Contents

0.0.1 The Meta Compiler

$$\textbf{generation-syntax} \quad : \quad \textit{theory} \rightarrow \textit{theory}$$
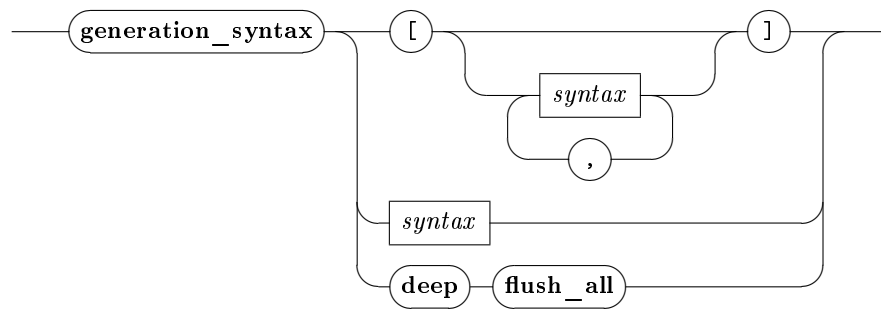
syntax
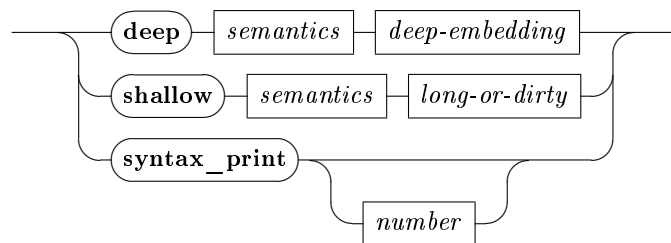
semantics

deep-embedding

skip_export

(THEORY *name*)

(IMPORTS [*name* ,] *name*)

SECTION

long-or-dirty

[*export-code* ,]

(output_directory *name*)

export-code

in
Haskell
OCaml
Scala
SML
module_name *name* (*args*)

3
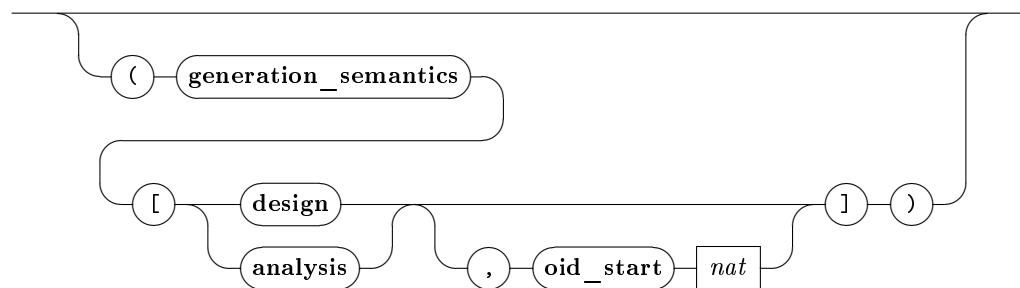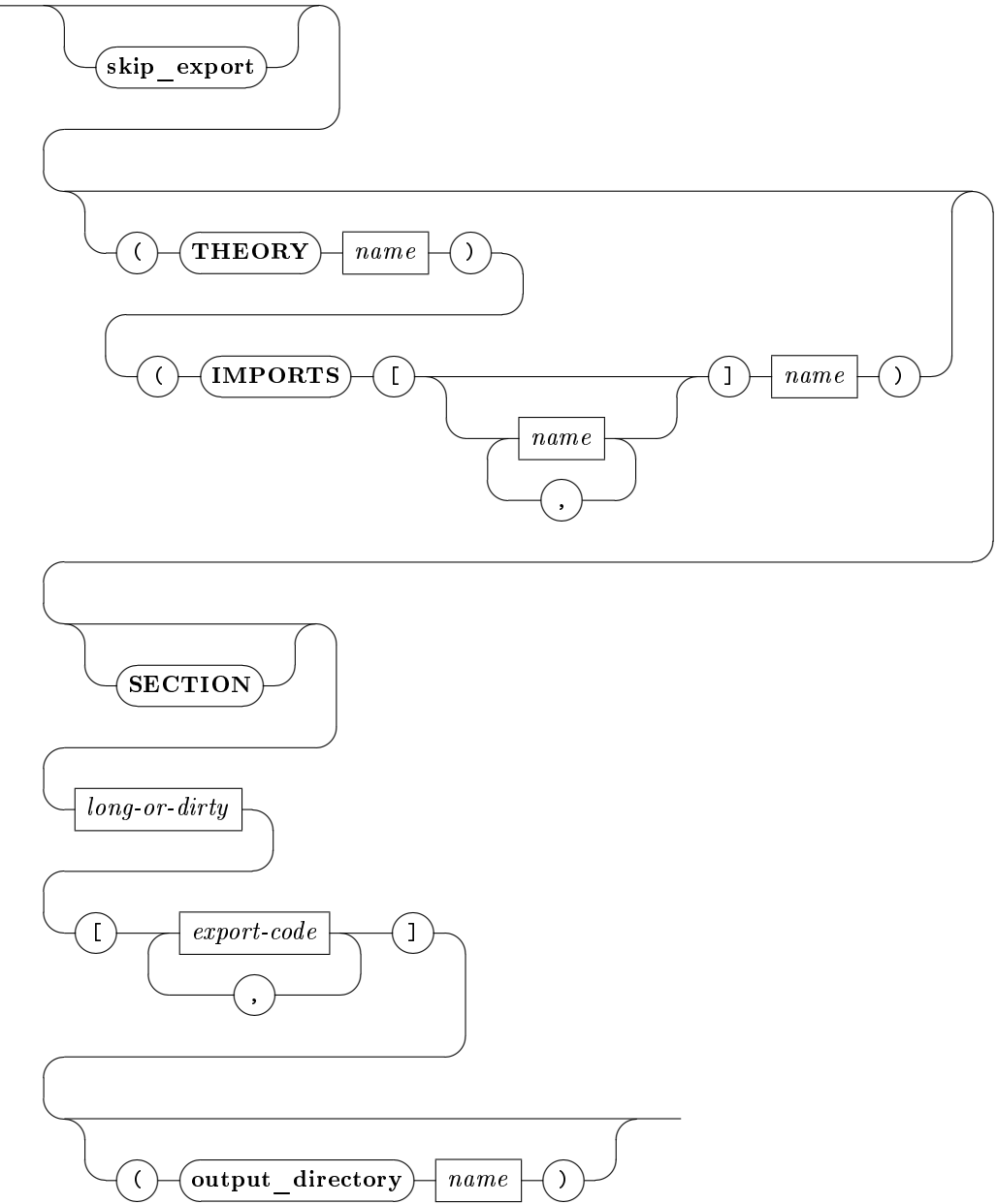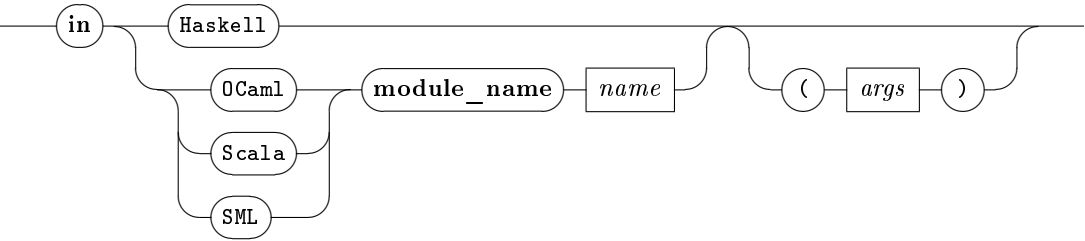
long-or-dirty

```
        ┌──────────────────────────┐
        │    ╭─────────╮           │
        ├────┤  SORRY  ├───────────┤
             ╰─────────╯
             ╭─────────╮
        ╰────┤ no_dirty├───────────╯
             ╰─────────╯
```
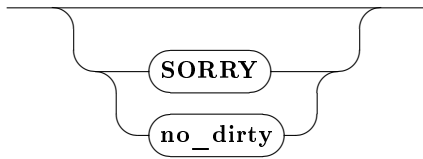
0.0.2 UML/OCL: Extended USE Grammar

