

**Universidade do Minho**

***Realizado por***

*Ana Duarte, A74407*

*Ângela Gonçalves, A76542*

*Diogo Bessa, A75544*

***Entregue em***

*3 de janeiro de 2019*

***Docente***

*Professor César Analide*

***Disciplina de Programação em Lógica, Conhecimento e Raciocínio***

*Mestrado Integrado em*

*Engenharia Biomédica*

*Ano Letivo de 2018/2019*

**Trabalho Prático 3**

Regras de Produção

**Sumário**

A inserção de regras de produção, que definem factos que dependem de outros que lhe são antecessores, permite a execução de programas em PROLOG que contenham um grande conjunto de conclusões sem que estas tenham sido declaradas expressamente. Assim, com este trabalho, procedeu-se à elaboração de um sistema de apoio à decisão clínica que comprove diagnósticos e tratamentos médicos, tendo por base os sintomas que foram declarados.

A estrutura do trabalho é composta por uma parte com declarações iniciais, outra com os sistemas de inferência, uma outra com funcionalidades e uma parte com predicados auxiliares.

Após as simulações efetuadas, verificou-se que o programa representava adequadamente o caso em estudo podendo, de futuro, interligar-se em programas mais complexos, com mais funcionalidades e com a inclusão de conhecimento imperfeito.

**Palavras-chave:** PROLOG, Backward Chaining, Forward Chaining.

Índice

[**1.Introdução** 1](#_Toc534351398)

[**2.Preliminares** 2](#_Toc534351399)

[2.1. Sistema de Inferência *Backward Chaining* 2](#_Toc534351400)

[2.2. Sistema de Inferência *Forward Chaining* 3](#_Toc534351401)

[**3.Descrição do Trabalho e Análise de Resultados** 5](#_Toc534351402)

[3.1. Declarações Iniciais 5](#_Toc534351403)

[3.2. Base de Conhecimento 5](#_Toc534351404)

[3.3. Regras de Produção 6](#_Toc534351405)

[3.3.1. Sistema *Backward Chaining* 7](#_Toc534351406)

[3.3.2. Sistema *Forward Chaining* 10](#_Toc534351407)

[3.4. Funcionalidades 12](#_Toc534351408)

[3.4.1. Consultar Sintomas, Diagnósticos e/ou Tratamentos 12](#_Toc534351409)

[3.4.2. Consultar Utentes que apresentam um determinado Diagnóstico 15](#_Toc534351410)

[3.4.3. Evolução do Conhecimento: Adicionar e Remover Sintomas 15](#_Toc534351411)

[3.5. Predicados Auxiliares 16](#_Toc534351412)

[3.6. Invariantes 16](#_Toc534351413)

[**4. Conclusão** 18](#_Toc534351414)

[**5. Bibliografia** 19](#_Toc534351415)

# **1.Introdução**

Este trabalho tem como principal objetivo a utilização de regras de produção para a representação de conhecimento, num sistema de apoio à decisão em ambiente clínico, aplicável ao diagnóstico médico e prescrição de medicamentos. O sistema deve ser capaz de representar, de forma adequada, condições que implicam conclusões como, as relações do tipo “SE Condição ENTÃO Conclusão”. Além disso, o sistema, deve possuir funcionalidades de resposta a questões de verificação dos diagnósticos associados aos utentes e respetivos tratamentos e grau de probabilidade de ocorrência.

O objetivo do trabalho prende-se, assim, com a geração de um programa baseado em lógica matemática, que consiga deduzir conclusões sem que estas tenham sido expressamente declaradas.

Para isso, é então necessária a representação de conhecimento, intencional (regras) e extensional (evidências), diminuindo-se, deste modo, a necessidade de memória física na base do conhecimento, uma vez que, parte do conhecimento (evidências) é deduzido a partir de regras.

# **2.Preliminares**

Um sistema com base em regras de produção é capaz de resolver um intervalo de problemas alargado, por vezes complexos, recorrendo a regras simples e combinação de resultados. Representa-se o conhecimento como uma coleção de regras do tipo: “SE Condição ENTÃO conclusão”. O conjunto de regras de produção é visto como uma representação de conhecimento procedimental.

A vantagem do uso deste tipo de representação de conhecimento, está na facilidade de determinação de um dado diagnóstico médico consoante um conjunto de sintomas, contemplando vários diagnósticos assim como tratamentos médicos associados.

