Tablix User's Manual

Tomaž Šolc
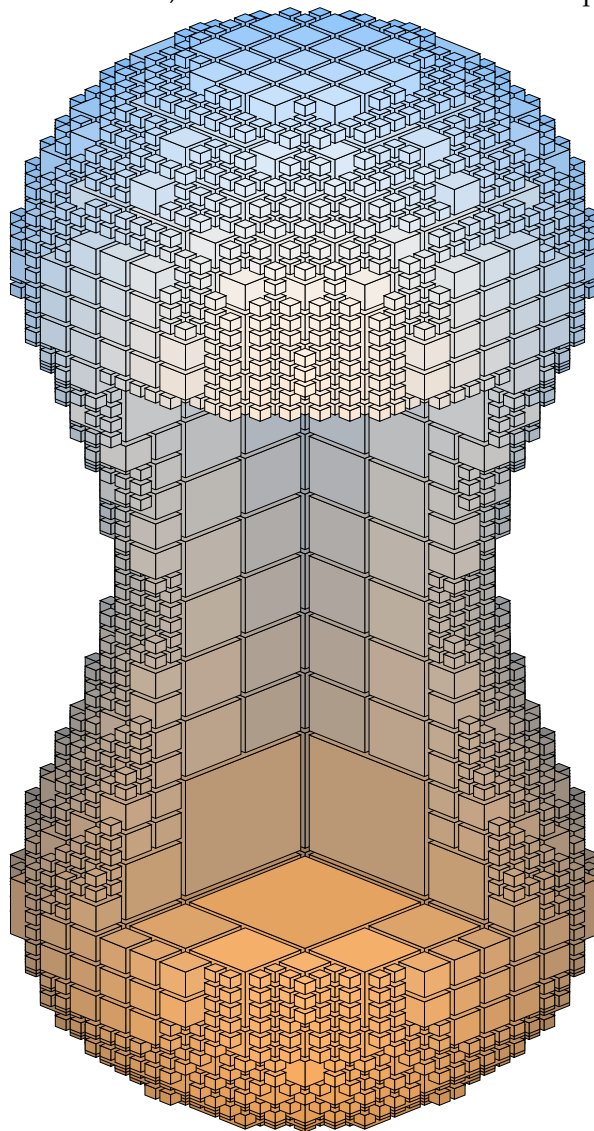
Tablix User's Manual

Tablix is a powerful free software kernel for solving general timetabling problems. It uses a coarse-grained parallel genetic algorithm in combination with other techniques to construct sensible timetables from XML formatted problem descriptions. Tablix can run on a single host as well as on a heterogeneous parallel virtual machine using PVM3. This document tries to introduce Tablix to a regular user. It includes instructions for installation, usage and a section on troubleshooting.

Table of Contents

List of Figures

Chapter 1. Introduction

1.1. What is Tablix?

Tablix is a powerful free software kernel for solving general timetabling problems. It uses a coarse-grained parallel genetic algorithm in combination with other techniques to construct sensible timetables from XML formatted problem descriptions. Tablix can run on a single host as well as on a heterogeneous parallel virtual machine using PVM3.

Tablix kernel supports a very wide range of timetabling problems, from high school timetabling to barge scheduling. A number of timetable constraints are already implemented in the default installation. Because of kernel's modular design it is easy to add custom timetable constraints and/or modify existing ones. Kernel modules are written in C. Extensive API documentation is available on the internet and in the source distribution.

The use of the Parallel Virtual Machine enables Tablix to use the combined power of any group of computers connected by a network. It can run for example on a cluster of old machines that are no longer used or on workstations in the computer science labs that aren't used during the night. Cluster can be composed of machines of different architectures and/or running different operating systems. Bandwidth requirements are low (a 10 Mbps LAN is sufficient). Tablix will also run on a number of bootable CDs that support clustering. A specialized GNU/Linux bootable CD distribution called Tablix on Morphix (http://www.kiberpipa.org/~tomaz/tom) is also freely available on the internet.

The default installation will export finished timetables into XHTML 1.1 format ready for publishing on the internet or into a "comma separated values" format (CSV) that is suitable for import into a spreadsheet application and further processing. New export formats can be added by writing custom export modules.

Tablix kernel is developed according to the rule of separation of interfaces and engines. The kernel can be used standalone with a command line interface. Because it uses XML formatted files for input and output files it is very simple to incorporate into other software packages that provide either customized user interfaces or various forms of pre- and post-processing. A separately developed friendly graphics user interface to Tablix kernel called G-Tablix (http://gtablix.homelinux.org/wordpress) is freely available on the internet. Problem description files can also be edited with general-purpose XML editing software like MlView (http://www.freespiders.org/projects/gmlview) or KXML Editor (http://kxmleditor.sourceforge.net/).

Tablix is free software and is available under the terms of GNU General Public License.

1.2. Supported platforms

Tablix is written with portability in mind. It should compile without problems on most UNIX-like operating systems that support PVM3 (http://www.csm.ornl.gov/pvm/pvm_home.html) and libxml (http://xmlsoft.org/).

Following is a list of operating systems that are known to work with Tablix:

- Debian GNU/Linux i386 (2.1, 2.2, 3.0)

- NetBSD/i386 (1.5.2, 1.6.1, 1.6.2)

- FreeBSD (4.7)

- Mac OS X (10.3.9)

Tested with following versions of PVM3: 3.4.5

Tablix will work with any version of libxml later than 2.4.3. However using versions later than 2.4.11 is recommended because it supports printing line numbers with parser error messages.

> **Note:** Utilities in the `utils` subdirectory also need GNU plot (http://www.gnuplot.info), bash shell and various Unix utilities like grep, sed, awk, etc. to run.

Chapter 2. Installation

Note: You will need root privileges on your system, unless you install from source.

2.1. Tablix packages

At the time of writing Ubuntu GNU/Linux (http://www.ubuntulinux.org/) and Gentoo GNU/Linux (http://www.gentoo.org/) include Tablix package in their package managing systems. Packages should also be soon be available for Debian GNU/Linux.

Before compiling and installing Tablix from source you should check if your operating system distribution already includes Tablix packages. If it does, then the easiest way to install Tablix on your machine is to use the package manager your distribution supplies. Note however that the included Tablix package may be outdated and that you may want to install from source anyway to get the latest version of Tablix (specifically Gentoo package is terribly outdated at the time of writing). The newest Tablix releases can always be downloaded from tablix.org (http://www.tablix.org).

Note: Even if you can't or do not want to install Tablix from a package for your distribution you may want to consider using libxml and/or PVM3 packages and then compiling Tablix using libraries from these packages. Note however that in this case your have to install packages that include development versions of libraries (such packages usually have `-dev` in their names).

2.2. Installation for Debian users

2.2.1. Using apt-get

Debian packages of the stable releases of Tablix are available from an unofficial Debian repository (http://www.kiberpipa.org/~tomaz/tablix/stable). To use this repository with `apt-get` add the following line to your `/etc/apt/sources.list` file:

```
deb http://www.kiberpipa.org/~tomaz/tablix/stable ./
```

Then you can install the latest Tablix release by running the following commands as root:

```
# apt-get update
# apt-get install tablix2
```

Note: Please note that Tablix packages may soon be included in the official Debian package repositories so it may no longer be necessary to change your `sources.list` file.

Note: If you are running an older Debian distribution or a current distribution on an architecture other than i386 you may have to compile your own Debian package. Packages available on the internet will only install on the latest Debian GNU/Linux distribution on i386 architecture. See the following section for more details..

