

TOrPEDO Manual

1 Preliminaries

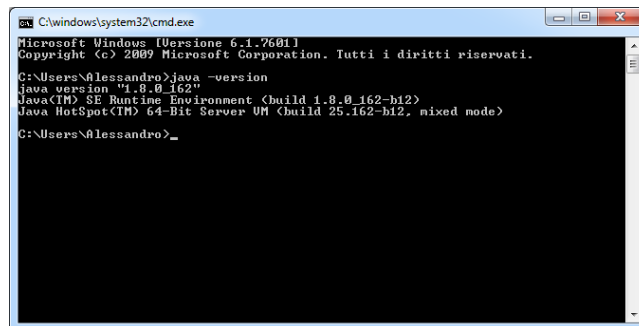
In this section we describe the steps required to prepare the system to run the TOrPEDO tool, namely installing JRE and Docker on Microsoft Windows, Mac OS X, and GNU/LINUX operating systems. The *torpedo.jar* application can be found in the same zip archive as this manual. Alternatively, if the user wishes to use TOrPEDO from an already installed and configured setting, a VirtualBox virtual machine with TOrPEDO is available at <https://bit.ly/2K7D1NI> (username: “ubuntu”, password: “torpedo”).

1.1 Microsoft Windows

1.1.1 Install JRE

1. Download from ¹ the JRE matching your system (we recommend using `jre-8u162-windows-x64.exe`/`jre-8u162-windows-i586.exe` for 64bit/32bit respectively).
2. Install the JRE from the downloaded installer.
3. Open a command prompt (usually, it is sufficient to open the **Start** menu and type `cmd`) and check that JRE is correctly installed by typing `java -version`. You should see something like Fig. 1.

¹<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

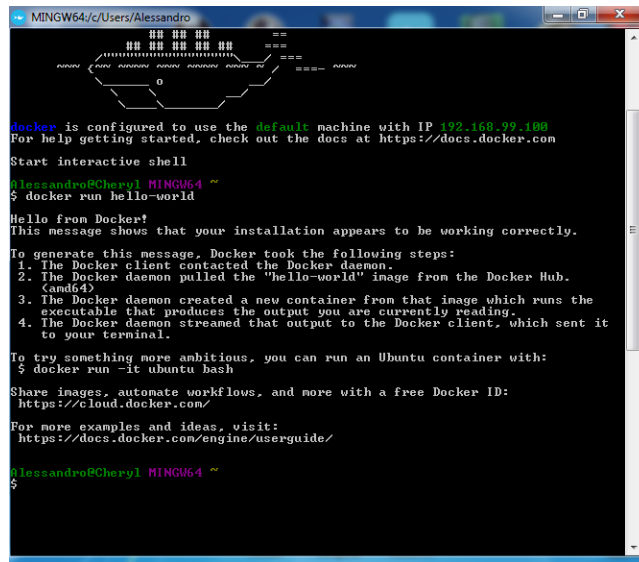


```
C:\windows\system32\cmd.exe
Microsoft Windows [Versione 6.1.7601]
Copyright (c) 2009 Microsoft Corporation. Tutti i diritti riservati.

C:\Users\Alessandro>java -version
java version "1.8.0_162"
Java(TM) SE Runtime Environment (build 1.8.0_162-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.162-b12, mixed mode)

C:\Users\Alessandro>_
```

Figure 1: Sample of java version



```

MINGW64/c/Users/Alessandro
docker is configured to use the default machine with IP 192.168.99.100
For help getting started, check out the docs at https://docs.docker.com
Start interactive shell
Alessandro@Cheryl MINGW64 ~
$ docker run hello-world

Hello from Docker!
This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:
1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.
   (amd64)
3. The Docker daemon created a new container from that image which runs the
   executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it
   to your terminal.

To try something more ambitious, you can run an Ubuntu container with:
$ docker run -it ubuntu bash

Share images, automate workflows, and more with a free Docker ID:
https://cloud.docker.com/

For more examples and ideas, visit:
https://docs.docker.com/engine/userguide/

Alessandro@Cheryl MINGW64 ~
$
```

Figure 2: Sample of docker run

1.1.2 Install Docker

In order to correctly run TORPEDO you should install Docker. For Windows 10, a complete guide is available at ². For older versions of Windows, Docker Toolbox is required. The complete guide is available at ³, here we summarize the main steps.