Além disso, há ainda a possibilidade de se associar um grau de confiança a uma dada regra de produção, que corresponde à probabilidade de ocorrência de uma determinada conclusão. O grau de confiança poderá representar um valor qualitativo ou quantitativo, e uma possível representação é: “SE Condição ENTÃO Conclusão ::: GRAU”.

Como forma de verificação do funcionamento do sistema poderão ser utilizados sistemas de inferência baseados em processos *Backward Chaining* ou *Forward Chaining*. O processo de *Backward chaining* é similar ao teste de hipóteses em problemas com resolução humana, por exemplo, um médico ao suspeitar que um paciente tem um determinado problema, vai procurar certos sintomas de forma a comprovar a hipótese. Assim este modelo é caraterizado por focar no objetivo, ou seja, começa no topo e vai descendo até atingir as condições iniciais que provem esse objetivo.[1][2]

Por sua vez, no processo *Forward Chaining*, a solução começa com a informação básica, onde se vai relacionar essa informação de forma a obter conclusões lógicas. Por exemplo, ao contrário do *Backward chaining*, um médico, neste modelo, vai começar por perguntar ao paciente os seus sintomas e, depois, a partir dessa informação, vai estabelecer uma hipótese para ser testada. É então chamado de um modelo “*bottom-up*”, ou seja, começa com factos e aplica regras para gerar novos factos  que, por sua vez, vão originar premissas, continuando o ciclo até chegar a um determinado objetivo ou à falta de factos nos quais as premissas correspondam aos factos conhecidos.[1][2]

## 2.1. Sistema de Inferência *Backward Chaining*

O sistema de inferência designado por *Backward chaining* apresenta-se, assim, como uma forma de colocar questões e obter respostas, tendo em conta os o conhecimento existente. As suas principais características de pesquisa são:[2]

* Inicia a pesquisa com alguma hipótese ou conclusão;
* Faz internamente apenas algumas perguntas;
* Testa apenas algumas regras;
* É um mecanismo de pesquisa rápido porque testa poucas regras;
* Fornece uma pequena quantidade de informação, limitada a apenas uma pequena quantidade de dados;
* A pesquisa é efetuada apenas na parte da base de conhecimento que é relevante para o problema atual;
* É orientado por objetivos (conclusões);
* Inicia com uma hipótese e procura as informações até a hipótese poder ser aceite ou rejeitada;
* Tem uma lógica de pesquisa *bottom-up* (de baixo para cima);
* Pesquisa para trás até encontrar os factos que sustentam a hipótese/conclusão;
* Efetua primeiro uma pesquisa em profundidade
* É um sistema adequado para problemas que se iniciam a partir de uma hipótese como, por exemplo, a indicação de um determinado diagnóstico médico;
* Trata-se de um sistema que foca as perguntas, na base de conhecimento, tendo em conta o objetivo e pesquisando apenas o conhecimento que está relacionado com esse objetivo;
* O sistema facilita a explicação do diagnóstico;
* Os dados devem ser adquiridos de forma interativa;
* Um pequeno número de objetivos iniciais e um grande número de regras correspondem aos factos que utiliza;
* Fácil para definir uma decisão.

O sistema *Backward Chaining* é mais adequado para problemas que se “alargam” a partir dos factos como os do esquema da Figura 1.

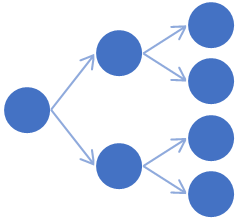


Figura - Esquema tipo mais adequado para o Backward Chaining

## 2.2. Sistema de Inferência *Forward Chaining*

Outro sistema de inferência utilizado em conjunto com as regras de produção é o *Forward Chaining*. As suas principais características são as que se enumeram de seguida:[2]

* Inicia a pesquisa a partir dos factos iniciais;
* Executa internamente muitas perguntas;
* Testa todas as regras existentes;
* Tem um processamento lento porque testa todas as regras;
* Fornece uma enorme quantidade de informações a partir de um pequeno conjunto de dados;
* Infere tudo o que é possível a partir das informações existente na base de conhecimento;
* É sobretudo orientado aos dados;
* A partir da entrada pesquisa as regras até obter a resposta;
* Tem um raciocínio do tipo *top-down* (de cima para baixo);
* Pesquisa para a frente para encontrar as conclusões dos factos;
* Ao contrário do *Backward*, faz a pesquisa com prioridade à largura;
* Está vocacionado para problemas que se iniciam na coleta de dados como, por exemplo, o seu planeamento, monitorização e controlo;
* Não é um sistema focado porque infere todas as conclusões que for possível obter, respondendo a perguntas não relacionadas;
* O sistema não é adequado para explicar as conclusões;
* Todos os dados são disponibilizados;
* A partir de um pequeno número de dados iniciais gera um alto número de conclusões;
* É um sistema mais difícil para atingir as conclusões.

