Noncanonical Parsing

Sylvain Schmitz
schmitz@i3s.unice.fr

10 mai 2006

Java Declarations

Gosling et al. [1996]

Example

```
public class Declarations {
    public final static int FIELD = 10;
    public static int method (int i) {
        // ...
    }
}
```

Java Grammar

Gosling et al. [1996]

Example

```
public class Declarations {
    public final static int FIELD = 10;
    public static int method (int i) {
        // ...
    }
}
```

$$<FDecl> \rightarrow <FModS>\ <Type>\ <VDeclS>\ ;$$
$$<FModS> \rightarrow <FModS><FMod> \mid \varepsilon$$
$$<FMod> \rightarrow \texttt{public} \mid \texttt{final} \mid \texttt{static} \mid \texttt{transient} \mid \ldots$$
$$<Type> \rightarrow <NonVoidType>$$
$$<MHead> \rightarrow <MModS><ResType><MDecl><Throws>$$
$$<MModS> \rightarrow <MModS><MMod> \mid \varepsilon$$
$$<MMod> \rightarrow \texttt{public} \mid \texttt{final} \mid \texttt{static} \mid \texttt{abstract} \mid \ldots$$
$$<ResType> \rightarrow <NonVoidType> \mid \texttt{void}$$

Java Grammar

Gosling et al. [1996]

Example

```
public class Declarations {
    public final static int FIELD = 10;
    public static int method (int i) {
        // ...
    }
}
```

$$<FDecl> \rightarrow <FModS> \ <Type> \ <VDeclS> \ ;$$
$$<FModS> \rightarrow <FModS><FMod> \mid \varepsilon$$
$$<FMod> \rightarrow \texttt{public} \mid \texttt{final} \mid \texttt{static} \mid \texttt{transient} \mid \ldots$$
$$<Type> \rightarrow <NonVoidType>$$
$$<MHead> \rightarrow <MModS><ResType><MDecl><Throws>$$
$$<MModS> \rightarrow <MModS><MMod> \mid \varepsilon$$
$$<MMod> \rightarrow \texttt{public} \mid \texttt{final} \mid \texttt{static} \mid \texttt{abstract} \mid \ldots$$
$$<ResType> \rightarrow <NonVoidType> \mid \texttt{void}$$

Java Grammar

Gosling et al. [1996]

Example

```
public class Declarations {
    public final static int FIELD = 10;
    public static int method (int i) {
        // ...
    }
}
```

$$<FDecl> \rightarrow <FModS> \ <Type> \ <VDeclS> \ ;$$

$$<FModS> \rightarrow <FModS><FMod> \mid \varepsilon$$

$$<FMod> \rightarrow \texttt{public} \mid \texttt{final} \mid \texttt{static} \mid \texttt{transient} \mid \ldots$$

$$<Type> \rightarrow <NonVoidType>$$

$$<MHead> \rightarrow <MModS><ResType><MDecl><Throws>$$

$$<MModS> \rightarrow <MModS><MMod> \mid \varepsilon$$

$$<MMod> \rightarrow \texttt{public} \mid \texttt{final} \mid \texttt{static} \mid \texttt{abstract} \mid \ldots$$

$$<ResType> \rightarrow <NonVoidType> \mid \texttt{void}$$

Java Grammar

Gosling et al. [1996]

Example

```
public class Declarations {
    public final static int FIELD = 10;
    public static int method (int i) {
        // ...
    }
}
```

$$<FDecl> \rightarrow <FModS>\ <Type>\ <VDeclS>\ ;$$
$$<FModS> \rightarrow <FModS><FMod> \mid \varepsilon$$
$$<FMod> \rightarrow \texttt{public} \mid \texttt{final} \mid \texttt{static} \mid \texttt{transient} \mid \ldots$$
$$<Type> \rightarrow <NonVoidType>$$
$$<MHead> \rightarrow <MModS><ResType><MDecl><Throws>$$
$$<MModS> \rightarrow <MModS><MMod> \mid \varepsilon$$
$$<MMod> \rightarrow \texttt{public} \mid \texttt{final} \mid \texttt{static} \mid \texttt{abstract} \mid \ldots$$
$$<ResType> \rightarrow <NonVoidType> \mid \texttt{void}$$

Java Grammar

Gosling et al. [1996]

Example

```
public class Declarations {
    public final static int FIELD = 10;
    public static int method (int i) {
        // ...
    }
}
```

$$<FDecl> \rightarrow <FModS>\ <Type>\ <VDeclS>\ ;$$
$$<FModS> \rightarrow <FModS><FMod> \mid \varepsilon$$
$$<FMod> \rightarrow \texttt{public} \mid \texttt{final} \mid \texttt{static} \mid \texttt{transient} \mid \ldots$$
$$<Type> \rightarrow <NonVoidType>$$
$$<MHead> \rightarrow <MModS><ResType><MDecl><Throws>$$
$$<MModS> \rightarrow <MModS><MMod> \mid \varepsilon$$
$$<MMod> \rightarrow \texttt{public} \mid \texttt{final} \mid \texttt{static} \mid \texttt{abstract} \mid \ldots$$
$$<ResType> \rightarrow <NonVoidType> \mid \texttt{void}$$

Java Grammar

Gosling et al. [1996]

Example

```
public class Declarations {
    public final static int FIELD = 10;
    public static int method (int i) {
        // ...
    }
}
```

$$<FDecl> \rightarrow <FModS> \ <Type> \ <VDeclS> \ ;$$

$$<FModS> \rightarrow <FModS><FMod> \mid \varepsilon$$

$$<FMod> \rightarrow \texttt{public} \mid \texttt{final} \mid \texttt{static} \mid \texttt{transient} \mid \ldots$$

$$<Type> \rightarrow <NonVoidType>$$

$$<MHead> \rightarrow <MModS><ResType><MDecl><Throws>$$

$$<MModS> \rightarrow <MModS><MMod> \mid \varepsilon$$

$$<MMod> \rightarrow \texttt{public} \mid \texttt{final} \mid \texttt{static} \mid \texttt{abstract} \mid \ldots$$

$$<ResType> \rightarrow <NonVoidType> \mid \texttt{void}$$

Java Grammar

Gosling et al. [1996]

Example

```
public class Declarations {
    public final static int FIELD = 10;
    public static int method (int i) {
        // ...
    }
}
```

$$<FDecl> \rightarrow <FModS> \ <Type> \ <VDeclS> \ ;$$

$$<FModS> \rightarrow <FModS><FMod> \mid \varepsilon$$

$$<FMod> \rightarrow \texttt{public} \mid \texttt{final} \mid \texttt{static} \mid \texttt{transient} \mid \ldots$$

$$<Type> \rightarrow <NonVoidType>$$

$$<MHead> \rightarrow <MModS><ResType><MDecl><Throws>$$

$$<MModS> \rightarrow <MModS><MMod> \mid \varepsilon$$

$$<MMod> \rightarrow \texttt{public} \mid \texttt{final} \mid \texttt{static} \mid \texttt{abstract} \mid \ldots$$

$$<ResType> \rightarrow <NonVoidType> \mid \texttt{void}$$

Java Grammar

Gosling et al. [1996]

Example

```
public class Declarations {
    public final static int FIELD = 10;
    public static int method (int i) {
        // ...
    }
}
```

$$\langle FDecl \rangle \rightarrow \langle FModS \rangle \ \langle Type \rangle \ \langle VDeclS \rangle \ ;$$

$$\langle FModS \rangle \rightarrow \langle FModS \rangle \langle FMod \rangle \mid \varepsilon$$

$$\langle FMod \rangle \rightarrow \texttt{public} \mid \texttt{final} \mid \texttt{static} \mid \texttt{transient} \mid \ldots$$

$$\langle Type \rangle \rightarrow \langle NonVoidType \rangle$$

$$\langle MHead \rangle \rightarrow \langle MModS \rangle \langle ResType \rangle \langle MDecl \rangle \langle Throws \rangle$$

$$\langle MModS \rangle \rightarrow \langle MModS \rangle \langle MMod \rangle \mid \varepsilon$$

$$\langle MMod \rangle \rightarrow \texttt{public} \mid \texttt{final} \mid \texttt{static} \mid \texttt{abstract} \mid \ldots$$

$$\langle ResType \rangle \rightarrow \langle NonVoidType \rangle \mid \texttt{void}$$

Java Grammar

Gosling et al. [1996]

Example

```
public class Declarations {
    public final static int FIELD = 10;
    public static int method (int i) {
        // ...
    }
}
```

$$\langle FDecl \rangle \rightarrow \langle FModS \rangle \ \langle Type \rangle \ \langle VDeclS \rangle \ ;$$

$$\langle FModS \rangle \rightarrow \langle FModS \rangle \langle FMod \rangle \mid \varepsilon$$

$$\langle FMod \rangle \rightarrow \texttt{public} \mid \texttt{final} \mid \texttt{static} \mid \texttt{transient} \mid \ldots$$

$$\langle Type \rangle \rightarrow \langle NonVoidType \rangle$$

$$\langle MHead \rangle \rightarrow \langle MModS \rangle \langle ResType \rangle \langle MDecl \rangle \langle Throws \rangle$$

$$\langle MModS \rangle \rightarrow \langle MModS \rangle \langle MMod \rangle \mid \varepsilon$$

$$\langle MMod \rangle \rightarrow \texttt{public} \mid \texttt{final} \mid \texttt{static} \mid \texttt{abstract} \mid \ldots$$

$$\langle ResType \rangle \rightarrow \langle NonVoidType \rangle \mid \texttt{void}$$

Java Grammar

Gosling et al. [1996]

Example

```
public class Declarations {
    public final static int FIELD = 10;
    public static int method (int i) {
        // ...
    }
}
```

$$<FDecl> \rightarrow <FModS>\ <Type>\ <VDeclS>\ ;$$
$$<FModS> \rightarrow <FModS><FMod> \mid \varepsilon$$
$$<FMod> \rightarrow \texttt{public} \mid \texttt{final} \mid \texttt{static} \mid \texttt{transient} \mid \ldots$$
$$<Type> \rightarrow <NonVoidType>$$
$$<MHead> \rightarrow <MModS><ResType><MDecl><Throws>$$
$$<MModS> \rightarrow <MModS><MMod> \mid \varepsilon$$
$$<MMod> \rightarrow \texttt{public} \mid \texttt{final} \mid \texttt{static} \mid \texttt{abstract} \mid \ldots$$
$$<ResType> \rightarrow <NonVoidType> \mid \texttt{void}$$

Java Grammar

Gosling et al. [1996]

Example

```
public class Declarations {
    public final static int FIELD = 10;
    public static int method (int i) {
        // ...
    }
}
```

$$\langle FDecl \rangle \rightarrow \langle FModS \rangle \ \langle Type \rangle \ \langle VDeclS \rangle \ ;$$

$$\langle FModS \rangle \rightarrow \langle FModS \rangle \langle FMod \rangle \mid \varepsilon$$

$$\langle FMod \rangle \rightarrow \texttt{public} \mid \texttt{final} \mid \texttt{static} \mid \texttt{transient} \mid \ldots$$

$$\langle Type \rangle \rightarrow \langle NonVoidType \rangle$$

$$\langle MHead \rangle \rightarrow \langle MModS \rangle \langle ResType \rangle \langle MDecl \rangle \langle Throws \rangle$$

$$\langle MModS \rangle \rightarrow \langle MModS \rangle \langle MMod \rangle \mid \varepsilon$$

$$\langle MMod \rangle \rightarrow \texttt{public} \mid \texttt{final} \mid \texttt{static} \mid \texttt{abstract} \mid \ldots$$

$$\langle ResType \rangle \rightarrow \langle NonVoidType \rangle \mid \texttt{void}$$

Java Grammar

Gosling et al. [1996]

Example

```
public class Declarations {
    public final static int FIELD = 10;
    public static int method (int i) {
        // ...
    }
}
```

$$\langle FDecl \rangle \rightarrow \langle FModS \rangle \ \langle Type \rangle \ \langle VDeclS \rangle \ ;$$

$$\langle FModS \rangle \rightarrow \langle FModS \rangle \langle FMod \rangle \mid \varepsilon$$

$$\langle FMod \rangle \rightarrow \texttt{public} \mid \texttt{final} \mid \texttt{static} \mid \texttt{transient} \mid \ldots$$

$$\langle Type \rangle \rightarrow \langle NonVoidType \rangle$$

$$\langle MHead \rangle \rightarrow \langle MModS \rangle \langle ResType \rangle \langle MDecl \rangle \langle Throws \rangle$$

$$\langle MModS \rangle \rightarrow \langle MModS \rangle \langle MMod \rangle \mid \varepsilon$$

$$\langle MMod \rangle \rightarrow \texttt{public} \mid \texttt{final} \mid \texttt{static} \mid \texttt{abstract} \mid \ldots$$

$$\langle ResType \rangle \rightarrow \langle NonVoidType \rangle \mid \texttt{void}$$

Java Grammar

Gosling et al. [1996]

Example

```
public class Declarations {
    public final static int FIELD = 10;
    public static int method (int i) {
        // ...
    }
}
```

$<FDecl> \rightarrow <FModS>\ <Type>\ <VDeclS>\ ;$

$<FModS> \rightarrow <FModS><FMod> \mid \varepsilon$

$<FMod> \rightarrow$ `public` $\mid$ `final` $\mid$ `static` $\mid$ `transient` $\mid \ldots$

$<Type> \rightarrow <NonVoidType>$

$<MHead> \rightarrow <MModS><ResType><MDecl><Throws>$

$<MModS> \rightarrow <MModS><MMod> \mid \varepsilon$

$<MMod> \rightarrow$ `public` $\mid$ `final` $\mid$ `static` $\mid$ `abstract` $\mid \ldots$

$<ResType> \rightarrow <NonVoidType> \mid$ `void`

Java Grammar

Gosling et al. [1996]

Example

```
public class Declarations {
    public final static int FIELD = 10;
    public static int method (int i) {
        // ...
    }
}
```

$$<FDecl> \rightarrow <FModS> \ <Type> \ <VDeclS> \ ;$$

$$<FModS> \rightarrow <FModS><FMod> \mid \varepsilon$$

$$<FMod> \rightarrow \texttt{public} \mid \texttt{final} \mid \texttt{static} \mid \texttt{transient} \mid \ldots$$

$$<Type> \rightarrow <NonVoidType>$$

$$<MHead> \rightarrow <MModS><ResType><MDecl><Throws>$$

$$<MModS> \rightarrow <MModS><MMod> \mid \varepsilon$$

$$<MMod> \rightarrow \texttt{public} \mid \texttt{final} \mid \texttt{static} \mid \texttt{abstract} \mid \ldots$$

$$<ResType> \rightarrow <NonVoidType> \mid \texttt{void}$$

Java Grammar

Gosling et al. [1996]

Example

```
public class Declarations {
    public final static int FIELD = 10;
    public static int method (int i) {
        // ...
    }
}
```

$$\langle FDecl \rangle \rightarrow \langle FModS \rangle \ \langle Type \rangle \ \langle VDeclS \rangle \ ;$$

$$\langle FModS \rangle \rightarrow \langle FModS \rangle \langle FMod \rangle \mid \varepsilon$$

$$\langle FMod \rangle \rightarrow \texttt{public} \mid \texttt{final} \mid \texttt{static} \mid \texttt{transient} \mid \ldots$$

$$\langle Type \rangle \rightarrow \langle NonVoidType \rangle$$

$$\langle MHead \rangle \rightarrow \langle MModS \rangle \langle ResType \rangle \langle MDecl \rangle \langle Throws \rangle$$

$$\langle MModS \rangle \rightarrow \langle MModS \rangle \langle MMod \rangle \mid \varepsilon$$

$$\langle MMod \rangle \rightarrow \texttt{public} \mid \texttt{final} \mid \texttt{static} \mid \texttt{abstract} \mid \ldots$$

$$\langle ResType \rangle \rightarrow \langle NonVoidType \rangle \mid \texttt{void}$$

Parsing Java Declarations

Example

```
public final static int FIELD = 10 ;
```

$ ‖ public final static int FIELD = 10;$

Parsing Java Declarations

Example

FModS

ε public final static int FIELD = 10 ;

$ε ‖ public final static int FIELD = 10;$

$\vdash_{\text{reduce}}$ $<FModS> ‖ public final static int FIELD = 10;$

Parsing Java Declarations

Example

FModS

ε public final static int FIELD = 10 ;

$\$ <\mathit{FModS}> \parallel \texttt{public final static int FIELD = 10;}\$$

$\vdash_{\text{shift}} \$ <\mathit{FModS}> \texttt{public} \parallel \texttt{final static int FIELD = 10;}\$$

Parsing Java Declarations

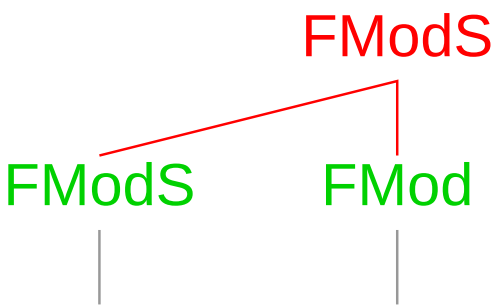
Example

$$\$ < FModS > \texttt{public} \parallel \texttt{final static int FIELD = 10;\$}$$

$$\vdash_{\text{reduce}} \$ < FModS > < FMod > \parallel \texttt{final static int FIELD = 10;\$}$$

Parsing Java Declarations

Example

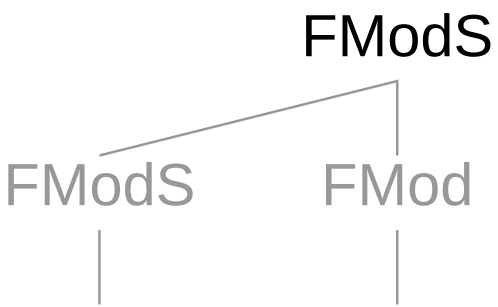


$$\$ < FModS > < FMod > \parallel \texttt{final static int FIELD = 10;} \$$$

$$\vdash_{\text{reduce}} \$ < FModS > \parallel \texttt{final static int FIELD = 10;} \$$$

Parsing Java Declarations

Example



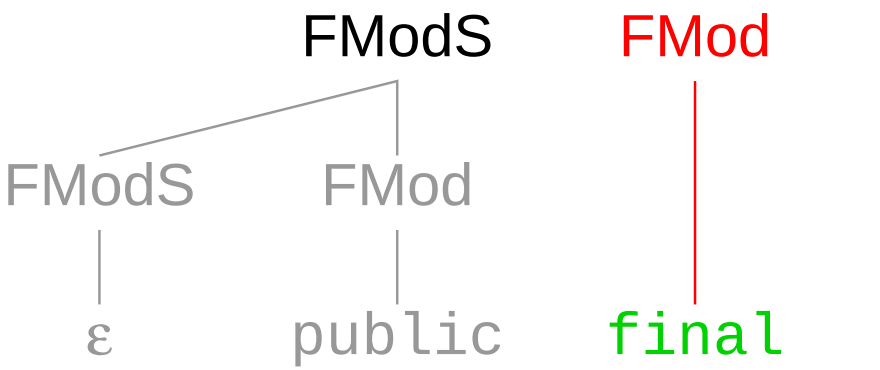
`public final static int FIELD = 10 ;`

$$\$ < FModS > \parallel \texttt{final static int FIELD = 10;} \$$$

$$\vdash_{\text{shift}} \$ < FModS > \texttt{final} \parallel \texttt{static int FIELD = 10;} \$$$

Parsing Java Declarations

Example

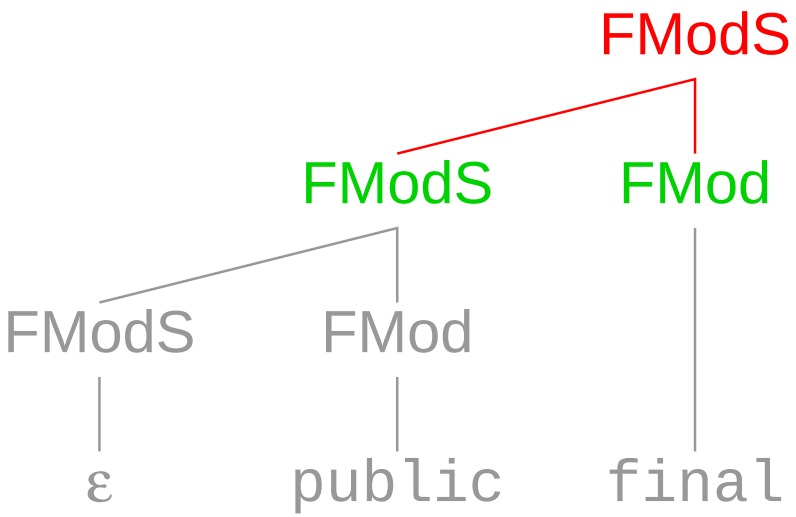


$$\$ < \mathit{FModS} > \texttt{final} \parallel \texttt{static int FIELD = 10;\$}$$

$$\models_{\text{reduce}} \$ < \mathit{FModS} > < \mathit{FMod} > \parallel \texttt{static int FIELD = 10;\$}$$

Parsing Java Declarations

Example

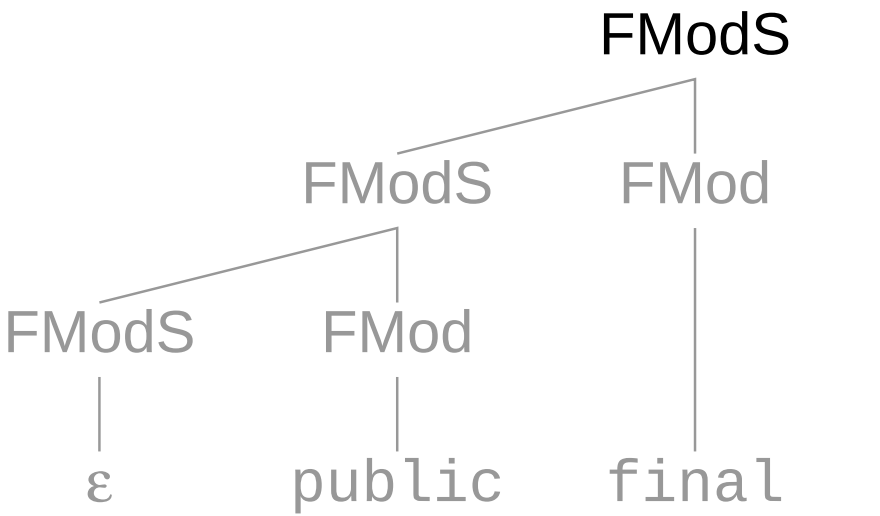


$$\$ < \mathit{FModS} > < \mathit{FMod} > \| \texttt{static int FIELD = 10;\$}$$

$$\models_{\text{reduce}} \$ < \mathit{FModS} > \| \texttt{static int FIELD = 10;\$}$$

Parsing Java Declarations

Example



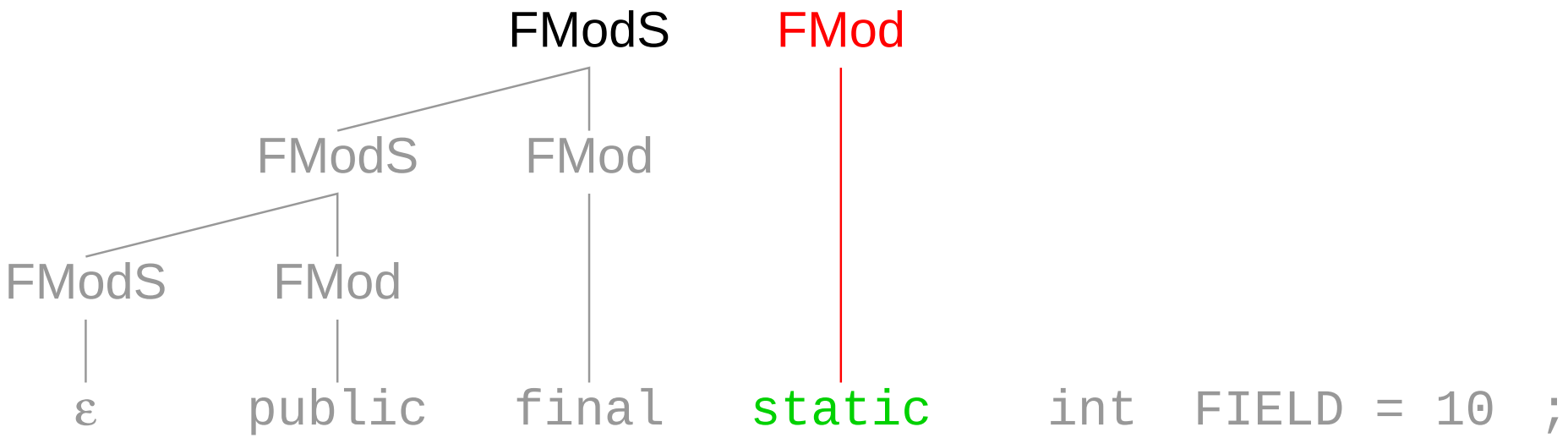
`static int FIELD = 10 ;`

$$\$ < FModS > \parallel \texttt{static int FIELD = 10;} \$$$

$$\vdash_{\text{shift}} \$ < FModS > \texttt{static} \parallel \texttt{int FIELD = 10;} \$$$

