Connectivity

SMTLIB

The SMTLIB package provides support for converting Maple expressions to input in the SMT-LIB language format.

SMT-LIB is an interface language frequently used by programs designed to solve SMT (*Satisfiability Modulo Theories*) problems.

The SMTLIB[ToString] command converts a Maple expression into SMT-LIB input which can then be fed as input into an SMT solver program. By default, the generated SMT-LIB script performs a simple satisfiability query.

Example 1: Generate an SMT-LIB script testing whether a logical formula has a satisfying assignment (that is, an assignment of values making the formula true).

```
expr := Import("example/3sat.cnf", base = datadir, variables = [V1, V2, V3])
```

$$expr := (V1 \text{ or } V3 \text{ or not } V2) \text{ and } (V2 \text{ or } V3 \text{ or not } V1) \text{ and } (V3 \text{ or not } V1 \text{ or not } V2)$$

```
with(SMTLIB)
```

$$[ToString]$$

```
ToString( expr )
```

```
"(set-logic QF_UF)
(declare-fun V1 () Bool)
(declare-fun V2 () Bool)
(declare-fun V3 () Bool)
(assert (and (or V1 V3 (not V2)) (or V2 V3 (not V1)) (or V3 (not V1) (not V2))))
(check-sat)
(exit)
"
```

If an explicit satisfying assignment is desired, we can optionally generate a script requesting the SMT solver to produce one (here, a set of values for $V1$, $V2$, $V3$ for which the formula is true).

$$ToString(\ expr, getvalue = [V1, V2, V3]\ )$$

```
"(set-option :produce-models true)
(set-logic QF_UF)
(declare-fun V1 () Bool)
(declare-fun V2 () Bool)
(declare-fun V3 () Bool)
(assert (and (or V1 V3 (not V2)) (or V2 V3 (not V1)) (or V3 (not V1) (not V2))))
(check-sat)
(get-value (V1 V2 V3))
(exit)
"
```

In addition to Boolean-valued variables, SMT-LIB variables may also be numeric. The generated SMT-LIB requests a nontrivial solution in positive integers to the equation $w^3 + x^3 = y^3 + z^3$. (When such a solution exists, the integer $w^3 + x^3$ is known as a *taxicab number*, the smallest of which is 1729.)

$$ToString(\ \{w^3 + x^3 = y^3 + z^3, w > 0, x > 0, y > 0, z > 0, w \neq y, w \neq z\}, getvalue = [w, x, y, z]\ )$$

```
"(set-option :produce-models true)
(set-logic QF_NIA)
(declare-fun w () Int)
(declare-fun x () Int)
(declare-fun y () Int)
(declare-fun z () Int)
(assert (and (= (+ (* w w w) (* x x x)) (+ (* y y y) (* z z z))) (distinct w y) (distinct w z) (< 0 w)
    (< 0 x) (< 0 y) (< 0 z)))
(check-sat)
(get-value (w x y z))
(exit)
"
```

URL

The existing URL package has been extended with three new commands: URL[Delete], URL[Head], and URL[Put]. These correspond directly to the HTTP (Hypertext Transfer Protocol) commands DELETE, HEAD, and PUT, respectively.

The inclusion of **Delete**, **Head**, and **Put** in the URL package, along with the existing URL[Get] and URL[Post], enables Maple code to make use of **REST** (*REpresentational State Transfer*) web interfaces. These interfaces offer access and manipulation of online resources by use of a uniform set of stateless operations based on HTTP.

Example: The **Head** command issues an HTTP HEAD request. This requests a response from the server identical to that of a GET request, but without the body of the response. Among many other uses, this command may be used to confirm that a remote server is online and accepting requests.

```
with(URL)
```

[*Construct*, *Delete*, *Escape*, *Get*, *Head*, *Parse*, *Post*, *Put*, *Unescape*]

```
Head( "http://www.maplesoft.com/", output = [headers, content, code])
```

```
table( ["Date" = "Mon, 23 Jan 2017 22:17:11 GMT", "X-Powered-By" = "ASP.NET",
    "X-AspNet-Version" = "2.0.50727", "Cache-Control" = "private", "Content-Length" = "53676",
    "Content-Type" = "text/html; charset=utf-8", "Set-Cookie"
     = "ASP.NET_SessionId=b0eexnvcsrscup45x14dftid; path=/; HttpOnly", "Server"
     = "Microsoft-IIS/8.5"]), "", 200
```

YAML

The new YAML package allows import and export of files and strings in the YAML format, a lightweight file format for exchanging structured data.
The commands YAML[ParseFile] and YAML[ParseString] parse YAML files and strings, respectively, to Maple expressions. The inverse operation, conversion of Maple structures to YAML, is provided by YAML[ToString].

The general-purpose commands Import and Export have also been extended to support YAML.

Example: Import YAML data encoding the mailing address of Maplesoft headquarters.

```
yamlFile := FileTools:-JoinPath( ["example/address.yaml"], base = datadir)
```

```
T := YAML:-ParseFile( yamlFile )
```

```
T :=table( ["address" = table( ["city" = "Waterloo", "country" = "Canada", "province" = "ON",
    "streetAddress" = "615 Kumpf Drive", "postalCode" = "N2V 1K8"]), "companyName"
     = "Maplesoft", "phoneNumbers" = [table( ["type" = "local", "number"
     = "+1 (519) 747-2373"]), table( ["type" = "toll-free", "number" = "+1 (800) 267-6583"]),
    table( ["type" = "fax", "number" = "+1 (519) 747-5284"]) ], "founded" = 1988])
```

```
T["companyName"]
```

"Maplesoft"

```
T["address"]["city"], T["address"]["country"]
```

"Waterloo", "Canada"