1. Download from ⁴ Docker Toolbox.
2. Install Docker Toolbox from the downloaded installer.
3. Launch **Docker QuickStart** from the Desktop icon. Wait until the booting process completes.
4. Check that Docker is correctly installed by typing in the terminal: `docker run hello-world`. You should see something like Fig. 2.

1.2 Mac OS X

1.2.1 Install JRE

1. Download from ⁵ the JRE matching your system (we recommend using `jre-8u162-macosx-x64.dmg`).

²<https://docs.docker.com/docker-for-windows/install/>

³https://docs.docker.com/toolbox/toolbox_install_windows/

⁴<https://download.docker.com/win/stable/DockerToolbox.exe>

⁵<http://www.oracle.com/technetwork/java/javase/downloads/index.html>

2. Install the JRE from the downloaded installer.

1.2.2 Install Docker

In order to correctly run TORPEDO you should install Docker. For Mac OS X 10.11 and later, a complete guide is available at ⁶, we summarize here the main steps.

1. Download from ⁷ Docker.
2. Install Docker from the downloaded installer.
3. Drag Docker application to the Application folder.
4. Double-click the Docker application. Wait until a pop-up on the status bar notifies that Docker is correctly running.)
5. Check that Docker is correctly installed by typing in the terminal:
`docker run hello-world.`

1.3 GNU/Linux

The steps to be followed depends on the considered distribution. The reported instructions refer to Ubuntu 17.10.

1.3.1 Install JRE

1. Open a terminal window.
2. Type: `sudo apt install openjdk-8-jre.`
3. Provide the user password.
4. Check that JRE is correctly installed by typing `java -version.`

1.3.2 Install Docker

1. Open a terminal window.
2. Type: `sudo apt install docker.io.`
3. Provide the user password.
4. Start the docker daemon, by typing:
`sudo systemctl start docker.service.`
5. Add the current user to the `docker` group by typing: `sudo usermod -aG docker $USER`
6. Check that Docker is correctly running by typing `docker run hello-world.`

⁶<https://docs.docker.com/docker-for-mac/install/>

⁷<https://download.docker.com/mac/stable/Docker.dmg>

2 Running TOrPEDO

In this section we describe the steps required to run TOrPEDO .

2.1 Microsoft Windows

1. Launch **Docker QuickStart** from the Desktop icon. Wait until the prompt becomes available.
2. Change directory (using the `cd` command) to the one containing the `torpedo.jar` file.
3. Run TOrPEDO by typing:

```
java -jar torpedo.jar analysis [options] <PKS.xml> <property.ltl>
```

or

```
java -jar torpedo.jar recheck <PKS1.xml> <slice.xml>
```

in the prompt.

2.2 Mac OS X

1. Make sure that the Docker daemon is running by checking the presence of the Docker icon in the status bar.
2. Open a terminal window.
3. Change directory (using the `cd` command) to the one containing the `torpedo.jar` file.
4. Run TOrPEDO by typing:

```
java -jar torpedo.jar analysis [options] <PKS.xml> <property.ltl>
```

or

```
java -jar torpedo.jar recheck <PKS1.xml> <slice.xml>
```

in the terminal.

2.3 GNU/Linux

1. Open a terminal window.
2. Make sure that the Docker daemon is running. If not, start it with the appropriate command (for Ubuntu 17.10 `sudo systemctl start docker.service`).
3. Change directory (using the `cd` command) to the one containing the `torpedo.jar` file.
4. Run TOrPEDO by typing:

```
java -jar torpedo.jar analysis [options] <PKS.xml> <property.ltl>
```

or

```
java -jar torpedo.jar recheck <PKS1.xml> <slice.xml>
```

in the terminal.

