PICAT: Uma Linguagem de Programação Multiparadigma

Miguel Alfredo Nunes, Jeferson L. R. Souza, Claudio Cesar de Sá

miguel.nunes@edu.udesc.br

jeferson.souza@udesc.br

claudio.sa@udesc.br

Departamento de Ciência da Computação
Centro de Ciências e Tecnológias
Universidade do Estado de Santa Catarina

12 de abril de 2019

Contribuições

- Alexandre Gonçalves;
- João Herique Faes Battisti;
- Paulo Victor de Aguiar;
- Rogério Eduardo da Silva;
- Hakan Kjellerstrand – (`http://www.hakank.org/picat/`)
- Neng-Fa Zhou – (`http://www.picat-lang.org/`)
- Outros anônimos que auxiliaram na produção deste documento;

Sumário I

Planejamento

- Requisito: conceitos de listas e recursividade dominados!

Planejamento

- Requisito: conceitos de listas e recursividade dominados!
- Além destes: conceitos grafos, árvores de busca, nós, etc

Planejamento

- Requisito: conceitos de listas e recursividade dominados!
- Além destes: conceitos grafos, árvores de busca, nós, etc
- *Planejamento* é um termo amplo e em vários domínios

Planejamento

- Requisito: conceitos de listas e recursividade dominados!
- Além destes: conceitos grafos, árvores de busca, nós, etc
- *Planejamento* é um termo amplo e em vários domínios
- O que **não** é o nosso contexto de *planejamento* ?
  Exemplo: planejamento estratégico das empresas, planejar como distribuir os dividendos da empresa, orçamento familiar, etc

Planejamento

- Requisito: conceitos de listas e recursividade dominados!
- Além destes: conceitos grafos, árvores de busca, nós, etc
- *Planejamento* é um termo amplo e em vários domínios
- O que **não** é o nosso contexto de *planejamento* ?
  Exemplo: planejamento estratégico das empresas, planejar como distribuir os dividendos da empresa, orçamento familiar, etc
- O que é o nosso contexto de *planejamento* ?

Planejamento

- Requisito: conceitos de listas e recursividade dominados!
- Além destes: conceitos grafos, árvores de busca, nós, etc
- *Planejamento* é um termo amplo e em vários domínios
- O que **não** é o nosso contexto de *planejamento* ?
  Exemplo: planejamento estratégico das empresas, planejar como distribuir os dividendos da empresa, orçamento familiar, etc
- O que é o nosso contexto de *planejamento* ? Questões que envolvam um ambiente, um agente, sensores, e ações que modifiquem estados.
  Exemplo clássico: robótica em geral

Planejamento

- Problemas em geral necessitam de um **plano** para serem solucionados
- Em resumo, a área de planejamento é bem complexa, antiga na área da IA e robótica (1970 – STRIPS), efervescente, e de muito interesse na indústria.

Planejamento

- Problemas em geral necessitam de um **plano** para serem solucionados
- Em resumo, a área de planejamento é bem complexa, antiga na área da IA e robótica (1970 – STRIPS), efervescente, e de muito interesse na indústria.
- Vários problemas ainda sem solução

Planejamento

- Problemas em geral necessitam de um **plano** para serem solucionados
- Em resumo, a área de planejamento é bem complexa, antiga na área da IA e robótica (1970 – STRIPS), efervescente, e de muito interesse na indústria.
- Vários problemas ainda sem solução
- Várias abordagens sobre a visão clássica da IA. Mas temos evoluções significativas ...

Planejamento

- Problemas em geral necessitam de um **plano** para serem solucionados
- Em resumo, a área de planejamento é bem complexa, antiga na área da IA e robótica (1970 – STRIPS), efervescente, e de muito interesse na indústria.
- Vários problemas ainda sem solução
- Várias abordagens sobre a visão clássica da IA. Mas temos evoluções significativas ...
- PDDL (*Planning Domain Definition Language*) : unanimidade (ou próxima a esta) entre os pesquisadores de planejamento

Definições

- Plano: seqüência ordenada de ações

Definições

- Plano: seqüência ordenada de ações
  - problema: obter banana, leite e uma furadeira
  - plano: ir ao supermercado, ir à seção de frutas, pegar as bananas, ir à seção de leite, pegar uma caixa de leite, ir ao caixa, pagar tudo, ir a uma loja de ferramentas, ..., voltar para casa.
- Um Planejador:

Definições

- Plano: seqüência ordenada de ações
  - problema: obter banana, leite e uma furadeira
  - plano: ir ao supermercado, ir à seção de frutas, pegar as bananas, ir à seção de leite, pegar uma caixa de leite, ir ao caixa, pagar tudo, ir a uma loja de ferramentas, ..., voltar para casa.
- Um Planejador:
  Combina conhecimento de um ambiente, um agente e suas ações possíveis, entradas (luz, cor, cheiro, sensor, etc), um estado corrente e/ou inicial, e com isto resolve de problemas planejar sequência de ações, que mudam de estados a cada ação, até atingir um estado final.

Exemplos do que é planejamento ...

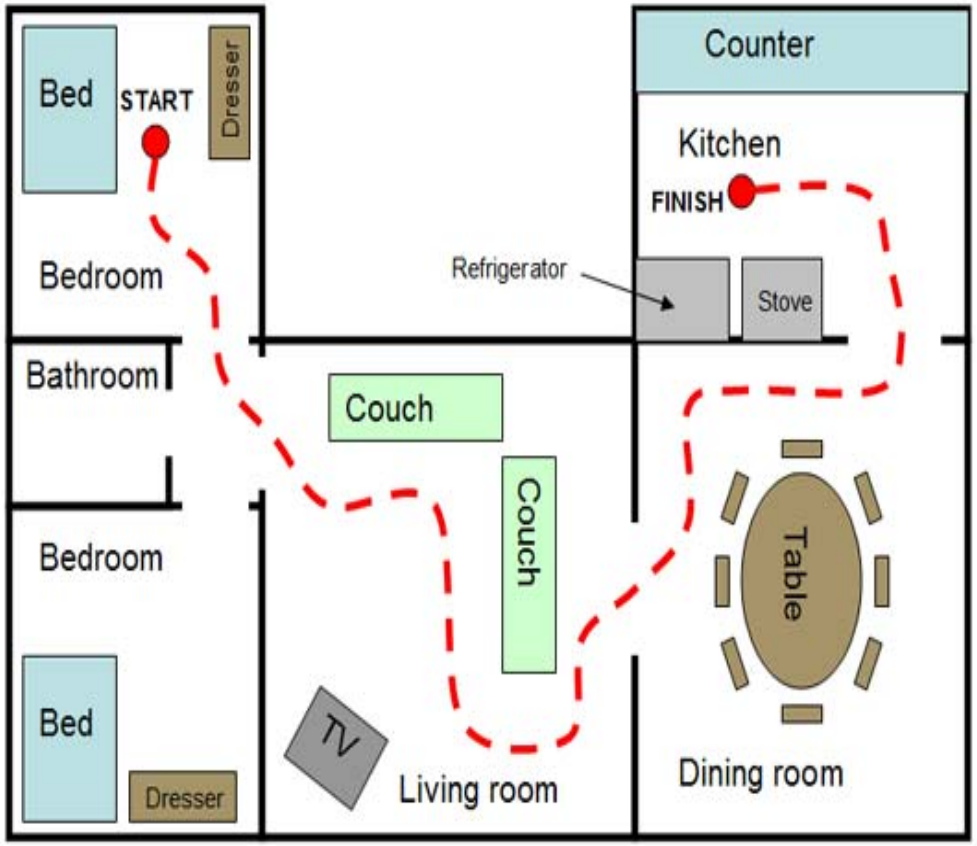

Figura 1: A fome no meio da noite!

Exemplos do que é planejamento ...

