



AspectJ grammar in TXL

Mariano Ceccato

ITC-Irst

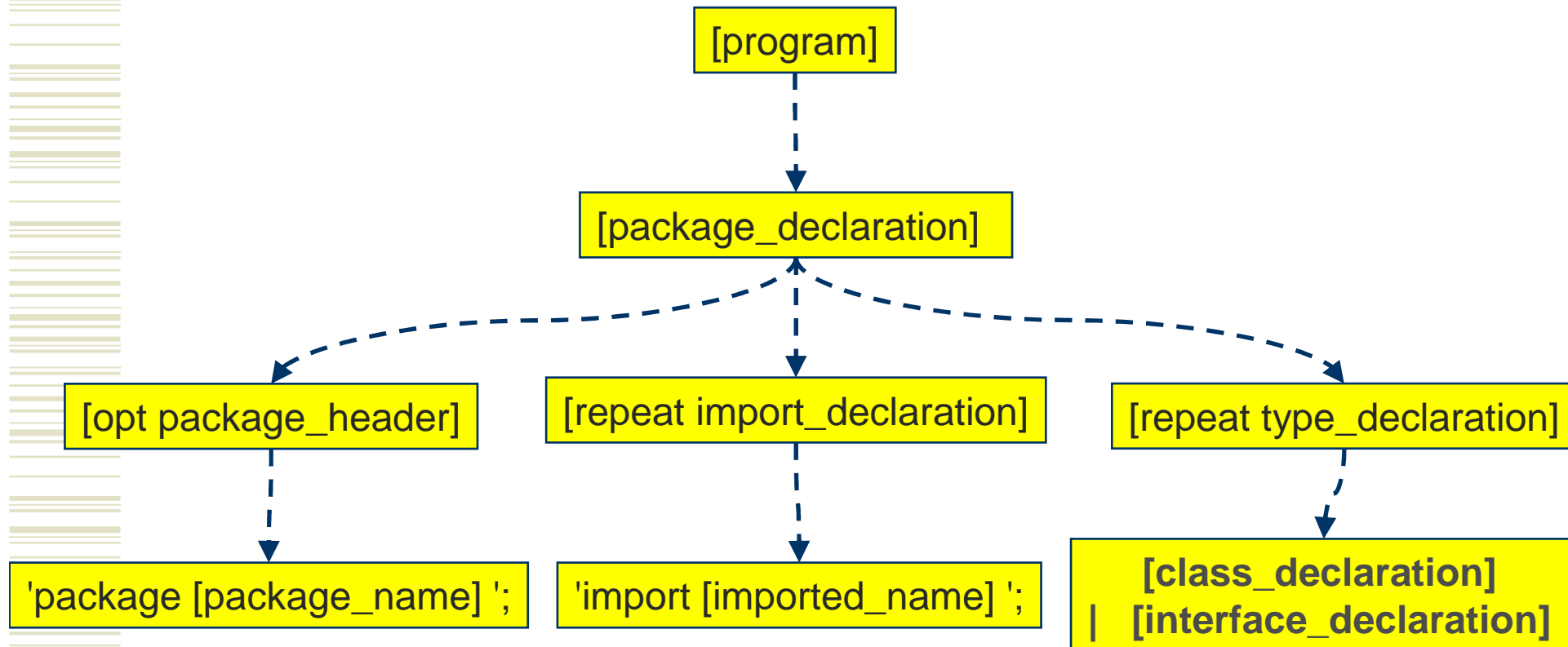
Centro per la ricerca
Scientifica e Tecnologica

