dcss-ai-wrapper: An API for Dungeon Crawl Stone Soup providing both Vector and Symbolic State Representations

Dustin Dannenhauer, [1] **Zohreh A. Dannenhauer,** [2] **Jonathan Decker,** [3] **Adam Amos-Binks,** [4] **Michael W. Floyd,** [2] **David W. Aha** [3]

[1] Parallax Advanced Research Corporation | Dayton, OH
[2] Knexus Research Corporation | National Harbor, MD
[3] Navy Center for Applied Research in AI | Naval Research Laboratory (Code 5510) | Washington, DC
[4] Applied Research Associates | Raleigh, NC
dustin.dannenhauer@parallaxresearch.org, zohreh.dannenhauer@knexusresearch.com, jonathan.decker@nrl.navy.mil, aamosbinksaamosbinks@ara.com, michael.floyd@knexusresearch.com, david.aha@nrl.navy.mil

Abstract

Dungeon Crawl Stone Soup is a single-player, free, and open-source rogue-like video game with a variety of features that make it a challenge for artificial intelligence (AI) research. *dcss-ai-wrapper* is the first API designed to enable intelligent agents to play Dungeon Crawl Stone Soup. We describe the vector and symbolic relational state representations available through the *dcss-ai-wrapper*, as well as how to use the API to develop custom agents. By providing both vector and relational representations, we hope to spur advances in reinforcement learning, automated planning, and other cognitive and learning techniques. This API is similar in spirit to recent game APIs such as the Nethack Learning Environment, MALMO, ELF, and the Starcraft II API. The complexities of Dungeon Crawl Stone Soup include actions with delayed consequences, partial observability, stochastic actions where probabilities change over time, extremely sparse rewards, procedurally generated environments, sensing actions, and dynamic monsters and level-specific events. Our contributions are (1) a description of the publicly available *dcss-ai-wrapper*, (2) an API that supports both vector and PDDL representations of the DCSS game state, and (3) a high-level PDDL model of Dungeon Crawl Stone Soup compatible with the FastDownward planner. *dcss-ai-wrapper* is available at https://github.com/dtdannen/dcss-ai-wrapper.

Introduction

Dungeon Crawl Stone Soup[1] (DCSS) is a single-player, free, and open-source rogue-like turn-based video game that consists of a procedurally generated 2-dimensional grid world. To win the game, a player must navigate their character through a series of levels to collect 'The Orb of Zot' and then return to the starting location. Along the way, players encounter a wide variety of monsters and items. Players equip and use items to make themselves stronger or consume them to aid in difficult situations. The DCSS world is dynamic, stochastic, partially observable, and complex: when considering the all tiles in a game, the number of possible game states is orders of magnitude larger than games such as Starcraft and Go, and the number of instantiated actions the player may take can reach into the hundreds.

DCSS is notoriously hard for humans. Comments such as "*Wow. I've finally gotten my first win since I started playing, almost exactly 3 years ago.*"[2] frequently appear on DCSS message boards. More experienced players regularly answer questions and provide advice to newer players. A single game takes on the order of hours to complete; for example, the average playtime for games in a large-scale tournament of human players in 2016 was 8.5 hours.

Rogue-like games are famous for their characteristic of *permanent death*: when the player dies, the game ends. Making a single mistake, or a series of small mistakes, will often lead to failure. Worse, sometimes these mistakes are realized only hundreds or thousands of turns later. For example, a player may use a one-time-use life-saving item when they could have used a repeatable ability or the player may have trained skills in such a way that they have vulnerabilities against more powerful monsters found later in the game.

We have developed the first DCSS API designed to support AI research studies. This API is written primarily in Python and is freely available at the public GitHub repository (https://github.com/dtdannen/dcss-ai-wrapper), which also includes a public Gitter chatroom (https://gitter.im/dcss-ai-wrapper/community) with permanently hosted, publicly available discussions on using the API. This *dcss-ai-wrapper* API offers several desirable characteristics for evaluating new and existing AI techniques:

- A simulated environment that is partially observable, dynamic, and stochastic, with an environment model that changes over time (i.e., the probabilities associated with the success of a player's actions change over time).
- An environment requiring rich knowledge to progress. This includes multiple types of knowledge such as factual (e.g., the player must obtain 3 runes before entering *The Realm of Zot* level), strategic (e.g., aviod fighting a hydra monster with a non-fire bladed weapon), and descriptive knowledge (almost every aspect of the game has an associated English text description designed for a human user including all objects, tiles, and monsters).

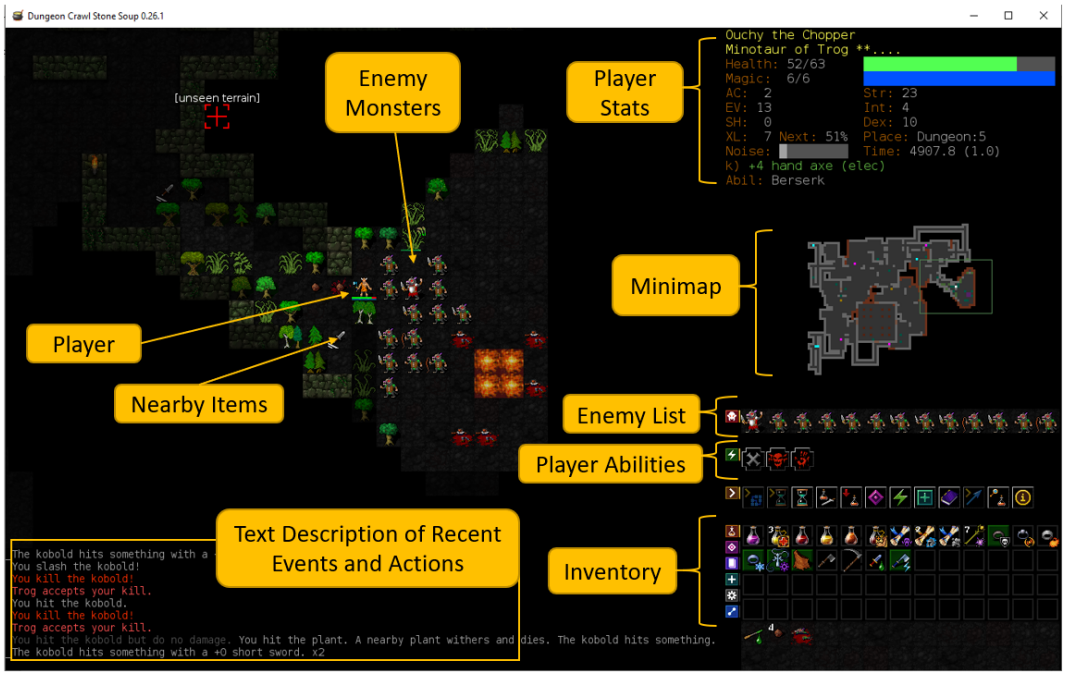

Figure 1: Annotated Screenshot of the Dungeon Crawl Stone Soup Interface

- A game that requires long-term strategic planning where early decisions can have a significant impact on later game play. Poor decisions early can have irreversible effects and critical consequences (e.g., permanent death).
- An environment that does not penalize slow reaction times. DCSS is a turn-based game with no time limit on deciding which action to take next. New players are often advised to pause when they realize they are in a dangerous situation in order to (1) carefully consider all of their options and (2) learn about the monsters and items in the current situation from online knowledge sources (e.g., a wiki, forum, and live IRC chat with other players).
- There is existing data on human performance for thousands of previously-played games. This provides an opportunity for comparing the performance of human and intelligent agents using DCSS.

We next describe the state space and environment properties that make DCSS an interesting research domain. We then describe the skill level of human DCSS players from an annual tournament, followed by the *dcss-ai-wrapper* API, how to develop custom agents, and the available vector and PDDL state representations. We then discuss similar environments and their use in AI research, including the recent Nethack Learning Environment. We conclude with a discussion of research topics for which *dcss-ai-wrapper* may be of use.

DCSS State Space and Environment Properties

The graphical user interface to DCSS is shown in Figure 1. It includes the following information on the current state:

Top Right The player's stats, which include the player's piety level with the god they are currently worshipping, health points, magic points, attributes describing armor, evasiveness, shielding, strength, intelligence, and dexterity, the player's experience level, place, and depth in the game, the current noise level, the amount of time (number of turns taken), the currently equipped weapon, and the current item type in the quiver.

Middle Right The minimap of the entire level explored so far.

Bottom Right Nearby monsters, player-activated abilities, and inventory items (the player can hold 52 unique item types, there is no limit to the quantity of the number of items of a single type).

Center This is the main area of the game. It displays the tiles around the player, with the view being centered on the player's tile. This area is also where menus (inventory, etc.) appear.

Bottom Left The most recent messages (natural language text) that describe what's happened in the game. Every action and event in the game is associated with one or more natural language statements. Multiple attacks by both the player and enemies can occur from a single keypress that initiates the next action. The text messages describe what happened and their order, and are the only way to obtain a higher-fidelity sequence of events. All messages, in the order they are produced by the game, are available via the *dcss-ai-wrapper*.

DCSS has multiple characteristics that contribute to its high complexity. We describe these here, followed by a the-

oretical analysis on the lower bound of state and action space size.

- 650+ unique monster types that the player may encounter, many of which require specific actions, attributes, or special knowledge to defeat. For example, if you attack a hydra monster with a weapon that has a blade (e.g., axe, sword) you will chop off it's head and it will grow more in its place, and as a result become much stronger. A good approach to defeating hydras is to use a bladed weapon enchanted with fire (which sears the wound), use a blunt force object such as a mace, or use sufficiently strong magic spells.
- 13,800 possible starting character configurations formed by choosing: one of 23 species (e.g., vampire, ogre), one of 24 backgrounds (e.g., fighter, wizard, berserker), and one of 25 deities for your character to worship that may provide additional benefits (e.g., worshiping *Gozag Ym Sagoz* turns slain enemy corpses into gold). Some are considered easier than others; a minotaur berserker worshiping Trog is the recommended starting character for new players who have yet to win a game.
- 31 skills (e.g., fighting, short blades, hexes, charms, and shields) and 3 attributes (strength, intellect, and dexterity) that are increased by spending experience points. The value of each skill ranges from 0 to a maximum of value of 27. Spending experience is permanent and cannot be undone (except under special circumstances). Poor decisions in allocating experience points for skills and attributes can prevent players from winning a game, since improperly raising your character's attributes yields deficiencies against certain monster types found later in a game. It is also specific to the items and spells a character will focus on, which often changes during the course of a game. Finding a rare and powerful item meant for melee may warrant an entire strategy change for a character that is currently magic-based. It is not always an easy decision because there may not be enough time to raise skill and attribute values before encountering monsters that require high values to defeat.
- 100+ spell actions a player can learn. A player can retain a maximum of only 21 spells at any time. Spells have unique effects that sometimes require careful planning. Some spells buff the player with attributes that affect later actions. For example, when in a situation where time is of utmost importance, casting a spell that temporarily increases the player's speed should often be cast first.
- 48 unique types of melee and ranged weapons that a player may encounter and use. Each weapon may be branded to give it additional effects (e.g., fire, frost, or venom) that may cause additional damage and special effects (e.g., a monster hit with a venom brand will gain a temporary poison status that causess damage over time).
- 15 runes to be collected. Runes are special items; they do not require inventory space, and they enable a player to visit new branches (series of levels) of the dungeon. Collecting a minimum of three runes is necessary to access the *Realm of Zot* level, which is required to win the game. While 3 runes are the minimum requirement, many players challenge themselves to see how many runes they can acquire. Runes are associated with special areas in the game (i.e. the serpentine rune requires fighting snake-themed monsters and a resistance to poison is highly recommended). Some runes are significantly more difficult to obtain than others.
- Approximately 65,000 to 80,000 turns is typical for a 3-rune game. Turns can be considered an approximation of the number of actions taken. This can vary depending on the speed of the player, which may be faster or slower than the turn rate, in which case a fast player may take 2 actions in 1.5 turns or a slow player may take 1 action in 1.5 turns. Speed is an attribute of the player's character depending on their attributes and items (e.g. equipping heavier armor can slow attack speed; other items may increase or decrease the player's movement speed). Speed here does not refer to how long a player takes to select the next action, as the game is turn-based.
- 40+ consumable resource items, including: 18 potions, 10 scrolls, 11 wands and a small number of specialty items. Potions and scrolls are single-use and offer some of the most important life saving capabilities, such as a scroll of blinking that instantly teleports a character to another tile within their line of sight.
- Players may encounter more than 70,000 tiles before completing a game. A tile is a location on the grid that may hold a combination of monsters, items, and special terrain features (e.g., lava, water, or steam).
- 100+ levels. Levels are composed of tiles that are procedurally generated to form rooms, passageways, etc. using a variety of terrain elements such as walls, shallow water, deep water, or lava. Levels are connected via staircases that act as portals from one level to the next. A 3-rune game requires visiting at least 45 levels. Most levels, after they have been procedurally generated when the game starts, have a static arrangement of tiles. The two exceptions are the levels Abyss and Labyrinth where the number of tiles is infinite and the layout of tiles outside the player's line of sight constantly changes.
- Partially observable: A player does not see a tile until it is within the character's line of sight, which is normally within seven tiles in any cardinal direction.
- Dynamic: Monsters take their own actions, independent of the player, and some events (e.g., as entrances to special areas) close after a time limit (e.g., volcano and sewer levels).
- Stochastic: Most actions (e.g., melee attacks or spells) are probabilistic and often fail. As the player increases or decreases their attributes, the probability of success changes.
- Natural language text accompanies every item and action in the game. The player can ask for a description of any tile, object, monster, etc., within view or in the player's inventory.
- Permanent death: If a player dies, the game ends and they must start again in a newly generated world. The only way