2.2.2. Compiling Debian package from source

Following section describes how to make your own Debian package out of the Tablix source distribution.

1. Install libxml2 and pvm3 development packages.

   ```
   # apt-get install libxml2-dev pvm-dev
   ```

2. Download and untar Tablix source distribution.

   ```
   $ wget http://www.tablix.org/releases/stable/tablix2-0.2.2.tar.gz
   $ tar -xzf tablix2-0.2.2.tar.gz
   $ cd tablix2-0.2.2
   ```

3. Build Debian package.

   ```
   $ fakeroot debian/rules binary
   $ cd ..
   ```

4. Install the Debian package.

   ```
   $ su
   # dpkg --install tablix2_0.2.2-1_i386.deb
   ```

2.3. Installation from source

1. Build and install supported versions of libxml2 and PVM3. If you don't have root access to your machine, you can install them in your home directory. Be sure to set `PVM_ROOT` and `PVM_ARCH` correctly in the latter case. See installation instructions included with libxml2 and PVM3 distributions for more information.

2. Download and untar Tablix distribution into your home directory.

   ```
   $ wget http://www.tablix.org/releases/stable/tablix2-0.2.2.tar.gz
   $ tar -xzf tablix2-0.2.2.tar.gz
   $ cd tablix2-0.2.2
   ```

3. Configure distribution. Check `./configure --help` for available options. Defaults should work in most cases.

```
$ ./configure
```

If you don't have root privileges, you can specify *--prefix* to install Tablix into a subdirectory in your home directory.

> ## Warning
>
> If the configure script can't find your PVM3 installation or the *--without-pvm3* flag was used, Tablix will be compiled in debug mode and a warning will be printed at the end of the configure process.
>
> Tablix compiled in debug mode uses a linear genetic algorithm instead of a coarse grained parallel genetic algorithm. This makes debugging easier but also severely decreases the probability that Tablix will find an optimal solution to the given timetabling problem. Solving even moderately complex problems is impossible in this mode.
>
> Unless you are debugging the kernel or one of the modules you should always use PVM3 (even if you are going to use Tablix on a single machine).

4. Compile. You can add some optimization flags by setting *CFLAGS* environment variable. Defining you CPU architecture with a parameter like *-march=athlon* can improve performance. See documentation for the version of C compiler you are using for more information.

```
$ make CFLAGS=-march=pentium
```

5. Install compiled binaries. If you specified a proper *--prefix* option above, you may not need to use root privileges.

```
$ su
# make install
```

2.4. Clusters

The same version of Tablix must be installed on all machines in the cluster. The `tablix2_kernel` executable must be located somewhere where the `pvmd` daemon will find it. This in most cases means that it must be in the default `PATH` for the user under which `pvmd` is running and / or the `ep` option in the PVM's `hostfile` must be set correctly. See PVM3 documentation for more details.

The most straight forward way to install Tablix on a cluster is to repeat the installation steps described above on every machine. If all machines are identical, you can simplify this task by using a utility like clusterssh (http://clusterssh.sourceforge.net/).

A less boring way is to set up network booting for machines in the cluster (using a setup like the Linux terminal server project (http://www.ltsp.org)). You can also just use NFS exported directories to store the Tablix installation in. Install PVM, libxml2 and Tablix all in a NFS exported directory on a master computer. Then mount this directory on every computer in the cluster.

Another way to do this is with bootable CDs. You can use Tablix on Morphix (http://www.kiberpipa.org/~tomaz/tom), a modified version of Morphix that has Tablix already installed, together with automatic cluster configuration tool. Any other bootable CD that supports PVM3 clusters can also be used.

If you want to make your own bootable CD, you can find step by step instructions in Tablix on Morphix HOW-TO.

Chapter 3. Getting started

After Tablix has been installed on your workstation follow these steps to test your installation. These are the basic steps required to obtain a solution to a timetabling problem. You will repeat them every time you will use Tablix (unless you will be using some front-end other than the default command line interface, in which case the front-end will do these steps for you).

If you are running Tablix on a cluster, choose one machine to be the master node. You will control Tablix from that machine. Do the following steps only on that computer.

1. First you have to describe your timetabling problem in the format that Tablix understands. Tablix uses an unified XML format to describe problem definitions and solutions to these problems.

   You can find some example configuration files in the `examples/` subdirectory. These steps presume that you are using the `examples/sample2.xml` configuration. See sections later in this document for instructions on how to write your own problem definitions.

   ```
   $ cd examples
   ```

2. If you are running Tablix on a single machine and you have installed Tablix system-wide (e.g. you didn't supply any *--prefix* options to the `./configure` script) you probably do not need a PVM hostfile and you can skip this step. If you have problems starting Tablix later on one of the reasons can be that you need a hostfile with a correct *ep* parameter.

   If you are running Tablix on a cluster a hostfile is usually unavoidable. See PVM3 documentation for details. Following is an example hostfile for a cluster composed of three different machines.

   ```
   orion   ep=/home/avian/tablix/src \
           sp=7000
   dolphin ep=/home/avian/tablix/src \
           sp=1600
   europa  ep=/home/avian/tablix-0.0.1/src \
           sp=800 \
           dx=/home/avian/src/pvm3/lib/pvmd
   ```

   > **Note:** The *sp* field is important. Try to match relative speeds of your machines as accurately as possible. This will help Tablix balance the load more evenly and will increase performance.

3. Start pvm. If you don't need a hostfile, you can omit it on the command line.

   ```
   $ pvm hostfile
   ```

   > **Note:** Tablix does not automatically detect if new nodes have been added to the cluster while it is running. All nodes must be properly configured before Tablix starts.

4. Start Tablix. See `tablix2 -h` or the man page for a list of the available command line arguments.

   ```
   $ tablix2 -n 10 -o test1_ sample2.xml
   ```

   The *-n* specifies the number of computational nodes Tablix will start on the cluster. For a single machine the default number (4) will usually be sufficient for simple problems. On a cluster of machines a good guess is N*4, where N is the number of machines.

   The *-o* option instructs Tablix to prefix all output files with `test1_`.

5. Now you have to wait for Tablix to find a solution. Time depends on your configuration file, number of nodes, speed and architecture of nodes, etc. See below for the explanation of numbers that show up the screen. On a reasonably fast machine (e.g. single Athlon XP 2500+) the example problem `sample2.xml` will take around 30 minutes to solve.

   You can track the progress of a running Tablix process by running `tablix2_plot` utility. It can draw a number of different graphs that show how successful is the algorithm at finding the solution to the problem. You have to run it from a terminal emulator running on X Window System.

   ```
   $ tablix2_plot --conv-fitness test1_
   ```

6. If Tablix found the solution, it has written the solution to a number of files with names `test1_result0.xml`, `test1_result1.xml` and so on. The number of files should match the number of computational nodes specified with the *-n*. Each computational node will return its own result. These results will usually be very similar of even identical.

   XML files can be read directly, but in most cases you will want to convert them into a format that is easier to read by humans. One available option is to convert it to a web page (XHTML format) and open it in a browser.