....

$$\mathbf{Class} \quad : \quad \textit{theory} \rightarrow \textit{theory}$$
$$\mathbf{Abstract\text{-}class} \quad : \quad \textit{theory} \rightarrow \textit{theory}$$

```
         ╭───────────╮        ┌─────────┐  ┌──┐  ┌───────────┐
    ─────┤   Class   ├────────┤ binding ├──┤ =├──┤ type-base ├──────
         ╰───────────╯        └─────────┘  └──┘  └───────────┘
         ╭───────────────╮    ┌─────────────┐  ┌───────┐
    ─────┤ Abstract_class├────┤ type-object ├──┤ class ├────────────
         ╰───────────────╯    └─────────────┘  └───────┘
```

class

```
    ┌─────────────────────────────────────────────────────┐
    │    ╭──────────────╮    ┌─────────┐  ┌──┐  ┌──────────┐│
    ├────┤  Attributes  ├────┤ binding ├──┤ :├──┤ uml-type ││
         ╰──────────────╯    └─────────┘  └──┘  └──────────┘
                                        ╭───╮
                                        ┤ ; ├
                                        ╰───╯
    ┌─────────────────────────────────────────────────────┘
    │    ┌─────────┐
    ╰────┤ context ├───────
         └─────────┘
```

4
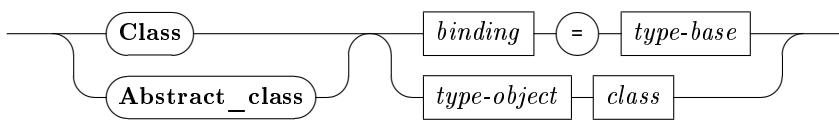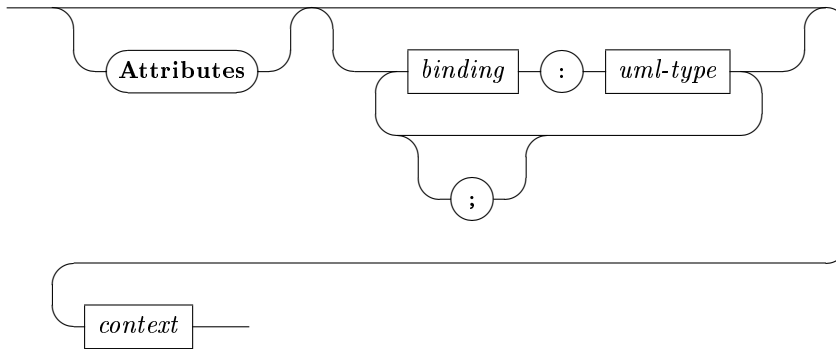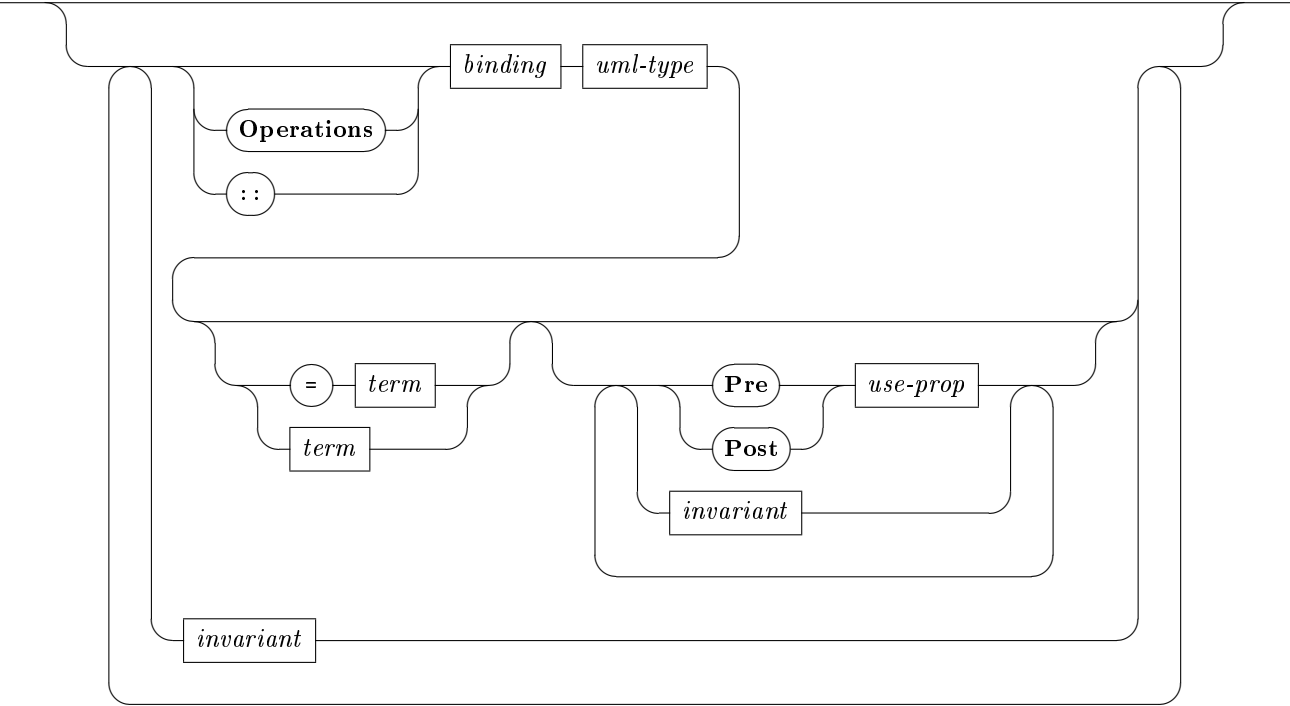
context

invariant
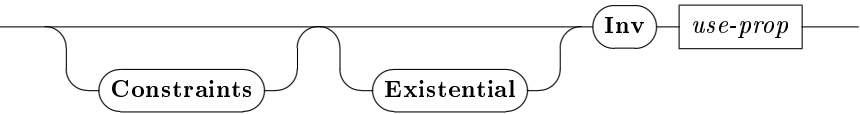
Aggregation	:	*theory* $\rightarrow$ *theory*
Association	:	*theory* $\rightarrow$ *theory*
Composition	:	*theory* $\rightarrow$ *theory*
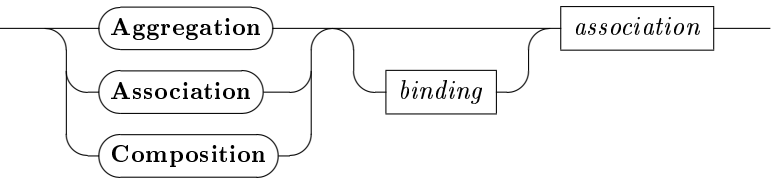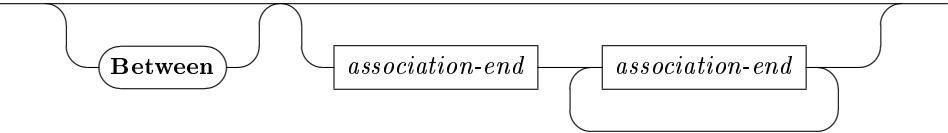
association

association-end

. . . .
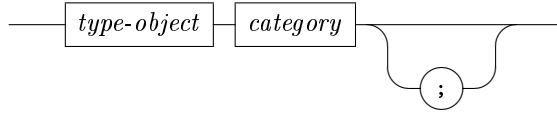
Associationclass   :   *theory* → *theory*
Abstract-associationclass   :   *theory* → *theory*

. . . .
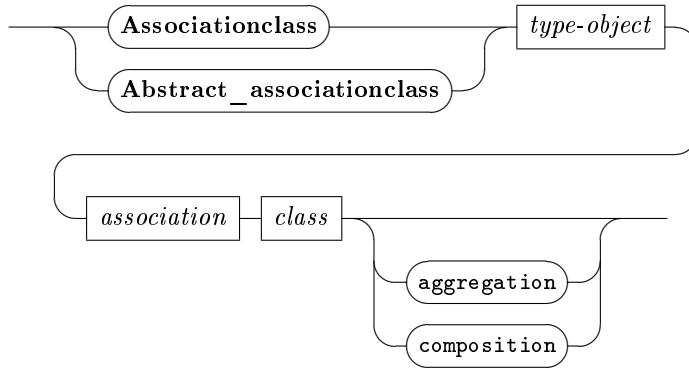
Context   :   *theory* → *theory*

. . . .

