1 Построение TNFA

Для построения TNFA по дереву можно использовать несколько изменённый алгоритм Томпсона. Узлы дерва YARD можно разделить на 4 типа: лист, последовательность, альтернатива, замыкание. Эти типы соответствуют основным конструкциям расширенных регулярных выражений.

Атрибут ребра (дуги, перехода в NFA) - функция. В общем случае ей доступна вся информация о состоянии автомата и она может производить любые действия. В нашем случае она бутет записывать информацию о начале и конце распознования конструкции регулярного выражения в список.

Каждая метка соответствует началу или концу распознования той или иной конструкции регулярного выражения и содержит:

- уникальный идентификатор, одинаковый для метки начала и конца одной и той же конструкции
- информацию о конструкции, которой она соответствует
- позицию маркера в строке, в момент генерации метки

Определим следующий набор меток:

```
type tag =
  //Последовательность
  | TSeqStart of int*int //(уникальный идентификатор, позиция маркера)
  | TSeqEnd of int*int
  //Первая ветка альтернативы
  | TAlt1Start of int*int
  | TAlt1End of int*int
  //Вторая ветка альтернативы
  | TAlt2Start of int*int
  | TAlt2End of int*int
  //Замыкание
  | TClsStart of int*int
  | TClsEnd of int*int
```

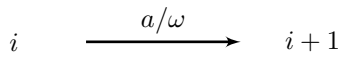
Определим функции, которые будем использовать в качестве меток для рёбер. Общим параметром для всех функций будет текущая позиция указателя в строке *pos*. Так же каждой функции доступна некоторая общая структура данных, в которую ведётся запись данных, *globalTagsList*. Набор функций выглядит следующим образом:

- ω – `fun _ -> ()` Будет означать отсутствие метки.
- $SeqStart_k(pos)$ – `fun pos -> TSeqStart(k, pos) :: globalTagsList` Подразумевается, что уникальный идентификатор вычисляется заранее, поэтому для разных конструкций строятся разные функции. Чтобы различать их, присвоим им индекс, соответствующий идентификатору.
- $SeqEnd_k(pos)$ – `fun pos -> TSeqEnd(k, pos) :: globalTagsList`

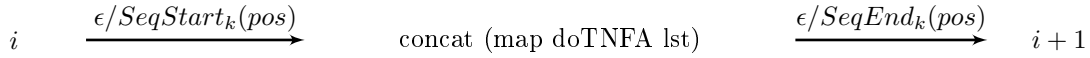
- $Alt1Start_k(pos)$ – `fun pos -> TAlt1Start(k, pos) :: globalTagsList`
- $Alt1End_k(pos)$ – `fun pos -> TAlt1End(k, pos) :: globalTagsList`
- $Alt2Start_k(pos)$ – `fun pos -> TAlt2Start(k, pos) :: globalTagsList`
- $Alt2End_k(pos)$ – `fun pos -> TAlt2End(k, pos) :: globalTagsList`
- $ClsStart_k(pos)$ – `fun pos -> TClsStart(k, pos) :: globalTagsList`
- $ClsEnd_k(pos)$ – `fun pos -> TClsEnd(k, pos) :: globalTagsList`

Тогда для расстановки меток можно будет дополнить алгоритм Томпсона. Результат будет таким:

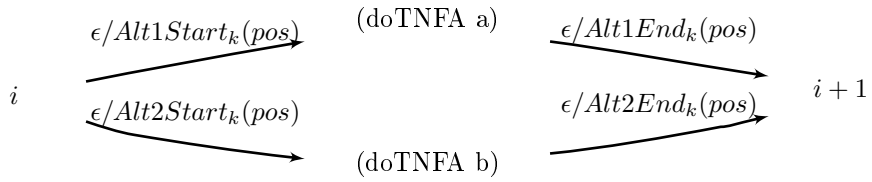
- Лист : Leaf(a)



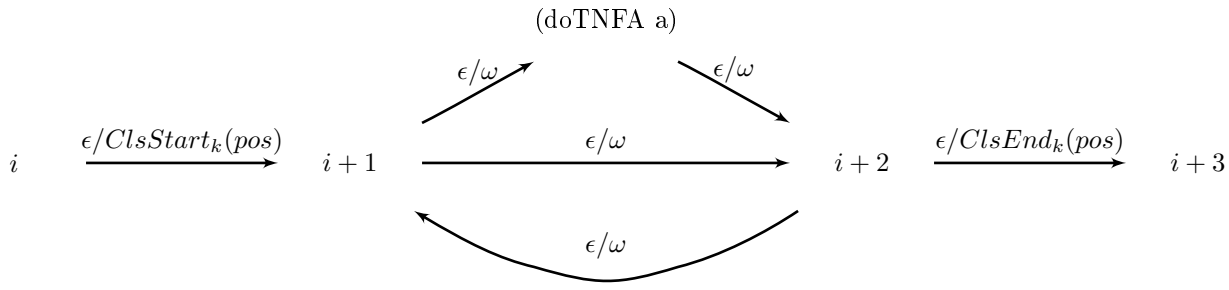
- Последовательность : Seq(lst)



- Альтернатива : Alt(a,b)



- Замыкание : Cls(a)



После работы построенного таким образом автомата в *globalTagsList* будет лежать структура, описывающая проесс распознования строк. Она будет являться привильной скобочной структурой(если считать, что метки начала и конца конструкции образуют скобчную пару).

2 Построение дерева вывода

Для построения дерева вывода строки, поданной на вход автомату, можно воспользоваться деей, описанной выше. Для этого потребуется только изменить фукции меток и глобальную стуктуру данных.

