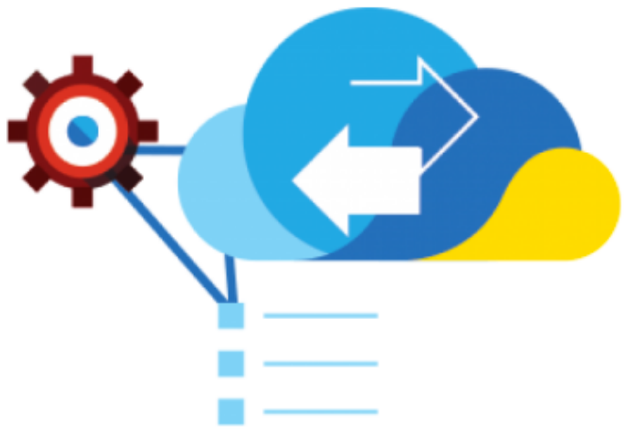
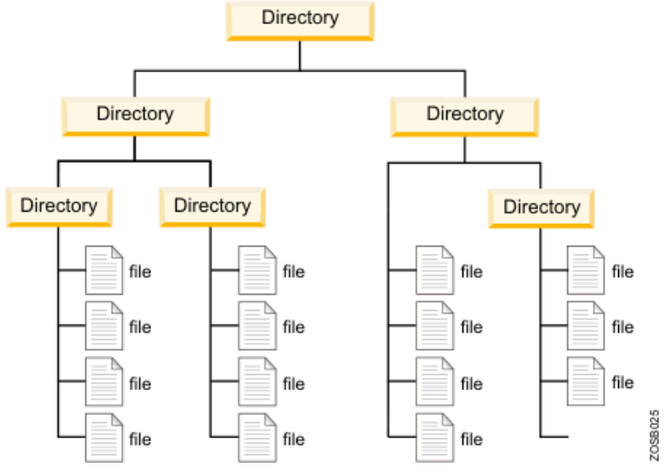
Synthesizing Bijective String Transformers

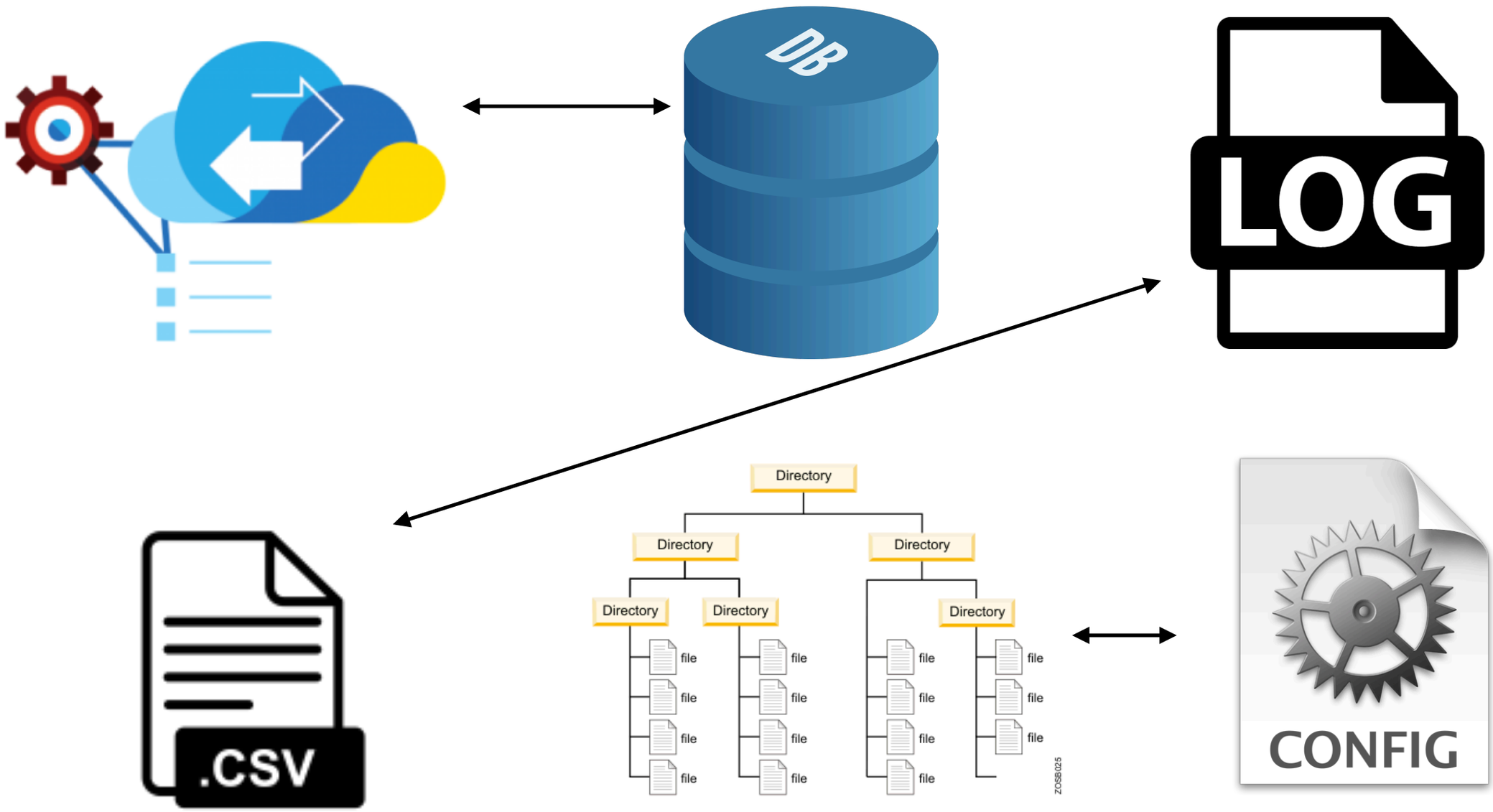
Anders Miltner
David Walker (advisor)
Kathleen Fisher
Benjamin Pierce
Steve Zdancewic

Data is **Everywhere**





Data is **Everywhere**



Example: BibTex and Endnote

BibTex

```
@book{
  title={Introduction to
    MetaMathematics},
  author={Kleane, Stephen
    Cole},
}
```

EndNote

```
%0 Book
%T Introduction to
   MetaMathematics
%A Stephen Cole Kleane
```

Example: BibTex and Endnote

BibTex

```
@book{
  title={Introduction to
    MetaMathematics},
  author={Kleene, Stephen
    Cole},
}
```

EndNote

```
%0 Book
%T Introduction to
   MetaMathematics
%A Stephen Cole Kleane
```

Example: BibTex and Endnote

BibTex

```
@book{
  title={Introduction to
    MetaMathematics},
  author={Kleene, Stephen
    Cole},
}
```

EndNote

```
%0 Book
%T Introduction to
   MetaMathematics
%A Stephen Cole Kleene
```

Example: BibTex and Endnote

BibTex

```
@book{
  title={Introduction to
    MetaMathematics},
  author={Kleene, Stephen
    Cole},
}
```

EndNote

```
%0 Book
%T Introduction to
   MetaMathematics
%A Stephen Cole Kleene
```

Example: BibTex and Endnote

BibTex

```
@book{
  title={Introduction to
    MetaMathematics},
  author={Kleene, Stephen
    Cole},
}
```

EndNote

```
%0 Book
%T Introduction to
   MetaMathematics
%A Stephen Cole Kleene

%0 Article
%T Type-and-Example-Directed
   Program Synthesis
%A Peter-Michael Osera
%A Steve Zdancewic
```

Example: BibTex and Endnote

BibTex

```
@book{
  title={Introduction to
    MetaMathematics},
  author={Kleene, Stephen
    Cole},
}

@article{
  title={Type-and-Example-
    Directed Program
    Synthesis},
  author={Osera, Peter-Michael
    and Zdancewic, Steve},
}
```

EndNote

```
%0 Book
%T Introduction to
   MetaMathematics
%A Stephen Cole Kleene

%0 Article
%T Type-and-Example-Directed
   Program Synthesis
%A Peter-Michael Osera
%A Steve Zdancewic
```

Bidirectional Programming

- Use a single piece of code to express functions between data formats
- Desire code to be correct by construction
  - Inversion properties guaranteed
- We focus on a bijective fragment of Boomerang [BFPPS ’08]
  - Terms are “lenses”
  - Mappings between string data formats
  - Generated functions are inverses of each other

Issues with Bidirectional Programming

- Difficult to program in!
  - Complicated Type Systems
  - New Syntax
  - Fiddly details
  - New way of thinking about programs

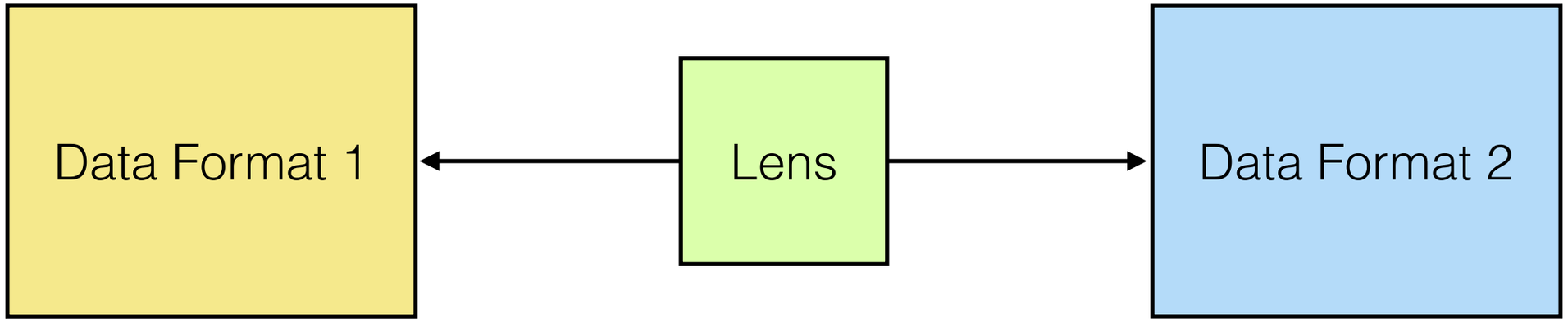
Issues with Bidirectional Programming

Miltner,Anders,NJ, \
08544,Princeton,11 Dickinson \
St.,Apt 11

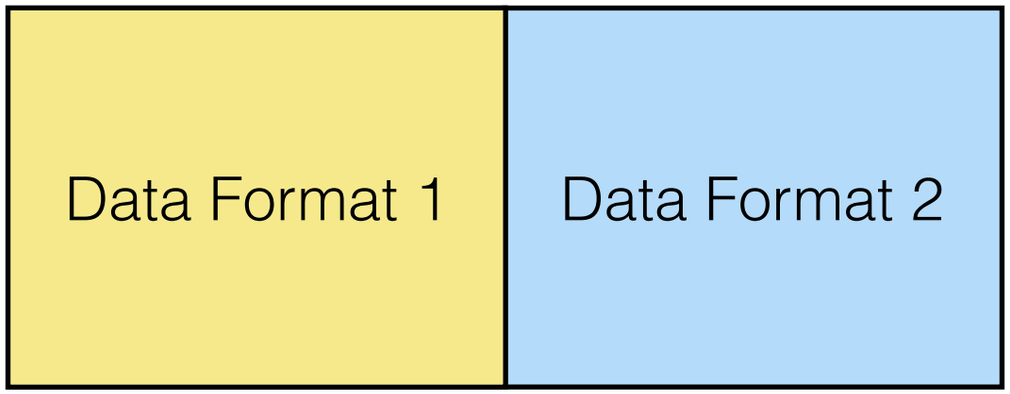
Anders Miltner
11 Dickinson St.
Apt 11
Princeton, NJ 08544

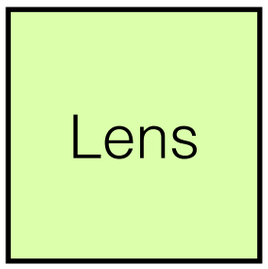
concat(swap(concat(…),swap(swap(…),concat(…))

Synthesis as a Solution!

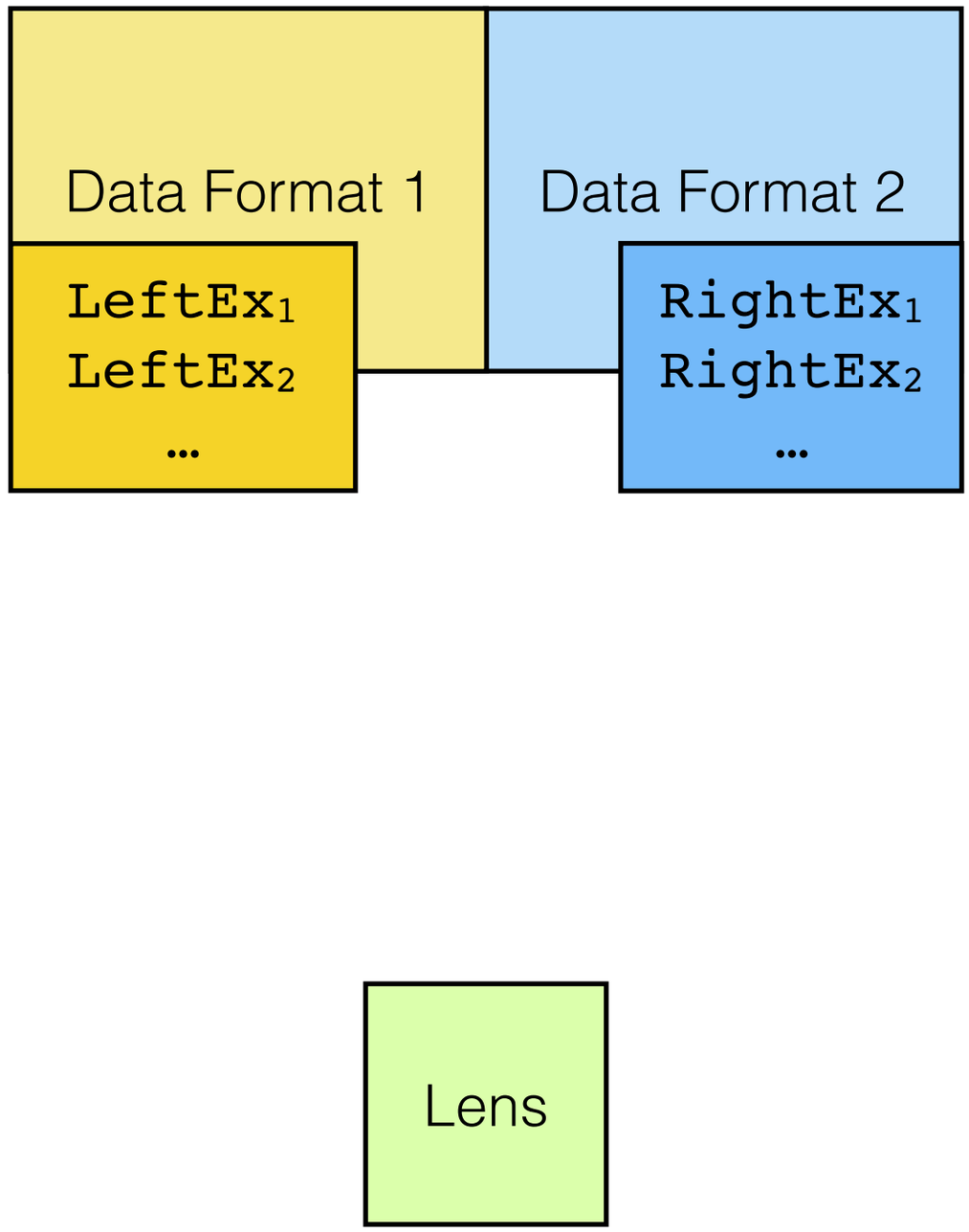


Synthesis as a Solution!

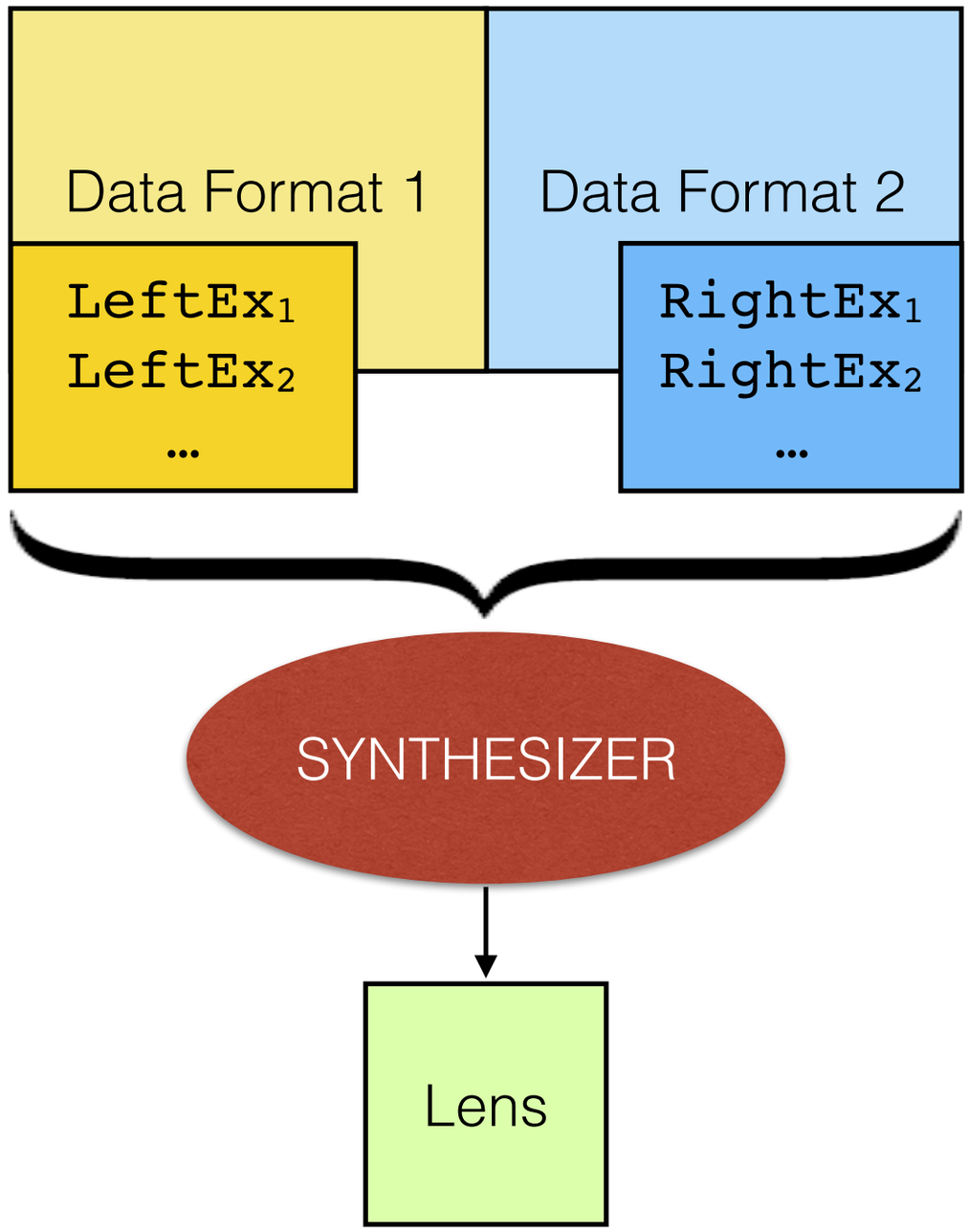




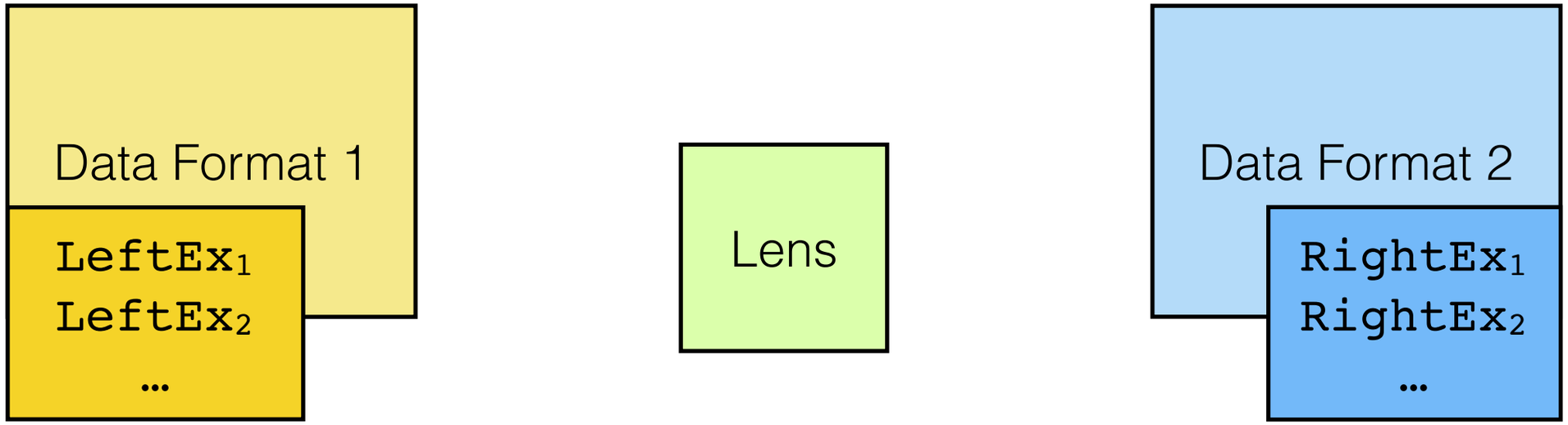
Synthesis as a Solution!



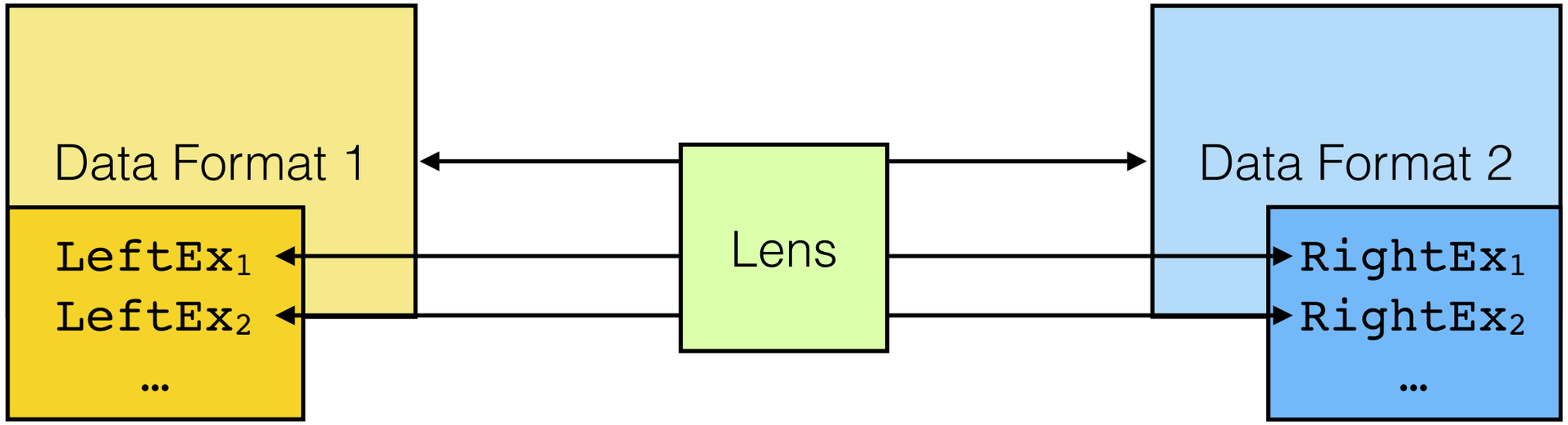
Synthesis as a Solution!



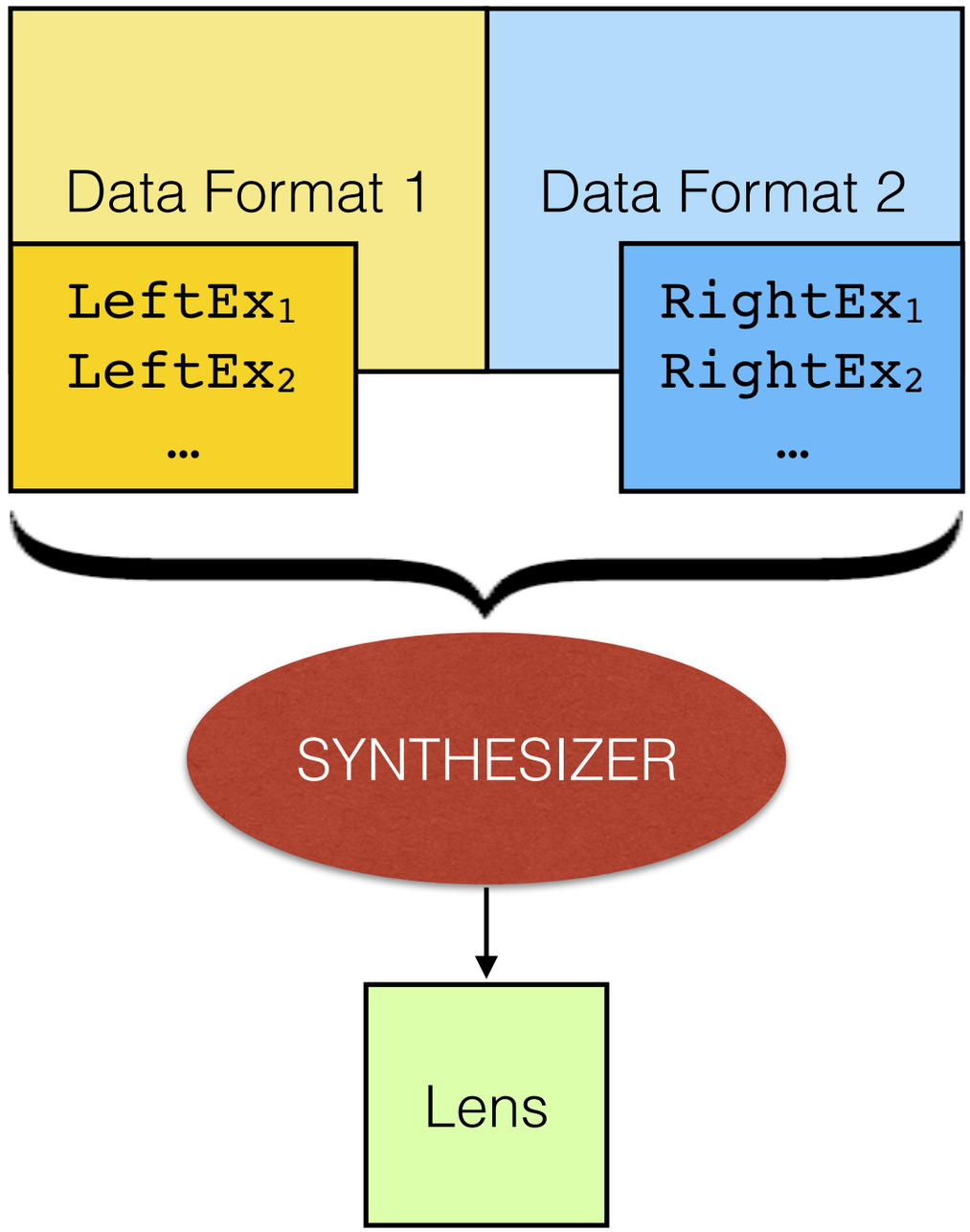
Synthesis as a Solution!



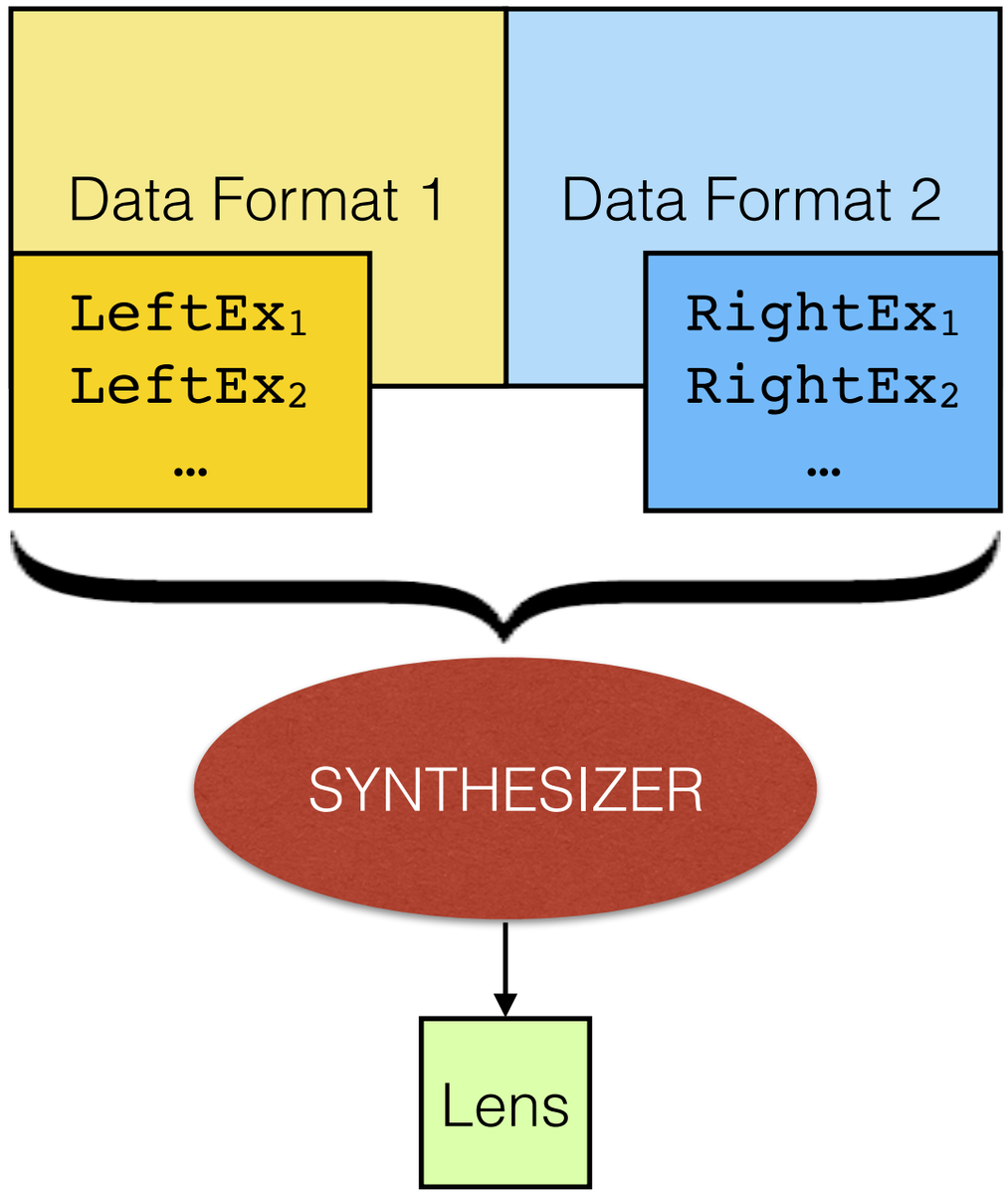
Synthesis as a Solution!



