PMR-3510 Inteligência Artificial

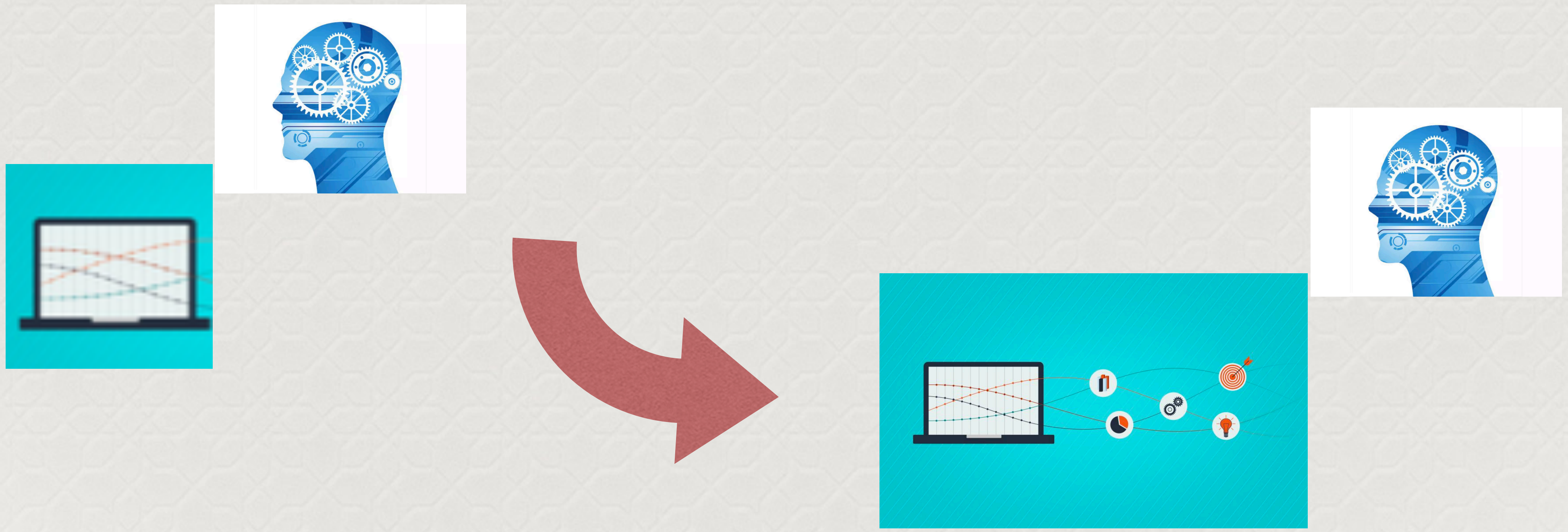
Aula 9- Planejamento automático

Prof. José Reinaldo Silva
reinaldo@usp.br

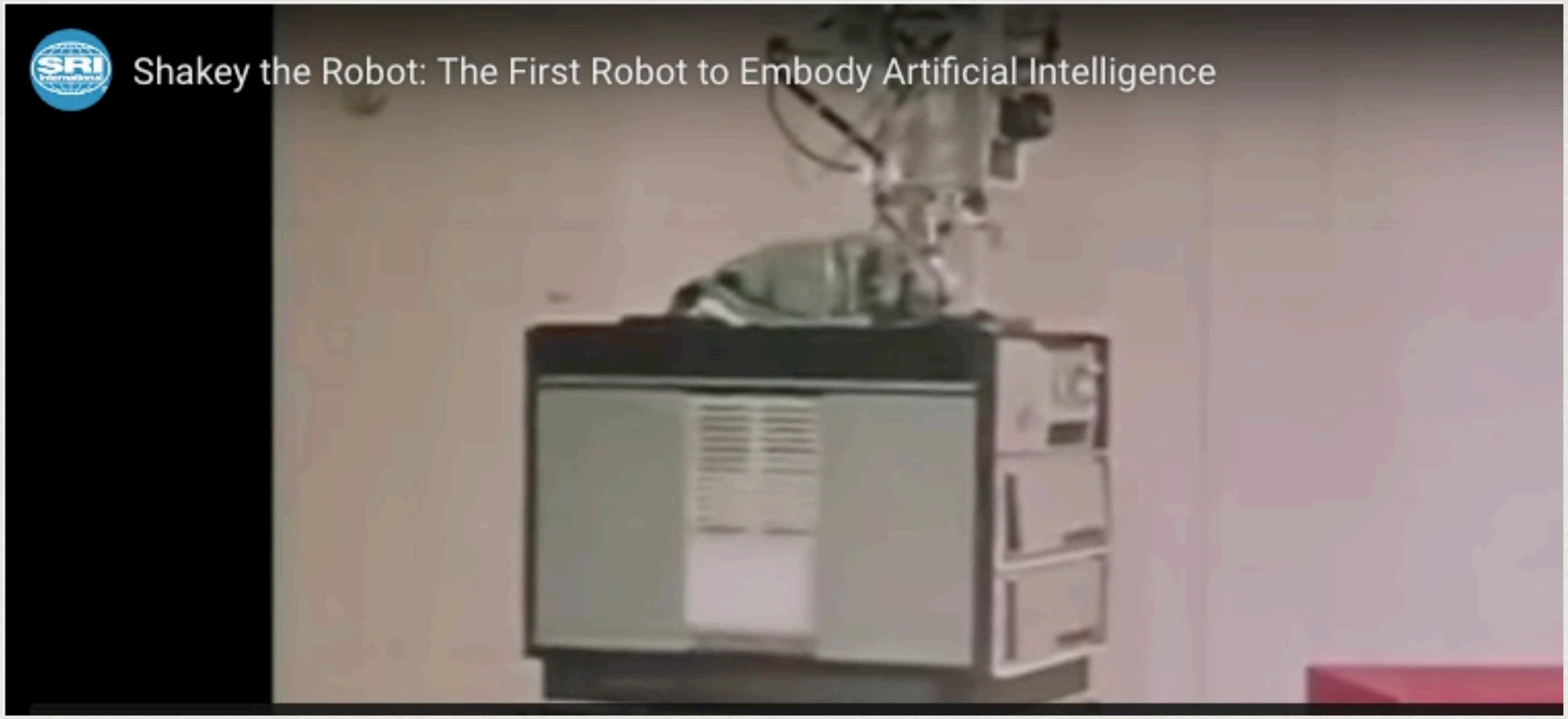
d-l@b

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.



Resolução de problemas com IA e o papel do SRI


SRI
Shakey the Robot: The First Robot to Embody Artificial Intelligence

Planejamento automático

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

métodos de busca clássica e informada

programação lógica

métodos de inferência

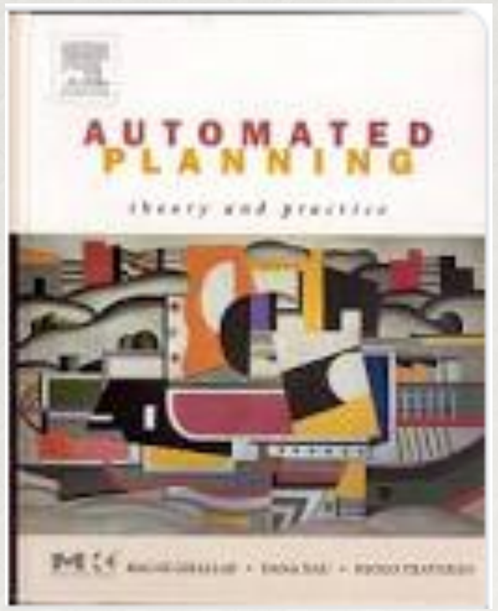
Automated Planning

"Planning is the reasoning side of acting. It is an abstract, explicit deliberation process that chooses and organizes actions by anticipating their expected outcomes. This deliberation aims at achieving as best as possible some pre-stated objectives. Automated planning is an area of Artificial Intelligence (AI) that studies this deliberation process computationally."

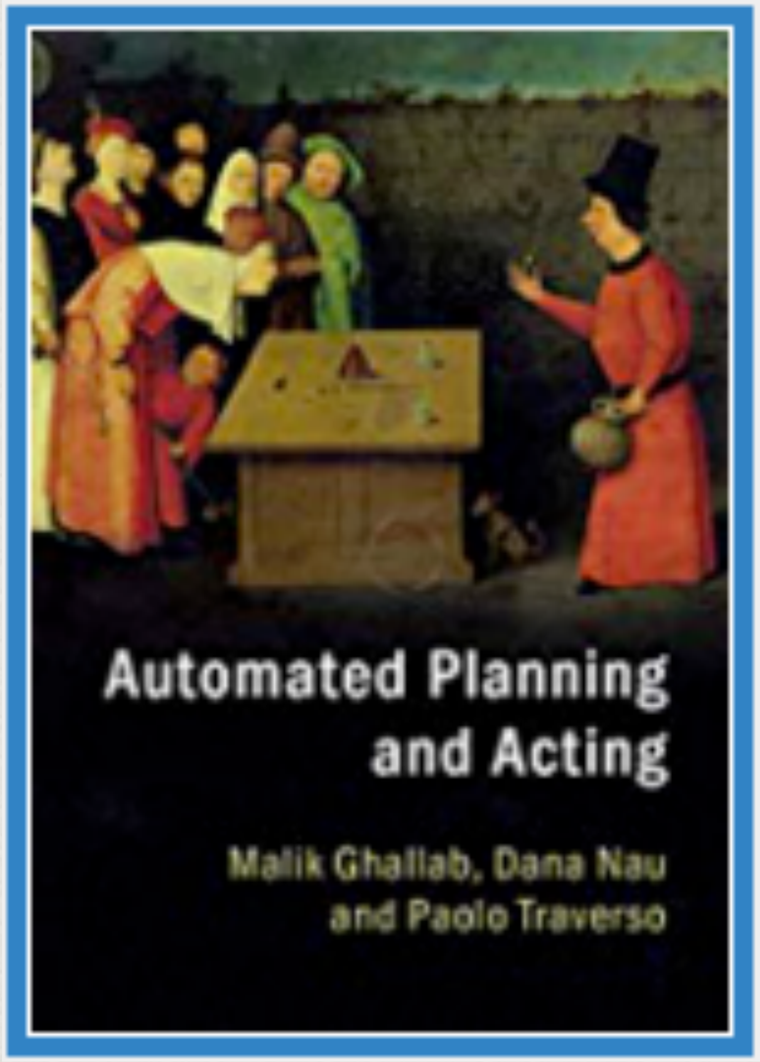
Malik Ghallab

Dana Nau

Paolo Traverso



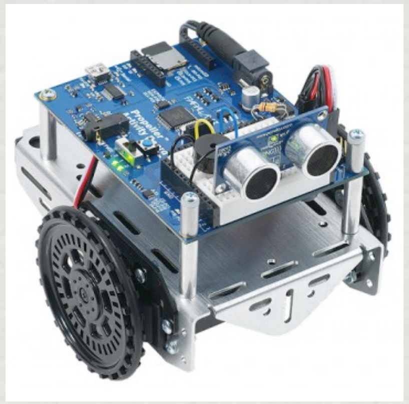
Morgan Kauffman, 2004


Automated Planning and Acting

Malik Ghallab, Dana Nau and Paolo Traverso

Cambridge University Press, 2016

Forms of (AI) planning: motion planning



Path and motion planning is concerned with the synthesis of a geometric path from a starting position in space to a goal and of a control trajectory along that path that specifies the state variables in the configuration space of a mobile system, such as a truck, a mechanical arm, a robot, or a virtual character.