   ```
   $ tablix2_output -o my_timetable.html htmlcss test1_result1.xml
   ```

7. You can now open the resulting timetable in you favorite browser and check the results:

   ```
   $ firefox file://`pwd`/my_timetable.html
   ```

**Figure 3-1. Graphical representation of a typical Tablix session**
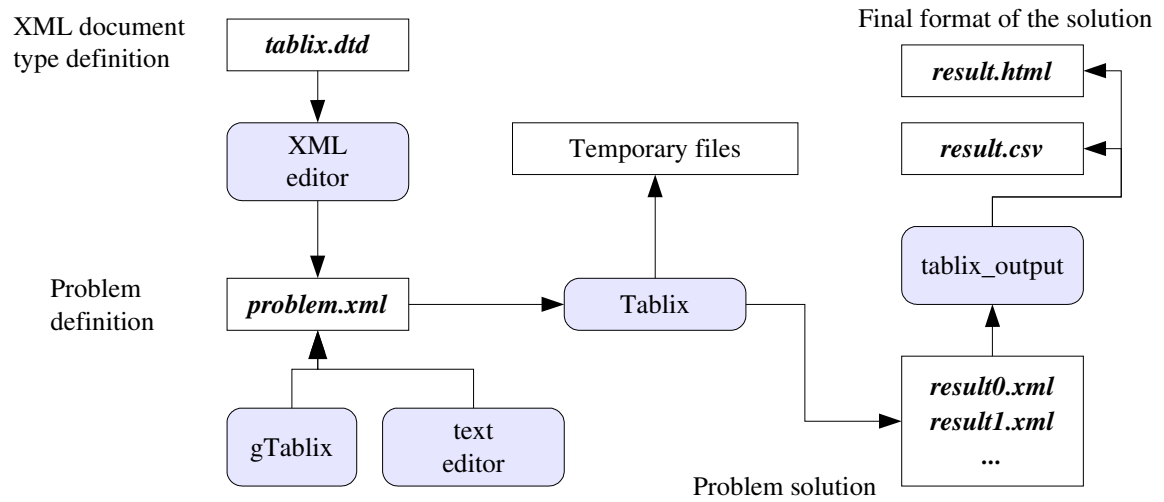
# Chapter 4. Using Tablix

## 4.1. Introduction to genetic algorithms

Tablix uses a modified genetic algorithm to solve timetabling problems. Genetic algorithm works by first evaluating a large number of random timetables (this group of timetables is called a population). It assigns a fitness value to each timetable. This fitness value indicates how well the timetable satisfies the restrictions given in the definition of the timetabling problem. Lower fitness values mean a better solution and fitness value 0 indicates a perfect solution. The type of genetic algorithm Tablix uses maintains a separate population on each computing node.

> **Note:** Fitness value 0 indicates a perfect solution, but solutions with fitness values greater than 0 may also be acceptable, depending on the problem description).

Genetic algorithm leaves the part of the population with lower fitness values unmodified and replaces the part of the population with higher fitness values with new timetables that are based on the first part. One such step of the algorithm (evaluation and replacement of the worse part of the population) is a called a generation.

When an acceptable solution is found in the population, Tablix writes it into a file and exits.

There are several important points you should keep in mind when using Tablix:

- The genetic algorithm isn't deterministic. Because it starts with a random population, the results will differ each time you run Tablix, even if you use exactly the same configuration file.

  Most users run Tablix several times and then choose the best solution.

- There is no way for Tablix to determine if a solution exists for given the timetabling problem. Tablix will do its best to find out any obvious flaws in the problem definition, but if it doesn't find any doesn't mean that the solution exists.

  If the solution doesn't exist then the genetic algorithm will never finish.

- The total size of the population determines how exhaustive search for the solution will be. The size of the population can be increased by increasing the number of computing nodes (the `-n` parameter) or by increasing the population of one node (the `popsize` algorithm parameter).

  Problems that are harder to solve require larger populations for Tablix to reliably find a solution.

## 4.2. Tablix master process

When you run `tablix2`, you actually start the master process that will start the requested number (`-n` option) of slave processes (Tablix kernels) on the virtual machine. It will then multicast the configuration file to all the computing nodes and start listening for their reports.

Tablix isn't very verbose by default. You can enable additional informative and debug messages with the *-d N* option, where *N* is an integer from 0 (only fatal error messages are shown) to 4 (display debug messages). Default is 2.

Following is an explanation of the progress indicator that is shown at verbosity settings greater than 1:

```
[4000b] reports 87271 (0) at 1, 480.0 GPM, 00:00:05 elapsed, 4/4 running
  ^              ^      ^    ^ ^            ^                  ^
  |              |      |    | |           |     _____|
  |              |      |    | |           |    |
  |              |      |    | |           |    |_ Number of running
  |              |      |    | |           |        nodes / number of
  |              |      |    | |           |        nodes in the cluster.
  |              |      |    | |           |
  |              |      |    | |           |        Node stops when it finds
  |              |      |    | |           |        an acceptable solution.
  |              |      |    | |           |
  |              |      |    | |           |_ Elapsed time since Tablix
  |              |      |    | |               was started.
  |              |      |    | |
  |              |      |    | |_ Total generations per minute count for
  |              |      |    |       the cluster.
  |              |      |    |
  |              |      |    |_ Population serial number
  |              |      |
  |              |      |_ 0 means that an acceptable solution was not found
  |              |          in the population. 1 means the population contains
  |              |          an acceptable solution.
  |              |
  |              |          This must be one for at least 300
  |              |          generations for Tablix to stop.
  |              |
  |              |_ Weighted sum of all errors
  |
  |_ PVM Task ID of the node that sent the last report.
```

You can press Ctrl-C (or send SIGINT) to stop the process. Tablix will save its state in a number of files called `save0.txt`, `save1.txt`, etc. (it will prepend your prefix, if given). You can later resume the process by running Tablix with the *-r* parameter. You should not change the XML configuration file in any way between stopping and restoring a Tablix computation. You may however change the number of computing nodes.

When all the criteria for an acceptable solution are satisfied, Tablix will output one XML file for each node (file name will be prefix + `result0.xml`, `result1.xml`, etc.). These files can then be processed with `tablix2_output` to produce a timetable in a format suitable for further processing or display.

During the computation, Tablix saves the progress of the computation in files named `conv0.txt`, `conv1.txt`, etc. (unless you compiled Tablix with convergence information saving disable (*--disable-conv* `./configure` parameter). You can plot this data using `tablix2_plot` script. See the man page for more information.

4.3. Tablix timetabling model

Note: See Tablix timetabling model formal description (http://www.tablix.org/~avian/ttm2.pdf) for more detailed description.

4.3.1. Timetable information

Timetable in Tablix is defined as a group of events. Each event (sometimes also called a tuple) uses a fixed number of resources. Resources are grouped into several resource types.

In school scheduling, examples of resource types would be: teachers, groups of students, classrooms and timeslots. Each resource in the "teachers" resource type would represent on teacher. Each resource in the "groups of students" resource type would represent on group of students. Each resource in the "timeslots" resource type would represent on timeslot.

Resource types are divided into two groups: constant resource types are assigned to events by the user. Tablix will never change these assignments. variable resource types on the other hand are usually not assigned to events by the user. Tablix will change this assignments in order to find a solution to the timetabling problem. The solution therefore consists of proper assignments of variable resources to the events.

Note: Each event uses one resource of each defined resource type.

Note: It is possible to also assign variable resource to events. These assignments are called hints. Tablix will try to use them when finding the solution.

In the above example, teachers and groups of students would be constant resources and classrooms and timeslots would be variable resources. The user would assign teachers and groups of students to events (events would represent lectures in this case). Tablix will then try to find suitable time and place for lectures to take place.

Resources can be defined one by one, as a single row of resources or as a two dimensional matrix of resources. A resource matrix for example is usually used for the time resource. Each column in the matrix then represents one day in the week.