Instance   :   *theory* → *theory*

Instance

binding :: *type-object* =

term-object

object-cast

and

term-object

[(*binding* , *binding*) |=

binding = *uml-term*

,

]
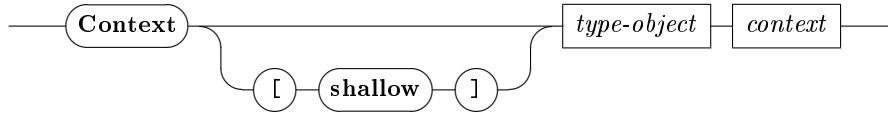
object-cast

(*term-object* :: *type-object*)
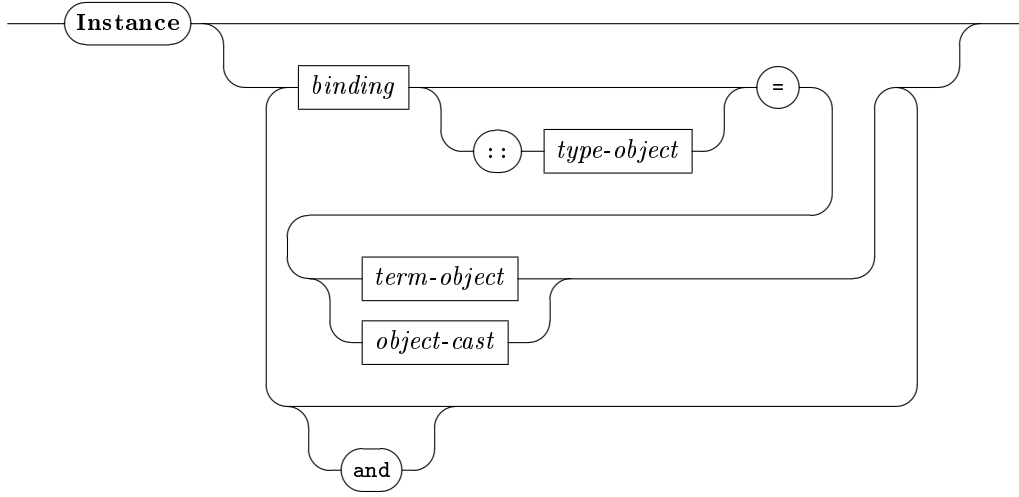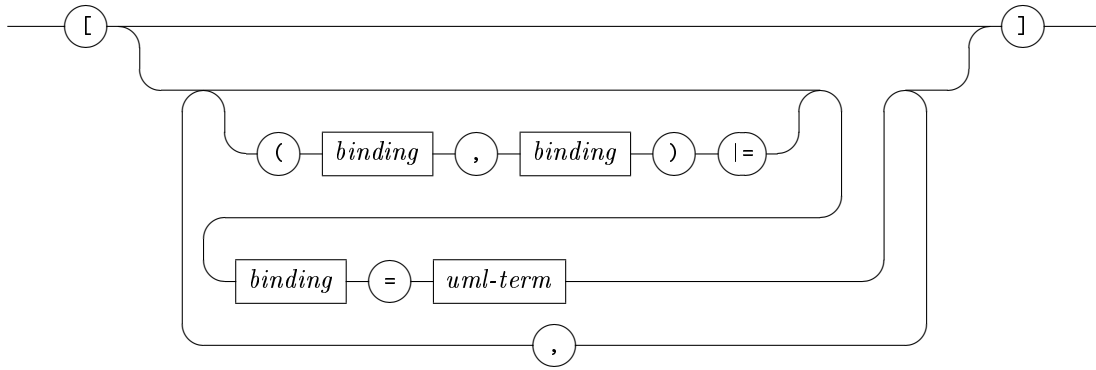
→ oclAsType (*type-object*)
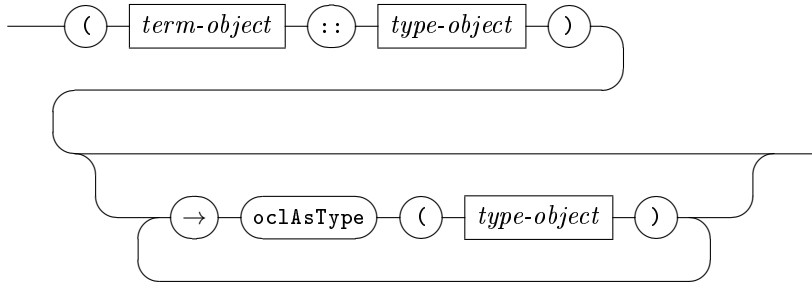
State : *theory* → *theory*