ceccato@itc.it

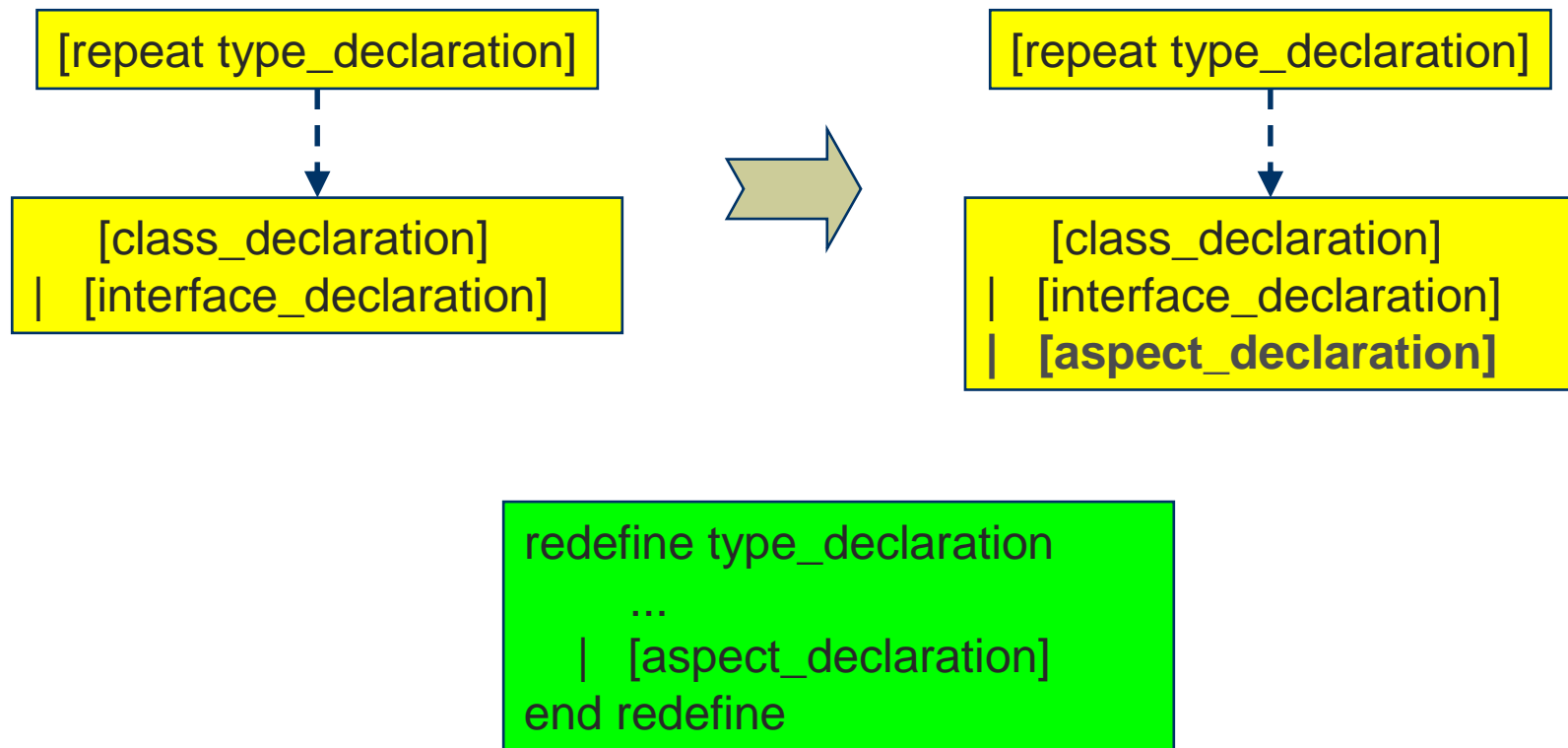
Purpose of this grammar

- ◆ Parse and transform AspectJ code
 - Instrumentation of the new aspect code for JConsole
 - Evaluate the coverage of the aspect code
 - Develop new test cases to get a complete coverage for the aspect code