Note: It may seem that the fact that all events use exactly the same number of resources makes it impossible to apply this model to timetables where an event may use more than one event of a single resource type. In reality various timetable constraints (for example resource conflicts) allow this model to be very flexible and to also be applied to that kind of timetables.

4.3.2. Timetable constraints

Each timetable constraint is defined by a Tablix kernel module (called a fitness module. For example, the `sametime.so` module in school scheduling ensures that no teacher and no student has two lectures scheduled at the same time.

Different modules are intended for different purposes and make different assumptions at what resource types are defined in the configuration file. This means that to use a certain module, the timetabling problem must include certain resource types and sometimes even that some resource types must include for example a matrix of resources. At the time of writing, most modules are intended for school scheduling.

You must assign a weight value to each module. Weights are used when calculating the fitness value of a timetable and define how much computational effort is used for reducing a certain type of errors. Each module can also be defined as either mandatory or non-mandatory. All fitness functions defined by mandatory must return zero errors in order for a timetable to be considered an acceptable solution. An example of a module that is always defined as mandatory is the `sametime.so` module mentioned above. A non-mandatory module would be a module that detects timetable features that are not wanted, but does not make a timetable impossible.

> **Note:** Mandatory modules should have larger weight values than non-mandatory modules in order for Tablix to reliably find a solution.

> **Note:** Some modules do not define fitness functions. These modules ignore the weight and mandatory settings given to them.

You can find the descriptions of all modules included in the Tablix distribution in the Tablix modules reference manual. It is available on-line (http://www.tablix.org/releases/doc/modules) or in the `doc` subdirectory in the source tree.

Each fitness module can define one or more restrictions and can accept several module options. Two types of restrictions exist: event restrictions that are applied to events and resource restrictions that are applied to resources. A resource restriction that is defined by a module can sometimes only be applied to resources of a certain resource type and can sometimes be applied to any defined resource.

4.4. Setting weights

Weight values tell Tablix how exactly to shape the timetable. Setting weights can be tricky because wrong combinations usually lead to either sub-optimal timetables or Tablix working in an endless loop.

There is currently no way to check weights values for sanity. You will have to experiment and see which values fit best.

The best way to start is to set all weights for non-mandatory modules to a very low value (for example 1) and all weight for mandatory modules to a very high value (for example 500). This ensures that Tablix will direct most effort to reducing the number of errors that are keeping the solution from being acceptable.

If Tablix does not finish after a reasonable time (hard problems can take a long time to solve), then use the `tablix2_plot` utility with `--functions` parameter to see which mandatory fitness functions are keeping Tablix from finding a solution.

Look at the far right end of the graph. Return values of mandatory fitness functions are plotted with thicker lines. If Tablix did not finish even after a large number of populations, one or more of the mandatory functions will probably have a flat line at the end at some value higher than zero. These mandatory functions are keeping Tablix from finding a solution. You should increase the weight values for these functions and try again. Use this procedure until you find the combination of weights that will work. If you can't find the combination (it shouldn't take more than a one or two tries), then either the problem is too hard for the size of the population you are using (increase the `-n` parameter and add more hardware into the cluster).

When you have the combination of the weights for the mandatory modules, you can try to adjust the weights for non-mandatory modules. Take a look at the produced timetables and/or the convergence graph and see which non-mandatory errors are the most outstanding. Increase the weight value for the module that is responsible for that type of errors and try again. When you increase non-mandatory weights to the point just before Tablix will no longer find a solution because it is putting to much effort towards the non-mandatory errors, you have found the optimal setting.

> **Note:** The gain in timetable quality achieved by this slow tweaking of the non-mandatory values is usually low.

4.5. Configuration file format

Tablix uses a XML formatted file to store timetable information. XML consists of tags, nested in each other. Each tag can have one or more properties and some tags can have text content in them. The top level tag, which should include all other tags, is called the root tag.

> **Note:** The XML Document Type Definition document for the Tablix configuration format can be found on-line (http://www.tablix.org/releases/dtd/).

> **Tip:** The most common mistake new users make is to put `<restriction>` tags in wrong places in the XML file. Tablix will in most cases silently ignore unknown extra tags in the XML file and will therefore also ignore any `<restriction>` tags that are not in the right places.
>
> If you are not sure if your XML syntax is correct or if Tablix seems to be ignoring some of your `<restriction>` tags, you can use the `xmllint` utility that comes with libxml2 to check your XML file against

the Document Type Definition. Type `xmllint --valid --noout your-config-file.xml` at the command line. If it returns no error messages then the XML is most likely syntactically correct.

Tablix configuration consists of several parts. They are delimited by top level tags. The root tag should be named `<ttm>`. The top level tag must also contain the `version` property that contains version number of the format. This version number should be "0.2.0" for use with Tablix kernels versions 0.2.1 and later.

Note: All strings (resource names, restriction types, etc.) that appear in the XML file are case sensitive. You can have for example three different resources in the configuration file with names "ALPHA", "Alpha" and "alpha".

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE ttm PUBLIC "-//Tablix//DTD TTM 0.2.0//EN" "http://www.tablix.org/releases/dtd/tablix2r1.dtd">

<ttm version="0.2.0">
        .
        .
        .
</ttm>
```

4.5.1. Title, address, author

First part of the configuration is optional. You can include the information about the author of the timetable, the address of the institution and the title of the timetable under the `<info>` tag.

```
<info>
        <title>Example high school</title>
        <address>Somewhere</address>
        <author>Someone</author>
</info>
```

This information will be printed on the top of the timetable when exported into the HTML format.

4.5.2. Modules

Next part of the configuration defines the modules that will be used by Tablix. This part and all subsequent parts of the configuration are mandatory.

Modules are listed with `<module>` tags inside the `<modules>` tag.

```
<modules>
        <module name="sametime.so" weight="200" mandatory="yes"/>
        .
        .
        .
<modules>
```

Some modules may accept module options. They can be specified with *<option>* tags. Each module option has a name (specified with the *name* property) and a value.

```
<modules>
        <module name="holes.so" weight="6" mandatory="no">
                <option name="resourcetype">class</option>
        </module>
        .
        .
        .
</modules>
```

> **Note:** With the selection of modules you choose the group of errors for which Tablix will try to optimize your timetable. With weight values you can tell Tablix which errors are more important than others.

> **Note:** Each module can define one or more restriction types. See the Modules reference manual for more information on which module defines which restrictions. Tablix will ignore any unknown restriction types in the configuration file. A warning is printed for each unknown restriction encountered if the verbosity level is set to 3 or higher (default is 2).

> **Tip:** You can find working examples for all modules in the `ttf/tests/` subdirectory. Each module included in the distribution has one or more test cases stored there. These test cases are used by the Tablix Testing Framework to automatically check if modules are working correctly. For now, you can treat these files as normal problem descriptions and ignore the extra Scheme code at the beginning.

4.5.3. Resources

All resources types that will be used in your timetable should be defined under the *<resources>* tag.

```
<resources>
        <constant>
        .
        .
        .
        </constant>
        <variable>
        .
        .
        .
        </variable>
</resources>
```

Constant resource types should be defined under the *<constant>* tag and variable resource types under the *<variable>* tag.

A resource type is defined with a *<resourcetype>* tag. Name of the type is specified inside a *type* property.

```
<constant>
        <resourcetype type="teacher">
                .
                .
                .
        </resourcetype>
        .
        .
        .
</constant>
```

Each resource type can contain one or more resources. A single resource is defined with a *<resource>* tag inside *<resourcetype>*. Each resource must have its name defined with a *name* property.

> **Note:** Two resources of the same resource type can not have identical names.

