SWI-Prolog SGML/XML parser

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\n',
```

```
load_html_file(File, Term) :-
        dtd(html, DTD),
        load_structure(File, Term,
                       [ dtd(DTD),
```

```
[ entity(alpha), ' < ', entity(beta) ]
```

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

The *Options* list controls the conversion process. Currently de ned options are:

dtd(*?DTD*)
: Reference to a DTT object. If speci ed, the `<!DOCTYPE ...>` declaration is ignored and the document is parsed and validated against the provided DTD. If provided as a variable, the implicitly created DTD is returned. See section 3.4.

dialect(*+Dialect*)
: Specify the parsing dialect. Supported are `sgml` (default), `xml` and `smlns`. See section 3.2 for details on the di erences.

 le(*+Name*)
: Sets the name of the le on which errors are reported. Sets the linenumber to 1.

line(*+Line*)
: Sets the starting line-number for reporting errors.

max_errors(*+Max*)
: Sets the maximum number of errors. If this number i(the)-4reached, an exception of the format below is raised. The default is 50.

```
54x0  10.909 2I-28(eciafl/F2)n 0 1 5.45mit4(The)-361.807 0 maxCode0454 0 cm BT70
```

```
...,
absolute_file_name(dtd(Type),
                   [ extensions([dtd]),
                     access(read)
```

dtd(*?DTD*)

If speci ed with an initialised DTD, this DTD is used for parsing the document, regardless of the document prologue. If speci ed using as a variable, a reference to the created DTD is returned. This DTD may be created from the document prologue or build implicitely from the document's content.

free_sgml_parser(+*Parser*)

Destroy all resources related to the parser. This does not destroy the DTD if the parser was created using the dtd(*DTD*) option.

set_sgml_parser(+*Parser, +Option*)

Sets attributes to the parser. Currently de ned attributes:

le(*File*)

source(+*Stream*)

An input stream that is read. Either this option or the goal(*Goal*) option must be provided.

goal(+*Goal*)

Goal is a callable term. The predicate sgml_parse/2 opens an output stream to the parser and invokes call(*Goal, Stream*), where *Goal*

```
                [ document(Content),
                  parse(content)
                ]),
        process_rdf_description(element(Tag, Attr, Content)).
```

4 Processing Indexed Files

In some cases applications which process small portions of large SGML, XML or RDF