# **TextWorld Documentation**

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Ta

# **CONTENTS**

1	What is TextWorld?	3
2	Known Issues	5
3	tw-play	7
4	tw-make	9
5	tw-view	17
6	tw-extract	19
7	A Simple Game	23
8	Coin Collector	25
9	The Cooking Game	27
10	Treasure Hunter	29
11	textworld	31
12	textworld.gym	37
13	textworld.envs	47
14	textworld.agents	53
15	textworld.generator	55
16	textworld.challenges	101
17	textworld.logic	103
18	textworld.render	119
19	textworld.utils	123
20	Indices and tables	125
Bil	bliography	127
Py	thon Module Index	129

Index 131

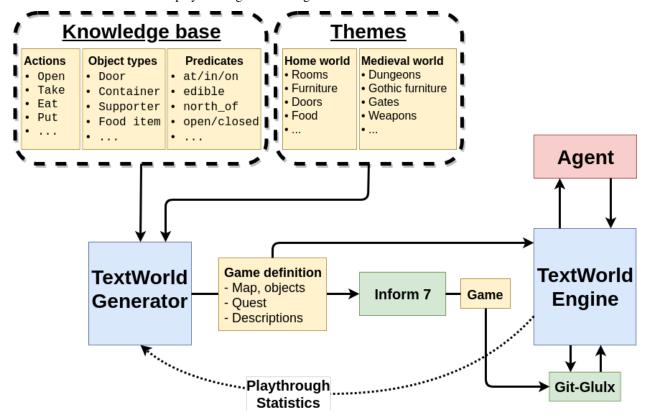
TextWorld is a text-based learning environment for Reinforcement Learning agent.

CONTENTS 1

2 CONTENTS

## WHAT IS TEXTWORLD?

TextWorld is a sandbox learning environment for training and testing reinforcement learning (RL) agents on text-based games. It enables generating games from a game distribution parameterized by the map size, the number of objects, quest length and complexity, richness of text descriptions, and more. Then, one can sample game from that distribution. TextWorld can also be used to play existing text-based games.



# **TWO**

# **KNOWN ISSUES**

# 2.1 Inform 7

Inform 7 command line tools don't support Windows Linux Subsystem (a.k.a Bash on Ubuntu on Windows).

## THREE

## TW-PLAY

Play a TextWorld game (.z8 or .ulx).

```
usage: tw-play [-h] [--mode MODE] [--max-steps STEPS] [--viewer [PORT]]
[--hint] [-v] [-vv]
game
```

# 3.1 Positional Arguments

game

# 3.2 Named Arguments

**--mode** Possible choices: random, human, random-cmd, walkthrough

Select an agent to play the game: ['random', 'human', 'random-cmd', 'walk-

through']. Default: "human".

Default: "human"

**--max-steps** Limit maximum number of steps.

Default: 0

**--viewer** Start web viewer.

**--hint** Display the oracle trajectory leading to winning the game.

Default: False

**-v, --verbose** Verbose mode.

Default: False

-vv, --very-verbose Print debug information.

8 Chapter 3. tw-play

## **FOUR**

### TW-MAKE

```
usage: tw-make [-h] [--third-party PATH]
{custom,tw-coin_collector,tw-treasure_hunter,tw-simple,tw-cooking}
...
```

# 4.1 Named Arguments

**--third-party** Load an external python file. Useful to register custom challenges on-the-fly.

# 4.2 Types of game to create

**subcommand** Possible choices: custom, tw-coin\_collector, tw-treasure\_hunter, tw-simple, tw-cooking

## 4.3 Sub-commands:

#### 4.3.1 custom

Make a custom game.

#### **Custom game settings**

**--world-size** Nb. of rooms in the world.

Default: 5

**--nb-objects** Minimum nb. of objects in the world.

Default: 10

#### **Grammar settings**

**--theme** Theme to use for generating the text. Default: "house"

Default: "house"

**--include-adj** Turn on adjectives.

Default: False

**--blend-descriptions** Blend descriptions across consecutive sentences.

Default: False

**--ambiguous-instructions** Refer to an object using its type (e.g. red container vs. red chest).

Default: False

**--only-last-action** Intruction only describes the last action of quest.

Default: False

**--blend-instructions** Blend instructions across consecutive actions.

Default: False

--entity-numbering Append a number after an entity name if there is not enough variation for it (e.g.

'red apple 2').

Default: False

### **Quest settings**

**--nb-parallel-quests** Nb. of parallel quests the game will have. Default: 1.

Default: 1

**--quest-length** Nb. of actions the quest requires to be completed. It is a shorthand for '-quest-

min-length N -quest-max-length N -quest-max-depth N'.

**--quest-breadth** Nb. of subquests the quests will have. It is a shorthand for '-quest-min-breadth

N –quest-max-breadth N'.

#### Quest settings (advanced)

**--quest-min-length** Minimum nb. of actions the quest requires to be completed. This setting is ignored

if -quest-length is provided. Default: 1.

Default: 1

--quest-max-length Maximum nb. of actions the quest requires to be completed. This setting is ig-

nored if –quest-length is provided. Default: 5.

Default: 5

--quest-min-breadth Minimum nb. of subquests the quests can have. This setting is ignored if -quest-

breadth is provided. Default: 1.

Default: 1

--quest-max-breadth Maxmimum nb. of subquests the quests can have. This setting is ignored if

-quest-breadth is provided. Default: 5.

Default: 5

**--quest-min-depth** Minimum nb. of actions the subquests can have. Default: 1.

Default: 1

**--quest-max-depth** Maximum nb. of actions the subquests can have. This setting is ignored if –quest-

length is provided. Default: 5.

Default: 5

#### **General settings**

**--output** Path where to save the generated game. If it points to a folder, the game's UUID

will be used as the filename.

Default: "./tw\_games/"

--seed

**--format** Possible choices: ulx, z8

Which format to use when compiling the game. Default: "z8"

Default: "z8"

**--overview** Display an overview of the generated game.

Default: False

**--save-overview** Save the overview image of the generated game alongside the game as a PNG file.

Default: False

-f, --force Default: False--silent Default: False-v, --verbose Default: False

4.3. Sub-commands:

### 4.3.2 tw-coin collector

Generate a Coin Collector game

### **Coin Collector game settings**

**--level** The difficulty level. Must be between 1 and 300 (included).

#### **General settings**

**--output** Path where to save the generated game. If it points to a folder, the game's UUID

will be used as the filename.

Default: "./tw\_games/"

--seed

**--format** Possible choices: ulx, z8

Which format to use when compiling the game. Default: "z8"

Default: "z8"

**--overview** Display an overview of the generated game.

Default: False

**--save-overview** Save the overview image of the generated game alongside the game as a PNG file.

Default: False

-f, --force Default: False--silent Default: False-v, --verbose Default: False

## 4.3.3 tw-treasure\_hunter

Generate a Treasure Hunter game

```
tw-make tw-treasure_hunter [-h] --level LEVEL [--output PATH] [--seed SEED]

[--format {ulx,z8}] [--overview] [--save-overview]

[-f] [--silent | -v]
```

#### **Treasure Hunter game settings**

**--level** The difficulty level. Must be between 1 and 30 (included).

#### **General settings**

**--output** Path where to save the generated game. If it points to a folder, the game's UUID

will be used as the filename.

Default: "./tw\_games/"

--seed

**--format** Possible choices: ulx, z8

Which format to use when compiling the game. Default: "z8"

Default: "z8"

**--overview** Display an overview of the generated game.

Default: False

**--save-overview** Save the overview image of the generated game alongside the game as a PNG file.

Default: False

-f, --force Default: False--silent Default: False-v, --verbose Default: False

## 4.3.4 tw-simple

Generate simple challenge game

## Simple game settings

**--rewards** Possible choices: dense, balanced, sparse

The reward frequency: dense, balanced, or sparse.

**--goal** Possible choices: detailed, brief, none

The description of the game's objective shown at the beginning of the game: de-

tailed, bried, or none

**--test** Whether this game should be drawn from the test distributions of games.

Default: False

4.3. Sub-commands:

#### **General settings**

**--output** Path where to save the generated game. If it points to a folder, the game's UUID

will be used as the filename.

Default: "./tw\_games/"

--seed

**--format** Possible choices: ulx, z8

Which format to use when compiling the game. Default: "z8"

Default: "z8"

**--overview** Display an overview of the generated game.

Default: False

**--save-overview** Save the overview image of the generated game alongside the game as a PNG file.

Default: False

-f, --force Default: False--silent Default: False-v, --verbose Default: False

## 4.3.5 tw-cooking

Generate cooking games similar to those used for the First TextWorld Problem (FTWP) competition (https://aka.ms/ftwp).

#### The Cooking Game settings

**--recipe** Number of ingredients in the recipe. Default: 1

Default: 1

**--take** Number of ingredients to find. It must be less or equal to the value of --recipe.

Default: 0

Default: 0

**--go** Possible choices: 1, 6, 9, 12

Number of locations in the game (1, 6, 9, or 12). Default: 1

Default: 1

**--open** Whether containers/doors need to be opened.

**--cook** Whether some ingredients need to be cooked.

Default: False

**--cut** Whether some ingredients need to be cut.

Default: False

**--drop** Whether the player's inventory has limited capacity.

Default: False

**--recipe-seed** Random seed used for generating the recipe. Default: 0

Default: 0

**--split** Possible choices: train, valid, test

Specify the game distribution to use. Food items (adj-noun pairs) are split in three subsets. Also, the way the training food items can be prepared is further divided

in three subsets.

• train: training food and their corresponding training preparations

• valid: valid food + training food but with unseen valid preparations

• test: test food + training food but with unseen test preparations

Default: game is drawn from the joint distribution over train, valid, and test.

## **General settings**

**--output** Path where to save the generated game. If it points to a folder, the game's UUID

will be used as the filename.

Default: "./tw\_games/"

--seed

**--format** Possible choices: ulx, z8

Which format to use when compiling the game. Default: "z8"

Default: "z8"

**--overview** Display an overview of the generated game.

Default: False

**--save-overview** Save the overview image of the generated game alongside the game as a PNG file.

Default: False

-f, --force Default: False--silent Default: False-v, --verbose Default: False

4.3. Sub-commands:

16 Chapter 4. tw-make

# **FIVE**

# **TW-VIEW**

Display the graph representation of a game's initial state.

usage: tw-view [-h] [-v] game

# **5.1 Positional Arguments**

game JSON file containing infos about the game.

# **5.2 Named Arguments**

**-v, --verbose** Verbose mode.

18 Chapter 5. tw-view

SIX

## **TW-EXTRACT**

Extract information from of a list of TextWorld games.

```
usage: tw-extract [-h] [-f] [--merge] [-q | -v] {vocab,entities,walkthroughs,commands} ...
```

# **6.1 Positional Arguments**

subcommand Possible choices: vocab, entities, walkthroughs, commands

Type of information to extract.

# **6.2 Named Arguments**

-q, --quiet Default: False-v, --verbose Default: False

# 6.3 General settings

**-f, --force** Default: False

**--merge** Merge extracted information to existing output file.

Default: False

## 6.4 Sub-commands:

### 6.4.1 vocab

Extract vocabulary.

```
tw-extract vocab [-h] [-f] [--merge] [-q \mid -v] [--output OUTPUT] [--theme THEME] [game [game ...]]
```

#### **Positional Arguments**

**game** List of TextWorld games (.ulx|.z8|.json).

### **Named Arguments**

-q, --quiet Default: False-v, --verbose Default: False

**--output** Output file containing all words (.txt). Default: "vocab.txt"

Default: "vocab.txt"

**--theme** Provide a text grammar theme from which to extract words.

#### **General settings**

**-f, --force** Default: False

**--merge** Merge extracted information to existing output file.

Default: False

### 6.4.2 entities

Extract entity names.

```
tw-extract entities [-h] [-f] [--merge] [-q \mid -v] [--output OUTPUT] game [game ...]
```

### **Positional Arguments**

**game** List of TextWorld games (.ulx|.z8|.json).

#### **Named Arguments**

-q, --quiet Default: False-v, --verbose Default: False

**--output** Output file containing all entity names (.txt). Default: "entities.txt"

Default: "entities.txt"

### **General settings**

**-f, --force** Default: False

**--merge** Merge extracted information to existing output file.

Default: False

## 6.4.3 walkthroughs

Extract walkthroughs.

```
tw-extract walkthroughs [-h] [-f] [--merge] [-q \mid -v] [--output OUTPUT] game [game ...]
```

### **Positional Arguments**

game List of TextWorld games (.ulx|.json).

### **Named Arguments**

-q, --quiet Default: False-v, --verbose Default: False

**--output** Output file containing all walkthroughs (.txt). Default: "walkthroughs.txt"

Default: "walkthroughs.txt"

## **General settings**

**-f, --force** Default: False

**--merge** Merge extracted information to existing output file.

Default: False

#### 6.4.4 commands

Extract all possible commands.

```
tw-extract commands [-h] [-f] [--merge] [-q \mid -v] [--output OUTPUT] game [game ...]
```

6.4. Sub-commands: 21

## **Positional Arguments**

**game** List of TextWorld games (.ulx|.json).

## **Named Arguments**

-q, --quiet Default: False-v, --verbose Default: False

**--output** Output file containing all commands (.txt). Default: "commands.txt"

Default: "commands.txt"

## **General settings**

**-f, --force** Default: False

**--merge** Merge extracted information to existing output file.

## **SEVEN**

## **A SIMPLE GAME**

This simple game takes place in a typical house and consists in finding the right food item and cooking it. Here's the map of the house.

# 7.1 Usage

```
usage: tw-make tw-simple [-h] --rewards {dense,balanced,sparse} --goal {detailed,brief,none} [--test]
```

## 7.1.1 Simple game settings

**--rewards** Possible choices: dense, balanced, sparse

The reward frequency: dense, balanced, or sparse.

**--goal** Possible choices: detailed, brief, none

The description of the game's objective shown at the beginning of the game: de-

tailed, bried, or none

**--test** Whether this game should be drawn from the test distributions of games.

**EIGHT** 

# **COIN COLLECTOR**

In this type of game, the world consists in a chain of quest\_length rooms with potentially distractors rooms (i.e. leading to a dead end). The agent stats on one end and has to collect a "coin" object which is placed at the other end. There is no other objects present in the world other than the coin to collect.

# 8.1 Usage

usage: tw-make tw-coin\_collector [-h] --level LEVEL

# 8.1.1 Coin Collector game settings

**--level** The difficulty level. Must be between 1 and 300 (included).

**NINE** 

### THE COOKING GAME

This type of game was used for the competition *First TextWorld Problems*<sup>1</sup>. The overall objective of the game is to locate the kitchen, read the cookbook, fetch the recipe's ingredients, process them accordingly, prepare the meal, and eat it. To control the game's difficulty, one can specify the amount of skills that are involved to solve it (see skills section below).

#### References

## 9.1 Usage

```
usage: tw-make tw-cooking [-h] [--recipe INT] [--take INT] [--go {1,6,9,12}]
[--open] [--cook] [--cut] [--drop]
[--recipe-seed INT] [--split {train,valid,test}]
```

## 9.1.1 The Cooking Game settings

**--recipe** Number of ingredients in the recipe. Default: 1

Default: 1

**--take** Number of ingredients to find. It must be less or equal to the value of --recipe.

Default: 0
Default: 0

**--go** Possible choices: 1, 6, 9, 12

Number of locations in the game (1, 6, 9, or 12). Default: 1

Default: 1

**--open** Whether containers/doors need to be opened.

Default: False

**--cook** Whether some ingredients need to be cooked.

Default: False

**--cut** Whether some ingredients need to be cut.

<sup>1</sup> https://aka.ms/ftwp

#### **TextWorld Documentation**

**--drop** Whether the player's inventory has limited capacity.

Default: False

**--recipe-seed** Random seed used for generating the recipe. Default: 0

Default: 0

**--split** Possible choices: train, valid, test

Specify the game distribution to use. Food items (adj-noun pairs) are split in three subsets. Also, the way the training food items can be prepared is further divided in three subsets.

• train: training food and their corresponding training preparations

• valid: valid food + training food but with unseen valid preparations

• test: test food + training food but with unseen test preparations

Default: game is drawn from the joint distribution over train, valid, and test.

**TEN** 

# TREASURE HUNTER

In this type of game, the agent spawns in a randomly generated maze and must find a specific object which is mentioned in the objective displayed when game starts. This is a text version of the task proposed in [Parisotto2017].

#### References

# 10.1 Usage

usage: tw-make tw-treasure\_hunter [-h] --level LEVEL

# 10.1.1 Treasure Hunter game settings

**--level** The difficulty level. Must be between 1 and 30 (included).

## **ELEVEN**

### **TEXTWORLD**

## 11.1 Core

#### exception textworld.core.EnvInfoMissingError(requester, info)

Bases: NameError

Thrown whenever some environment information EnvInfos.

#### exception textworld.core.GameNotRunningError(msg=")

Bases: RuntimeError

Error when game is not running (either has terminiated or crashed).

#### class textworld.core.Agent

Bases: object

Interface for any agent that want to play a text-based game.

act(game\_state, reward, done)

Acts upon the current game state.

#### **Parameters**

- game\_state (GameState) Current game state.
- reward (float) Accumulated reward up until now.
- **done** (bool) Whether the game is finished.

#### Return type str

**Returns** Text command to be performed in this current state.

finish(game\_state, reward, done)

Let the agent know the game has finished.

#### **Parameters**

- **game\_state** (*GameState*) Game state at the moment the game finished.
- reward (float) Accumulated reward up until now.
- **done** (bool) Whether the game has finished normally or not. If False, it means the agent's used up all of its actions.

#### Return type None

### reset(env)

Let the agent set some environment's flags.

**Parameters** env (*Environment*) – TextWorld environment.

#### Return type None

#### property wrappers

#### class textworld.core.EnvInfos(\*\*kwargs)

Bases: object

Customizing what information will be returned by an environment.

Information can be requested by setting one or more attributes to True. The attribute *extras* should be a list of strings corresponding to keys in the metadata dictionary of TextWorld generated games.

#### copy()

#### admissible\_commands

All commands relevant to the current state. This information changes from one step to another.

Type bool

#### property basics: Iterable[str]

Information requested excluding the extras.

**Return type** Iterable[str]

#### command\_templates

Templates for commands understood by the game. This information *doesn't* change from one step to another.

Type bool

#### description

Text description of the current room, i.e. output of the look command. This information changes from one step to another.

Type bool

#### entities

Names of all entities in the game. This information *doesn't* change from one step to another.

Type bool

#### extras

Names of extra information which are game specific.

**Type** List[str]

#### facts

All the facts that are currently true about the world. This information changes from one step to another.

Type bool

### fail\_facts

Mutually exclusive sets of failing facts for each quest. This information *doesn't* change from one step to another.

Type bool

#### feedback

Text observation produced by the game in response to previous command. This information changes from one step to another.

Type bool

#### game

Current game in its serialized form. Use with textworld.Game.deserialize.

### Type bool

#### intermediate\_reward

Reward (proxy) indicating if the player is making progress. This information changes from one step to another.

Type bool

### inventory

Text listing of the player's inventory, i.e. output of the *inventory* command. This information changes from one step to another.

Type bool

### last\_action

The last action performed where None means it was not a valid action. This information changes from one step to another.

Type bool

### last\_command

The last command performed where None means it was not a valid command. This information changes from one step to another.

Type bool

#### location

Name of the player's current location. This information changes from one step to another.

Type bool

#### lost

Whether the player lost the game. This information changes from one step to another.

Type bool

### max\_score

Maximum reachable score of the game. This information doesn't change from one step to another.

Type bool

## moves

Number of moves done so far in the game. This information changes from one step to another.

Type bool

## objective

Objective of the game described in text. This information *doesn't* change from one step to another.

Type bool

### policy\_commands

Sequence of commands leading to a winning state. This information changes from one step to another.

Type bool

#### score

Current score of the game. This information changes from one step to another.

Type bool

### verbs

Verbs understood by the the game. This information *doesn't* change from one step to another.

Type bool

11.1. Core 33

#### win\_facts

Mutually exclusive sets of winning facts for each quest. This information *doesn't* change from one step to another.

Type bool

won

Whether the player won the game. This information changes from one step to another.

Type bool

class textworld.core.Environment(infos=None)

Bases: object

Class allowing to interact with the game's interpreter.

The role of an *Environment* is to handle the communication between user code and the backend interpreter that manages the text-based game. The overall *Environment* structure is highly inspired by OpenAI's gym.

### **Example**

Here's a minimal example of how to interact with an Environment

```
>>> import textworld
>>> options = textworld.GameOptions()
>>> options.seeds = 1234
>>> options.nb_objects = 5
>>> options.quest_length = 2
>>> game_file, _ = textworld.make(options, path='./') # Generate a random game.
>>> env = textworld.start(game_file) # Load the game.
>>> game_state = env.reset() # Start a new game.
>>> env.render()
I hope you're ready to go into rooms and interact with objects, because you've
just entered TextWorld! Here is how to play! First thing I need you to do is to
ensure that the type G chest is open. And then, pick up the keycard from the
type G chest inside the attic. Got that? Good!
-= Attic =-
You arrive in an attic. A normal kind of place. You begin to take stock of
what's in the room.
You make out a type G chest. You can see a TextWorld style locker. The TextWorld
style locker contains a frisbee and a sock.
There is a TextWorld style key on the floor.
>>> command = "take key" # Command to send to the game.
>>> game_state, reward, done = env.step(command)
>>> env.render()
(the TextWorld style key)
You pick up the TextWorld style key from the ground.
```

**Parameters infos** (Optional[*EnvInfos*]) – Information to be included in the game state. By default, only the game's narrative is included.

```
close()
          Ends the game.
              Return type None
     copy()
          Return a copy of this environment at the same state.
              Return type Environment
              Returns A copy of this environment at the same state.
     load(path)
          Loads a new text-based game.
              Parameters path (str) – Path to the game file to load.
              Return type None
     render(mode='human')
          Renders the current state of the game.
              Parameters mode (str) – The mode to use for rendering.
              Return type Optional[str]
     reset()
          Starts game from the beginning.
              Return type GameState
              Returns Initial state of the game.
     seed(seed=None)
          Sets the seed for the random number generator.
              Return type None
     step(command)
          Performs a given command.
              Parameters command (str) – Text command to send to the interpreter.
              Return type Tuple[GameState, float, bool]
              Returns A tuple containing the new game state, a reward for performing that command and reach-
                  ing this new state, and whether the game is finished or not.
     property display_command_during_render: bool
          Enables/disables displaying the command when rendering.
              Return type bool
class textworld.core.GameState
     Bases: dict
     copy()
          Returns a deepcopy of this game state.
              Return type GameState
class textworld.core.Wrapper(env=None)
     Bases: object
```

11.1. Core 35

Special environment that wraps others to provide new functionalities.

Special environment that wraps other *Environment* objects to provide new functionalities (e.g. transcript recording, viewer, etc).

```
Parameters env (Optional[Environment]) – environment to wrap.
close()
        Return type None
copy()
        Return type Wrapper
load(path)
        Return type None
render(mode='human')
        Return type Optional[Any]
reset()
        Return type GameState
seed(seed=None)
        Return type List[int]
step(command)
        Return type Tuple[GameState, float, bool]
property display_command_during_render: bool
        Return type bool
property unwrapped
```

## **TEXTWORLD.GYM**

textworld.gym.utils.register\_game(gamefile, request\_infos=None, batch\_size=None, auto\_reset=False, max\_episode\_steps=50, asynchronous=True, action\_space=None, observation\_space=None, name=", \*\*kwargs)

Make an environment for a particular game.

#### **Parameters**

- gamefile (str) Path for the TextWorld game (\*.ulx|\*.z[1-8]).
- **request\_infos** (Optional[*EnvInfos*]) For customizing the information returned by this environment (see textworld.EnvInfos for the list of available information).

**Warning:** Only supported for TextWorld games (i.e., with a corresponding \*.json file).

• **batch\_size** (Optional[int]) – If provided, it indicates the number of games to play at the same time. By default, a single game is played at once.

**Warning:** When batch\_size is provided (even for batch\_size=1), env.step expects a list of commands as input and outputs a list of states. env.reset also outputs a list of states.

- auto\_reset (bool) If True, each game *independently* resets once it is done (i.e., reset happens on the next env.step call). Otherwise, once a game is done, subsequent calls to env.step won't have any effects.
- max\_episode\_steps (int) Number of steps allocated to play each game. Once exhausted, the game is done.
- **asynchronous** (bool) If True, games in the batch are played in parallel. Only when batch size is greater than one.
- action\_space (Optional[Space]) The action space be used with OpenAI baselines. (see textworld.gym.spaces.Word).
- **observation\_space** (Optional[Space]) The observation space be used with OpenAI baselines (see *textworld.gym.spaces.Word*).
- **name** (str) Name for the new environment, i.e. "tw-{name}-v0". By default, the returned env\_id is "tw-v0".

#### Return type str

**Returns** The corresponding gym-compatible env\_id to use.

### **Example**

```
>>> from textworld.generator import make_game, compile_game
>>> options = textworld.GameOptions()
>>> options.seeds = 1234
>>> game = make_game(options)
>>> game.extras["more"] = "This is extra information."
>>> gamefile = compile_game(game)
>>> import gym
>>> import textworld.gym
>>> from textworld import EnvInfos
>>> request_infos = EnvInfos(description=True, inventory=True, extras=["more"])
>>> env_id = textworld.gym.register_game(gamefile, request_infos)
>>> env = gym.make(env_id)
>>> ob, infos = env.reset()
>>> print(infos["extra.more"])
This is extra information.
```

textworld.gym.utils.register\_games(gamefiles, request\_infos=None, batch\_size=None, auto\_reset=False, max\_episode\_steps=50, asynchronous=True, action\_space=None, observation\_space=None, name=", \*\*kwargs)

Make an environment that will cycle through a list of games.

#### **Parameters**

- **gamefiles** (List[str]) Paths for the TextWorld games (\*.ulx|\*.z[1-8]).
- **request\_infos** (Optional[*EnvInfos*]) For customizing the information returned by this environment (see textworld.EnvInfos for the list of available information).

```
Warning: Only supported for TextWorld games (i.e., with a corresponding *.json file).
```

• batch\_size (Optional[int]) – If provided, it indicates the number of games to play at the same time. By default, a single game is played at once.

**Warning:** When batch\_size is provided (even for batch\_size=1), env.step expects a list of commands as input and outputs a list of states. env.reset also outputs a list of states.

- **auto\_reset** (bool) If True, each game *independently* resets once it is done (i.e., reset happens on the next env.step call). Otherwise, once a game is done, subsequent calls to env.step won't have any effects.
- max\_episode\_steps (int) Number of steps allocated to play each game. Once exhausted, the game is done.
- **asynchronous** (bool) If True, games in the batch are played in parallel. Only when batch size is greater than one.

- action\_space (Optional[Space]) The action space be used with OpenAI baselines. (see textworld.gym.spaces.Word).
- **observation\_space** (Optional[Space]) The observation space be used with OpenAI baselines (see *textworld.gym.spaces.Word*).
- name (str) Name for the new environment, i.e. "tw-{name}-v0". By default, the returned env\_id is "tw-v0".

#### Return type str

**Returns** The corresponding gym-compatible env\_id to use.

## **Example**

```
>>> from textworld.generator import make_game, compile_game
>>> options = textworld.GameOptions()
>>> options.seeds = 1234
>>> game = make_game(options)
>>> game.extras["more"] = "This is extra information."
>>> gamefile = compile_game(game)
>>> import gym
>>> import textworld.gym
>>> from textworld import EnvInfos
>>> request_infos = EnvInfos(description=True, inventory=True, extras=["more"])
>>> env_id = textworld.gym.register_games([gamefile], request_infos)
>>> env = gym.make(env_id)
>>> ob, infos = env.reset()
>>> print(infos["extra.more"])
This is extra information.
```

# 12.1 Agent

class textworld.gym.core.Agent

Bases: object

Interface for any agent playing TextWorld games.

act(obs, score, done, infos)

Acts upon the current list of observations.

