Die Installation der t.Sprachen

Achim Clausing

1. August 2011

Inhaltsverzeichnis

→ Die blauen Textstellen sind externe, die roten interne Links.

1 Überblick

Die t.Sprachen können Sie von der Webseite `http://cs.uni-muenster.de/tanagra` herunterladen. Klicken Sie auf den Download-Button, um die Archivdatei `tanagra.zip` auf Ihren Rechner zu kopieren. Sie ist 0,7 MB groß, entpackt 1,8 MB.

Die Software installiert sich nicht ganz automatisch; es ist ein bisschen (tatsächlich sehr wenig) Handarbeit gefragt. Der Mehraufwand verhilft aber letztlich zu einem besseren Verständnis der internen Abläufe.

Die Installation verläuft in drei Schritten. Einige Einzelheiten dieser Schritte hängen von Ihrem Betriebssystem ab, sie werden anschließend erläutert. Zunächst die summarische Beschreibung:

1. Stellen Sie sicher, dass auf Ihrem Rechner eine Java-Laufzeitumgebung (JRE) installiert ist.
2. Entpacken Sie die Datei `tanagra.zip` in ein Verzeichnis Ihrer Wahl.
3. Setzen Sie Ihre Pfadvariable so, dass die Kommandos zum Aufruf der t.Sprachen gefunden werden.

Auf der folgenden Seite wird genauer gesagt, was dafür zu tun ist.

2 Die Installation im Detail

1. Zum Testen Ihrer Java-Umgebung öffnen Sie ein Terminal (eine Shell): unter Windows `cmd.exe` (Windows-Taste + R, dann `cmd` eingeben), unter Linux z. B. `xterm` und unter Mac OS das Menü Programme→Dienstprogramme→Terminal. Tippen sie den Befehl `java -version` ein. Sie sollten etwas in der folgenden Art sehen:

   ```
   C:\Users\meier> java -version
   java version "1.6.0_26"
   Java(TM) SE Runtime Environment (build 1.6.0_26-b03)
   Java HotSpot(TM) 64-Bit Server VM (build 20.1-022, mixed mode, sharing)
   ```

   Es muss mindestens ein JRE der Version 1.6 installiert sein; die „minor number“ (hier `0_26`) spielt keine Rolle.

   Es kann vorkommen, dass der Kommandointerpreter Java nicht findet, obwohl auf dem Rechner ein JRE vorhanden ist. Dann müssen Sie herausfinden, in welchem Verzeichnis der Befehl `java` – genauer: die ausführbare Datei `java.exe` (Windows) bzw. `java` (Linux/Mac OS) – steht und dieses Verzeichnis in Ihre Pfadvariable schreiben. Wie das geht, wird in Abschnitt 4 erklärt.

   Wenn Sie kein JRE haben, müssen Sie Java nachinstallieren. Tipps dafür in finden Sie in Abschnitt 5 (und haufenweise im Internet).

2. Beim Herunterladen von der Webseite der t.Sprachen werden Sie von Ihrem Download-Programm gefragt, ob Sie die Datei `tanagra.zip` speichern oder entpacken möchten. Wenn Sie sich für Speichern entschieden haben, können Sie das Entpacken nachholen, indem Sie in einem Dateimanager `tanagra.zip` markieren und auf „Alle Dateien extrahieren“ oder etwas Ähnliches klicken.[1]

3. In dem entpackten Tanagra-Ordner gibt es die beiden Dateien `install_Tanagra_for_Windows` und `install_Tanagra_for_Linux_or_MacOS`. Klicken Sie die für Ihr Betriebssystem richtige Datei an.[2] Dadurch wird eine Umgebungsvariable `TANAGRA` erzeugt und die Pfadvariable `PATH` um das Verzeichnis ergänzt, in dem die ausführbaren Befehle von Tanagra stehen.