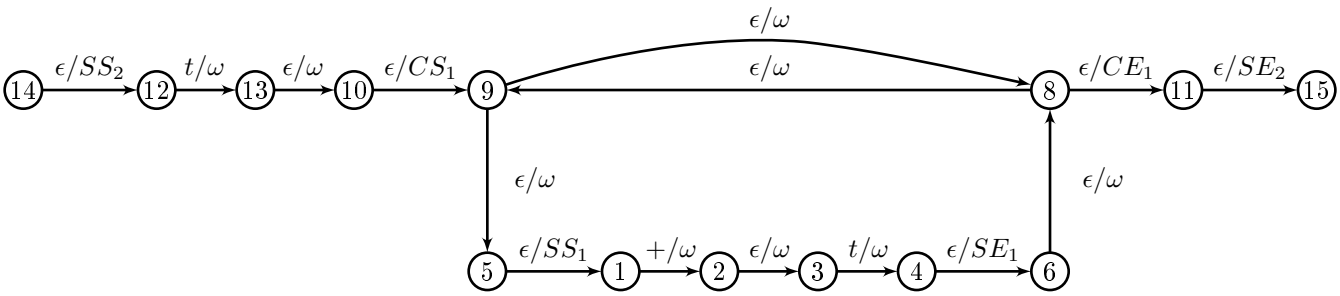
Необходимо преобразовать *globalTagsList* в стек. Теперь в нём будут лежать не только метки, но и деревья вывода. Фкнкции *Start* по прежнему будут класть в *globalTagsList* метку о начале соответствующей конструкции, а *End* функции будут снимать все элементы до соответствующей метки, строть из них очередной узел, соответствующий распознанной конструкции и класть его обратно.

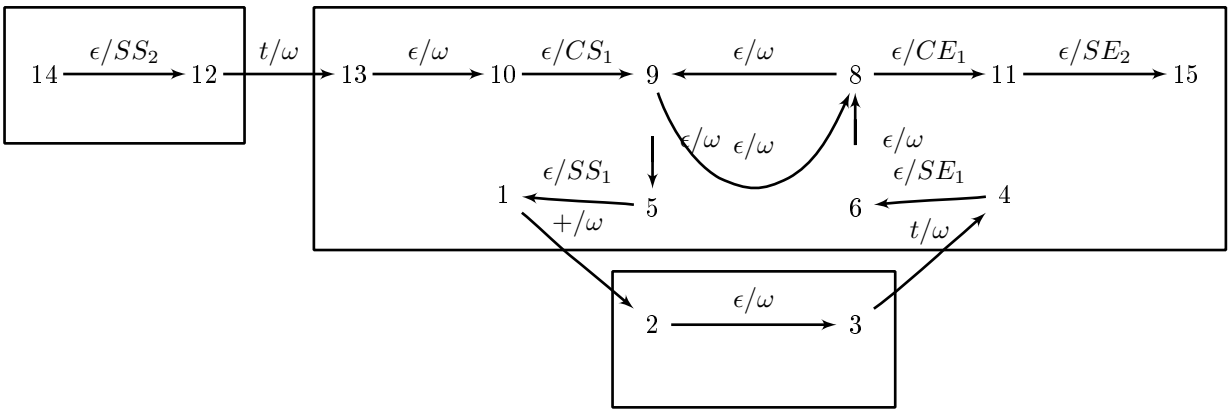
Таким образом, в конце работы автомата, если строка принята, то в *globalTagsList* будет лежать дерево вывода этой строки.

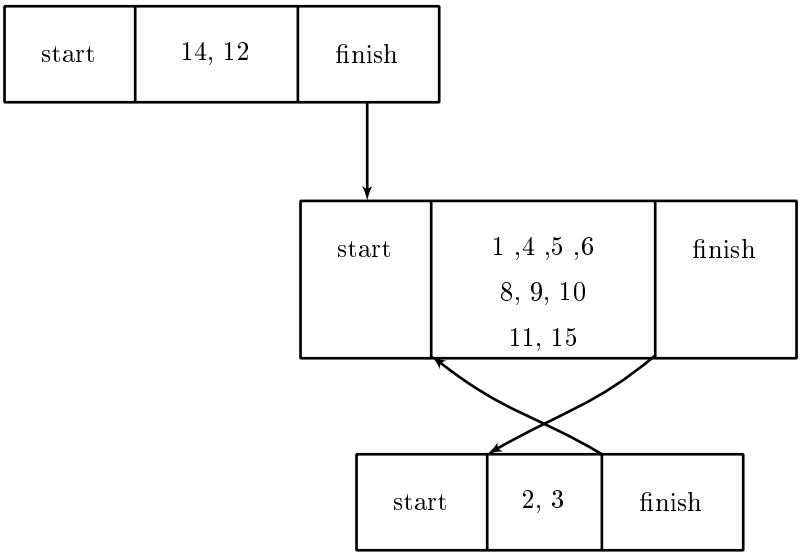
3 Преобразование TNFA в TDFA

Общая идея преобразования заключается в том, что можно расширить понятие состояния таким образом, чтобы стал возможен перенос в него функций с рёбер. Часть функций будут выполняться на входе в состояние, часть на выходе.

4 Пример




14
12
13
10
9
8
11
15
1
5
6
4
2
3
ϵ/SS_2
t/ω
ϵ/ω
ϵ/CS_1
ϵ/ω
ϵ/CE_1
ϵ/SE_2
ϵ/ω
ϵ/ω
ϵ/ω
ϵ/SS_1
ϵ/SE_1
$+/\omega$
t/ω
ϵ/ω


start
14, 12
finish
start
1 ,4 ,5 ,6
8, 9, 10
11, 15
finish
start
2, 3
finish