Na Figura 2 apresenta-se um esquema de encadeamento de regras de produção que mais se adequa a um sistema *Forward Chaining*.

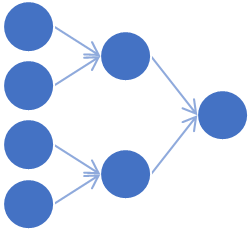


Figura - Esquema tipo mais adequado para o Forward Chaining

# **3.Descrição do Trabalho e Análise de Resultados**

## 3.1. Declarações Iniciais

Com o intuito de se evitarem erros e avisos desnecessário, antes de se implementar o programa propriamente dito, é preciso inserir-se um conjunto de declarações que visam garantir o seu correto funcionamento. Além disso, importa ainda declarar a simbologia a ser utilizada durante a implementação do programa. Dado o contexto em que se insere este trabalho prático, considerou-se necessário definir-se os elementos “e”, “ou”, “se”, “então”, “com”, “porque”, “de”, “facto”, “::” e “:::”.

Assim, foram indicadas as seguintes definições iniciais:

*:- set\_prolog\_flag(discontiguous\_warnings,off).*

*:- set\_prolog\_flag(single\_var\_warnings,off).*

*:- set\_prolog\_flag(unknown,fail).*

*:- op( 800,fx,se ).*

*:- op( 800,fx,facto ).*

*:- op( 700,xfx,entao ).*

*:- op( 300,xfy,ou ).*

*:- op( 200,xfy,e ).*

*:- op( 900,xfx,com).*

*:- op( 900,xfx,porque).*

*:- op( 900,xfx,de).*

*:- op( 900,xfx,::).*

*:- op( 900,xfx,:::).*

Adicionalmente, é preciso ainda declararem-se os predicados dinâmicos, cujos argumentos podem ser modificados durante a execução do programa. Note-se que, para cada um destes predicados é ainda especificado o número de argumentos pelos quais é composto.

*:- dynamic facto/2.*

*:- dynamic utente/5.*

*:- dynamic '::'/2.*

*:- dynamic ':::'/2.*

## 3.2. Base de Conhecimento

De forma a desenvolver-se um programa que funcione como um sistema de apoio à decisão em contexto clínico, que seja capaz de lidar com sintomas e diagnósticos médicos e a respetiva prescrição de medicamentos, é preciso construir-se uma base de conhecimento adequada. Esta base de conhecimento deverá ser capaz de lidar com conhecimento extensional (factos) e conhecimento intencional (regras).

Para tal, optou-se por se incluírem registos iniciais relativos a utentes, factos e regras, que seguem a seguinte estrutura:

*utente( #idUtente, Nome, Idade, Género, Morada).*

*facto( #idUtente, Sintoma) :: Grau.*

*se (#idUtente, Sintoma) então (#idUtente, Diagnóstico) :: (Grau com Prescrição).*

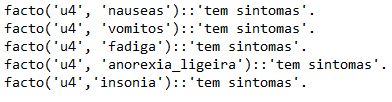
A título de exemplo, para se aproximar as simulações a situações em contexto real, inseriu-se utentes, factos e regras da seguinte forma:

*utente( u1, rita\_pinto, 41, feminino, braga).*

*facto( u2, stress\_cronico) :: ‘tem sintomas’.*

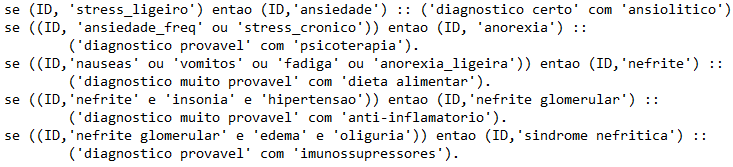
*se (ID, stress\_ligeiro) então (ID, ansiedade) :: (‘diagnóstico certo’ com ansiolítico’).*

