Inf2D - Coursework 2

Situation Calculus and Planning

Mihai Dobre
M.S.Dobre@sms.ed.ac.uk

March 4, 2016

THE UNIVERSITY *of* EDINBURGH
informatics

Outline

- Golog and Prolog
- Parking Agent

Golog

- Language based on Situation Calculus.
  - Situations
  - Actions
  - Predicates (atemporal, fluent)
  - Axioms (descriptive, precondition, successor state)
- How to automatize inference in this language?
  - We have Prolog
- Golog interpreter is written in Prolog

Prolog

- Declarative language (relations)
  - Knowledge base (facts and rules)
  - Queries (can you prove/satisfy this?)
- User can provide a KB and ask a query.
- Prolog uses resolution (backward chaining) over KB to answer.

The Syntax of Prolog

- Predicates & constants start with lower-case.
- Variables start with upper-case.
- A **Term** is a constant, variable or composite from other terms.
- No quantifiers
  - variables in KB are implicitly universal
  - queries ask for satisfaction (existential-like)

,	conjunction
;	disjunction
:-	if (rev implication)
.	end of sentence
=	unification
_	anonymous variable

Example of Prolog syntax

KB:

```
horse(bluebeard).
offspring(charlie, bluebeard).
horse(H) :- offspring(H,X), horse(X).
```

Queries:

```
horse(charlie).
horse(X).
lion(X).
```

“**:-**” is only used in rules. Rules cannot be queries.

Prolog Demo

- Create a KB file, e.g., `demo.pl`
- Load the file in Prolog with command `[demo].`
- Useful predicates:
    - `listing/0` : list all facts in KB
    - `assert/1` : save a fact into KB
    - `retract/1` : remove a fact from KB
    - `halt/0` : exit Prolog

Numbers in Prolog

- Numbers are constants.
- Some arithmetic operators:

$$+ \quad - \quad / \quad * \quad > \quad < \quad >= \quad <= \quad \texttt{is} \quad =:=$$

KB:

```
price(book,10).
price(coffee,3).
canBuy(Money,Item) :-
  price(X,P), Money>=P, Item=X.
```

Queries:

```
canBuy(2,Item).
canBuy(6,Item).
canBuy(12,Item).
```

Numbers in Prolog

Create alternative definition of numbers:

Use a predicate “s” that stands for successor!

Rule:

```
can(A,X) :-   A = move, X = s(s(_));
              A = jump, X = s(s(s(_))).
```

Queries:

```
can(jump, s(s(s(s(s(0)))))).
can(move, s(0)).
can(X, s(s(s(s(s(0)))))).
```

Parking Agent

Coursework 2

You are asked to:

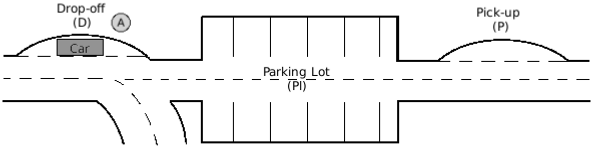
1. Formalise the problem in the Situation Calculus
2. Implement in Golog

Problem Description

A: agent

A can move and drive cars between connected areas.

The grounds:



A can also park or deliver cars.

Part 1: Formalise

The environment:

- which locations are connected?
- where is **A** and **Car**?
- what is the state of the **Car**?

The actions:

- what can **A** do?
- when can it do it? (preconditions)
- how do actions change the environment? (effects)

Part 1: Formalise

Please:

- be as compact as possible (remove unused predicates or fluents)
- use Russell & Norvig notations
  - Constants and predicates begin with a capital letter
  - Variables begin with a lower-case letter
  - and, or, not, $=>$, $<=>$

Part 2 & 3: Implement and extend

Part 2:

- Translate Part 1 into the syntax of Golog.
- Test on some initial states & goals.

Part 3:

- Add more predicates
- Change your axioms to suit the new predicates.
- Test.

Tips

Being concise will help you:

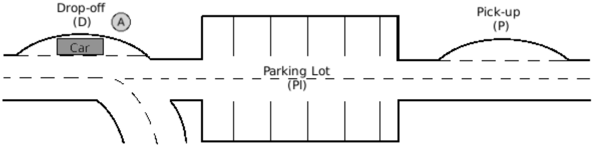
- Remove unused predicates
- Remove arguments in predicates that don't need them.

Balance abstraction:

- Don't add a new predicate for every action (e.g., `moveAfromDtoP1`).
- Don't use same predicates for things that are essentially different.

Tips

Test on simple tasks, specially when debugging.



Tips

Start with:

- The first two chapters of “Learn Prolog Now!” by Patrick Blackburn, Johan Bos, and Kristina Striegnitz (online version);
- Play with the block example;
- Start with the simple tasks and build up.

Assignment package

- 8 files
- A Golog-based planner - only available on DICE machines
- Domain and problem template files
- All text should go in answer.txt
- Blocks world example.

Submission Package

All original files

- Part 1 answers should be in **answer.txt**
- Part 2
  - a domain file: **domain-task21.pl**
  - problem instances: **instance-task22.pl**, **instance-task23.pl**, **instance-task24.pl**
- Part 3
  - **domain-task31.pl**, **instance-task31.pl**
  - **domain-task32.pl**, **instance-task32.pl**
  - **domain-task33.pl**, **instance-task33a.pl** and **instance-task33b.pl**

To conclude

Worth 12.5% of the final mark

Deadline:

- **4pm on Monday 21st March**
- Electronic submission through **submit** software (available on any DICE machine)