One text command must be returned for each observation.

### **Parameters**

- **obs** (str) Previous command's feedback (game's narrative).
- **score** (int) The score obtained so far.
- **done** (bool) Whether the game is finished.
- **infos** (Mapping[str, Any]) Additional information requested.

### Return type str

**Returns** Text command to be performed. If episode has ended (i.e. done is True), the returned value is expected to be ignored.

12.1. Agent 39

```
property infos_to_request: textworld.core.EnvInfos
```

Returns what additional information should be made available at each game step.

Requested information will be included within the infos dictionary passed to *Agent.act()*. To request specific information, create a textworld. EnvInfos and set its attributes to True accordingly.

In addition to the standard information, certain games may have specific information that can be requested via the extras attribute. Refer to the documentation specific to the game to know more (see textworld. challenges).

### **Example**

Here is an example of how to request information and retrieve it.

```
>>> from textworld import EnvInfos
>>> request_infos = EnvInfos(description=True, inventory=True)
...
>>> env = gym.make(env_id)
>>> ob, infos = env.reset()
>>> print(infos["description"])
>>> print(infos["inventory"])
```

Return type EnvInfos

## **12.2 Envs**

 $Bases:\ textworld.gym.envs.textworld\_batch.TextworldBatchGymEnv$ 

Environment for playing text-based games.

#### **Parameters**

- gamefiles (List[str]) Paths of every game composing the pool (\*.ulx|\*.z[1-8]).
- **request\_infos** (Optional[*EnvInfos*]) For customizing the information returned by this environment (see textworld.EnvInfos for the list of available information).

**Warning:** Only supported for TextWorld games (i.e., that have a corresponding \*.json file).

- max\_episode\_steps (Optional[int]) Number of steps allocated to play each game. Once exhausted, the game is done.
- action\_space (Optional[Space]) The action space be used with OpenAI baselines. (see textworld.gym.spaces.Word).
- **observation\_space** (Optional[Space]) The observation space be used with OpenAI baselines (see *textworld.gym.spaces.Word*).

#### reset()

Resets the text-based environment.

Resetting this environment means starting the next game in the pool.

**Return type** Tuple[str, Dict[str, Any]]

### Returns

A tuple (observation, info) where

- observation: text observed in the initial state;
- infos: additional information as requested.

### step(command)

Runs a command in the text-based environment.

**Parameters command** – Text command to send to the game interpreter.

Return type Tuple[str, Dict[str, Any]]

#### Returns

A tuple (observation, score, done, info) where

- observation: text observed in the new state;
- score: total number of points accumulated so far;
- done: whether the game is finished or not;
- infos: additional information as requested.

```
metadata = {'render.modes': ['human', 'ansi', 'text']}
```

class textworld.gym.envs.textworld\_batch.TextworldBatchGymEnv(gamefiles, request\_infos=None,

batch\_size=1, asynchronous=True, auto\_reset=False, max\_episode\_steps=None, action\_space=None, observation\_space=None)

Bases: gym.core.Env

Environment for playing text-based games in batch.

### **Parameters**

- **gamefiles** (List[str]) Paths of every game composing the pool (\*.ulx|\*.z[1-8]|\*. json).
- **request\_infos** (Optional[*EnvInfos*]) For customizing the information returned by this environment (see textworld.EnvInfos for the list of available information).

**Warning:** Only supported for TextWorld games (i.e., that have a corresponding \*.json file).

• batch\_size (int) – If provided, it indicates the number of games to play at the same time. By default, a single game is played at once.

Warning: When batch\_size is provided (even for batch\_size=1), env.step expects a list of commands as input and outputs a list of states. env.reset also outputs a list of states.

12.2. Envs 41

- asynchronous (bool) If True, wraps the environments in an AsyncBatchEnv (which uses multiprocessing to run the environments in parallel). If False, wraps the environments in a SyncBatchEnv. Default: True.
- auto\_reset (bool) If True, each game *independently* resets once it is done (i.e., reset happens on the next env.step call). Otherwise, once a game is done, subsequent calls to env.step won't have any effects.
- max\_episode\_steps (Optional[int]) Number of steps allocated to play each game. Once exhausted, the game is done.
- action\_space (Optional[Space]) The action space be used with OpenAI baselines. (see textworld.gym.spaces.Word).
- **observation\_space** (Optional[Space]) The observation space be used with OpenAI baselines (see textworld.gym.spaces.Word).

### close()

Close this environment.

### Return type None

#### render(mode='human')

Renders the current state of each environment in the batch.

Each rendering is composed of the previous text command (if there's one) and the text describing the current observation.

Parameters mode (str) - Controls where and how the text is rendered. Supported modes are:

- human: Display text to the current display or terminal and return nothing.
- ansi: Return a StringIO containing a terminal-style text representation. The text can include newlines and ANSI escape sequences (e.g. for colors).
- text: Return a string (str) containing the text without any ANSI escape sequences.

Return type Union[StringIO, str, None]

**Returns** Depending on the mode, this method returns either nothing, a string, or a StringIO object.

### reset()

Resets the text-based environment.

Resetting this environment means starting the next game in the pool.

**Return type** Tuple[List[str], Dict[str, List[Any]]]

## Returns

A tuple (observations, infos) where

- observation: text observed in the initial state for each game in the batch;
- infos: additional information as requested for each game in the batch.

#### seed(seed=None)

Set the seed for this environment's random generator(s).

This environment use a random generator to shuffle the order in which the games are played.

**Parameters seed** (Optional[int]) – Number that will be used to seed the random generators.

Return type List[int]

**Returns** All the seeds used to set this environment's random generator(s).

```
skip(nb_games=1) Skip games.
```

**Parameters nb\_games** (int) – Number of games to skip.

Return type None

step(commands)

Runs a command in each text-based environment of the batch.

Parameters commands – Text command to send to the game interpreter.

Return type Tuple[List[str], List[float], List[bool], Dict[str, List[Any]]]

#### **Returns**

A tuple (observations, scores, dones, infos) where

- observations: text observed in the new state for each game in the batch;
- scores: total number of points accumulated so far for each game in the batch;
- dones: whether each game in the batch is finished or not;
- infos: additional information as requested for each game in the batch.

```
metadata = {'render.modes': ['human', 'ansi', 'text']}
```

```
textworld.gym.envs.utils.shuffled_cycle(iterable, rng, nb_loops=- 1)
```

Yield each element of iterable one by one, then shuffle the elements and start yielding from the start. Stop after nb\_loops loops.

#### **Parameters**

- iterable (Iterable[Any]) Iterable containing the elements to yield.
- rng (RandomState) Random generator used to shuffle the elements after each loop.
- **nb\_loops** (int) Number of times to go through all the elements. If set to -1, loop an infinite number of times.

Return type Iterable[Any]

# 12.3 Spaces

### exception textworld.gym.spaces.text\_spaces.VocabularyHasDuplicateTokens

Bases: ValueError

class textworld.gym.spaces.text\_spaces.Char(max\_length, vocab=None, extra\_vocab=[])

Bases: gym.spaces.multi\_discrete.MultiDiscrete

Character observation/action space

This space consists of a series of gym.spaces.Discrete objects all with the same parameters. Each gym.spaces.Discrete can take integer values between 0 and len(self.vocab).

12.3. Spaces 43

#### **Notes**

The following special token will be prepended (if needed) to the vocabulary:

· '#': Padding token

#### **Parameters**

- max\_length (int) Maximum number of characters in a text.
- **vocab** (*list of char*, *optional*) Vocabulary defining this space. It shouldn't contain any duplicate characters. If not provided, the vocabulary will consists in characters [a-z0-9], punctuations ["", "-", """] and padding "#".
- extra\_vocab (list of char, optional) Additional tokens to add to the vocabulary.

### filter\_unknown(text)

Strip out all characters not in the vocabulary.

```
tokenize(text, padding=False)
```

Tokenize characters found in the vocabulary.

Note: text will be padded up to self.max\_length.

```
class textworld.gym.spaces.text_spaces.Word(max_length, vocab)
```

Bases: gym.spaces.multi\_discrete.MultiDiscrete

Word observation/action space

This space consists of a series of gym.spaces.Discrete objects all with the same parameters. Each gym.spaces.Discrete can take integer values between 0 and len(self.vocab).

#### **Notes**

The following special tokens will be prepended (if needed) to the vocabulary:

- '<PAD>': Padding
- '<UNK>': Unknown word
- '<S>': Beginning of sentence
- '': End of sentence

## **Example**

Let's create an action space that can be used with textworld.gym.register\_game. We are going to assume actions are short phrases up to 8 words long.

```
>>> import textworld
>>> gamefiles = ["/path/to/game.ulx", "/path/to/another/game.z8"]
>>> vocab = textworld.vocab.extract_from(gamefiles)
>>> vocab = sorted(vocab) # Sorting the vocabulary, optional.
>>> action_space = textworld.gym.text_spaces.Word(max_length=8, vocab=vocab)
```

#### **Parameters**

• max\_length (int) - Maximum number of words in a text.

• **vocab** (*list of strings*) – Vocabulary defining this space. It shouldn't contain any duplicate words.

## tokenize(text, padding=False)

Tokenize words found in the vocabulary.

Note: text will be padded up to self.max\_length.

12.3. Spaces 45

**CHAPTER** 

## **THIRTEEN**

## **TEXTWORLD.ENVS**

## 13.1 TextWorld

class textworld.envs.tw.TextWorldEnv(infos=None)

Bases: textworld.core.Environment

Environment for playing games by TextWorld.

**Parameters infos** (Optional[*EnvInfos*]) – Information to be included in the game state. By default, only the game's narrative is included.

copy()

Return a copy of this environment.

It is safe to call *step* and *reset* on the copied environment.

**Warning:** The Game and Inform7Game private objects are *soft* copies.

### Return type TextWorldEnv

### **load**(*path*)

Loads a new text-based game.

**Parameters** path (str) – Path to the game file to load.

Return type None

#### reset()

Starts game from the beginning.

**Returns** Initial state of the game.

### step(command)

Performs a given command.

**Parameters command** (str) – Text command to send to the interpreter.

**Returns** A tuple containing the new game state, a reward for performing that command and reaching this new state, and whether the game is finished or not.

## **13.2 Glulx**

class textworld.envs.glulx.git\_glulx.GitGlulxEnv(\*args, \*\*kwargs)

Bases: textworld.core.Environment

Environment to support playing Glulx games.

This environment supports playing text-based games that were compiled for the Glulx virtual machine. The main advantage of using Glulx over Z-Machine is it uses 32-bit data and addresses, so it can handle game files up to four gigabytes long. This comes handy when we want to generate large world with a lot of objects in it.

We use a customized version of git-glulx as the glulx interpreter. That way we don't rely on stdin/stdout to communicate with the interpreter but instead use UNIX sockets.

**Parameters infos** – Information to be included in the game state. By default, only the game's narrative is included.

close()

Ends the game.

Return type None

load(ulx\_file)

Loads a new text-based game.

**Parameters** path – Path to the game file to load.

Return type None

render(mode='human')

Renders the current state of the game.

**Parameters mode** (str) – The mode to use for rendering.

Return type None

reset()

Starts game from the beginning.

Return type str

Returns Initial state of the game.

step(command)

Performs a given command.

**Parameters command** (str) – Text command to send to the interpreter.

Return type str

**Returns** A tuple containing the new game state, a reward for performing that command and reaching this new state, and whether the game is finished or not.

property game\_running: bool

Determines if the game is still running.

Return type bool

# 13.3 Wrappers

```
class textworld.envs.wrappers.recorder.Recorder
     Bases: textworld.core.Wrapper
          Parameters env – environment to wrap.
     reset()
              Return type GameState
     step(command)
              Return type Tuple[GameState, float, bool]
class textworld.envs.wrappers.viewer.HtmlViewer(env, open automatically=True, port=8080)
     Bases: textworld.core.Wrapper
     Wrap a TextWorld environment to provide visualization.
     During a playthrough, the game can be visualized via local webserver http://localhost:<port>.
     :param : The TextWorld environment to wrap. :type : param env: :param : Port to use for the web viewer. :type
     : param port:
     close()
          Close the game.
          In addition to shutting down the game, this closes the local webserver.
     reset()
          Reset the game.
              Return type Initial game state.
     step(command)
          Perform a game step.
              Parameters command (str) – Text command to send to the game engine.
              Return type Tuple[GameState, float, bool]
              Returns
                   • game_state - Updated game state.
                   • score – Score for reaching this state.
                   • done – Whether the same is done or not.
     property port
class textworld.envs.wrappers.filter.Filter(env=None)
     Bases: textworld.core.Wrapper
     Environment wrapper to filter what information is made available.
     Requested information will be included within the infos dictionary returned by Filter.reset() and Filter.
     step(...). To request specific information, create a textworld. EnvInfos and set the appropriate attributes
```

13.3. Wrappers 49

to True. Then, instantiate a *Filter* wrapper with the EnvInfos object.

### **Example**

Here is an example of how to request information and retrieve it.

```
>>> from textworld import EnvInfos
     >>> from textworld.envs.wrappers import Filter
     >>> request_infos = EnvInfos(description=True, inventory=True, extras=["more"])
     >>> env = textworld.start(gamefile, request_infos)
     >>> env = Filter(env)
     >>> ob, infos = env.reset()
     >>> print(infos["description"])
     >>> print(infos["inventory"])
     >>> print(infos["extra.more"])
         Parameters env (Optional[Environment]) – environment to wrap.
     copy()
             Return type Filter
     reset()
             Return type Tuple[str, Mapping[str, Any]]
     step(command)
             Return type Tuple[str, Mapping[str, Any]]
13.4 Z-Machine
class textworld.envs.zmachine.jericho.JerichoEnv(*args, **kwargs)
     Bases: textworld.core.Environment
         Parameters infos - Information to be included in the game state. By default, only the game's
             narrative is included.
     close()
         Ends the game.
     copy()
         Return a copy of this environment at the same state.
             Return type JerichoEnv
     load(z, file)
         Loads a new text-based game.
             Parameters path – Path to the game file to load.
             Return type None
     reset()
```

Starts game from the beginning.

**Returns** Initial state of the game.

### seed(seed=None)

Sets the seed for the random number generator.

### step(command)

Performs a given command.

**Parameters command** – Text command to send to the interpreter.

**Returns** A tuple containing the new game state, a reward for performing that command and reaching this new state, and whether the game is finished or not.

### property game\_running: bool

Determines if the game is still running.

Return type bool

13.4. Z-Machine 51

## **FOURTEEN**

## **TEXTWORLD.AGENTS**

```
class textworld.agents.human.HumanAgent(autocompletion=True, oracle=False)
```

Bases: textworld.core.Agent

act(game\_state, reward, done)

Acts upon the current game state.

#### **Parameters**

- game\_state Current game state.
- reward Accumulated reward up until now.
- **done** Whether the game is finished.

**Returns** Text command to be performed in this current state.

#### reset(env)

Let the agent set some environment's flags.

**Parameters** env – TextWorld environment.

### class textworld.agents.random.NaiveAgent(seed=1234)

Bases: textworld.core.Agent

act(game\_state, reward, done)

Acts upon the current game state.

### **Parameters**

- **game\_state** Current game state.
- reward Accumulated reward up until now.
- **done** Whether the game is finished.

**Returns** Text command to be performed in this current state.

## reset(env)

Let the agent set some environment's flags.

**Parameters** env – TextWorld environment.

### class textworld.agents.random.RandomCommandAgent(seed=1234)

Bases: textworld.core.Agent

act(game\_state, reward, done)

Acts upon the current game state.

#### **Parameters**

• **game\_state** – Current game state.

- **reward** Accumulated reward up until now.
- **done** Whether the game is finished.

**Returns** Text command to be performed in this current state.

### reset(env)

Let the agent set some environment's flags.

**Parameters** env – TextWorld environment.

## class textworld.agents.simple.NaiveAgent(seed=1234)

Bases: textworld.core.Agent

act(game\_state, reward, done)

Acts upon the current game state.

#### **Parameters**

- **game\_state** Current game state.
- **reward** Accumulated reward up until now.
- **done** Whether the game is finished.

**Returns** Text command to be performed in this current state.

#### reset(env)

Let the agent set some environment's flags.

**Parameters** env – TextWorld environment.

### exception textworld.agents.walkthrough.WalkthroughDone

Bases: NameError

### class textworld.agents.walkthrough.WalkthroughAgent(commands=None)

Bases: textworld.core.Agent

Agent that simply follows a list of commands.

act(game\_state, reward, done)

Acts upon the current game state.

#### **Parameters**

- **game\_state** Current game state.
- reward Accumulated reward up until now.
- **done** Whether the game is finished.

**Returns** Text command to be performed in this current state.

### reset(env)

Let the agent set some environment's flags.

**Parameters** env – TextWorld environment.

## TEXTWORLD.GENERATOR

exception textworld.generator.GenerationWarning

Bases: UserWarning

 ${\tt textworld.generator.compile\_game} (\textit{game}, \textit{options} = \textit{None})$ 

Compile a game.

### **Parameters**

- game (Game) Game object to compile.
- **options** (Optional[GameOptions]) For customizing the game generation (see textworld.GameOptions for the list of available options).

**Returns** The path to compiled game.

textworld.generator.make\_game(options)

Make a game (map + objects + quest).

**Parameters options** (GameOptions) — For customizing the game generation (see textworld. GameOptions for the list of available options).

Return type Game

Returns Generated game.

 $\verb|textworld.generator.make_game_with(|world|, |quests=None|, |grammar=None|)|$ 

textworld.generator.make\_grammar(options={}, rng=None)

Return type Grammar

textworld.generator.make\_map(n\_rooms, size=None, rng=None, possible\_door\_states=['open', 'closed', 'locked'])

Make a map.

#### **Parameters**

- **n\_rooms** (*int*) Number of rooms in the map.
- **size** (tuple of int) Size (height, width) of the grid delimiting the map.

textworld.generator.make\_quest(world, options=None)

textworld.generator.make\_small\_map( $n\_rooms$ , rng=None,  $possible\_door\_states=['open', 'closed', 'locked']$ ) Make a small map.

The map will contains one room that connects to all others.

#### **Parameters**

- **n\_rooms** (*int*) Number of rooms in the map (maximum of 5 rooms).
- possible\_door\_states (list of str, optional) Possible states doors can have.

textworld.generator.make\_world(world\_size, nb\_objects=0, rngs=None)

Make a world (map + objects).

#### **Parameters**

- world\_size (int) Number of rooms in the world.
- **nb\_objects** (*int*) Number of objects in the world.

textworld.generator.make\_world\_with(rooms, rng=None)

Make a world that contains the given rooms.

**Parameters rooms** (list of textworld.logic.Variable) – Rooms in the map. Variables must have type 'r'.

### exception textworld.generator.chaining.QuestGenerationError

Bases: Exception

#### **class** textworld.generator.chaining.**Chain**(initial state, nodes)

Bases: object

An initial state and a chain of actions forming a quest.

#### nodes

The dependency tree of this quest.

#### initial\_state

The initial state from which the actions start.

#### actions

The sequence of actions forming this quest.

## class textworld.generator.chaining.ChainNode(action, depth, breadth, parent)

Bases: object

A node in a chain of actions.

### action

The action to perform at this step.

#### depth

This node's depth in the dependency tree.

#### breadth

This node's breadth in the dependency tree.

#### parent

This node's parent in the dependency tree.

## class textworld.generator.chaining.ChainingOptions

Bases: object

Options for customizing the behaviour of chaining.

#### backward

Whether to run chaining forwards or backwards. Forward chaining produces a sequence of actions that start at the provided state, while backward chaining produces a sequence of actions that end up at the provided state.

### min\_length

The minimum length of the generated quests.

### max\_length

The maximum length of the generated quests.

### min\_depth

The minimum depth (length) of the generated independent subquests.

### max\_depth

The maximum depth (length) of the generated independent subquests.

#### min\_breadth

The minimum breadth of the generated quests. When this is higher than 1, the generated quests will have multiple parallel subquests. In this case, min\_depth and max\_depth limit the length of these independent subquests, not the total size of the quest.

#### max\_breadth

The maximum breadth of the generated quests.

### subquests

Whether to also return incomplete quests, which could be extended without reaching the depth or breadth limits.

### independent\_chains

Whether to allow totally independent parallel chains.

#### create\_variables

Whether new variables may be created during chaining.

#### fixed mapping

A fixed mapping from placeholders to variables, for singletons.

#### rng

If provided, randomize the order of the quests using this random number generator.

## logic

The rules of the game.

### rules\_per\_depth

A list of lists of rules for restricting the allowed actions at certain depths.

### restricted\_types

A set of types that may not have new variables created.

### allowed\_types

A set of types that are allowed to have new variables created.

#### check\_action(state, action)

Check if an action should be allowed in this state.

The default implementation disallows actions that would create new facts that don't mention any new variables.

#### **Parameters**

- **state** (*State*) The current state.
- action (Action) The action being applied.

#### Return type bool

**Returns** Whether that action should be allowed.

### check\_new\_variable(state, type, count)

Check if a new variable should be allowed to be created in this state.

#### **Parameters**

```
    state (State) – The current state.
    type (str) – The type of variable being created.
    count (int) – The total number of variables of that type.
    Return type bool
    Returns Whether that variable should be allowed to be created.
    copy()
    Return type ChainingOptions
    get_rules(depth)
    Get the relevant rules for this depth.
    Parameters depth (int) – The current depth in the chain.
    Return type Iterable[Rule]
```

**Returns** The rules that may be applied at this depth in the chain. property fixed\_mapping: textworld.logic.GameLogic

Return type GameLogic

property logic: textworld.logic.GameLogic

Return type GameLogic

textworld.generator.chaining.get\_chains(state, options)

Generates chains of actions (quests) starting from or ending at the given state.

#### **Parameters**

- **state** (*State*) The initial state for chaining.
- options (ChainingOptions) Options to configure chaining behaviour.

**Return type** Iterable[*Chain*]

**Returns** All possible quests according to the constraints.

textworld.generator.chaining.sample\_quest(state, options)

Samples a single chain of actions (a quest) starting from or ending at the given state.

#### **Parameters**

- **state** (*State*) The initial state for chaining.
- **options** (*ChainingOptions*) Options to configure chaining behaviour. Set options.rng to sample a random quest.

Return type Optional[Chain]

**Returns** A single possible quest.

**Raises** *QuestGenerationError* – No quest could be generated given the provided chaining options.

class textworld.generator.dependency\_tree.DependencyTree(element\_type=<class</pre>

'textworld.generator.dependency\_tree.DependencyTreeElemetrees=[])

Bases: object

### copy()

#### **Return type** *DependencyTree*

```
push(value, allow_multi_root=False)
```

Add a value to this dependency tree.

Adding a value already present in the tree does not modify the tree.

#### **Parameters**

- value (Any) value to add.
- allow\_multi\_root (bool) if True, allow the value to spawn an additional root if needed.

### Return type bool

### remove(value)

Remove all leaves having the given value.

The value to remove needs to belong to at least one leaf in this tree. Otherwise, the tree remains unchanged.

```
Parameters value (Any) – value to remove from the tree.
```

Return type bool

**Returns** Whether the tree has changed or not.

```
property empty: bool
```

Return type bool

### property leaves\_elements:

List[textworld.generator.dependency\_tree.DependencyTreeElement]

```
Return type List[DependencyTreeElement]
```

property leaves\_values: List[Any]

Return type List[Any]

property values: List[Any]

Return type List[Any]

### class textworld.generator.dependency\_tree.DependencyTreeElement(value)

Bases: object

Representation of an element in the dependency tree.

The notion of dependency and ordering should be defined for these elements.

Subclasses should override  $depends\_on$ ,  $\_\_lt\_\_$  and  $\_\_str\_\_$  accordingly.

#### **depends\_on**(*other*)

Check whether this element depends on the other.

### Return type bool

### is\_distinct\_from(others)

Check whether this element is distinct from others.

```
Return type bool
class textworld.generator.logger.GameLogger(group_actions=True)
     Bases: object
     aggregate(other)
     collect(game)
     display_stats()
     static load(filename)
     save(filename)
     stats()
exception textworld.generator.vtypes.NotEnoughNounsError
     Bases: NameError
class textworld.generator.vtypes.VariableType(type, name, parent=None)
     Bases: object
     classmethod deserialize(data)
              Return type VariableType
     classmethod parse(expr)
          Parse a variable type expression.
              Parameters expr (str) - The string to parse, in the form name: type -> parent1 &
                 parent2 or name: type for root node.
              Return type VariableType
     serialize()
              Return type str
class textworld.generator.vtypes.VariableTypeTree(vtypes)
     Bases: object
     Manages hierarchy of types defined in ./grammars/variables.txt. Used for extending the rules.
     count(state)
          Counts how many objects there are of each type.
     descendants(vtype)
          Given a variable type, return all possible descendants.
     classmethod deserialize(data)
              Return type VariableTypeTree
     get_ancestors(vtype)
          List all ancestors of a type where the closest ancetors are first.
     get_description(vtype)
     is_constant(vtype)
     is_descendant_of(child, parents)
          Return if child is a descendant of parent
```

```
classmethod load(path)
          Read variables from text file.
     sample(parent_type, rng, exceptions=[], include_parent=True, probs=None)
          Sample an object type given the parent's type.
     serialize()
              Return type List
     CHEST = 'c'
     CLASS_HOLDER = ['c', 's']
     SUPPORTER = 's'
textworld.generator.vtypes.get_new(type, types_counts, max_types_counts=None)
     Get the next available id for a given type.
textworld.generator.vtypes.parse_variable_types(content)
     Parse a list VariableType expressions.
15.1 Game
exception textworld.generator.game.UnderspecifiedEventError
     Bases: NameError
exception textworld.generator.game.UnderspecifiedQuestError
     Bases: NameError
class textworld.generator.game.ActionDependencyTree(*args, kb=None, **kwargs)
     Bases: textworld.generator.dependency_tree.DependencyTree
     copy()
              Return type ActionDependencyTree
     flatten()
          Generates a flatten representation of this dependency tree.
          Actions are greedily yielded by iteratively popping leaves from the dependency tree.
              Return type Iterable[Action]
     remove(action)
          Remove all leaves having the given value.
          The value to remove needs to belong to at least one leaf in this tree. Otherwise, the tree remains unchanged.
              Parameters value – value to remove from the tree.
              Return type Tuple[bool, Optional[Action]]
              Returns Whether the tree has changed or not.
class textworld.generator.game.ActionDependencyTreeElement(value)
     Bases: textworld.generator.dependency_tree.DependencyTreeElement
     Representation of an Action in the dependency tree.
     The notion of dependency and ordering is defined as follows:
```

15.1. Game 61

- action 1 depends on action 2 if action 1 needs the propositions added by action 2;
- action1 should be performed before action2 if action2 removes propositions needed by action1.

## depends\_on(other)

Check whether this action depends on the other.

Action1 depends on action2 when the intersection between the propositions added by action2 and the preconditions of the action1 is not empty, i.e. action1 needs the propositions added by action2.

### Return type bool

### is\_distinct\_from(others)

Check whether this element is distinct from others.

We check if self.action has any additional information that others actions don't have. This helps us to identify whether a group of nodes in the dependency tree already contain all the needed information that self.action would bring.

### Return type bool

property action: textworld.logic.Action

**Return type** *Action* 

### class textworld.generator.game.EntityInfo(id, type)

Bases: object

Additional information about entities in the game.

