BESCHREIBUNG DER MYPDDL-DATEIEN

In dieser Beschreibung möchte ich kurz auf die Implementierung von myPddl eingehen, d. h. einen Überblick über die Dateien geben und ihre Bedeutung kurz erklären. Details (und Kommentare) zur Implementierung können dann in der jeweiligen Datei gefunden werden. Einen kleinen Überblick über die Bestandteile meiner Bachelorarbeit (und die Beschreibung, die du gerade liest) findest du auf:
https://github.com/Pold87/ba-thesis/
Bei Fragen stehe ich jederzeit gerne zur Verfügung:
Volker Strobel
volker.strobel87@gmail.com

SYNTAX HIGHLIGHTING (MYPDDL-SYNTAX)

Ein kurzes Tutorial zum Erstellen von Sublime-Text-Syntax-Definitionen findet man auf:
http://docs.sublimetext.info/en/sublime-text-2/extensibility/syntaxdefs.html

pddl.YAML-tmLanguage
pddl.tmLanguage

Die Datei `pddl.YAML-tmLanguage`[1] beinhaltet die Syntaxdefinitionen für PDDL in YAML[2]. Durch reguläre Ausdrücke[3] wird jeder PDDL-Ausdruck erkannt und kann somit benannt werden. Außerdem werden reguläre Ausdrücke durch die Definition von *Repositories* in andere reguläre Ausdrücke eingebunden. Aus der YAML-Datei wird Hilfe von AAAPackageDev[4] eine Sublime-Text-Syntax-Definition im Plist-Format erstellt und in der Datei `pddl.tmLanguage`[5] gespeichert. Ein schönes Beispiel für Syntax-

1 https://github.com/Pold87/myPDDL/blob/master/syntax/pddl.YAML-tmLanguage
2 http://www.yaml.org/
3 Sublime Text verwendet reguläre Ausdrücke im Oniguruma-Format: http://www.geocities.jp/kosako3/oniguruma/doc/RE.txt
4 https://github.com/SublimeText/AAAPackageDev
5 https://github.com/Pold87/myPDDL/blob/master/syntax/pddl.tmLanguage

definitionen findet man auf:
https://github.com/SublimeText/AAAPackageDev/blob/master/Syntax%20Definitions/Sublime%20Text%20Syntax%20Def%20(YAML).YAML-tmLanguage

Comments.tmPreferences

In der Datei `Comments.tmPreferences`[6] wird festgelegt, mit welchem Zeichen ein Kommentar in PDDL beginnt. Dadurch kann man in Sublime Text mit dem Befehl `Toggle Comment` eine Zeile auskommentieren (bzw. wieder einkommentieren).

completion.sublime-completions

requirements.sublime-completions

Die Datei `completion.sublime-completions`[7] beinhaltet Schlüsselwörter, die durch die PDDL-Auto-Completion-Funktion eingefügt werden können. Zusätzlich befinden sich in der Datei `requirements.sublime-completions`[8] Completions für den PDDL-Requirements-Block.

CODE SNIPPETS (MYPDDL-SNIPPET)

Snippets-Ordner

Im Snippets-Ordner[9] befinden sich die PDDL-Grundgerüste. Jedes Snippet ist in einer Datei gespeichert, in der der Trigger (d.h. das Schlüsselwort, das zum Einfügen verwendet wird) und der Inhalt festgelegt werden. Durch die Definition von Platzhaltern (`${n:...}`) kann man dadurch in Sublime Text von Feld zu Feld springen (mit der Tab-Taste).

CLOJURE

Die Grundfunktionen von MYPDDL sind, mit Ausnahme des Syntax Highlighters und der Code Snippets, in Clojure implementiert. Die Basis dafür bildet ein Interface zwischen PDDL und Clojure.[10] Dieses Interface kann als Bibliothek in Clojure benutzt werden um PDDL-Dateien zu lesen und zu erweitern (die Methoden und Kommentare findet man in der Datei `core.clj`[11]. Die Library ist auf Clojars (Repository für Open Source Clojure Libraries) zur Verfügung gestellt[12], so dass sie einfach

pddl_clojure_interface/core.clj

6 https://github.com/Pold87/myPDDL/blob/master/syntax/Comments.tmPreferences
7 https://github.com/Pold87/myPDDL/blob/master/syntax/completion.sublime-completions
8 https://github.com/Pold87/myPDDL/blob/master/syntax/requirements.sublime-completions
9 https://github.com/Pold87/myPDDL/tree/master/snippet
10 https://github.com/Pold87/pddl-clojure-interface
11 https://github.com/Pold87/pddl-clojure-interface/blob/master/src/pddl_clojure_interface/core.clj
12 https://clojars.org/pddl-clojure-interface

in PDDL-Projekte eingebunden werden kann. Es gibt auch noch eine Standalone-Version[13], damit man die Funktionen in einer Unix-Shell verwenden kann. Auf den jeweiligen GitHub-Seiten findet man weitere Informationen zur Installation und zur Verwendung.

