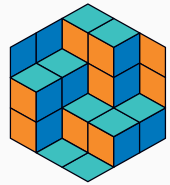

Арифметические операции в TSG, их обращение

Оля Шиманская, 3 курс СП

11 июня 2021 г.

Реализация операций + и *

Сложение

- Вход программы: два числа в двоичной системе счисления Например:**CONS 1 (CONS 0 (ATOM ""))**
- Алгоритм:
  - переворачиваем первое число
  - переворачиваем второе число
  - складываем, запоминая остаток (0 или 1) и складываем столбиком, пока не встретим пустой атом

Пример

plus [C 1 (C 0 empty), C 1 (C 0 (C 1 (C 1 empty)))]

↓ разворачиваем числа и вызываем вспомогательную функцию plusR

plusR [C 0 (C 1 empty), C 1 (C 1 (C 0 (C 1 empty))), 0, empty]

0 - остаток, empty - наш будущий результат, к нему будем прицеплять атомы

$$\begin{array}{r} \mathbf{0} \\ 10 \\ +\ 1011 \\ \hline 1 \end{array} \rightarrow \begin{array}{r} \mathbf{0}0 \\ 10 \\ +\ 1011 \\ \hline 01 \end{array} \rightarrow \begin{array}{r} \mathbf{1}00 \\ 10 \\ +\ 1011 \\ \hline 101 \end{array} \rightarrow \begin{array}{r} \mathbf{0}100 \\ 10 \\ +\ 1011 \\ \hline 1101 \end{array} \rightarrow \begin{array}{r} \mathbf{0}0100 \\ 10 \\ +\ 1011 \\ \hline 1101 \end{array}$$

Обратимость с алгоритмом ura

```
*Main Lib Mult Paths_TSG_arithmetics_simple Plus Util> ura plusProg  ([CVE 1, de
cimalToBinaryExp 2], RESTR []) (decimalToBinaryExp 13)
[([CVE 1 :-> CONS (ATOM "1") (CONS (ATOM "0") (CONS (ATOM "1") (CONS (ATOM "1")
(ATOM ""))))],RESTR [])
```

```
*Main Lib Mult Paths_TSG_arithmetics_simple Plus Util> ura plusProg  ([decimalTo
BinaryExp 2, CVE 1], RESTR []) (decimalToBinaryExp 13)
[([CVE 1 :-> CONS (ATOM "1") (CONS (ATOM "0") (CONS (ATOM "1") (CONS (ATOM "1")
(ATOM ""))))],RESTR [])
```

```
*Main Lib Mult Paths_TSG_arithmetics_simple Plus Util> head $ ura plusProg  ([CVE 1,
CVE 2], RESTR []) (decimalToBinaryExp 13)
[([CVE 2 :-> CONS (ATOM "1") (ATOM ""),CVE 1 :-> CONS (ATOM "1") (CONS (ATOM "1") (CON
S (ATOM "0") (CONS (ATOM "0") (ATOM ""))))],RESTR [])
```

Умножение

- Вход программы: два числа в двоичной системе счисления
  Например:**CONS 1 (CONS 0 (ATOM ¨¨))**
- Алгоритм:
  - переворачиваем первое число
  - переворачиваем второе число
  - с помощью вспомогательной функции умножаем и складываем по два числа столбиком

Аргументы:

- e_a_shifted — первое число (перевернутое), к которому добавили нули (то есть сдвинули), его надо прибавить к e_new_res
- e_b_rest — второе число (перевернутое), отщепляем цифры от него, если 1, то надо сдвинуть первое число и прибавить, иначе просто сдвинуть
- e_res — текущий результат, используем для сложения и получения нового результата
- e_a_add — то самое сдвинутое первое число, которое тоже используем для сложения
- a_rem — остаток
- e_new_res — результат

Обратимость с алгоритмом ura

```
*Main Lib Mult Paths_TSG_arithmetics_simple Plus Util> ura multProg  ([CVE 1, decimal
ToBinaryExp 3], RESTR []) (decimalToBinaryExp 12)
[([CVE 1 :-> CONS (ATOM "1") (CONS (ATOM "0") (CONS (ATOM "0") (ATOM "")))],RESTR [])
```

```
*Main Lib Mult Paths_TSG_arithmetics_simple Plus Util> ura multProg  ([decimalToBinar
yExp 3, CVE 1], RESTR []) (decimalToBinaryExp 12)
[([CVE 1 :-> CONS (ATOM "1") (CONS (ATOM "0") (CONS (ATOM "0") (ATOM "")))],RESTR [])
```

```
*Main Lib Mult Paths_TSG_arithmetics_simple Plus Util> head $ ura multProg  ([CVE 1,
CVE 2], RESTR []) (decimalToBinaryExp 12)
([CVE 2 :-> CONS (ATOM "1") (ATOM ""),CVE 1 :-> CONS (ATOM "1") (CONS (ATOM "1") (CON
S (ATOM "0") (CONS (ATOM "0") (ATOM ""))))],RESTR [])
```

```
*Main Lib Mult Paths_TSG_arithmetics_simple Plus Util> head $ ura multProg  ([CVE 1,
CVE 1], RESTR []) (decimalToBinaryExp 16)
([CVE 1 :-> CONS (ATOM "1") (CONS (ATOM "0") (CONS (ATOM "0") (ATOM "")))],RESTR [])
```

Спасибо за внимание!