Escola Politécnica da USP - Depto. de Enga. Mecatrônica

PMR-3510 Inteligência Artificial

Aula 9- Inferência Lógica

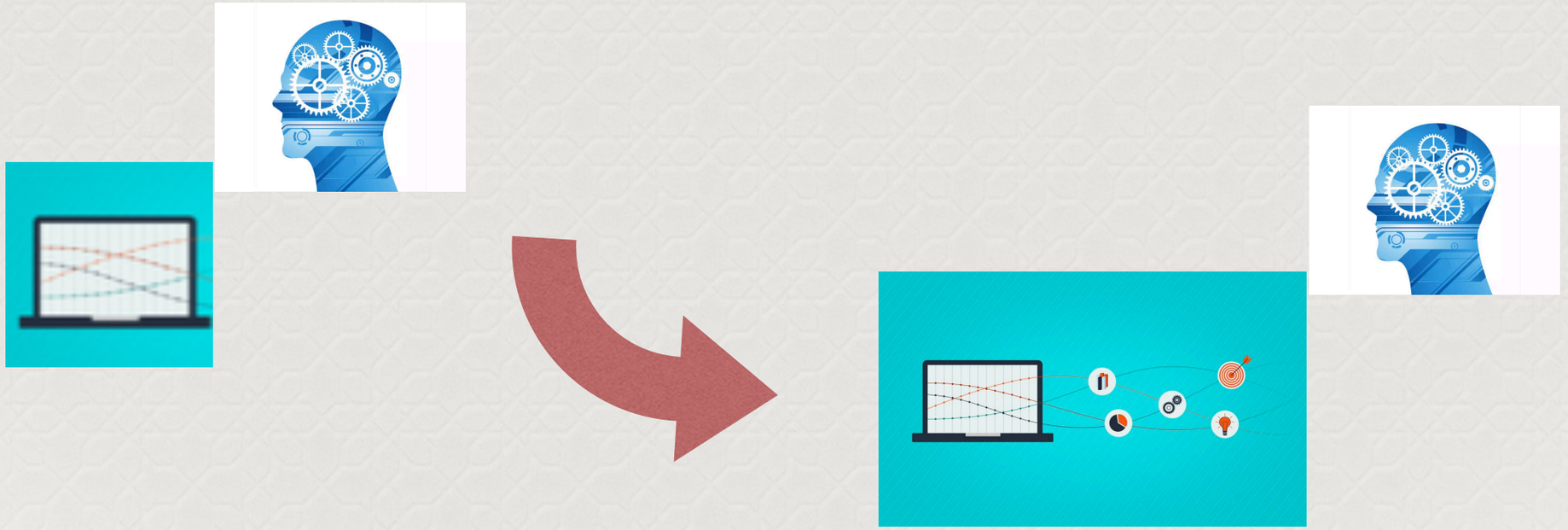
Prof. José Reinaldo Silva

reinaldo@usp.br

d-l@b

Resolução de problemas com IA

Existem duas maneiras de resolver problemas usando IA: uma é algorítmica, usando busca (clássica ou informada); a outra é usando inferência lógica.