Parsing Java Declarations

Example

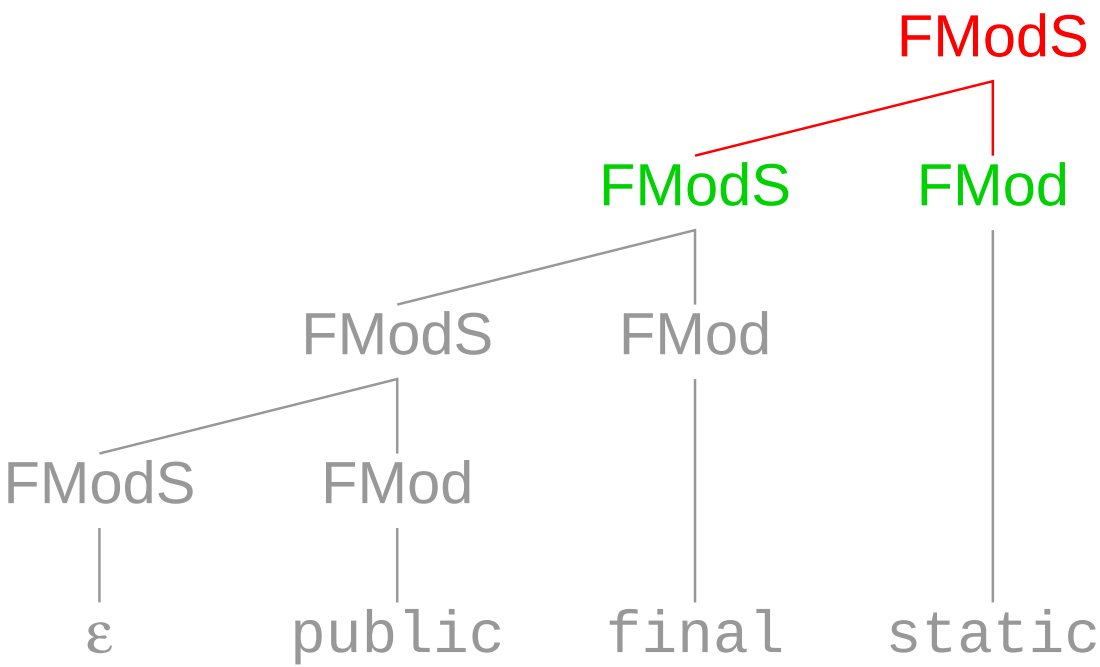


$$\$ < \mathit{FModS} > \texttt{static} \parallel \texttt{int FIELD = 10;\$}$$

$$\models_{\text{reduce}} \$ < \mathit{FModS} > < \mathit{FMod} > \parallel \texttt{int FIELD = 10;\$}$$

Parsing Java Declarations

Example

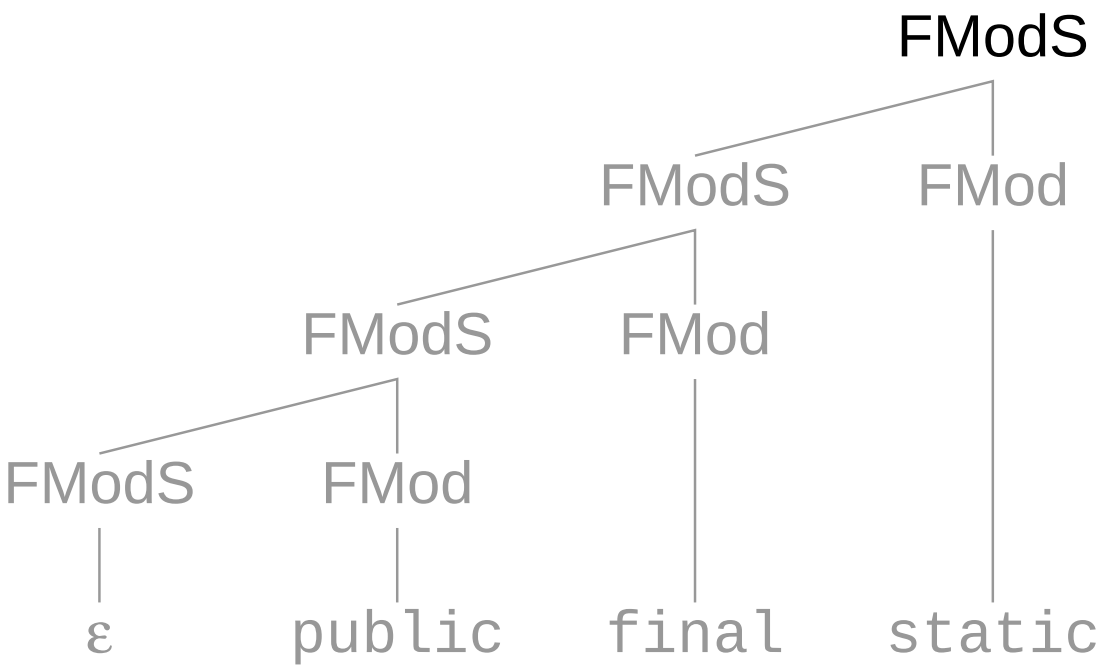


$$\$ < \mathit{FModS} > < \mathit{FMod} > \parallel \texttt{int FIELD = 10;\$}$$

$$\vdash_{\text{reduce}} \$ < \mathit{FModS} > \parallel \texttt{int FIELD = 10;\$}$$

Parsing Java Declarations

Example

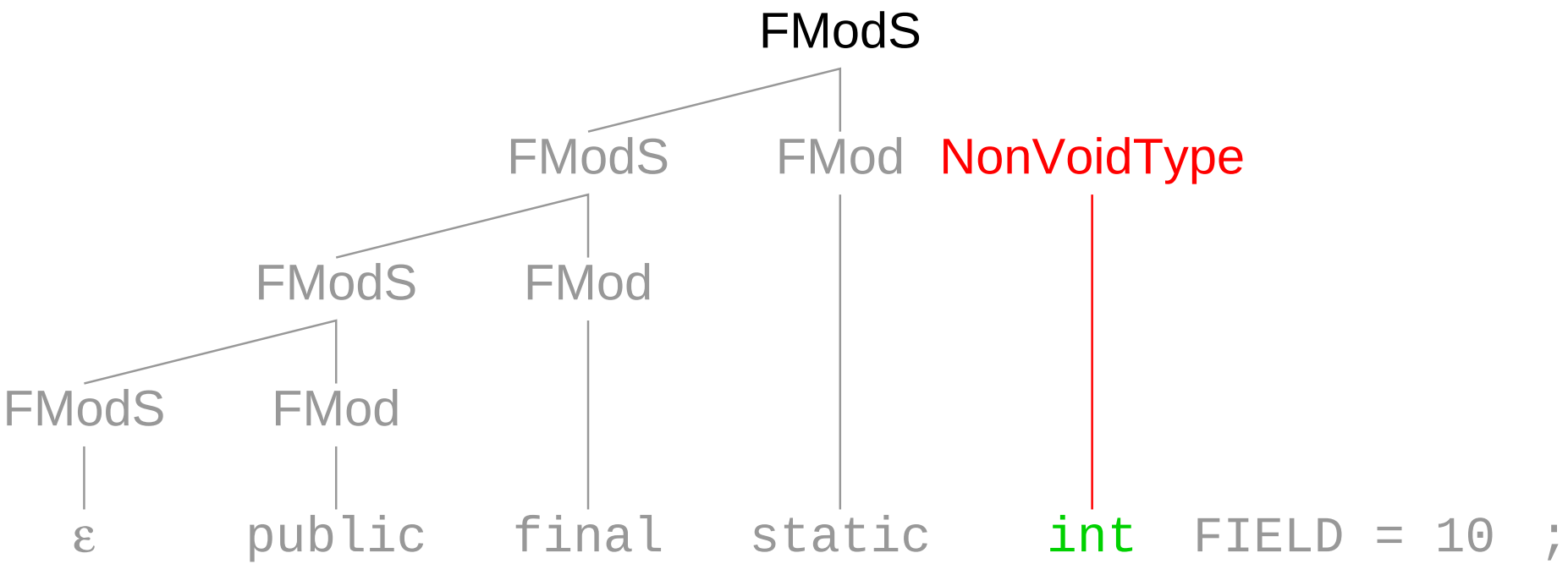


$$\$ < \mathit{FModS} > \parallel \texttt{int FIELD = 10;\$}$$

$$\vdash_{\text{shift}} \$ < \mathit{FModS} > \texttt{int} \parallel \texttt{FIELD = 10;\$}$$

Parsing Java Declarations

Example

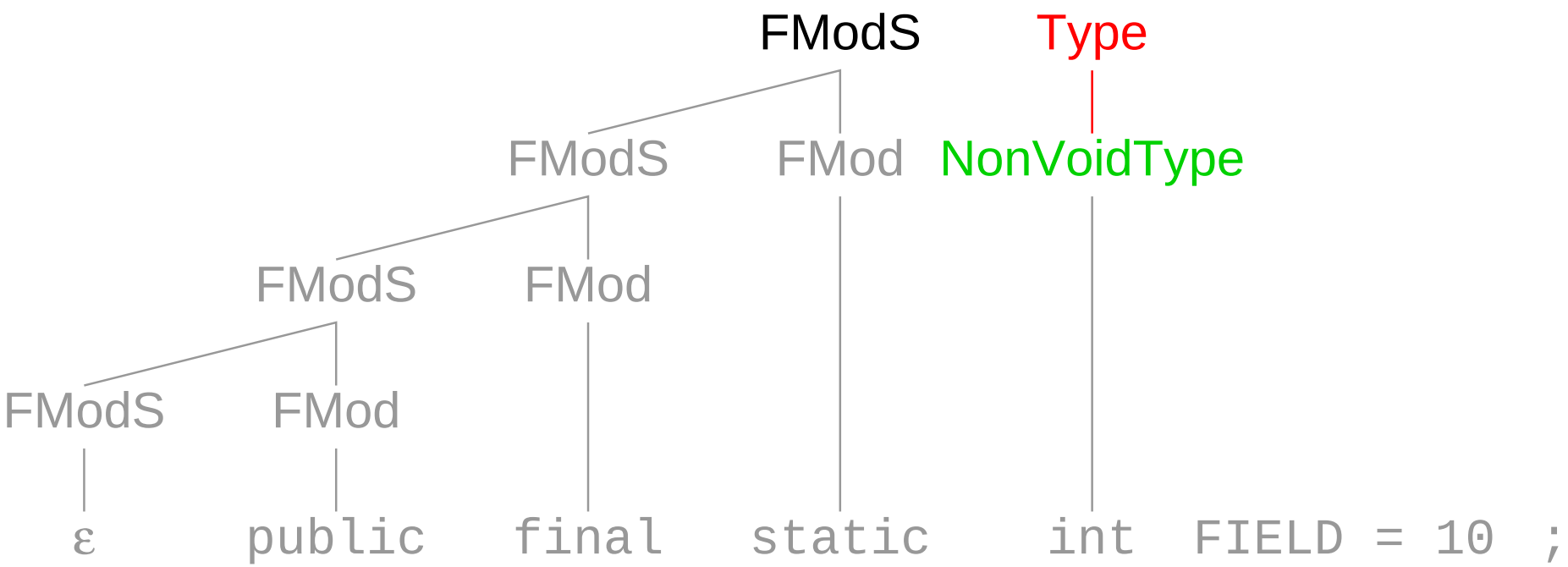


$$\$ < FModS > \texttt{int} \parallel \texttt{FIELD} = \texttt{10;}\$$$

$$\models_{\text{reduce}} \$ < FModS > < NonVoidType > \parallel \texttt{FIELD} = \texttt{10;}\$$$

Parsing Java Declarations

Example

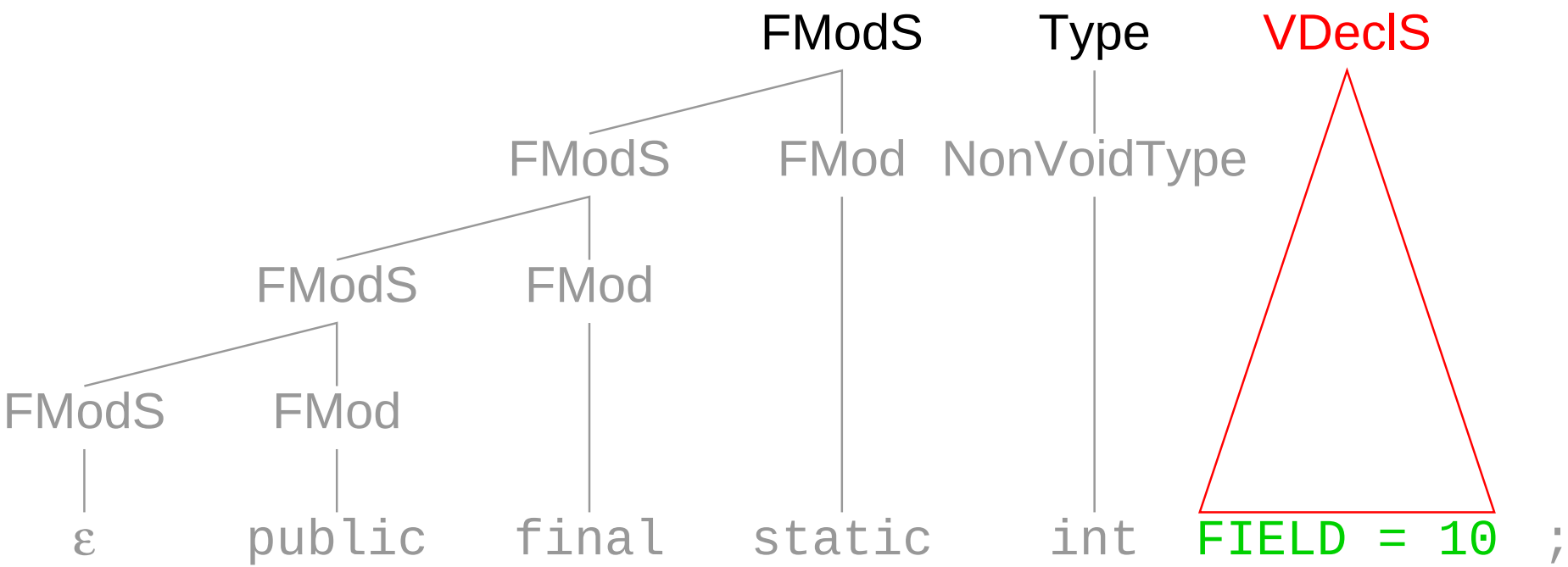


$$\$ < FModS > < NonVoidType > \| \texttt{FIELD} = \texttt{10;}\$$$

$$\models_{\text{reduce}} \$ < FModS > < Type > \| \texttt{FIELD} = \texttt{10;}\$$$