Damit ist die Installation fertig. Die ausführbaren Dateien mit den Kommandos `tzero`, `tlisp`, `tpascal`, `tscheme`, `tlambda`, `tjava` und `tprolog` stehen in den Unterverzeichnissen `bat` (für Windows) bzw. `bin` (für Linux und Mac OS) des Tanagra-Ordners. Steuern Sie Ihren Browser dorthin und klicken Sie sie an.

Sie können auch ein Terminal öffnen und eines der Kommandos eingeben, dann startet der entsprechende Interpreter. Falls Sie wider Erwarten doch eine Fehlermeldung bekommen, können Sie in Abschnitt 7 nach einem Tipp zur Behebung suchen.

Sie können die t.Sprachen jederzeit wieder deinstallieren, indem Sie das Verzeichnis `tanagra` und die Umgebungsvariable `TANAGRA` löschen.

[1] Alternativ können Sie ein Terminal öffnen. Mit dem Befehl `cd C:\Users\meier` – oder wo immer `tanagra.zip` gespeichert wurde – wechseln Sie in den Tanagra-Ordner und entpacken dort die Zip-Datei „von Hand“ mit dem Befehl `unzip tanagra.zip`.

[2] Auch hier gibt es die Alternative, die Installationsdatei als Befehl anstatt durch Anklicken auszuführen. Manche Dateimanager sind nämlich aus Sicherheitsgründen so eingestellt, dass sich mit ihnen ausführbare Dateien nicht durch Anklicken ausführen lassen. In diesem Fall öffnen Sie ein Terminal, wechseln in den Ordner `tanagra` und geben den Befehl `install_Tanagra_for_Windows` bzw. `install_Tanagra_for_Linux_or_MacOS` ein.

3 Wie werden die t.Sprachen aufgerufen?

Es wird kein eigenes Desktop-Icon für die t.Sprachen eingerichtet – sieben neue Icons wären ein bisschen viel auf einmal. Wenn Sie möchten, können Sie natürlich Verknüpfungen (für Mac-Benutzer: Aliasse) zu den Befehlen in `bat` bzw. `bin` auf Ihren Desktop legen.

Machen Sie ein Terminal auf[3] und geben Sie, zum Beispiel, `tzero` ein. Der Interpreter für t.Zero meldet sich mit seinem Prompt. Jetzt können Sie die Sprache im interaktiven Modus ausprobieren. Alle Interpreter kann man mit `(quit)` oder durch das Drücken von `Strg-C` beenden.

Die Beispiele aus dem Buch[4] finden Sie kapitelweise gesammelt im Unterverzeichnis `examples` des Ordners `tanagra`. Kopieren Sie sich die Datei `Kapitel1.tzero` in Ihr momentanes Arbeitsverzeichnis als, beispielsweise, `Kap1.tzero`. Möchten Sie alle Beispiele auf einmal probieren, geben Sie `tzero Kap1.tzero` ein. Ein paar Seiten Ausgaben werden an Ihnen vorbeirauschen, von denen Sie nicht viel erkennen. Führen Sie `Kap1.tzero` deshalb besser mit dem Kommando

```
tzero Kap1.tzero > Kap1.session
```

aus. Dann werden alle Ausgaben in die Datei `Kap1.session` geschrieben, die sie anschließend in Ihrem Lieblingseditor lesen können. Der Inhalt von `Kap1.session` sollte mit dem von `Kapitel1.session` im Ordner `examples` übereinstimmen. Nachdem der t.Zero-Interpreter die Ausdrücke in `Kap1.tzero` abgearbeitet hat, wartet das Programm nicht auf weitere Eingaben, sondern beendet sich selbst. Das können Sie ändern, indem Sie die letzte Zeile in `Kap1.tzero`, die Eingabe `(exit)`, entfernen.