Synthesis as a Solution!



Synthesis as a Solution!



Talk Structure

Present…

- Existing lens language (bijective boomerang)
- Naive approach to synthesis, to motivate our approach
- **DNF lenses** which are more suited to synthesis
- **Algorithm** for synthesizing DNF lenses from types and examples
- **Evaluation of our algorithm** of a test suite of 25 synthesis tasks

Bijective Boomerang

Bijective Boomerang

- `l : R <=> S`
- `l.get : R → S`
- `l.put : S → R`
- Invertibility Guarantees?
  - `l.put ○ l.get = id|R`
  - `l.get ○ l.put = id|S`
  - From side conditions
- Extended example with BibTex to EndNote author field

Constant Lens

```
author={Kleene, Stephen Cole}
```

```
%A Stephen Cole Kleene
```

Constant Lens

```
author={Kleene, Stephen Cole}
```

```
%A Stephen Cole Kleene
```

```
const("author={","%A") : "author={" <=> "%A"
```

Constant Lens

```
author={Kleene, Stephen Cole}          %A Stephen Cole Kleene
```

```
const("author={","%A") : "author={" <=> "%A"
```

```
const(s,t) : s <=> t
```

Identity Lens

```
author={Kleene, Stephen Cole}          %A Stephen Cole Kleene
```

Identity Lens

```
author={Kleene, Stephen Cole}          %A Stephen Cole Kleene
```

```
id([A-Z][a-z]*) : [A-Z][a-z]* <=> [A-Z][a-z]*
```

Identity Lens

```
author={Kleene, Stephen Cole}          %A Stephen Cole Kleene
```

```
id([A-Z][a-z]*) : [A-Z][a-z]* <=> [A-Z][a-z]*
```

$$\frac{\textit{side conditions}}{\texttt{id(R) : R <=> R}}$$

Concat Lens

```
author={Kleene, Stephen Cole}        %A Stephen Cole Kleene
```

Concat Lens

```
author={Kleene, Stephen Cole}          %A Stephen Cole Kleene
```

$$\text{concat}(l_1, l_2) : R_1.R_2 \iff S_1.S_2$$

Concat Lens

```
author={Kleene, Stephen Cole}          %A Stephen Cole Kleene
```

$$\texttt{concat}(l_1, l_2) : R_1.R_2 \Longleftrightarrow S_1.S_2$$

$$\frac{\begin{array}{c} l_1 : R_1 \Longleftrightarrow S_1 \\ l_2 : R_2 \Longleftrightarrow S_2 \\ \textit{side conditions} \end{array}}{\texttt{concat}(l_1, l_2) : R_1.R_2 \Longleftrightarrow S_1.S_2}$$

Swap Lens

```
author={Kleene, Stephen Cole}
```

```
%A Stephen Cole Kleene
```

Swap Lens

```
author={Kleene, Stephen Cole}          %A Stephen Cole Kleene
```

$$swap(l_1, l_2) : R_1.R_2 <=> S_2.S_1$$

Swap Lens

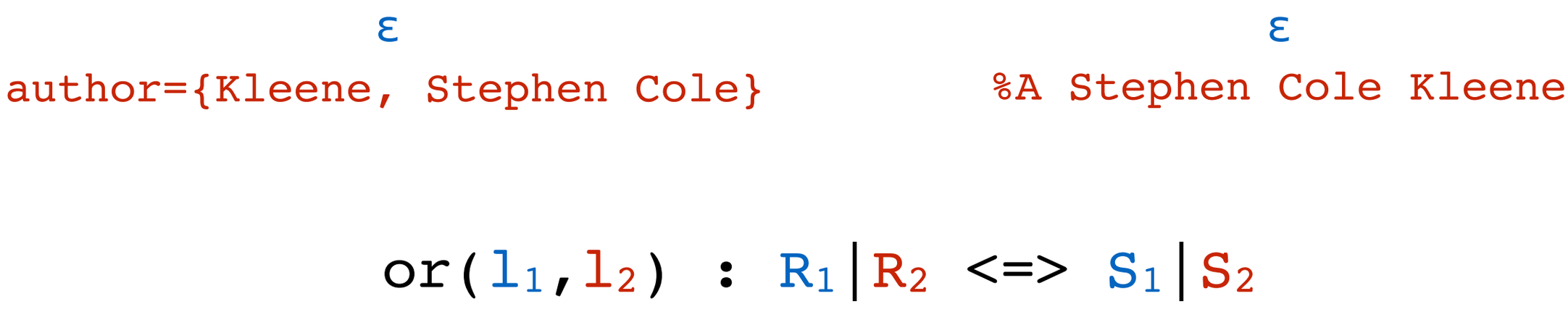
```
author={Kleene, Stephen Cole}
```

```
%A Stephen Cole Kleene
```

$$\frac{\begin{array}{c} l_1 : R_1 \mathrel{<=>} S_1 \\ l_2 : R_2 \mathrel{<=>} S_2 \\ \textit{side conditions} \end{array}}{\mathtt{swap}(l_1, l_2) : R_1.R_2 \mathrel{<=>} S_2.S_1}$$

Or Lens

Or Lens


ε
author={Kleene, Stephen Cole}
ε
%A Stephen Cole Kleene
or(l1,l2) : R1|R2 <=> S1|S2

Or Lens

```
                    ε                                        ε
author={Kleene, Stephen Cole}          %A Stephen Cole Kleene
```

$$\texttt{or}(l_1, l_2) \;:\; R_1 \mid R_2 \;\texttt{<=>}\; S_1 \mid S_2$$

$$\frac{\begin{array}{c} l_1 \;:\; R_1 \;\texttt{<=>}\; S_1 \\ l_2 \;:\; R_2 \;\texttt{<=>}\; S_2 \\ \textit{side conditions} \end{array}}{\texttt{or}(l_1, l_2) \;:\; R_1 \mid R_2 \;\texttt{<=>}\; S_1 \mid S_2}$$

Iterate Lens

```
author={Hofmann, Martin
  and Pierce, Benjamin
  and Wagner, Daniel}
```

```
%A Martin Hofmann
%A Benjamin Pierce
%A Daniel Wagner
```

Iterate Lens

```
author={Hofmann, Martin
  and Pierce, Benjamin
  and Wagner, Daniel}
```

```
%A Martin Hofmann
%A Benjamin Pierce
%A Daniel Wagner
```

```
iterate(l) : R* <=> S*
```

Iterate Lens

```
author={Hofmann, Martin
  and Pierce, Benjamin
  and Wagner, Daniel}
```

```
%A Martin Hofmann
%A Benjamin Pierce
%A Daniel Wagner
```

```
iterate(l) : R* <=> S*
```

$$\frac{\texttt{l : R <=> S} \quad \textit{side conditions}}{\texttt{iterate(l) : R* <=> S*}}$$

Single Author Lens!

```
author={Kleene, Stephen Cole},          %A Stephen Cole Kleene
```

Single Author Lens!

```
author={Kleene, Stephen Cole},
```

```
%A Stephen Cole Kleene
```

```
concat(
  const("author={","%A"),
  l)
```

Single Author Lens!

```
author={Kleene, Stephen Cole},
```

```
%A Stephen Cole Kleene
```

```
concat(
  const("author={","%A"),
concat(
  l,
  const("},",ε)))
```

Single Author Lens!

```
author={Kleene, Stephen Cole},
```

```
%A Stephen Cole Kleene
```

```
concat(
  const("author={","%A"),
concat(
  l,
  const("},",ε)))
```

```
id(" "[A-Z][a-z]*)
```

Single Author Lens!

```
author={Kleene, Stephen Cole},
```

```
concat(
  const("author={","%A"),
concat(
  l,
  const("},",ε)))
```

```
%A Stephen Cole Kleene
```

```
iterate(id(" "[A-Z][a-z]*))
```

Single Author Lens!

```
author={Kleene, Stephen Cole},          %A Stephen Cole Kleene
```

```
concat(
  const("author={","%A"),
concat(
  swap(
    l,
    iterate(id(" "[A-Z][a-z]*))),
  const("},",ε)))
```

Single Author Lens!

```
author={Kleene, Stephen Cole},          %A Stephen Cole Kleene
```

```
concat(
  const("author={","%A"),
  concat(
    swap(
      concat(
        id([A-Z][a-z]*),
        l),
      iterate(id(" "[A-Z][a-z]*))),
    const("},",ε)))
```

Single Author Lens!

```
author={Kleene, Stephen Cole},          %A Stephen Cole Kleene
```

```
concat(
  const("author={","%A"),
concat(
  swap(
    concat(
      id([A-Z][a-z]*),
      const(",",ε)),
    iterate(id(" "[A-Z][a-z]*))),
  const("},",ε)))
```

Single Author Lens!

```
author={Kleene, Stephen Cole},
```

```
concat(
  const("author={","%A"),
  concat(
    swap(
      concat(
        id([A-Z][a-z]*),
        const(",",ε)),
      iterate(id(" "[A-Z][a-z]*))),
    const("},",ε)))
```

•
•

```
%A Stephen Cole Kleene
```

```
"author={"
[A-Z][a-z]
    ","
(" "[A-Z][a-z])*
"}"
```

<=>

```
"%A"
(" "[A-Z][a-z])*
[A-Z][a-z]
    ε
    ε
```

Single Author Lens!

```
author={Kleene, Stephen Cole},
```

```
%A Stephen Cole Kleene
```

```
concat(
  const("author={","%A"),
  concat(
    swap(
      concat(
        id([A-Z][a-z]*),
        const(",",ε)),
      iterate(id(" "[A-Z][a-z]*))),
    const("},",ε)))
```

```
"author={"
  (((([A-Z][a-z]
      ",")
    (" "[A-Z][a-z])*)
  "}")
```

<=>

```
"%A"
  ((((" "[A-Z][a-z])*
    ([A-Z][a-z]
      ε))
    ε)
```

Potential Problem!

```
"author={"
  [A-Z][a-z]
    ","
      (" "[A-Z][a-z])*
        "}"
<=>
"%A"
  (" "[A-Z][a-z])*
  [A-Z][a-z]
    ε
      ε
```

```
"author={"
  (((([A-Z][a-z]
          ",")
       (" "[A-Z][a-z])*)
    "}")
<=>
"%A"
  ((" "[A-Z][a-z])*
  ([A-Z][a-z]
    ε))
  ε)
```

Potential Problem!

```
"author={"
  ([A-Z][a-z]
    (","
      ((" "[A-Z][a-z])*
        "}")))
<=>
"%A"
  ((" "[A-Z][a-z])*
([A-Z][a-z]
  (ε
    ε)))
```

```
"author={"
  (((([A-Z][a-z]
        ",")
      (" "[A-Z][a-z])*)
    "}")
<=>
"%A"
  (((" "[A-Z][a-z])*
([A-Z][a-z]
  ε))
ε)
```

Potential Problem!

```
"author={"
  [A-Z][a-z]
    ","
      (" "[A-Z][a-z])*
        "}"
<=>
"%A"
  (" "[A-Z][a-z])*
  [A-Z][a-z]
    ε
      ε
```

```
"author={"
  (((([A-Z][a-z]
        ",")
      (" "[A-Z][a-z])*)
    "}")
<=>
"%A"
  ((" "[A-Z][a-z])*
  ([A-Z][a-z]
    ε))
  ε)
```

Potential Problem!

```
Name = [A-Z][a-z]

"author={"
  Name
    ","
      (" "Name)*
        "}"
<=>
"%A"
  (" "Name)*
    Name
      ε
        ε
```

```
"author={"
  ((([A-Z][a-z]
        ",")
      (" "[A-Z][a-z])*)
    "}")
<=>
"%A"
  ((" "[A-Z][a-z])*
  ([A-Z][a-z]
    ε))
ε)
```

Potential Problem!

```
Name = [A-Z][a-z]
Names = (" "Name)*

"author={"
  Name
    ","
      Names
        "}"
<=>
"%A"
  Names
    Name
      ε
        ε
```

```
"author={"
  ((([A-Z][a-z]
        ",")
      (" "[A-Z][a-z])*)
    "}")
<=>
"%A"
  ((" "[A-Z][a-z])*
  ([A-Z][a-z]
    ε))
ε)
```

Potential Problem!

```
Name = [A-Z][a-z]
Names = (" “Name)*

SingleBibAuth =
  "author={" Name "," Names "}"

SingleEndAuth =
  “%A” Names Name

SingleBibAuth <=> SingleEndAuth
```

```
"author={"
  ((([A-Z][a-z]
        ",")
      (" "[A-Z][a-z])*)
    "}")
<=>
"%A"
  ((" "[A-Z][a-z])*
  ([A-Z][a-z]
    ε))
ε)
```

Retyping

```
Name = [A-Z][a-z]
Names = (" “Name)*

SingleBibAuth =
  "author={" Name "," Names "}"

SingleEndAuth =
  “%A” Names Name

SingleBibAuth <=> SingleEndAuth
```

```
"author={"
  ((([A-Z][a-z]
        ",")
      (" "[A-Z][a-z])*)
    "}")
<=>
"%A"
  ((" "[A-Z][a-z])*
  ([A-Z][a-z]
    ε))
ε)
```

$$\frac{R \equiv R' \quad S \equiv S' \quad l : R' \mathrel{<=>} S'}{l : R \mathrel{<=>} S}$$

$$R \equiv R'$$

- Semiring Axioms

R ≡ R′

- Semiring Axioms

  - R | ∅ ≡ R
  - R . ∅ ≡ ∅
  - ∅ . R ≡ ∅
  - (R . R′) . R′′ ≡ R . (R′ . R′′)
  - (R | R′) | R′′ ≡ R | (R′ | R′′)
  - R | T ≡ T | R
  - R . (R′ | R′′) ≡ (R . R′) | (R . R′′)
  - (R′ | R′′) . R ≡ (R′ . R) | (R′′ . R)
  - ε . R ≡ R
  - R . ε ≡ R

$$R \equiv R'$$

- Semiring Axioms
- ε-unrolling for *

R ≡ R′

- Semiring Axioms
- ε-unrolling for *
  - R* ≡ ε | (R . R*)
  - R* ≡ ε | (R* . R)

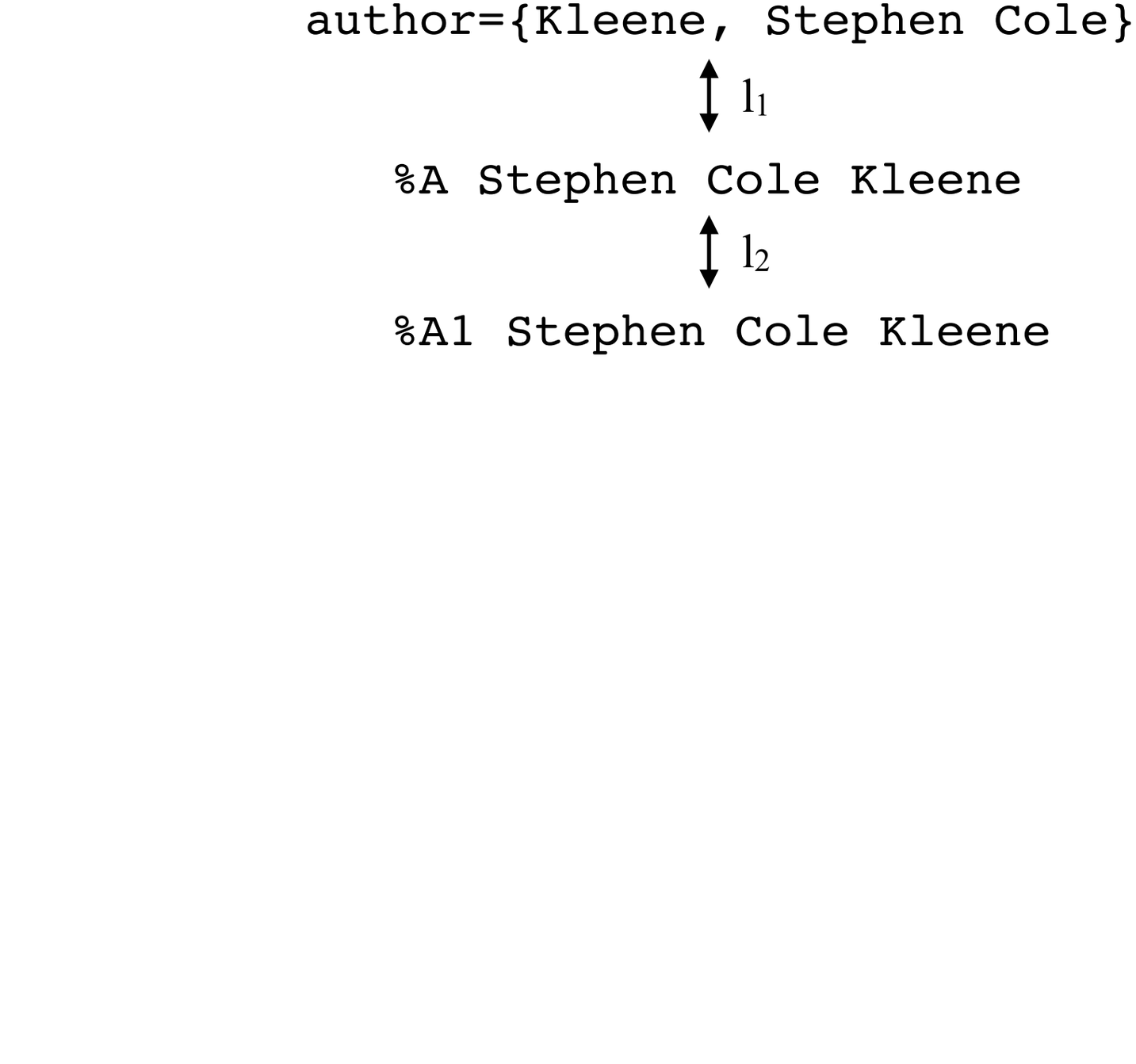
$$R \equiv R'$$

- Semiring Axioms
- ε-unrolling for *
- R is *definitionally equivalent* to R′

$$R \equiv R'$$

- Semiring Axioms
- ε-unrolling for *
- `R` is *definitionally equivalent* to `R'`
- Not full equivalence, but sufficient in practice!

Composition



Composition

```
author={Kleene, Stephen Cole}
```

↕ l_1

```
%A Stephen Cole Kleene
```

↕ l_2

```
%A1 Stephen Cole Kleene
```

$l_1;l_2$: SingleBibAuth <=> NewSingleEndAuth

Composition

```
author={Kleene, Stephen Cole}
```

$\updownarrow l_1$

```
%A Stephen Cole Kleene
```

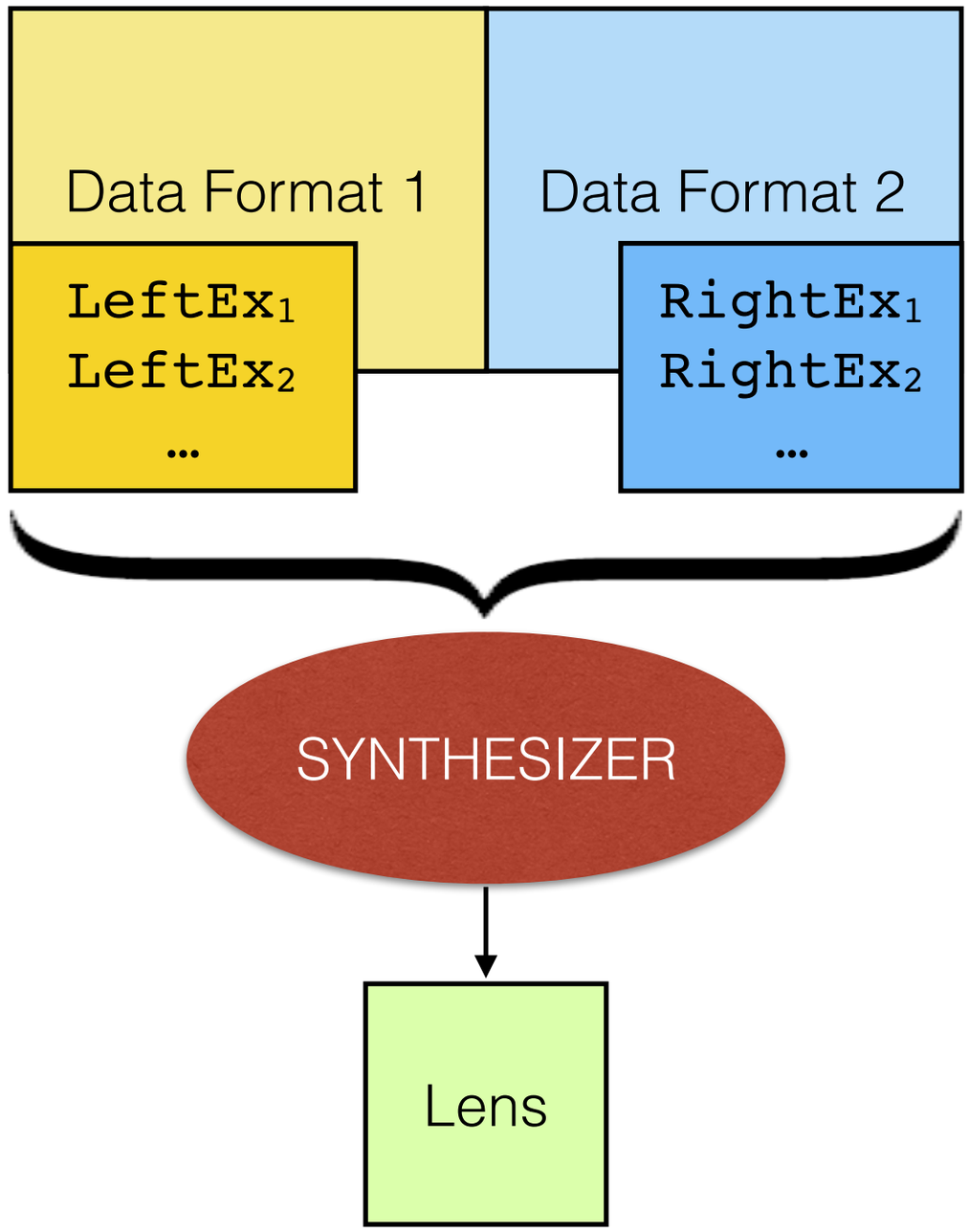
$\updownarrow l_2$

```
%A1 Stephen Cole Kleene
```

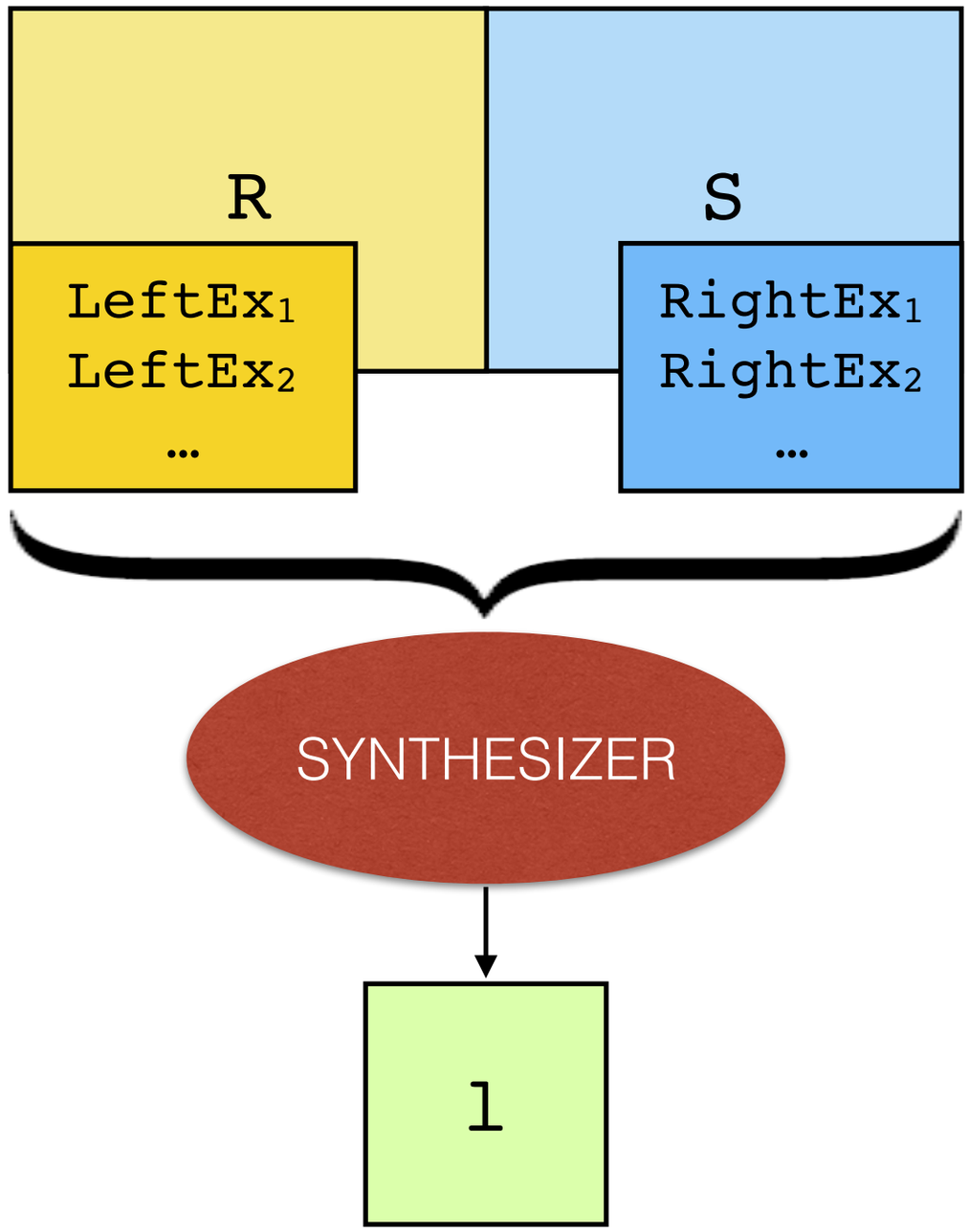
$l_1;l_2$: SingleBibAuth <=> NewSingleEndAuth

$$\frac{l_1 : R \Leftrightarrow S \qquad l_2 : S \Leftrightarrow T}{l_1;l_2 : R \Leftrightarrow T}$$

