



# PROGRAMAÇÃO AVANÇADA INTELIGÊNCIA ARTIFICIAL

PROF. JOSENALDE OLIVEIRA

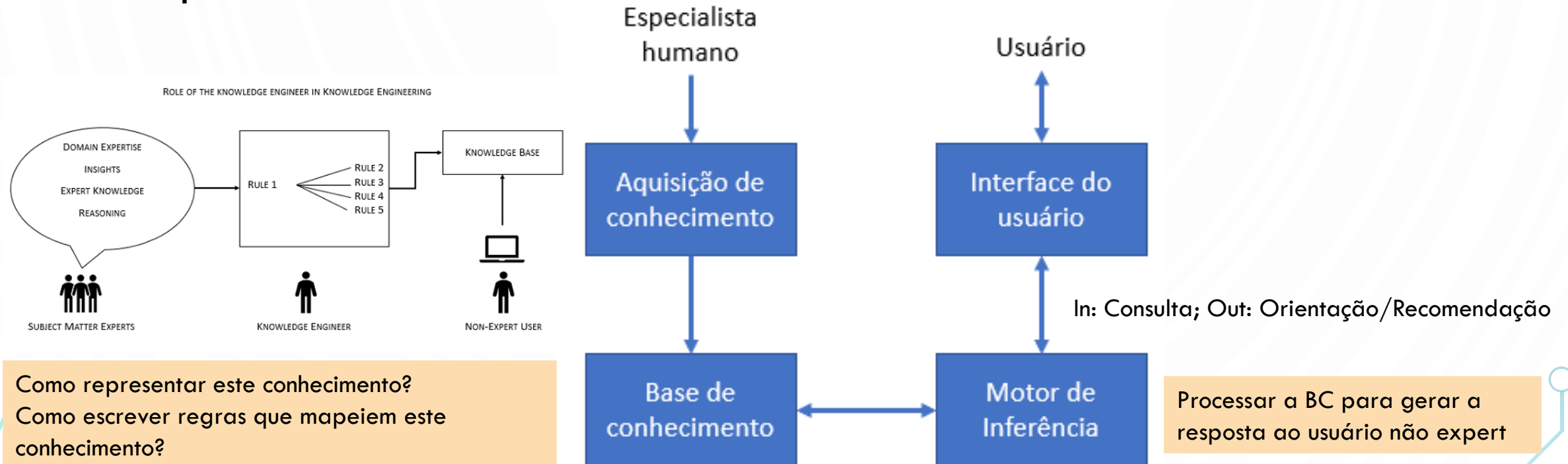
josenalde.oliveira@ufrn.br

<https://github.com/josenalde/ai-advanced-programming>

TÉCNICO EM INFORMÁTICA – UFRN - EAJ

# SISTEMAS ESPECIALISTAS

- Sistemas inteligentes que incorporam **conhecimento especializado em domínio específico**, sendo este volumoso, impreciso, dinâmico – é possível **‘simular’** a habilidade de tomada de decisão humana?

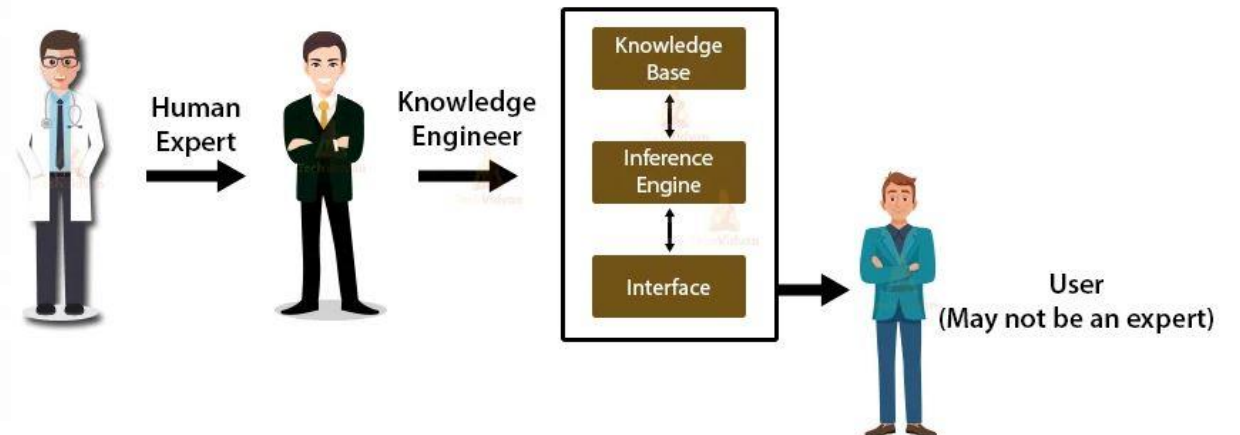


# SISTEMAS ESPECIALISTAS

- Diversas aplicações e contextos, normalmente associados à diagnóstico
  - *Uzum Embrapa*: <https://www.cnpuv.embrapa.br/uzum/uva/>
  - *Dendral, Mycin, R1/xcon, pxdes, dxplain, etc.*

- Permanência 'pós especialista'
- Distribui expertise humana
- Pode contemplar base com vários especialistas
- Reduz custo de consultar especialista
- Podem resolver problemas complexos deduzindo novos fatos a partir de fatos existentes/conhecidos, representados em geral por regras se-então
- Requer treinamento e conjunto significativo de dados

