Летняя школа по параллельному программированию 2016

Разработка интерпретатора языка LuNA для веб-фреймворка

Последовательная исполняющая система

Выполнил: Ажбаков Артем, ФИТ НГУ

Руководитель: Перепелкин В.А.

15.07.2016

Введение: LuNA

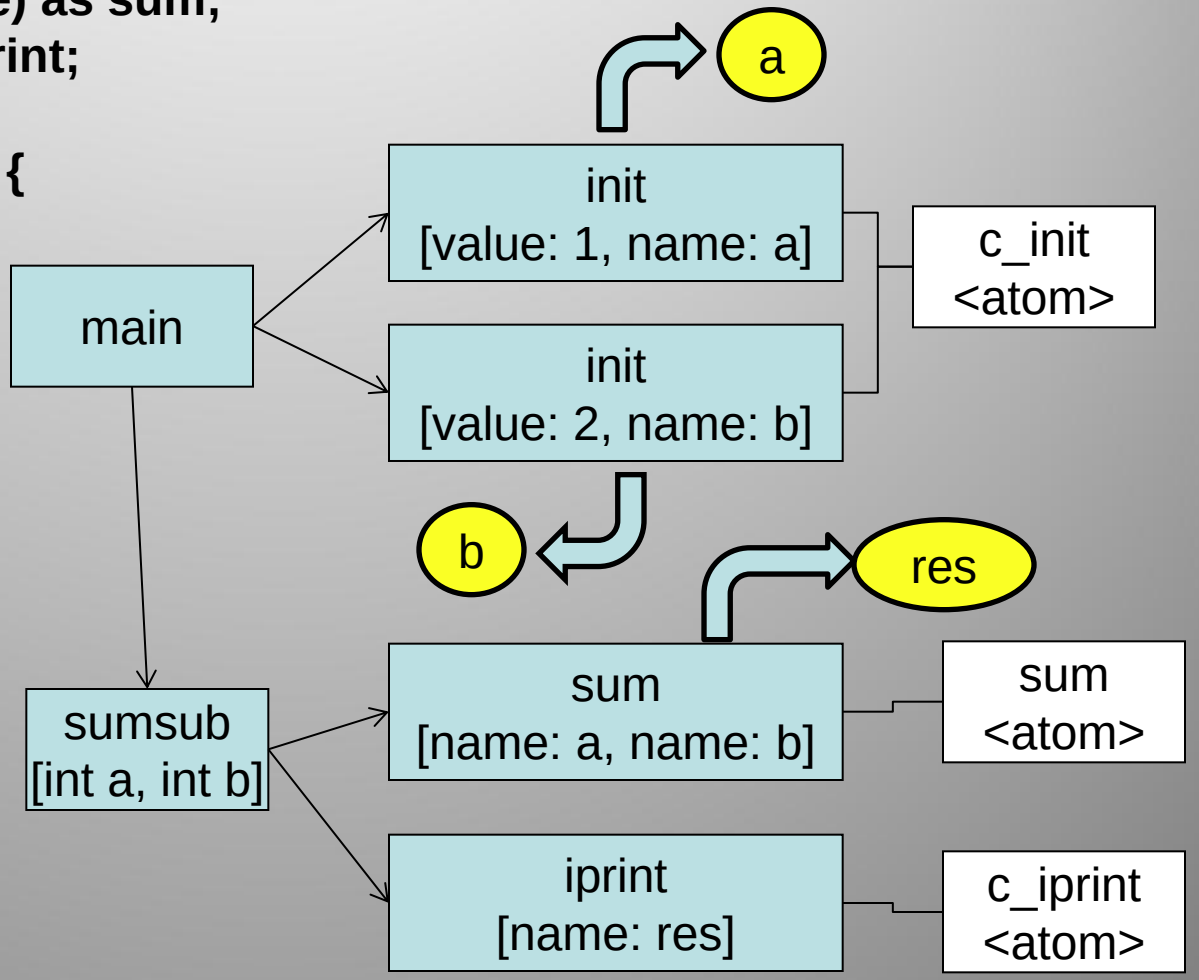
- LuNA – система автоматизации параллельного программирования
- LuNA – система фрагментированного программирования
- Компоненты LuNA - язык, компилятор, система исполнения

Введение : программа LuNA

```
import c_init(int, name) as init;
import sum(int, int, name) as sum;
import c_iprint(int) as iprint;

sub sumsub (int a, int b) {
    df res;
    sum (a, b, res);
    iprint(res);
}

sub main()
{
    df a, b;
    init(1, a);
    init(2, b);
    sumsub(a, b);
}
```



Проблема

- Отсутствует возможность наблюдать за тем, как может происходить параллельное исполнение, хочется видеть, какие задачи могут исполняться независимо и когда.
- Шаг к решению – интерпретатор, пошаговая система исполнения.

Внутреннее представление программы

```
"iprint": {
   "type": "extern",
   "code": "c_iprint",
   "args":{
            "type": "int"
   }
}
```

```
"init": {
   "type": "extern",
   "code": "c_init",
   "args": {
     {"type": "int“},
     {"type": "name“}
   }
}
```

```
"sum": {
      "type": "extern",
      "code": "sum",
      "args": {
        {"type": "int"},
        {"type": "int"},
        {"type": "name"}
      }
  }
```

```
"main": {
 "body":
  {
  "code": "init",
  "args": [
    {"type": "iconst", "value": 1},
    {"ref": ["a"], "type": "id"}
   ],
   "type": "exec",
  },
  {
     …..
  }
}
```

Псевдокод

```
main = convertCfToTask;
while (hasNextTasks(main))  {
        time++;
        tick (main);
        printTask (main);
}
```

```
tick (task, passed_args) {
        acceptPassedArgs (passed)
        if (!task.nextTasksInstantiated) {
                instantiateNextTasks(task);
        }
        foreach (next_task in task.next_tasks) {
                if (next_task.type == atom) {
                        callUserFunction (getArgs(task, next_task));
                        collectResults();
                        removeTask(nextTask);
                } else {
                        tick (next_task, getArgs(task, next_task));
                }
        }
}
```

Пример вывода

- **Task: main**
  - **has tasks:**

• **init** – **depends on:** » **1 type of iconst, is output: false** » **a[1] type of id, is output: true** – **all args ready!**
• **init** – **depends on:** » **1 type of iconst, is output: false** » **a[2] type of id, is output: true** – **all args ready!**
• **sumsub** – **depends on:** » **a[1] type of id, is output: false** » **a[2] type of id, is output: false**

Пример вывода

Task: main

- ready:

a[1] == 1

a[2] == 2

- has tasks:

Task: sumsub
depends on:
a[1] type of id, is output: false
a[2] type of id, is output: false
has tasks:

Task: sum
depends on:
a type of id, is output: false
b type of id, is output: false
res type of id, is output: true

Task: iprint
depends on:
res type of id, is output: false

Результат

Поддерживается:

- Пошаговое исполнение программы
- Печать состояния
- Передача в качестве аргументов целочисленных констант и фрагментов данных (нет строковых констант и др.)
- Рекурсия
- Цикл for
- Лог

Не поддерживается:

- Цикл while, конструкция if и др. конструкции языка
- Логирование некоторых событий
- Строковые константы и др. типы данных

Заключение

- Таким образом, данный инструмент подходит для пошагового исполнения простых программ среды LuNA и, в сочетании с визуализатором, облегчает понимание работы программы.
- Дальнейшее развитие подразумевает собственно управление ходом исполнения программы, поддержку виртуальных исполняющих узлов, поддержку недостающих конструкций языка.

Спасибо за внимание

Ажбаков Артем, ФИТ НГУ