Forms of (AI) planning: perception planning

Self-driving car

Perception planning is concerned with plans **involving sensing actions** for gathering information. It arises in tasks such as modelling environments or objects, identifying objects, localising through sensing a mobile system, or more generally identifying the current state of the environment. An example of these tasks is the design of a precise virtual model of an urban scene from a set of images.

Forms of (AI) planning: navigation planning

<u>Navigation planning</u> combines the two previous problems of <u>motion</u> and <u>perception</u> planning in order to reach a goal or to explore an area. The purpose of navigation planning is to synthesize a policy that combines localization primitives and sensor-based motion primitives, e.g., visually following a road until reaching some landmark, moving along some heading while avoiding obstacles, and so forth.

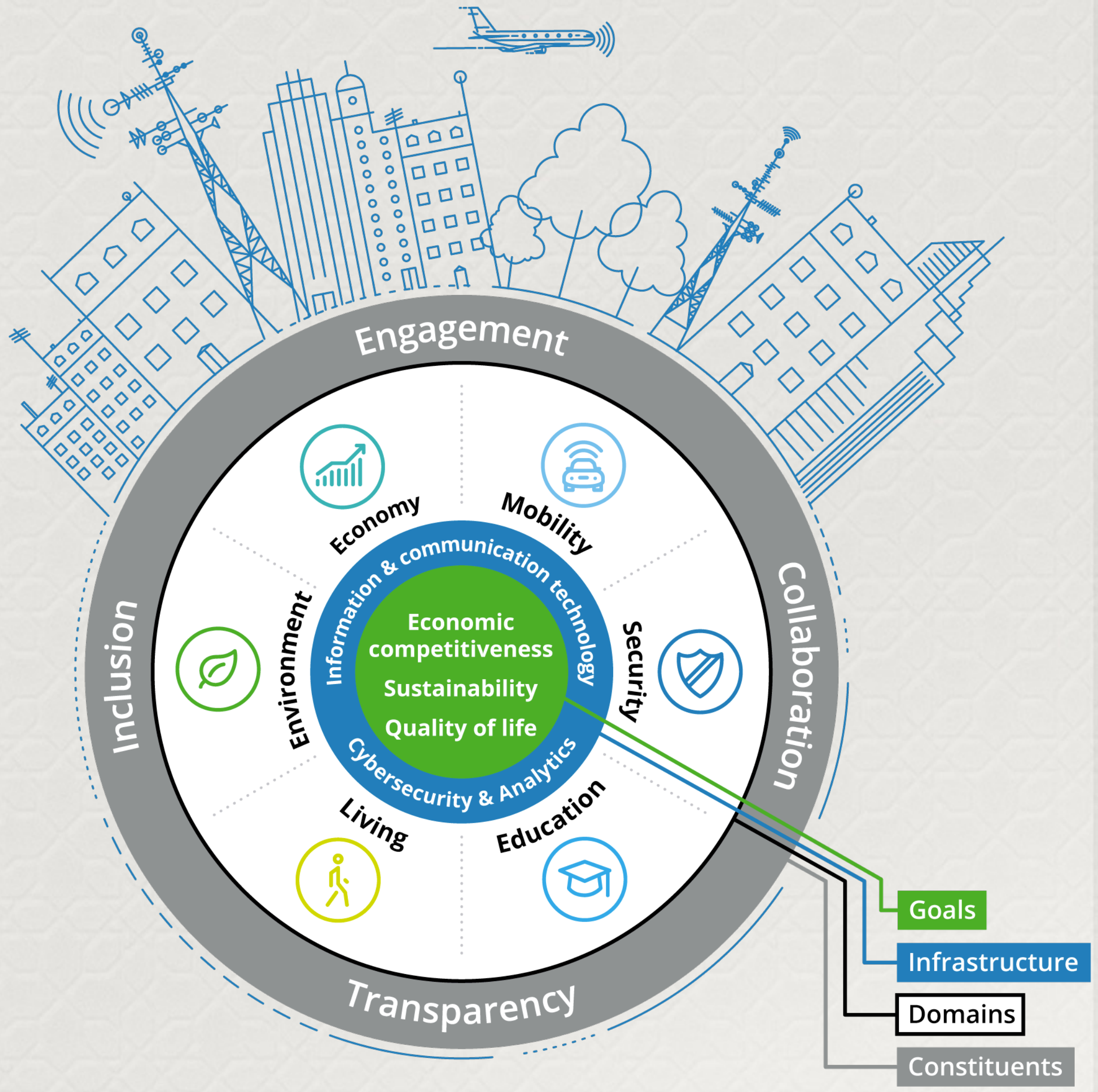
Forms of (AI) planning: manipulation planning

Manipulation planning is concerned with **handling objects, e.g., to build assemblies**. The actions include sensory-motor primitives that involve forces, touch, vision, range, and other sensory information. A plan might involve picking up an object from its marked sides, returning it if needed, inserting it into an assembly, and pushing lightly till it clips mechanically into position.

Forms of (AI) planning: communication planning

Communication planning arises in **dialog** and in **cooperation** problems between several agents, human or artificial. It addresses issues such as when and how to query needed information and which feedback should be provided.

Intelligent cities projects use AI planning



Relating AI Planning and Problem Solving

Domain-depend planning	One develops predictive models for the type of actions to be planned for and for the states of the system in which they take place. Computational tools for running these models, in order to predict and assess the effects of alternate actions and plans in various situations, exploit the specifics of the domain.
Domain-independent planning	Domain-independent planning relies on abstract, general models of actions. These models range from very simple ones that allow only for limited forms of reasoning to models with richer prediction capabilities. There are in particular the following forms of models and planning capabilities.

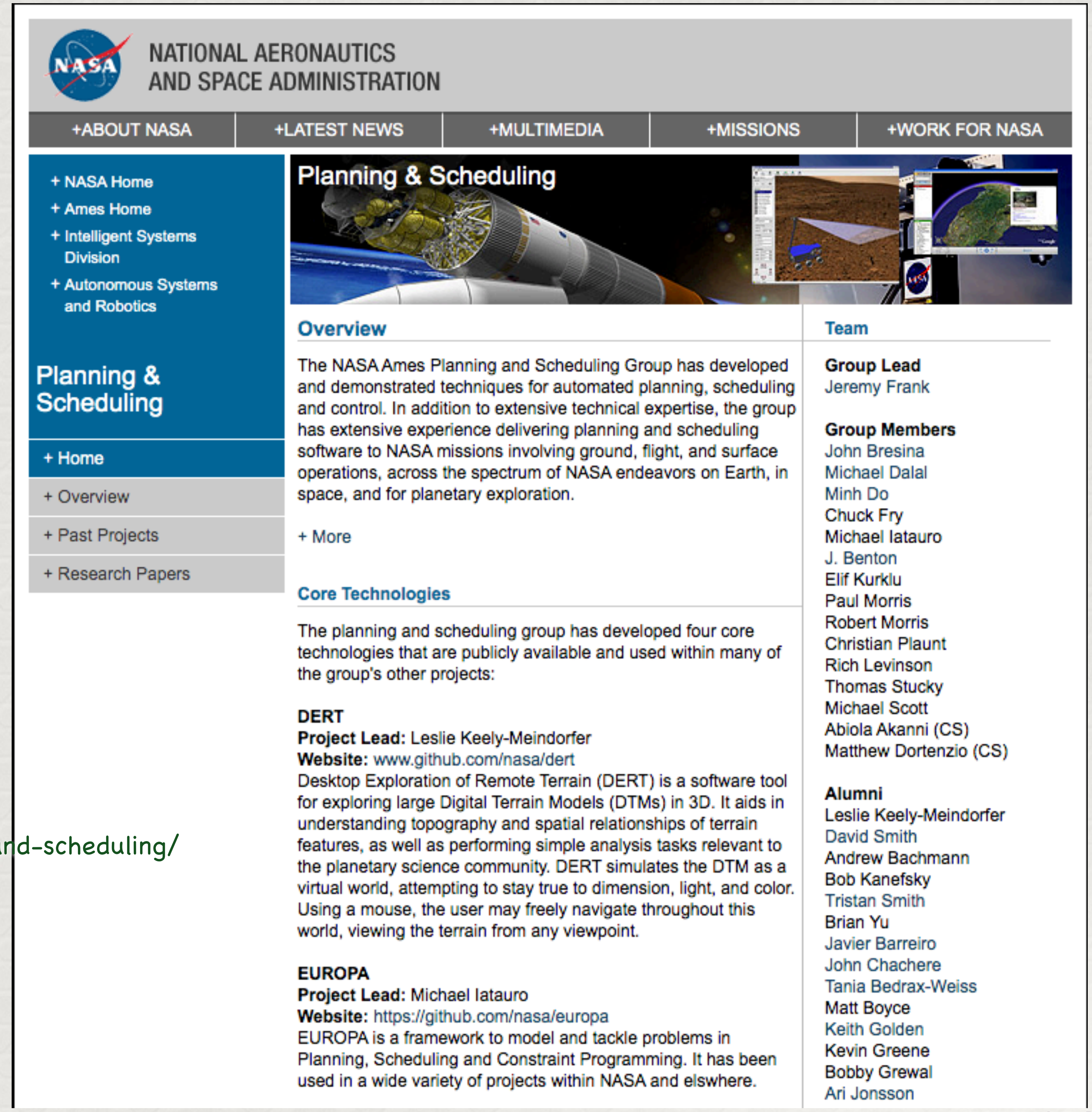
Type of modelling in planning

Project Planning	in which models of actions are reduced mainly to temporal and precedence constraints, e.g., the earliest and latest start times of an action or its latency with respect to another action.
Scheduling and resource allocation	in which the action models include the above types of constraints plus constraints on the resources to be used by each action.or its latency with respect to another action.
Plan Synthesis	in which the action models enrich the precedent models with the conditions needed for the applicability of an action and the effects of the action on the state of the world.

Deep Space 1

Europa - NASA

https://ti.arc.nasa.gov/tech/asr/groups/planning-and-scheduling/



NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Planning & Scheduling

Overview

The NASA Ames Planning and Scheduling Group has developed and demonstrated techniques for automated planning, scheduling and control. In addition to extensive technical expertise, the group has extensive experience delivering planning and scheduling software to NASA missions involving ground, flight, and surface operations, across the spectrum of NASA endeavors on Earth, in space, and for planetary exploration.

+ More

Core Technologies

The planning and scheduling group has developed four core technologies that are publicly available and used within many of the group's other projects:

DERT
Project Lead: Leslie Keely-Meindorfer
Website: www.github.com/nasa/dert
Desktop Exploration of Remote Terrain (DERT) is a software tool for exploring large Digital Terrain Models (DTMs) in 3D. It aids in understanding topography and spatial relationships of terrain features, as well as performing simple analysis tasks relevant to the planetary science community. DERT simulates the DTM as a virtual world, attempting to stay true to dimension, light, and color. Using a mouse, the user may freely navigate throughout this world, viewing the terrain from any viewpoint.