#### classmethod deserialize(data)

Creates a EntityInfo from serialized data.

**Parameters data** (Mapping) — Serialized data with the needed information to build a *EntityInfo* object.

Return type EntityInfo

### serialize()

Serialize this object.

**Results:** EntityInfo's data serialized to be JSON compatible

### Return type Mapping

adj

The adjective (i.e. descriptive) part of the name, if available.

Type str

### definite

The definite article to use for this entity.

Type str

#### desc

Text description displayed when examining this entity in the game.

Type str

id

Unique name for this entity. It is used when generating

Type str

### indefinite

The indefinite article to use for this entity.

Type str

#### name

The name that will be displayed in-game to identify this entity.

Type str

#### noun

The noun part of the name, if available.

Type str

#### room\_type

Type of the room this entity belongs to. It used to influence its *name* during text generation.

Type str

### synonyms

Alternative names that can be used to refer to this entity.

**Type** List[str]

### type

The type of this entity.

Type str

### class textworld.generator.game.Event(actions=(), conditions=(), commands=())

Bases: object

Event happening in TextWorld.

An event gets triggered when its set of conditions become all statisfied.

#### actions

Actions to be performed to trigger this event

#### commands

Human readable version of the actions.

### condition

textworld.logic.Action that can only be applied when all conditions are statisfied.

### **Parameters**

- **actions** (Iterable[Action]) The actions to be performed to trigger this event. If an empty list, then conditions must be provided.
- **conditions** (Iterable[*Proposition*]) Set of propositions which need to be all true in order for this event to get triggered.
- **commands** (Iterable[str]) Human readable version of the actions.

## copy()

Copy this event.

Return type Event

## classmethod deserialize(data)

Creates an Event from serialized data.

15.1. Game 63

```
Parameters data (Mapping) – Serialized data with the needed information to build a Event
                   object.
               Return type Event
     is_triggering(state)
          Check if this event would be triggered in a given state.
               Return type bool
     serialize()
          Serialize this event.
          Results: Event's data serialized to be JSON compatible.
               Return type Mapping
     set_conditions(conditions)
          Set the triggering conditions for this event.
               Parameters conditions (Iterable[Proposition]) – Set of propositions which need to be
                   all true in order for this event to get triggered.
               Return type Action
               Returns Action that can only be applied when all conditions are statisfied.
     property actions: Iterable[textworld.logic.Action]
               Return type Iterable[Action]
     property commands: Iterable[str]
               Return type Iterable[str]
class textworld.generator.game.EventProgression(event, kb)
     Bases: object
     EventProgression monitors a particular event.
     Internally, the event is represented as a dependency tree of relevant actions to be performed.
          Parameters quest – The quest to keep track of its completion.
     compress_policy(state)
          Compress the policy given a game state.
               Parameters state (State) – Current game state.
               Return type bool
               Returns Whether the policy was compressed or not.
     copy()
          Return a soft copy.
               Return type EventProgression
     update(action=None, state=None)
          Update event progression given available information.
               Parameters
                   • action (Optional[Action]) – Action potentially affecting the event progression.
```

• **state** (Optional[State]) – Current game state.

### Return type None

### property done: bool

Check if the quest is done (i.e. triggered or untriggerable).

#### Return type bool

### property triggered: bool

Check whether the event has been triggered.

#### Return type bool

### property triggering\_policy: List[textworld.logic.Action]

Actions to be performed in order to trigger the event.

### **Return type** List[Action]

### property untriggerable: bool

Check whether the event is in an untriggerable state.

### Return type bool

### class textworld.generator.game.Game(world, grammar=None, quests=())

Bases: object

Game representation in TextWorld.

A *Game* is defined by a world and it can have quest(s) or not. Additionally, a grammar can be provided to control the text generation.

#### **Parameters**

- world (World) The world to use for the game.
- **quests** (Iterable[*Quest*]) The quests to be done in the game.
- grammar(Optional[Grammar]) The grammar to control the text generation.

### change\_grammar(grammar)

Changes the grammar used and regenerate all text.

#### Return type None

### copy()

Make a shallow copy of this game.

## Return type Game

### classmethod deserialize(data)

Creates a Game from serialized data.

**Parameters data** (Mapping) – Serialized data with the needed information to build a *Game* object.

Return type Game

#### classmethod load(filename)

Creates Game from serialized data saved in a file.

### Return type Game

### save(filename)

Saves the serialized data of this game to a file.

### Return type None

15.1. Game 65

```
serialize()
         Serialize this object.
          Results: Game's data serialized to be JSON compatible
              Return type Mapping
     property command_templates: List[str]
          All command templates understood in this game.
              Return type List[str]
     property directions_names: List[str]
              Return type List[str]
     property entity_names: List[str]
              Return type List[str]
     property infos: Dict[str, textworld.generator.game.EntityInfo]
          Information about the entities in the game.
              Return type Dict[str, EntityInfo]
     property max_score: float
         Sum of the reward of all quests.
              Return type float
     property objective: str
              Return type str
     property objects_names: List[str]
          The names of all relevant objects in this game.
              Return type List[str]
     property objects_names_and_types: List[str]
          The names of all non-player objects along with their type in this game.
              Return type List[str]
     property objects_types: List[str]
          All types of objects in this game.
              Return type List[str]
     property verbs: List[str]
          Verbs that should be recognized in this game.
              Return type List[str]
     property walkthrough: Optional[List[str]]
              Return type Optional[List[str]]
class textworld.generator.game.GameOptions
     Bases: object
```

Options for customizing the game generation.

#### nb\_rooms

Number of rooms in the game.

Type int

#### nb\_objects

Number of objects in the game.

Type int

### nb\_parallel\_quests

Number of parallel quests, i.e. not sharing a common goal.

Type int

### quest\_length

Number of actions that need to be performed to complete the game.

Type int

### quest\_breadth

Number of subquests per independent quest. It controls how nonlinear a quest can be (1: linear).

Type int

### quest\_depth

Number of actions that need to be performed to solve a subquest.

Type int

#### path

Path of the compiled game (.ulx or .z8). Also, the source (.ni) and metadata (.json) files will be saved along with it.

Type str

#### force\_recompile

If True, recompile game even if it already exists.

Type bool

### file\_ext

Type of the generated game file. Either .z8 (Z-Machine) or .ulx (Glulx). If *path* already has an extension, this is ignored.

Type str

#### seeds

Seeds for the different generation processes.

- If None, seeds will be sampled from textworld.g\_rng.
- If int, it acts as a seed for a random generator that will be used to sample the other seeds.
- If dict, the following keys can be set:
  - 'map': control the map generation;
  - 'objects': control the type of objects and their location;
  - 'quest': control the quest generation;
  - 'grammar': control the text generation.

For any key missing, a random number gets assigned (sampled from textworld.g\_rng).

**Type** Optional[Union[int, Dict]]

15.1. Game 67

```
kb
```

The knowledge base containing the logic and the text grammars (see textworld.generator. KnowledgeBase for more information).

```
Type KnowledgeBase
```

### chaining

For customizing the quest generation (see textworld.generator.ChainingOptions for the list of available options).

Type Chaining Options

#### grammar

For customizing the text generation (see textworld.generator.GrammarOptions for the list of available options).

**Type** *GrammarOptions* 

copy()

Return type GameOptions

property kb: textworld.generator.data.KnowledgeBase

Return type KnowledgeBase

property quest\_breadth: int

Return type int

property quest\_length: int

Return type int

property rngs: Dict[str, numpy.random.mtrand.RandomState]

Return type Dict[str, RandomState]

property seeds

property uuid: str

Return type str

class textworld.generator.game.GameProgression(game, track\_quests=True)

Bases: object

GameProgression keeps track of the progression of a game.

If *tracking\_quests* is True, then *winning\_policy* will be the list of Action that need to be applied in order to complete the game.

#### **Parameters**

- game (Game) The game for which to track progression.
- **track\_quests** (bool) whether quest progressions are being tracked.

### copy()

Return a soft copy.

## Return type GameProgression

#### update(action)

Update the state of the game given the provided action.

**Parameters action** (*Action*) – Action affecting the state of the game.

Return type None

## property completed: bool

Whether all non-optional quests are completed.

Return type bool

## property done: bool

Whether all non-optional quests are completed or at least one has failed or is unfinishable.

Return type bool

# property failed: bool

Whether at least one non-optional quest has failed or is unfinishable.

Return type bool

# property score: int

Sum of the reward of all completed quests.

Return type int

# property tracking\_quests: bool

Whether quests are being tracked or not.

Return type bool

# property valid\_actions: List[textworld.logic.Action]

Actions that are valid at the current state.

Return type List[Action]

# property winning\_policy: Optional[List[textworld.logic.Action]]

Actions to be performed in order to complete the game.

**Return type** Optional[List[Action]]

**Returns** A policy that leads to winning the game. It can be None if tracking\_quests is False or the quest has failed.

Bases: object

Quest representation in TextWorld.

A quest is defined by a mutually exclusive set of winning events and a mutually exclusive set of failing events.

#### win events

Mutually exclusive set of winning events. That is, only one such event needs to be triggered in order to complete this quest.

#### fail\_events

Mutually exclusive set of failing events. That is, only one such event needs to be triggered in order to fail this quest.

## reward

Reward given for completing this quest.

15.1. Game 69

#### desc

A text description of the quest.

#### commands

List of text commands leading to this quest completion.

## optional

Whether this quest is optional or not to finish the game.

## repeatable

Whether this quest can be completed more than once.

#### **Parameters**

- win\_events (Iterable[Event]) Mutually exclusive set of winning events. That is, only one such event needs to be triggered in order to complete this quest.
- **fail\_events** (Iterable[*Event*]) Mutually exclusive set of failing events. That is, only one such event needs to be triggered in order to fail this quest.
- **reward** (Optional[int]) Reward given for completing this quest. By default, reward is set to 1 if there is at least one winning events otherwise it is set to 0.
- **desc** (Optional[str]) A text description of the quest.
- **commands** (Iterable[str]) List of text commands leading to this quest completion.
- **optional** (bool) If True, this quest is optional to finish the game.
- **repeatable** (bool) If True, this quest can be completed more than once.

## copy()

Copy this quest.

Return type Quest

#### classmethod deserialize(data)

Creates a Quest from serialized data.

**Parameters data** (Mapping) – Serialized data with the needed information to build a *Quest* object.

Return type Quest

# is\_failing(state)

Check if this quest is failing in that particular state.

Return type bool

## is\_winning(state)

Check if this quest is winning in that particular state.

Return type bool

#### serialize()

Serialize this quest.

**Results:** Quest's data serialized to be JSON compatible

Return type Mapping

property commands: Iterable[str]

Return type Iterable[str]

```
property fail_events: Iterable[textworld.generator.game.Event]
              Return type Iterable[Event]
     property win_events: Iterable[textworld.generator.game.Event]
              Return type Iterable[Event]
class textworld.generator.game.QuestProgression(quest, kb)
     Bases: object
     QuestProgression keeps track of the completion of a quest.
     Internally, the quest is represented as a dependency tree of relevant actions to be performed.
          Parameters quest (Quest) – The quest to keep track of its completion.
     copy()
          Return a soft copy.
              Return type QuestProgression
     update(action=None, state=None)
          Update quest progression given available information.
              Parameters
                  • action (Optional[Action]) – Action potentially affecting the quest progression.
                  • state (Optional[State]) – Current game state.
              Return type None
     property completable: bool
          Check if the quest has winning events.
              Return type bool
     property completed: bool
          Check whether the quest is completed.
              Return type bool
     property done: bool
          Check if the quest is done (i.e. completed, failed or unfinishable).
              Return type bool
     property failed: bool
          Check whether the quest has failed.
              Return type bool
     property unfinishable: bool
          Check whether the quest is in an unfinishable state.
              Return type bool
     property winning_policy: Optional[List[textworld.logic.Action]]
          Actions to be performed in order to complete the quest.
              Return type Optional[List[Action]]
textworld.generator.game.gen_commands_from_actions(actions, kb=None)
```

15.1. Game 71

**Return type** List[str]

# **15.2 World**

```
exception textworld.generator.world.NoFreeExitError
     Bases: Exception
class textworld.generator.world.World(kb=None)
     Bases: object
     add_fact(fact)
             Return type None
     add_facts(facts)
             Return type None
     classmethod deserialize(serialized facts, kb=None)
             Return type World
     find_object_by_id(id)
             Return type Optional[WorldObject]
     find_room_by_id(id)
             Return type Optional[WorldRoom]
     classmethod from_facts(facts, kb=None)
             Return type World
     classmethod from_map(map, kb=None)
             Parameters map (Graph) – Graph defining the structure of the world.
             Return type World
     get_all_objects_in(obj)
             Return type List[WorldObject]
     get_entities_per_type(type)
          Get all entities of a certain type.
             Return type List[WorldEntity]
     get_facts_in_scope()
             Return type List[Proposition]
     get_objects_in_inventory()
```

```
Return type List[WorldObject]
get_visible_objects_in(obj)
        Return type List[WorldObject]
populate(nb objects, rng=None, object types probs=None)
        Return type List[Proposition]
populate_room(nb_objects, room, rng=None, object_types_probs=None)
        Return type List[Proposition]
populate_room_with(objects, room, rng=None)
        Return type List[Proposition]
populate_with(objects, rng=None)
        Return type List[Proposition]
serialize()
        Return type List
set_player_room(start_room=None)
        Return type Proposition
property entities: ValuesView[textworld.generator.world.WorldEntity]
        Return type ValuesView[WorldEntity]
property facts: List[textworld.logic.Proposition]
        Return type List[Proposition]
property objects: List[textworld.generator.world.WorldObject]
        Return type List[WorldObject]
property player_room: textworld.generator.world.WorldRoom
        Return type WorldRoom
property rooms: List[textworld.generator.world.WorldRoom]
        Return type List[WorldRoom]
property state: textworld.logic.State
        Return type State
```

15.2. World 73

```
class textworld.generator.world.WorldEntity(*args, **kwargs)
     Bases: textworld.logic.Variable
     A WorldEntity is an abstract concept representing anything with a name and a type.
     Create a Variable.
          Parameters
                • name – The (unique) name of the variable.
                • type (optional) – The type of the variable. Defaults to the same as the name.
     add_related_fact(fact)
              Return type None
     classmethod create(var)
              Return type Union[WorldRoom, WorldObject]
     get_attributes()
              Return type List[Proposition]
     property id: str
              Return type str
     name
     type
class textworld.generator.world.WorldObject(*args, **kwargs)
     Bases: textworld.generator.world.WorldEntity
     A WorldObject is anything we can directly interact with.
     Create a Variable.
          Parameters
                • name – The (unique) name of the variable.
                • type (optional) – The type of the variable. Defaults to the same as the name.
     name
     type
class textworld.generator.world.WorldRoom(*args, **kwargs)
     Bases: textworld.generator.world.WorldEntity
     WorldRooms can be linked with each other through exits.
     Create a Variable.
          Parameters
                • name – The (unique) name of the variable.
                • type (optional) – The type of the variable. Defaults to the same as the name.
     name
```

## type

textworld.generator.world.connect(room1, direction, room2, door=None)
Generate predicates that connect two rooms.

#### **Parameters**

- room1 (Variable) A room variable.
- **direction** (str) Direction that we need to travel to go from room1 to room2.
- room2 (Variable) A room variable.
- **door** (Optional[*Variable*]) The door separating the two rooms. If None, there is no door between the rooms.

Return type List[Proposition]

 ${\tt textworld.generator.world.graph2state}(\textit{G}, \textit{rooms})$ 

Convert Graph object to a list of Proposition.

#### **Parameters**

- **G** (Graph) Graph defining the structure of the world.
- **rooms** (Dict[str, Variable]) information about the rooms in the world.

**Return type** List[*Proposition*]

```
\label{local_condition} \verb|textworld.generator.graph_networks.create_map| (rng, n\_nodes, h, w, possible\_door\_states = ['open', 'locked'])| \\
```

textworld.generator.graph\_networks.create\_small\_map(rng, n\_rooms, possible\_door\_states=['open', 'closed', 'locked'])

textworld.generator.graph\_networks.direction(x, y)

 $textworld.generator.graph_networks.extremes(G)$ 

Find left most and bottom nodes in the cartesian sense.

 $\texttt{textworld.generator.graph\_networks.gen\_layout}(rng, n\_nodes = 5, h = 10, w = 10)$ 

Generate a map with n\_nodes rooms by picking a subgraph from a h,w grid.

textworld.generator.graph\_networks.get\_path(G, room1, room2)

textworld.generator.graph\_networks.mark\_doors(*G*, rng, possible\_door\_states=['open', 'closed', 'locked'])

Put doors between neighbouring articulation points.

 ${\tt textworld.generator.graph\_networks.plot\_graph}(\textit{G}, \textit{show=True})$ 

Plot TextWorld's graph representation of a world.

## Return type None

```
textworld.generator.graph_networks.relabel(G)
```

Relabel G so that its origin is (0, 0)

textworld.generator.graph\_networks.reverse\_direction(direction)

textworld.generator.graph\_networks.shortest\_path(G, source, target)

Return shortest path in terms of directions.

textworld.generator.graph\_networks.xy\_diff(x, y)

15.2. World 75

# 15.3 GameMaker

```
exception textworld.generator.maker.ExitAlreadyUsedError
     Bases: ValueError
exception textworld.generator.maker.FailedConstraintsError(failed_constraints)
     Bases: ValueError
     Thrown when a constraint has failed during generation.
          Parameters failed_constraints (List[Action]) – The constraints that have failed
exception textworld.generator.maker.MissingPlayerError
     Bases: ValueError
exception textworld.generator.maker.PlayerAlreadySetError
     Bases: ValueError
exception textworld.generator.maker.QuestError
     Bases: ValueError
class textworld.generator.maker.GameMaker(options=None)
     Bases: object
     Stateful utility class for handcrafting text-based games.
     player
          Entity representing the player.
              Type WorldEntity
     inventory
          Entity representing the player's inventory.
              Type WorldEntity
     nowhere
          List of out-of-world entities (e.g. objects that would only appear later in a game).
              Type List[WorldEntity]
     rooms
          The rooms present in this world.
              Type List[WorldRoom]
     paths
          The connections between the rooms.
              Type List[WorldPath]
     Creates an empty world, with a player and an empty inventory.
     add_fact(name, *entities)
          Adds a fact.
              Parameters
                  • name (str) – The name of the new fact.
                  • *entities – A list of WorldEntity as arguments to this fact.
              Return type None
     build(validate=True)
```

Create a Game instance given the defined facts.

**Parameters validate** (optional) – If True, check if the game is valid, i.e. respects all constraints.

Return type Generated game.

## compile(path)

Compile this game.

**Parameters path** (str) – Path where to save the generated game.

**Returns** Path to the game file.

**Return type** game\_file

## connect(exit1, exit2)

Connect two rooms using their exits.

#### **Parameters**

- exit1 (WorldRoomExit) The exit of the first room to link.
- exit2 (WorldRoomExit) The exit of the second room to link.

Return type WorldPath

**Returns** The path created by the link between two rooms, with no door.

## find\_by\_name(name)

Find an entity using its name.

Return type Optional[WorldEntity]

## find\_path(room1, room2)

Get the path between two rooms, if it exists.

# **Parameters**

- **room1** (*WorldRoom*) One of the two rooms.
- **room2** (*WorldRoom*) The other room.

**Return type** Optional[WorldEntity]

**Returns** The matching path path, if it exists.

# findall(type)

Gets all entities of the given type.

**Parameters type** (str) – The type of entity to find.

**Return type** List[WorldEntity]

**Returns** All entities which match.

## generate\_distractors(nb\_distractors)

Generates a number of distractors - random objects.

**Parameters nb\_distractors** (int) – The number of distractors to game will contain.

Return type None

#### generate\_random\_quests(nb\_quests=1, length=1, breadth=1)

Generates random quests for the game.

Warning: This method overrides any previous quests the game had.

15.3. GameMaker 77

#### **Parameters**

- **nb\_quests** Number of parallel quests, i.e. not sharing a common goal.
- length (int) Number of actions that need to be performed to complete the game.
- **breadth** (int) Number of subquests per independent quest. It controls how nonlinear a quest can be (1: linear).

## Return type List[Quest]

Returns The generated quests.

## import\_graph(G)

Convert Graph object to a list of Proposition.

**Parameters G** (Graph) – Graph defining the structure of the world.

Return type List[WorldRoom]

move(entity, new\_location)

Move an entity to a new location.

### **Parameters**

- **entity** (*WorldEntity*) Entity to move.
- **new\_location** (*WorldEntity*) Where to move the entity.

## Return type None

**new**(*type*, *name=None*, *desc=None*)

Creates new entity given its type.

# **Parameters**

- **type** (str) The type of the entity.
- **name** (Optional[str]) The name of the entity.
- **desc** (Optional[str]) The description of the entity.

Return type Union[WorldEntity, WorldRoom]

# Returns

The newly created entity.

- If the type is 'r', then a WorldRoom object is returned.
- Otherwise, a WorldEntity is returned.

new\_door(path, name=None, desc=None)

Creates a new door and add it to the path.

## **Parameters**

- path (WorldPath) A path between two rooms where to add the door.
- name (Optional[str]) The name of the door. Default: generate one automatically.
- **desc** (Optional[str]) The description of the door.

Return type WorldEntity

**Returns** The newly created door.

#### new\_event\_using\_commands(commands)

Creates a new event using predefined text commands.

This launches a textworld.play session to execute provided commands.

**Parameters commands** (List[str]) – Text commands.

Return type Event

Returns The resulting event.

# new\_fact(name, \*entities)

Create new fact.

## **Parameters**

- name (str) The name of the new fact.
- \*entities A list of entities as arguments to the new fact.

Return type None

## new\_quest\_using\_commands(commands)

Creates a new quest using predefined text commands.

This launches a textworld.play session to execute provided commands.

Parameters commands (List[str]) - Text commands.

Return type Quest

**Returns** The resulting quest.

## new\_room(name=None, desc=None)

Create new room entity.

# **Parameters**

- name (Optional[str]) The name of the room.
- **desc** (Optional[str]) The description of the room.

Return type WorldRoom

**Returns** The newly created room entity.

## record\_quest()

Defines the game's quest by recording the commands.

This launches a textworld.play session.

Return type Quest

**Returns** The resulting quest.

# render(interactive=False)

Returns a visual representation of the world. :type interactive: bool :param interactive: opens an interactive session in the browser instead of returning a png. :return: :param save\_screenshot: ONLY FOR WHEN interactive == False. Save screenshot in temp directory. :param filename: filename for screenshot

#### set\_player(room)

Place the player in room.

**Parameters room** (WorldRoom) – The room the player will start in.

15.3. GameMaker 79

#### **Notes**

At the moment, the player can only be place once and cannot be moved once placed.

Raises PlayerAlreadySetError – If the player has already been set.

Return type None

## set\_quest\_from\_commands(commands)

Defines the game's quest using predefined text commands.

This launches a textworld.play session.

**Parameters commands** (List[str]) – Text commands.

Return type Quest

**Returns** The resulting quest.

#### set\_walkthrough(commands)

## test(walkthrough=False)

Test the game being built.

This launches a textworld.play session.

## Return type None

## validate()

Check if the world is valid and can be compiled.

A world is valid is the player has been place in a room and all constraints (defined in the *knowledge base*) are respected.

#### Return type bool

## property facts: Iterable[textworld.logic.Proposition]

All the facts associated to the current game state.

Return type Iterable[Proposition]

# property state: textworld.logic.State

Current state of the world.

Return type State

## **class** textworld.generator.maker.WorldEntity(var, name=None, desc=None, kb=None)

Bases: object

Represents an entity in the world.

Example of entities commonly found in text-based games: rooms, doors, items, etc.

#### **Parameters**

- **var** (*Variable*) The underlying variable for the entity which is used by TextWorld's inference engine.
- name (Optional[str]) The name of the entity that will be displayed in-game. Default: generate one according the variable's type.
- **desc** (Optional[str]) The description of the entity that will be displayed when examining it in the game.

# add(\*entities)

Add children to this entity.

## Return type None

```
add_fact(name, *entities)
```

Adds a fact to this entity.

#### **Parameters**

- name (str) The name of the new fact.
- \*entities A list of entities as arguments to the new fact.

## Return type None

```
add_property(name)
```

Adds a property to this entity.

A property is a fact that only involves one entity. For instance, 'closed(c)', 'open(c)', and 'locked(c)' are all properties.

Parameters name (str) – The name of the new property.

Return type None

# has\_property(name)

Determines if this object has a property with the given name.

Parameters property. (The name of the) -

## **Example**

```
>>> from textworld import GameMaker
>>> M = GameMaker()
>>> chest = M.new(type="c", name="chest")
>>> chest.has_property('closed')
False
>>> chest.add_property('closed')
>>> chest.has_property('closed')
True
```

## Return type bool

```
remove(*entities)

remove_fact(name, *entities)

Return type None

remove_property(name)

Return type None

property facts: List[textworld.logic.Proposition]

All facts related to this entity (or its children content).

Return type List[Proposition]

property id: str

Unique name used internally.

Return type str
```

15.3. GameMaker 81

```
property name: str
          Name of this entity.
              Return type str
     property properties: List[textworld.logic.Proposition]
          Properties of this object are things that refer to this object and this object alone. For instance, 'closed',
          'open', and 'locked' are possible properties of 'containers'.
               Return type List[Proposition]
     property type: str
          Type of this entity.
              Return type str
class textworld.generator.maker.WorldPath(src, src_exit, dest, dest_exit, door=None, kb=None)
     Bases: object
     Represents a path between two WorldRoom objects.
     A WorldPath encapsulates the source WorldRoom, the source WorldRoomExit, the destination WorldRoom
     and the destination WorldRoom. Optionally, a linking door can also be provided.
          Parameters
                • src (WorldRoom) – The source room.
                • src_exit (WorldRoomExit) – The exit of the source room.
                • dest (WorldRoom) – The destination room.
                • dest_exit (WorldRoomExit) – The exist of the destination room.
                • door (Optional[WorldEntity]) – The door between the two rooms, if any.
     property door: Optional[textworld.generator.maker.WorldEntity]
          The entity representing the door or None if there is none.
               Return type Optional[WorldEntity]
     property facts: List[textworld.logic.Proposition]
          Facts related to this path.
              Return type List[Proposition]
              Returns The facts that make up this path.
class textworld.generator.maker.WorldRoom(*args, **kwargs)
     Bases: textworld.generator.maker.WorldEntity
     Represents a room in the world.
     Takes the same arguments as WorldEntity.
     Then, creates a WorldRoomExit for each direction defined in graph_networks.DIRECTIONS, and sets exits to
     be a dict of those names to the newly created rooms. It then sets an attribute to each name.
          Parameters
                • args – The args to pass to WorldEntity
                • kwargs – The kwargs to pass to WorldEntity
     east
     north
```

#### south

west

class textworld.generator.maker.WorldRoomExit(src, direction, dest=None)

Bases: object

Represents an exit from a Room.

These are used to connect WorldRoom`s to form `WorldPath`s. `WorldRoomExit`s are linked to each other through their :py:attr:`dest.

When dest is None, it means there is no path leading to this exit yet.

#### **Parameters**

- **src** (*WorldRoom*) The WorldRoom that the exit is from.
- direction (str) The direction the exit is in: north, east, south, and west are common.
- **dest** (Optional[WorldRoom]) The WorldRoomExit that this exit links to (exits are linked to each other).

textworld.generator.maker.get\_failing\_constraints(state, kb=None)

# 15.4 Grammar

class textworld.generator.text\_generation.CountOrderedDict

Bases: collections.OrderedDict

An OrderedDict whose empty items are 0

class textworld.generator.text\_generation.MergeAction

Bases: object

Group of actions merged into one.