Para além dos utentes foram também introduzidos dados iniciais relativos a factos. Por exemplo, para o utente u4, foram introduzidos os seguintes factos:



## 3.3. Regras de Produção

As regras de produção traduzem para o programa as situações de causalidade do tipo ‘Se … Então’ e podem ser, por exemplo, aplicadas em contexto clínico para a realização de um diagnóstico e da respetiva prescrição tendo em conta os sintomas apresentados pelo utente. A este nível foram inseridas as seguintes regras de produção:



Estas regras traduzem para o PROLOG os diagnósticos e os tratamentos que estão associados a determinados sintomas, tal como está ilustrado na Figura 3.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Stress Ligeiro  Figura - Esquema do Problema  **OU**  **OU**  **OU**  **OU**  **E**  **E**  **E**  **E** |  | Ansiedade |  |  |  |  |
|  |  |  |  |  |  |  |
| Ansiedade Frequente |  |  |  |  |  |  |
|  |  | Anorexia Mental |  |  |  |  |
| Stress crónico |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| Náuseas |  |  |  |  |  |  |
|  |  |  |  |  |  |  |
| Vómitos |  |  |  |  |  |  |
|  |  | Nefrite |  |  |  |  |
| Fadiga |  |  |  |  |  |  |
|  |  | Insónia |  | Nefrite Glomerular |  |  |
| Ansiedade Ligeira |  |  |  |  |  |  |
|  |  | Hipertensão |  | Edema |  | Síndrome Nefrítica |
|  |  |  |  |  |  |  |
|  |  |  |  | Oligúria |  |  |

No caso do problema em estudo, como se comprova pela análise da Figura 3, o sistema de inferência mais adequado é o de *Forward Chaining*. No entanto, tendo em conta as características da figura, também se pode considerar adequado utilizar-se o *Backward Chaining.* Assim, optou-se por se construir os dois sistemas de inferência para verificar o comportamento de cada um deles.

### 3.3.1. Sistema *Backward Chaining*

Para a implementação deste sistema em concreto foram definidas as seguintes instruções:

*% Extensao do meta-predicado rp: ID, Diagnóstico, Grau, Explicacao, Tratamento -> {V,F}*

*rp(ID, D, G, G de D, []):-nonvar(D), (facto(ID, D))::G.*

*rp(ID, D, G, (D com G) porque Ex, [X|L]):-(se (ID, C) entao (ID, D))::(Gr com X),*

*rp(ID, C, Gc, Ex, L), grau\_multiplo(Gr, Gc, G).*

*rp(ID,(C1 e C2), G,(E1 e E2),Lista):- nonvar(C1),nonvar(C2), rp(ID, C1, G1, E1, L1),*

*rp(ID, C2, G2, E2, L2), menor(G1,G2,G), concatenar(L1,L2,Lista).*

*rp(ID,(C1 ou C2), G,(E1 e E2),Lista):- nonvar(C1),nonvar(C2), rp(ID,C1,G1,E1,L1),*

*rp(ID,C2,G2,E2,L2), maior(G1,G2,G), concatenar(L1,L2,Lista).*

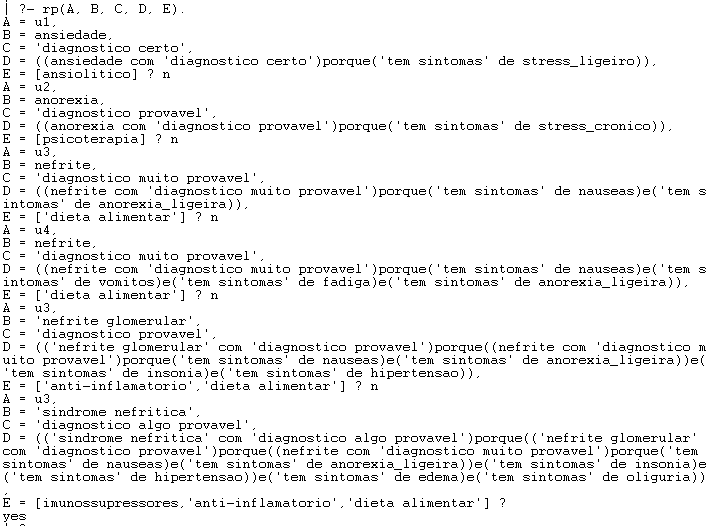
*rp(ID,(C1 ou C2), G1, E1, L1):- nonvar(C1),nonvar(C2), rp(ID,C1,G1, E1, L1),*

