Welcome to the Grid!

Introduction ▪ QuickStart ▪ Library Predicates ▪ User Shell ▪ Outline

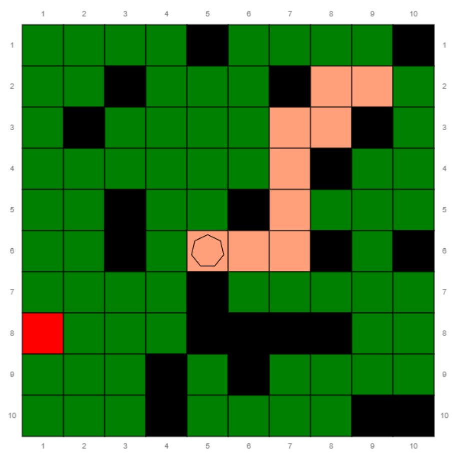
This document is an introduction to the **GridWorld** platform used in the Prolog labs of the 3rd year AI unit **COMS30014** (and associated assessments **COMS30013** and **COMS30062**).

1 Introduction

The purpose of Grid World lab platform is to provide an inspiring and **fun environment** to help you develop your **practical Prolog** programming skills and thereby obtain a deeper **conceptual understanding** of the underpinning theory. These labs are **vital for all students** because the skills and concepts you learn will be tested in **both** the **exam** and **coursework** assessments on this unit.

Formally, these Prolog labs are split into the following **three parts**, which introduce the foundational features of the GridWorld Webserver and Wikipedia library functionality:

- lab **grid** - (**week 2**) interact with an **empty grid** by writing predicates to **spiral** your agent from an outside corner into the centre.
- lab **identity** - (**week 3**) interact with **Wikipedia** by writing predicates to infer a secret actor identity using clues given by a disembodied (off-grid) oracle.
- lab **search** - (**week 7**) interact with a **non-empty grid** by writing predicates to find a **path** that allows your agent to visit an embodied (on-grid) oracle.



In these labs, you will **not** need to **understand** or even **look** at the **library code** — although you are welcome to try and do so if you wish!

These labs **assume** you have a **running version** of SWI Prolog and are developing a **basic knowledge** of logic programs obtained by working through the recommended Learn Prolog Now! tutorial. You should be starting to get comfortable **running programs** in SWIPL and **editing code** in a text editor (e.g. in PceEMacs, Atom, etc). You should also be starting to use the online manual and exploring the use of the built-in debugger.

In order develop **good coding practice** you are strongly advised to: **comment your code** to explain the meaning of each argument and the behaviour of each predicate; **format your code** to make the logical flow clear (using **informative variable and predicate names**); **test your code** to make sure it compiles **without errors** and that predicates **terminate properly** which means that wherever possible they should be **(semi-)deterministic** in the sense that they should terminate after succeeding once (or failing). You should especially try to avoid **non-termination** and **run-time exceptions** or predicates that return **duplicate solutions** or leave unnecessary **choice points**.

2 QuickStart

a) First you'll need to **install** the GridWorld library on your machine:

- **Download** the GridWorld library ZIP file
- **Extract** the GridWorld library files to a convenient location on your machine
- **Move** into the GridWorld library **root directory** in which you should see:
  - **three skeleton answer files** (`lab_grid_12345.pl`, `lab_identity_12345.pl` and `lab_search_12345.pl`) where you'll write your solutions for each lab
  - **one lab runner file** (`ailp.pl`) which is responsible for loading the library functions and solution file for each of the respective labs
- **Rename** the skeleton files by replacing `12345` with your **student number**

While this last step of renaming the skeleton files it is not strictly necessary for these labs, it is good practice, especially if you plan to go on to do the coursework assessment option (where this will be required).

BUT, if you do **rename** these files, then please make sure to use your (7-digit) **student number** and **not** your **username** (which contains letters that will disrupt the loading mechanism) **or** your **candidate number** (which should only be used when you are taking exams)!

b) Then you'll need to **invoke** one of the labs using the loader:

- **Open** a terminal window (e.g. `bash`, `cmd`, `powershell`, etc.)
- **Run** one of the labs using a command of the form `swipl ailp.pl lab` X, where X stands for one of the three Prolog terms `grid` or `identity` or `search`.

On **Linux** or **Mac** you may also be able to use slightly shorter commands like `./ailp.pl lab grid` if you make the loader **executable** using `chmod +x ailp.pl`

On **Windows** you may also be able to double click on `aipl.pl` in an `explorer` window and then enter a term like `lab grid` at the Prolog prompt. Or you could type `swipl-win ailp.pl lab grid` in a (`cmd` or `powershell`) terminal.