EUROPA
Project Lead: Michael Iatauro
Website: https://github.com/nasa/europa
EUROPA is a framework to model and tackle problems in Planning, Scheduling and Constraint Programming. It has been used in a wide variety of projects within NASA and elswhere.

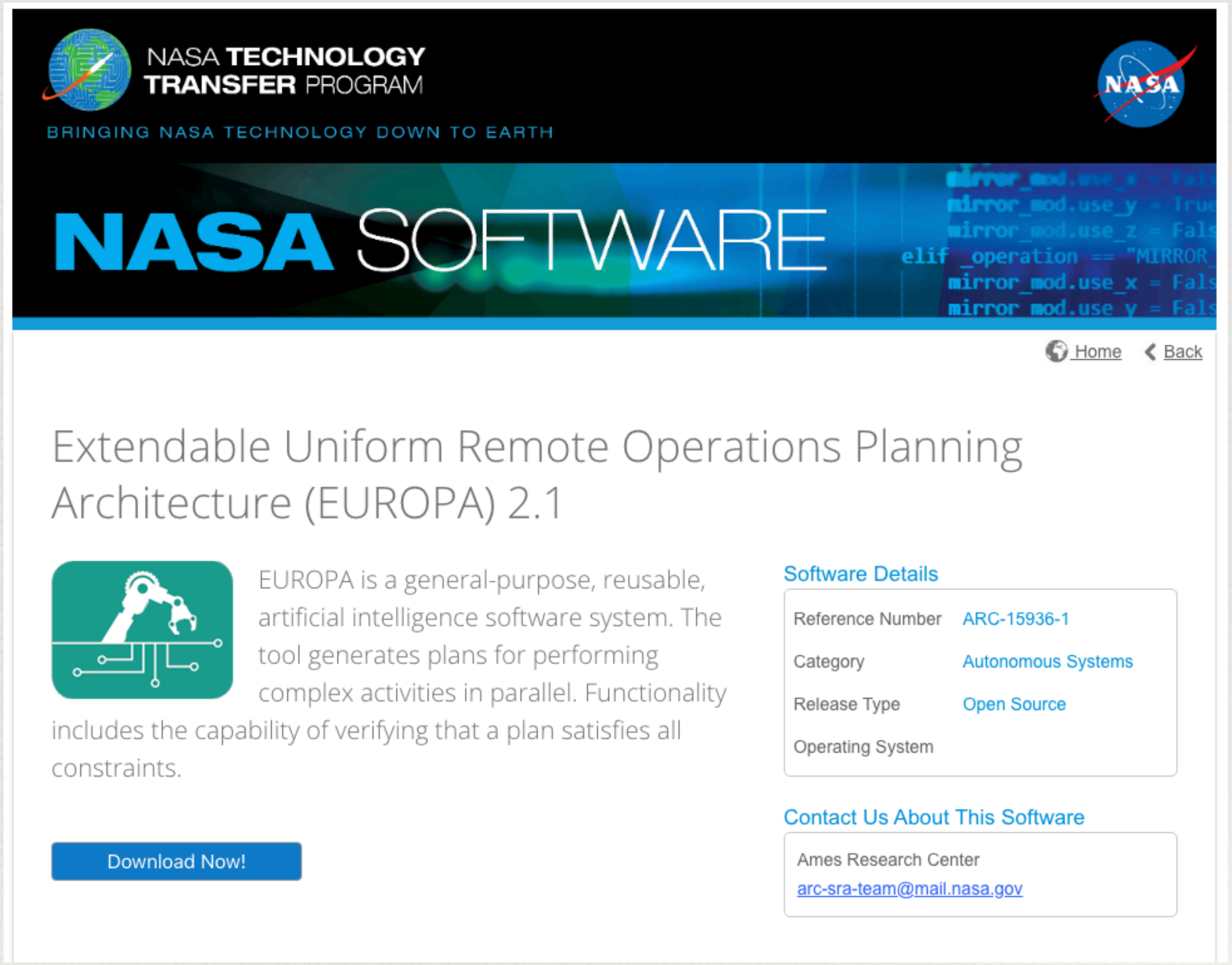
Team

Group Lead
Jeremy Frank

Group Members
John Bresina
Michael Dalal
Minh Do
Chuck Fry
Michael Iatauro
J. Benton
Elif Kurklu
Paul Morris
Robert Morris
Christian Plaunt
Rich Levinson
Thomas Stucky
Michael Scott
Abiola Akanni (CS)
Matthew Dortenzio (CS)

Alumni
Leslie Keely-Meindorfer
David Smith
Andrew Bachmann
Bob Kanefsky
Tristan Smith
Brian Yu
Javier Barreiro
John Chachere
Tania Bedrax-Weiss
Matt Boyce
Keith Golden
Kevin Greene
Bobby Grewal
Ari Jonsson

https://software.nasa.gov/software/ARC-15936-1


NASA TECHNOLOGY TRANSFER PROGRAM
BRINGING NASA TECHNOLOGY DOWN TO EARTH
NASA SOFTWARE
Home
Back
Download Now!

Extendable Uniform Remote Operations Planning Architecture (EUROPA) 2.1

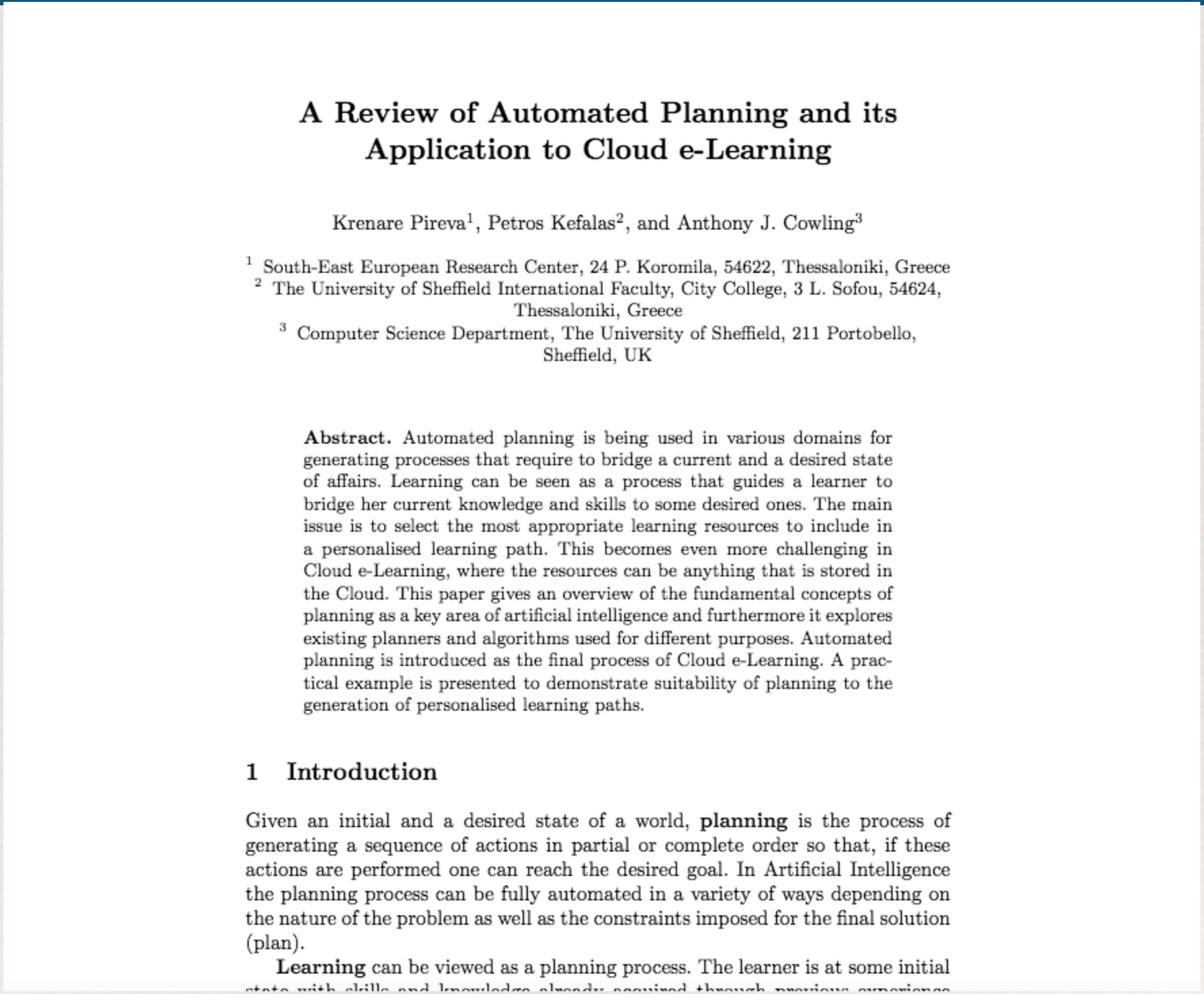
EUROPA is a general-purpose, reusable, artificial intelligence software system. The tool generates plans for performing complex activities in parallel. Functionality includes the capability of verifying that a plan satisfies all constraints.

Software Details

Reference Number	ARC-15936-1
Category	Autonomous Systems
Release Type	Open Source
Operating System	

Contact Us About This Software

Ames Research Center
arc-sra-team@mail.nasa.gov



A Review of Automated Planning and its Application to Cloud e-Learning

Krenare Pireva[1], Petros Kefalas[2], and Anthony J. Cowling[3]

