Programming language Prolog — Draft standard – Editor's foreword

1 Introduction

The standard for Prolog will appear in two parts. This is a draft for part 1 and covers the core aspects of the language. Part 2 will contain a definition of modules.

Several expert members of WG17 have contributed significantly to this draft. In alphabetical order, they are:

a) David Bowen (Quintus Corporation) – Input/output, errors,

b) Mats Carlsson (SICS) – Syntax,

c) Klaus Daessler (Siemens) – Terminology, executing a goal,

d) Pierre Deransart (INRIA), AbdelAli Ed-Dbali (Université d'Orléans) – Formal semantics (Annex A),

e) Tony Dodd (University of Bristol) – `bagof/3` and `setof/3`,

A Working Draft (SC22 N1133) was balloted and accepted for CD registration (SC22 N1205, N1210, N1242: resolution 226). The official (and unofficial) comments accompanying the ballot results were considered at a WG17 meeting in Copenhagen, and appropriate changes have been made. A response from WG17 to all the comments is in preparation and will follow shortly.

This revised CD is the result, and is put forward for registration as a DIS.

This foreword is not part of the draft International Standard.

Roger Scowen
ISO/IEC JTC1 SC22 WG17 (Prolog) convener,
DITC/93, National Physical Laboratory
TEDDINGTON, Middlesex TW11 0LW
UNITED KINGDOM
April 7, 1993

2 A brief rationale

This draft International Standard for Prolog is based on "Edinburgh" Prolog – a *de facto* standard. Like many such standards, its definition is incomplete and implementations differ in many surprising ways. Thus for many Prolog programmers a standard will tell them not just what they already know, but also provide answers to questions they have not yet asked.

Many implementors seek to provide in a Prolog predicate the ability to perform all possible tasks concerned with that functionality in a single call. For a programmer, this generality may be convenient. But for a standardizer it is a nightmare trying to define the precise effect of erroneous cases. This draft International Standard therefore normally aims to provide any particular functionality in the simplest possible way, but with sufficient power to provide portability to those who need it.

WG17 is aware that every time an International Standard defines a feature as implementation dependent, the standardizers have failed to provide portable functionality. The number of such features has therefore been minimized.

3 Features of draft standard Prolog

Many parts of the draft are both close to existing practice and implementable efficiently. For example:

— Syntax (6) is close to traditional Edinburgh Prolog syntax, specifies which terms are equivalent, identifies ambiguous cases and defines their meaning.

— Unification (7.3, 8.2) resolves the problem of the occurs-check. This is not normally implemented, but the failure to do so leaves Prolog with an unclear, ambiguous semantics where steadfast libraries are impossible (see the example 7.3.4.1).

— Arithmetic (7.9, 8.6, 8.7, 10.1) is based on a language-independent arithmetic standard designed by experts, can be implemented efficiently on almost any computer processor, and is accurate so that programmers know their answers can be trusted.

— Execution of a goal (7.7, 7.8).

Other features are uncontroversial and agreed, for example: type-testing (7.1, 8.3), term comparison (7.2, 8.4), clause retrieval (8.8), clause creation (8.9), more complicated evaluable functors (10.2-10.4).

A few features are novel, but have been accepted and are stable, for example: flags (7.11, 8.17), error processing (7.12), atom processing (8.16), converting characters read as data (8.14).