Abstract Parsing for Two-staged Languages with Concatenation

Soonho Kong Wontae Choi Kwangkeun Yi

Programming Research Lab.
Seoul National University

{soon,wtchoi,kwang}@ropas.snu.ac.kr

2009/07/10

ROSAEC center 2nd Workshop

Why?

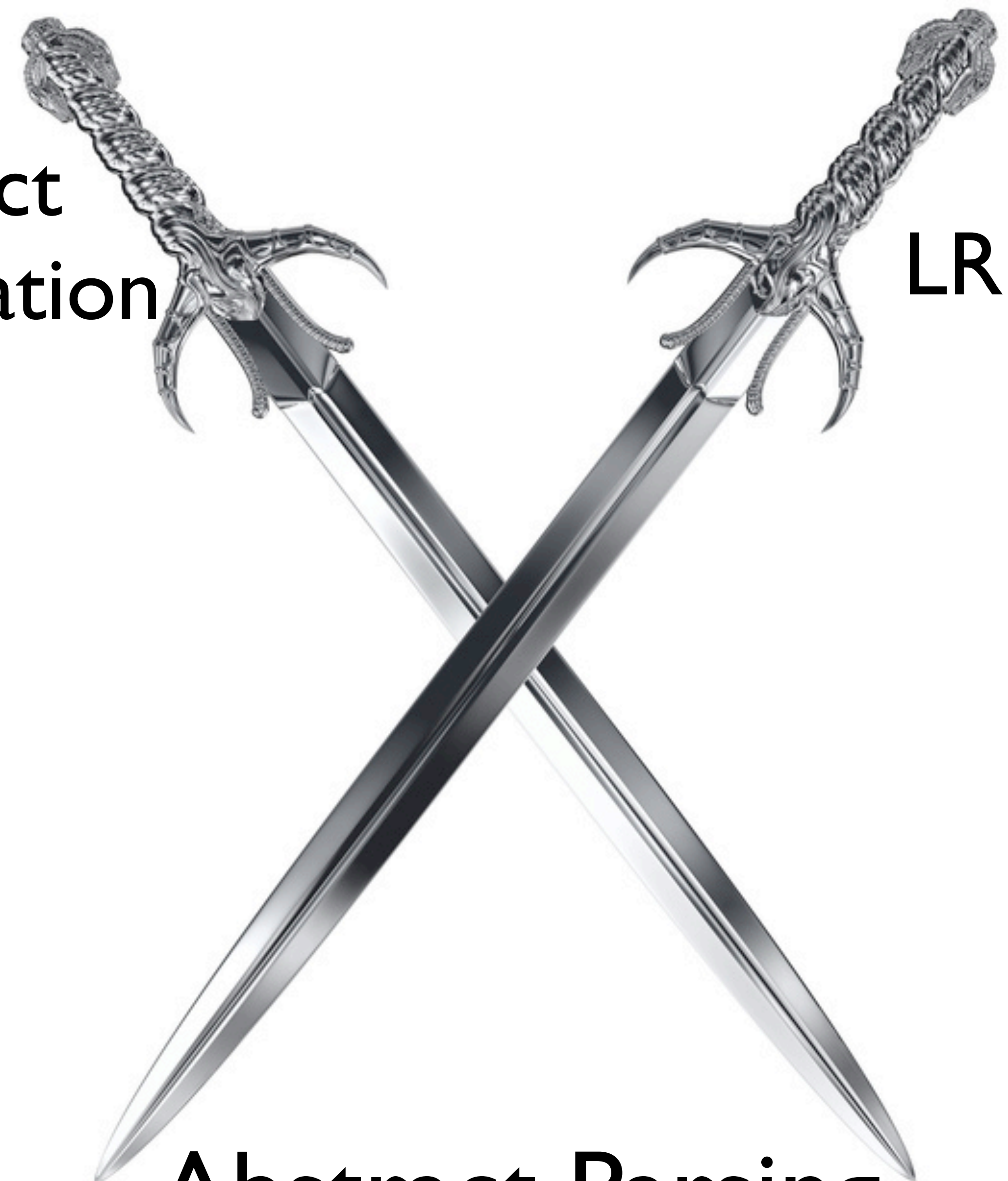
Why should I Listen to You?

Why?

Abstract Parsing is
a *powerful* static analysis technique
which has many applications.

Why Powerful?

Abstract Interpretation



LR Parsing

Abstract Parsing

Combine Two Powerful Techniques

Why Powerful?

Many Applications

- Syntax Check of Generated Programs in Two-Staged Languages
- Shape Analysis using Abstract Parsing
- Proof Carrying Code Framework for Program Generators

Why Powerful?

Many Applications

- Syntax Check of Generated Programs in Two-Staged Languages
- Shape Analysis using Abstract Parsing
- Proof Carrying Code Framework for Program Generators

Motivation

- Two-staged languages with Concatenation:
  Program generates programs
- Want to check:
  Syntax of generated programs

Language: Syntax & Semantics

Syntax

$$e \in \mathit{Exp} ::= x \mid \texttt{let}\ x\ e_1\ e_2 \mid \texttt{or}\ e_1\ e_2 \mid \texttt{re}\ x\ e_1\ e_2\ e_3 \mid \text{`}f$$

$$f \in \mathit{Frag} ::= x \mid \texttt{let} \mid \texttt{or} \mid \texttt{re} \mid \texttt{(} \mid \texttt{)} \mid f_1.f_2 \mid \texttt{,}e$$

Semantics

$\boxed{\sigma \vdash^0 e \Rightarrow v}$

$$\sigma \vdash^0 x \Rightarrow \sigma(x) \qquad \text{(variable)}$$

$$\frac{\sigma \vdash^0 e_1 \Rightarrow v \qquad \sigma[x \mapsto v] \vdash^0 e_2 \Rightarrow v'}{\sigma \vdash^0 \texttt{let}\ x\ e_1\ e_2 \Rightarrow v'} \qquad \text{(let binding)}$$

$$\frac{\sigma \vdash^0 e_1 \Rightarrow v}{\sigma \vdash^0 \texttt{or}\ e_1\ e_2 \Rightarrow v} \qquad \frac{\sigma \vdash^0 e_2 \Rightarrow v}{\sigma \vdash^0 \texttt{or}\ e_1\ e_2 \Rightarrow v} \qquad \text{(branch)}$$

$$\frac{\sigma \vdash^0 e_1 \Rightarrow v \qquad \sigma[x \mapsto v] \vdash^0 \mathit{loop}\ x\ e_2\ e_3 \Rightarrow v'}{\sigma \vdash^0 \texttt{re}\ x\ e_1\ e_2\ e_3 \Rightarrow v'} \qquad \text{(loop)}$$

$$\frac{\sigma \vdash^0 e_2 \Rightarrow v \qquad \sigma[x \mapsto v] \vdash^0 \mathit{loop}\ x\ e_2\ e_3 \Rightarrow v'}{\sigma \vdash^0 \mathit{loop}\ x\ e_2\ e_3 \Rightarrow v'}$$

$$\frac{\sigma \vdash^0 e_3 \Rightarrow v}{\sigma \vdash^0 \mathit{loop}\ x\ e_2\ e_3 \Rightarrow v}$$

$$\frac{\sigma \vdash^1 f \Rightarrow v}{\sigma \vdash^0 \text{`}f \Rightarrow v} \qquad \text{(back quote)}$$

$\boxed{\sigma \vdash^1 f \Rightarrow v}$

$$\frac{}{\sigma \vdash^1 x \Rightarrow x} \qquad \frac{}{\sigma \vdash^1 \texttt{let} \Rightarrow \texttt{let}} \qquad \text{(token)}$$

$$\frac{}{\sigma \vdash^1 \texttt{or} \Rightarrow \texttt{or}} \qquad \frac{}{\sigma \vdash^1 \texttt{re} \Rightarrow \texttt{re}}$$

$$\frac{}{\sigma \vdash^1 \texttt{(} \Rightarrow \texttt{(}} \qquad \frac{}{\sigma \vdash^1 \texttt{)} \Rightarrow \texttt{)}}$$

$$\frac{\sigma \vdash^1 f_1 \Rightarrow v_1 \qquad \sigma \vdash^1 f_2 \Rightarrow v_2}{\sigma \vdash^1 f_1.f_2 \Rightarrow v_1 v_2} \qquad \text{(concatenation)}$$

$$\frac{\sigma \vdash^0 e \Rightarrow v}{\sigma \vdash^1 \texttt{,}e \Rightarrow v} \qquad \text{(comma)}$$

Language: Example

```
let x `a
let y `b
   or x y

=> a
 | b
```

```
let x = `a
let y = `b
   `x.y.,y

=> x y b
```

```
re x `a `b x

=> a
 | b
```

```
re x `a (`b.,x) x

=> a
 | b a
 | b b a
 | b b b a
 ...
```

```
re x `a (`or . ,x) (`,x . b)

=> a b
 | or a b
 | or or a b
 | or or or a b
 | ...
```

Language: Collecting Semantics

$$
\begin{aligned}
\mathit{Code} &= \mathit{Token\ sequence} \\
\sigma \in \mathit{Env} &= \mathit{Var} \to \mathit{Code} \\
[\![e]\!]^0 &\in 2^{\mathit{Env}} \to 2^{\mathit{Code}} \\
[\![f]\!]^1 &\in 2^{\mathit{Env}} \to 2^{\mathit{Code}}
\end{aligned}
$$

$$
\begin{aligned}
[\![x]\!]^0\Sigma &= \{\sigma(x) \mid \sigma \in \Sigma\} \\
[\![\texttt{let}\ x\ e_1\ e_2]\!]^0\Sigma &= \bigcup_{\sigma\in\Sigma} \bigcup_{c \in [\![e_1]\!]^0\{\sigma\}} [\![e_2]\!]^0\{\sigma[x \mapsto c]\} \\
[\![\texttt{or}\ e_1\ e_2]\!]^0\Sigma &= [\![e_1]\!]^0\Sigma \cup [\![e_2]\!]^0\Sigma \\
[\![\texttt{re}\ x\ e_1\ e_2\ e_3]\!]^0\Sigma &= \bigcup_{\sigma\in\Sigma} [\![e_3]\!]^0\{\sigma[x \mapsto c] \mid c \in \\
&\qquad \mathit{fix}\lambda C.[\![e_1]\!]^0\{\sigma\} \cup [\![e_2]\!]^0\{\sigma[x \mapsto c'] \mid c' \in C\}\} \\
[\![\text{`}f]\!]^0\Sigma &= [\![f]\!]^1\Sigma \\
[\![x]\!]^1\Sigma &= \{x\} \\
[\![\texttt{let}]\!]^1\Sigma &= \{\texttt{let}\} \\
[\![\texttt{or}]\!]^1\Sigma &= \{\texttt{or}\} \\
[\![\texttt{re}]\!]^1\Sigma &= \{\texttt{re}\} \\
[\![\texttt{(}]\!]^1\Sigma &= \{\texttt{(}\} \\
[\![\texttt{)}]\!]^1\Sigma &= \{\texttt{)}\} \\
[\![f_1.f_2]\!]^1\Sigma &= \bigcup_{\sigma\in\Sigma} \{xy \mid x \in [\![f_1]\!]^1\{\sigma\} \wedge y \in [\![f_2]\!]^1\{\sigma\}\} \\
[\![,e]\!]^1\Sigma &= [\![e]\!]^0\Sigma
\end{aligned}
$$

ROSAEC center
Research On Software Analysis for Error-free Computing
소프트웨어 무결점 연구센터 KOSEF ERC

Language: Collecting Semantics

- Example

```
re x `a (`or . ,x) (`,x . b)


=> a b
 | or a b
 | or or a b
 | or or or a b
 | ...
```

$$[\![\texttt{re}\ x\ \text{`}a\ (\text{`}\texttt{or}\ .\ ,x)\ (\text{`},\texttt{x}\ .\ \texttt{b})]\!]^0\{\sigma_0\}$$

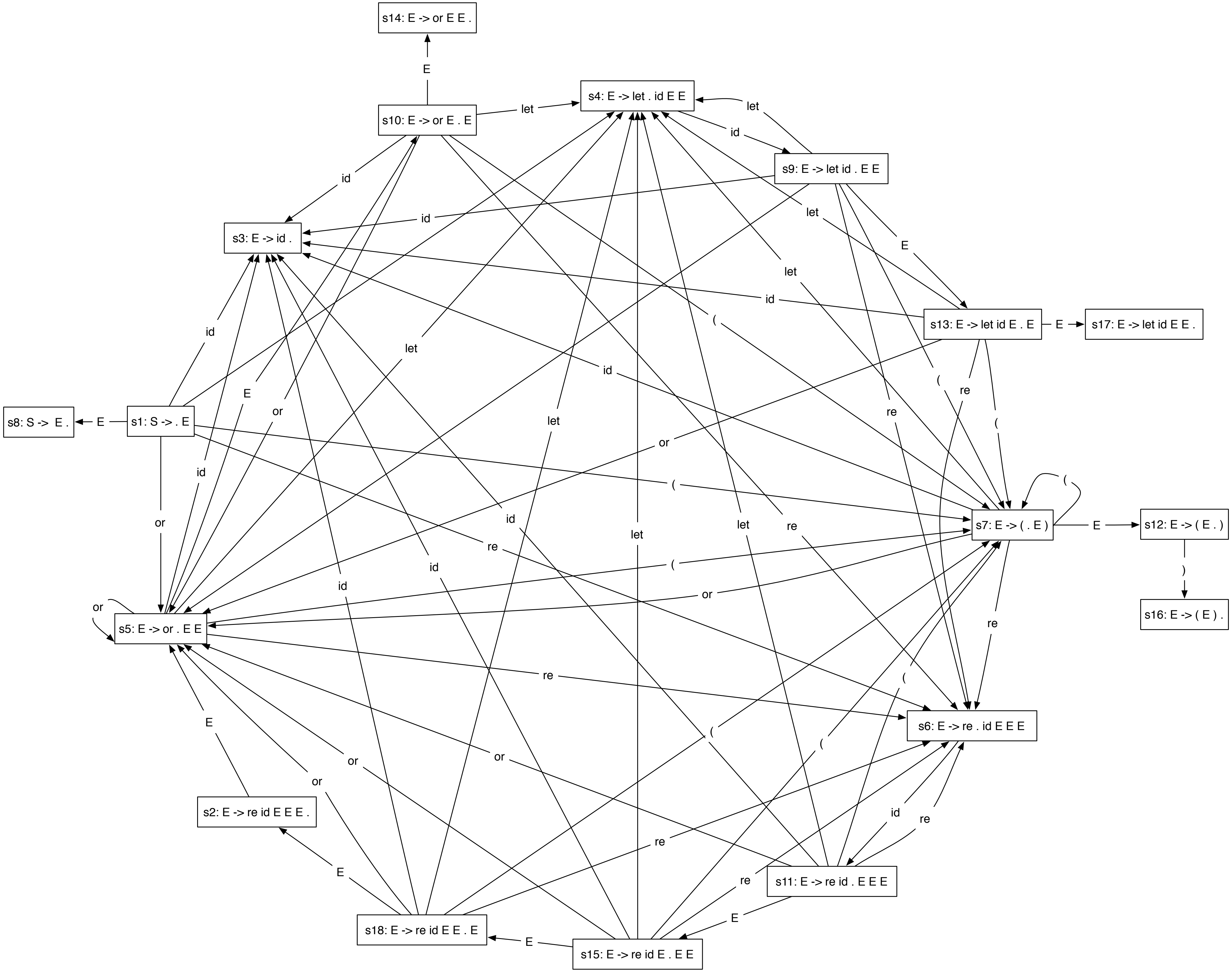
$$= \{\texttt{a}\ \texttt{b}, \texttt{or}\ \texttt{a}\ \texttt{b}, \texttt{or}\ \texttt{or}\ \texttt{a}\ \texttt{b}, \texttt{or}\ \texttt{or}\ \texttt{or}\ \texttt{a}\ \texttt{b}, \ldots\}$$

LR Parsing

- Determine whether input string S conforms to the grammar G
- In our case, the reference grammar is

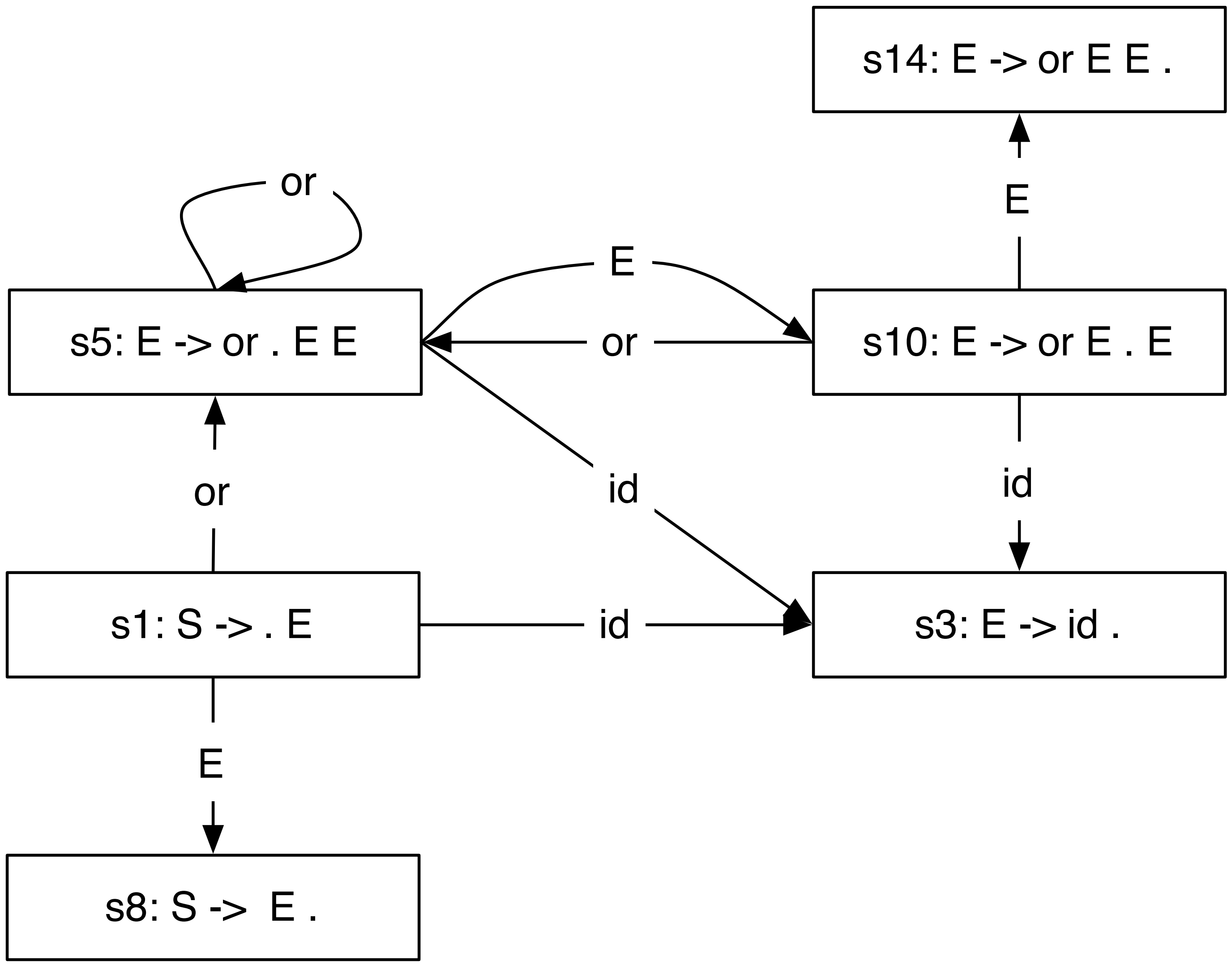
$$e \in \mathit{Exp} ::= x \mid \texttt{let}\ x\ e_1\ e_2 \mid \texttt{or}\ e_1\ e_2 \mid \texttt{re}\ x\ e_1\ e_2\ e_3$$

- LR parser generator builds a state machine for the given grammar.



