7. Library of Specification Modules

A reusable specification modul

- solves a frequently occurring task,
  e.g. name analysis according Algol-like scope rules,
- provides abstract symbol roles (**`CLASS`**) with computations that contribute to the solution of the task, z. B. **`IdUseEnv`** for applied occurrences,
- contains all specifications, functions, etc. that are necessary to implement the task's solution (FunnelWeb file)
- is a member of a library of modules that support related topics,
  e.g. name analysis according to different scope rules
- has a descriptive documentation

Users

- select a suitable module,
- instantiate it,
- let symbols of their abstract syntax inherit some of the symbol roles,
- use the computed attributes for their own computations.

Basic Module for Name Analysis

Symbol roles:
Grammar root:

```
SYMBOL Program INHERITS RootScope END;
```

Ranges containing definitions:

```
SYMBOL Block INHERITS RangeScope END;
```

Defining identifier occurrence:

```
SYMBOL DefIdent INHERITS IdDefScope END;
```

Applied identifier occurrence:

```
SYMBOL UseIdent
   INHERITS IdUseEnv,ChkIdUse END;
```

Provided attributes:

```
   DefIdent, UseIdent: Key, Bind
   Program, Block: Env
```

Instantiation
in a .specs file
for Algol-like scope rules:

```
$/Name/AlgScope.gnrc:inst
```

for C-like scope rules:

```
$/Name/CScope.gnrc: inst
```

for a new name space

```
$/Name/AlgScope.gnrc
      +instance=Label
      :inst
```

Symbol roles:
LabelRootScope,
LabelRangeScope, ...

Specification Libraries in Eli

Contetnts of the Eli Documentation
Specification Module Library:

- Introduction of a running example
- How to use Specification Modules

- Name analysis according to scope rules
- Association of properties to definitions
- Type analysis tasks
- Tasks related to input processing
- Tasks related to generating output
- Abstract data types to be used in specifications
- Solutions of common problems

- Migration of Old Library Module Usage

Name Analysis, Type Analysis

Name analysis according to scope rules

- Tree Grammar Preconditions
- Basic Scope Rules, 3 variants: Algol-like, C-like, Bottom-Up
- Predefined Identifiers
- Joined Ranges (3 variants)
- Scopes being Properties of Objects (4 variants)
- Inheritance of Scopes (3 variants)
- Name Analysis Test
- Environment Module

Type analysis tasks

- Types, operators, and indications
- Typed entities
- Expressions
- User-defined types
- Structural type equivalence
- Error reporting in type analysis
- Dependence in type analysis

Association of Properties to Entities

Association of properties to definitions

- Common Aspects of Property Modules
- Count Occurrences of Objects
- Set a Property at the First Object Occurrence
- Check for Unique Object Occurrences
- Determine First Object Occurrence
- Map Objects to Integers
- Associate Kinds to Objects
- Associate Sets of Kinds to Objects
- Reflexive Relations Between Objects
- Some Useful PDL Specifications

Input and Output

Tasks related to input processing

- Insert a File into the Input Stream
- Accessing the Current Token
- Command Line Arguments for Included Files

Tasks related to generating output

- PTG Output for Leaf Nodes
- Commonly used Output patterns for PTG
- Indentation
- Output String Conversion
- Pretty Printing
- Typesetting for Block Structured Output
- Processing Ptg-Output into String Buffers
- Introduce Separators in PTG Output

Other Useful Modules

Abstract data types to be used in specifications

- Lists in LIDO Specifications
- Linear Lists of Any Type
- Bit Sets of Arbitrary Length
- Bit Sets of Integer Size
- Stacks of Any Type
- Mapping Integral Values To Other Types
- Dynamic Storage Allocation

Solutions of common problems

- String Concatenation
- Counting Symbol Occurrences
- Generating Optional Identifiers
- Computing a hash value
- Sorting Elements of an Array
- Character string arithmetic