The Mechanical Evaluation Of Expressions

Aditya Siram (@deech)

January 30, 2017

Outline

SECD - Square Root

```
Transform(S,E,C,D) =
  nullC → [hS:S',E',C',D']
          where S',E',C',D' = D
  else →
    identifier X → [locationEXE:S, E, tC, D]
    λexp X →
      [constructclosure((E, bvX),unitlist(body X)):S,
      E, tC, D]
    X = ap → closure(hS) →
                  [( ), derive(assoc(J,2ndS)E'),
                       C,
                       (t(tS), E, tC, D)]
                  where E',J = environmentpart(hS)
                    and C' = controlpart(hS)
                else → [(1stS)(2nd):t(tS), E, tC, D]
    else → [S, E, randX:(ratorX:(ap:tC)), D]
    where X = hC
```

```
sqrt = \n . n * n
S = []
E = []
C = [sqrt(2)]
D = []
```

SECD - Square Root

```
Transform(S,E,C,D) =
  nullC → [hS:S',E',C',D']
          where S',E',C',D' = D
  else →
    identifier X → [locationEXE:S, E, tC, D]
    λexp X →
      [constructclosure((E, bvX),unitlist(body X)):S,
      E, tC, D]
    X = ap → closure(hS) →
                  [( ), derive(assoc(J,2ndS)E'),
                       C,
                       (t(tS), E, tC, D)]
                  where E',J = environmentpart(hS)
                    and C' = controlpart(hS)
                else → [(1stS)(2nd):t(tS), E, tC, D]
    else → [S, E, randX:(ratorX:(ap:tC)), D]
    where X = hC
```

```
sqrt = \n . n * n
S = []
E = []
C = [\n . n * n, 2]
D = []
```

SECD - Square Root

```
Transform(S,E,C,D) =
  nullC → [hS:S',E',C',D']
          where S',E',C',D' = D
  else →
    identifier X → [locationEXE:S, E, tC, D]
    λexp X →
      [constructclosure((E, bvX),unitlist(body X)):S,
      E, tC, D]
    X = ap → closure(hS) →
                  [( ), derive(assoc(J,2ndS)E'),
                       C,
                       (t(tS), E, tC, D)]
                  where E',J = environmentpart(hS)
                    and C' = controlpart(hS)
                else → [(1stS)(2nd):t(tS), E, tC, D]
    else → [S, E, randX:(ratorX:(ap:tC)), D]
    where X = hC
```

```
sqrt = \n . n * n
S = []
E = []
C = [2, \n . n * n, ap]
D = []
```

SECD - Square Root

```
Transform(S,E,C,D) =
  nullC → [hS:S',E',C',D']
          where S',E',C',D' = D
  else →
    identifier X → [locationEXE:S, E, tC, D]
    λexp X →
      [constructclosure((E, bvX),unitlist(body X)):S,
      E, tC, D]
    X = ap → closure(hS) →
                  [( ), derive(assoc(J,2ndS)E'),
                       C,
                       (t(tS), E, tC, D)]
                  where E',J = environmentpart(hS)
                    and C' = controlpart(hS)
                else → [(1stS)(2nd):t(tS), E, tC, D]
    else → [S, E, randX:(ratorX:(ap:tC)), D]
    where X = hC
```

```
sqrt = \n . n * n
S = [2]
E = []
C = [\n . n * n, ap]
D = []
```

SECD - Square Root

```
Transform(S,E,C,D) =
  nullC → [hS:S',E',C',D']
          where S',E',C',D' = D
  else →
    identifier X → [locationEXE:S, E, tC, D]
    λexp X →
      [constructclosure((E, bvX),unitlist(body X)):S,
      E, tC, D]
    X = ap → closure(hS) →
                  [( ), derive(assoc(J,2ndS)E'),
                       C,
                       (t(tS), E, tC, D)]
                  where E',J = environmentpart(hS)
                    and C' = controlpart(hS)
                else → [(1stS)(2nd):t(tS), E, tC, D]
    else → [S, E, randX:(ratorX:(ap:tC)), D]
    where X = hC
```

```
sqrt = \n . n * n
S = [closure [] x [x * x], 2]
E = []
C = [ap]
D = []
```