Die jeweils aktuelle Eingabedatei, mit der Sie arbeiten – zum Beispiel `Kap1.tzero` – darf beliebig heißen, es ist keine besondere Endung vorgeschrieben. Ich halte die Datei, an der ich gerade arbeite, normalerweise in einem zweiten Fenster im Editor geöffnet. Nach einer Änderung (speichern nicht vergessen!) hole ich im Interpreter-Fenster mit der ↑-Taste den Befehl zum Ausführen der Datei wieder – hier also `tzero Kap1.tzero`. Auf diese Weise sieht man sofort die Wirkung der Änderung. Es lohnt meistens nicht, größere Ausdrücke im interaktiven Modus in das Interpreterfenster einzugeben. Der Umweg über das Editorfenster stellt sicher, dass alle wesentlichen Eingaben auch gespeichert sind.

Apropos Editor: Welchen Editor man wählt, ist zwar im Prinzip egal, in der Praxis aber doch ziemlich wesentlich. Wer für die Java-Programmierung, das Programmieren in den t.Sprachen, das Schreiben von Emails, für die Erstellung von html-Seiten und LaTeX-Dokumenten und was es sonst so an textorientierten Arbeiten gibt, jeweils einen anderen Editor verwendet, der oder die muss sich ständig umstellen. Besser ist es, man gewöhnt sich an einen einzigen Universaleditor. Er sollte zumindest das Syntax-Highlighting von Java-Programmen und das „Matchen“ von öffnenden und schließenden Klammern beherrschen. Die Klassiker sind Vim und Emacs. Bei beiden ist der Lernaufwand etwas größer als bei einem Wysiwyg-Editor, aber die Investition zahlt sich aus. Versuchen Sie im Internet herauszufinden, welcher Editor für Ihre Zwecke geeignet ist. Für Einsteiger, die keinen neuen Editor lernen wollen, ist in der Windows-Welt der Editor Notepad++ zu empfehlen. Er erfordert praktisch keine Umgewöhnung.

[3] Das `cmd`-Terminal von Windows startet wie eh und je mit schwarzem Hintergrund. Man kann Farbe, Größe und Schrift aber einstellen (Rechtsklick auf die Kopfzeile, Menüpunkt „Eigenschaften“). Als Alternative kann man die PowerShell aufrufen, sie kennt die notwendigen Umgebungsvariablen auch.

[4] Achim Clausing: Programmiersprachen – Konzepte, Strukturen und Implementierung in Java. Spektrum Akademischer Verlag, 2011.

4 Was geschieht bei der Installation?

Die grafische Oberfläche des Rechners ist eine Art Karosserie, unter der die technischen Einzelheiten verborgen sind. Den Umgang mit Computern ohne diese Bequemlichkeit würden die meisten von uns wohl als Zumutung empfinden. Manchmal ist es aber doch nützlich, unter das Blech zu schauen. Deshalb soll hier ganz knapp erläutert werden, was bei der Installation abläuft.

Shells

Vorweg ein paar Bemerkungen zu den heute von „Normal"-Benutzern nur noch wenig verwendeten *Kommandointerpretern*, auch *Terminals* oder *Shells* genannt. Es gibt sie für alle Betriebssysteme:

- Auf Windows-Systemen den Interpreter `cmd.exe` (zu dem es eine Einführung in `cmd` als Wikibuch gibt) und die für Einsteiger nicht ganz einfach zu handhabende PowerShell.
- Auf Apple-Rechnern existiert das *Terminal* (siehe z. B. Mac OS X für Profis, Kapitel 1, von Michael Kofler).
- In der Unix-Welt, zu der Linux und genau besehen auch Apples Mac OS X gehören, gibt es Shells in fast beliebig großer Auswahl. Der Standard-Kommandointerpreter unter Unix ist die `bash`. Eine sehr lesbares und informatives Tutorial zur `bash` findet man im Kapitel 4 der Einführung in Linux von Michael Redinger.