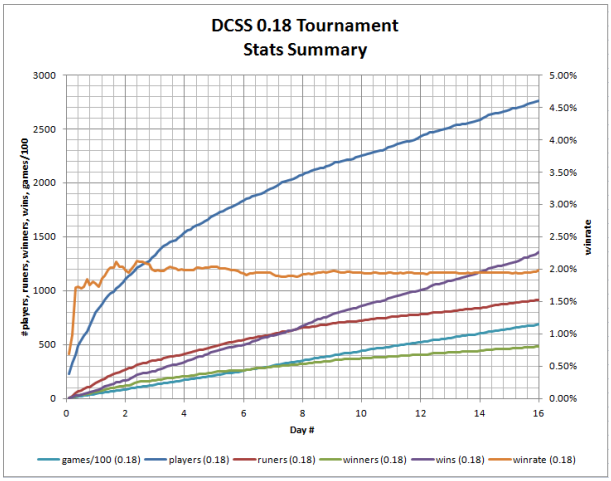

Figure 2: Results from the v. 0.18 tournament

to replay a game is to manually set the seed for the procedural generation.

We provide a lower bound complexity analysis of the state space for a complete game using lower-bound assumptions on the numbers of tiles, items, monsters, etc. that the player would encounter in a 3-rune game. Technically DCSS has an infinite state space[3].

- 70,000 tiles
- 900 items (an estimated 20 items per level $\times$ 45 levels)
- 2000 monsters

For simplicity, we assume that monsters and items will not be generated on the same tile. With these minimum assumptions, the state space is

$$|S| = 70000^{2900} \approx 10^{14000}$$

which is significantly more than StarCraft (an estimated lower bound of 10^{1685}), Go (10^{170}), and Chess (10^{50}) (Ontanón et al. 2013). However, StarCraft has a significantly higher action space, estimated at 10^8 (Vinyals et al. 2017); DCSS' number of grounded actions is no more than 1000 in any given state. A primary difference between StarCraft and DCSS is real-time decision making. Since DCSS does not penalize long reaction times, cognitive approaches for more deliberate reasoning (such as planning and inference mechanisms) can be effectively evaluated in DCSS, while still ensuring a highly complex environment.

Annual DCSS Tournament

A tournament is held with every major release of the game (e.g., v. 0.17, v. 0.18, or v. 0.19); it includes thousands of players and spans 16 days. During this time, players try to collect as many points as possible by playing a variety of different character configurations (i.e., species, background, and deity combinations). The results for the tournament using v. 0.18 of the game (PleasingFungus 2016) were posted on 3 June 2016, and some are shown in Figure 2. Tournaments attract the best human players to exhibit their skill and serve as one possible benchmark with which to evaluate AI agents against humans. For those unfamiliar with DCSS, here are some statistics from the v. 0.18 tournament:

- The average won game required around eight and half hours of human playtime.
- The fastest run was 41:00 minutes.
- 2500 human players competed, and only about 20% won a game.
- The overall win rate of games attempted was slightly higher than 2%.

In addition to winning the game, many intermediate measures can be used to assess agent performance. These include number of runes collected, number of levels reached, time, number of actions taken, number of monsters killed, etc.

API

Our *dcss-ai-wrapper* is the first DCSS API designed to facilitate AI research. We use the term 'wrapper' because this API does not modify the original DCSS source code, allowing it to support multiple versions of DCSS (which is under active development itself[4]) such that the user can treat the game source code as a black box. Instead, our wrapper provides features for easily obtaining different types of state representations and capabilities for running experiments. *dcss-ai-wrapper* is written in Python 3 and is available on Github[5] under the MIT license. Currently, the wrapper is meant to be used with DCSS in webserver mode (DCSS can be run locally in the terminal using ASCII graphics and on a webserver where games are played in a browser; we provide a docker container preloaded with DCSS in webserver mode for immediate use). The API is under regular, active development.

DCSS Game Modes

Dungeon Crawl Stone Soup can be played in one of four different game modes: *trunk*, *trunk seeded*, *tutorial*, and *sprint*. **Trunk** refers to the full game of DCSS as it's meant to be played, procedurally generated each time with a random seed. **Trunk Seeded** is the full game of DCSS with the extra option to provide a seed value. A seed number affects most calls to the random number generator, including dungeon layout generation, however it does not guarantee all actions over an entire game will lead to the same result. The developers of DCSS do not guarantee that two games with exactly the same seed will behave identically for the same set of actions, which my be relevant for highly precise experimentation. Since this is an issue with the DCSS game engine, it is outside the scope of our API. **Tutorial** game mode allows the user to specify one of 5 tutorial levels with special instructions and mini challenges designed to introduce the player to different aspects of the game (e.g., movement, fighting, items, or casting spells). Tutorials are fixed and therefore not procedurally generated. **Sprint** game mode is a "fast-paced Crawl variant in which you must explore a single brutally hard floor in pursuit of the orb of Zot"[6]. A player's experience and piety increases at a rate

of 9x compared to **Trunk** game mode. The floorplan, monsters, and items are usually fixed, meaning you can expect the exact same encounters each time.

Create and Run a Custom Agent

To create a custom agent after installing dcss-ai-wrapper, two steps are needed: (1) create a new agent that is a subclass of the *BaseAgent* class (a simple example is shown in Listing 1); and (2) create a connection to the DCSS game using the WebSockGame class, set configuration settings (or use the defaults), and call run() to start the agent (an example is shown in Listing 2). Together Listings 1 and 2 provide a simple yet complete example of developing a custom agent and running it on Tutorial 1.

```
1  from dcss.agent.base import BaseAgent
2  from dcss.state.game import GameState
3  from dcss.actions.action import Action
4
5  class MyAgent(BaseAgent):
6
7      def __init__(self):
8          super().__init__()
9          self.gamestate = None
10
11     def get_action(self, gamestate: GameState):
12         self.gamestate = gamestate
13         # get all possible actions
14         actions = Action.get_all_move_commands()
15         # call your planner or policy instead of random:
16         return random.choice(actions)
```

Listing 1: Example Custom Agent named MyAgent

Lines 1-3 in Listing 1 import several fundamental classes from the *dcss-ai-wrapper* Python package: *BaseAgent* (the parent class for DCSS agents), *GameState* (the DCSS game state information), and *Action* (actions an agent can perform in DCSS). Line 5 is the class signature for the new agent, *MyAgent*, that is a subclass of the *BaseAgent* class. All custom-made agents need to subclass the *BaseAgent* class for the *WebSockGame* class to be able to load and run the agent. Lines 7-9 are optional, but allow for creating a custom constructor method in case any additional initialization is necessary when an instance of the class is created. In this example, it calls *BaseAgent*'s constructor (line 8) and sets the *gamestate* to *None* (since the game has not yet started). The most important parts of the *MyAgent* class are lines 11-16 which depict the primary method for which the agent interacts with the game through the get_action(gamestate) method. This is called by *dcss-ai-wrapper* when it requires the next action from the agent. The method receives the current *gamestate* object as a parameter and returns the *action* the agent wishes to perform. In this example, the *MyAgent* randomly selects an action to perform from the set of possible movement actions. For a more complex agent, the logic in this method would be replaced by a custom action selection policy.

Listing 2 depicts the code needed to run the game. Because the current version of the API supports playing DCSS via websockets we use the WebSockGame and WebserverConfig classes (for future work we will extend the API to support playing the game locally in a terminal). Line 5 shows how to obtain the default webserver configuration

```
1  from dcss.websockgame import WebSockGame
2  from dcss.connection.config import WebserverConfig
3
4  def main():
5      my_config = WebserverConfig
6
7      # set game mode to Tutorial #1
8      my_config.game_id = 'tut-web-trunk'
9      my_config.tutorial_number = 1
10
11     # create game
12     game = WebSockGame(config=my_config,
                          agent_class=MyAgent)
13     game.run()
```

Listing 2: Running MyAgent on Tutorial 1

class. This configuration supports many options including: which game mode to select, whether to start a new game automatically after the current one has finished, what seed to use for procedural generated game modes, and the character options of species (i.e. *Minotaur*), background (i.e., *Berserker*), and starting weapon (i.e. *Hand Axe*). Lines 8 and 9 modify two of the default values of the *WebserverConfig* to run the game in Tutorial 1. A new *WebSockGame* is then created using the custom configuration, *my_config*, and custom agent, *MyAgent*, as shown on Line 12, and finally the game is run (Line 13). Recall that when the game is run, the game will periodically call the agent's `get_action(...)` method by providing the current game state, at which time the agent selects an action to perform. In this example, *MyAgent* will move around the environment using random movement actions.

Vector State Representations

During a DCSS game, a player moves around a gridworld with multiple levels, and the view of the game is centered on the player. Since the view is egocentric, our wrapper provides partial state representations of the area around the player, as well as global information of all areas visited. Since DCSS is partially observable and the player has a line of sight (LOS) of only 7 tiles in any direction (not including the tile the player is on), information on tiles outside its LOS may be outdated. Additional data, such as player stats, inventory, etc. that are independent of the gridworld tiles are given as fixed-size vectors. Each vector can be obtained independently through the API, and multiple options exist for obtaining the map data vector of different sizes:

`get_player_stats_vector()` Returns a vector of size 170 representing player stats such as health, gold, status effect, or mutations.

`get_player_inventory_vector()` Returns a vector of size 364 representing player inventory containing up to 52 items where each item has multiple attributes.

`get_player_spells_vector()` Returns a vector of size 244 representing player spells, including which spells the player knows and which spells are available to learn, and attributes of currently known spells.

`get_player_abilities_vector()` Returns a vector of size 376 representing player abilities, including

whether the player has the ability, its costs, and its likelihood of success.

`get_player_skills_vector()` Returns a vector of size 62 representing player skills. Each skill has two values associated with it: the current value of the skill and how much new experience will be allocated to increasing that skill.

`get_egocentric_LOS_map_data_vector()` Returns a vector of size up to 2,176 representing map data only within the player's current line of sight. Each cell is represented by 34 attributes.

