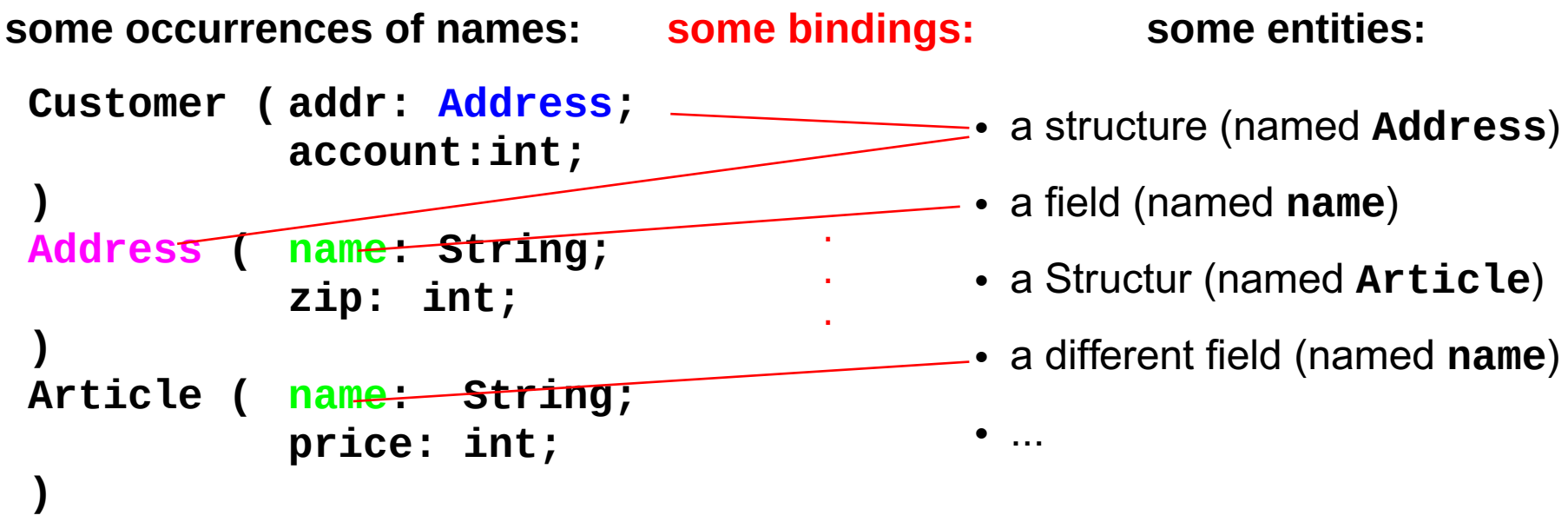
5. Binding Names to Entities

Names in the source code represent **entities** to describe the meaning of the text.

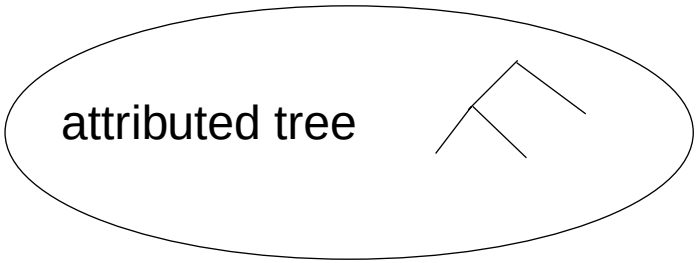
Occurrences of names are bound to **entities**.

Scope rules of the language specify how names are to be bound. E.g.:

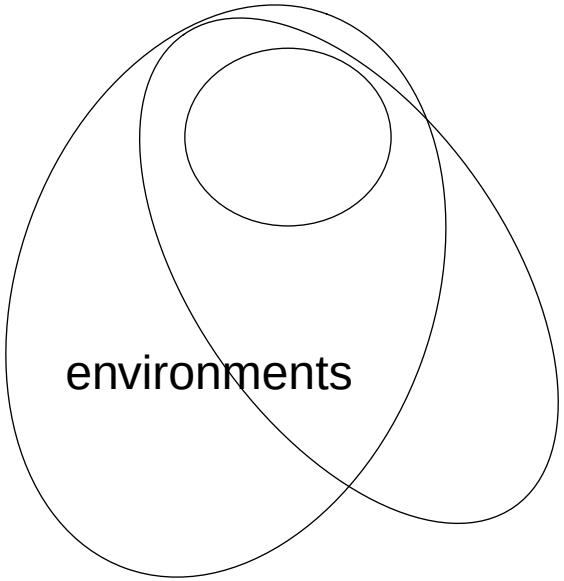
- Every name **a**, used as a structure name or as a type name is bound to the same entity.
- A type name **a** is an applied occurrence of a name. There must be a defining occurrences of **a** somewhere in the text.
- Field names are bound separately for every structure.



Keys and Properties



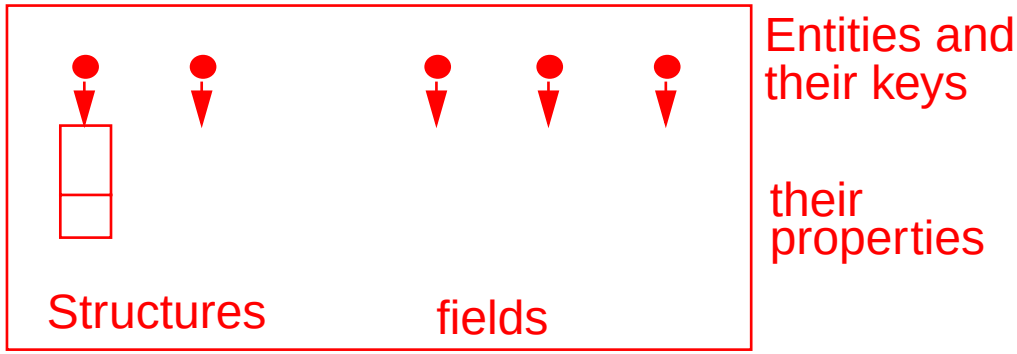
Eli tools implement properties of entities and of envivronments



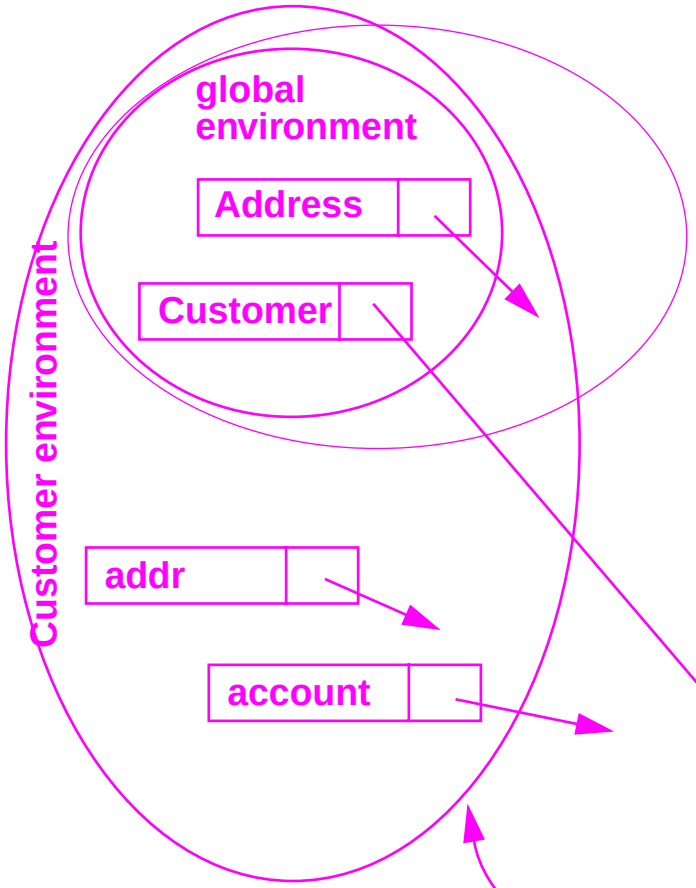
Entities are represented by keys.
Properties are associated to them.