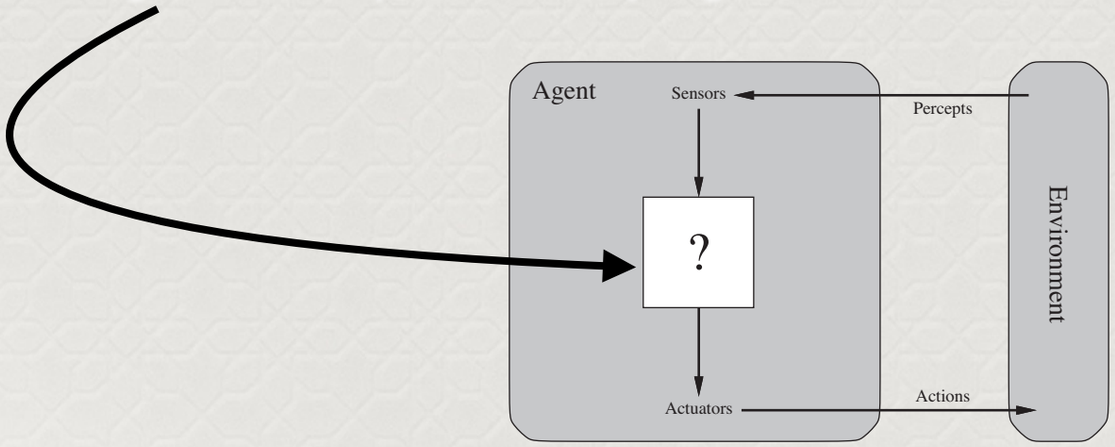


Introduction

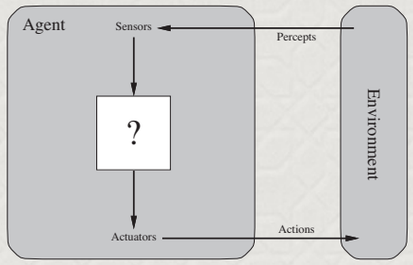
From Data to Intelligent Agents

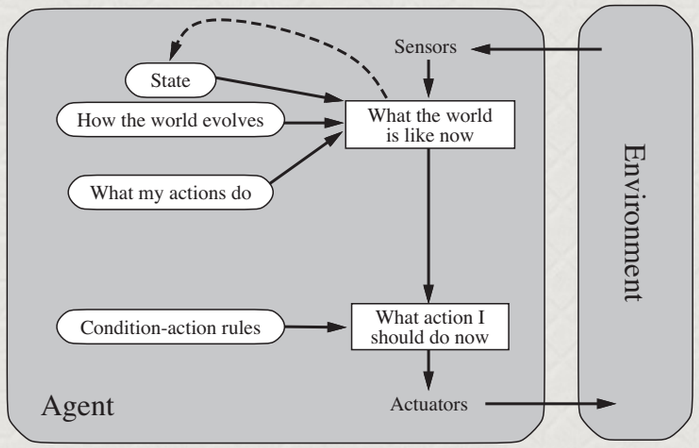
Agentes inteligentes autônomos

Até então definimos o agente inteligente pela sua relação com o environment com o entorno (environment). Note-se que os mecanismos de percepção e ação podem envolver hardware (sensores e atuadores). Mas a autonomia está ligada ao processamento (a interrogação na figura), que deve ser necessariamente executada por uma arquitetura computacional, que por sua vez tem os processos guias por um programa. Portanto o nosso agente pode ser definido como: agente = arquitetura + programa.

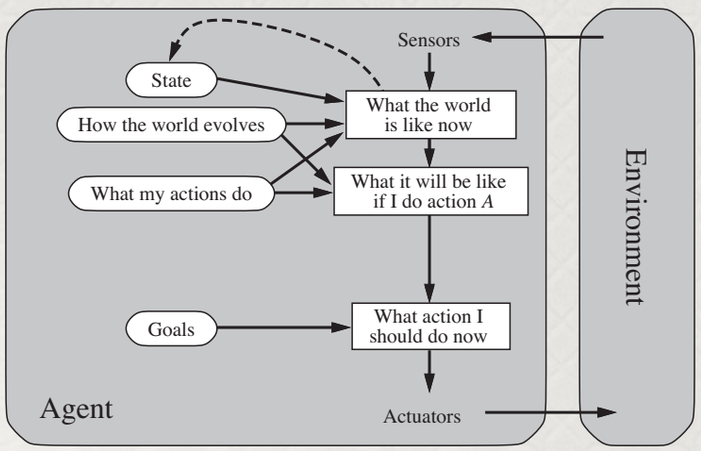


Agentes Lógicos





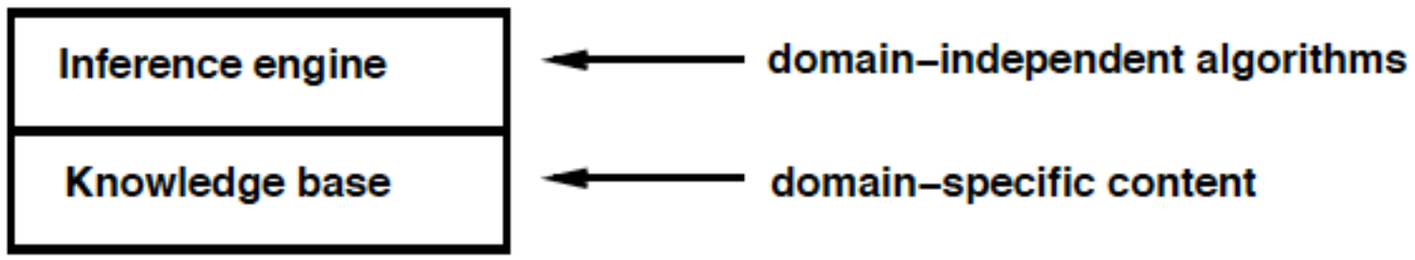
Agente baseado em modelo



Agente baseado em objetivos

Agentes automatizados (eventualmente móveis) podem agir em um ambiente real de forma reativa, baseado em um modelo de conhecimento do seu entorno ou com objetivos a cumprir como em um jogo. Vamos tomar como base este tipo de agente para um dispositivo móvel que atua em um jogo que tem um ambiente em torno chamado Mundo de Wumpus. Nesse caso o agente precisa agir de forma inteligente para evitar desastres e conseguir o seu objetivo (o que nem sempre será possível). A forma "inteligente" (racional) de agir e selecionar ações será baseada em uma base de conhecimento do domínio.

Knowledge bases



Knowledge base = set of sentences in a formal language

Declarative approach to building an agent (or other system):
TELL it what it needs to know

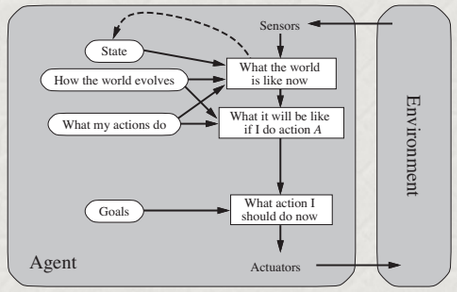
Then it can ASK itself what to do—answers should follow from the KB

Agents can be viewed at the knowledge level
i.e., what they know, regardless of how implemented

Or at the implementation level
i.e., data structures in KB and algorithms that manipulate them

Stuart Russell, Univ. of California - Berkeley, autor do livro texto.

Introdução informal aos agentes lógicos

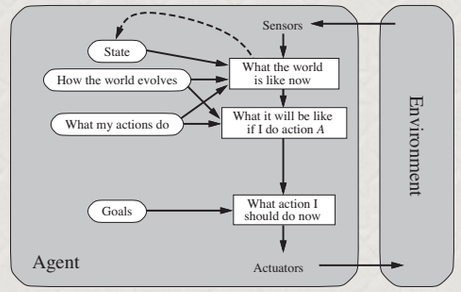


Vamos inicialmente fazer uma abordagem informal aos agentes lógicos utilizando um jogo (mais uma vez) como sugerido no livro texto.