## Components of Expert Systems in AI



# SISTEMAS ESPECIALISTAS

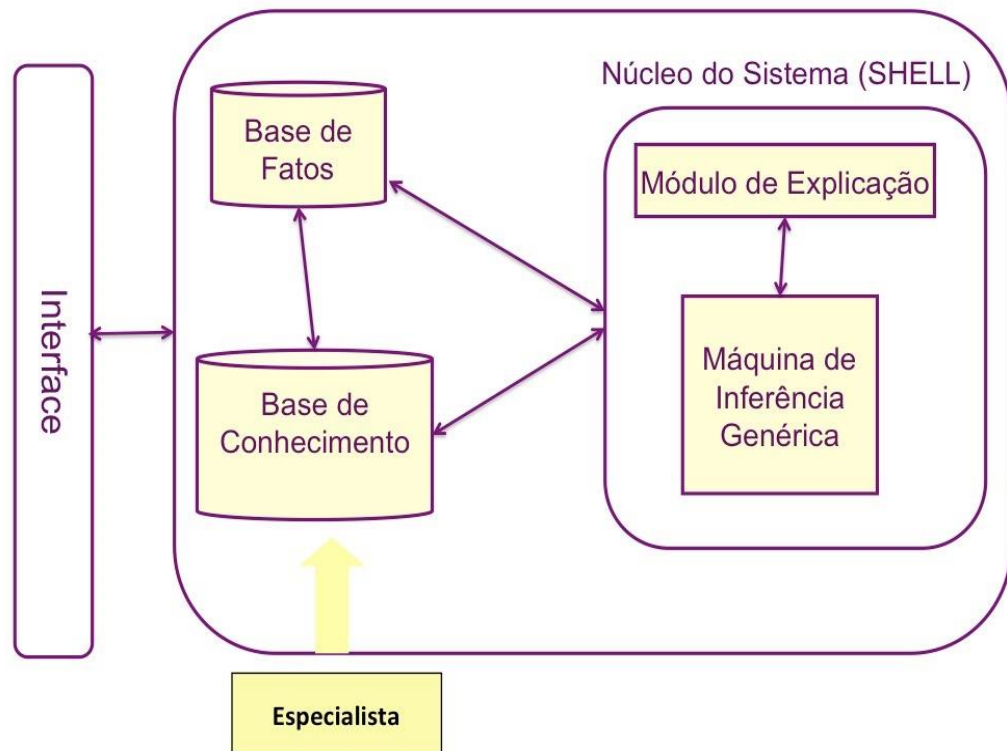
- Estrutura geral:

**Interface:** Responsável pela obtenção de informações junto ao usuário, além da apresentação dos resultados e explicações

**Base de conhecimento:** onde fica armazenado o conhecimento sobre o domínio do problema de que trata o sistema, descrito explicitamente por um **formalismo processável computacionalmente** e com representação compatível com a máquina de inferência específica (algoritmo que simula o raciocínio). É dependente do domínio de aplicação.

Pode ser composta de conceitos básicos, relações entre conceitos, regras, procedimentos, sendo representado este conhecimento por lógica, regras, redes semânticas, orientação a objetos etc.

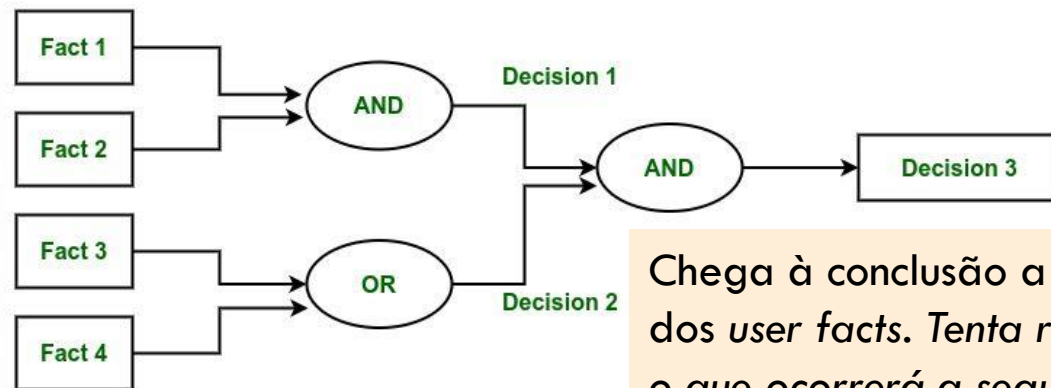
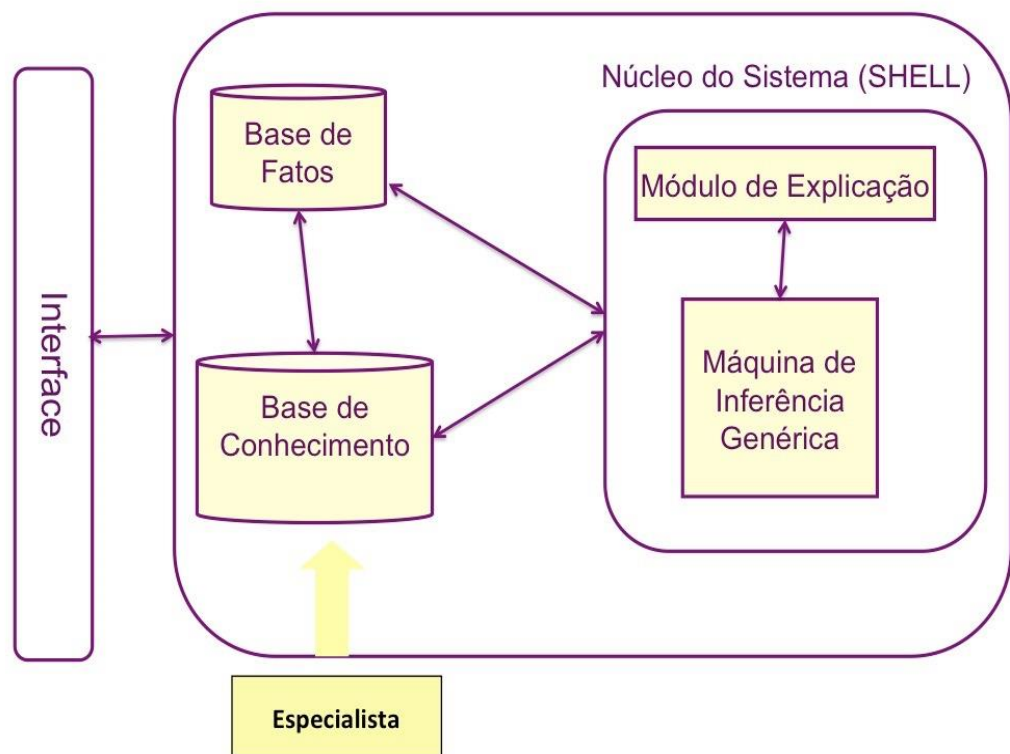
**Base de fatos (memória de trabalho):** onde são armazenadas as respostas do usuário na interação com o SE e as conclusões intermediárias do processo de raciocínio - volátil