`get_egocentric_level_map_data_vector()` Returns a vector of size up to 170,000+ representing map data only on the player's current level. Levels vary in their number of cells. Most have at least 2000, while some can have upwards of 5000 cells.

`get_all_map_data_vector()` Returns a vector in size up to 3,400,000+ representing map data containing up to all cells the player has encountered.

The latest documentation describing these vectors can be found on the projects online documentation[7].

PDDL State Representations

Our API, *dcss-ai-wrapper*, provides multiple functions to obtain the current state in the Planning Definition Domain Language (PDDL) (McDermott et al. 1998). These functions return a list of PDDL facts that, when combined with the static background knowledge, enable automated planning. Currently, agents have been developed to play the game using the FastDownward planner (Helmert 2006), and we plan to extend the PDDL representations to accommodate more planners in the future. The PDDL representation functions to obtain state information are similar to the vector representation functions, except that the number of facts returned is dynamic, and there is an additional, special function to provide a list of static background facts (such as which places in the dungeon are connected to other places). Briefly:

`get_player_stats_pddl()` Returns facts about the player such as health, gold, piety, strength, and known spells.

`get_player_inventory_pddl()` Returns facts about inventory items including which items the player has obtained and/or has equipped.

`get_player_skills_pddl()` Returns facts about the current level of each skill and whether the player is currently training that skill.

`get_egocentric_LOS_map_data_pddl(`*`radius`*`=7)` Returns only those facts about cells within the given *radius* of the player.

`get_egocentric_level_map_data_pddl()` Returns only those facts about cells on the player's current level.

`get_all_map_data_pddl()` Returns all facts regarding cells the player has encountered in the game thus far.

`get_background_pddl()` Returns the list of static facts that are provided as part of this API. This function always returns the same list of facts.

PDDL Domain Model

We provide a high-level PDDL domain model of Dungeon Crawl Stone Soup capable of achieving goals to move to different tiles, destroy monsters, pickup and equip items, train skills, cast spells, and travel to the next level. This domain file is not meant to accurately represent DCCS at a high level of fidelity. We speculate such a domain model would likely be its own significant research contribution, and a useful domain model will need to be constantly updated online because the likelihood of effects of an action occurring will change over the course of a single game. Rogue-like games such as DCCS and Nethack remain an unsolved challenge for AI research, and a high-fidelity model is outside the scope of our API. Instead, our PDDL model is meant to enable basic planning to achieve goals for moving, acquiring items, and attacking specific monsters. Given the inaccuracy of this model, a planner that uses it should be incorporated into an agent with plan execution monitoring and goal generation capabilities. We have tested this domain model with the Fastdownward planner. This complete domain model is provided in the Github repository (under the `/models` folder) and the current version at the time of writing is shown in Appendix A.

Related Work

The rogue-like genre of video games began with the original Rogue that was developed in 1980, having been inspired by the text adventure games of the 1970's. The genre now boasts many different titles, of which Nethack (released in 1987) and Dungeon Crawl Stone Soup (released circa 2006) remain highly popular. There has been prior work motivating Nethack for AI research: Winder and desJardins (2018) identified NetHack as "an immensely rich domain" worth using to evaluate concept-aware task transfer as future work, and Steinkraus and Kaelbling (2004) used a simplified version of NetHack[8] to evaluate learning abstractions for large MDPs. Regarding Rogue, Asperti et al. (2019) present an interface to the game where they show model-based reinforcement learning (RL) approaches can learn to perform navigation tasks.

Recently, Küttler et al. (2020) released the Nethack Learning Environment to evaluate RL algorithms. They provide the most comprehensive interface to the kinds of modern and highly difficult rogue-likes like Nethack and DCSS to date, and this marks a major step forward in providing access to these environments to facilitate current AI research.

Comparison to the Nethack Learning Environment

In the pursuit of advancing the field of RL, the *dcss-ai-wrapper* API complements the Nethack Learning Environment (Küttler et al. 2020) by providing a second, unique testbed with similar properties for RL research. Dungeon Crawl Stone Soup, while of the same genre, has a number of important differences compared to Nethack, such as more

branches of the dungeon to explore, special levels that are constantly changing as the player moves (Abyss, Labryinth), and DCSS has been described as more balanced such that death is more likely due to a player's mistake, rather than arbitrary luck[9] however there are a greater variety of ways to die in Nethack than in DCSS. DCSS is slightly more complex than Nethack in the number of starting character configurations and available magic spells to use. Interestingly from a non-combat planning perspective, DCSS play involves fewer situations that require specific chaining of actions and items to unlock a secret or advance a story line than in Nethack.

Our API provides a similar observation mechanism to the game to the Nethack Learning Environment's observation mechanism. The vector representation of *dcss-ai-wrapper* that provides an observation vector of tiles of a given radius around the player is similar to the fixed 21 x 79 matrix that Nethack offers as the observations of nearby tiles. Similarly, Nethack offers a 21-dimensional vector representing agent features while our API offers a 170 dimensional vector. Therefore we expect that RL approaches developed for the Nethack Learning Environment should be compatible with the *dcss-ai-wrapper* with only minor modifications. Using both DCCS and Nethack as evaluation environments for RL research should encourage developing domain-independent agents.

Unlike the Nethack Learning Environment or other interfaces to rogue-like games, the *dcss-ai-wrapper* API is the first environment to support AI algorithms based on symbolic relational representations. This offers unique challenges to the automated planning and cognitive systems research communities due to the large domain size, stochastic actions, and exploration in partially observable states. Also, there is the ability to compare vector with relational representation approaches, and perhaps approaches that mix these two representations. We hope *dcss-ai-wrapper* will enable research on empirically investigating the relative tradeoffs of vector and symbolic relational approaches, including experiments that measure exactly which types of information (and the quantity of information) must be provided to each agent, and the corresponding impact on performance.

Unlike the *dcss-ai-wrapper*, which is still under active development, the Nethack Learning Environment is a more mature testbed that clearly outlines specific tasks and rewards, and offers a cutting edge RL agent solution. While intelligent agents can make substantial progress on a game of Nethack, they cannot beat the game and perform on par with expert humans. Additionally, the Nethack Learning Environment boasts a speed of 14.4K environment steps per second, whereas our *dcss-ai-wrapper* does not yet surpass 1k actions per second when running on the terminal version on Linux platforms (this functionality is planned to be fully supported after v0.1 of our API). As of this writing, both Nethack and DCSS remain unsolved AI challenges, for RL, automated planning, and other cognitive systems approaches.

Prior work on developing programs to play DCSS and Nethack

Computer programs that play DCSS and Nethack have been hand-coded. **qw**[10] is the best known bot for DCSS; its highest winrate is about 15% for 3-rune games with the starting character of Deep Dwarf Fighter worshipping Makhleb, and it also achieves a 1% winrate for a 15-rune game with a Gargoyle Fighter worshipping Okawaru. The first bot to beat NetHack with no human intervention was created by Reddit user *duke-nh*[11]. Both of these bots rely extensively on expert-coded knowledge and rules, and do not perform learning. They demonstrate that programs can win these games under certain conditions and, being open source, provide baselines for AI agents playing these games.

Video games such as DCSS offer some of the complexities of real-world environments: dynamic, partially observable, open, etc., in a software simulation that is often less expensive and/or faster to evaluate new approaches. Several simulated environments have released in recent years: these include Microsoft Research's MALMO API for Minecraft (Johnson et al. 2016), Deepmind and Blizzard's Starcraft II API (Vinyals et al. 2017), and Facebook's ELF platform for Game Research (Tian et al. 2017). DCSS fills a gap in the available simulation environments because it it characterized by higher complexity, partial observability, and nondeterminism, yet does not require decision-making in real-time. This makes DCSS more manageable for agents that require deliberation in their decision making such as automated planning, inference, and online learning mechanisms.

Promising Research Questions Facilitated by the DCSS Domain

DCCS is unique in that it is a highly complex game in terms of both state and action space while also being difficult for humans to win. Playing well requires large amounts of different types of knowledge (factual and strategic, for example). Human performance data is available to compare against AI systems. ALso, almost everything in the game is accompanied by natural language text. Because of these characteristics, DCSS is an excellent research testbed to explore solutions to the following problems:

- **Achievement goals vs. learning goals:** An agent may find it is constantly dying in a situation and should consider taking different actions to explore the situation, or seek external information (such as querying the online wiki or asking a question on an online IRC chatroom) to understand why it's failing. Once it has found knowledge relevant to the problem the agent must decide how to use such knowledge.
- **Planning and acting with learned models:** The probabilities of an agent's effects (e.g., combat, likelihood to land a hit, likelihood to block or dodge an attack) change as the agent gets stronger and with respect to different types of monsters. How can an agent plan and act with these changing models, and how can it update its models?
- **Intelligent assistants and tutors:** Can an assistant be developed that aids players in completing the game by of-

fering advice or guidance? Perhaps an intelligent tutoring agent could observe a human player fail repeatedly in a situation (e.g., every time the human player faces a hydra monster, the character dies) and generate custom scenarios designed to teach the human player proper strategies to running from or defeating hydras. This could include lessons in allocating skill points, selecting among a variety of weapons, and using a variety of escape related items.

- **Explainable planning and goal reasoning agents:** The interpretability of AI systems has been an especially popular topic of workshops and related events since 2016, and in 2017 DARPA launched the Explainable AI (XAI) Program. Most of these efforts have focused on providing transparency to the decision making of machine learning (ML) systems in general, and deep networks more specifically[12]. While XAI research on data-driven ML is well-motivated, AI Planning is well placed to address the challenges of transparency and explainability in a broad range of interactive AI systems. For example, research on Explainable Planning has focused on helping humans to understand a plan produced by a planner (e.g., (Sohrabi, Baier, and McIlraith 2011; Bidot et al. 2010)), on reconciling the models of agents and humans (e.g., (Chakraborti et al. 2017)), and on explaining why a particular action was chosen by a planner rather than a different one (e.g., (Smith 2012; Langley et al. 2017; Fox, Long, and Magazzeni 2017).

  DCSS' rich environment contains different types of knowledge that make understanding decision making difficult. Novice players watching an expert player may not understand why certain decisions or actions were taken. Thus, DCSS could be a suitable environment for evaluating agents that explain their planning and other decision making components to humans.

- **Knowledge Extraction from Games:** DCSS is a knowledge rich game that takes humans many hours of playing and reading before acquiring enough knowledge to complete the game. While our API provides a starting point to use techniques such as automated planning, there are opportunities for new approaches to knowledge extraction that could be evaluated with DCSS.

- **Curriculum-based RL:** In environments such as DCSS there is delayed reward. The most obvious reward function is winning a game, but since this requires tens of thousands of actions to do so, intermediate reward functions are needed. The player's cumulative game score could be used, but this may not be enough to determine such actions as spending experience points to increase skill levels. Could an agent identify for itself what rewards it should pursue? Will a curriculum-based RL approach yield an agent that can complete the game?

- **Execution monitoring, replanning and goal reasoning:** Consider an example where an agent is executing a plan to achieve the goal of killing a monster when the agent observes a rare weapon item nearby. The agent may decide to replan to pick up that object and use it to kill the monster, but to do so would require kiting (a tactic where the player keeps an enemy chasing them while also keeping it at a range where it cannot attack the player) the monster around an obstacle to reach the item without being attacked first. Can we build agents capable of reasoning about goals and plans in an environment such as DCSS that could lead to such behavior?

Conclusion

DCSS is an excellent evaluation testbed for many AI problems and is supported by an active community of players and developers. We describe properties of DCSS that warrant its consideration as an evaluation testbed, particularly because it is partially observable, dynamic, stochastic, and requires a variety of decision making capabilities to win a game. We describe the vector and symbolic relational state representations provided by *dcss-ai-wrapper*, as well as the first PDDL domain file for Dungeon Crawl Stone Soup.

