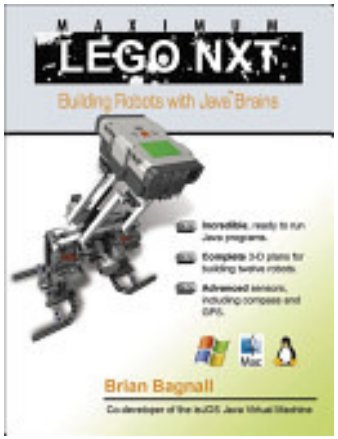
Departamento de Sistemas e Computação – FURB
Curso de Ciência da Computação
Disciplina de Robótica

Paradigmas em robótica

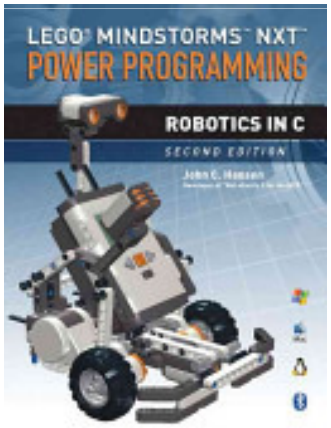
Prof. Aurélio Hoppe
aurelio.hoppe@gmail.com
http://www.inf.furb.br/~aurelio/

Grupo de Pesquisa em Computação
Gráfica, Processamento de Imagens e
Entretenimento Digital
http://www.inf.furb.br/gcg

Bibliografia



Maximum lego NXT: building robots with Java brains
Brian Bagnall
Winnipeg : Variant Press, 2007, 505p.



Lego Mindstorms NXT power programming (2 edição)
John Hansen
Winnipeg : Variant, c2009, 543p.

Paradigmas em Robótica

- Definem como as primitivas principais “**sentir**”, “**planejar**”, “**agir**” são organizadas em um programa de controle de um robô

  - SENTIR
    - obter dados sensoriais (brutos) do mundo e gerar informações úteis ao robô (percepção)

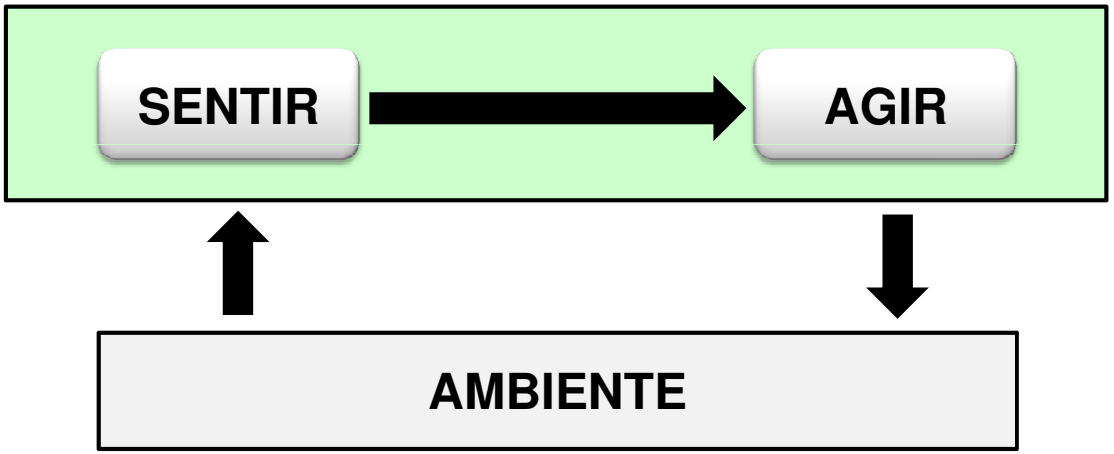
  - PLANEJAR
    - produzir diretivas (tarefas), considerando o objetivo do robô, e suas informações sensoriais e/ou cognitivas (conhecimento)

  - AGIR
    - produzir efeito no mundo através de atuadores (ex: motores)

Paradigmas em robótica

Primitiva	Entrada	Saída
SENTIR	dados sensoriais	informações sensoriais (percepção)
PLANEJAR	informações (sensoriais ou cognitivas)	diretivas (tarefas)
AGIR	informações (sensoriais ou diretivas)	comandos de atuação