3 Basic TOrPEDO usage

TOrPEDO provides two functionalities: *analysis* and *recheck*. Analysis checks the satisfaction of a property in Linear-time Temporal Logic (LTL) on a Partial Kripke Structure (PKS) model. The property can be satisfied/possibly satisfied/not satisfied on the PKS. The tool eventually provides a proof of satisfaction or a counterexample showing a violation of the property. Recheck checks if a proof produced for a PKS M can still be applied to a new PKS M' derived from M .

3.1 Analysis

The simplest way to run TOrPEDO is the following:

```
java -jar torpedo.jar analysis <PKS XML file> <LTL property file>
```

With this command the tool is run on the PKS contained in `<PKS XML file>` for the properties contained in `<LTL property file>`.

The PKS is passed as an XML file in a format inspired by the one of tool χ Check. The property is provided in a textual file containing an LTL formula following the grammar shown in Fig. 3. In this grammar `'&'`, `'|'`, `'~'`, `'<->'`, `'->'` represent logical and, or, not, double implication, and implication respectively. Instead, `'B'`, `'U'`, `'W'`, `'X'`, `'F'`, `'G'` represent temporal operators before, until, weak-until, next, eventually, and globally respectively.

The aforementioned usage provides only the verification result without producing either counterexample or proof. To obtain these, some options between `analysis` and `<PKS XML file>` must be added.

```

⟨expression⟩ ::= ⟨implication⟩ (‘<->’ ⟨implication⟩)*
⟨implication⟩ ::= ⟨disjunction⟩ (‘->’ ⟨disjunction⟩)*
⟨disjunction⟩ ::= ⟨conjunction⟩ (‘|’ ⟨conjunction⟩)*
⟨conjunction⟩ ::= ⟨temporal⟩ (‘&’ ⟨temporal⟩)*
⟨temporal⟩ ::= ⟨unary⟩ (‘B’|‘U’|‘W’) ⟨temporal⟩ | ⟨unary⟩
⟨unary⟩ ::= (‘X’|‘F’|‘G’|‘~’) ⟨unary⟩ | ⟨atom⟩
⟨atom⟩ ::= ‘(’ ⟨expression⟩ ‘)’ | ‘True’ | ‘False’ | IDENTIFIER

```

Figure 3: Property grammar

To obtain a counterexample, add the option `-c <ce file>`. This option enables the counterexample generation; the counterexample will be placed in `<ce file>`.

To obtain a proof, add the option `-o <proof file>`. This option enables the proof generation; the proof will be placed in `<proof file>`. It is also possible to produce the proof as a PKS, using the option `-t <slice file>`. In this way, the proof contained in `<slice file>` can be used while running the recheck functionality of the tool.

Note that the first time the tool is run some Docker containers must be installed. Thus, the computer must be connected to the internet and the first run may take longer than the following ones.

3.2 Recheck

The usage of recheck functionality is the following:

```
java -jar torpedo.jar recheck <PKS XML file> <Slice XML file>
```

where `<PKS XML file>` contains the new PKS; `<Slice XML file>` contains the previously obtained proof as a PKS. There are no additional options.

4 Advanced TOrPEDO analysis options

We now describe some advanced options applicable for the analysis phase.

It is possible to specify which specific solver instance should be used to obtain the proof, by adding the option `-s solver`, where `solver` is the solver to be used. Currently two solvers are available: `pltlmup` and `hybrid`. The default one is `hybrid`.

It is possible, for debugging purposes, to obtain the input given to the external tools, with the option `-i <prefix>`. Using this option, the input of the external tools is saved in files starting with `<prefix>`.

It is possible, for debugging purposes, to obtain the whole output of the external tools, with the option `-l <prefix>`. Using this option, the output of the external tools is saved in files starting with `<prefix>`.