Overview of ANDF Validation

Stavros Macrakis
Open Software Foundation

Applications distributed with ANDF must have consistent semantics across platforms. OSF and its partners are developing validation methods and tools which will give a high level of confidence to application vendors and users.

1. Introduction

An ANDF producer translates programs from a source language to ANDF form, and an ANDF installer translates from ANDF form to a machine language. The ANDF form itself is architecture-neutral, and is defined independently of any particular language or machine by a specification [TDF92] [Macr93a]. The producer depends on an abstract specification of an API (platform-independent header file), and the installer on its instantiation on the platform (platform-dependent token definition library).[1]

OSF and its partners are involved in several efforts to assure that all steps in this process conform to their specifications, and that the specifications themselves are well-defined. These efforts should lead to more precise specifications, and tools for checking conformance to specifications.

A precise specification and effective testing tools will increase confidence that applications distributed in ANDF will execute correctly upon installation.

1. See [Macr92a], [Macr93b].

2. The importance of validation to ANDF

Validation is an important kind of quality control for software. It is particularly important for software which interoperates with other software written to the same specification. ANDF tools, like (for instance) network protocol implementations, must work correctly with all other ANDF tools, regardless of their provenance.

OSF is thus engaged in a variety of validation activities for ANDF. The goal is to demonstrate that the technology correctly implements application programs regardless of the combination of producer, installer, and header files used.

3. End-to-end testing: language tests and sample applications

The first kind of validation testing we undertook was compiling and running C language validation suites such as Plum-Hall under the ANDF technology. These discovered some minor errors, and gave us confidence that the technology covered the full C language to first approximation.

We then proceeded to port a variety of applications to ANDF [Macr93a] and run them through the compilation chain on a variety of machines. In particular, we successfully ported major applications such as Informix's Wingz [John92] and Gnu Emacs [John91]. Work on Oracle is ongoing [Watt93a] Other experiments have been performed with Postgres [Watt92], Ghostscript [Ford92], and Cscope [Macr92].

This work has contributed to our understanding of portability issues, and has found surprisingly few bugs in the technology itself.

Although such informal testing does find real bugs, and measures the robustness of the technology when confronted with full-size software, it does not provide systematic coverage.

4. Correctness and completeness of the ANDF *specification*

The ANDF specification itself must be correct and unambiguous, since any other validation depends on this.

ANDF is currently defined by an English-language specification written by DRA. There are several ways that OSF and its partners are working to improve it: independent implementations, formal specification, and test writing.

— *Implementation*

Independent implementations of producers and installers are a good test of the specification. The Gandf project is an example, which has pointed out some difficult areas. [Ford93]

— *Formal specification*

As part of the Esprit project GLUE, DDC-I (Lyngby, Denmark) is developing a formal specification of the ANDF language using RSL in an Action Semantics style. [Hans92] [Toft93]

We expect that, besides the intrinsic value of having a formal specification, the process of creating it will bring up questions which need to be resolved in the informal specification.

— *Test writing*

The producer and installer validation efforts at OSF Research Institute (Grenoble) and its GLUE partners should also produce feedback on the specification.

Improved ANDF specification

At the end of the Esprit project, the ANDF specification will have received comments from producer writers, installer writers, formal specifiers, and test writers. The informal specification, formal specification, and tests should reflect precisely the same semantics. The cross-checking possible among the three forms should increase confidence in each one. Moreover, a side product

of these specification and test writing efforts will be an implementor's guide which points out tricky or surprising aspects of the language.

5. Independent validation of producers and installers

End-to-end tests as described in section 3. on page 2 cannot show that producers are producing correct ANDF, nor that installers are interpreting ANDF correctly. There are several kinds of problems that might remain undetected:

- Incomplete coverage in installers;
- Erroneous usage in producers;
- Target-dependent semantics.

The validation project addresses them through an installer test suite, a producer test suite, and a generalized ANDF interpreter. This work is being done at the OSF Research Institute (Grenoble) [Brou93] and its partners in the Esprit GLUE project.

6. Installer testing

Installer testing checks that installers correctly implement the ANDF language. It is essentially the same problem as a classic language validation suite such as the Ada validation capability [Good86].

End-to-end tests miss many things

Separate installer testing is necessary because end-to-end tests miss many things. This is because installers used as part of a producer/installer chain are only exposed to that ANDF which the producer produces.

Currently, there is only one operational producer of ANDF, the DRA C producer. This producer may not use all the capabilities of the ANDF language; indeed, the ANDF language has capabilities which are unlikely to be necessary for any C producer. Thus, ANDF generated by the DRA C producer cannot exercise the whole of the installer.

There may be combinations of constructs which will not be used by any one existing language. Nonetheless, these combinations must be installed correctly, because future languages may require them, or hand-written ANDF may use them. For that matter, in-line inclusion of ANDF generated from one language into ANDF generated from another language is a legal and useful optimization.

Finally, all current producers and installers are partially or wholly based on DRA engineering. This means that there may be implicit assumptions shared by producers and installers, which are not documented in the ANDF specification.

The ANDF validation suite offers systematic coverage

Unlike a producer, the validation suite is organized around the list of ANDF constructions. Each construction is tested in its normal and extreme cases. Already, preliminary testing has shown undocumented limitations in the DRA installers which were not detected using producer-generated ANDF.

