Tutorial Overview

(5min) Introduction to Planning.Domains

(15min) api.planning.domains
(10min) solver.planning.domains
(15min) editor.planning.domains

(5min) Break

(40min) Putting it all together
• Three initiatives; one platform.
• Strong focus on planning problems.
• Tools for (and by) the community.
- Three initiatives; one platform.
- Strong focus on planning problems.
- Tools for (and by) the community.

**Thanks!** The planning.domains services are graciously supported by the ICAPS organization and community.
Wouldn’t it be cool if...

...we had a central repository for PDDL files.
API

- Open repository of over 125 domains
- Open repository of over 125 domains
- API access for:
  - Collections
  - Domains
  - Problems

```
GET api.planning.domains/collections
Returns all of the collections.
```

```
GET api.planning.domains/collection/{col-id}
Returns the collection matching col-id.
```

```
GET api.planning.domains/domain/{dom-id}
Returns the domain matching dom-id.
```

```
GET api.planning.domains/problems/search
Returns all of the problems matching the query provided. The following parameters can be used for the query:
```

<table>
<thead>
<tr>
<th>Param</th>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>domain</td>
<td>Number</td>
<td>Matches the provided domain ID.</td>
</tr>
<tr>
<td>domain_name</td>
<td>String</td>
<td>Matches when the problem's domain name contains the provided string.</td>
</tr>
<tr>
<td>problem_name</td>
<td>String</td>
<td>Matches when the problem's name contains the provided string.</td>
</tr>
<tr>
<td>min_lower_bound</td>
<td>Number</td>
<td>Matches all problems with a lower bound no smaller than the provided number.</td>
</tr>
<tr>
<td>max_upper_bound</td>
<td>Number</td>
<td>Matches all problems with an upper bound no larger than the provided number.</td>
</tr>
</tbody>
</table>

Courtesy of Ramirez, Lipovetzky, Haslum, ...
- Open repository of over 125 domains
- API access for:
  - Collections
  - Domains
  - Problems
- Python library and command utility

Courtesy of Andrew Coles.
Open repository of over 125 domains

API access for:
- Collections
- Domains
- Problems

Python library and command utility

JavaScript library

<table>
<thead>
<tr>
<th>ID</th>
<th>Domain</th>
<th>Requirements</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>80</td>
<td>cybersecurity</td>
<td>:action-costs :strips</td>
<td>A domain that models the cyber security model of vulnerability analysis for cyber defense.</td>
</tr>
<tr>
<td>58</td>
<td>elevators</td>
<td>:action-costs :typing</td>
<td>(opt08) The scenario is the following: There is a building with N+1 floors, numbered from 0 to N. The building can be separated in blocks of size M+1, where M divides N. Adjacent blocks have a common floor. For example, suppose N=12 and M=4, then we have 13 floors in total (ranging from 0 to 12), which form 3 blocks of 5 floors each, being 0 to 4, 4 to 8 and 8 to 12. The building has K fast (accelerating) elevators that stop only in floors that are multiple of M/2 (so M has to be an even number). Each fast elevator has a capacity of X persons. Furthermore, within each block, there are L slow elevators, that stop at every floor of the block. Each slow elevator has a capacity of Y persons (usually Y</td>
</tr>
<tr>
<td>95</td>
<td>elevators</td>
<td>:action-costs :typing</td>
<td>(sato8) The scenario is the following: There is a building with N+1 floors, numbered from 0 to N. The building can be separated</td>
</tr>
</tbody>
</table>
Wouldn’t it be cool if...

...we had a planner in the cloud.
- **Call via a URL**
- **Call using JSON**

Plan Found:

```plaintext
(unstack b c)
(put-down b)
(unstack c a)
(put-down c)
(unstack a d)
(stack a b)
(pick-up c)
(stack c a)
(pick-up d)
(stack d c)
```

```javascript
$.ajax({
  type: "POST",
  contentType: 'application/json',
  data: JSON.stringify({"domain": domText, "problem": probText}))
  .done(function (res) {
    if (res['result'] === 'ok') {
      window.alert('Plan found!');
    } else {
      window.alert('Planning failed.');
    }
    console.log(res);
  });
```
Call via a URL
Call using JSON
FOSS project to deploy your own

PDDL Solver (in the cloud!)

This project is the bases for solver.planning.domains -- a web service that provides access to an automated planner. Please report any bugs or feature requests you may have on the [issue list] for the project.