Goto controller of the LR(0) parser for the reference grammar

LR Parsing



Part of goto controller of the LR(0) parser for the reference grammar

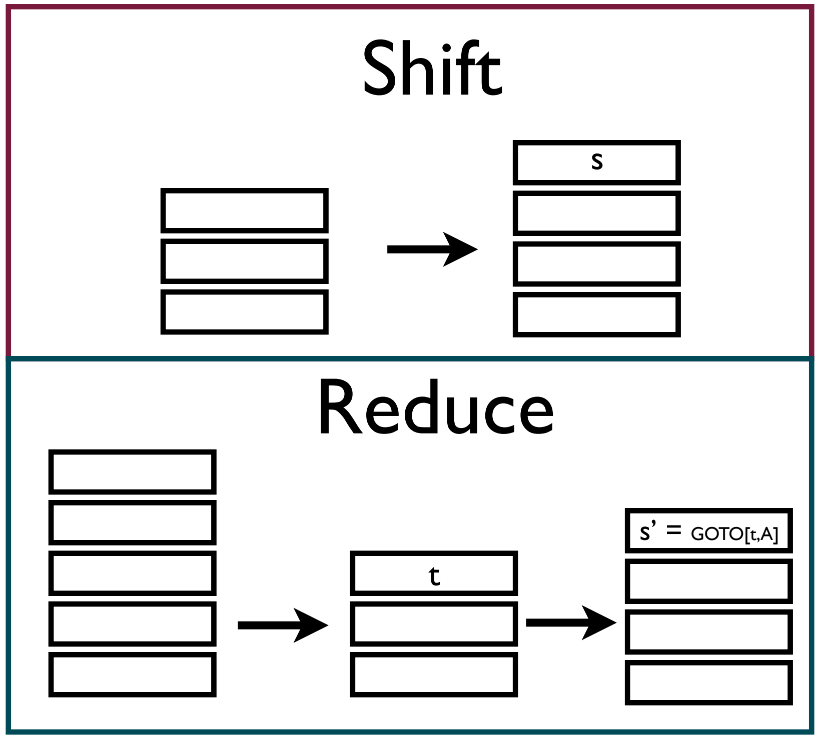
LR Parsing

- Atomic function : $parse\_action : Token \rightarrow P \rightarrow P$

Algorithm 1 $parse\_action$ algorithm

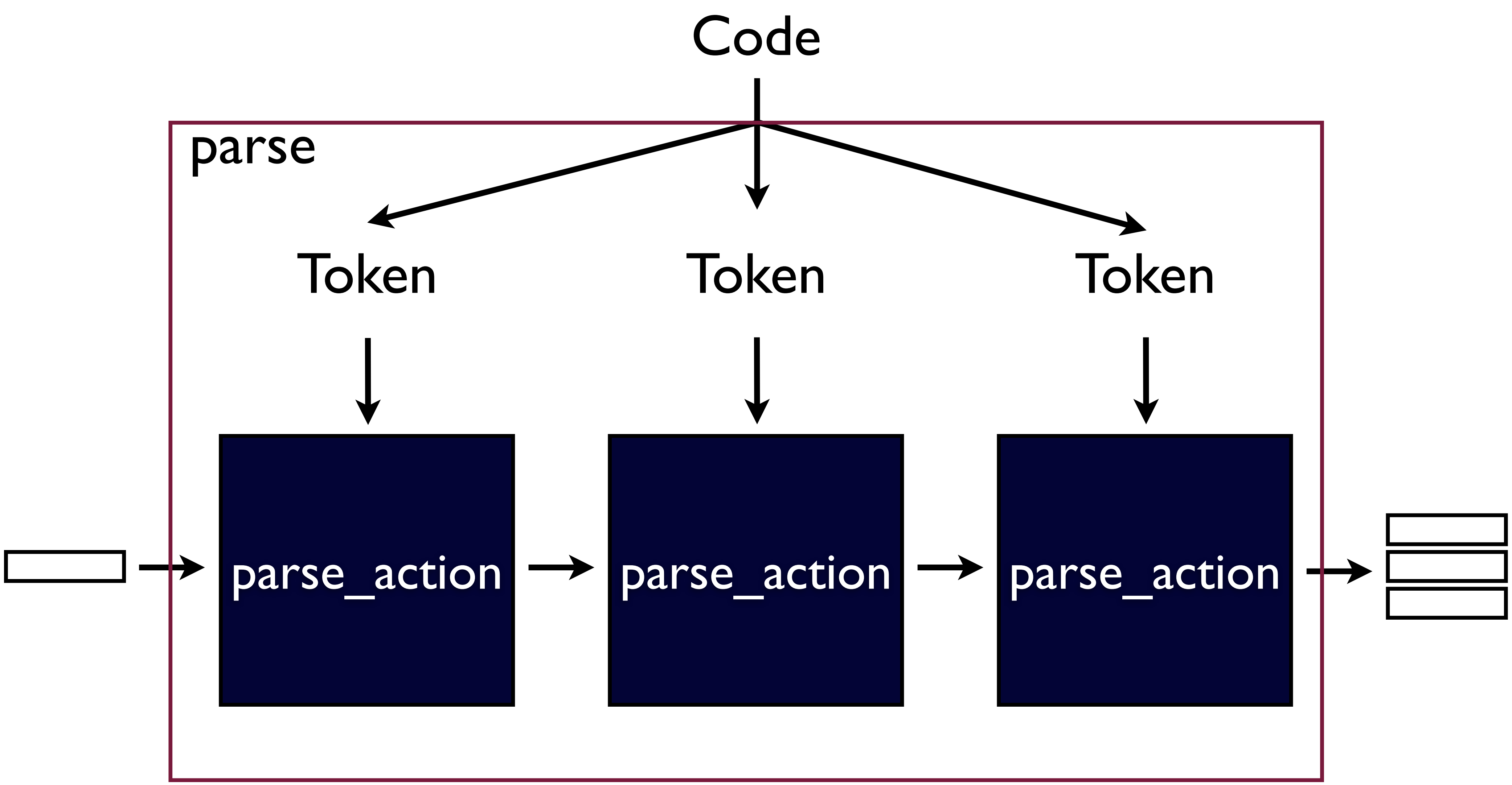
---

procedure $parse\_action(p, t)$
    $s_{top} \leftarrow$ the state on top of stack p
    if $ACTION[s_{top}, t]$ = shift s **then**
        push s onto the stack p
        return p
    else if $ACTION[s_{top}, t]$ = reduce $A \rightarrow \beta$ **then**
        pop $|\beta|$ symbol off the stack p
        $s_{top} \leftarrow$ the state on top of stack p
        push $GOTO[s_{top}, A]$ onto the stack p
        return $parse\_action(p, t)$
    end if
end procedure



LR Parsing

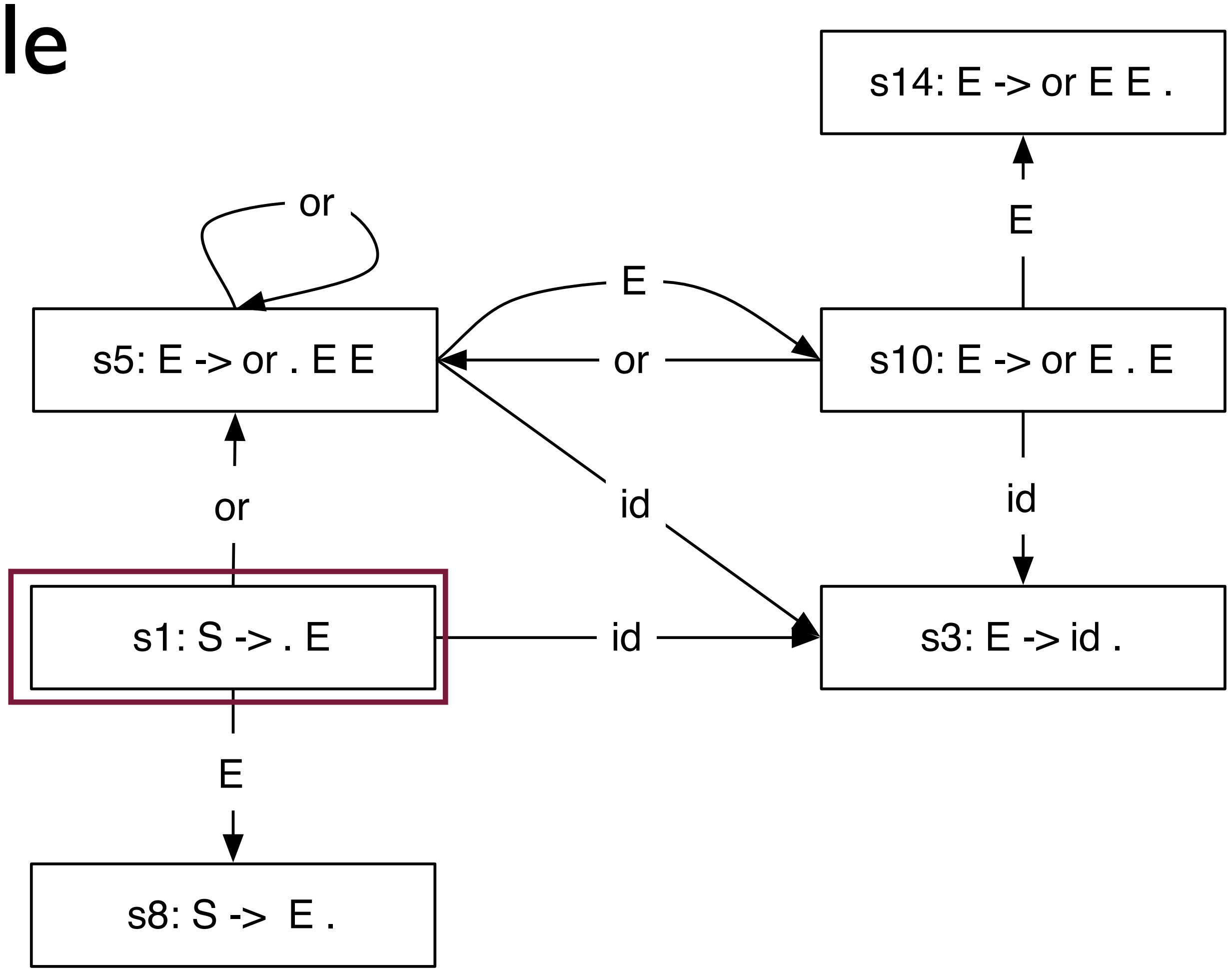
- Parsing is composition of parse_action