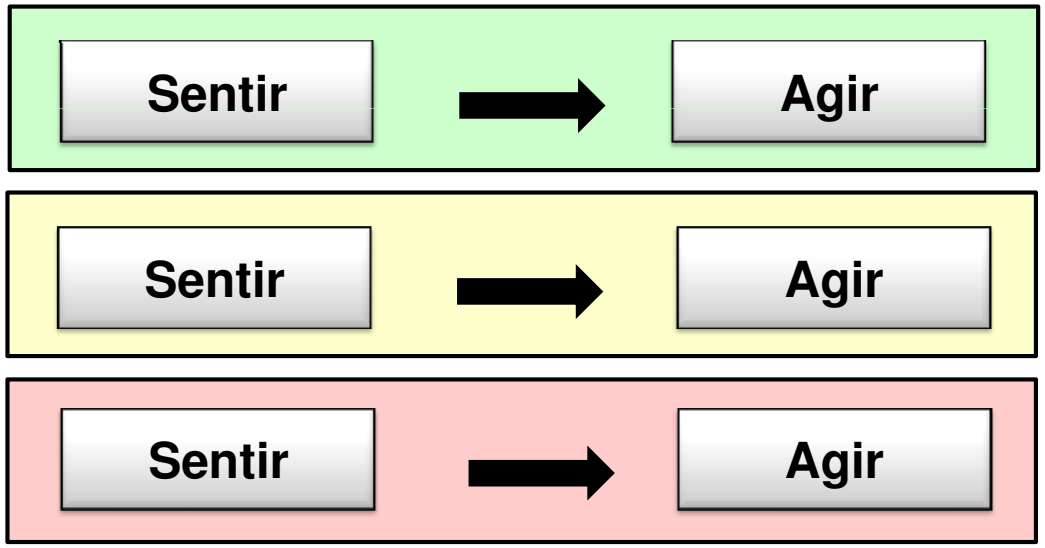
Paradigma Reativo



- Inspirado na idéia dos **comportamentos** “padrão”
- Não há **modelo de mundo**
- **Percepção** é transformada imediatamente em **ação**
- Vários comportamentos podem ser ativados simultaneamente
  - (se o hardware permitir)

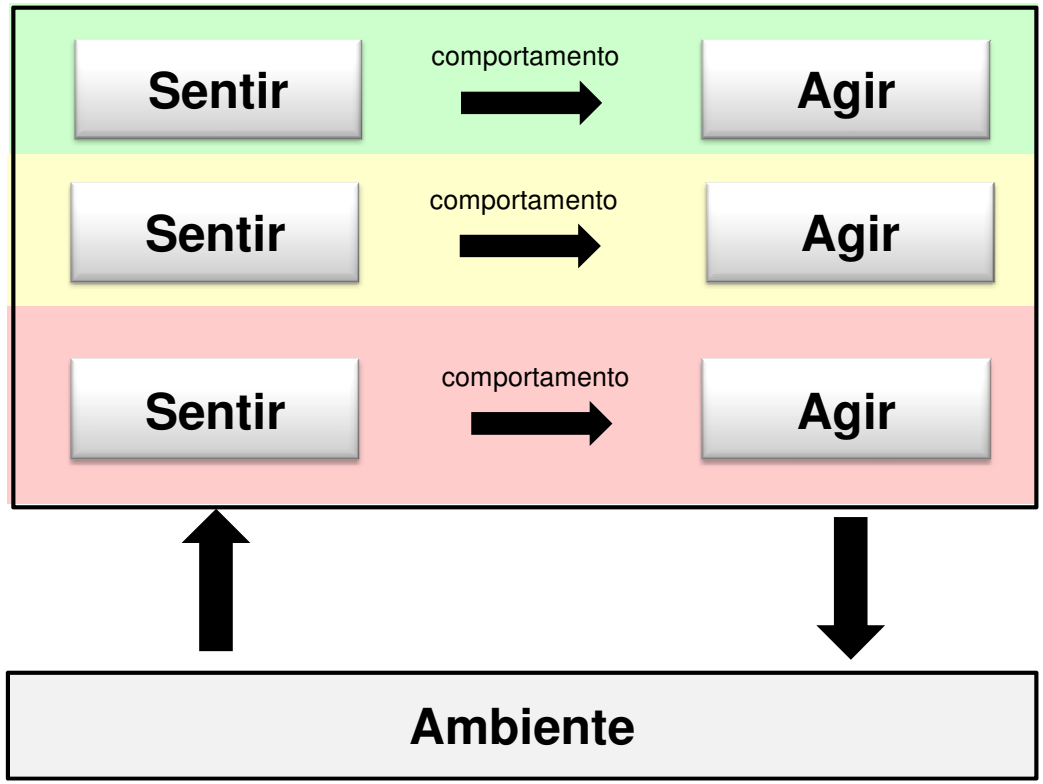
Paradigma Reativo

- Ações do robô são decompostas em comportamentos
  - Não há um modelo do mundo nem planejamento
    - Sistemas puramente reativos
- Comportamentos executados em paralelo
- Mapeamento direto entre sentir e agir



Paradigma Reativo:
Comportamentos

- Toda ação do robô é resultado de um comportamento.
- Várias ações == vários comportamentos



Paradigma Reativo:
Comportamentos

- Cada comportamento define sua percepção
  - só percebe o que é de interesse
  - percepção é **ego-cêntrica**
    - local e em nível do robô
    - não há percepção em nível de mundo.
- Comportamentos são os “blocos de construção” da inteligência do robô.
  - são independentes
- Comportamento final do robô é emergente.
  - Emerge da interação de vários comportamentos
    - A inteligência do robô está nos olhos do observador, e não em um trecho específico de código.