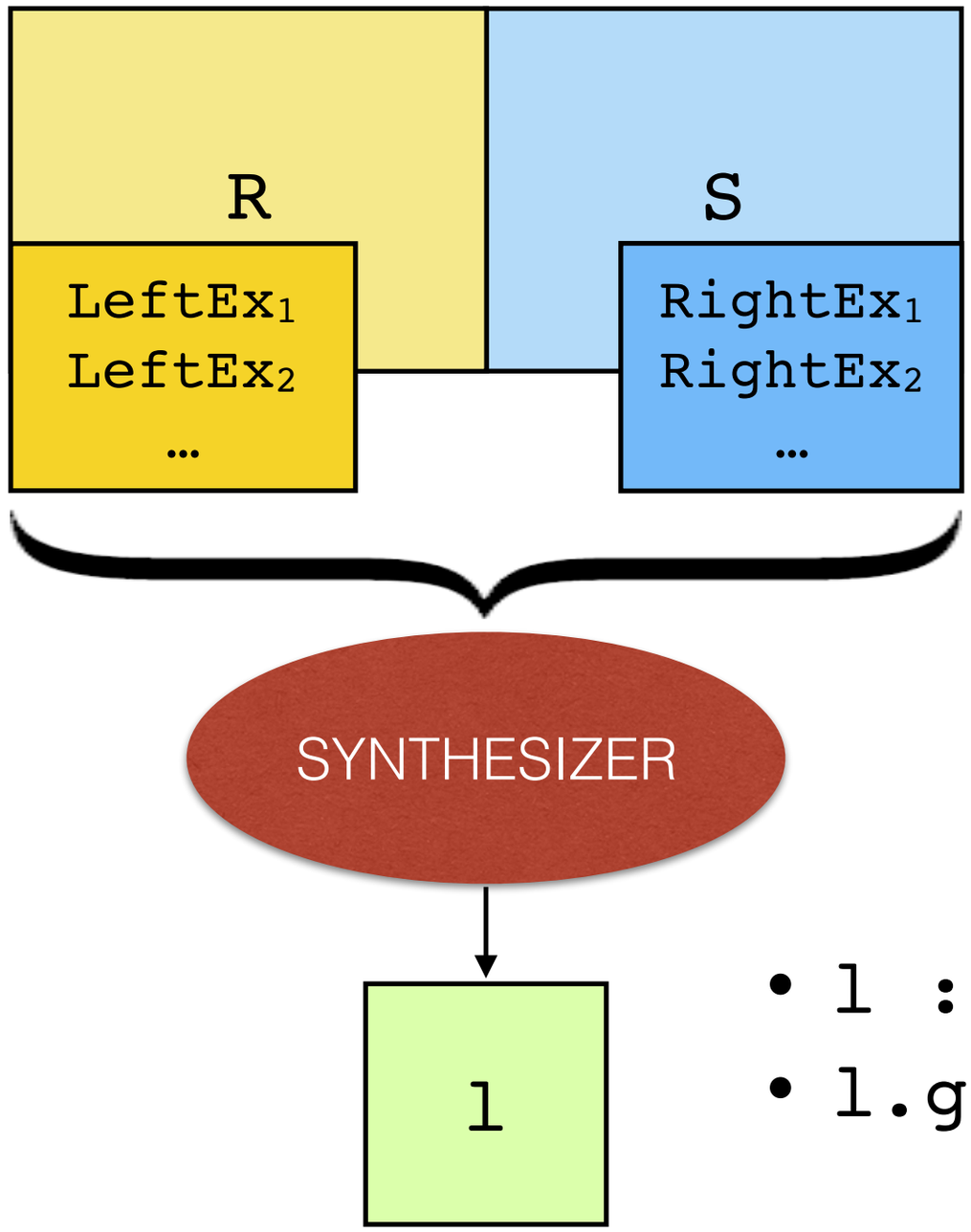
Lenses from Data Formats



Lenses from Data Formats

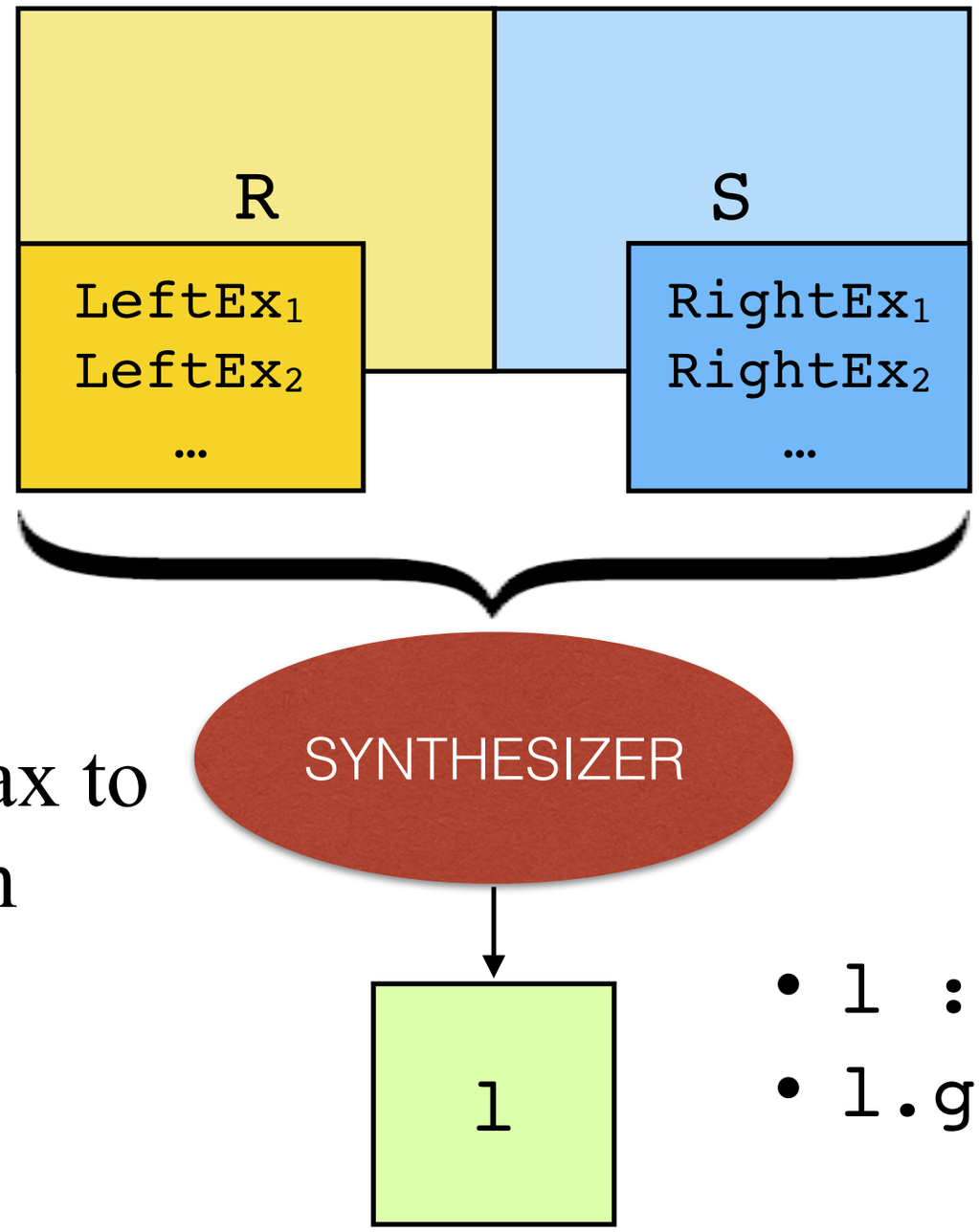


Lenses from Data Formats



- `l : R <=> S`
- `l.get` $\texttt{LeftEx}_i$ = $\texttt{RightEx}_i$

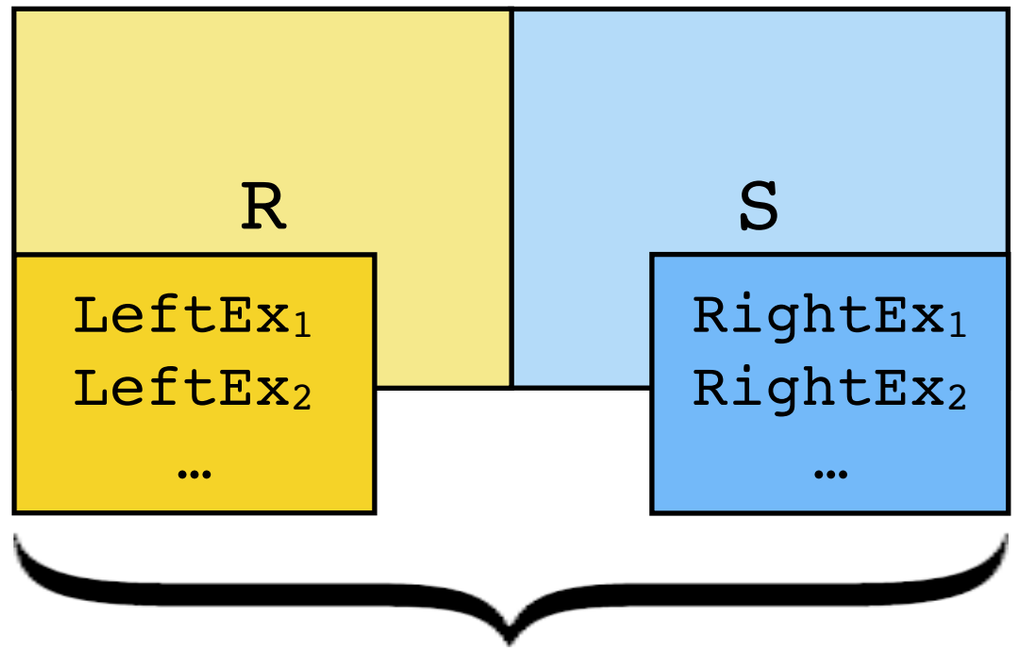
Lenses from Data Formats



Use the type syntax to direct the term creation!

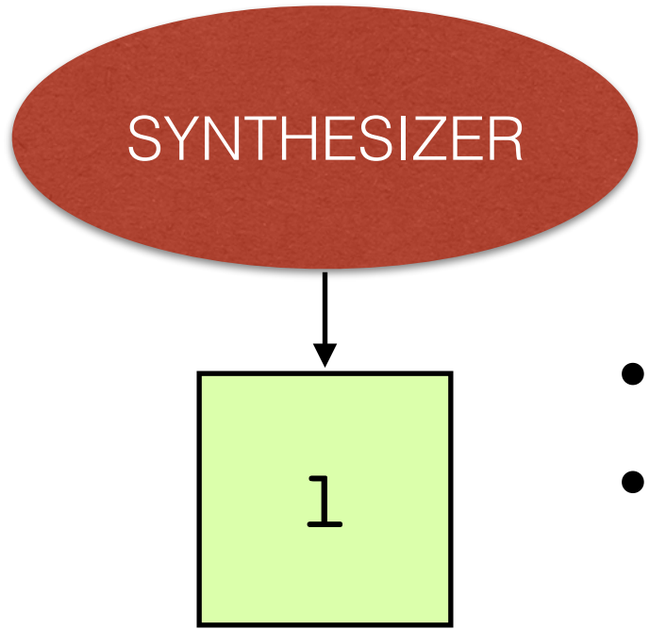
- `l : R <=> S`
- `l.get` $\texttt{LeftEx}_i$ = $\texttt{RightEx}_i$

Lenses from Data Formats



Use the type syntax to direct the term creation!

`? : R <=> S`



- `l : R <=> S`
- `l.get` $\texttt{LeftEx}_i$ = $\texttt{RightEx}_i$

Naive Algorithm

The Easy Cases!

$$? : R_1 \mid R_2 \iff S_1 \mid S_2$$

The Easy Cases!

$$\frac{?_1 : R_1 \mathrel{<=>} S_1 \qquad ?_2 : R_2 \mathrel{<=>} S_2}{\mathtt{or}(?_1,?_2) : R_1 \mid R_2 \mathrel{<=>} S_1 \mid S_2}$$

The Easy Cases!

$$\frac{?_1 : R_1 <=> S_1 \qquad ?_2 : R_2 <=> S_2}{\mathtt{or}(?_1,?_2) : R_1 \mid R_2 <=> S_1 \mid S_2}$$

The Easy Cases!

$$\frac{?_1 : R_1 \mathrel{<=>} S_1 \qquad ?_2 : R_2 \mathrel{<=>} S_2}{\mathrm{or}(?_1,?_2) : R_1 \mid R_2 \mathrel{<=>} S_1 \mid S_2}$$

$$? : R^* \mathrel{<=>} S^*$$

The Easy Cases!

$$\frac{?_1 : R_1 \Longleftrightarrow S_1 \qquad ?_2 : R_2 \Longleftrightarrow S_2}{\mathtt{or}(?_1,?_2) : R_1 \mid R_2 \Longleftrightarrow S_1 \mid S_2}$$

$$\frac{? : R \Longleftrightarrow S}{\mathtt{iterate}(?) : R^* \Longleftrightarrow S^*}$$

The Easy Cases!

$$\frac{?_1 : R_1 \mathrel{<=>} S_1 \qquad ?_2 : R_2 \mathrel{<=>} S_2}{\mathtt{or}(?_1,?_2) : R_1 \mid R_2 \mathrel{<=>} S_1 \mid S_2}$$

$$\frac{? : R \mathrel{<=>} S}{\mathtt{iterate}(?) : R^* \mathrel{<=>} S^*}$$

The Easy Cases!

$$\frac{?_1 : R_1 \mathrel{<=>} S_1 \qquad ?_2 : R_2 \mathrel{<=>} S_2}{\mathrm{or}(?_1,?_2) : R_1 \mid R_2 \mathrel{<=>} S_1 \mid S_2}$$

$$\frac{? : R \mathrel{<=>} S}{\mathrm{iterate}(?) : R^* \mathrel{<=>} S^*}$$

$$? : R_1.R_2 \mathrel{<=>} S_1.S_2$$

The Easy Cases!

$$\frac{?_1 : R_1 \Leftrightarrow S_1 \quad ?_2 : R_2 \Leftrightarrow S_2}{\mathrm{or}(?_1,?_2) : R_1 \mid R_2 \Leftrightarrow S_1 \mid S_2}$$

$$\frac{? : R \Leftrightarrow S}{\mathrm{iterate}(?) : R^* \Leftrightarrow S^*}$$

$$\frac{?_1 : R_1 \Leftrightarrow S_1 \quad ?_2 : R_2 \Leftrightarrow S_2}{\mathrm{concat}(?_1,?_2) : R_1.R_2 \Leftrightarrow S_1.S_2}$$

The Easy Cases!

$$\frac{?_1 : R_1 \Leftrightarrow S_1 \qquad ?_2 : R_2 \Leftrightarrow S_2}{\mathtt{or}(?_1,?_2) : R_1 \mid R_2 \Leftrightarrow S_1 \mid S_2}$$

$$\frac{?_1 : R_1 \Leftrightarrow S_1 \qquad ?_2 : R_2 \Leftrightarrow S_2}{\mathtt{concat}(?_1,?_2) : R_1.R_2 \Leftrightarrow S_1.S_2}$$

$$\frac{? : R \Leftrightarrow S}{\mathtt{iterate}(?) : R^* \Leftrightarrow S^*}$$

$$? : R_1.R_2 \Leftrightarrow S_1.S_2$$

The Easy Cases!

$$\frac{?_1 : R_1 \Leftrightarrow S_1 \quad ?_2 : R_2 \Leftrightarrow S_2}{\mathtt{or}(?_1,?_2) : R_1 \mid R_2 \Leftrightarrow S_1 \mid S_2}$$

$$\frac{?_1 : R_1 \Leftrightarrow S_1 \qquad ?_2 : R_2 \Leftrightarrow S_2}{\mathtt{concat}(?_1,?_2) : R_1.R_2 \Leftrightarrow S_1.S_2}$$

$$\frac{? : R \Leftrightarrow S}{\mathtt{iterate}(?) : R^* \Leftrightarrow S^*}$$

$$\frac{?_1 : R_1 \Leftrightarrow S_2 \quad ?_2 : R_2 \Leftrightarrow S_1}{\mathtt{swap}(?_1,?_2) : R_1.R_2 \Leftrightarrow S_1.S_2}$$

The Easy Cases!

$$\frac{?_1 : R_1 \Longleftrightarrow S_1 \qquad ?_2 : R_2 \Longleftrightarrow S_2}{\mathtt{or}(?_1,?_2) : R_1 \mid R_2 \Longleftrightarrow S_1 \mid S_2}$$

$$\frac{? : R \Longleftrightarrow S}{\mathtt{iterate}(?) : R^* \Longleftrightarrow S^*}$$

$$\frac{?_1 : R_1 \Longleftrightarrow S_1 \qquad ?_2 : R_2 \Longleftrightarrow S_2}{\mathtt{concat}(?_1,?_2) : R_1.R_2 \Longleftrightarrow S_1.S_2}$$

$$\frac{?_1 : R_1 \Longleftrightarrow S_2 \qquad ?_2 : R_2 \Longleftrightarrow S_1}{\mathtt{swap}(?_1,?_2) : R_1.R_2 \Longleftrightarrow S_1.S_2}$$

$$? : s \Longleftrightarrow t$$

The Easy Cases!

$$\frac{?_1 : R_1 \Longleftrightarrow S_1 \qquad ?_2 : R_2 \Longleftrightarrow S_2}{\mathtt{or}(?_1,?_2) : R_1 \mid R_2 \Longleftrightarrow S_1 \mid S_2}$$

$$\frac{?_1 : R_1 \Longleftrightarrow S_1 \qquad ?_2 : R_2 \Longleftrightarrow S_2}{\mathtt{concat}(?_1,?_2) : R_1.R_2 \Longleftrightarrow S_1.S_2}$$

$$\frac{? : R \Longleftrightarrow S}{\mathtt{iterate}(?) : R^* \Longleftrightarrow S^*}$$

$$\frac{?_1 : R_1 \Longleftrightarrow S_2 \qquad ?_2 : R_2 \Longleftrightarrow S_1}{\mathtt{swap}(?_1,?_2) : R_1.R_2 \Longleftrightarrow S_1.S_2}$$

$$\frac{}{\mathtt{const}(s,t) : s \Longleftrightarrow t}$$

The Hard Cases

$$\frac{?_1 : R_1 \Longleftrightarrow S_1 \qquad ?_2 : R_2 \Longleftrightarrow S_2}{\mathtt{or}(?_1,?_2) : R_1 \mid R_2 \Longleftrightarrow S_1 \mid S_2}$$

$$\frac{?_1 : R_1 \Longleftrightarrow S_1 \qquad ?_2 : R_2 \Longleftrightarrow S_2}{\mathtt{concat}(?_1,?_2) : R_1.R_2 \Longleftrightarrow S_1.S_2}$$

$$\frac{?_1 : R_1 \Longleftrightarrow S_2 \qquad ?_2 : R_2 \Longleftrightarrow S_1}{\mathtt{swap}(?_1,?_2) : R_1.R_2 \Longleftrightarrow S_1.S_2}$$

$$\frac{? : R \Longleftrightarrow S}{\mathtt{iterate}(?) : R^* \Longleftrightarrow S^*}$$

$$\frac{}{\mathtt{const}(s,t) : s \Longleftrightarrow t}$$

$$? : R_1 \,.\, (R_2 \mid R_3) \Longleftrightarrow (R_1 \,.\, R_2) \mid (R_1 \,.\, R_3)$$

The Hard Cases

$$\frac{?_1 : R_1 \Longleftrightarrow S_1 \qquad ?_2 : R_2 \Longleftrightarrow S_2}{\mathtt{or}(?_1,?_2) : R_1 \mid R_2 \Longleftrightarrow S_1 \mid S_2}$$

$$\frac{?_1 : R_1 \Longleftrightarrow S_1 \qquad ?_2 : R_2 \Longleftrightarrow S_2}{\mathtt{concat}(?_1,?_2) : R_1.R_2 \Longleftrightarrow S_1.S_2}$$

$$\frac{?_1 : R_1 \Longleftrightarrow S_2 \qquad ?_2 : R_2 \Longleftrightarrow S_1}{\mathtt{swap}(?_1,?_2) : R_1.R_2 \Longleftrightarrow S_1.S_2}$$

$$\frac{? : R \Longleftrightarrow S}{\mathtt{iterate}(?) : R^* \Longleftrightarrow S^*}$$

$$\frac{}{\mathtt{const}(s,t) : s \Longleftrightarrow t}$$

$$? : \frac{(R_1 \,.\, R_2) \mid (R_1 \,.\, R_3) \Longleftrightarrow (R_1 \,.\, R_2) \mid (R_1 \,.\, R_3)}{? : R_1 \,.\, (R_2 \mid R_3) \Longleftrightarrow (R_1 \,.\, R_2) \mid (R_1 \,.\, R_3)}$$

The Hard Cases

$$\frac{?_1 : R_1 \Longleftrightarrow S_1 \qquad ?_2 : R_2 \Longleftrightarrow S_2}{\mathtt{or}(?_1,?_2) : R_1 \mid R_2 \Longleftrightarrow S_1 \mid S_2}$$

$$\frac{?_1 : R_1 \Longleftrightarrow S_1 \qquad ?_2 : R_2 \Longleftrightarrow S_2}{\mathtt{concat}(?_1,?_2) : R_1.R_2 \Longleftrightarrow S_1.S_2}$$

$$\frac{?_1 : R_1 \Longleftrightarrow S_2 \qquad ?_2 : R_2 \Longleftrightarrow S_1}{\mathtt{swap}(?_1,?_2) : R_1.R_2 \Longleftrightarrow S_1.S_2}$$

$$\frac{? : R \Longleftrightarrow S}{\mathtt{iterate}(?) : R^* \Longleftrightarrow S^*}$$

$$\frac{}{\mathtt{const}(s,t) : s \Longleftrightarrow t}$$

$$? : R_1 \,.\, (R_2 \,.\, R_3) \Longleftrightarrow (R_1 \,.\, R_2) \,.\, R_3$$

The Hard Cases

$$\frac{?_1 : R_1 \Longleftrightarrow S_1 \qquad ?_2 : R_2 \Longleftrightarrow S_2}{\mathtt{or}(?_1,?_2) : R_1 \mid R_2 \Longleftrightarrow S_1 \mid S_2}$$

$$\frac{?_1 : R_1 \Longleftrightarrow S_1 \qquad ?_2 : R_2 \Longleftrightarrow S_2}{\mathtt{concat}(?_1,?_2) : R_1.R_2 \Longleftrightarrow S_1.S_2}$$

$$\frac{?_1 : R_1 \Longleftrightarrow S_2 \qquad ?_2 : R_2 \Longleftrightarrow S_1}{\mathtt{swap}(?_1,?_2) : R_1.R_2 \Longleftrightarrow S_1.S_2}$$

$$\frac{? : R \Longleftrightarrow S}{\mathtt{iterate}(?) : R^* \Longleftrightarrow S^*}$$

$$\frac{}{\mathtt{const}(s,t) : s \Longleftrightarrow t}$$

$$\frac{? : R_1 \,.\, (R_2 \,.\, R_3) \Longleftrightarrow R_1 \,.\, (R_2 \,.\, R_3)}{? : R_1 \,.\, (R_2 \,.\, R_3) \Longleftrightarrow (R_1 \,.\, R_2) \,.\, R_3}$$

The Hard Cases

$$\frac{?_1 : R_1 \Longleftrightarrow S_1 \qquad ?_2 : R_2 \Longleftrightarrow S_2}{\mathtt{or}(?_1,?_2) : R_1 \mid R_2 \Longleftrightarrow S_1 \mid S_2}$$

$$\frac{?_1 : R_1 \Longleftrightarrow S_1 \qquad ?_2 : R_2 \Longleftrightarrow S_2}{\mathtt{concat}(?_1,?_2) : R_1.R_2 \Longleftrightarrow S_1.S_2}$$

$$\frac{?_1 : R_1 \Longleftrightarrow S_2 \qquad ?_2 : R_2 \Longleftrightarrow S_1}{\mathtt{swap}(?_1,?_2) : R_1.R_2 \Longleftrightarrow S_1.S_2}$$

$$\frac{? : R \Longleftrightarrow S}{\mathtt{iterate}(?) : R^* \Longleftrightarrow S^*}$$

$$\frac{}{\mathtt{const}(s,t) : s \Longleftrightarrow t}$$

$$\frac{? : ?_1 \Longleftrightarrow ?_2 \qquad ?_1 \equiv R \qquad ?_2 \equiv S}{? : R \Longleftrightarrow S}$$

The Hard Cases

$$\frac{?_1 : R_1 \Longleftrightarrow S_1 \qquad ?_2 : R_2 \Longleftrightarrow S_2}{\mathtt{or}(?_1,?_2) : R_1 \mid R_2 \Longleftrightarrow S_1 \mid S_2}$$

$$\frac{? : R \Longleftrightarrow S}{\mathtt{iterate}(?) : R^* \Longleftrightarrow S^*}$$

$$\frac{}{\mathtt{const}(s,t) : s \Longleftrightarrow t}$$

$$\frac{?_1 : R_1 \Longleftrightarrow S_1 \qquad ?_2 : R_2 \Longleftrightarrow S_2}{\mathtt{concat}(?_1,?_2) : R_1.R_2 \Longleftrightarrow S_1.S_2}$$

$$\frac{?_1 : R_1 \Longleftrightarrow S_2 \qquad ?_2 : R_2 \Longleftrightarrow S_1}{\mathtt{swap}(?_1,?_2) : R_1.R_2 \Longleftrightarrow S_1.S_2}$$

$$\frac{? : ?_1 \Longleftrightarrow ?_2 \qquad ?_1 \equiv R \qquad ?_2 \equiv S}{? : R \Longleftrightarrow S}$$

$$? : R1 \,.\, R2 \,.\, R3 \,.\, R4 \Longleftrightarrow R3 \,.\, R1 \,.\, R4 \,.\, R2$$

The Hard Cases

$$\frac{?_1 : R_1 \mathrel{<=>} S_1 \qquad ?_2 : R_2 \mathrel{<=>} S_2}{\mathtt{or}(?_1,?_2) : R_1 \mid R_2 \mathrel{<=>} S_1 \mid S_2}$$

$$\frac{? : R \mathrel{<=>} S}{\mathtt{iterate}(?) : R^* \mathrel{<=>} S^*}$$

$$\frac{}{\mathtt{const}(s,t) : s \mathrel{<=>} t}$$

$$\frac{?_1 : R_1 \mathrel{<=>} S_1 \qquad ?_2 : R_2 \mathrel{<=>} S_2}{\mathtt{concat}(?_1,?_2) : R_1.R_2 \mathrel{<=>} S_1.S_2}$$

$$\frac{?_1 : R_1 \mathrel{<=>} S_2 \qquad ?_2 : R_2 \mathrel{<=>} S_1}{\mathtt{swap}(?_1,?_2) : R_1.R_2 \mathrel{<=>} S_1.S_2}$$

$$\frac{? : ?_1 \mathrel{<=>} ?_2 \qquad ?_1 \equiv R \qquad ?_2 \equiv S}{? : R \mathrel{<=>} S}$$

$$\frac{?_1 : R \mathrel{<=>} ? \qquad ?_2 : ? \mathrel{<=>} S}{?_1;?_2 : R \mathrel{<=>} S}$$

The Hard Cases

$$\frac{?_1 : R_1 \mathrel{<=>} S_1 \qquad ?_2 : R_2 \mathrel{<=>} S_2}{\mathtt{or}(?_1,?_2) : R_1 \mid R_2 \mathrel{<=>} S_1 \mid S_2}$$

$$\frac{? : R \mathrel{<=>} S}{\mathtt{iterate}(?) : R^* \mathrel{<=>} S^*}$$

$$\frac{}{\mathtt{const}(s,t) : s \mathrel{<=>} t}$$

$$\frac{?_1 : R_1 \mathrel{<=>} S_1 \qquad ?_2 : R_2 \mathrel{<=>} S_2}{\mathtt{concat}(?_1,?_2) : R_1.R_2 \mathrel{<=>} S_1.S_2}$$

$$\frac{?_1 : R_1 \mathrel{<=>} S_2 \qquad ?_2 : R_2 \mathrel{<=>} S_1}{\mathtt{swap}(?_1,?_2) : R_1.R_2 \mathrel{<=>} S_1.S_2}$$

$$\frac{? : ?_1 \mathrel{<=>} ?_2 \qquad ?_1 \equiv R \qquad ?_2 \equiv S}{? : R \mathrel{<=>} S}$$

$$\frac{?_1 : R \mathrel{<=>} ? \qquad ?_2 : ? \mathrel{<=>} S}{?_1;?_2 : R \mathrel{<=>} S}$$

The Hard Cases

$$\frac{?_1 : R_1 \Longleftrightarrow S_1 \qquad ?_2 : R_2 \Longleftrightarrow S_2}{\mathtt{or}(?_1,?_2) : R_1 \mid R_2 \Longleftrightarrow S_1 \mid S_2}$$

$$\frac{? : R \Longleftrightarrow S}{\mathtt{iterate}(?) : R^* \Longleftrightarrow S^*}$$

$$\frac{}{\mathtt{const}(s,t) : s \Longleftrightarrow t}$$

$$\frac{?_1 : R_1 \Longleftrightarrow S_1 \qquad ?_2 : R_2 \Longleftrightarrow S_2}{\mathtt{concat}(?_1,?_2) : R_1.R_2 \Longleftrightarrow S_1.S_2}$$

$$\frac{?_1 : R_1 \Longleftrightarrow S_2 \qquad ?_2 : R_2 \Longleftrightarrow S_1}{\mathtt{swap}(?_1,?_2) : R_1.R_2 \Longleftrightarrow S_1.S_2}$$

$$\frac{? : ?_1 \Longleftrightarrow ?_2 \qquad ?_1 \equiv R \qquad ?_2 \equiv S}{? : R \Longleftrightarrow S}$$

$$\frac{?_1 : R \Longleftrightarrow ? \qquad ?_2 : ? \Longleftrightarrow S}{?_1;?_2 : R \Longleftrightarrow S}$$

HUGE SEARCH SPACE

Approach

- Reduce Equivalences
  - Alternate language of types: *DNF Regular Expressions*
  - Restrict where retyping can be used
- Remove Composition
  - Alternate language of terms: *DNF Lenses*
  - DNF lenses closed under composition

DNF Regular Expressions and DNF Lenses

DNF Regular Expressions

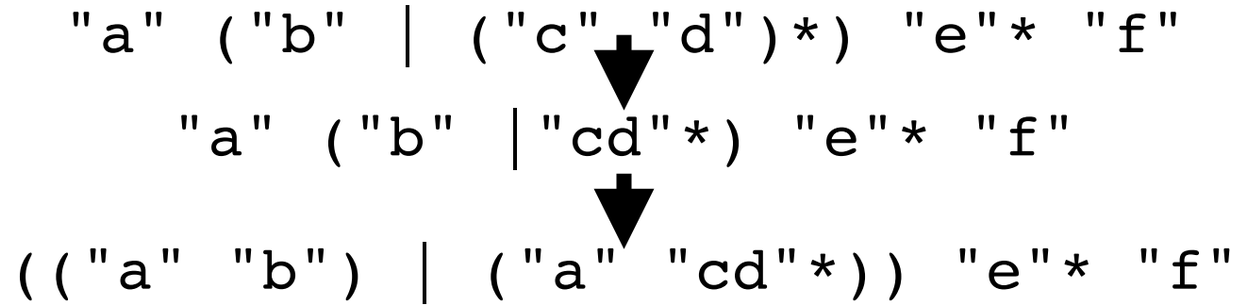
- Fully distributed regular expression
- No associativity information
- Base strings enforced at fixed locations