$$parse : Code \rightarrow P \rightarrow P$$

LR Parsing

- Example

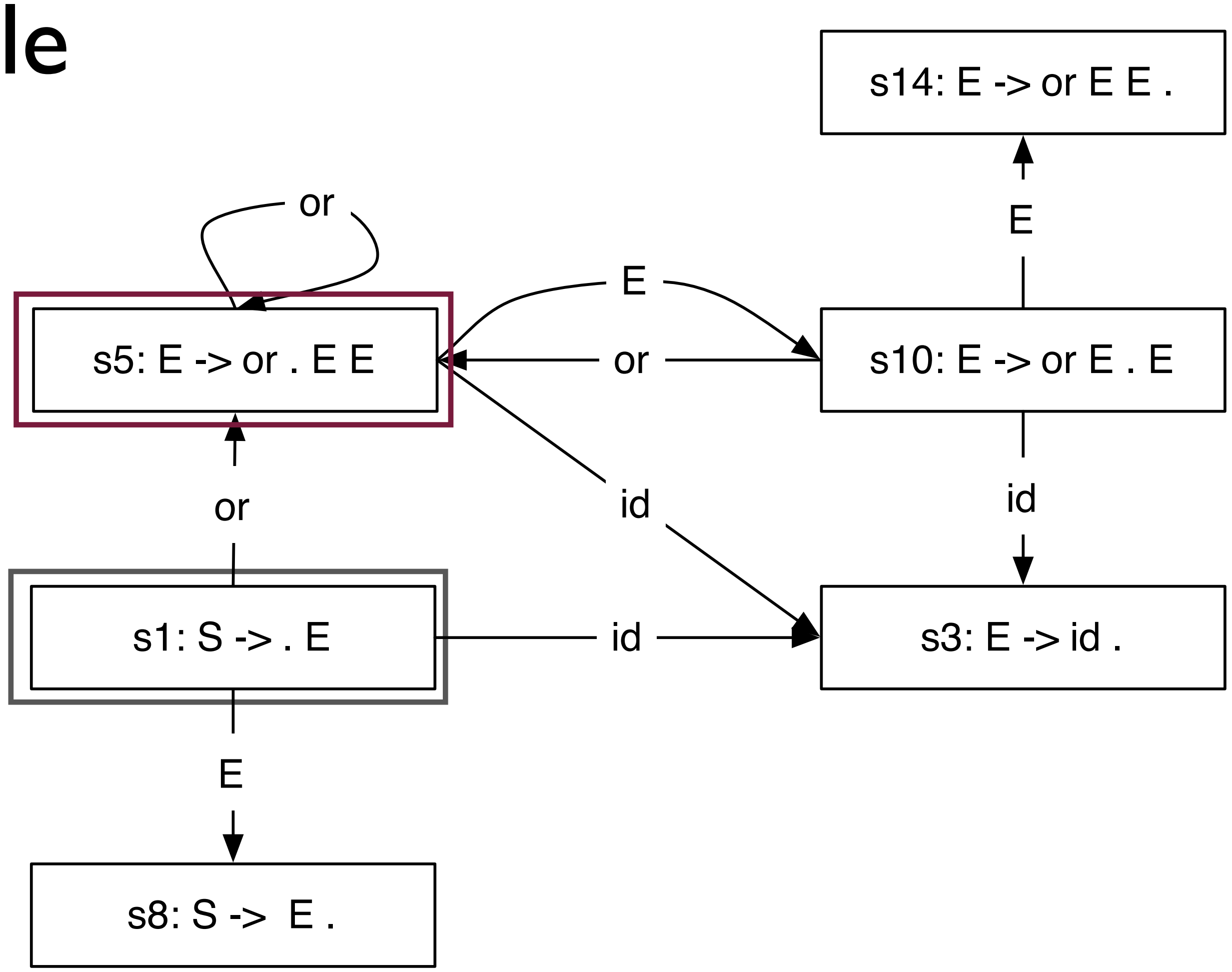


. <u>or</u> a b

s_1

LR Parsing

- Example

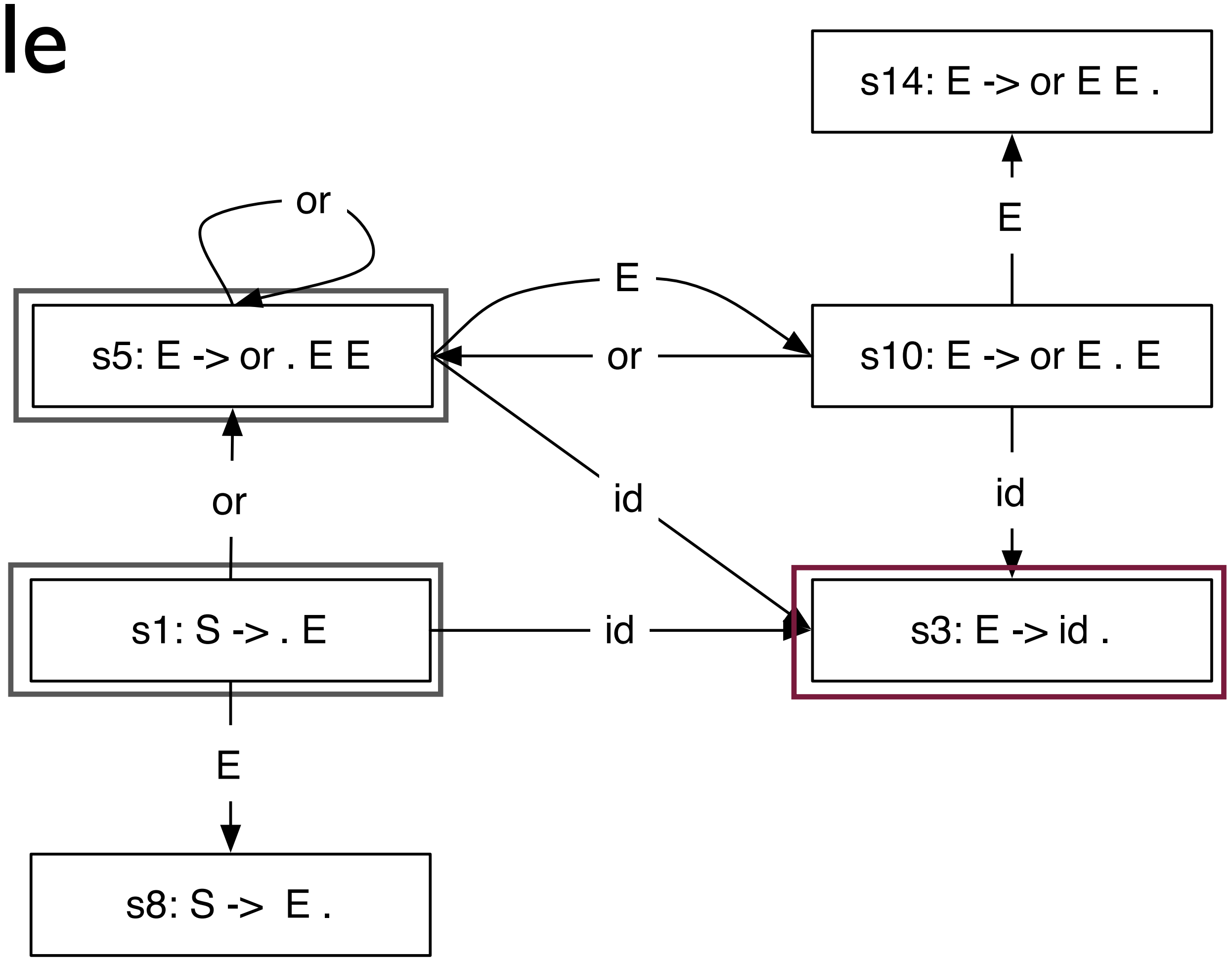


or . <u>a</u> b

$s_5 s_1$

LR Parsing

- Example

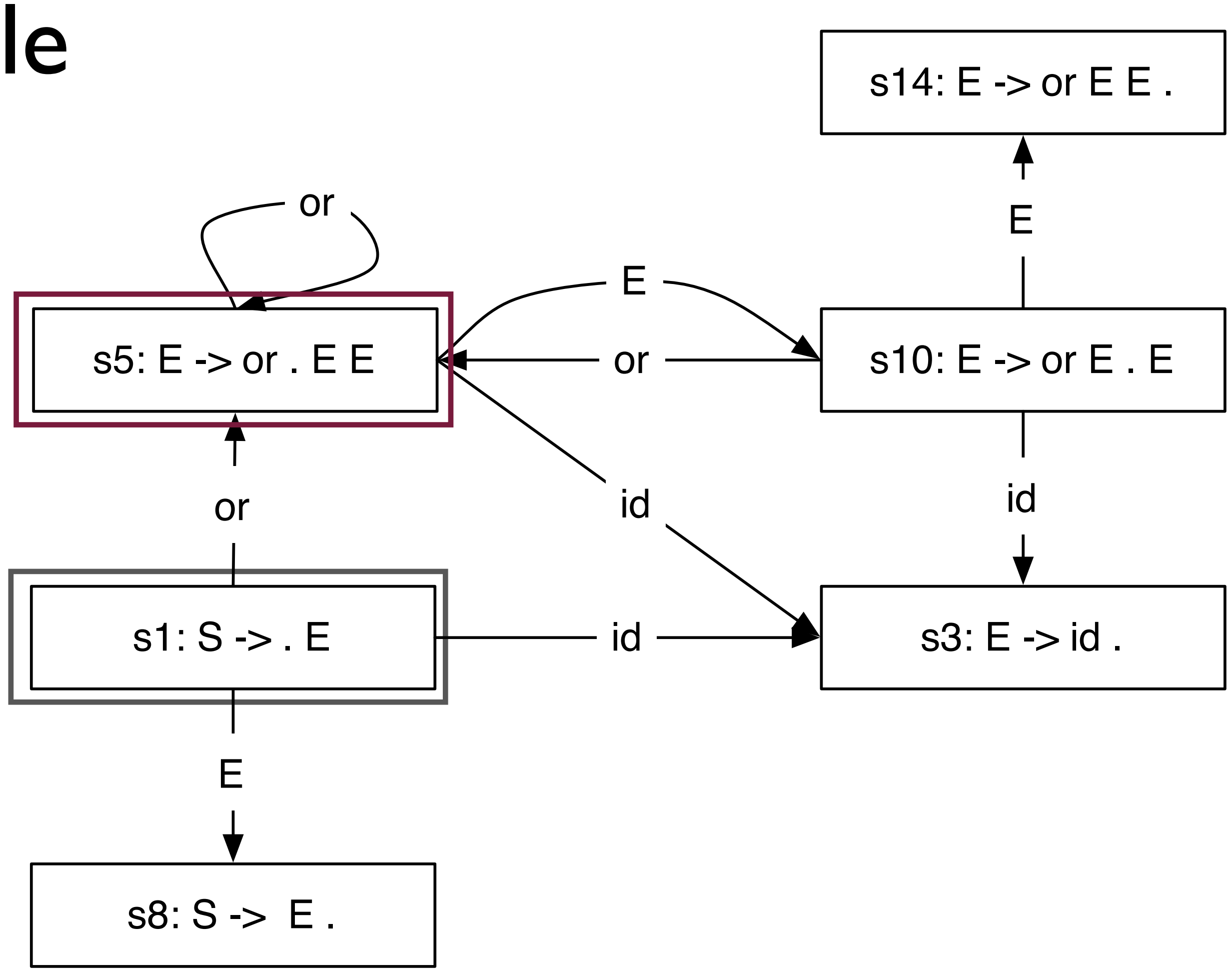


or a . b

$s_3 s_5 s_1$

LR Parsing

- Example

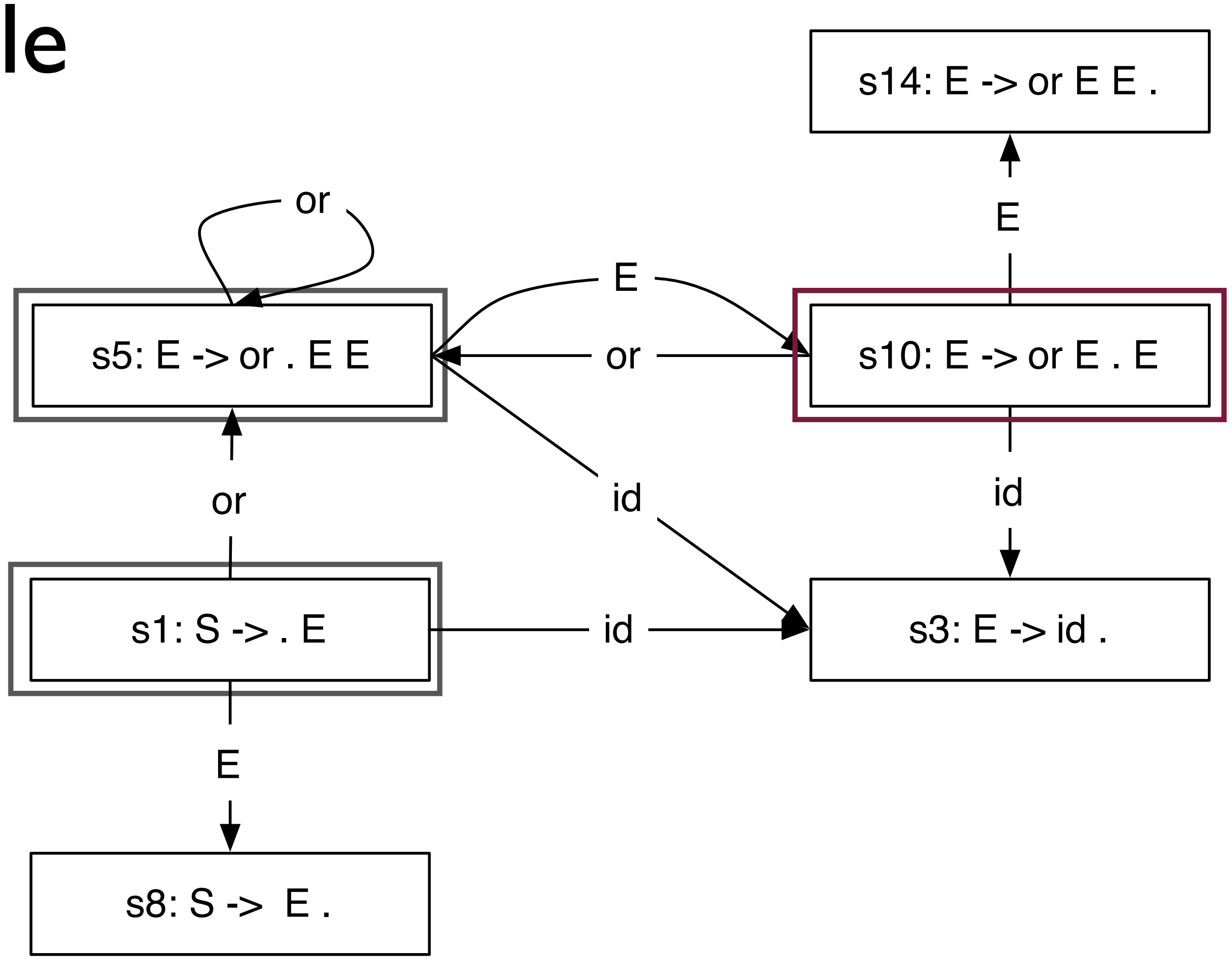


`or a . b`

$s_5 s_1$

LR Parsing

- Example

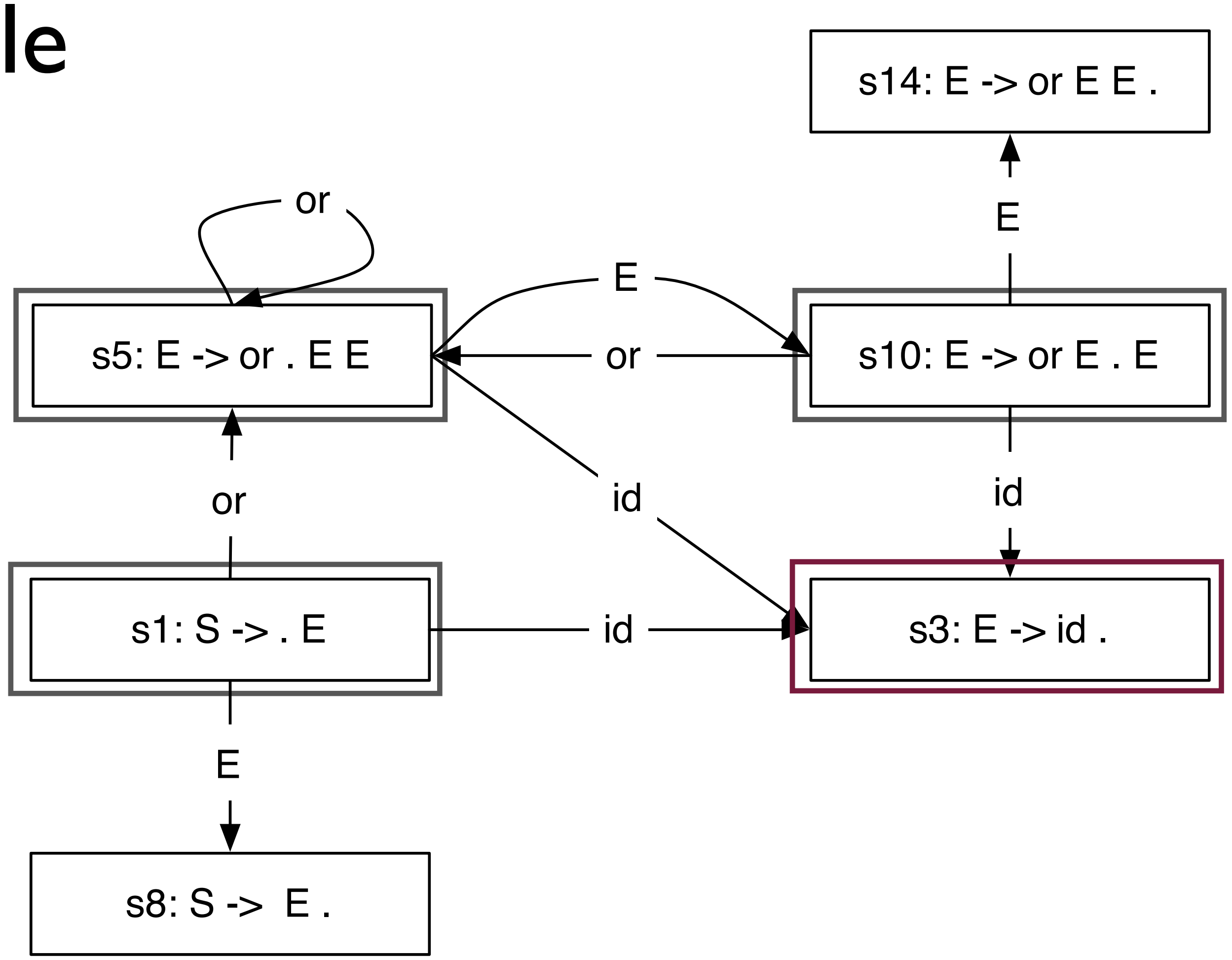


`or a . b`

$s_{10} s_5 s_1$

LR Parsing

- Example

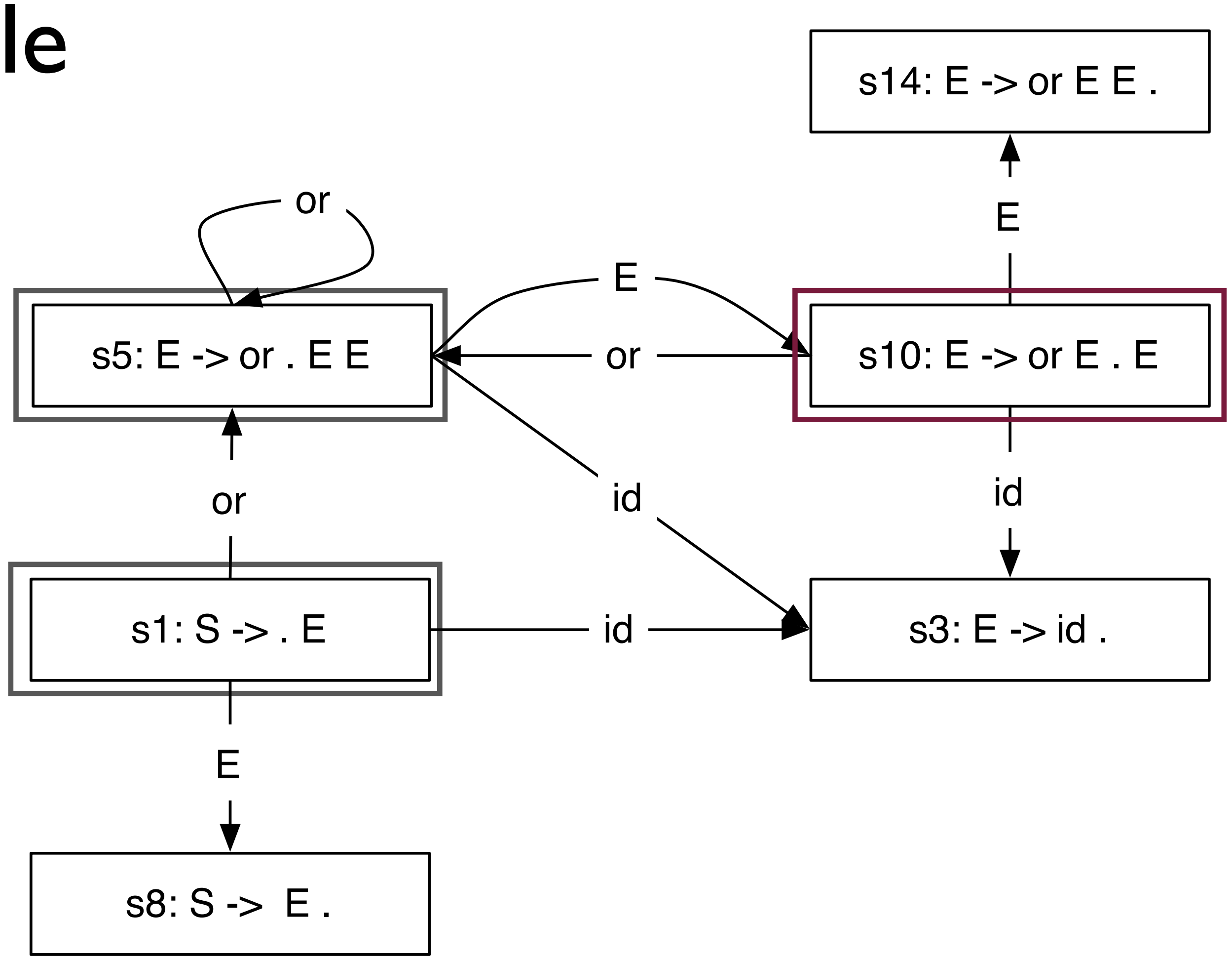


`or a b .`

$s_3 s_{10} s_5 s_1$

LR Parsing

- Example

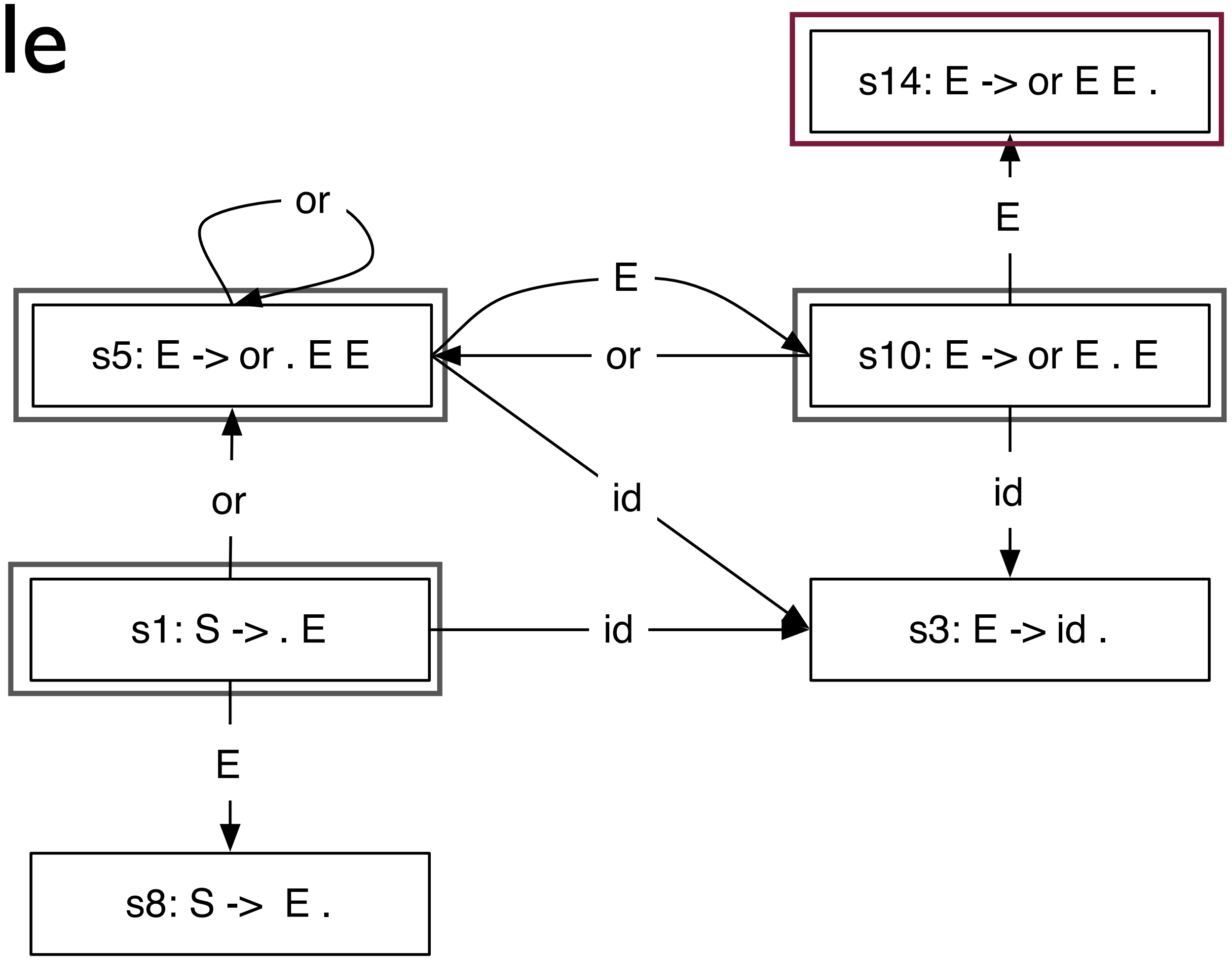


or a b .