State [**shallow**] *binding* = *state*

7
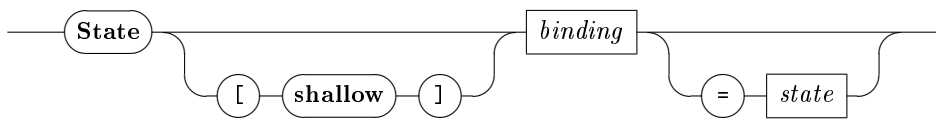
state

....
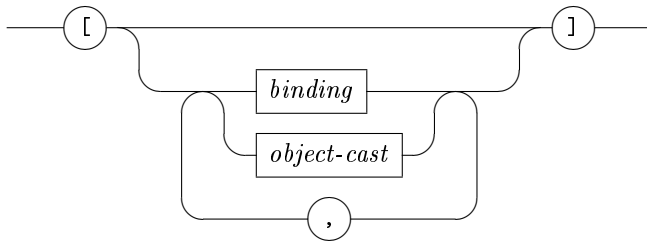
$$\textbf{PrePost} \quad : \quad theory \rightarrow theory$$
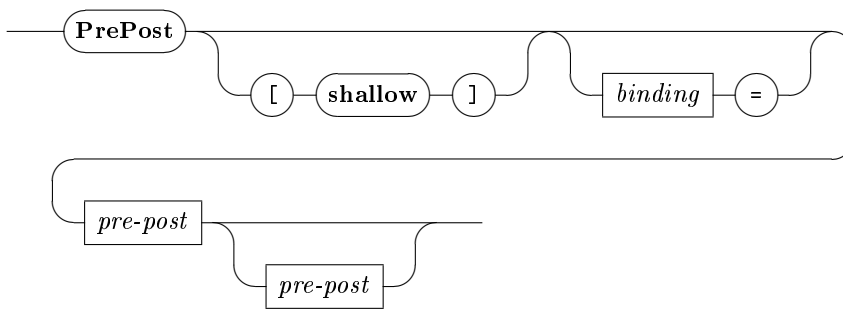
pre-post
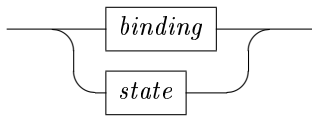
0.0.3 UML/OCL: Type System

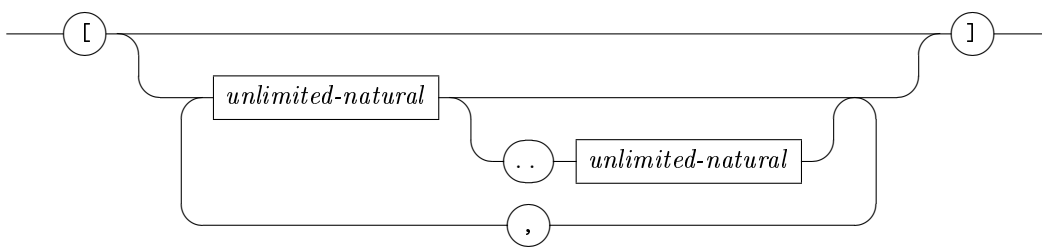..
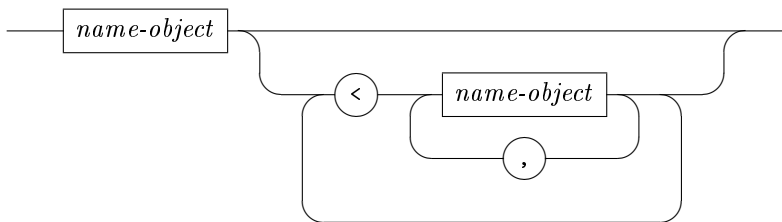
unlimited-natural
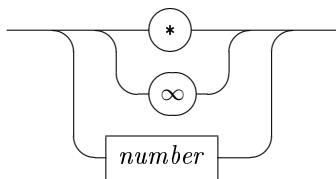
term-base
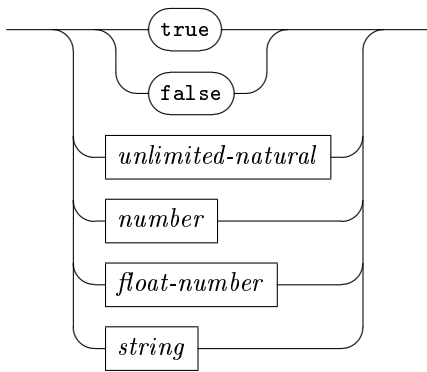
multiplicity

uml-term

9
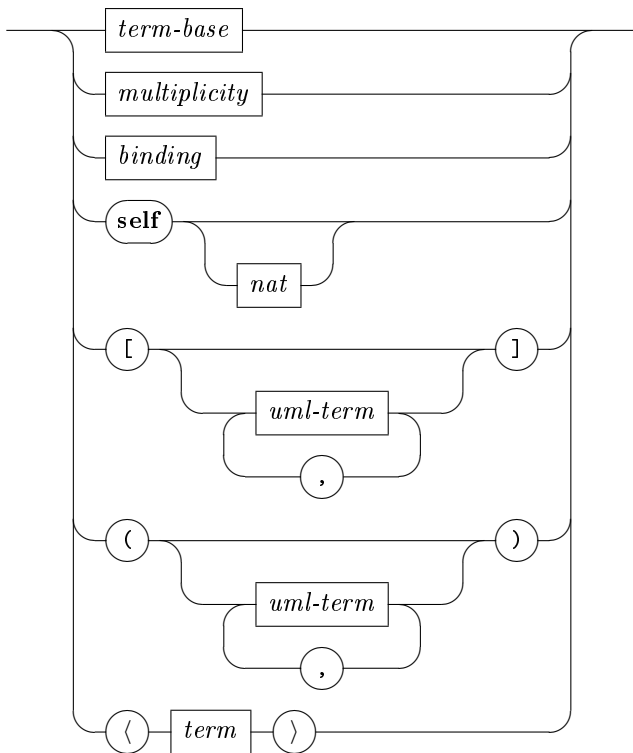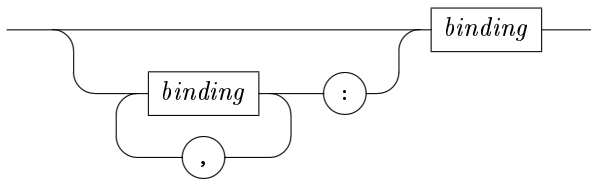
name-object