Ein Kommandointerpreter läuft auf einem Rechner mit grafischer Benutzerschnittstelle in einem Fenster. In diesem Fenster wird ein Programm ausgeführt, das Befehle zur Steuerung des Rechners entgegen nimmt; nur dieses Programm ist genau genommen die Shell. Wer sich ernsthaft mit dem befassen will, was bei Computern „unter der Motorhaube" los ist, kommt um den Einsatz einer Shell nicht herum.

Eine Shell ist bei genauerem Hinsehen nichts anderes als eine Programmiersprache. Ein *Shell-Skript* ist ein Programm in dieser Sprache. Die Dateien `install_Tanagra_for_Windows.bat` und `install_Tanagra_for_Linux_or_MacOS` zur Installation der t.Sprachen enthalten (sehr einfache) Programme, die von `cmd.exe` bzw. einer `bash` ausgeführt werden. Um sie auszuführen, öffnet man ein entsprechendes Terminal und gibt den Namen der Datei mit den Kommandos ein. Unter Windows kann man `install_Tanagra_for_Windows.bat` auch aus einem Dateimanager, etwa dem Explorer, anklicken. Dieser erkennt anhand der Dateiendung `.bat`, dass es sich um eine Stapeldatei (einen *batch file*) handelt, also eine Liste von Kommandos für den Interpreter `cmd.exe`.

Kommandos sind entweder in die Shell *eingebaut* oder sie werden im Filesystem *gesucht*, d. h. es wird nach einer ausführbaren Datei mit dem Namen des Kommandos gesucht. Es wäre allerdings wenig sinnvoll, die ganze Festplatte abzugrasen, bis irgendwo eine Datei gefunden wird, die (vielleicht nur zufällig) denselben Namen hat wie das Kommando. Die Shell durchsucht deshalb nur die Verzeichnisse, die in der *Pfadvariablen* eingetragen sind. Sie ist eine der *Umgebungsvariablen* Ihres Systems.

Umgebungsvariablen

Umgebungsvariablen enthalten Werte, von denen das Verhalten eines Computers abhängt. Man sieht diese Informationen im Normalbetrieb nicht, aber sie werden ständig benutzt und sind von zentraler Bedeutung, weil sie den *Kontext* schaffen, in dem alle Programme ablaufen.

Auf Windows- und Unix-Rechnern kann man in einer Shell mit dem Befehl `set` abfragen, welche Umgebungsvariablen momentan definiert sind und was für Werte sie haben.

Eine zentrale Umgebungsvariable ist die Pfadvariable PATH. Sie enthält den *Pfad*, die Liste der Verzeichnisse, in denen Befehle gesucht werden. In einem Unix-Rechner könnte sie beispielsweise so aussehen:

```
/home/meier> echo $PATH
.:/opt/kde3/bin:/usr/lib64/mpi/gcc/openmpi/bin:/usr/local/bin:/usr/bin:/bin:/usr/bin/X11:
/usr/X11R6/bin:/usr/games:/opt/real/RealPlayer:/home/meier/bin
```

Die Variable heißt zwar `PATH`, wenn man ihren Wert verwenden will, muss man aber ein Dollarzeichen voranstellen. Das gilt für alle Umgebungsvariablen. Die einzelnen Verzeichnisse werden in Unix durch Doppelpunkte voneinander getrennt. Das Verzeichnis „`.`“ am Anfang des Pfads steht für das jeweilige Arbeitsverzeichnis.