$s_{10} s_5 s_1$

LR Parsing

- Example

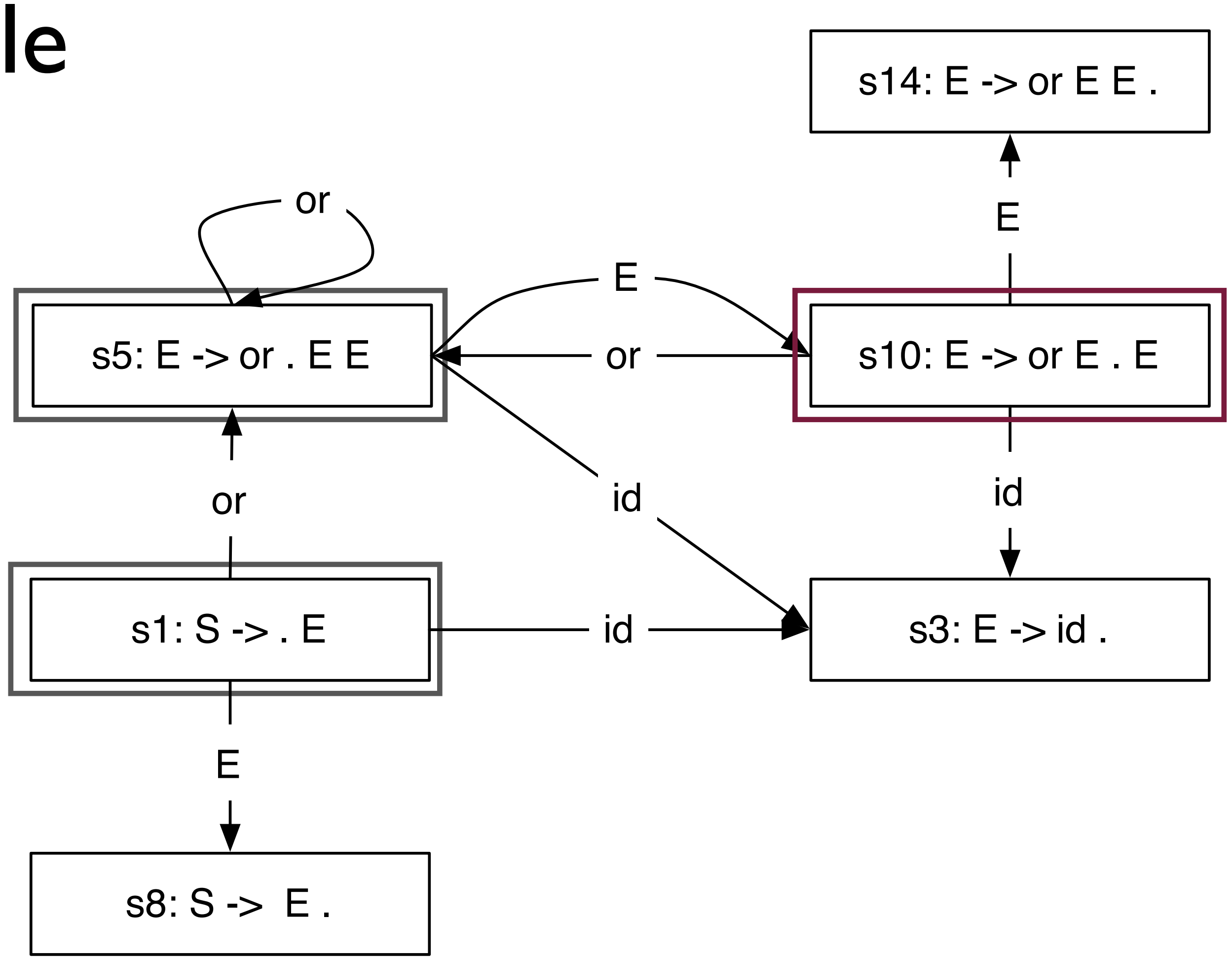



```
or a b .
```

$s_{14} s_{10} s_5 s_1$

LR Parsing

- Example

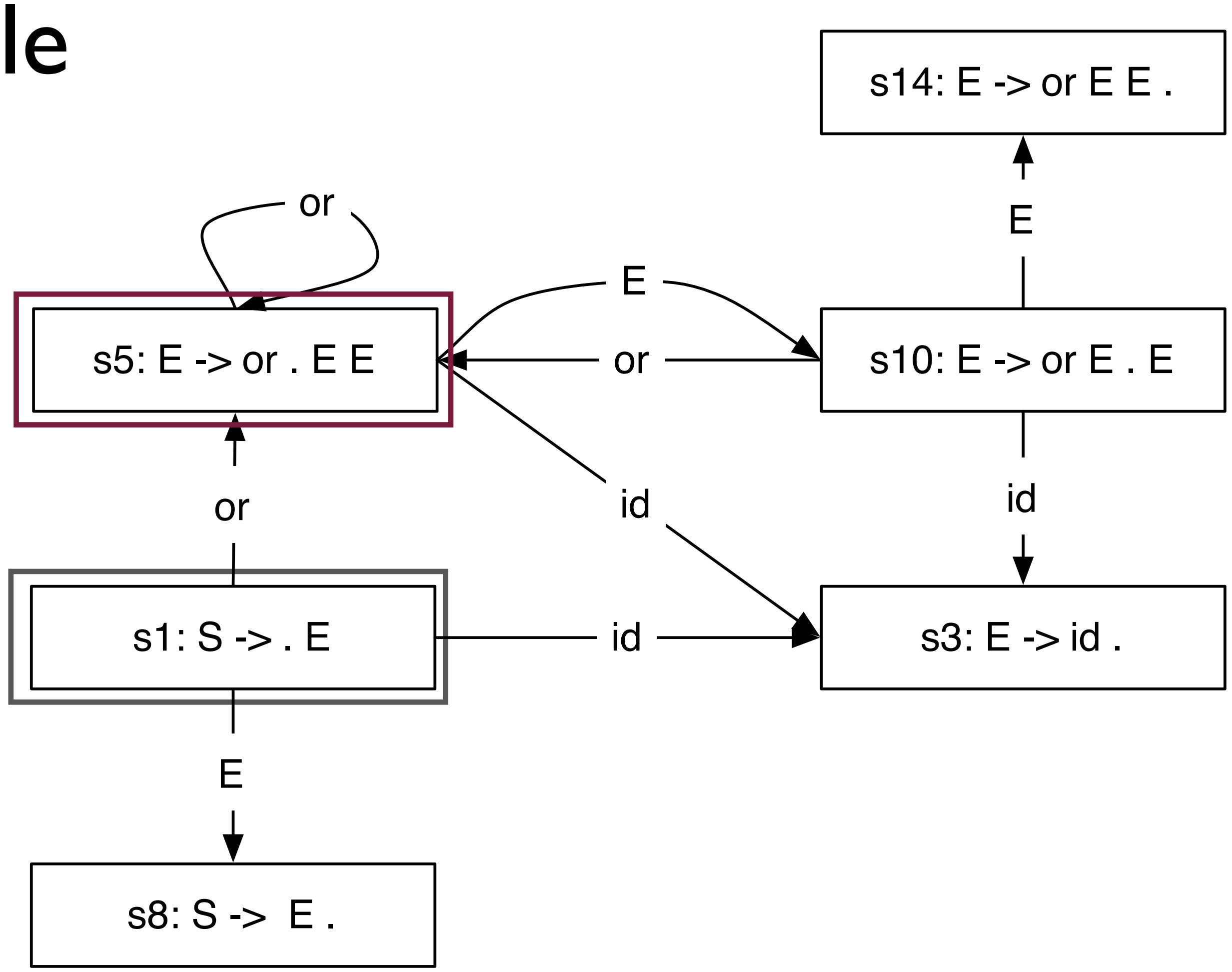


`or a b .`

$s_{10} s_5 s_1$

LR Parsing

- Example

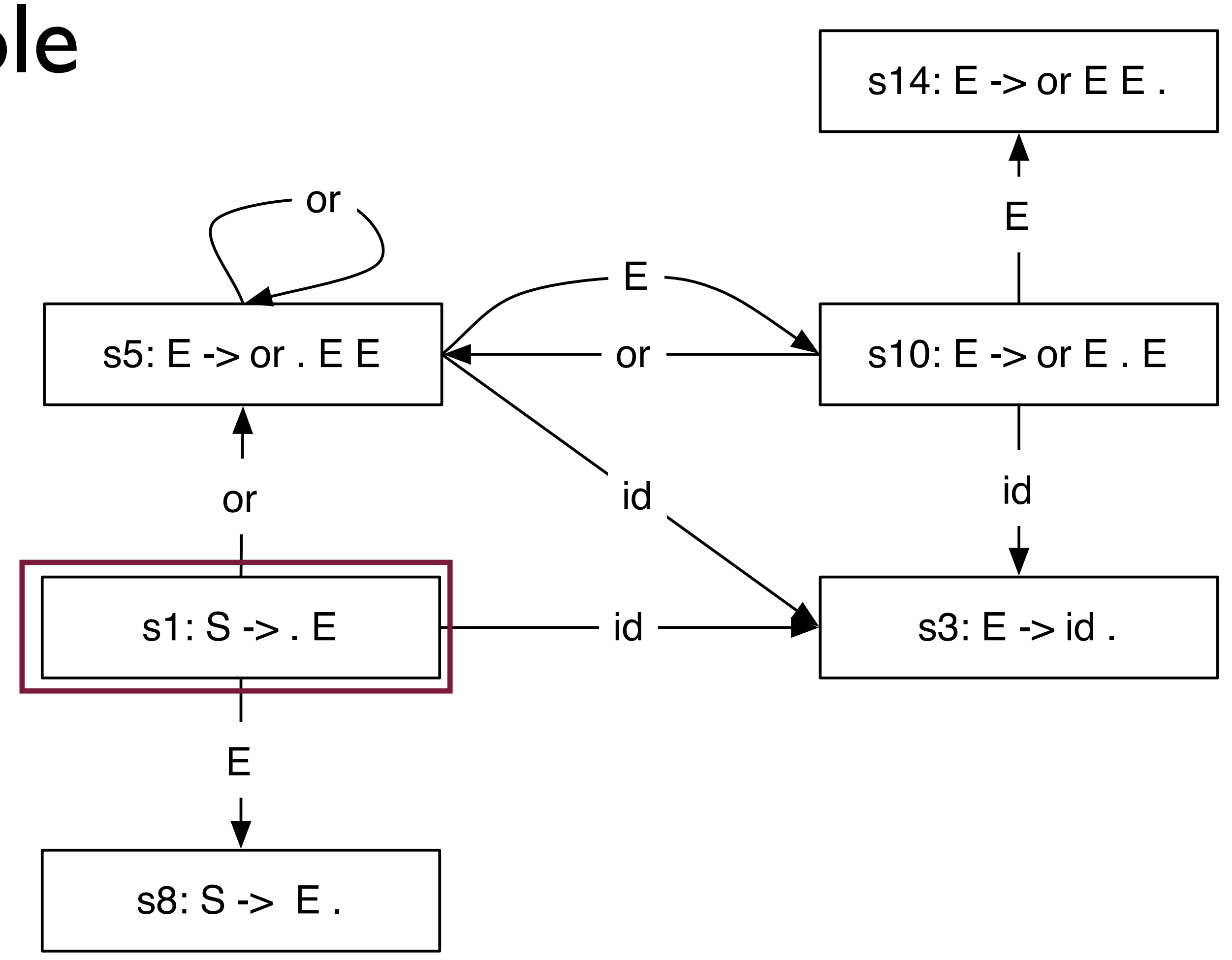


`or a b .`

$s_5 s_1$

LR Parsing

- Example

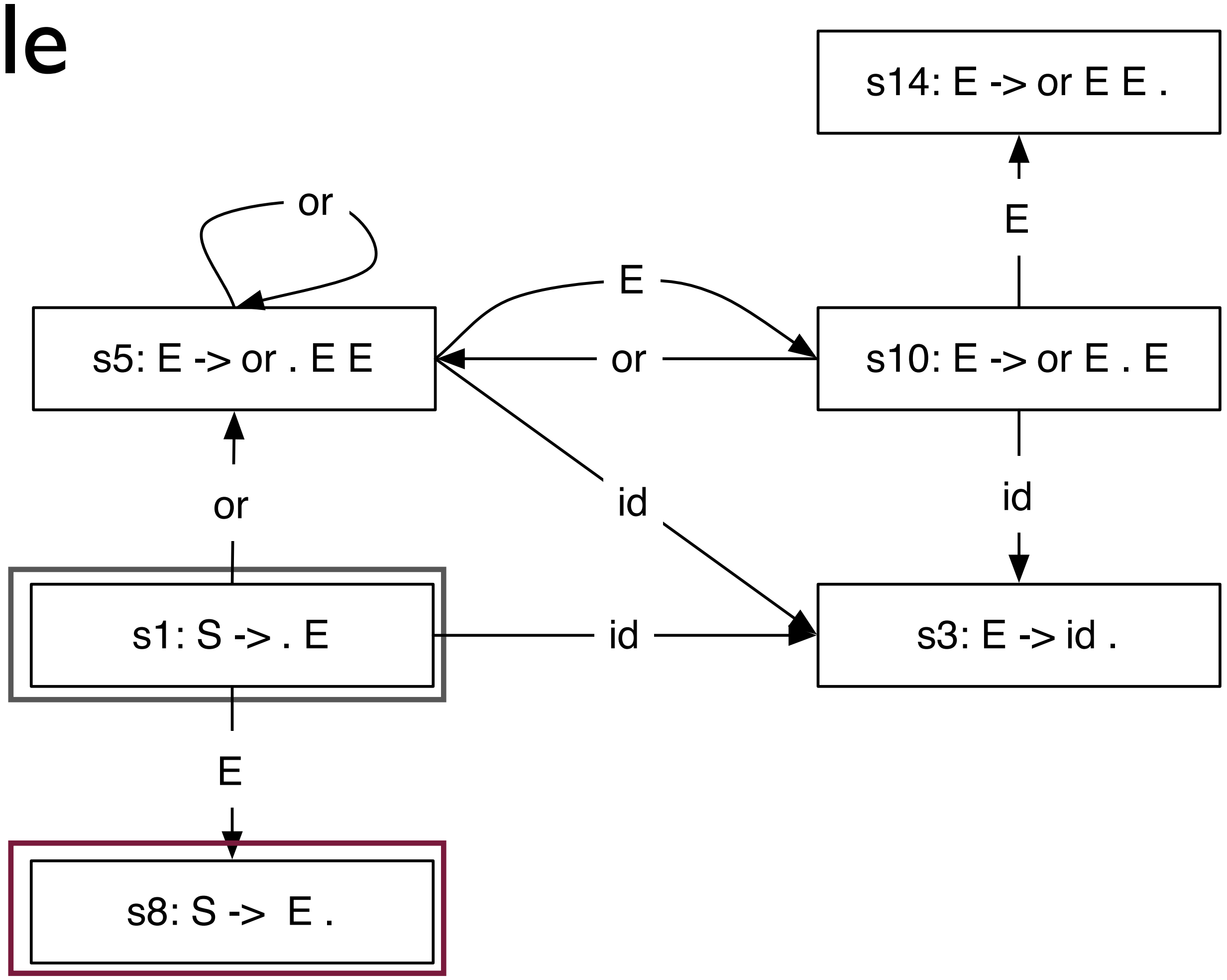


`or a b .`

s_1

LR Parsing

- Example

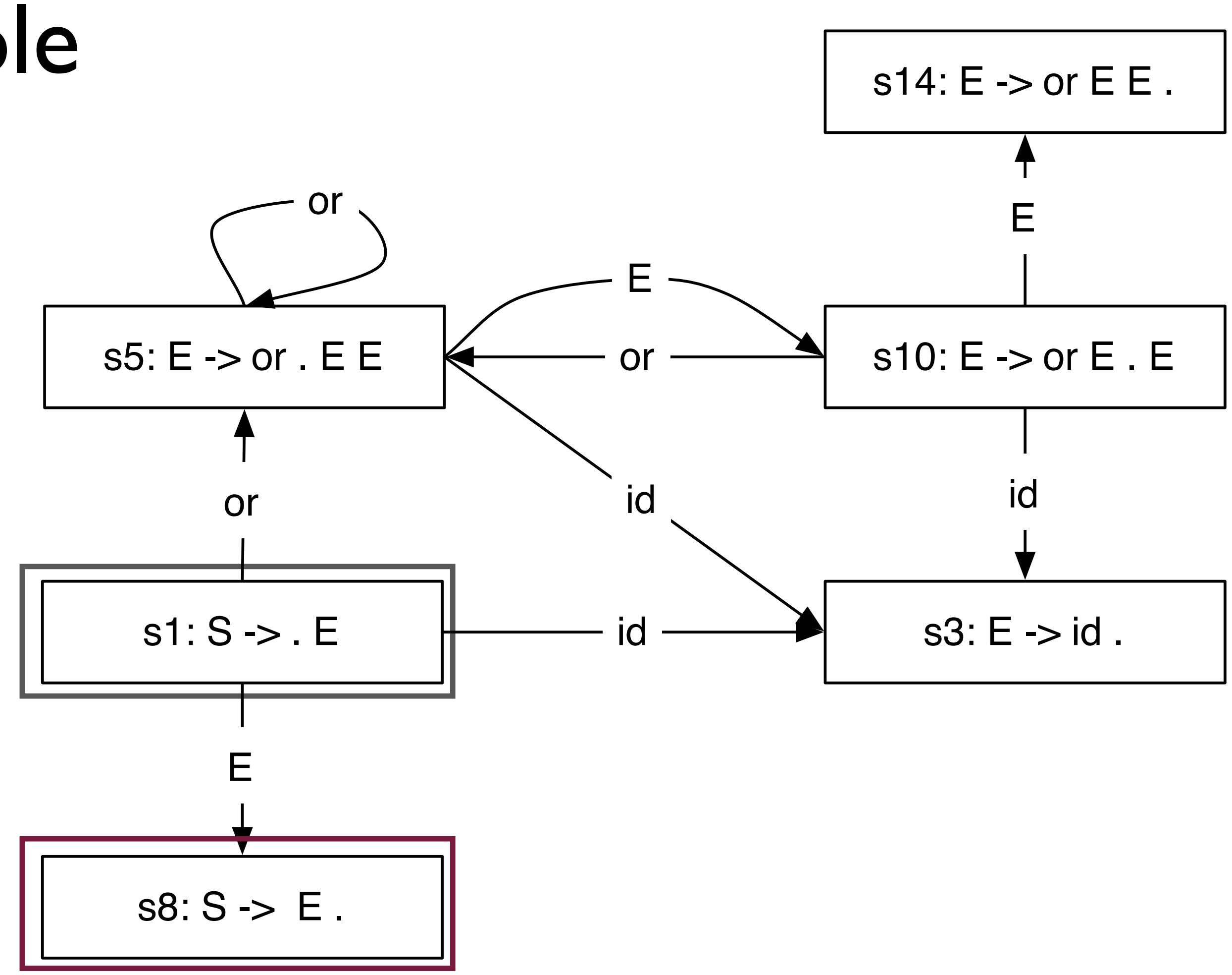


`or a b .`

$s_8 s_1$

LR Parsing

- Example



`or a b .`

$s_8 s_1$

Accept!

Abstract Parsing: Idea

- Instead of executing the program and parsing the result,

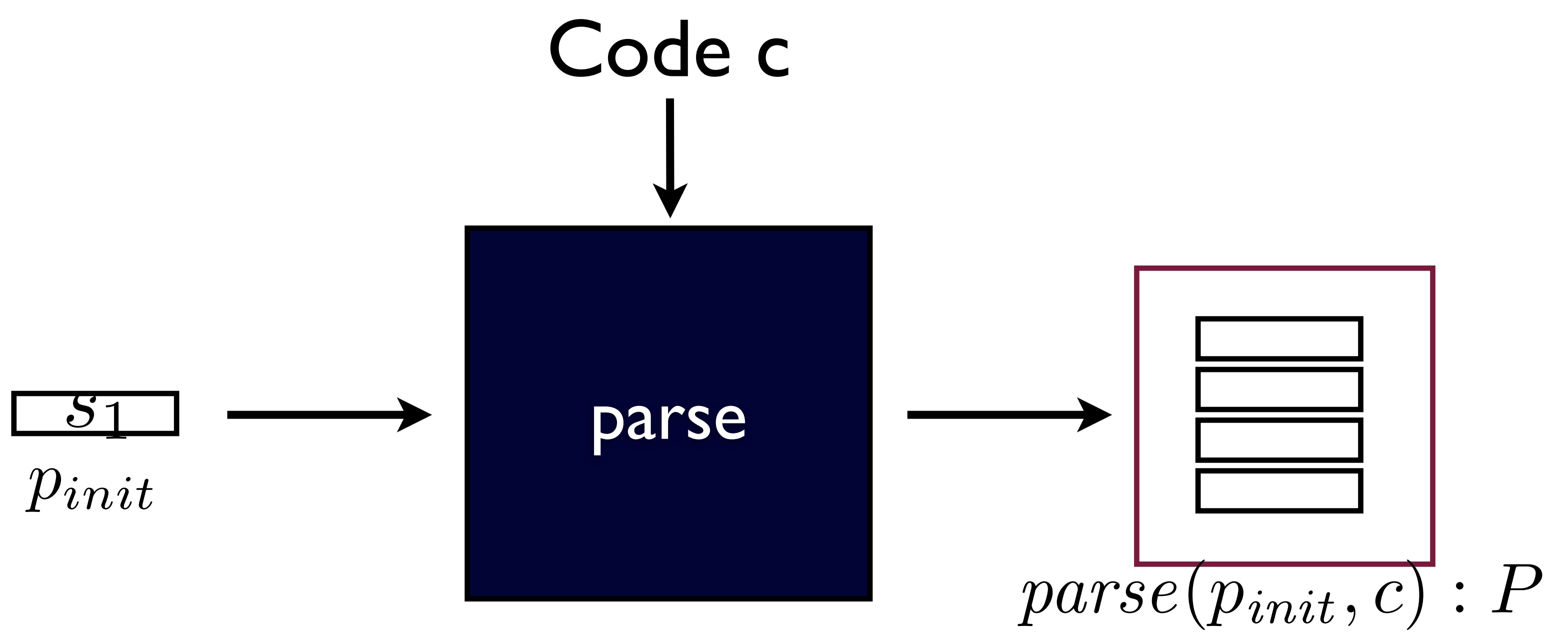
$$[\![e]\!]^0\Sigma = \{c_1, c_2, \ldots, c_n\} \quad parse(c_i) = O/X$$

- Define abstract semantics using parse stack and execute the program on it.

$$[\![\hat{e}]\!]^0\Sigma\{p_{init}\} = \{p_1, p_2, \ldots, p_n\}$$

Code as Parse Stack Transition Function

- Q: What should be the abstract value for Code c?
- A1: Parse Stack: $parse(p_{init}, c) : P$

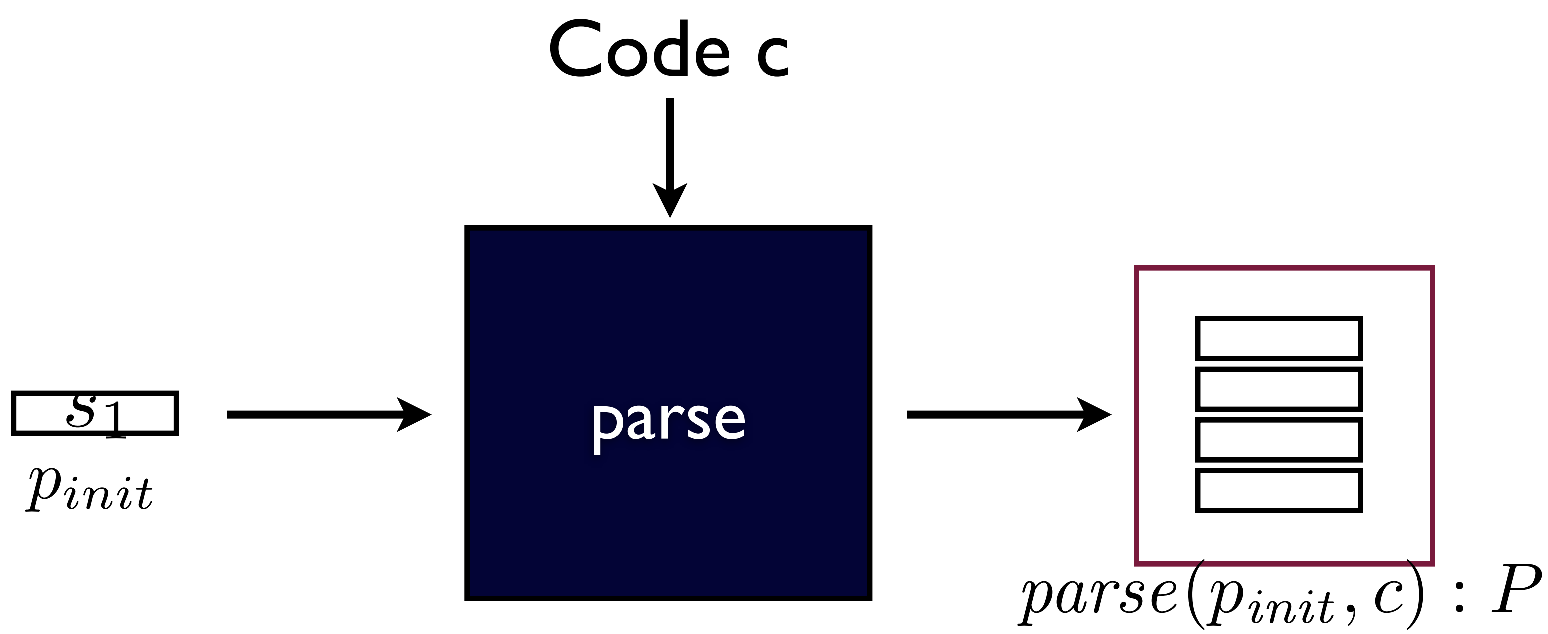


- Example

$$\texttt{or} \longrightarrow s_1 s_5$$
$$\texttt{a} \longrightarrow s_1 s_8$$

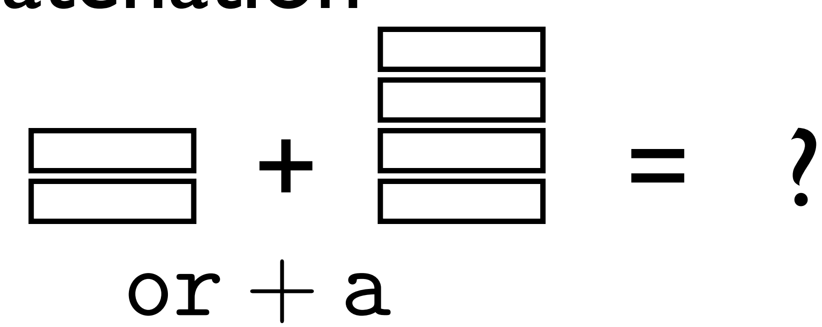
Code as Parse Stack Transition Function

- Q: What should be the abstract value for Code c?
- A1: Parse Stack: $parse(p_{init}, c) : P$