```
<resourcetype type="teacher">
        <resource name="Prof. SJK 1"/>
        .
        .
        .
</resourcetype>
```

If some modules have defined resource restrictions for the defined resource types, then you can apply a resource restriction to a resource with a *<restriction>* tag.

```
<resourcetype type="class">
        <resource name="3 B">
                <restriction type="conflicts-with">3 MA-FIZ</restriction>
        </resource>
        .
        .
        .
</resourcetype>
```

Each restriction has a type (specified with the *type* property) and a value.

A row of resources can be defined with a *<linear>* tag. This is a simple way to defined a larger number of identical resources.

```
<resourcetype type="room">
        <linear name="K#" from="1" to="5">
                <restriction type="capability">Kemija</restriction>
        </linear>
        .
        .
        .
```

```
</resourcetype>
```

You must assign a name to the row of resources with the `name` property. The character "#" in the name will be replaced by an integer ranging from the number in the `from` property to the number in the `to` property.

Resource restrictions can be applied in the same way as with the `<resource>` tag.

The example above will define 5 resources with names from K1 to K5. All resources will have restriction "capability" applied.

A matrix of resources is defined in a similar way with the `<matrix>` tag.

```
<resourcetype type="time">
        <matrix width="5" height="8"/>
</resourcetype>
```

The `width` and `height` properties define the dimensions of the matrix. You can not define a name for the resources in the matrix. All resources will be named "X Y", where X is the X coordinate of the resource in the matrix and Y is the Y coordinate.

Resource restrictions can be applied to a resource matrix in the same way as with the `<resource>` tag.

4.5.4. Events

Events must be defined with `<event>` tags under the `<events>` tag.

```
<events>
        <event name="SJK" repeats="4">
                .
                .
                .
        </event>
        .
        .
        .
</events>
```

Each event has a name defined with the `name`. Two different events can share the same name, however this practice is not recommended since it may confuse some modules.

The `repeats` property defines how many times this event is repeated. For example if an event has `repeats` equal to 4, this has the same effect as if the the `<event>` tag defining this event would be copied four times in the configuration file.

> **Note:** The `repeats` has no other hidden effects. It exists only for convenience when writing XML files by hand. This means that the following two examples are identical as far as Tablix is concerned. In fact if you look at the XML file with the problem solution you will see that all `<event>` tags are expanded so that none have `repeats` greater than 1.
>
> This example...

```
<event name="SJK" repeats="3">
        .
        .
        .
</event>
```

...is identical to this one:

```
<event name="SJK" repeats="1">
        .
        .
        .
</event>
<event name="SJK" repeats="1">
        .
        .
        .
</event>
<event name="SJK" repeats="1">
        .
        .
        .
</event>
```

Resources that are used by an event are defined with the `<resource>` tags inside the `<event>` tag.

```
<event name="SJK" repeats="4">
        <resource type="teacher" name="Prof. SJK 1"/>
        <resource type="class" name="1 A"/>
</event>
```

`<resource>` tags must have `type` and `name` properties, defining resource type and name of the resource. It is mandatory to assign one resource of each constant resource type to each event. Optionally, you can also assign one resource of some or all variable resource types to some or all events. Such assignments will be used as hints by Tablix kernel. They may be used in the final solution if they will help to reduce the fitness value of the timetables.

> **Note:** Please note that these hints are not equal to the "fixed tuples" in 0.1 branch. If you would like to permanently assign some variable resources to an event, you have to use a module that defines a restriction that implements this.
>
> For example, to schedule an event at a fixed time, you can use the module `fixedtime.so`.

If some modules have defined event restrictions, then you can apply an event restriction to an event with a `<restriction>` tag inside the `<event>` tag.

```
<event name="KEM" repeats="2">
        <resource type="teacher" name="Prof. KEM 2"/>
        <resource type="class" name="1 B"/>
        <restriction type="capability">Kemija</restriction>
</event>
```

As with resource restrictions each event restriction has a type (specified with the `type` property) and a value.

Chapter 5. Native language support

5.1. User interface translation

Tablix uses GNU gettext to provide translated run-time messages and predefined text in XHTML output. Translation will occur automatically if your environment is set correctly, Tablix wasn't compiled with `--disable-nls` option and there is a .po file for your language available in Tablix distribution. On some systems locale(1) man page holds some details on setting up localization.

Run-time message translation involves some CPU overhead. This should not be noticeable on recent machines, but if you will be running Tablix on older hardware I recommend you compile Tablix with the `--disable-nls` option.

Messages from all nodes in the cluster will be translated to the language of the master process. It is possible however to run a cluster where a part of machines are running Tablix with NLS disabled. In that case messages from those machines will be displayed in English by the master process.

> **Note:** Please note that this section covers only the default command-line interface to the kernel. Any graphical user interfaces may have different levels of support for native languages.

New Tablix translations are always welcome at `<tomaz.solc@tablix.org>`. See `ABOUT-NLS` file for more information about gettext and translations.

5.2. Text encoding support

Tablix stores internally all strings in the UTF-8 encoding. You can use any encoding for the XML configuration file, as long as it can be converted to UTF-8 by your libxml2 library. Remember to declare the encoding of your file in the XML prolog. For example:

```
<?xml version="1.0" encoding="iso-8859-2" ?>
```

The XHTML output always uses UTF-8 encoding and is in strict compliance with XHTML 1.1 standard. It should preserve all non-English characters that are used in the XML configuration.

Chapter 6. Compatibility with older versions

6.1. Changes between releases 0.3.3 and 0.3.4

Some changes have been made to the module loading code to make Tablix portable to systems that do not use `.so` extension for dynamic libraries.

Since release 0.3.4 module names in `<module>` tags can be written without the `.so` extension. Tablix kernel will automatically add the proper extension at runtime. To remain backward compatible with problem descriptions written for earlier versions Tablix also still accepts module names written with the `.so` extension. In that case the `.so` part will be automaticaly replaced on systems that use a different extension.

It is recommended that new problem descriptions do not use the `.so` extension with module names.

6.2. Changes between releases 0.3.1 and 0.3.2

A design error has been discovered in the dependent tuples functionality of Tablix 0.3.1. To deal with this problem the part of the kernel API that provides support for updater functions has been redesigned in Tablix 0.3.2.

Any fitness modules using updater functions in Tablix 0.3.1 will not work with with Tablix 0.3.2. These modules have to be rewritten to use the new API if you wish to use them with Tablix 0.3.2 and later. This is recommended in all cases since the design error mentioned above can introduce subtle errors in the timetables produced from problem descriptions using these modules (see Design error in Tablix 0.3.1 (http://www.tablix.org/archives/2006-04-07T16_00_44.html) for more information).

> **Note:** The new API is designed to be as similar to the old one as possible and is in some cases even easier to use. In most cases only minor modifications are necessary. See Tablix modules HOW-TO, part 1, for more information.

Modules written for Tablix 0.3.1 that do not use updater functions will work on Tablix 0.3.2 without modifications. Tablix 0.3.2 also remains source and binary compatible with modules written for the 0.3.0 release.

6.3. Changes between branches 0.1 and 0.3

Tablix kernel has been completely rewritten between these two stable branches. Because of the large difference in features there is no straightforward way of migrating configuration files and modules.

Care has been taken to rename all installed files (executables from the branch 0.3 are prefixed with `tablix2` compared with `tablix` in branch 0.1). Therefore one version of Tablix from each branch can be installed on the same machine or cluster without interference.

Some modules have been renamed during the 0.2 development branch and sometimes multiple modules have been merged together into a more general module. Following is a list of modules that were available during the 0.1 branch together with the names of modules that are available in the 0.3 branch that provide identical or similar functionality. Please note that this list reflects the situation at the time of writing of this manual. You can always find an up-to-date list of modules in the 0.3 branch at tablix.org (http://www.tablix.org).

`class_freeperiod.so`

No module is currently available with this functionality. `freeperiod.so` module provides similar functionality for teachers.

`double_period.c`

Replaced by `consecutive.so`.

`forcesametime.so`

Replaced by `sametimeas.so`.

`more_teachers.so`

The same functionality can be achieved by using the *conflicts-with* restrictions defined by the `sametime.so` module.

`placecapability.so`

Identical functionality provided by a module with the same name.

`preferred.so`

No module is currently available with this functionality. A mandatory restriction of this type can be achieved by using the `fixedtime.so` module.

`sametime.so`

Identical functionality provided by a module with the same name.

`student_afternoon.so`

No module is currently available with this functionality.

`student_freemorning.so`

Replaced by `freemorning.so`.

`student_holes.so`

Replaced by `holes.so`.

`student_perday.so`

Replaced by `perday.so`.

`student_walk.so`

>   Replaced by `walk.so`.

`subject_dispersion.so`

>   No module is currently available with this functionality.

`subject_morning.so`

>   No module is currently available with this functionality.

`subject_notsameday.so`

>   No module is currently available with this functionality.

`subject_preferred.so`

>   No module is currently available with this functionality. A mandatory restriction of this type
>   can be achieved by using the `fixedtime.so` module.

`subject_sameperiod.so`

>   No module is currently available with this functionality.

`subject_sameroom.so`

>   No module is currently available with this functionality.

`teacher_afternoon.so`

>   No module is currently available with this functionality.

`teacher_dayoff.so`

>   Replaced by `freeperiod.so`.

`teacher_first_last.so`

>   No module is currently available with this functionality.

`teacher_freemorning.so`

>   Replaced by `freemorning.so`.

`teacher_holes.so`

>   Replaced by `holes.so`.

`teacher_maxperday.so`

>   No module is currently available with this functionality.

`teacher_perday.so`

>   Replaced by `perday.so`.

`timeplace.so`

Identical functionality provided by a module with the same name.

6.3.1. Converting configuration files

One way to convert configuration files from one format to the other is to use G-Tablix (http://gtablix.homelinux.org/wordpress). At the time of writing it supports reading and writing to both formats, so it is possible to read a file in the old format and write the file in the new format.

Be warned though that code for writing files in the new format currently contains bugs. It is recommended that you check the converted files by hand before using them.

Arief M Utama also made a specialized utility for converting problem descriptions from the old format to the new one. More information about his program is available in the Tablix Wiki (http://www.tablix.org/wiki/wiki.pl?TabConv).

6.3.2. Porting modules

Because of the major changes in the kernel API there is currently no automatic way to convert module source code to fit the new API. Modules need to be manually ported to the new kernel.

Chapter 7. Troubleshooting

7.1. Frequently asked questions

7.1.1. General

Q: Is there a graphical front end for Tablix?

Yes. Boštjan Špetič released the first version of G-Tablix in August 2004. G-Tablix is a GTK front end for Tablix written in Perl. See G-Tablix home page (http://gtablix.homelinux.org/wordpress) for more information.

You can also use one of the free general-purpose graphical XML editors available (you can find a document type definition file in the `examples/` subdirectory).

Q: Is there a Microsoft Windows version?

Maybe. It is theoretically possible to compile Tablix under a UNIX-emulation layer like Cygwin (http://www.cygwin.com/). I know it is supported by both PVM3 and libxml. However as far as I know, nobody actually tried to compile Tablix on that platform. If you are brave enough to be the first to try, please send a mail to the mailing list (or even better, make a page on the Wiki about how you did it).

You can still run Tablix on a machine that has Microsoft Windows installed by using a bootable CD or by booting from network.

Q: I have to add a certain constraint to my timetabling problem. How do I do that in Tablix?

Tablix distribution comes with a number of modules that implement the most common used restrictions. See the Modules reference manual for details on which module implements which restriction and how to use it in your timetabling problem.

Some more complicated restrictions can be implemented by using a combination of simpler restrictions from different modules. There are to many of these combinations to document them all. Use some imagination.

If you really can't find any way specifying the constraint you want with the existing set of modules, you will probably have to write a new module. Sometimes only some minor modifications to existing modules are needed, so look into that first! See Modules HOW-TO, part 1 for more information on how to write fitness modules.

Q: Will you write the fitness module I need for me?

A: Not likely. Tablix has a modular design exactly because the author does not have time to implement every single requirement needed by every user. The best I can do is to answer occasional questions on the mailing list.

You are of course welcome to contribute your modules to the distribution and make life easier for other users that may need the same functionality.

Q: Tablix won't stop even after a long time. All the lines scrolling up in my terminal show the same fitness. Progress indicator shows that no acceptable solutions have been found. Convergence (plotted with `tablix2_plot`) graph shows a long horizontal line at the end. *

A1: The number one reason for this is that you have given Tablix a problem that is impossible to solve. Tablix has only a limited ability to check problem descriptions for errors. A typical example of an error that prevents Tablix from finding a solution is that one restriction forces an event to be using a specific resource and another restriction forbids the same event from using the same resource (and both restrictions are mandatory).

A2: The population of the genetic algorithm has converged to a local minimum. This can always happen. Try running Tablix again. You can reduce the probability of premature convergence by increasing the number of nodes in the cluster (see the `-n` parameter). Alternatively you can increase the weights assigned to mandatory errors. Note that more complex a timetable is a bigger cluster you need to solve it with Tablix.

Q: Tablix won't stop even after a long time. Progress indicator in my terminal shows different fitness values. Convergence graph shows a chopped horizontal line (line jumping up and down) at the end. This can sometimes be seen as a thick line at the end of the graph if the graph is very compressed.

A: Good question. This happens occasionally with no apparent reason. I have two theories, first one a bit more interesting then the other. If you notice Tablix behaving like that please send me a mail and describe the circumstances. It will help a lot.