Converting to DNF Regular Expressions ⇓

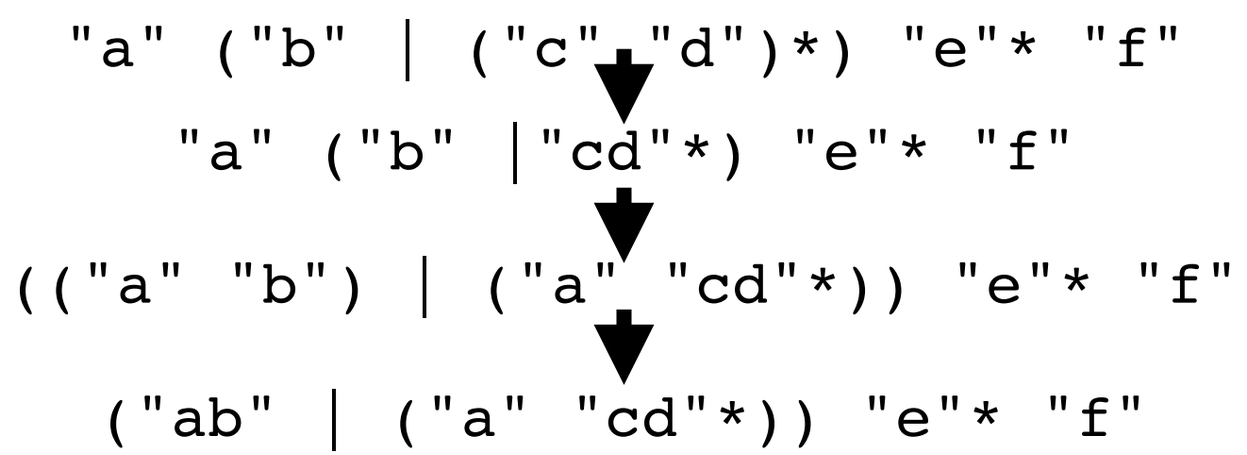
```
"a" ("b" | ("c" "d")*) "e"* "f"
```

Converting to DNF Regular Expressions ⇓

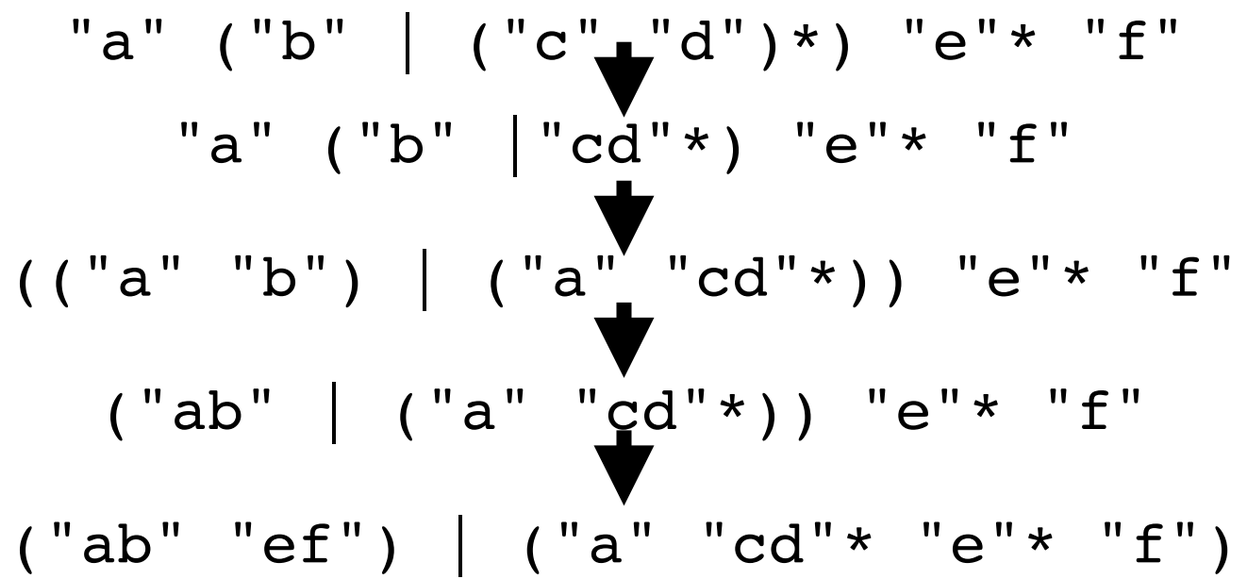
Converting to DNF Regular Expressions ⇓



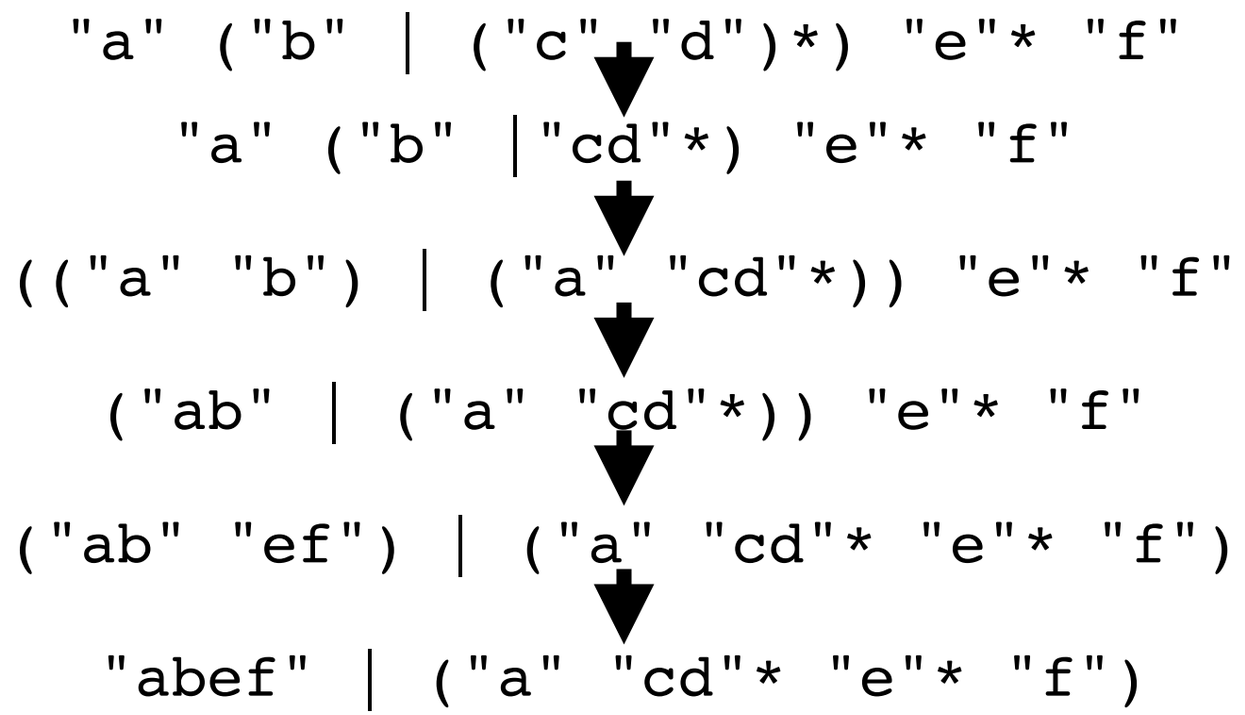
Converting to DNF Regular Expressions ⇓



Converting to DNF Regular Expressions ⇓



Converting to DNF Regular Expressions ⇓



Converting to DNF Regular Expressions ⇓

`"abef" | ("a" "cd"* "e"* "f")`

Atoms

$A ::= DR*$	iterate

Sequences

$SQ ::= \langle s_0; A_1; \ldots; A_n; s_n \rangle$	sequence

DNF Regular Expressions

$DR ::= \{SQ_1; \ldots; SQ_n\}$	choose

Converting to DNF Regular Expressions ⇓

```
"abef" | ("a" "cd"* "e"* "f")
```

```
{"abef" ; "a" "cd"* "e"* "f"}
```

Atoms

$A ::= DR^*$	iterate

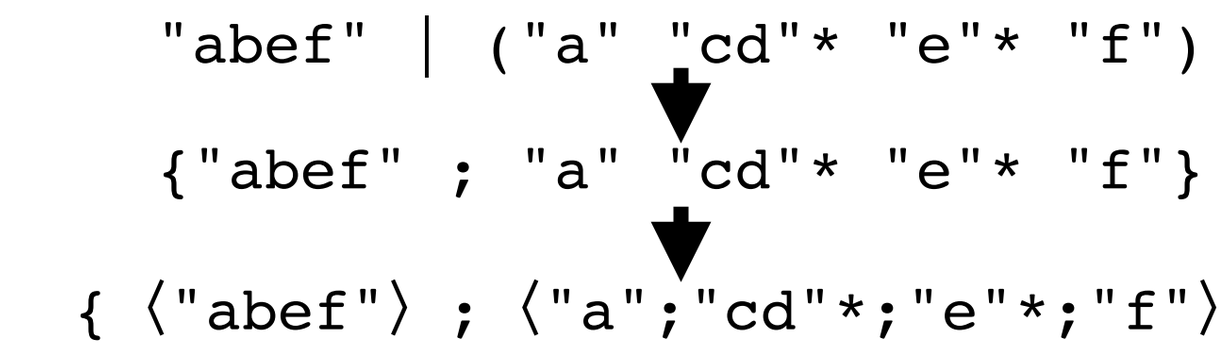
Sequences

$SQ ::= \langle s_0; A_1; \ldots; A_n; s_n \rangle$	sequence

DNF Regular Expressions

$DR ::= \{SQ_1; \ldots; SQ_n\}$	choose

Converting to DNF Regular Expressions ⇓



Atoms

A ::= DR* iterate

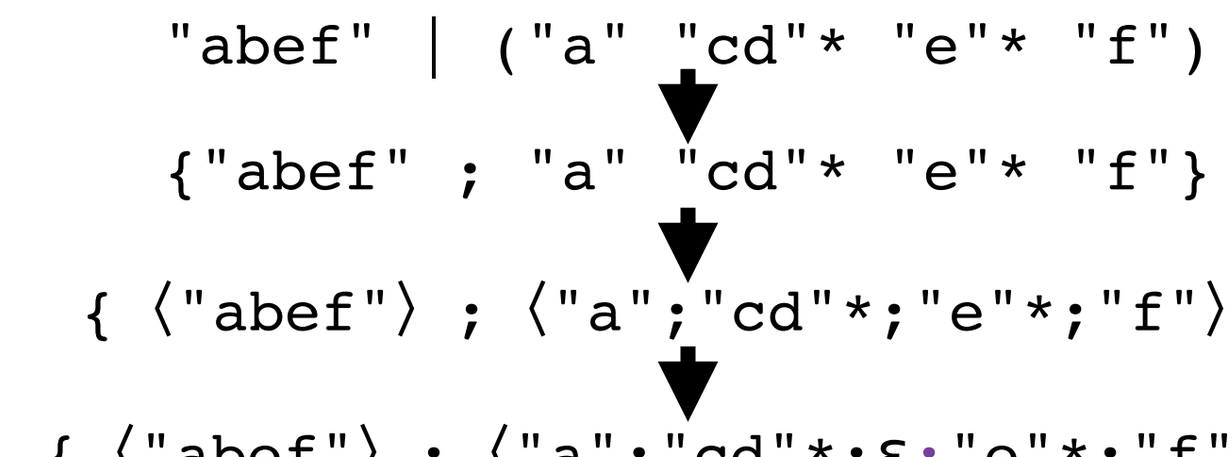
Sequences

SQ ::= ⟨s_0;A_1;...;A_n;s_n⟩ sequence

DNF Regular Expressions

DR ::= {SQ_1;...;SQ_n} choose

Converting to DNF Regular Expressions ⇓



Atoms

A ::= DR* iterate

Sequences

SQ ::= ⟨s_0;A_1;...;A_n;s_n⟩ sequence

DNF Regular Expressions

DR ::= {SQ_1;...;SQ_n} choose

Converting to DNF Regular Expressions ⇓

```
"a" ("b" | ("c" "d")*) "e"* "f"
```

```
{ ⟨"abef"⟩ ; ⟨"a";"cd"*;ε;"e"*;"f"⟩ }
```

Converting to DNF Regular Expressions ⇓

```
"a" ("b" | ("c" "d")*) "e"* "f"

"a" ε ("b" |"cd"* | ∅) (“e"* “f")


{ ⟨"abef"⟩ ; ⟨"a";"cd"*;ε;"e"*;"f"⟩ }
```

Converting to DNF Regular Expressions ⇓

```
"a" ("b" | ("c" "d")*) "e"* "f"

"a" ε ("b" |"cd"* | ∅) (“e"* “f")

⋮

{ ⟨"abef"⟩ ; ⟨"a";"cd"*;ε;"e"*;"f"⟩ }
```

Converting to DNF Regular Expressions ⇓

```
"a" ("b" | ("c" "d")*) "e"* "f"

"a" ε ("b" |"cd"* | ∅) (“e"* “f")

⋮

{ ⟨"abef"⟩ ; ⟨"a";"cd"*;ε;"e"*;"f"⟩ }
```

```
"abef" | ("a" (ε | ("cd" "cd"*)) "e"* "f")

⋮

{ ⟨"abef"⟩ ; ⟨"a";"e"*;"f"⟩ ;
  ⟨"acd";"cd"*;ε;"e"*;"f"⟩ }
```

Converting to DNF Regular Expressions ⇓

```
"a" ("b" | ("c" "d")*) "e"* "f"

"a" ε ("b" |"cd"* | ∅) (“e"* “f")

⋮

{ ⟨"abef"⟩ ; ⟨"a";"cd"*;ε;"e"*;"f"⟩ }
```

```
"abef" | ("a" (ε | ("cd" "cd"*)) "e"* "f")

⋮

{ ⟨"abef"⟩ ; ⟨"a";"e"*;"f"⟩ ;
  ⟨"acd";"cd"*;ε;"e"*;"f"⟩ }
```

```
("a" "cd"* "e"* "f") | "abef"

⋮

{ ⟨"a";"cd"*;ε;"e"*;"f"⟩ ; ⟨"abef"⟩ }
```

DNF Regular Expression Rewrites

- A | B ≡ B | A not part of DNF rewrites
  - DNF Lens terms address these equivalences
- `R* ≡ ε | (R.R*)` and `R* ≡ ε | (R*.R)` ARE part of DNF rewrites
  - `R* → ε | (R.R*)`
  - `R* → ε | (R*.R)`
  - `DR →* DR'`

DNF Lenses and Subcomponents

`dl :: D <=> E`

Atom Lenses

`al` ::= `iterate(dl)` iterate

Sequence Lenses

`sl` ::= `(` $\langle$`(`s_0`,`t_0`);`al_1`;...;`al_n`;(`s_n`,`t_n`)`$\rangle$ `,σ)` sequence

DNF Lenses

`dl` ::= `({`sl_1`;...;`sl_n`},σ)` choose

Iterate Lens

```
author={Hofmann, Martin
  and Pierce, Benjamin
  and Wagner, Daniel}
```

```
%A Martin Hofmann
%A Benjamin Pierce
%A Daniel Wagner
```

Iterate Lens

```
author={Hofmann, Martin
  and Pierce, Benjamin
  and Wagner, Daniel}
```

```
%A Martin Hofmann
%A Benjamin Pierce
%A Daniel Wagner
```

```
iterate(dl) :: D* <=> E*
```

Iterate Lens

```
author={Hofmann, Martin
  and Pierce, Benjamin
  and Wagner, Daniel}
```

```
%A Martin Hofmann
%A Benjamin Pierce
%A Daniel Wagner
```

```
iterate(dl) :: D* <=> E*
```

$$\frac{\texttt{dl :: D <=> E} \quad \textit{side conditions}}{\texttt{iterate(l) :: D* <=> E*}}$$

Sequence Lens

```
author={Kleene, Stephen Cole}
```

```
%A Stephen Cole Kleene
```

Sequence Lens

```
author={Kleene, Stephen Cole}
```

```
%A Stephen Cole Kleene
```

$$(\langle ("\texttt{author=}","\texttt{\%A}");al_1;(",",\varepsilon);al_2;("\}",\varepsilon)\rangle ,[1;0]) ::$$

$$\langle "\texttt{author=}";a_1;",";a_2;"\}"\rangle <=> \langle "\texttt{\%A}";b_2;\varepsilon;b_1;\varepsilon\rangle$$

Sequence Lens

```
author={Kleene, Stephen Cole}        %A Stephen Cole Kleene
```

$$(\ \langle (\text{"author="},\text{"\%A"});al_1;(\text{","},\varepsilon);al_2;(\text{"}\}\text{"},\varepsilon)\rangle\ ,[1;0])\ ::$$

$$\langle \text{"author="};a_1;\text{","};a_2;\text{"}\}\text{"}\rangle \mathrel{<=>} \langle \text{"\%A"};b_2;\varepsilon;b_1;\varepsilon\rangle$$

$$al_1 :: a_1 \mathrel{<=>} b_1$$
$$\ldots$$
$$al_n :: a_n \mathrel{<=>} b_n$$
side conditions

$$(\ \langle (s_0,t_0);al_1;\ldots;al_n;(s_n,t_n)\rangle\ ,\sigma)\ ::$$

$$\langle s_0;a_1;\ldots;a_n;s_n\rangle \mathrel{<=>} \langle t_0;b_{\sigma 1};\ldots;b_{\sigma n};t_n\rangle$$

DNF Lens

```
                ε                                %A Stephen Cole Kleene
author={Kleene, Stephen Cole}                            ε
```

DNF Lens

```
                ε                                %A Stephen Cole Kleene
author={Kleene, Stephen Cole}                            ε
```

$$(\{l_1, l_2\}, [1,0]) :: \{R_1; R_2\} \Longleftrightarrow \{S_2; S_1\}$$

DNF Lens

```
                  ε                                %A Stephen Cole Kleene
author={Kleene, Stephen Cole}                              ε
```

$$(\{l_1, l_2\}, [1,0]) :: \{R_1; R_2\} \Longleftrightarrow \{S_2; S_1\}$$

$$sl_1 :: sq_1 \Longleftrightarrow tq_1$$
$$\ldots$$
$$sl_n :: sq_n \Longleftrightarrow tq_n$$
side conditions

$$(\{sl_1; \ldots; sl_n\}, \sigma) ::$$

$$\{sq_0; \ldots; sq_n\} \Longleftrightarrow \{tq_{\sigma 1}; \ldots; tq_{\sigma n}\}$$

Rewrites at End

```
dl :: D' <=> E'      D →* D'      E →* E'
-------------------------------------------
              dl :: D <=> E
```

Rewrites at End

```
dl :: D' <=> E'      D →* D'      E →* E'
-------------------------------------------
              dl :: D <=> E
```

AND NO COMPOSITION!

Equivalence Statements

Soundness

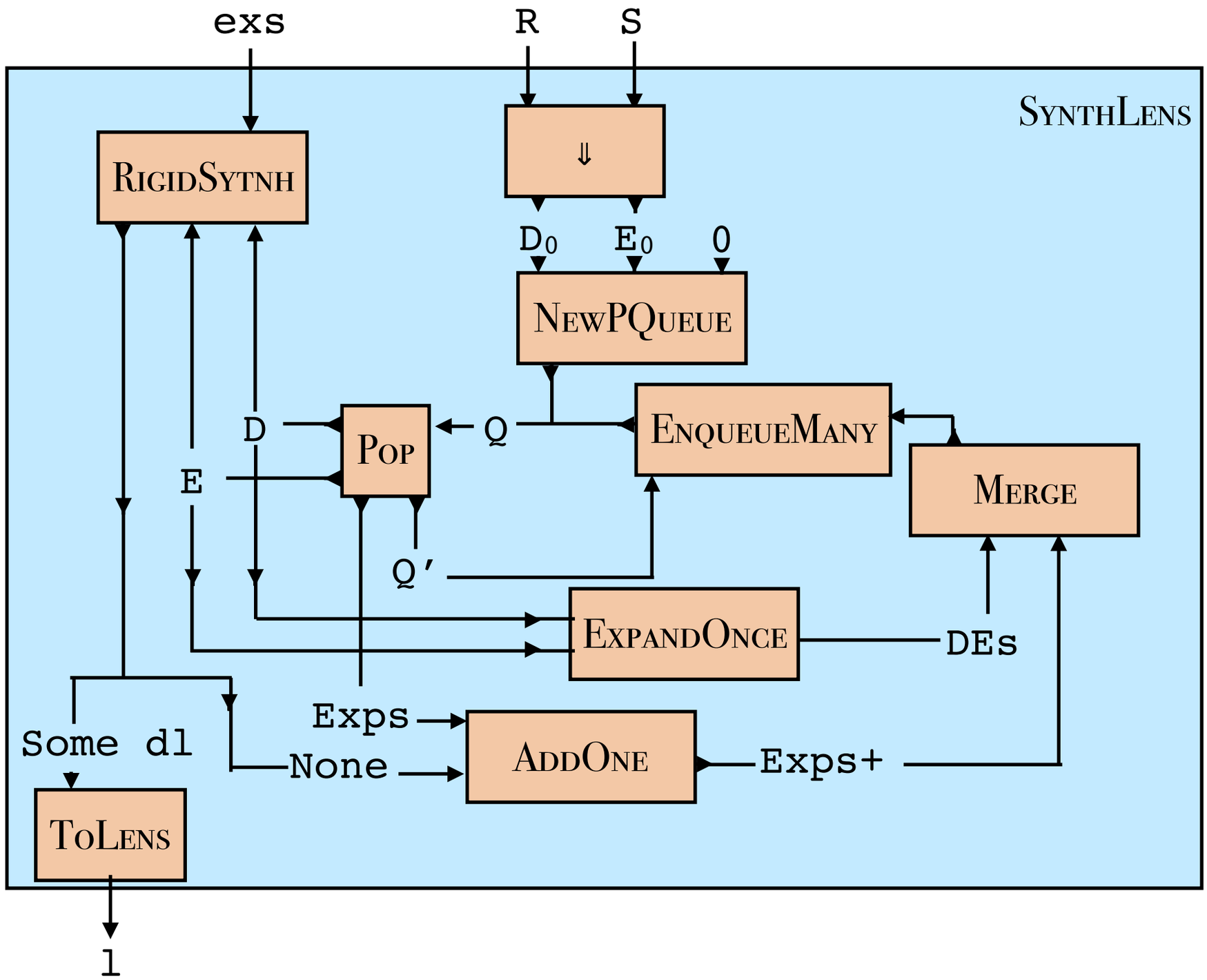
If `dl :: DR <=> DS` then there exist `l`, `R`, `S` such that
`l : R <=> S`, with the same semantics

Completeness

If `l : R <=> S` then there exists `dl` such that
`dl :: ⇓(R) <=> ⇓(S)`, with the same semantics

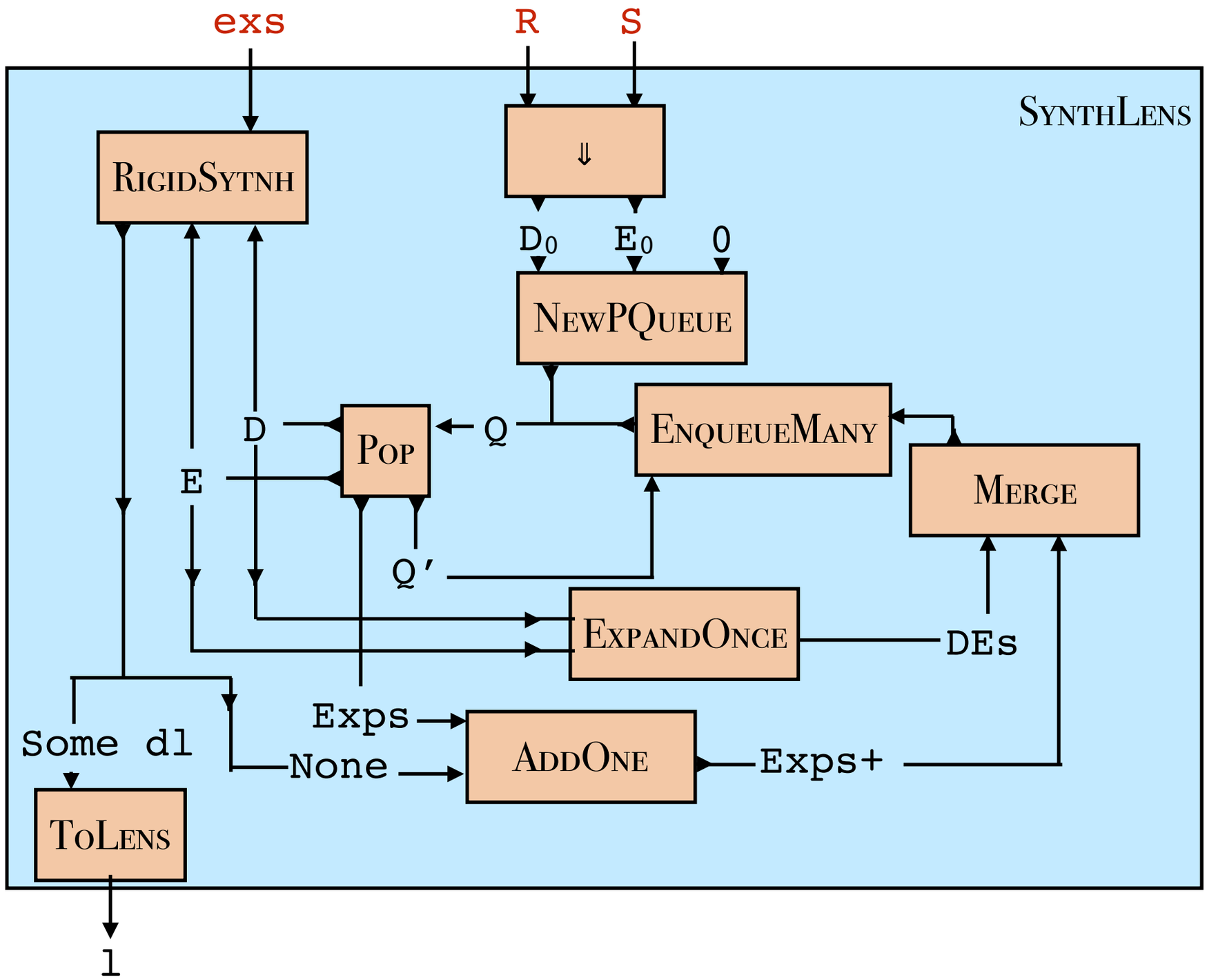
Synthesis Algorithm

Synthesis Diagram



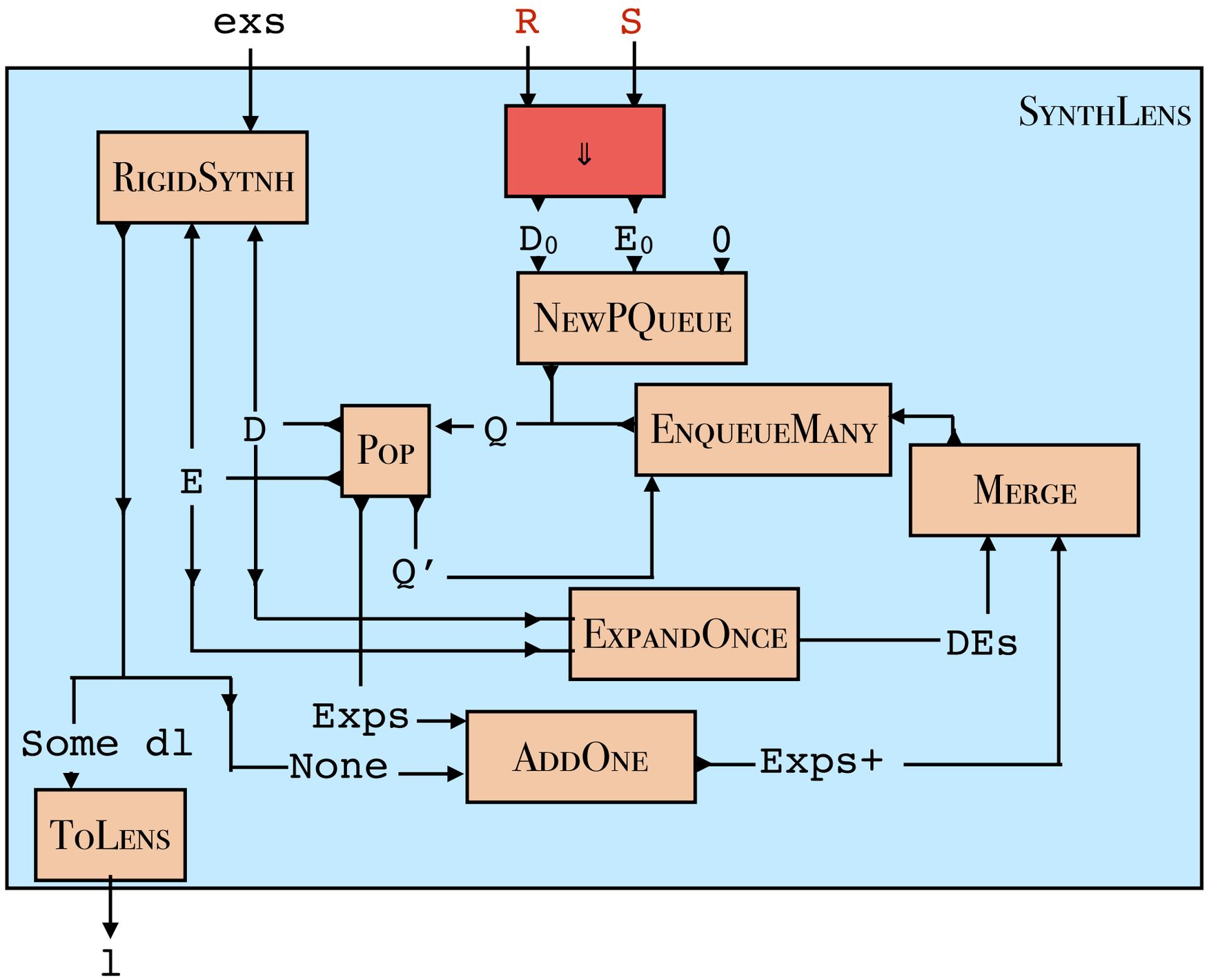

```
exs : string*string
      list
R : RX
S : RX
D0 : DNF RX
E0 : DNF RX
D : DNF RX
E : DNF RX
0 : int
Exps : int
Exps+ : int
Q : Pri Queue
Q′ : Pri Queue
Some dl : DNF Lens
          option
None : DNF Lens
       option
l : Lens
DEs: DNF RX * DNF RX
     list
```

Synthesis Diagram



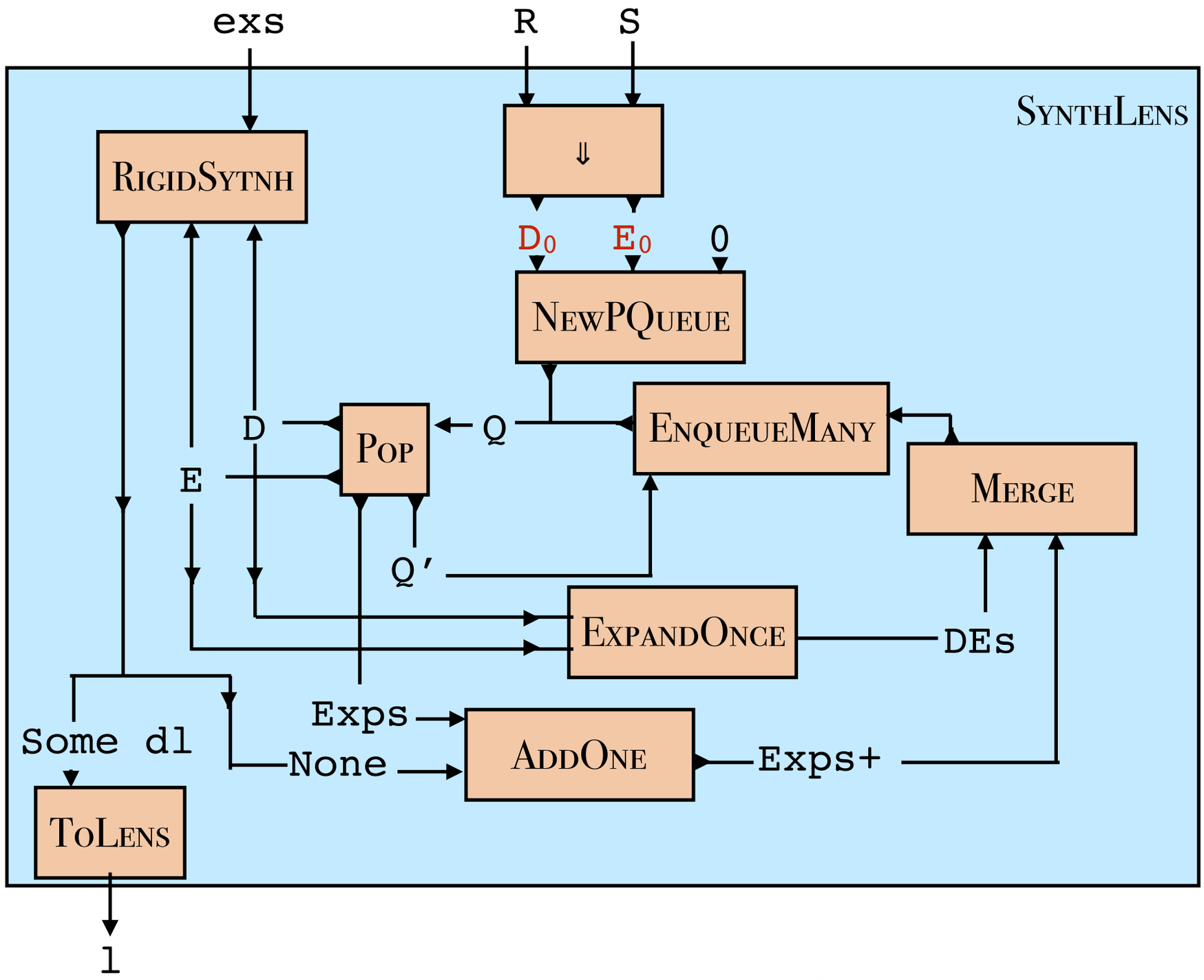

```
exs : string*string
      list
R : RX
S : RX
D0 : DNF RX
E0 : DNF RX
D : DNF RX
E : DNF RX
0 : int
Exps : int
Exps+ : int
Q : Pri Queue
Q′ : Pri Queue
Some dl : DNF Lens
          option
None : DNF Lens
       option
l : Lens
DEs: DNF RX * DNF RX
     list
```

