Web-SynDic

Web System for Demonstrating the Syntactic Algorithms for Solving Linear Equations in Nonnegative Integers (Nonnegative Linear Diophantine Equations)

Implementation Document

Department of Computer Science, Petrozavodsk State University, Russia

15th November 2004

Contents

1 General Description

This document is focused on changes to the design specification that were necessary for the implementation phase. We also give short descriptions of interesting algorithms that are not already described in the design specification.

2 Web Server Subsystem

2.1 Web pages

- In the user interface implementation, “bounds on user limits” term used instead of the “default limits”.
- Limits and Default Limits forms are implemented in separate jsp-files (limits.jsp and limits-default.jsp) instead of using single limits.jsp.

2.2 Parsers

2.2.1 class ANLDEParser

Class ANLDEParser is used by web server to translate single ANLDE systems and ANLDE system sets from traditional mathematical style to internal format. Public member ANLDERequest parseANLDE(String source, boolean set, Limits limits) is used to translate ANLDE system (if *set* is false) or ANLDE system set (if *set* is true) to the internal format (instance of ANLDERequest class) from given *source* string according to *limits* specified. ANLDEFormatException is thrown in case of parsing error or when the ANLDE system doesn't conform to the limits.

2.2.2 class ANLDEFormatException

ANLDEFormatException is thrown by ANLDEParser.parseANLDE() when it fails to parse ANLDE system (set). Public member getMessage() is used to get error description and public method getLine() is used to get line of text where error is found.

2.2.3 ANLDE system parser

ANLDE system parser is constructed from two files containing specifications for syntax analyzer and lexical analyzer. Files with specifications are translated to Java source files using Byacc/J and JFlex. The generated class ANLDESystemParser is used by class ANLDEParser to translate single ANLDE systems. The generated class ANLDESystemLexer is used by ANLDESystemParser as a scanner.

YACC specification:

```
input:                          /* empty input */
        | input line
;
line:   NL
        | left '=' right NL     /* a line containing an equation */
        | left '=' INT NL       /* a line containing an equation with zero right-
                                   hand part */ ;
;
left:   VAR                     /* rule for left-hand part of ANLDE */
        | left '+' VAR
;
right:  item                    /* rule for right-hand part of ANLDE */
```



```
        | right '+' item
;
item:   VAR                     /* coefficient before variable equals to 1 */
        | INT '*' VAR           /* coefficient before variable equals to INT */
;
```

Flex specification:

```
INT = [0-9]+                    /* integer value */
VAR = [A-Za-z][A-Za-z0-9]*      /* variable name */
NL = [\r\n]+                    /* newline */
```

If an error has been matched while parsing, ANLDEFormatException containing error description and line number is thrown. If during the translation coefficients, equations, or unknowns limit has been exceeded, ANLDEFormatException is also thrown.

3 Algorithm Server subsystem

3.1 class ANLDESystem

Private fields int n and int m removed. All public methods set* were removed because of object's immutable structure. For the same reason constructor ANLDESystem () was replaced by constructor ANLDESystem(String[], int[], int[][]). Public methods int getN() and int getM() were replaced by corresponding public methods int getNumEqautions() and int getNumUnknowns(). Public method String toString() was introduced (i. e. for debugging).

3.2 class Generator

Parser checks the limits for generating ANLDE systems. Generator do not check number of solutions when it generating ANLDE system. If the set of ANLDE systems has at least one ANLDE system which has incorrect limits, we reject whole set. Constructor Generator() was replaced by constructor Generator(String absPath).

3.3 class SolverMetrics

Added field String errorMessage; for storing error message like "algorithm failure" etc. Added methods void setError(String) and String getError().

3.4 class Solution

Private field int q removed. Public method void setQ(int) removed. Added constructor Solution(int[][]).

3.5 class Solver

Added private field String solpath for storing solver path. Also added protected methods setSolPath(String) and getSolPath(). Constructor Solver() was replaced by constructor Solver(String absPath).

3.6 class LpSolveSolver

For using solving algorithm we make simple optimization task:

$$\begin{aligned}
&\sum_{i=1}^{m} x_i \to \min \\
&\sum_{i=1}^{m} x_i(A[j][i] - E(I)[j][i])x_i = 0, \quad j = \overline{1,n} \\
&\sum_{i=1}^{m} x_i \geq 1 \\
&x_i - integer, \quad i = \overline{1,n}
\end{aligned}$$

Let $A[j][i] - E(I)[j][i] = B[j][i]$. Then in lp_solve input format this task written as:

$$\begin{aligned}
&\min: X1 + X2 + \ldots + Xm; \\
&B[1][1]X1 + B[1][2]X2 + \ldots + B[1][m]Xm = 0; \\
&\qquad\qquad \ldots \\
&B[n][1]X1 + B[n][2]X2 + \ldots + B[n][m]Xm = 0; \\
&X1 + X2 + \ldots + Xm >= 1; \\
&\text{int} X1, X2, \ldots, Xm;
\end{aligned}$$

Solution of this task is a element of Hilbert basis because it's a minimal solution. Therefore when we compare solutions we seek this element in Hilbert basis.