type-base

type-object

category

```
──── multiplicity ─┐
                   │
   ┌───────────────┴──────────────────────────────┐
   │  Role ── binding ─┬──────────────────────────┤
                       │   Derived ── = ── term    │
                       ├── Nonunique ──────────────┤
                       ├── Ordered ────────────────┤
                       ├── Qualifier ── uml-type ──┤
                       ├── Redefines ── binding ───┤
                       ├── Sequence_ ──────────────┤
                       ├── Subsets ── binding ─────┤
                       └── Union ──────────────────┘
```

uml-type

type-base

type-object

Sequence — *uml-type*

Set

category

Pair — *uml-type* — *uml-type*

(binding : (*uml-type*) — uml-type-weak ,)

: *uml-type*

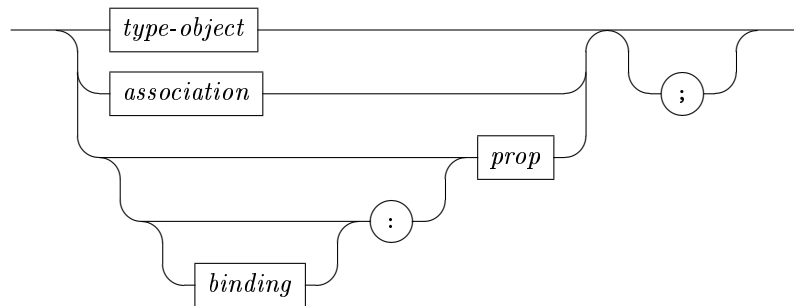(*uml-type*)

⟨ *type* ⟩

use-prop

type-object

association

prop

binding :