- Problem: Parse Stack Concatenation

$$\texttt{or} \rightarrow s_1 s_5$$
$$\texttt{a} \rightarrow s_1 s_8$$



Code as Parse Stack Transition Function

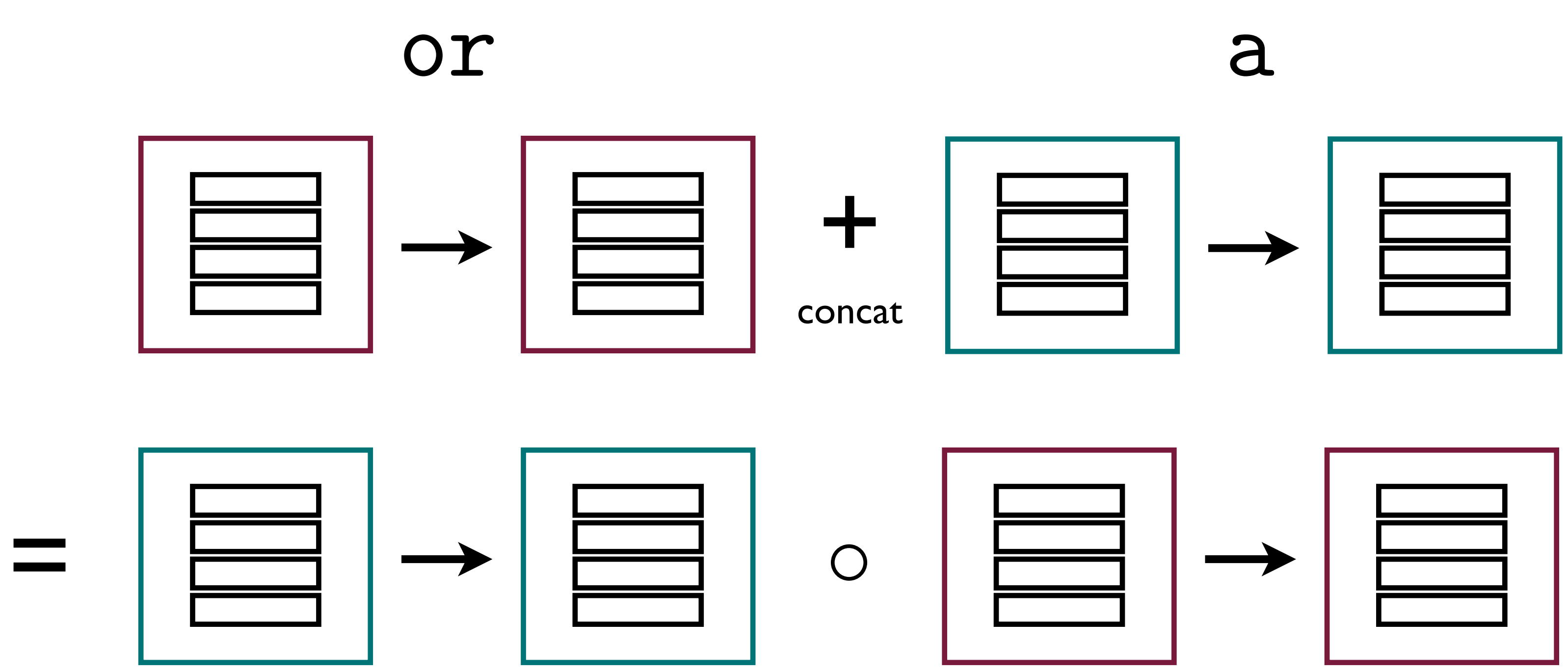
- Q: What should be the abstract value for Code c?
- A2: Parse Stack Transition Function : $\lambda p.parse(p, c) : P \to P$

`or`		`a`		`or a`
...	+	...	=>	...
$s_1 \mapsto s_1 s_5$		$s_1 s_5 \mapsto s_1 s_5 s_{10}$		$s_1 \mapsto s_1 s_5 s_{10}$
...		...		...

Code concatenation => Function Composition

Code as Parse Stack Transition Function

- Q: What should be the abstract value for Code c?
- A2: Parse Stack Transition Function : $\lambda p.parse(p, c) : P \rightarrow P$



$$f_{\texttt{a}} \circ f_{\texttt{or}}$$

Code concatenation => Function Composition

Concrete Parsing Semantics

- Using the abstraction from Code to $P \to P$ establish a Galois connection $2^{Code} \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} V_P = 2^{P \to P}$
- Derive concrete parsing semantics

$$\sigma \in Env_P = Var \to \boxed{V_P}$$
$$[\![e]\!]_P^0 \in Env_P \to \boxed{V_P}$$
$$[\![f]\!]_P^1 \in Env_P \to \boxed{V_P}$$

$$
\begin{aligned}
[\![x]\!]_P^0\sigma &= \sigma(x) \\
[\![\texttt{let}\ x\ e_1\ e_2]\!]_P^0\sigma &= [\![e_2]\!]_P^0(\sigma[x \mapsto [\![e_1]\!]_P^0\sigma]) \\
[\![\texttt{or}\ e_1\ e_2]\!]_P^0\sigma &= [\![e_1]\!]_P^0\sigma \cup [\![e_2]\!]_P^0\sigma \\
[\![\texttt{re}\ x\ e_1\ e_2\ e_3]\!]_P^0\sigma &= [\![e_3]\!]_P^0(\sigma[x \mapsto \\
&\qquad \mathit{fix}\ \lambda k.[\![e_1]\!]_P^0\sigma \cup [\![e_2]\!]_P^0(\sigma[x \mapsto k])]) \\
[\![`f]\!]_P^0\sigma &= [\![f]\!]_P^1\sigma \\
[\![t]\!]_P^1\sigma &= \boxed{\{\lambda p.parse\_action(p,t)\}} \\
[\![f_1.f_2]\!]_P^1\sigma &= \boxed{\{p_2 \circ p_1 \mid p_1 \in [\![f_1]\!]_P^1\sigma \wedge p_2 \in [\![f_2]\!]_P^1\sigma\}} \\
[\![,e]\!]_P^1\sigma &= [\![e]\!]_P^0\sigma
\end{aligned}
$$

ROSAEC center
Research On Software Analysis for Error-free Computing
소프트웨어 무결점 연구센터 KOSEF ERC

First Abstraction

- First, we abstract $2^{P \to P}$ to $2^P \to 2^P$ by,

$$\alpha_{2^{P\to P} \boxed{\to (2^P \to 2^P)}} = \boxed{\lambda F \lambda P. \{f(p) \mid f \in F \wedge p \in P\}}$$

- Semantics:

$$\begin{aligned}
\sigma \in Env_{\hat{P}} &= Var \to \boxed{V_{\hat{P}}} \\
[\![e]\!]^0_{\hat{P}} &\in Env_{\hat{P}} \to \boxed{V_{\hat{P}}} \\
[\![f]\!]^1_{\hat{P}} &\in Env_{\hat{P}} \to \boxed{V_{\hat{P}}}
\end{aligned}$$

$$\begin{aligned}
[\![x]\!]^0_{\hat{P}} &= \sigma(x) \\
[\![\texttt{let}\ x\ e_1\ e_2]\!]^0_{\hat{P}}\sigma &= [\![e_2]\!]^0_{\hat{P}}(\sigma[x \mapsto [\![e_1]\!]^0_{\hat{P}}\sigma]) \\
[\![\texttt{or}\ e_1\ e_2]\!]^0_{\hat{P}}\sigma &= \lambda P.[\![e_1]\!]^0_{\hat{P}}\sigma P \cup [\![e_2]\!]^0_{\hat{P}}\sigma P \\
[\![\texttt{re}\ x\ e_1\ e_2\ e_3]\!]^0_{\hat{P}}\sigma &= [\![e_3]\!]^0_{\hat{P}}(\sigma[x \mapsto \\
&\qquad fix\ \lambda k.\lambda P.[\![e_1]\!]^0_{\hat{P}}\sigma P \cup [\![e_2]\!]^0_{\hat{P}}(\sigma[x \mapsto k])P]) \\
[\![\text{`}f]\!]^0_{\hat{P}}\sigma &= [\![f]\!]^1_{\hat{P}}\sigma \\
[\![t]\!]^1_{\hat{P}}\sigma &= \boxed{\lambda P.Parse_action(P, t)} \\
[\![f_1 . f_2]\!]^1_{\hat{P}}\sigma &= \boxed{[\![f_2]\!]^1_{\hat{P}}\sigma \circ [\![f_1]\!]^1_{\hat{P}}\sigma} \\
[\![, e]\!]^1_{\hat{P}}\sigma &= [\![e]\!]^0_{\hat{P}}\sigma
\end{aligned}$$

Need More Abstraction

- Since P is infinite, computing f : $2^P \rightarrow 2^P$ may not terminate.
- Example:

```
re x (`or . , x) (`,x . b)
=> a b
   or a b
   or or a b
   ...
```

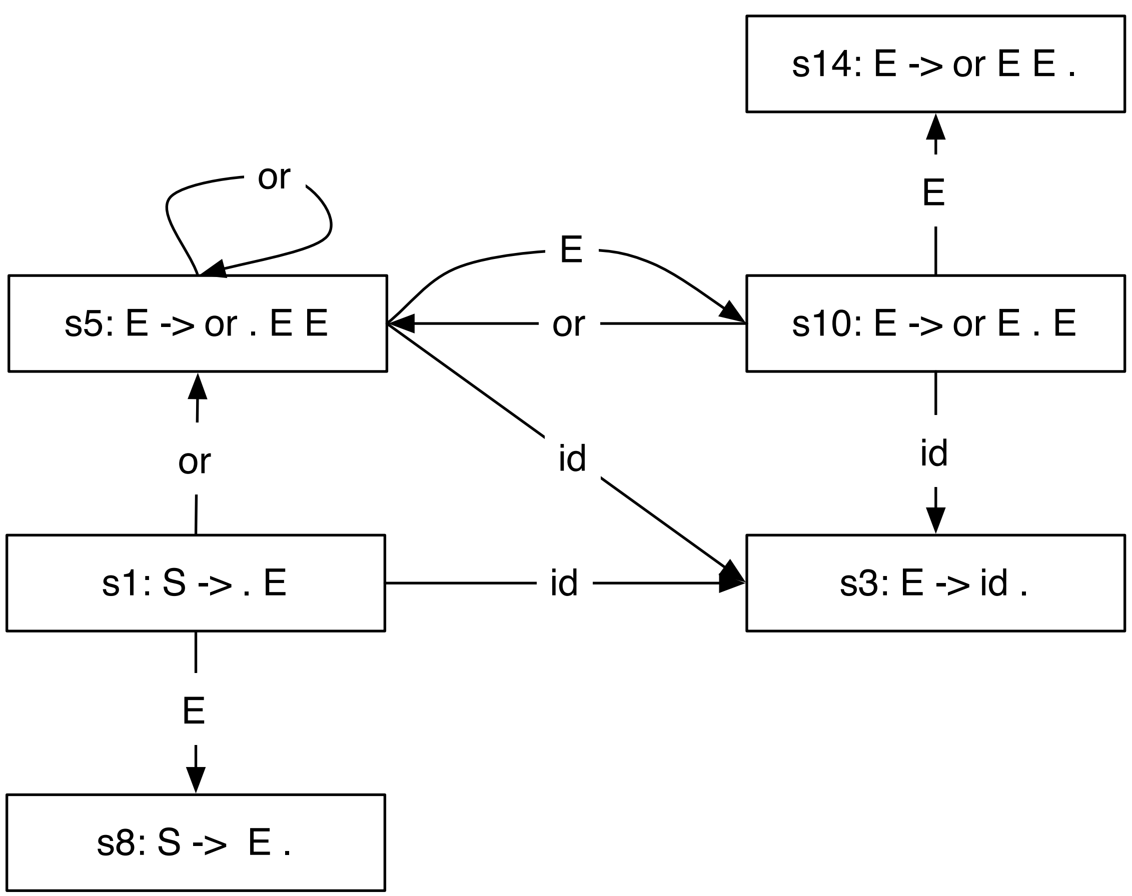
$$[\![\texttt{re}\ x\ `a\ \ (`\texttt{or}\ \ .\ \ ,x)\ \ (`,\texttt{x}\ \ .\ \ \texttt{b})]\!]^0_{\hat{P}}\sigma_0\{s_1\}$$
$$= (\lambda P.PA(P,\texttt{b}) \circ (\mathit{fix}\,\lambda k.\lambda P.(PA(P,\texttt{a}) \sqcup k \circ PA(P,\texttt{or}))))\{s_1\}$$

Need More Abstraction

- Example:

```
re x (`or . ,x) (`,x . b)
```

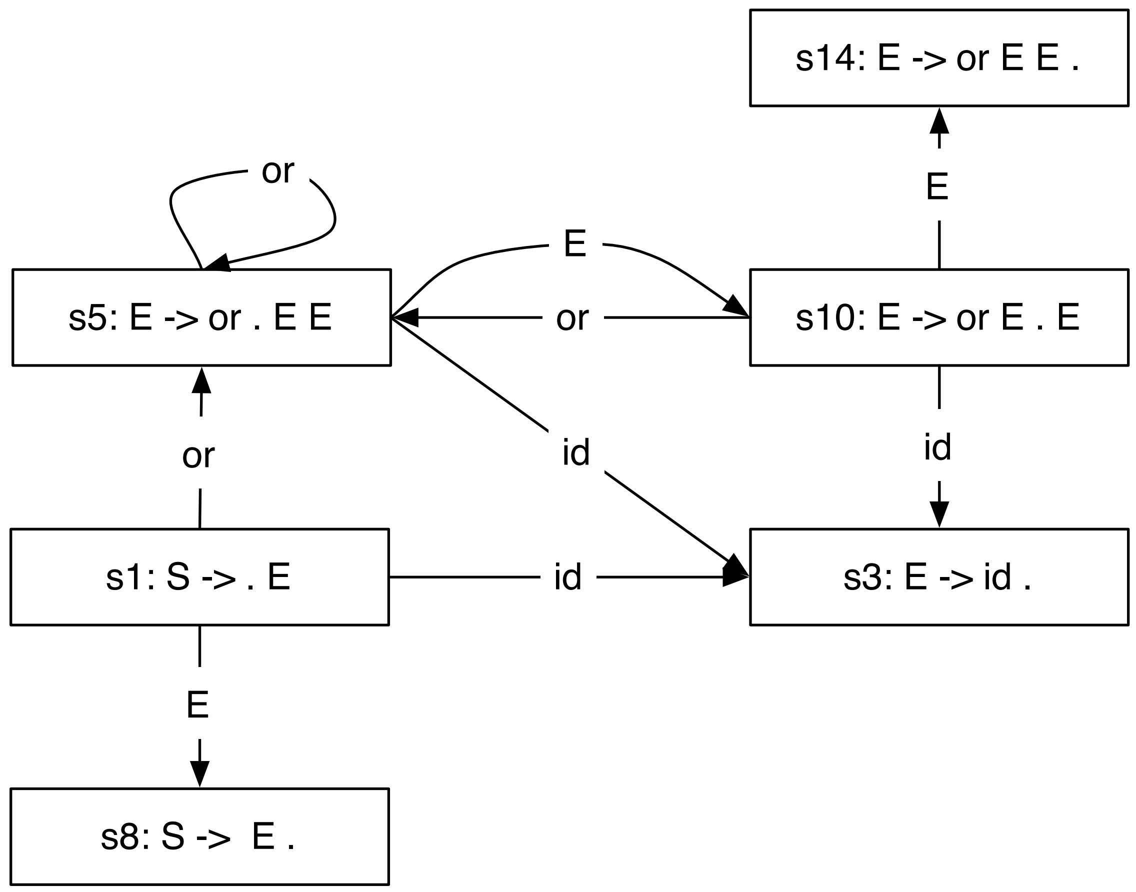
$$\llbracket \texttt{re}\ x\ \text{‘}a\ (\text{‘}\texttt{or}\ .\ ,x)\ (\text{‘},\texttt{x}\ .\ \texttt{b})\rrbracket^{0}_{\hat{P}}\sigma_0\{s_1\}$$
$$= (\lambda P.PA(P,\texttt{b}) \circ (\mathit{fix}\,\lambda k.\lambda P.(PA(P,\texttt{a}) \sqcup k \circ PA(P,\texttt{or})))) \{s_1\}$$



Need More Abstraction

- Example: `re x (`or . ,x) (`,x . b)`

$$[\![\texttt{re}\ x\ \text{`}a\ (\text{`}\texttt{or}\ .\ ,x)\ (\text{`},\texttt{x}\ .\ \texttt{b})]\!]^{0}_{\hat{P}}\sigma_0\{s_1\}$$
$$= (\lambda P.PA(P,\texttt{b}) \circ \underline{(\mathit{fix}\,\lambda k.\lambda P.(PA(P,\texttt{a}) \sqcup k \circ PA(P,\texttt{or}))))}\{s_1\}$$

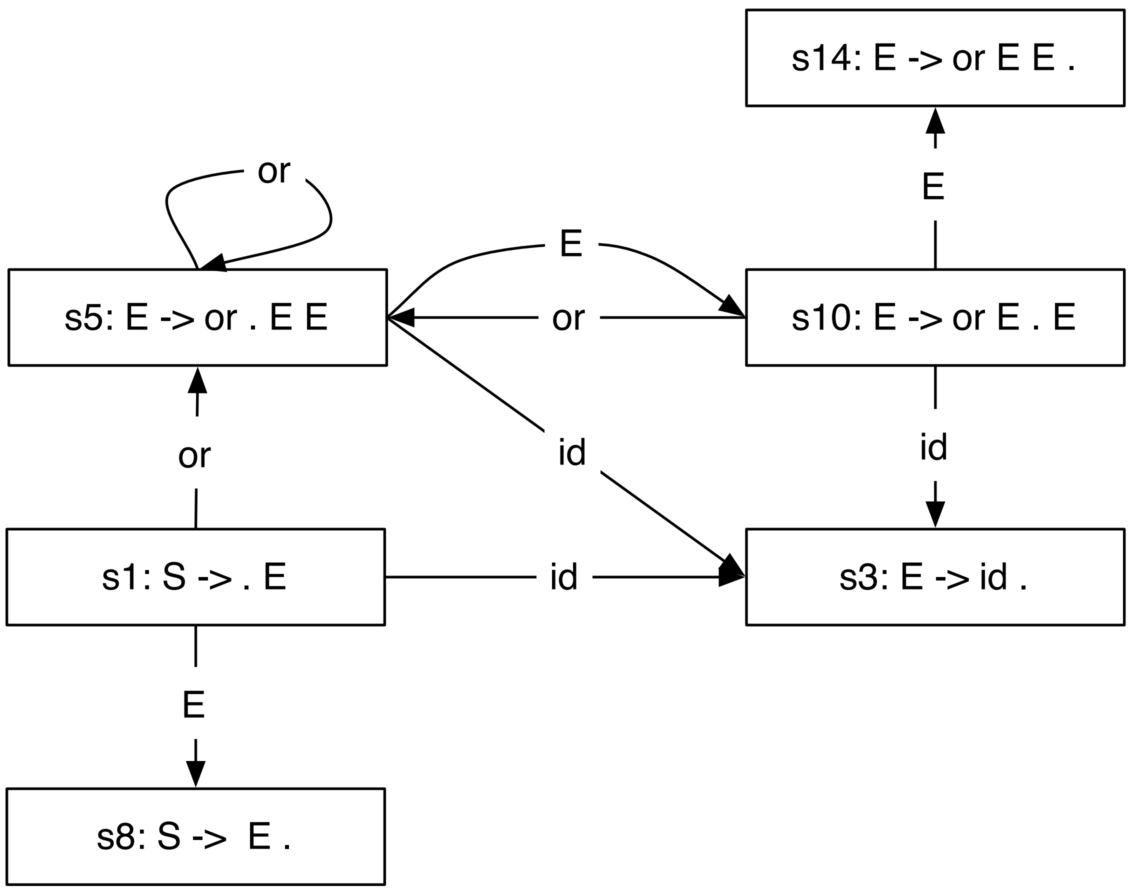


ROSAEC center
Research On Software Analysis for Error-free Computing
소프트웨어 무결점 연구센터 KOSEF ERC