O jogo se chama Mundo de Wumpus e é baseado em um grid onde um agente móvel pode se deslocar (o deslocamento é restrito a movimentos laterais e verticais apenas) evitando cair em um poço ou se deparar com o monstro Wumpus (neste caso o agente pode ser eliminado).

O agente dispõe de uma (única) flecha e pode usá-la para matar o Wumpus. Mas não pode errar ou será eliminado por ele.

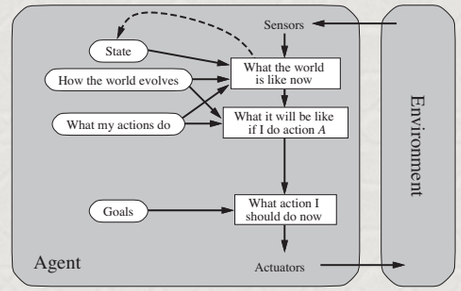
Base de conhecimento



As regras que orientam o objetivo são divididas em dois blocos:

Regras do domínio (descrição do grid): O mundo de Wumpus é uma "caverna" (um grid) com salas conectadas por passagens. Em uma destas salas estará o monstro Wumpus à espreita para devorar qualquer guerreiro (agente) que apareça. O Wumpus pode ser abatido por uma flecha, mas o agente móvel só pode ter uma flecha e portanto só uma chance. Algumas salas têm poços sem fundo e se o agente entrar em uma delas perde o jogo. Em uma das salas tem também um pote de ouro.

Base de conhecimento



Regras de pontuação: +1000 pontos se o agente conseguir sair da caverna com o pode de ouro; -1000 se cair em um poço ou for devorado pelo Wumpus; -1 para cada ação (de movimento) executada e -10 pelo uso da flecha.

O agente começa pelo quadro inferior esquerdo e deve sair da "caverna" por este mesmo quadro.

O Wumpus é circundado (nos quadros vizinhos) por um cheiro horrível, e o poço por uma brisa suave. O quadrado vizinho ao ouro emana um brilho intenso. Assim, o agente pode usar os seus sensores para tentar inferir o caminho melhor.

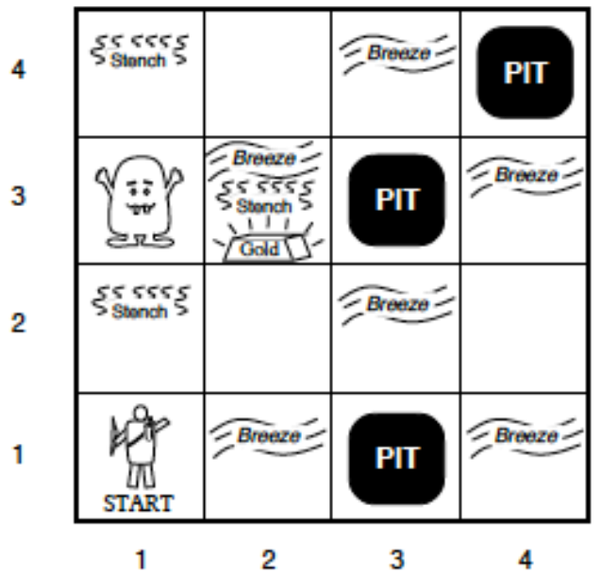
Wumpus World PEAS description

Performance measure
- gold +1000, death -1000
- -1 per step, -10 for using the arrow

Environment
- Squares adjacent to wumpus are smelly
- Squares adjacent to pit are breezy
- Glitter iff gold is in the same square
- Shooting kills wumpus if you are facing it
- Shooting uses up the only arrow
- Grabbing picks up gold if in same square
- Releasing drops the gold in same square

Actuators Left turn, Right turn, Forward, Grab, Release, Shoot

Sensors Breeze, Glitter, Smell



Stuart Russell, Univ. of California - Berkeley, autor do livro texto.

Wumpus world characterization

Observable?? No—only local perception

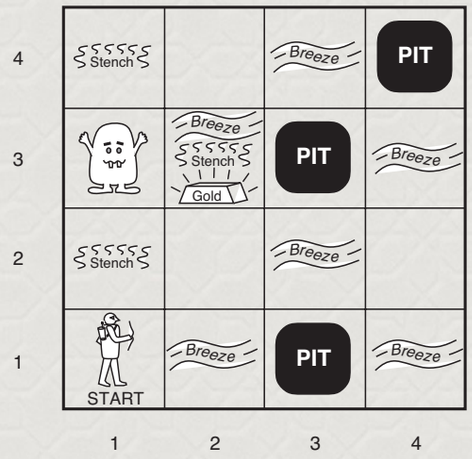
Deterministic?? Yes—outcomes exactly specified

