ASL Reference Progress
DDI 0623

Arm Architecture Technology Group

January 23, 2024

Contents

Chapter 1

Non-Confidential Proprietary Notice

This material covers both ASLv0 (viz, the existing ASL pseudocode language which appears in the Arm Architecture Reference Manual) and ASLv1, a new, experimental, and as yet unreleased version of ASL.

This material is work in progress, more precisely at pre-Alpha quality as per Arm's quality standards. In particular, this means that it would be premature to base any production tool development on this material.

However, any feedback, question, query and feature request would be most welcome; those can be sent to Arm's Architecture Formal Team Lead Jade Alglave (`jade.alglave@arm.com`) or by raising issues or PRs to the herdtools7 github repository.

Chapter 2

Disclaimer

This document is part of the ASLRef material. It is a snapshot of: `https://github.com/herd/herdtools7/commit/6dd15fe7833fea24eb94933486d0858038f0c2e8`

This material covers both ASLv0 (viz, the existing ASL pseudocode language which appears in the Arm Architecture Reference Manual) and ASLv1, a new, experimental, and as yet unreleased version of ASL.

The development version of ASLRef can be found here `~/herdtools7/asllib`.

A list of open items being worked on can be found here `~/herdtools7/asllib/doc/ASLRefProgress.tex`.

This material is work in progress, more precisely at Alpha quality as per Arm's quality standards. In particular, this means that it would be premature to base any production tool development on this material.

However, any feedback, question, query and feature request would be most welcome; those can be sent to Arm's Architecture Formal Team Lead Jade Alglave `(jade.alglave@arm.com)` or by raising issues or PRs to the herdtools7 github repository.

Chapter 3

Not implemented in ASLRef

This chapter describes what is not yet present in the executable version of ASLRef.

3.1 Syntax

3.1.1 Constrained bitvectors

Constrained bitvectors are not yet implemented in ASLRef.

For example, the following function is not understood by ASLRef:

```
func foo(x: bits({3, 5})) => integer
begin
  return Len(x) * 2;
end
```

Here the length of the formal argument `x` needs to be declared as a parameter, for example as follows:

```
func foo {N: integer {3, 5}} (x: bits(N)) => integer
begin
  return Len(x) * 2;
end
```

This also applies to bitvectors constrained by a type.

This is related to $\texttt{R}_{\texttt{FHYZ}}$, $\texttt{R}_{\texttt{VBMX}}$, $\texttt{R}_{\texttt{XVWK}}$, $\texttt{R}_{\texttt{PMQB}}$, $\texttt{R}_{\texttt{VCZX}}$, $\texttt{I}_{\texttt{TZVJ}}$, $\texttt{I}_{\texttt{GLWM}}$, $\texttt{I}_{\texttt{MTWL}}$, $\texttt{R}_{\texttt{QYZD}}$, $\texttt{I}_{\texttt{NFBN}}$, $\texttt{R}_{\texttt{LJBG}}$, $\texttt{I}_{\texttt{YBHF}}$, $\texttt{R}_{\texttt{FZSD}}$.

3.1.2 Guards

Guards are used on `case` and `catch` statements, to restrict matching on the evaluation of a boolean expression. They are not yet implemented in ASLRef.

This relates to R_{WGSY}.

3.2 Typing

Note that as it stands, the type system of ASL is not sound. Our proposal can be found at: `https://github.com/herd/herdtools7/pull/747`

This is related to I_{JSKW}.

3.2.1 Immutability of expressions in constraint ranges

The constraints of an integer type should be *constrained*, *statically evaluable expressions*, i. e. the variables used by expressions describing an integer constraint should be immutable and of a constrained type, and those expressions should be side-effect-free.

Furthermore, those constraints should not include Checked Type Conversions.

This also applies to bitvector widths.

This is related to R_{BSMK}, R_{LSNP}, R_{GHRP}, R_{LVTH}, I_{ZLZC}, R_{RHTN}, R_{XNBN}, I_{GQYG}.

3.2.2 Sound domains for under-constrained integers

Under-constrained integers have for the moment the same domain. This makes the type-system unsound, as one can assign one under-constrained integer to another.

This is related to I_{JSKW}.

3.2.3 Statically evaluable programs

Side effects analysis has not been implemented yet. This makes detection of statically evaluable subprograms impossible.

Furthermore, non-execution time subprograms, expressions, and types have not been implemented.

This is related to I_{LZCX}, I_{NXJR}, R_{CSFT}, I_{HYBT}, D_{CCTY}, I_{LYKD}, I_{ZPWM}, D_{KCKX}, I_{NTYZ}, I_{MSZT}, D_{QNHM}, I_{XYKC}, D_{ZPMF}, I_{XSDY}, D_{XRBT}, I_{WVGG}, D_{JLJD}, I_{KKDY}, D_{MTQJ}, I_{YBGL}, I_{YMRT}, I_{QJTN}, I_{GFZT}.

3.2.4 Precise implementation of parameters type-checking

The code handling parameter type-checking at a function declaration is not safe: some checks are missing.

Furthermore, the handling of formal arguments that should be considered as parameters is also not finished.

This is related to I_{BLVP}, R_{RKBV}, I_{TQGH}.