The Java syntax tree



The first extension



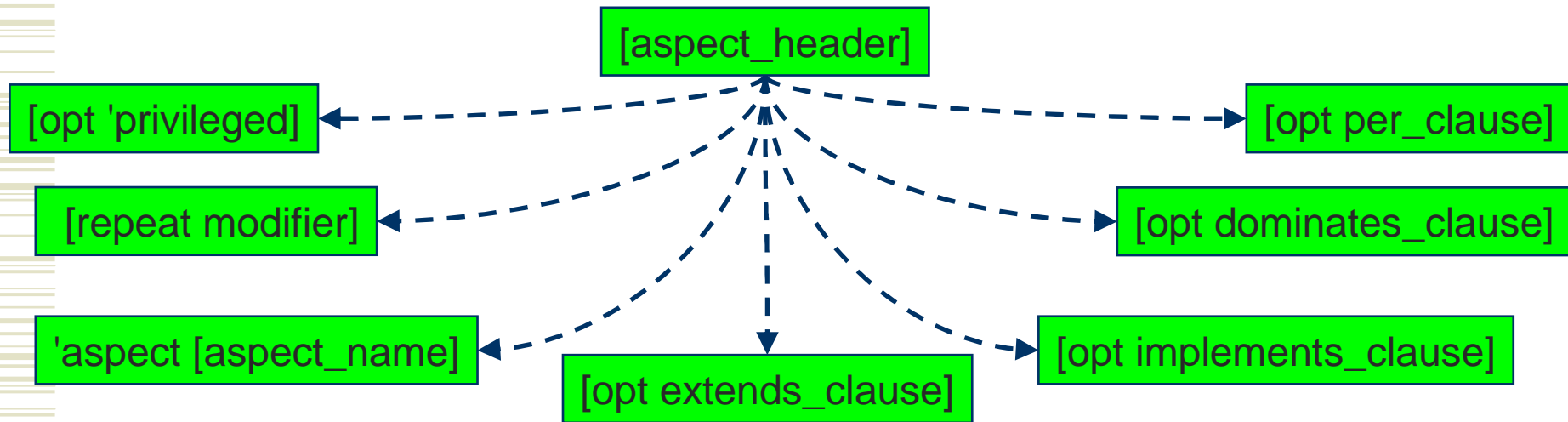
Aspect declaration

```
define class_declaration  
  [class_header] [class_body]  
end define
```

```
define interface_declaration  
  [interface_header] [interface_body]  
end define
```