Parsing Java Declarations

Example

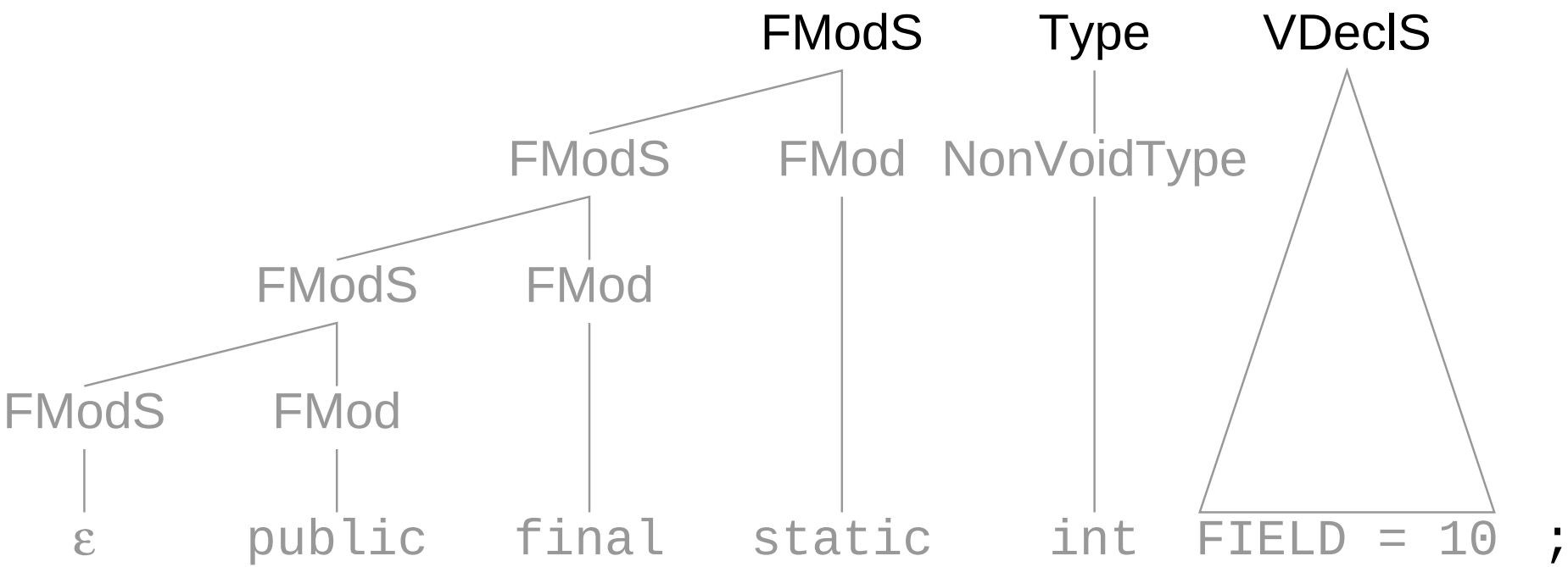


$$\$ < FModS > < Type > \texttt{FIELD} = 10 \parallel ;\$$$

$$\models_{\text{reduce}} \$ < FModS > < Type > < VDeclS > \parallel ;\$$$

Parsing Java Declarations

Example

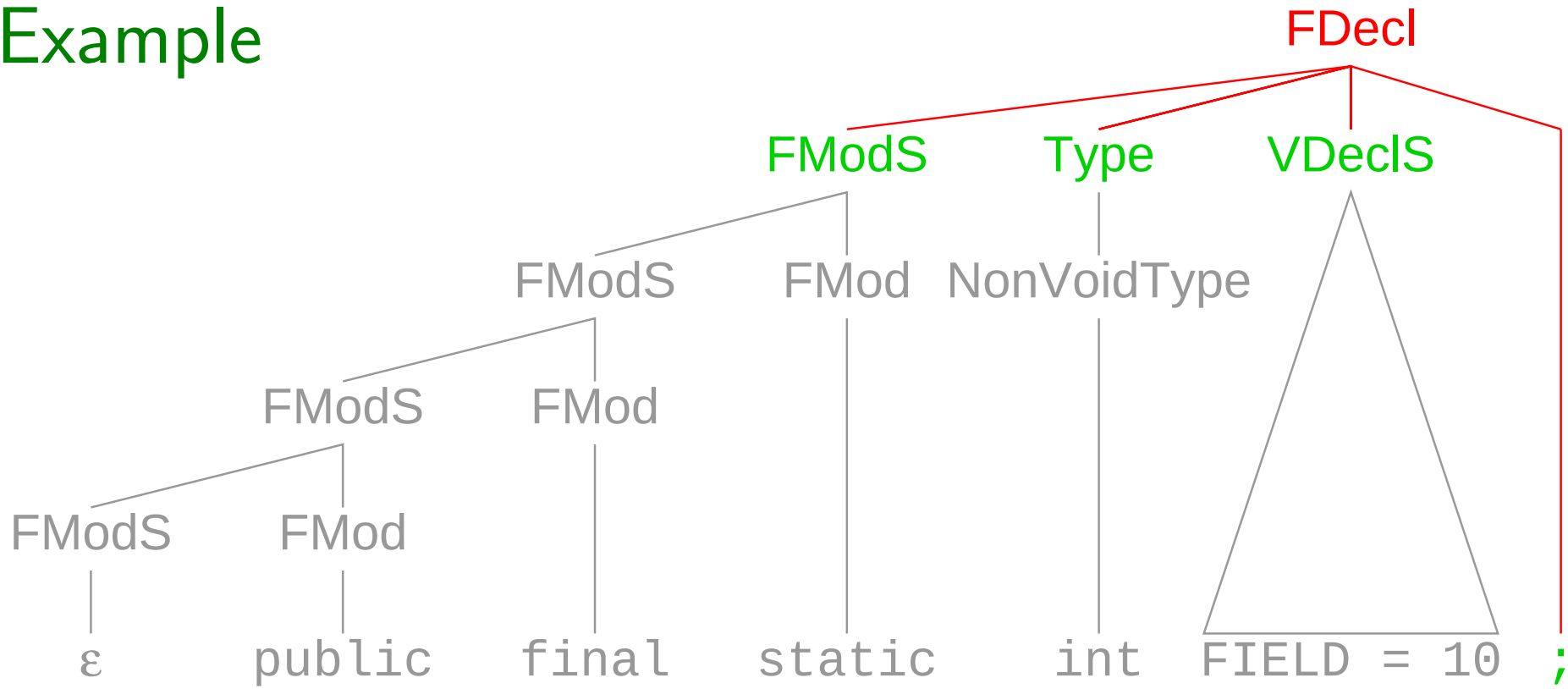


$$\$ < FModS >< Type >< VDeclS > \parallel ; \$$$

$$\vdash_{\text{shift}} \$ < FModS >< Type >< VDeclS > ; \parallel \$$$

Parsing Java Declarations

Example

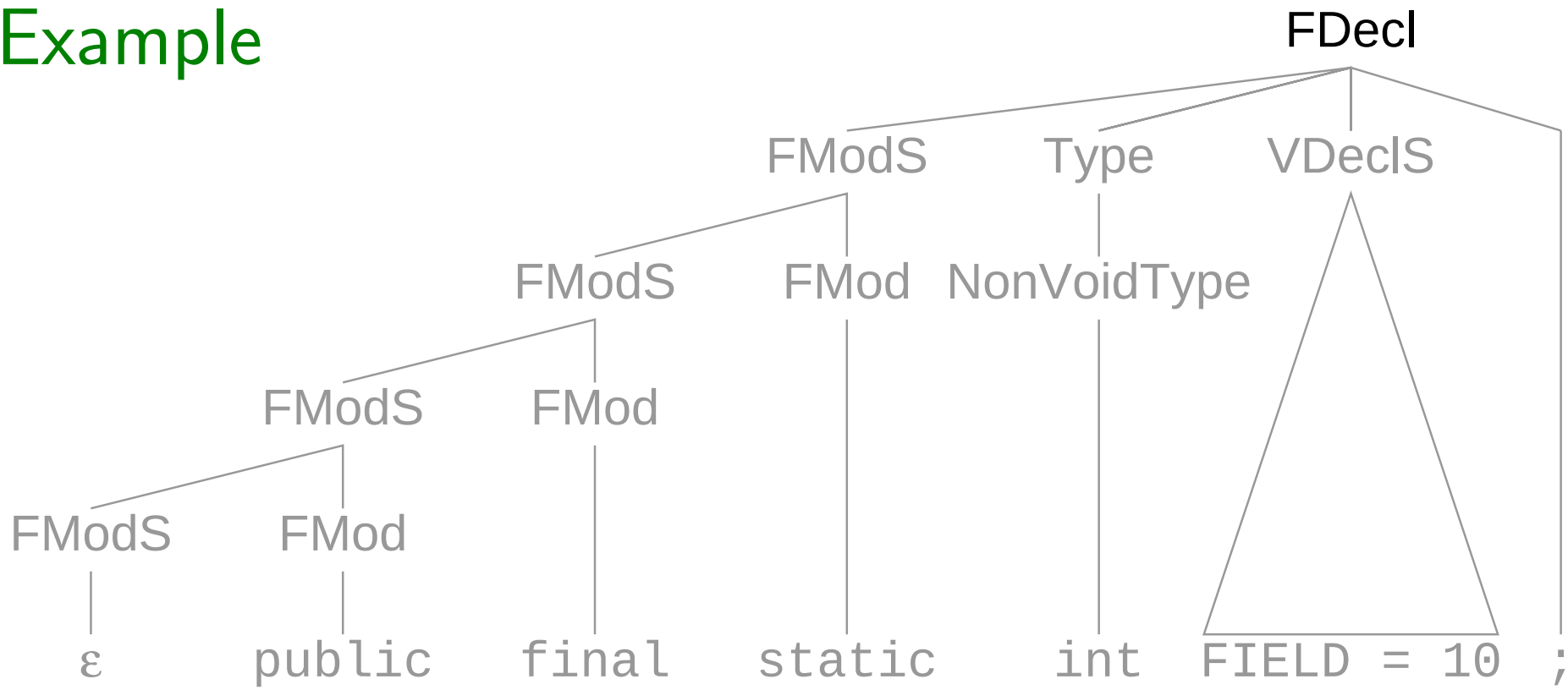


$$\$ < FModS > < Type > < VDeclS > \; ; \parallel \$$$

$$\vdash_{\text{reduce}} \$ < FDecl > \parallel \$$$

Parsing Java Declarations

Example



$$\$ < FDecl > \ \| \ \$$$

Nondeterminism

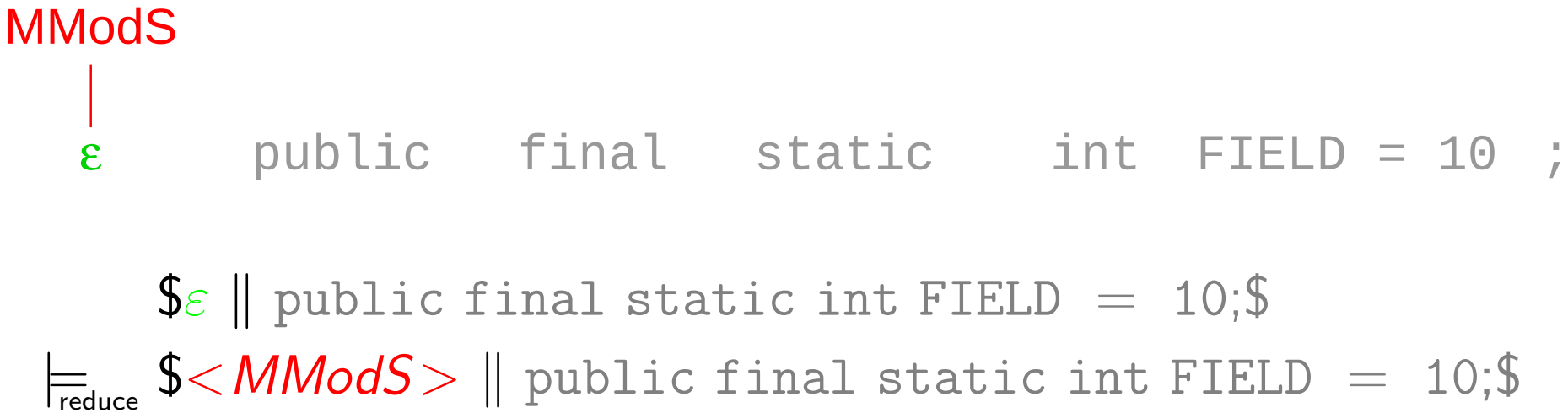
Example

```
public final static int FIELD = 10 ;
```

$ ‖ public final static int FIELD = 10;$

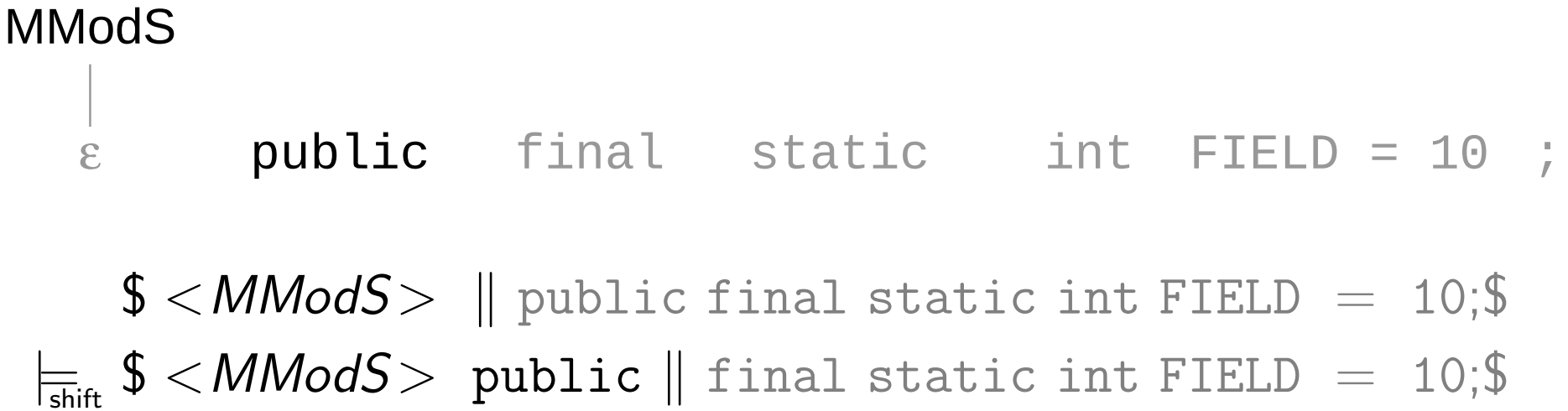
Nondeterminism

Example



Nondeterminism

Example



Nondeterminism

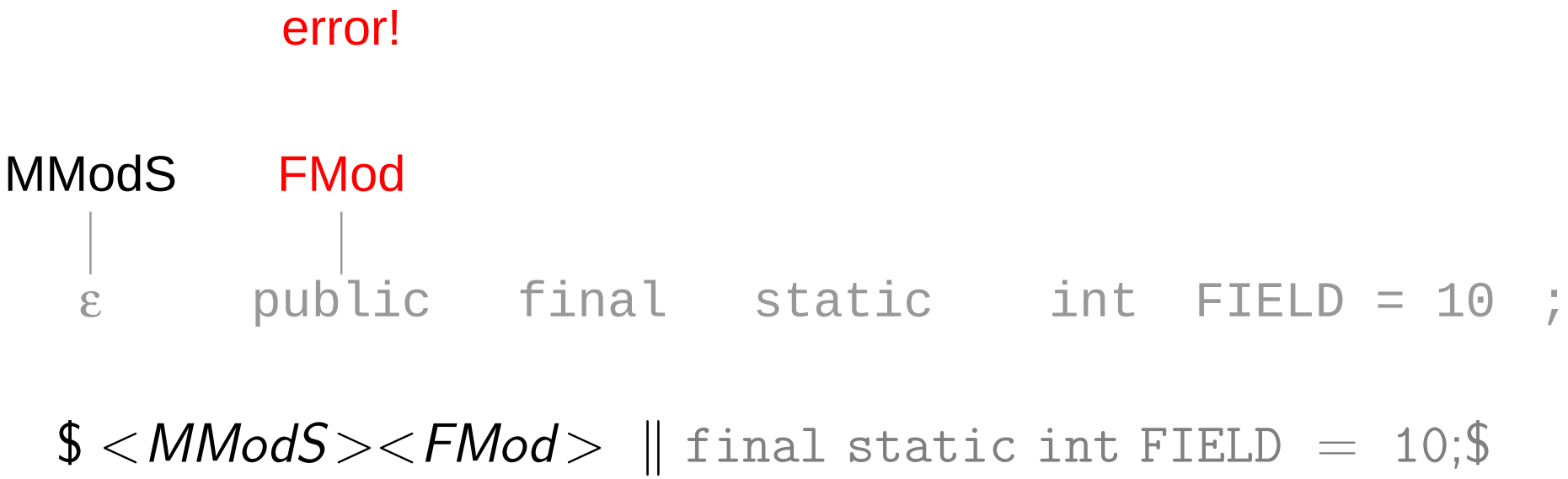
Example

$$\$ < \mathit{MModS} > \texttt{public} \parallel \texttt{final static int FIELD = 10;} \$$$

$$\vdash_{\text{reduce}} \$ < \mathit{MModS} > < \mathit{FMod} > \parallel \texttt{final static int FIELD = 10;} \$$$

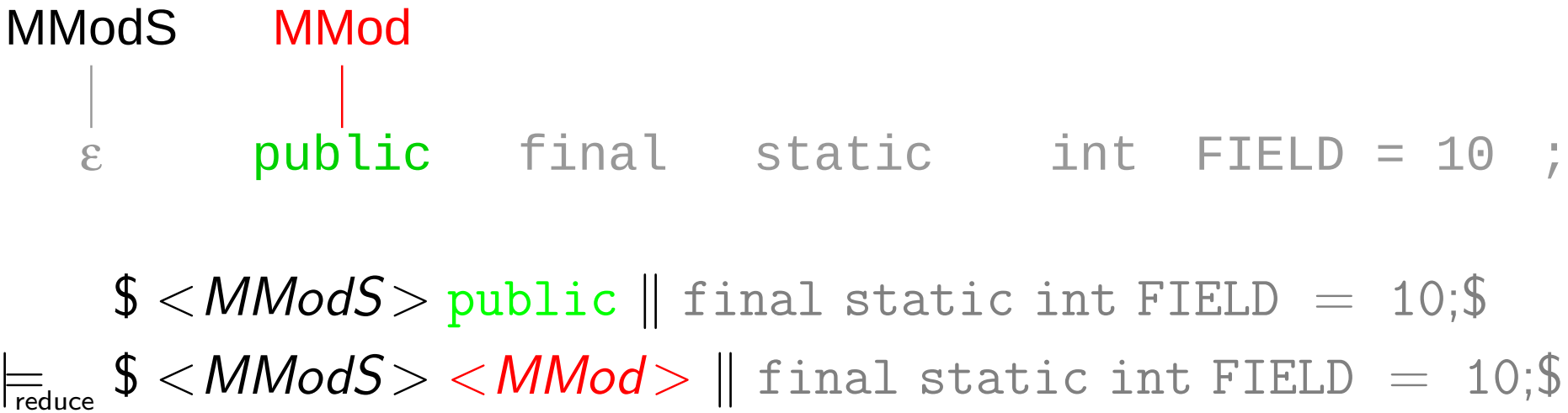
Nondeterminism

Example



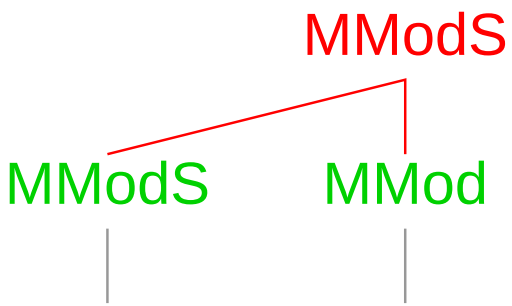
Nondeterminism

Example



Nondeterminism

Example

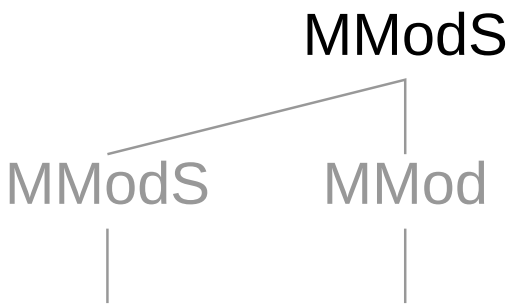


$$\$ \langle \mathit{MModS} \rangle \langle \mathit{MMod} \rangle \parallel \texttt{final static int FIELD = 10;\$}$$

$$\models_{\text{reduce}} \$ \langle \mathit{MModS} \rangle \parallel \texttt{final static int FIELD = 10;\$}$$

Nondeterminism

Example



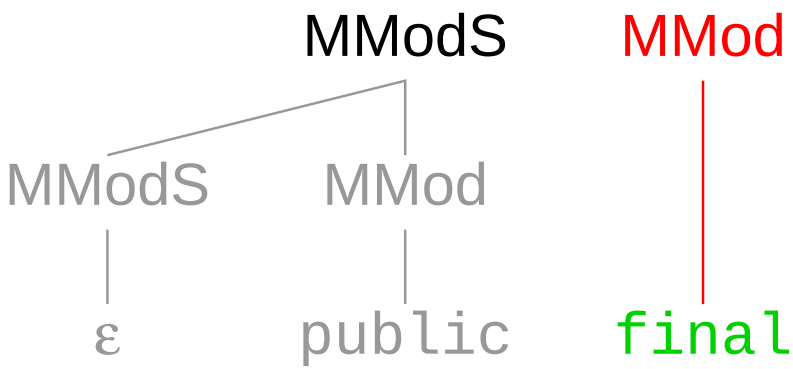

```
public final static int FIELD = 10 ;
```

$$\$ < \mathit{MModS} > \parallel \texttt{final static int FIELD = 10;\$}$$

$$\vdash_{\text{shift}} \$ < \mathit{MModS} > \texttt{final} \parallel \texttt{static int FIELD = 10;\$}$$

Nondeterminism

Example

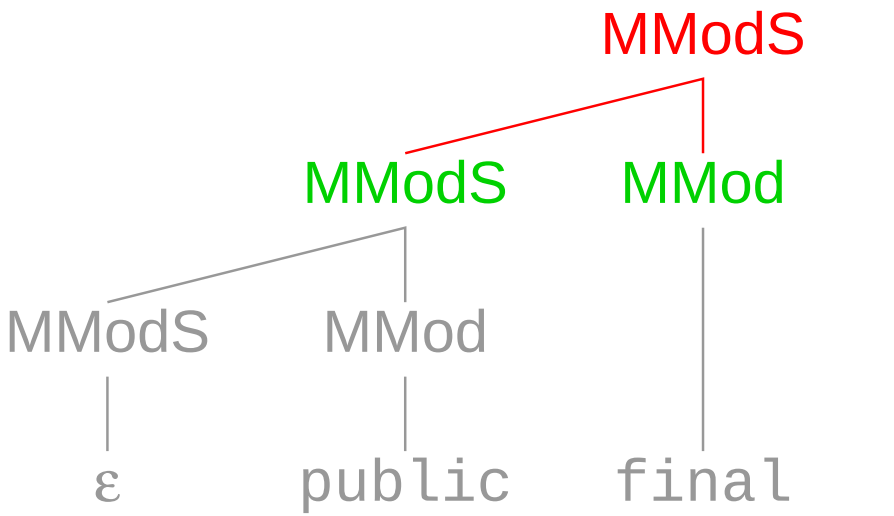


$$\$ < \mathit{MModS} > \texttt{final} \parallel \texttt{static int FIELD = 10;\$}$$

$$\vdash_{\text{reduce}} \$ < \mathit{MModS} > < \mathit{MMod} > \parallel \texttt{static int FIELD = 10;\$}$$

Nondeterminism

Example

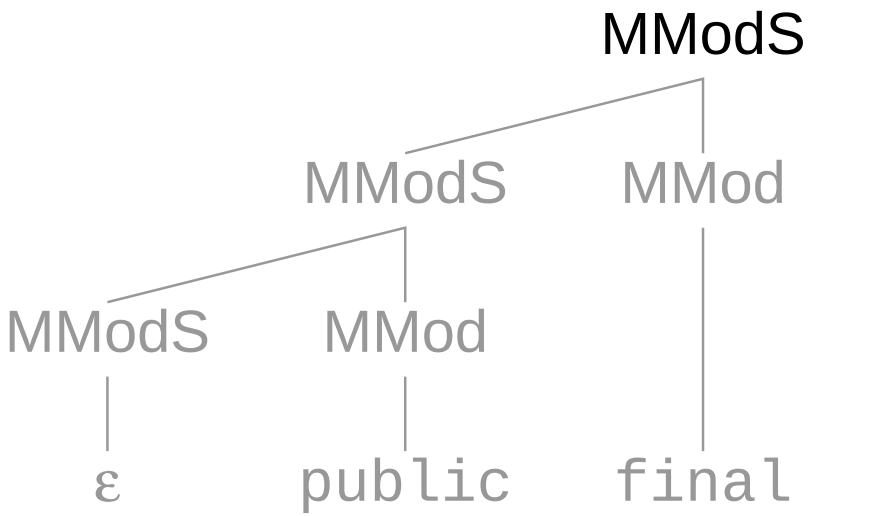


$$\$ < \mathit{MModS} > < \mathit{MMod} > \| \texttt{static int FIELD = 10;\$}$$

$$\vdash_{\text{reduce}} \$ < \mathit{MModS} > \| \texttt{static int FIELD = 10;\$}$$

Nondeterminism

Example



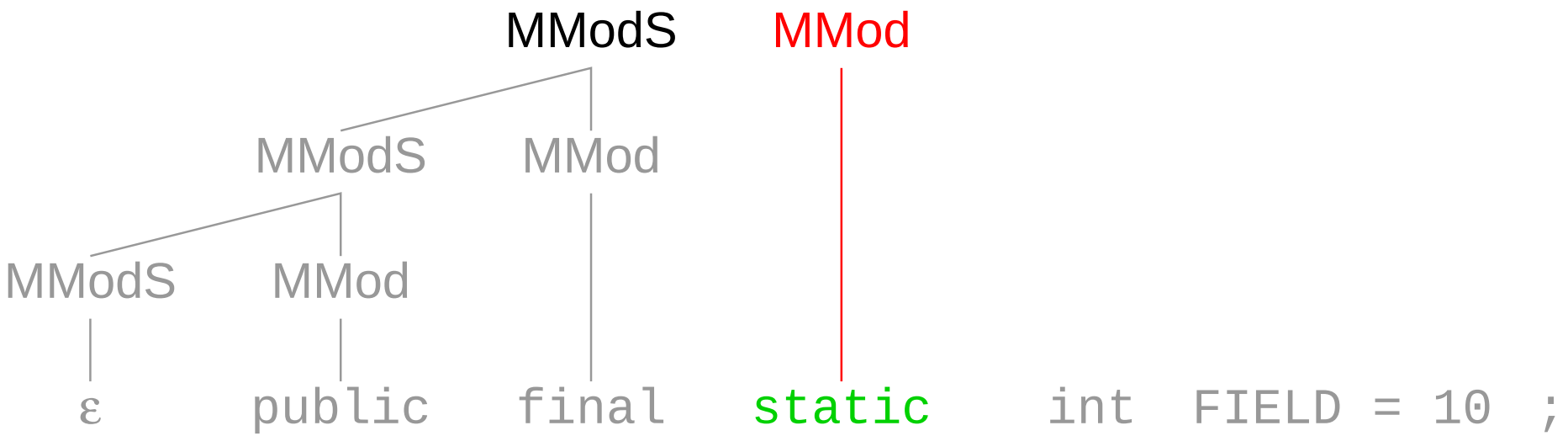
`static int FIELD = 10 ;`

$$\$ < MModS > \parallel \texttt{static int FIELD = 10;} \$$$

$$\vdash_{\text{shift}} \$ < MModS > \texttt{static} \parallel \texttt{int FIELD = 10;} \$$$

Nondeterminism

Example

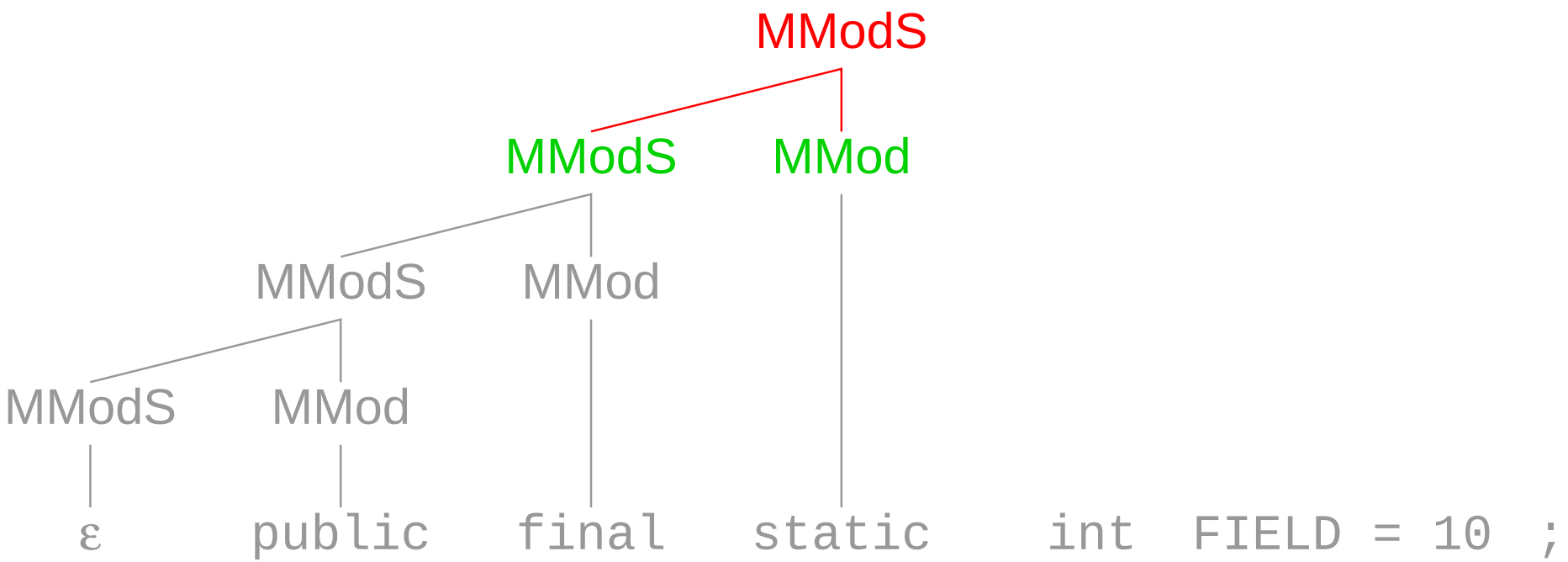


$$\$ < MModS > \texttt{static} \parallel \texttt{int FIELD = 10;\$}$$

$$\models_{\text{reduce}} \$ < MModS > < MMod > \parallel \texttt{int FIELD = 10;\$}$$