Episodic?? No—sequential at the level of actions

Static?? Yes—Wumpus and Pits do not move

Discrete?? Yes

Single-agent?? Yes—Wumpus is essentially a natural feature



Sensoriamento do ambiente

1. No compartimento onde está o wumpus e nos compartimentos adjacentes (não diagonais) haverá um cheiro horroroso!

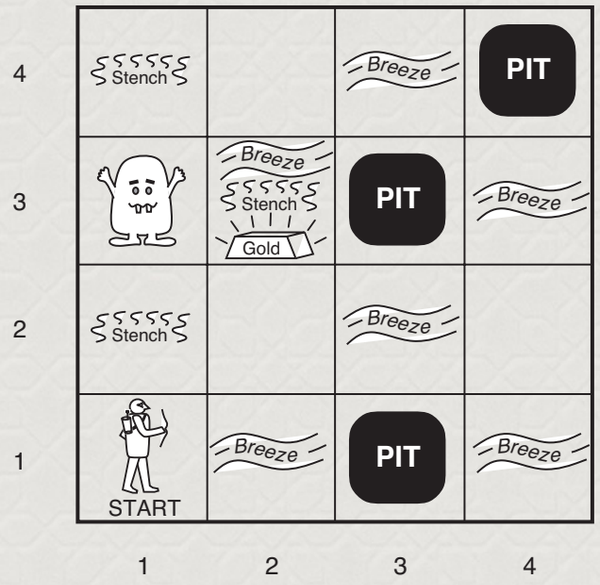
2. Nos quadros adjacentes (não diagonais) a um poço o agente sentirá uma brisa suave;

3. No quadrado onde está o ouro o agente verá um brilho intenso;

4. quando se encaminhar para uma parede o agente perceberá um impacto (como um robô móvel);

5. se o wumpus for morto por uma flecha soltará um grito triste que será ouvido em toda a caverna.

Portanto o agente verá em cada quadro um vetor (uma lista) [Fedor, Brisa, Brilho, Impacto, Grito], ou resumidamente [S, B, G, I, W].



1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 OK	2,2	3,2	4,2
1,1 A OK	2,1 OK	3,1	4,1

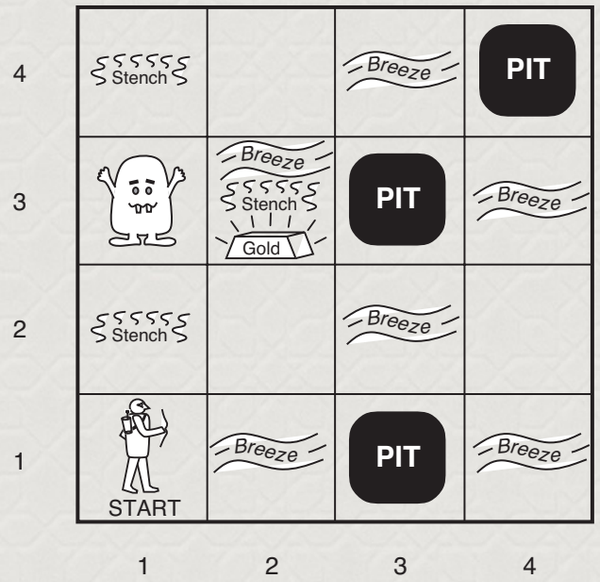
(a)

A = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*

1,4	2,4	3,4	4,4
1,3	2,3	3,3	4,3
1,2 OK	2,2 P?	3,2	4,2
1,1 V OK	2,1 A B OK	3,1 P?	4,1

(b)

Forward



1,4	2,4	3,4	4,4
1,3 **W!**	2,3	3,3	4,3
1,2 **A** **S** **OK**	2,2 **OK**	3,2	4,2
1,1 **V** **OK**	2,1 **B** **V** **OK**	3,1 **P!**	4,1

(a)

A = *Agent*
B = *Breeze*
G = *Glitter, Gold*
OK = *Safe square*
P = *Pit*
S = *Stench*
V = *Visited*
W = *Wumpus*

1,4	2,4 **P?**	3,4	4,4
1,3 **W!**	2,3 **A** **S G** **B**	3,3 **P?**	4,3
1,2 **S** **V** **OK**	2,2 **V** **OK**	3,2	4,2
1,1 **V** **OK**	2,1 **B** **V** **OK**	3,1 **P!**	4,1

(b)

O jogo

O jogo consiste em ter várias rodadas com configuração diferente do grid (a caverna) e fazer com que o agente acumule o maior número de pontos possível. Portanto este deve vagar, sempre por lugares seguros procurando o pote de ouro (que vale +1000 pontos) evitando os poços e evitando ser morto pelo wumpus.

Um agente inteligente

Em cada estado o agente percebe o ambiente → Infere qual a melhor ação em função desta informação → Executa a ação

memória

O processo de inferência

O processo (racional) de inferência consiste em "deduzir" qual ou quais os compartimentos seguros a partir do conhecimento adquirido e do sensorialmente da situação no estado corrente. É portanto um processo mais "inteligente" do que a busca direta em grafo (ou árvore) por um estado-objetivo bem definido.

Entailment

Entailment means that one thing follows from another:

$$KB \models \alpha$$

Knowledge base KB entails sentence α
if and only if
α is true in all worlds where KB is true

E.g., the KB containing "the Giants won" and "the Reds won" entails "Either the Giants won or the Reds won"

E.g., $x + y = 4$ entails $4 = x + y$

Entailment is a relationship between sentences (i.e., syntax) that is based on semantics

Note: brains process syntax (of some sort)

Stuart Russell, Univ. of California - Berkeley, autor do livro texto.