-

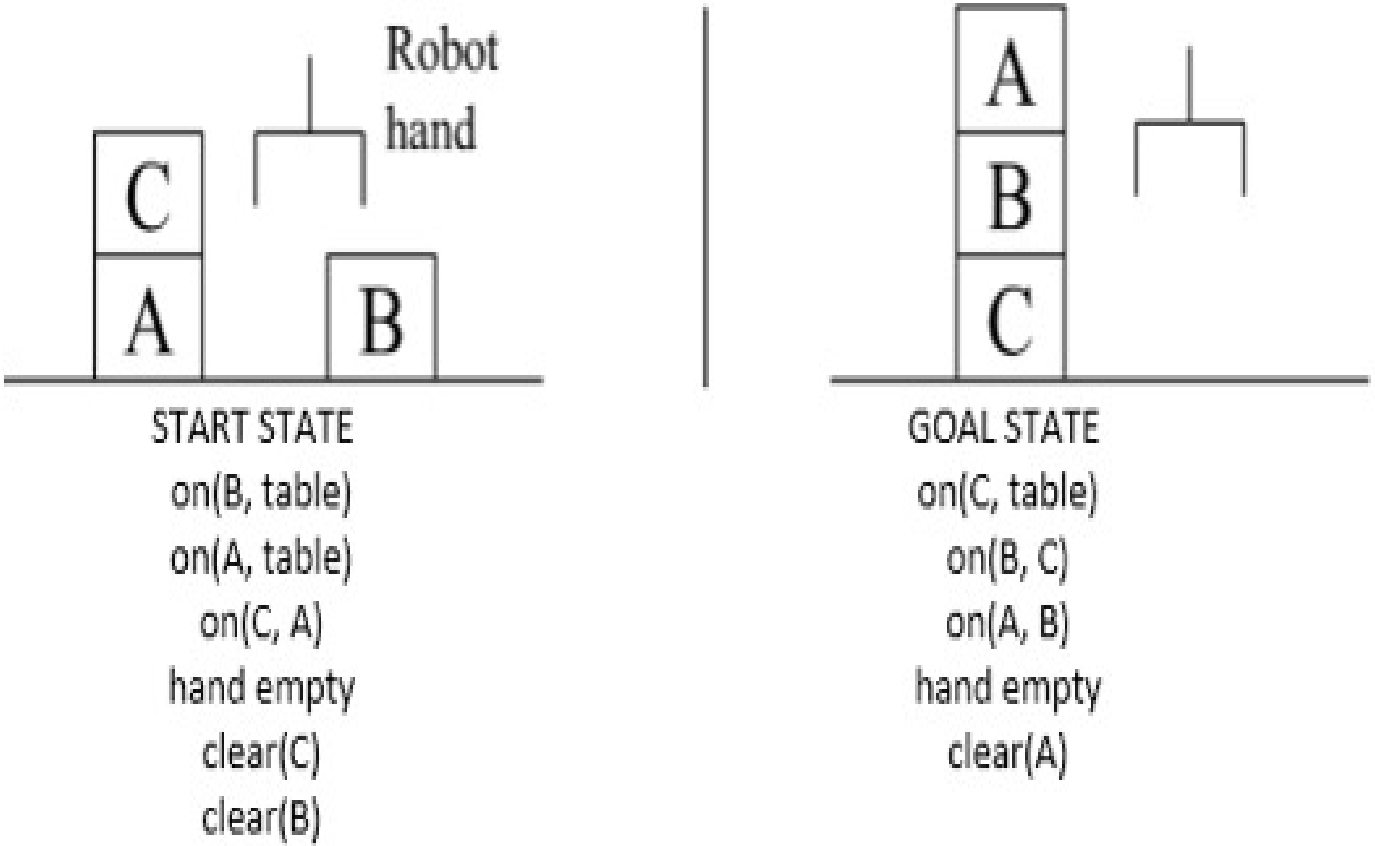

Figura 2: O *mundo dos blocos*

Espaço de Estados

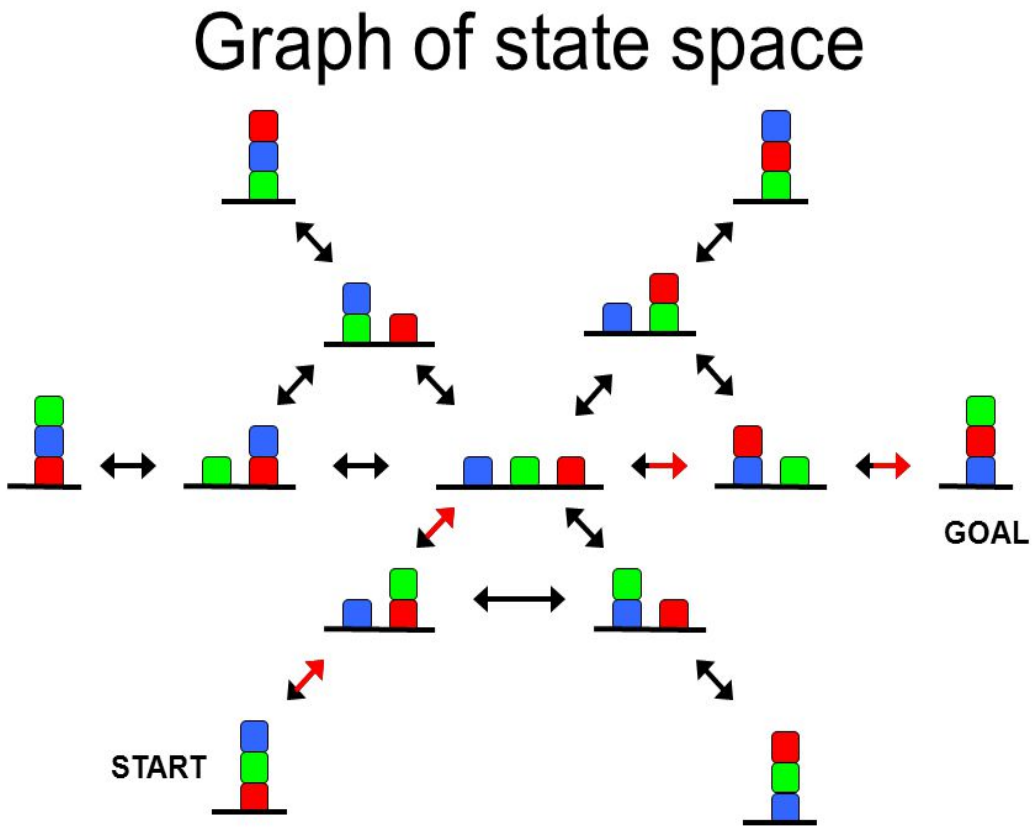

Figura 3: O espaço de estados do *mundo dos blocos*

Elementos de um Planejador – Vocabulário I

PAREI AQUI

- Plano: uma sequência ordenada de ações, criada incrementalmente a partir do estado inicial
  Ex. posições das peças de um jogo

$$S_1 < S_2 < ... < S_n$$

- Ambiente: agente, Wumpus, cavernas, buracos, ouro
- Estados: descrição completa de possíveis estados atingíveis
  Problema: quanto aos estados não-previstos, inacessíveis?
- Estado inicial: agente na caverna (1,1) com apenas uma flecha
  Wumpus e buracos em cavernas quaisquer
- Objetivos: pegar a barra de ouro e voltar à caverna (1,1) com vida

Elementos de um Planejador – Vocabulário II

- Percepções: fedor, brisa, luz, choque (contra a parede da caverna) e grito do Wumpus
- Ações: programas que geram o estado sucessor avançar para próxima caverna girar 90 graus à direita ou à esquerda pegar um objeto na mesma caverna que o agente atirar na direção para onde o agente está olhando (a flecha pára quando encontra uma parede ou mata o Wumpus) sair da caverna
- Operadores: tudo o que o agente pode fazer
- Heurística: número de objetos ainda não possuídos