This allows for blending consecutive instructions.

textworld.generator.text\_generation.assign\_description\_to\_object(obj, grammar, game)
Assign a descripton to an object.

textworld.generator.text\_generation.assign\_description\_to\_quest(quest, game, grammar)

textworld.generator.text\_generation.assign\_description\_to\_room(room, game, grammar) Assign a descripton to a room.

textworld.generator.text\_generation.assign\_name\_to\_object(obj, grammar, game\_infos)
Assign a name to an object (if needed).

textworld.generator.text\_generation.assign\_new\_matching\_names(obj1\_infos, obj2\_infos, grammar, exclude)

textworld.generator.text\_generation.clean\_replace\_objs(grammar, desc, objs, game)
Return a cleaned/keyword replaced for a list of objects.

textworld.generator.text\_generation.describe\_event(event, game, grammar)
Assign a descripton to a quest.

#### Return type str

textworld.generator.text\_generation.expand\_clean\_replace(symbol, grammar, obj, game)
Return a cleaned/keyword replaced symbol.

15.4. Grammar 83

```
textworld.generator.text_generation.generate_instruction(action, grammar, game, counts)
     Generate text instruction for a specific action.
textworld.generator.text_generation.generate_text_from_grammar(game, grammar)
textworld.generator.text_generation.get_action_chains(actions, grammar, game)
     Reduce the action list by combining similar actions.
textworld.generator.text_generation.is_seq(chain, game)
     Check if we have a theoretical chain in actions.
textworld.generator.text_generation.list_to_string(lst, det, det_type='a')
     Convert a list to a natural language string.
textworld.generator.text_generation.obj_list_to_prop_string(objs, property, game, det=True,
                                                                    det_type='a'
     Convert an object list to a nl string list of names.
textworld.generator.text_generation.repl_sing_plur(phrase, length)
     Alter a sentence depending on whether or not we are dealing with plural or singular objects (for counting)
textworld.generator.text_generation.replace_num(phrase, val)
     Add a numerical value to a string.
exception textworld.generator.text_grammar.MissingTextGrammar(path)
     Bases: NameError
class textworld.generator.text_grammar(Grammar(options={}), rng=None, kb=None)
```

Context-Free Grammar for text generation.

## **Parameters**

Bases: object

- **options** (Union[GrammarOptions, Mapping[str, Any]]) For customizing text generation process (see textworld.generator.GrammarOptions for the list of available options).
- rng (Optional[RandomState]) Random generator used for sampling tag expansions.

## check()

Check if this grammar is valid.

TODO: use logging mechanism to report warnings and errors.

# Return type bool

```
expand(text, rng=None)
```

Expand some text until there is no more tag to expand.

#### **Parameters**

- **text** (str) Text potentially containing grammar tags to be expanded.
- **rng** (*optional*) Random generator used to chose an expansion when there is many. By default, it used the random generator of this grammar object.

**Returns** Resulting text in which there is no grammar tag left to be expanded.

**Return type** expanded\_text

```
generate_name(obj_type, room_type=", include_adj=None, exclude=[]) Generate a name given an object type and the type room it belongs to.
```

#### **Parameters**

- **obj\_type** (str) Type of the object for which we will generate a name.
- room\_type (optional) Type of the room the object belongs to.
- **include\_adj** (*optional*) If True, the name can contain a generated adjective. If False, any generated adjective will be discarded. Default: use value grammar.options.include\_adj
- **exclude** (*optional*) List of names we should avoid generating.

Return type Tuple[str, str, str]

#### Returns

- name The whole name, i.e. adj + " " + noun.
- *adj* The adjective part of the name.
- noun The noun part of the name.

# get\_all\_adjective\_for\_type(type)

Get all possible adjectives for a given object type.

**Parameters type** (str) – Object type.

**Returns** All possible adjectives sorted in alphabetical order.

Return type adjectives

# get\_all\_expansions\_for\_tag(tag, max\_depth=500)

Get all possible expansions for a grammar tag.

#### **Parameters**

- tag (str) Grammar tag to be expanded.
- max\_depth (optional) Maximum recursion depth when expanding tag.

**Returns** All possible expansions.

Return type expansions

## get\_all\_expansions\_for\_type(type)

Get all possible expansions for a given object type.

**Parameters** type (str) – Object type.

**Returns** All possible names.

Return type names

# get\_all\_names\_for\_type(type, include\_adj)

Get all possible names for a given object type.

## **Parameters**

- **type** (str) Object type.
- **include\_adj** (*optional*) If True, names can contain generated adjectives. If False, any generated adjectives will be discarded.

**Returns** All possible names sorted in alphabetical order.

Return type names

# get\_all\_nouns\_for\_type(type)

Get all possible nouns for a given object type.

Parameters type (str) – Object type.

15.4. Grammar 85

Returns All possible nouns sorted in alphabetical order.

Return type nouns

# get\_random\_expansion(tag, rng=None)

Return a randomly chosen expansion for the given tag.

#### **Parameters**

- tag (str) Grammar tag to be expanded.
- **rng** (*optional*) Random generator used to chose an expansion when there is many. By default, it used the random generator of this grammar object.

**Returns** An expansion chosen randomly for the provided tag.

Return type expansion

get\_vocabulary()

Return type List[str]

has\_tag(tag)

Check if the grammar has a given tag.

Return type bool

split\_name\_adj\_noun(candidate, include\_adj)

Extract the full name, the adjective and the noun from a string.

#### **Parameters**

- candidate (str) String that may contain one adjective-noun sperator '|'.
- **include\_adj** (*optional*) If True, the name can contain a generated adjective. If False, any generated adjective will be discarded.

Return type Optional[Tuple[str, str, str]]

#### **Returns**

- name The whole name, i.e. adj + " " + noun.
- *adj* The adjective part of the name.
- *noun* The noun part of the name.

 $\textbf{class} \ \texttt{textworld.generator.text\_grammar.GrammarOptions} (\textit{options} = None, **kwargs)$ 

Bases: object

copy()

Return type GrammarOptions

## classmethod deserialize(data)

Creates a *GrammarOptions* from serialized data.

**Parameters data** (Mapping) – Serialized data with the needed information to build a *GrammarOptions* object.

Return type GrammarOptions

# serialize()

Serialize this object.

**Results:** GrammarOptions's data serialized to be JSON compatible.

# Return type Mapping

#### allowed\_variables\_numbering

Append numbers after an object name if there is not enough variation for it.

Type bool

#### ambiguous\_instructions

When True, in the game objective, objects of interest might be refer to by their type or adjective rather than full name.

Type bool

## blend\_descriptions

When True, objects sharing some properties might be described in a single sentence rather than separate consecutive ones.

Type bool

# blend\_instructions

When True, consecutive actions to be accomplished might be described in a single sentence rather than separate ones.

Type bool

## include\_adj

When True, object names can be preceded by an adjective.

Type bool

#### names\_to\_exclude

List of names the text generation should not use.

Type List[str]

## only\_last\_action

When True, only the last action of a quest will be described in the generated objective.

Type bool

# theme

Grammar theme's name. All \*.twg files starting with that name will be loaded.

Type str

# unique\_expansion

When True, #symbol# are force to be expanded to unique text.

Type bool

## property uuid: str

Generate UUID for this set of grammar options.

Return type str

textworld.generator.text\_grammar.fix\_determinant(var)

15.4. Grammar 87

# 15.5 Knowledge Base

#### **Parameters**

- target\_dir (Optional[str]) Folder containing two subfolders: logic and text\_grammars. If provided, both logic\_path and grammar\_path are ignored.
- **logic\_path** (Optional[str]) Folder containing \*.twl files that describe the logic of a game.
- **grammar\_path** (Optional[str]) Folder containing \*.twg files that describe the grammar used for text generation.

Return type KnowledgeBase

Returns KnowledgeBase object.

serialize()

Return type str

textworld.generator.data.create\_data\_files(dest='./textworld\_data', verbose=False, force=False)

Creates grammar files in the target directory.

Will NOT overwrite files if they alredy exist (checked on per-file basis).

# **Parameters**

- **dest** (str) The path to the directory where to dump the data files into.
- **verbose** (bool) Print when skipping an existing file.
- **force** (bool) Overwrite all existing files.

# 15.5.1 Data

#### container.twl

```
# container
type c : t {
    predicates {
        open(c);
        closed(c);
        locked(c);
```

```
in(o, c);
   }
   rules {
       lock/c :: sat(P, r) & sat(c, r) & sin(k, I) & smatch(k, c) & closed(c) ->_
→locked(c);
       unlock/c :: at(P, r) & at(c, r) & in(k, I) & match(k, c) & locked(c) ->_ unlock/c :: 
→closed(c);
       open/c :: $at(P, r) & $at(c, r) & closed(c) -> open(c);
       close/c :: $at(P, r) & $at(c, r) & open(c) -> closed(c);
   }
   reverse_rules {
       lock/c :: unlock/c;
       open/c :: close/c;
   }
   constraints {
       c1 :: open(c)
                       & closed(c) -> fail();
       c2 :: open(c) & locked(c) -> fail();
       c3 :: closed(c) & locked(c) -> fail();
   }
   inform7 {
       type {
           kind :: "container";
           definition :: "containers are openable, lockable and fixed in place.
}
       predicates {
           open(c) :: "The {c} is open";
           closed(c) :: "The {c} is closed";
           locked(c) :: "The {c} is locked";
           in(o, c) :: "The {o} is in the {c}";
       }
       commands {
           open/c :: "open {c}" :: "opening the {c}";
           close/c :: "close {c}" :: "closing the {c}";
           lock/c :: "lock {c} with {k}" :: "locking the {c} with the {k}";
           unlock/c :: "unlock {c} with {k}" :: "unlocking the {c} with the {k}";
       }
   }
}
```

#### door.twl

```
# door
type d : t {
   predicates {
        open(d);
        closed(d);
        locked(d);
        link(r, d, r);
   }
   rules {
        lock/d :: at(P, r) & link(r, d, r') & link(r', d, r) & link(r', d, r) & link(r, l) & match(k, l)
\rightarrow d) & closed(d) \rightarrow locked(d);
        unlock/d :: $at(P, r) & $link(r, d, r') & $link(r', d, r) & $in(k, I) & $match(k,
\rightarrow d) & locked(d) -> closed(d);
        open/d :: at(P, r) & link(r, d, r') & link(r', d, r) & closed(d) -> open(d)
\rightarrow& free(r, r') & free(r', r);
        close/d :: $at(P, r) & $link(r, d, r') & $link(r', d, r) & open(d) & free(r, r
\rightarrow') & free(r', r) -> closed(d);
        examine/d :: at(P, r) & link(r, d, r') \rightarrow at(P, r); # Nothing changes.
   }
   reverse_rules {
        lock/d :: unlock/d;
        open/d :: close/d;
        examine/d :: examine/d;
   }
   constraints {
        d1 :: open(d)
                        & closed(d) -> fail();
        d3 :: closed(d) & locked(d) -> fail();
        # A door can't be used to link more than two rooms.
        link1 :: link(r, d, r') & link(r, d, r'') -> fail();
        link2 :: link(r, d, r') & link(r'', d, r''') -> fail();
        # There's already a door linking two rooms.
        link3 :: link(r, d, r') & link(r, d', r') -> fail();
        # There cannot be more than four doors in a room.
        too_many_doors :: link(r, d1: d, r1: r) & link(r, d2: d, r2: r) & link(r, d3: d,__
\rightarrowr3: r) & link(r, d4: d, r4: r) & link(r, d5: d, r5: r) -> fail();
        # There cannot be more than four doors in a room.
        dr1 :: free(r, r1: r) & link(r, d2: d, r2: r) & link(r, d3: d, r3: r) & link(r, u)
\rightarrow d4: d, r4: r) & link(r, d5: d, r5: r) -> fail();
        dr2 :: free(r, r1: r) & free(r, r2: r) & link(r, d3: d, r3: r) & link(r, d4: d, ...
\rightarrowr4: r) & link(r, d5: d, r5: r) -> fail();
                                                                              (continues on next page)
```

```
dr3 :: free(r, r1: r) & free(r, r2: r) & free(r, r3: r) & link(r, d4: d, r4: r) &
\rightarrow link(r, d5: d, r5: r) -> fail();
        dr4 :: free(r, r1: r) & free(r, r2: r) & free(r, r3: r) & free(r, r4: r) & ...
\rightarrowlink(r, d5: d, r5: r) -> fail();
        free1 :: link(r, d, r') & free(r, r') & closed(d) -> fail();
        free2 :: link(r, d, r') & free(r, r') & locked(d) -> fail();
    }
    inform7 {
        type {
            kind :: "door";
            definition :: "door is openable and lockable.";
        predicates {
            open(d) :: "The {d} is open";
            closed(d) :: "The {d} is closed";
            locked(d) :: "The {d} is locked";
            link(r, d, r') :: ""; # No equivalent in Inform7.
        }
        commands {
            open/d :: "open {d}" :: "opening {d}";
            close/d :: "close {d}" :: "closing {d}";
            unlock/d :: "unlock {d} with {k}" :: "unlocking {d} with the {k}";
            lock/d :: "lock {d} with {k}" :: "locking {d} with the {k}";
            examine/d :: "examine {d}" :: "examining {d}";
        }
    }
}
```

## food.twl

```
# food
type f : 0 {
    predicates {
        edible(f);
        eaten(f);
    }

    rules {
        eat :: in(f, I) -> eaten(f);
    }

    constraints {
        eaten1 :: eaten(f) & in(f, I) -> fail();
        eaten2 :: eaten(f) & in(f, c) -> fail();
}
```

```
eaten3 :: eaten(f) & on(f, s) -> fail();
        eaten4 :: eaten(f) & at(f, r) \rightarrow fail();
    }
    inform7 {
        type {
            kind :: "food";
            definition :: "food is edible.";
        }
        predicates {
            edible(f) :: "The {f} is edible";
            eaten(f) :: "The {f} is nowhere";
        commands {
            eat :: "eat \{f\}" :: "eating the \{f\}";
        }
    }
}
```

# inventory.twl

```
# Inventory
type I {
   predicates {
        in(o, I);
   }
   rules {
        inventory :: at(P, r) \rightarrow at(P, r); # Nothing changes.
       take :: at(P, r) & at(o, r) -> in(o, I);
        drop :: at(P, r) & in(o, I) -> at(o, r);
        take/c :: at(P, r) & at(c, r) & pen(c) & in(o, c) -> in(o, I);
        insert :: $at(P, r) & $at(c, r) & $open(c) & in(o, I) -> in(o, c);
        take/s :: $at(P, r) & $at(s, r) & on(o, s) -> in(o, I);
            :: $at(P, r) & $at(s, r) & in(o, I) -> on(o, s);
        examine/I :: in(o, I) \rightarrow in(o, I); # Nothing changes.
        examine/s :: at(P, r) & at(s, r) & n(o, s) \rightarrow at(P, r); # Nothing changes.
        examine/c :: at(P, r) & $at(c, r) & $open(c) & $in(o, c) -> at(P, r); # Nothing_
}
   reverse_rules {
        inventory :: inventory;
```

```
take :: drop;
        take/c :: insert;
        take/s :: put;
        examine/I :: examine/I;
        examine/s :: examine/s;
        examine/c :: examine/c;
   }
   inform7 {
       predicates {
            in(o, I) :: "The player carries the {o}";
        }
       commands {
            take :: "take {o}" :: "taking the {o}";
            drop :: "drop {o}" :: "dropping the {o}";
            take/c :: "take {o} from {c}" :: "removing the {o} from the {c}";
            insert :: "insert {o} into {c}" :: "inserting the {o} into the {c}";
            take/s :: "take {o} from {s}" :: "removing the {o} from the {s}";
            put :: "put {o} on {s}" :: "putting the {o} on the {s}";
            inventory :: "inventory" :: "taking inventory";
            examine/I :: "examine {o}" :: "examining the {o}";
            examine/s :: "examine {o}" :: "examining the {o}";
            examine/c :: "examine {o}" :: "examining the {o}";
        }
   }
}
```

## key.twl

```
# key
type k : o {
    predicates {
        match(k, c);
        match(k, d);
    }

    constraints {
        k1 :: match(k, c) & match(k', c) -> fail();
        k2 :: match(k, c) & match(k, c') -> fail();
        k3 :: match(k, d) & match(k', d) -> fail();
        k4 :: match(k, d) & match(k, d') -> fail();
    }

    inform7 {
```

```
type {
        kind :: "key";
}

predicates {
        match(k, c) :: "The matching key of the {c} is the {k}";
        match(k, d) :: "The matching key of the {d} is the {k}";
}
}
```

# object.twl

```
# object
type o : t {
    constraints {
        obj1 :: in(o, I) & in(o, c) -> fail();
        obj2 :: in(o, I) & on(o, s) -> fail();
        obj3 :: in(o, I) & at(o, r) -> fail();
        obj4 :: in(o, c) & on(o, s) -> fail();
        obj5 :: in(o, c) & at(o, r) -> fail();
        obj6 :: on(o, s) & at(o, r) \rightarrow fail();
        obj7 :: at(o, r) & at(o, r') -> fail();
        obj8 :: in(o, c) & in(o, c') -> fail();
        obj9 :: on(o, s) & on(o, s') -> fail();
    }
    inform7 {
        type {
            kind :: "object-like";
            definition :: "object-like is portable.";
        }
    }
}
```

## player.twl

```
# Player
type P {
    rules {
        look :: at(P, r) -> at(P, r); # Nothing changes.
    }

    reverse_rules {
        look :: look;
    }

    inform7 {
        commands {
```

```
look :: "look" :: "looking";
}
}
```

#### room.twl

```
# room
type r {
   predicates {
        at(P, r);
        at(t, r);
       north_of(r, r);
       west_of(r, r);
       north_of/d(r, d, r);
       west_of/d(r, d, r);
       free(r, r);
        south_of(r, r') = north_of(r', r);
        east_of(r, r') = west_of(r', r);
        south_of/d(r, d, r') = north_of/d(r', d, r);
        east_of/d(r, d, r') = west_of/d(r', d, r);
   }
   rules {
        go/north :: at(P, r) & $north_of(r', r) & $free(r, r') & $free(r', r) -> at(P, r
→');
       go/south :: at(P, r) & $south_of(r', r) & $free(r, r') & $free(r', r) -> at(P, r
');
       go/east :: at(P, r) & $east_of(r', r) & $free(r, r') & $free(r', r) -> at(P, r
');
        go/west :: at(P, r) & $west_of(r', r) & $free(r, r') & $free(r', r) -> at(P, r
');
   }
   reverse_rules {
        go/north :: go/south;
        go/west :: go/east;
   }
   constraints {
       r1 :: at(P, r) & at(P, r') -> fail();
       r2 :: at(s, r) & at(s, r') \rightarrow fail();
       r3 :: at(c, r) \& at(c, r') \rightarrow fail();
        # An exit direction can only lead to one room.
```

```
nav_r1 :: north_of(r, r') & north_of(r'', r') \rightarrow fail();
        \label{eq:nav_rr2} \begin{array}{l} \mbox{nav\_rr2} \mbox{ :: } \mbox{south\_of(r, r') \& south\_of(r'', r') -> fail();} \\ \mbox{nav\_rr3} \mbox{ :: } \mbox{east\_of(r, r') \& east\_of(r'', r') -> fail();} \\ \end{array}
         nav_rr4 :: west_of(r, r') & west_of(r'', r') -> fail();
         # Two rooms can only be connected once with each other.
        nav_rrA :: north_of(r, r') & south_of(r, r') -> fail();
         nav_rrB :: north_of(r, r') \& west_of(r, r') \rightarrow fail();
        nav_rrC :: north_of(r, r') & east_of(r, r') -> fail();
        nav_rD :: south_of(r, r') & west_of(r, r') \rightarrow fail();
        nav_rrE :: south_of(r, r') & east_of(r, r') -> fail();
        nav_rrF :: west_of(r, r') & east_of(r, r') \rightarrow fail();
    }
    inform7 {
         type {
             kind :: "room";
         }
        predicates {
             at(P, r) :: "The player is in {r}";
             at(t, r) :: "The {t} is in {r}";
             free(r, r') :: ""; # No equivalent in Inform7.
             north_of(r, r') :: "The {r} is mapped north of {r'}";
             south_of(r, r') :: "The {r} is mapped south of {r'}";
             east_of(r, r') :: "The {r} is mapped east of {r'}";
             west_of(r, r') :: "The {r} is mapped west of {r'}";
             north_of/d(r, d, r') :: "South of {r} and north of {r'} is a door called {d}
             south_of/d(r, d, r') :: "North of {r} and south of {r'} is a door called {d}
             east_of/d(r, d, r') :: "West of {r} and east of {r'} is a door called {d}";
             west_of/d(r, d, r') :: "East of {r} and west of {r'} is a door called {d}";
         }
         commands {
             go/north :: "go north" :: "going north";
             go/south :: "go south" :: "going south";
             go/east :: "go east" :: "going east";
             go/west :: "go west" :: "going west";
         }
    }
}
```

# supporter.twl

```
# supporter
type s : t {
    predicates {
        on(o, s);
    }

    inform7 {
        type {
            kind :: "supporter";
            definition :: "supporters are fixed in place.";
        }

        predicates {
            on(o, s) :: "The {o} is on the {s}";
        }
    }
}
```

# thing.twl

```
# thing
type t {
    rules {
        examine/t :: at(P, r) & $at(t, r) -> at(P, r);
    }

    reverse_rules {
        examine/t :: examine/t;
    }

    inform7 {
        type {
            kind :: "thing";
        }

        commands {
            examine/t :: "examine {t}" :: "examining the {t}";
        }
    }
}
```

# 15.6 Inform 7

```
exception textworld.generator.inform7.world2inform7.CouldNotCompileGameError
     Bases: RuntimeError
exception textworld.generator.inform7.world2inform7.TextworldInform7Warning
     Bases: UserWarning
class textworld.generator.inform7.world2inform7.Inform7Game(game)
     Bases: object
     define_inform7_kinds()
          Generate Inform 7 kind definitions.
             Return type str
     detect_action(i7_event, actions)
          Detect which action corresponds to a Inform7 event.
             Parameters
                 • i7_event (str) – Inform7 event detected.
                 • actions (Iterable[Action]) – List of action to match the Inform7 event against.
             Return type Optional[Action]
             Returns Action corresponding to the provided Inform7 event.
     gen_commands_from_actions(actions)
             Return type List[str]
     gen_source(seed=1234)
             Return type str
     gen_source_for_attribute(attr)
             Return type Optional[str]
     gen_source_for_attributes(attributes)
             Return type str
     gen_source_for_conditions(conds)
          Generate Inform 7 source for winning/losing conditions.
             Return type str
     gen_source_for_map(src_room)
             Return type str
     gen_source_for_objects(objects)
             Return type str
     gen_source_for_rooms()
```

```
Return type str

get_human_readable_action(action)

Return type Action

get_human_readable_fact(fact)

Return type Proposition

VERSION = 1

textworld.generator.inform7.world2inform7.compile_inform7_game(source, output, verbose=False)

Return type None

textworld.generator.inform7.world2inform7.generate_inform7_source(game, seed=1234, use_i7_description=False)

Return type str

textworld.generator.inform7.world2inform7.split_string(string, name, cutoff=200)
```

15.6. Inform 7

CHAPTER	
SIXTEEN	

# **TEXTWORLD.CHALLENGES**

# **SEVENTEEN**

# TEXTWORLD.LOGIC

```
class textworld.logic.Action(name, preconditions, postconditions)
     Bases: object
     An action in the environment.
     Create an Action.
          Parameters
                • name (str) – The name of this action.
                • preconditions (Iterable[Proposition]) – The preconditions that must hold before this
                  action is applied.
                • postconditions (Iterable[Proposition]) - The conditions that replace the precondi-
                  tions once applied.
     classmethod deserialize(data)
              Return type Action
     format_command(mapping={})
     inverse(name=None)
          Invert the direction of this action.
              Parameters name (optional) – The new name for the inverse action.
              Return type An action that does the exact opposite of this one.
     classmethod parse(expr)
          Parse an action expression.
              Parameters expr (str) – The string to parse, in the form name ::
                                                                                [$]proposition [&
                  [$]proposition]* -> proposition [& proposition]*.
              Return type Action
     serialize()
              Return type Mapping
     property added: Collection[textworld.logic.Proposition]
          All the new propositions being introduced by this action.
              Return type Collection[Proposition]
```

property all\_propositions: Collection[textworld.logic.Proposition]

All the pre- and post-conditions.

```
Return type Collection[Proposition]
     property removed: Collection[textworld.logic.Proposition]
          All the old propositions being removed by this action.
              Return type Collection[Proposition]
     property variables
class textworld.logic.Alias(pattern, replacement)
     Bases: object
     A shorthand predicate alias.
     expand(predicate)
          Expand a use of this alias into its replacement.
              Return type Collection[Predicate]
class textworld.logic.GameLogic
     Bases: object
     The logic for a game (types, rules, etc.).
     classmethod deserialize(data)
              Return type GameLogic
     classmethod load(paths)
     normalize_rule(rule)
              Return type Rule
     classmethod parse(cls, document)
              Return type GameLogic
     serialize()
              Return type str
class textworld.logic.Inform7Command(rule, command, event)
     Bases: object
     Information about an Inform 7 command.
class textworld.logic.Inform7Logic
     Bases: object
     The Inform 7 bindings of a GameLogic.
class textworld.logic.Inform7Predicate(predicate, source)
     Bases: object
     Information about an Inform 7 predicate.
class textworld.logic.Inform7Type(name, kind, definition=None)
     Bases: object
     Information about an Inform 7 kind.
```

```
class textworld.logic.Placeholder(name, type=None)
     Bases: object
     A symbolic placeholder for a variable in a Predicate.
     Create a Placeholder.
          Parameters
                • name (str) – The name of this placeholder.
                • type (optional) – The type of variable represented. Defaults to the name with any trailing
                  apostrophes stripped.
     classmethod deserialize(data)
              Return type Placeholder
     classmethod parse(expr)
          Parse a placeholder expression.
              Parameters expr (str) – The string to parse, in the form name or name: type.
              Return type Placeholder
     serialize()
              Return type Mapping
     name
     type
class textworld.logic.Predicate(name, parameters)
     Bases: object
     A boolean-valued function over variables.
     Create a Predicate.
          Parameters
                • name (str) – The name of this predicate.
                • parameters (Iterable[Placeholder]) – The symbolic arguments to this predicate.
     classmethod deserialize(data)
              Return type Predicate
     instantiate(mapping)
          Instantiate this predicate with the given mapping.
              Parameters mapping (Mapping[Placeholder, Variable]) - A mapping from Placeholders to
                  Variables.
              Return type The instantiated Proposition with each Placeholder mapped to the corresponding
                  Variable.
     match(proposition)
          Match this predicate against a concrete proposition.
              Parameters proposition (Proposition) – The proposition to match against.
              Return type Optional[Mapping[Placeholder, Variable]]
```

#### **Returns**

- The mapping from placeholders to variables such that self.instantiate(mapping) == proposition, or None if no
- such mapping exists.

## classmethod parse(expr)

Parse a predicate expression.

**Parameters expr** (str) – The string to parse, in the form name(placeholder [, placeholder]\*).

Return type Predicate

serialize()

Return type Mapping

# substitute(mapping)

Copy this predicate, substituting certain placeholders for others.

**Parameters mapping** (Mapping[*Placeholder*, *Placeholder*]) – A mapping from old to new placeholders.

Return type Predicate

# property names: Collection[str]

The names of the placeholders in this predicate.

Return type Collection[str]

# property types: Collection[str]

The types of the placeholders in this predicate.

Return type Collection[str]

## class textworld.logic.Proposition(name, arguments=[])

Bases: mementos.core.NewBase

An instantiated Predicate, with concrete variables for each placeholder.