Theory 1: Genetic algorithm may become unstable with certain parameters. I think this can happen when the fitness function begins to behave chaotically. This behavior can be the effect of some parts of the fitness function strongly opposing each other (meaning that solving a number of problems A spawns approximately the same number of problems B, and vice versa). This would explain why removing some restrictions usually helps in this case.

Theory 2: This is a very long-lived and hard-to-spot bug in Tablix.

Q: Timetables created with Tablix aren't as good as they could be. **

A: Tablix, as is the case with any program using a genetic algorithm, isn't perfect. It will find a different solution each time you run it. Try running Tablix multiple times and then choose the best solution.

You can try to identify which errors are the most common in the generated timetable (see saved convergence info or use the `tablix2_plot` utility). Increase the weights for those errors.

You can get better results by increasing the total size of the population. This can be done either by increasing the number of computing nodes (the `-n` parameter) or by increasing the population of one node (the `popsize` algorithm parameter). This will of course result in fewer generations per minute and it will take Tablix longer to find a result unless you increase the computing power of the cluster (e.g. add more machines) at the same time.

You can also try changing some algorithm parameters. tablix2(1) man page lists all available parameters together with their descriptions. For example, increasing the `localstep` parameter will usually result in better results and slower computation.

Q: I have been to question *, then to question **, then to * and back again. Now I'm at question *. What else can I do.

A: You have probably reached the optimal timetable Tablix can make for you out of the configuration you gave it. Only thing you can do is to either change your data or hack the Tablix's genetic algorithm. If you do the latter, send me a mail. I would love to hear from you.

Q: Some nodes won't send in their population when I press Ctrl-C.

A: This is a bug in some older versions of PVM3.

Q: All nodes will send in the population when I press Ctrl-C, but Tablix will still wait as if some nodes haven't sent it.

A: This is a bug in some older versions of PVM3.

Q: I still don't understand.

A: Read this file again. Then read everything under `doc` subdirectory. Go through example files in the `example` subdirectory. Try experimenting first. If you still don't know where to start, send an email to the mailing list and somebody will help you.

7.1.2. Error messages

Q: I get the following error: `libpvm [t40002]: tablix: Not Found`

A: `pvmd` can't find `tablix2_kernel` executable. Check the *ep* option in your `hostfile` (`man pvmd`), check *PATH* environment variable. For some unknown reason this also happens if you are using IP addresses instead of hostnames in your `hostfile`.

Q: I get the following error when compiling on NetBSD: `'master.c:262: 'LC_ALL' undeclared (first use in this function)`

A: Use `./configure --with-included-gettext`.

Q: When I run PVM3 it just waits for a few minutes without displaying the command prompt, then prints `Terminated` and exits without any other explanation. No additional error messages can be found in the logs.

A: This seems to be a weird feature / bug in PVM3. The usual cause of this behavior is that the contents of `/etc/hostname` on one of the machines in the cluster is not equal to the DNS hostname of that machine. For example: If you can ping a machine in the cluster using the hostname `node4.cluster.lan`, then the contents of the `/etc/hostname` on that machine should be `node4` and nothing else.

Q: What do `Domain '12' for resource type 'room' is empty` and similar errors mean?

A: The direct meaning of this error message is that Tablix detected an inconsistency in the problem description. Your timetabling problem does not have a solution because one or more events can not be assigned to any defined room. This error should be followed with some more error messages that should give you some pointers on where to look for the cause.

In practice this error often means that you have made a mistake in one of the restrictions. For example a typographical error in a `capability` restriction can cause this.

Chapter 8. Legal

8.1. Contact

Please direct any questions, suggestions and bug reports to the tablix-list mailing list. Instructions on how to subscribe and unsubscribe are available are available from the Tablix home page (http://www.tablix.org/articles/list.html).

Author can be contacted at `<tomaz.solc@tablix.org>`.

8.2. Copyright

TABLIX, PGA general timetable solver. Copyright (c) 2002-2006 Tomaž Šolc.

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307 USA

TinyScheme, Copyright (c) 2000, Dimitrios Souflis. All rights reserved.

Exact distribution terms can be found in the appendix.

Appendix A. GNU General Public License

A.1. Preamble

The licenses for most software are designed to take away your freedom to share and change it. By contrast, the GNU General Public License is intended to guarantee your freedom to share and change free software - to make sure the software is free for all its users. This General Public License applies to most of the Free Software Foundation's software and to any other program whose authors commit to using it. (Some other Free Software Foundation software is covered by the GNU Library General Public License instead.) You can apply it to your programs, too.

When we speak of free software, we are referring to freedom, not price. Our General Public Licenses are designed to make sure that you have the freedom to distribute copies of free software (and charge for this service if you wish), that you receive source code or can get it if you want it, that you can change the software or use pieces of it in new free programs; and that you know you can do these things.

To protect your rights, we need to make restrictions that forbid anyone to deny you these rights or to ask you to surrender the rights. These restrictions translate to certain responsibilities for you if you distribute copies of the software, or if you modify it.

For example, if you distribute copies of such a program, whether gratis or for a fee, you must give the recipients all the rights that you have. You must make sure that they, too, receive or can get the source code. And you must show them these terms so they know their rights.

We protect your rights with two steps:

1. copyright the software, and
2. offer you this license which gives you legal permission to copy, distribute and/or modify the software.

Also, for each author's protection and ours, we want to make certain that everyone understands that there is no warranty for this free software. If the software is modified by someone else and passed on, we want its recipients to know that what they have is not the original, so that any problems introduced by others will not reflect on the original authors' reputations.

Finally, any free program is threatened constantly by software patents. We wish to avoid the danger that redistributors of a free program will individually obtain patent licenses, in effect making the program proprietary. To prevent this, we have made it clear that any patent must be licensed for everyone's free use or not licensed at all.

The precise terms and conditions for copying, distribution and modification follow.

A.2. TERMS AND CONDITIONS FOR COPYING, DISTRIBUTION AND MODIFICATION

A.2.1. Section 0

This License applies to any program or other work which contains a notice placed by the copyright holder saying it may be distributed under the terms of this General Public License. The "Program", below, refers to any such program or work, and a "work based on the Program" means either the Program or any derivative work under copyright law: that is to say, a work containing the Program or a portion of it, either verbatim or with modifications and/or translated into another language. (Hereinafter, translation is included without limitation in the term "modification".) Each licensee is addressed as "you".

Activities other than copying, distribution and modification are not covered by this License; they are outside its scope. The act of running the Program is not restricted, and the output from the Program is covered only if its contents constitute a work based on the Program (independent of having been made by running the Program). Whether that is true depends on what the Program does.

A.2.2. Section 1

You may copy and distribute verbatim copies of the Program's source code as you receive it, in any medium, provided that you conspicuously and appropriately publish on each copy an appropriate copyright notice and disclaimer of warranty; keep intact all the notices that refer to this License and to the absence of any warranty; and give any other recipients of the Program a copy of this License along with the Program.

You may charge a fee for the physical act of transferring a copy, and you may at your option offer warranty protection in exchange for a fee.

A.2.3. Section 2

You may modify your copy or copies of the Program or any portion of it, thus forming a work based on the Program, and copy and distribute such modifications or work under the terms of Section 1 above, provided that you also meet all of these conditions:

a. You must cause the modified files to carry prominent notices stating that you changed the files and the date of any change.

b. You must cause any work that you distribute or publish, that in whole or in part contains or is derived from the Program or any part thereof, to be licensed as a whole at no charge to all third parties under the terms of this License.

c. If the modified program normally reads commands interactively when run, you must cause it, when started running for such interactive use in the most ordinary way, to print or display an

announcement including an appropriate copyright notice and a notice that there is no warranty (or else, saying that you provide a warranty) and that users may redistribute the program under these conditions, and telling the user how to view a copy of this License. (Exception: If the Program itself is interactive but does not normally print such an announcement, your work based on the Program is not required to print an announcement.)