A roadmap for future work can be found in the repository. It includes adding support for more complicated actions such as spells, extending the API to run on Linux using the terminal interface of crawl, add sensing actions in the PDDL domain model, and more.

Acknowledgements

The views, opinions and/or findings expressed are those of the authors and should not be interpreted as representing the official views or policies of the Department of Defense or the U.S. Government.

Notes

1. https://github.com/crawl/crawl
2. This comment was posted on October 24, 2018: https://www.reddit.com/r/dcss/comments/9qzfmy/vavp_mibe_my_first_win_after_3_years/
3. Additionally, the lower bound complexity analysis we give here considers all levels together as a single state. AI agents will likely use abstractions over the state space to reduce complexity, or consider only a single level at a time.
4. The current version of our API supports DCSS v0.26, which is the latest stable version of DCSS at the time of this writing
5. https://github.com/dtdannen/dcss-ai-wrapper
6. http://crawl.chaosforge.org/Dungeon_Sprint
7. The latest documentation on vector and PDDL State Representations can be found here: https://dcss-ai-wrapper.readthedocs.io/en/latest/usage/state_representation.html
8. https://www.nethack.org/
9. https://news.ycombinator.com/item?id=602808
10. https://github.com/elliptic/qw
11. https://www.reddit.com/r/nethack/comments/2tluxv/yaap_fullauto_bot_ascension_bothack/
12. Exceptions, for example, include the broader intent of XAI Workshops at the IJCAI, ICCBR, and ICAPS conferences.

References

Asperti, A.; Cortesi, D.; De Pieri, C.; Pedrini, G.; and Sovrano, F. 2019. Crawling in Rogue's Dungeons With Deep Reinforcement Techniques. *IEEE Transactions on Games* 12(2): 177–186.

Bidot, J.; Biundo, S.; Heinroth, T.; Minker, W.; Nothdurft, F.; and Schattenberg, B. 2010. Verbal Plan Explanations for Hybrid Planning. In *Proceedings of Multikonferenz Wirtschaftsinformatik*, 2309–2320.

Chakraborti, T.; Sreedharan, S.; Zhang, Y.; and Kambhampati, S. 2017. Plan explanations as model reconciliation: Moving beyond explanation as soliloquy. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 156–163.

Fox, M.; Long, D.; and Magazzeni, D. 2017. Explainable Planning. In *Explainable Artificial Intelligence: Papers from the IJCAI Workshop*.

Helmert, M. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research* 26: 191–246.

Johnson, M.; Hofmann, K.; Hutton, T.; and Bignell, D. 2016. The Malmo Platform for Artificial Intelligence Experimentation. In *IJCAI*, 4246–4247.

Küttler, H.; Nardelli, N.; Miller, A. H.; Raileanu, R.; Selvatici, M.; Grefenstette, E.; and Rocktäschel, T. 2020. The nethack learning environment. *arXiv preprint arXiv:2006.13760* .

Langley, P.; Meadows, B.; Sridharan, M.; and Choi, D. 2017. Explainable Agency for Intelligent Autonomous Systems. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 4762–4764. AAAI Press.

McDermott, D.; Ghallab, M.; Howe, A.; Knoblock, C.; Ram, A.; Veloso, M.; Weld, D.; and Wilkins, D. 1998. PDDL-the planning domain definition language .

Ontanón, S.; Synnaeve, G.; Uriarte, A.; Richoux, F.; Churchill, D.; and Preuss, M. 2013. A survey of real-time strategy game AI research and competition in StarCraft. *IEEE Transactions on Computational Intelligence and AI in games* 5(4): 293–311.

PleasingFungus. 2016. Dungeon Crawl Stone Soup Tournament Results v 0.18. https://crawl.develz.org/wordpress/0-18-tournament-results. Accessed: 2017-09-13.

Smith, D. E. 2012. Planning as an Iterative Process. In *Proceedings of the 26th AAAI Conference on Artificial Intelligence*, 2180–2185. AAAI Press.

Sohrabi, S.; Baier, J. A.; and McIlraith, S. A. 2011. Preferred Explanations: Theory and Generation via Planning. In *Proceedings of the 25th AAAI Conference on Artificial Intelligence*. AAAI Press.

Steinkraus, K.; and Kaelbling, L. P. 2004. Combining dynamic abstractions in large MDPs .

Tian, Y.; Gong, Q.; Shang, W.; Wu, Y.; and Zitnick, C. L. 2017. ELF: An Extensive, Lightweight and Flexible Research Platform for Real-time Strategy Games. *Advances in Neural Information Processing Systems (NIPS)* .

Vinyals, O.; Ewalds, T.; Bartunov, S.; Georgiev, P.; Vezhnevets, A. S.; Yeo, M.; Makhzani, A.; Küttler, H.; Agapiou, J.; Schrittwieser, J.; et al. 2017. StarCraft II: a new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782* .

Winder, J.; and desJardins, M. 2018. Concept-Aware Feature Extraction for Knowledge Transfer in Reinforcement Learning. In *Knowledge Extraction from Games Workshop at AAAI-18*.

Appendix A: STRIPS-level PDDL Domain File for Dungeon Crawl Stone Soup

```
;;; File: models/fastdownward_simple.pddl
;;;
;;; Simple domain representation for dungeon crawl stone soup compatible with
;;; the fastdownward planner and other pddl planning systems.
;;;
;;; Author: Dustin Dannenhauer
;;; Email: dannenhauerdustin@gmail.com
;;;
;;; Notes:
;;;     0. This domain file is not meant to accurately represent dungeon crawl
;;;        stone soup. Rather it is meant as a low fidelity approximation
;;;        of the real environment that is meant to be incorporated into a
;;;        planning system embedded in an agent with plan execution monitoring
;;;        and other capabilities, to produce goal-directed behavior capable of
;;;        basic reasoning about most player actions.
;;;
;;;     1. This domain file was created using the best available information
;;;        from the crawl wiki, which is not always kept up to date. Please
;;;        submit an issue on the github if any errors or inconsistencies are
;;;        found. Github: https://github.com/dtdannen/dcss-ai-wrapper
;;;        Crawl wiki: http://crawl.chaosforge.org/


(define (domain dcss)
(:requirements :strips :negative-preconditions :existential-preconditions)
(:types monster
        cell
        place ; examples: zot_4, dungeon_12, vaults_2
        skill
        ability
        spell
        god
        qualitative_quantity
        status
        mutation
        terrain
        danger_rating
        item
        rune
        status_effect
        target_ability_text_message

        non_target_based_spell - spell
        target_based_spell - spell

        non_target_ability - ability
        target_ability - ability
        target_ability_location - target_ability
        target_ability_menu - target_ability
        target_ability_text_message_choice - target_ability

        consumeitem - item
        equipitem - item
        potion - consumeitem
        scroll - consumeitem
```

```
        weapon - equipitem
        armour - equipitem
)

(:constants
;; background objects

  none - qualitative_quantity
  low - qualitative_quantity
  medium_low - qualitative_quantity
  medium - qualitative_quantity
  medium_high - qualitative_quantity
  high - qualitative_quantity
  maxed - qualitative_quantity

  serpentine_rune - rune
  decaying_rune - rune
  barnacled_rune - rune
  gossamer_rune - rune
  abyssal_rune - rune
  silver_rune - rune
  slimy_rune - rune
  dark_rune - rune
  glowing_rune - rune
  fiery_rune - rune
  magical_rune - rune
  demonic_rune - rune
  golden_rune - rune
  iron_rune - rune
  icy_rune - rune
  obsidian_rune - rune
  bone_rune - rune

  shallow_water - terrain
  deep_water - terrain
  lava - terrain
  rock_wall - terrain
  translucent_rock_wall - terrain
  green_crystal_wall - terrain
  stone_wall - terrain
  translucent_stone_wall - terrain
  metal_wall - terrain
  unnaturally_hard_wall - terrain
  bush - terrain
  fungus - terrain
  plant - terrain
  trees - terrain

  easy - danger_rating
  dangerous - danger_rating
  very_dangerous - danger_rating

  abomination - monster
  acid_blob - monster
  acid_dragon - monster
  adder - monster
  agate_snail - monster
  agnes - monster
  air_elemental - monster
```

aizul - monster
alligator - monster
alligator_snapping_turtle - monster
anaconda - monster
ancient_bear - monster
ancient_champion - monster
ancient_lich - monster
ancient_zyme - monster
angel - monster
ant_larva - monster
antaeus - monster
anubis_guard - monster
apis - monster
apocalypse_crab - monster
arachne - monster
archer_statue - monster
asmodeus - monster
asterion - monster
azrael - monster
azure_jelly - monster
baby_alligator - monster
bai_suzhen - monster
ball_lightning - monster
ball_python - monster
ballistomycete - monster
ballistomycete_spore - monster
balrug - monster
barachi_monster - monster
basilisk - monster
bat - monster
battlesphere - monster
bennu - monster
big_fish - monster
big_kobold - monster
black_bear - monster
black_draconian - monster
black_mamba - monster
black_sun - monster
blink_frog - monster
blizzard_demon - monster
bloated_husk - monster
block_of_ice - monster
blood_saint - monster
blork_the_orc - monster
blue_death - monster
blue_devil - monster
bog_body - monster
bog_mummy - monster
boggart - monster
bone_dragon - monster
boring_beetle - monster
boris - monster
boulder_beetle - monster
brain_worm - monster
briar_patch - monster
brimstone_fiend - monster
brown_ooze - monster
bullfrog - monster
bumblebee - monster

bunyip - monster
burning_bush - monster
butterfly - monster
cacodemon - monster
cane_toad - monster
catoblepas - monster
caustic_shrike - monster
centaur_monster - monster
centaur_warrior - monster
cerebov - monster
chaos_butterfly - monster
chaos_champion - monster
chaos_spawn - monster
charred_statue - monster
cherub - monster
chimera - monster
chuck - monster
clay_golem - monster
conjurer_statue - monster
corrupter - monster
crawling_corpse - monster
crazy_yiuf - monster
crimson_imp - monster
crocodile - monster
crystal_golem - monster
crystal_guardian - monster
curse_skull - monster
curse_toe - monster
cyclops - monster
daeva - monster
dancing_weapon - monster
dart_slug - monster
death_cob - monster
death_drake - monster
death_knight - monster
death_ooze - monster
death_scarab - monster
death_yak - monster
deathcap - monster
deep_dwarf_monster - monster
deep_dwarf_artificer - monster
deep_dwarf_berserker - monster
deep_dwarf_death_knight - monster
deep_dwarf_necromancer - monster
deep_dwarf_scion - monster
deep_elf_annihilator - monster
deep_elf_archer - monster
deep_elf_blademaster - monster
deep_elf_conjurer - monster
deep_elf_death_mage - monster
deep_elf_demonologist - monster
deep_elf_elementalist - monster
deep_elf_fighter - monster
deep_elf_high_priest - monster
deep_elf_knight - monster
deep_elf_mage - monster
deep_elf_master_archer - monster
deep_elf_priest - monster
deep_elf_soldier - monster

deep_elf_sorcerer - monster
deep_elf_summoner - monster
deep_troll - monster
deep_troll_earth_mage - monster
deep_troll_shaman - monster
demigod_monster - monster
demon - monster
demonic_crawler - monster
demonic_monsters - monster
demonspawn_monster - monster
derived_undead - monster
diamond_obelisk - monster
dire_elephant - monster
dispater - monster
dissolution - monster
donald - monster
doom_hound - monster
dowan - monster
draconian_monster - monster
draconian_annihilator - monster
draconian_knight - monster
draconian_monk - monster
draconian_scorcher - monster
draconian_shifter - monster
draconian_stormcaller - monster
draconian_zealot - monster
dream_sheep - monster
drowned_soul - monster
dryad - monster
duane - monster
duvessa - monster
dwarf - monster
earth_elemental - monster
edmund - monster
efreet - monster
eidolon - monster
eldritch_tentacle - monster
electric_eel - monster
electric_golem - monster
eleionoma - monster
elemental_wellspring - monster
elephant - monster
elephant_slug - monster
elf - monster
emperor_scorpion - monster
entropy_weaver - monster
ereshkigal - monster
erica - monster
erolcha - monster
ettin - monster
eustachio - monster
executioner - monster
eye_of_devastation - monster
eye_of_draining - monster
fannar - monster
faun - monster
felid_monster - monster
fenstrider_witch - monster
fire_bat - monster