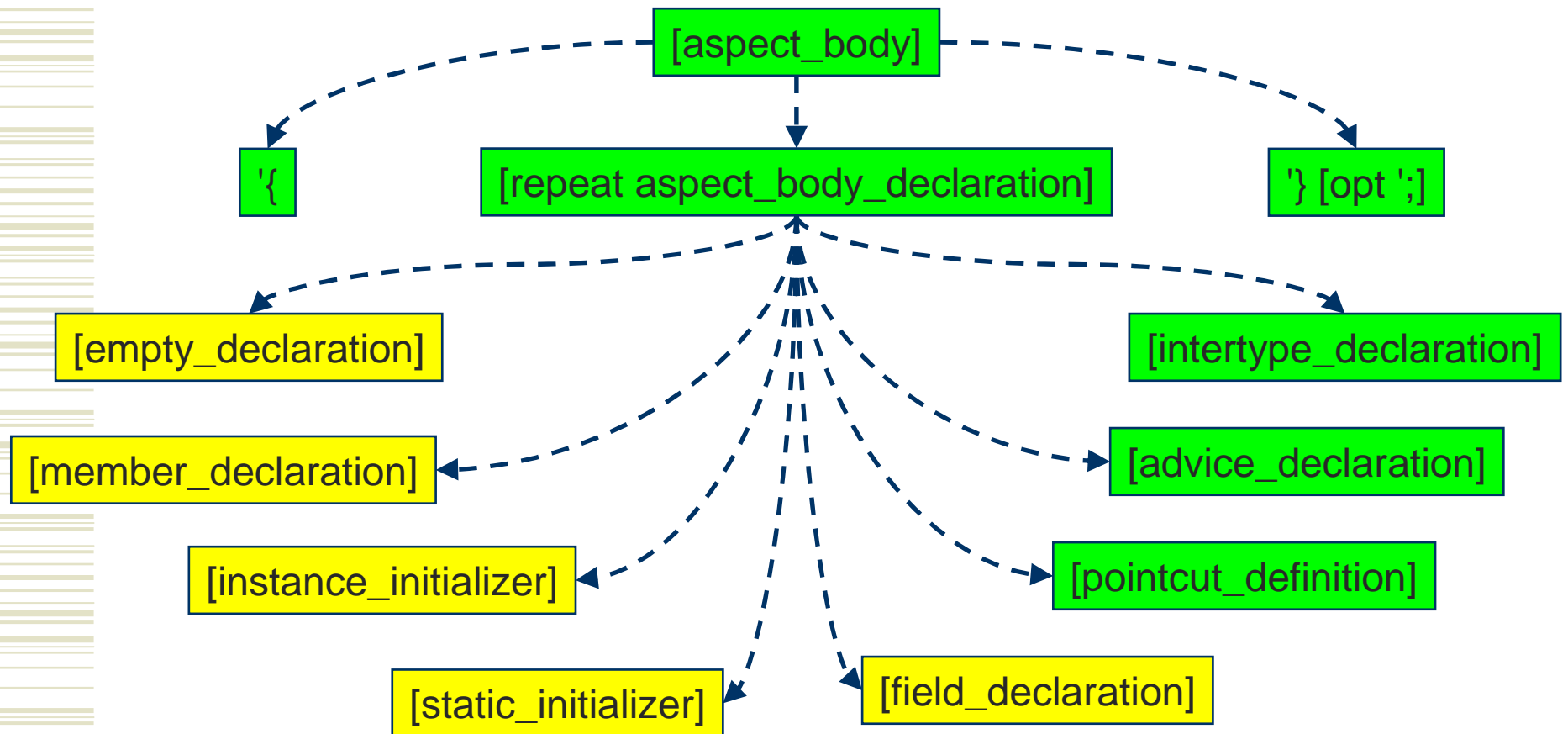
```
define aspect_declaration  
  [aspect_header] [aspect_body]  
end define
```

Aspect header

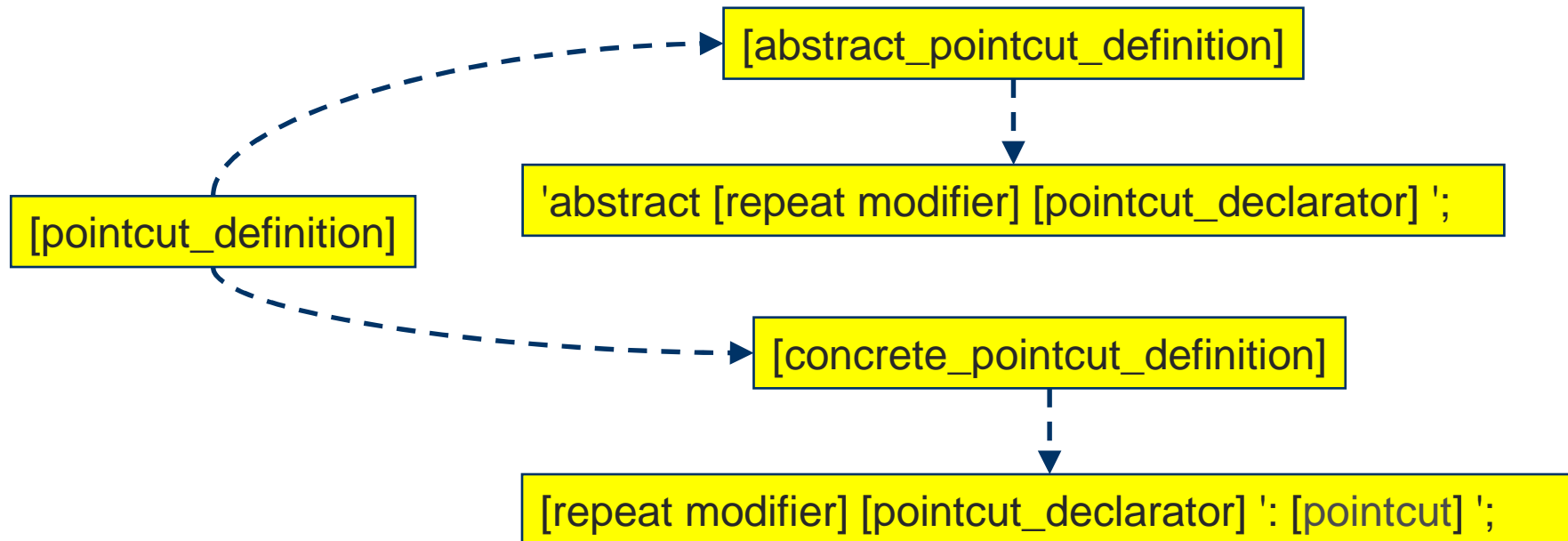


```
public aspect A {...}
privileged aspect B {...}
aspect C extends D implements I,J {...}
aspect E dominates A,B,C {...}
```