Some tests are inspired by particular machine architectures' limitations.

Others are inspired by peculiar combinations of ANDF features which seem unlikely to be used by producers. Although these cases may appear pathological, they are typical of the cases which arise in certain applications such as the output of program generation systems or inlining of code written in one language into code written in another.

Others are inspired by common optimizer errors.

7. The Generalized ANDF interpreter for producer testing

Producers are harder to test independently than installers. The problem is that their output is ANDF, and some way is needed of checking whether it is the correct ANDF without depending on a particular installer. Of course, executing C validation suites compiled with a given producer on multiple installers is a good first step.

The solution that has been adopted is a generalized ANDF interpreter (GAI). Unlike all existing installers, it will have no dependence on DRA technology. As an independent implementation, it is expected not to share implicit assumptions with the DRA technology. Moreover, it will have extensive debugging facilities.

But its greatest value will be that it can create a particular execution environment which might not exist on any current installer. For instance, it might emulate a very large word length (128 bits), or a word length which is not a power of two (40 bits). Its `long`'s might be just one bit wider than its `int`'s. Its address space may have only the minimal structure defined by the ANDF specification, rather than being linear or segmented.

The GAI will also attempt to provide "perverse" but legal interpretations of programs. For instance (depending on the final form of the specification), it may return random results for overflow calculations rather than the expected modular result.

The GAI is also expected to support all legal permutations of the order of evaluation. In effect, ANDF is quite liberal in permitting reordering and indeed interleaving of expression evaluation. The GAI will take maximum advantage of this to detect dependence on evaluation order.

The GAI will be used in conjunction with C test programs and validation suites to detect incorrect assumptions made by ANDF producers.

8. API testing

The platform-independent abstract header files and the platform-dependent concrete header files used with the ANDF technology must correspond to the API they implement. To check this conformance, OSF has run the VSX test suite for XPG/3 on the ANDF technology, and also the gcc technology (as a baseline). [Watt93b]

Several classes of errors have appeared:

Specification inconsistencies and errors
: The XPG/3 specification contains some internal inconsistencies and non-ANSI C requirements. These errors were detected by tcc, where ordinary compilers could not check them (since they depend on concrete, not abstract, interface specifications). For instance, the `nl_catd` type is left unspecified, yet it is assumed that `-1` can be converted to it to designate an error.

Test suite errors
: The VSX test suite contains several non-ANSI C constructions.

Abstract header errors
: We have found two missing interfaces in the ANDF headers.

Producer errors
: The macro and function namespaces are not completely separate.

ANDF language errors
: We have not detected any errors in the specification of ANDF.

Installer errors
: Several CISC installers commit least-significant digit rounding errors on floating-point constants.

Concrete header errors
: We have not detected any concrete header (platform-dependent token library) errors.

In several cases, we have not yet determined in which category an error falls. However, it currently appears that there are more errors in the XPG/3 specification and the VSX tests than in the ANDF technology. Many of these errors were not reported by previous compilers.

9. Conclusion

OSF and its partners are taking multiple approaches to increasing confidence in the ANDF technology.

The ANDF specification itself, producers, installers, and header files are all being strengthened by the work in Cambridge, in Grenoble, and at Esprit partners' sites.

10. Bibliography

[Brou93] Fréderic Broustaut, Christian Fabre, François de Ferrière, Eric Ivanov, ANDF *Validation Suites Specification.* OSF Research Institute, Grenoble, March 1993.

[Ford92] Richard Ford, OSF RI internal report.

[Ford93] —, GANDF: *Status and Design*, OSF Research Institute, April 1993.

[Good86] John Goodenough, *The Ada compiler validation capability.* Softech, Inc., December 1986.

[Hans92] Bo Stig Hansen, Jørgen Bundgaard, *The Role of the* ANDF *Formal Specification*. DDC International A/S, Document code 202104/RPT/5, December 1992.

[John91] Andrew Johnson, OSF RI internal report.

[John92] —, OSF RI internal report.

[Macr92a] Stavros Macrakis, *The Structure of* ANDF. OSF Research Institute, October 1992.

[Macr92b] —, OSF RI internal report.

[Macr93a] —, *Porting to* ANDF. OSF Research Institute, January 1993.

[Macr93b] —, *Building Applications using* ANDF. OSF Research Institute, February 1993.

[TDF92] *TDF Specification* (December, 1992). Defence Research Agency, Malvern, U.K.

[Toft93] Jens Ulrik Toft, *Feasibility of Using RSL as the Specification Language for the* ANDF *Formal Specification*. DDC International A/S, Document code 202104/RPT/8, January 1993.

[Watt92] Thomas J. Watt, *Preliminary Report of Experience Porting Postgres with the Research Prototype ANDF Technology,* OSF Research Institute, August, 1992.

[Watt93a] —, *Porting Oracle with the ANDF Compiler Technology—A Progress Report,* OSF Research Institute, March 1993.

[Watt93b] —, *A Conformance Comparison between* ANDF *and GCC for X/Open Verification of an OSF/1*MK SS *Platform,* OSF Research Institute, March 1993.