Structures have a property called **`Environment`**

Definition module



Bindings and Environments



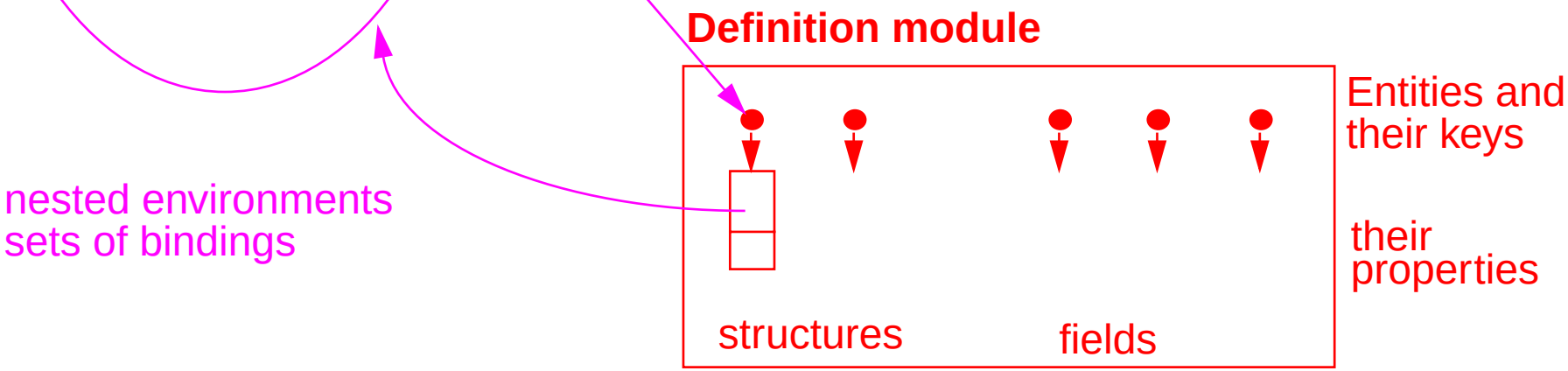
Environment: nested sets of bindings

Binding: associates a name with a key

The global environment binds all structure and type names.

The environment of a structure binds its field names.

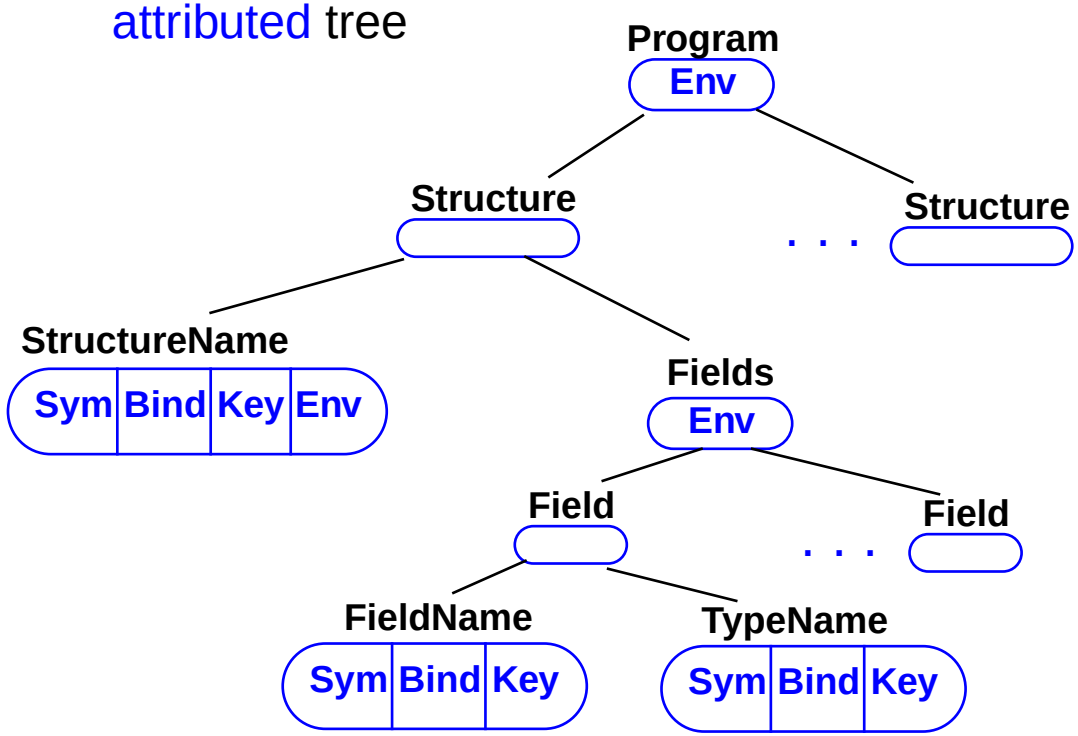
Eli tools implement properties of entities and of envivronments