;
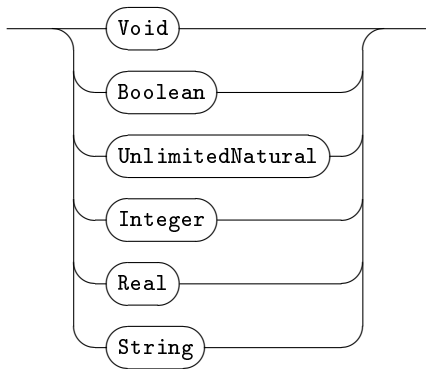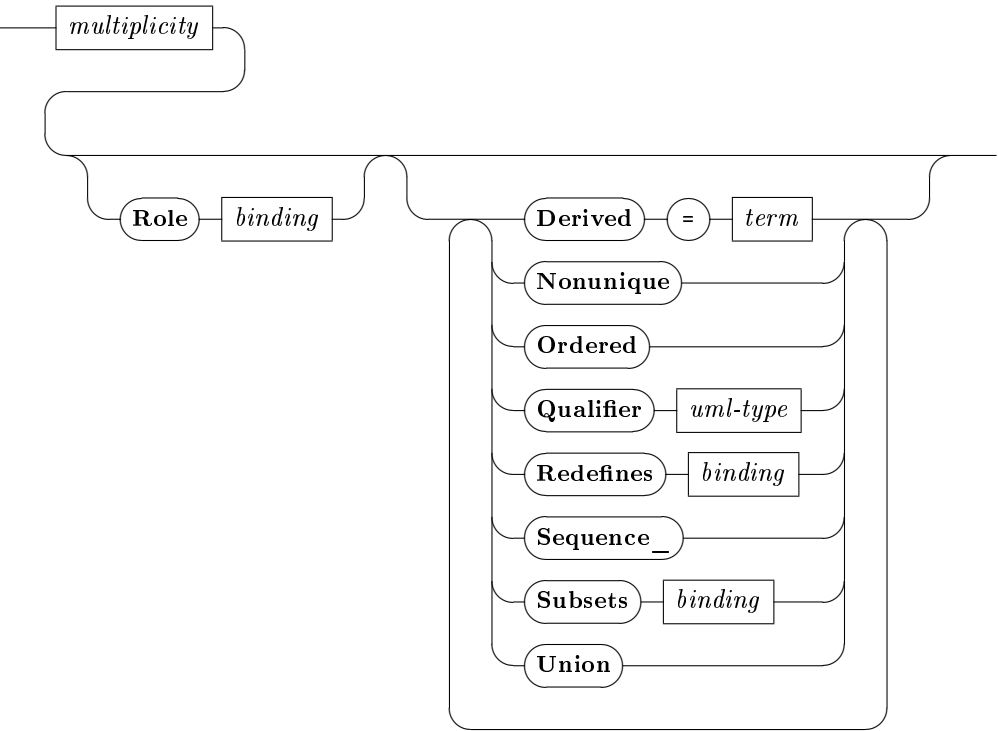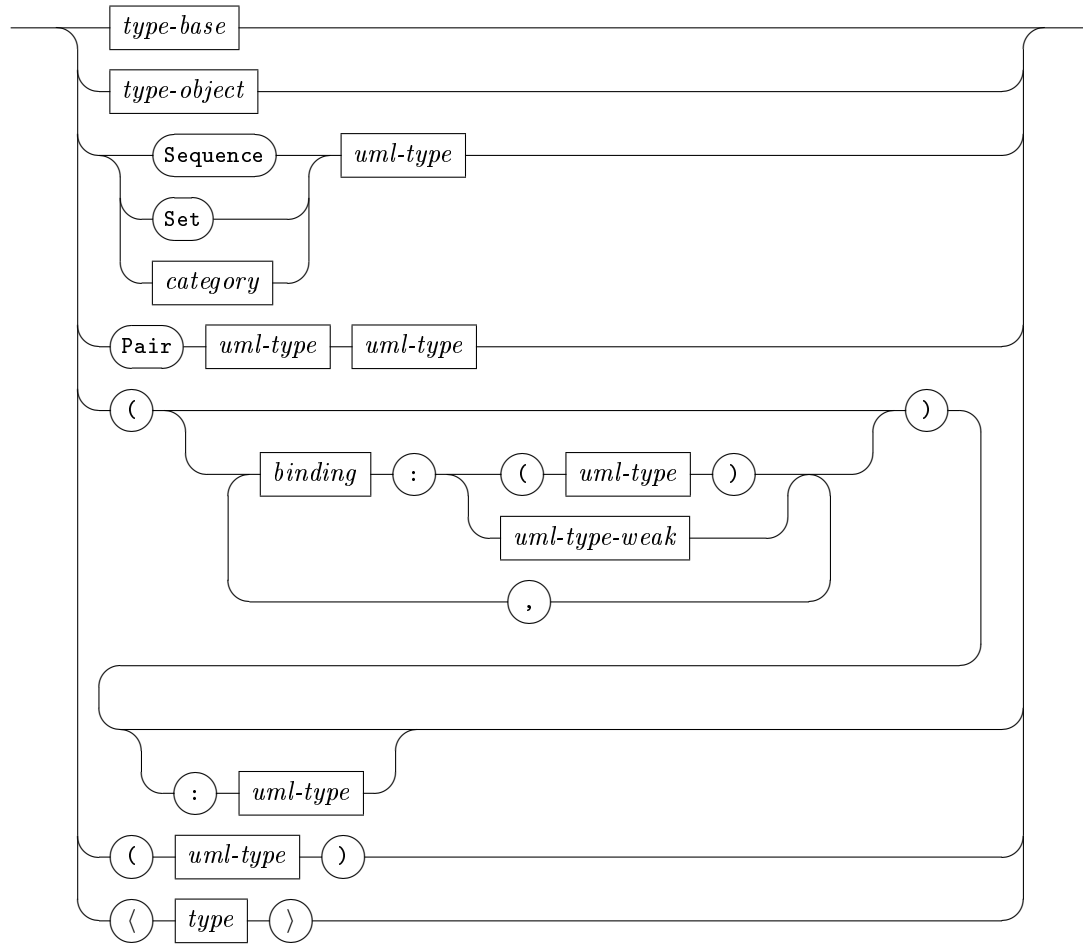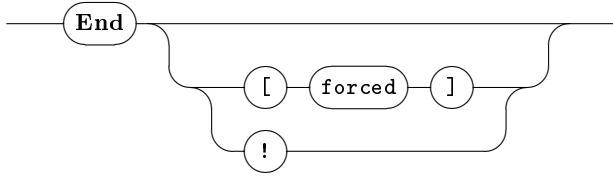
uml_type_weak is like uml_type except that type_object can not contain quantified names.

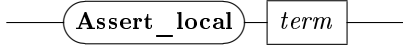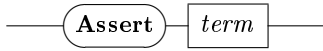## 0.0.4 UML/OCL: Miscellaneous

End : *theory* → *theory*

```
........................................................................................
```

$$\textbf{BaseType} \quad : \quad \textit{theory} \rightarrow \textit{theory}$$
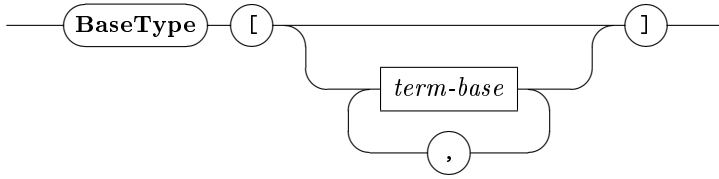
0.0.5 Featherweight OCL: Library

$$
\begin{aligned}
\textbf{Assert} \quad &: \quad \textit{theory} \rightarrow \textit{theory} \\
\textbf{Assert-local} \quad &: \quad \textit{local-theory} \rightarrow \textit{local-theory}
\end{aligned}
$$