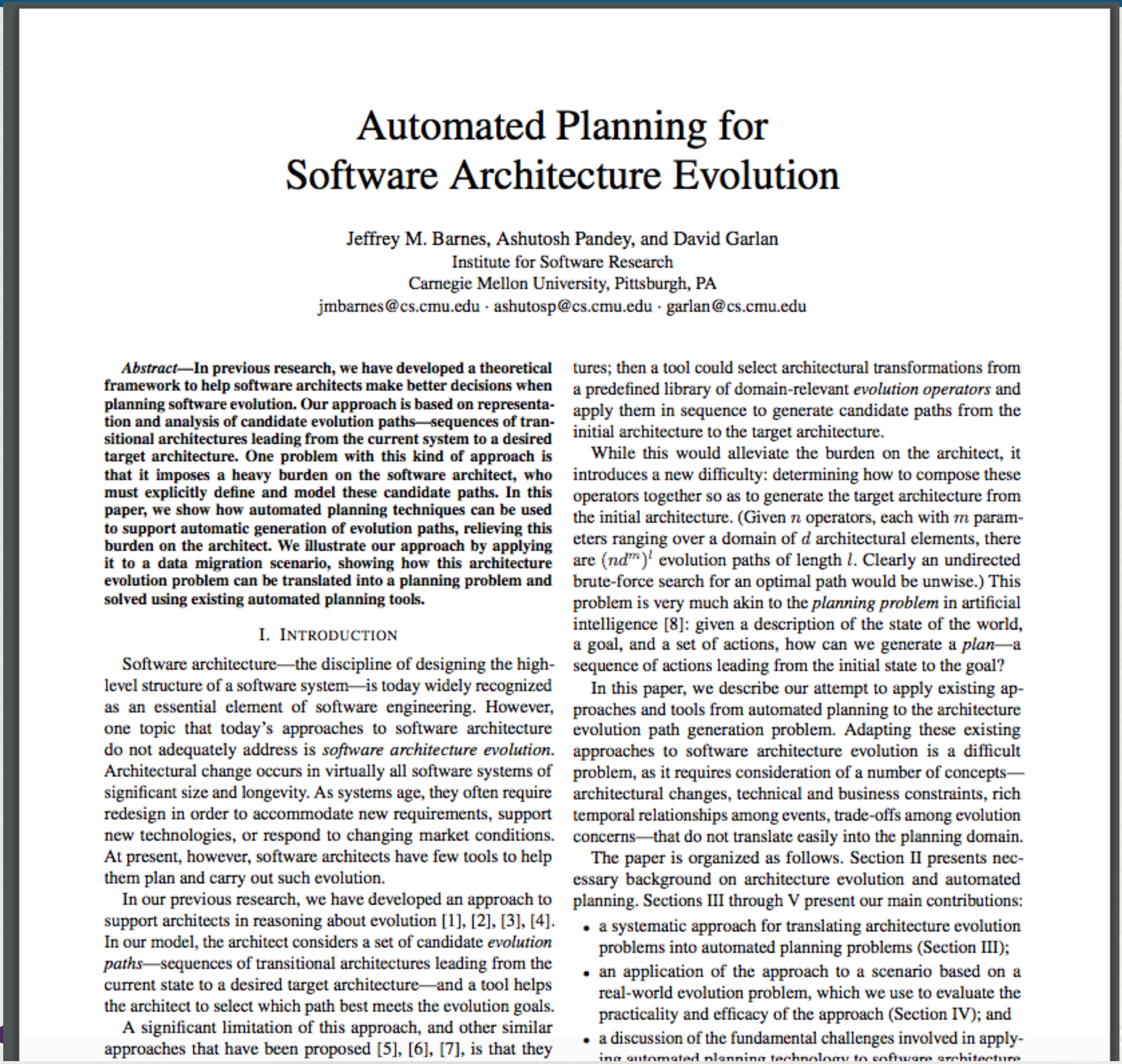
[1] South-East European Research Center, 24 P. Koromila, 54622, Thessaloniki, Greece
[2] The University of Sheffield International Faculty, City College, 3 L. Sofou, 54624, Thessaloniki, Greece
[3] Computer Science Department, The University of Sheffield, 211 Portobello, Sheffield, UK

Abstract. Automated planning is being used in various domains for generating processes that require to bridge a current and a desired state of affairs. Learning can be seen as a process that guides a learner to bridge her current knowledge and skills to some desired ones. The main issue is to select the most appropriate learning resources to include in a personalised learning path. This becomes even more challenging in Cloud e-Learning, where the resources can be anything that is stored in the Cloud. This paper gives an overview of the fundamental concepts of planning as a key area of artificial intelligence and furthermore it explores existing planners and algorithms used for different purposes. Automated planning is introduced as the final process of Cloud e-Learning. A practical example is presented to demonstrate suitability of planning to the generation of personalised learning paths.

1 Introduction

Given an initial and a desired state of a world, **planning** is the process of generating a sequence of actions in partial or complete order so that, if these actions are performed one can reach the desired goal. In Artificial Intelligence the planning process can be fully automated in a variety of ways depending on the nature of the problem as well as the constraints imposed for the final solution (plan).

Learning can be viewed as a planning process. The learner is at some initial



Automated Planning for Software Architecture Evolution

Jeffrey M. Barnes, Ashutosh Pandey, and David Garlan
Institute for Software Research
Carnegie Mellon University, Pittsburgh, PA
jmbarnes@cs.cmu.edu · ashutosp@cs.cmu.edu · garlan@cs.cmu.edu

***Abstract*—In previous research, we have developed a theoretical framework to help software architects make better decisions when planning software evolution. Our approach is based on representation and analysis of candidate evolution paths—sequences of transitional architectures leading from the current system to a desired target architecture. One problem with this kind of approach is that it imposes a heavy burden on the software architect, who must explicitly define and model these candidate paths. In this paper, we show how automated planning techniques can be used to support automatic generation of evolution paths, relieving this burden on the architect. We illustrate our approach by applying it to a data migration scenario, showing how this architecture evolution problem can be translated into a planning problem and solved using existing automated planning tools.**

I. Introduction

Software architecture—the discipline of designing the high-level structure of a software system—is today widely recognized as an essential element of software engineering. However, one topic that today's approaches to software architecture do not adequately address is *software architecture evolution*. Architectural change occurs in virtually all software systems of significant size and longevity. As systems age, they often require redesign in order to accommodate new requirements, support new technologies, or respond to changing market conditions. At present, however, software architects have few tools to help them plan and carry out such evolution.

In our previous research, we have developed an approach to support architects in reasoning about evolution [1], [2], [3], [4]. In our model, the architect considers a set of candidate *evolution paths*—sequences of transitional architectures leading from the current state to a desired target architecture—and a tool helps the architect to select which path best meets the evolution goals.

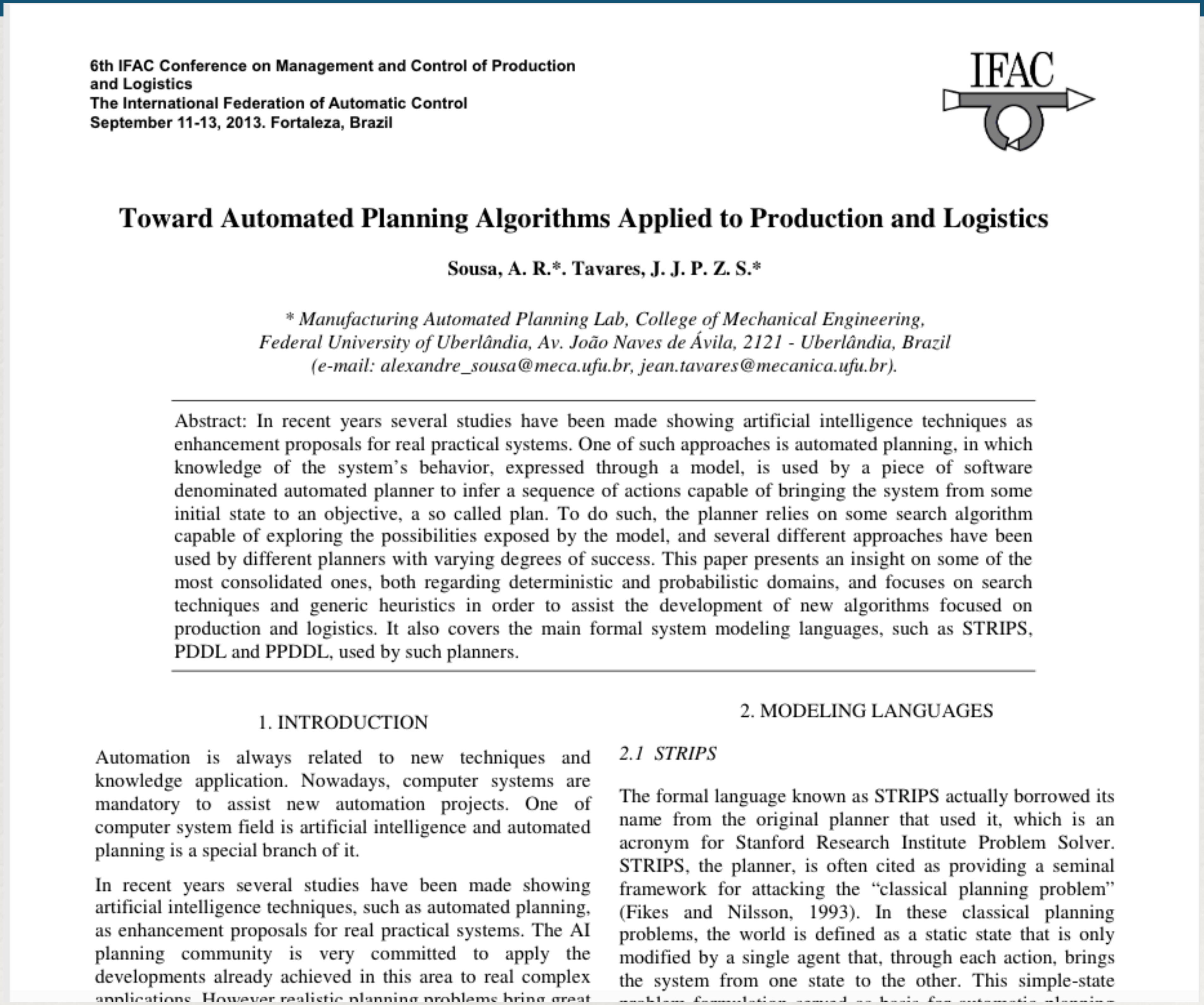
A significant limitation of this approach, and other similar approaches that have been proposed [5], [6], [7], is that they tures; then a tool could select architectural transformations from a predefined library of domain-relevant *evolution operators* and apply them in sequence to generate candidate paths from the initial architecture to the target architecture.

While this would alleviate the burden on the architect, it introduces a new difficulty: determining how to compose these operators together so as to generate the target architecture from the initial architecture. (Given n operators, each with m parameters ranging over a domain of d architectural elements, there are $(nd^m)^l$ evolution paths of length l. Clearly an undirected brute-force search for an optimal path would be unwise.) This problem is very much akin to the *planning problem* in artificial intelligence [8]: given a description of the state of the world, a goal, and a set of actions, how can we generate a *plan*—a sequence of actions leading from the initial state to the goal?

In this paper, we describe our attempt to apply existing approaches and tools from automated planning to the architecture evolution path generation problem. Adapting these existing approaches to software architecture evolution is a difficult problem, as it requires consideration of a number of concepts—architectural changes, technical and business constraints, rich temporal relationships among events, trade-offs among evolution concerns—that do not translate easily into the planning domain.

The paper is organized as follows. Section II presents necessary background on architecture evolution and automated planning. Sections III through V present our main contributions:

- a systematic approach for translating architecture evolution problems into automated planning problems (Section III);
- an application of the approach to a scenario based on a real-world evolution problem, which we use to evaluate the practicality and efficacy of the approach (Section IV); and
- a discussion of the fundamental challenges involved in apply-



6th IFAC Conference on Management and Control of Production and Logistics
The International Federation of Automatic Control
September 11-13, 2013. Fortaleza, Brazil

IFAC

Toward Automated Planning Algorithms Applied to Production and Logistics

Sousa, A. R.*. Tavares, J. J. P. Z. S.*

* *Manufacturing Automated Planning Lab, College of Mechanical Engineering, Federal University of Uberlândia, Av. João Naves de Ávila, 2121 - Uberlândia, Brazil (e-mail: alexandre_sousa@meca.ufu.br, jean.tavares@mecanica.ufu.br).*

Abstract: In recent years several studies have been made showing artificial intelligence techniques as enhancement proposals for real practical systems. One of such approaches is automated planning, in which knowledge of the system's behavior, expressed through a model, is used by a piece of software denominated automated planner to infer a sequence of actions capable of bringing the system from some initial state to an objective, a so called plan. To do such, the planner relies on some search algorithm capable of exploring the possibilities exposed by the model, and several different approaches have been used by different planners with varying degrees of success. This paper presents an insight on some of the most consolidated ones, both regarding deterministic and probabilistic domains, and focuses on search techniques and generic heuristics in order to assist the development of new algorithms focused on production and logistics. It also covers the main formal system modeling languages, such as STRIPS, PDDL and PPDDL, used by such planners.