Auf Windows-Rechnern heißt die Pfadvariable `Path`, ihr Wert wird als `%Path%` geschrieben. Windows unterscheidet – anders als Unix – bei Umgebungsvariablen nicht zwischen Groß- und Kleinschreibung, man könnte auch `%PATH%` opder `%path%` schreiben. Trennzeichen für die Verzeichnisse im Pfad ist bei Windows das Semikolon:

```
C:\Users\meier> echo %Path%
Path=C:\Windows\SYSTEM32;C:\Windows;C:\Windows\SYSTEM32\WBEM;C:\Windows\SYSTOEM32\WINDOWS
POWERSHELL\V1.0\;C:\PROGRAM FILES\INTEL\WIFI\BIN\;C:\PROGRAM FILES\COMMON FILES\INTEL\WIR
ELESSCOMMON\;C:\PROGRAM FILES (X86)\COMMON FILES\LENOVO;C:\PROGRAM FILES (X86)\INTEL\SERV
ICES\IPT\;C:\PROGRAM FILES (X86)\LENOVO\ACCESS CONNECTIONS\;C:\Program Files\Java\jdk1.6.
0_26\bin\
```

Eine andere wichtige Umgebungsvariable ist `HOME` (alle Unix-Systeme) bzw. `USERPROFILE` (seit Windows Vista, bis XP hieß sie `HOMEPATH`). Diese Variable gibt den Ordner an, in dem sich das persönliche Verzeichnis des Benutzers befindet:

```
C:\> echo %USERPROFILE%
C:\Users\meier
```

Das jeweilige Arbeitsverzeichnis steht in der Variablen `CD` (Windows) bzw. `PWD` (Unix). Viele Shells schreiben es in ihren Eingabeprompt, sodass man sehen kann, wo man sich gerade befindet.

Wie definiert man eine Umgebungsvariable? In einer `cmd`-Shell unter Windows geht das mit `setx`:

```
setx TANAGRA C:\Users\meier\tanagra
```

Ab sofort hat die Variable `TANAGRA` den Wert `C:\Users\meier\tanagra`. Man kann Umgebungsvariablen auch ohne Shell über die Menüs von Windows setzen. Dazu ruft man in Windows 7 das Menü Start→Systemsteuerung auf, sucht nach dem Stichwort „Umgebungsvariablen“ und findet den Menüpunkt „Umgebungsvariablen für dieses Konto bearbeiten”. Damit kann man Variablen setzen, korrigieren und auch wieder entfernen. (In anderen Windows-Versionen heißt der Menüpunkt teilweise etwas anders.)

Unix-Umgebungsvariablen definiert man, indem man `export NAME=`$\langle Wert \rangle$ in eine Shell eingibt:

```
export TANAGRA=$HOME/tanagra
```

Wenn das Arbeitsverzeichnis im Moment `$HOME/tanagra` ist, wie das bei der Installation der t.Sprachen der Fall sein sollte, dann ist der vorangehende Befehl offensichtlich gleichwertig mit

```
export TANAGRA=$PWD
```

Die Definition einer Windows-Umgebungsvariablen, ob mit `setx` oder über das System-Menü, wird in die Windows-Registry eingetragen.

Bei Unix-Systemen gibt es keine Registry. Das Installationsskript trägt stattdessen den Befehl zum Setzen der persönlichen Umgebungsvariablen in die Datei `$HOME/.bashrc` ein. Weil ihr Name mit einem Punkt beginnt, ist diese Datei normalerweise verborgen – man sieht sie im Dateimanager nur, wenn man „verborgene Dateien anzeigen“ angeklickt hat und in einer Shell nur, wenn man `ls -a` (*list all*) eingibt. Die Datei `.bashrc` wird bei jedem Start einer Bash von dieser gelesen und ausgeführt.

Der Pfad zu den t.Sprachen

Mit dem Setzen der Variablen `TANAGRA` allein ist es noch nicht getan. Erst wenn das Unterverzeichnis mit den ausführbaren Befehlen `tzero`, `tlisp` etc. „im Pfad“ steht, können diese aufgerufen werden.