0.0.6 Featherweight OCL: Auxiliary

$$
\begin{aligned}
\textbf{lazy-text} \quad &: \quad \textit{local-theory} \rightarrow \textit{local-theory} \\
\textbf{apply-text} \quad &: \quad \textit{local-theory} \rightarrow \textit{local-theory} \\
\textbf{reset-text} \quad &: \quad \textit{local-theory} \rightarrow \textit{local-theory}
\end{aligned}
$$
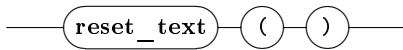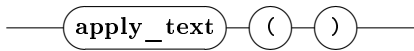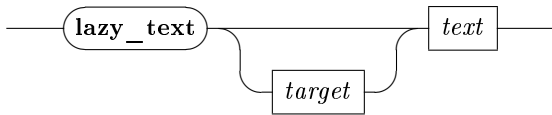
0.0.7 Featherweight OCL: Compiler

$$
\begin{array}{rcl}
\textbf{fun}' & : & \textit{local-theory} \rightarrow \textit{local-theory} \\
\textbf{definition}' & : & \textit{local-theory} \rightarrow \textit{local-theory} \\
\textbf{code-reflect}' & : & \textit{theory} \rightarrow \textit{theory}
\end{array}
$$

....
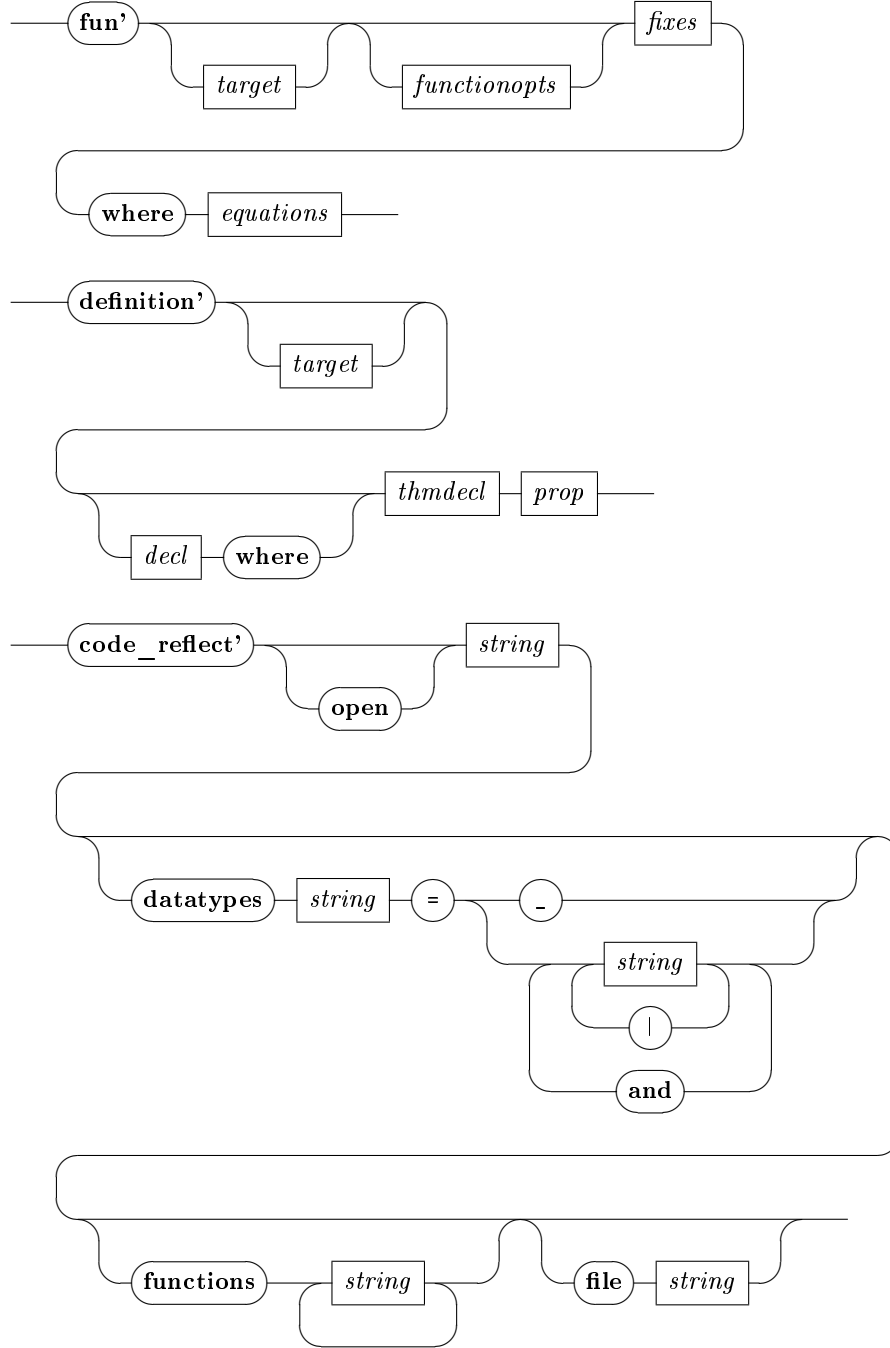
$$\begin{array}{rcl}
\textbf{lazy-code-printing} & : & \textit{theory} \rightarrow \textit{theory} \\
\textbf{apply-code-printing} & : & \textit{theory} \rightarrow \textit{theory} \\
\textbf{apply-code-printing-reflect} & : & \textit{local-theory} \rightarrow \textit{local-theory}
\end{array}$$

(lazy_code_printing)
- *printing-const*
- *printing-typeconstructor*
- *printing-class*
- *printing-class-relation*
- *printing-class-instance*
- *printing-module*
- |

(apply_code_printing) ( ) ( )

(apply_code_printing_reflect) *text*

0.1 …

[1]

Bibliography

[1] ...