Deploying your own solver

This project should get you from zero to having your own hosted planner in the cloud (heroku to be specific) in under 5 minutes (yes, I've timed myself). It could be considerably less if you already have a heroku account and the appropriate software installed. The steps to having things setup and running are as follows:

1. Head over to http://heroku.com and get yourself an account.
2. Install the [heroku toolbelt] which will allow you to deploy new applications.
3. Login using your credentials from step 1.
4. Clone this project (if you haven't already) and navigate to it: git clone git@bitbucket.org:pddl-tools/solver.git; cd solver
5. Run heroku create from the directory this file exists. Take note of the URL.
6. Run git push heroku master to deploy the software.

Est voilà! You now have your very own planner-in-the-cloud.
- Call via a URL
- Call using JSON
- FOSS project to deploy your own
- Ultra-agile track for king-of-the-hill

≡ BFS(f)
Wouldn’t it be cool if...

...we had a dedicated editor for PDDL.
Online editor
Editor

- Online editor
- Syntax highlighting

```prolog
(define (domain BLOCKS)
 (:requirements :strips)
 (:predicates (on ?x ?y)
   (ontable ?x)
   (clear ?x)
   (handempty)
   (holding ?x))
 )

(:action pick-up
 :parameters (?x)
 :precondition (and (clear ?x) (ontable ?x) (handempty))
 :effect
   (and (not (ontable ?x)))
   (not (clear ?x))
   (not (handempty))
   (holding ?x))
```
Editor

- Online editor
- Syntax highlighting
- Bracket folding
- Online editor
- Syntax highlighting
- Bracket folding
- Auto-completion
Editor

- Online editor
- Syntax highlighting
- Bracket folding
- Auto-completion
- Save/load locally
- Online editor
- Syntax highlighting
- Bracket folding
- Auto-completion
- Save/load locally
- Import via the API
- Online editor
- Syntax highlighting
- Bracket folding
- Auto-completion
- Save/load locally
- Import via the API
- Compute plans via the online solver
Editor

- Online editor
- Syntax highlighting
- Bracket folding
- Auto-completion
- Save/load locally
- Import via the API
- Compute plans via the online solver
- Analyze using TorchLight

**Torchlight Output**

TorchLight: parsing domain file
domain 'BLOCKS' defined
done.
TorchLight: parsing problem file
problem 'BLOCKS-7-0' defined
done.

TorchLight: running Fast-Downward translator to generate variables ... done.
TorchLight: creating SG and DTG structures
Warning: didn’t find variable value for FF ft ON(E E). Skipping the fact from variables structures.
Warning: didn’t find variable value for FF ft ON(G G). Skipping the fact from variables structures.
Warning: didn’t find variable value for FF ft ON(B B). Skipping the fact from variables structures.
Warning: didn’t find variable value for FF ft ON(A A). Skipping the fact from variables structures.
Warning: didn’t find variable value for FF ft ON(P F). Skipping the fact from variables structures.
Warning: didn’t find variable value for FF ft ON(C C). Skipping the fact from variables structures.
Warning: didn’t find variable value for FF ft ON(D D). Skipping the fact from variables structures.
TorchLight: static examination of SG and DTG structures ... done.

TorchLight guaranteed global analysis:
Failed.
Percentage of successful x0/t0 gDGs : 7.06% (6 of 85)

TorchLight guaranteed local analysis of initial state:
Failed.

TorchLight approximate local analysis of initial state:
Failed.

TorchLight: sampling random states ... done.

TorchLight guaranteed local analysis of sampled states:
Success and hence no local minima under h*: 0.00%
Online editor
Syntax highlighting
Bracket folding
Auto-completion
Save/load locally
Import via the API
Compute plans via the online solver
Analyze using TorchLight

An anatomy of a plugin (JavaScript file):

```javascript
define(function () {
  return {
    name: "Plan-o-matic1000",
    author: "John_Smith",
    email: "yeah@right.com",
    description: "A plugin template",

    // Called when loaded or enabled
    initialize: function () { },

    // Called when disabled
    disable: function () { },

    // Used to save settings
    save: function () { return {}; },

    // Restore any previous settings
    load: function (settings) { }
  };
});
```
1. **api.planning.domains**
   - Central PDDL repository
   - API interface to all benchmarks
   - Suite of tools to interface with API

2. **solver.planning.domains**
   - Planner in the cloud
   - Open source project
   - Rolling ultra-agile contest