Attributed Tree for Name Analysis

Attributes of the tree nodes describe properties of the program construct

attributed tree



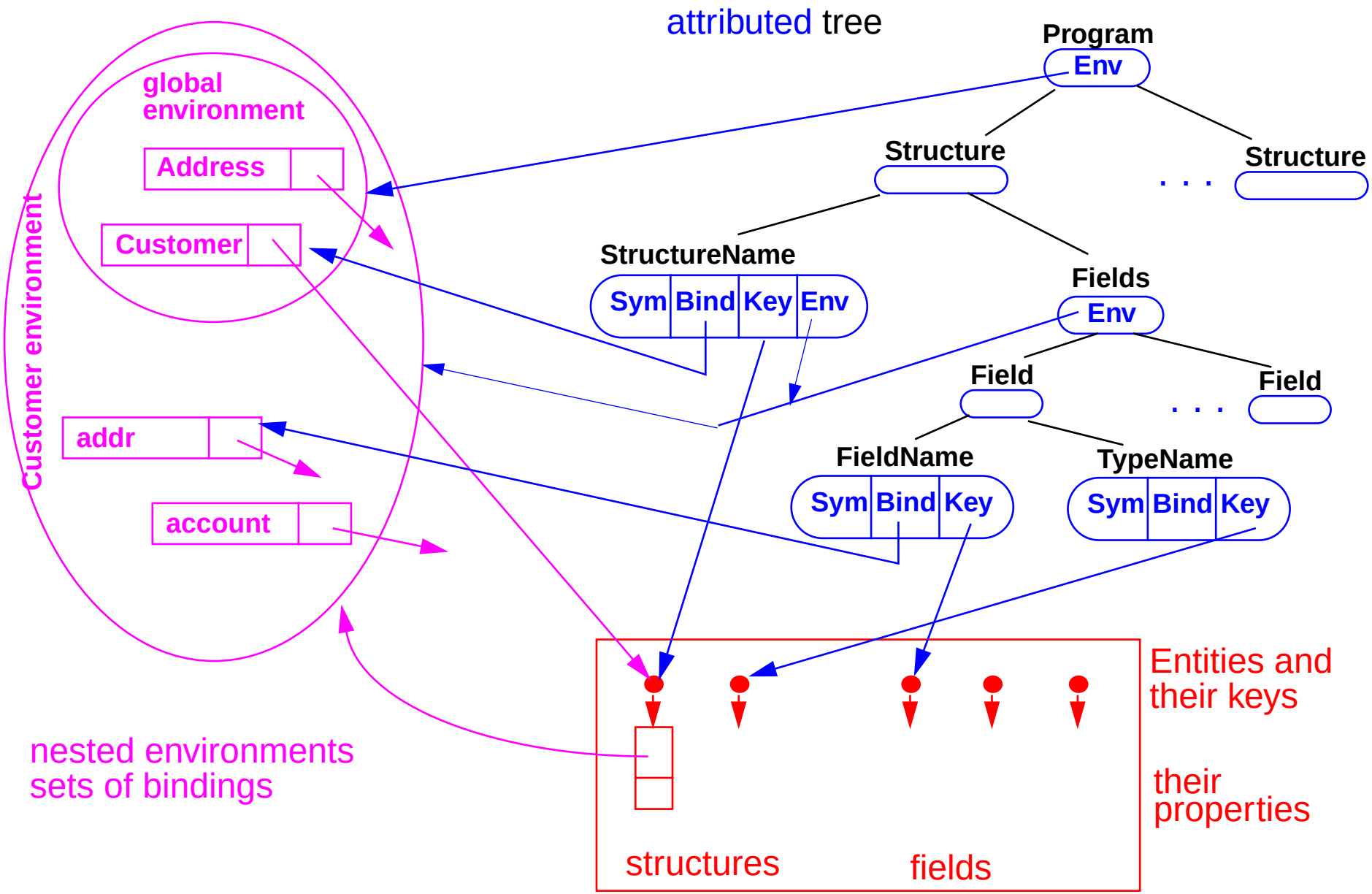
Program has the global environment

StructureName and Fields have the environment of the structure

Every node for a name occurrences has attributes for

- the code of the identifier,
- the binding of its name, and
- its key

Attributes, Environments, and Keys



Environment Module

Implements the abstract data type **Environment**:
hierarchally nested sets (tree) of **bindings (name, environment, key)**

Functions:

NewEnv ()	creates a new environment e, that is the root of a new tree; used in **root context**
NewScope (e_1)	creates a new environment e_2 that is nested in e_1. Every binding of e_1 is a binding of e_2, too, if it is not hidden by a binding established for the same name in e_2; used in **range context**
BindIdn (e, id)	creates a new binding (id, e, k), if e does not yet have a binding for id; k is then a new key for a new entity; the result is in both cases the binding (id, e, k); used for **defining occurrences**.
BindingInEnv (e, id)	yields a binding (id, e_1, k) of e oder of a surrounding environment of e; if there is no such binding it yields NoBinding; used for **applied occurrences**
BindingInScope (e, id)	yields a binding (id, e, k) of e, if e directly contains such a binding; NoBinding otherwise; e.g. used for **qualified names**

Example: Names and Entities for the Structure Generator

Abstract syntax

```
RULE: Descriptions  LISTOF Import | Structure              END;
RULE: Import ::= 'import' ImportNames 'from' FileName      END;
RULE: ImportNames   LISTOF ImportName                      END;
RULE: Structure ::= StructureName '(' Fields ')'           END;
RULE: Fields        LISTOF Field                           END;
RULE: Field ::=     FieldName ':' TypeName ';'             END;