O Problema Exemplo I

Figura 4: Um quebra-cabeça (2×3 ou 3×2) *simplificado* do conhecido 3×3

O Problema Exemplo II

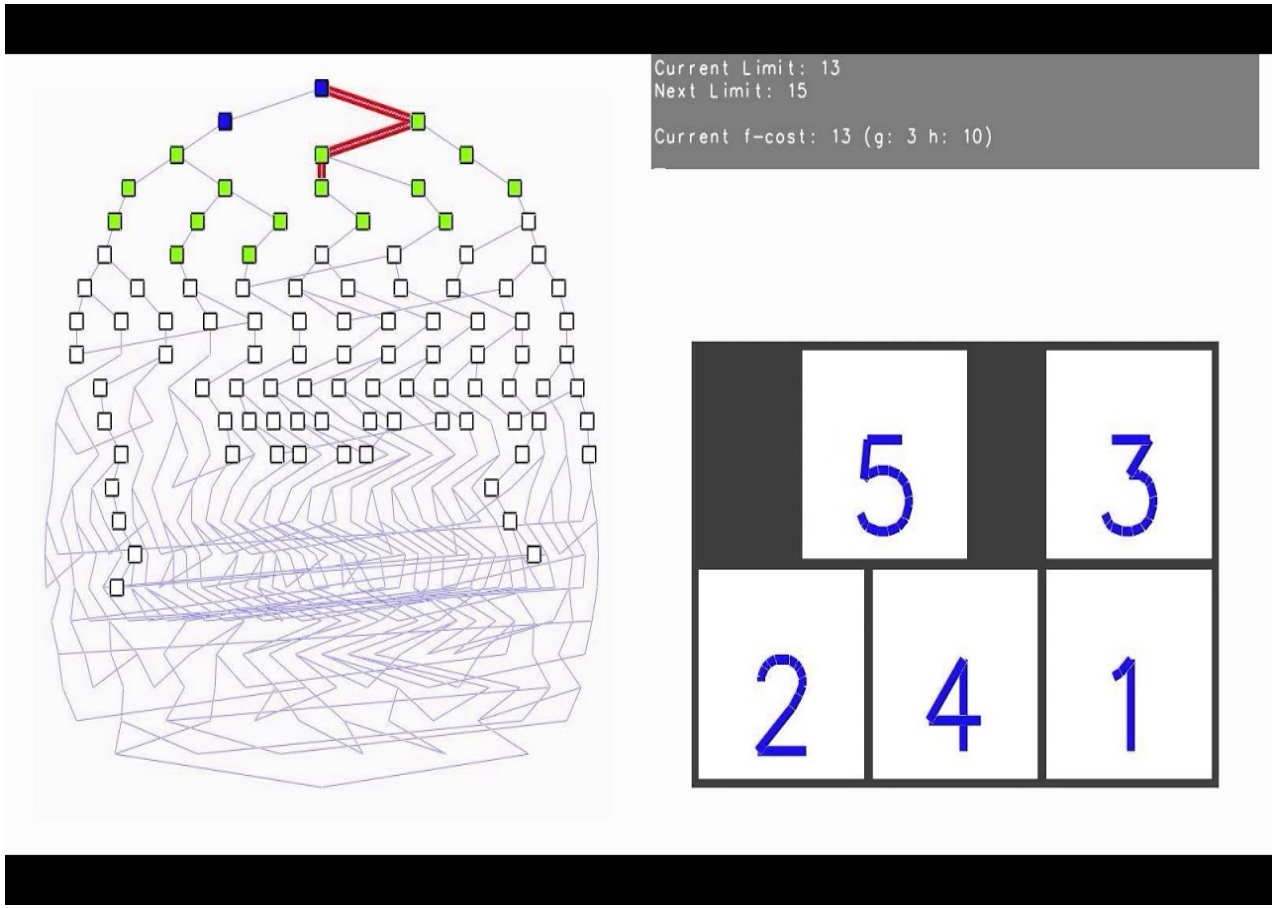

Figura 5: Sim, *simplificado* mas não muito!

Partes do código comentado I

```
/*
A  B  C
D  E  F
*/

%%%%%%%%%
%   1 5 %
% 4 3 2 %
%%%%%%%%%

import datetime.
import planner.
```

Partes do código comentado II

```
index(-)
estado_inicial( [0,1,5,4,3,2]  ).

%% funcao final do planner
final( [1,2,3,4,5,0] ) => true .
```

Partes do código comentado III

```
% Up <-> Down
/* Descrevendo as possiveis acoes para o planner */
action([A,B,C, D,E,F], S1, Acao, Custo_Acao ) ?=>
    Custo_Acao = 1,
    ( A == 0 ),  %% conj. condicoes
    S1 = [D,B,C, 0,E,F],
    Acao = ($up(D),S1). %%a acao + estado modificado

action([A,B,C, D,E,F], S1, Acao, Custo_Acao ) ?=>
    Custo_Acao = 1,
    (A == 0 ),  %% conj. condicoes
    S1 = [0,B,C, A,E,F],
    Acao = ($dow(A),S1). %%a acao + estado modificado
.....................................................
```