Create a Proposition.

#### **Parameters**

- **name** (str) The name of the proposition.
- **arguments** (Iterable[*Variable*]) The variables this proposition is applied to.

classmethod deserialize(data)

Return type Proposition

# classmethod parse(expr)

Parse a proposition expression.

**Parameters** expr (str) – The string to parse, in the form name(variable [, variable]\*).

Return type Proposition

serialize()

Return type Mapping

## arguments

#### name

# property names: Collection[str]

The names of the variables in this proposition.

Return type Collection[str]

# signature

# property types: Collection[str]

The types of the variables in this proposition.

Return type Collection[str]

class textworld.logic.Rule(name, preconditions, postconditions)

Bases: object

A template for an action.

Create a Rule.

#### **Parameters**

- name (str) The name of this rule.
- **preconditions** (Iterable[*Predicate*]) The preconditions that must hold before this rule is applied.
- **postconditions** (Iterable[*Predicate*]) The conditions that replace the preconditions once applied.

## classmethod deserialize(data)

# Return type Rule

## instantiate(mapping)

Instantiate this rule with the given mapping.

**Parameters mapping** (Mapping[*Placeholder*, *Variable*]) – A mapping from Placeholders to Variables.

**Return type** The instantiated Action with each Placeholder mapped to the corresponding Variable.

## inverse(name=None)

Invert the direction of this rule.

**Parameters name** (optional) – The new name for the inverse rule.

**Return type** A rule that does the exact opposite of this one.

# match(action)

Match this rule against a concrete action.

**Parameters action** (*Action*) – The action to match against.

**Return type** Optional[Mapping[Placeholder, Variable]]

# Returns

- The mapping from placeholders to variables such that self.instantiate(mapping) == action, or None if no such
- mapping exists.

```
classmethod parse(expr)
          Parse a rule expression.
              Parameters expr (str) - The string to parse, in the form name :: [$]predicate [&
                   [$]predicate]* -> predicate [& predicate]*.
              Return type Rule
     serialize()
              Return type Mapping
     substitute(mapping, name=None)
          Copy this rule, substituting certain placeholders for others.
              Parameters mapping (Mapping[Placeholder, Placeholder]) – A mapping from old to new
                  placeholders.
              Return type Rule
     property all_predicates: Iterable[textworld.logic.Predicate]
          All the pre- and post-conditions.
              Return type Iterable[Predicate]
class textworld.logic.Signature(name, types)
     Bases: mementos.core.NewBase
     The type signature of a Predicate or Proposition.
     Create a Signature.
          Parameters
                • name (str) – The name of the proposition/predicate this signature is for.
                • types (Iterable[str]) – The types of the parameters to the proposition/predicate.
     classmethod parse(expr)
          Parse a signature expression.
              Parameters expr (str) – The string to parse, in the form name(type [, type]*).
              Return type Signature
     name
     types
class textworld.logic.State(logic, facts=None)
     Bases: object
     The current state of a world.
     Create a State.
          Parameters
                • logic (GameLogic) – The logic for this state's game.
                • facts (optional) – The facts that will be true in this state.
     add_fact(prop)
          Add a fact to the state.
     add_facts(props)
          Add some facts to the state.
```

# all\_applicable\_actions(rules, mapping=None)

Get all the rule instantiations that would be valid actions in this state.

#### **Parameters**

- **rules** (Iterable[*Rule*]) The possible rules to instantiate.
- **mapping** (*optional*) An initial mapping to start from, constraining the possible instantiations.

**Return type** The actions that can be instantiated from the given rules in this state.

#### **all\_assignments**(rule, mapping=None, partial=False, allow\_partial=None)

Find all possible placeholder assignments that would allow a rule to be instantiated in this state.

#### **Parameters**

- **rule** (*Rule*) The rule to instantiate.
- **mapping** (*optional*) An initial mapping to start from, constraining the possible instantiations.
- **partial** (*optional*) Whether incomplete mappings, that would require new variables or propositions, are allowed.
- **allow\_partial** (*optional*) A callback function that returns whether a partial match may involve the given placeholder.

**Return type** Iterable[Mapping[*Placeholder*, Optional[*Variable*]]]

#### Returns

- The possible mappings for instantiating the rule. Partial mappings requiring new variables will have None in
- place of existing Variables.

# all\_instantiations(rule, mapping=None)

Find all possible actions that can be instantiated from a rule in this state.

#### **Parameters**

- **rule** (*Rule*) The rule to instantiate.
- **mapping** (*optional*) An initial mapping to start from, constraining the possible instantiations.

**Return type** The actions that can be instantiated from the rule in this state.

## apply(action)

Apply an action to the state.

**Parameters action** (*Action*) – The action to apply.

**Return type** Whether the action could be applied (i.e. whether the preconditions were met).

# apply\_on\_copy(action)

Apply an action to a copy of this state.

**Parameters action** (*Action*) – The action to apply.

Return type Optional[State]

# Returns

- The copied state after the action has been applied or None if action
- wasn't applicable.

```
are_facts(props)
     Returns whether the propositions are all true in this state.
         Return type bool
copy()
     Create a copy of this state.
         Return type State
classmethod deserialize(data)
     Deserialize a State object from data.
         Return type State
facts_with_signature(sig)
     Returns all the known facts with the given signature.
         Return type Set[Proposition]
has_variable(var)
     Returns whether this state is aware of the given variable.
         Return type bool
is_applicable(action)
     Check if an action is applicable in this state (i.e. its preconditions are met).
         Return type bool
is_fact(prop)
     Returns whether a proposition is true in this state.
         Return type bool
is_sequence_applicable(actions)
     Check if a sequence of actions are all applicable in this state.
         Return type bool
remove_fact(prop)
     Remove a fact from the state.
remove_facts(props)
     Remove some facts from the state.
serialize()
     Serialize this state.
         Return type Sequence
variable_named(name)
     Returns the variable with the given name, if known.
         Return type Variable
variables_of_type(type)
     Returns all the known variables of the given type.
         Return type Set[Variable]
property facts: Iterable[textworld.logic.Proposition]
```

All the facts in the current state.

**Return type** Iterable[*Proposition*]

```
property variables: Iterable[textworld.logic.Variable]
          All the variables tracked by the current state.
             Return type Iterable[Variable]
class textworld.logic.Type(name, parents)
     Bases: object
     A variable type.
     has_subtype_named(name)
             Return type bool
     has_supertype_named(name)
             Return type bool
     is_subtype_of(other)
             Return type bool
     is_supertype_of(other)
             Return type bool
     property ancestors: Iterable[textworld.logic.Type]
          The ancestors of this type (not including itself).
             Return type Iterable[Type]
     property child_types: Iterable[textworld.logic.Type]
          The direct children of this type.
              Return type Iterable[Type]
     property children: Iterable[str]
          The names of the direct children of this type.
              Return type Iterable[str]
     property descendants: Iterable[textworld.logic.Type]
          The descendants of this type (not including itself).
             Return type Iterable[Type]
     property parent_types: Iterable[textworld.logic.Type]
          The parents of this type as Type objects.
             Return type Iterable[Type]
     property subtypes: Iterable[textworld.logic.Type]
          This type and its descendants.
             Return type Iterable[Type]
     property supertypes: Iterable[textworld.logic.Type]
         This type and its ancestors.
             Return type Iterable[Type]
```

# class textworld.logic.TypeHierarchy

Bases: object

A hierarchy of types.

add(type)

## closure(type, expand)

Compute the transitive closure in a type lattice according to some type relationship (generally direct sub-/super-types).

Such a lattice may look something like this:



so the closure of D would be something like [B, C, A].

Return type Iterable[Type]

get(name)

Return type Type

# multi\_ancestors(types)

Compute the ancestral closure of a sequence of types. If these are the types of some variables, the result will be all the function parameter types that could also accept those variables.

**Return type** Iterable[Collection[*Type*]]

# multi\_closure(types, expand)

Compute the transitive closure of a sequence of types in a type lattice induced by some per-type relationship (generally direct sub-/super-types).

For a single type, such a lattice may look something like this:

```
A / \ B C \ \ / D
```

so the closure of D would be something like [B, C, A]. For multiple types at once, the lattice is more complicated:

**Return type** Iterable[Collection[*Type*]]

```
multi_descendants(types)
          Compute the descendant closure of a sequence of types. If these are the types of some function parameters,
          the result will be all the variable types that could also be passed to this function.
              Return type Iterable[Collection[Type]]
     multi_subtypes(types)
          Computes the descendant closure of a sequence of types, including the initial types.
              Return type List[Collection[Type]]
     multi_supertypes(types)
          Computes the ancestral closure of a sequence of types, including the initial types.
              Return type Iterable[Collection[Type]]
class textworld.logic.Variable(name, type=None)
     Bases: object
     A variable representing an object in a world.
     Create a Variable.
          Parameters
                • name (str) – The (unique) name of the variable.
                • type (optional) – The type of the variable. Defaults to the same as the name.
     classmethod deserialize(data)
              Return type Variable
     is_a(type)
              Return type bool
     classmethod parse(expr)
          Parse a variable expression.
              Parameters expr (str) – The string to parse, in the form name or name: type.
              Return type Variable
     serialize()
              Return type Mapping
     name
     type
class textworld.logic.model.ActionNode(ctx=None, ast=None, parseinfo=None, **kwargs)
     Bases: textworld.logic.model.ModelBase
     name = None
     postconditions = None
     preconditions = None
class textworld.logic.model.ActionPreconditionNode(ctx=None, ast=None, parseinfo=None, **kwargs)
     Bases: textworld.logic.model.ModelBase
     condition = None
```

```
preserve = None
class textworld.logic.model.AliasNode(ctx=None, ast=None, parseinfo=None, **kwargs)
     Bases: textworld.logic.model.ModelBase
     lhs = None
     rhs = None
class textworld.logic.model.ConstraintsNode(ctx=None, ast=None, parseinfo=None, **kwargs)
     Bases: textworld.logic.model.ModelBase
     constraints = None
class textworld.logic.model.DocumentNode(ctx=None, ast=None, parseinfo=None, **kwargs)
     Bases: textworld.logic.model.ModelBase
     types = None
class textworld.logic.model.GameLogicModelBuilderSemantics(context=None, types=None)
     Bases: tatsu.semantics.ModelBuilderSemantics
class textworld.logic.model.Inform7CodeNode(ctx=None, ast=None, parseinfo=None, **kwargs)
     Bases: textworld.logic.model.ModelBase
     code = None
class textworld.logic.model.Inform7CommandNode(ctx=None, ast=None, parseinfo=None, **kwargs)
     Bases: textworld.logic.model.ModelBase
     command = None
     event = None
     rule = None
class textworld.logic.model.Inform7CommandsNode(ctx=None, ast=None, parseinfo=None, **kwargs)
     Bases: textworld.logic.model.ModelBase
     commands = None
class textworld.logic.model.Inform7Node(ctx=None, ast=None, parseinfo=None, **kwargs)
     Bases: textworld.logic.model.ModelBase
     parts = None
class textworld.logic.model.Inform7PredicateNode(ctx=None, ast=None, parseinfo=None, **kwargs)
     Bases: textworld.logic.model.ModelBase
     predicate = None
     source = None
class textworld.logic.model.Inform7PredicatesNode(ctx=None, ast=None, parseinfo=None, **kwargs)
     Bases: textworld.logic.model.ModelBase
     predicates = None
class textworld.logic.model.Inform7TypeNode(ctx=None, ast=None, parseinfo=None, **kwargs)
     Bases: textworld.logic.model.ModelBase
     definition = None
     kind = None
class textworld.logic.model.ModelBase(ctx=None, ast=None, parseinfo=None, **kwargs)
     Bases: tatsu.objectmodel.Node
```

```
class textworld.logic.model.PlaceholderNode(ctx=None, ast=None, parseinfo=None, **kwargs)
     Bases: textworld.logic.model.ModelBase
     name = None
     type = None
class textworld.logic.model.PredicateNode(ctx=None, ast=None, parseinfo=None, **kwargs)
     Bases: textworld.logic.model.ModelBase
     name = None
     parameters = None
class textworld.logic.model.PredicatesNode(ctx=None, ast=None, parseinfo=None, **kwargs)
     Bases: textworld.logic.model.ModelBase
     predicates = None
class textworld.logic.model.PropositionNode(ctx=None, ast=None, parseinfo=None, **kwargs)
     Bases: textworld.logic.model.ModelBase
     arguments = None
     name = None
class textworld.logic.model.ReverseRuleNode(ctx=None, ast=None, parseinfo=None, **kwargs)
     Bases: textworld.logic.model.ModelBase
     lhs = None
     rhs = None
class textworld.logic.model.ReverseRulesNode(ctx=None, ast=None, parseinfo=None, **kwargs)
     Bases: textworld.logic.model.ModelBase
     reverse_rules = None
class textworld.logic.model.RuleNode(ctx=None, ast=None, parseinfo=None, **kwargs)
     Bases: textworld.logic.model.ModelBase
     name = None
     postconditions = None
     preconditions = None
class textworld.logic.model.RulePreconditionNode(ctx=None, ast=None, parseinfo=None, **kwargs)
     Bases: textworld.logic.model.ModelBase
     condition = None
     preserve = None
class textworld.logic.model.RulesNode(ctx=None, ast=None, parseinfo=None, **kwargs)
     Bases: textworld.logic.model.ModelBase
     rules = None
class textworld.logic.model.SignatureNode(ctx=None, ast=None, parseinfo=None, **kwargs)
     Bases: textworld.logic.model.ModelBase
     name = None
     types = None
class textworld.logic.model.TypeNode(ctx=None, ast=None, parseinfo=None, **kwargs)
     Bases: textworld.logic.model.ModelBase
```

```
name = None
     parts = None
     supertypes = None
class textworld.logic.model.VariableNode(ctx=None, ast=None, parseinfo=None, **kwargs)
     Bases: textworld.logic.model.ModelBase
     name = None
     type = None
class textworld.logic.parser.GameLogicBuffer(text, whitespace=None, nameguard=None,
                                                 comments_re=None, eol_comments_re='#.*$',
                                                 ignorecase=None, namechars=", **kwargs)
     Bases: tatsu.buffering.Buffer
class textworld.logic.parser.GameLogicParser(whitespace=None, nameguard=None,
                                                 comments_re=None, eol_comments_re='#.*$',
                                                 ignorecase=None, left_recursion=True, parseinfo=True,
                                                 keywords=None, namechars=", buffer_class=<class
                                                 'textworld.logic.parser.GameLogicBuffer'>, **kwargs)
     Bases: tatsu.parsing.Parser
class textworld.logic.parser.GameLogicSemantics
     Bases: object
     action(ast)
     actionPrecondition(ast)
     alias(ast)
     constraints(ast)
     document(ast)
     inform7(ast)
     inform7Code(ast)
     inform7Command(ast)
     inform7Commands(ast)
     inform7Part(ast)
     inform7Predicate(ast)
     inform7Predicates(ast)
     inform7Type(ast)
     name(ast)
     onlyAction(ast)
     onlyPlaceholder(ast)
     onlyPredicate(ast)
     onlyProposition(ast)
     onlyRule(ast)
     onlySignature(ast)
```

```
onlyVariable(ast)
     phName(ast)
     placeholder(ast)
     predName(ast)
     predicate(ast)
     predicateDecls(ast)
     predicates(ast)
     proposition(ast)
     reverseRule(ast)
     reverseRuleDecls(ast)
     reverseRules(ast)
     rule(ast)
     ruleDecls(ast)
     ruleName(ast)
     rulePrecondition(ast)
     rules(ast)
     signature(ast)
     signatureOrAlias(ast)
     start(ast)
     str(ast)
     strBlock(ast)
     type(ast)
     typePart(ast)
     variable(ast)
textworld.logic.parser.main(filename, start=None, **kwargs)
```

# **EIGHTEEN**

# **TEXTWORLD.RENDER**

```
exception textworld.render.render.WebdriverNotFoundError
     Bases: Exception
class textworld.render.render.GraphItem(type, name)
     Bases: object
     add_content(content)
     add_unknown_predicate(predicate)
     get_max_depth()
          Returns the maximum nest depth of this plus all children. A container with no items has 1 depth, a container
          containing one item has 2 depth, a container containing a container which contains an item has 3 depth,
          and so on. :return: maximum nest depth
     set_open_closed_locked(status)
     to_dict()
     property infos
class textworld.render.render.GraphRoom(name, base_room)
     Bases: object
     add_item(item)
              Return type None
     position_string()
              Return type str
textworld.render.render.concat_images(*images)
textworld.render.get_webdriver(path=None)
     Get the driver and options objects. :param path: path to browser binary. :return: driver
textworld.render.load_state(world, game_infos=None, action=None, format='png',
                                        limit_player_view=False)
     Generates serialization of game state.
          Parameters
                • world (World) – The current state of the world to visualize.
```

• game\_infos (Optional[Dict[str, EntityInfo]]) - The mapping needed to get objects

names.

- action (Optional [Action]) If provided, highlight the world changes made by that action.
- **format** (str) The graph output format (gv, svg, png...)
- limit\_player\_view (bool) Whether to limit the player's view (defaults to false)

# Return type dict

**Returns** The graph generated from this World

textworld.render.render.load\_state\_from\_game\_state(game\_state, format='png', limit\_player\_view=False)

Generates serialization of game state.

#### **Parameters**

- **game\_state** (*GameState*) The current game state to visualize.
- **format** (str) The graph output format (png, svg, pdf, ...)
- limit\_player\_view (bool) Whether to limit the player's view. Default: False.

#### Return type dict

**Returns** The graph generated from this World

## textworld.render.render.take\_screenshot(url, id='world')

Takes a screenshot of DOM element given its id. :type url: str :param url: URL of webpage to open headlessly. :type id: str :param id: ID of DOM element. :return: Image object.

```
textworld.render.render.temp_viz(nodes, edges, pos, color=[])
```

#### textworld.render.render.visualize(world, interactive=False)

Show the current state of the world. :type world: Union[Game, State, GameState, World] :param world: Object representing a game state to be visualized. :type interactive: bool :param interactive: Whether or not to visualize the state in the browser. :return: Image object of the visualization.

# textworld.render.render.which(program)

helper to see if a program is in PATH :param program: name of program :return: path of program or None

Creates server for streamed game state

#### class textworld.render.serve.Server(game\_state, port)

Bases: object

Visualization server. Uses Server-sent Events to update game state for visualization.

Note: Flask routes are defined in app.add\_url\_rule in order to call self in routes. :type game\_state: dict :param game\_state: game state returned from load\_state\_from\_game\_state :type port: int :param port: port to run visualization on

#### gen()

Our generator for listening for updating state. We poll for results to return us something. If nothing is returned then we just pass and keep polling. :return: yields event-stream parsed data.

#### index()

Index route ("/"). Returns HTML template processed by handlebars. :rtype: str :return: Flask response object

#### static listen(conn, results)

Listener for updates. Runs on separate thread. :type conn: Connection :param conn: child connection from multiprocessing.Pipe. :type results: Queue :param results: thread-safe queue for results.

#### start(child conn)

Starts the WSGI server and listen for updates on a separate thread.

Parameters child\_conn (Connection) - Child connection from multiprocessing. Pipe.

#### subscribe()

Our Server-sent Event stream route. :return: A stream

#### update\_subscribers(game state)

Updates all subscribers and updates their data. This is for multiple subscribers on the visualization service. :type game\_state: dict :param game\_state: parsed game\_state from load\_state\_from\_game\_state

## class textworld.render.serve.ServerSentEvent(data)

Bases: object

Object helper to parse dict into SSE data. :type data: any :param data: data to pass to SSE

encode()

## class textworld.render.serve.SupressStdStreams

Bases: object

for surpressing std.out streams

# class textworld.render.serve.VisualizationService(game\_state, open\_automatically)

Bases: object

Server for visualization.

We instantiate a new process for our flask server, so our game can send updates to the server. The server instantiates new gevent Queues for every connection.

# start(parent\_thread, port)

Start visualization server on a new process. :type parent\_thread: Thread :param parent\_thread: the parent thread that called start. :type port: int :param port: Port to run visualization on.

# Return type None

# start\_server(game\_state, port, child\_conn)

function for starting new server on new process. :type game\_state: dict :param game\_state: initial game state from load :type port: int :param port: port to run server :type child\_conn: Connection :param child\_conn: child connection from multiprocessing.Pipe

stop\_server()

# update\_state(game\_state, command)

Propogate state update to server. We use a multiprocessing. Pipe to pass state into flask process. :type game\_state: GameState :param game\_state: Glulx game state. :type command: str :param command: previous command

textworld.render.serve.find\_free\_port(port\_range)

textworld.render.serve.get\_html\_template(game state=None)

# **NINETEEN**

# **TEXTWORLD.UTILS**

```
class textworld.utils.RandomGenerator(seed=None)
     Bases: object
     Random generator controlling the games generation.
     next()
          Start a new random generator using a new seed.
     set_seed(seed)
     property seed
class textworld.utils.RegexDict
     Bases: collections.OrderedDict
     Ordered dictionary that supports querying with regex.
     References
     Adapted from https://stackoverflow.com/questions/21024822/python-accessing-dictionary-with-wildcards.
     get_matching(*regexes, exclude=[])
          Query the dictionary using one or several regular expressions.
                   • *regexes - List of regular expressions determining which keys of this dictionary are rel-
                    evant to this query.
                   • exclude (List[str]) - List of regular expressions determining which keys of this dictio-
                    nary should be excluded from this query.
              Return type List[Any]
              Returns The value associated to each relevant (and not excluded) keys.
textworld.utils.check_modules(required_modules, missing_modules)
     Check whether some required modules are missing.
textworld.utils.chunk(iterable, n, fct=<function <lambda>>)
          Return type Iterable[Iterable]
textworld.utils.encode_seeds(seeds)
     Generate UID from a list of seeds.
textworld.utils.make_temp_directory(suffix=", prefix='tw_', dir=None)
```

Create temporary folder to used in a with statement.

```
textworld.utils.maybe_mkdir(dirpath)
```

Create all parent folders if needed.

```
textworld.utils.save_graph_to_svg(G, labels, filename, backward=False)
```

Generate a figure of a networkx's graph object and save it.

```
textworld.utils.str2bool(v)
```

Convert string to a boolean value. .. rubric:: References

https://stackoverflow.com/questions/715417/converting-from-a-string-to-boolean-in-python/715468#715468

```
textworld.utils.take(n, iterable)
```

Return first n items of the iterable as a list.

## References

https://docs.python.org/3/library/itertools.html#itertools-recipes

```
Return type Iterable
```

```
textworld.utils.unique_product(*iterables)
```

Cartesian product of input iterables with pruning.

This method prunes any product tuple with duplicate elements in it.

# **Example**

```
unique_product('ABC', 'Ax', 'xy') -> Axy BAx BAy Bxy CAx CAy Cxy
```

# **Notes**

This method is faster than the following equivalent code:

# ${\tt textworld.utils.} \textbf{uniquify} (\textit{seq})$

Order preserving uniquify.

#### References

Made by Dave Kirby https://www.peterbe.com/plog/uniqifiers-benchmark

```
textworld.utils.g_rng = <textworld.utils.RandomGenerator object> Global random generator.
```

# **CHAPTER**

# **TWENTY**

# **INDICES AND TABLES**

- genindex
- modindex
- search

# **BIBLIOGRAPHY**

[Parisotto2017] Emilio Parisotto and Ruslan Salakhutdinov. Neural map: Structured memory for deep reinforcement learning. arXiv:1702.08360, 2017.

128 Bibliography

# **PYTHON MODULE INDEX**

```
t
                                                textworld.gym.spaces.text_spaces, 43
                                                textworld.gym.utils, 37
textworld, 31
                                                textworld.logic, 103
textworld.agents, 53
                                                textworld.logic.model, 113
textworld.agents.human, 53
                                                textworld.logic.parser, 116
textworld.agents.random, 53
                                                textworld.render, 119
textworld.agents.simple, 54
                                                textworld.render.render, 119
textworld.agents.walkthrough, 54
                                                textworld.render.serve, 120
textworld.challenges.coin_collector, 23
                                                textworld.utils, 123
textworld.challenges.cooking, 25
textworld.challenges.simple, 22
textworld.challenges.treasure_hunter, 28
textworld.core, 31
textworld.envs.47
textworld.envs.glulx, 48
textworld.envs.glulx.git_glulx, 48
textworld.envs.tw, 47
textworld.envs.wrappers, 49
textworld.envs.wrappers.filter, 49
textworld.envs.wrappers.recorder, 49
textworld.envs.wrappers.viewer, 49
textworld.envs.zmachine, 50
textworld.envs.zmachine.jericho, 50
textworld.generator, 55
textworld.generator.chaining, 56
textworld.generator.data, 88
textworld.generator.dependency_tree, 58
textworld.generator.game, 61
textworld.generator.graph_networks, 75
textworld.generator.inform7,98
textworld.generator.inform7.world2inform7,98
textworld.generator.logger, 60
textworld.generator.maker, 76
textworld.generator.text_generation, 83
textworld.generator.text_grammar, 84
textworld.generator.vtypes, 60
textworld.generator.world,72
textworld.gym, 37
textworld.gym.envs, 40
textworld.gym.envs.textworld, 40
textworld.gym.envs.textworld_batch, 41
textworld.gym.envs.utils, 43
```

