SWI-Prolog SGML/XML parser

Jan Wielemaker
SWI,
University of Amsterdam
The Netherlands
E-mail: jan@swi.psy.uva.nl

May 24, 2000

Abstract

Markup languages are an increasingly important method for data-representation and exchange. This article documents the package *sgml2pl*, a foreign library for SWI-Prolog to parse SGML and XML documents, returning information on both the document and the document's DTD. The parser is designed to be small, fast and exible.

Contents

1 Introduction

Markup languages have recently regained popularity for two reasons. One is document exchange, which is largely based on HTML, an instance of SGML and the other is for data-exchange between programs, which is often based on XML, which can be considered simpli ed and rationalised version of SGML.

James Clark's SP parser is a exible SGML and XML parser. Unfortunately it has some drawbacks. It is very big, not very fast, cannot work under event-driven input and is generally

```
            [],
            [ element(head,
                      [],
                      [ element(title,
                                [],
                                [ 'Demo'
                                ])
                      ]),
              element(body,
                      [],
                      [ '\n',
                        element(h1,
                                [ align = center
                                ],
                                [ 'This is a demo'
                                ]),
                        '\n\n',
                        element(p,
                                [],
                                [ 'Paragraphs in HTML need not be closed.\n'
                                ]),
                        element(p,
                                [],
                                [ 'This is called `omitted-tag\' handling.'
                                ])
                      ])
            ])
].
```

```
load_html_file(File, Term) :-
        dtd(html, DTD),
        load_structure(File, Term,
                       [ dtd(DTD),
                         dialect(sgml)
                       ]).
```

3 Predicate Reference

3.1 Loading Structured Documents

SGML or XML files are loaded through the common predicate `load_structure/3`. This is a predicate with many options. For simplicity a number of commonly used shorthands are provided: `load_sgml_file/2`, `load_xml_file/2`, and `load_html_file/2`.

load_structure(+*File, {ListOfContent, +Options*)

Load the XML file *File* and return the resulting structure in *ListOfContent*. *Options* is a list of options controlling the conversion process.

A proper XML document contains only a single toplevel element whose name matches the document type. Nevertheless, a list is returned for consistency with the representation of element content. The *ListOfContent* consists of three types:

Atom

Atoms are `CDATA`. Note is possible SWI-Prolog, as is no length-limit on atoms and atom garbage collection is provided.

element(*Name, ListAttributes, ListOfContent*)

Name

```
[ entity('Alpha'), ' < ', entity('Beta') ]
```

This is a special case of `entity`(*Code*), intended to handle special symbols by their name rather then character code.

sdata(*Text*)

space(*sgml*)

In SGML, newlines at the start and end of an element are removed.[1] This is the default

White-space handling
White space mode is set to `preserve`. In addition to setting white-space handling at the toplevel the XML reserved attribute `<xml:space>`

3.4 DTD-Handling

element(*Name, Omit, Content*

notations(*ListOfNotations*)
: Returns a list holding the names of all `NOTATION` declarations.

notation(*Name, File*)
: Yields the declared file for from a `NOTATION` declaration.

3.4.1 The DOCTYPE declaration

free_sgml_parser(+*Parser*)

begin

```
set_sgml_parser(Parser, dialect(xml)),
sgml_parse(Parser,
           [ source(In),
             call(begin, on_begin),
             call(end, on_end)
           ]),
```

```
                    [ source(In),
                      call(begin, index_on_begin)
                    ]),
        close(In).

index_on_begin(_Element, Attributes, Parser) :-
        memberchk('r:id'=Id, Attributes),
        get_sgml_parser(Parser, charpos(Offset)),
        get_sgml_parser(Parser, file(File)),
        assert(location(Id, File, Offset))[(get1,)]TJ --59 Tf 0 0 Td7 -27.098 0.24index_Thefo
                memberon(Id, File, Offset))[(0 -13.549 Td[5emberonad_structurile) (File,)Term
```