Need More Abstraction

- Example: `re x (`or . ,x) (`,x . b)`

$$[\![\texttt{re}\ x\ `a\ (`\texttt{or}\ .\ ,x)\ (`,\texttt{x}\ .\ \texttt{b})]\!]^0_{\hat{P}}\sigma_0\{s_1\}$$
$$= (\lambda P.PA(P,\texttt{b}) \circ (\mathit{fix}\,\lambda k.\lambda P.(\underline{PA(P,\texttt{a})} \sqcup k \circ PA(P,\texttt{or}))))\{s_1\}$$

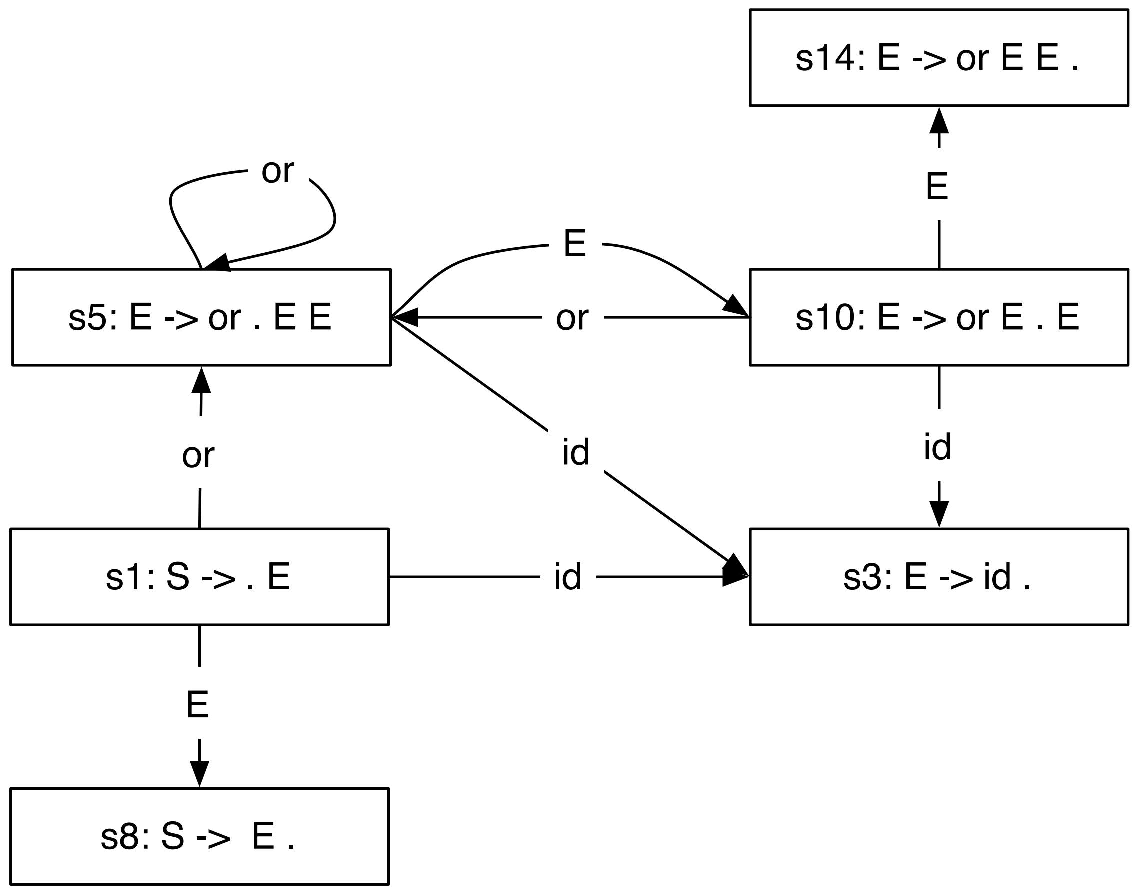


$$\{s_1\} \mapsto \{s_8 s_1\}$$

Need More Abstraction

- Example: `re x (`or . ,x) (`,x . b)`

$$
\begin{aligned}
&[\![\texttt{re}\ x\ \text{`}a\ (\text{`}\texttt{or}\ .\ ,x)\ (\text{`},\texttt{x}\ .\ \texttt{b})]\!]^0_{\hat{P}}\sigma_0\{s_1\} \\
&\qquad = (\lambda P.PA(P,\texttt{b}) \circ (\mathit{fix}\,\lambda k.\lambda P.(PA(P,\texttt{a}) \sqcup \underline{k \circ PA(P,\texttt{or})})))\{s_1\}
\end{aligned}
$$



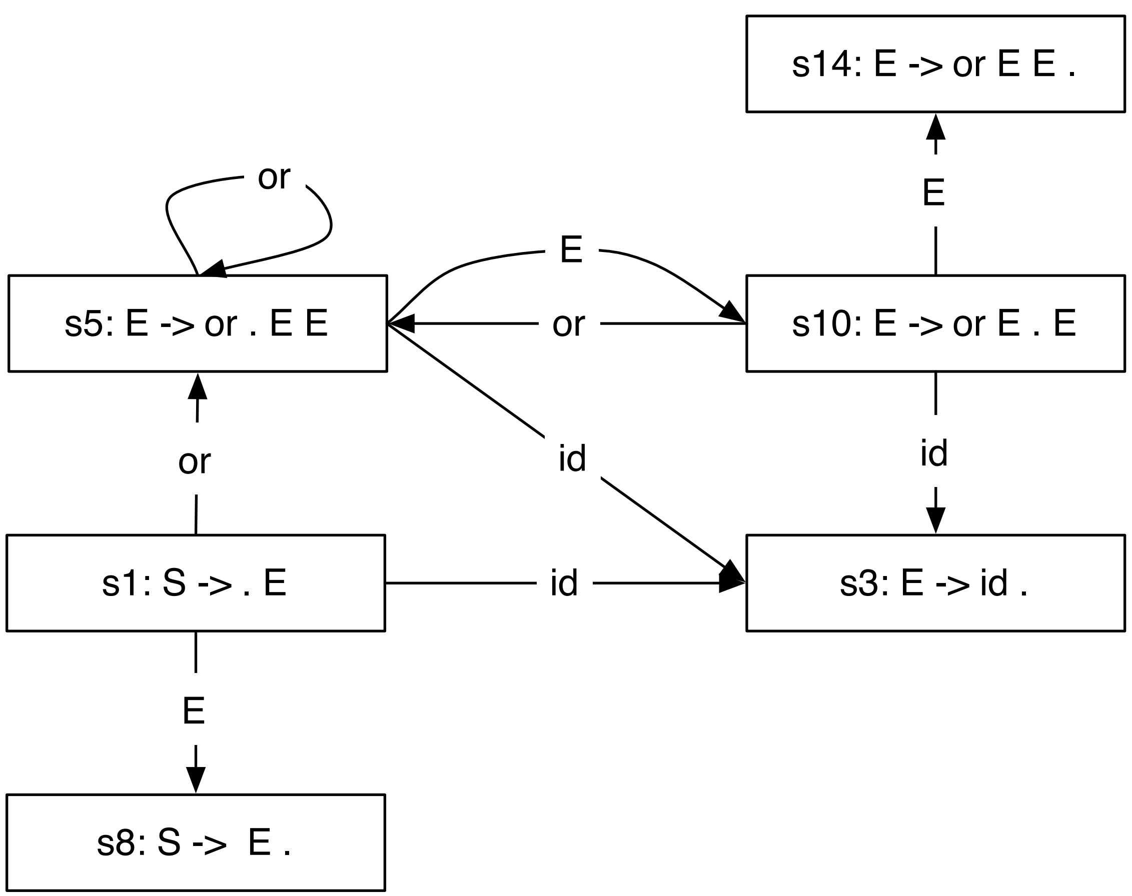
$$\{s_1\} \mapsto \{s_8 s_1\} \cup k(\{s_5 s_1\})$$

ROSAEC center
Research On Software Analysis for Error-free Computing
소프트웨어 무결점 연구센터 KOSEF ERC

Need More Abstraction

- Example: `re x (`or . ,x) (`,x . b)`

$$
\begin{aligned}
&[\![\texttt{re}\ x\ \text{`}a\ (\text{`}\texttt{or}\ .\ ,x)\ (\text{`},\texttt{x}\ .\ \texttt{b})]\!]^{0}_{\hat{P}}\sigma_0\{s_1\} \\
&\qquad = (\lambda P.PA(P,\texttt{b}) \circ (\mathit{fix}\,\lambda k.\lambda P.(\underline{PA(P,\texttt{a})} \sqcup k \circ PA(P,\texttt{or}))))\{s_1\}
\end{aligned}
$$



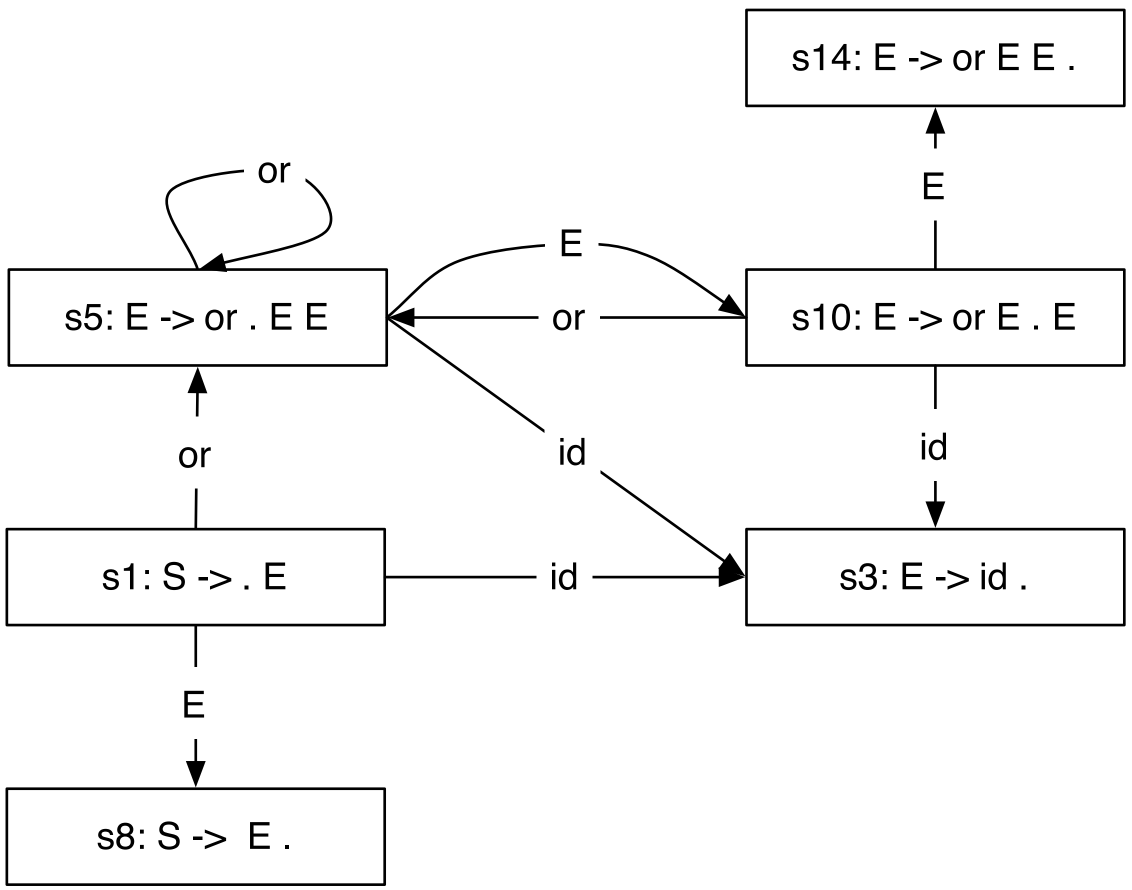
$$\{s_1\} \mapsto \{s_8 s_1\}$$

$$\{s_5 s_1\} \mapsto \{s_{10} s_5 s_1\}$$

Need More Abstraction

- Example: `re x (`or . ,x) (`,x . b)`

$$[\![\texttt{re}\ x\ `a\ (`\texttt{or}\ .\ ,x)\ (`,\texttt{x}\ .\ \texttt{b})]\!]^0_{\hat{P}}\sigma_0\{s_1\}$$
$$= (\lambda P.PA(P,\texttt{b}) \circ (\mathit{fix}\,\lambda k.\lambda P.(PA(P,\texttt{a}) \sqcup \underline{k \circ PA(P,\texttt{or})})))\{s_1\}$$



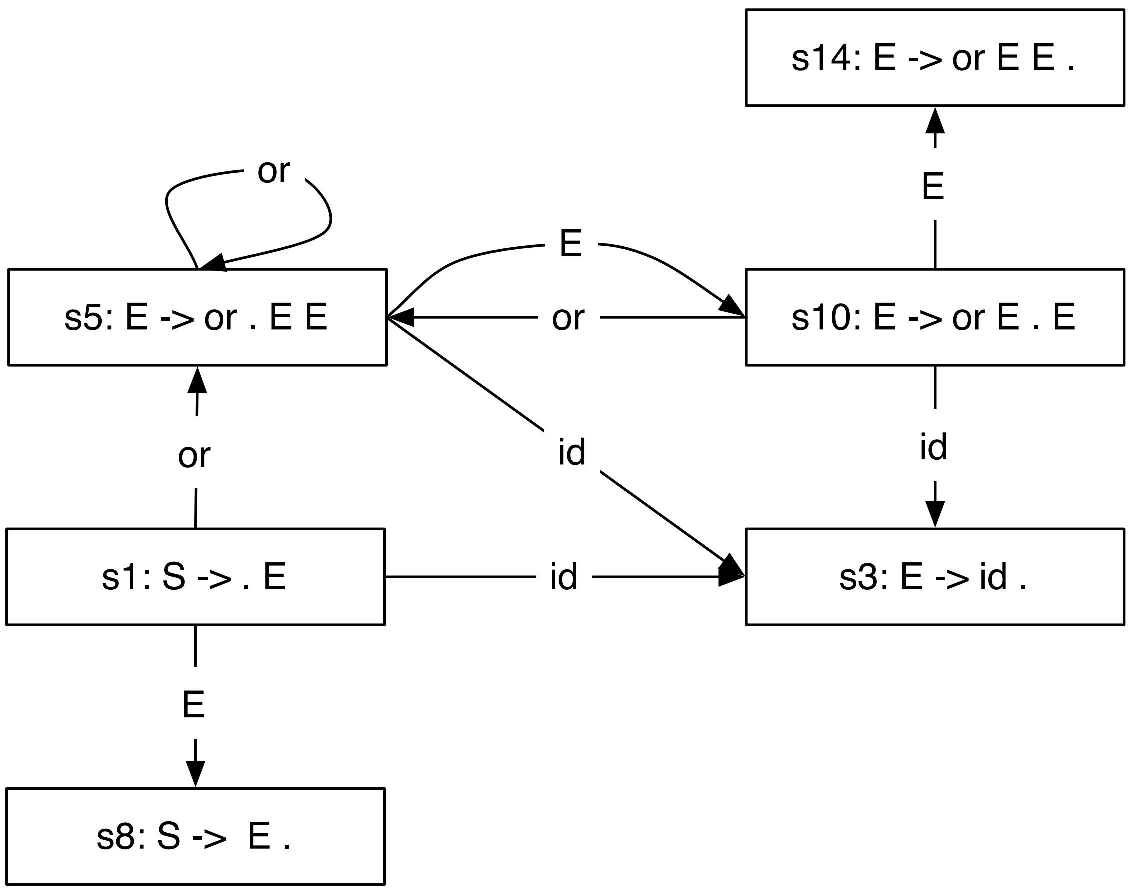
$$\{s_1\} \mapsto \{s_8 s_1\}$$
$$\{s_5 s_1\} \mapsto \{s_{10} s_5 s_1\} \cup k(\{s_5 s_5 s_1\})$$

ROSAEC center
Research On Software Analysis for Error-free Computing
소프트웨어 무결점 연구센터 KOSEF ERC

Need More Abstraction

- Example: `re x (`or . ,x) (`,x . b)`

$$[\![\texttt{re}\ x\ \text{`}a\ (\text{`}\texttt{or}\ .\ ,x)\ (\text{`},\texttt{x}\ .\ \texttt{b})]\!]^0_{\hat{P}}\sigma_0\{s_1\}$$
$$= (\lambda P.PA(P,\texttt{b}) \circ (\mathit{fix}\,\lambda k.\lambda P.(\underline{PA(P,\texttt{a})} \sqcup k \circ PA(P,\texttt{or}))))\{s_1\}$$

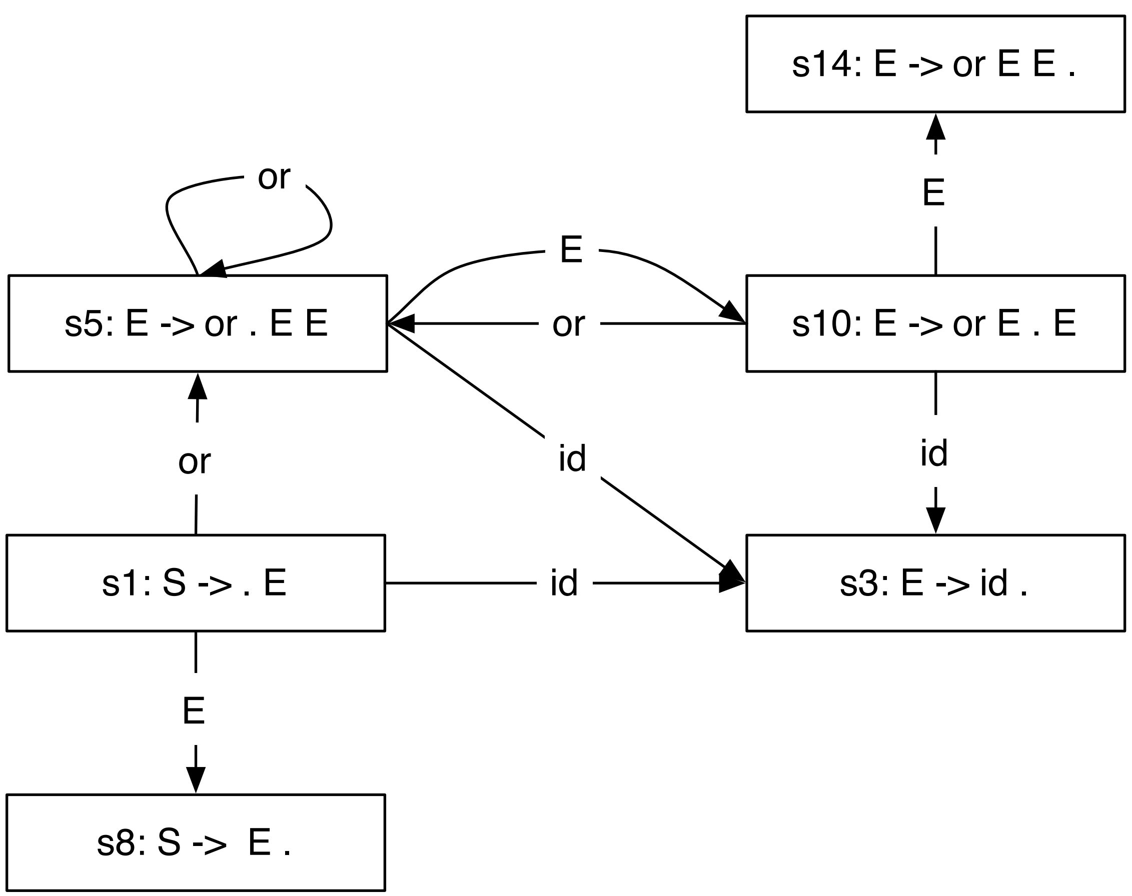


$$\{s_1\} \mapsto \{s_8 s_1\}$$
$$\{s_5 s_1\} \mapsto \{s_{10} s_5 s_1\}$$
$$\{s_5 s_5 s_1\} \mapsto \{s_{10} s_5 s_5 s_1\}$$

