

# Funcons-beta: Objects \*

The PPlanCompS Project

Objects.cbs | PLAIN | PRETTY

---

## Objects

```
[ Datatype  objects
  Funcon    object
  Funcon    object-identity
  Funcon    object-class-name
  Funcon    object-feature-map
  Funcon    object-subobject-sequence
  Funcon    object-tree
  Funcon    object-single-inheritance-feature-map ]
```

*Datatype* `objects` ::= `object`(`_` : `atoms`, `_` : `identifiers`, `_` : `environments`, `_` : `objects`\*)

`object`( $A, C, Env, O^*$ ) is an object:  $*$  distinguished by an atom  $A$ ,  $*$  of a class named  $C$ ,  $*$  with an environment  $Env$  with the features of the object, and  $*$  a sequence  $O^*$  of subobjects of the direct superclasses of  $C$ . `object`( $A, C, Env$ ) is an object of a base class. `object`( $A, C, Env, O'$ ) is an object of a class with a single superclass. With multiple inheritance, subobjects due to repeated inheritance of the same class may be shared.

Implementations of objects generally represent an object as a vector of fields, and use pointers and offsets for efficient access to individual fields. The representation of objects used in this specification is independent of such implementation concerns.

```
Funcon  object-identity(_ : objects) :  $\Rightarrow$  atoms
Rule    object-identity
          object( $A$  : atoms, _ : identifiers, _ : environments, _ : objects*)  $\rightsquigarrow$ 
           $A$ 
```

```
Funcon  object-class-name(_ : objects) :  $\Rightarrow$  identifiers
Rule    object-class-name
          object(_ : atoms,  $C$  : identifiers, _ : environments, _ : objects*)  $\rightsquigarrow$ 
           $C$ 
```

```
Funcon  object-feature-map(_ : objects) :  $\Rightarrow$  environments
Rule    object-feature-map
          object(_ : atoms, _ : identifiers,  $Env$  : environments, _ : objects*)  $\rightsquigarrow$ 
           $Env$ 
```

---

\*Suggestions for improvement: [plancomps@gmail.com](mailto:plancomps@gmail.com).  
Reports of issues: <https://github.com/plancomps/CBS-beta/issues>.

*Funcon*  $\text{object-subobject-sequence}(\_ : \text{objects}) : \Rightarrow \text{objects}^*$

*Rule*  $\text{object-subobject-sequence}$   
 $\text{object}(\_ : \text{atoms}, \_ : \text{identifiers}, \_ : \text{environments}, O^* : \text{objects}^*) \rightsquigarrow$   
 $O^*$

*Funcon*  $\text{object-tree}(\_ : \text{objects}) : \Rightarrow \text{trees}(\text{objects})$

$\text{object-tree } O$  forms a tree where the branches are the object trees for the direct subobjects of  $O$ .

*Rule*  $\text{object-tree}(O : \text{objects}) \rightsquigarrow$   
 $\text{tree}(\$   
     $O,$   
     $\text{interleave-map}(\$   
         $\text{object-tree given},$   
         $\text{object-subobject-sequence } O))$

*Funcon*  $\text{object-single-inheritance-feature-map}(O : \text{objects}) : \Rightarrow \text{environments}$   
 $\rightsquigarrow \text{map-override left-to-right-map}(\$   
     $\text{object-feature-map given},$   
     $\text{single-branching-sequence object-tree } O)$

For multiple inheritance, different resolution orders can be specified by using difference linearisations of the object tree.