочена вероятност в условието. За получаване на следващият интервал се сумира края на предходния интервал със следващата посочена вероятност. Общо вероятностите се сумират до *100%*, което съответства на *100.0000* от тип *Double* в настоящата имплементация.

1.2. Генериране на произволни проценти

Съществена трудност при осъществяване на проекта е сблъсъкът с генераторите на произволни проценти. При подаване на един и същ генератор, числата, които се създават са еднакви. Оттук възниква и нуждата за подаване на нови генератори при многократното извикване на дадена функция. Връщаната стойност на вградената

функция `randomR` е *(a, g)*, където *g* е нов генератор. За решение на проблема с генераторите новият генератор, получен от функцията `RandomR`, се подава като генератор при следващото извикване на даден метод. За улеснение и добиване на известно ниво на абстракция е създаден типът `Random' a`, който приема генератор от тип `StdGen` и връща двойка `(a, StdGen)`.
При разпад на дадена частица се генерират два произволни процента, а именно - първи процент, по който съдим дали дадена частица ще се разпадне или не, и втори процент, спрямо който (при наличие на благоприятен резултат за първия процент) разпадаме частицата на две нови частици. Поради наличието на тази предпоставка, е избран подход, при който се генерира двойка различни проценти, а заедно с нея се връща и нов генератор. Функцията, която онагледява този подход е `randTuple :: Random' (Double, Double)`.

Глава 3. Структура на Импелментацията

3.1. Имплементация на частица

С цел да се онагледи ключовата характеристика стабилност на дадена частица е дефинирано:

```
data Stability = Stable | Unstable Double
```

Ако дадена частица не е стабилна, вероятността ѝ за разпад е представена като число с плаваща запетая с точност до четвъртия знак след запетаята.

Една частица съответно е представена с двете си основни характеристики: име и стабилност:

```
data Particle = Particle {name :: String, stable :: Stability}
```

Тъй като програмата извежда частиците от един списък във формата: “<брой срещания> x <име на частица>” е създадена функцията:

```
tellcurrentParticles :: [Particle] -> String
```

Ако конкретна частица се среща повече от веднъж, функцията прави нужните преобразувания за извеждане на една частица без повторения.

3.1. Разпад на частици

Разпадът на частиците е дефиниран посредством две функции: `decayOneParticle`, която симулира разпад на една единствена частица и `decayOnceList` - рекурсивна функция, която прави опит за едновременен разпад на всички частици от списъка. При нея, за всяка отделна частица от списъка се извиква функцията `decayOneParticle` и в зависимост от генерираната вероятност, която за всяка частица е различна, разпадът ѝ се или не се осъществява. Функцията връща също и следващ генератор, който улеснява многократното ѝ извикване.

Доказателство на твърдението, че за всяка една частица от списъка генерираната вероятност е различна наблюдаваме на Фиг. 1. Списъкът се състои от един W Бозон и един Z Бозон. Вероятността за разпад на двете частици е еднаква (50%) и въпреки това след първия опит за разпад на списъка, наблюдаваме, че само едната нестабилна частица (W Бозон) се е разпаднала, а другата не.

```
*Simulator> let d = decayOnceList [w_boson, z_boson] (mkStdGen 2)
*Simulator> tellcurrentParticles (fst d)
"1x Antitau Lepton, 1x Neutrino, 1x Z Boson."
```

Фиг. 1 : Генериране на различни проценти за разпад на частиците

Поради преценката, че точният момент на разпад на дадена частица предизвиква интерес за проучващия поведението ѝ, симулаторът принтира всяко едно от състоянията на разпад по следния начин: “<номер на момент>. <състав на списъка в конкретния момент>” дори разпад в съответния момент да не се е осъществил. Известен е фактът, че ниската вероятност за разпад на Хигс Бозона може да доведе до принтиране на голямо количество редове. Въпреки това гореописаната предпоставка надделя при избора. При всяко едно извикване на функцията `simulator` допълнително се създава нов начален за процеса генератор чрез `newStdGen`. Следващите генератори се създават посредством `decayOnceList`.