Nondeterminism

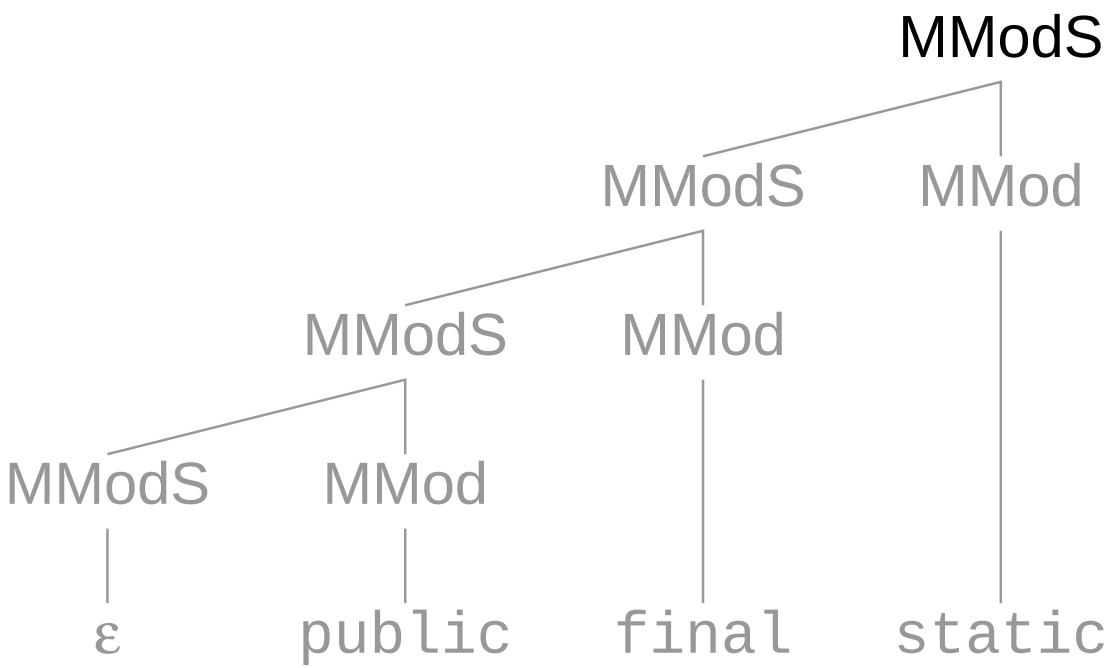
Example



$$\$ < \mathit{MModS} > < \mathit{MMod} > \| \texttt{int FIELD = 10;\$}$$
$$\models_{\text{reduce}} \$ < \mathit{MModS} > \| \texttt{int FIELD = 10;\$}$$

Nondeterminism

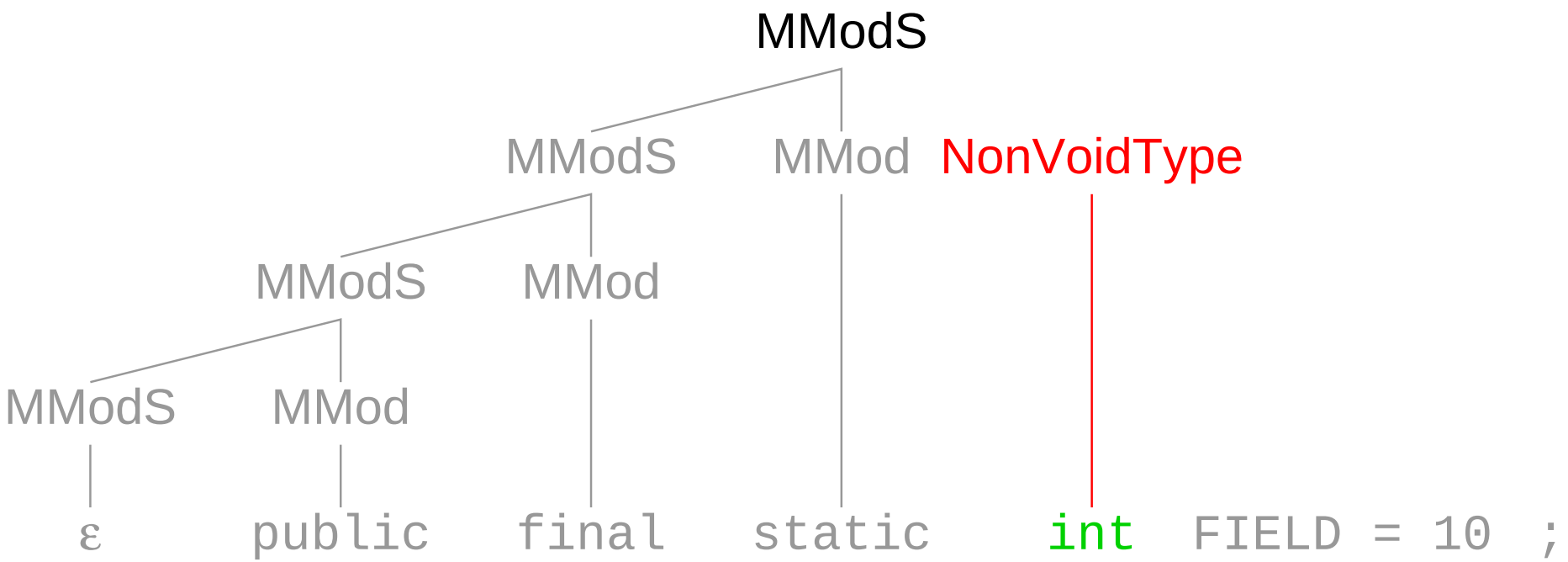
Example



$$\$ < \mathit{MModS} > \parallel \texttt{int FIELD = 10;\$}$$
$$\models_{\text{shift}} \$ < \mathit{MModS} > \texttt{int} \parallel \texttt{FIELD = 10;\$}$$

Nondeterminism

Example

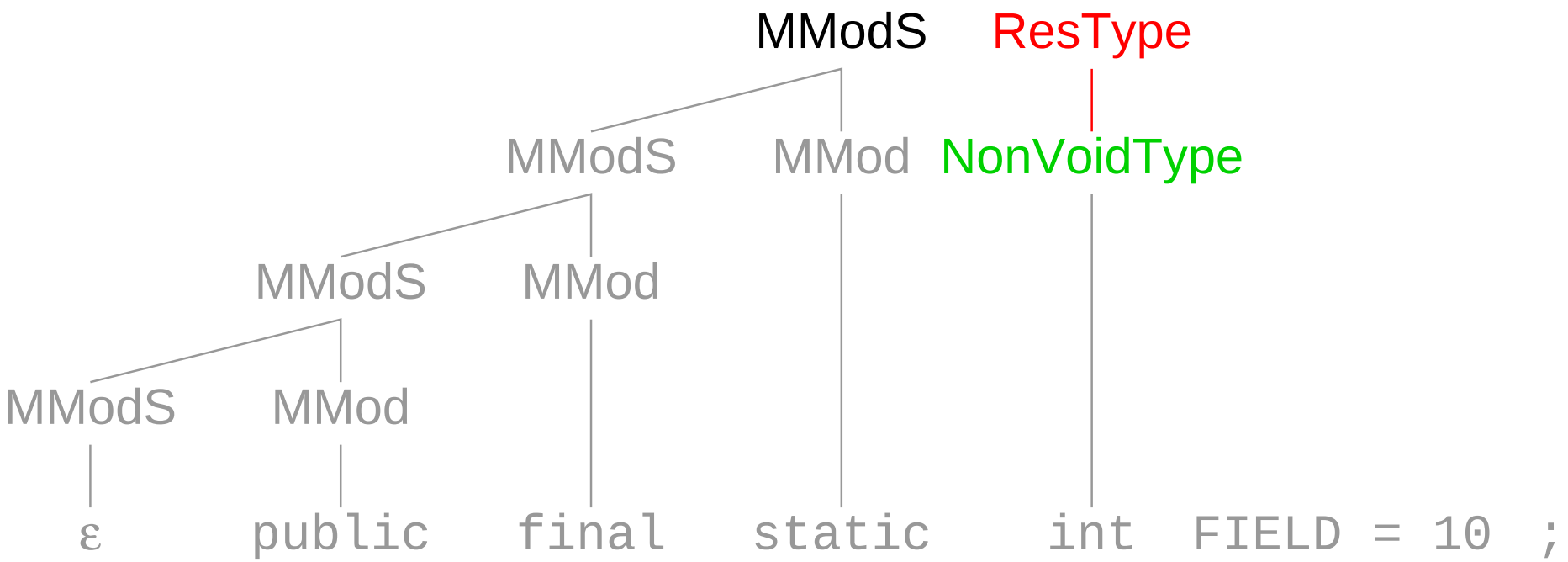


$$\$ < \mathit{MModS} > \texttt{int} \parallel \texttt{FIELD} = \texttt{10;}\$$$

$$\models_{\text{reduce}} \$ < \mathit{MModS} > < \mathit{NonVoidType} > \parallel \texttt{FIELD} = \texttt{10;}\$$$

Nondeterminism

Example

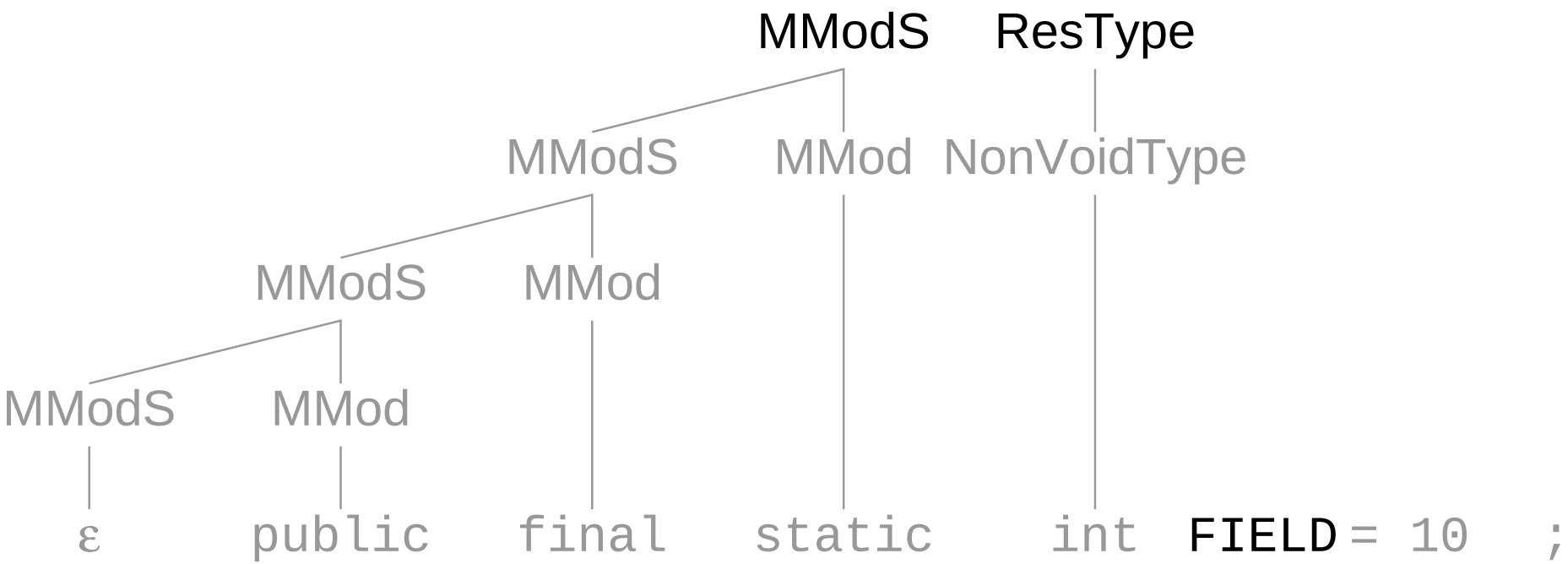


$$\$ < \mathit{MModS} > < \mathit{NonVoidType} > \| \texttt{FIELD} = \texttt{10;\$}$$

$$\models_{\text{reduce}} \$ < \mathit{MModS} > < \mathit{ResType} > \| \texttt{FIELD} = \texttt{10;\$}$$

Nondeterminism

Example

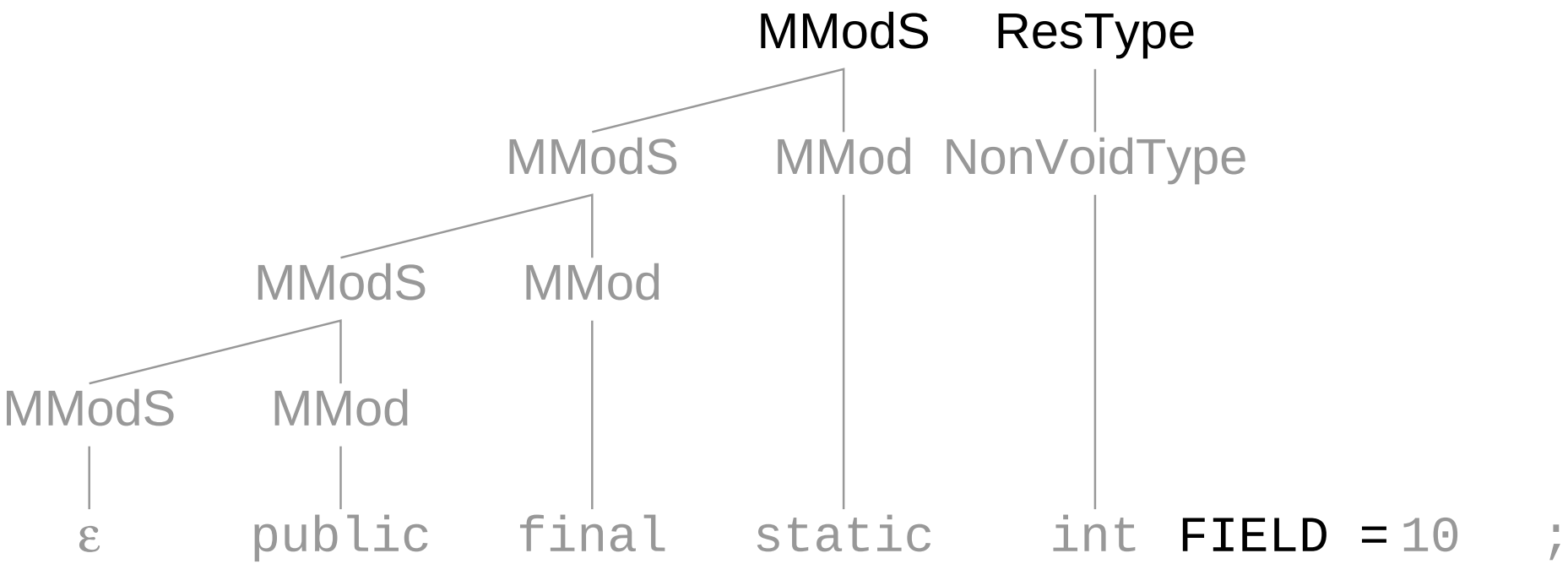


$$\$ < MModS >< ResType > \| \texttt{FIELD} = 10;\$$$

$$\models_{\text{shift}} \$ < MModS >< ResType > \texttt{FIELD} \| = 10;\$$$

Nondeterminism

Example

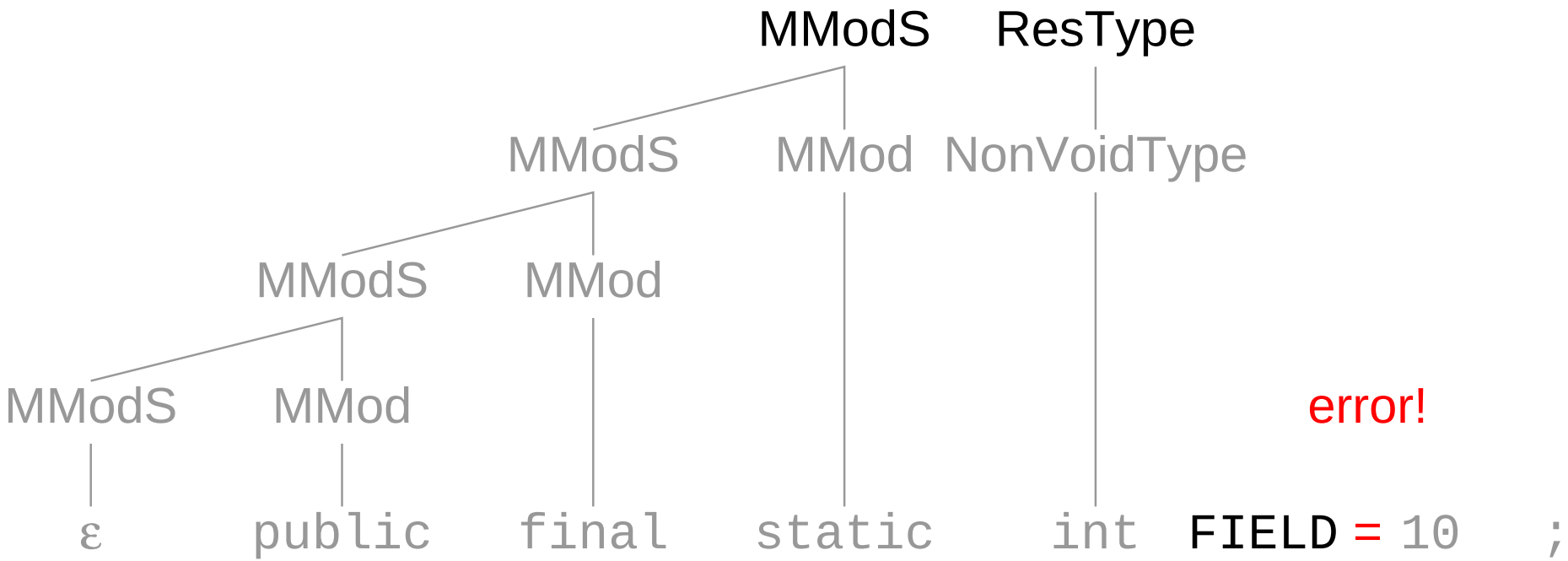


$$\$ < MModS >< ResType > \texttt{FIELD} \parallel = \ 10;\$$$

$$\models_{\text{shift}} \$ < MModS >< ResType > \texttt{FIELD} = \parallel 10;\$$$

Nondeterminism

Example



$$\$ < MModS > < ResType > \texttt{FIELD} = \| \texttt{10;} \$$$

Nondeterminism

Example

FModS
|
ε public final static int FIELD = 10 ;

$ε || public final static int FIELD = 10;$
$\vdash_{\text{reduce}}$ $<FModS> || public final static int FIELD = 10;$

Why Determinism Matters

+ Performance
+ Safety
− Generality

LR(k) Parsing

Knuth [1965]

- complete stack information
- k symbols of lookahead

Example

Choice between reductions $< FModS > \to \varepsilon$ and $< MModS > \to \varepsilon$:

```
public static final ... final static int l =
public static final ... final static int l (
```

- Infinite lookahead needed for better generality

LR(k) Parsing

Knuth [1965]

- complete stack information
- k symbols of lookahead

Example

Choice between reductions $< FModS > \rightarrow \varepsilon$ and $< MModS > \rightarrow \varepsilon$:

```
public static final ... final static int l =
public static final ... final static int l (
```

- Infinite lookahead needed for better generality

LR(k) Parsing

Knuth [1965]

- complete stack information
- k symbols of lookahead

Example

Choice between reductions $< FModS > \rightarrow \varepsilon$ and $< MModS > \rightarrow \varepsilon$:

```
||public static final ... final static int l =
||public static final ... final static int l (
```

- Infinite lookahead needed for better generality

Syntactic Predicates

Parr and Quong [1995]

- User defined lookahead pattern

Example

```
((MMod)* ResType MDecl)?
```

- Error-prone
- Exponential time complexity

Syntactic Predicates

Parr and Quong [1995]

- User defined lookahead pattern

Example

```
((MMod)* ResType MDecl)?
```

- Error-prone
- Exponential time complexity

Syntactic Predicates

Parr and Quong [1995]

- User defined lookahead pattern

Example

```
((MMod)* ResType MDecl)?
```

- Error-prone
- Exponential time complexity

Syntactic Predicates

Parr and Quong [1995]

- User defined lookahead pattern

Example

```
((MMod)* ResType MDecl)?
```

- Error-prone
- Exponential time complexity

LR-Regular Parsing

Baker [1981], Bermudez and Schimpf [1990], Farré and Fortes Gálvez [2001]

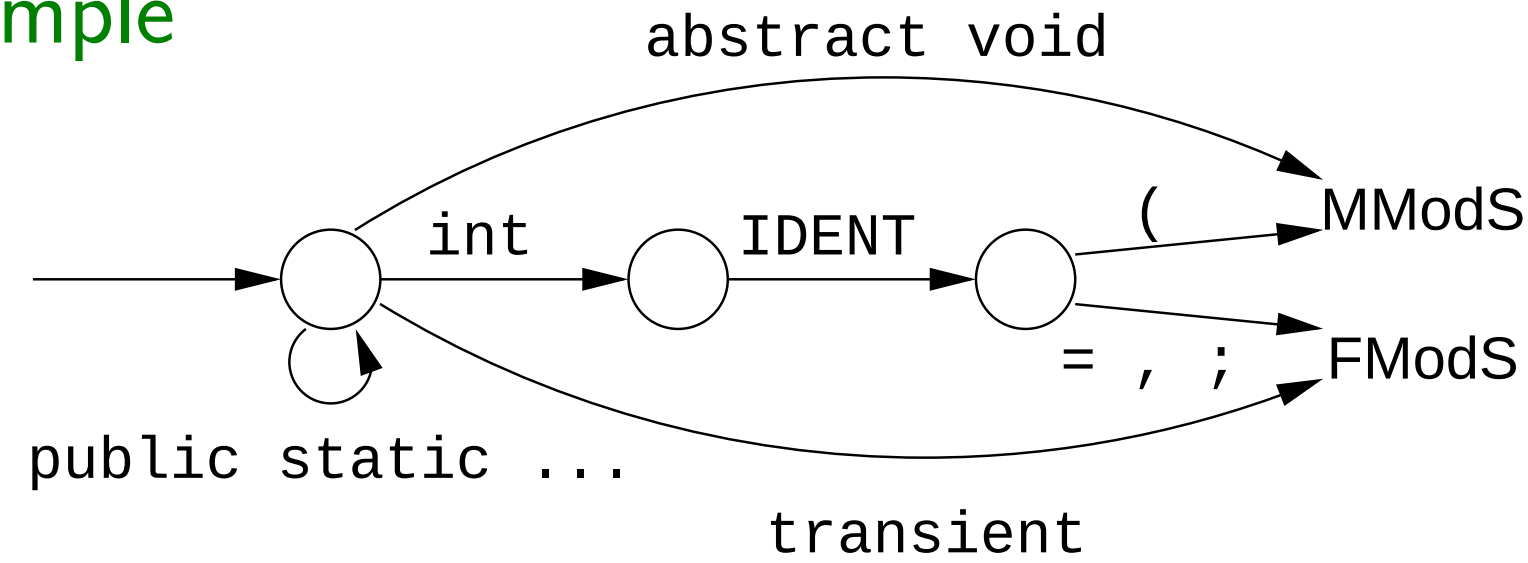
- Finite State lookahead discrimination
- $\mathcal{O}(n^2)$ time complexity
- Only considers terminal symbols

LR-Regular Parsing

Baker [1981], Bermudez and Schimpf [1990], Farré and Fortes Gálvez [2001]

- Finite State lookahead discrimination

Example



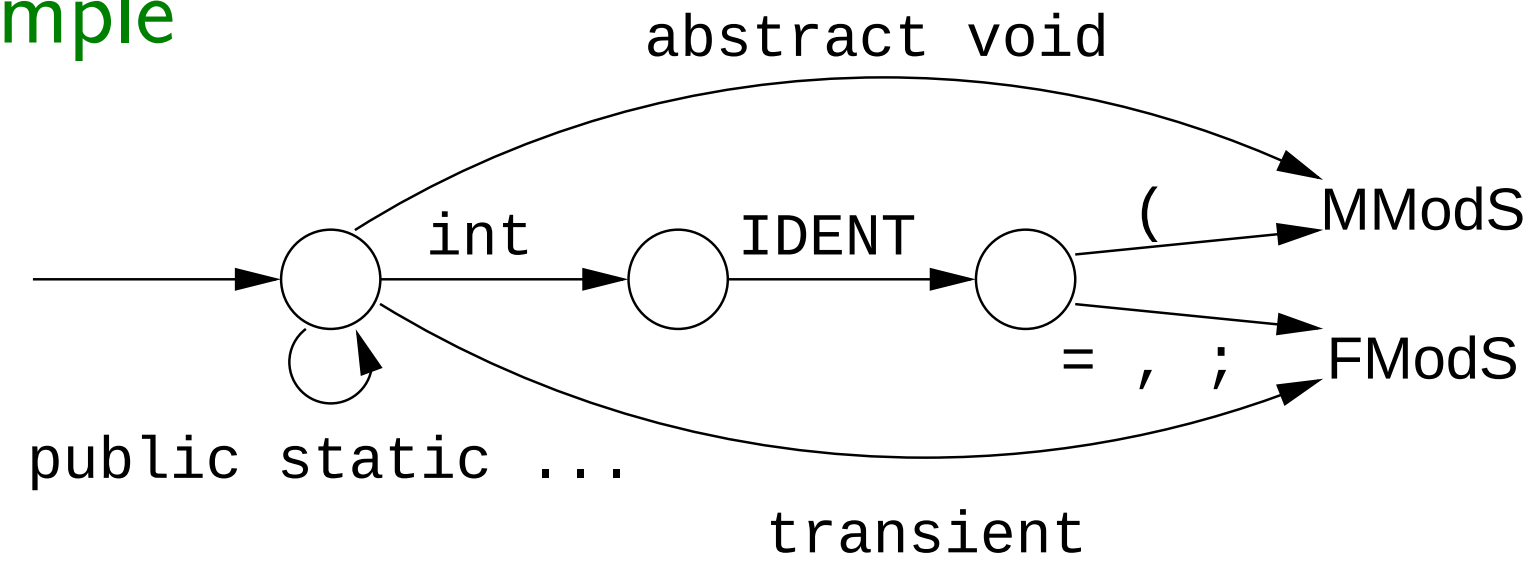
- $\mathcal{O}(n^2)$ time complexity
- Only considers terminal symbols