Models

Logicians typically think in terms of models, which are formally structured worlds with respect to which truth can be evaluated

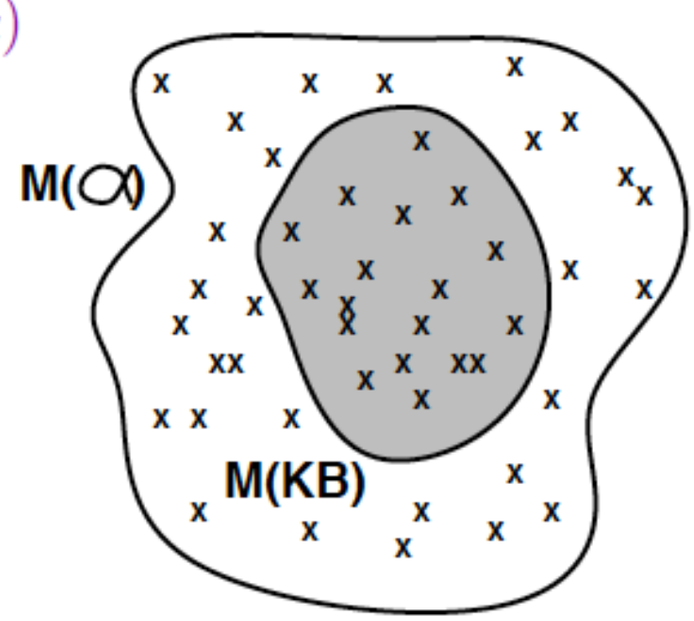
We say m is a model of a sentence α if α is true in m

$M(\alpha)$ is the set of all models of α

Then $KB \models \alpha$ if and only if $M(KB) \subseteq M(\alpha)$

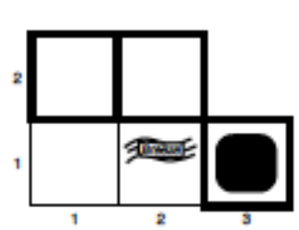
E.g. KB = Giants won and Reds won

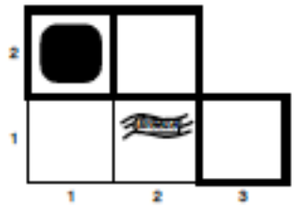
α = Giants won

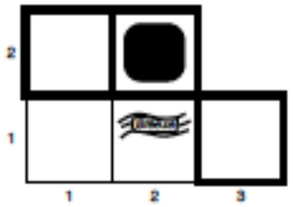


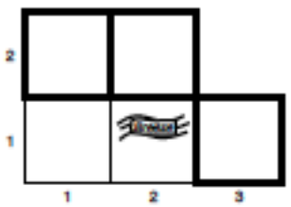
Stuart Russell, Univ. of California - Berkeley, autor do livro texto.