Фиг. 2 показва нагледно процеса на извеждане на частиците чрез функцията `universeSimulator`, която обвива `simulator`, задавайки номера на началния момент (1.). Обърнато е внимание на факта, че повторното изпълнение на същата функция довежда до различен резултат, което доказва недетерминизма на програмата.

```
*Simulator> universeSimulator [w_boson, tau_lepton, z_boson, neutrino]
1. 1x W Boson, 1x Tau Lepton, 1x Z Boson, 1x Neutrino.
2. 1x Positron, 2x Neutrino, 1x Tau Lepton, 1x Down Quark, 1x Down Antiquark.
*Simulator> universeSimulator [w_boson, tau_lepton, z_boson, neutrino]
1. 1x W Boson, 1x Tau Lepton, 1x Z Boson, 1x Neutrino.
2. 1x W Boson, 1x Tau Lepton, 1x Bottom Quark, 1x Bottom Antiquark, 1x Neutrino.
3. 1x Positron, 2x Neutrino, 1x Tau Lepton, 1x Bottom Quark, 1x Bottom Antiquark.
```

Фиг. 2: Финален симулатор на разпад.

Глава 4. Въвеждане на първоначално състояние от потребителя

Създадена е функция за съставяне на начален списък от частици от самият потребител. Ако потребителят въведе непозната дума за частица или индикатор за прекъсване на въвеждането, се приема, че началният списък е създаден. С този списък се извиква функцията `universeSimulator`.
Пример за това може да се наблюдава на Фиг. 3. и Фиг. 4.

```
*Main> main
Enter Particle:
W Boson
Enter Particle:
Neutrino
Enter Particle:
Z Boson
Enter Particle:

1. 1x W Boson, 1x Neutrino, 1x Z Boson.
2. 1x W Boson, 1x Neutrino, 1x Z Boson.
3. 1x W Boson, 1x Neutrino, 1x Z Boson.
4. 1x Positron, 2x Neutrino, 1x Z Boson.
5. 1x Positron, 2x Neutrino, 1x Muon, 1x Antimuon.
*Main>
```

Фиг. 3: Въвеждане от потребителя

```
Enter Particle:
Higgs Boson
Enter Particle:
W Boson
Enter Particle:

1. 1x Higgs Boson, 1x W Boson.
2. 1x Higgs Boson, 1x Antimuon, 1x Neutrino.
3. 1x Higgs Boson, 1x Antimuon, 1x Neutrino.
4. 1x Higgs Boson, 1x Antimuon, 1x Neutrino.
5. 1x Higgs Boson, 1x Antimuon, 1x Neutrino.
198. 1x Higgs Boson, 1x Antimuon, 1x Neutrino.
199. 1x Higgs Boson, 1x Antimuon, 1x Neutrino.
200. 1x Higgs Boson, 1x Antimuon, 1x Neutrino.
201. 1x Higgs Boson, 1x Antimuon, 1x Neutrino.
202. 1x Higgs Boson, 1x Antimuon, 1x Neutrino.
203. 1x Higgs Boson, 1x Antimuon, 1x Neutrino.
204. 1x Higgs Boson, 1x Antimuon, 1x Neutrino.
205. 1x Charm Quark, 1x Charm Antiquark, 1x Antimuon, 1x Neutrino.
```

(…)

Фиг. 4: Въведени от потребителя Хигс Бозон и W Бозон. Разпад на Хигс Бозон в момент 205.

Използвана Литература

(1) http://learnyouahaskell.com/making-our-own-types-and-typeclasses
(2) https://hackage.haskell.org/package/random-1.1/docs/System-Random.html
(3) https://www.reddit.com/r/haskell/comments/5mx8av/saving_user_input_to_array_in_io/
(4) https://www.schoolofhaskell.com/school/starting-with-haskell/libraries-and-frameworks/randoms