fire_crab - monster
fire_dragon - monster
fire_drake - monster
fire_elemental - monster
fire_giant - monster
fire_vortex - monster
firespitter_statue - monster
flaming_corpse - monster
flayed_ghost - monster
floating_eye - monster
flying_skull - monster
formicid_monster - monster
formicid_drone - monster
formicid_venom_mage - monster
frances - monster
francis - monster
frederick - monster
freezing_wraith - monster
frilled_lizard - monster
frost_giant - monster
frost-covered_statue - monster
gargoyle_monster - monster
gastronok - monster
gelid_demonspawn - monster
geryon - monster
ghost_crab - monster
ghost_moth - monster
ghoul_monster - monster
giant_amoeba - monster
giant_blowfly - monster
giant_centipede - monster
giant_cockroach - monster
giant_firefly - monster
giant_goldfish - monster
giant_mite - monster
giant_slug - monster
giant_toad - monster
gila_monster - monster
gloorx_vloq - monster
glowing_orange_brain - monster
glowing_shapeshifter - monster
gnoll_monster - monster
gnoll_sergeant - monster
gnoll_shaman - monster
goblin - monster
golden_dragon - monster
golden_eye - monster
goliath_beetle - monster
goliath_frog - monster
grand_avatar_monster - monster
great_orb_of_eyes - monster
green_death - monster
green_draconian - monster
grey_draconian - monster
grey_rat - monster
griffon - monster
grinder - monster
grizzly_bear - monster
grum - monster

guardian_golem - monster
guardian_mummy - monster
guardian_naga - monster
guardian_serpent - monster
hairy_devil - monster
halazid_warlock - monster
halfling_monster - monster
harold - monster
harpy - monster
hell_beast - monster
hell_hog - monster
hell_hound - monster
hell_knight - monster
hell_rat - monster
hell_sentinel - monster
hellephant - monster
hellion - monster
hellwing - monster
hill_giant - monster
hippogriff - monster
hobgoblin - monster
hog - monster
holy_swine - monster
hornet - monster
hound - monster
howler_monkey - monster
human_monster - monster
hungry_ghost - monster
hydra - monster
ice_beast - monster
ice_devil - monster
ice_dragon - monster
ice_fiend - monster
ice_statue - monster
ignacio - monster
ignis - monster
iguana - monster
ijyb - monster
ilsuiw - monster
imperial_myrmidon - monster
inept_mimic - monster
infernal_demonspawn - monster
insubstantial_wisp - monster
iron_devil - monster
iron_dragon - monster
iron_elemental - monster
iron_giant - monster
iron_golem - monster
iron_imp - monster
iron_troll - monster
ironbrand_convoker - monster
ironheart_preserver - monster
jackal - monster
jelly - monster
jellyfish - monster
jessica - monster
jiangshi - monster
jorgrun - monster
jory - monster

joseph - monster
josephine - monster
jozef - monster
juggernaut - monster
jumping_spider - monster
khufu - monster
killer_bee - monster
killer_bee_larva - monster
killer_klown - monster
kirke - monster
kobold_monster - monster
kobold_demonologist - monster
komodo_dragon - monster
kraken - monster
laboratory_rat - monster
lamia - monster
large_abomination - monster
lasher_statue - monster
lava_fish - monster
lava_snake - monster
lava_worm - monster
lemure - monster
leopard_gecko - monster
lich - monster
lightning_spire - monster
lindwurm - monster
list_of_monsters - monster
lom_lobon - monster
lorocyproca - monster
lost_soul - monster
louise - monster
lurking_horror - monster
macabre_mass - monster
maggie - monster
mana_viper - monster
manticore - monster
mara - monster
margery - monster
master_blaster - monster
master_elementalist - monster
maud - monster
maurice - monster
meliai - monster
menkaure - monster
mennas - monster
merfolk - monster
merfolk_monster - monster
merfolk_aquamancer - monster
merfolk_avatar - monster
merfolk_impaler - monster
merfolk_javelineer - monster
merfolk_siren - monster
mermaid - monster
metal_gargoyle - monster
michael - monster
midge - monster
mimic_monster - monster
minotaur_monster - monster
mnoleg - monster

molten_gargoyle - monster
monster_attributes - monster
monster_generation - monster
monsters - monster
monstrous_demonspawn - monster
moth_of_suppression - monster
moth_of_wrath - monster
mottled_draconian - monster
mottled_dragon - monster
mummy_monster - monster
mummy_priest - monster
murray - monster
naga_monster - monster
naga_mage - monster
naga_ritualist - monster
naga_sharpshooter - monster
naga_warrior - monster
nagaraja - monster
nameless_horror - monster
natasha - monster
necromancer_monster - monster
necrophage - monster
nellie - monster
neqoxec - monster
nergalle - monster
nessos - monster
nikola - monster
norbert - monster
norris - monster
obsidian_statue - monster
octopode_monster - monster
octopode_crusher - monster
ogre_monster - monster
ogre_mage - monster
oklob_plant - monster
oklob_sapling - monster
ooze - monster
ophan - monster
orange_crystal_statue - monster
orange_demon - monster
orb_guardian - monster
orb_of_fire - monster
orb_spider - monster
orc - monster
orc_high_priest - monster
orc_knight - monster
orc_priest - monster
orc_sorcerer - monster
orc_warlord - monster
orc_warrior - monster
orc_wizard - monster
paladin_monster - monster
pale_draconian - monster
pan_monster - monster
pandemonium_lord - monster
peacekeeper - monster
pearl_dragon - monster
phantasmal_warrior - monster
phantom - monster

phoenix - monster
pikel - monster
pillar_of_salt - monster
pit_fiend - monster
plague_shambler - monster
polar_bear - monster
polymoth - monster
polyphemus - monster
porcupine - monster
priests - monster
prince_ribbit - monster
profane_servitor - monster
psyche - monster
pulsating_lump - monster
purgy - monster
purple_draconian - monster
putrid_demonspawn - monster
quasit - monster
queen_ant - monster
queen_bee - monster
quicksilver_dragon - monster
quokka - monster
ragged_hierophant - monster
raiju - monster
rakshasa - monster
rat - monster
raven - monster
ravenous_mimic - monster
reaper - monster
red_devil - monster
red_draconian - monster
redback - monster
revenant - monster
rime_drake - monster
river_rat - monster
robin - monster
rock_troll - monster
rock_worm - monster
rotting_devil - monster
rotting_hulk - monster
roxanne - monster
royal_mummy - monster
rupert - monster
rust_devil - monster
saint_roka - monster
salamander - monster
salamander_firebrand - monster
salamander_mystic - monster
salamander_stormcaller - monster
saltling - monster
satyr - monster
scorpion - monster
sea_snake - monster
seraph - monster
serpent_of_hell_cocytus - monster
serpent_of_hell_dis - monster
serpent_of_hell_gehenna - monster
serpent_of_hell_tartarus - monster
servant_of_whispers - monster

shadow - monster
shadow_demon - monster
shadow_dragon - monster
shadow_imp - monster
shadow_wraith - monster
shambling_mangrove - monster
shapeshifter - monster
shard_shrike - monster
shark - monster
shedu - monster
sheep - monster
shining_eye - monster
shock_serpent - monster
sigmund - monster
silent_spectre - monster
silver_star - monster
silver_statue - monster
simulacrum_monster - monster
sixfirhy - monster
skeletal_warrior - monster
skeleton_monster - monster
sky_beast - monster
slave - monster
slime_creature - monster
small_abomination - monster
smoke_demon - monster
snail_statue - monster
snaplasher_vine - monster
snapping_turtle - monster
snorg - monster
sojobo - monster
soldier_ant - monster
sonja - monster
soul_eater - monster
spark_wasp - monster
spatial_maelstrom - monster
spatial_vortex - monster
spectral_thing - monster
spellforged_servitor_monster - monster
sphinx - monster
spider - monster
spiny_worm - monster
spirit - monster
spirit_wolf - monster
spooky_statue - monster
spriggan_monster - monster
spriggan_air_mage - monster
spriggan_assassin - monster
spriggan_berserker - monster
spriggan_defender - monster
spriggan_druid - monster
spriggan_enchanter - monster
spriggan_rider - monster
starcursed_mass - monster
steam_dragon - monster
stone_giant - monster
stone_golem - monster
storm_dragon - monster
subtractor_snake - monster

sun_demon - monster
swamp_dragon - monster
swamp_drake - monster
swamp_worm - monster
tarantella - monster
template:monster_info - monster
tengu_monster - monster
tengu_conjurer - monster
tengu_reaver - monster
tengu_warrior - monster
tentacled_monstrosity - monster
tentacled_starspawn - monster
terence - monster
terpsichore - monster
test_spawner - monster
the_enchantress - monster
the_iron_giant - monster
the_lernaean_hydra - monster
the_royal_jelly - monster
thorn_hunter - monster
thorn_lotus - monster
thrashing_horror - monster
tiamat - monster
titan - monster
toadstool - monster
toenail_golem - monster
tormentor - monster
torpor_snail - monster
torturous_demonspawn - monster
training_dummy - monster
troll - monster
troll_monster - monster
twister - monster
two-headed_ogre - monster
tyrant_leech - monster
tzitzimitl - monster
ufetubus - monster
ugly_thing - monster
unborn - monster
unborn_deep_dwarf - monster
unseen_horror - monster
urug - monster
ushabti - monster
vampire_monster - monster
vampire_bat - monster
vampire_knight - monster
vampire_mage - monster
vampire_mosquito - monster
vapour - monster
vashnia - monster
vault_guard - monster
vault_sentinel - monster
vault_warden - monster
very_ugly_thing - monster
vine_stalker_monster - monster
viper - monster
wandering_mushroom - monster
war_dog - monster
war_gargoyle - monster

warg - monster
warmonger - monster
wasp - monster
water_elemental - monster
water_moccasin - monster
water_nymph - monster
white_draconian - monster
white_imp - monster
wight - monster
wiglaf - monster
will-o-the-wisp - monster
wind_drake - monster
wizard_monster - monster
wizard_statue - monster
wolf - monster
wolf_spider - monster
wood_golem - monster
worker_ant - monster
worldbinder - monster
worm - monster
wraith - monster
wretched_star - monster
wyvern - monster
xtahua - monster
yak - monster
yaktaur - monster
yaktaur_captain - monster
yellow_draconian - monster
ynoxinul - monster
zombie - monster
zot_statue - monster

agile_status - status_effect
antimagic_status - status_effect
augmentation_status - status_effect
bad_forms_status - status_effect
berserk_status - status_effect
black_mark_status - status_effect
blind_status - status_effect
brilliant_status - status_effect
charm_status - status_effect
confusing_touch_status - status_effect
confusion_status - status_effect
constriction_status - status_effect
cooldowns_status - status_effect
corona_status - status_effect
corrosion_status - status_effect
darkness_status - status_effect
dazed_status - status_effect
death_channel_status - status_effect
deaths_door_status - status_effect
deflect_missiles_status - status_effect
disjunction_status - status_effect
divine_protection_status - status_effect
divine_shield_status - status_effect
doom_howl_status - status_effect
drain_status - status_effect
engorged_status - status_effect