Aspect body



Pointcut definition



```
define pointcut_declarator
    'pointcut [pointcut_name] '( [list formal_parameter] '
end define
```

Pointcut definition

```
'abstract [repeat modifier] [pointcut_declarator] ';
```

```
abstract pointcut allConstructor;
```

```
[repeat modifier] [pointcut_declarator] ': [pointcut] ';
```

```
pointcut allConstructor():  
    call( public *.new(..) );  
  
private pointcut stetterinvocation(Object o):  
    call( * *.set*(..) ) && target( o );
```

Pointcut definition

define pointcut

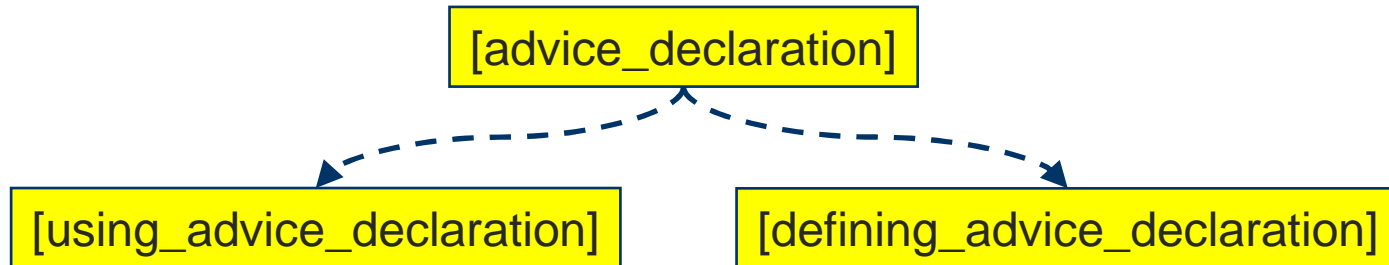
```
'call '( [method_pattern] ' )
| 'call '( [constructor_pattern] ' )
| 'execution '( [method_pattern] ' )
| 'execution '( [constructor_pattern] ' )
| 'initialization '( [constructor_pattern] ' )
| 'preinitialization '( [constructor_pattern] ' )
| 'get '( [field_pattern] ' )
| 'set '( [field_pattern] ' )
| 'handler '( [type_pattern] ' )
| 'adviceexecution '( ' )
| 'within '( [type_pattern] ' )
```

```
| 'withincode '( [method_pattern] ' )
| 'withincode '( [constructor_pattern] ' )
| 'cflow '( [pointcut] ' )
| 'cflowbelow '( [pointcut] ' )
| 'if '( [expression] ' )
| 'this '( [var_or_type] ' )
| 'target '( [var_or_type] ' )
| 'args '( [list var_or_type_or_twoPoint] ' )

| [pointcut] '|| [pointcut]
| [pointcut] '&& [pointcut]
| ! [pointcut]
| '( [pointcut] ' )
| [pointcut_name]
```