Paradigma Reativo:
Comportamentos

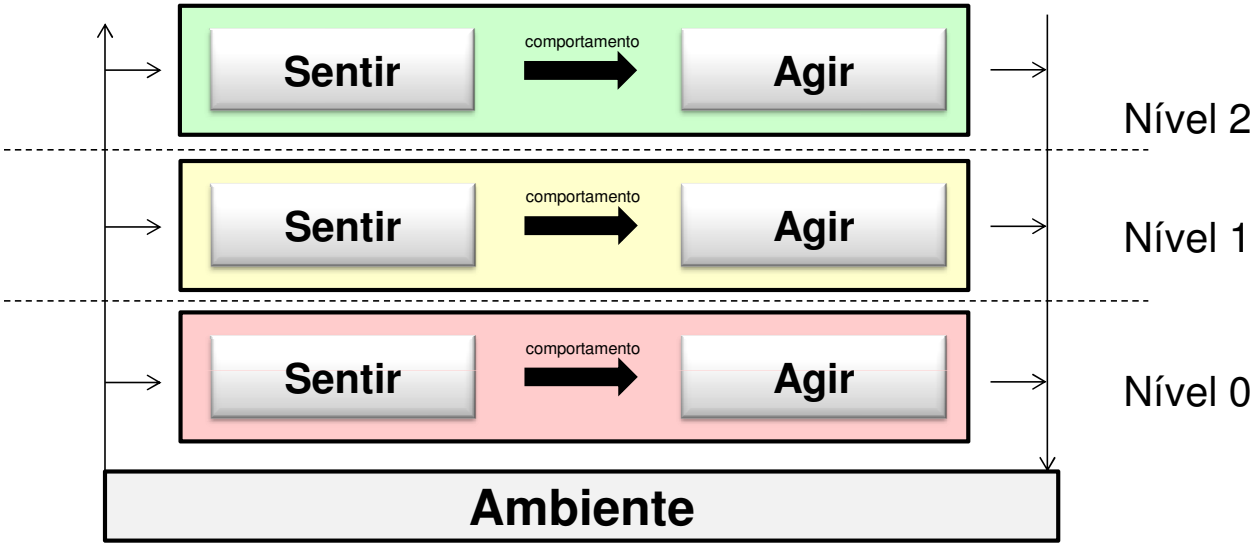
- Como coordenar vários comportamentos?
  - Arquiteturas

- Sistema Video Locadora
  - Paradigma:
    - OO
  - Arquiteturas:
    - Cliente – servidor

- Arquitetura define a interação entre os componentes
- Arquiteturas para o paradigma reativo:
  - Subsumption
  - Campos potenciais

Paradigma Reativo:
Arquitetura Subsumption

- Comportamentos são organizados em níveis de competência
  - Todos os módulos de processamento tem acesso aos sensores e atuadores



  - Níveis baixos → comportamentos vitais
    - ex: andar, desviar de obstáculos
  - Níveis altos → comportamentos para buscar objetivos
    - ex: exploração

Arquitetura Subsumption:
Lejos

- Behavior – interface que representa um comportamento
- É necessário definir:
  - circunstância em que o comportamento é ativado
    - boolean takeControl()
    - sensor de toque ativado = colisão
  - ação realizada pelo comportamento
    - void action()
    - colisão = parar de andar
  - desativação do comportamento
    - void suppress()

Arquitetura Subsumption:
Lejos

- Exemplo: robô que anda em frente e desvia de obstáculos

```
public class NaoBater implements Behavior {
    TouchSensor touch;

    public NaoBater(TouchSensor touch) {
        this.touch = touch;
    }

    public void action() {
        Motor.A.rotate(-360);
    }

    public void suppress() {}

    public boolean takeControl() {
        return touch.isPressed();
    }
}
```

Arquitetura Subsumption:
Lejos

- Exemplo: robô que anda em frente e desvia de obstáculos

```
public class AndarFrente implements Behavior {

    public AndarFrente() { }

    public void action() {
        Motor.A.forward();
    }

    public void suppress() {
        Motor.A.flt();
    }

    public boolean takeControl() {
        // nivel zero - sempre é ativado se
        // nenhum outro comportamento foi ativado
        return true;
    }
}
```

Arquitetura Subsumption:
Lejos

- Arbitrator – interface que gerencia os comportamentos.
  - Requer um array de Behavior
    - quanto maior o índice do comportamento, maior é a prioridade

- A cada ciclo, determina qual comportamento será ativado
  - itera descendentemente no array de comportamentos
  - será ativado o primeiro com takeControl() == true

Arquitetura Subsumption:
Lejos

- Exemplo: robô que anda em frente e desvia de obstáculos

```
public class SubsumptionRobot {

    public static void main(String[] args) {
        TouchSensor touch = new TouchSensor(SensorPort.S1);

        Behavior andar = new AndarFrente();
        Behavior desviar = new NaoBater(touch);

        Behavior[] comportamentos = { andar, desviar };
        // quanto maior o índice, maior a
        // prioridade do comportamento

        Arbitrator arb = new Arbitrator(comportamentos);
        arb.start();
    }
}
```

Arquitetura Subsumption:

Exercício