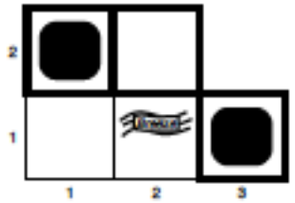
Wumpus models

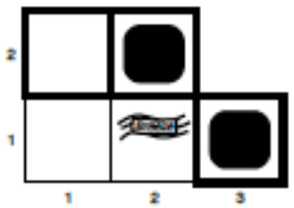


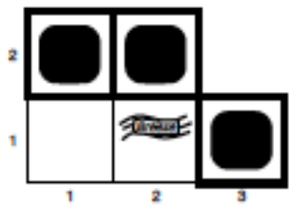


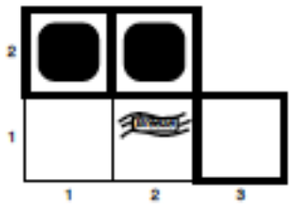






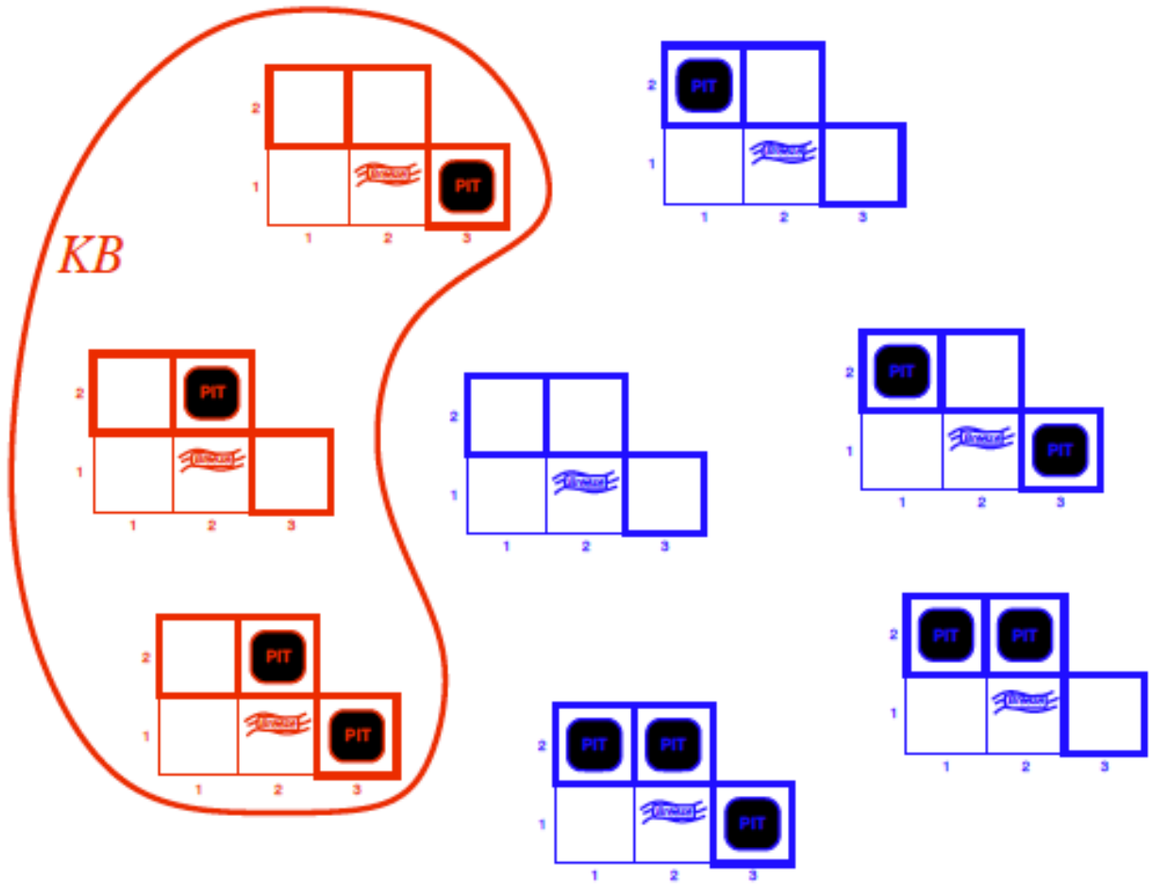






Stuart Russell, Univ. of California - Berkeley, autor do livro texto.

Wumpus models



KB = wumpus-world rules + observations

Stuart Russell, Univ. of California - Berkeley, autor do livro texto.

Inference

$KB \vdash_i \alpha$ = sentence α can be derived from KB by procedure i

Consequences of KB are a haystack; α is a needle.
Entailment = needle in haystack; inference = finding it

Soundness: i is sound if
whenever $KB \vdash_i \alpha$, it is also true that $KB \models \alpha$

Completeness: i is complete if
whenever $KB \models \alpha$, it is also true that $KB \vdash_i \alpha$

Preview: we will define a logic (first-order logic) which is expressive enough to say almost anything of interest, and for which there exists a sound and complete inference procedure.

That is, the procedure will answer any question whose answer follows from what is known by the KB.

Stuart Russell, Univ. of California - Berkeley, autor do livro texto.

Propositional logic: Syntax

Propositional logic is the simplest logic—illustrates basic ideas

The proposition symbols P_1, P_2 etc are sentences

If S is a sentence, $\neg S$ is a sentence (negation)

If S_1 and S_2 are sentences, $S_1 \wedge S_2$ is a sentence (conjunction)

If S_1 and S_2 are sentences, $S_1 \vee S_2$ is a sentence (disjunction)

If S_1 and S_2 are sentences, $S_1 \Rightarrow S_2$ is a sentence (implication)

If S_1 and S_2 are sentences, $S_1 \Leftrightarrow S_2$ is a sentence (biconditional)

Fórmulas-bem-formadas (well-formed-formulas) formam sentenças usando predicados e conectivos lógicos. Estas devem ser "interpretadas"para inferir o seu valor lógico. As sentenças mais simples formadas com conectivos lógicos seguem as regras mostradas na tabela abaixo.

Truth tables for connectives

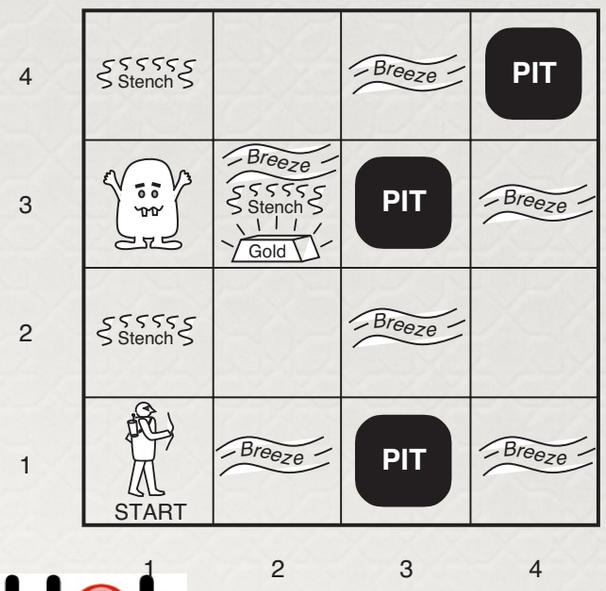
P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	*false*	*true*	*false*	*false*	*true*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

Wumpus world sentences

Let $P_{i,j}$ be true if there is a pit in $[i, j]$.
Let $B_{i,j}$ be true if there is a breeze in $[i, j]$.

$$\neg P_{1,1}$$
$$\neg B_{1,1}$$
$$B_{2,1}$$

"Pits cause breezes in adjacent squares"



$$S_{12} \rightarrow W_{31} \vee W_{22}$$
$$W_{22} \rightarrow S_{21} \wedge S_{12}$$
$$\neg S_{12+}$$

$$\models \quad W_{31}$$

Wumpus world sentences

Let $P_{i,j}$ be true if there is a pit in $[i, j]$.
Let $B_{i,j}$ be true if there is a breeze in $[i, j]$.