LR-Regular Parsing

Baker [1981], Bermudez and Schimpf [1990], Farré and Fortes Gálvez [2001]

- Finite State lookahead discrimination

Example



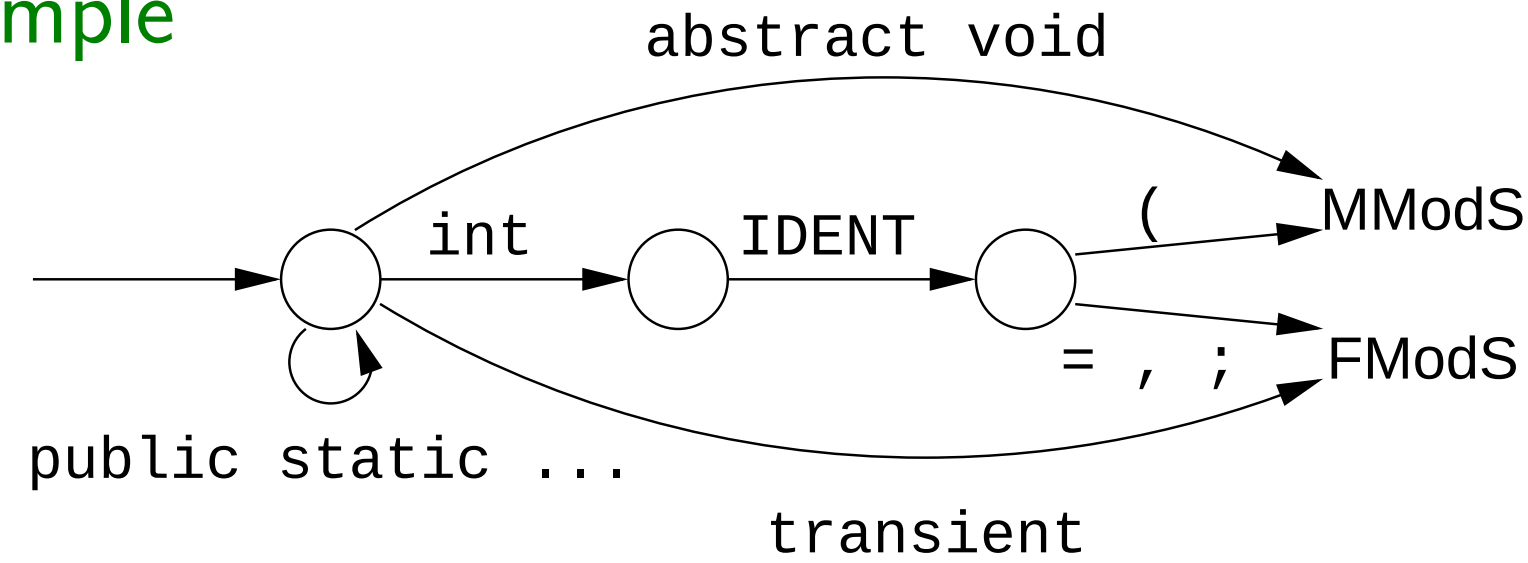
- $\mathcal{O}(n^2)$ time complexity
- Only considers terminal symbols

LR-Regular Parsing

Baker [1981], Bermudez and Schimpf [1990], Farré and Fortes Gálvez [2001]

- Finite State lookahead discrimination

Example



- $\mathcal{O}(n^2)$ time complexity
- Only considers terminal symbols

Parsing a Program Source

- Good generality
- Determinism
- $\mathcal{O}(n)$ time complexity

Parsing a Program Source

- Good generality
- Determinism
- $\mathcal{O}(n)$ time complexity

Noncanonical parsing

Noncanonical Parsing

Colmerauer [1970], Szymanski and Williams [1976], Tai [1979]

Example

```
public final static int FIELD = 10 ;
```

$ ‖ public final static int FIELD = 10;$

Noncanonical Parsing

Colmerauer [1970], Szymanski and Williams [1976], Tai [1979]

Example

```
public final static int FIELD = 10 ;
```

$$\$ \parallel \texttt{public final static int FIELD = 10;\$}$$

$$\models_{\text{shift}} \$\ \texttt{public} \parallel \texttt{final static int FIELD = 10;\$}$$

Noncanonical Parsing

Colmerauer [1970], Szymanski and Williams [1976], Tai [1979]

Example

```
public final static int FIELD = 10 ;
```

$$\$\texttt{public} \parallel \texttt{final static int FIELD = 10;}\$$$

$$\vdash_{\text{shift}} \$\texttt{public final} \parallel \texttt{static int FIELD = 10;}\$$$

Noncanonical Parsing

Colmerauer [1970], Szymanski and Williams [1976], Tai [1979]

Example

```
public final static int FIELD = 10 ;
```

$$\$\texttt{public final} \parallel \texttt{static int FIELD = 10;}\$$$

$$\vdash_{\text{shift}} \$\texttt{public final static} \parallel \texttt{int FIELD = 10;}\$$$

Noncanonical Parsing

Colmerauer [1970], Szymanski and Williams [1976], Tai [1979]

Example

```
public final static int FIELD = 10 ;
```

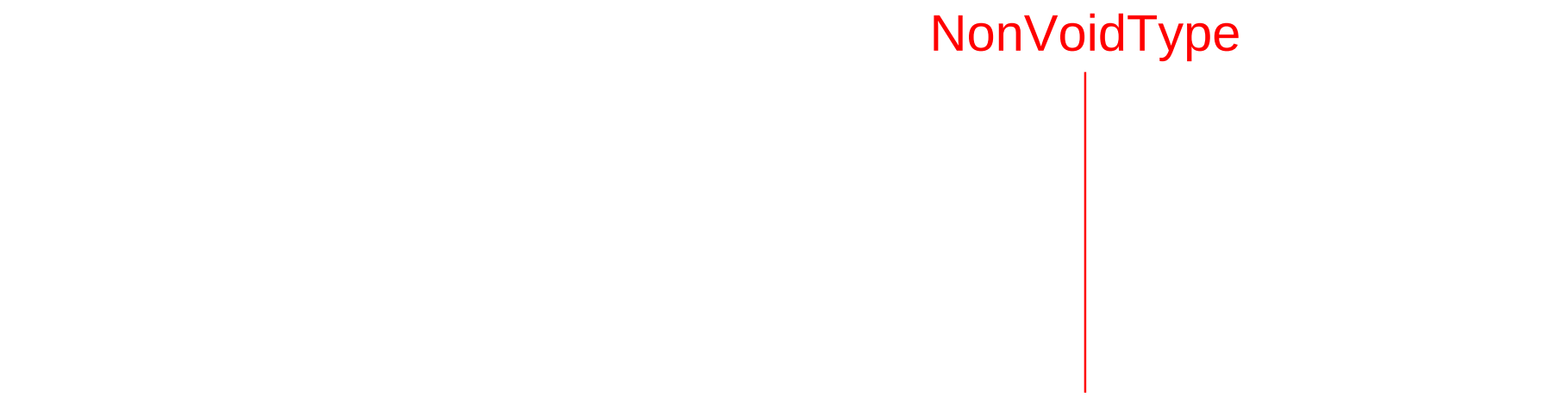
$$\$\texttt{public final static} \parallel \texttt{int FIELD = 10;}\$$$

$$\vdash_{\text{shift}} \$\texttt{public final static int} \parallel \texttt{FIELD = 10;}\$$$

Noncanonical Parsing

Colmerauer [1970], Szymanski and Williams [1976], Tai [1979]

Example



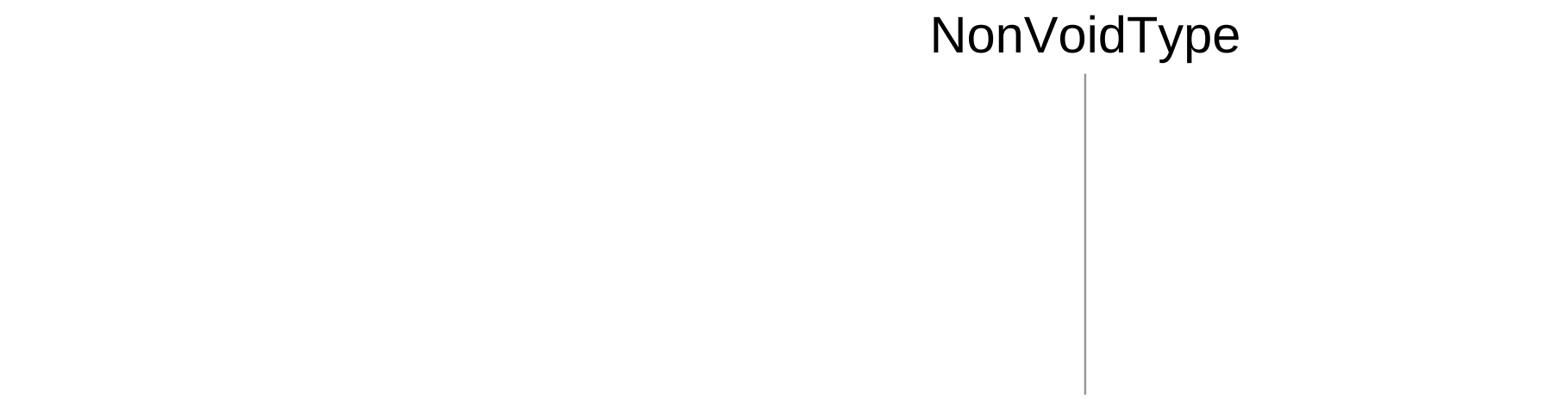
$$\$\texttt{public final static int} \parallel \texttt{FIELD = 10;}\$$$

$$\models_{\text{reduce}} \$\texttt{public final static} \parallel \langle \textit{NonVoidType} \rangle\ \texttt{FIELD = 10;}\$$$

Noncanonical Parsing

Colmerauer [1970], Szymanski and Williams [1976], Tai [1979]

Example



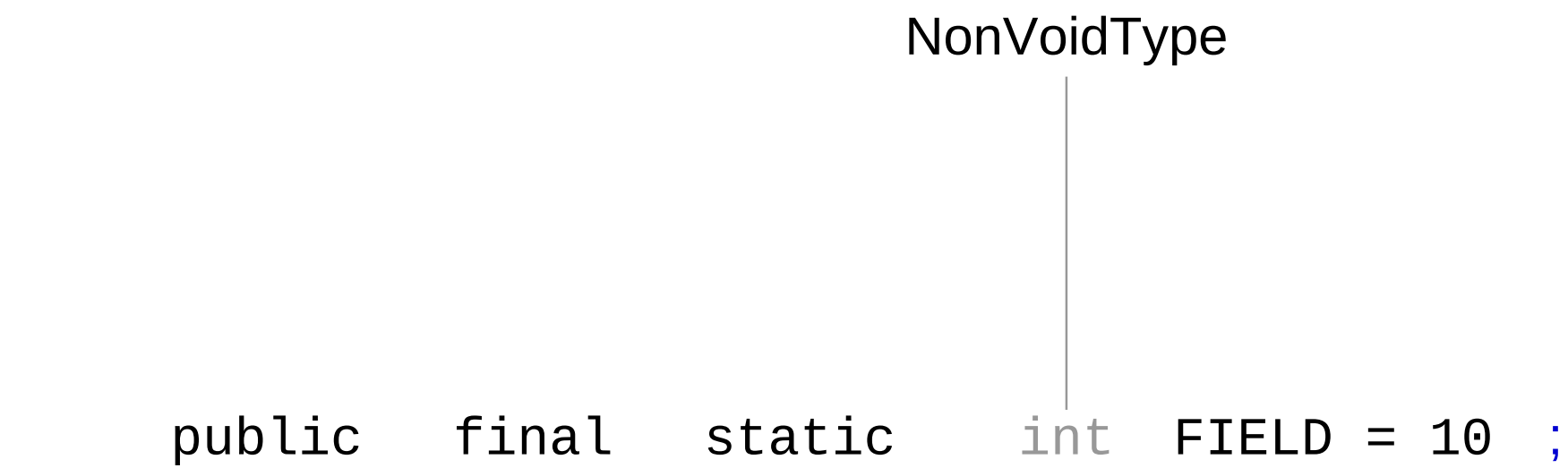
$$\$\texttt{public final static} \parallel \langle \textit{NonVoidType} \rangle\ \texttt{FIELD = 10;\$}$$

$$\vdash_{\text{shift}} \$\texttt{public final static}\ \langle \textit{NonVoidType} \rangle \parallel \texttt{FIELD = 10;\$}$$

Noncanonical Parsing

Colmerauer [1970], Szymanski and Williams [1976], Tai [1979]

Example



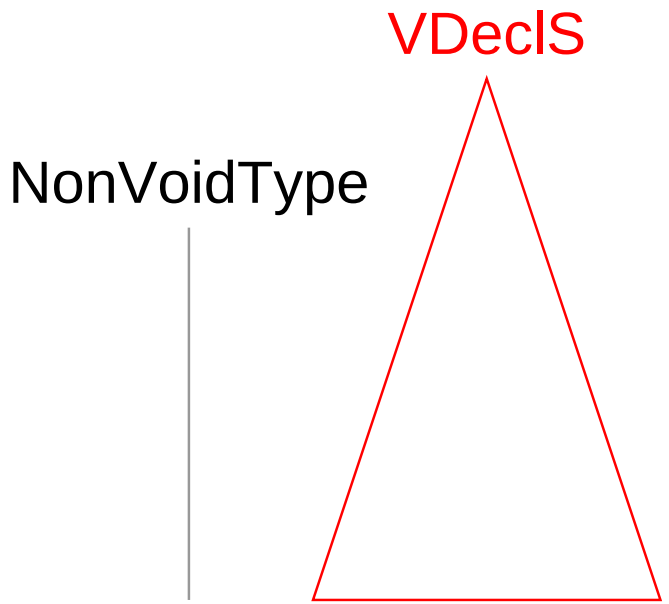
$$\$\texttt{public final static} \langle \textit{NonVoidType} \rangle \parallel \texttt{FIELD = 10 ;\$}$$

$$\vdash_{\text{shift}} \$\texttt{public final static} \langle \textit{NonVoidType} \rangle \texttt{ FIELD = 10} \parallel \texttt{;\$}$$

Noncanonical Parsing

Colmerauer [1970], Szymanski and Williams [1976], Tai [1979]

Example




```
public  final  static  int  FIELD = 10  ;
```

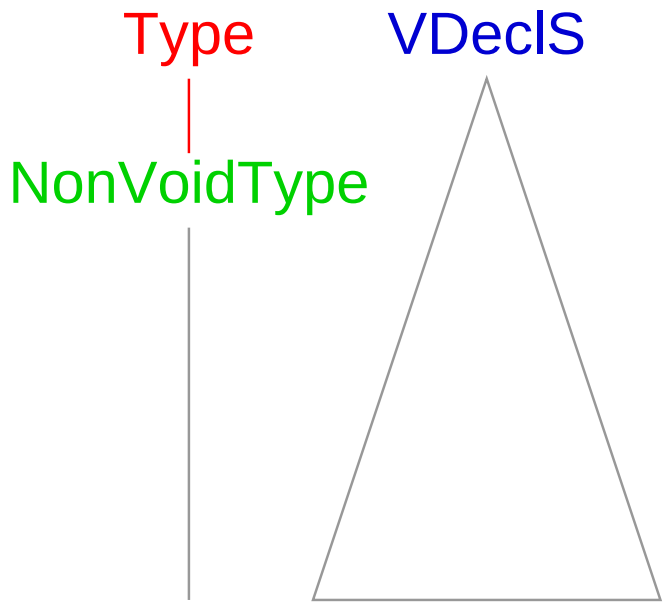
$$\$\texttt{public final static}\ \langle \textit{NonVoidType} \rangle\ \texttt{FIELD = 10}\ \|\ \texttt{;\$}$$

$$\models_{\text{reduce}}\ \$\texttt{public final static}\ \langle \textit{NonVoidType} \rangle\ \|\ \langle \textit{VDeclS} \rangle\ \texttt{;\$}$$

Noncanonical Parsing

Colmerauer [1970], Szymanski and Williams [1976], Tai [1979]

Example




```
public  final  static  int  FIELD = 10  ;
```

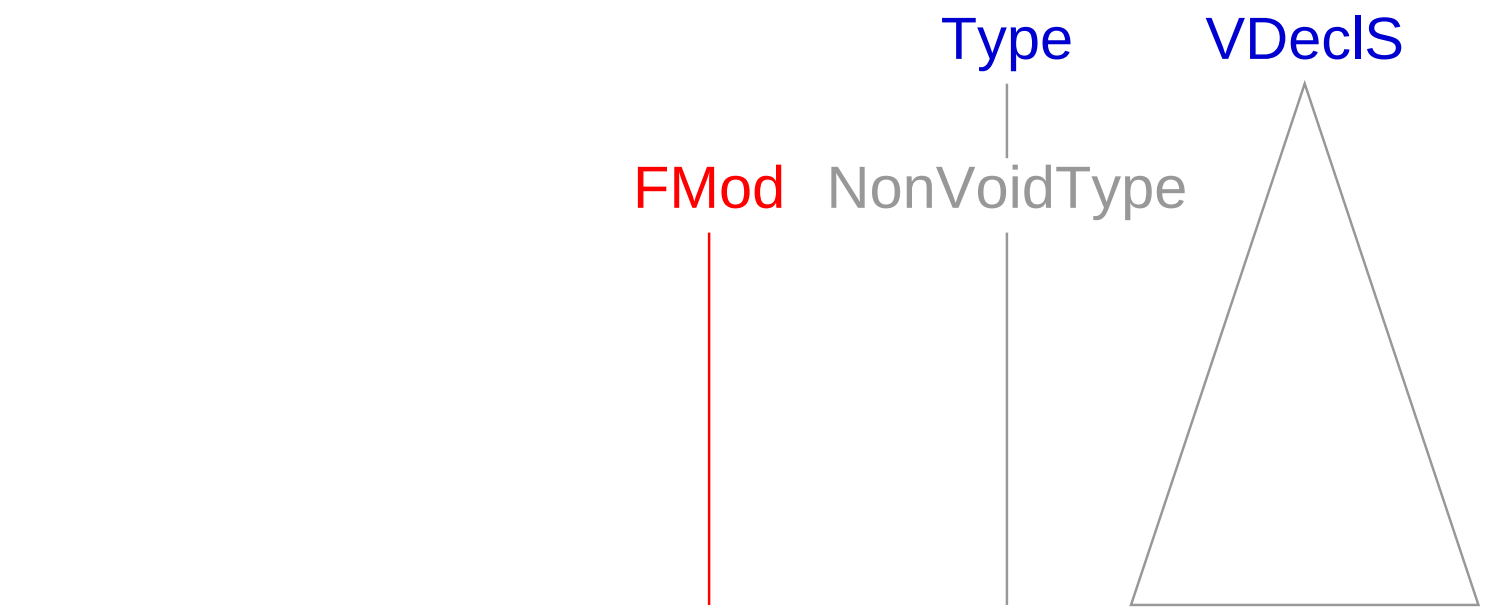
$$\$\texttt{public final static} \langle \textit{NonVoidType} \rangle \parallel \langle \textit{VDeclS} \rangle \ \texttt{;\$}$$

$$\models_{\text{reduce}} \ \$\texttt{public final static} \parallel \langle \textit{Type} \rangle \ \langle \textit{VDeclS} \rangle \ \texttt{;\$}$$

Noncanonical Parsing

Colmerauer [1970], Szymanski and Williams [1976], Tai [1979]

Example



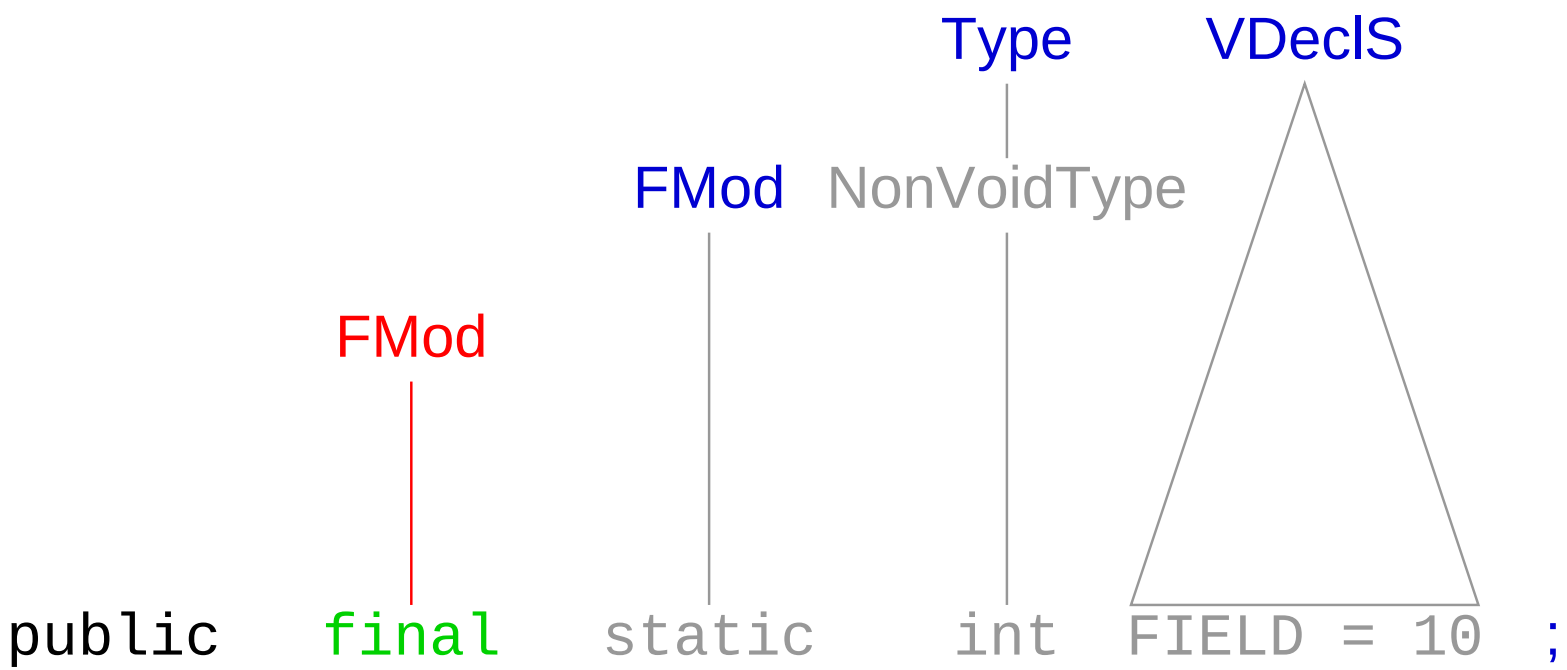
$$\$\texttt{public final static} \parallel \langle \mathit{Type} \rangle \langle \mathit{VDeclS} \rangle \ \texttt{;}\$$$

$$\models_{\text{reduce}} \ \$\texttt{public final} \parallel \langle \mathit{FMod} \rangle \ \langle \mathit{Type} \rangle \langle \mathit{VDeclS} \rangle \ \texttt{;}\$$$