Need More Abstraction

- Example: `re x (`or . ,x) (`,x . b)`

$$
\begin{aligned}
&[\![\texttt{re}\ x\ \text{`}a\ (\text{`}\texttt{or}\ .\ ,x)\ (\text{`},\texttt{x}\ .\ \texttt{b})]\!]^{0}_{\hat{P}}\sigma_0\{s_1\} \\
&\quad = (\lambda P.PA(P,\texttt{b}) \circ (\mathit{fix}\,\lambda k.\lambda P.(PA(P,\texttt{a}) \sqcup \underline{k \circ PA(P,\texttt{or})})))\{s_1\}
\end{aligned}
$$



$$\{s_1\} \mapsto \{s_8 s_1\}$$

$$\{s_5 s_1\} \mapsto \{s_{10} s_5 s_1\}$$

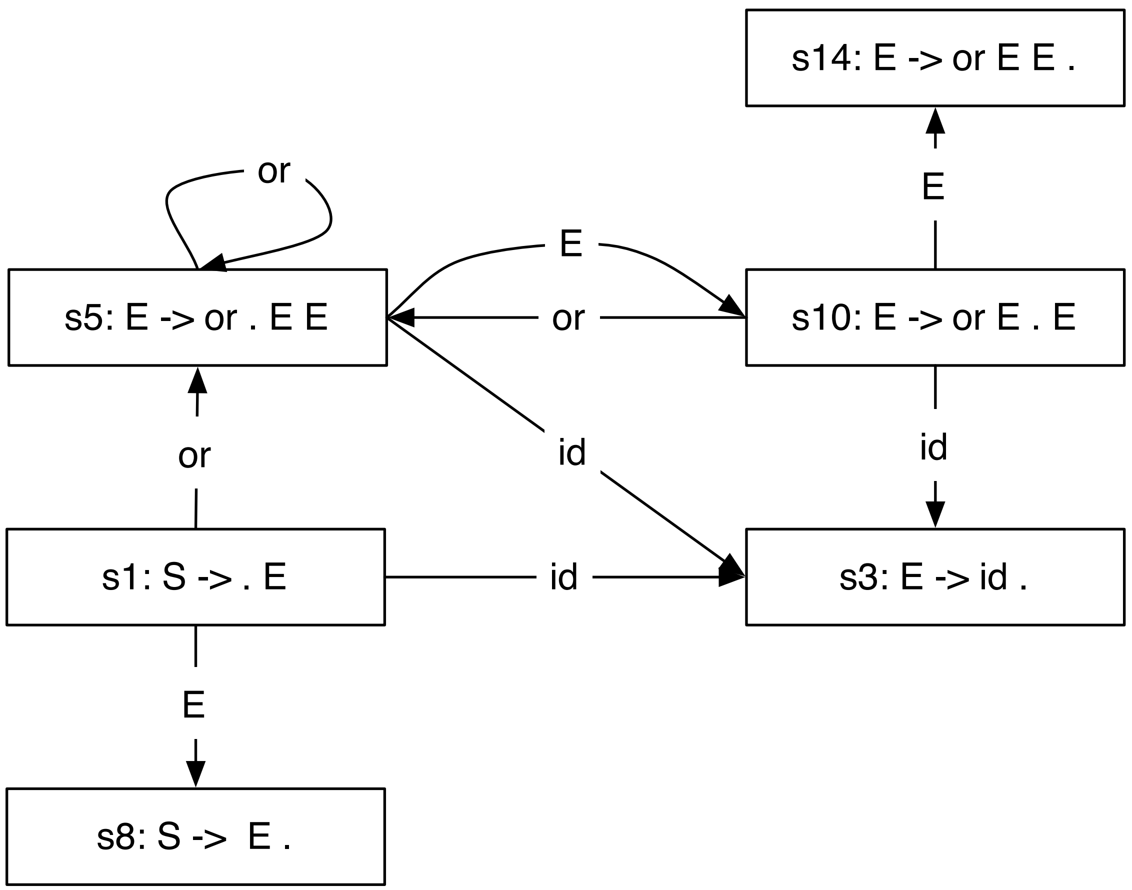
$$\{s_5 s_5 s_1\} \mapsto \{s_{10} s_5 s_5 s_1\} \cup k(\{s_5 s_5 s_5 s_1\})$$

Need More Abstraction

- Example:

```
re x (`or . ,x) (`,x . b)
```

$$[\![\texttt{re}\ x\ \text{`}a\ (\text{`}\texttt{or}\ .\ ,x)\ (\text{`},\texttt{x}\ .\ \texttt{b})]\!]^0_{\hat{P}}\sigma_0\{s_1\}$$
$$= (\lambda P.PA(P,\texttt{b}) \circ (\mathit{fix}\,\lambda k.\lambda P.(\underline{PA(P,\texttt{a})} \sqcup k \circ PA(P,\texttt{or}))))\{s_1\}$$

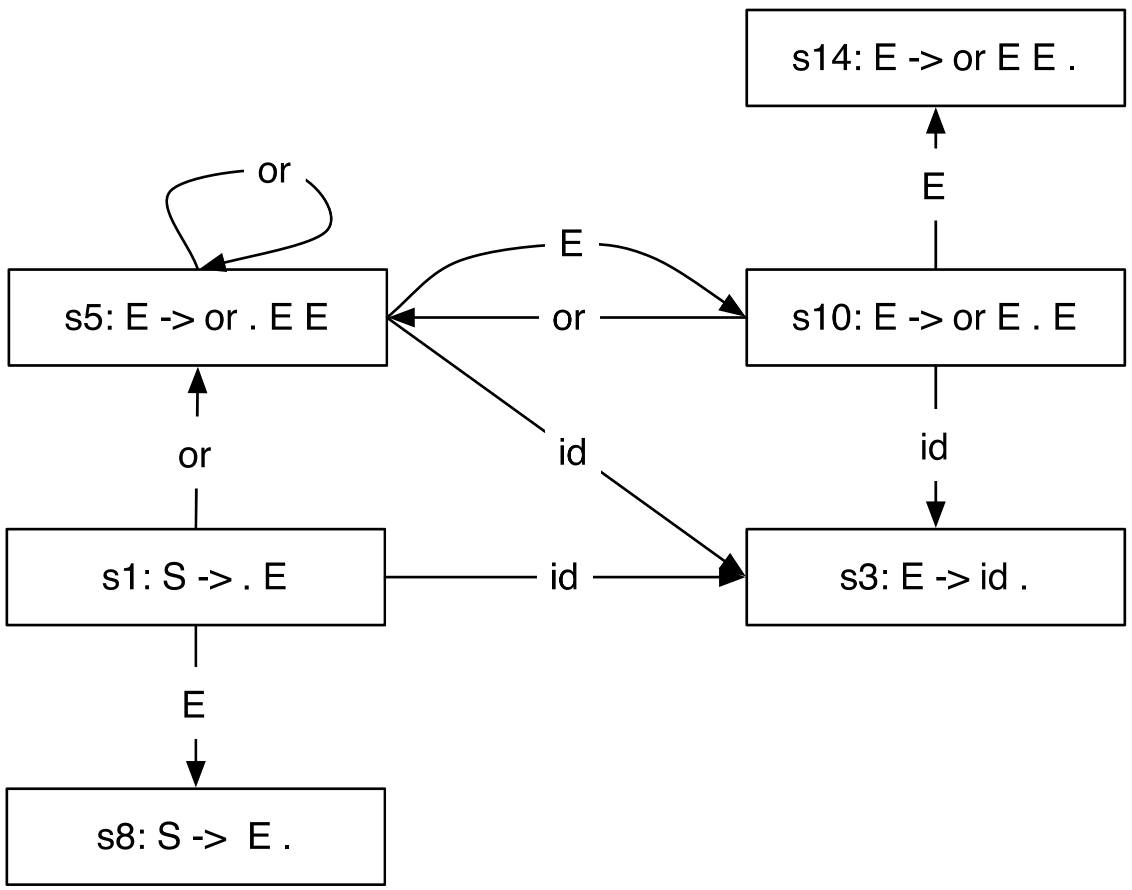


$$\{s_1\} \mapsto \{s_8 s_1\}$$
$$\{s_5 s_1\} \mapsto \{s_{10} s_5 s_1\}$$
$$\{s_5 s_5 s_1\} \mapsto \{s_{10} s_5 s_5 s_1\}$$
$$\{s_5 s_5 s_5 s_1\} \mapsto \ldots$$

Need More Abstraction

- Example: `re x (`or . ,x) (`,x . b)`

$$[\![\texttt{re}\ x\ \text{`}a\ (\text{`}\texttt{or}\ .\ ,x)\ (\text{`},\texttt{x}\ .\ \texttt{b})]\!]^0_{\hat{P}}\sigma_0\{s_1\}$$
$$= (\lambda P.PA(P,\texttt{b}) \circ (\mathit{fix}\lambda k.\lambda P.(PA(P,\texttt{a}) \sqcup k \circ PA(P,\texttt{or}))))\{s_1\}$$



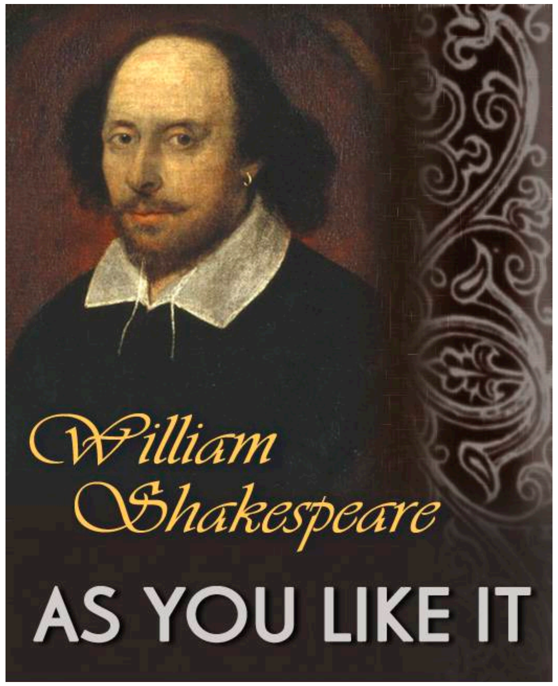
$$\{s_1\} \mapsto \{s_8 s_1\}$$
$$\{s_5 s_1\} \mapsto \{s_{10} s_5 s_1\}$$
$$\{s_5 s_5 s_1\} \mapsto \{s_{10} s_5 s_5 s_1\}$$
$$\{s_5 s_5 s_5 s_1\} \mapsto \ldots$$

...

Not Terminated.

Parameterized Framework

Instead of providing *particular abstract domain for* 2^P, *Parameterize* an abstract domain with the *conditions* it should satisfy.



Parameterized Framework

We can abstract $2^P \rightarrow 2^P$ to $D^\sharp \rightarrow D^\sharp$

if $D^\sharp$ satisfies the following conditions.

1. $(D^\sharp, \sqsubseteq, \sqcup, \bot_{D^\sharp})$ is CPO
2. 2^P and $D^\sharp$ are Galois connected via $\alpha_{2^P \rightarrow D^\sharp}$ and $\gamma_{D^\sharp \rightarrow 2^P}$
3. $parse\_action^\sharp : Token \rightarrow D^\sharp \rightarrow D^\sharp$ is a sound abstraction of

   $parse\_action : Token \rightarrow 2^P \rightarrow 2^P$. That is,

   $\forall a \in Token. \forall P \in 2^P.$

   $\alpha_{2^P \rightarrow D^\sharp}(\{parse\_action\ a\ p \mid p \in P\}) \sqsubseteq parse\_action^\sharp\ a\ \alpha_{2^P \rightarrow D^\sharp}(P)$

Parameterized Framework

We define abstract semantics function $[\![\cdot]\!]_{D^\sharp}$

$$\begin{aligned}
\sigma \in \mathit{Env}_{D^\sharp} &= \mathit{Var} \to V^\sharp \\
[\![e]\!]^0_{D^\sharp} &\in \mathit{Env}_{D^\sharp} \to V^\sharp \\
[\![f]\!]^1_{D^\sharp} &\in \mathit{Env}_{D^\sharp} \to V^\sharp
\end{aligned}$$

$$\begin{aligned}
[\![x]\!]^0_{D^\sharp}\sigma &= \sigma(x) \\
[\![\texttt{let}\ x\ e_1\ e_2]\!]^0_{D^\sharp}\sigma &= [\![e_2]\!]^0_{D^\sharp}(\sigma[x \mapsto [\![e_1]\!]^0_{D^\sharp}\sigma]) \\
[\![\texttt{or}\ e_1\ e_2]\!]^0_{D^\sharp}\sigma &= [\![e_1]\!]^0_{D^\sharp}\sigma \sqcup [\![e_2]\!]^0_{D^\sharp}\sigma \\
[\![\texttt{re}\ x\ e_1\ e_2\ e_3]\!]^0_{D^\sharp}\sigma &= [\![e_3]\!]^0_{D^\sharp}(\sigma[x \mapsto \\
&\qquad \mathit{fix}\ \lambda k.[\![e_1]\!]^0_{D^\sharp}\sigma \sqcup [\![e_2]\!]^0_{D^\sharp}(\sigma[x \mapsto k])]) \\
[\![\text{`}f]\!]^0_{D^\sharp}\sigma &= [\![f]\!]^1_{D^\sharp}\sigma \\
[\![t]\!]^1_{D^\sharp}\sigma &= \lambda D.\mathit{Parse_action}^\sharp(D,t) \\
[\![f_1.f_2]\!]^1_{D^\sharp}\sigma &= [\![f_2]\!]^1_{D^\sharp}\sigma \circ [\![f_1]\!]^1_{D^\sharp}\sigma \\
[\![,e]\!]^1_{D^\sharp}\sigma &= [\![e]\!]^0_{D^\sharp}\sigma
\end{aligned}$$

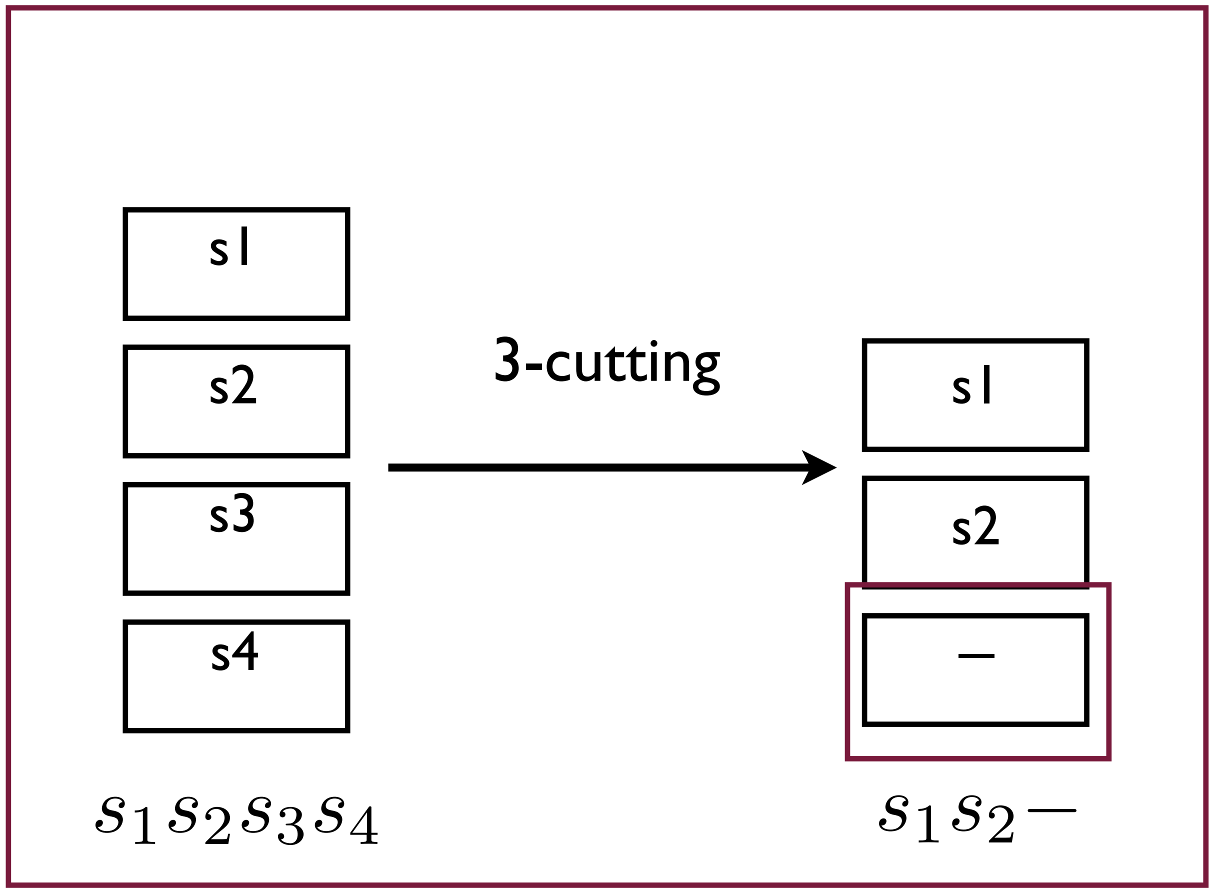
Then $[\![\cdot]\!]_{D^\sharp}$ is a sound approximation of $[\![\cdot]\!]_{\hat{P}}$.

$$\begin{aligned}
&\forall e \in \mathit{Exp}.\forall \sigma \in \mathit{Env}_{\hat{P}}. \\
&\alpha_{V_{\hat{P}} \to V^\sharp}([\![e]\!]_{\hat{P}}\sigma) \sqsubseteq [\![e]\!]_{D^\sharp}(\alpha_{\mathit{Env}_{\hat{P}} \to \mathit{Env}_{D^\sharp}}(\sigma))
\end{aligned}$$

Instantiation of $D^\sharp$

$\widehat{D}$: Abstract Parsing Stack with k-cutting

IDEA : limit the length of parsing stack with k



Instantiation of $D^\sharp$

$\widehat{D}$: Abstract Parsing Stack with k-cutting

1. Define Abstract Parse Stack

$$\bar{P} = \{p \cdot - \mid p \in \Sigma^*\}$$

$$\hat{P} = P \cup \bar{P}$$

$$\rho_1 \sqsubseteq_{\hat{P}} \rho_2 \overset{\text{def}}{=} \mathit{prefix}(\rho_1) \text{ starts with } \mathit{prefix}(\rho_2)$$

$$\mathit{prefix}(\rho) = \begin{cases} \rho & \text{if } \rho \in P \\ s_1 \ldots s_n & \text{if } \rho = s_1 \ldots s_n - \\ \epsilon \text{ (empty string)} & \rho = - \end{cases}$$

Example

$$s_1 s_2 s_3 - = \begin{array}{|c|}\hline \text{s1} \\ \hline \text{s2} \\ \hline \text{s3} \\ \hline - \\ \hline \end{array}$$

$$s_4 s_5 s_6 \sqsubseteq s_4 -$$

Instantiation of $D^\sharp$

$\widehat{D}$: Abstract Parsing Stack with k-cutting

2. Define Abstract Domain $\hat{D}$

$$\hat{D} = \{norm(\hat{d}) \mid \hat{d} \in 2^{\hat{P}}\}$$

$$\hat{d}_1 \sqsubseteq \hat{d}_2 \overset{\text{def}}{=} \forall \rho_1 \in \hat{d}_1 . \exists \rho_2 \in \hat{d}_2 . \rho_1 \sqsubseteq_{\hat{P}} \rho_2$$

$$\hat{d}_1 \sqcup \hat{d}_2 \overset{\text{def}}{=} norm(\hat{d}_1 \cup \hat{d}_2)$$

$$norm(\hat{d}) = \{\rho \in \hat{d} \mid \forall \rho' \in \hat{d} . \rho \not\sqsubset_{\hat{P}} \rho'\}$$

Example

$$norm\{s_1 -, s_1 s_2, s_1 s_3 s_4\} = \{s_1\}$$

Instantiation of $D^\sharp$
$\widehat{D}$: Abstract Parsing Stack with k-cutting

3. Galois Connection $2^P \underset{\alpha}{\overset{\gamma}{\leftrightarrows}} \hat{D}$

$$\alpha = id$$
$$\gamma = \lambda \hat{d}.Expand(\hat{d})$$

$$expand(\rho) = \begin{cases} \{\rho\} & \text{if } \rho \in P \\ \{prefix(\rho) \cdot p \mid p \in P\} & \text{if } \rho \in \bar{P} \end{cases} \qquad Expand(\hat{d}) = \bigcup_{\rho \in \hat{d}} expand(\rho)$$