engulf_status - status_effect
fast_slow_status - status_effect
fear_status - status_effect
finesse_status - status_effect
fire_vulnerable_status - status_effect
flayed_status - status_effect
flight_status - status_effect
frozen_status - status_effect
haste_status - status_effect
heavenly_storm_status - status_effect
held_status - status_effect
heroism_status - status_effect
horrified_status - status_effect
inner_flame_status - status_effect
invisibility_status - status_effect
in_lava_status - status_effect
ledas_liquefaction_status - status_effect
magic_contamination_status - status_effect
mark_status - status_effect
mesmerised_status - status_effect
might_status - status_effect
mirror_damage_status - status_effect
no_potions_status - status_effect
no_scrolls_status - status_effect
olgrebs_toxic_radiance_status - status_effect
orb_status - status_effect
ozocubus_armour_status - status_effect
paralysis_status - status_effect
petrifying_or_petrified_status - status_effect
poison_status - status_effect
powered_by_death_status - status_effect
quad_damage_status - status_effect
recall_status - status_effect
regenerating_status - status_effect
repel_missiles_status - status_effect
resistance_status_effect_status - status_effect
ring_of_flames_status - status_effect
sapped_magic_status - status_effect
scrying_status - status_effect
searing_ray_status - status_effect
serpents_lash_status - status_effect
shroud_of_golubria_status - status_effect
sickness_status - status_effect
silence_status - status_effect
sleep_status - status_effect
slimify_status - status_effect
slow_status - status_effect
sluggish_status - status_effect
starving_status - status_effect
stat_zero_status - status_effect
sticky_flame_status - status_effect
still_winds_status - status_effect
swiftness_status - status_effect
teleport_status - status_effect
teleport_prevention_status - status_effect
tornado_status - status_effect
transmutations_status - status_effect
umbra_status - status_effect
vitalisation_status - status_effect

vulnerable_status - status_effect
water_status - status_effect
weak_status - status_effect

acute_vision - mutation
antennae - mutation
beak - mutation
big_wings - mutation
blink - mutation
camouflage - mutation
clarity - mutation
claws - mutation
cold_resistance - mutation
electricity_resistance - mutation
evolution - mutation
fangs - mutation
fire_resistance - mutation
high_mp - mutation
hooves - mutation
horns - mutation
icy_blue_scales - mutation
improved_attributes - mutation
iridescent_scales - mutation
large_bone_plates - mutation
magic_resistance - mutation
molten_scales - mutation
mutation_resistance - mutation
passive_mapping - mutation
poison_breath - mutation
poison_resistance - mutation
regeneration - mutation
repulsion_field - mutation
robust - mutation
rugged_brown_scales - mutation
shaggy_fur - mutation
slimy_green_scales - mutation
stinger - mutation
strong_legs - mutation
talons - mutation
tentacle_spike - mutation
thin_metallic_scales - mutation
thin_skeletal_structure - mutation
tough_skin - mutation
wild_magic - mutation
yellow_scales - mutation

ashenzari - god
beogh - god
cheibriados - god
dithmenos - god
elyvilon - god
fedhas - god
gozag - god
hepliaklqana - god
jiyva - god
kikubaaqudgha - god
lugonu - god
makhleb - god
nemelex - god

okawaru - god
qazlal - god
ru - god
sif - god
trog - god
uskayaw - god
vehumet - god
wu_jian - god
xom - god
yredelemnul - god
zin - god
shining_one - god

unknown - god

ambrosia_potion - potion
berserkrage_potion - potion
brilliance_potion - potion
cancellation_potion - potion
curing_potion - potion
degeneration_potion - potion
experience_potion - potion
flight_potion - potion
haste_potion - potion
healwounds_potion - potion
invisibility_potion - potion
lignification_potion - potion
magic_potion - potion
might_potion - potion
mutation_potion - potion
resistance_potion - potion
stabbing_potion - potion

acquirement_scroll - scroll
amnesia_scroll - scroll
blinking_scroll - scroll
brandweapon_scroll - scroll
enchantarmour_scroll - scroll
enchantweapon_scroll - scroll
fear_scroll - scroll
fog_scroll - scroll
holyword_scroll - scroll
identity_scroll - scroll
immolation_scroll - scroll
magicmapping_scroll - scroll
noise_scroll - scroll
randomuselessness_scroll - scroll
removecurse_scroll - scroll
silence_scroll - scroll
summoning_scroll - scroll
teleportation_scroll - scroll
torment_scroll - scroll
vulnerability_scroll - scroll

alistairs_intoxication_spell - non_target_based_spell
animate_dead_spell - non_target_based_spell
animate_skeleton_spell - non_target_based_spell
aura_of_abjuration_spell - non_target_based_spell
beastly_appendage_spell - non_target_based_spell

blade_hands_spell - non_target_based_spell
blink_spell - non_target_based_spell
borgnjors_revivification_spell - non_target_based_spell
call_canine_familiar_spell - non_target_based_spell
call_imp_spell - non_target_based_spell
chain_lightning_spell - non_target_based_spell
confusing_touch_spell - non_target_based_spell
conjure_ball_lightning_spell - non_target_based_spell
conjure_flame_spell - non_target_based_spell
controlled_blink_spell - non_target_based_spell
corpse_rot_spell - non_target_based_spell
death_channel_spell - non_target_based_spell
deaths_door_spell - non_target_based_spell
discord_spell - non_target_based_spell
disjunction_spell - non_target_based_spell
dragon_form_spell - non_target_based_spell
dragons_call_spell - non_target_based_spell
eringyas_noxious_bog_spell - non_target_based_spell
excruciating_wounds_spell - non_target_based_spell
foxfire_spell - non_target_based_spell
hydra_form_spell - non_target_based_spell
ice_form_spell - non_target_based_spell
ignite_poison_spell - non_target_based_spell
ignition_spell - non_target_based_spell
infusion_spell - non_target_based_spell
iskenderuns_battlesphere_spell - non_target_based_spell
ledas_liquefaction_spell - non_target_based_spell
malign_gateway_spell - non_target_based_spell
metabolic_englaciation_spell - non_target_based_spell
monstrous_menagerie_spell - non_target_based_spell
necromutation_spell - non_target_based_spell
olgrebs_toxic_radiance_spell - non_target_based_spell
ozocubus_armour_spell - non_target_based_spell
ozocubus_refrigeration_spell - non_target_based_spell
portal_projectile_spell - non_target_based_spell
recall_spell - non_target_based_spell
ring_of_flames_spell - non_target_based_spell
shadow_creatures_spell - non_target_based_spell
shatter_spell - non_target_based_spell
shroud_of_golubria_spell - non_target_based_spell
silence_spell_spell - non_target_based_spell
simulacrum_spell - non_target_based_spell
song_of_slaying_spell - non_target_based_spell
spectral_weapon_spell - non_target_based_spell
spellforged_servitor_spell - non_target_based_spell
spider_form_spell - non_target_based_spell
statue_form_spell - non_target_based_spell
sticks_to_snakes_spell - non_target_based_spell
sublimation_of_blood_spell - non_target_based_spell
summon_demon_spell - non_target_based_spell
summon_forest_spell - non_target_based_spell
summon_greater_demon_spell - non_target_based_spell
summon_guardian_golem_spell - non_target_based_spell
summon_horrible_things_spell - non_target_based_spell
summon_hydra_spell - non_target_based_spell
summon_ice_beast_spell - non_target_based_spell
summon_mana_viper_spell - non_target_based_spell
summon_small_mammal_spell - non_target_based_spell
swiftness_spell - non_target_based_spell

absolute_zero_spell - target_based_spell
agony_spell - target_based_spell
airstrike_spell - target_based_spell
apportation_spell - target_based_spell
bolt_of_magma_spell - target_based_spell
borgnjors_vile_clutch_spell - target_based_spell
cause_fear_spell - target_based_spell
corona_spell - target_based_spell
dazzling_flash_spell - target_based_spell
dispel_undead_spell - target_based_spell
dispersal_spell - target_based_spell
ensorcelled_hibernation_spell - target_based_spell
fire_storm_spell - target_based_spell
fireball_spell - target_based_spell
freeze_spell - target_based_spell
freezing_cloud_spell - target_based_spell
frozen_ramparts_spell - target_based_spell
fulminant_prism_spell - target_based_spell
gells_gravitas_spell - target_based_spell
hailstorm_spell - target_based_spell
haunt_spell - target_based_spell
infestation_spell - target_based_spell
inner_flame_spell - target_based_spell
invisibility_spell_spell - target_based_spell
iron_shot_spell - target_based_spell
irradiate_spell - target_based_spell
iskenderuns_mystic_blast_spell - target_based_spell
lees_rapid_deconstruction_spell - target_based_spell
lehudibs_crystal_spear_spell - target_based_spell
lesser_beckoning_spell - target_based_spell
lightning_bolt_spell - target_based_spell
magic_dart_spell - target_based_spell
mephitic_cloud_spell - target_based_spell
orb_of_destruction_spell - target_based_spell
pain_spell - target_based_spell
passage_of_golubria_spell - target_based_spell
passwall_spell - target_based_spell
petrify_spell - target_based_spell
poisonous_vapours_spell - target_based_spell
sandblast_spell - target_based_spell
searing_ray_spell - target_based_spell
shock_spell - target_based_spell
slow_spell - target_based_spell
starburst_spell - target_based_spell
static_discharge_spell - target_based_spell
sticky_flame_spell - target_based_spell
sting_spell - target_based_spell
stone_arrow_spell - target_based_spell
summon_lightning_spire_spell - target_based_spell
teleport_other_spell - target_based_spell
tornado_spell - target_based_spell
tukimas_dance_spell - target_based_spell
vampiric_draining_spell - target_based_spell
yaras_violent_unravelling_spell - target_based_spell

fighting - skill
long_blades - skill
short_blades - skill
axes - skill

maces_&_flails - skill
polearms - skill
staves - skill
unarmed_combat - skill
bows - skill
crossbows - skill
throwing - skill
slings - skill
armour - skill
dodging - skill
shields - skill
spellcasting - skill
conjurations - skill
hexes - skill
charms - skill
summonings - skill
necromancy - skill
translocations - skill
transmutation - skill
fire_magic - skill
ice_magic - skill
air_magic - skill
earth_magic - skill
poison_magic - skill
invocations - skill
evocations - skill
stealth - skill

apocalypse_ability - non_target_ability
banish_self_ability - non_target_ability
bat_form_ability - non_target_ability
bend_space_ability - non_target_ability
bend_time_ability - non_target_ability
berserk_ability - non_target_ability
blink_ability - non_target_ability
briar_patch_ability - non_target_ability
bribe_branch_ability - non_target_ability
brothers_in_arms_ability - non_target_ability
call_merchant_ability - non_target_ability
channel_magic_ability - non_target_ability
cleansing_flame_ability - non_target_ability
corrupt_ability - non_target_ability
cure_bad_mutations_ability - non_target_ability
depart_abyss_ability - non_target_ability
disaster_area_ability - non_target_ability
divine_protection_ability - non_target_ability
divine_shield_ability - non_target_ability
divine_vigour_ability - non_target_ability
drain_life_ability - non_target_ability
draw_out_power_ability - non_target_ability
elemental_force_ability - non_target_ability
finesse_ability - non_target_ability
flight_ability_ability - non_target_ability
gain_random_mutations_ability - non_target_ability
greater_healing_ability - non_target_ability
grow_ballistomycete_ability - non_target_ability
grow_oklob_plant_ability - non_target_ability
heal_wounds_ability - non_target_ability
heavenly_storm_ability - non_target_ability