# SISTEMAS ESPECIALISTAS

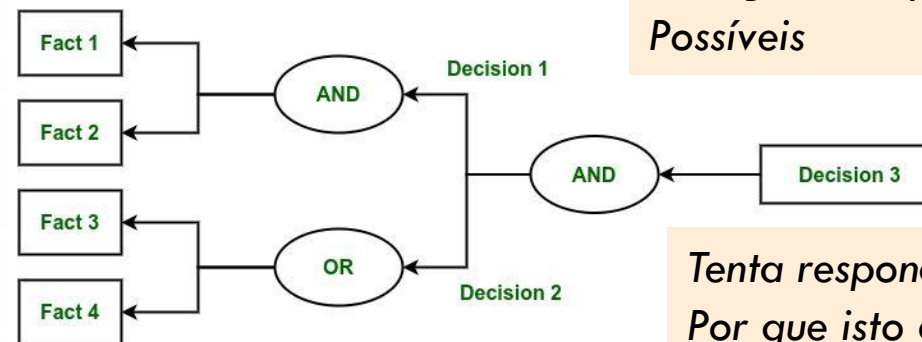
- Estrutura geral:

**Máquina de inferência (MI):** A máquina de inferência processa a linguagem de representação usada na base de conhecimento, gerando e percorrendo o espaço de busca (algoritmo) sempre que necessário. É independente do domínio de aplicação.



Forward Chaining

Chega à conclusão a partir dos *user facts*. Tenta responder o que ocorrerá a seguir? *Predição. Útil para muitas conclusões Possíveis*



Backward Chaining

Tenta responder Por que isto acontece? *Raiz ou causa da questão. Testa Hipóteses (diagnóstico), conclusão Específica, mais apropriado*



# SISTEMAS ESPECIALISTAS

- Máquina de inferência:
- A máquina/motor de inferência é a responsável por selecionar as regras para determinar se ela é aplicável ou não
- Isso é feito selecionando perguntas para o usuário
- Se for aplicável, esta regra poderá afetar a escolha de outras regras
- O mecanismo de inferência continuará a aplicar regras até não haver mais regras aplicáveis
- **Não pense nas regras como um caminho linear a ser percorrido**, já que a máquina de inferência fica em um laço e testa quaisquer regras que puder até obter uma conclusão. A **MI não faz perguntas sobre coisas que ela pode obter internamente**, como a data ou a estação do ano, por exemplo



# SISTEMAS ESPECIALISTAS

- Máquina de inferência (exemplo raciocínio guiado por dados):

Regra 1: se  
o motor recebe combustível e  
o motor tenta pegar,  
então  
o problema é vela.

Regra 2: se  
o motor não pega e  
as luzes não acendem,  
então  
o problema é bateria ou cabo.

Regra 3: se  
o motor não pega e  
as luzes acendem,  
então  
o problema é o motor de partida.

Regra 4: se  
há combustível no tanque de combustível e  
há combustível no carburador,  
então  
o motor está recebendo combustível.

- Considera base de fatos e verifica se alguma combinação deles combina com os antecedentes de uma das regras da base de regras
- Se coincidem, a regra é disparada e sua conclusão adicionada à base de fatos
- Se a conclusão por recomendação (objetivo), encerra

- PREMISSAS das regras são analisadas para verificar quais são questionáveis e quais são alcançáveis por conclusão de outras regras. As questionáveis geram perguntas ao usuário. Pode-se fazer uma busca eliminando regras com premissas não questionáveis

SE para análise de sistemas automotivos

# SISTEMAS ESPECIALISTAS

- Máquina de inferência (exemplo raciocínio guiado por dados):

Regra 1: se  
o motor recebe combustível e  
o motor tenta pegar,  
então  
o problema é vela.

Regra 2: se  
o motor não pega e  
as luzes não acendem,  
então  
o problema é bateria ou cabo.

Regra 3: se  
o motor não pega e  
as luzes acendem,  
então  
o problema é o motor de partida.

Regra 4: se  
há combustível no tanque de combustível e  
há combustível no carburador,  
então  
o motor está recebendo combustível.

- Iteração #1 pelas REGRAS
  - R1.p1 não questionável E R1.p2 questionável: R1 não dispara
  - R2.p1 e R2.p2 são questionáveis. Supor respostas negativas.  
Fato O motor tenta pegar é colocado na base de fatos. R2 não dispara
  - R3.p1 falha, passa para R4
  - R4.p1 e R4.p2 são questionáveis. Supor respostas positivas.  
Combustível no tanque de combustível e no carburador, junto com a conclusão entram na base de fatos. Próxima iteração.
- Iteração #2
  - R1.p1 e R1.p2 são positivas, logo gera conclusão objetivo (problema)

SE para análise de sistemas automotivos



# SISTEMAS ESPECIALISTAS

- Máquina de inferência (exemplo raciocínio guiado por dados):

Regra 1: se  
o motor recebe combustível e  
o motor tenta pegar,  
então  
o problema é vela.

Regra 2: se  
o motor não pega e  
as luzes não acendem,  
então  
o problema é bateria ou cabo.

Regra 3: se  
o motor não pega e  
as luzes acendem,  
então  
o problema é o motor de partida.

Regra 4: se  
há combustível no tanque de combustível e  
há combustível no carburador,  
então  
o motor está recebendo combustível.