3.2.5 Restriction on use of under-constrained types

As storage types

Restrictions on use of under-constrained types as storage element types are not implemented.

This is related to $\mathtt{R}_{\mathtt{ZCVD}}$.

As expression with a constrained type

Restriction on the use of under-constrained parameters as left-hand-side of a Checked Typed Conversion is not implemented in ASLRef. For example, the following will not raise a type-error:

```
  return N as integer {0..2*N};
end

func main () => integer
begin
  assert foo ('100') == 3;
```

This is related to $\mathtt{I}_{\mathtt{TBHH}}$, $\mathtt{R}_{\mathtt{ZDKC}}$.

3.3 Semantics

3.3.1 Real exponentiation

The exponentiation operation `exp_real` has not been implemented. Note: this would construct non-rational numbers which are not supported by ASLRef, e. g. $2^{\frac{1}{2}} = \sqrt{2}$.

This is related to $\mathtt{R}_{\mathtt{BNCY}}$.

Chapter 4

Not transliterated in ASL reference documents

4.1 Typing

4.1.1 Domains

The transliteration of the notion of domains will be made more complete in the future.

4.1.2 Global storage declarations

Global storage declarations are not transliterated.

This is related to $\mathtt{R}_{\mathtt{FWQM}}$.

4.1.3 Statically evaluable expressions

The part of statically evaluable expressions that has been implemented in ASLRef (see Section 3.2.3) has not been transliterated.

Equivalence of statically evaluable expressions has been implemented in a restricted setting (see Section 3.2.3). Mainly, expressions that reduce to polynomials can be checked for equivalence. This has not been transliterated either.

This is related to $\mathtt{R}_{\mathtt{PKXK}}$, $\mathtt{D}_{\mathtt{YYDW}}$, $\mathtt{D}_{\mathtt{CWVH}}$, $\mathtt{D}_{\mathtt{HLQC}}$, and $\mathtt{I}_{\mathtt{LHLR}}$, $\mathtt{R}_{\mathtt{RFQP}}$, $\mathtt{R}_{\mathtt{VNKT}}$.

4.1.4 Polymorphism

Polymorphism in ASL is the ability to have multiple subprograms with the same name that do not have the same signature.

Although polymorphism is implemented in ASLRef, it has not yet been transliterated.

This is related to $\mathtt{D}_{\mathtt{BTBR}}$, $\mathtt{I}_{\mathtt{FSFQ}}$, $\mathtt{I}_{\mathtt{FCTB}}$, $\mathtt{I}_{\mathtt{PFGQ}}$, $\mathtt{R}_{\mathtt{PGFC}}$, $\mathtt{I}_{\mathtt{BTMT}}$.

4.1.5 Calls to setters and getters

The replacement of implicit calls to getters and setters (written for example as slices) to explicit calls to subprograms has not been transliterated. In terms of abstract syntax, this corresponds to the translation between a `E_Slice` and a `E_Call`.

This is related to $\texttt{I}_{\texttt{YYQX}}$, $\texttt{I}_{\texttt{LJLW}}$, $\texttt{I}_{\texttt{MFBC}}$.

4.1.6 Type inference from literals

Finding the type of a literal value, or a compile-time constant expression, is not yet transliterated.

This is related to $\texttt{R}_{\texttt{ZJKY}}$ and $\texttt{I}_{\texttt{RYRP}}$.

4.2 Semantics

4.2.1 Base values of types

Finding the base value of a type is not transliterated.

This is related to $\texttt{R}_{\texttt{NJDZ}}$, $\texttt{R}_{\texttt{CFTD}}$, $\texttt{R}_{\texttt{QGGH}}$, $\texttt{I}_{\texttt{WVQZ}}$, $\texttt{R}_{\texttt{GYCG}}$, $\texttt{R}_{\texttt{WKCY}}$, $\texttt{R}_{\texttt{LCCN}}$, $\texttt{R}_{\texttt{CPCK}}$, $\texttt{R}_{\texttt{ZVPT}}$, $\texttt{I}_{\texttt{QFZH}}$, $\texttt{R}_{\texttt{ZGVR}}$, $\texttt{I}_{\texttt{PGSS}}$, $\texttt{R}_{\texttt{QWSQ}}$, $\texttt{R}_{\texttt{HMRK}}$, $\texttt{R}_{\texttt{MBRM}}$, $\texttt{R}_{\texttt{SVJB}}$.

4.2.2 Operations

Operations are not transliterated from ASLRef.

This is related to $\texttt{R}_{\texttt{NCWM}}$, $\texttt{R}_{\texttt{VGZF}}$, $\texttt{R}_{\texttt{THSV}}$, $\texttt{R}_{\texttt{CRQJ}}$, $\texttt{R}_{\texttt{ZTJN}}$, $\texttt{R}_{\texttt{SVMM}}$, $\texttt{R}_{\texttt{WWTV}}$, $\texttt{R}_{\texttt{GHXR}}$, $\texttt{I}_{\texttt{NBCT}}$.

4.2.3 Standard library and primitives

The standard library is not transliterated.

This is related to $\texttt{R}_{\texttt{RXYN}}$.