Synthesis Diagram



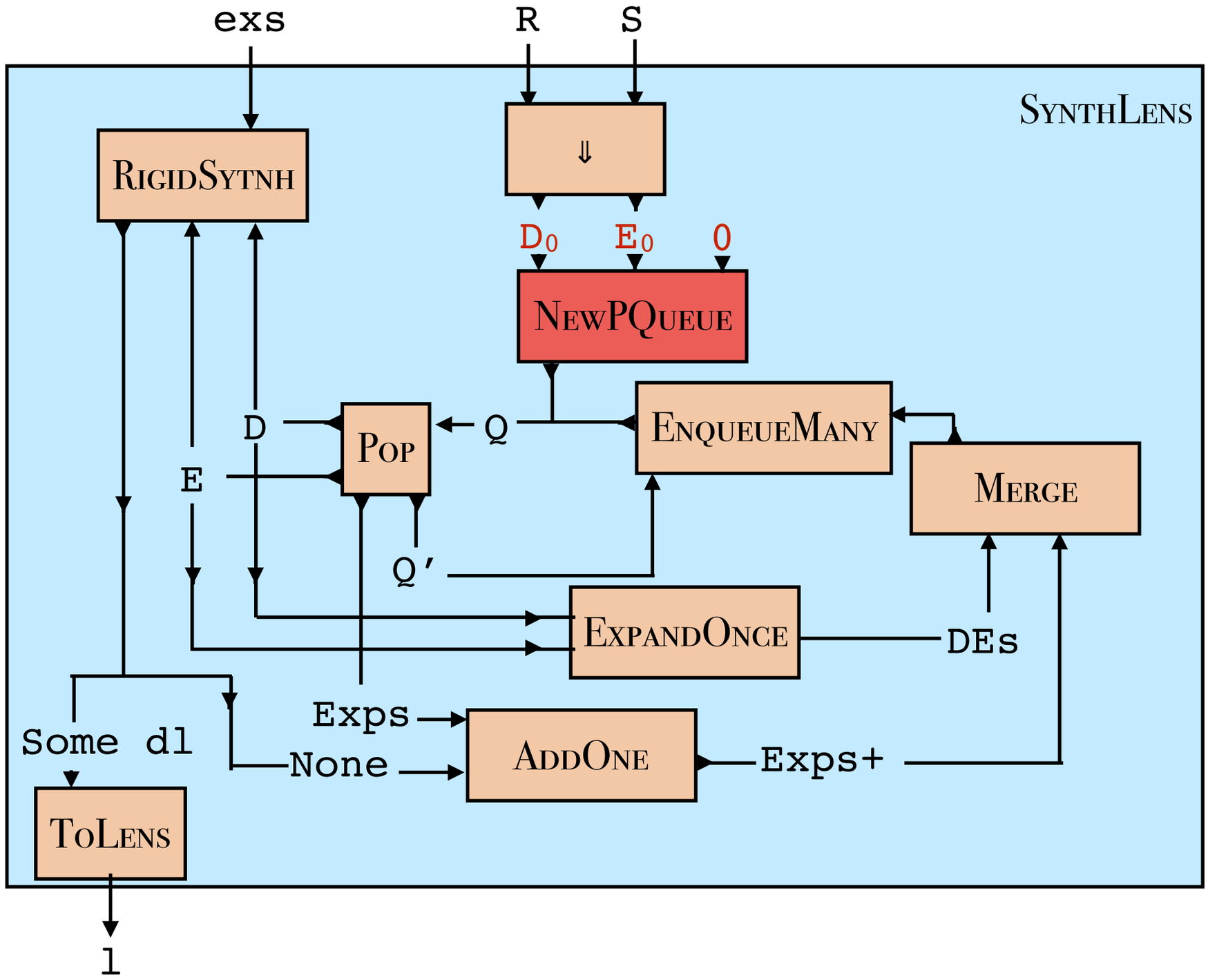

```
exs : string*string
      list
R : RX
S : RX
D0 : DNF RX
E0 : DNF RX
D : DNF RX
E : DNF RX
0 : int
Exps : int
Exps+ : int
Q : Pri Queue
Q′ : Pri Queue
Some dl : DNF Lens
          option
None : DNF Lens
       option
l : Lens
DEs: DNF RX * DNF RX
     list
```

Synthesis Diagram



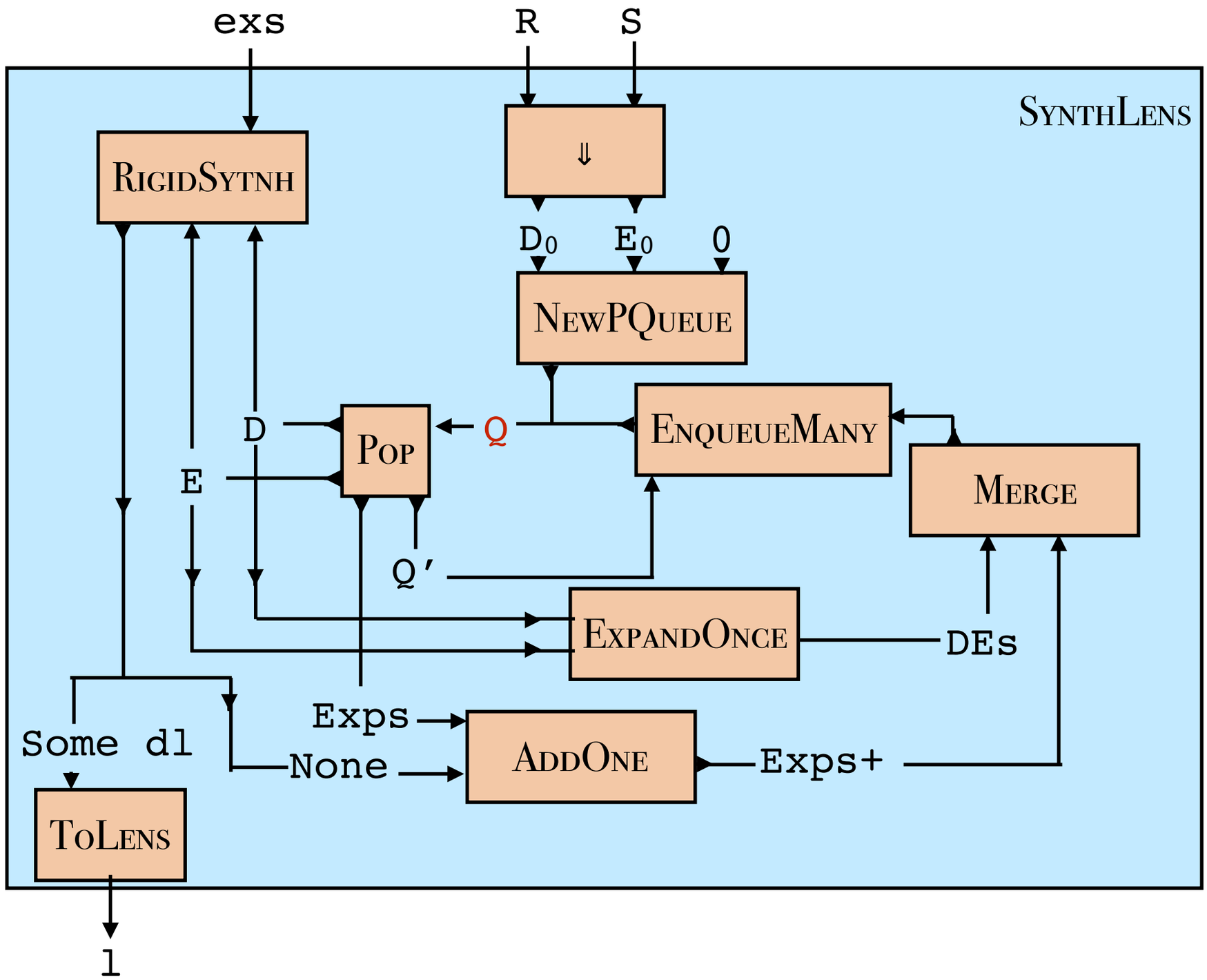

```
exs : string*string
      list
R : RX
S : RX
D0 : DNF RX
E0 : DNF RX
D : DNF RX
E : DNF RX
0 : int
Exps : int
Exps+ : int
Q : Pri Queue
Q′ : Pri Queue
Some dl : DNF Lens
          option
None : DNF Lens
       option
l : Lens
DEs: DNF RX * DNF RX
     list
```

Synthesis Diagram



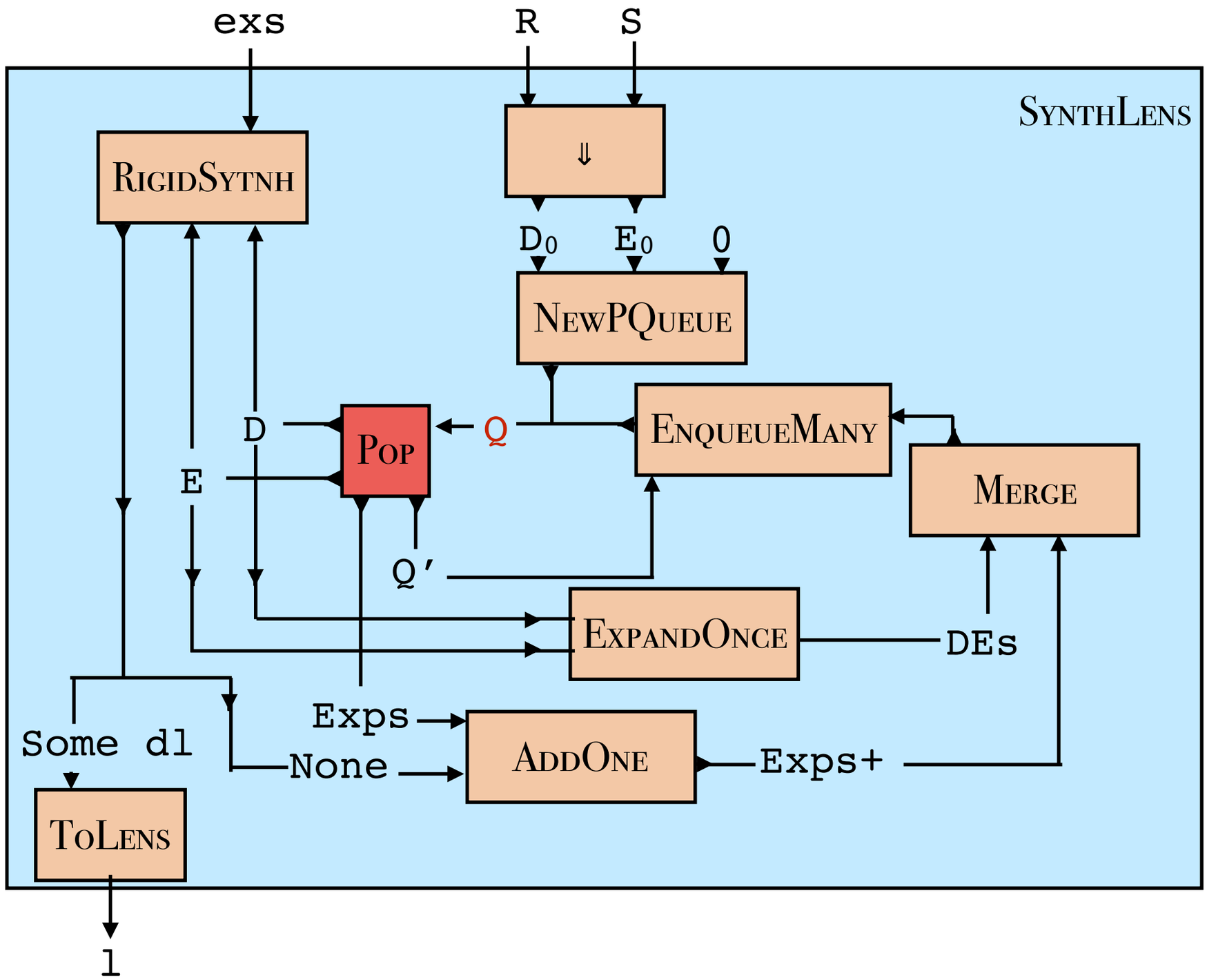

```
exs : string*string
      list
R : RX
S : RX
D0 : DNF RX
E0 : DNF RX
D : DNF RX
E : DNF RX
0 : int
Exps : int
Exps+ : int
Q : Pri Queue
Q′ : Pri Queue
Some dl : DNF Lens
          option
None : DNF Lens
       option
l : Lens
DEs: DNF RX * DNF RX
     list
```

Synthesis Diagram



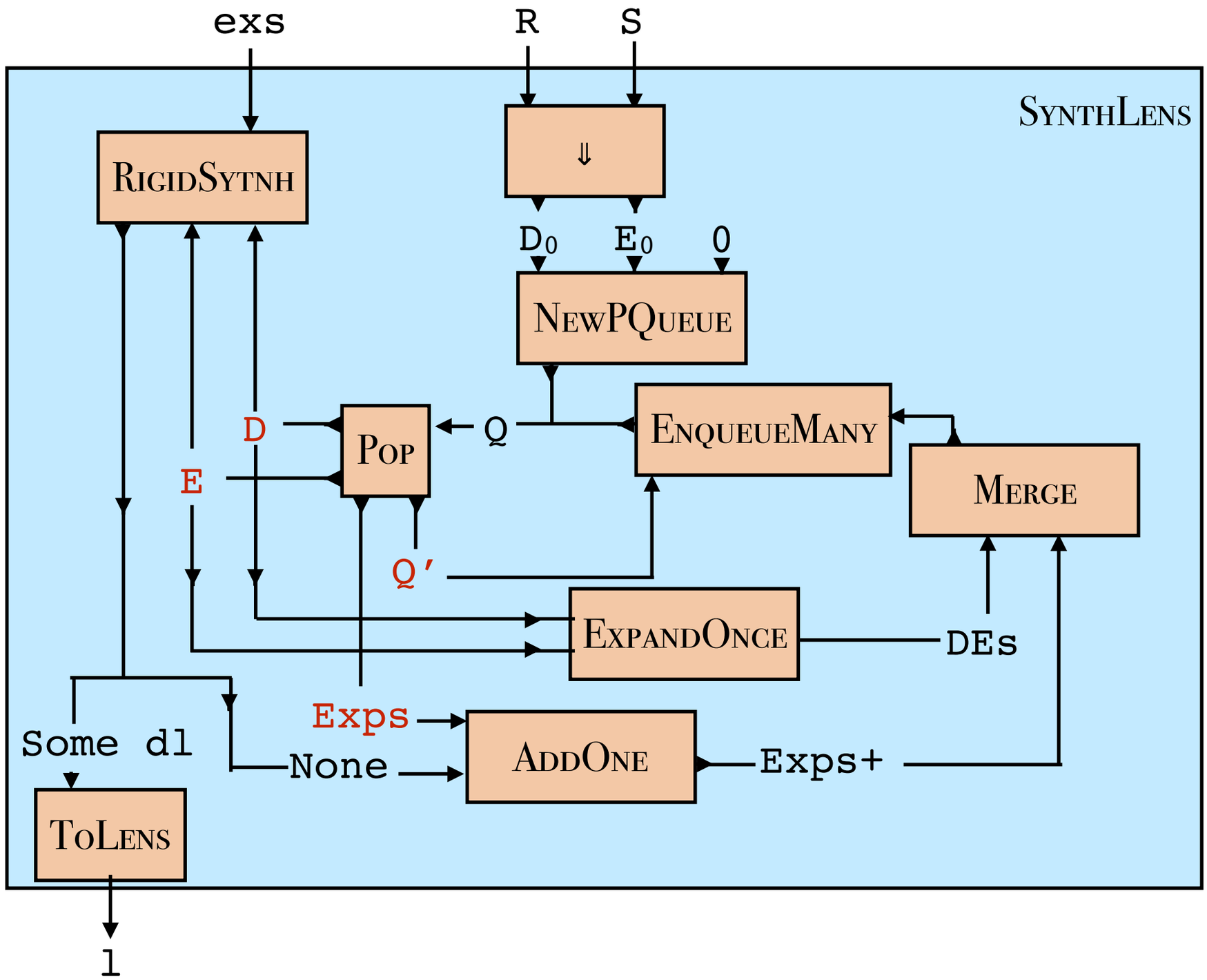

```
exs : string*string
      list
R : RX
S : RX
D0 : DNF RX
E0 : DNF RX
D : DNF RX
E : DNF RX
0 : int
Exps : int
Exps+ : int
Q : Pri Queue
Q′ : Pri Queue
Some dl : DNF Lens
          option
None : DNF Lens
       option
l : Lens
DEs: DNF RX * DNF RX
     list
```

Synthesis Diagram



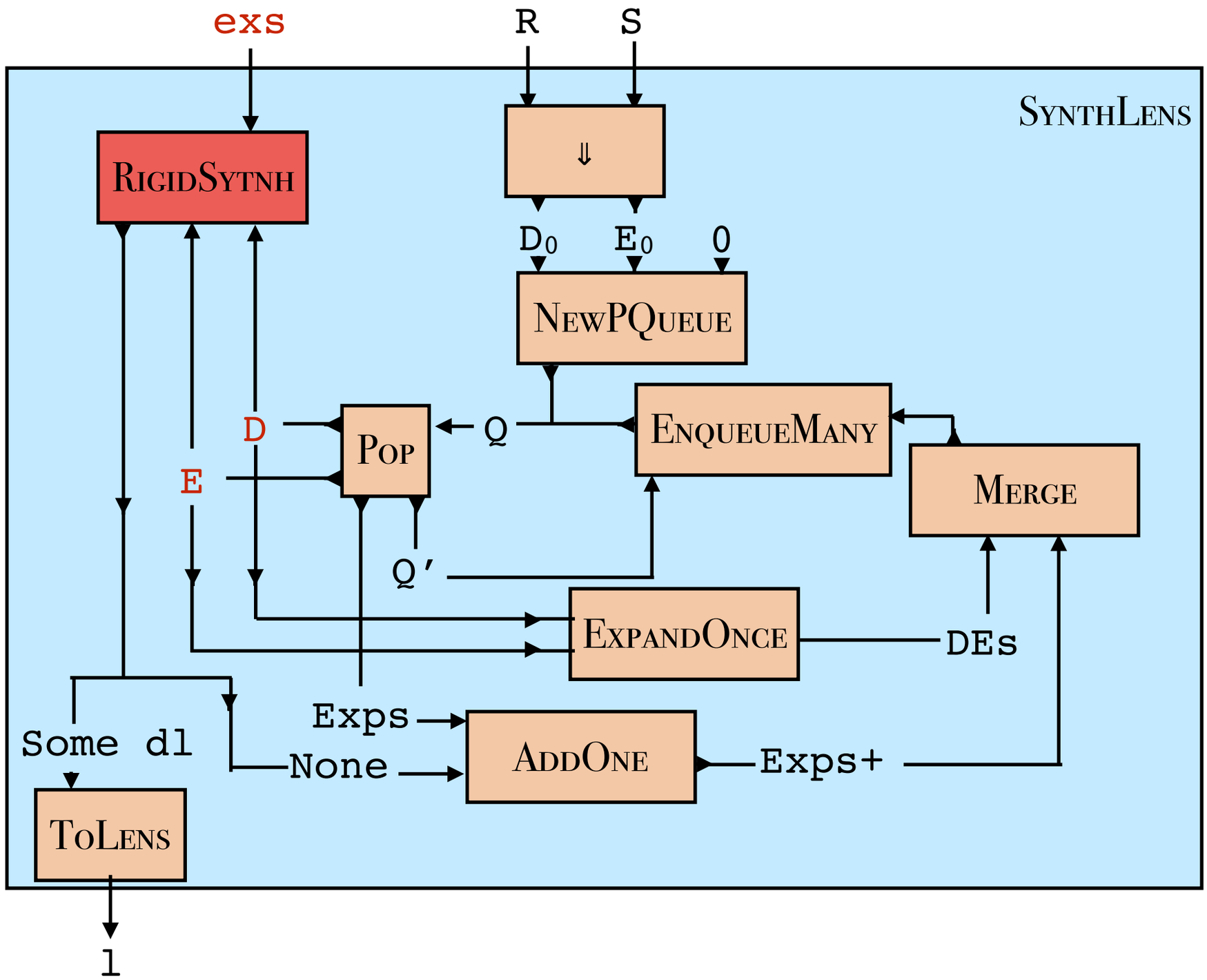

```
exs : string*string
      list
R : RX
S : RX
D0 : DNF RX
E0 : DNF RX
D : DNF RX
E : DNF RX
0 : int
Exps : int
Exps+ : int
Q : Pri Queue
Q′ : Pri Queue
Some dl : DNF Lens
          option
None : DNF Lens
       option
l : Lens
DEs: DNF RX * DNF RX
     list
```

Synthesis Diagram



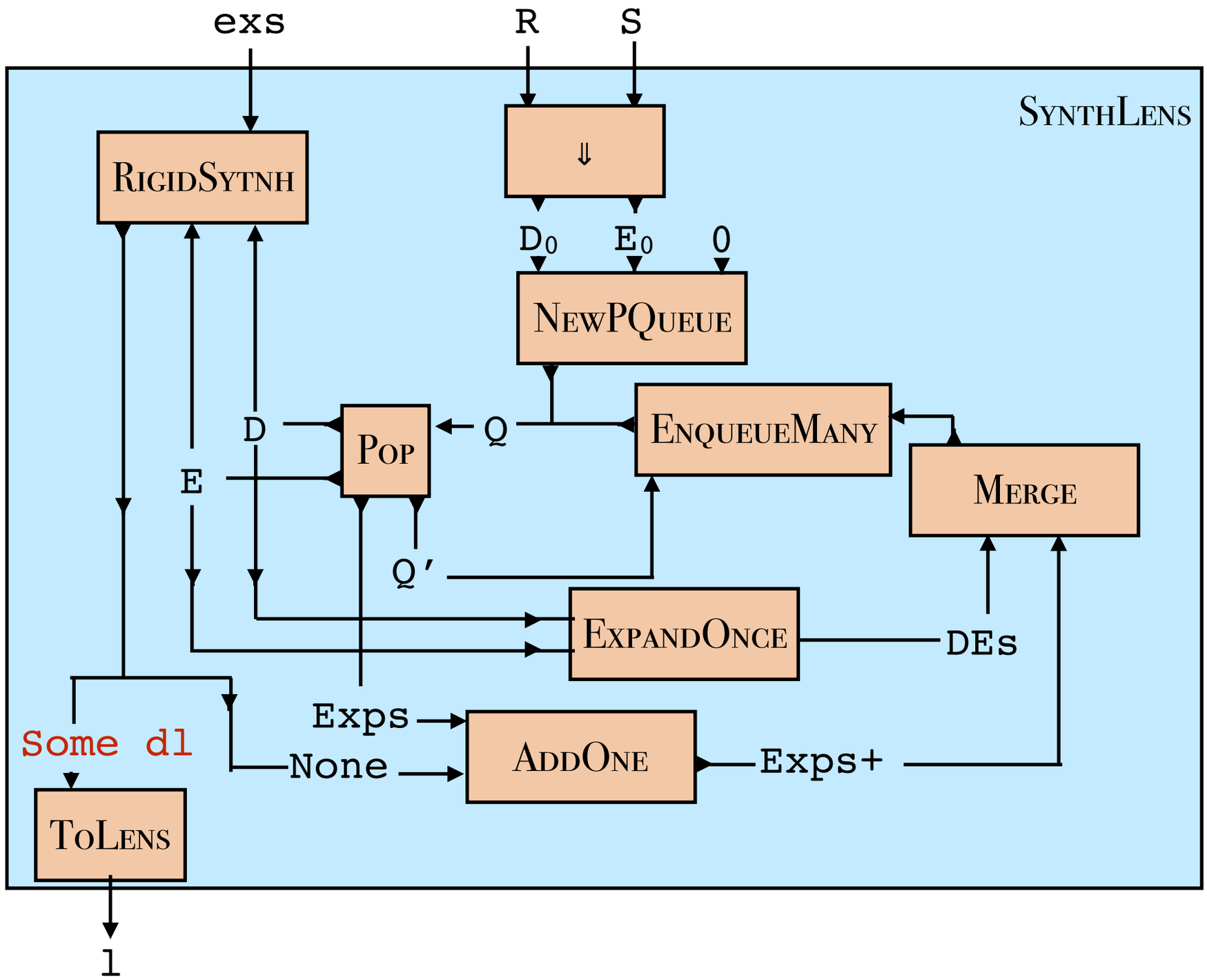

```
exs : string*string
      list
R : RX
S : RX
D0 : DNF RX
E0 : DNF RX
D : DNF RX
E : DNF RX
0 : int
Exps : int
Exps+ : int
Q : Pri Queue
Q′ : Pri Queue
Some dl : DNF Lens
          option
None : DNF Lens
       option
l : Lens
DEs: DNF RX * DNF RX
     list
```

Synthesis Diagram



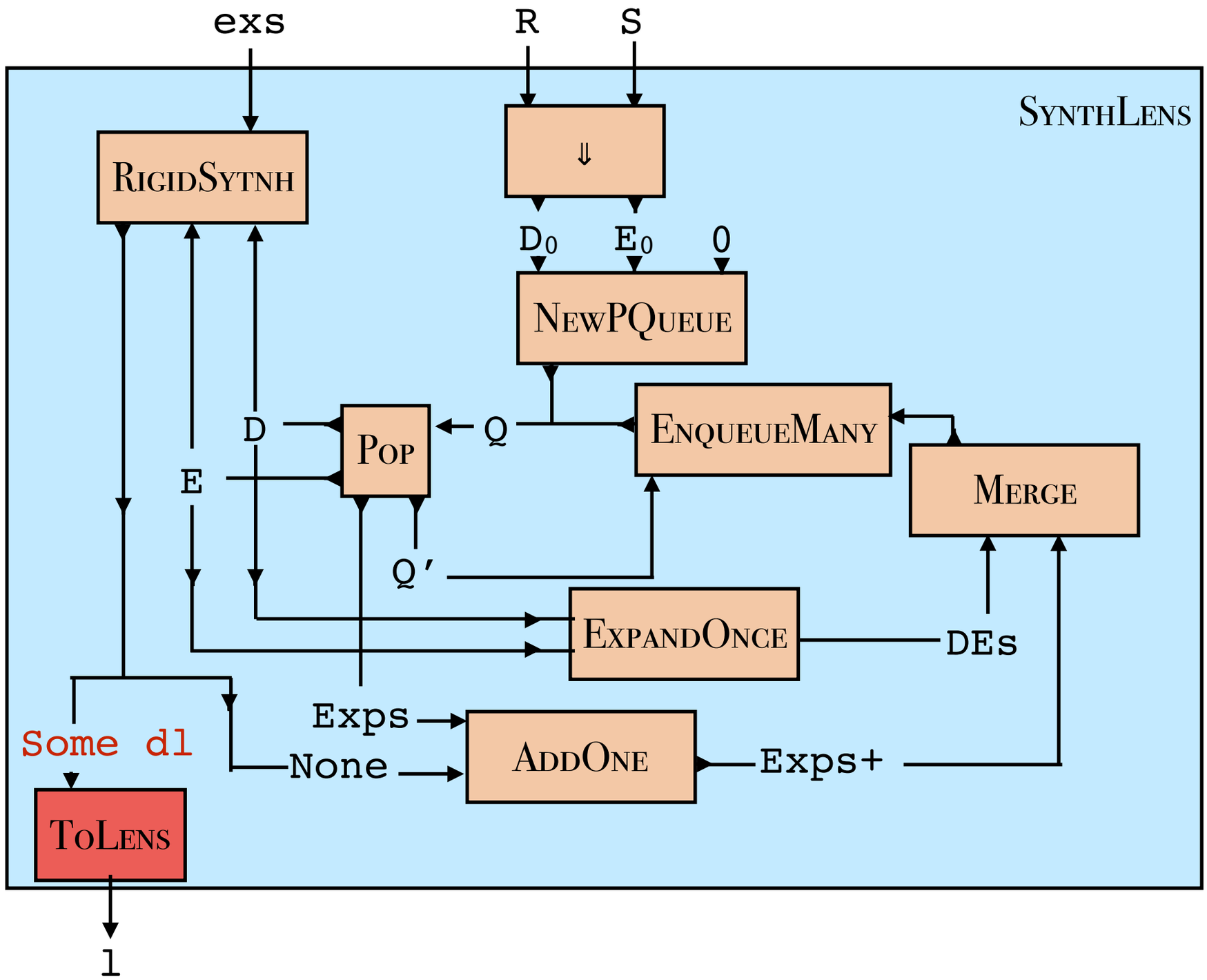

```
exs : string*string
      list
R : RX
S : RX
D0 : DNF RX
E0 : DNF RX
D : DNF RX
E : DNF RX
0 : int
Exps : int
Exps+ : int
Q : Pri Queue
Q′ : Pri Queue
Some dl : DNF Lens
          option
None : DNF Lens
       option
l : Lens
DEs: DNF RX * DNF RX
     list
```

Synthesis Diagram



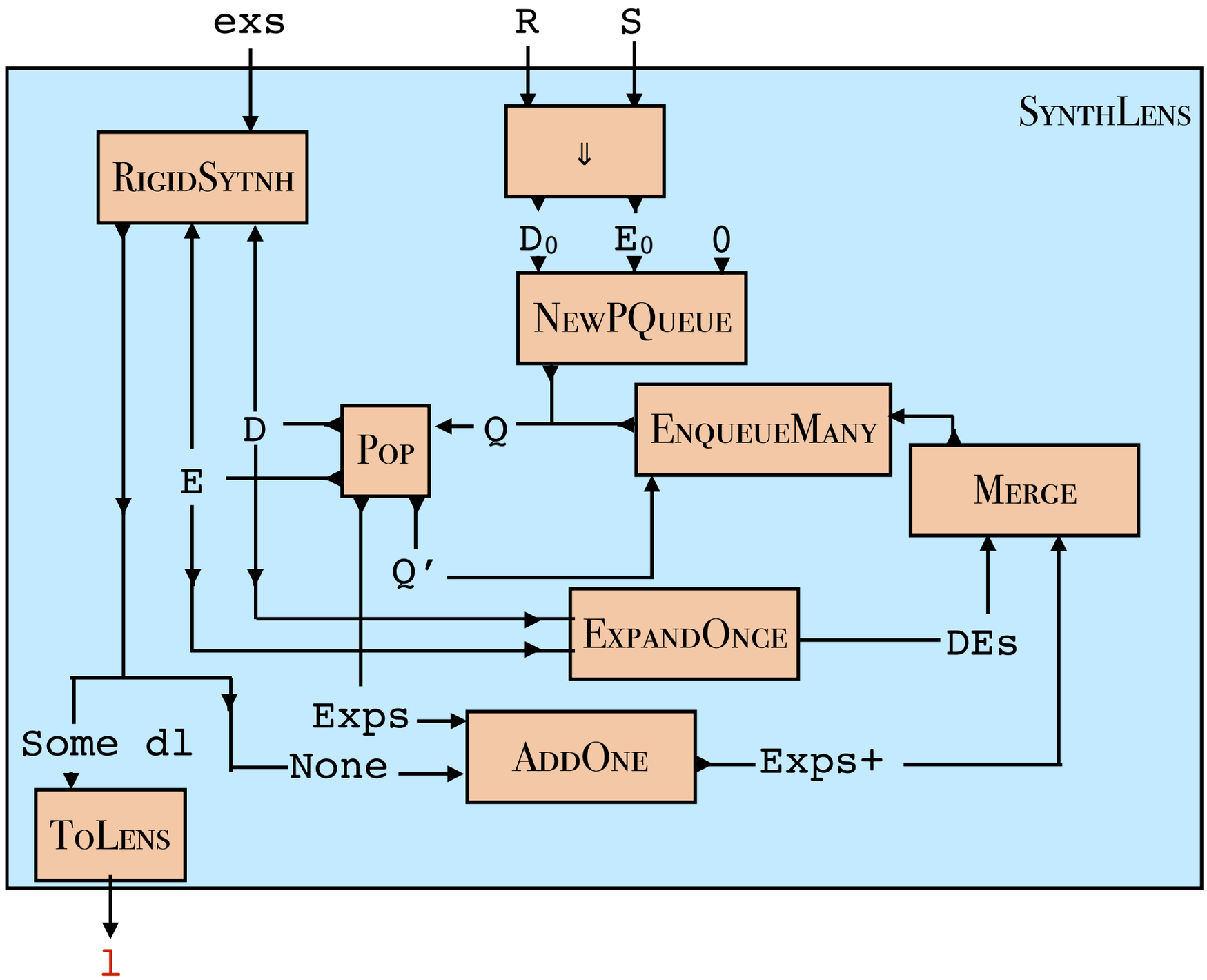

```
exs : string*string
      list
R : RX
S : RX
D0 : DNF RX
E0 : DNF RX
D : DNF RX
E : DNF RX
0 : int
Exps : int
Exps+ : int
Q : Pri Queue
Q′ : Pri Queue
Some dl : DNF Lens
          option
None : DNF Lens
       option
l : Lens
DEs: DNF RX * DNF RX
     list
```

