Klaus Däßler
DIN NI22 AK17 Prolog
Germany

Dresden, **11.Nov.2008**

German comments, proposals on and a modification of

ISO/IEC DTR 13211-3:2006, April 23, 2008

0. General

DIN AK17 has followed the discussions on the difficulties of definite clause grammars. We identified three major topics. One is the restriction to type "list" for terminal sequences, second (sometimes unintentional) misuse of grammar rule expansion to create clauses for existing BIPs and third the restriction to parsing jobs, although DCGs are employed as least as oftenly for generating jobs.
Although quite late, we've created now some more general concepts and thereby solved most of these problems and discussions by slight changes to the already very good draft of Paulo. Additionally we fixed some minor typos and textual flaws.
The result comes with a proposed modified version of this DTR. The actual version is November 11, 2008.
The rest of the paragraph numbering follows the section numbering of the accompanying modified DTR. Here you find all important modifications described.

3 Definitions

3.1 removed 2nd line (reference to phrase/2/3), because it's worrying the reader.

3.4 introduced the concept of generating (a terminal sequence), to avoid the unsymmetric restriction to parsing in this draft, which is misleading.

3.5 used singular for *grammar-body-alternative.* Alternative is one or more concurrent clauses. Compared with choice its more unique.

3.5.1 removed the definition of cut (!), because there is no difference to 1321-1

3.9 generalized the concept of grammar-body-nonterminal to simply non-terminal (of a grammar), --> 3.14

3.17 introduced the concept of parsing as opposite to generating.

3.19 renamed "sequence of terminals" to "terminal-sequence" for mor ease of reading. Simplified its description, because the description of a list, including empty list and partial list, is part of 13211-1. Thereby avoided all discussions on partial lists versus lists. The terminal-sequence is a very central concept thereby, taking in respect the ideas of DCG's originator, R. O'Keefe, which correctly states, that not lists but terminal sequences are handled with grammars, but on the other side Paulo correctly states, that lists >are< used in the grammar concept to represent terminal-sequences.

3.20 introduced the concept of "comprehensive terminal-sequence" to avoid the misleading input-list and output-list. Input and output hold only for parsing, not for generating. cf. the concept of "remaining terminal-sequence". Both concepts have proven as quite valuable.

3.21 introduced the concept of "remaining terminal-sequence" for the "output list".

6.1.3 replaced the "list of terminals" by "terminal-sequence" everywhere in the draft.

7.11 We propose for the DTR

Prolog flag *terminal-sequence* with values {*list, term*}

This allows a portable Prolog program to inquire, if the both terminal sequence variables denote lists or any term, as in SWIPL e.g.

7.12
Consequently we propose for the BIP phrase/3:
- if flag terminal-sequence is set to *term* -

error-type "*terminal-sequence*" instead of "*list*", thereby freeing macro based systems from simulating lists for tests of correctness in phrase/3. Yet terminal-sequences are still defined as lists. We have to decide in Udine on this once more.

These proposals are not yet included in the accompanying German DTR modification paper.

7.14.1 included empty terminal-sequence as empty list containing zero terminals.

7.14.3 more precise description of the so called push-back list. Replaced the somewhat odd concepts of input list and output list, dedicated to parsing only.

7.14.5 clarified concept "processing" wrt. expanding = preparation for execution

7.14.6 somewhat clarified control constructs and bips. (as done by Paulo elsewhere already).

7.14.7 resolved the problem of unintenionally creating clauses for built-in predicates, whose static behaviour may differ between implementations, by declaring the use of their names as nonterminals as implementation dependent. Thereby all problems are solved and standardization of dirty tricks is has been avoided now. Users of SWIPL may continue such usage if they want.

7.14.7 made the error-type of missing grammar rule mandatory.

7.14.7 Notes: made section symmetric wrt. parsing and generating

8.1.1.1 made section symmetric wrt. parsing and generating.

8.1.1.4 symmetric wrt parsing and generating

8.1.1.5 addditional examples for more clarity

10.1 introduced comprehensive resp. remaining terminal-sequence instead of input/output lists. Replaced parsing by more general wording.

10.3 - 10.5 used terminal-sequence instead of list throughout

11.1 used comprehensive resp. remaining terminal-sequence instead of input/output list.

Done.