

## 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.

**some occurrences of names:**

**some bindings:**

**some entities:**

```

Customer ( addr: Address;
           account: int;
)
Address ( name: String;
          zip: int;
)
Article ( name: String;
          price: int;
)

```

- a structure (named **Address**)
- a field (named **name**)
- a Structur (named **Article**)
- a different field (named **name**)
- ...

### Lecture Generating Software from Specifications SS 2012 / Slide 501

#### Objectives:

Understand binding of names to entities

#### In the lecture:

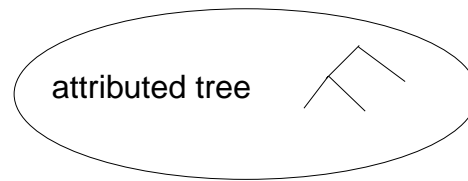
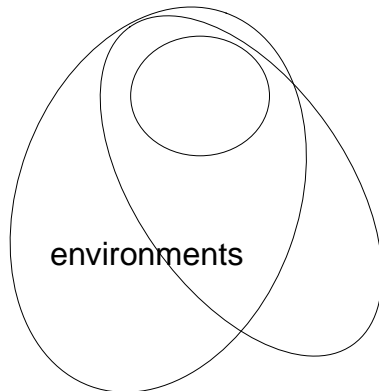
Explain the notions using the example:

- entities the text refers to,
- names of entities,
- occurrence of a name bound to an entity,
- scope of bindings.

#### Suggested reading:

GdP-3.1 ff

# Keys and Properties

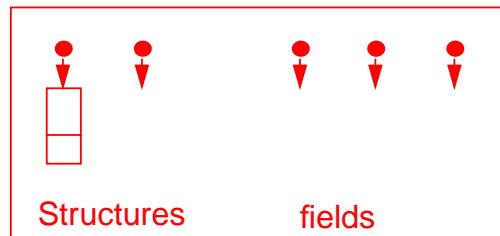


**Eli tools** implement properties of entities and of environments

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

Structures have a property called **Environment**

## Definition module



Entities and their keys

their properties

## Lecture Generating Software from Specifications SS 2012 / Slide 502

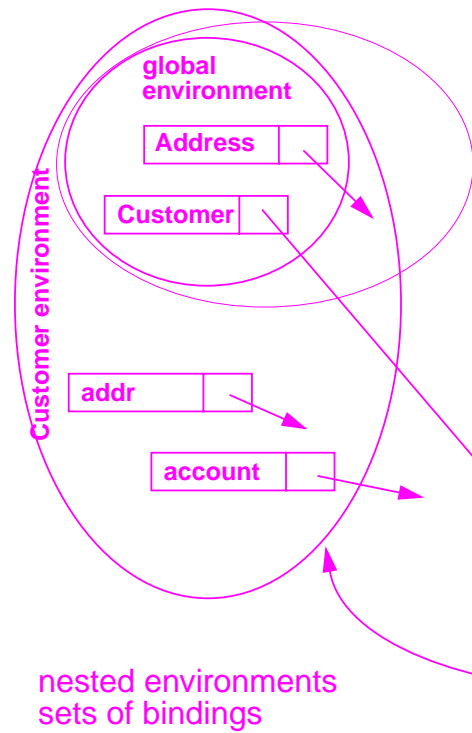
### Objectives:

Overview over properties of entities

### In the lecture:

The topics of the slide are explained.

# 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 environments

## Lecture Generating Software from Specifications SS 2012 / Slide 503

### Objectives:

Overview over bindings

### In the lecture:

The topics of the slide are explained.

# Attributed Tree for Name Analysis

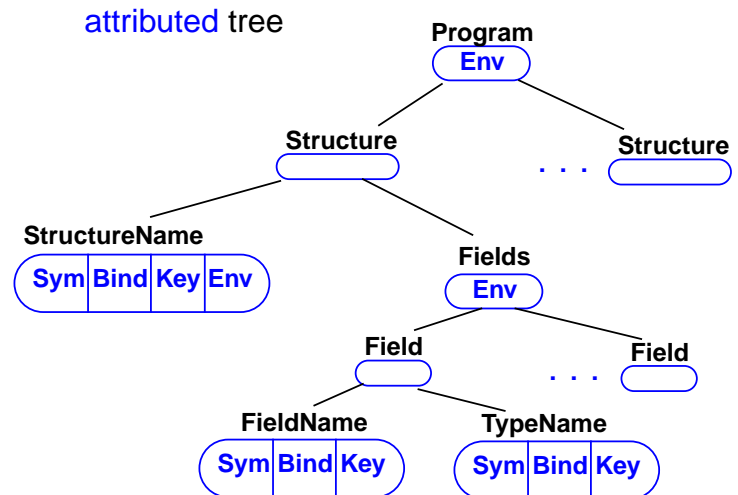
**Attributes of the tree nodes**  
describe properties of the program  
construct