Synthesis Diagram



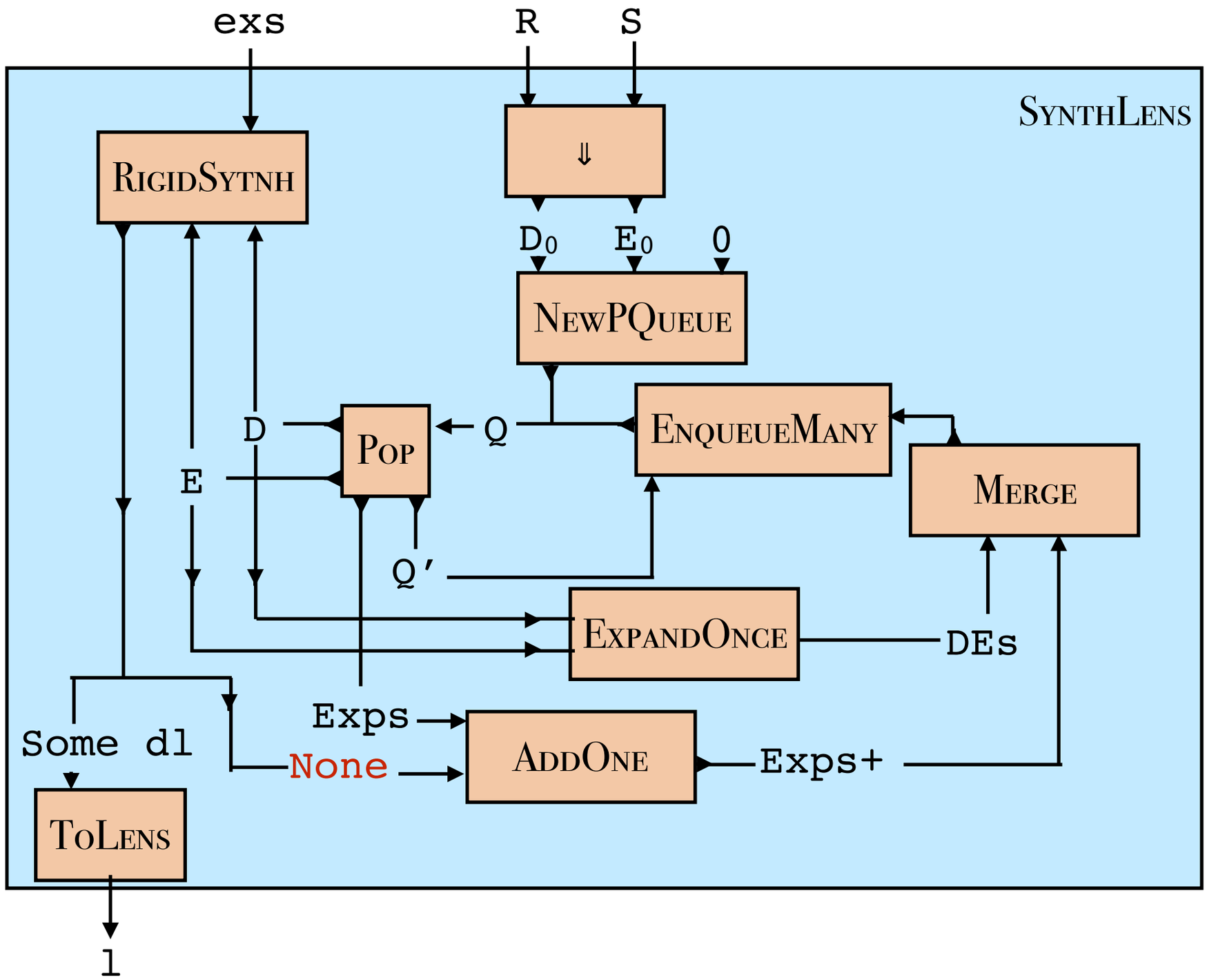

```
exs : string*string
      list
R : RX
S : RX
D0 : DNF RX
E0 : DNF RX
D : DNF RX
E : DNF RX
0 : int
Exps : int
Exps+ : int
Q : Pri Queue
Q′ : Pri Queue
Some dl : DNF Lens
          option
None : DNF Lens
       option
l : Lens
DEs: DNF RX * DNF RX
     list
```

Synthesis Diagram



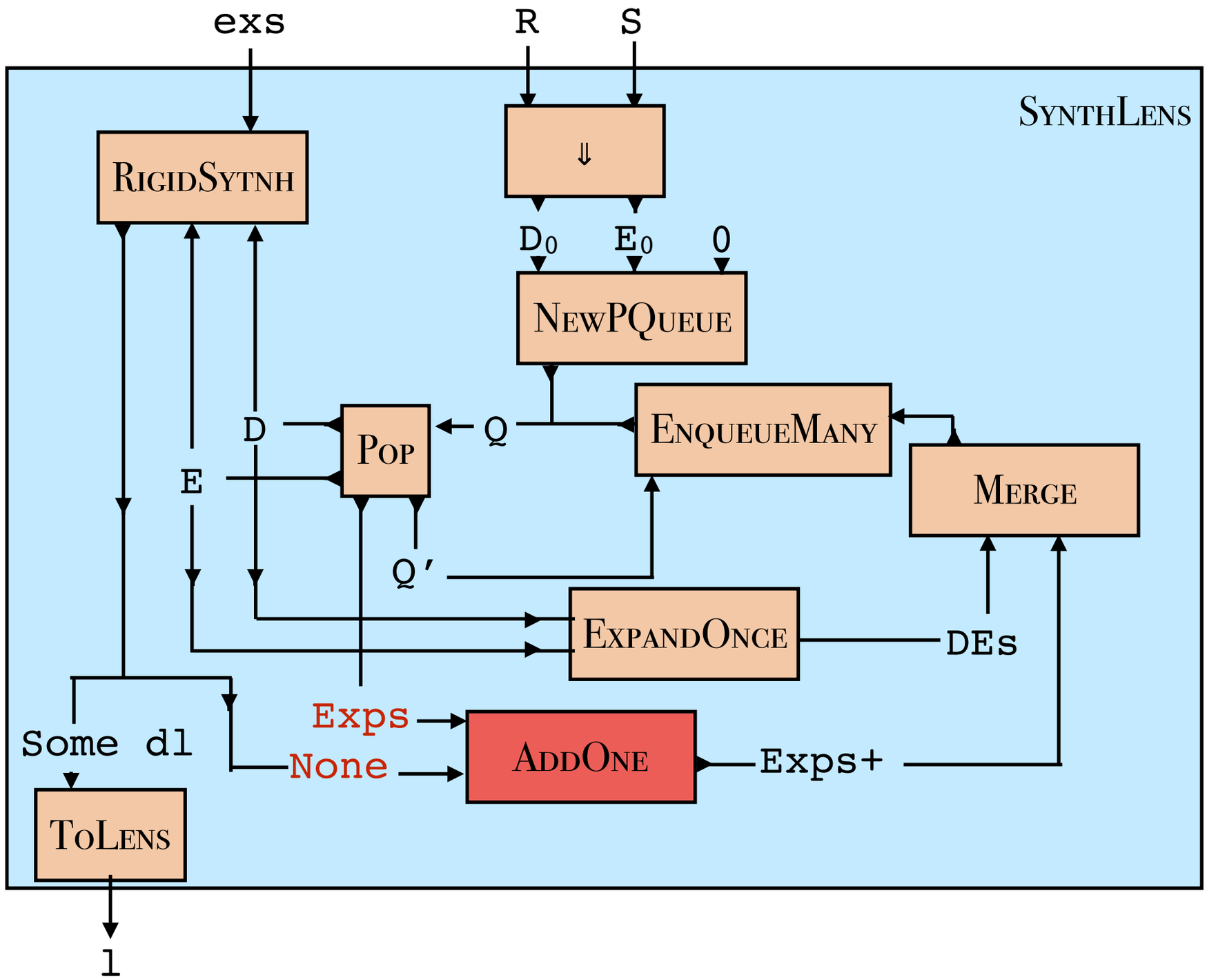

```
exs : string*string
      list
R : RX
S : RX
D0 : DNF RX
E0 : DNF RX
D : DNF RX
E : DNF RX
0 : int
Exps : int
Exps+ : int
Q : Pri Queue
Q′ : Pri Queue
Some dl : DNF Lens
          option
None : DNF Lens
       option
l : Lens
DEs: DNF RX * DNF RX
     list
```

Synthesis Diagram



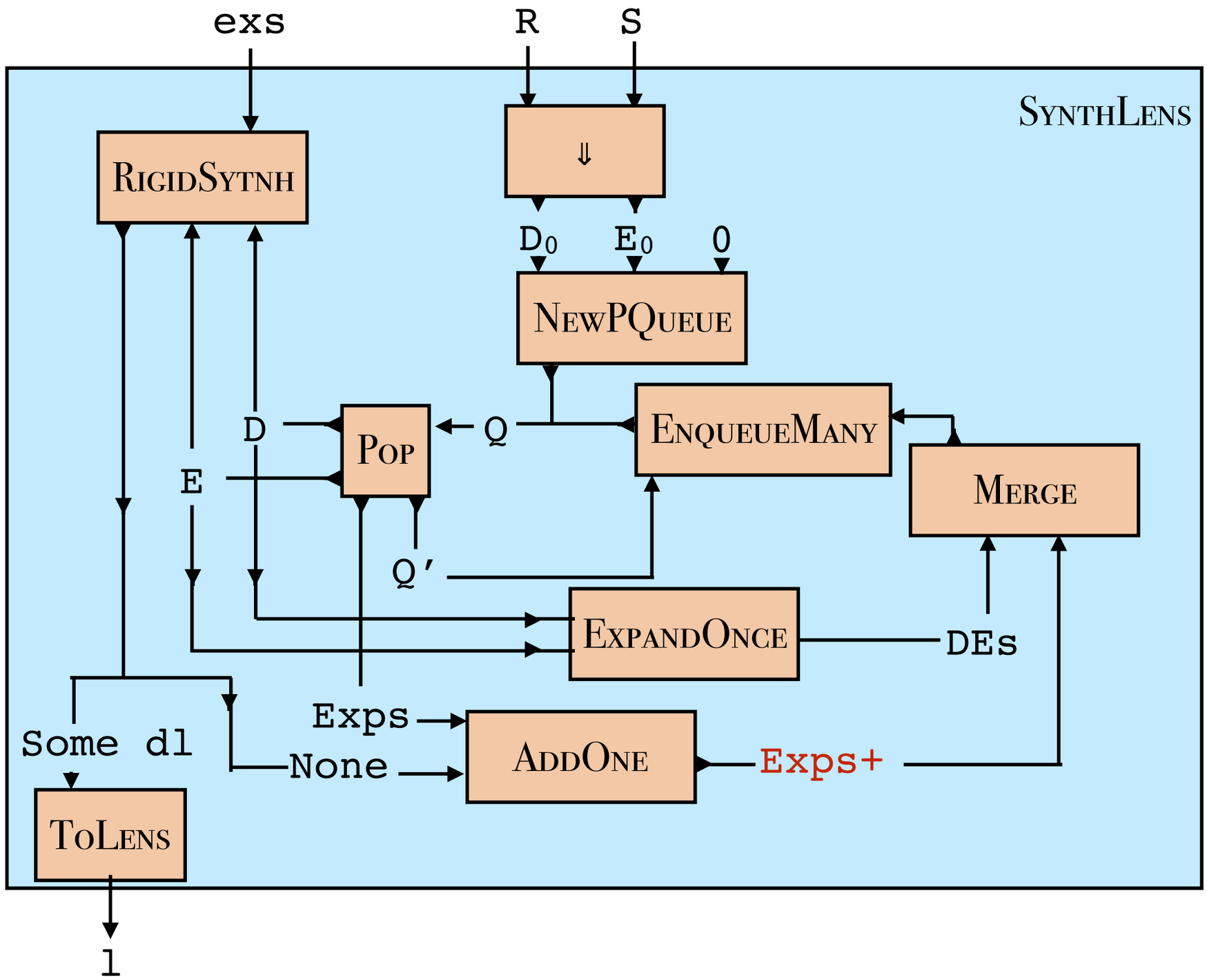

```
exs : string*string
      list
R : RX
S : RX
D0 : DNF RX
E0 : DNF RX
D : DNF RX
E : DNF RX
0 : int
Exps : int
Exps+ : int
Q : Pri Queue
Q′ : Pri Queue
Some dl : DNF Lens
          option
None : DNF Lens
       option
l : Lens
DEs: DNF RX * DNF RX
     list
```

Synthesis Diagram



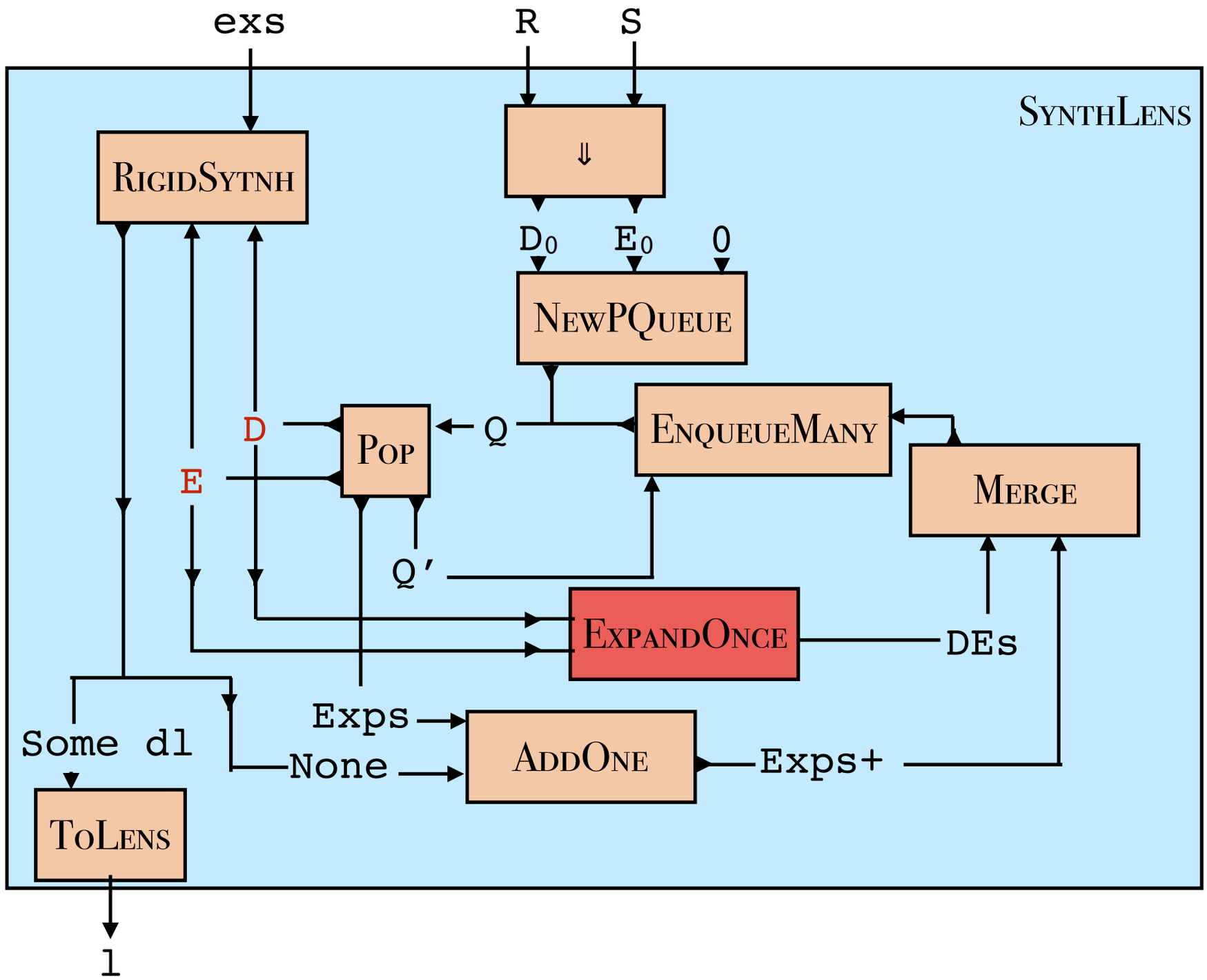

```
exs : string*string
      list
R : RX
S : RX
D0 : DNF RX
E0 : DNF RX
D : DNF RX
E : DNF RX
0 : int
Exps : int
Exps+ : int
Q : Pri Queue
Q′ : Pri Queue
Some dl : DNF Lens
          option
None : DNF Lens
       option
l : Lens
DEs: DNF RX * DNF RX
     list
```

Synthesis Diagram



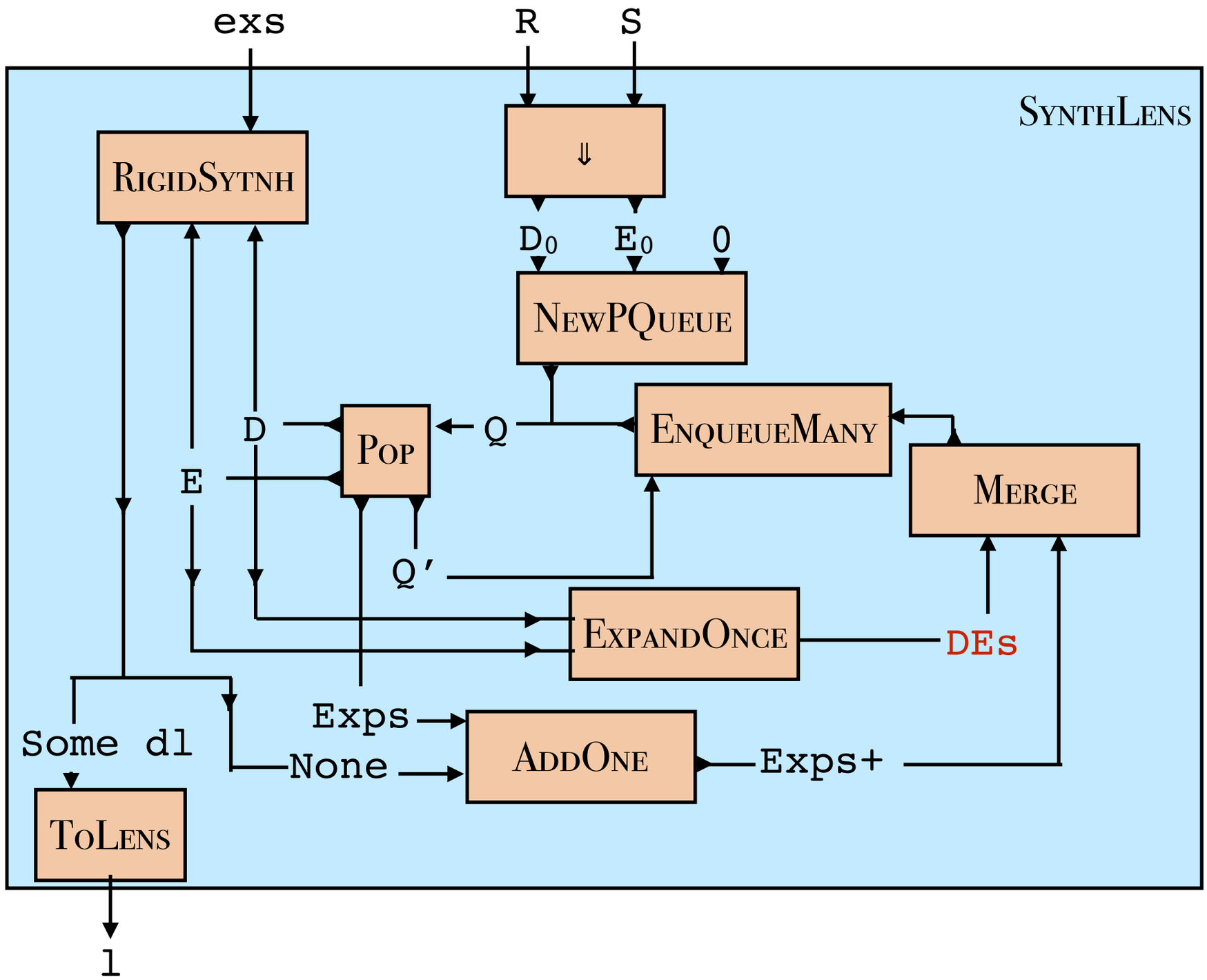

```
exs : string*string
      list
R : RX
S : RX
D0 : DNF RX
E0 : DNF RX
D : DNF RX
E : DNF RX
0 : int
Exps : int
Exps+ : int
Q : Pri Queue
Q′ : Pri Queue
Some dl : DNF Lens
          option
None : DNF Lens
       option
l : Lens
DEs: DNF RX * DNF RX
     list
```

Synthesis Diagram



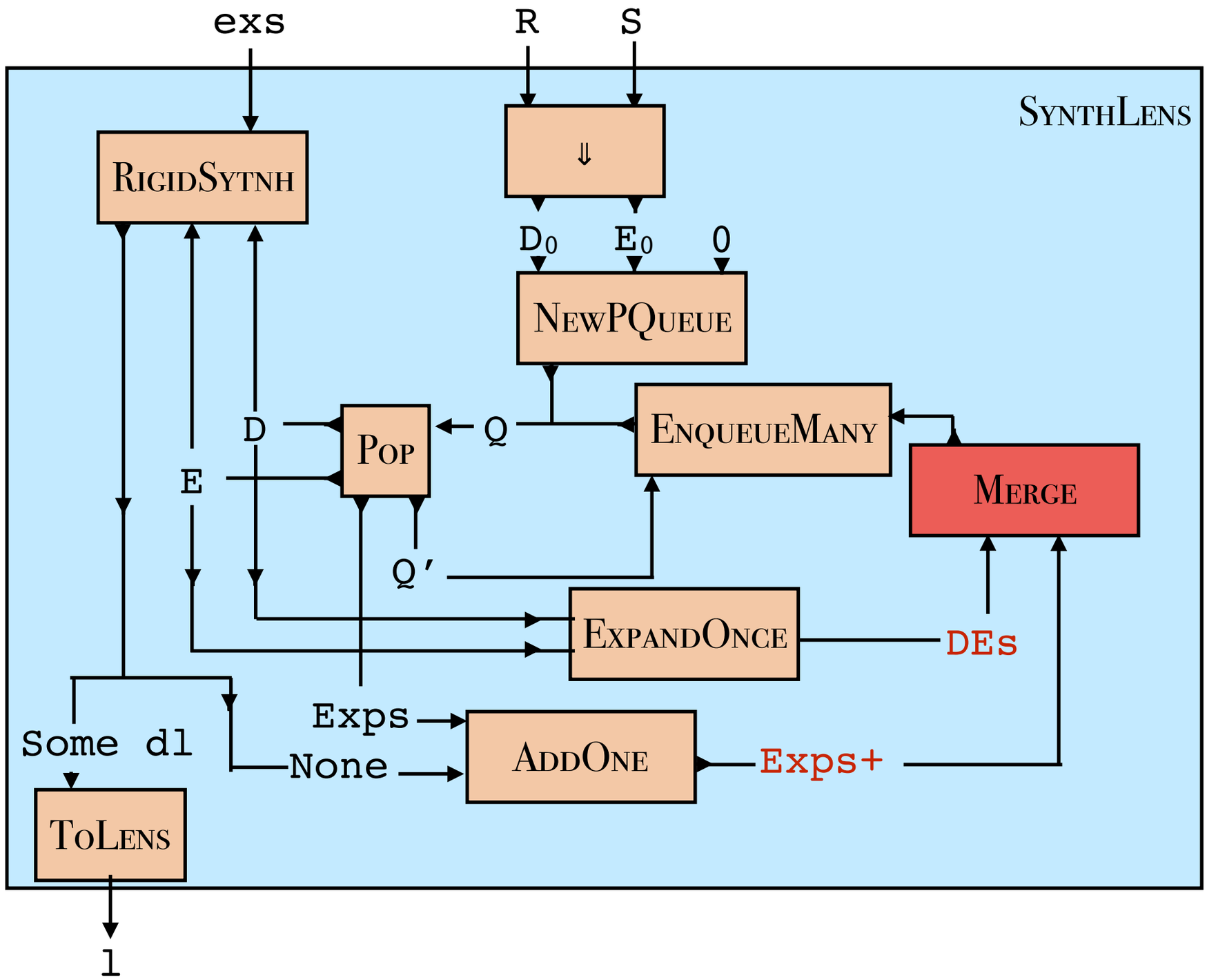

```
exs : string*string
      list
R : RX
S : RX
D0 : DNF RX
E0 : DNF RX
D : DNF RX
E : DNF RX
0 : int
Exps : int
Exps+ : int
Q : Pri Queue
Q′ : Pri Queue
Some dl : DNF Lens
          option
None : DNF Lens
       option
l : Lens
DEs: DNF RX * DNF RX
     list
```

Synthesis Diagram



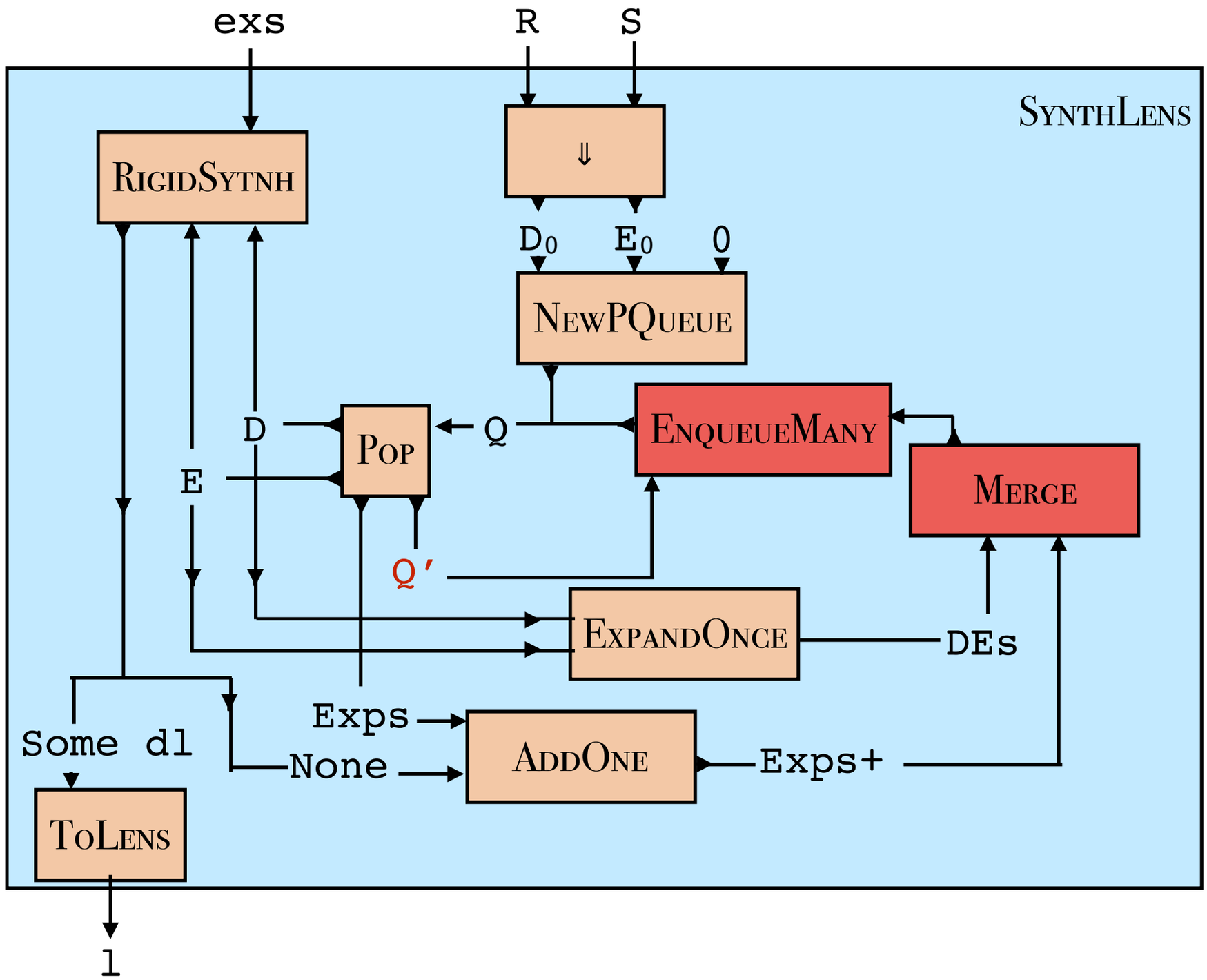

```
exs : string*string
      list
R : RX
S : RX
D0 : DNF RX
E0 : DNF RX
D : DNF RX
E : DNF RX
0 : int
Exps : int
Exps+ : int
Q : Pri Queue
Q′ : Pri Queue
Some dl : DNF Lens
          option
None : DNF Lens
       option
l : Lens
DEs: DNF RX * DNF RX
     list
```