These requirements apply to the modified work as a whole. If identifiable sections of that work are not derived from the Program, and can be reasonably considered independent and separate works in themselves, then this License, and its terms, do not apply to those sections when you distribute them as separate works. But when you distribute the same sections as part of a whole which is a work based on the Program, the distribution of the whole must be on the terms of this License, whose permissions for other licensees extend to the entire whole, and thus to each and every part regardless of who wrote it.

Thus, it is not the intent of this section to claim rights or contest your rights to work written entirely by you; rather, the intent is to exercise the right to control the distribution of derivative or collective works based on the Program.

In addition, mere aggregation of another work not based on the Program with the Program (or with a work based on the Program) on a volume of a storage or distribution medium does not bring the other work under the scope of this License.

A.2.4. Section 3

You may copy and distribute the Program (or a work based on it, under Section 2 in object code or executable form under the terms of Sections 1 and 2 above provided that you also do one of the following:

a. Accompany it with the complete corresponding machine-readable source code, which must be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

b. Accompany it with a written offer, valid for at least three years, to give any third party, for a charge no more than your cost of physically performing source distribution, a complete machine-readable copy of the corresponding source code, to be distributed under the terms of Sections 1 and 2 above on a medium customarily used for software interchange; or,

c. Accompany it with the information you received as to the offer to distribute corresponding source code. (This alternative is allowed only for noncommercial distribution and only if you received the program in object code or executable form with such an offer, in accord with Subsection b above.)

The source code for a work means the preferred form of the work for making modifications to it. For an executable work, complete source code means all the source code for all modules it contains, plus any associated interface definition files, plus the scripts used to control compilation and installation of the executable. However, as a special exception, the source code distributed need not

include anything that is normally distributed (in either source or binary form) with the major components (compiler, kernel, and so on) of the operating system on which the executable runs, unless that component itself accompanies the executable.

If distribution of executable or object code is made by offering access to copy from a designated place, then offering equivalent access to copy the source code from the same place counts as distribution of the source code, even though third parties are not compelled to copy the source along with the object code.

A.2.5. Section 4

You may not copy, modify, sublicense, or distribute the Program except as expressly provided under this License. Any attempt otherwise to copy, modify, sublicense or distribute the Program is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

A.2.6. Section 5

You are not required to accept this License, since you have not signed it. However, nothing else grants you permission to modify or distribute the Program or its derivative works. These actions are prohibited by law if you do not accept this License. Therefore, by modifying or distributing the Program (or any work based on the Program), you indicate your acceptance of this License to do so, and all its terms and conditions for copying, distributing or modifying the Program or works based on it.

A.2.7. Section 6

Each time you redistribute the Program (or any work based on the Program), the recipient automatically receives a license from the original licensor to copy, distribute or modify the Program subject to these terms and conditions. You may not impose any further restrictions on the recipients' exercise of the rights granted herein. You are not responsible for enforcing compliance by third parties to this License.

A.2.8. Section 7

If, as a consequence of a court judgment or allegation of patent infringement or for any other reason (not limited to patent issues), conditions are imposed on you (whether by court order, agreement or otherwise) that contradict the conditions of this License, they do not excuse you from the conditions of this License. If you cannot distribute so as to satisfy simultaneously your obligations under this License and any other pertinent obligations, then as a consequence you may not distribute the Program at all. For example, if a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.

If any portion of this section is held invalid or unenforceable under any particular circumstance, the balance of the section is intended to apply and the section as a whole is intended to apply in other circumstances.

It is not the purpose of this section to induce you to infringe any patents or other property right claims or to contest validity of any such claims; this section has the sole purpose of protecting the integrity of the free software distribution system, which is implemented by public license practices. Many people have made generous contributions to the wide range of software distributed through that system in reliance on consistent application of that system; it is up to the author/donor to decide if he or she is willing to distribute software through any other system and a licensee cannot impose that choice.

This section is intended to make thoroughly clear what is believed to be a consequence of the rest of this License.

A.2.9. Section 8

If the distribution and/or use of the Program is restricted in certain countries either by patents or by copyrighted interfaces, the original copyright holder who places the Program under this License may add an explicit geographical distribution limitation excluding those countries, so that distribution is permitted only in or among countries not thus excluded. In such case, this License incorporates the limitation as if written in the body of this License.

A.2.10. Section 9

The Free Software Foundation may publish revised and/or new versions of the General Public License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns.

Each version is given a distinguishing version number. If the Program specifies a version number of this License which applies to it and "any later version", you have the option of following the terms and conditions either of that version or of any later version published by the Free Software Foundation. If the Program does not specify a version number of this License, you may choose any version ever published by the Free Software Foundation.

A.2.11. Section 10

If you wish to incorporate parts of the Program into other free programs whose distribution conditions are different, write to the author to ask for permission. For software which is copyrighted by the Free Software Foundation, write to the Free Software Foundation; we sometimes make exceptions for this. Our decision will be guided by the two goals of preserving the free status of all derivatives of our free software and of promoting the sharing and reuse of software generally.

A.2.12. NO WARRANTY Section 11

BECAUSE THE PROGRAM IS LICENSED FREE OF CHARGE, THERE IS NO WARRANTY FOR THE PROGRAM, TO THE EXTENT PERMITTED BY APPLICABLE LAW. EXCEPT WHEN OTHERWISE STATED IN WRITING THE COPYRIGHT HOLDERS AND/OR OTHER PARTIES PROVIDE THE PROGRAM "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESSED OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE. THE ENTIRE RISK AS TO THE QUALITY AND PERFORMANCE OF THE PROGRAM IS WITH YOU. SHOULD THE PROGRAM PROVE DEFECTIVE, YOU ASSUME THE COST OF ALL NECESSARY SERVICING, REPAIR OR CORRECTION.

A.2.13. Section 12

IN NO EVENT UNLESS REQUIRED BY APPLICABLE LAW OR AGREED TO IN WRITING WILL ANY COPYRIGHT HOLDER, OR ANY OTHER PARTY WHO MAY MODIFY AND/OR REDISTRIBUTE THE PROGRAM AS PERMITTED ABOVE, BE LIABLE TO YOU FOR DAMAGES, INCLUDING ANY GENERAL, SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES ARISING OUT OF THE USE OR INABILITY TO USE THE PROGRAM (INCLUDING BUT NOT LIMITED TO LOSS OF DATA OR DATA BEING RENDERED INACCURATE OR LOSSES SUSTAINED BY YOU OR THIRD PARTIES OR A FAILURE OF THE PROGRAM TO OPERATE WITH ANY OTHER PROGRAMS), EVEN IF SUCH HOLDER OR OTHER PARTY HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES.

END OF TERMS AND CONDITIONS

A.3. How to Apply These Terms to Your New Programs

If you develop a new program, and you want it to be of the greatest possible use to the public, the best way to achieve this is to make it free software which everyone can redistribute and change under these terms.

To do so, attach the following notices to the program. It is safest to attach them to the start of each source file to most effectively convey the exclusion of warranty; and each file should have at least the "copyright" line and a pointer to where the full notice is found.

<one line to give the program's name and a brief idea of what it does.> Copyright (C) <year> <name of author>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

Also add information on how to contact you by electronic and paper mail.

If the program is interactive, make it output a short notice like this when it starts in an interactive mode:

Gnomovision version 69, Copyright (C) year name of author Gnomovision comes with ABSOLUTELY NO WARRANTY; for details type "show w". This is free software, and you are welcome to redistribute it under certain conditions; type "show c" for details.

The hypothetical commands "show w" and "show c" should show the appropriate parts of the General Public License. Of course, the commands you use may be called something other than "show w" and "show c"; they could even be mouse-clicks or menu items--whatever suits your program.

You should also get your employer (if you work as a programmer) or your school, if any, to sign a "copyright disclaimer" for the program, if necessary. Here is a sample; alter the names:

Yoyodyne, Inc., hereby disclaims all copyright interest in the program "Gnomovision" (which makes passes at compilers) written by James Hacker.

<signature of Ty Coon>, 1 April 1989 Ty Coon, President of Vice

This General Public License does not permit incorporating your program into proprietary programs. If your program is a subroutine library, you may consider it more useful to permit linking proprietary applications with the library. If this is what you want to do, use the GNU Library General Public License instead of this License.

Appendix B. TinyScheme license terms