These instructions assume the SWI installation directory is on your system **path** (which will be the case if you follow the installation advice given in prior lectures).

c) In the `grid` and `search` labs (but **not** in the `identity` lab) you'll need to open a GridWorld webserver by running the following library commands at the Prolog prompt:

- `start.`
    - then **hit** the "y" key (or any key except "n" or "N") at the prompt to open a browser
    - and **click** the "**Run**" button at the bottom left of the resulting browser window
- `join_game(A).`
- `reset_game.`
- `start_game.`

To save pressing "y" you can instead launch the webserver with `start. .`

To save some typing you can instead run the last three commands with the command `shell.` followed by the macro `setup.`

In the grid and search labs, you'll only be able to add a single agent which will always be given the identifier `A=1`. In the identity lab, you won't be able to run a GridWorld server but will use a special agent `oscar` that exists off grid!

In order to see the "Run" button, you may need to scroll down past an initially empty space (where the grid will be subsequently drawn once a game is started)

Please note that nothing will happen below until you click "Run" in the browser!

d) To **interact with the GridWorld** use the library and macro commands defined in Section 3 and Section 4 below! For example:

- in **lab grid** you can try `agent_do_moves(1,[p(1,2),p(1,3),p(1,4)]).` when your agent is located at the initial position `p(1,1)` in the top left corner.
- in **lab identity** you can try `agent_ask_oracle(oscar,o(1),link,L).` to get a link from the Wikipedia page of some secret actor you are trying to identify.

☒ After you **make any changes** to your code in the skeleton answer files don't forget to run the commands `make` followed by `reset_game/start_game` (at which point your GridWorld browser window should automatically refresh)

☒ Also, please make sure the game is **not paused** in the browser when you call `reset_game` or the server may hang due to a bug in the way http responses are assumed to be sequenced.

e) In the `grid` and `search` labs (but **not** in the `identity` lab) it is good practice to close the GridWorld webserver by running the following library commands at the Prolog prompt:

- `leave_game.`
- `stop.`

ⓘ To **quit** Prolog altogether use the command `halt.`

☒ To **abort** a computation that seems to be hanging, you can try hitting `ctrl-c`, `ctrl-d` or `ctrl-x` a few times possibly followed by hitting the `a` key

3 Library Predicates

These labs comprise a set of three consecutive games that involve interacting (over http using the `ailp` library predicates described below for working with) either with a localhost grid server (in the grid and search labs in weeks 2 and 7) or with live Wikipedia pages (in the identity lab in week 3).

The two grid-based games will involve you writing Prolog code to navigate a single agent around a 10×10 square grid rendered in a browser window. The **location of each cell** a grid will be represented by a Prolog term `p(X,Y)` where X and Y are natural numbers denoting the horizontal and vertical offsets (rightwards and downwards) from the top left corner (as labelled on the grid). The **contents of each cell** in the grid is represented by exactly one of the following Prolog terms (where N is an integer identifier of the corresponding object):

Term	Colour	Object	Meaning
`a(N)`	random	agent	a user-controllable agent is located at this position (colour and shape chosen at random)
`empty`	green	empty space	an agent adjacent to this cell can move here
`o(N)`	red	oracle	an agent adjacent to this cell can ask this oracle a question
`t(N)`	black	thing	these represent “walls” or “obstacles” that your agent cannot move through

Your agent will be allowed **move one step** at a time to any empty adjacent (on-grid) cell to the **immediate South, East, North or West** of its current position.

You will control your agent using the following library predicates.

⚠ Note that, in the remainder of this documentation, predicates are often annotated with their respective **arities** (`name/arity`) and arguments are often annotated with their intended **modes** (`+In`, `-Out` or `?Any`).

BUT these **arity and mode decorations should *not* be typed in any actual code** - there are included in the documentation to show how the predicates should be used. As explained in the SWI manual:

- an **input argument** (+) must be instantiated to a correctly typed term when the predicate is called,
- an **output argument** (–) will become instantiated (if it wasn't already) when the predicate succeeds,
- and **partial arguments** (?) may be variable, ground or partially instantiated when the predicate is called and/or succeeds.

Please note that (SWI built-in and GridWorld library) predicates are only guaranteed to work properly (or even at all) when used in the correct way!