Partes do código comentado IV

```
% Left <-> Right
action([A,B,C, D,E,F], S1, Acao, Custo_Acao ) ?=>
    Custo_Acao = 1,
    (A == 0),  %% conj. condicoes
    S1 = [B,0,C, D,E,F],
    Acao = ($left(B), S1). %%a acao + estado modificado

action([A,B,C, D,E,F], S1, Acao, Custo_Acao ) ?=>
    Custo_Acao = 1,
    (B == 0),  %% conj. condicoes
    S1 = [0,A,C, D,E,F],
    Acao = ($right(A), S1). %%a acao + estado modificado
.....................................................
```

Partes do código comentado V

```
main  ?=>
    estado_inicial( Q ),

    best_plan_unbounded( Q , Sol_Acoes),
     println(sol=Sol_Acoes),

    printf("\n Estado Inicial: "),
    w_Quadro( Q ),
    w_L_Estado( Sol_Acoes ),

    Total := length(Sol_Acoes) ,
    Num_Movts := (Total -1) ,
    printf("\n Inicial  (estado): %w ", Q),
    printf("\n Total de acoes: %d", Total),
    printf(" \n ========================================\n ")
    %%% , fail descomente para multiplas solucoes
```

Partes do código comentado VI

```
    .

main => printf("\n Para uma solução .... !!!!" ) .
.....................................................
```

O código

- Acompanhar as explicações do código de: `https://github.com/claudiosa/CCS/blob/master/picat/puzzle_2x3_planner.pi`
- Confira a execução

Parte da Saída I

```
[ccs@gerzat picat]$ picat puzzle_2x3_planner.pi
sol = [(left(1),[1,0,5,4,3,2]),(left(5),[1,5,0,4,3,2]),
(up(2),[1,5,2,4,3,0]),(right(3),[1,5,2,4,0,3]),(dow(5),[1,0,2,4,
(left(2),[1,2,0,4,5,3]),(up(3),[1,2,3,4,5,0])]

 Estado Inicial:
 0 1 5
 4 3 2

Acao: left(1)
 1 0 5
 4 3 2

Acao: left(5)
 1 5 0
 4 3 2
```

Parte da Saída II

. . . . . .

Parte da Saída III

```
.................................
Acao: left(2)
 1 2 0
 4 5 3

Acao: up(3)
 1 2 3
 4 5 0

 Inicial  (estado): [0,1,5,4,3,2]
 Total de acoes: 7
 ========================================
```

O módulo do *planner* do Picat

- O que efetivamente voce precisa saber

O módulo do *planner* do Picat

- O que efetivamente voce precisa saber
- Importar um módulo

O módulo do *planner* do Picat

- O que efetivamente voce precisa saber
- Importar um módulo
- O predicado *final*

O módulo do *planner* do Picat

- O que efetivamente voce precisa saber
- Importar um módulo
- O predicado *final*
- O predicado *action*

O módulo do *planner* do Picat

- O que efetivamente voce precisa saber
- Importar um módulo
- O predicado *final*
- O predicado *action*
- Os planejadores disponíveis:

Uma Solução – Saída

Reflexões

- Outros métodos para se resolver estes problemas

Reflexões

- Outros métodos para se resolver estes problemas
- Mas perdemos na portabilidade de usar em outros planejadores

Reflexões

- Outros métodos para se resolver estes problemas
- Mas perdemos na portabilidade de usar em outros planejadores
- Os modelos escritos em PDDL (*Planning Domain Definition Language*) facilmente portáveis para Picat

Reflexões

- Outros métodos para se resolver estes problemas
- Mas perdemos na portabilidade de usar em outros planejadores
- Os modelos escritos em PDDL (*Planning Domain Definition Language*) facilmente portáveis para Picat
- Sob um uso mais restrito, um modelo em PDDL é executado diretamente em Picat

Reflexões

- Outros métodos para se resolver estes problemas
- Mas perdemos na portabilidade de usar em outros planejadores
- Os modelos escritos em PDDL (*Planning Domain Definition Language*) facilmente portáveis para Picat
- Sob um uso mais restrito, um modelo em PDDL é executado diretamente em Picat
-

Resumindo

- Picat é jovem (nascida em 2013);
- Uma evolução ao Prolog após seus mais de 40 anos de existência e sucesso!
- Sua sintaxe é moderna;
- Código aberto, multi-plataforma, e repleta de possibilidades;
- Uso para fins diversos;
- Muitas bibliotecas específicas prontas: CP, SAT, Planner, etc;
-
- .