3.7 class CheckSolutions

All comparing methods of solutions we collecting in class CheckSolutions. In Solver class method checkSolution() calls corresponding method in class CheckSolutions.

Class CheckSolutions contain methods:
boolean hilbertToHilbert(Solution mainSol, Solution currentSol); comparing Hilbert basises and return true if Hilbert basises ar equal or false if not equal.

boolean solutionToHilbert(Solution mainSol, Solution currentSol); separating solution currentSol into Hilbert basis mainSol. Return true if solution currentSol can separate into Hilbert basis mainSol or false if it's impossible.

3.8 class GenTask

Added private field boolean set which contain true if need generating a set of ANLDE systems. Also added methods void setSet(boolean) and boolean getSet(). Constructor GenTask() was replaced by constructor GenTask(ANLDE, Limits, String, boolean).

3.9 class GeneratorSpooler

Changed generators.properties format:

Generator*i*.name = *Package*

Generator*i*.path = *Path*

where i — generator number (i=1,2,...), *Package* — where is generator class located, *Path* — where is algorithm execution file lacated. All paths should be relatives.

In class GeneratorSpooler added field private Thread theThread and methods public void start(), public void stop() and public void run() for implementing separate thread. Constructor GeneratorSpooler(path) was replased by GeneratorSpooler(path, maxsize). Field private int maxBufferSize was added.

3.10 class SolTask

Constructor SolTask() was replaced by constructor SolTask(ANLDE, Limits, String[]).

3.11 class SolverSpooler

Changed solvers.properties format:

Solver*i*.name = *Package*

Solver*i*.path = *Path*

where i — solver number (i=1,2,...), *Package* — where is solver class located, *Path* — where is algorithm execution file lacated. All paths should be relatives.

In class SolverSpooler added field private Thread theThread and methods public void start(), public void stop() and public void run() for implementing separate thread. Constructor SolverSpooler(path) was replased by SolverSpooler(path, maxsize). Field private int maxBufferSize was added.

3.12 class Lp_solveOutputParser

In method parseOutcome added int size (number of elements in solution) for verification.

3.13 class BaseSolverOutputParser

In method parseOutcome added int size (number of elements in solution) for verification.

3.14 class CeneratorOutputParser

In method parseANLDE added boolean set field for signification of set of ANLDE systems. It is used for adding set of ANLDE systems in list.

3.15 class ANLDECh

Method void init(SolverOutcome) for start collecting ANLDE system characteristics was added. Method void add(java.util.List) for adding information about solutions was added. Fields int count and int countSols was added. Field List solvers and methods void setSolvers(List) and List getSolvers() was added.

4 Management Subsystem

4.1 class UserProfile

4.1.1 User Permissions Checking

Added `String group` field, `public void setGroup(String)` and `public String getGroup()` methods to manage user groups. Three user groups are supported: admin, user and guest. Also methods for checking user permisions have been added:

- `public boolean canViewStatistics()` returns `true` for users from the admin group;
- `public boolean canChangeProfile()` returns `true` for users from the admin and user groups;
- `public boolean canChangeDefaultLimits()` returns `true` for users from the admin group;
- `public boolean canManageUsers()` returns `true` for users from the admin group;

4.1.2 Profile Invalidation

Added `boolean valid` field, `void invalidate()` and `boolean isValid()` functions to mark and check removed profiles.

4.2 class Management

Conformity of user limits with default limits is not checked for system administrator.

4.2.1 sendNotes(note, anlde, admEmail, userProfile)

Added `String admEmail` and `UserProfile userProfile` fields. `userProfile` is used for setting "From" field in a email message. Also `anlde` type is changed to the String, because a textual representation of ANLDE system is needed. In method `sendNotes()` added parameter `String noteType`. This parameter used in "subject".

4.2.2 getLimitsFor(UserProfile)

New method to get user limits bounded by the default (bounding) limits.

4.2.3 setLimitsFor(UserProfile, Limits)

New method, does nothing but invoking UserProfile.setLimits(Limits).

5 Data Store Subsystem

5.1 class UserProfileStore

Profile data file format is changed. File used by class.java.Util.Properties, where keys are:

- `email` — user's email
- `password` — user's password (required)
- `fullname` — user's full name
- `information` — user's information
- `group` — user's group

User limits (Limits format):

- `limits.time`

- `limits.memory`
- `limits.coefficients`
- `limits.solutionValues`
- `limits.equations`
- `limits.unknowns`
- `limits.systems`
- `limits.solutions`
- `limits.reportSolutions`

Possible values are described in the User Information format. Name of file is the same with user nickname.

5.2 class DefaultLimitsStore

Default limits file format is changed. File used by class.java.Util.Properties, where keys are:

Default limits (Limits format):

- `time`
- `memory`
- `coefficients`
- `solutionValues`
- `equations`
- `unknowns`
- `systems`
- `solutions`
- `reportSolutions`

Possible values are described in the Limits format.

5.3 class StatisticsStore

Added easy checking of length of IP address. The blank activity statistics file and file with one commentary line are considered as corrupted.