Zu diesem Zweck erweitert das Installationsskript die Pfadvariable. Unter Windows erreicht es das mit dem Befehl

```
setx PATH "%TANAGRA%\bat"
```

Die Anführungszeichen um die neue Variable darf man nicht weglassen. Die Variable `PATH` wird dabei nicht überschrieben; alle vom System in `PATH` geschriebenen Verzeichnisse bleiben erhalten. In jeder nach dieser Zuweisung an `PATH` neu gestarteten Shell steht dann `C:\Users\meier\tanagra\bat` „im Pfad“, das ist der Ordner mit den Batch-Dateien `tzero.bat`, `tlisp.bat` etc. Sehen Sie sich Ihren Pfad in einer `cmd`-Shell mit dem Kommando `echo %PATH%` an, um zu kontrollieren, dass die Variable richtig gesetzt ist!

Die Unix-Version sieht analog aus: Beim Installieren wird in die Datei `.bashrc` im Verzeichnis `$HOME` etwas ähnliches wie die Zeile

```
export PATH=/home/meier/tanagra/bin:$PATH
```

geschrieben. Man sollte daran denken, den bisherigen Wert `$PATH` an `PATH` mit zuzuweisen, sonst wird der alte Pfad überschrieben. Beim nächsten Start einer Bash zeigt die Eingabe `echo $PATH` den erweiterten Suchpfad an.

5 Java installieren

JRE und JDK

Mit „Java“ kann zweierlei gemeint sein: Ein JRE (Java Runtime Environment), die Laufzeitumgebung für Java-Programme oder ein JDK (Java Development Kit), die Entwicklungsumgebung für Java. Ein JRE gibt Ihnen die Möglichkeit, Java-Programme auszuführen; mit einem JDK haben Sie zusätzlich die Möglichkeit, Java-Quellcode zu übersetzen.

Um die t.Sprachen zu benutzen, genügt ein JRE. Wenn Sie etwas daran verändern möchten, brauchen Sie einen Java-Compiler, also ein JDK.

Um zu prüfen, was auf Ihrem Rechner vorhanden ist, gibt es mehrere Möglichkeiten. Am einfachsten ist es, ein Kommandofenster zu öffnen und die beiden Befehle

```
java -version
```

und

```
javac -version
```

einzugeben. Wenn der erste davon erfolgreich ist, kennt Ihr Rechner das Kommando `java` zum Ausführen von Java-Programmen; ist zusätzlich die angezeigte Versionsnummer nicht kleiner als 1.6, so sind die t.Sprachen ausführbar. Der zweite Befehl, `javac`, bezieht sich auf den Java-Compiler. Stimmt auch hier die Version, so können Sie die t.Sprachen neu kompilieren. Für den Anfang ist das aber nicht nötig.

Sie können das Vorhandensein eines JRE (nur das, nicht des JDK) auch über das Internet testen. Gehen Sie auf die Webseite `http://www.java.com/de/download/installed.jsp` und klicken Sie auf den roten Button „Java-Version überprüfen“.

Die Installation von Java

Wenn Sie kein Java oder nur eines mit einer zu niedrigen Versionsnummer haben, können Sie auf die Webseite `http://www.java.com/de/download` gehen und „Kostenloser Java-Download“ anklicken. Dann wird auf Ihrem Rechner ein JRE 1.6 installiert.

Ich würde Ihnen allerdings empfehlen, gleich die Java-Entwicklungsumgebung zu holen. Unter der URL `http://www.oracle.com/technetwork/java/javase/downloads` finden Sie das jeweils neuste JDK. Sie können es auch installieren, wenn auf Ihrem Rechner schon eine ältere Java-Version vorhanden ist.

In beiden Fällen sollten Sie Administrator-Rechte haben. Das kann man zwar umgehen, aber während der Installation werden Sie nach dem Administrator-Passwort gefragt; halten Sie es also besser bereit.

Wenn Sie als Windows-Benutzer nach der Installation ein `cmd`-Fenster aufmachen und mit der Eingabe `java -version` testen, ob alles geklappt hat, werden Sie vermutlich feststellen, dass die Shell das Kommando `java` nicht findet. Der Grund ist wieder die Pfadvariable. Sie müssen das Verzeichnis, in dem Ihre Java-Befehle stehen, in den Pfad aufnehmen[5].