*nao(rp(ID,C2, G2, E2, L2)).*

*rp(ID,(C1 ou C2), G2, E2, L2):- nonvar(C1),nonvar(C2), rp(ID, C2, G2, E2, L2),*

*nao(rp(ID, C1, G1, E1, L1)).*

***Justificação:*** *Na primeira cláusula define-se o critério de paragem que é dado pela lista vazia no campo da prescrição. A lista vazia acontece quando não se aplicam tratamentos, ou seja, quando só existem os sintomas. A segunda cláusula é a base do sistema de inferência que vai procurar, para um dado diagnóstico, os factos que o originam. No caso de encontrar factos que justificam o diagnóstico remete, recursivamente para outros factos que justifiquem os primeiros e, simultaneamente calcula o seu grau de probabilidade. Este grau aplica-se de uma forma idêntica às probabilidades de ocorrerem factos de acontecimentos sucessivos, ou seja, através da multiplicação. Como os parâmetros são qualitativos foi criada a Tabela 1 que atribui a probabilidade de acontecerem dois factos sucessivos. As restantes cláusulas definem as regras lógicas da conjunção e disjunção, em que existem 4 situações: a conjunção entre dois acontecimentos é verdade se os cada um deles é verdade; a disjunção é verdade se ambos forem verdade ou se um é verdade e outro falso (e vice-versa). Além disso, na conjunção, isto é, dois acontecimentos justificam uma conclusão se ambos os acontecimentos ocorrerem, que se reflete em termos de probabilidades no valor referente à mais baixa probabilidade de ocorrer. No caso da disjunção, como basta que um dos dois ocorra, a probabilidade de ocorrência é a maior dos dois acontecimentos. Como as probabilidades, ou graus de ocorrência, são qualitativos existiu a necessidade de se definirem todas as situações de maior e de menor probabilidade, tal como se indica nas Tabelas 2 e 3.*

***Simulação:***



A tabela que relaciona os graus (probabilidades de ocorrência) é a Tabela 1.

Tabela : Grau Múltiplo

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Grau Múltiplo | Tem sintomas | Diagnóstico Certo | Diagnóstico muito provável | Diagnóstico provável | Diagnóstico algo provável | Diagnóstico pouco provável |
| Tem sintomas | Tem sintomas | Diagnóstico Certo | Diagnóstico muito provável | Diagnóstico provável | Diagnóstico algo provável | Diagnóstico pouco provável |
| Diagnóstico Certo | Diagnóstico Certo | Diagnóstico Certo | Diagnóstico muito provável | Diagnóstico provável | Diagnóstico algo provável | Diagnóstico pouco provável |
| Diagnóstico muito provável | Diagnóstico muito provável | Diagnóstico muito provável | Diagnóstico provável | Diagnóstico provável | Diagnóstico provável | Diagnóstico algo provável |
| Diagnóstico provável | Diagnóstico provável | Diagnóstico provável | Diagnóstico provável | Diagnóstico algo provável | Diagnóstico algo provável | Diagnóstico algo provável |
| Diagnóstico algo provável | Diagnóstico algo provável | Diagnóstico algo provável | Diagnóstico provável | Diagnóstico algo provável | Diagnóstico pouco provável | Diagnóstico pouco provável |
| Diagnóstico pouco provável | Diagnóstico pouco provável | Diagnóstico pouco provável | Diagnóstico pouco provável | Diagnóstico algo provável | Diagnóstico pouco provável | Diagnóstico pouco provável |

Apresenta-se de seguida a Tabela 2 que relaciona a probabilidade de ocorrência quando se pretende determinar a maior probabilidade (ou grau).