SECD - Square Root

```
Transform(S,E,C,D) =
  nullC → [hS:S',E',C',D']
          where S',E',C',D' = D
  else →
    identifier X → [locationEXE:S, E, tC, D]
    λexp X →
      [constructclosure((E, bvX),unitlist(body X)):S,
      E, tC, D]
    X = ap → closure(hS) →
                  [( ), derive(assoc(J,2ndS)E'),
                       C,
                       (t(tS), E, tC, D)]
                  where E',J = environmentpart(hS)
                    and C' = controlpart(hS)
                else → [(1stS)(2nd):t(tS), E, tC, D]
    else → [S, E, randX:(rator X:(ap:tC)), D]
    where X = hC
```

```
sqrt = \n . n * n
S = []
E = [[(x 2)]]
C = [x * x]
D = []
```

SECD - Square Root

```
Transform(S,E,C,D) =
  nullC → [hS:S',E',C',D']
          where S',E',C',D' = D
  else →
    identifier X → [locationEXE:S, E, tC, D]
    λexp X →
      [constructclosure((E, bvX),unitlist(body X)):S,
      E, tC, D]
    X = ap → closure(hS) →
                  [( ), derive(assoc(J,2ndS)E'),
                       C,
                       (t(tS), E, tC, D)]
                  where E',J = environmentpart(hS)
                    and C' = controlpart(hS)
                else → [(1stS)(2nd):t(tS), E, tC, D]
    else → [S, E, randX:(ratorX:(ap:tC)), D]
    where X = hC
```

```
sqrt = \n . n * n
S = []
E = [[(x 2)]]
C = [x x * ap]
D = []
```

SECD - Square Root

```
Transform(S,E,C,D) =
  nullC → [hS:S',E',C',D']
          where S',E',C',D' = D
  else →
    identifier X → [locationEXE:S, E, tC, D]
    λexp X →
      [constructclosure((E, bvX),unitlist(body X)):S,
      E, tC, D]
    X = ap → closure(hS) →
                  [( ), derive(assoc(J,2ndS)E'),
                       C,
                       (t(tS), E, tC, D)]
                  where E',J = environmentpart(hS)
                    and C' = controlpart(hS)
                else → [(1stS)(2nd):t(tS), E, tC, D]
    else → [S, E, randX:(ratorX:(ap:tC)), D]
    where X = hC
```

```
sqrt = \n . n * n
S = [[x x]]
E = [[(x 2)]]
C = [* ap]
D = []
```

SECD - Square Root

```
Transform(S,E,C,D) =
  nullC → [hS:S',E',C',D']
          where S',E',C',D' = D
  else →
    identifierX → [locationEXE:S, E, tC, D]
    λexpX →
      [constructclosure((E, bvX),unitlist(bodyX)):S,
      E, tC, D]
    X = ap → closure(hS) →
                  [( ), derive(assoc(J,2ndS)E'),
                       C,
                       (t(tS), E, tC, D)]
                  where E',J = environmentpart(hS)
                    and C' = controlpart(hS)
                else → [(1stS)(2nd):t(tS), E, tC, D]
    else → [S, E, randX:(ratorX:(ap:tC)), D]
    where X = hC
```

```
sqrt = \n . n * n
S = [[2 2]]
E = [[(x 2)]]
C = [* ap]
D = []
```

SECD - Square Root

```
Transform(S,E,C,D) =
  nullC → [hS:S',E',C',D']
          where S',E',C',D' = D
  else →
    identifierX → [locationEXE:S, E, tC, D]
    λexpX →
      [constructclosure((E, bvX),unitlist(bodyX)):S,
      E, tC, D]
    X = ap → closure(hS) →
                  [( ), derive(assoc(J,2ndS)E'),
                       C,
                       (t(tS), E, tC, D)]
                  where E',J = environmentpart(hS)
                    and C' = controlpart(hS)
                else → [(1stS)(2nd):t(tS), E, tC, D]
    else → [S, E, randX:(ratorX:(ap:tC)), D]
    where X = hC
```

```
sqrt = \n . n * n
S = [4]
E = [[(x 2)]]
C = []
D = []
```