[5] oder in eines der schon im Pfad stehenden Verzeichnisse Links auf die entsprechenden Befehle schreiben – aber das ist nicht einfacher.

Zu diesem Zweck gehen Sie genauso vor wie in Abschnitt 4: Geben Sie

```
setx PATH "%JAVA%\bin;%TANAGRA%\bat"
```

in ein Terminal ein, bzw. das Folgende, wenn Sie ein JDK installiert haben:

```
setx PATH "%JAVAC%\bin;%TANAGRA%\bat"
```

Die Umgebungsvariablen `JAVA` bzw. `JAVAC` werden bei der Installation des JRE bzw. des JDK automatisch erzeugt. Wenn Sie von anderen Quellen installiert haben, müssen Sie nachsehen, ob evtl. Variablen wie `JAVA_ROOT`, `JAVA_HOME`, `JAVA_BINDIR`, `SDK_HOME` oder `JDK_HOME` definiert sind. Sie können auch in das Verzeichnis `C:\` wechseln und mit `dir /s C:\java.exe` selbst suchen, in welchem Ordner die ausführbare Datei für den Befehl `java` steht.

Unter Linux und Mac OS müssen die Verzeichnisse mit den Java-Binaries normalerweise nicht in der Pfadvariablen vorkommen. Hier werden bei der Installation symbolische Links auf die Java-Befehle in das Verzeichnis `/usr/bin` geschrieben, das bei jedem System im Pfad steht.

Auf welchem Betriebssystem auch immer: Testen Sie den Befehl `java -version` in einer Shell. Wenn ein Java 1.6 oder 1.7 gemeldet wird, ist alles in Ordnung.

Hinweise zur Installation von Java gibt es in beliebiger Zahl im Internet. Eine detaillierte Anleitung zur Installation des JDK finden Sie z. B. hier.

6 Tanagra neu kompilieren

Früher oder später werden Sie an den t.Sprachen etwas ändern wollen. Der Eingabeprompt gefällt Ihnen nicht, Sie wollen Leerzeichen in Bezeichnern zulassen, ein Operator oder ein Datentyp fehlt, oder Sie möchten den Interpreter an der einen oder anderen Stelle effizienter machen.

Gehen Sie in einen der Ordner `sources/tanagra` oder `sources/expression`, ändern oder ergänzen Sie die entsprechende Java-Datei und übersetzen Sie das ganze System mit `buildtanagra`. Dazu muss ein JDK installiert sein!

Erschrecken Sie nicht, wenn Ihnen der Compiler mehrere Warnungen der Form

```
/home/meier/tanagra/sources/tanagra/Sys.java:9: warning: sun.misc.Signal is Sun proprietary
API and may be removed in a future release
```

um die Ohren wirft. Das sind nicht-abstellbare Meldungen; warum Sun-Oracle nicht erlaubt, sie mit `@SuppressWarnings` zu unterdrücken, ist ein im Internet reichlich diskutiertes Firmengeheimnis. Tatsache ist, dass das Paket `sun.misc.Signal` nicht zum offiziellen Java-API gehört, obwohl es damit verteilt wird, und dass Programme, die es verwenden, auf manchen JVMs nicht laufen. Durch Auskommentieren einiger Zeilen am Anfang von `sources/tanagra/Sys.java` kann man die Verwendung dieses Pakets deaktivieren. Allerdings lassen sich dann „amoklaufende“ Programme in den t.Sprachen nicht mehr mit `Strg-C` beenden, ohne dass gleich der ganze Interpreter beendet wird.

7 Frequently Asked Questions

1. Q: Durch Anklicken der Datei `install_Tanagra_for_Windows.bat` im Explorer wurde eine Umgebungsvariable `TANAGRA` mit einem Verzeichnis erzeugt, das auf meinem Rechner nicht gar existiert.