SE para análise de sistemas automotivos

- Iteração #1 pelas REGRAS
  - R1.p1 não questionável E R1.p2 questionável: R1 não dispara
  - R2.p1 e R2.p2 são questionáveis. **Supor respostas negativas.**Fato O motor tenta pegar é colocado na base de fatos. R2 não dispara
  - R3.p1 falha, passa para R4
  - R4.p1 e R4.p2 são questionáveis. Supor respostas positivas. Combustível no tanque de combustível e no carburador, junto com a conclusão entram na base de fatos. Próxima iteração.
- Iteração #2
  - R1.p1 e R1.p2 são positivas, logo gera conclusão objetivo (problema)

**R2.p1: o motor não pega? Sim, ele tenta pegar!**

**R2.p2: as luzes não acendem? Sim, elas acendem!**

O motor tenta pegar

Base de fatos

# SISTEMAS ESPECIALISTAS

- Máquina de inferência (exemplo raciocínio guiado por dados):

Regra 1: se  
o motor recebe combustível e  
o motor tenta pegar,  
então  
o problema é vela.

Regra 2: se  
o motor não pega e  
as luzes não acendem,  
então  
o problema é bateria ou cabo.

Regra 3: se  
o motor não pega e  
as luzes acendem,  
então  
o problema é o motor de partida.

Regra 4: se  
há combustível no tanque de combustível e  
há combustível no carburador,  
então  
o motor está recebendo combustível.

SE para análise de sistemas automotivos

- Iteração #1 pelas REGRAS
  - R1.p1 não questionável E R1.p2 questionável: R1 não dispara
  - R2.p1 e R2.p2 são questionáveis. Supor respostas negativas.  
Fato O motor tenta pegar é colocado na base de fatos. R2 não dispara
  - R3.p1 falha, passa para R4
  - R4.p1 e R4.p2 são questionáveis. **Supor respostas positivas.**  
Combustível no tanque de combustível e no carburador, junto com a conclusão entram na base de fatos. Próxima iteração.
- Iteração #2
  - R1.p1 e R1.p2 são positivas, logo gera conclusão objetivo (problema)

**R4.p1: há combustível no tanque? Sim, há**

**R4.p2: há combustível no carburador? Sim, há!**

O motor tenta pegar  
Há combustível no tanque  
Há combustível no carburador  
O motor está recendo combustível

Base de fatos

# SISTEMAS ESPECIALISTAS

- Máquina de inferência (exemplo raciocínio guiado por dados):

Regra 1: se  
o motor recebe combustível e  
o motor tenta pegar,  
então  
→ o problema é vela.

Regra 2: se  
o motor não pega e  
as luzes não acendem,  
então  
o problema é bateria ou cabo.

Regra 3: se  
o motor não pega e  
as luzes acendem,  
então  
o problema é o motor de partida.

Regra 4: se  
há combustível no tanque de combustível e  
há combustível no carburador,  
então  
o motor está recebendo combustível.

SE para análise de sistemas automotivos

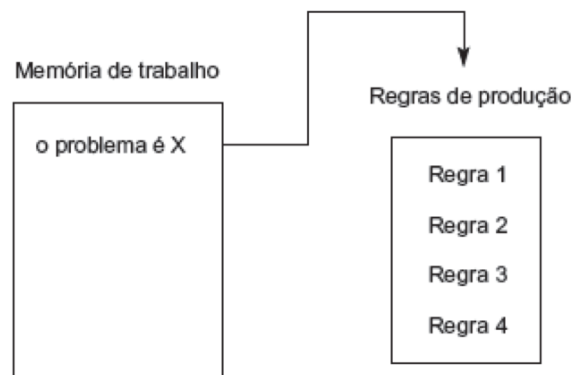
- Iteração #1 pelas REGRAS
  - R1.p1 não questionável E R1.p2 questionável: R1 não dispara
  - R2.p1 e R2.p2 são questionáveis. Supor respostas negativas.  
Fato O motor tenta pegar é colocado na base de fatos. R2 não dispara
  - R3.p1 falha, passa para R4
  - R4.p1 e R4.p2 são questionáveis. Supor respostas positivas.  
Combustível no tanque de combustível e no carburador, junto com a conclusão entram na base de fatos. Próxima iteração.
- Iteração #2
  - R1.p1 e R1.p2 são positivas, logo gera conclusão objetivo (problema VELA)

O motor tenta pegar  
Há combustível no tanque  
Há combustível no carburador  
O motor está recendo combustível

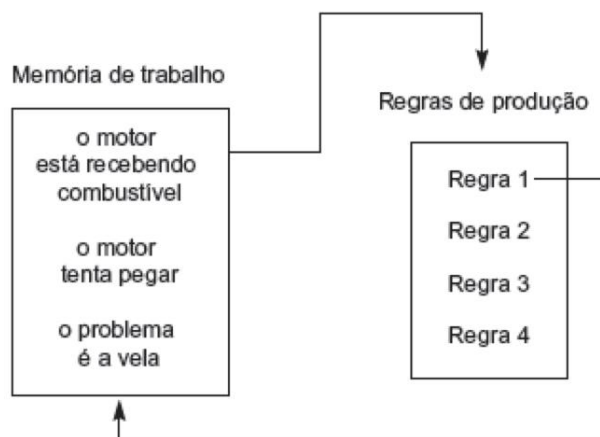
Base de fatos

# SISTEMAS ESPECIALISTAS

- Máquina de inferência (exemplo raciocínio guiado por objetivos):



- Coloca-se a conclusão do que se quer provar na memória de trabalho
  - Algo do tipo:...o problema é...
- Regras 1, 2 e 3 casam com conclusões deste tipo (começa-se pela primeira)
- Regra 1 coloca suas premissas e conclusão na memória de trabalho, para serem confirmadas ou não
  - Para confirmar R1.p1, testa-se R4 com perguntas ao usuário:



...▶  
combustível no tanque de combustível?  
**sim**  
combustível no carburador?  
**sim**  
o motor tenta pegar?  
**por quê?**

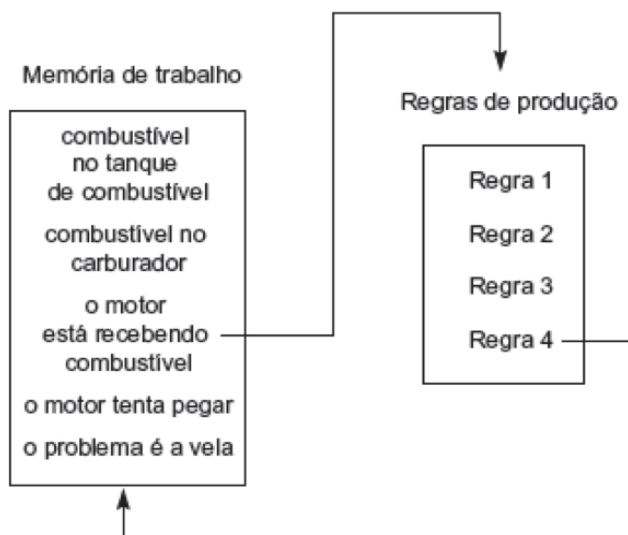
Foi estabelecido que:

1. o motor está recebendo combustível, portanto, se
2. o motor tenta pegar, então o problema é vela.