Synthesis Diagram



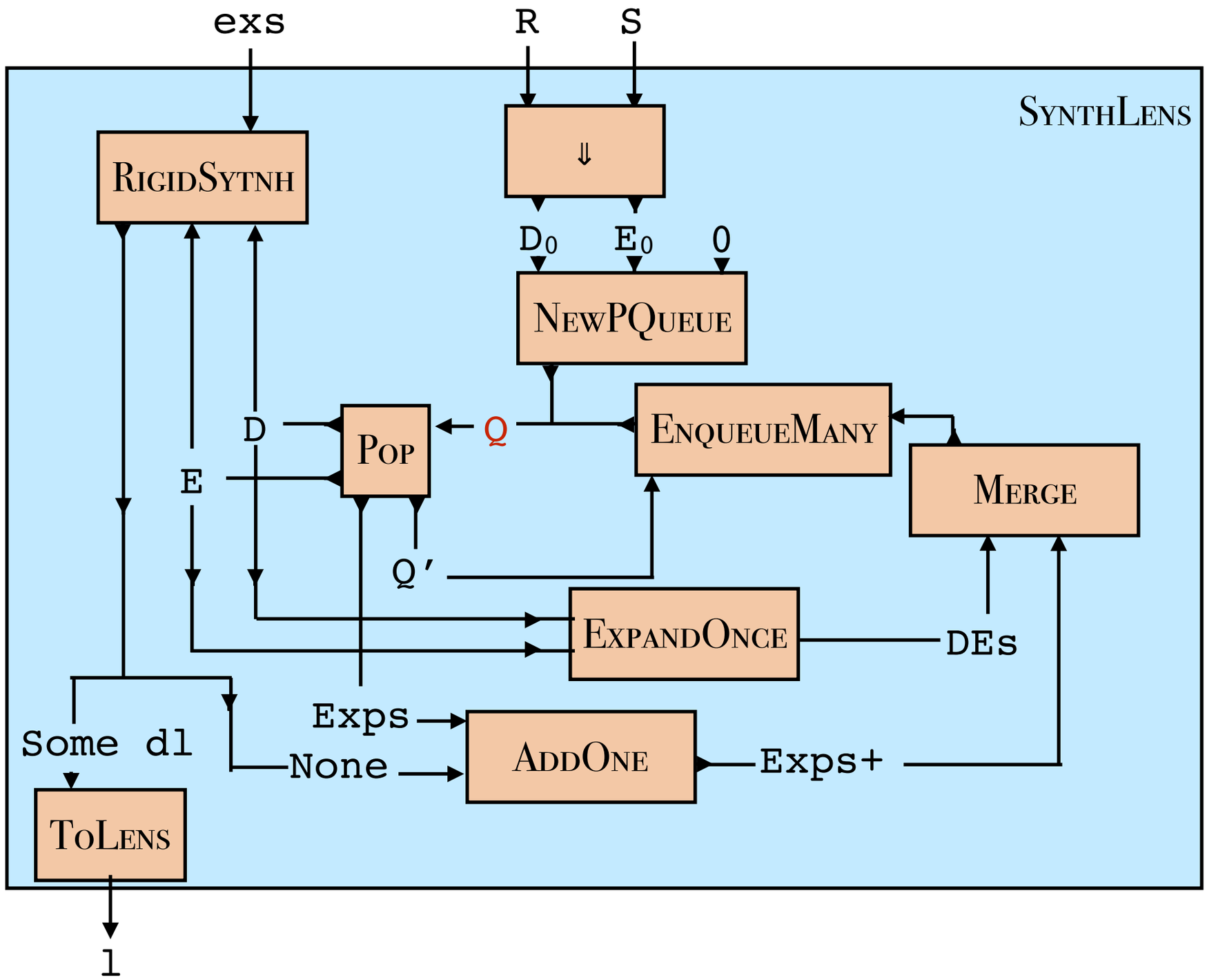

```
exs : string*string
      list
R : RX
S : RX
D0 : DNF RX
E0 : DNF RX
D : DNF RX
E : DNF RX
0 : int
Exps : int
Exps+ : int
Q : Pri Queue
Q' : Pri Queue
Some dl : DNF Lens
          option
None : DNF Lens
       option
l : Lens
DEs: DNF RX * DNF RX
     list
```

Synthesis Diagram



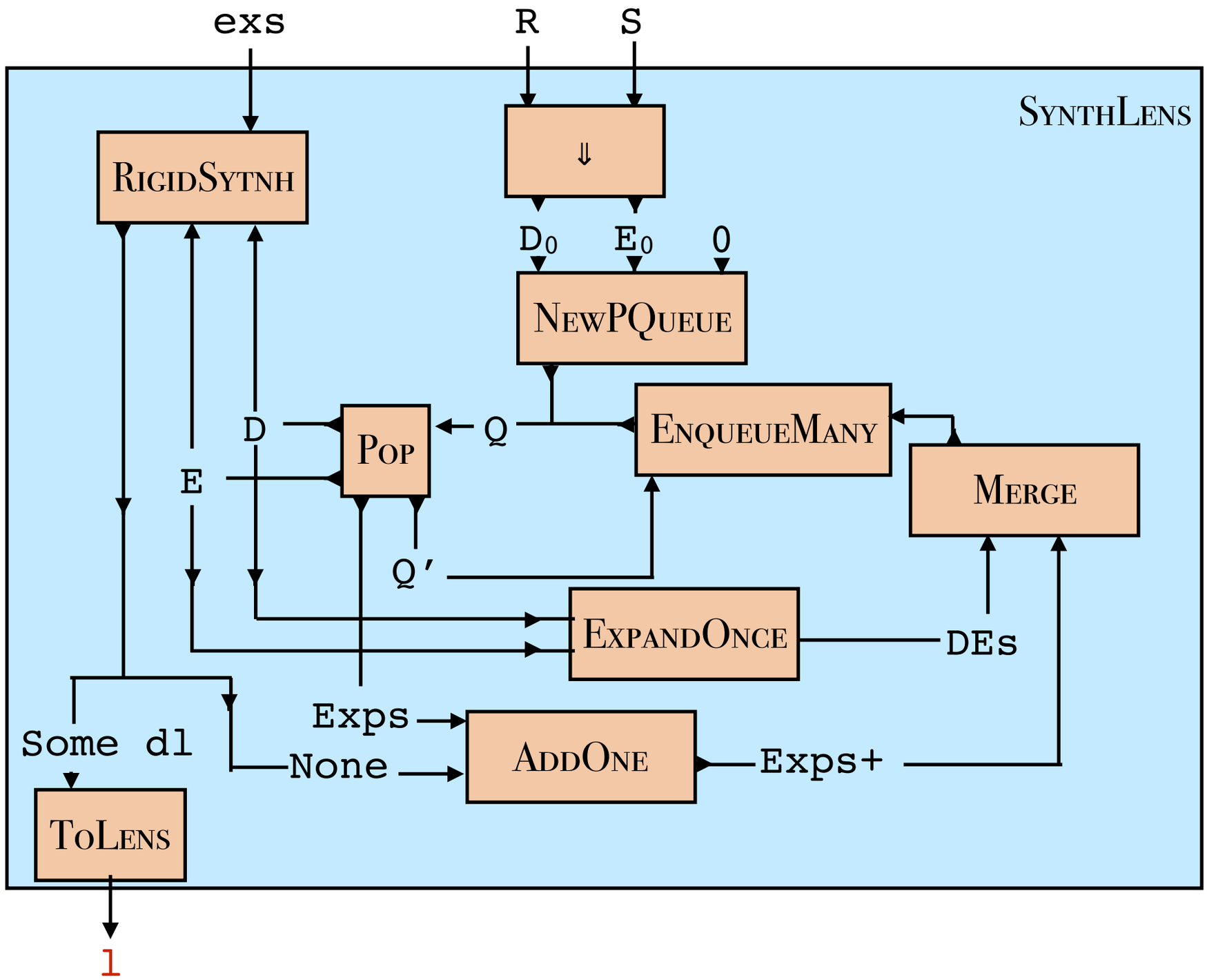

```
exs : string*string
      list
R : RX
S : RX
D0 : DNF RX
E0 : DNF RX
D : DNF RX
E : DNF RX
0 : int
Exps : int
Exps+ : int
Q : Pri Queue
Q′ : Pri Queue
Some dl : DNF Lens
          option
None : DNF Lens
       option
l : Lens
DEs: DNF RX * DNF RX
     list
```

Synthesis Diagram




```
exs : string*string
      list
R : RX
S : RX
D0 : DNF RX
E0 : DNF RX
D : DNF RX
E : DNF RX
0 : int
Exps : int
Exps+ : int
Q : Pri Queue
Q′ : Pri Queue
Some dl : DNF Lens
          option
None : DNF Lens
       option
l : Lens
DEs: DNF RX * DNF RX
     list
```

Synthesis Diagram




```
exs : string*string
      list
R : RX
S : RX
D0 : DNF RX
E0 : DNF RX
D : DNF RX
E : DNF RX
0 : int
Exps : int
Exps+ : int
Q : Pri Queue
Q′ : Pri Queue
Some dl : DNF Lens
          option
None : DNF Lens
       option
l : Lens
DEs: DNF RX * DNF RX
     list
```

Priority Queue

- Higher priority for fewer expansions preformed
- Higher priority for more similar looking regular expressions
- Lower score means higher priority

$$\textsc{Score}(\texttt{D},\texttt{E},\texttt{Exps}) = d(\texttt{D},\texttt{E}) + 8*\texttt{Exps}$$

DNF RX Distance

Based on the distribution of the user-defined Regular Expressions

```
Entry."test"|Names|(Names.Entry)*|Other
Names|Alt*.Entry|Names.Names
```

DNF RX Distance

Based on the distribution of the user-defined Regular Expressions

```
Entry."test"|Names|(Names.Entry)*|Other
Names|Alt*.Entry|Names.Names
```

```
(Entry,0) -> 1
```

DNF RX Distance

Based on the distribution of the user-defined Regular Expressions

```
Entry."test"|Names|(Names.Entry)*|Other
Names|Alt*.Entry|Names.Names
```

```
(Entry,0) -> 1
```

DNF RX Distance

Based on the distribution of the user-defined Regular Expressions

```
Entry."test"|Names|(Names.Entry)*|Other
Names|Alt*.Entry|Names.Names
```

```
(Entry,0) -> 1
(Names,0) -> 1
```

DNF RX Distance

Based on the distribution of the user-defined Regular Expressions

```
Entry."test"|Names|(Names.Entry)*|Other
Names|Alt*.Entry|Names.Names
```

```
(Entry,0) -> 1
(Names,0) -> 1
(Names,1) -> 1
```

DNF RX Distance

Based on the distribution of the user-defined Regular Expressions

```
Entry."test"|Names|(Names.Entry)*|Other
Names|Alt*.Entry|Names.Names
```

```
(Entry,0) -> 1
(Names,0) -> 1
(Names,1) -> 1
(Entry,1) -> 1
```

DNF RX Distance

Based on the distribution of the user-defined Regular Expressions

```
Entry."test"|Names|(Names.Entry)*|Other
Names|Alt*.Entry|Names.Names
```

```
(Entry,0) -> 1
(Names,0) -> 1
(Names,1) -> 1
(Entry,1) -> 1
(Other,0) -> 1
```

DNF RX Distance

Based on the distribution of the user-defined Regular Expressions

```
Entry."test"|Names|(Names.Entry)*|Other
Names|Alt*.Entry|Names.Names
```

```
(Entry,0) -> 1
(Names,0) -> 1
(Names,1) -> 1
(Entry,1) -> 1
(Other,0) -> 1
```

```
(Names,0) -> 1
```

DNF RX Distance

Based on the distribution of the user-defined Regular Expressions

```
Entry."test"|Names|(Names.Entry)*|Other
Names|Alt*.Entry|Names.Names
```

```
(Entry,0) -> 1
(Names,0) -> 1
(Names,1) -> 1
(Entry,1) -> 1
(Other,0) -> 1
```

```
(Names,0) -> 1
 (Alt,1) -> 1
```

DNF RX Distance

Based on the distribution of the user-defined Regular Expressions

```
Entry."test"|Names|(Names.Entry)*|Other
Names|Alt*.Entry|Names.Names
```

```
(Entry,0) -> 1
(Names,0) -> 1
(Names,1) -> 1
(Entry,1) -> 1
(Other,0) -> 1
```

```
(Names,0) -> 1
 (Alt,1) -> 1
(Entry,0) -> 1
```

DNF RX Distance

Based on the distribution of the user-defined Regular Expressions

```
Entry."test"|Names|(Names.Entry)*|Other
Names|Alt*.Entry|Names.Names
```

```
(Entry,0) -> 1
(Names,0) -> 1
(Names,1) -> 1
(Entry,1) -> 1
(Other,0) -> 1
```

```
(Names,0) -> 2
 (Alt,1) -> 1
(Entry,0) -> 1
```

DNF RX Distance

Based on the distribution of the user-defined Regular Expressions

```
Entry."test"|Names|(Names.Entry)*|Other
Names|Alt*.Entry|Names.Names
```

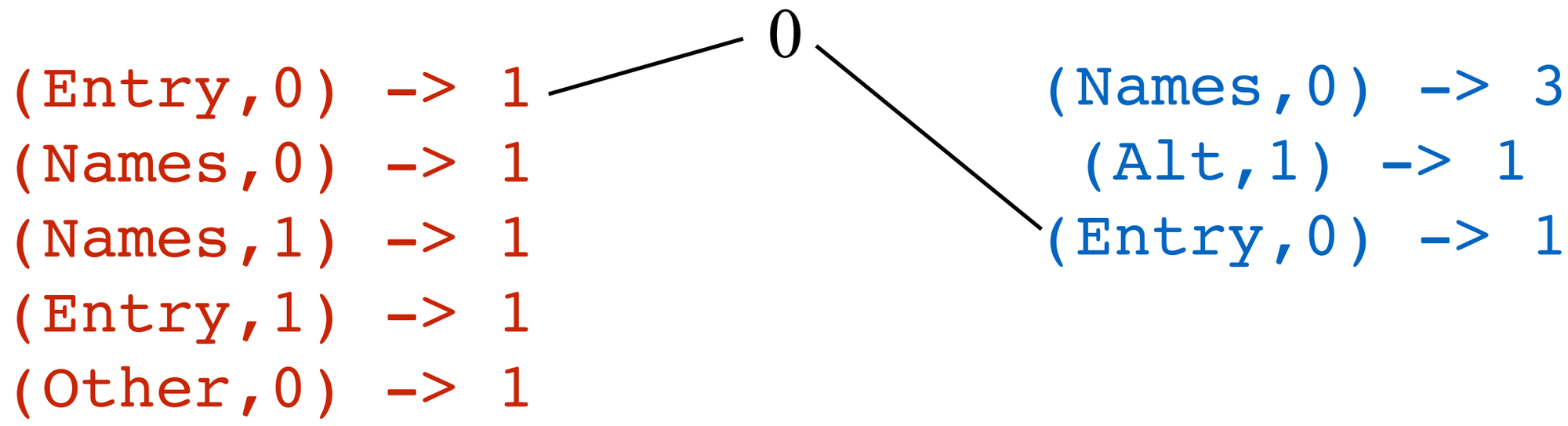
```
(Entry,0) -> 1
(Names,0) -> 1
(Names,1) -> 1
(Entry,1) -> 1
(Other,0) -> 1
```

```
(Names,0) -> 3
 (Alt,1) -> 1
(Entry,0) -> 1
```

DNF RX Distance

Based on the distribution of the user-defined Regular Expressions

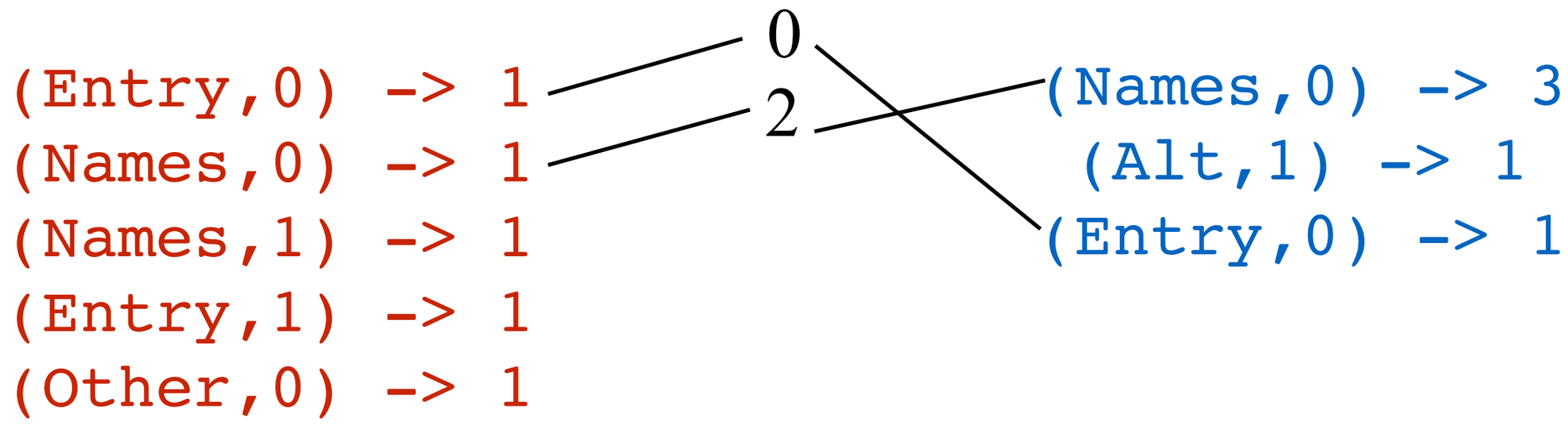
```
Entry."test"|Names|(Names.Entry)*|Other
Names|Alt*.Entry|Names.Names
```



DNF RX Distance

Based on the distribution of the user-defined Regular Expressions

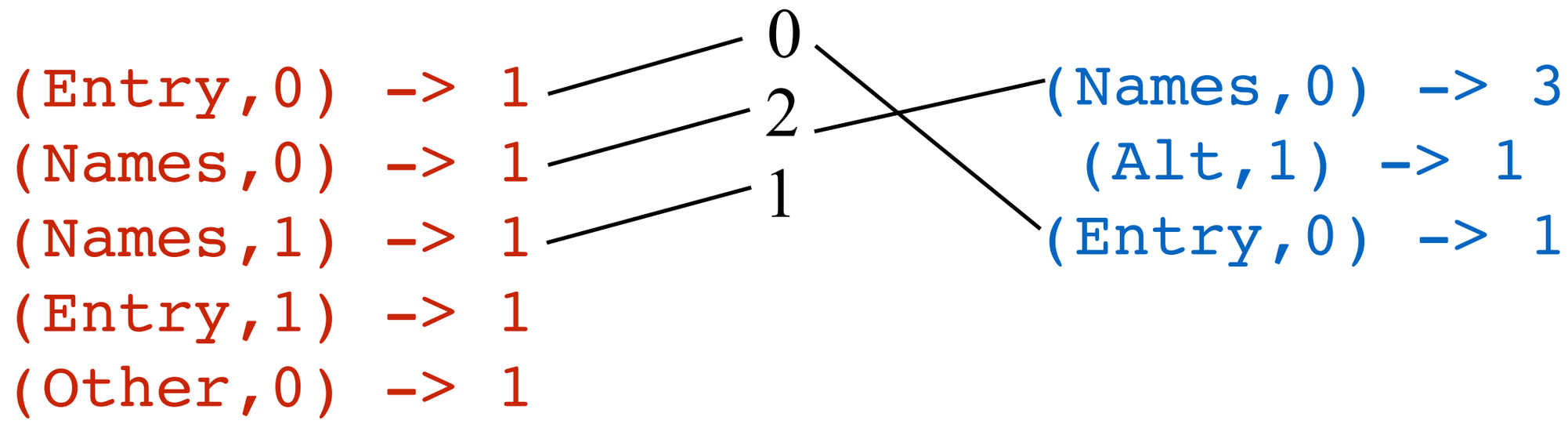
```
Entry."test"|Names|(Names.Entry)*|Other
Names|Alt*.Entry|Names.Names
```



DNF RX Distance

Based on the distribution of the user-defined Regular Expressions

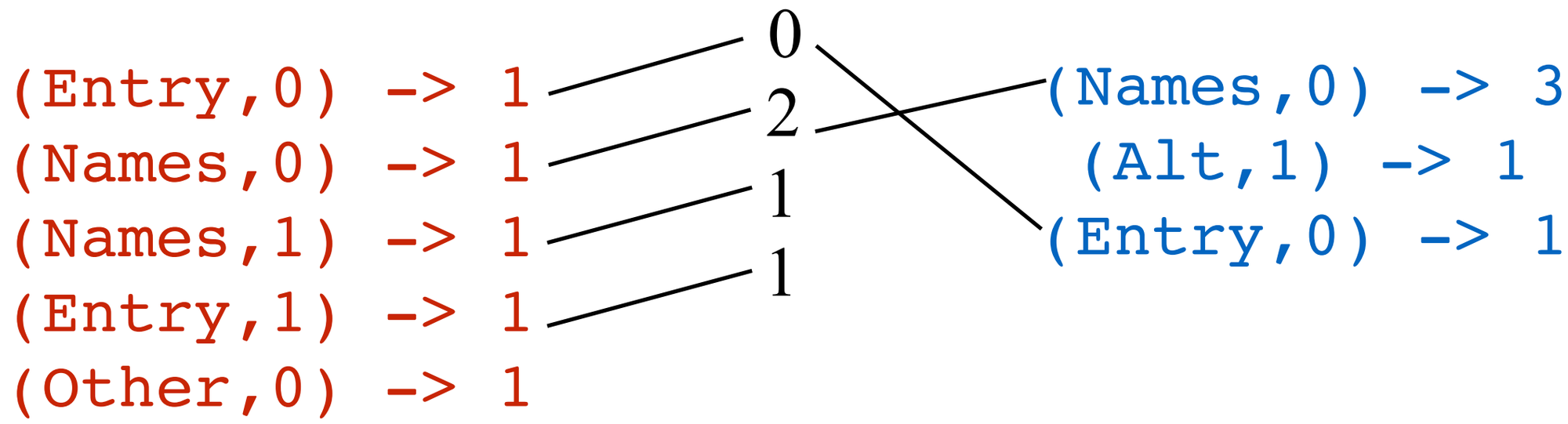
```
Entry."test"|Names|(Names.Entry)*|Other
Names|Alt*.Entry|Names.Names
```



DNF RX Distance

Based on the distribution of the user-defined Regular Expressions

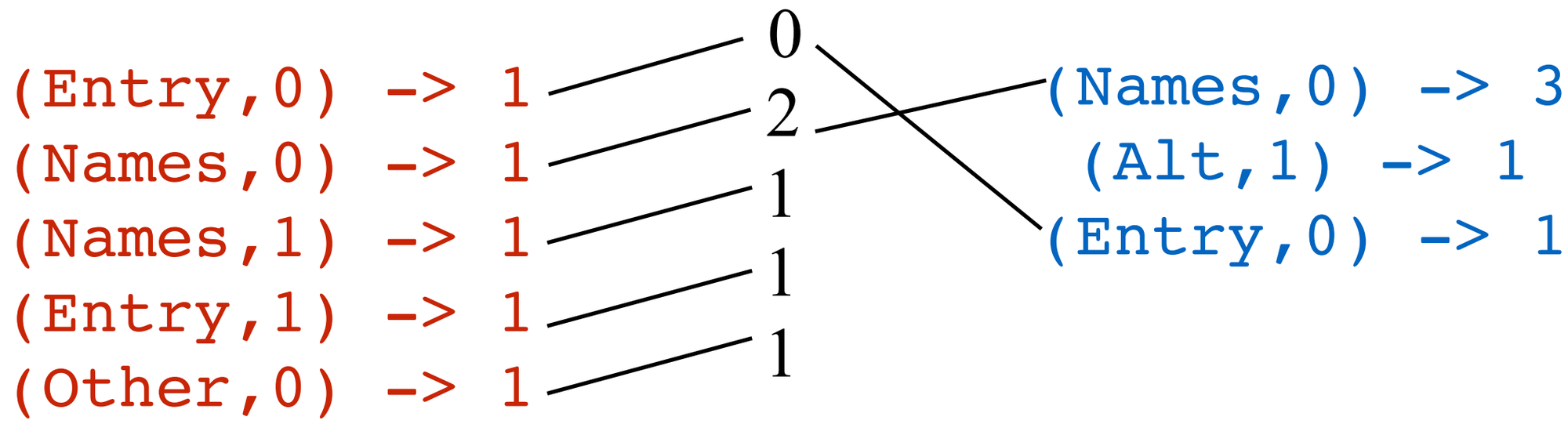
```
Entry."test"|Names|(Names.Entry)*|Other
Names|Alt*.Entry|Names.Names
```



DNF RX Distance

Based on the distribution of the user-defined Regular Expressions

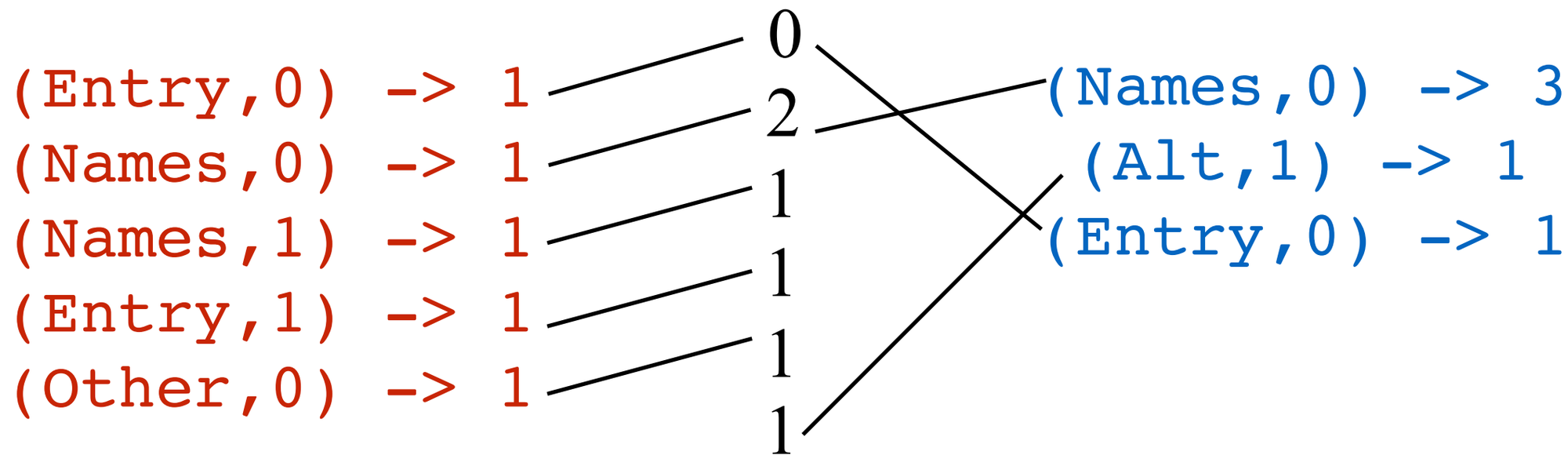
```
Entry."test"|Names|(Names.Entry)*|Other
Names|Alt*.Entry|Names.Names
```



DNF RX Distance

Based on the distribution of the user-defined Regular Expressions

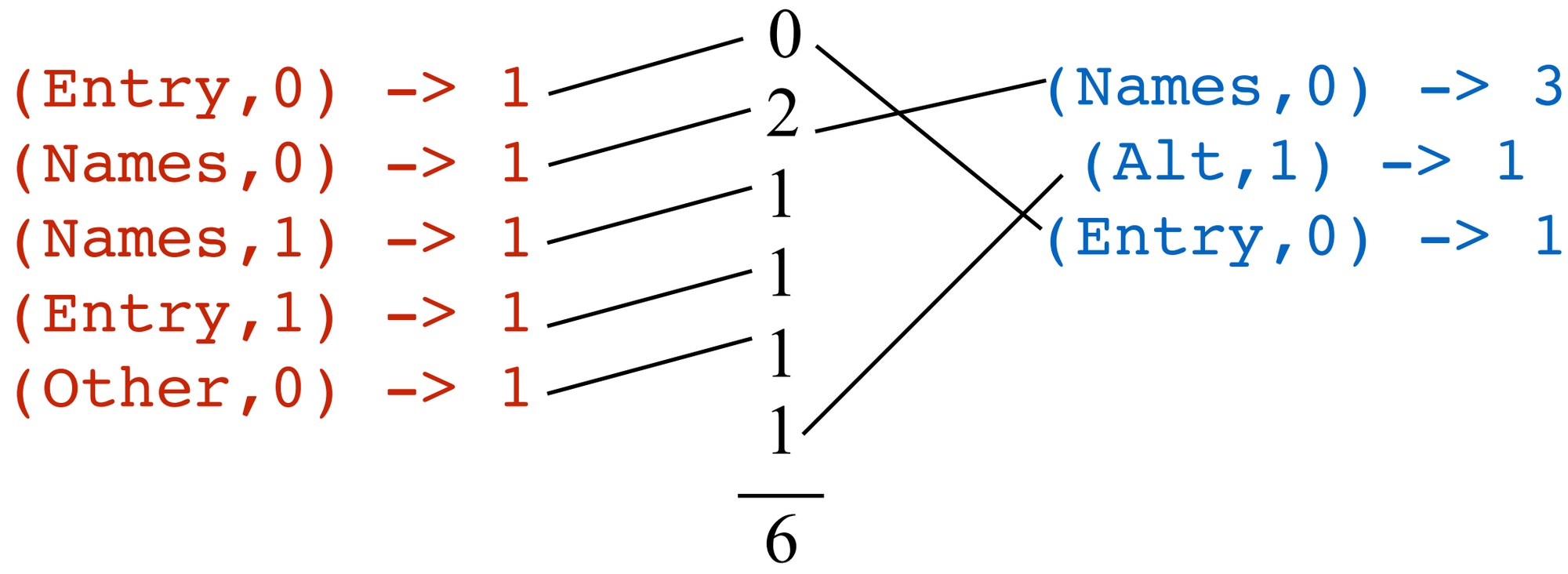
```
Entry."test"|Names|(Names.Entry)*|Other
Names|Alt*.Entry|Names.Names
```



DNF RX Distance

Based on the distribution of the user-defined Regular Expressions

```
Entry."test"|Names|(Names.Entry)*|Other
Names|Alt*.Entry|Names.Names
```