Example

$$\gamma\{s_1 -\} = \{s_1 s_2, s_1 s_3, \ldots, s_1 s_2 s_3 \ldots\}$$

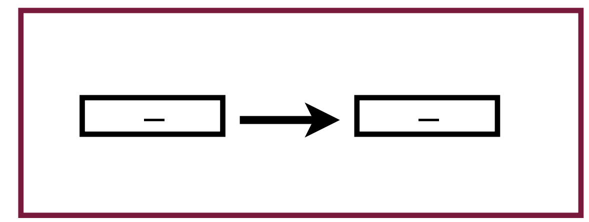
Instantiation of $D^\sharp$

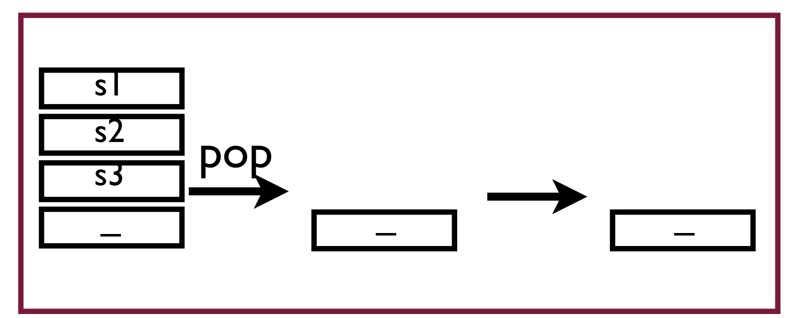
$\widehat{D}$: Abstract Parsing Stack with k-cutting

4. $\widehat{Parse\_action}$

Algorithm 2 $\widehat{parse\_action}$ algorithm

procedure $\widehat{parse\_action}(\rho, t)$
 if $\rho = -$ **then**
  return ρ
 end if
 $s_{top} \leftarrow$ the state on top of stack ρ
 if $ACTION[s_{top}, t]$ = shift s **then**
  push s onto the stack ρ
  return ρ
 else if $ACTION[s_{top}, t]$ = reduce $A \rightarrow \beta$ **then**
  pop $|\beta|$ symbol off the stack ρ
  $s_{top} \leftarrow$ the state on top of stack ρ
  if $s_{top} = -$ **then**
   return ρ
  end if
  push $GOTO[s_{top}, A]$ onto the stack ρ
  return $\widehat{parse\_action}(\rho, t)$
 end if
end procedure





Instantiation of $D^\sharp$

$\widehat{D}$: Abstract Parsing Stack with k-cutting

5. Widening

A. Define widening on $\widehat{D}$

$$A \nabla_{\hat{D}} B = \{norm(cut_k(\rho)) \mid \rho \in A \cup B\}$$

$$cut_k(\rho) = \begin{cases} \rho & \text{if } |\rho| \leq k \\ s_1 \ldots s_{k-1} - & \text{if } \rho = s_1 \ldots s_{k-1} s_k \ldots s_n. \end{cases}$$

Example

$$\{s_1 s_2 s_3\} \nabla_{\hat{D}} \{s_4 s_1 s_2 s_3\} = \{s_1 s_2 s_3, s_4 s_1 -\}$$

B. Define widening on $\hat{V} = \hat{D} \to \hat{D}$

$$f \nabla_{\hat{V}} g = \lambda \hat{d}. \begin{cases} f(\hat{d}) \nabla_{\hat{D}} g(\hat{d}) & \text{if } \forall \rho \in \hat{d}. |\rho| \leq l \\ \{-\} & \text{otherwise.} \end{cases}$$

Instantiation of $D^\sharp$

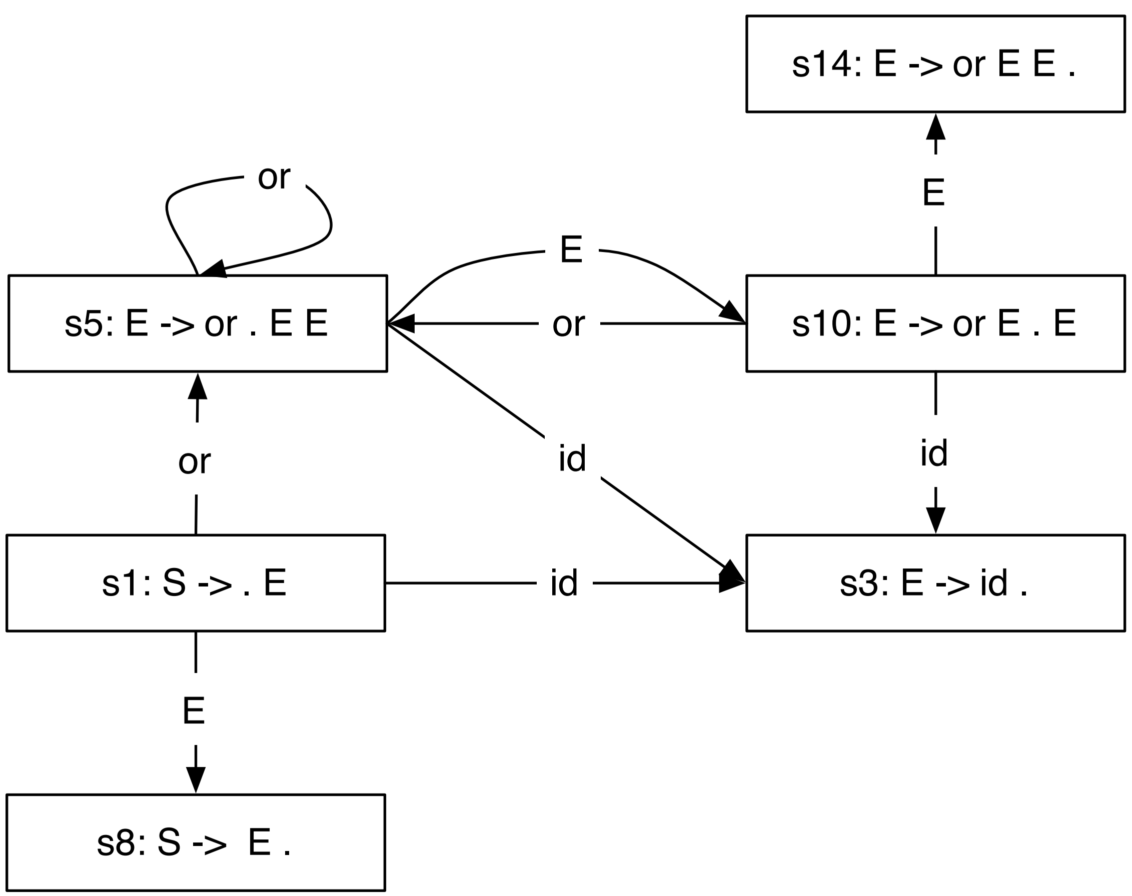
$\widehat{D}$: Abstract Parsing Stack with k-cutting

- Example: `re x (`or . ,x) (`,x . b)` k = 3

$$[\![\texttt{re}\ x\ \text{‘}a\ (\text{‘}\texttt{or}\ .\ ,x)\ (\text{‘},\texttt{x}\ .\ \texttt{b})]\!]^0_{\widehat{D}}\sigma_0\{s_1\}$$

$$= (\lambda P.\widehat{PA}(P,\texttt{b}) \circ (\mathit{fix}\,\lambda k.\lambda P.(\widehat{PA}(P,\texttt{a}) \sqcup k \circ \widehat{PA}(P,\texttt{or}))))\{s_1\}$$

1st iteration



$$\{s_1\} \mapsto \{s_8 s_1\}$$

$$\{s_5 s_1\} \mapsto \{s_{10} s_5 s_1\}$$

$$\{s_5 s_5 s_1\} \mapsto \{s_5 s_5 -\}$$

Instantiation of $D^\sharp$

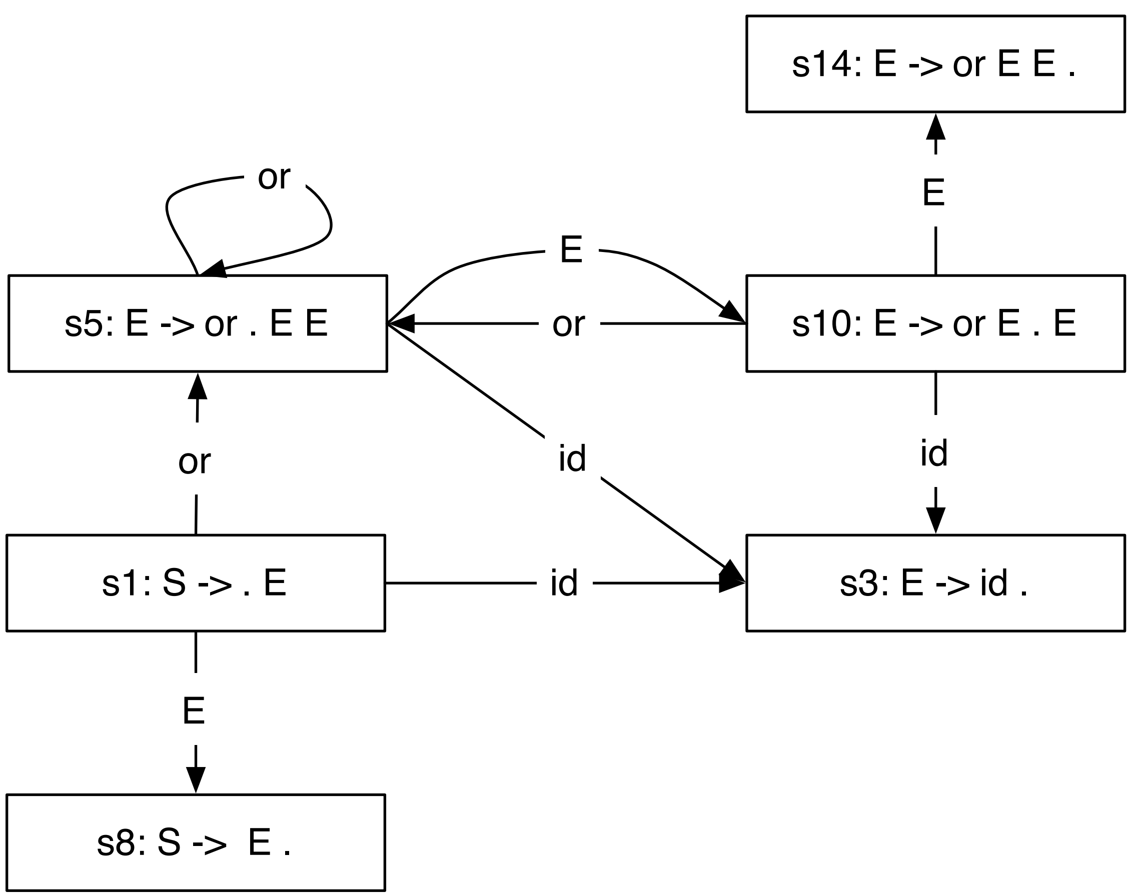
$\widehat{D}$: Abstract Parsing Stack with k-cutting

- Example: `re x (`or . ,x) (`,x . b)` k = 3

$$[\![\texttt{re}\ x\ \texttt{`}a\ (\texttt{`or}\ .\ ,x)\ (\texttt{`,x}\ .\ \texttt{b})]\!]^0_{\widehat{D}}\sigma_0\{s_1\}$$

$$= (\lambda P.\widehat{PA}(P,\texttt{b})\circ(\mathit{fix}\,\lambda k.\lambda P.(\widehat{PA}(P,\texttt{a})\sqcup k\circ\widehat{PA}(P,\texttt{or}))))\{s_1\}$$

2nd iteration



$$\{s_1\}\mapsto\{s_8 s_1, s_{10}s_5-\}$$

$$\{s_5 s_1\}\mapsto\{s_{10}s_5 s_1, s_5 s_5-, -\}$$

$$\{s_5 s_5 s_1\}\mapsto\{s_5 s_5-, -\}$$

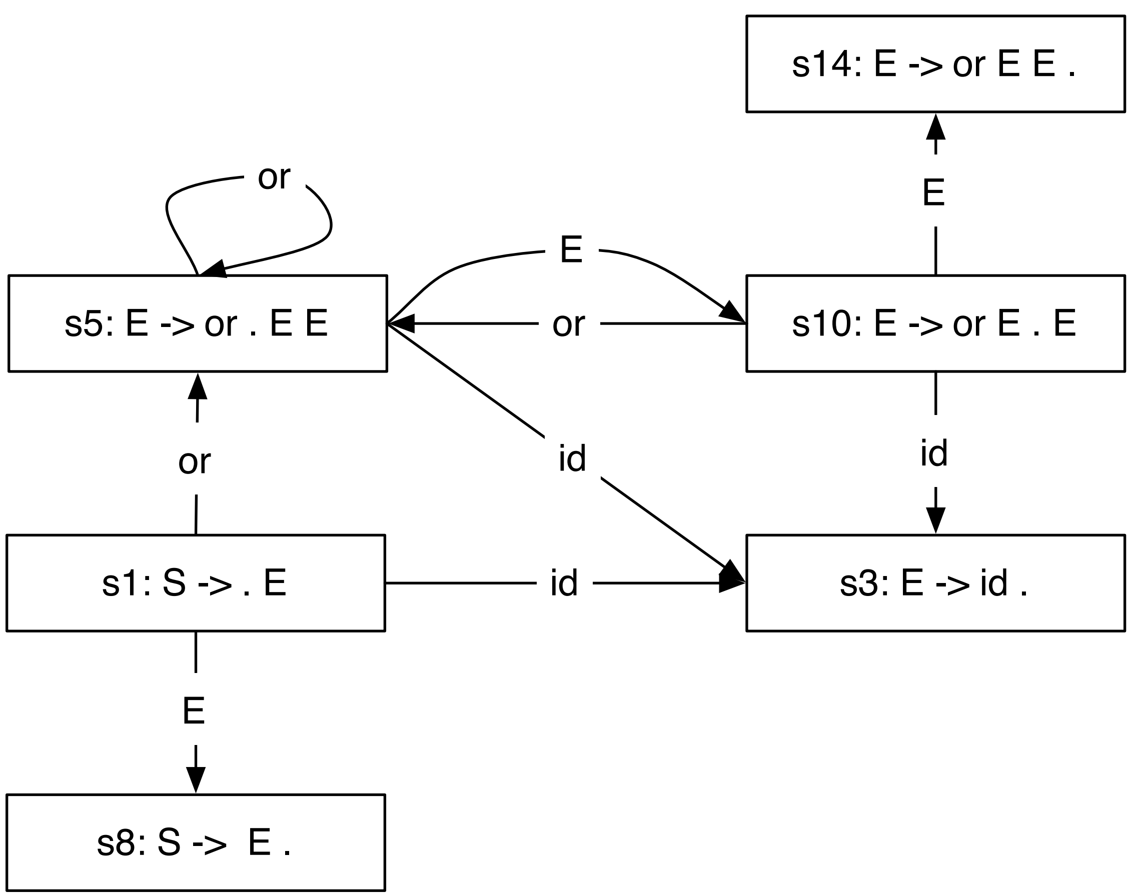
Instantiation of $D^\sharp$
$\widehat{D}$: Abstract Parsing Stack with k-cutting

- Example: `re x (`or . ,x) (`,x . b)` k = 3

$$[\![\texttt{re}\ x\ \text{‘}a\ (\text{‘}\texttt{or}\ .\ ,x)\ (\text{‘},\texttt{x}\ .\ \texttt{b})]\!]^0_{\widehat{D}}\sigma_0\{s_1\}$$

$$= (\lambda P.\widehat{PA}(P,\texttt{b}) \circ (\mathit{fix}\,\lambda k.\lambda P.(\widehat{PA}(P,\texttt{a}) \sqcup k \circ \widehat{PA}(P,\texttt{or}))))\{s_1\}$$

3rd iteration



$$\{s_1\} \mapsto \{s_8 s_1, s_{10} s_5 s_1, s_5 s_5 -, -\}$$

$$\{s_5 s_1\} \mapsto \{s_{10} s_5 s_1, s_5 s_5 -, -\}$$

$$\{s_5 s_5 s_1\} \mapsto \{s_5 s_5 -, -\}$$

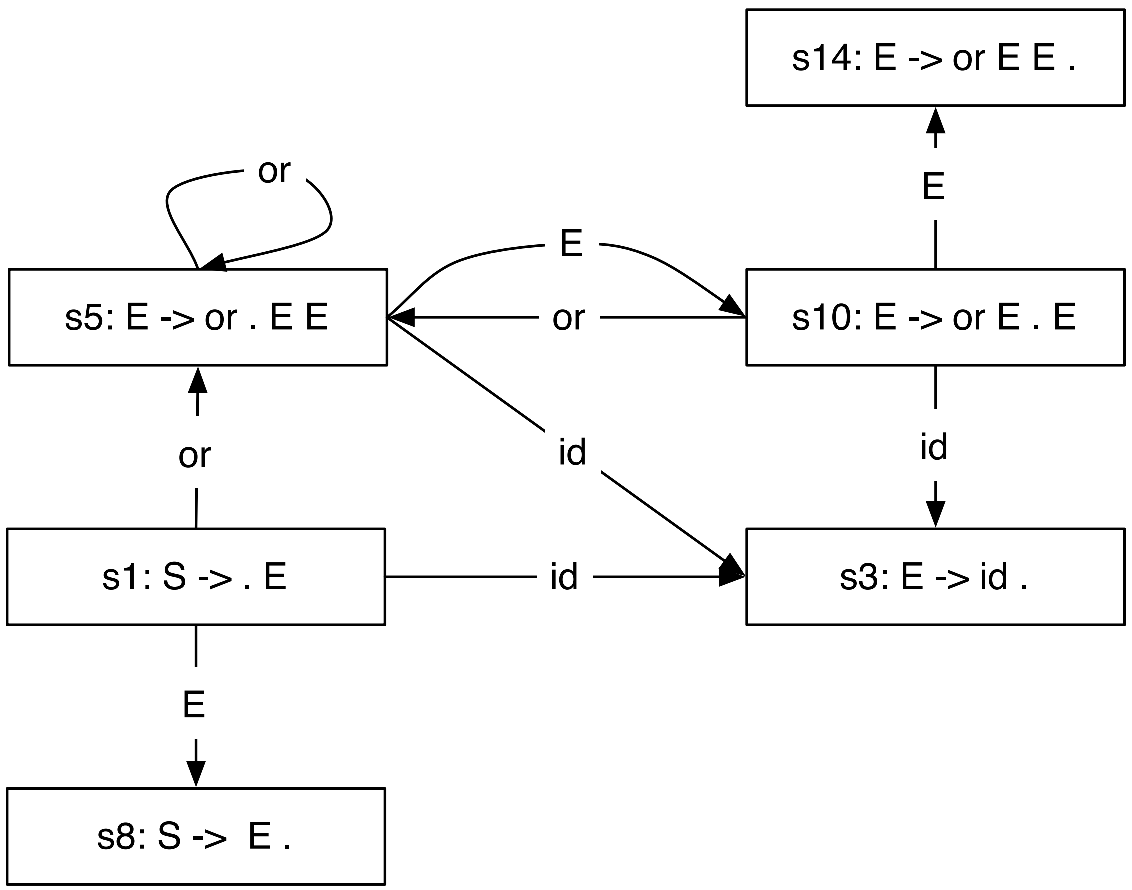
Instantiation of $D^\sharp$
$\widehat{D}$: Abstract Parsing Stack with k-cutting

- Example: `re x (`or . ,x) (`,x . b)` k = 3

$$[\![\texttt{re}\ x\ \text{‘}a\ (\text{‘}\texttt{or}\ .\ ,x)\ (\text{‘},\texttt{x}\ .\ \texttt{b})]\!]^0_{\widehat{D}}\sigma_0\{s_1\}$$

$$= (\lambda P.\widehat{PA}(P,\texttt{b}) \circ (\mathit{fix}\,\lambda k.\lambda P.(\widehat{PA}(P,\texttt{a}) \sqcup k \circ \widehat{PA}(P,\texttt{or}))))\{s_1\}$$

4th iteration



$$\{s_1\} \mapsto \{s_8 s_1, s_{10} s_5 s_1, s_5 s_5 -, -\}$$

$$\{s_5 s_1\} \mapsto \{s_{10} s_5 s_1, s_5 s_5 -, -\}$$

$$\{s_5 s_5 s_1\} \mapsto \{s_5 s_5 -, -\}$$

Fixed Point!

ROSAEC center
Research On Software Analysis for Error-free Computing
소프트웨어 무결점 연구센터 KOSEF ERC

Conclusion

- We formalize and generalize abstract parsing in the abstract interpretation framework.

Collecting Semantics	$\longrightarrow$	Concrete Parsing Semantics	$\longrightarrow$	First Step Abstraction Semantics	$\longrightarrow$	Parameterized Abstract Parsing Semantics
2^{Code}		$2^{P \rightarrow P}$		$2^P \rightarrow 2^P$		$D^\sharp \rightarrow D^\sharp$

Abstraction Steps for the Value Domain

- Apply abstract parsing to the two-staged languages.

Thank you

ROSAEC center
Research On Software Analysis for Error-free Computing
소프트웨어 무결점 연구센터 KOSEF ERC