1. INTRODUCTION

Automation is always related to new techniques and knowledge application. Nowadays, computer systems are mandatory to assist new automation projects. One of computer system field is artificial intelligence and automated planning is a special branch of it.

In recent years several studies have been made showing artificial intelligence techniques, such as automated planning, as enhancement proposals for real practical systems. The AI planning community is very committed to apply the developments already achieved in this area to real complex applications. However realistic planning problems bring great

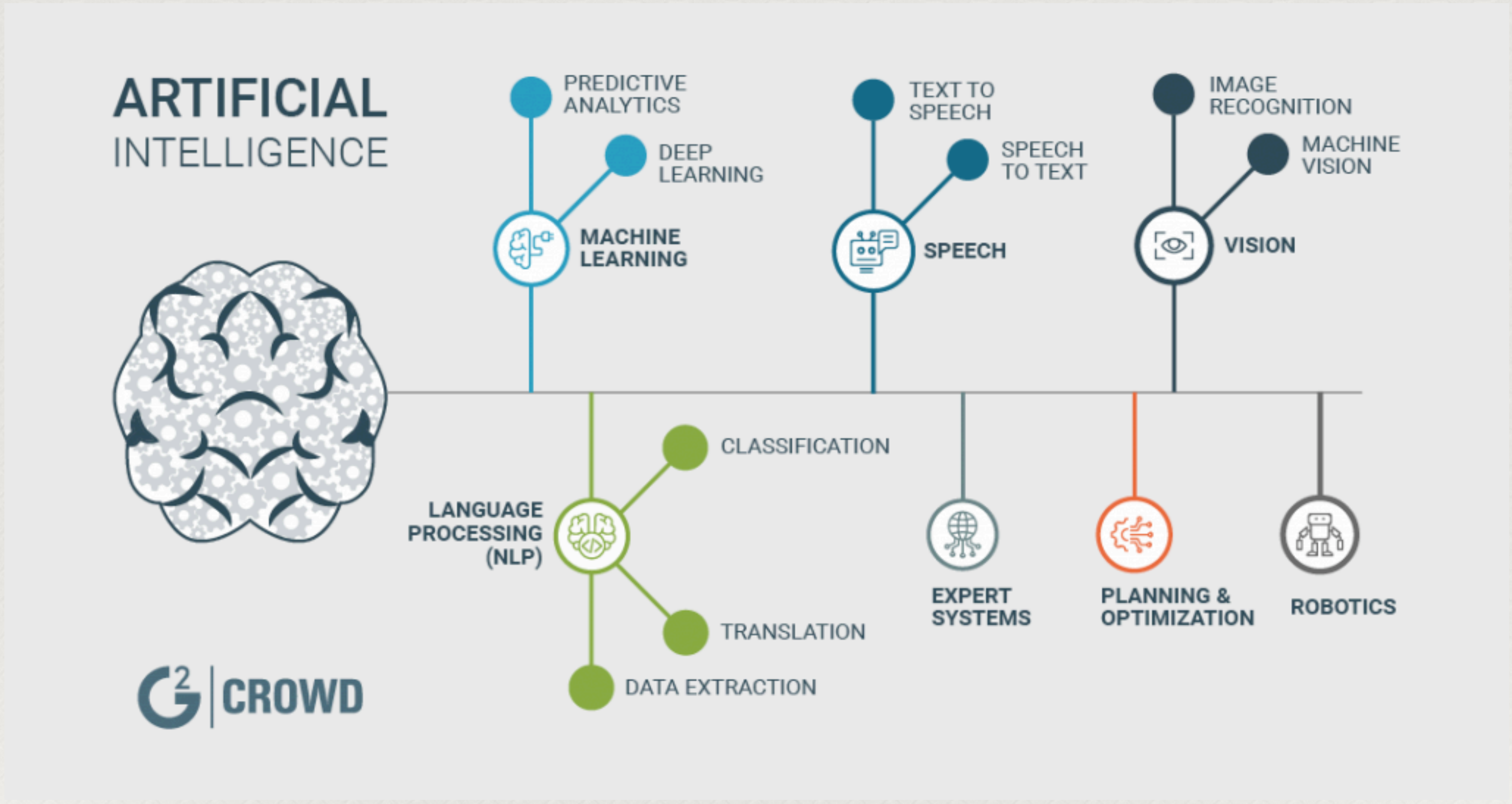
2. MODELING LANGUAGES

2.1 STRIPS

The formal language known as STRIPS actually borrowed its name from the original planner that used it, which is an acronym for Stanford Research Institute Problem Solver. STRIPS, the planner, is often cited as providing a seminal framework for attacking the "classical planning problem" (Fikes and Nilsson, 1993). In these classical planning problems, the world is defined as a static state that is only modified by a single agent that, through each action, brings the system from one state to the other. This simple-state

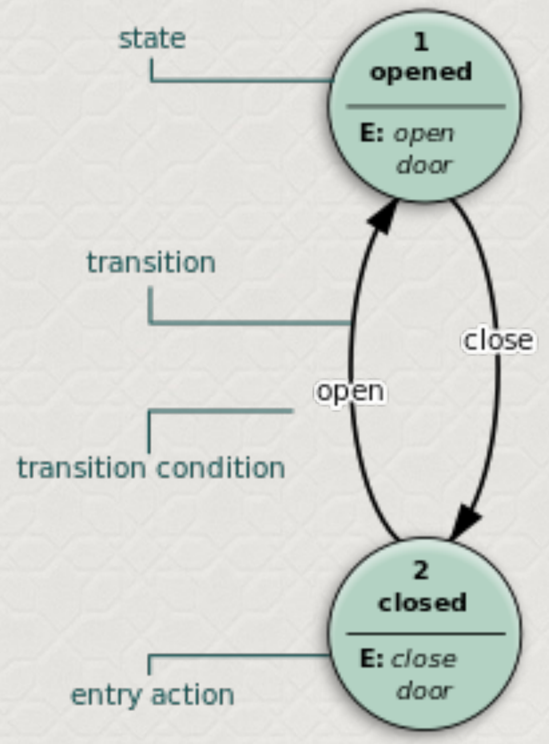
The logistic of containers in ports

Space is valuable in port operation as well as the operation of storing and retrieving the proper container.


ARTIFICIAL
INTELLIGENCE
PREDICTIVE ANALYTICS
DEEP LEARNING
MACHINE LEARNING
TEXT TO SPEECH
SPEECH TO TEXT
SPEECH
IMAGE RECOGNITION
MACHINE VISION
VISION
LANGUAGE PROCESSING (NLP)
CLASSIFICATION
TRANSLATION
DATA EXTRACTION
EXPERT SYSTEMS
PLANNING & OPTIMIZATION
ROBOTICS
G2 CROWD

Summarising planning concepts

The most important point is that automatic planning was created as an **inspired problem solving methods**, and therefore had to be adherent to a problem solving method and basic formalism. This formalism is based on **Transition Systems**.



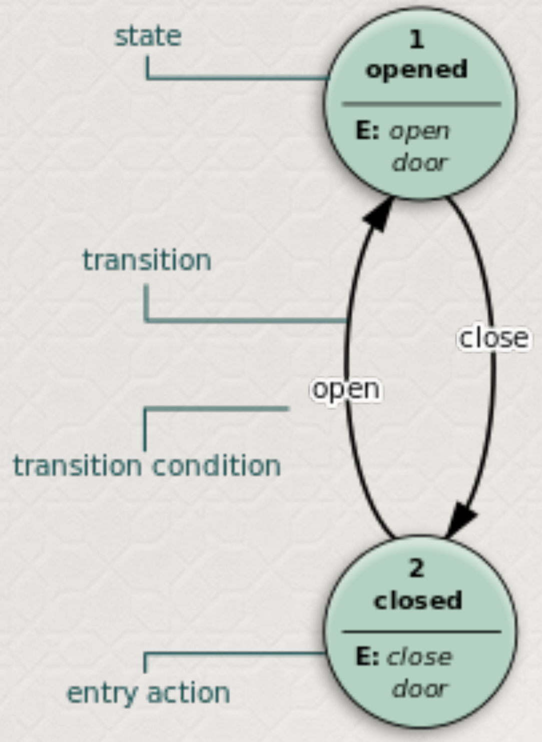
A conceptual modelling for planning

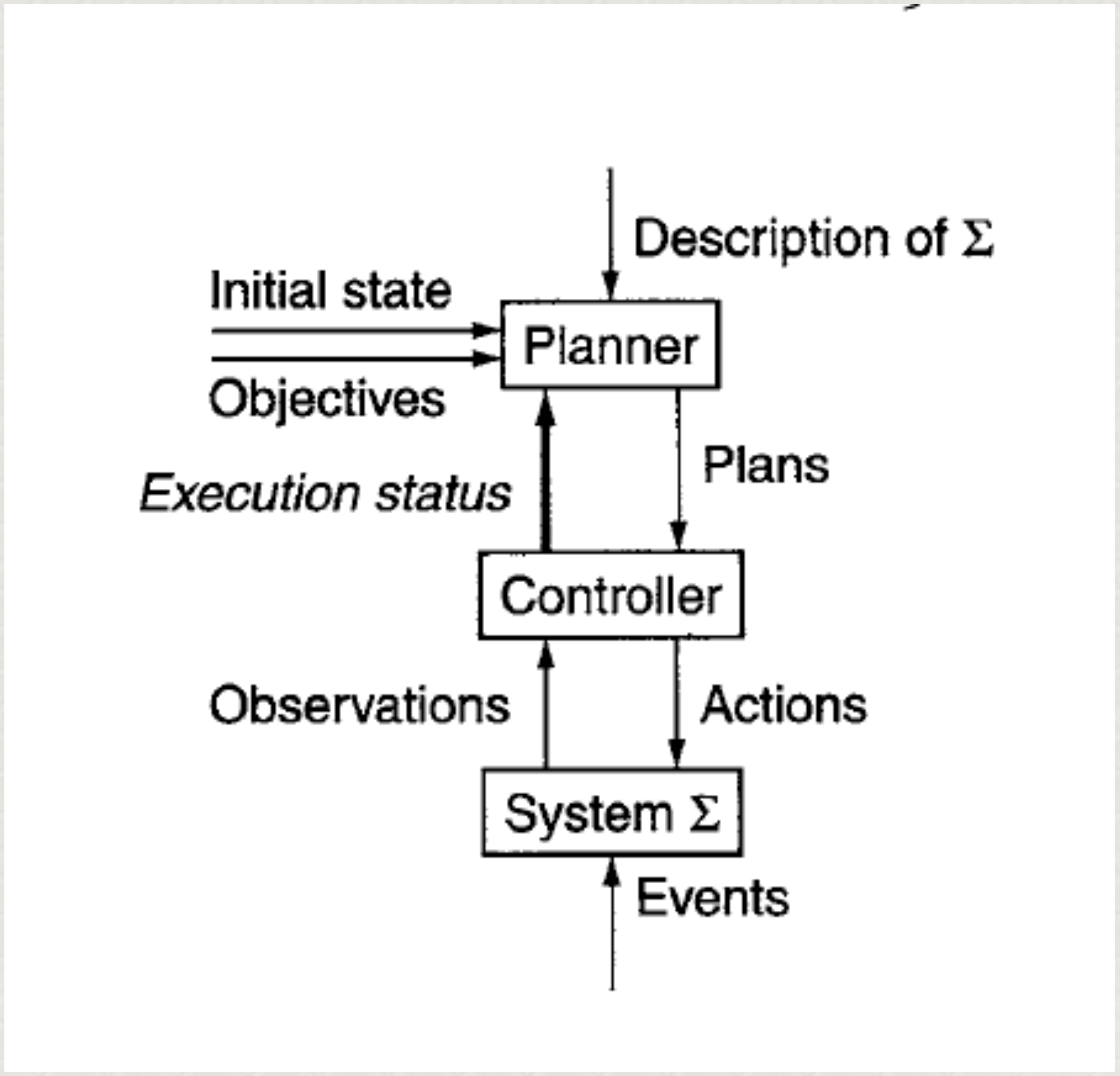
Transition Systems are also known as State-Transition Systems and are in the basis of the discrete systems approach.