Noncanonical Parsing

Colmerauer [1970], Szymanski and Williams [1976], Tai [1979]

Example



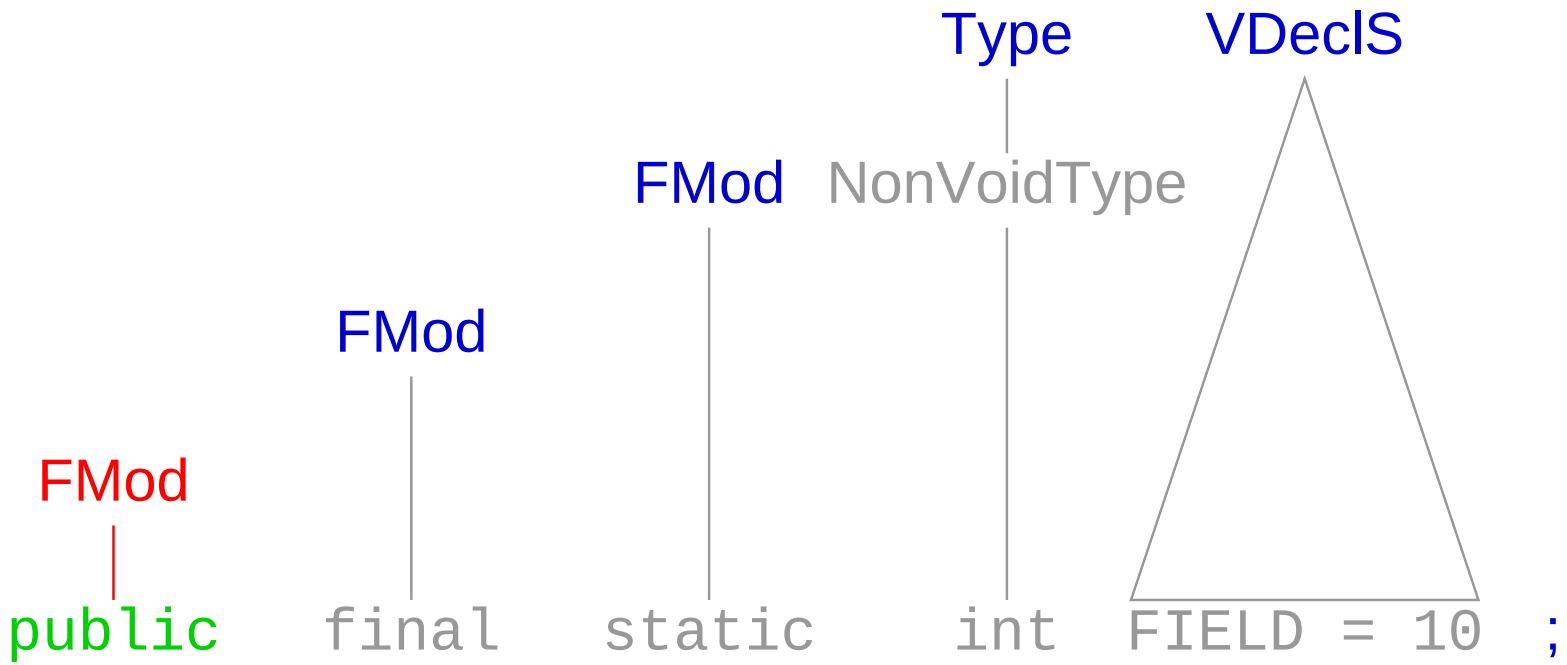
$$\$\texttt{public final} \parallel \langle \mathit{FMod} \rangle \langle \mathit{Type} \rangle \langle \mathit{VDeclS} \rangle \ \texttt{;\$}$$

$$\models_{\text{reduce}} \ \$\texttt{public} \parallel \langle \mathit{FMod} \rangle \ \langle \mathit{FMod} \rangle \langle \mathit{Type} \rangle \langle \mathit{VDeclS} \rangle \ \texttt{;\$}$$

Noncanonical Parsing

Colmerauer [1970], Szymanski and Williams [1976], Tai [1979]

Example



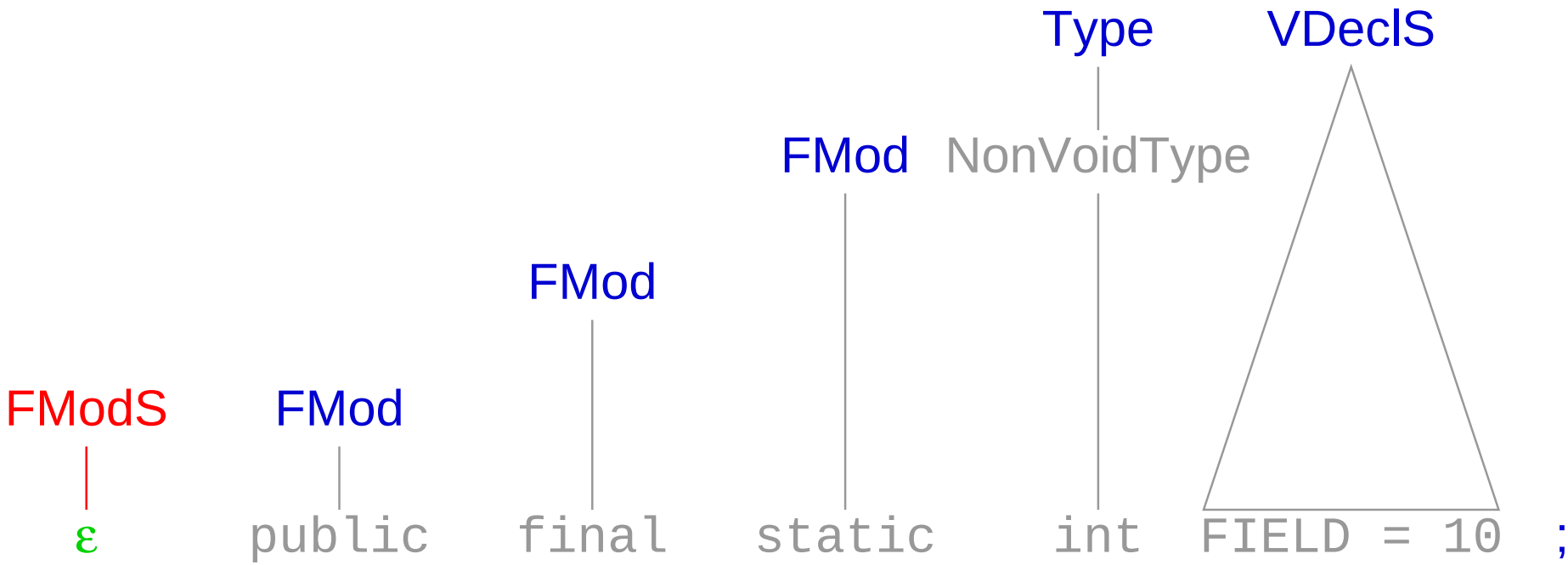
$ public ‖ <*FMod*><*FMod*><*Type*><*VDeclS*> ;$

$\vdash_{\text{reduce}}$ $ ‖ <*FMod*> <*FMod*><*FMod*><*Type*><*VDeclS*> ;$

Noncanonical Parsing

Colmerauer [1970], Szymanski and Williams [1976], Tai [1979]

Example



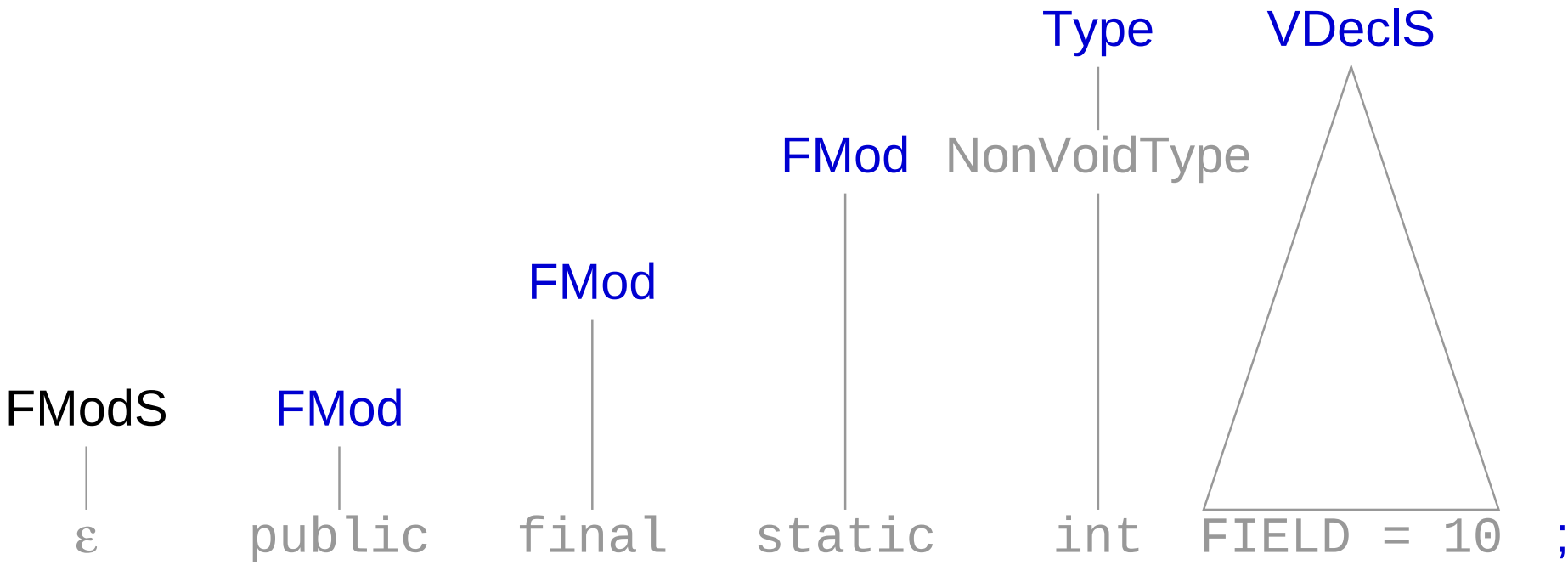
$$\$\ \varepsilon \parallel \langle FMod \rangle \langle FMod \rangle \langle FMod \rangle \langle Type \rangle \langle VDeclS \rangle\ ;\$$$

$$\models_{\text{reduce}} \$ \parallel \langle FModS \rangle\ \langle FMod \rangle \langle FMod \rangle \langle FMod \rangle \langle Type \rangle \langle VDeclS$$

Noncanonical Parsing

Colmerauer [1970], Szymanski and Williams [1976], Tai [1979]

Example



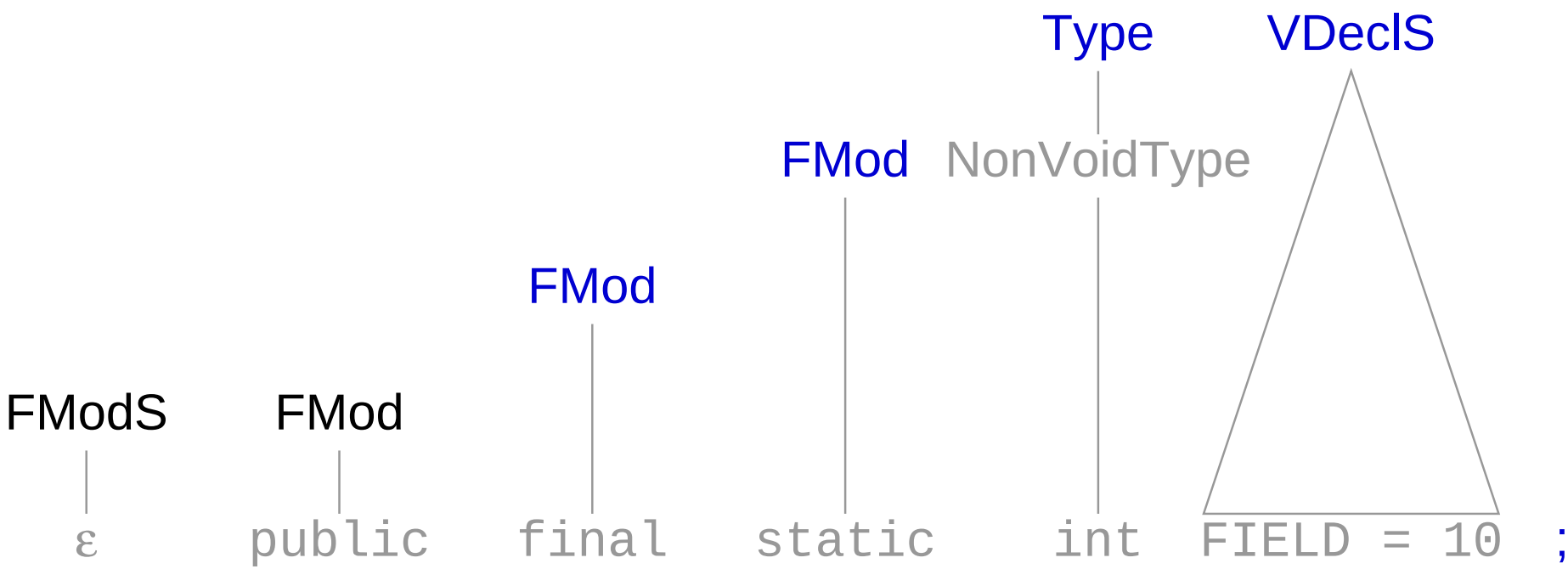
$$\$ \parallel \langle FModS \rangle \quad \langle FMod \rangle \langle FMod \rangle \langle FMod \rangle \langle Type \rangle \langle VDeclS \rangle$$

$$\vdash_{\text{shift}} \$ \ \langle FModS \rangle \parallel \langle FMod \rangle \langle FMod \rangle \langle FMod \rangle \langle Type \rangle \langle VDeclS \rangle$$

Noncanonical Parsing

Colmerauer [1970], Szymanski and Williams [1976], Tai [1979]

Example



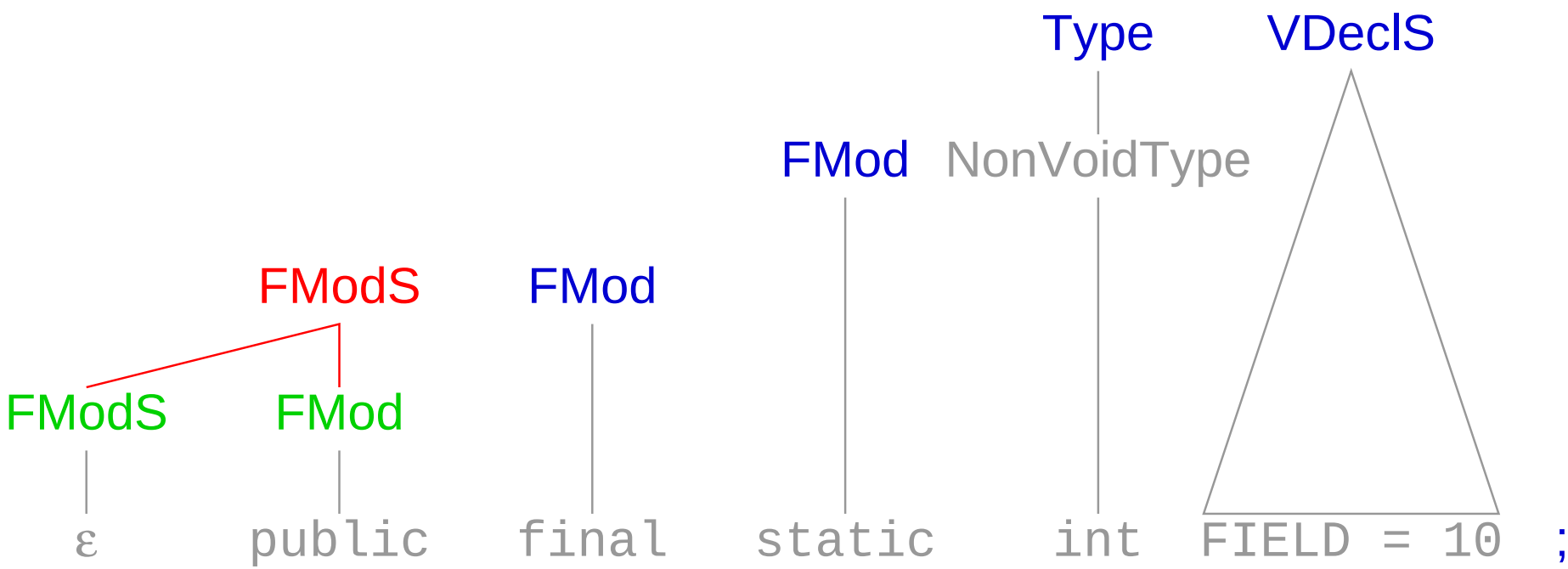
$$\$ < FModS > \ \| < FMod > \quad < FMod >< FMod >< Type >< VDecl S$$

$$\models_{\text{shift}} \$ < FModS > \quad < FMod > \ \| < FMod >< FMod >< Type >< VDecl S$$

Noncanonical Parsing

Colmerauer [1970], Szymanski and Williams [1976], Tai [1979]

Example



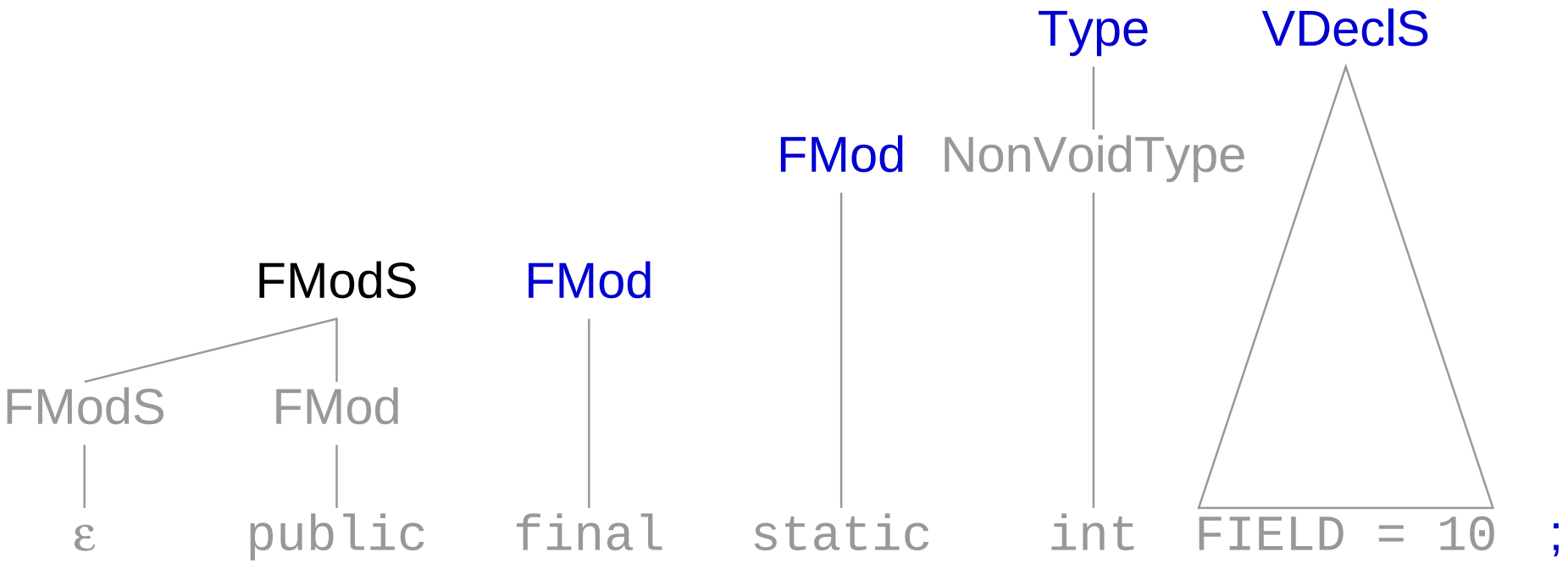
$ $\langle FModS \rangle \langle FMod \rangle \parallel \langle FMod \rangle \langle FMod \rangle \langle Type \rangle \langle VDeclS \rangle$

$\models_{\text{reduce}}$ $ $\parallel \langle FModS \rangle \langle FMod \rangle \langle FMod \rangle \langle Type \rangle \langle VDeclS \rangle$;$

Noncanonical Parsing

Colmerauer [1970], Szymanski and Williams [1976], Tai [1979]

Example



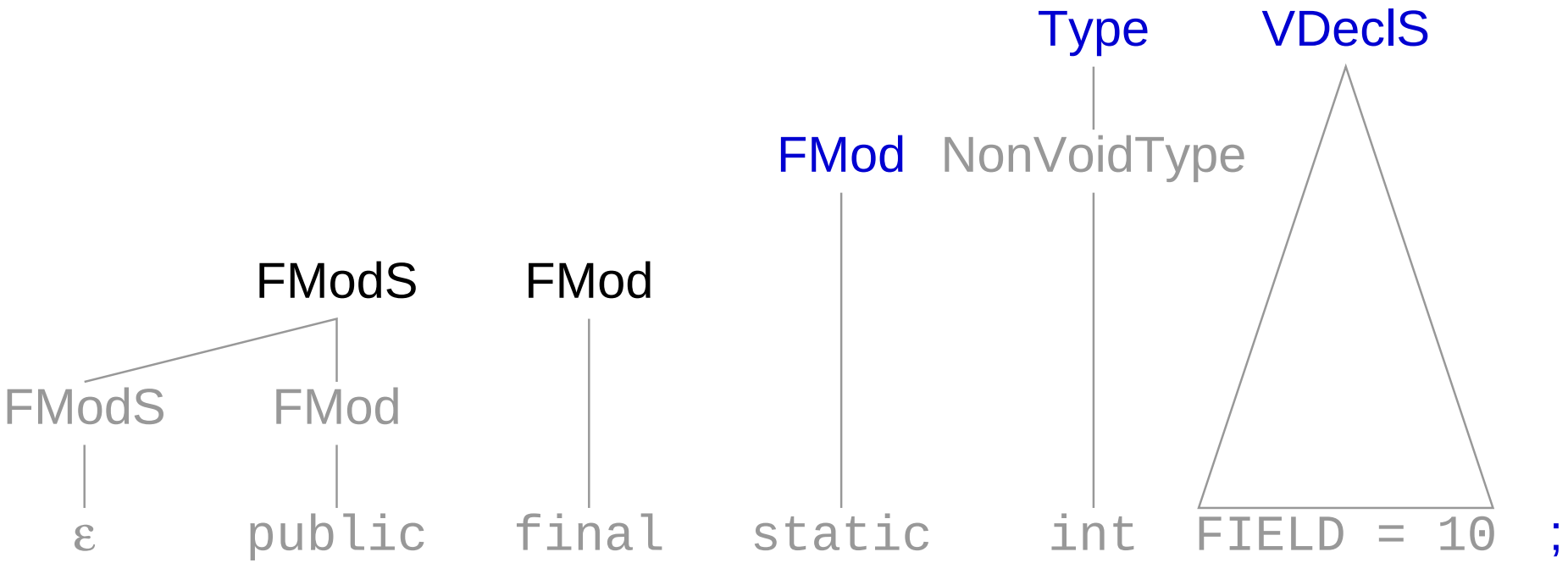
$$\$ \parallel \langle FModS \rangle \quad \langle FMod \rangle \langle FMod \rangle \langle Type \rangle \langle VDeclS \rangle \; ;\$$$

$$\models_{\text{shift}} \$ \; \langle FModS \rangle \parallel \langle FMod \rangle \langle FMod \rangle \langle Type \rangle \langle VDeclS \rangle \; ;\$$$

Noncanonical Parsing

Colmerauer [1970], Szymanski and Williams [1976], Tai [1979]

