6. Bibliothek mit Spezifikationsmodulen

Spezifikationsmodul

- löst eine häufig vorkommende Aufgabe,
  z. B. Bindung von Namen gemäß Algol-Verdeckungsregel
- bietet abstrakte Symbole (**`CLASS`**) mit vorgefertigten Berechnungen,
  Symbolrollen mit vorgefertigten Berechnungen, z. B. **`IdUseEnv`**
- enthält (als FunnelWeb-Datei) alle sonst notwendigen Spezifikationen,
  z. B. Implementierung von C-Funktionen
- ist in thematisch zusammenhängenden Bibliotheken organisiert,
  z. B. Namensanlalyse, Ausgabe mit PTG
- ist ausführlich dokumentiert

Anwender

- wählt Modul aus,
- instanziiert ihn,
- ordnet Symbolrollen einigen Symbolen seiner abstrakten Syntax zu,
- benutzt die berechneten Attribute in eigenen Berechnungen

Grundlegender Modul zur Namensanalyse

Symbolrollen:
Wurzel der Grammatik:

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

Abschnitte mit Definitionen:

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

Definierter Bezeichner:

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

Angewandte Bezeichner:

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

Benutzbare Attribute:

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

Instantiierung
in .specs-Datei
für Algol-Scope-Rules:

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

für C-Scope-Rules:

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

für neuen Namensraum

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

Symbolrollen:

```
     LabelRootScope,
     LabelRangeScope, ...
```

Struktur-Generator mit Namensmodul spezifizieren

`Fields` spielt die **Rolle eine Range**: Unterbaum begrenzt Gültigkeit darin enthaltener Definitionen

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

Wurzel der Umgebungshierarchie

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

Alle definierenden Auftreten **binden** ihren Namen im Env-Attribut des **nächst-umfassenden Range**

```
CLASS SYMBOL DefineAnIdent
   INHERITS IdentOcc, IdDefScope END;
```

Für den Bezeichner eine Bindung in den umfassenden Umgebungen suchen

Misserfolg melden

Nur ein angewandtes Auftreten

```
CLASS SYMBOL UseAnIdent
   INHERITS IdentOcc, IdUseEnv, ChkIdUse
END;
```

vergl. Folien 5.9 -5.11

Spezifikations Bibliotheken in Eli

Inhalt der Eli Dokumentation
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

Namen und Typen

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

- Basic Type Analysis
  Example for Basic Type Analysis
- Properties of Types
  Array Types
  Pointer Types
  Function Types
  Types Having Scope Properties
- Operator Identification
- Deferred Property Association

Zuordnen von Eigenschaften

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

Eingabe und Ausgabe

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

Allgemein nützliche Module

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

- Error Reports
- String Concatenation
- Counting Symbol Occurrences
- Generating Optional Identifiers
- Sorting Elements of an Array