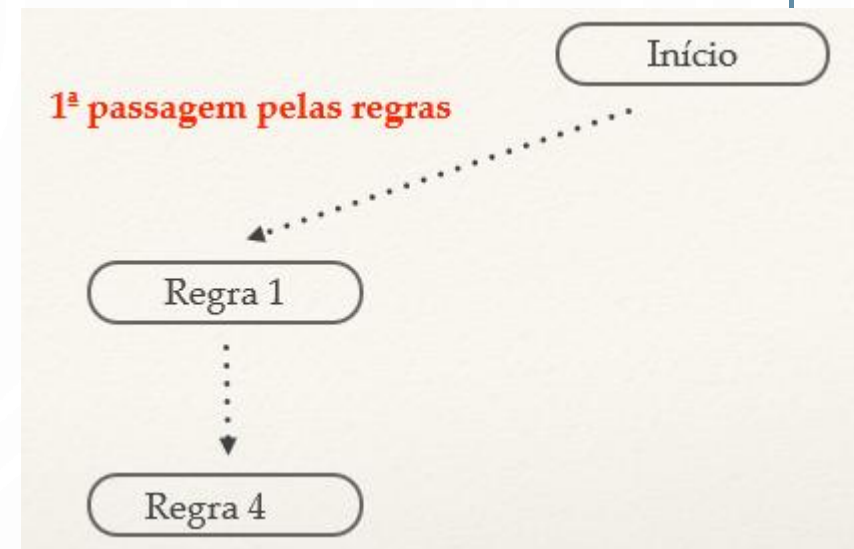
SE para análise de sistemas automotivos

# SISTEMAS ESPECIALISTAS

- Máquina de inferência (exemplo raciocínio guiado por objetivos):



- Regra 1: se  
o motor recebe combustível e  
o motor tenta pegar,  
então  
o problema é vela.
- Regra 2: se  
o motor não pega e  
as luzes não acendem,  
então  
o problema é bateria ou cabo.
- Regra 3: se  
o motor não pega e  
as luzes acendem,  
então  
o problema é o motor de partida.
- Regra 4: se  
há combustível no tanque de combustível e  
há combustível no carburador,  
então  
o motor está recebendo combustível.



## Memória de trabalho ao final da execução (Base de Fatos)

1. O problema é vela
2. O motor está recebendo combustível
3. O motor tenta pegar (3ª sim)
4. Há combustível no tanque de combustível (1ª sim)
5. Há combustível no carburador (2ª sim)

- Conclui-se que o problema é VELA

SE para análise de sistemas automotivos

# SISTEMAS ESPECIALISTAS

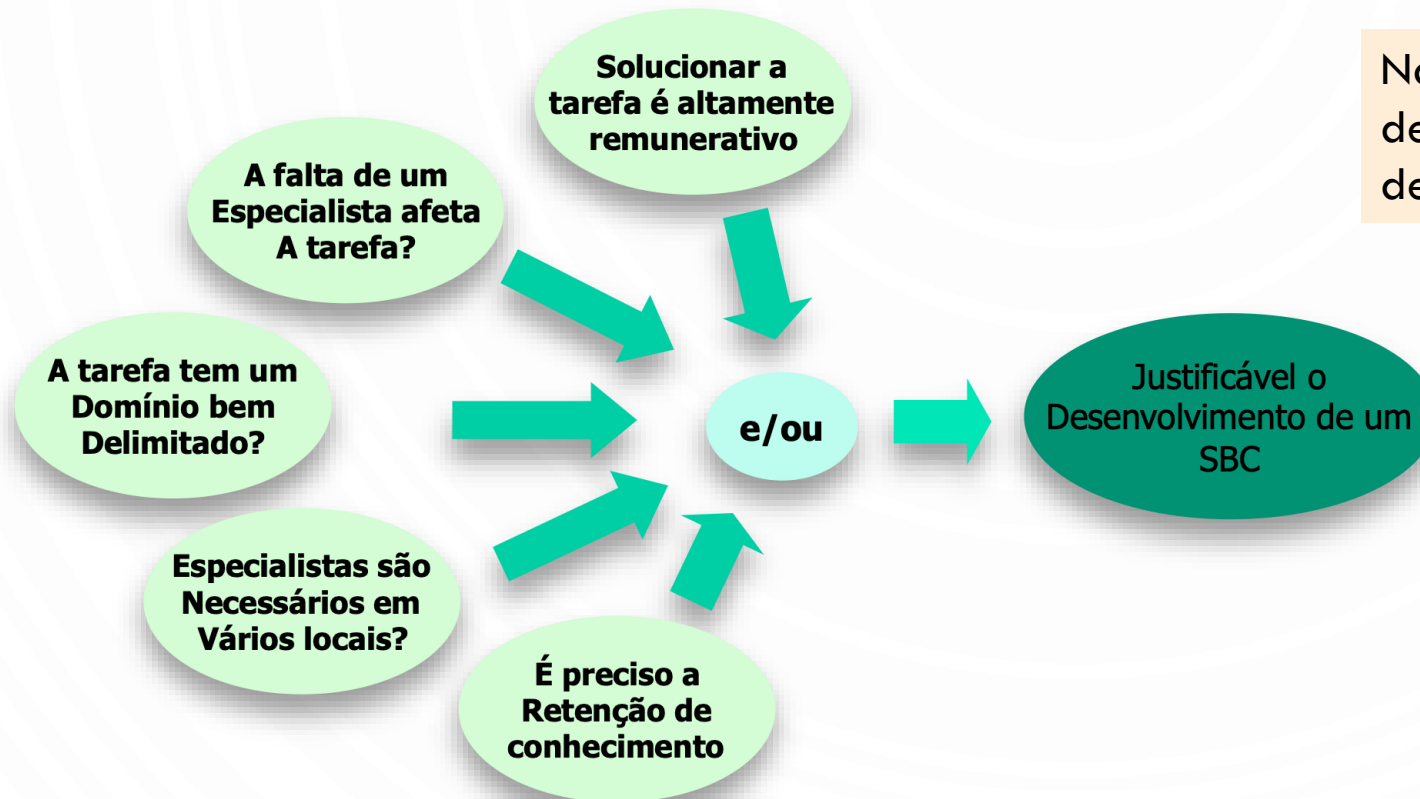
- *Módulo de explicação*
  - *Informa justificativa das conclusões obtidas e dos motivos pelos quais o SE fez determinadas perguntas*
  - *Usuário pode validar se deve ou não seguir o conselho*