Tabela : Maior Grau

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Maior Grau | Tem sintomas | Diagnóstico Certo | Diagnóstico muito provável | Diagnóstico provável | Diagnóstico algo provável | Diagnóstico pouco provável |
| Tem sintomas | Tem sintomas | Tem sintomas | Tem sintomas | Tem sintomas | Tem sintomas | Tem sintomas |
| Diagnóstico Certo | Tem sintomas | Diagnóstico Certo | Diagnóstico Certo | Diagnóstico Certo | Diagnóstico Certo | Diagnóstico Certo |
| Diagnóstico muito provável | Tem sintomas | Diagnóstico Certo | Diagnóstico muito provável | Diagnóstico muito provável | Diagnóstico muito provável | Diagnóstico muito provável |
| Diagnóstico provável | Tem sintomas | Diagnóstico Certo | Diagnóstico muito provável | Diagnóstico provável | Diagnóstico provável | Diagnóstico provável |
| Diagnóstico algo provável | Tem sintomas | Diagnóstico Certo | Diagnóstico muito provável | Diagnóstico provável | Diagnóstico algo provável | Diagnóstico algo provável |
| Diagnóstico pouco provável | Tem sintomas | Diagnóstico Certo | Diagnóstico muito provável | Diagnóstico provável | Diagnóstico algo provável | Diagnóstico pouco provável |

Apresenta-se ainda a correspondência de “menor” probabilidade:

Tabela : Menor Grau

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Menor Grau | Tem sintomas | Diagnóstico Certo | Diagnóstico muito provável | Diagnóstico provável | Diagnóstico algo provável | Diagnóstico pouco provável |
| Tem sintomas | Tem sintomas | Diagnóstico Certo | Diagnóstico muito provável | Diagnóstico provável | Diagnóstico algo provável | Diagnóstico pouco provável |
| Diagnóstico Certo | Diagnóstico Certo | Diagnóstico Certo | Diagnóstico muito provável | Diagnóstico provável | Diagnóstico algo provável | Diagnóstico pouco provável |
| Diagnóstico muito provável | Diagnóstico muito provável | Diagnóstico muito provável | Diagnóstico muito provável | Diagnóstico provável | Diagnóstico algo provável | Diagnóstico pouco provável |
| Diagnóstico provável | Diagnóstico provável | Diagnóstico provável | Diagnóstico provável | Diagnóstico provável | Diagnóstico algo provável | Diagnóstico pouco provável |
| Diagnóstico algo provável | Diagnóstico algo provável | Diagnóstico algo provável | Diagnóstico algo provável | Diagnóstico algo provável | Diagnóstico algo provável | Diagnóstico pouco provável |
| Diagnóstico pouco provável | Diagnóstico pouco provável | Diagnóstico pouco provável | Diagnóstico pouco provável | Diagnóstico pouco provável | Diagnóstico pouco provável | Diagnóstico pouco provável |

### 3.3.2. Sistema *Forward Chaining*

O sistema *Forward Chaining* foi construído pela introdução da seguinte sequência de instruções:

*% Extensao do meta-predicado rpf: ID, Diagnóstico, Grau, Explicacao, Tratamento -> {V,F},*

*rpf(ID, D, G, (D com G) porque (Ex com G), T1):-*

*(se (ID, C) entao (ID, D)) :: (G1 com T1), composicao(ID, C, G2, Ex, T2),*

*grau\_multiplo(G1, G2, G), inserir\_facto((facto(ID, D)) :: G).*

*inserir\_facto((facto(ID, F)) :: G):- nao((facto(ID, F)) :: G), assert((facto(ID, F)) :: G).*

*inserir\_facto(facto(ID, F) :: G):- (facto(ID, F)) :: G.*

*composicao(ID, C, G, (C com G), []):- (facto(ID, C)) :: G.*

*composicao(ID, (C1 e C2), G, (E1 e E2), [T1|T2]):-*

*composicao(ID, C1, G1, E1, T1), composicao(ID, C2, G2, E2, T2), menor(G1, G2, G).*

*composicao(ID, (C1 ou C2), G, (E1 e E2), [T1|T2]):-*

*composicao(ID, C1, G1, E1, T1), composicao(ID, C2, G2, E2, T2), maior(G1, G2, G).*

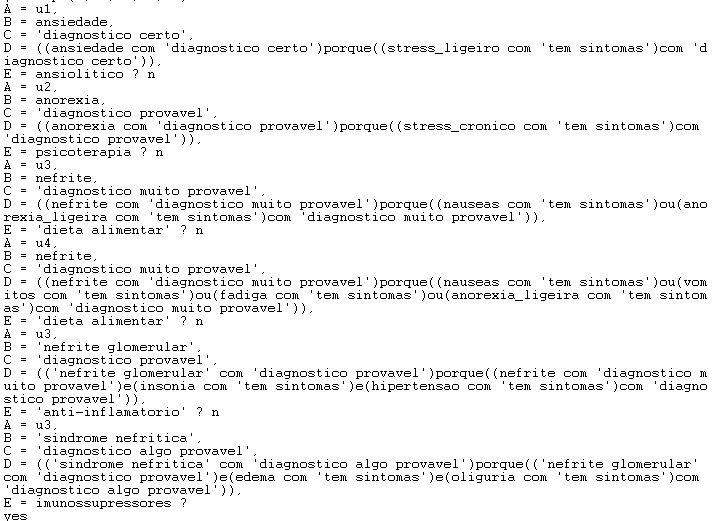
*composicao(ID, (C1 ou C2), G1, E1, T1):-*