$$\neg P_{1,1}$$
$$\neg B_{1,1}$$
$$B_{2,1}$$

"Pits cause breezes in adjacent squares"

$$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$$
$$B_{2,1} \Leftrightarrow (P_{1,1} \vee P_{2,2} \vee P_{3,1})$$

"A square is breezy **if and only if** there is an adjacent pit"

Logical equivalence

Two sentences are logically equivalent iff true in same models:

$\alpha \equiv \beta$ if and only if $\alpha \models \beta$ and $\beta \models \alpha$

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{De Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{De Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

Validity and satisfiability

A sentence is valid if it is true in **all** models,

e.g., $True$, $A \vee \neg A$, $A \Rightarrow A$, $(A \wedge (A \Rightarrow B)) \Rightarrow B$

Validity is connected to inference via the Deduction Theorem:

$KB \models \alpha$ if and only if $(KB \Rightarrow \alpha)$ is valid

A sentence is satisfiable if it is true in **some** model

e.g., $A \vee B$, C

A sentence is unsatisfiable if it is true in **no** models

e.g., $A \wedge \neg A$

Satisfiability is connected to inference via the following:

$KB \models \alpha$ if and only if $(KB \wedge \neg\alpha)$ is unsatisfiable

i.e., prove α by *reductio ad absurdum*

Proof methods

Proof methods divide into (roughly) two kinds:

Application of inference rules
- Legitimate (sound) generation of new sentences from old
- Proof = a sequence of inference rule applications
  Can use inference rules as operators in a standard search alg.
- Typically require translation of sentences into a normal form

Model checking

truth table enumeration (always exponential in n)
improved backtracking, e.g., Davis–Putnam–Logemann–Loveland
heuristic search in model space (sound but incomplete)
e.g., min-conflicts-like hill-climbing algorithms

Forward and backward chaining

Horn Form (restricted)

KB = conjunction of Horn clauses

Horn clause =

- ♢ proposition symbol; or
- ♢ (conjunction of symbols) $\Rightarrow$ symbol

E.g., $C \wedge (B \Rightarrow A) \wedge (C \wedge D \Rightarrow B)$

Modus Ponens (for Horn Form): complete for Horn KBs

$$\frac{\alpha_1, \ldots, \alpha_n, \qquad \alpha_1 \wedge \cdots \wedge \alpha_n \Rightarrow \beta}{\beta}$$

Can be used with forward chaining or backward chaining.
These algorithms are very natural and run in linear time

Robô Valkyrie mede 1,8 metro e pesa 125 quilos. (Foto: Nasa)