Base da xAI (Explainable AI – IA Explicável)



# SISTEMAS ESPECIALISTAS

- *Análise de viabilidade de um SE*



Na atualidade, aquisição automática de conhecimento alinhada à aprendizagem de máquina é um tópico em pesquisas...

# FERRAMENTAS PARA CONSTRUÇÃO DE SE (JAVA)


- **Exemplo:** Drools. O caminho de um nó raiz até folha representa o lado esquerdo da regra, ou seja, quando chega ao nó folha todas as premissas foram satisfeitas e a regra é disparada (fired); Jboss Drools com plugins Eclipse, Netbeans etc. [drools.org](http://drools.org)

Table of Contents

- 1. Introduction
  - 1.1. Introduction
  - 1.2. Getting Involved
    - 1.2.1. Sign up to jboss.org
    - 1.2.2. Sign the Contributor Agreement
    - 1.2.3. Submitting issues via JIRA
    - 1.2.4. Fork GitHub
    - 1.2.5. Writing Tests
    - 1.2.6. Commit with Correct Conventions
    - 1.2.7. Submit Pull Requests
  - 1.3. Installation and Setup (Core and IDE)
    - 1.3.1. Installing and using
    - 1.3.2. Building from source
    - 1.3.3. Eclipse
- 2. Release Notes
  - 2.1. What is New and Noteworthy in Drools 7.1
  - 2.2. New and Noteworthy in KIE Workbench 7.1.0
    - 2.2.1. Project Metrics Dashboard
    - 2.2.2. Teams Metrics Dashboard
  - 2.3. What is New and Noteworthy in Drools 7.0
    - 2.3.1. Core Engine
    - 2.3.2. Workbench
  - 2.4. Breaking changes in Drools 7.0 from 6.x
    - 2.4.1. Property reactivity enabled by default
    - 2.4.2. Type preserving accumulate

## Drools Documentation

The JBoss Drools Team <<https://www.drools.org/community/team.html>> – Version 7.1.0.Final



### Welcome and Release Notes

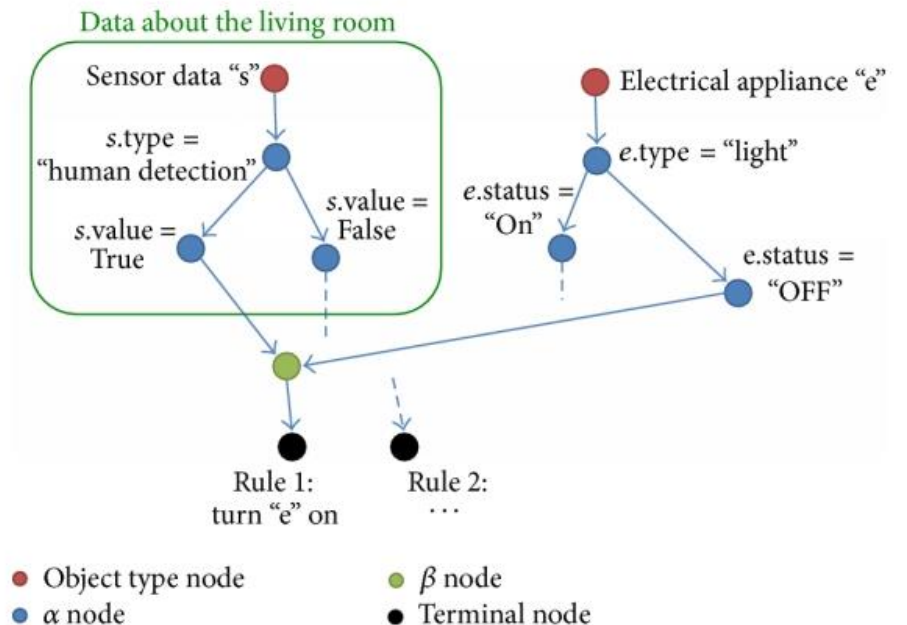
## 1. Introduction

### 1.1. Introduction

KIE (Knowledge Is Everything) is an umbrella project introduced to bring our related technologies together under one roof. It also acts as the core shared between our projects.

KIE contains the following different but related projects offering a complete portfolio of solutions for business automation and management:

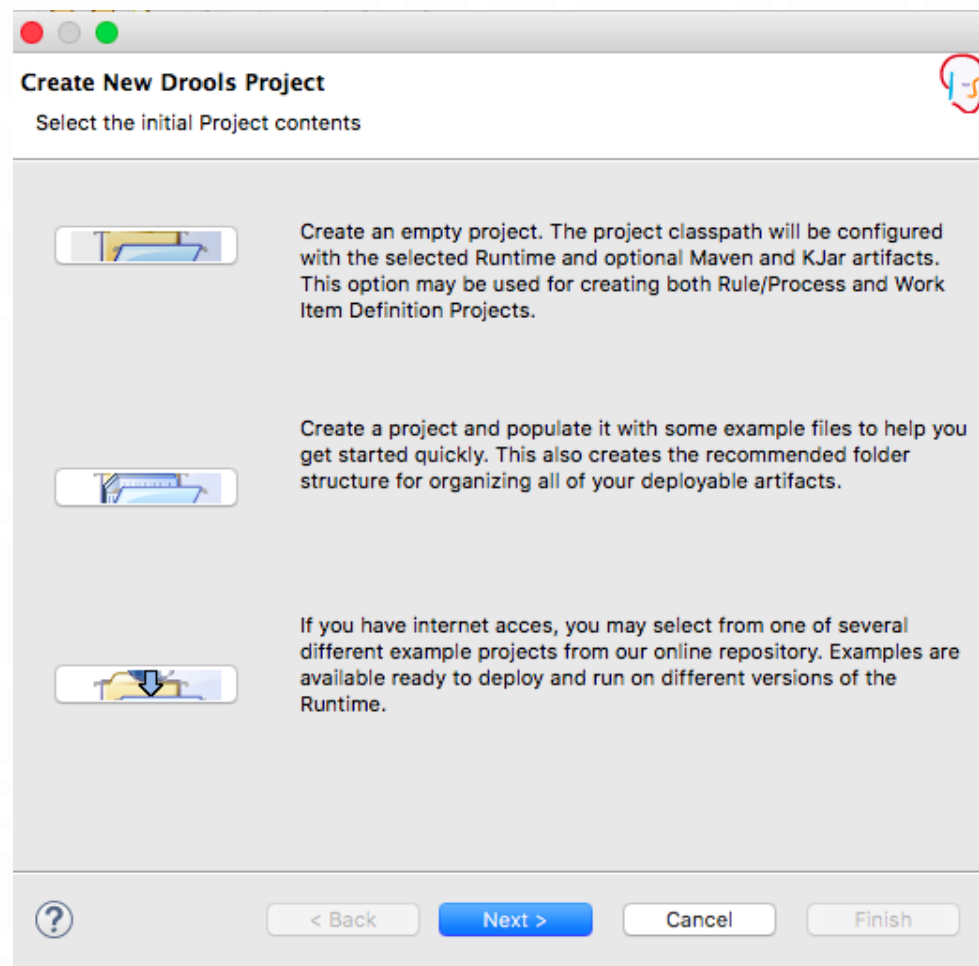
1. **Drools** is a business rule management system with a forward-chaining and backward-chaining inference based rules engine, allowing fast and reliable evaluation of business rules and complex event processing. A rule engine is also a fundamental building block to create an expert system which, in artificial intelligence, is a computer system that emulates the decision-making ability of a human expert.
2. **jBPM** is a flexible Business Process Management suite allowing you to model your business goals by describing the steps that need to be executed to achieve those goals.
3. **OptaPlanner** is a constraint solver that optimizes use cases such as employee rostering, vehicle routing, task assignment and cloud optimization.



Exemplo algoritmo Rete

# FERRAMENTAS PARA CONSTRUÇÃO DE SE

- *Drools: dependência drools e gef, que é GUI para o drools*
- Criar novo projeto Drools e escolher inicialmente a segunda opção



# EXEMPLO DE REGRAS EM DROOLS

- Arquivo de regras com extensão *.drl* (drools rules) com o formato abaixo

```
1 rule "Nova Regra" // Inicio da regra
2   when
3     // Condicoes
4   then
5     // Acoes
6 end // Fim da regra
```

- Na cláusula *When* faz-se testes sobre os objetos inseridos na memória de trabalho (base de fatos)
- Na ação colocam-se comandos Java a serem executados se a condição for verdadeira
- A regra especifica que quando um determinado conjunto de condições ocorrer, especificado no lado esquerdo (LHS), é realizado o que está no lado direito RHS
- O quanto (*When*) indica que a avaliação da condição ocorre continuamente, em qualquer momento da vida útil do motor de regras e que sempre que forem atendidas as condições a regra é disparada

# EXEMPLO: LICENÇA PARA DIRIGIR

- Duas regras para testar se um candidato pode realizar o teste a fim de obter licença para dirigir*

```
drivingLicence.drl  Candidato.java  Main.java
1 package drivingLicenseExample;
2
3 public class Candidato {
4     private String nome;
5     private int idade;
6     private boolean valido;
7
8     public Candidato(String nome, int idade){
9         this.nome = nome;
10        this.idade = idade;
11    }
12
13    public String getNome() {
14        return nome;
15    }
16    public void setNome(String nome) {
17        this.nome = nome;
18    }
19    public int getIdade() {
20        return idade;
21    }
22    public void setIdade(int idade) {
23        this.idade = idade;
24    }
25    public boolean isValid() {
26        return valido;
27    }
28    public void setValido(boolean valido) {
29        this.valido = valido;
30    }
31
32
33 }
34
```

```
drivingLicence.drl  Candidato.java  Main.java
1 //created on: 18/07/2016|
2 package rules
3
4 //list any import classes here.
5 import drivingLicenseExample.Candidato;
6
7
8 rule "idade valida"
9
10     when
11         //conditions
12         c : Candidato(idade < 18)
13     then
14         //actions
15         c.setValido(false);
16         System.out.println(c.getNome() + " não pode ter uma licença para dirigir");
17
18     end
19
20 rule "idade nao valida"
21     //include attributes such as "salience" here...
22     when
23         //conditions
24         c : Candidato(idade >= 18)
25     then
26         //actions
27         c.setValido(true);
28         System.out.println(c.getNome() + " pode ter uma licença para dirigir");
29
30     end
31
32
```

# EXEMPLO: LICENÇA PARA DIRIGIR

- *Duas regras para testar se um candidato pode realizar o teste a fim de obter licença para dirigir*

- As linhas 13-15 criam e carregam memória de trabalho (kSession)
- As linhas 20-21 inserem fatos kSession.insert
- Regras são chamadas para que se busque uma solução .fireAllRules()

```
drivingLicence.drl  Candidato.java  Main.java
1 package drivingLicenseExample;
2
3 import org.kie.api.KieServices;
4
5
6
7
8 public class Main {
9
10     public static void main(String[] args) {
11         try {
12             // load up the knowledge base
13             KieServices ks = KieServices.Factory.get();
14             KieContainer kContainer = ks.getKieClasspathContainer();
15             KieSession kSession = kContainer.newKieSession("ksession-rules");
16
17             // go !
18             Candidato c1 = new Candidato("João da Silva", 16);
19             Candidato c2 = new Candidato("Maria José", 21);
20             kSession.insert(c1);
21             kSession.insert(c2);
22             kSession.fireAllRules();
23         } catch (Throwable t) {
24             t.printStackTrace();
25         }
26     }
27
28 }
29
```