*composicao(ID, C1, G1, E1, T1), nao(composicao(ID, C2, G2, E2, T2)).*

*composicao(ID, (C1 ou C2), G2, E2, T2):-*

*composicao(ID, C2, G2, E2, T2), nao(composicao(ID, C1, G1, E1, T1)).*

***Justificação:*** *Na primeira cláusula é indicado o funcionamento do sistema rpf que é do tipo: (ID\_utente, Diagnóstico, Grau, Diagnóstico com Grau porque (Explicação com Grau), Tratamento). Aqui, verifica-se se existe uma condição “se (ID, condição) então (ID, diagnóstico)” com um determinado grau e tratamento associados. Neste caso, remete-se para a composição e executa-se o grau\_multiplo para conjugar probabilidades consecutivas. Além disso, insere-se o facto (ID, Diagnóstico) com o Grau associado. O inserir introduz o facto na base de conhecimento e necessita que previamente se faça “derivar.” para não serem retornado os registos duplicados. A composição serve para agrupar os registos consecutivos de diagnóstico e tratamento. A composição é recursiva e coloca o primeiro resultado numa lista e executa a composição dos restantes elementos. Como tem um funcionamento recursivo, tem de se colocarem as cláusulas de paragem que impõem o Tratamento como uma lista vazia. De uma forma semelhante ao sistema Forward Chaining são colocadas as cláusulas de conjunção e disjunção de condições.*

***Simulação:***

## 3.4. Funcionalidades

Para se obter um sistema de diagnóstico médico e de prescrição que sirva de apoio à decisão, construíram-se um conjunto de funcionalidades que visam permitir obter o máximo de eficiência do sistema. Assim, foram construídas as seguintes funcionalidades:

* Consultar o histórico clínico para um determinado utente, seja ao nível dos sintomas, dos diagnósticos, da prescrição ou de todos associados;
* Determinar todos os utentes que foram diagnosticados com uma determinada doença;
* Adicionar e remover sintomas.

### 3.4.1. Consultar Sintomas, Diagnósticos e/ou Tratamentos

Os comandos de programação são os que se indicam e justificam de seguida para a funcionalidade de consultar o histórico de um doente, utilizando o sistema *Backward chaining*:

*%Extensao do predicado cs\_historico\_utente: CS, IdU, S -> {V,F}*

*% CS pode ser 1 para consultar os sintomas*

*% CS pode ser 2 para consultar os diagnosticos*

*% CS pode ser 3 para consultar os tratamentos*

*% CS pode ser 4 para consultar os sintomas, diagnosticos e tratamentos*

*cs\_historico\_utente(CS, IdU, S):- CS=1, solucoes((IdU, N), utente(IdU, N, \_, \_, \_), L1),*

*solucoes(D, (facto(IdU,D)::'tem sintomas'), L2), concatenar(L1,L2,S).*

*cs\_historico\_utente(CS, IdU, S):- CS=2, solucoes((IdU, N), utente(IdU, N, \_, \_, \_), L1),*

*solucoes(D, rp(IdU, D, G, E, T), L2), concatenar(L1,L2,S).*

*cs\_historico\_utente(CS, IdU, S):- CS=3, solucoes((IdU, N), utente(IdU, N, \_, \_, \_), L1),*

*solucoes(X, rp(IdU, D, G, E, [X|L]), L2), concatenar(L1,L2,S).*

*cs\_historico\_utente(CS, IdU, S):- CS=4, solucoes((IdU, N), utente(IdU, N, \_, \_, \_), L1), solucoes(('sintomas' de D1), (facto(IdU,D1)::'tem sintomas', rp(IdU, D1, \_, \_, \_)), L2), solucoes(('diagnostico' de D2), (facto(IdU,D1)::'tem sintomas', rp(IdU, D2, \_, \_, \_)), L3), sem\_repetidos(L3,R3), solucoes(('tratamento' de X), (facto(IdU,D1):: 'tem sintomas', rp(IdU,D2,G,E,[X|L])), L4), sem\_repetidos(L4,R4), concatenar(L1,L2,L5), concatenar(L5,R3,L6),concatenar(L6,R4,S).*