textworld.gym.spaces, 43

130 Python Module Index

# **INDEX**

act() (textworld.agents.human.HumanAgent method), 53 act() (textworld.agents.random.NaiveAgent method), 53 act() (textworld.agents.random.RandomCommandAgent method), 53 act() (textworld.agents.simple.NaiveAgent method), 54 act() (textworld.agents.walkthrough.WalkthroughAgent method), 54 act() (textworld.agents.walkthrough.WalkthroughAgent method), 54 act() (textworld.core-Agent method), 31 act() (textworld.core-Agent method), 31 act() (textworld.generator.chaining.ChainNode attribute), 56 action (textworld.logic.parser.GameLogicSemantics method), 116 actionDependencyTree (class in textworld.logic.parser.GameLogicSemantics method), 116 ActionDependencyTreeElement (class in textworld.generator.game), 61 ActionPependencyTreeElement (class in textworld.generator.game.Generator.game), 61 ActionPependencyTreeElement (class in textworld.generator.game.Generator.game), 61 ActionPependencyTreeElement (class in textworld.logic.parser.GameLogicSemantics in textworld.logic.model), 113 actionS (textworld.generator.game.Event attribute), 63 actionS (textworld.generator.game.Event property), 64 add() (textworld.gene	A	72
act() (textworld.agents.random.NaiveAgent method), 53 act() (textworld.agents.random.RandomCommandAgent method), 53 act() (textworld.agents.simple.NaiveAgent method), 54 act() (textworld.agents.simple.NaiveAgent method), 54 act() (textworld.agents.walkthrough.WalkthroughAgent method), 54 act() (textworld.agents.malchol.gic.) 103 act() (textworld.core.Agent method), 39 act() (textworld.gencator.Agent method), 39 act() (textworld.generator.chaining.ChainNode attribute), 56 action (textworld.generator.game.ActionDependencyTree action( (textworld.logic.parser.GameLogicSemantics method), 116 ActionDependencyTreeElement (class in textworld.generator.game), 61 ActionNode (class in textworld.logic.model), 113 action( (textworld.logic.parser.GameLogicSemantics method), 116 ActionNode (class in textworld.logic.model), 113 action( (textworld.generator.game.Event attribute), 55 actions (textworld.generator.chaining.Chain attribute), 56 actions (textworld.generator.game.Event attribute), 63 actions (textworld.generator.game.Event attribute), 63 actions (textworld.generator.maker.WorldEntity method), 81  AliasNode (class in textworld.logic.state method), 108 all_applicable_actions() (textworld.logic.State method), 109 all_instantiations() (textworld.logic.State method), 109 all_predicates (textworld.logic.State method), 109 all_predicates (textworld.generator.chaining.ChainingOptions attribute), 57 allowed_types (textworld.generator.chaining.ChainingOptions attribute), 57 allowed_types (textworld.generator.chaining.ChainingOptions attribute), 57 ambiguous_instructions	act() (textworld.agents.human.HumanAgent method),	
act () (textworld.agents.random.RandomCommandAgent method), 53 act () (textworld.agents.random.RandomCommandAgent method), 53 act () (textworld.agents.simple.NaiveAgent method), 54 act () (textworld.synt.synthylough.WalkthroughAgent method), 54 act () (textworld.gynt.core.Agent method), 31 act () (textworld.gynt.core.Agent method), 39 Act () (textworld.gynt.core.Agent method), 31 act () (textworld.gynt.core.Agent method), 39 Act () (textworld.gynt.core.Agent method), 30 add_reatex() (textworld.gynt.core.Agent.gynt.gynt.gynt.gynt.gynt.gynt.gynt.gy	53	
act() (textworld.agents.simple.NaiveAgent method), 54 act() (textworld.agents.walkthrough.WalkthroughAgent method), 54 act() (textworld.agents.maken.WorldEntity method), 81 add_related_fact() (textworld.generator.world.WorldEntity method), 81 add_related_fact() (textworld.generator.world.WorldEntity method), 81 add_related_fact() (textworld.generator.world.WorldEntity method), 81 add_related_fact() (textworld.generator.world.WorldEntity method), 74 add_unknom_predicate() (textworld.legic.Action property), 103 adj (textworld.legic.Action property), 103 adj (textworld.generator.game.EntityInfo attribute), 52 admissible_commands (textworld.generator.game.EntityInfo attribute), 32 Agent (class in textworld.generator.logger.GameLogger method), 109 allias(class in textworld.logic), 104 alias() actionPrecondition() (textworld.generator.game.EntityInfo attribute), 53 aggregate() (textworld.generator.logger.GameLogger method), 116 ActionNode (class in textworld.logic.model), 113 actionPrecondition() (textworld.generator.game.Event attribute), 53 actions (textworld.generator.maker.WorldEntity method), 80 add() (textworld.logic.model), 113 actionS(textworld.generator.game.Event attribute), 63 actions (textworld.generator.maker.WorldEntity method), 80 add() (textworld.generator.maker.WorldEntity method), 80 add() (textworld.logic.TypeHierarchy method), 112 add_content() (textworld.render.render.GraphItem method), 119 add_logic.Action property), 108 all_applicable_actions() (textworld.logic.State method), 109 all_instantiations() (textworld.logic.State method), 109 all_propositions (textworld.logic.Action property), 103 allowed_veriables_num	<pre>act() (textworld.agents.random.NaiveAgent method), 53</pre>	· <del>-</del>
act() (textworld.agents.simple.NaiveAgent method), 54 act() (textworld.agents.walkthrough.WalkthroughAgent method), 54 act() (textworld.core.Agent method), 31 act() (textworld.core.Agent method), 39 Action (class in textworld.logic), 103 action (textworld.generator.chaining.ChainNode attribute), 56 action(textworld.generator.game.ActionDependencyTreeEndedf (textworld.logic.Action property), 103 aproperty), 62 action() (textworld.logic.parser.GameLogicSemantics method), 116 ActionDependencyTree (class in textworld.logic.parser.GameLogicSemantics method), 116 ActionPeendencyTreeEndedf (textworld.generator.game.EntityInfo attribute), 62 add (textworld.generator.game), 61 ActionPeendencyTreeCement (class in textworld.core), 31 Agent (class in textworld.core), 31 Agent (class in textworld.generator.logger.GameLogger method), 116 ActionPeendencyTreeCement (class in textworld.logic.model), 113 actionSequencyTreeCement (class in textworld.logic.model), 113 actionPeendencyTreeCement (class in textworld.logic.model), 113 actionPeendencyTreeCenent (class in textworld.logic.model), 114 all_applicable_actions() (textworld.logic.State method), 109 all_applicable_actions() (textworld.logic.State method), 109 all_propositions (textworld.logic.State method), 109 all_propositions (textworld.logic.Action property), 108 all_propositions (textworld.logic.State method), 109 all_propositions (textworld.log	act() (textworld.agents.random.RandomCommandAgent	
act() (textworld.agents.walkthrough.WalkthroughAgent method), 54 act() (textworld.core.Agent method), 31 act() (textworld.core.Agent method), 31 act() (textworld.generator.edgent method), 31 act() (textworld.generator.chaining.ChainNode action (textworld.generator.chaining.ChainNode artibute), 56 action (textworld.generator.game.ActionDependencyTreeEnddeft (textworld.logic.Action property), 62 action() (textworld.generator.game.ActionDependencyTreeEnddeft (textworld.logic.Action property), 103 adj (textworld.logic.Action property), 103 adj (textworld.generator.game.EntityInfo attribute), 62 addinals (textworld.generator.game.EntityInfo attribute), 32 Agent (class in textworld.generator.game.EntityInfo attribute), 32 Agent (class in textworld.generator.game.EntityInfo attribute), 33 ActionPeendencyTree (class in textworld.generator.game), 61 ActionNode (class in textworld.logic.model), 113 actionPeendencyTree (class in textworld.logic.generator.game), 61 ActionNode (class in textworld.logic.model), 113 actionPeendencyTree (class in textworld.logic.model), 113 actionPeendencyTree (class in textworld.logic.generator.game.Event attribute), 63 action (textworld.generator.game.Event attribute), 63 action (textworld.generator.game.Event attribute), 63 action (textworld.generator.game.Event attribute), 63 actions (textworld.generator.game.Event attribute), 63 actions (textworld.generator.maker.WorldEntity method), 109 all_assignments() (textworld.logic.State method), 109 all_propositions (textworld.logic.State method), 109 all_propositions (textworld.logic.Action property), 108 all_propositions (textworld.generator.chaining.ChainingOptions attribute), 57 allowed_types (textworld.generator.chaining.ChainingOptions attribute), 87 ambiguous_instructions	method), 53	=
method), 54 act() (textworld.core.Agent method), 31 act() (textworld.generator.chaining.ChainNode attribute), 56 action(textworld.generator.game.ActionDependencyTreeEndmethol), 116 ActionDependencyTreeElement (class in textworld.logic.parser.game), 61 ActionDependencyTreeElement (class in textworld.logic.parser.GameLogicSemantics method), 116 ActionDependencyTreeElement (class in textworld.logic.parser.GameLogicSemantics method), 116 ActionDependencyTreeElement (class in textworld.logic.parser.GameLogicSemantics method), 116 ActionDependencyTreeElement (class in textworld.generator.game), 61 ActionDependencyTreeElement (class in textworld.logic.model), 113 actions (textworld.logic.parser.GameLogicSemantics method), 116 ActionPrecondition() (textworld.logic.model), 113 actions (textworld.logic.model), 113 actions (textworld.generator.chaining.Chain attribute), 55 actions (textworld.generator.game.Event property), 64 add() (textworld.generator.game.Event attribute), 63 actions (textworld.generator.maker.WorldEntity method), 119 add_fact() (textworld.generator.maker.GameMaker method), 76 add_fact() (textworld.generator.maker.WorldEntity method), 81	act() (textworld.agents.simple.NaiveAgent method), 54	
action (class in textworld.logic), 103 action (textworld.generator.chaining.ChainNode attribute), 56 action(textworld.generator.game.ActionDependencyTreeEndded(textworld.logic.parser.GameLogicSemantics method), 116 ActionDependencyTree (class in textworld.generator.game), 61 ActionDependencyTreeElement (class in textworld.generator.game), 61 ActionPrecondition() (textworld.logic.model), 113 actionPreconditionNode (class in textworld.logic.model), 113 actionStextworld.generator.gameLogicSemantics method), 116 ActionPreconditionNode (class in textworld.logic.model), 113 actions (textworld.generator.game.Event attribute), 53 actions (textworld.generator.game.Event attribute), 63 actions (textworld.generator.game.Event attribute), 63 add() (textworld.generator.game.Event property), 64 add()		method), 81
Action (class in textworld.logic), 103 action (textworld.generator.game.ActionDependencyTreeEndled (textworld.render.render.GraphItem method), 119 action(textworld.generator.game.ActionDependencyTreeEndled (textworld.logic.Action property), 103 adj (textworld.logic.Action property), 103 adj (textworld.generator.game.EntityInfo attribute), 62 admissible_commands (textworld.core.EnvInfos attribute), 32 addinssible_commands (textworld.core.EnvInfos attribute), 32 addinssible_commands (textworld.core.EnvInfos attribute), 33 adj (textworld.generator.game.EntityInfo attribute), 62 admissible_commands (textworld.core.EnvInfos attribute), 33 aggregate() (textworld.generator.logger.GameLogger method), 60 ActionNode (class in textworld.logic.parser.GameLogger method), 60 ActionPreconditionNode (class in textworld.logic.parser.GameLogger method), 116 ActionPreconditionNode (class in textworld.logic.parser.GameLogicSemantics method), 116 ActionPreconditionNode (class in textworld.logic.State method), 118 actions (textworld.generator.game.Event attribute), 63 actions (textworld.generator.game.Event attribute), 63 actions (textworld.generator.game.Event attribute), 63 actions (textworld.generator.game.Event attribute), 63 actions (textworld.generat	act() (textworld.core.Agent method), 31	
action (textworld.generator.chaining.ChainNode attribute), 56 action(textworld.generator.game.ActionDependencyTreeEnded (textworld.logic.Action property), 103 adj (textworld.generator.game.EntityInfo attribute), 62 action() (textworld.logic.parser.GameLogicSemantics method), 116 ActionDependencyTree (class in textworld.generator.game), 61 ActionDependencyTreeElement (class in textworld.generator.game), 61 ActionNode (class in textworld.logic.model), 113 actionPrecondition() (textworld.logic.parser.GameLogicSemantics method), 116 ActionPreconditionNode (class in textworld.logic.parser.GameLogicSemantics method), 118 actions (textworld.generator.game.Event attribute), 63 actions (textworld.generator.game.Event attribut	act() (textworld.gym.core.Agent method), 39	
ribute), 56 action (textworld.generator.game.ActionDependencyTreeEndded (textworld.logic.Action property), 62 action() (textworld.logic.parser.GameLogicSemantics method), 116 ActionDependencyTree (class in textworld.generator.game), 61 ActionDependencyTreeElement (class in textworld.generator.logger.GameLogger method), 60 ActionDependencyTreeElement (class in textworld.logic.parser.GameLogger.gametlogger method), 60 ActionDependencyTreeElement (class in textworld.logic.parser.GameLogger.gametlogger.ga	Action (class in textworld.logic), 103	
action() (textworld.logic.parser.GameLogicSemantics     method), 116  ActionDependencyTree (class in textworld.generator.game), 61 ActionDependencyTreeElement (class in textworld.generator.game), 61 ActionNode (class in textworld.logic.model), 113 actionPrecondition() (textworld.logic.parser.GameLogicSemantics method), 116 ActionPreconditionNode (class in textworld.logic.parser.GameLogicSemantics method), 116 AliasNode (class in textworld.logic.model), 114 all_applicable_actions() (textworld.logic.State method), 109 all_assignments() (textworld.logic.State method), 109 all_instantiations() (textworld.logic.Rule property), 108 all_propositions (textworld.logic.Action property), 108 all_propositions (textworld.generator.chaining.ChainingOptions attribute), 57 allowed_variables_numbering (textworld.generator.text_grammar.GrammarOptions attribute), 87 ambiguous_instructions	tribute) 56	119
action() (textworld.logic.parser.GameLogicSemantics     method), 116  ActionDependencyTree (class in textworld.generator.game), 61 ActionDependencyTreeElement (class in textworld.generator.game), 61 ActionNode (class in textworld.logic.model), 113 actionPrecondition() (textworld.logic.parser.GameLogicSemantics method), 116 ActionPreconditionNode (class in textworld.logic.parser.GameLogicSemantics method), 116 AliasNode (class in textworld.logic.model), 114 all_applicable_actions() (textworld.logic.State method), 109 all_assignments() (textworld.logic.State method), 109 all_instantiations() (textworld.logic.Rule property), 108 all_propositions (textworld.logic.Action property), 108 all_propositions (textworld.generator.chaining.ChainingOptions attribute), 57 allowed_variables_numbering (textworld.generator.text_grammar.GrammarOptions attribute), 87 ambiguous_instructions	action(textworld.generator.game.ActionDependencyTree)	Endded (textworld.logic.Action property), 103
method), 116 ActionDependencyTree (class in textworld.generator.game), 61 ActionDependencyTreeElement (class in textworld.generator.game), 61 ActionDependencyTreeElement (class in textworld.generator.logger.GameLogger method), 60 ActionNode (class in textworld.logic.model), 113 actionPrecondition()     (textworld.logic.parser.GameLogicSemantics method), 116 ActionPreconditionNode (class in textworld.logic.parser.GameLogicSemantics method), 116 AliasNode (class in textworld.logic.model), 114 all_applicable_actionS() (textworld.logic.State method), 108 all_apsignments() (textworld.logic.State method), 109 all_predicates (textworld.logic.Rule property), 108 all_propositions (textworld.logic.Action property), 108 all_propositions (textworld.logic.Action property), 103 allowed_types (textworld.generator.chainingOptions attribute), 57 allowed_types (textworld.generator.chainingOptions attribute), 87 ambiguous_instructions	property), 62	au j (lexiworia.generator.game.Emityinjo auribute), 02
method), 116 ActionDependencyTree (class in textworld.generator.game), 61 ActionDependencyTreeElement (class in textworld.generator.game), 61 ActionNode (class in textworld.logic.model), 113 actionPrecondition() (textworld.logic.parser.GameLogicSemantics method), 116 ActionPreconditionNode (class in textworld.logic.generator.gameLogicSemantics method), 116 ActionPreconditionNode (class in textworld.logic.parser.GameLogicSemantics method), 116 AliasNode (class in textworld.logic.model), 114 all_applicable_actions() (textworld.logic.State method), 109 all_assignments() (textworld.logic.State method), 109 all_instantiations() (textworld.logic.Rule property), 108 all_predicates (textworld.logic.Rule property), 108 all_predicates (textworld.logic.Rule property), 108 all_predicates (textworld.logic.Action property), 103 allowed_types (textworld.generator.chaining.ChainingOptions attribute), 57 allowed_types (textworld.generator.text_grammar.GrammarOptions attribute), 87 ambiguous_instructions	<pre>action() (textworld.logic.parser.GameLogicSemantics</pre>	
textworld.generator.game), 61 ActionDependencyTreeElement (class in textworld.generator.logger.GameLogger method), 60 ActionNode (class in textworld.logic.model), 113 actionPrecondition() (textworld.logic.parser.GameLogicSemantics method), 116 ActionPreconditionNode (class in textworld.logic.model), 113 actions (textworld.logic.model), 113 actions (textworld.generator.chaining.Chain attribute), 56 actions (textworld.generator.game.Event attribute), 56 actions (textworld.generator.maker.WorldEntity method), 119 add_content() (textworld.generator.maker.GameMaker method), 119 add_fact() (textworld.generator.maker.GameMaker method), 81  Agent (class in textworld.generator.logger.GameLogger method), 60 Alias (class in textworld.logic.parser.GameLogicSemantics method), 116 AliasNode (class in textworld.logic.model), 114 all_applicable_actions() (textworld.logic.State method), 108 all_assignments() (textworld.logic.State method), 109 all_instantiations() (textworld.logic.State method), 109 all_predicates (textworld.logic.Rule property), 108 all_propositions (textworld.logic.Action property), 103 allowed_types (textworld.generator.chaining.ChainingOptions attribute), 57 allowed_variables_numbering (textworld.generator.text_grammar.GrammarOptions attribute), 87 ambiguous_instructions		
Agent (class in textworld.gym.core), 39  ActionDependencyTreeElement (class in textworld.generator.game), 61  ActionNode (class in textworld.logic.model), 113  actionPrecondition() (textworld.logic.parser.GameLogicSemantics method), 116  ActionPreconditionNode (class in textworld.logic.model), 113  actionSeconditionNode (class in textworld.logic.model), 113  actions (textworld.logic.model), 113  actions (textworld.generator.chaining.Chain attribute), 56  actions (textworld.generator.game.Event attribute), 56  add() (textworld.generator.maker.WorldEntity method), 119  add_content() (textworld.render.render.GraphItem method), 119  add_fact() (textworld.generator.maker.GameMaker method), 76  add_fact() (textworld.generator.maker.WorldEntity method), 81  Agent (class in textworld.generator.logger.GameLogger method), 60  Alias (class in textworld.logic.parser.GameLogicSemantics method), 116  AliasNode (class in textworld.logic.model), 114  all_applicable_actions() (textworld.logic.State method), 109  all_assignments() (textworld.logic.State method), 109  all_instantiations() (textworld.logic.Rule property), 108  all_predicates (textworld.logic.Rule property), 108  all_propositions (textworld.generator.chaining.ChainingOptions attribute), 57  allowed_variables_numbering	ActionDependencyTree (class in	
ActionDependencyTreeElement (class in textworld.generator.game), 61 ActionNode (class in textworld.logic.model), 113 actionPrecondition()		= :
ActionNode (class in textworld.logic.model), 113 actionPrecondition()	ActionDependencyTreeElement (class in	
actionPrecondition()		Alias (class in textworld.logic), 104
(textworld.logic.parser.GameLogicSemantics method), 116  ActionPreconditionNode (class in textworld.logic.model), 113  actions (textworld.logic.model), 113  actions (textworld.generator.chaining.Chain attribute), 56  actions (textworld.generator.game.Event attribute), 63  actions (textworld.generator.game.Event property), 64  add() (textworld.generator.maker.WorldEntity method), 109  add() (textworld.logic.TypeHierarchy method), 112  add_content() (textworld.render.render.GraphItem method), 119  add_fact() (textworld.generator.maker.GameMaker method), 76  add_fact() (textworld.generator.maker.WorldEntity method), 81  method), 116  AliasNode (class in textworld.logic.model), 114  all_applicable_actions() (textworld.logic.State method), 109  all_instantiations() (textworld.logic.Rule property), 108  all_predicates (textworld.logic.Rule property), 108  all_predicates (textworld.logic.Action property), 103  allowed_types (textworld.generator.chaining.ChainingOptions attribute), 57  allowed_variables_numbering (textworld.generator.text_grammar.GrammarOptions attribute), 87  ambiguous_instructions		alias() (textworld.logic.parser.GameLogicSemantics
method), 116  ActionPreconditionNode (class in textworld.logic.model), 113  actions (textworld.generator.chaining.Chain attribute), 56  actions (textworld.generator.game.Event attribute), 63 actions (textworld.generator.game.Event attribute), 64 add() (textworld.generator.maker.WorldEntity method), 112 add_content() (textworld.generator.maker.GameMaker method), 119 add_fact() (textworld.generator.maker.GameMaker method), 76 add_fact() (textworld.generator.maker.WorldEntity method), 81  AliasNode (class in textworld.logic.model), 114 all_applicable_actions() (textworld.logic.State method), 108 all_assignments() (textworld.logic.State method), 109 all_instantiations() (textworld.logic.Rule property), 108 all_predicates (textworld.logic.Rule property), 108 all_propositions (textworld.logic.Action property), 103 allowed_types (textworld.generator.chaining.ChainingOptions attribute), 57 allowed_variables_numbering (textworld.generator.text_grammar.GrammarOptions attribute), 87 ambiguous_instructions		method), 116
ActionPreconditionNode (class in textworld.logic.model), 113 actions (textworld.generator.chaining.Chain attribute), 56 actions (textworld.generator.game.Event attribute), 63 actions (textworld.generator.game.Event attribute), 63 actions (textworld.generator.game.Event property), 64 add() (textworld.generator.maker.WorldEntity method), 112 add_content() (textworld.render.render.GraphItem method), 119 add_fact() (textworld.generator.maker.GameMaker method), 76 add_fact() (textworld.generator.maker.WorldEntity method), 81  all_applicable_actions() (textworld.logic.State method), 109 all_instantiations() (textworld.logic.State method), 109 all_predicates (textworld.logic.Rule property), 108 all_propositions (textworld.logic.Action property), 108 all_propositions (textworld.logic.Action property), 108 all_predicates (textworld.logic.Rule property), 108 all_predicates (textworld.logic.Rule property), 108 all_predicates (textworld.logic.Rule property), 108 all_predicates (textworld.logic.Rule property), 108 all_propositions (textworld.logic.State method), 109 all_predicates (textworld.logic.Action property), 108 all_predicates (textworld.logic.Rule property), 108 all_propositions (textworld.logic.Rule property), 108 all_pr		AliasNode (class in textworld.logic.model), 114
actions (textworld.generator.chaining.Chain attribute), 56 actions (textworld.generator.game.Event attribute), 56 actions (textworld.generator.game.Event attribute), 63 actions (textworld.generator.game.Event property), 64 add() (textworld.generator.maker.WorldEntity method), 80 add() (textworld.logic.TypeHierarchy method), 112 add_content() (textworld.render.render.GraphItem method), 119 add_fact() (textworld.generator.maker.GameMaker method), 76 add_fact() (textworld.generator.maker.WorldEntity method), 81  method), 108 all_assignments() (textworld.logic.State method), 109 all_instantiations() (textworld.logic.Rule property), 108 all_propositions (textworld.logic.Action property), 103 allowed_types (textworld.generator.chaining.ChainingOptions attribute), 57 allowed_variables_numbering (textworld.generator.chaining.ChainingOptions attribute), 87 ambiguous_instructions		all_applicable_actions() (textworld.logic.State
actions (textworld.generator.chaining.Chain attribute), 56 actions (textworld.generator.game.Event attribute), 63 actions (textworld.generator.game.Event attribute), 63 actions (textworld.generator.game.Event property), 64 add() (textworld.generator.maker.WorldEntity method), 80 add() (textworld.logic.TypeHierarchy method), 112 add_content() (textworld.render.render.GraphItem method), 119 add_fact() (textworld.generator.maker.GameMaker method), 76 add_fact() (textworld.generator.maker.WorldEntity method), 81  all_instantiations() (textworld.logic.State method), 109 all_predicates (textworld.logic.Rule property), 108 all_propositions (textworld.logic.Action property), 103 allowed_types (textworld.generator.chaining.ChainingOptions attribute), 57 allowed_variables_numbering	`	method), 108
actions (textworld.generator.game.Event attribute), 63 actions (textworld.generator.game.Event property), 64 add() (textworld.generator.maker.WorldEntity method), 80 add() (textworld.logic.TypeHierarchy method), 112 add_content() (textworld.render.render.GraphItem method), 119 add_fact() (textworld.generator.maker.GameMaker method), 76 add_fact() (textworld.generator.maker.GameMaker method), 81 all_instantiations() (textworld.logic.State method), 109 all_predicates (textworld.logic.Rule property), 108 all_propositions (textworld.logic.Action property), 103 allowed_types (textworld.generator.chaining.ChainingOptions attribute), 57 allowed_variables_numbering (textworld.generator.text_grammar.GrammarOptions attribute), 87 ambiguous_instructions	${\tt actions}\ ({\it textworld.generator.chaining.Chain}\ {\it attribute}),$	
actions (textworld.generator.game.Event property), 64 add() (textworld.generator.maker.WorldEntity method), 80 add() (textworld.logic.TypeHierarchy method), 112 add_content() (textworld.render.render.GraphItem method), 119 add_fact() (textworld.generator.maker.GameMaker method), 76 add_fact() (textworld.generator.maker.WorldEntity method), 81  109 all_predicates (textworld.logic.Rule property), 108 all_propositions (textworld.logic.Action property), 103 allowed_types (textworld.generator.chaining.ChainingOptions attribute), 57 allowed_variables_numbering (textworld.generator.text_grammar.GrammarOptions attribute), 87 ambiguous_instructions	actions (textworld, generator, game, Event attribute), 63	<pre>all_instantiations() (textworld.logic.State method),</pre>
add() (textworld.logic.TypeHierarchy method), add() (textworld.logic.TypeHierarchy method), 112 add_content() (textworld.render.render.GraphItem method), 119 add_fact() (textworld.generator.maker.GameMaker method), 76 add_fact() (textworld.generator.maker.WorldEntity method), 81 all_predicates (textworld.logic.Rule property), 108 all_propositions (textworld.logic.Rule property), 108 all_propositions (textworld.logic.Action property), 103 allowed_types (textworld.generator.chaining.ChainingOptions attribute), 57 allowed_variables_numbering (textworld.generator.text_grammar.GrammarOptions attribute), 87 ambiguous_instructions		109
add() (textworld.logic.TypeHierarchy method), 112 add_content() (textworld.render.render.GraphItem     method), 119 add_fact() (textworld.generator.maker.GameMaker     method), 76 add_fact() (textworld.generator.maker.WorldEntity     method), 81  all_propositions (textworld.logic.Action property),  103 allowed_types (textworld.generator.chaining.ChainingOptions     attribute), 57 allowed_variables_numbering     (textworld.generator.text_grammar.GrammarOptions     attribute), 87 ambiguous_instructions		all_predicates (textworld.logic.Rule property), 108
add() (textworld.logic.TypeHierarchy method), 112 add_content() (textworld.render.render.GraphItem method), 119 add_fact() (textworld.generator.maker.GameMaker method), 76 add_fact() (textworld.generator.maker.WorldEntity method), 81  103 allowed_types (textworld.generator.chaining.ChainingOptions attribute), 57 allowed_variables_numbering (textworld.generator.text_grammar.GrammarOptions attribute), 87 ambiguous_instructions	•	all_propositions (textworld.logic.Action property),
add_content() (textworld.render.render.GraphItem method), 119  add_fact() (textworld.generator.maker.GameMaker method), 76  add_fact() (textworld.generator.maker.WorldEntity method), 81  allowed_types (textworld.generator.chaining.ChainingOptions attribute), 57  allowed_variables_numbering (textworld.generator.text_grammar.GrammarOptions attribute), 87  ambiguous_instructions		103
add_fact() (textworld.generator.maker.GameMaker method), 76 add_fact() (textworld.generator.maker.WorldEntity method), 81  allowed_variables_numbering (textworld.generator.text_grammar.GrammarOptions attribute), 87 ambiguous_instructions	add_content() (textworld.render.render.GraphItem	
method), 76 (textworld.generator.text_grammar.GrammarOptions add_fact() (textworld.generator.maker.WorldEntity method), 81 (textworld.generator.text_grammar.GrammarOptions attribute), 87 ambiguous_instructions		allowed_variables_numbering
add_fact() (textworld.generator.maker.WorldEntity method), 81 attribute), 87 ambiguous_instructions		$(textworld.generator.text\_grammar.GrammarOptions$
method), 81 ambiguous_instructions		attribute), 87
		ambiguous_instructions
	<pre>add_fact() (textworld.generator.world.World method),</pre>	$(textworld.generator.text\_grammar.GrammarOptions$