RULE: StructureName ::= Ident                              END;
RULE: ImportName ::=    Ident                              END;
RULE: FieldName ::=     Ident                              END;
RULE: TypeName ::=      Ident                              END;
```

Different nonterminals for identifiers in different roles, because different computations are expected, e.g. for defining and applied occurrences.

Computation of Environment Attributes

Root of the environment hierarchy

Fields play the **role of a Range**.

The inherited computation of **Env** is overridden.

```
SYMBOL Descriptions INHERITS RootScope END;

SYMBOL Fields  INHERITS RangeScope END;

RULE: Structure ::= StructureName '(' Fields ')'
COMPUTE
   Fields.Env = StructureName.Env;
END;
```

Each structure entity has an **environment as its property**.

It is **created only once** for every occurrence of a structure entity.

That environment is **embedded in the global environment.**

In that environment the field names are bound.

```
SYMBOL StructureName COMPUTE
   SYNT.GotEnvir =
      IF (EQ (GetEnvir (THIS.Key, NoEnv), NoEnv),
         ResetEnvir
            (THIS.Key,
             NewScope (INCLUDING Range.Env)));

   SYNT.Env =
      GetEnvir (THIS.Key, NoEnv) <- SYNT.GotEnvir;
END;
```

Defining and Applied Occurrences of Identifiers

Computations **`IdentOcc`** for all identifier occurrences.

```
CLASS SYMBOL IdentOcc: Sym: int,
CLASS SYMBOL IdentOcc COMPUTE
   SYNT.Sym = TERM;
END;
```

All defining occurrences **bind** their names in the **next enclosing Range**

```
SYMBOL StructureName
   INHERITS IdentOcc, IdDefScope END;
SYMBOL ImportName
   INHERITS IdentOcc, IdDefScope END;
SYMBOL FieldName
   INHERITS IdentOcc, IdDefScope END;
```

Bind an applied occurrence of an identifier in the enclosing environment;
report an error if there is no valid binding.

```
SYMBOL TypeName
   INHERITS IdentOcc, IdUseEnv, ChkIdScope END;
```