Example



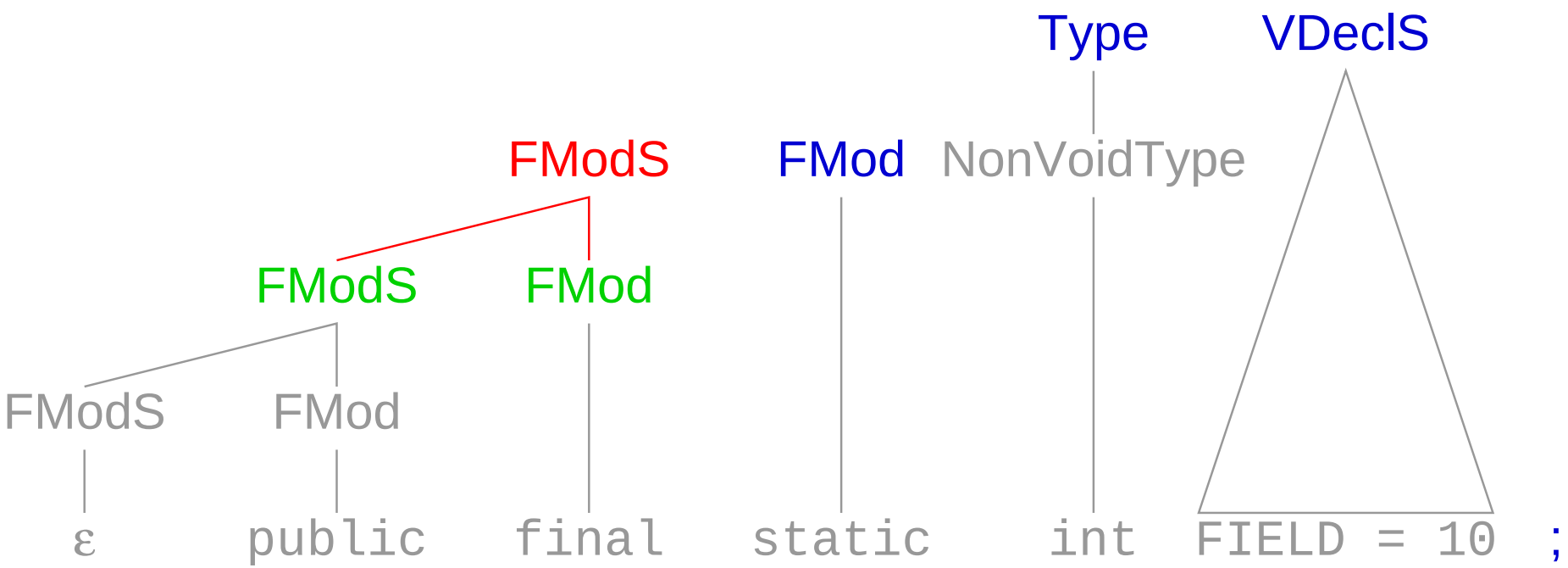
$$\$ < FModS > \ \| < FMod > \quad < FMod >< Type >< VDeclS > \ ;\$$$

$$\models_{\text{shift}} \$ < FModS > \quad < FMod > \ \| < FMod >< Type >< VDeclS > \ ;\$$$

Noncanonical Parsing

Colmerauer [1970], Szymanski and Williams [1976], Tai [1979]

Example



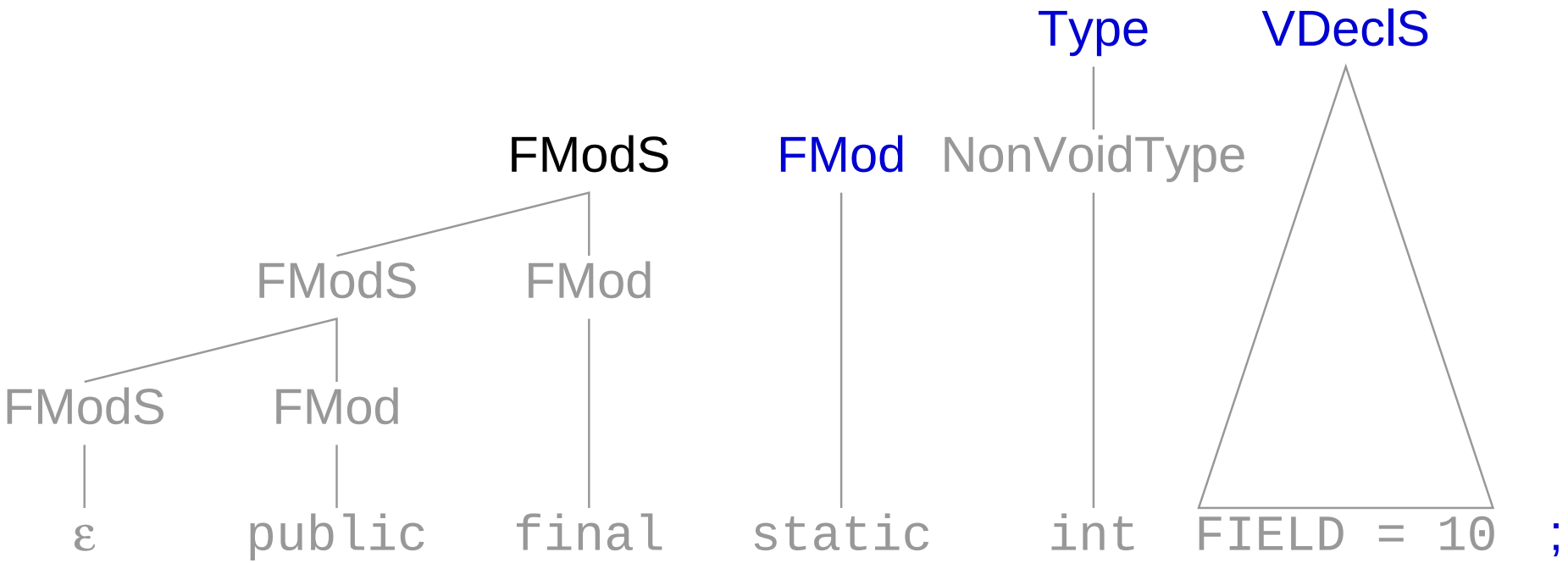
$ $\langle FModS \rangle \langle FMod \rangle \parallel \langle FMod \rangle \langle Type \rangle \langle VDeclS \rangle$;$

$\models_{\text{reduce}}$ $ $\parallel \langle FModS \rangle \langle FMod \rangle \langle Type \rangle \langle VDeclS \rangle$;$

Noncanonical Parsing

Colmerauer [1970], Szymanski and Williams [1976], Tai [1979]

Example



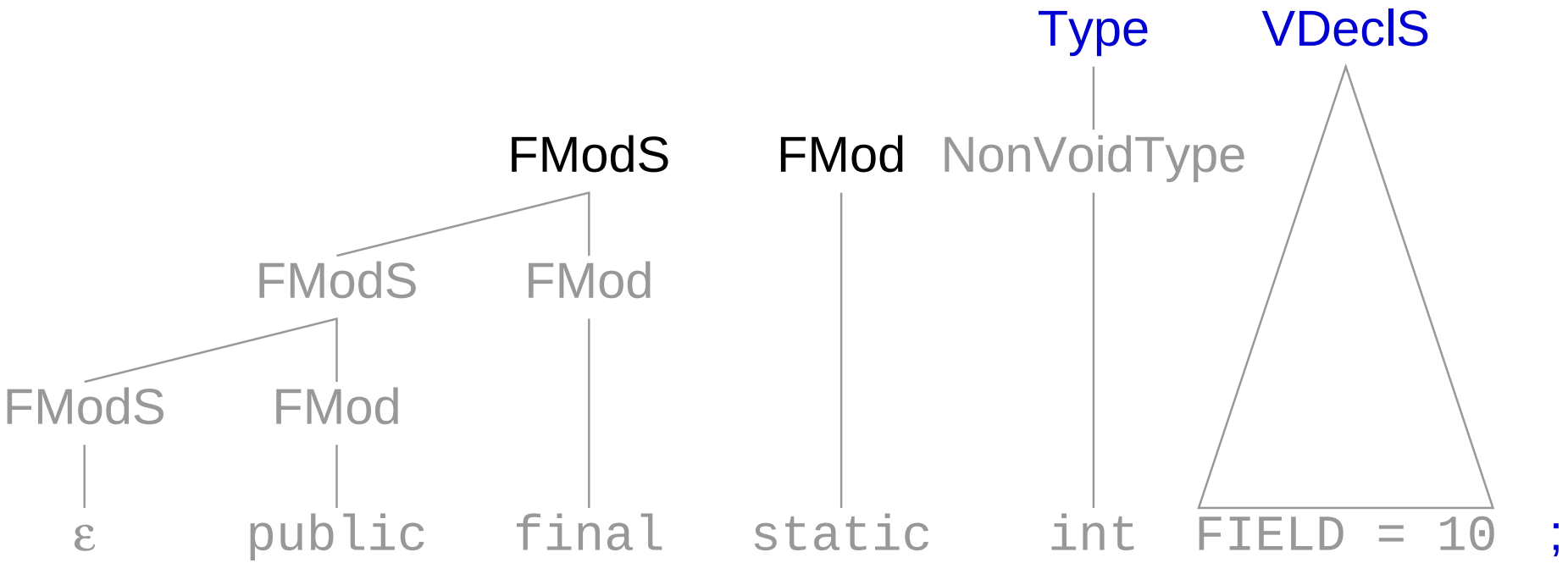
$$\$ \parallel \langle FModS \rangle \quad \langle FMod \rangle \langle Type \rangle \langle VDeclS \rangle \; ;\$$$

$$\models_{\text{shift}} \$ \; \langle FModS \rangle \parallel \langle FMod \rangle \langle Type \rangle \langle VDeclS \rangle \; ;\$$$

Noncanonical Parsing

Colmerauer [1970], Szymanski and Williams [1976], Tai [1979]

Example



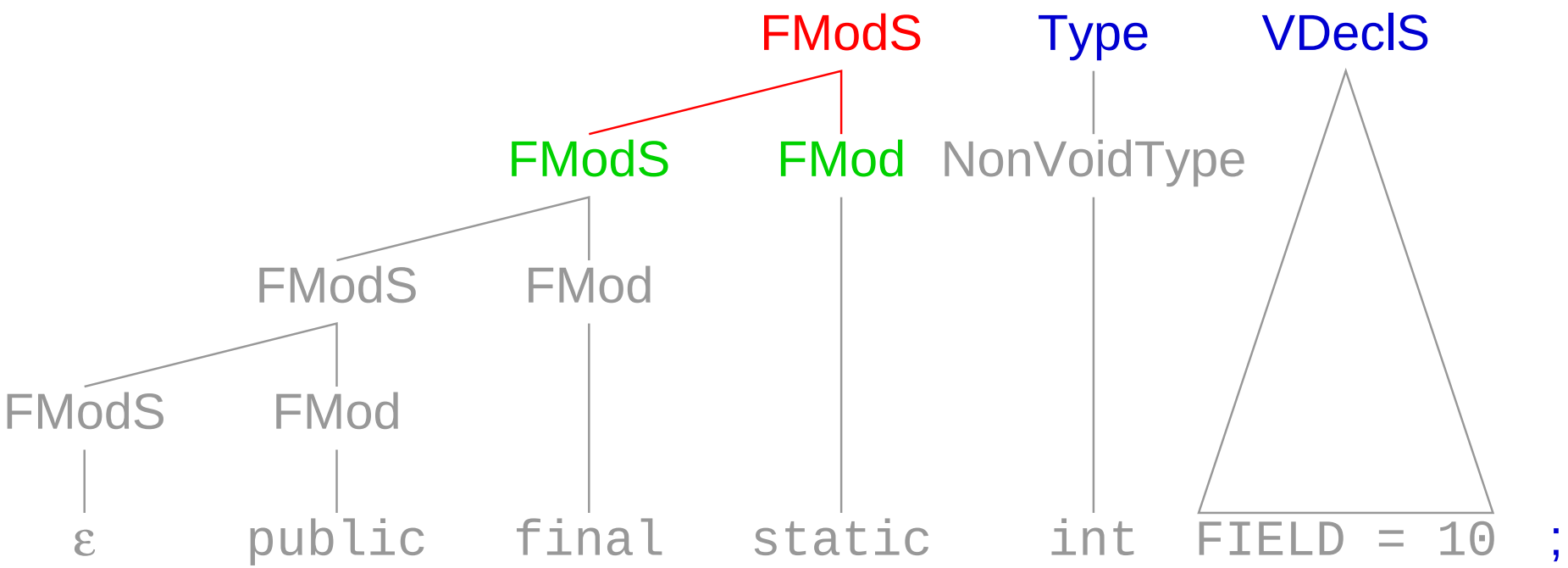
$$\$ < FModS > \ \| < FMod > \ < Type >< VDeclS > \ ;\$$$

$$\models_{\text{shift}} \$ < FModS > \ < FMod > \ \| < Type >< VDeclS > \ ;\$$$

Noncanonical Parsing

Colmerauer [1970], Szymanski and Williams [1976], Tai [1979]

Example



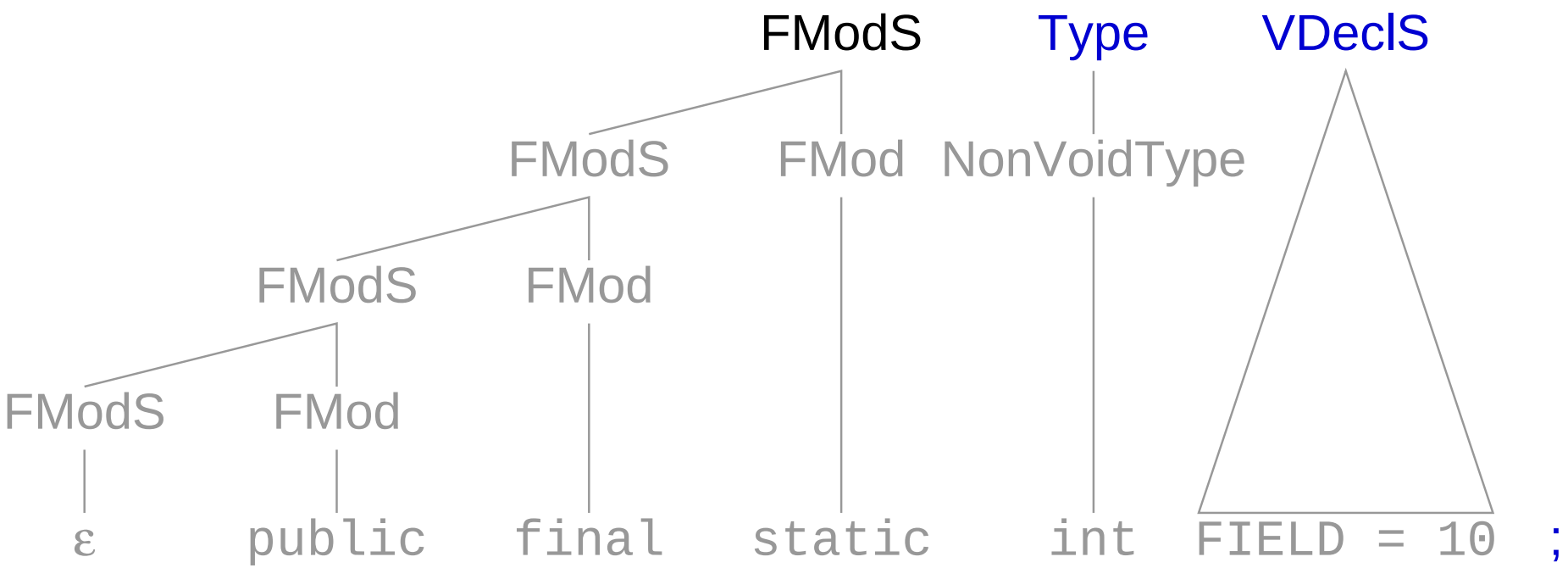
$ $\langle FModS \rangle \langle FMod \rangle \parallel \langle Type \rangle \langle VDeclS \rangle$;$

$\models_{\text{reduce}}$ $ $\parallel \langle FModS \rangle \langle Type \rangle \langle VDeclS \rangle$;$

Noncanonical Parsing

Colmerauer [1970], Szymanski and Williams [1976], Tai [1979]

Example



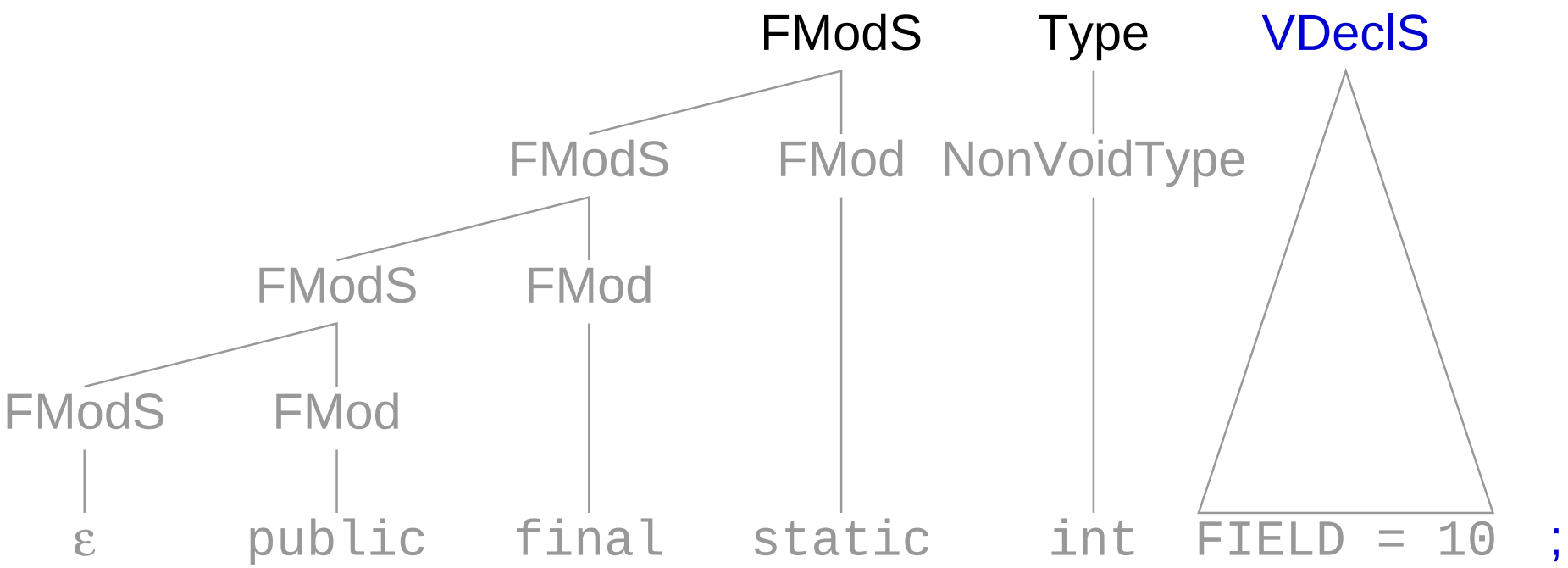
$$\$ \parallel \langle FModS \rangle \quad \langle Type \rangle \langle VDeclS \rangle \; ;\$$$

$$\models_{\text{shift}} \$ \; \langle FModS \rangle \parallel \langle Type \rangle \langle VDeclS \rangle \; ;\$$$

Noncanonical Parsing

Colmerauer [1970], Szymanski and Williams [1976], Tai [1979]

Example



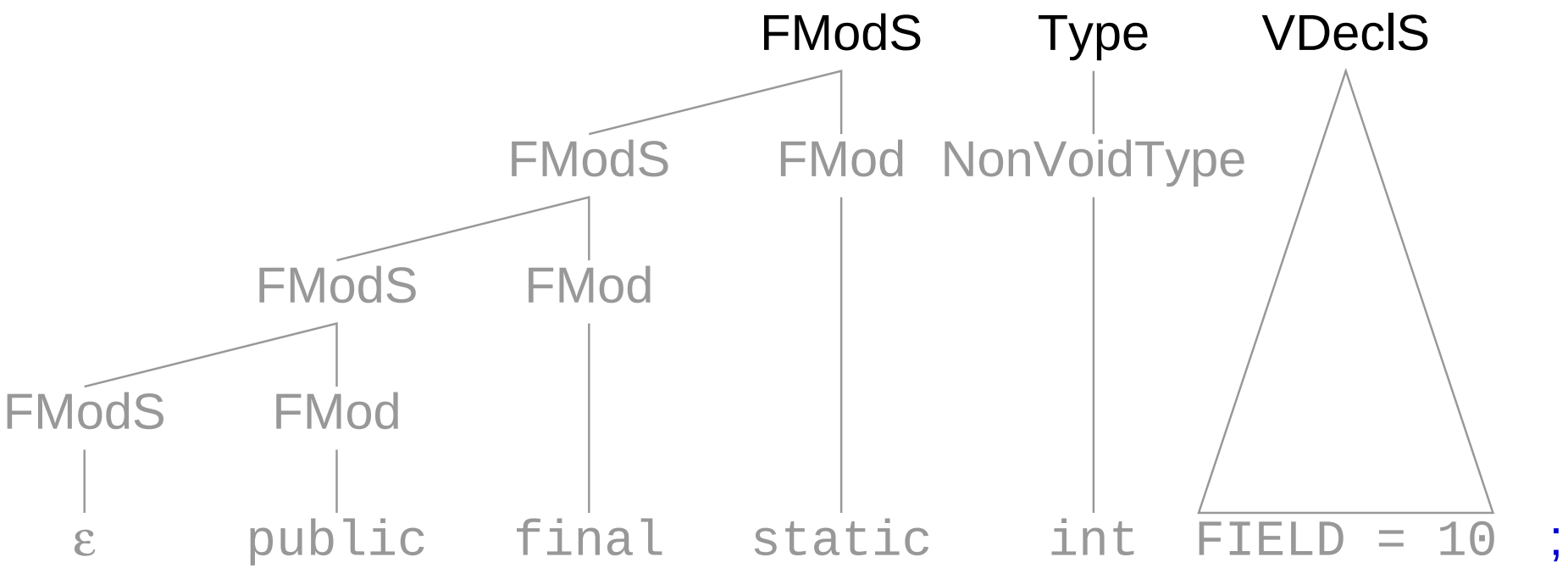
$$\$ < FModS > \ \| < Type > \ < VDeclS > \ ;\$$$

$$\models_{\text{shift}} \$ < FModS > \ < Type > \ \| < VDeclS > \ ;\$$$

Noncanonical Parsing

Colmerauer [1970], Szymanski and Williams [1976], Tai [1979]