attribute), 87	chunk() (in module textworld.utils), 123
ancestors (textworld.logic.Type property), 111	CLASS_HOLDER (textworld.generator.vtypes.VariableTypeTree
apply() (textworld.logic.State method), 109	attribute), 61
apply_on_copy() (textworld.logic.State method), 109	clean_replace_objs() (in module
are_facts() (textworld.logic.State method), 109	textworld.generator.text_generation), 83
arguments (textworld.logic.model.PropositionNode at-	close() (textworld.core.Environment method), 34
tribute), 115	close() (textworld.core.Wrapper method), 36
arguments (textworld.logic.Proposition attribute), 106	close() (textworld.envs.glulx.git_glulx.GitGlulxEnv
<pre>assign_description_to_object() (in module</pre>	method), 48 close() (textworld.envs.wrappers.viewer.HtmlViewer
assign_description_to_quest() (in module	method), 49
textworld.generator.text_generation), 83	close() (textworld.envs.zmachine.jericho.JerichoEnv
assign_description_to_room() (in module	method), 50
textworld.generator.text_generation), 83	close() (textworld.gym.envs.textworld_batch.TextworldBatchGymEnv
assign_name_to_object() (in module	method), 42
textworld.generator.text_generation), 83	closure() (textworld.logic.TypeHierarchy method), 112
<pre>assign_new_matching_names() (in module</pre>	code (textworld.logic.model.Inform7CodeNode at-
$textworld.generator.text\_generation), 83$	tribute), 114
П	collect() (textworld.generator.logger.GameLogger
В	method), 60
${\tt backward} (\textit{textworld}. \textit{generator}. \textit{chaining}. \textit{Chaining} \textit{Options}$	command (textworld.logic.model.Inform7CommandNode
attribute), 56	attribute), 114
basics (textworld.core.EnvInfos property), 32	command_templates (textworld.core.EnvInfos at-
blend_descriptions (textworld.generator.text_grammar.	Grammar (ปักษณะ), 32 command_templates (textworld.generator.game.Game
attribute), 87 blend_instructions (textworld.generator.text_grammar.	
attribute), 87	commands (textworld.generator.game.Event attribute), 63
breadth (textworld.generator.chaining.ChainNode	commands (textworld.generator.game.Event property), 64
attribute), 56	commands (textworld.generator.game.Quest attribute), 70
build() (textworld.generator.maker.GameMaker	commands (textworld.generator.game.Quest property), 70
method), 76	commands (textworld.logic.model.Inform7CommandsNode
0	attribute), 114
C	compile() (textworld.generator.maker.GameMaker
Chain (class in textworld.generator.chaining), 56	method), 77
chaining (textworld.generator.game.GameOptions at-	compile_game() (in module textworld.generator), 55 compile_inform7_game() (in module
tribute), 68	textworld.generator.inform7.world2inform7),
ChainingOptions (class in	99
textworld.generator.chaining), 56	completable (textworld.generator.game.QuestProgression
ChainNode (class in textworld.generator.chaining), 56 change_grammar() (textworld.generator.game.Game	property), 71
change_grammar() (textworld.generator.game.Game method), 65	completed (textworld.generator.game.GameProgression
Char (class in textworld.gym.spaces.text_spaces), 43	property), 69
check() (textworld.generator.text_grammar.Grammar	${\tt completed}\ (\textit{textworld.generator.game.QuestProgression}$
J. D. 04	property), 71
<pre>check_action() (textworld.generator.chaining.Chaining()</pre>	compress_policy() (textworld.generator.game.EventProgression
method), 57	memoa), 04
<pre>check_modules() (in module textworld.utils), 123</pre>	concat_images() (in module textworld.render.render),
<pre>check_new_variable()</pre>	119 condition (textuewed consector come Front attribute)
(textworld.generator.chaining.ChainingOptions	condition (textworld.generator.game.Event attribute),
method), 57	condition(textworld.logic.model.ActionPreconditionNode
CHEST (textworld.generator.vtypes.VariableTypeTree at-	attribute), 113
tribute), 61 child_types (textworld.logic.Type property), 111	condition(textworld.logic.model.RulePreconditionNode
CHILL CADES DESIMONO 1051C LADE DIODELLA LL	Condition (lexiworid.logic.model.Kuler reconditionNode
children (textworld.logic.Type property), 111	attribute), 115

connect() (in module textworld.generator.world), 75 connect() (textworld.generator.maker.GameMaker method), 77	class method), 88 define_inform7_kinds()     (textworld.generator.inform7.world2inform7.Inform7Game
constraints (textworld.logic.model.ConstraintsNode attribute), 114	method), 98 definite (textworld.generator.game.EntityInfo at-
<pre>constraints() (textworld.logic.parser.GameLogicSeman</pre>	
method), 116	definition (textworld.logic.model.Inform7TypeNode
ConstraintsNode (class in textworld.logic.model), 114	attribute), 114
copy() (textworld.core.EnvInfos method), 32	DependencyTree (class in
copy() (textworld.core.Environment method), 35	textworld.generator.dependency_tree), 58
copy() (textworld.core.GameState method), 35	DependencyTreeElement (class in
copy() (textworld.core.Wrapper method), 36	textworld.generator.dependency_tree), 59
copy() (textworld.envs.tw.TextWorldEnv method), 47	depends_on() (textworld.generator.dependency_tree.DependencyTreeElen
copy() (textworld.envs.wrappers.filter.Filter method), 50	method), 59
copy() (textworld.envs.zmachine.jericho.JerichoEnv	depends_on() (textworld.generator.game.ActionDependencyTreeElement
method), 50	method), 62
copy() (textworld.generator.chaining.ChainingOptions	depth (textworld.generator.chaining.ChainNode at-
method), 58	tribute), 56
copy() (textworld.generator.dependency_tree.Dependency	
method), 58	desc (textworld.generator.game.Quest attribute), 69
copy() (textworld.generator.game.ActionDependencyTree	
method), 61	descendants() (textworld.generator.vtypes.VariableTypeTree
copy() (textworld.generator.game.Event method), 63	method), 60
copy() (textworld.generator.game.EventProgression	describe_event() (in module
method), 64	textworld.generator.text_generation), 83
copy() (textworld.generator.game.Game method), 65	description (textworld.core.EnvInfos attribute), 32
copy() (textworld.generator.game.GameOptions	deserialize() (textworld.generator.data.KnowledgeBase
method), 68	class method), 88
	deserialize() (textworld.generator.game.EntityInfo
copy() (textworld.generator.game.GameProgression method), 68	class method), 62
copy() (textworld.generator.game.Quest method), 70	
	deserialize() (textworld.generator.game.Event class
copy() (textworld.generator.game.QuestProgression	method), 63 deserialize() (textworld.generator.game.Game class
method), 71	
copy() (textworld.generator.text_grammar.GrammarOptio	ns method), 65 deserialize() (textworld.generator.game.Quest class
method), 86	
copy() (textworld.logic.State method), 110	method), 70
CouldNotCompileGameError, 98 count() (textworld.generator.vtypes.VariableTypeTree	deserialize() (textworld.generator.text_grammar.GrammarOptions
	class method), 86
method), 60	deserialize() (textworld.generator.vtypes.VariableType
CountOrderedDict (class in	class method), 60
textworld.generator.text_generation), 83	deserialize() (textworld.generator.vtypes.VariableTypeTree
create() (textworld.generator.world.WorldEntity class	class method), 60
method), 74	deserialize() (textworld.generator.world.World class
create_data_files() (in module	method), 72
textworld.generator.data), 88	deserialize() (textworld.logic.Action class method),
create_map() (in module	103
textworld.generator.graph_networks), 75	deserialize() (textworld.logic.GameLogic class
create_small_map() (in module	method), 104
textworld.generator.graph_networks), 75	deserialize() (textworld.logic.Placeholder class
create_variables(textworld.generator.chaining.Chainin	
attribute), 57	deserialize() (textworld.logic.Predicate class
D	method), 105
	deserialize() (textworld.logic.Proposition class
default() (textworld generator data Knowledge Rase	method), 106

deserialize() (textworld.logic.Rule class method), 107	expand_clean_replace() (in module
deserialize() (textworld.logic.State class method),	textworld.generator.text_generation), 83
110	extras (textworld.core.EnvInfos attribute), 32
deserialize() (textworld.logic.Variable class method), 113	extremes() (in module textworld.generator.graph_networks), 75
<pre>detect_action() (textworld.generator.inform7.world2inf</pre>	
direction() (in module	facts (textworld.core.EnvInfos attribute), 32
$textworld.generator.graph\_networks), 75$	facts (textworld.generator.maker.GameMaker prop-
${\tt directions\_names}  (\textit{textworld.generator.game.Game}$	erty), 80
property), 66	facts (textworld.generator.maker.WorldEntity property),
display_command_during_render	81
(textworld.core.Environment property), 35	facts (textworld.generator.maker.WorldPath property),
display_command_during_render	82
(textworld.core.Wrapper property), 36	facts (textworld.generator.world.World property), 73
display_stats() (textworld.generator.logger.GameLogg	
method), 60	facts_with_signature() (textworld.logic.State
document() (textworld.logic.parser.GameLogicSemantics method), 116	method), 110
DocumentNode (class in textworld.logic.model), 114	fail_events (textworld.generator.game.Quest at-
done (textworld.generator.game.EventProgression prop-	tribute), 69 fail_events (textworld.generator.game.Quest prop-
erty), 65	erty), 71
done (textworld.generator.game.GameProgression prop-	fail_facts (textworld.core.EnvInfos attribute), 32
erty), 69	failed (textworld.generator.game.GameProgression
done (textworld.generator.game.QuestProgression prop-	property), 69
erty), 71	failed (textworld.generator.game.QuestProgression
${\tt door}\ (\textit{textworld.generator.maker.WorldPath property}),82$	property), 71
Г	FailedConstraintsError, 76
E	feedback (textworld.core.EnvInfos attribute), 32
east (textworld.generator.maker.WorldRoom attribute), 82	file_ext (textworld.generator.game.GameOptions at- tribute), 67
$\verb"empty" (textworld.generator.dependency\_tree.Dependency Tree.Dependency Tre$	
property), 59	<pre>filter_unknown() (textworld.gym.spaces.text_spaces.Char</pre>
encode() (textworld.render.serve.ServerSentEvent	method), 44
method), 121	find_by_name() (textworld.generator.maker.GameMaker
encode_seeds() (in module textworld.utils), 123	method), 77
entities (textworld.core.EnvInfos attribute), 32	find_free_port() (in module textworld.render.serve),  121
entities (textworld.generator.world.World property), 73	121
entity_names (textworld.generator.game.Game prop-	find chiect by id()
	find_object_by_id()  (textworld generator world World method)
	$(textworld. generator. world. World \qquad method),$
erty), 66	(textworld.generator.world.World method), 72
erty), 66 EntityInfo (class in textworld.generator.game), 62	(textworld.generator.world.World method), 72 find_path() (textworld.generator.maker.GameMaker
erty), 66	(textworld.generator.world.World method), 72 find_path() (textworld.generator.maker.GameMaker method), 77
erty), 66 EntityInfo (class in textworld.generator.game), 62 EnvInfoMissingError, 31	(textworld.generator.world.World method), 72 find_path() (textworld.generator.maker.GameMaker method), 77
erty), 66 EntityInfo (class in textworld.generator.game), 62 EnvInfoMissingError, 31 EnvInfos (class in textworld.core), 32	<pre>(textworld.generator.world.World method), 72 find_path() (textworld.generator.maker.GameMaker</pre>
erty), 66 EntityInfo (class in textworld.generator.game), 62 EnvInfoMissingError, 31 EnvInfos (class in textworld.core), 32 Environment (class in textworld.core), 34	<pre>(textworld.generator.world.World method), 72 find_path() (textworld.generator.maker.GameMaker</pre>
erty), 66 EntityInfo (class in textworld.generator.game), 62 EnvInfoMissingError, 31 EnvInfos (class in textworld.core), 32 Environment (class in textworld.core), 34 Event (class in textworld.generator.game), 63 event (textworld.logic.model.Inform7CommandNode attribute), 114	(textworld.generator.world.World method), 72  find_path() (textworld.generator.maker.GameMaker method), 77  find_room_by_id() (textworld.generator.world.World method), 72  findall() (textworld.generator.maker.GameMaker method), 77  finish() (textworld.core.Agent method), 31
erty), 66 EntityInfo (class in textworld.generator.game), 62 EnvInfoMissingError, 31 EnvInfos (class in textworld.core), 32 Environment (class in textworld.core), 34 Event (class in textworld.generator.game), 63 event (textworld.logic.model.Inform7CommandNode attribute), 114 EventProgression (class in textworld.generator.game),	(textworld.generator.world.World method), 72  find_path() (textworld.generator.maker.GameMaker method), 77  find_room_by_id() (textworld.generator.world.World method), 72  findall() (textworld.generator.maker.GameMaker method), 77  finish() (textworld.core.Agent method), 31  fix_determinant() (in module
erty), 66 EntityInfo (class in textworld.generator.game), 62 EnvInfoMissingError, 31 EnvInfos (class in textworld.core), 32 Environment (class in textworld.core), 34 Event (class in textworld.generator.game), 63 event (textworld.logic.model.Inform7CommandNode attribute), 114 EventProgression (class in textworld.generator.game), 64	(textworld.generator.world.World method), 72  find_path() (textworld.generator.maker.GameMaker method), 77  find_room_by_id() (textworld.generator.world.World method), 72  findall() (textworld.generator.maker.GameMaker method), 77  finish() (textworld.core.Agent method), 31  fix_determinant() (in module textworld.generator.text_grammar), 87
erty), 66 EntityInfo (class in textworld.generator.game), 62 EnvInfoMissingError, 31 EnvInfos (class in textworld.core), 32 Environment (class in textworld.core), 34 Event (class in textworld.generator.game), 63 event (textworld.logic.model.Inform7CommandNode attribute), 114 EventProgression (class in textworld.generator.game), 64 ExitAlreadyUsedError, 76	<pre>(textworld.generator.world.World method),</pre>
erty), 66 EntityInfo (class in textworld.generator.game), 62 EnvInfoMissingError, 31 EnvInfos (class in textworld.core), 32 Environment (class in textworld.core), 34 Event (class in textworld.generator.game), 63 event (textworld.logic.model.Inform7CommandNode attribute), 114 EventProgression (class in textworld.generator.game), 64 ExitAlreadyUsedError, 76 expand() (textworld.generator.text_grammar.Grammar	(textworld.generator.world.World method), 72  find_path() (textworld.generator.maker.GameMaker method), 77  find_room_by_id() (textworld.generator.world.World method), 72  findall() (textworld.generator.maker.GameMaker method), 77  finish() (textworld.core.Agent method), 31  fix_determinant() (in module textworld.generator.text_grammar), 87  fixed_mapping (textworld.generator.chaining.ChainingOptions attribute), 57
erty), 66 EntityInfo (class in textworld.generator.game), 62 EnvInfoMissingError, 31 EnvInfos (class in textworld.core), 32 Environment (class in textworld.core), 34 Event (class in textworld.generator.game), 63 event (textworld.logic.model.Inform7CommandNode attribute), 114 EventProgression (class in textworld.generator.game), 64 ExitAlreadyUsedError, 76	<pre>(textworld.generator.world.World method),</pre>

$\verb flatten()  (textworld.generator.game.ActionDependency Table 1)    (textworld.generator.game.ActionDependency Table 2)    (textworld.game.ActionDependency Table 2)    (textworld.game.actionDependenc$	ree method), 98
method), 61	<pre>gen_source_for_objects()</pre>
force_recompile(textworld.generator.game.GameOptionattribute), 67	as (textworld.generator.inform7.world2inform7.Inform7Game method), 98
<pre>format_command() (textworld.logic.Action method),</pre>	<pre>gen_source_for_rooms()</pre>
103	(textworld.generator.inform7.world2inform7.Inform7Game
<pre>from_facts() (textworld.generator.world.World class</pre>	method), 98
method), 72	<pre>generate_distractors()</pre>
<pre>from_map() (textworld.generator.world.World class     method), 72</pre>	(textworld.generator.maker.GameMaker method), 77
G	<pre>generate_inform7_source() (in module</pre>
g_rng (in module textworld.utils), 124	99
Game (class in textworld.generator.game), 65	<pre>generate_instruction()</pre>
game (textworld.core.EnvInfos attribute), 32	textworld.generator.text_generation), 83
game_running(textworld.envs.glulx.git_glulx.GitGlulxEnv property), 48	<pre>generate_name() (textworld.generator.text_grammar.Grammar</pre>
game_running(textworld.envs.zmachine.jericho.JerichoEn	generate_random_quests()
property), 51	(textworld.generator.maker.GameMaker
GameLogger (class in textworld.generator.logger), 60	method), 77
GameLogic (class in textworld.logic), 104	<pre>generate_text_from_grammar() (in module</pre>
GameLogicBuffer (class in textworld.logic.parser), 116	textworld.generator.text_generation), 84
GameLogicModelBuilderSemantics (class in	GenerationWarning, 55
textworld.logic.model), 114	<pre>get() (textworld.logic.TypeHierarchy method), 112</pre>
GameLogicParser (class in textworld.logic.parser), 116	<pre>get_action_chains()</pre>
GameLogicSemantics (class in textworld.logic.parser),	<pre>textworld.generator.text_generation), 84 get_all_adjective_for_type()</pre>
	(textworld.generator.text_grammar.Grammar
GameMaker (class in textworld.generator.maker), 76	method), 85
GameNotRunningError, 31 GameOptions (class in textworld.generator.game), 66	<pre>get_all_expansions_for_tag()</pre>
GameProgression (class in textworld.generator.game),	(textworld.generator.text_grammar.Grammar method), 85
68	get_all_expansions_for_type()
GameState (class in textworld.core), 35	(textworld.generator.text_grammar.Grammar
gen() (textworld.render.serve.Server method), 120 gen_commands_from_actions() (in module	method), 85
<pre>gen_commands_from_actions() (in module</pre>	<pre>get_all_names_for_type()</pre>
gen_commands_from_actions()	(textworld.generator.text_grammar.Grammar
(textworld.generator.inform7.world2inform7.Info	
method), 98	get_all_nouns_for_type()
gen_layout() (in module	(textworld.generator.text_grammar.Grammar
textworld.generator.graph_networks), 75	method), 85
gen_source() (textworld.generator.inform7.world2inform	
method), 98	(textworld.generator.world.World method),
<pre>gen_source_for_attribute()</pre>	72
(textworld.generator.inform7.world2inform7.Informethod), 98	method), 60
<pre>gen_source_for_attributes()</pre>	<pre>get_attributes() (textworld.generator.world.WorldEntity</pre>
(textworld.generator.inform 7. world 2 inform 7. Infor	rm7Game method), 74 get_chains() (in module textworld.generator.chaining),
method), 98	58
gen_source_for_conditions()  (tartworld generator inform 7 world?inform 7 Info	rgercidescription() (textworld.generator.vtypes.VariableTypeTree
method), 98	method), 60
<pre>gen_source_for_map()</pre>	<pre>get_entities_per_type()</pre>
(textworld.generator.inform7.world2inform7.Infor	rm7Game (textworld.generator.world.World method),

72	has_subtype_named() (textworld.logic.Type method),
<pre>get_facts_in_scope()</pre>	111
(textworld.generator.world.World method), 72	has_supertype_named() (textworld.logic.Type method), 111
<pre>get_failing_constraints() (in module</pre>	has_tag() (textworld.generator.text_grammar.Grammar method), 86
<pre>get_html_template()</pre>	has_variable() (textworld.logic.State method), 110
textworld.render.serve), 121	HtmlViewer (class in textworld.envs.wrappers.viewer),
<pre>get_human_readable_action()</pre>	49
	r HumanAgent (class in textworld.agents.human), 53
get_human_readable_fact()	
	rm7Game 11
method), 99	THA GENTWorld generator game. EntityInfo attribute), 62
<pre>get_matching() (textworld.utils.RegexDict method),</pre>	id (textworld.generator.maker.WorldEntity property), 81
123	id (textworld.generator.world.WorldEntity property), 74
	import_graph() (textworld.generator.maker.GameMaker
get_max_depth() (textworld.render.render.GraphItem	method), 78
method), 119	include_adj (textworld.generator.text_grammar.GrammarOptions
get_new() (in module textworld.generator.vtypes), 61	attribute), 87
<pre>get_objects_in_inventory()</pre>	indefinite (textworld.generator.game.EntityInfo
(textworld.generator.world.World method),	attribute), 62
72	$\verb independent_chains   (\textit{textworld}. \textit{generator}. \textit{chaining}. \textit{Chaining} Options  $
get_path() (in module	attribute), 57
textworld.generator.graph_networks), 75	index() (textworld.render.serve.Server method), 120
get_random_expansion()	<pre>inform7() (textworld.logic.parser.GameLogicSemantics</pre>
(textworld.generator.text_grammar.Grammar	method), 116
method), 86	$\verb"inform7Code"()" (\textit{textworld.logic.parser.GameLogicSemantics}"$
get_reverse_action()	method), 116
(textworld.generator.data. Knowledge Base	Inform7CodeNode (class in textworld.logic.model), 114
method), 88	Inform7Command (class in textworld.logic), 104
<pre>get_rules() (textworld.generator.chaining.ChainingOpti</pre>	<sup>0</sup> #form7Command() (textworld.logic.parser.GameLogicSemantics method), 116
get_visible_objects_in()	Inform7CommandNode (class in textworld.logic.model),
(textworld.generator.world.World method),	114
73	inform7Commands() (textworld.logic.parser.GameLogicSemantics
<pre>get_vocabulary() (textworld.generator.text_grammar.Generator), 86</pre>	rammar method), 116
<pre>get_webdriver() (in module textworld.render.render),</pre>	Inform7CommandsNode (class in textworld.logic.model),
119	114
GitGlulxEnv (class in textworld.envs.glulx.git_glulx), 48	Inform7Game (class in
Grammar (class in textworld.generator.text_grammar), 84	textworld.generator.inform7.world2inform7),
grammar (textworld.generator.game.GameOptions	98
attribute), 68	Inform7Logic (class in textworld.logic), 104
GrammarOptions (class in	Inform7Node (class in textworld.logic.model), 114
textworld.generator.text_grammar), 86	inform7Part() (textworld.logic.parser.GameLogicSemantics method), 116
<pre>graph2state() (in module textworld.generator.world),</pre>	Inform7Predicate (class in textworld.logic), 104
75	<pre>inform7Predicate() (textworld.logic.parser.GameLogicSemantics</pre>
GraphItem (class in textworld.render.render), 119	method), 116
GraphRoom (class in textworld.render.render), 119	Inform7PredicateNode (class in
	textworld.logic.model), 114
H	inform7Predicates()
has_property() (textworld.generator.maker.WorldEntity	(textworld.logic.parser.GameLogicSemantics
method). 81	method) 116

Inform7PredicatesNo textworld.logic	,	elass in	Knowled	lgeBase (class in textwo	orld.generator.da	ata), 88
Inform7Type (class in to		104	1			
inform7Tupe()(texture	ziworia.iogic), rld logic parser	104 Camal agiaSama	nt <del>i</del> os		* 0	
inform7Type()(textwo	ra.iogic.parser.	.GameLogicSema	""Fast_ac	ction (textworld.core.E	nvInfos attribute	?), 33
method), 116	111			ommand (textworld.core.)		
Inform7TypeNode (classinfos (textworld.general			leaves_	_elements( <i>textworld.ge</i> <i>property</i> ), 59	enerator.depende	ency_tree.DependencyTree
infos (textworld.render.	render.GraphIte	em property), 119	leaves		erator dependenc	cy_tree.DependencyTree
<pre>infos_to_request (t      erty), 40</pre>	extworld.gym.co	ore.Agent prop-		property), 59		
initial_state (textwo	rld generator cl	naining Chain at-		world.logic.model.Alias		
tribute), 56	ria.generaior.en	iaining.Chain ai	ins (tex	tworld.logic.model.Reve	erseKuteNoae a	ttribute),
instantiate() (textwo	rld logic Prodic	rate method) 105		115		
instantiate() (textwo			list_to	_string()	(in	module
intermediate_reward				textworld.generator.tex	-	
tribute), 33			listen	() (textworld.render.ser 120	ve.Server static	method),
inventory (textworld.co			load()	(textworld.core.Environ	ment method), 3	5
inventory (textworld.g	enerator.maker	.GameMaker at-	load()	(textworld.core.Wrapper	r method), 36	
tribute), 76			load()	(textworld.envs.glu	ılx.git_glulx.Git(	GlulxEnv
<pre>inverse() (textworld.lo</pre>	~			method), 48		
<pre>inverse() (textworld.lo</pre>			load()	(textworld.envs.tw.TextW	VorldEnv method	<i>l</i> ), 47
$is_a()$ (textworld.logic.			load()	(textworld.envs.zma	chine.jericho.Je	richoEnv
<pre>is_applicable() (text</pre>	vorld.logic.Stat	te method), 110		method), 50		
is_constant() (textwo) method), 60				(textworld.generator.da method), 88	_	
<pre>is_descendant_of()(     method), 60</pre>	textworld.gener	ator.vtypes.Varial	ole <b>TonaTi</b> ye	(textworld.generator.gai	me.Game class	method),
<pre>is_distinct_from()(</pre>	textworld.gener	ator.dependency	tree Danen	denenTreeElement <sub>tor to</sub>	ager Gamel ogg	er static
method), 59	0	1 3-	1000()	method), 60	gger.GameLogg	er static
<pre>is_distinct_from()(</pre>	textworld.gener	ator.game.Actionl	Depandenc	yTreeElement generator	vtvnes Variable	Tyne Tree
method), 62	0	0	Loau()	class method), 60	.viypes.variabie	Туреттее
<pre>is_fact() (textworld.lo</pre>	gic.State metho	d), 110	()bsol	(textworld.logic.GameL	ogic class metho	nd) 104
	-	rator.game.Quest		:ate() (in module textw		
method), 70		0 2		:ate_from_game_stat		module
is_seq() (in module text	tworld.generato	r.text generation)		textworld.render.rende		тошие
84		_8		on (textworld.core.EnvIn		3
is_sequence_applica <i>method</i> ), 110	ble() (text	tworld.logic.State	logic	(textworld.generator.cl		
is_subtype_of() (text	world logic Typ	a mathad) 111		attribute), 57		
is_supertype_of() (text			logic	(textworld.generator.cl	haining.Chainin	gOptions
	_	-		property), 58	u \ 00	
method), 64	(iexiworia.gene	rator.game.Event	lost (te.	xtworld.core.EnvInfos a	ttribute), 33	
<pre>is_winning()</pre>	(textworld.gene	rator.game.Quest	M			
method), 70			main()	(in module textworld.log	vic narsar) 117	
Ī				ume() (in module textwo	_	55
J				ume_with() (in module	-	
JerichoEnv (class in t	extworld.envs.zi	machine.jericho),		rammar() (in module te:	_	
50	_	3	_	up() (in module textwor		
				ip() (in module textwor iest() (in module textw		
K				nall_map() (in module nall_map() (in module		
kb (textworld.generator.g	ame.GameOnti	ons attribute) 68		emp_directory() (in	_	
kb (textworld.generator.g	_		marc_tt	123	mounte textwo	·
kind (textworld.logic.mo	_		maka 1.10	orld() (in module textw	orld generator)	56
114		, , , , , , , , , , , , , , , , , , , ,	marc_w(	ri ia () (in mounte textin	ona.generaior),	, 50