Formally, a state-transition system is a 4-tuple E = (S, A, E, y), where:

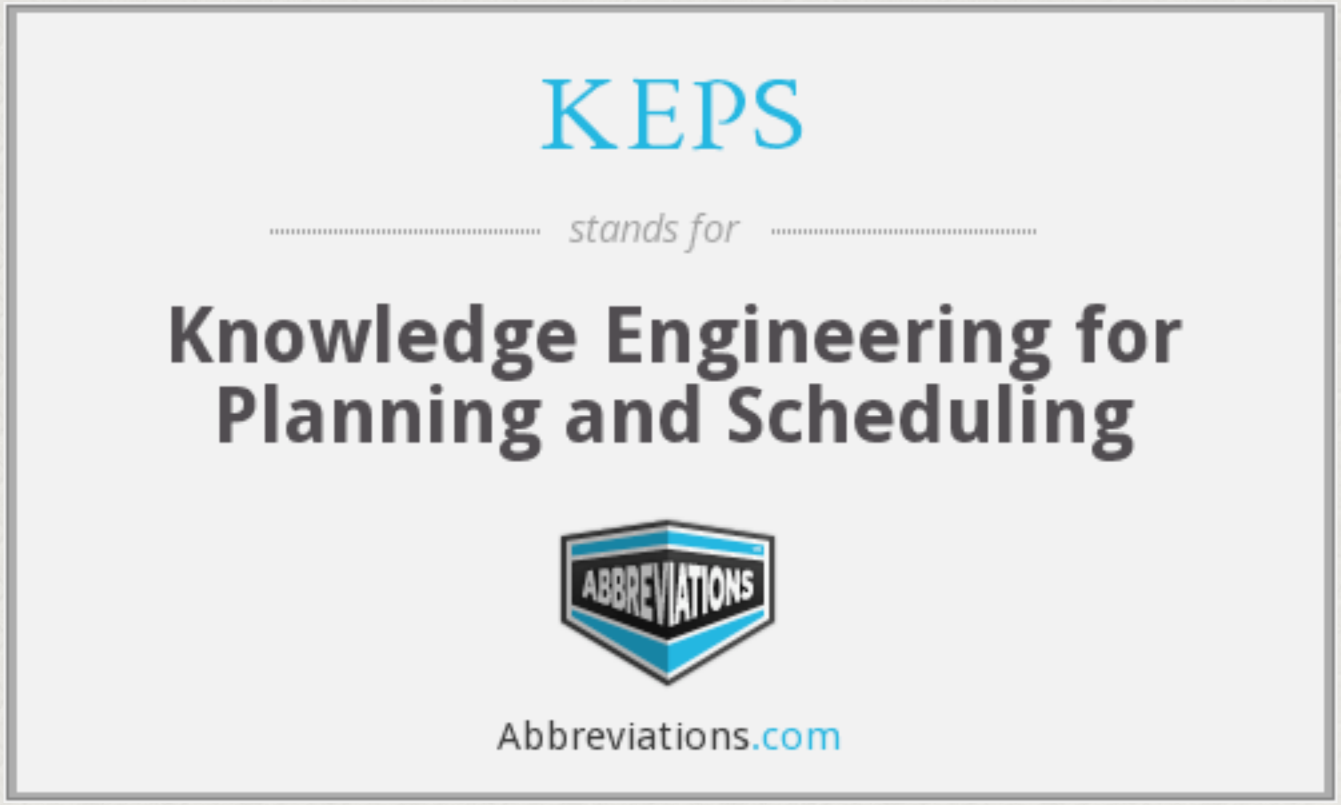
- S = {Sl, S2 . . . . } is a finite or recursively enumerable set of states;
- A = {al, a2,...} is a finite or recursively enumerable set of actions;
- E = {el, e2,...} is a finite or recursively enumerable set of events; and
- $\gamma : S \times A \times E \rightarrow 2^S$ is a state-transition function.

A state-transition system may be represented by a directed graph whose nodes are the states in S. If $s' \in \gamma(s, u)$, where u is a pair (a, e), $a \in A$ and $e \in E$, then the graph contains an arc from s to s' that is labeled with u. Each such arc is called a state transition.




Description of Σ
Initial state
Planner
Objectives
Execution status
Plans
Controller
Observations
Actions
System Σ
Events

