SQL2PDDL

Generated by Doxygen 1.7.6.1

Tue Jun 17 2014 11:42:03

Contents

Chapter 1

SQL2PDDL.cpp

Main driver that writes a PDDL problem file

Input and Output:

- Needs connection to SQL database and mapping.txt file
- Need to set environment variables for SQL connection
- writes PDDL problem file to problem-[name of problem].pddl

The code queries the SQL database by calling Database/DAO.cpp, which is part of the autogenerated C++ suite.

Input mapping.txt file is filled only with entries that look like:

```
(kit-has-kitTray ?kit - Kit ?kittray - KitTray)
A Kit : ( E hasKit_KitTray ) print _NAME hasKit_KitTray
```

The first line is copied from the domain file, listing the predicate or function prototype, and the second line describes where this information lies in the database. There must be a blank line between each pair of content lines - this is hard-coded into the parser currently. It uses set theory syntax - for example, this example would be read "for all rows in the Kit table, such that the attribute hasKit_KitTray exists, print _NAME and has-Kit_KitTray". Note that currently the exists filter only supports checking the existence of attributes in the main table (next to the "for each")

Another example that looks like this:

```
(endEffector-has-no-heldObject ?endeffector - EndEffector)
A EndEffector : ( _NAME ! c SolidObject/hadByHeldObject_EndEffector ) print
     _NAME
```

This would be read as "For all rows in EndEffector such that _NAME is not a member of the set queried from SolidObject/hadByHeldObject_EndEffector, print _NAME". This filter supports is a member of and is not a member of. The most complicated supported input looks like:

```
 (kit-has-physicalLocation-refObject-lbwk ?kit - Kit ?largeboxwithkits -
      LargeBoxWithKits)
 A PhysicalLocation : ( _NAME > SolidObject/hasSolidObject_PrimaryLocation :
      SolidObject/_NAME \c Kit/_NAME ) && ( hasPhysicalLocation_RefObject c
      LargeBoxWithKits/_NAME ) print SolidObject/_NAME hasPhysicalLocation_RefObject
```

As one can see, the parser also supports the operator "&&", where it filters entries using both statements around it. Additionally, note the first statement, which includes the ">" operator. The entire statement can be read as "for all rows in PhysicalLocation, such that (_NAME contains the same string as an entry in SolidObject/hasSolidObject-_PrimaryLocation in the same row as a SolidObject/_NAME entry which is a member of Kit/_NAME) AND (that same row in PhysicalLocation has for the attribute hasPhysical-Location_RefObject a member of the set LargeBoxWithKits/_NAME), print SolidObject/-_NAME and PhysicalLocation/ hasPhysicalLocation_RefObject". If a table and forward slash are left off the path to an attribute, the default table is the main table

Author

Christopher Lawler

Version

1.0

Date

3 Jun 2014

Chapter 2

Class Index

2.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

Chapter 3

Class Documentation

3.1 PDDLWriter Class Reference

Worker class that generates problem file.

`#include <PDDLWriter.h>`

Public Member Functions

- PDDLWriter ()

  Constructor.
- void writeObjectSection (std::ofstream &, std::ifstream &)

  Writes the header and object section of the new problem file by copying it from the original problem file - lists of objects are updated from database.
- void writeGoalSection (std::ofstream &, std::ifstream &)

  Writes the goal section of the new problem file by copying it directly from the original problem file.
- void writePredicate (std::ofstream &, std::string, std::vector< std::string >)

  writes quieried values in correct predicate syntax
- void writeFunction (std::ofstream &, std::string, std::vector< std::string >)

  writes quieried values in correct function syntax
- void writeGroupPredicates (std::ofstream &, std::string, std::vector< std::vector< std::string > >)

  writes all the predicates related to each predicate prototype/line in input file
- void writeGroupFunctions (std::ofstream &, std::string, std::vector< std::vector< std::string > >)

  writes all the functions related to each function prototype/line in input file
- void parseAndWriteInit (std::ofstream &, std::ifstream &, std::ifstream &)

  writes all the predicates related to each predicate prototype/line in input file

3.1.1 Detailed Description

Worker class that generates problem file.

Author

Christopher Lawler

Version

1.0

Date

3 Jun 2014

3.1.2 Member Function Documentation

3.1.2.1 void PDDLWriter::parseAndWriteInit (std::ofstream & *problemFile,* std::ifstream & *oldProblemFile,* std::ifstream & *mappingText*)

writes all the predicates related to each predicate prototype/line in input file

This method loops through the input file. It takes the name of the predicate from the first line, and sends the information from the database information line to a parser which queries the relevant information. It sends the resulting columns to writeGroupPredicates or writeGroupFunctions, respectively. It stops parsing when it sees the line ":end".

Data from the database is passed between functions in a two-dimensional vector of strings, where the top-level vector is a vector of columns pulled out of the database. The structure looks as follows: columns = {

column1:	column2:
_NAME 1	attribute 1
_NAME 2	attribute 2
_NAME n	attribute n

}

Parameters

outFile	the problem file
mappingText	the input file

3.1.2.2 void PDDLWriter::writeFunction (std::ofstream & *outFile,* std::string *functionName,* std::vector< std::string > *args*)

writes quieried values in correct function syntax

Parameters

outFile	the problem file
function-Name	read in from the input file
args	the list of tokens that will follow the function name

3.1.2.3 void PDDLWriter::writeGroupFunctions (std::ofstream & *outFile,* std::string *functionName,* std::vector< std::vector< std::string > > *columns*)

writes all the functions related to each function prototype/line in input file

Parameters

outFile	the problem file
function-Name	read in from the input file
columns	contains the relevant columns taken from the database, sorted and filtered

3.1.2.4 void PDDLWriter::writeGroupPredicates (std::ofstream & *outFile,* std::string *predicateName,* std::vector< std::vector< std::string > > *columns*)

writes all the predicates related to each predicate prototype/line in input file

Parameters

outFile	the problem file
predicate-Name	read in from the input file
columns	contains the relevant columns taken from the database, sorted and filtered

3.1.2.5 void PDDLWriter::writePredicate (std::ofstream & *outFile,* std::string *predicateName,* std::vector< std::string > *args*)

writes quieried values in correct predicate syntax

Parameters

outFile	the problem file
predicate-Name	read in from the input file
args	the list of tokens that will follow the predicate name

The documentation for this class was generated from the following files:

- /home/chris/Documents/APRS_workspace/SQL2PDDL/src/PDDLWriter.h
- /home/chris/Documents/APRS_workspace/SQL2PDDL/src/PDDLWriter.cpp