end define

Advice declaration



```
define using_advice_declaration
  [repeat modifier] [advice_specification] [opt throws] ':
    [pointcut_name] [pointcut_argument]
    [advice_body]
end define
```

```
define defining_advice_declaration
  [repeat modifier] [advice_specification] [opt throws] ':
    [pointcut] [advice_body]
end define
```

Advice declaration

[using_advice_declaration]

```
define using_advice_declaration
  [repeat modifier] [advice_specification] [opt throws] ':
    [pointcut_name] [pointcut_argument]
    [advice_body]
end define
```

```
before ( Object caller, Object called ):
  observerInvocation ( caller, called ) {...}
```

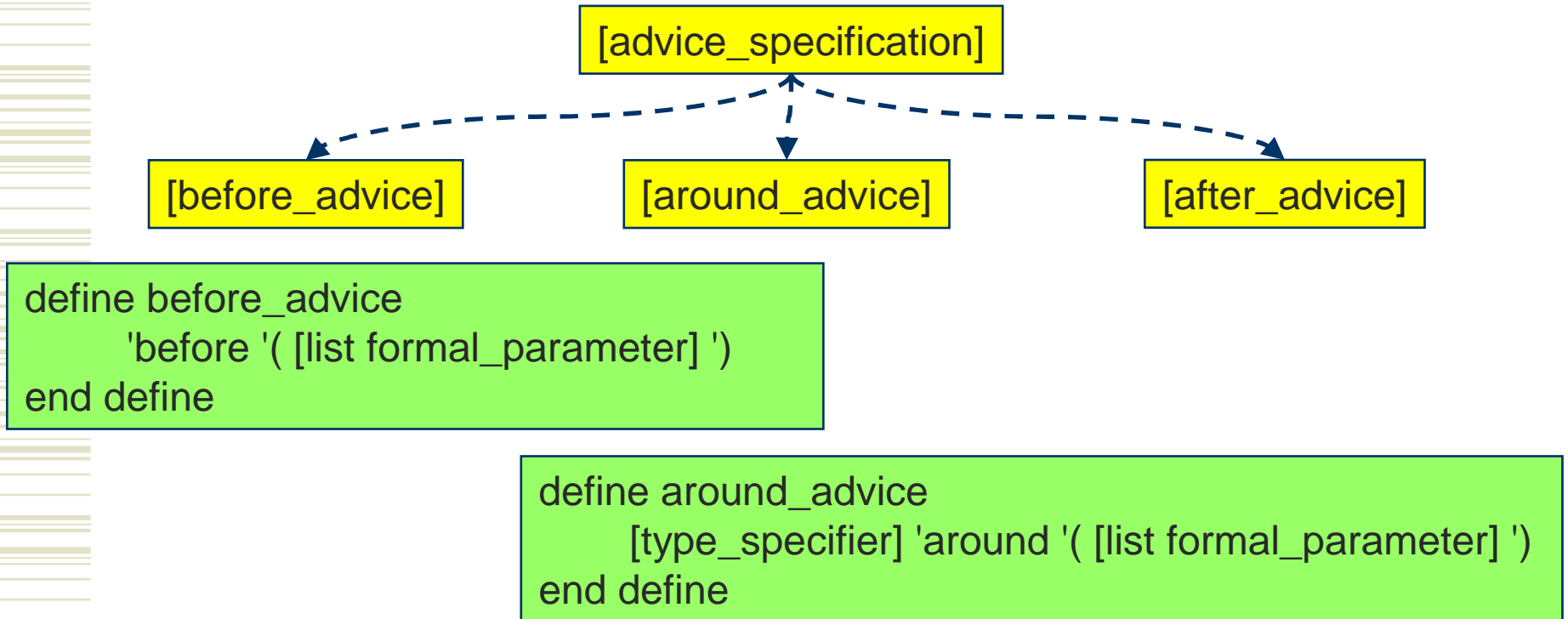
Advice declaration

[defining_advice_declaration]

```
define defining_advice_declaration
  [repeat modifier] [advice_specification] [opt throws] ':
    [pointcut] [advice_body]
end define
```

```
before ( Object caller, Object called ):
  call ( * observer.*(..) ) &&
  this ( caller ) && target ( called ) {...}
```