   A: Möglicherweise haben Sie `install_Tanagra_for_Windows.bat` angeklickt, ohne vorher `tanagra.zip` zu entpacken. Die Datei `tanagra.zip` wird nämlich im Explorer genauso dargestellt wie das Verzeichnis `tanagra`, das sie in gepackter Form enthält. Entpacken Sie zuerst `tanagra.zip` und klicken Sie `install_Tanagra_for_Windows.bat` in dem entpackten Verzeichnis an.

2. Q: Ich habe ein älteres Windows und bei mir klappt die Installation einfach nicht – weder durch Anklicken von `install_Tanagra_for_Windows.bat` noch mit einem `cmd`-Terminal. Wenn ich es in einem Terminal versuche, kommt die Fehlermeldung `Der Befehl "setx" ist entweder falsch geschrieben oder konnte nicht gefunden werden`.

   A: Den Befehl `setx`, der von `install_Tanagra_for_Windows.bat` benutzt wird, gibt es erst seit Windows XP SP2. Auf älteren Windows-Systemen können Sie entweder die Umgebungsvariablen `TANAGRA` und `PATH` über das Menü „Start→Arbeitsplatz→Eigenschaften→Erweitert→Umgebungsvariablen“ von Hand setzen (mühsam) oder aber den Befehl `setx` nachinstallieren (einfacher). Oder Sie leisten sich ein Update Ihres Betriebssystems.

3. Q: Ich habe Tanagra unter Linux installiert, aber die Eingabe `tzero` ergibt trotzdem die Meldung „`command not found`“. Was stimmt nicht?

   A: Haben Sie nach der Installation kein neues Terminal aufgemacht? Die Datei `.bashrc`, in der die Pfadvariablen gesetzt werden, wird nur beim Start einer Bash gelesen. Ihr Inhalt kann deshalb erst beim nächsten Öffnen eines Terminals wirksam werden.

   Haben Sie mit `echo $PATH` Ihre Pfadvariable kontrolliert?

   Oder haben Sie `tzero` eingegeben und nicht `tZero`? Unix unterscheidet – anders als Windows – bei Befehlen zwischen Groß- und Kleinschreibung.

4. Q: Ich bekomme beim Aufruf von `tzero` die Fehlermeldung `java.lang.UnsupportedClassVersionError`.

   A: Die t.Sprachen funktionieren nur ab Java 1.6. Installieren Sie ein aktuelles Java.

5. Q: Ich bekomme beim Aufruf von `tzero` die Fehlermeldung `Permission denied` (betroffen sind Linux und Mac OS).

   A: Die Zugriffsrechte für die Datei `tanagra/bin/tzero` sind falsch gesetzt. Öffnen Sie ein Terminal, wechseln Sie in das Verzeichnis `tanagra/bin` und geben Sie `chmod u=rwx tzero` ein.

6. Q: Beim Aufruf rekursiv definierter Funktionen in t.Lisp bekomme ich ganz schnell die Meldung `Error: Stack overflow` (manchmal auch `Error: Not enough memory`). Was kann ich tun?

   A: Setzen Sie in der ausführbaren Datei `tanagra/bin/tscheme` (Linux, Mac OS) bzw. in `tanagra\bat\tscheme.bat` (Windows) die Option `-Xss16M` herauf. Diese Option steuert die Größe des Stacks in der virtuellen Java-Maschine, die beim Start eines Java-Programms

erzeugt wird. Mit `java -X` können Sie sich Optionen von Java anzeigen lassen, mit denen Sie die anfängliche und die maximale Größe des Stacks und des Heaps ändern können.

7. Q: Ich bekomme beim Aufruf von `tscheme` den Fehler `java.lang.OutOfMemoryError: unable to create new native thread`.

   A. Haben Sie evtl. in der Datei `tscheme` (Linux, Mac OS) bzw. in `tscheme.bat` (Windows) die Option `-Xss16M` zu sehr heraufgesetzt?