New Features of Eli Version 4.5

Uwe Kastens

University of Paderborn
D-33098 Paderborn
FRG

A. M. Sloane

Department of Computing
Division of Information and Communication Sciences
Macquarie University
Sydney, NSW 2109
Australia

W. M. Waite

Department of Electrical and Computer Engineering
University of Colorado
Boulder, CO 80309-0425
USA

$Date: 2008/08/27 13:44:24 $

Table of Contents

New Features of Eli Version 4.5

This document gives information about new facilities available in Eli version 4.5 and those modifications made since the previous distributed Eli version 4.4 that might be of general interest. Numerous corrections, improvements, and additions have been made without being described here.

1 New type analysis modules

Type analysis is complex, even for simple languages, but it is based upon a number of language-independent concepts. The type analysis of source text in a particular language can be described in terms of those concepts, by identifying constructs playing specific roles in the type system of that language. Once that identification has been made, most of the code required to carry out the analysis can be deduced automatically. We have added attribution modules to Eli that are sufficient to describe most type analysis problems in a straightforward manner (see Section "Overview" in *Type Analysis Reference Manual*).

These modules support language concepts such as the following:

- language- and user-defined types
- name or structural equivalence of types
- language- and user-defined operators, procedures, and methods
- expressions with coercion, explicit type conversion, and overloaded operators

They encapsulate the necessary computations and the dependence relationships among them, and export a nomenclature for language constructs. Type analyzer code for a particular compiler can be created by instantiating the appropriate modules and then classifying the constructs of the given source language according to that scheme. For example, after instantiating the `Expression` module, all of the code needed to analyze an infix operator with two operands can be obtained by classifying the appropriate abstract syntax rule as a `DyadicContext`.

For a tutorial on the use of the new modules, see Section "Overview" in *Tutorial on Type Analysis*.

2 New name analysis modules

The name analysis modules that support scopes being properties and scopes being inherited have been updated mainly in oder to fit together with the updated modules for type analysis.

In the module `ScopeProp` the following role names have been changed: `ExportRange` replaces `RangeScopeProp`, `QualIdUse` replaces `IdUseScopeProp`. `ChkQualIdUse` has been added. Furthermore, the dependence pattern used for the computations of the module roles have been changed to a less restrictive one.

The module `AlgScopeProp` has been removed. The module `ScopeProp` is to be used instead.

In the module `CScopeProp` the role names are adapted to those of `ScopeProp`.

The module `BuScopeProp` has been left unchanged.

In the module `AlgInh` the following role names have been changed: `InhRange` replaces `RangeSnglInh` and `RangeMulInh`, `ChkQualIdUse` replaces `ChkIdUseScopeProp`.

A new role `ExportInhRange` combines `InhRange` and `ExportInhRange`. `RangeQualInhScope` has been deleted.

`Inheritscope` now does not require to compute `Inheritscope.OuterScope` by a user computation, `Inheritscope.ScopeKey` may be computed instead.

In the module `CInh` the role names are adapted to those of `AlgInh`.

The module `BuInh` has been left unchanged.

For further information see the Name Analysis documentation see Section “Overview” in *Name Analysis Reference Manual*. and the see Section “Overview” in *Tutorial on Name Analysis*.

3 New LIGA front-end

The front-end of the Liga attribute evaluator generator has been re-implemented. The main changes are:

- Old-style terminal attribution which had already been marked "outdated" in the previous Eli versions has now been disabled. This means that specifications like

```
ATTR c: int;

RULE p: R ::= X COMPUTE
  R.c = X.c;
END;
```

  where `X` is a terminal symbol now lead to an error message:

```
"ERROR:  ERROR: A terminal has no attributes"
```

- The key word NONTERM is no longer supported. It should be replaced by SYMBOL.

- Improved error messages, e.g.

```
"ERROR: Symbol does neither occur in syntax nor in symbol attribution"
```

  becomes

```
"ERROR: Symbol does not occur in rule or definition: IddDefScope"
```

- New protocol feature of the new liga frontend. This protocol provides information about the compound liga input specification:
  - tree and class symbols, their attributes and their inheritance relation
  - rule attributes
  - remote access constructs INCLUDING, CONSTITUENT, and CONSTITUENTS
  - inheritance of computations

  Invoke the liga frontend protocol generator by:

```
<yourspec>:feInfo:viewlist
```

Index