heroism_ability - non_target_ability
idealise_ability - non_target_ability
lesser_healing_ability - non_target_ability
purification_ability - non_target_ability
recall_ability - non_target_ability
recall_undead_slaves_ability - non_target_ability
receive_corpses_ability - non_target_ability
receive_necronomicon_ability - non_target_ability
recite_ability - non_target_ability
request_jelly_ability - non_target_ability
resurrection_ability - non_target_ability
sanctuary_ability - non_target_ability
scrying_ability - non_target_ability
serpents_lash_ability - non_target_ability
shadow_form_ability - non_target_ability
slimify_ability - non_target_ability
slouch_ability - non_target_ability
step_from_time_ability - non_target_ability
stomp_ability - non_target_ability
summon_divine_warrior_ability - non_target_ability
summon_greater_servant_ability - non_target_ability
summon_lesser_servant_ability - non_target_ability
temporal_distortion_ability - non_target_ability
toggle_divine_energy_ability - non_target_ability
toggle_injury_mirror_ability - non_target_ability
torment_ability - non_target_ability
trogs_hand_ability - non_target_ability
vitalisation_ability - non_target_ability
wall_jump_ability - non_target_ability
animate_dead - target_ability_location
animate_remains - target_ability_location
banish - target_ability_location
controlled_blink - target_ability_location
enslave_soul - target_ability_location
give_item_to_follower - target_ability_location
grand_finale - target_ability_location
heal_other - target_ability_location
hop - target_ability_location
Imprison - target_ability_location
line_pass - target_ability_location
major_destruction - target_ability_location
minor_destruction - target_ability_location
overgrow - target_ability_location
power_leap - target_ability_location
rolling_charge - target_ability_location
shadow_step - target_ability_location
smite - target_ability_location
spit_poison - target_ability_location
transference - target_ability_location
upheaval - target_ability_location
ancestor_identity - target_ability_menu
ancestor_life - target_ability_menu
brand_weapon_with_holy - target_ability_menu
brand_weapon_with_pain - target_ability_menu
corrupt_weapon - target_ability_menu
curse_item - target_ability_menu
deal_four - target_ability_menu
forget_spell - target_ability_menu
pick_a_card_any_card - target_ability_menu

```
stack_five - target_ability_menu
transfer_knowledge - target_ability_menu
triple_draw - target_ability_menu
potion_petition - target_ability_text_message

dungeon_1 - place
dungeon_2 - place
dungeon_3 - place
dungeon_4 - place
dungeon_5 - place
dungeon_6 - place
dungeon_7 - place
dungeon_8 - place
dungeon_9 - place
dungeon_10 - place
dungeon_11 - place
dungeon_12 - place
dungeon_13 - place
dungeon_14 - place
dungeon_15 - place

lair_1 - place
lair_2 - place
lair_3 - place
lair_4 - place
lair_5 - place
lair_6 - place

swamp_1 - place
swamp_2 - place
swamp_3 - place
swamp_4 - place

shoals_1 - place
shoals_2 - place
shoals_3 - place
shoals_4 - place

snake_pit_1 - place
snake_pit_2 - place
snake_pit_3 - place
snake_pit_4 - place

spiders_nest_1 - place
spiders_nest_2 - place
spiders_nest_3 - place
spiders_nest_4 - place

slime_pits_1 - place
slime_pits_2 - place
slime_pits_3 - place
slime_pits_4 - place
slime_pits_5 - place

orcish_mines_1 - place
orcish_mines_2 - place

elven_halls_1 - place
elven_halls_2 - place
```

```
elven_halls_3 - place

vaults_1 - place
vaults_2 - place
vaults_3 - place
vaults_4 - place
vaults_5 - place

crypt_1 - place
crypt_2 - place
crypt_3 - place

tomb_1 - place
tomb_2 - place
tomb_3 - place

depths_1 - place
depths_2 - place
depths_3 - place
depths_4 - place
depths_5 - place

abyss_1 - place
abyss_2 - place
abyss_3 - place
abyss_4 - place
abyss_5 - place

cocytus_1 - place
cocytus_2 - place
cocytus_3 - place
cocytus_4 - place
cocytus_5 - place
cocytus_6 - place
cocytus_7 - place

gehenna_1 - place
gehenna_2 - place
gehenna_3 - place
gehenna_4 - place
gehenna_5 - place
gehenna_6 - place
gehenna_7 - place

tartarus_1 - place
tartarus_2 - place
tartarus_3 - place
tartarus_4 - place
tartarus_5 - place
tartarus_6 - place
tartarus_7 - place

iron_city_of_dis_1 - place
iron_city_of_dis_2 - place
iron_city_of_dis_3 - place
iron_city_of_dis_4 - place
iron_city_of_dis_5 - place
iron_city_of_dis_6 - place
iron_city_of_dis_7 - place
```

```
  zot_1 - place
  zot_2 - place
  zot_3 - place
  zot_4 - place
  zot_5 - place

) ;; end constants


(:predicates
    ; N,S,E,W,NE,NW,SE,SW of a cell
    (northof ?cell1 ?cell2 - cell) ; ?cell2 is north of ?cell1
    (southof ?cell1 ?cell2 - cell)
    (eastof ?cell1 ?cell2 - cell)
    (westof ?cell1 ?cell2 - cell)
    (northeastof ?cell1 ?cell2 - cell)
    (northwestof ?cell1 ?cell2 - cell)
    (southeastof ?cell1 ?cell2 - cell)
    (southwestof ?cell1 ?cell2 - cell)

    (opendoor ?cell - cell)
    (closeddoor ?cell - cell)

    (statue ?cell - cell)

    (hasterrain ?cell - cell ?terrain - terrain)

    ;altars enable worshipping a god
    (altarat ?cell - cell ?god - god)
    ; player god
    (player_worshipping ?god - god)
    (player_piety ?amount - qualitative_quantity)

    ; player loc
    (playerat ?cell - cell)

    ; player health
    (playerhealth ?amount - qualitative_quantity)

    ; monster related predicates - only one monster per cell
    (hasmonster ?cell - cell)
    (monster_danger_rating ?cell - cell ?danger - danger_rating)
    (monster_health ?cell - cell ?amount - qualitative_quantity)
    (monster_status_effect ?cell - cell ?status - status)

    ; levels
    (playerplace ?place - place)
    (deeper ?place_above ?place_below - place)
    (connected ?currentplace ?nextlowestplace - place)
    (hasstairsdown ?cell - cell)
    (hasstairsup ?cell - cell)

    ; items
    (haspotion ?cell - cell)
    (hasscroll ?cell - cell)
    (hasweapon ?cell - cell)
    (hasarmour ?cell - cell)
    (hasfooditem ?cell - cell)
```

```
    (hasitem ?cell - cell ?item - item)

    ; special items
    (hasorbofzot ?cell - cell)
    (hasrune ?rune - rune ?cell - cell)

    ; special items that do not take up inventory space
    (playerhasorbofzot)
    (playerhasrune ?rune - rune)

    ; inventory
    (invhaspotion ?potion - potion)
    (invhasscroll ?scroll - scroll)
    (invhasarmour ?armour - armour)
    (invhasweapon ?weapon - weapon)
    (invhasitem ?item - item)

    ; what is equipped on the player
    (equippedarmour ?armour - armour)
    (equippedweapon ?weapon - weapon)

    ; placeholders for the effects of potions and scrolls
    ; these placeholders signify that the potion has some effect on the player
    ; and is useful when the player's goal is to consume an unidentified
    ; potion or scroll, usually in an attempt to either (1) identify the item or
    ; (2) because they are in a dire situation and are desparate for any help
    (has_generic_potion_effect ?potion - potion)
    (has_generic_scroll_effect ?scroll - scroll)
    (has_generic_spell_effect ?spell - spell)
    (has_generic_ability_effect ?ability - ability)

    ; skills are how the player allocates experience levels
    (training_off ?skill - skill)
    (training_low ?skill - skill)
    (training_high ?skill - skill)
    (player_skill_level ?skill - skill ?amount - qualitative_quantity)

    ; spells
    (player_memorised_spell ?spell - spell)
    (spell_chance_of_failure ?spell - spell ?amount - qualitative_quantity)
    (spell_available_to_memorise ?spell - spell)

    ;abilities
    (player_has_ability ?ability - ability)
    (ability_chance_of_failure ?ability - ability ?amount - qualitative_quantity)
)


(:action move_or_attack_s
    :parameters (?currcell ?destcell - cell)
    :precondition
    (and
        (southof ?currcell ?destcell)
        (not (hasterrain ?destcell stone_wall))
        (not (statue ?destcell))
        (not (hasterrain ?destcell lava))
        (not (hasterrain ?destcell plant))
        (not (hasterrain ?destcell trees))
        (playerat ?currcell)
```

```
    )
    :effect
    (and
        (playerat ?destcell)
        (not (playerat ?currcell))
    )
)

(:action move_or_attack_n
    :parameters (?currcell ?destcell - cell)
    :precondition
    (and
        (northof ?currcell ?destcell)
        (not (hasterrain ?destcell stone_wall))
        (not (statue ?destcell))
        (not (hasterrain ?destcell lava))
        (not (hasterrain ?destcell plant))
        (not (hasterrain ?destcell trees))
        (playerat ?currcell)
    )
    :effect
    (and
        (playerat ?destcell)
        (not (playerat ?currcell))
    )
)


(:action move_or_attack_e
    :parameters (?currcell ?destcell - cell)
    :precondition
    (and
        (eastof ?currcell ?destcell)
        (not (hasterrain ?destcell stone_wall))
        (not (statue ?destcell))
        (not (hasterrain ?destcell lava))
        (not (hasterrain ?destcell plant))
        (not (hasterrain ?destcell trees))
        (playerat ?currcell)
    )
    :effect
    (and
        (playerat ?destcell)
        (not (playerat ?currcell))
    )
)

(:action move_or_attack_w
    :parameters (?currcell ?destcell - cell)
    :precondition
    (and
        (westof ?currcell ?destcell)
        (not (hasterrain ?destcell stone_wall))
        (not (statue ?destcell))
        (not (hasterrain ?destcell lava))
        (not (hasterrain ?destcell plant))
        (not (hasterrain ?destcell trees))
        (playerat ?currcell)
    )
```

```
    :effect
    (and
        (playerat ?destcell)
        (not (playerat ?currcell))
    )
)

(:action move_or_attack_nw
    :parameters (?currcell ?destcell - cell)
    :precondition
    (and
        (northwestof ?currcell ?destcell)
        (not (hasterrain ?destcell stone_wall))
        (not (statue ?destcell))
        (not (hasterrain ?destcell lava))
        (not (hasterrain ?destcell plant))
        (not (hasterrain ?destcell trees))
        (playerat ?currcell)
    )
    :effect
    (and
        (playerat ?destcell)
        (not (playerat ?currcell))
    )
)

(:action move_or_attack_sw
    :parameters (?currcell ?destcell - cell)
    :precondition
    (and
        (southwestof ?currcell ?destcell)
        (not (hasterrain ?destcell stone_wall))
        (not (statue ?destcell))
        (not (hasterrain ?destcell lava))
        (not (hasterrain ?destcell plant))
        (not (hasterrain ?destcell trees))
        (playerat ?currcell)
    )
    :effect
    (and
        (playerat ?destcell)
        (not (playerat ?currcell))
    )
)

(:action move_or_attack_ne
    :parameters (?currcell ?destcell - cell)
    :precondition
    (and
        (northeastof ?currcell ?destcell)
        (not (hasterrain ?destcell stone_wall))
        (not (statue ?destcell))
        (not (hasterrain ?destcell lava))
        (not (hasterrain ?destcell plant))
        (not (hasterrain ?destcell trees))
        (playerat ?currcell)
    )
    :effect
    (and
```

```
        (playerat ?destcell)
        (not (playerat ?currcell))
    )
)

(:action move_or_attack_se
    :parameters (?currcell ?destcell - cell)
    :precondition
    (and
        (southeastof ?currcell ?destcell)
        (not (hasterrain ?destcell stone_wall))
        (not (statue ?destcell))
        (not (hasterrain ?destcell lava))
        (not (hasterrain ?destcell plant))
        (not (hasterrain ?destcell trees))
        (playerat ?currcell)
    )
    :effect
    (and
        (playerat ?destcell)
        (not (playerat ?currcell))
    )
)

(:action open-door-n
    :parameters (?currcell ?destcell - cell)