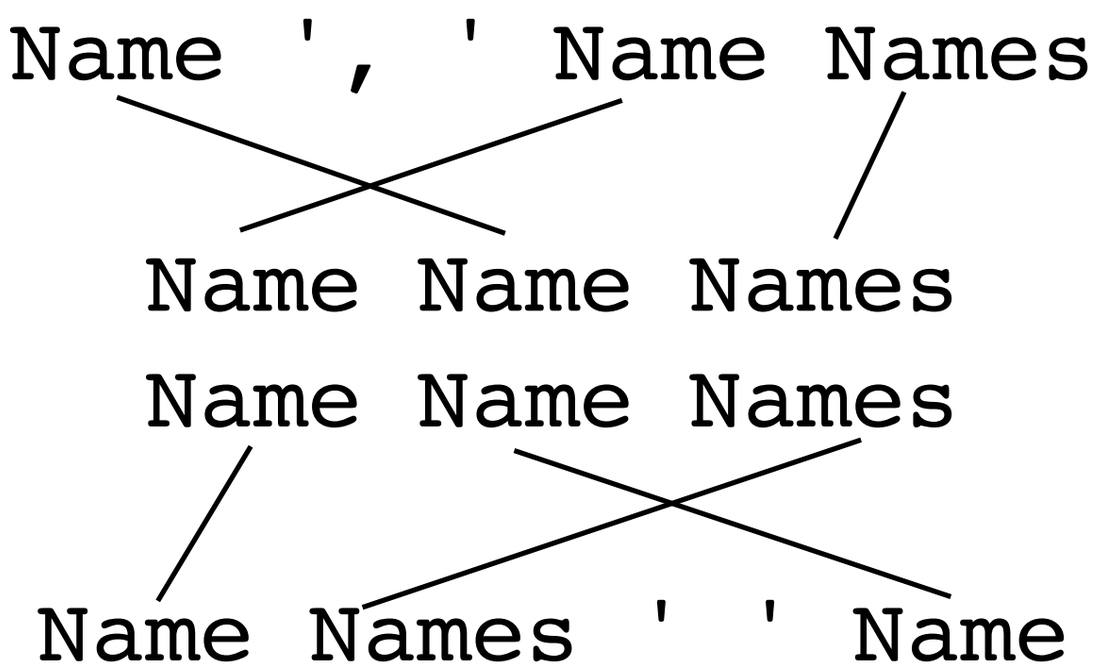


RigidSynth

Synthesize `dl` satisfying `dl :: DR <=> DS`

Difficulty lies in determining the permutations σ

Order the subcomponents of a DNF Regular Expression

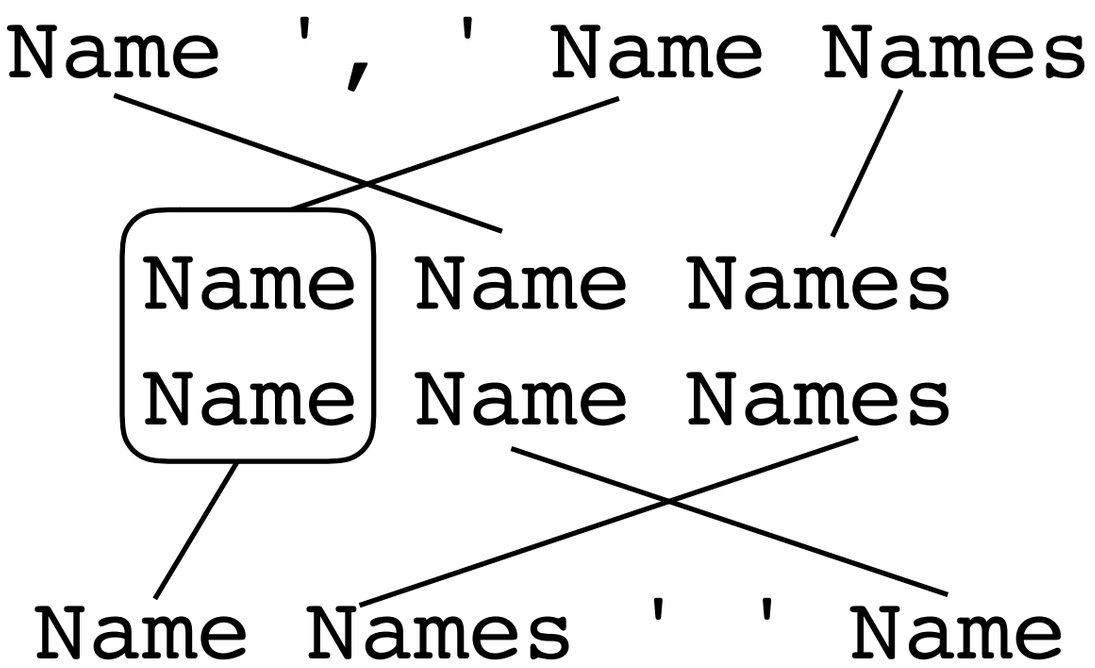


RigidSynth

Synthesize `dl` satisfying `dl :: DR <=> DS`

Difficulty lies in determining the permutations σ

Order the subcomponents of a DNF Regular Expression

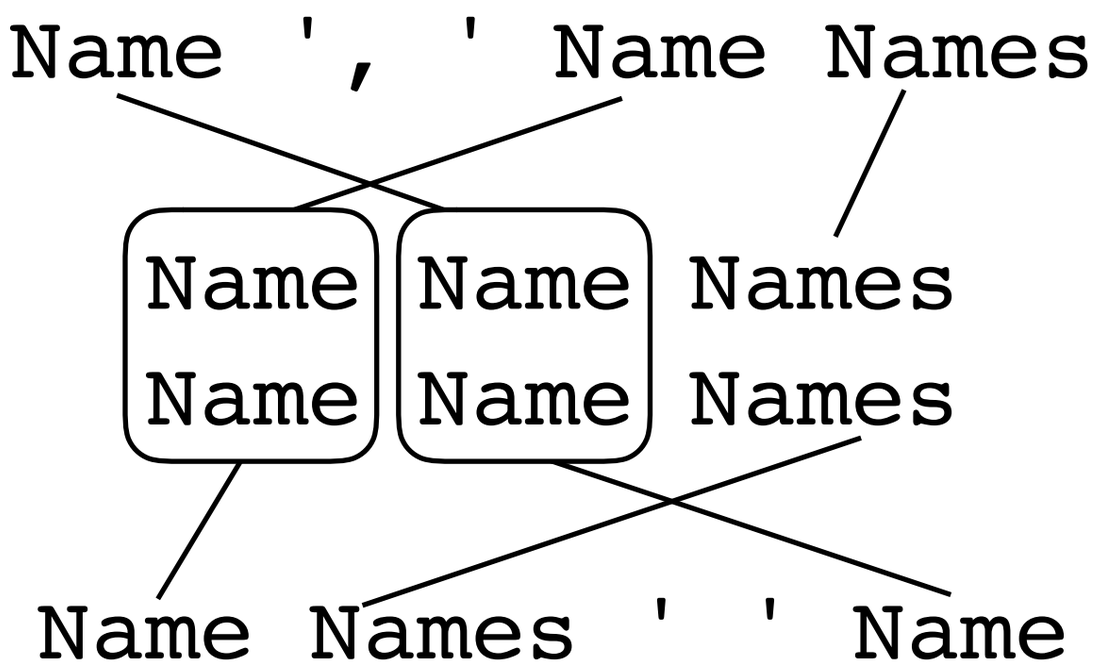


RigidSynth

Synthesize `dl` satisfying `dl :: DR <=> DS`

Difficulty lies in determining the permutations σ

Order the subcomponents of a DNF Regular Expression

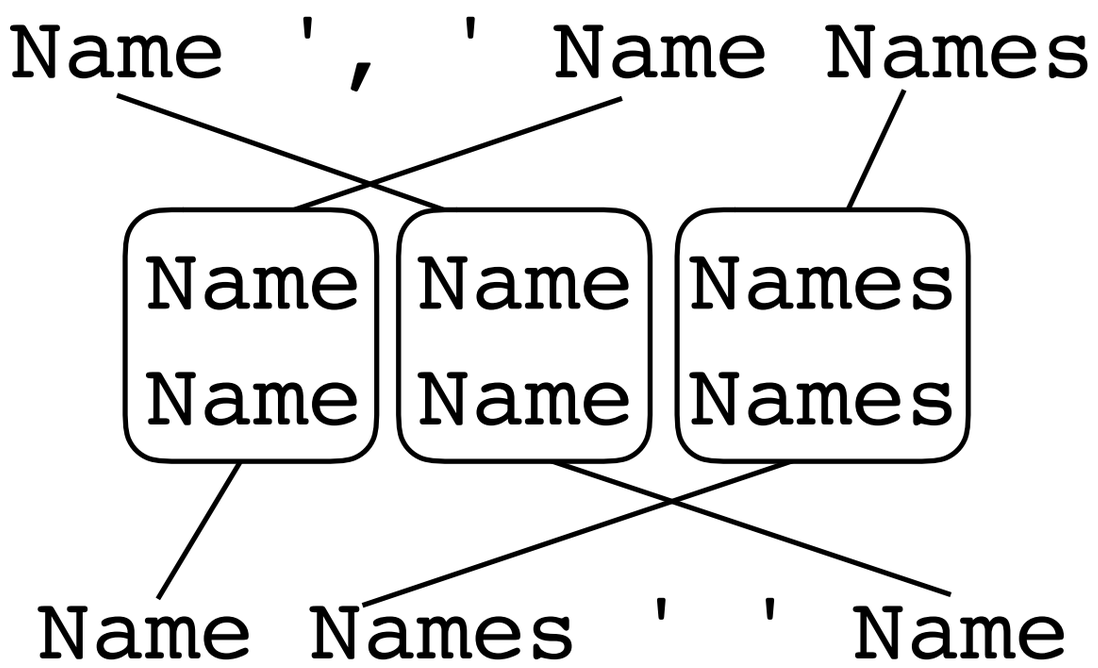


RigidSynth

Synthesize `dl` satisfying `dl :: DR <=> DS`

Difficulty lies in determining the permutations σ

Order the subcomponents of a DNF Regular Expression

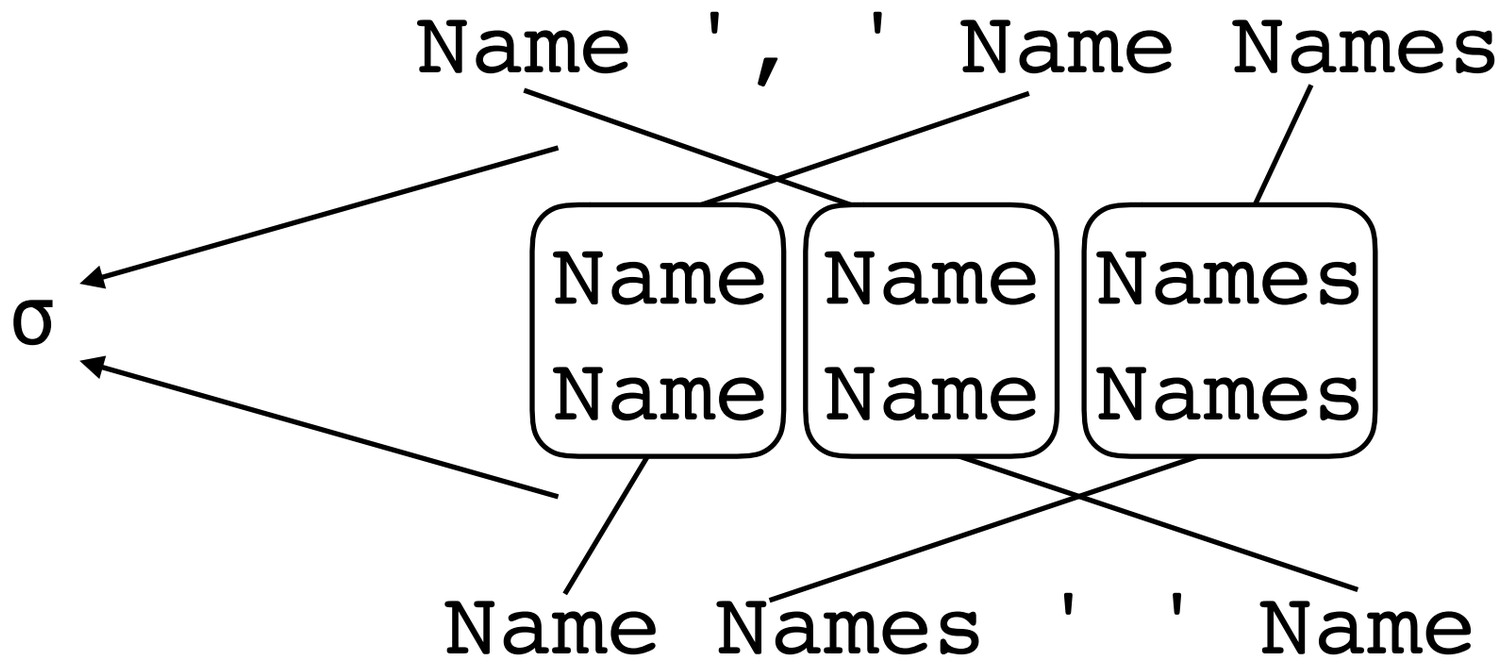


RigidSynth

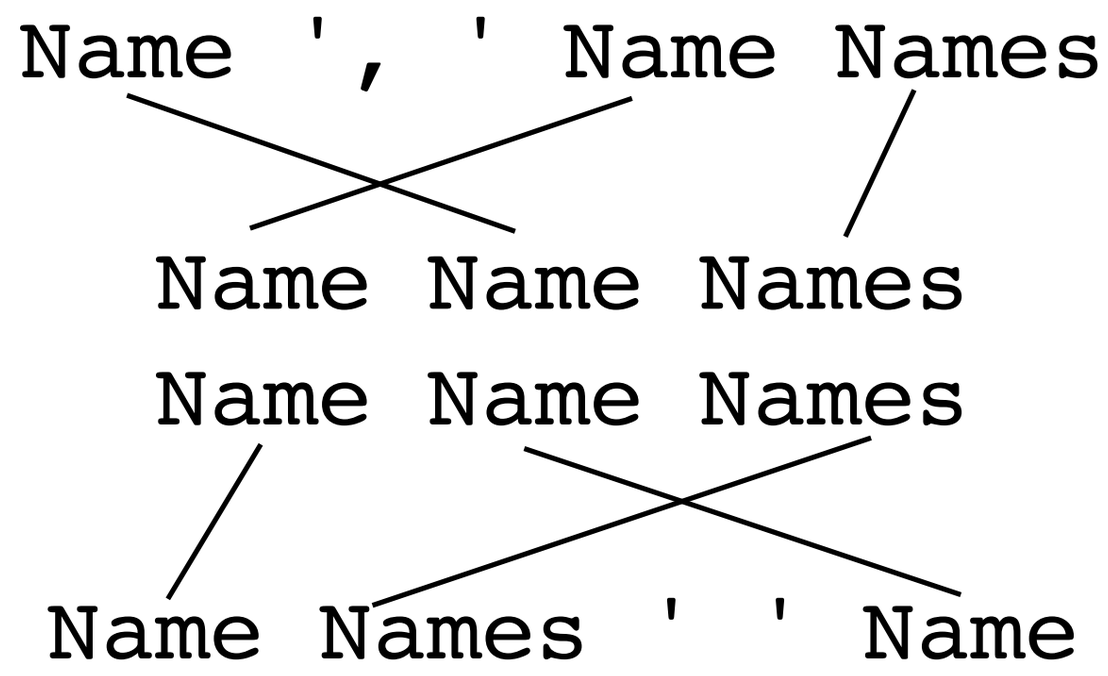
Synthesize `dl` satisfying `dl :: DR <=> DS`

Difficulty lies in determining the permutations σ

Order the subcomponents of a DNF Regular Expression



Satisfying Examples



Satisfying Examples

Add in additional information from the parsing of input/ output example strings!

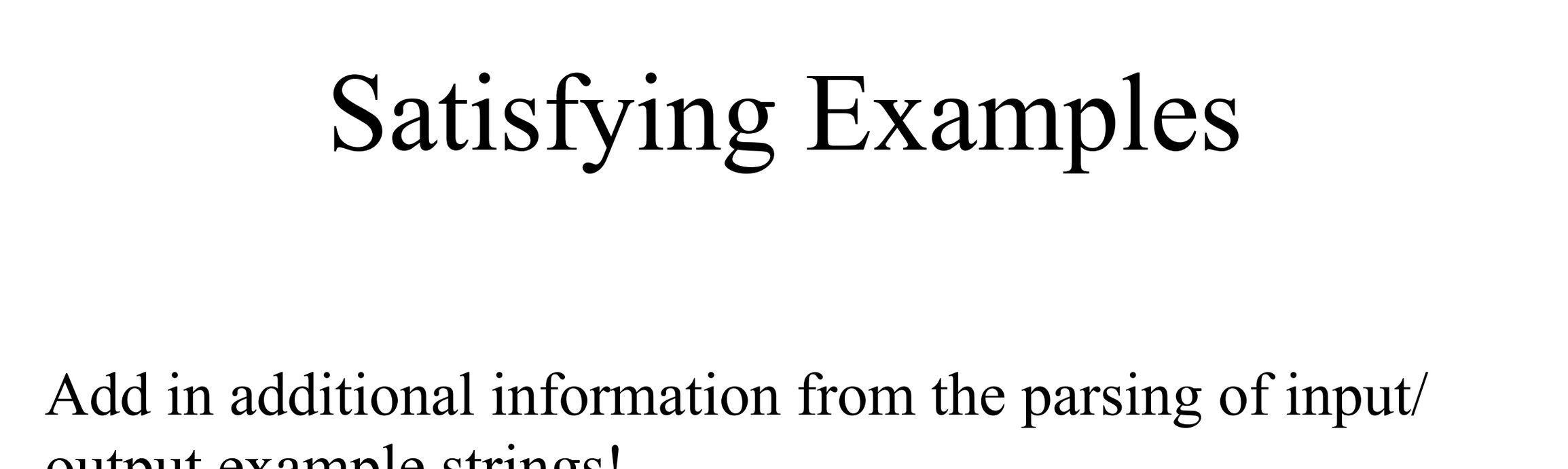
```
Name[Jones] ', ' Name[Simon] Names[Peyton]

   Name[Jones] Name[Simon] Names[Peyton]
   Name[Jones] Name[Simon] Names[Peyton]

Name[Simon] Names[Peyton] ' ' Name[Jones]
```

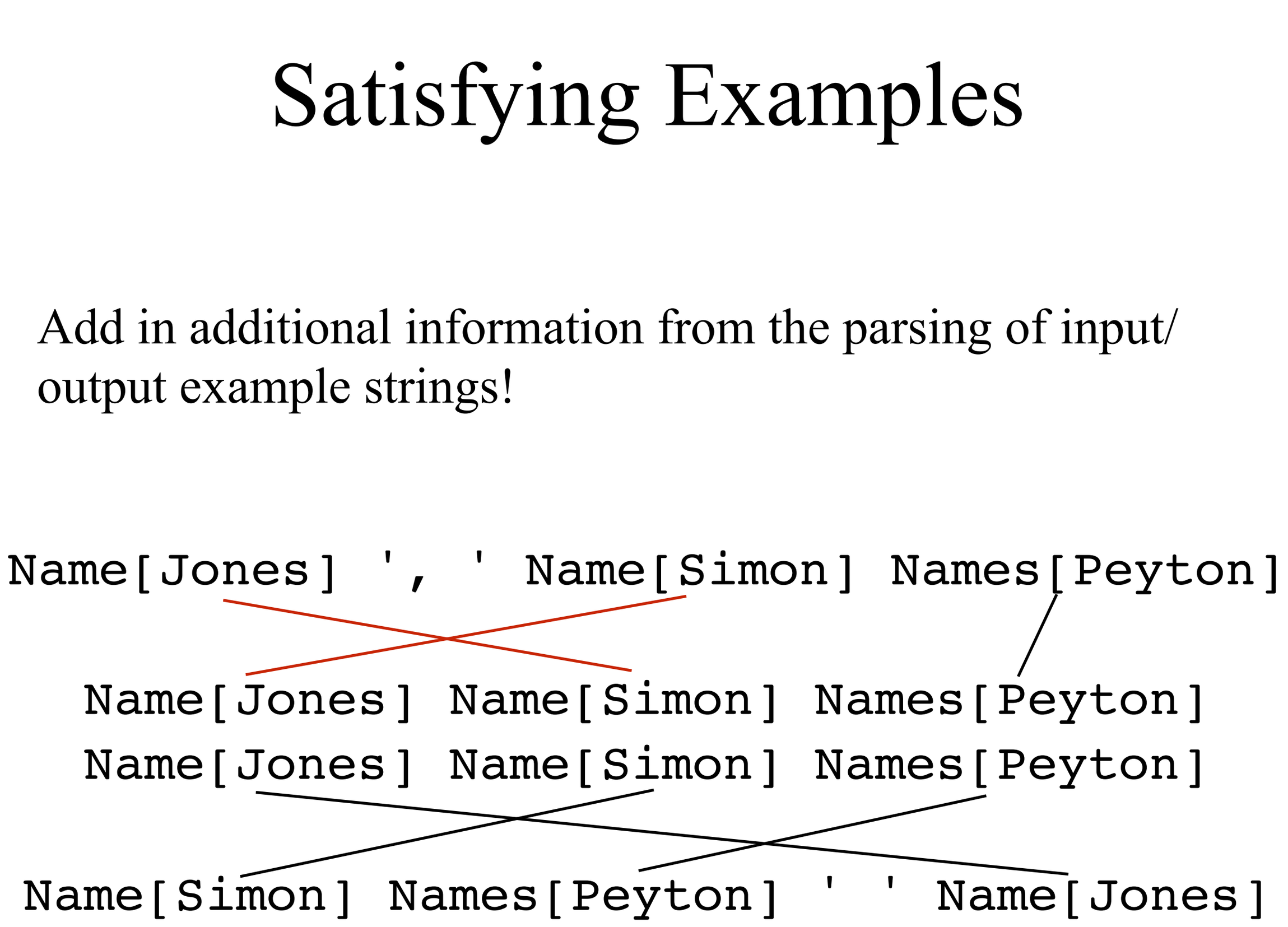
Satisfying Examples

Add in additional information from the parsing of input/ output example strings!



Satisfying Examples

Add in additional information from the parsing of input/ output example strings!



Evaluation

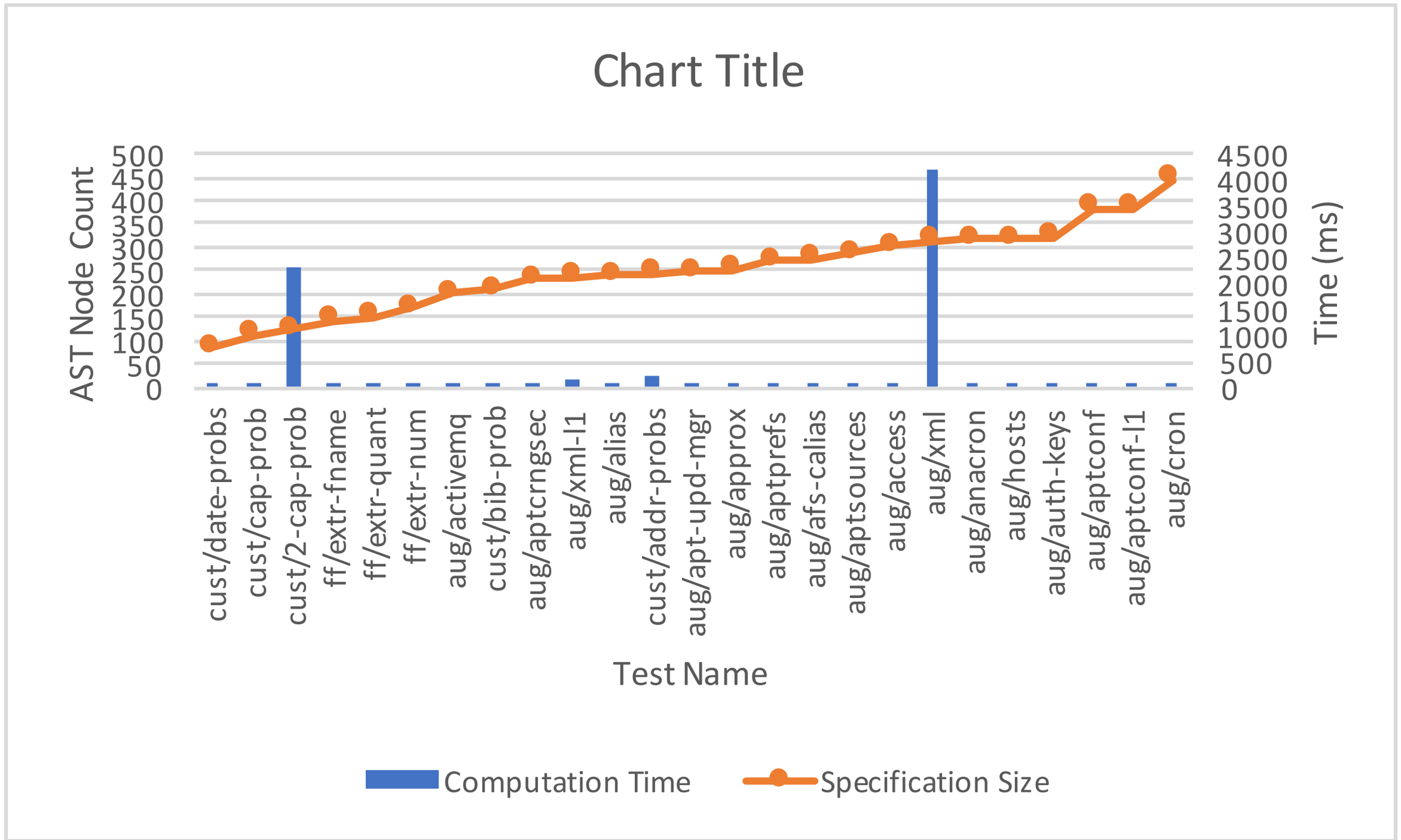
Evaluation Tests

- Custom built
  - Examples of good uses for it
  - Adversarial Examples
- FlashFill
  - Related work on string transformations
- Augeas
  - Bidirectional Language between configuration files and tree representation of the information

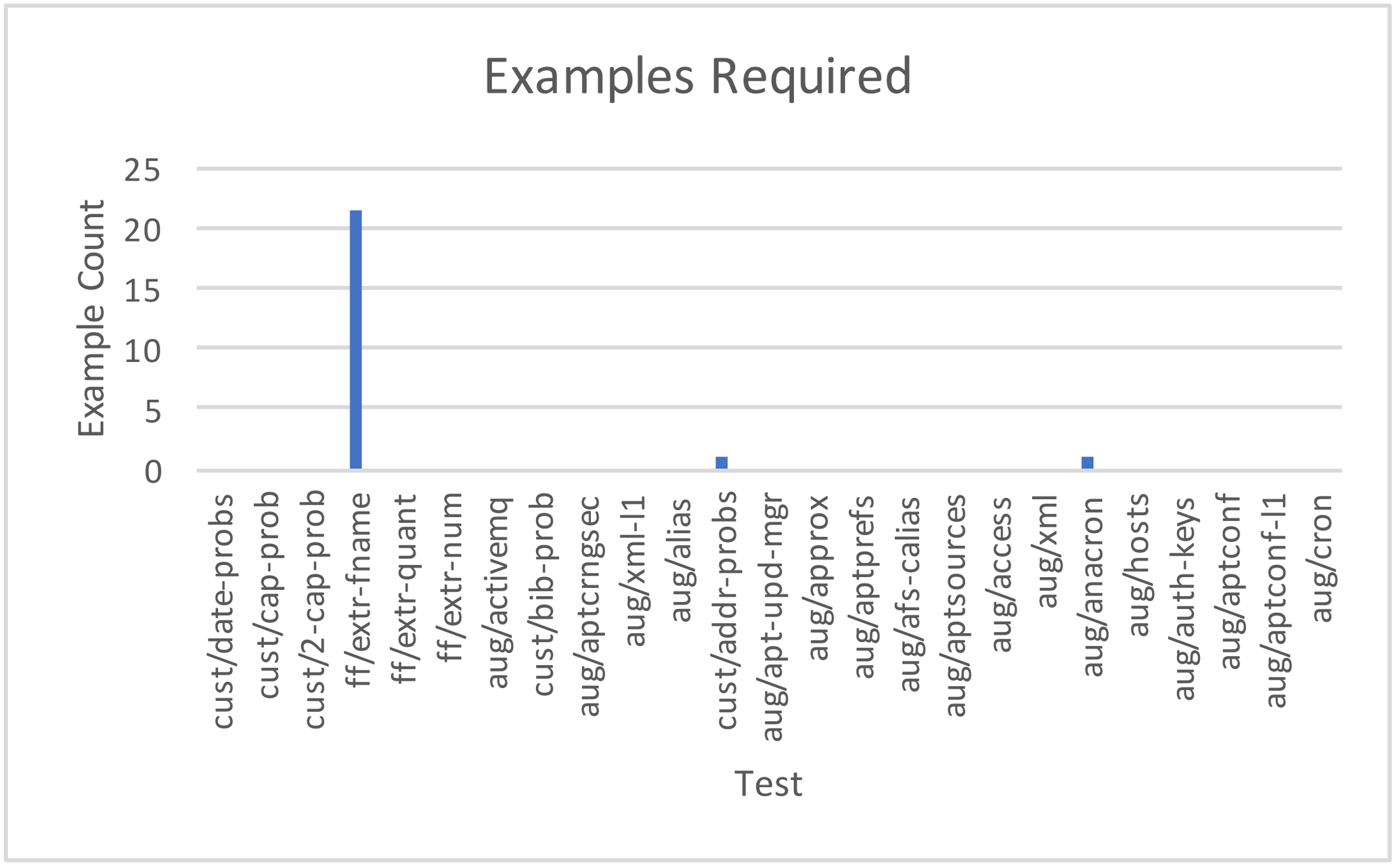
Evaluation Methods

- Evaluated on this very Mac!
  - 2.5 GHz Intel Core i7 processor
  - 16 GB of 1600 MHz DDR3
  - OS X Yosemite
- Average of 10 runs

Synthesis Time Graph



Examples Graph



What was hard?

- MANY were easy (13/25 synthesized in <10ms)
- custom 2-cap-prob (2.3s)
  - Exploits the fact we always distribute
  - Analogous 4-cap-prob takes 30 minutes
- Augeas's xml
  - Source and Target are very different
  - Large number of conversions needed to take place
    - 12 conversions required
    - 400 expansions explored

Surprising Facts

- Examples were often unnecessary
  - Tight type system makes few lenses per type
  - Usually little reordering took place
- How well it scales in practice!
  - Usually more complex programs become quickly impossible to synthesize
  - Difficulty comes from how different the formats are, not how complex the formats were

Related Work

- Type-Directed Synthesis [OZ ’15] [FOWZ ’16] [FCD ’15] [GKKP ’13] …
  - In general purpose programming languages
    - Type systems very different
- String Transformation Synthesis [G ’11] [LG ‘14] [PGDP ’14] [S ’16] [SG ’12] …
  - Did not take in specifications and examples

Synthesizing Bijective String Transformers

- Developed **DNF Lenses**
- Proven DNF Lenses **sound and complete** with respect to Bijective Lenses
- Developed an **efficient algorithm** for synthesis of DNF lenses
- **Evaluated our algorithm** on a suite of examples consisting of our own examples, examples derived from related work, and real world examples