Advice specification



Advice specification

```
before ( ) :  
    call ( public String MyClass.toString() )  
    {  
        System.out.println( prefix );  
    }
```

```
String around ( ) :  
    call ( public String MyClass.toString() )  
    {  
        String result = proceed( );  
        return new String( prefix + result );  
    }
```

Advice specification

```
define after_advice
  'after ' ( [list formal_parameter] ' )
  | 'after ' ( [list formal_parameter] ' ) 'returning
  | 'after ' ( [list formal_parameter] ' ) 'returning ' ( [formal_parameter] ' )
  | 'after ' ( [list formal_parameter] ' ) 'throwing
  | 'after ' ( [list formal_parameter] ' ) 'throwing ' ( [formal_parameter] ' )
end define
```

```
after ( ) throwing java.io.IOException:
  call ( public String MyClass.toString() )
  {...}
```

Intertype declaration

[intertype_declaration]

```
define intertype_declaration
  [method_introduction]
  | [field_introduction]
  | [constructor_introduction]
  | [parent_declaration]
  | [warning_declaration]
  | [error_declaration]
  | [soft_declaration]
  | [precedence_declaration]
end define
```

Method Introduction

```
define method_introduction
  [repeat modifier] [type_specifier]
    [type_specifier] '. [method_name] '( [list formal_parameter] ' )
    [opt throws]
    [method_body]
end define
```

```
public String MyClass.toString( ) throws java.io.IOException
{
  return ... ;
}
```

Field introduction

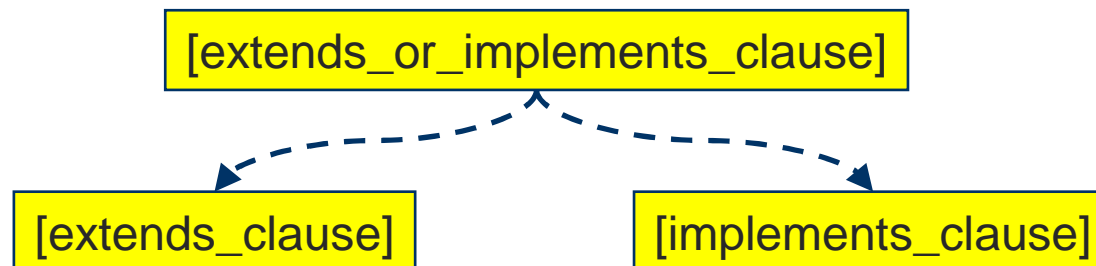
```
define field_introduction
    [repeat modifier] [type_specifier]
        [type_specifier] ' . [variable_name]
        [opt equals_variable_initializer] ';
end define
```

```
public String MyClass.default;

protected Vector MyClass.element = new Vector( 100 );
```

Parent declaration

```
define parent_declaration
  'declare 'parents ':
    [type_pattern] [extends_or_implements_clause] ';
end define
```



```
declare parents:
  MyString implements java.io.Serializable;

declare parents:
  MyString extends java.lang.String;
```

Language constraints

- ◆ The java grammar must be extended
- ◆ A new reserved word to prevent the usage of the word “**aspect**”
- ◆ A new compound, to prevent the separation of the two points in the wildcard “**..**”

```
include "Java.Grm"
```

```
keys  
  'aspect  
end keys
```

```
compounds  
  '..  
end compounds
```