3. **editor.planning.domains**
   - Custom PDDL editor
   - Tie-in to the API and Solver
   - TorchLight and other analysis soon
API Outline

- JSON read and write API
- Python interface
- JavaScript widgets
- Command-line utility
http://api.planning.domains

The start of every API call
http://api.planning.domains
/<format>

Can be json or xml (ommitted for post)
http://api.planning.domains
  /<format>
  /<genre>

For now, can only be **classical**
(**FOND, RDDL, RMPL, etc coming soon**)
http://api.planning.domains
  /<format>
  /<genre>
  /[collection|domain|problem](s)

Depends on if you want a list or single object
http://api.planning.domains
/<format>
/<genre>
/[collection|domain|problem](s)
/<id>|search?option=val|...

Options vary after this point...
api.planning.domains/json/classical/collection/12

{
  "error": false,
  "message": "Success!",
  "result": {
    "collection_id": 12,
    "collection_name": "All-IPC (STRIPS)",
    "description": "A selection of STRIPS...",
    "domain_set": "[8,14,17,19,24,27,...,129]",
    "tags": "[]"
  }
}
api.planning.domains/json/classical/domain/13
{
    "error": false,
    "message": "Success!",
    "result": {
        "domain_id": 13,
        "domain_name": "transport",
        "description": "(opt11) Each vehicle...",
        "tags": "["\":action-costs\",
    }"
}
Problems

api.planning.domains/json/classical/problem/13

```json
{
  "error": false,
  "message": "Success!",
  "result": {
    "problem_id": 13,
    "domain_id": 4,
    "domain": "sokoban",
    "problem": "p06.pddl",
    "domain_url": "http://www.haz.ca/planning-domains/...",
    "problem_url": "http://www.haz.ca/planning-domains/...",
    "domain_path": "classical/sokoban-opt08-strips/p06-domain.pddl",
    "problem_path": "classical/sokoban-opt08-strips/p06.pddl",
    "tags": "["]",
    "lower_bound": 5,
    "upper_bound": 11,
    "average_effective_width": null,
    "max_effective_width": null,
    "lower_bound_description": "haslum/pd-missing-hlb/...",
    "upper_bound_description": "Resetting the upper bounds",
    "average_effective_width_description": "",
    "max_effective_width_description": ""
  }
}
```
Demo!
Note: For now, modifications require special access

- Can update the attribute of any type:
  - POST API_PATH/updatecollection/{col-id}
  - POST API_PATH/updatedomain/{dom-id}
  - POST API_PATH/updateproblem/{prob-id}

- Following parameters are required:
  - user Email address
  - password Provided by admins
  - key Attribute name
  - value New value
  - desc Description indicating source
To find all problems that have a particular attribute set to null. For example, all problems missing an upper bound:

```
GET API_PATH/nullattribute/upper_bound
```

```
{
  "error": false,
  "message": "Success!",
  "result": [
    {
      "id": 3027,
      "domain_path": "classical/pathways/domain_p02.pddl",
      "problem_path": "classical/pathways/p02.pddl"
    },
    {
      "id": 485,
      "domain_path": "classical/floortile-opt11-strips/domain.pddl",
      "problem_path": "classical/floortile-opt11-strips/opt-p09-018.pddl"
    },
    ...
  ]
}
```
API Tags

Examples: PDDL Requirements, Unsolvables, Invertables

Note: For now, modifications require special access

- Listing all tags:
  - GET API_PATH/tags

- Adding tags:
  - POST API_PATH/tagcollection/{col-id}
  - POST API_PATH/tagdomain/{dom-id}
  - POST API_PATH/tagproblem/{prob-id}

- Removing tags:
  - POST API_PATH/untagcollection/{col-id}
  - POST API_PATH/untagdomain/{dom-id}
  - POST API_PATH/untagproblem/{prob-id}
Incumbent can be submitted and retrieved

- **GET** API PATH /plan/{prob-id}
  ```json
  {
    "error": false,
    "message": "Success!",
    "result": {
      "plan": "(move player-01 pos-6-4 pos-6-3 dir-up)\n..."
    }
  }
  ```

- **POST** API PATH /submitplan/{prob-id}
  ```
  plan String of IPC-style plan
  email User email (for the glory)
  ```

**Note:** No special access required!
Demo!
JSON API access
Open-source project
Ultra-agile track
### Solver API

Three main **POST** endpoints for solving and validating:

- `solver.planning.domains/solve`
- `solver.planning.domains/validate`
- `solver.planning.domains/solve-and-validate`

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>domain</strong></td>
<td>Either URL or raw PDDL for domain</td>
</tr>
<tr>
<td><strong>problem</strong></td>
<td>Either URL or raw PDDL for problem</td>
</tr>
<tr>
<td><strong>probID</strong></td>
<td>API ID to supersede <code>domain</code> and <code>problem</code></td>
</tr>
<tr>
<td><strong>is_url</strong></td>
<td>Set to <strong>true</strong> if using URLs</td>
</tr>
<tr>
<td><strong>plan</strong></td>
<td>IPC format plan (just for <code>/validate</code>)</td>
</tr>
</tbody>
</table>
Returned parameters if plan is computed:

- **length**: Number of actions
- **output**: Planner output
- **parse_status**: Status of the plan parsing (e.g., **ok**)
- **type**: Either **simple** or **full**
- **plan**: ...

Returned parameters if VAL is called:

- **cost**: Total plan cost
- **val_stdout**: VAL standard output
- **val_stderr**: VAL standard error
- **val_status**: Either **valid** or **err**
- **error**: Indication of any VAL error
Solver API: Returned Plan

(full)  
Ground action info included

Array of objects

name    Ground action name
action   Full ground action

"plan": {
  "action":
    " (:action move\n...",
  "name":
    "(move player-01...",
  "action":
    " (:action move\n...",
  "name":
    "(move player-02...",
  ...
}

(simple)  
Parser was unable to ground

Array of strings

"plan": {
  "(move player-01 pos-6-4...",
  "(move player-02 pos-3-5...",
  "(move player-01 pos-6-3...",
  ...
}

Over 14k plans computed since announcing at ICAPS!
Demo!
Editor Outline

- Editor Usage
- Editor Plugin Architecture
Demo!
define (function () {
    return {
        name: "Plan-o-matic 1000",
        author: "John Smith",
        email: "yeah@right.com",
        description: "A plugin template."
    },

    // Called when loaded or enabled
    initialize: function() { },

    // Called when disabled
    disable: function() { },

    // Used to save settings
    save: function() { return {}; },

    // Restore any previous settings
    load: function(settings) { }
});
define(function () {
    return {
        // Mandatory flag
        meta: true,

        // List of meta / normal plugins
        plugins: {
            "plugin1": {
                url: "http://path.to.plugin/1",
                settings: {} },
            "plugin2": {
                url: "http://path.to.plugin/2",
                settings: {option: "value"} },
            ...
        };
    }
});
Editor API: Menu Interface

add_menu(name, id, icon)

- name: Name for the menu
- id: HTML ID for reference
- icon: Bootstrap glyphicon

remove_menu_button(id)

- id: HTML ID for menu/button

add_menu_button(/*args*/)

- name: Name for the menu
- id: HTML id for later reference
- icon: Bootstrap glyphicon string
- cb_string: String of function call (no " permitted)
- parent_menu: (optional) ID for parent menu
new_tab(name, callback)

name Name for the new tab
callback Function that is called with the new view’s HTML ID (shown when tab is selected)

window.new_tab("My Tab", function(editor_name) {
  var newHTML = "<p>I’m in a tab!</p>";
  $("#"+editor_name).html(newHTML);
});
Editor API: Code Snippets

add_snippet(snippet, trigger)

snippet  Cloud9 style snippet
trigger  Text to trigger the auto-complete

window.add_snippet(
    "(when ${1:(and ())}\n     (${2}))",
    "condeff"
)
register_file_chooser(name, settings)
setup_file_chooser(btnName, desc)

- **name**: Slug or nickname for the chooser
- **settings**: Object including `showChoice` and `selectChoice` functions
- **btnName**: Name of the button for submission
- **desc**: Description for the top of the dialog

```javascript
window.register_file_chooser('planner', {
  showChoice: function() {
    window.setup_file_chooser('Plan','Compute Plan');
    $('#plannerURL').val(window.solverURL);
  },
  selectChoice: findPlan // Called when selected
});
```
injected_styles(css_style)

css_style  String of CSS to be included

window.inject_styles(
  ".some-divs { float: left; }\
  #some-other-div { padding: 13px; }"
)

window.toastr.success("Hurray!")
window.toastr.info("Things are happening...")
window.toastr.warning("Uh oh.")
window.toastr.error("I give up :(")
5min Break...

...plugin from start-to-finish