icaps



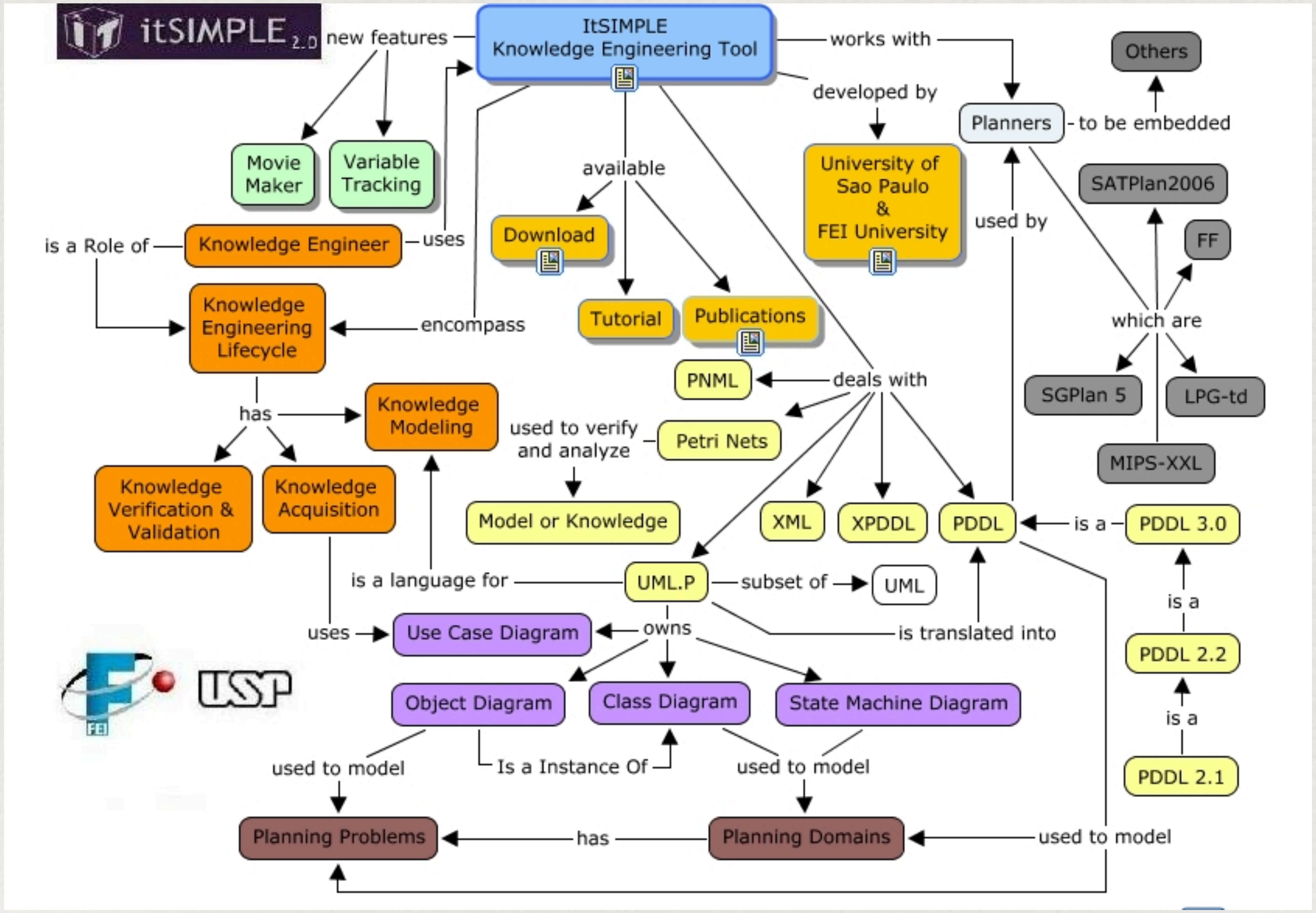
KEPS

stands for

Knowledge Engineering for Planning and Scheduling

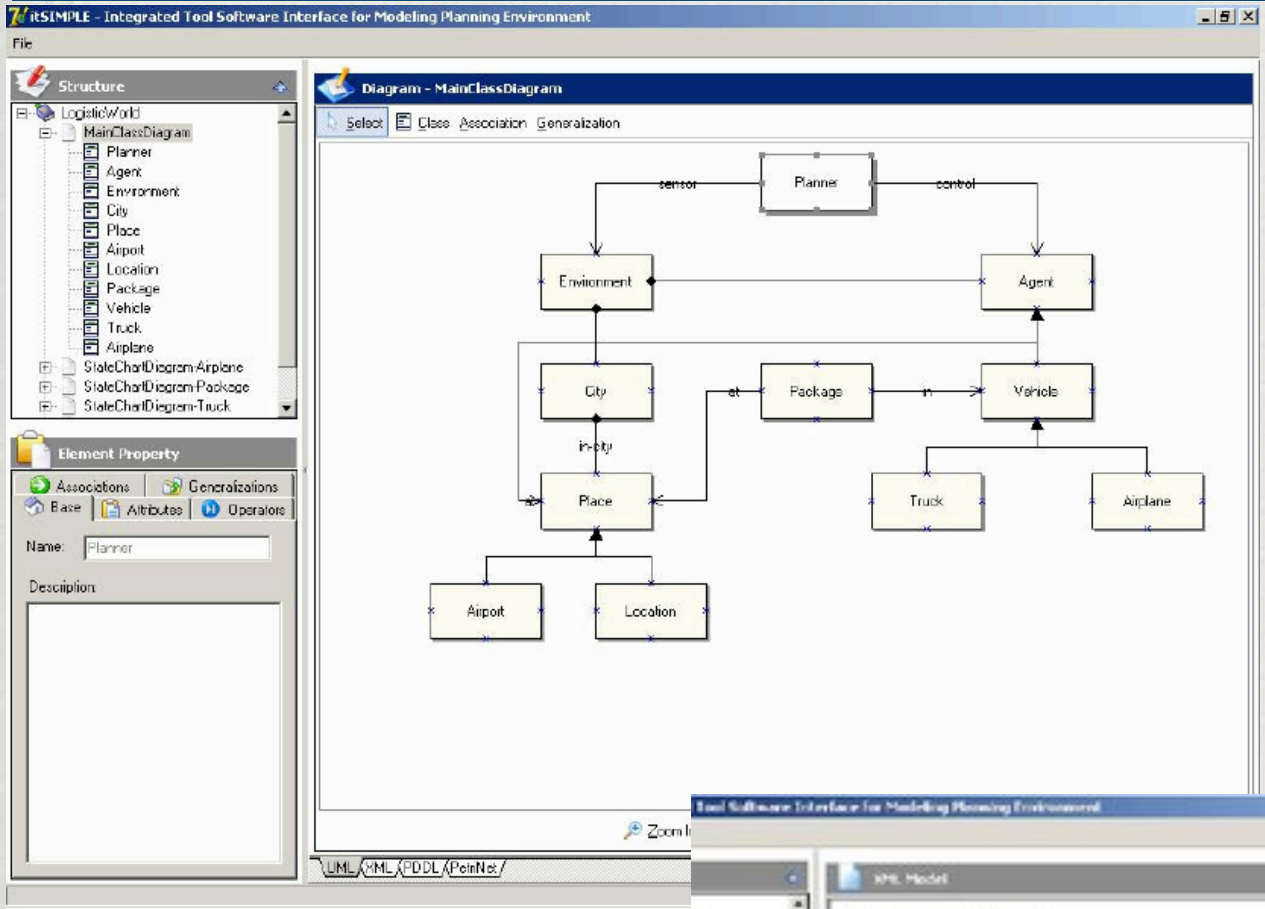
ABBREVIATIONS

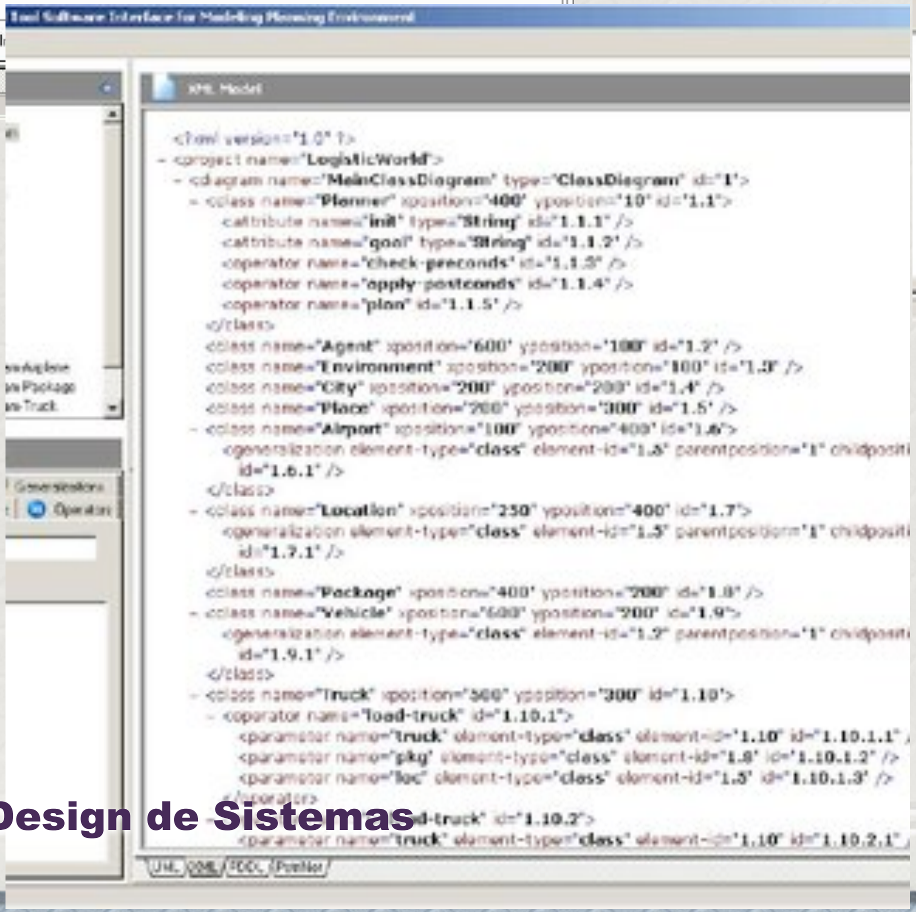
Abbreviations.com


itSIMPLE 2.0
new features
ItSIMPLE Knowledge Engineering Tool
works with
developed by
Others
Planners
to be embedded
Movie Maker
Variable Tracking
available
University of Sao Paulo & FEI University
SATPlan2006
used by
FF
is a Role of
Knowledge Engineer
uses
Download
Knowledge Engineering Lifecycle
encompass
Tutorial
Publications
which are
PNML
deals with
SGPlan 5
LPG-td
has
Knowledge Modeling
used to verify and analyze
Petri Nets
MIPS-XXL
Knowledge Verification & Validation
Knowledge Acquisition
Model or Knowledge
XML
XPDDL
PDDL
is a
PDDL 3.0
is a language for
UML.P
subset of
UML
is a
uses
Use Case Diagram
owns
is translated into
PDDL 2.2
is a
Object Diagram
Class Diagram
State Machine Diagram
PDDL 2.1
used to model
Is a Instance Of
used to model
Planning Problems
has
Planning Domains
used to model
USP

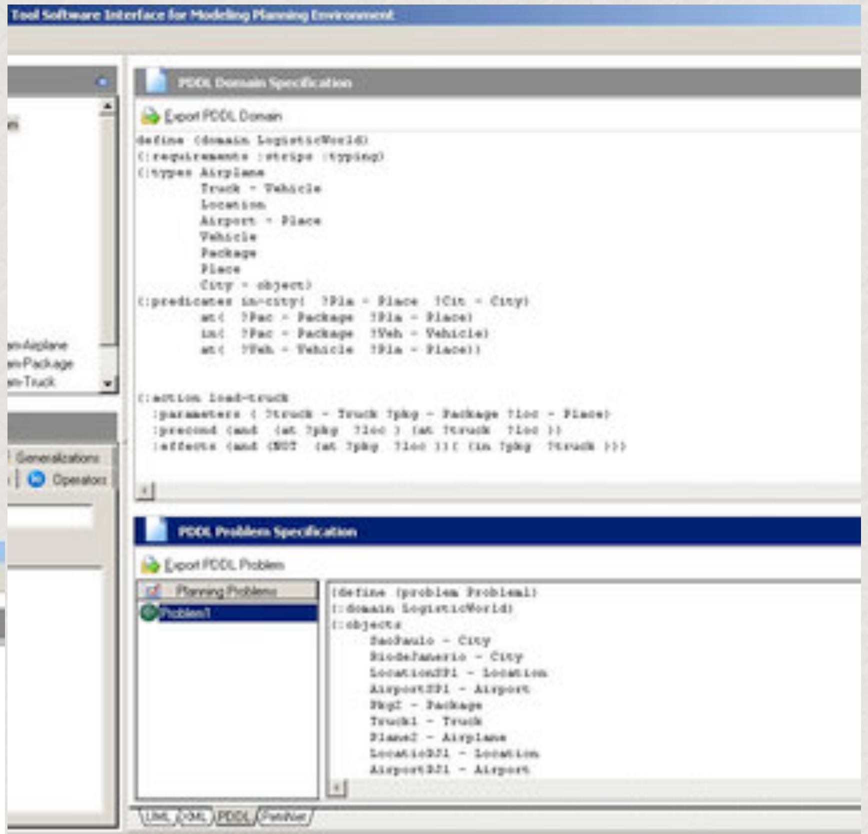
itSIMPLE

itSIMPLE





```
<?xml version="1.0" ?>
- <project name="LogisticWorld">
  - <diagram name="MainClassDiagram" type="ClassDiagram" id="1">
    - <class name="Planner" xposition="400" yposition="10" id="1.1">
        <attribute name="init" type="String" id="1.1.1" />
        <attribute name="goal" type="String" id="1.1.2" />
        <operator name="check-preconds" id="1.1.3" />
        <operator name="apply-postconds" id="1.1.4" />
        <operator name="plan" id="1.1.5" />
      </class>
      <class name="Agent" xposition="600" yposition="100" id="1.2" />
      <class name="Environment" xposition="200" yposition="100" id="1.3" />
      <class name="City" xposition="200" yposition="200" id="1.4" />
      <class name="Place" xposition="200" yposition="300" id="1.5" />
    - <class name="Airport" xposition="100" yposition="400" id="1.6">
        <generalization element-type="class" element-id="1.5" parentposition="1" childpositi
          id="1.6.1" />
      </class>
    - <class name="Location" xposition="250" yposition="400" id="1.7">
        <generalization element-type="class" element-id="1.5" parentposition="1" childpositi
          id="1.7.1" />
      </class>
      <class name="Package" xposition="400" yposition="200" id="1.8" />
    - <class name="Vehicle" xposition="600" yposition="200" id="1.9">
        <generalization element-type="class" element-id="1.2" parentposition="1" childpositi
          id="1.9.1" />
      </class>
    - <class name="Truck" xposition="500" yposition="300" id="1.10">
      - <operator name="load-truck" id="1.10.1">
          <parameter name="truck" element-type="class" element-id="1.10" id="1.10.1.1" /
          <parameter name="pkg" element-type="class" element-id="1.8" id="1.10.1.2" />
          <parameter name="loc" element-type="class" element-id="1.5" id="1.10.1.3" />
        </operator>
      - [illegible]d-truck" id="1.10.2">
          <parameter name="truck" element-type="class" element-id="1.10" id="1.10.2.1" /
```


Tool Software Interface for Modeling Planning Environment
PDDL Domain Specification
Export PDDL Domain
PDDL Problem Specification
Export PDDL Problem
Planning Problems
Problem1
Generalizations
Operators
UML
XML
PDDL
PetriNet