# EXEMPLO: AR-CONDICIONADO

```
ArCondicionado.java Casa.java Natureza.java Main.java
1 package com.sample;
2
3 public class Casa {
4     private String dono;
5     private ArCondicionado ar;
6
7     public Casa(String dono) {
8         this.dono = dono;
9         ar = new ArCondicionado(this);
10    }
11
12    public String getDono() {
13        return dono;
14    }
15
16    public ArCondicionado getAr() {
17        return ar;
18    }
19
20 }
21
```

```
ArCondicionado.java Casa.java Natureza.java Main.java
1 package com.sample;
2
3 public class Natureza {
4     private int temperatura;
5
6     public int getTemperatura() {
7         return temperatura;
8     }
9
10    public void setTemperatura(int temperatura) {
11        this.temperatura = temperatura;
12    }
13
14 }
15
16
```

# EXEMPLO: AR-CONDICIONADO

```
ArCondicionado.java Casa.java Natureza.java Main.java
1 package com.sample;
2
3 public class ArCondicionado {
4     private boolean ligado = false;
5     private Casa casaInstalada;
6
7     public ArCondicionado(Casa casaInstalada) {
8         this.casaInstalada = casaInstalada;
9     }
10
11    public void ligar(){
12        if(!ligado){
13            System.out.println("Ligando ar condicionado da casa de " + casaInstalada.getDono());
14            ligado = true;
15        }
16    }
17
18    public void desligar(){
19        if(ligado){
20            System.out.println("Desligando ar condicionado da casa de " + casaInstalada.getDono());
21            ligado = false;
22        }
23    }
24
25 }
26
27
```

# EXEMPLO: AR-CONDICIONADO

```
ArCondicionado.java Casa.java Natureza.java Main.java
1 package com.sample;
2
3 import org.kie.api.KieServices;
4
5
6
7
8 /**
9  * This is a sample class to launch a rule.
10  */
11 public class Main {
12
13     public static final void main(String[] args) {
14         try {
15             // load up the knowledge base
16             KieServices ks = KieServices.Factory.get();
17             KieContainer kContainer = ks.getKieClasspathContainer();
18             KieSession kSession = kContainer.newKieSession("ksession-rules");
19
20             // go !
21             Casa casaRafael = new Casa("Rafael");
22             Casa casaFrederico = new Casa("Frederico");
23             Natureza nature = new Natureza();
24             kSession.insert(casaRafael);
25             kSession.insert(casaFrederico);
26             FactHandle handleNatureza = kSession.insert(nature); //Fatos modificáveis
27
28             for(int temp = 30; temp > 15; temp--){
29                 System.out.println("Temperatura atual: "+temp);
30                 nature.setTemperatura(temp);
31                 kSession.update(handleNatureza, nature); //Atualiza o fato
32                 kSession.fireAllRules();
33             }
34
35         } catch (Throwable t) {
36             t.printStackTrace();
37         }
38     }
39
40
41
42 }
```

# EXEMPLO: AR-CONDICIONADO

```
ArCondicionado.java Casa.java Natureza.java Main.java Sample.drl
1 package com.sample
2
3 import com.sample.Main;
4
5 rule "Ligar ar condicionado da casa de Rafael quando a temperatura for maior que 26 graus"
6     when
7         Casa(dono == "Rafael", ac : ar)
8         Natureza(temperatura > 26)
9     then
10        ac.ligar();
11    end
12
13 rule "Desligar ar condicionado da casa de Rafael quando a temperatura for menor que 20 graus"
14     when
15         Casa(dono == "Rafael", ac : ar)
16         Natureza(temperatura < 20)
17     then
18        ac.desligar();
19    end
20
21 rule "Ligar ar condicionado da casa de Frederico quando a temperatura for maior que 21 graus"
22     when
23         Casa(dono == "Frederico", ac : ar)
24         Natureza(temperatura > 21)
25     then
26        ac.ligar();
27    end
28
29 rule "Desligar ar condicionado da casa de Frederico quando a temperatura for menor que 18 graus"
30     when
31         Casa(dono == "Frederico", ac : ar)
32         Natureza(temperatura < 18)
33     then
34        ac.desligar();
35    end
```

TECINFO-UFRN: INTELIGÊNCIA ARTIFICIAL, PROF. JOSENALDE OLIVEIRA

# EXEMPLO: AR-CONDICIONADO

- Na cláusula when pode-se realizar testes sobre valores de atributos dos objetos inseridos no kSession
  - Para isto ter na classe os métodos get
- Se for preciso um objeto chamar um método na cláusula then, então deve-se definir este objeto no when
- No caso de criar um objeto que é atributo da classe em teste, usa-se o nome do atributo, como em ac:ar

```
9 rule "idade valida"
10
11     when
12         //conditions
13         c : Candidato(idade < 18)
14     then
15         //actions
16         c.setValido(false);
17         System.out.println(c.getNome() + " não pode ter uma licença para dirigir");
18
19     end
```

```
4
5 rule "Ligar ar condicionado da casa de Rafael quando a temperatura for maior que 26 graus"
6     when
7         Casa(dono == "Rafael", ac : ar)
8         Natureza(temperatura > 26)
9     then
10         ac.ligar();
11     end
```

# EXEMPLO: SUPORTE TÉCNICO\_EAJ



Figura 18-tela de suporte de Hardware



Silva, Cláudia L.B.; Oliveira, J.B. Integração do motor JBoss Drools ao Desenvolvimento de Sistemas. TCC Tec. Info, UFRN/EAJ, 2015.



# FRAMEWORKS EM PYTHON

durable-rules: <https://github.com/jruizgit/rules>

...