<pre>make_world_with() (in module textworld.generator),</pre>	textworld.generator.chaining,56
56	textworld.generator.data,88
mark_doors() (in module	textworld.generator.dependency_tree,58
$textworld.generator.graph\_networks), 75$	textworld.generator.game,61
match() (textworld.logic.Predicate method), 105	textworld.generator.graph_networks,75
match() (textworld.logic.Rule method), 107	textworld.generator.inform7,98
max_breadth(textworld.generator.chaining.ChainingOptiattribute), 57	ons textworld.generator.inform7.world2inform7
<pre>max_depth(textworld.generator.chaining.ChainingOption</pre>	s textworld.generator.logger,60
attribute), 57	textworld.generator.maker,76
$\verb max_length  (\textit{textworld}. \textit{generator}. \textit{chaining}. \textit{Chaining} \textit{Optio}$	<pre>ns textworld.generator.text_generation,83</pre>
attribute), 56	textworld.generator.text_grammar,84
max_score (textworld.core.EnvInfos attribute), 33	textworld.generator.vtypes,60
<pre>max_score (textworld.generator.game.Game property),</pre>	textworld.generator.world,72
66	textworld.gym, 37
<pre>maybe_mkdir() (in module textworld.utils), 123</pre>	textworld.gym.envs, 40
MergeAction (class in	${\tt textworld.gym.envs.textworld}, 40$
textworld.generator.text_generation), 83	textworld.gym.envs.textworld_batch,41
$\verb"metadata" (textworld.gym.envs.textworld.TextworldGymEn") and the state of the s$	v textworld.gym.envs.utils,43
attribute), 41	textworld.gym.spaces,43
$metadata(textworld.gym.envs.textworld\_batch.TextworldBatch.Tex$	BatchGwextToworld.gym.spaces.text_spaces,43
attribute), 43	textworld.gym.utils,37
$\verb min_breadth  (textworld.generator.chaining.ChainingOptical and all all all all all all all all all al$	ons textworld.logic, 103
attribute), 57	textworld.logic.model, 113
min_depth(textworld.generator.chaining.ChainingOption	s textworld.logic.parser, 116
attribute), 57	textworld.render, 119
min_length(textworld.generator.chaining.ChainingOptio	ns textworld.render.render,119
attribute), 56	textworld.render.serve, 120
MissingPlayerError, 76	textworld.utils, 123
MissingTextGrammar, 84	move() (textworld.generator.maker.GameMaker
ModelBase (class in textworld.logic.model), 114	method), 78
module	moves (textworld.core.EnvInfos attribute), 33
textworld, 31	<pre>multi_ancestors() (textworld.logic.TypeHierarchy</pre>
textworld.agents, 53	method), 112
textworld.agents.human,53	<pre>multi_closure() (textworld.logic.TypeHierarchy</pre>
textworld.agents.random, 53	method), 112
textworld.agents.simple,54	<pre>multi_descendants() (textworld.logic.TypeHierarchy</pre>
textworld.agents.walkthrough,54	method), 113
textworld.challenges.coin_collector,23	<pre>multi_subtypes() (textworld.logic.TypeHierarchy</pre>
textworld.challenges.cooking,25	method), 113
textworld.challenges.simple,22	<pre>multi_supertypes() (textworld.logic.TypeHierarchy</pre>
textworld.challenges.treasure_hunter,28	method), 113
textworld.core, 31	
textworld.envs,47	N
textworld.envs.glulx,48	NaiveAgent (class in textworld.agents.random), 53
textworld.envs.glulx.git_glulx,48	NaiveAgent (class in textworld.agents.simple), 54
textworld.envs.tw,47	name (textworld.generator.game.EntityInfo attribute), 63
textworld.envs.wrappers,49	name (textworld.generator.maker.WorldEntity property),
textworld.envs.wrappers.filter,49	81
textworld.envs.wrappers.recorder, 49	name (textworld.generator.world.WorldEntity attribute),
textworld.envs.wrappers.viewer, 49	74
textworld.envs.zmachine,50	name (textworld.generator.world.WorldObject attribute),
textworld.envs.zmachine.jericho,50	74
textworld.generator,55	

name (textworld.generator.world.WorldRoom attribute),	tribute), 76
name (textworld.logic.model.ActionNode attribute), 113	0
name (textworld.logic.model.PlaceholderNode attribute),	
115	obj_list_to_prop_string() (in module
( 111	textworld.generator.text_generation), 84
name (textworld.logic.model.PredicateNode attribute),	objective (textworld.core.EnvInfos attribute), 33
	objective (textworld.generator.game.Game property),
name (textworld.logic.model.PropositionNode attribute),	66
115	objects (textworld.generator.world.World property), 73
name (textworld.logic.model.RuleNode attribute), 115	objects_names (textworld.generator.game.Game prop-
name (textworld.logic.model.SignatureNode attribute),	erty), 66
115	objects_names_and_types
name (textworld.logic.model.TypeNode attribute), 115	(textworld.generator.game.Game property),
name (textworld.logic.model.VariableNode attribute), 116	66
name (textworld.logic.Placeholder attribute), 105	objects_types (textworld.generator.game.Game prop-
name (textworld.logic.Proposition attribute), 107	erty), 66
name (textworld.logic.Signature attribute), 108	$\verb"only_last_action" (\textit{textworld}. \textit{generator}. \textit{text\_grammar}. \textit{Grammar} Options$
name (textworld.logic.Variable attribute), 113	attribute), 87
name() (textworld.logic.parser.GameLogicSemantics	<pre>onlyAction() (textworld.logic.parser.GameLogicSemantics</pre>
method), 116	method), 116
names (textworld.logic.Predicate property), 106	onlyPlaceholder() (textworld.logic.parser.GameLogicSemantics
names (textworld.logic.Proposition property), 107	method), 116
<pre>names_to_exclude(textworld.generator.text_grammar.Gr</pre>	anny Podivate() (textworld.logic.parser.GameLogicSemantics
attribute), 87	method), 116
nb_objects (textworld.generator.game.GameOptions at-	onlyProposition() (textworld.logic.parser.GameLogicSemantics
tribute), 67	method), 116
	PGAT-yRule() (textworld.logic.parser.GameLogicSemantics
attribute), 67	method), 116
nb_rooms (textworld.generator.game.GameOptions at-	onlySignature() (textworld.logic.parser.GameLogicSemantics
tribute), 66	method), 116
new() (textworld.generator.maker.GameMaker method),	onlyVariable() (textworld.logic.parser.GameLogicSemantics
78	method), 116
new_door() (textworld.generator.maker.GameMaker	optional (textworld.generator.game.Quest attribute), 70
method), 78	optional (lexiworth.generalor.game.Quest aurionie), 70
new_event_using_commands()	P
(textworld.generator.maker.GameMaker	•
method), 78	parameters (textworld.logic.model.PredicateNode at-
new_fact() (textworld.generator.maker.GameMaker	tribute), 115
method), 79	parent (textworld.generator.chaining.ChainNode at-
new_quest_using_commands()	tribute), 56
	<pre>parent_types (textworld.logic.Type property), 111</pre>
(textworld.generator.maker.GameMaker	<pre>parse() (textworld.generator.vtypes.VariableType class</pre>
method), 79	method), 60
new_room() (textworld.generator.maker.GameMaker	parse() (textworld.logic.Action class method), 103
method), 79	parse() (textworld.logic.GameLogic class method), 104
next() (textworld.utils.RandomGenerator method), 123	parse() (textworld.logic.Placeholder class method), 105
nodes (textworld.generator.chaining.Chain attribute), 56	parse() (textworld.logic.Predicate class method), 106
NoFreeExitError, 72	parse() (textworld.logic.Proposition class method), 106
normalize_rule() (textworld.logic.GameLogic	parse() (textworld.logic.Rule class method), 107
method), 104	parse() (textworld.logic.Signature class method), 108
north (textworld.generator.maker.WorldRoom attribute),	parse() (textworld.logic.Variable class method), 113
82	parse_variable_types() (in module
NotEnoughNounsError, 60	textworld.generator.vtypes), 61
noun (textworld.generator.game.EntityInfo attribute), 63	parts (textworld.logic.model.Inform7Node attribute),
nowhere (textworld.generator.maker.GameMaker at-	11/

114

parts (textworld.logic.model.TypeNode attribute), 116	method), 117
path (textworld.generator.game.GameOptions attribute),	PredicatesNode (class in textworld.logic.model), 115
67	${\tt predName()}\ (textworld.logic.parser.GameLogicSemantics$
paths (textworld.generator.maker.GameMaker at-	method), 117
tribute), 76	${\tt preserve}  (\textit{textworld.logic.model.ActionPreconditionNode} $
${\tt phName()}  (\textit{textworld.logic.parser.GameLogicSemantics}$	attribute), 113
method), 117	${\tt preserve} \ (\textit{textworld.logic.model.RulePreconditionNode}$
Placeholder (class in textworld.logic), 104	attribute), 115
placeholder() (textworld.logic.parser.GameLogicSeman method), 117	tiproperties (textworld.generator.maker.WorldEntity property), 82
PlaceholderNode (class in textworld.logic.model), 114	Proposition (class in textworld.logic), 106
player (textworld.generator.maker.GameMaker attribute), 76	proposition() (textworld.logic.parser.GameLogicSemantics method), 117
<pre>player_room (textworld.generator.world.World prop-</pre>	PropositionNode (class in textworld.logic.model), 115
erty), 73	<pre>push() (textworld.generator.dependency_tree.DependencyTree</pre>
PlayerAlreadySetError, 76	method), 59
plot_graph() (in module	
textworld.generator.graph_networks), 75	Q
policy_commands (textworld.core.EnvInfos attribute),	Quest (class in textworld.generator.game), 69
33	quest_breadth(textworld.generator.game.GameOptions
<pre>populate() (textworld.generator.world.World method),</pre>	attribute), 67
73	quest_breadth(textworld.generator.game.GameOptions
<pre>populate_room() (textworld.generator.world.World</pre>	property), 68
method), 73	quest_depth (textworld.generator.game.GameOptions
<pre>populate_room_with()</pre>	attribute), 67
(textworld.generator.world.World method), 73	<pre>quest_length (textworld.generator.game.GameOptions</pre>
<pre>populate_with()</pre>	quest_length (textworld.generator.game.GameOptions property), 68
$\verb"port" (\textit{textworld.envs.wrappers.viewer.HtmlV} iewer~prop-$	QuestError, 76
erty), 49	QuestGenerationError, 56
position_string() (textworld.render.render.GraphRoom method), 119	QuestProgression (class in textworld.generator.game), 71
postconditions (textworld.logic.model.ActionNode attribute), 113	R
postconditions (textworld.logic.model.RuleNode at-	RandomCommandAgent (class in
tribute), 115	textworld.agents.random), 53
preconditions (textworld.logic.model.ActionNode at-	RandomGenerator (class in textworld.utils), 123
tribute), 113	record_quest() (textworld.generator.maker.GameMaker
preconditions (textworld.logic.model.RuleNode	method), 79
attribute), 115	Recorder (class in textworld.envs.wrappers.recorder),
Predicate (class in textworld.logic), 105	49
<pre>predicate(textworld.logic.model.Inform7PredicateNode</pre>	RegexDict (class in textworld.utils), 123
attribute), 114	register_game() (in module textworld.gym.utils), 37
<pre>predicate() (textworld.logic.parser.GameLogicSemantic</pre>	relabel() (in module
<pre>predicateDecls() (textworld.logic.parser.GameLogicSet</pre>	mantics textworld.generator.graph_networks), 75
method), 117	${\tt remove()} \ (\textit{textworld}. \textit{generator}. \textit{dependency\_tree}. \textit{DependencyTree}. \\$
PredicateNode (class in textworld.logic.model), 115	method), 59
predicates (textworld.logic.model.Inform7PredicatesNoa attribute), 114	remove() (textworld.generator.game.ActionDependencyTree method), 61
${\tt predicates}\ ({\it textworld.logic.model.PredicatesNode}\ {\it at-}$	remove() (textworld.generator.maker.WorldEntity
tribute), 115	method), 81
predicates() (textworld logic parser GameLogic Semanti	cs

<pre>remove_fact() (textworld.generator.maker.WorldEntity</pre>	attribute), 115 reverseRule() (textworld.logic.parser.GameLogicSemantics
remove_fact() (textworld.logic.State method), 110	method), 117
remove_facts() (textworld.logic.State method), 110	reverseRuleDecls() (textworld.logic.parser.GameLogicSemantics
remove_property() (textworld.generator.maker.WorldEn	
method), 81	ReverseRuleNode (class in textworld.logic.model), 115
removed (textworld.logic.Action property), 104	reverseRules() (textworld.logic.parser.GameLogicSemantics
render() (textworld.core.Environment method), 35	method), 117
render() (textworld.core.Wrapper method), 36	ReverseRulesNode (class in textworld.logic.model), 115
render() (textworld.envs.glulx.git_glulx.GitGlulxEnv	reward (textworld.generator.game.Quest attribute), 69
method), 48	rhs (textworld.logic.model.AliasNode attribute), 114
render() (textworld.generator.maker.GameMaker	rhs (textworld.logic.model.ReverseRuleNode attribute),
method), 79	115
render() (textworld.gym.envs.textworld_batch.TextworldB	BatolgGytaEmworld.generator.chaining.ChainingOptions at-
method), 42	tribute), 57
repeatable (textworld.generator.game.Quest attribute),	rngs (textworld.generator.game.GameOptions property),
70	68
repl_sing_plur() (in module	room_type (textworld.generator.game.EntityInfo at-
textworld.generator.text_generation), 84	tribute), 63
replace_num() (in module	rooms (textworld.generator.maker.GameMaker at-
textworld.generator.text_generation), 84	tribute), 76
<pre>reset() (textworld.agents.human.HumanAgent method),</pre>	rooms (textworld.generator.world.World property), 73
53	Rule (class in textworld.logic), 107
<pre>reset() (textworld.agents.random.NaiveAgent method),</pre>	rule (textworld.logic.model.Inform7CommandNode at-
53	tribute), 114
reset() (textworld.agents.random.RandomCommandAgen method), 54	atrule() (textworld.logic.parser.GameLogicSemantics method), 117
	ruleDecls() (textworld.logic.parser.GameLogicSemantics
54	method), 117
${\tt reset()}\ (textworld.agents.walkthrough.WalkthroughAgent)$	ruleName()(textworld.logic.parser.GameLogicSemantics
method), 54	method), 117
reset() (textworld.core.Agent method), 31	RuleNode (class in textworld.logic.model), 115
reset() (textworld.core.Environment method), 35	${\tt rulePrecondition()} \ (\textit{textworld.logic.parser.GameLogicSemantics}$
reset() (textworld.core.Wrapper method), 36	method), 117
reset() (textworld.envs.glulx.git_glulx.GitGlulxEnv	RulePreconditionNode (class in
method), 48	textworld.logic.model), 115
reset() (textworld.envs.tw.TextWorldEnv method), 47	rules (textworld.logic.model.RulesNode attribute), 115
reset() (textworld.envs.wrappers.filter.Filter method), 50	rules() (textworld.logic.parser.GameLogicSemantics method), 117
reset() (textworld.envs.wrappers.recorder.Recorder	rules_per_depth(textworld.generator.chaining.ChainingOptions
method), 49	attribute), 57
reset() (textworld.envs.wrappers.viewer.HtmlViewer	RulesNode (class in textworld.logic.model), 115
method), 49	
reset() (textworld.envs.zmachine.jericho.JerichoEnv	S
method), 50	<pre>sample() (textworld.generator.vtypes.VariableTypeTree</pre>
reset() (textworld.gym.envs.textworld.TextworldGymEnv	method), 61
method), 40	sample quest() (in module
$\verb"reset()" (textworld.gym.envs.textworld\_batch.TextworldBatch.Te$	atchGymEnyextworld.generator.chaining), 58
method), 42	save() (textworld.generator.game.Game method), 65
${\tt restricted\_types} \ ({\it textworld.generator.chaining.Chaining.chaining$	saptions (textworld.generator.logger.GameLogger
attribute), 57	method), 60
reverse_direction() (in module	<pre>save_graph_to_svg() (in module textworld.utils), 124</pre>
$textworld.generator.graph\_networks), 75$	score (textworld.core.EnvInfos attribute), 33
${\tt reverse\_rules} \ ({\it textworld.logic.model.ReverseRulesNode}$	

score	(textworld.generator.game.GameProgression	123
	property), 69	<pre>set_walkthrough() (textworld.generator.maker.GameMaker</pre>
seed (te	xtworld.utils.RandomGenerator property), 123	method), 80
seed()	(textworld.core.Environment method), 35	<pre>shortest_path()</pre>
seed()	(textworld.core.Wrapper method), 36	$textworld. \textit{generator.} \textit{graph\_networks}), 75$
seed()	(textworld.envs.zmach in e.jericho.Jericho Env	shuffled_cycle() (in module
	method), 50	textworld.gym.envs.utils), 43
seed()	$(textworld.gym.envs.textworld\_batch.TextworldBat$	
	method), 42	signature (textworld.logic.Proposition attribute), 107
seeds	(textworld.generator.game.GameOptions attribute), 67	<pre>signature() (textworld.logic.parser.GameLogicSemantics</pre>
seeds	(textworld.generator.game.GameOptions prop-	
	erty), 68	<pre>signatureOrAlias() (textworld.logic.parser.GameLogicSemantic</pre>
serial:	ize() (textworld.generator.data.KnowledgeBase	
	method), 88	skip() (textworld.gym.envs.textworld_batch.TextworldBatchGymEn
serial:		
	method), 62	source (textworld.logic.model.Inform7PredicateNode at-
serial:	ize() (textworld.generator.game.Event method),	
	64	south (textworld.generator.maker.WorldRoom attribute),
serial	ize() (textworld.generator.game.Game method),	
	65	<pre>split_name_adj_noun()</pre>
serial	ize() (textworld.generator.game.Quest method),	
	70	method), 86
Seriai.	<pre>ize() (textworld.generator.text_grammar.Gramma   method), 86</pre>	
coriali		textworld.generator.inform7.world2inform7),
Serrar.	ize() (textworld.generator.vtypes.VariableType method), 60	start() (textworld.logic.parser.GameLogicSemantics
serial:	ize() (textworld.generator.vtypes.VariableTypeTre	
301141.	method), 61	start() (textworld.render.serve.Server method), 120
serial	ize() (textworld.generator.world.World method),	
	73	method), 121
serial	ize() (textworld.logic.Action method), 103	start_server() (textworld.render.serve.VisualizationService
	ize() (textworld.logic.GameLogic method), 104	method), 121
	ize() (textworld.logic.Placeholder method), 105	
	ize() (textworld.logic.Predicate method), 106	state (textworld.generator.maker.GameMaker prop-
	ize() (textworld.logic.Proposition method), 106	erty), 80
serial	ize() (textworld.logic.Rule method), 108	state (textworld.generator.world.World property), 73
serial	ize() (textworld.logic.State method), 110	stats() (textworld.generator.logger.GameLogger
serial	ize() (textworld.logic.Variable method), 113	method), 60
	(class in textworld.render.serve), 120	<pre>step() (textworld.core.Environment method), 35</pre>
Servers	SentEvent (class in textworld.render.serve), 121	<pre>step() (textworld.core.Wrapper method), 36</pre>
set_co	nditions() (textworld.generator.game.Event method), 64	step() (textworld.envs.glulx.git_glulx.GitGlulxEnv method), 48
set_ope	en_closed_locked()	<pre>step() (textworld.envs.tw.TextWorldEnv method), 47</pre>
	(textworld.render.render.GraphItem method),	step() (textworld.envs.wrappers.filter.Filter method), 50
	119	step() (textworld.envs.wrappers.recorder.Recorder
set_pla	ayer() (textworld.generator.maker.GameMaker	
	method), 79	step() (textworld.envs.wrappers.viewer.HtmlViewer
set_pla	ayer_room() (textworld.generator.world.World	
	method), 73	step() (textworld.envs.zmachine.jericho.JerichoEnv
set_que	est_from_commands()	method), 51
	(textworld.generator.maker.GameMaker	step() (textworld.gym.envs.textworld.TextworldGymEnv
co+ c=	method), 80	method), 41
set_se	ed() (textworld.utils.RandomGenerator method),	<pre>step() (textworld.gym.envs.textworld_batch.TextworldBatchGymEn</pre>

method), 43	textworld.envs
$\verb stop_server()  (textworld.render.serve. Visualization Services)                                      $	ce module, 47
method), 121	textworld.envs.glulx
<pre>str() (textworld.logic.parser.GameLogicSemantics</pre>	module, 48
method), 117	textworld.envs.glulx.git_glulx
str2bool() (in module textworld.utils), 124	module, 48
<pre>strBlock() (textworld.logic.parser.GameLogicSemantics</pre>	textworld.envs.tw
method), 117	module, 47
$\verb subquests  (textworld.generator.chaining.ChainingOptions ) \\$	textworld.envs.wrappers
attribute), 57	module, 49
<pre>subscribe() (textworld.render.serve.Server method),</pre>	textworld.envs.wrappers.filter
121	module, 49
<pre>substitute() (textworld.logic.Predicate method), 106</pre>	textworld.envs.wrappers.recorder
<pre>substitute() (textworld.logic.Rule method), 108</pre>	module, 49
subtypes (textworld.logic.Type property), 111	textworld.envs.wrappers.viewer
<pre>supertypes (textworld.logic.model.TypeNode attribute),</pre>	module, 49
116	textworld.envs.zmachine
supertypes (textworld.logic.Type property), 111	module, 50
${\tt SUPPORTER}(textworld.generator.vtypes.Variable Type Tree$	textworld.envs.zmachine.jericho
attribute), 61	module, 50
SupressStdStreams (class in textworld.render.serve),	textworld.generator
121	module, 55
synonyms (textworld.generator.game.EntityInfo at-	textworld.generator.chaining
tribute), 63	module, 56
<b>-</b>	textworld.generator.data
T	module, 88
take() (in module textworld.utils), 124	textworld.generator.dependency_tree
take_screenshot() (in module	module, 58
textworld.render.render), 120	textworld.generator.game
temp_viz() (in module textworld.render.render), 120	module, 61
test() (textworld.generator.maker.GameMaker	textworld.generator.graph_networks
method), 80	module, 75
	textworld.generator.inform7
module, 31	module, 98
textworld.agents	${\tt textworld.generator.inform7.world2inform7}$
module, 53	module, 98
textworld.agents.human	textworld.generator.logger
module, 53	module, 60
textworld.agents.random	textworld.generator.maker
module, 53	module, 76
textworld.agents.simple	textworld.generator.text_generation
module, 54	module, 83
textworld.agents.walkthrough	textworld.generator.text_grammar
module, 54	module, 84
textworld.challenges.coin_collector	textworld.generator.vtypes
module, 23	module, 60
textworld.challenges.cooking	textworld.generator.world
module, 25	module, 72
textworld.challenges.simple	textworld.gym
module, 22	module, 37
textworld.challenges.treasure_hunter	textworld.gym.envs
module, 28	module, 40
	textworld.gym.envs.textworld
module, 31	module, 40

toutuanld aum anus toutuanld hatch	+ ma (tartward logic model Placeholder Node attribute)
textworld.gym.envs.textworld_batch module, 41	type (textworld.logic.model.PlaceholderNode attribute), 115
textworld.gym.envs.utils	type (textworld.logic.model.VariableNode attribute), 116
module, 43	type (textworld.logic.Placeholder attribute), 105
textworld.gym.spaces	type (textworld.logic.Variable attribute), 113
module, 43	type() (textworld.logic.parser.GameLogicSemantics
textworld.gym.spaces.text_spaces	method), 117
module, 43	TypeHierarchy (class in textworld.logic), 111
textworld.gym.utils	TypeNode (class in textworld.logic.model), 115
module, 37	typePart() (textworld.logic.parser.GameLogicSemantics
textworld.logic	method), 117
module, 103	types (textworld.logic.model.DocumentNode attribute),
textworld.logic.model	114
module, 113	types (textworld.logic.model.SignatureNode attribute),
textworld.logic.parser	115
module, 116	types (textworld.logic.Predicate property), 106
textworld.render	types (textworld.logic.Proposition property), 107
module, 119	types (textworld.logic.Signature attribute), 108
textworld.render.render	cypes (territorialises), 100
module, 119	U
textworld.render.serve	UnderspecifiedEventError, 61
module, 120	UnderspecifiedQuestError, 61
textworld.utils	
module, 123	unfinishable (textworld.generator.game.QuestProgression
TextworldBatchGymEnv (class in	property), 71
textworld.gym.envs.textworld_batch), 41	unique_expansion(textworld.generator.text_grammar.GrammarOptions
TextWorldEnv (class in textworld.envs.tw), 47	attribute), 87
	unique_product() (in module textworld.utils), 124
textworld.gym.envs.textworld), 40	uniquify() (in module textworld.utils), 124
TextworldInform7Warning, 98	untriggerable (textworld.generator.game.EventProgression
	property), 65
theme (textworld.generator.text_grammar.GrammarOption attribute), 87	
	update() (textworld.generator.game.EventProgression
to_dict() (textworld.render.render.GraphItem method), 119	method), 64
	update() (textworld.generator.game.GameProgression
tokenize() (textworld.gym.spaces.text_spaces.Char	method), 69
method), 44	update() (textworld.generator.game.QuestProgression
tokenize() (textworld.gym.spaces.text_spaces.Word	method), 71
method), 45	update_state() (textworld.render.serve.VisualizationService
tracking_quests(textworld.generator.game.GameProgr	,,
property), 69	update_subscribers() (textworld.render.serve.Server
triggered (textworld.generator.game.EventProgression	method), 121
property), 65	uuid (textworld.generator.game.GameOptions property),
triggering_policy(textworld.generator.game.EventPro	
property), 65	uuid(textworld.generator.text_grammar.GrammarOptions
Type (class in textworld.logic), 111	property), 87
type (textworld.generator.game.EntityInfo attribute), 63	V
type (textworld.generator.maker.WorldEntity property),	V
82	valid_actions (textworld.generator.game.GameProgression
type (textworld.generator.world.WorldEntity attribute),	property), 69
74	validate() (textworld.generator.maker.GameMaker
type (textworld.generator.world.WorldObject attribute),	method), 80
74	values (textworld.generator.dependency_tree.DependencyTree
type (textworld.generator.world.WorldRoom attribute),	property), 59
74	Variable (class in textworld.logic), 113

```
variable() (textworld.logic.parser.GameLogicSemantics
                                                                 textworld.generator.graph networks), 75
         method), 117
variable_named() (textworld.logic.State method), 110
VariableNode (class in textworld.logic.model), 116
variables (textworld.logic.Action property), 104
variables (textworld.logic.State property), 110
variables_of_type() (textworld.logic.State method),
         110
VariableType (class in textworld.generator.vtypes), 60
VariableTypeTree
                                 (class
                                                   in
         textworld.generator.vtypes), 60
verbs (textworld.core.EnvInfos attribute), 33
verbs (textworld.generator.game.Game property), 66
VERSION (textworld.generator.inform7.world2inform7.Inform7Game
         attribute), 99
VisualizationService
                                   (class
                                                    in
         textworld.render.serve), 121
visualize() (in module textworld.render.render), 120
VocabularyHasDuplicateTokens, 43
W
walkthrough (textworld.generator.game.Game prop-
         erty), 66
WalkthroughAgent
                                 (class
                                                    in
         textworld.agents.walkthrough), 54
WalkthroughDone, 54
WebdriverNotFoundError, 119
west (textworld.generator.maker.WorldRoom attribute),
which() (in module textworld.render.render), 120
win_events (textworld.generator.game.Quest attribute),
win_events (textworld.generator.game.Quest property),
win_facts (textworld.core.EnvInfos attribute), 33
winning_policy (textworld.generator.game.GameProgression
         property), 69
winning_policy (textworld.generator.game.QuestProgression
         property), 71
won (textworld.core.EnvInfos attribute), 34
Word (class in textworld.gym.spaces.text_spaces), 44
World (class in textworld.generator.world), 72
WorldEntity (class in textworld.generator.maker), 80
WorldEntity (class in textworld.generator.world), 73
WorldObject (class in textworld.generator.world), 74
WorldPath (class in textworld.generator.maker), 82
WorldRoom (class in textworld.generator.maker), 82
WorldRoom (class in textworld.generator.world), 74
WorldRoomExit (class in textworld.generator.maker), 83
Wrapper (class in textworld.core), 35
wrappers (textworld.core.Agent property), 32
X
xy_diff()
                                               module
                           (in
```