Example



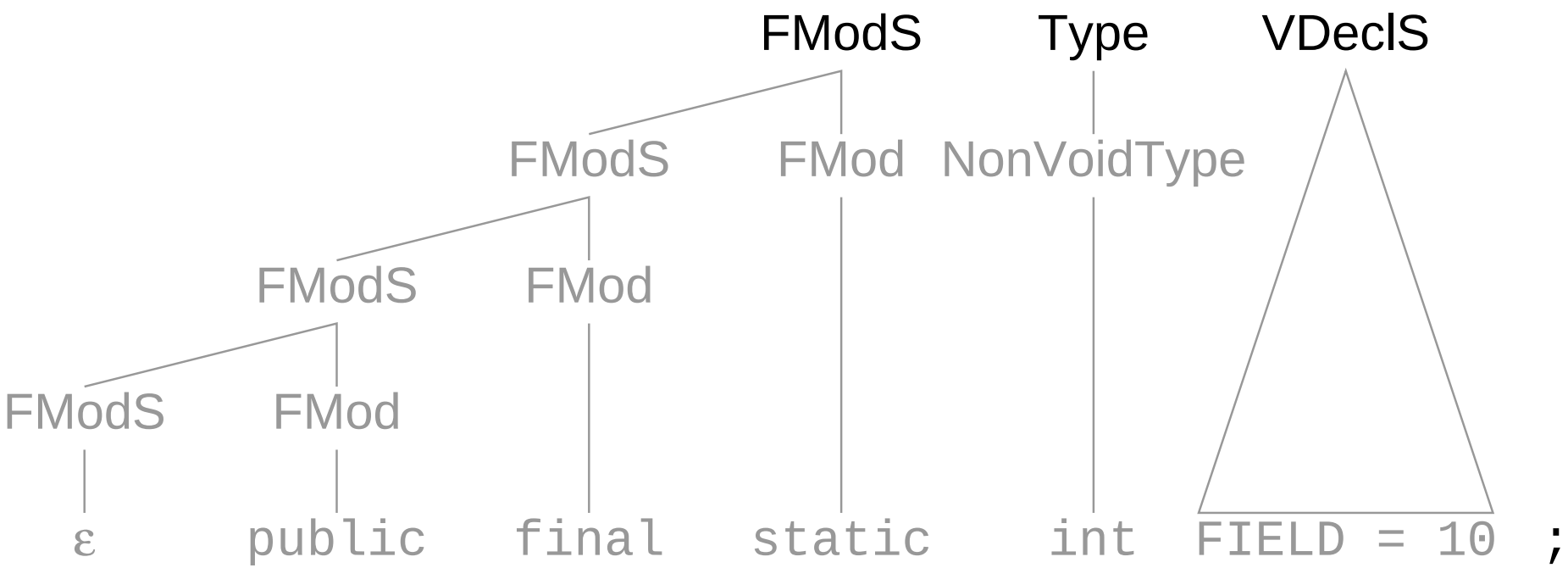
$$\$ < FModS > < Type > \ \| < VDeclS > \ ;\$$$

$$\models_{\text{shift}} \$ < FModS > < Type > \ < VDeclS > \ \| \ ;\$$$

Noncanonical Parsing

Colmerauer [1970], Szymanski and Williams [1976], Tai [1979]

Example



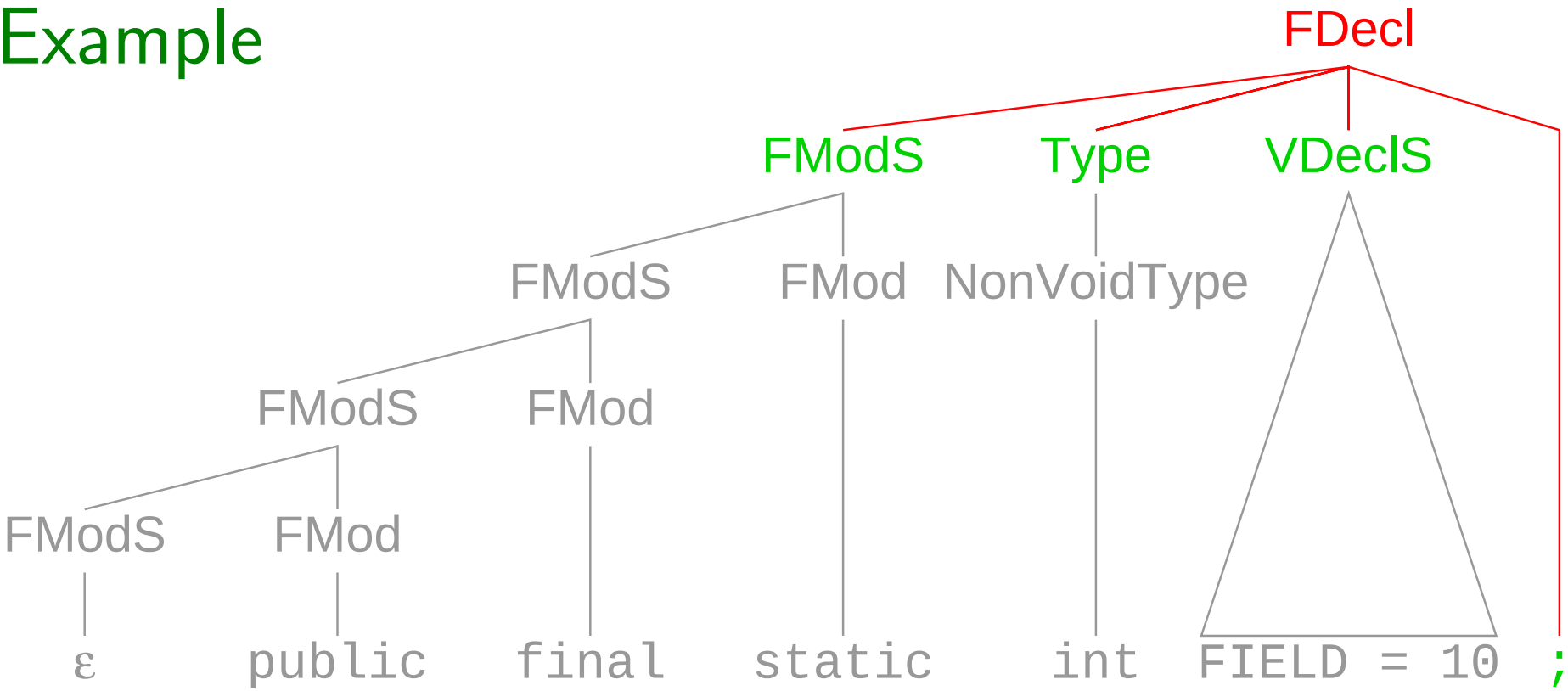
$$\$ < FModS >< Type >< VDeclS > \parallel ; \$$$

$$\vdash_{\text{shift}} \$ < FModS >< Type >< VDeclS > ; \parallel \$$$

Noncanonical Parsing

Colmerauer [1970], Szymanski and Williams [1976], Tai [1979]

Example



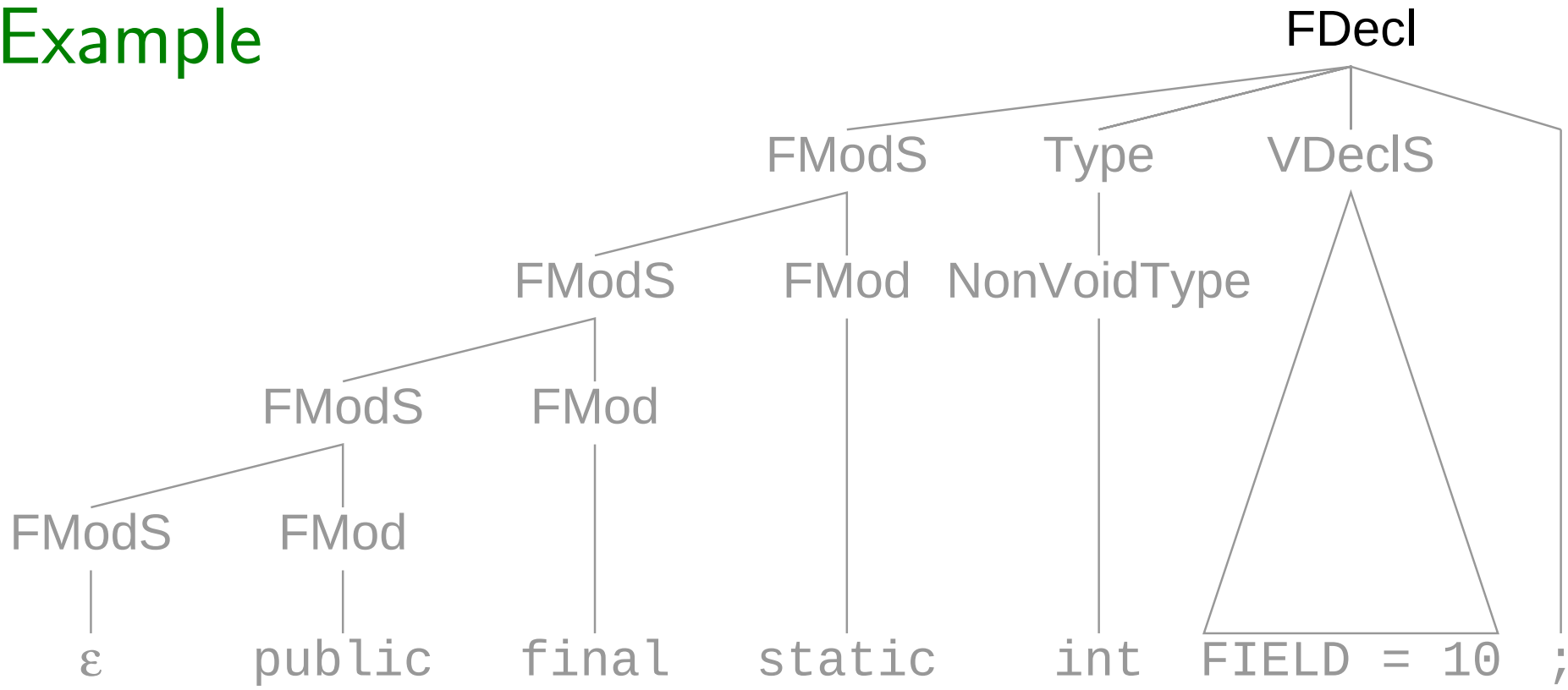
$$\$ < FModS >< Type >< VDeclS > \; ; \parallel \$$$

$$\vdash_{\text{reduce}} \$ \parallel < FDecl > \$$$

Noncanonical Parsing

Colmerauer [1970], Szymanski and Williams [1976], Tai [1979]

Example

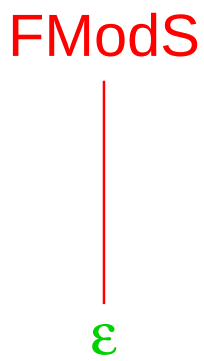


$$\$ \parallel \langle FDecl \rangle \ \$$$

$$\models_{\text{shift}} \$ \ \langle FDecl \rangle \parallel \$$$

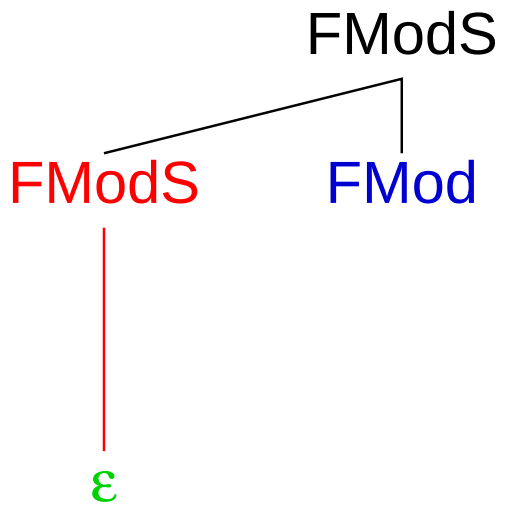
Noncanonical Lookaheads

Example



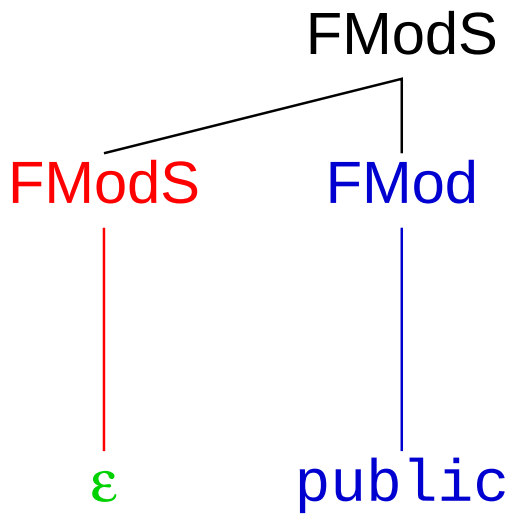
Noncanonical Lookaheads

Example



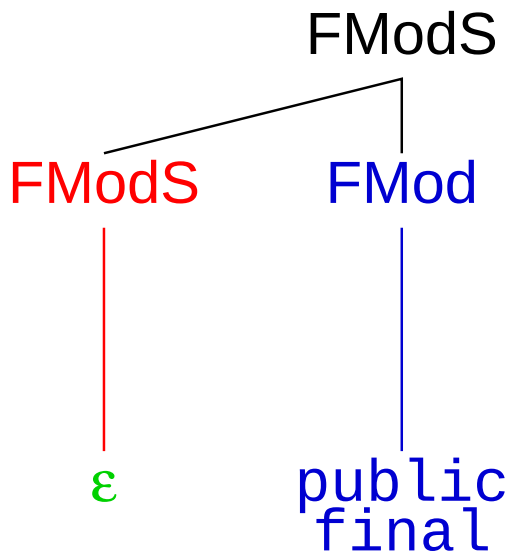
Noncanonical Lookaheads

Example



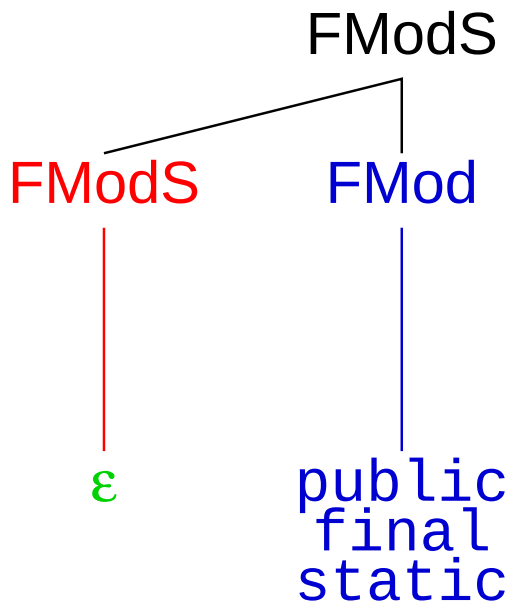
Noncanonical Lookaheads

Example



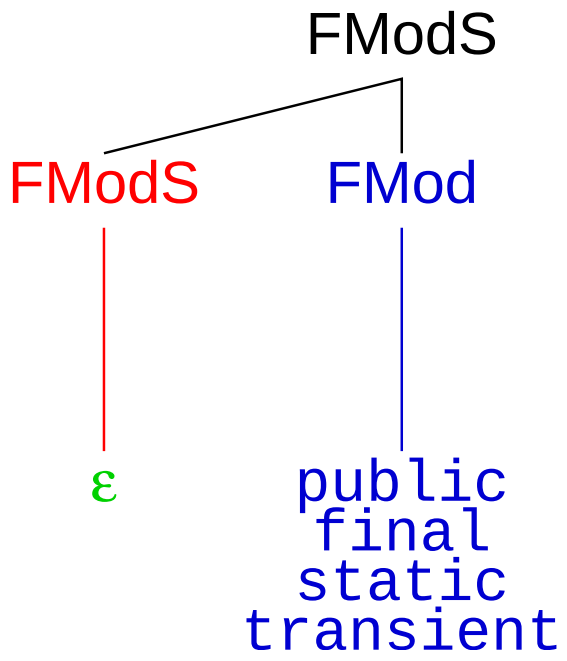
Noncanonical Lookaheads

Example



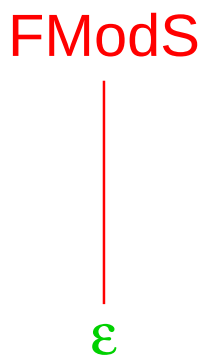
Noncanonical Lookaheads

Example



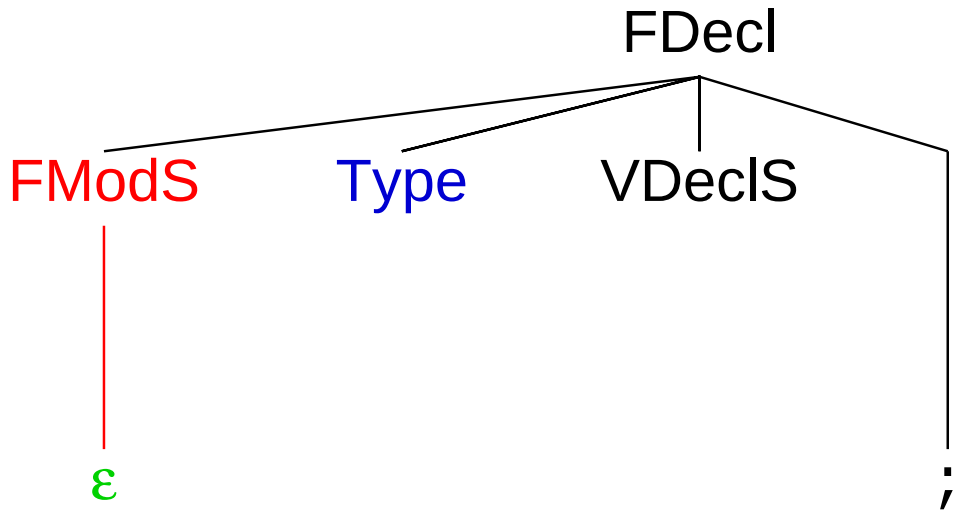
Noncanonical Lookaheads

Example



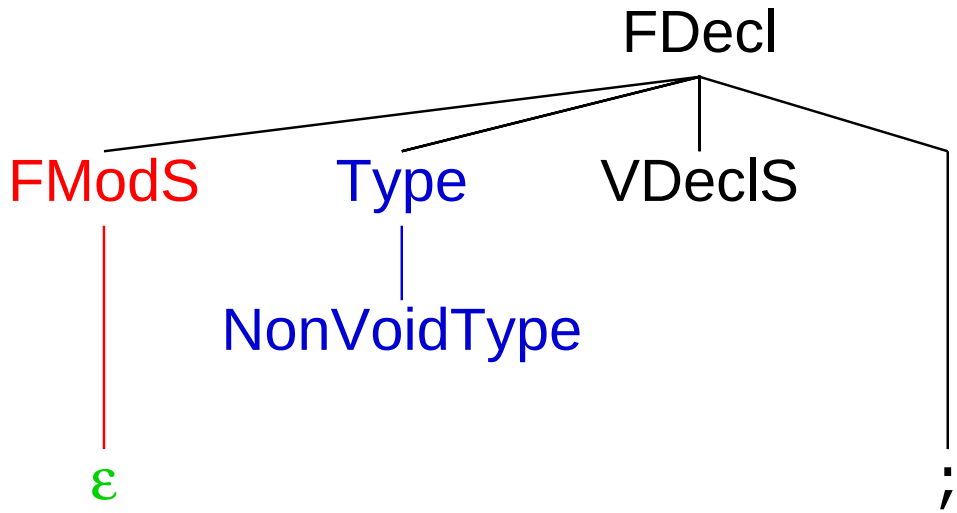
Noncanonical Lookaheads

Example



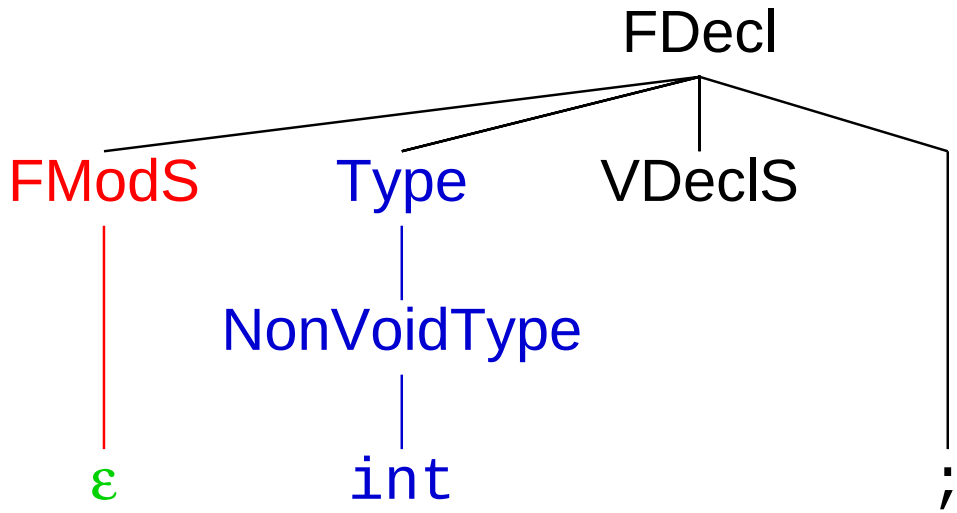
Noncanonical Lookaheads

Example



Noncanonical Lookaheads

Example



Noncanonical Lookaheads

Notation

$<FModS> \longrightarrow \varepsilon$ $\{\texttt{public}, \texttt{final}, \texttt{static}, \texttt{int}, \texttt{transient}, <FMod>, <Type>, <NonVoidType>\}$

Noncanonical Lookaheads

Comparison

$$<FModS> \to \varepsilon \qquad \{\texttt{public}, \texttt{final}, \texttt{static}, \texttt{int}, \texttt{transient}, <FMod>, <Type>, <NonVoidType>\}$$

$$<MModS> \to \varepsilon \qquad \{\texttt{public}, \texttt{final}, \texttt{static}, \texttt{int}, \texttt{abstract}, <MMod>, <ReturnType>, <NonVoidType>\}$$

Noncanonical Lookaheads

Conflicts

$<FModS> \longrightarrow \varepsilon$ $\{\texttt{public}, \texttt{final}, \texttt{static}, \texttt{int}, \texttt{transient}, <FMod>, <Type>, <NonVoidType>\}$

$<MModS> \longrightarrow \varepsilon$ $\{\texttt{public}, \texttt{final}, \texttt{static}, \texttt{int}, \texttt{abstract}, <MMod>, <ReturnType>, <NonVoidType>\}$

Noncanonical Lookaheads

Decisions

$$<FModS> \longrightarrow \varepsilon \qquad \{\texttt{public}, \texttt{final}, \texttt{static}, \texttt{int}, \texttt{transient}, <FMod>, <Type>, <NonVoidType>\}$$

$$<MModS> \longrightarrow \varepsilon \qquad \{\texttt{public}, \texttt{final}, \texttt{static}, \texttt{int}, \texttt{abstract}, <MMod>, <ReturnType>, <NonVoidType>\}$$

Noncanonical LALR(1)

Schmitz [2006]

NLALR(1) Construction

- Easy: computations on the LR(0) automaton
- Fast: relational expressions

Closing Comments

Noncanonical Parsing

Good generality, Determinism, $\mathcal{O}(n)$ time complexity

Improvements

Better generality: no preset lookahead length

T. P. Baker. Extending lookahead for LR parsers. J. Comput. Syst. Sci., 22(2):243–259, 1981. doi: 10.1016/0022-0000(81)90030-1.

M. E. Bermudez and K. M. Schimpf. Practical arbitrary lookahead LR parsing. J. Comput. Syst. Sci., 41(2): 230–250, 1990. doi: 10.1016/0022-0000(90)90037-L.

A. Colmerauer. Total precedence relations. J. ACM, 17 (1):14–30, Jan. 1970. doi: 10.1145/321556.321559.

J. Farré and J. Fortes Gálvez. A bounded-connect construction for LR-regular parsers. In R. Wilhelm, editor, CC'01, volume 2027 of Lecture Notes in Computer Science, pages 244–258. Springer, 2001. URL `http://www.springerlink.com/link.asp?id=e3e8g77kxevkyjfd`.

J. Gosling, B. Joy, and G. Steele. The Java™ Language Specification. Addison-Wesley Longman Publishing Co., Inc., first edition, Aug. 1996. ISBN 0-201-63451-1. URL http://java.sun.com/docs/books/jls/.

D. E. Knuth. On the translation of languages from left to right. Information and Control, 8:607–639, 1965.

T. J. Parr and R. W. Quong. ANTLR: A predicated-LL(k) parser generator. Software, Practice and Experience, 25(7):789–810, 1995. ISSN 0038-0644. URL http://citeseer.nj.nec.com/12770.

S. Schmitz. Noncanonical LALR(1) parsing. In Z. Dang and O. H. Ibarra, editors, DLT'06, volume 4036 of Lecture Notes in Computer Science, pages 95–107. Springer, 2006.

T. G. Szymanski and J. H. Williams. Noncanonical extensions of bottom-up parsing techniques. SIAM J. Comput., 5(2):231–250, June 1976. URL http://locus.siam.org/SICOMP/volume-05/art_0205019.html.

K.-C. Tai. Noncanonical SLR(1) grammars. ACM Trans. Prog. Lang. Syst., 1(2):295–320, 1979. ISSN 0164-0925. doi: 10.1145/357073.357083.