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**



## Lecture Generating Software from Specifications SS 2012 / Slide 504

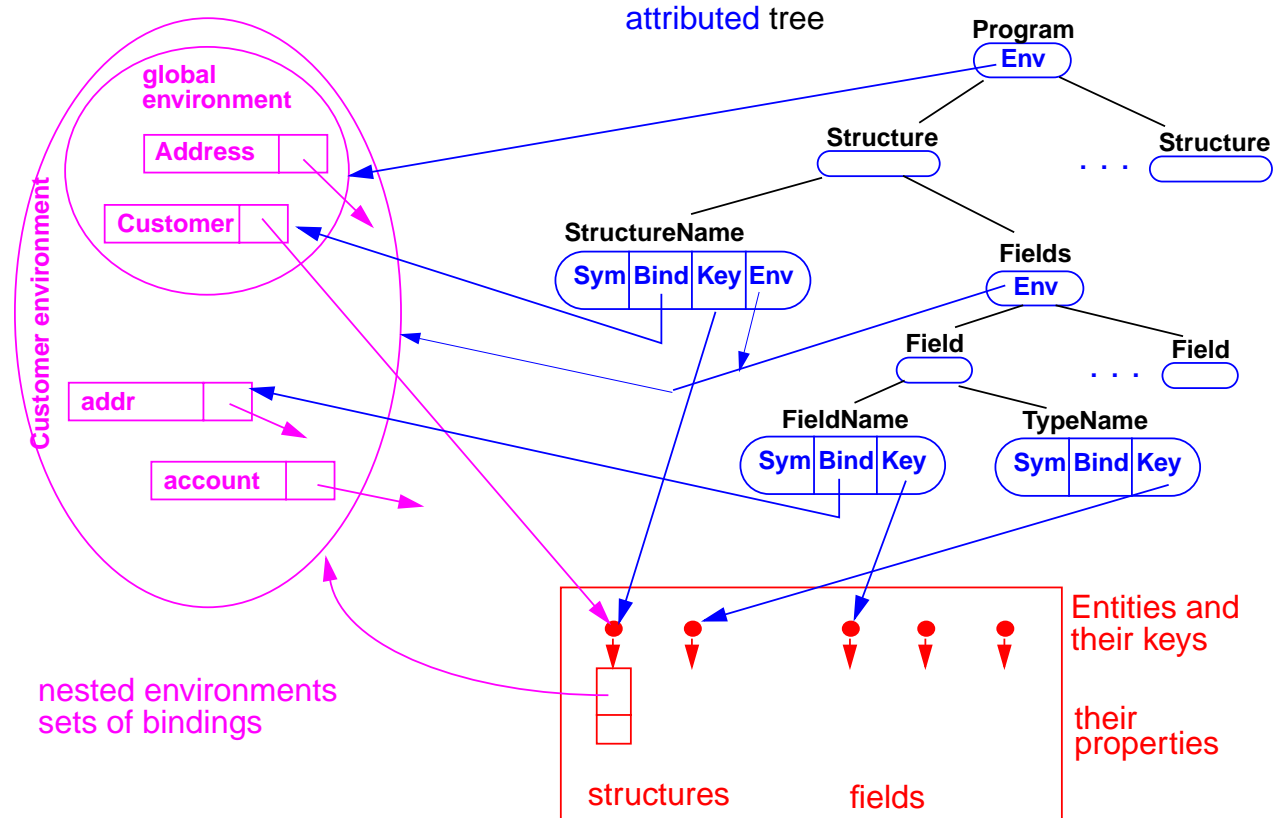
### Objectives:

Names and bindings in the tree

### In the lecture:

The topics of the slide are explained.

# Attributes, Environments, and Keys



## Lecture Generating Software from Specifications SS 2012 / Slide 505

### Objectives:

Roles of tree, bindings, and properties

### In the lecture:

The topics of the slide are explained.

## Environment Module

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

### Functions:

<b>NewEnv ()</b>	creates a new environment $e$ , that is the root of a new tree; used in <b>root context</b>
<b>NewScope (<math>e_1</math>)</b>	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 <b>range context</b>
<b>BindIdn (<math>e, id</math>)</b>	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 <b>defining occurrences</b> .
<b>BindingInEnv (<math>e, id</math>)</b>	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 <b>applied occurrences</b>
<b>BindingInScope (<math>e, id</math>)</b>	yields a binding ( $id, e, k$ ) of $e$ , if $e$ directly contains such a binding; NoBinding otherwise; e.g. used for <b>qualified names</b>

## Lecture Generating Software from Specifications SS 2012 / Slide 506

### Objectives:

Know the interface of the module

### In the lecture:

The roles of the functions are explained

## 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.

## Lecture Generating Software from Specifications SS 2012 / Slide 508

### Objectives:

Continue the running example

### In the lecture:

- refer to GSS-1.11 and GSS-5.1,
- present the abstract syntax,
- explain the identifier roles.

## Computation of Environment Attributes

**Root** of the environment hierarchy

```
SYMBOL Descriptions INHERITS RootScope END;
```

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

```
SYMBOL Fields INHERITS RangeScope END;
```

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

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

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

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

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

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

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

In that environment the field names are bound.

## Lecture Generating Software from Specifications SS 2012 / Slide 509

### Objectives:

Systematic computation of Env attributes

### In the lecture:

The topics of the slide are explained

- the Range role,
- root of nested environments created by NewEnv(), (computation can be omitted for the Grammar root).
- in the example language fields may be associated to one structure s in several structure descriptions for s.
- The property Envir stores one environment for each structure entity in the definition module.



## 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;
```

## Lecture Generating Software from Specifications SS 2012 / Slide 510

### Objectives:

Classify computations for identifier contexts

### In the lecture:

The following topics are explained:

- CLASS symbols represent computational roles.
- Establish a binding in an environment.
- Using the Range role.