myPDDL[14] verwendet nun dieses Interface[15], um damit mit PDDL zu interagieren. Die Implementierungen von myPDDL-new, -diagram, -distance findet man in der Datei `core.clj`[16]. Aus dem Clojure-Projekt wird dann mit Hilfe von Leiningen[17] eine JAR-Datei erstellt, die unabhängig von Clojure verwendet werden kann (vorausgesetzt man hat eine Java Virtual Machine installiert). Um den Aufruf dieser JAR-Datei zu vereinfachen, habe ich noch einen Shell-Wrapper geschrieben[18]. Fügt man den Pfad dieses Wrappers zu `$PATH` hinzu und macht diese Datei ausführbar (`chmod a+x myPDDL`) können myPDDL-new, -diagram und -distance überall in der Shell aufgerufen werden (mit `myPDDL ...`, z.B. `myPDDL diagram domain.pddl`). Durch diesen Wrapper können nun zahlreiche Anpassungen vorgenommen werden, ohne den Clojure-Source-Code zu editieren. Setzt man z.B. die Variable `revisioncontrol` auf `no` wird beim Aufruf von myPDDL-diagram keine Kopie der Domain-Datei gespeichert, durch Änderungen der Variable `viewer` wird festgelegt, mit welchem Programm das erstellte Diagramm angezeigt werden soll. Dort kann dann z.B. auch festgelegt werden, welche Vorlagen myPDDL-new (PDDL-Projekt-Erstellung) benutzt. Als Standard benutzt es die Templates im Ordner *Templates*[19]. Diese Templates können auch beliebig verändert werden, da es sich um einfache Textdateien handelt. myPDDL-new liest diese Dateien ein (in der Methode `create-PDDL-file`) und erstellt daraus die Code-Gerüste in den neuen Projekten (`domain-name` in den Templates wird dabei durch den Namen des neuen Projekts ersetzt).

mypddl/core.clj

mypddl-1.0.0.jar

myPDDL

Templates-Ordner

13 Programm: https://github.com/Pold87/pddl-clojure-interface-standalone
Quellcode: https://github.com/Pold87/pddl-clojure-interface-standalone-source

14 Source Code: https://github.com/Pold87/myPDDL-source/

15 Ich habe das Interface absichtlich nicht als Teil von myPDDL konzipiert, weil es für weitere Anwendungen dienen soll. myPDDL stellt einfach einen Weg dar, dieses Interface zu verwenden.

16 https://github.com/Pold87/myPDDL-source/blob/master/src/mypddl/core.clj

17 http://leiningen.org/

18 https://github.com/Pold87/myPDDL/blob/master/myPDDL

19 https://github.com/Pold87/myPDDL/tree/master/templates

myPddl-IDE

myPddl-IDE ist die integrierte Entwicklungsumgebung für myPddl, um alle Funktionen in Sublime Text verwenden zu können.

PDDL.sublime-settings

In der Datei `PDDL.sublime-settings`[20] werden Einstellungen festgelegt, die verwendet werden, wenn in Sublime Text eine pddl-Datei geöffnet wird.

Da pddl Lisp-basiert ist, habe ich mich dabei an den Clojure Style Guide gehalten "Use two spaces per indentation level. No hard tabs."[21]

PDDL.sublime-commands

`PDDL.sublime-commands` wird verwendet, um die myPddl-Tools in der Sublime Text Command Palette anzuzeigen.

messages.json

Die json-Datei `messages.json`[22] bestimmt, welche Nachricht angezeigt wird, nachdem myPddl mit Sublime Text Package Control[23] installiert wurde.

mypddl.py

In der Python-Datei `mypddl.py`[24] wird Sublime Text mit den anderen Tools von myPddl verknüpft. Außerdem werden spezifische Einstellungen für myPddl getroffen, die nur für Sublime Text gelten. Durch Threading wird sichergestellt, dass Sublime-Text nicht hängenbleibt, wenn die externen Tools aufgerufen werden.

20 https://github.com/Pold87/myPDDL/blob/master/PDDL.sublime-settings
21 https://github.com/bbatsov/clojure-style-guide
22 https://github.com/Pold87/myPDDL/blob/master/messages.json
23 Mit Package Control lassen sich Sublime Packages einfach installieren und verwalten: https://sublime.wbond.net/
24 https://github.com/Pold87/myPDDL/blob/master/mypddl.py