    :precondition
    (and
        (northof ?currcell ?destcell)
        (not (hasterrain ?destcell stone_wall))
        (not (statue ?destcell))
        (not (hasterrain ?destcell lava))
        (not (hasterrain ?destcell plant))
        (not (hasterrain ?destcell trees))
        (closeddoor ?destcell)
        (playerat ?currcell)
    )
    :effect
    (and
        (not (closeddoor ?destcell))
    )
)

(:action open-door-s
    :parameters (?currcell ?destcell - cell)
    :precondition
    (and
        (southof ?currcell ?destcell)
        (not (hasterrain ?destcell stone_wall))
        (not (statue ?destcell))
        (not (hasterrain ?destcell lava))
        (not (hasterrain ?destcell plant))
        (not (hasterrain ?destcell trees))
        (closeddoor ?destcell)
        (playerat ?currcell)
    )
    :effect
    (and
        (not (closeddoor ?destcell))
```

```
    )
)

(:action open-door-e
    :parameters (?currcell ?destcell - cell)
    :precondition
    (and
        (eastof ?currcell ?destcell)
        (not (hasterrain ?destcell stone_wall))
        (not (statue ?destcell))
        (not (hasterrain ?destcell lava))
        (not (hasterrain ?destcell plant))
        (not (hasterrain ?destcell trees))
        (closeddoor ?destcell)
        (playerat ?currcell)
    )
    :effect
    (and
        (not (closeddoor ?destcell))
    )
)

(:action open-door-w
    :parameters (?currcell ?destcell - cell)
    :precondition
    (and
        (westof ?currcell ?destcell)
        (not (hasterrain ?destcell stone_wall))
        (not (statue ?destcell))
        (not (hasterrain ?destcell lava))
        (not (hasterrain ?destcell plant))
        (not (hasterrain ?destcell trees))
        (closeddoor ?destcell)
        (playerat ?currcell)
    )
    :effect
    (and
        (not (closeddoor ?destcell))
    )
)

(:action open-door-nw
    :parameters (?currcell ?destcell - cell)
    :precondition
    (and
        (northwestof ?currcell ?destcell)
        (not (hasterrain ?destcell stone_wall))
        (not (statue ?destcell))
        (not (hasterrain ?destcell lava))
        (not (hasterrain ?destcell plant))
        (not (hasterrain ?destcell trees))
        (closeddoor ?destcell)
        (playerat ?currcell)
    )
    :effect
    (and
        (not (closeddoor ?destcell))
    )
)
```

```
(:action open-door-sw
    :parameters (?currcell ?destcell - cell)
    :precondition
    (and
        (southwestof ?currcell ?destcell)
        (not (hasterrain ?destcell stone_wall))
        (not (statue ?destcell))
        (not (hasterrain ?destcell lava))
        (not (hasterrain ?destcell plant))
        (not (hasterrain ?destcell trees))
        (closeddoor ?destcell)
        (playerat ?currcell)
    )
    :effect
    (and
        (not (closeddoor ?destcell))
    )
)

(:action open-door-ne
    :parameters (?currcell ?destcell - cell)
    :precondition
    (and
        (northeastof ?currcell ?destcell)
        (not (hasterrain ?destcell stone_wall))
        (not (statue ?destcell))
        (not (hasterrain ?destcell lava))
        (not (hasterrain ?destcell plant))
        (not (hasterrain ?destcell trees))
        (closeddoor ?destcell)
        (playerat ?currcell)
    )
    :effect
    (and
        (not (closeddoor ?destcell))
    )
)

(:action open-door-se
    :parameters (?currcell ?destcell - cell)
    :precondition
    (and
        (southeastof ?currcell ?destcell)
        (not (hasterrain ?destcell stone_wall))
        (not (statue ?destcell))
        (not (hasterrain ?destcell lava))
        (not (hasterrain ?destcell plant))
        (not (hasterrain ?destcell trees))
        (closeddoor ?destcell)
        (playerat ?currcell)
    )
    :effect
    (and
        (not (closeddoor ?destcell))
    )
)

(:action attack_without_move_n
```

```
    :parameters (?currcell ?destcell - cell)
    :precondition
    (and
        (northof ?currcell ?destcell)
        (hasmonster ?destcell)
        (playerat ?currcell)
    )
    :effect
    (and
        (not (hasmonster ?destcell))
    )
)

(:action attack_without_move_s
    :parameters (?currcell ?destcell - cell)
    :precondition
    (and
        (southof ?currcell ?destcell)
        (hasmonster ?destcell)
        (playerat ?currcell)
    )
    :effect
    (and
        (not (hasmonster ?destcell))
    )
)

(:action attack_without_move_s
    :parameters (?currcell ?destcell - cell)
    :precondition
    (and
        (southof ?currcell ?destcell)
        (hasmonster ?destcell)
        (playerat ?currcell)
    )
    :effect
    (and
        (not (hasmonster ?destcell))
    )
)

(:action attack_without_move_e
    :parameters (?currcell ?destcell - cell)
    :precondition
    (and
        (eastof ?currcell ?destcell)
        (hasmonster ?destcell)
        (playerat ?currcell)
    )
    :effect
    (and
        (not (hasmonster ?destcell))
    )
)

(:action attack_without_move_w
    :parameters (?currcell ?destcell - cell)
    :precondition
    (and
```

```
        (westof ?currcell ?destcell)
        (hasmonster ?destcell)
        (playerat ?currcell)
    )
    :effect
    (and
        (not (hasmonster ?destcell))
    )
)

(:action attack_without_move_ne
    :parameters (?currcell ?destcell - cell)
    :precondition
    (and
        (northeastof ?currcell ?destcell)
        (hasmonster ?destcell)
        (playerat ?currcell)
    )
    :effect
    (and
        (not (hasmonster ?destcell))
    )
)

(:action attack_without_move_nw
    :parameters (?currcell ?destcell - cell)
    :precondition
    (and
        (northwestof ?currcell ?destcell)
        (hasmonster ?destcell)
        (playerat ?currcell)
    )
    :effect
    (and
        (not (hasmonster ?destcell))
    )
)

(:action attack_without_move_se
    :parameters (?currcell ?destcell - cell)
    :precondition
    (and
        (southeastof ?currcell ?destcell)
        (hasmonster ?destcell)
        (playerat ?currcell)
    )
    :effect
    (and
        (not (hasmonster ?destcell))
    )
)

(:action attack_without_move_sw
    :parameters (?currcell ?destcell - cell)
    :precondition
    (and
        (southwestof ?currcell ?destcell)
        (hasmonster ?destcell)
        (playerat ?currcell)
```

```
    )
    :effect
    (and
        (not (hasmonster ?destcell))
    )
)

(:action rest_and_long_wait
    :parameters ()
    :precondition
    (and
        (not (playerhealth maxed))
    )
    :effect
    (and
        (playerhealth maxed)
    )
)

(:action travel_staircase_down
    :parameters (?currentplace ?currcell ?nextlowestplace)
    :precondition
    (and
        (playerat ?currcell)
        (hasstairsdown ?currcell)
        (playerplace ?currentplace)
        (connected ?currentplace ?nextlowestplace)
    )
    :effect
    (and
        (not (playerplace ?currentplace))
        (playerplace ?nextlowestplace)
    )
)

(:action travel_staircase_up
    :parameters (?currentplace ?currcell ?nexthighestplace)
    :precondition
    (and
        (playerat ?currcell)
        (hasstairsup ?currcell)
        (playerplace ?currentplace)
        (connected ?nexthighestplace ?currentplace)
    )
    :effect
    (and
        (not (playerplace ?currentplace))
        (playerplace ?nextlowestplace)
    )
)


(:action pickup_item
    :parameters (?item ?cell)
    :precondition
    (and
        (playerat ?cell)
        (hasitem ?cell ?item)
    )
```

```
    :effect
    (and
        (invhasitem ?item)
    )
)

(:action drop_item
    :parameters (?item ?cell)
    :precondition
    (and
        (playerat ?cell)
        (invhasitem ?item)
    )
    :effect
    (and
        (hasitem ?cell ?item)
    )
)


(:action equip_weapon
    :parameters (?weaponitem)
    :precondition
    (and
        (invhasweapon ?weaponitem)
        (not (equippedweapon ?weaponitem))
    )
    :effect
    (and
        (equippedweapon ?weaponitem)
    )
)


(:action equip_armour
    :parameters (?armouritem)
    :precondition
    (and
        (invhasarmour ?armouritem)
        (not (equippedarmour ?armouritem))
    )
    :effect
    (and
        (equippedarmour ?armouritem)
    )
)

(:action remove_weapon
    :parameters (?weaponitem)
    :precondition
    (and
        (invhasweapon ?weaponitem)
        (equippedweapon ?weaponitem)
    )
    :effect
    (and
        (not (equippedweapon ?weaponitem))
    )
)
```

```
(:action remove_armour
    :parameters (?armouritem)
    :precondition
    (and
        (invhasarmour ?armouritem)
        (equippedarmour ?armouritem)
    )
    :effect
    (and
        (not (equippedarmour ?armouritem))
    )
)


(:action consume_potion
    :parameters (?potion)
    :precondition
    (and
        (invhaspotion ?potion)
    )
    :effect
    (and
        (has_generic_potion_effect ?potion)
    )
)

(:action consume_scroll
    :parameters (?scroll)
    :precondition
    (and
        (invhasscroll ?scroll)
    )
    :effect
    (and
        (has_generic_scroll_effect ?scroll)
    )
)

(:action attack_by_throwing
    :parameters (?item ?targetcell)
    :precondition
    (and
        (invhasitem ?item)
    )
    :effect
    (and
        (not (hasmonster ?targetcell))
    )
)


(:action stop_training_skill
    :parameters (?skill - skill)
    :precondition
    (and
        (not (training_off ?skill))
        (or (training_low ?skill) (training_high ?skill))
```

```
    )
    :effect
    (and
        (training_off ?skill)
        (not (training_low ?skill))
        (not (training_high ?skill))
    )
)

(:action train_skill_low
    :parameters (?skill - skill)
    :precondition
    (and
        (not (training_low ?skill))
        (or (training_off ?skill) (training_high ?skill))
    )
    :effect
    (and
        (not (training_off ?skill))
        (training_low ?skill)
        (not (training_high ?skill))
    )
)

(:action train_skill_high
    :parameters (?skill - skill)
    :precondition
    (and
        (not (training_high ?skill))
        (or (training_off ?skill) (training_low ?skill))
    )
    :effect
    (and
        (not (training_off ?skill))
        (not (training_low ?skill))
        (training_high ?skill)
    )
)


(:action cast_spell_on_target
    :parameters (?spell - target_based_spell ?cell - cell)
    :precondition
    (and
        (player_memorised_spell ?spell)
        (hasmonster ?cell)
    )
    :effect
    (and
        (not (hasmonster ?cell))
    )
)

(:action cast_non_target_spell
    :parameters (?spell - non_target_based_spell)
    :precondition
    (and
        (player_memorised_spell ?spell)
    )
```

```
    :effect
    (and
        (has_generic_spell_effect ?spell)
    )
)

(:action use_non_target_ability
    :parameters (?ability - non_target_ability)
    :precondition
    (and
        (player_has_ability ?ability)
    )
    :effect
    (and
        (has_generic_ability_effect ?ability)
    )
)

(:action use_target_location_ability
    :parameters (?ability - target_ability_location ?cell - cell)
    :precondition
    (and
        (player_has_ability ?ability)
    )
    :effect
    (and
        (has_generic_ability_effect ?ability)
    )
)

(:action use_target_based_ability
    :parameters (?ability - target_ability_location)
    :precondition
    (and
        (player_has_ability ?ability)
    )
    :effect
    (and
        (has_generic_ability_effect ?ability)
    )
)


(:action worship_altar
    :parameters (?cell - cell ?god - god)
    :precondition
    (and
        (playerat ?cell)
        (altarat ?cell ?god)
    )
    :effect
    (and
        (player_worshipping ?god)
    )
)


)
```