Predicate	Meaning
`my_agent(-A)`	return in A the integer id of the last Agent joined to the game (which will usually be **1** though you **shouldn't rely** on that in code!)
`ailp_grid_size(-S)`	return in S the integer Size of (height and width) of the grid (which will usually be **10** though you **shouldn't rely** on that in code!)
`get_agent_position(+A,-Pos)`	return the specified Agent's current Position
`agent_do_moves(+A,+Path)`	move the specified Agent on the grid along the specified Path (which should be a list of grid locations consecutively accessible from the current position) or fail at the first point where a move is found to be invalid
`say(+Message,+A)`	print the given Message string next to the specified Agent's current position on the grid (which maybe useful for debugging?)

Table 1: *Library Predicates for Lab Grid*

Predicate	Meaning
`wp(+T)`	print to stdout an encoding of the Wiki-pedia page with the given Title (which may be an actor name written as a Prolog term like `'Billy Bob Thornton'`)
`wp(+T,-WT)`	return the WikiText of the Wikipedia page with the given Title
`wt_link(+WT,-Link)`	successively return each Link contained inside the given WikiText
`actor(-N)`	successively return each of 12 predefined possible secret actors Names
`link(-L)`	successively return each of 15 predefined possible Links from their Wikipedia pages
`agent_ask_oracle(oscar,o(1),link,-L)`	the agent oscar can (repeatedly) request the oracle `o(1)` to return a random Link from a secret actor's Wiki-pedia page
`test`	Test the solution predicate find_identity(A) succeeds for all possible secret identities

Table 2: *Library Predicates for Lab Identity*

Predicate	Meaning
`map_adjacent(+Pos,-Adj,-Obj)`	given a grid Position, return any Adjacent on-grid cell location along with the Object it contains (or the term `empty`)

Table 3: *Library Predicates for Lab Search (additional to Table 1)*

⚠ Note that `map_adjacent/3` with modes `map_adjacent(+Pos,-Adj,-Obj)` must have an instantiated first argument when you call it. So, given a Position, it returns Adjacent positions and Objects, but not vice versa!

⚠ Although you, the user, will be able to see the layout of the grid in the browser window, please remember your agent can only obtain that information by making calls to the relevant cells using `map_adjacent/3` (which are comparatively expensive as they operate over http); Thus, the tasks of efficiently finding an optimal path to a given location or finding the location of a given object are non-trivial (given the restrictions imposed by the above mode declarations)

Although you **won't need** to exploit this fact in these labs, the library allows multiple **clients** to interact with the **server** over **http** via an internal predicate `query_world/2` that enables one or more Prolog threads running on your machine to join agents to a game, move them around and query the grid. The only thing you need to know is that, because this all happens over http, calls that involve **looking up the contents of a cell** or **moving an agent** will have a significant **time overhead** as compared to standard Prolog queries.

You may also look at **http interactions** in your browser by looking in the **networking tab** of the dev menu for your web browser. This can usually be opened using F12.

4 User Shell

In order to further facilitate user interaction during a session, the GridWorld also provides an **interactive user shell** that can be invoked with the following command that allows the user to run the set of **macros** below (which can reduce the amount of typing you need to do):

Command	Meaning
`?-shell.`	open interactive shell providing the following macros % labs grid & search only

Macro	Meaning
`?help.`	% display a list of the macros below
`?stop.`	% exit from this command shell
`?setup.`	`?-join_game(A),reset_game,start_game.`
`?reset.`	`?-reset_game,start_game.`
`?status.`	`?-query_world(game_status,[A])` % e.g. running/ready
`?whoami.`	`?-my_agent(A)`
`?position.`	`?-my_agent(A),get_agent_position(A,P)`
`?search.`	`?-search_bf` % lab search only
`?call(+G).`	`?-findall(G,call(G),L).`

Table 4: *Possible shell macros.*

Note how the **shell prompt** "`?`" omits the dash in the standard **Prolog prompt** "`?-`".

You can enter and leave the user shell at any time; and you can run non-shell commands from within the shell by wrapping them up as an argument to a **`call`** macro - which will implicitly find *all solutions* of the specified goal.

If you want to make your head hurt, try running `call(shell)` from within the shell!

5 Outline

lab	grid	identity	search
Week	2	3	7
Solution filename	`lab_grid_12345.pl`	`lab_identity_12345.pl`	`lab_search_12345.pl`
Solution predicate	`spiral/1`	`find_identity/1`	`search_bf/0`
Needs internet?	No	Yes (access Wikipedia)	No
Needs localhost?	Yes (access grid)	No	Yes (access grid)
Provides shell?	Yes	No	Yes
Agent Name	a(1)	oscar	a(1)
Oracle Name	not applicable	o(1)	o(1)

Table 5: *Outline of labs.*

formatted by Markdeep 1.14