métodos de busca clássica e informada

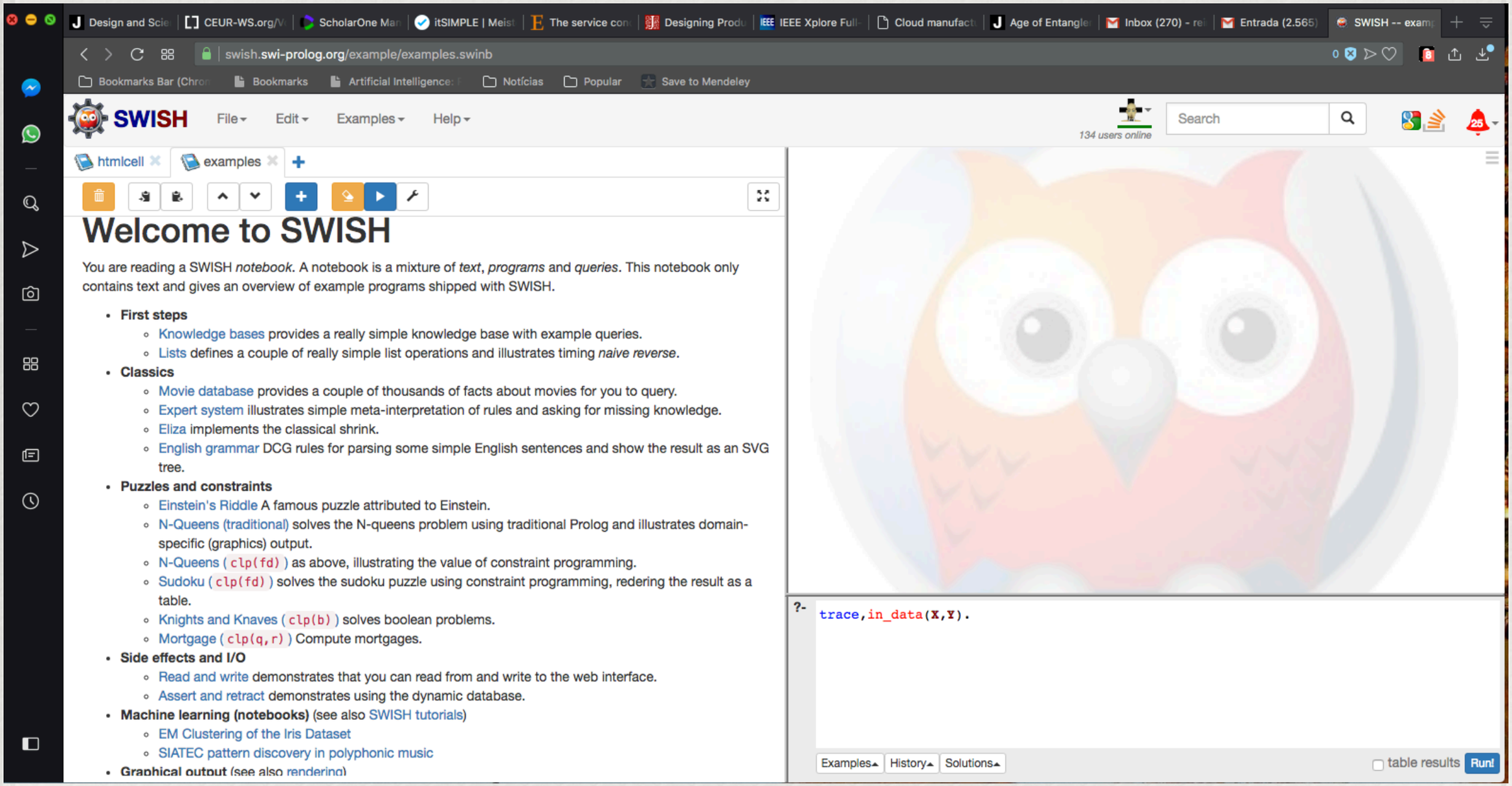
programação lógica

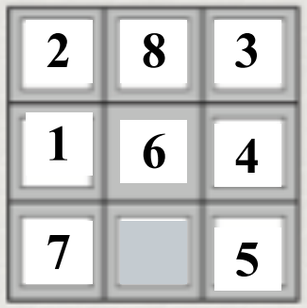
métodos de inferência

Até a próxima aula!

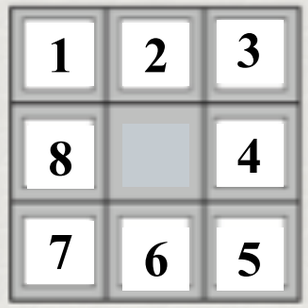
A competição entre equipes

No dia 31, segundo o nosso cronograma, teremos o desfecho do trabalho em grupo e faremos isso na forma de uma competição onde todos os grupos deverão resolver o mesmo problema dado como estado inicial. Todos devem ter o seu programa instalado no sistema Swish e pronto para rodar sua implementação do A*.


SWISH
File
Edit
Examples
Help
Search
134 users online
htmlcell
examples
Welcome to SWISH
You are reading a SWISH notebook. A notebook is a mixture of text, programs and queries. This notebook only contains text and gives an overview of example programs shipped with SWISH.
First steps
Knowledge bases provides a really simple knowledge base with example queries.
Lists defines a couple of really simple list operations and illustrates timing naive reverse.
Classics
Movie database provides a couple of thousands of facts about movies for you to query.
Expert system illustrates simple meta-interpretation of rules and asking for missing knowledge.
Eliza implements the classical shrink.
English grammar DCG rules for parsing some simple English sentences and show the result as an SVG tree.
Puzzles and constraints
Einstein's Riddle A famous puzzle attributed to Einstein.
N-Queens (traditional) solves the N-queens problem using traditional Prolog and illustrates domain-specific (graphics) output.
N-Queens (clp(fd)) as above, illustrating the value of constraint programming.
Sudoku (clp(fd)) solves the sudoku puzzle using constraint programming, redering the result as a table.
Knights and Knaves (clp(b)) solves boolean problems.
Mortgage (clp(q,r)) Compute mortgages.
Side effects and I/O
Read and write demonstrates that you can read from and write to the web interface.
Assert and retract demonstrates using the dynamic database.
Machine learning (notebooks) (see also SWISH tutorials)
EM Clustering of the Iris Dataset
SIATEC pattern discovery in polyphonic music
Graphical output (see also rendering)
?-
trace,in_data(X,Y).
Examples
History
Solutions
table results
Run!



g(0) = 0 + 4



estado final

O estado final será o mesmo que usamos na aula passada no exemplo de resolução da busca baseada no algoritmo Best-first.

O procedimento

- Todos devem estar preparados com o Swish e o seu algoritmo, mas sem tocar no teclado ou no mouse;
- O estado inicial será colocado em um slide;
- Todos esperam o "sinal de largada" para começar o processo;
- Devem inserir o estado inicial no programa e resolvê-lo;
- Chegando ao estado final o programa deve descrever este estado e colocar na tela o custo da busca (soma de g(x)).
- com esta informação na saída do programa a equipe faz um sinal de termino e o tempo é anotado.

Após a competição

Após a competição cada equipe deve fazer o upload no sistema e-disciplinas de um texto que descreve a heurística (g(x) e h(x)), suas propriedades, e uma listagem do código Prolog. A nota final será a média da nota atribuída a este documento e a nota da competição.

A nota da competição

A nota da competição será a composição da classificação por tempo (de zero a cinco), e cinco pontos se a equipe conseguiu resolver mesmo que em um tempo mais longo, e zero se não fechou o processo de busca.

Trabalho em grupo

Os grupos estão já definidos e registrados no e-disciplinas. A configuração é a seguinte:

Grupo 1:
Alyson Akio Haro
Alexandre Inoue
Eduardo Kose
Vitor Fukuda

Grupo 2:
Alex Majima
Gabriel Ferreira
Lucas de Angelis
Thiago Ferraz

Grupo 3:
Gabriel Tutia
Lucas Palopoli
Pedro dos Santos Melo
Samuel Monção

Grupo 4:
Gustavo Novello
Luiz Guilherme Sabino
Alessandro Ezequiel Junior
Gabriel Negre

Grupo 5:
Gabriel Yida
Vinicius Santiago
Danilo Polidoro

Grupo 6:
Guilherme Aires de França
Henrique Peterlevitz
Wagner Geraldo Ferreira