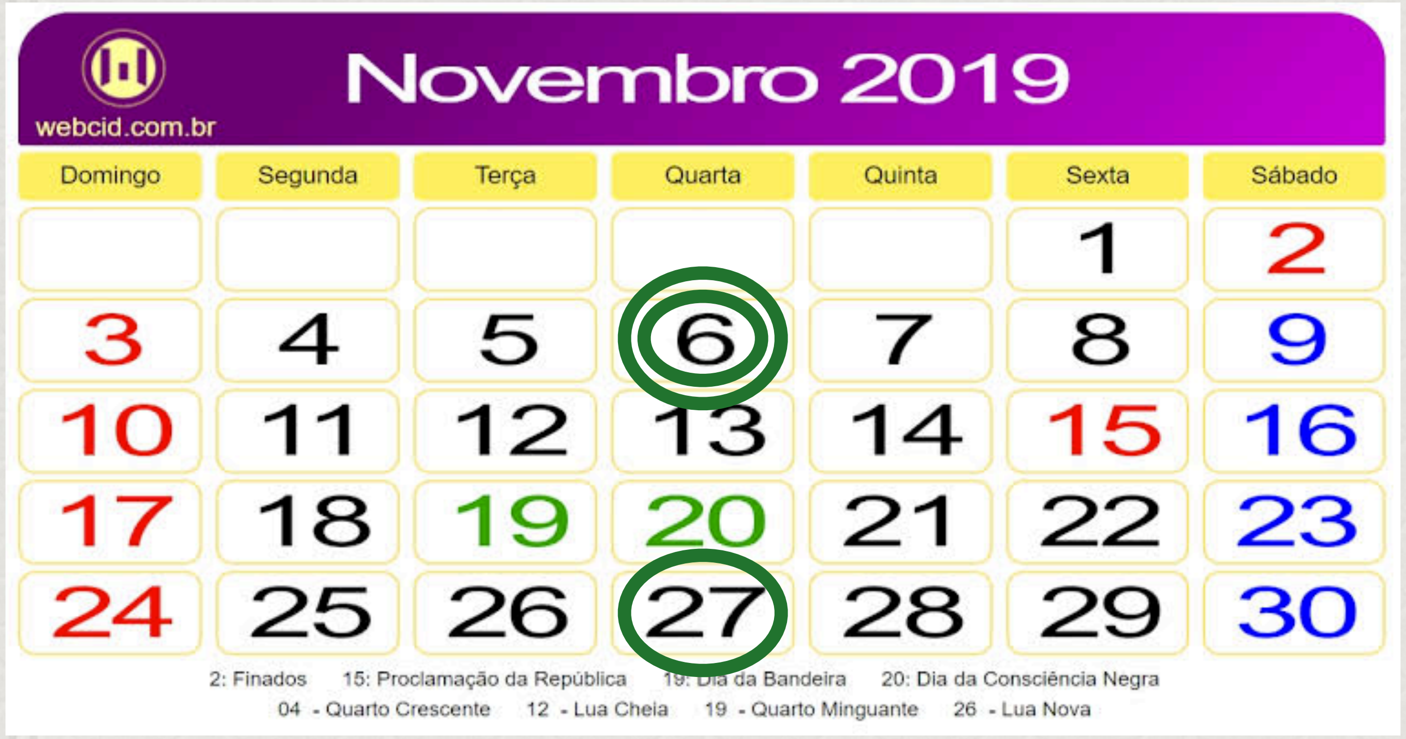
```
define (domain LogisticWorld)
(:requirements :strips :typing)
(:types Airplane
        Truck - Vehicle
        Location
        Airport - Place
        Vehicle
        Package
        Place
        City - object)
(:predicates in-city(  ?Pla - Place  ?Cit - City)
        at(  ?Pac - Package  ?Pla - Place)
        in(  ?Pac - Package  ?Veh - Vehicle)
        at(  ?Veh - Vehicle  ?Pla - Place))


(:action load-truck
  :parameters ( ?truck - Truck ?pkg - Package ?loc - Place)
  :precond (and  (at ?pkg  ?loc ) (at ?truck  ?loc ))
  :effects (and (NOT  (at ?pkg  ?loc ))( (in ?pkg  ?truck )))
```

```
(define (problem Problem1)
(:domain LogisticWorld)
(:objects
      SaoPaulo - City
      RioDeJaneiro - City
      LocationSP1 - Location
      AirportSP1 - Airport
      Pkg2 - Package
      Truck1 - Truck
      Plane2 - Airplane
      LocatioRJ1 - Location
      AirportRJ1 - Airport
```

In our next class (after the competition) we will explore a little further logic programming, how they supported Expert Systems in the past (80's) and what we should expect from this "classic AI" today and in the near future.

O novo cronograma



webcid.com.br

Novembro 2019

Domingo	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
					1	2
3	4	5	6	7	8	9
10	11	12	13	14	15	16
17	18	19	20	21	22	23
24	25	26	27	28	29	30

2: Finados 15: Proclamação da República 19: Dia da Bandeira 20: Dia da Consciência Negra
04 - Quarto Crescente 12 - Lua Cheia 19 - Quarto Minguante 26 - Lua Nova

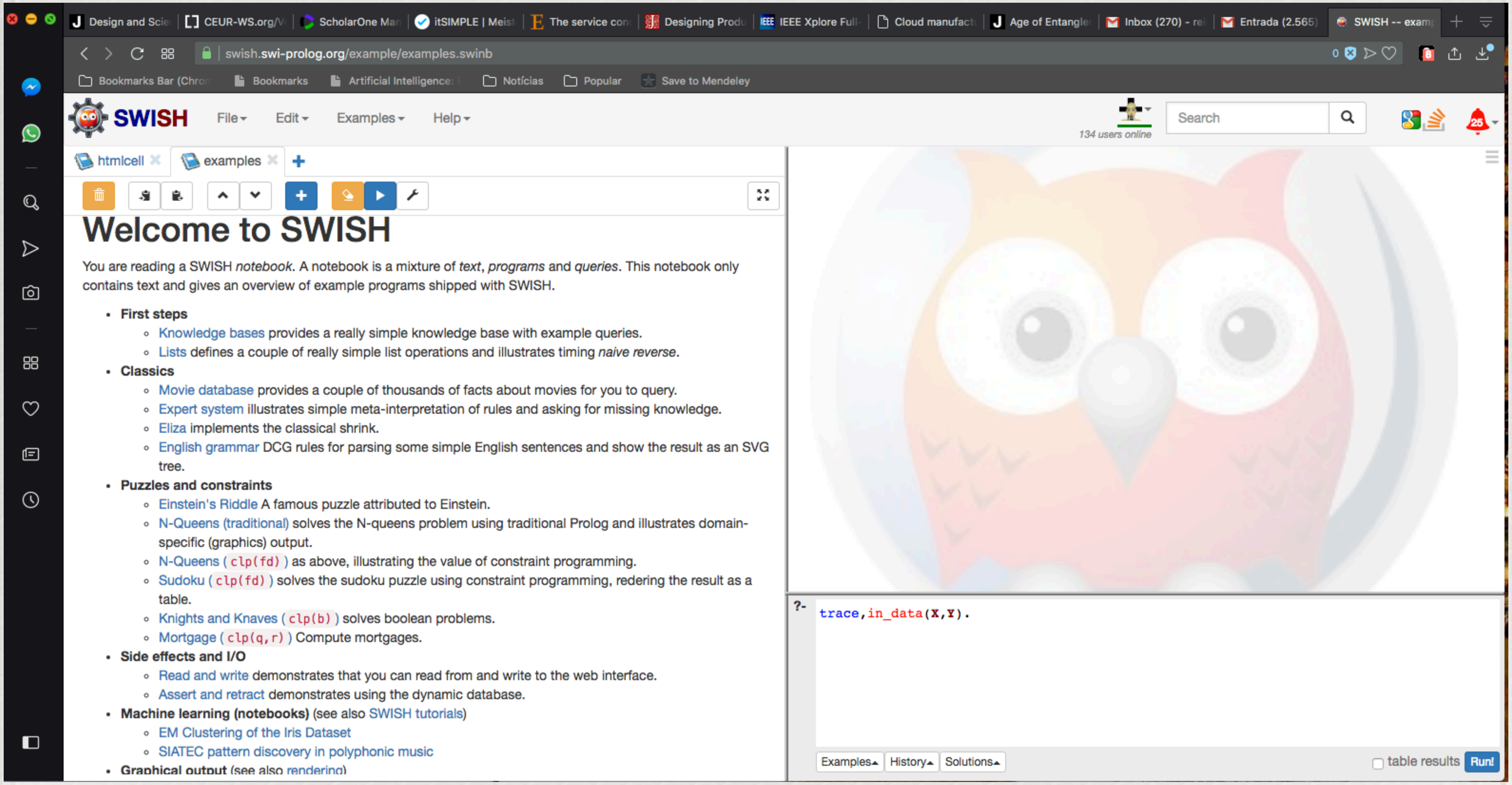
06/11 Competição

06/11 - texto (pdf) apresentando a aplicação

13/11 - resumo da análise completa

27/11 - apresentação do trabalho final

No novo cronograma teremos no dia 6/11 a competição e também a entrega da proposta para o trabalho final. A idéia é ter um feedback sobre a proposta e **entregar o trabalho final no dia 27/11.**



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).
```

O procedimento

- Todos devem estar preparados com o Swish e o seu algoritmo, mas sem tocar no teclado ou no mouse;
- Os estados inicial e final serão colocados em um slide;
- Todos esperam o "sinal de largada" para começar o processo;
- Devem inserir os estado inicial e final no programa;
- Chegando ao estado final o programa deve descrever este estado e colocar na tela o custo da busca.
- com esta informação na saída do programa a equipe faz um sinal de termino e o "tempo" é anotado (**posição no ranking**).

Classificação

1o. colocado : 10
2o. colocado : 9
3o. colocado : 8
4o. colocado : 7
5o. colocado : 6
Todos que finalizarem o plano em qualquer tempo (até 10min depois do 5o. Colocado) : 5
Os que não finalizarem nesse tempo
ou entrarem em loop : 0

Os empates são definidos pela posição inferior.

Após a competição

Após a competição cada equipe deve **fazer o upload no sistema e-disciplinas** (na página de PMR 3510) de um **arquivo PDF que descreve a heurística (f(x)= 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 obtida na 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 o plano mesmo que em um tempo mais longo, e zero se não fechou o processo de busca (sistema em loop infinito).

As equipes

Grupo 1:
Fernando Vicente Grando Monteiro 8992919
Marcos Menon José 8989112
Sverker Fabian Hugert 11462480
Vitor Augusto Martin 8993100

Grupo 2:
David Calil Spindola Pedro - 8989384
Diego Augusto Vieira Rodrigues - 8989276
Felipe Cominato Nemr - 9345662
Guilherme Sugahara Faustino - 9348971

Grupo 3:
Lucas Hideki Sakurai 8989193
Lucas Pereira Cotrim 8989092
Matheus Torres Guinezi 9345679
Monize Bessa Arabadgi 7961944
Renan Masashi Yamaguchi 8989151

Grupo 4:
Alexandre Zamora Zerbini Denigres - 8583072
Dylan Kim Heleno - 8586072
Henrique Yda Yamamoto - 9349502
Vinicius Augusto Carnevali Miquelin - 8988410
Vinicius Takiuti Miura - 9345874

Grupo 5:
Guilherme Dello Russo - 9345895
Nathan Géraud PERRIN- 10935360
Natália Thoma Ricardo - 9344806

Grupo 6:
Beatriz Santin de Araujo Pinho - 8533851
Bianca Faria Silva - 8991599
Bruna Sayuri de Souza Suzuki - 7987501
Murillo José Almeida Faria de Oliveira - 9436785

Grupo 7:
Daniel Tsutsumi - 9349005
Gabriel Pinto - 8988017
Juliana Lopes - 8512600
Kaio Takase - 9345690
Matheus Ramalho - 9345710

Grupo 8(?):
Danilo Polidoro - 8582982

Até a próxima aula!