Package ‘rpicosat’

November 15, 2017

Type Package

Title R Bindings for the ‘PicoSAT’ SAT Solver

Version 1.0.1

Description Bindings for the ‘PicoSAT’ solver to solve Boolean satisfiability problems (SAT).

The boolean satisfiability problem asks the question if a given boolean formula can be TRUE;
i.e. does there exist an assignment of TRUE/FALSE for each variable such that the whole for-
mula is TRUE?


License MIT + file LICENSE

Encoding UTF-8

LazyData true

ByteCompile true

RoxygenNote 6.0.1

URL https://github.com/dirkschumacher/rpicosat

BugReports https://github.com/dirkschumacher/rpicosat/issues

NeedsCompilation yes

Depends R (>= 3.4.0)

Suggests testthat, covr

Author Dirk Schumacher [aut, cre],
Armin Biere [ctb, cph] (Author and copyright holder of included PicoSAT
code)

Maintainer Dirk Schumacher <mail@dirk-schumacher.net>

Repository CRAN

Date/Publication 2017-11-15 22:48:38 UTC

R topics documented:

  picosat_added_original_clauses.................. 2
  picosat_decisions.............................. 2
2

\texttt{picosat\_decisions}

\begin{itemize}
\item \texttt{picosat\_propagations} ......................................................... 3
\item \texttt{picosat\_sat} ................................................................. 3
\item \texttt{picosat\_seconds} ......................................................... 4
\item \texttt{picosat\_solution\_status} .............................................. 5
\item \texttt{picosat\_variables} ....................................................... 5
\item \texttt{picosat\_visits} ............................................................. 6
\end{itemize}

\textbf{Index} 7

\begin{itemize}
\item \texttt{picosat\_added\_original\_clauses}
\end{itemize}

\textit{The number of original clauses}

\begin{description}
\item[Description] The number of original clauses
\item[Usage] \texttt{picosat\_added\_original\_clauses(x)}
\item[Arguments] \begin{description}
\item[$x$] a picosat solution object
\end{description}
\item[Value] an integer vector of length 1
\end{description}

\begin{itemize}
\item \texttt{picosat\_decisions} \textit{The number of decisions during a search}
\end{itemize}

\begin{description}
\item[Description] The number of decisions during a search
\item[Usage] \texttt{picosat\_decisions(x)}
\item[Arguments] \begin{description}
\item[$x$] a picosat solution object
\end{description}
\item[Value] an integer vector of length 1
\end{description}
**picosat_propagations**  
*The number of propagations during a search*

**Description**
The number of propagations during a search

**Usage**
```
picosat_propagations(x)
```

**Arguments**
- `x`  
a picoSAT solution object

**Value**
an integer vector of length 1

---

**picosat_sat**  
*Solve SAT problems with the 'PicoSAT' solver*

**Description**
The solver takes a formula in conjunctive normal form and finds a satisfiable assignment of the literals or returns that the formula is not satisfiable.

**Usage**
```
picosat_sat(formula, assumptions = integer(0L))
```

**Arguments**
- `formula`  
a list of integer vectors. Each vector is a clause. Each integer identifies a literal. No element must be 0. Negative integers are negated literals.
- `assumptions`  
an optional integer vector. Assumptions are fixed values for literals in your formula. Each element corresponds to a literal. Negative literals are FALSE, positive TRUE.

**Value**
a data.frame with two columns, variable and value. In case the solution status is not PICOSAT_SATISFIABLE the resulting data.frame has 0 rows. You can use `picosat_solution_status` to decide if the problem is satisfiable.
References

PicoSAT version 965 by Armin Biere: [http://fmv.jku.at/picosat/](http://fmv.jku.at/picosat/)

Examples

```r
# solve a boolean formula
# (not a or b) and (not b or c)
# each variable is an integer
# negations are negative integers
formula <- list(
  c(-1L, 2L),
  c(-2L, 3L)
)
res <- picosat_sat(formula)
picosat_solution_status(res)

# set a variable to a fixed value
# e.g. a = TRUE and b = TRUE
res <- picosat_sat(formula, assumptions = c(1L, 2L))
picosat_solution_status(res)

# get further information about the solution process
picosat_variables(res)
picosat_added_original_clauses(res)
picosat_decisions(res)
picosat_propagations(res)
picosat_visits(res)
picosat_seconds(res)
```

### picosat_seconds

- **Description:** Time spent in `picosat_sat`

- **Usage:**
  ```r
  picosat_seconds(x)
  ```

- **Arguments:**
  - `x` a picosat solution object

- **Value:** a numeric vector of length 1
### picosat_solution_status

*Get the solution status*

**Description**

Get the solution status

**Usage**

```r
cicosat_solution_status(x)
```

#### Arguments

- `x` a solution from the solver

#### Value

character either `PICOSAT_SATISFIABLE`, `PICOSAT_UNSATISFIABLE` or `PICOSAT_UNKNOWN`

---

### picosat_variables

*The number of variables in a model*

**Description**

The number of variables in a model

**Usage**

```r
cicosat_variables(x)
```

#### Arguments

- `x` a picosat solution object

#### Value

an integer vector of length 1
picosat_visits  The number of visits during a search

Description
The number of visits during a search

Usage
picosat_visits(x)

Arguments
x  a picosat solution object

Value
an integer vector of length 1
Index

picosat_added_original_clauses, 2
picosat_decisions, 2
picosat_propagations, 3
picosat_sat, 3
picosat_seconds, 4
picosat_solution_status, 5
picosat_variables, 5
picosat_visits, 6