***Justificação:*** *Ao selecionar-se 1, o programa procura, através do predicado soluções, o ID do utente e o seu nome e, em simultâneo, procura o diagnóstico dos factos registados com grau igual a ‘tem sintomas. As duas condições têm de ocorrer guardando-se os resultados nas listas L1 e L2 e concatenando-as na lista final S. Note-se que, caso se tivesse optado por fazer a pesquisa com apenas um predicado “soluções”, o resultado final repetiria o nome e o ID do utente em cada novo sintoma. No caso de se escolher o valor 2 para o CS, será pesquisado o diagnóstico de uma forma idêntica aos sintomas.*

*No caso da consulta aos tratamentos, o método é semelhante aos anteriores, mas são considerados os tratamentos em forma de lista [X|L], para se evitar que o resultado seja uma lista de tratamentos para cada diagnóstico. Ou seja, se em vez de X e [X|L] estivesse T, o resultado de “cs\_historico\_utente(3,u3,S).” seria dado por:*

*S = [(u3,rui\_sousa),['dieta alimentar'],['anti-inflamatorio','dieta alimentar'],[imunossupressores,'anti-inflamatorio','dieta alimentar']]*

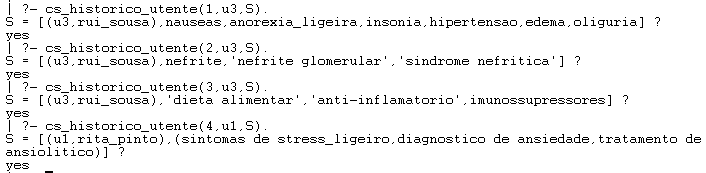
*Com a substituição de T por X e [X|L], o resultado passa a ser de:*

*S = [(u3,rui\_sousa),'dieta alimentar','anti-inflamatorio',imunossupressores]*

*Isto acontece porque com o T, sempre que se encontra um diagnóstico (por exemplo ‘nefrite’), será adicionado um tratamento e, quando surge outro diagnóstico que dependa sequencialmente do primeiro (por exemplo ‘nefrite glomerular’), o novo tratamento será incluído na lista. Note-se que, sendo o diagnóstico sequencial, o tratamento posterior inclui o tratamento anterior. A colocação de [X|L] irá incorporar na resposta o tratamento mais a jusante da cadeia de diagnóstico, que inclui todos os tratamentos anteriores.*

*Relativamente à funcionalidade de devolver os sintomas, os diagnósticos e os tratamentos em simultâneo utilizam-se dois predicados “soluções” de forma análoga aos anteriores em que, neste caso, se pesquisam com predicados individualizados de soluções para cada uma das pesquisas. Com estes predicados soluções, específicos para cada procura, eliminam-se os registos repetidos.*

***Simulação Backward Chaining:***



De uma forma semelhante à anterior, enumeram-se as instruções relativas às funcionalidades aplicando-se o sistema de *forward chaining*:

*%Extensao do predicado cs\_historico\_utente\_f: CS, IdU, S -> {V,F}*

*% CS pode ser 1 para consultar os sintomas*

*% CS pode ser 2 para consultar os diagnosticos*

*% CS pode ser 3 para consultar os tratamentos*

*% CS pode ser 4 para consultar os sintomas, diagnosticos e tratamentos*

*cs\_historico\_utente\_f(CS, IdU, S):- CS=1, solucoes((IdU, N), utente(IdU, N, \_, \_, \_), L1),*

*solucoes(D, (facto(IdU,D)::'tem sintomas'), L2), concatenar(L1,L2,S).*

*cs\_historico\_utente\_f(CS, IdU, S):- CS=2, solucoes((IdU, N), utente(IdU, N, \_, \_, \_), L1),*

*solucoes(D, rpf(IdU, D, G, E, T), L2), concatenar(L1,L2,S).*

*cs\_historico\_utente\_f(CS, IdU, S):- CS=3, solucoes((IdU, N), utente(IdU, N, \_, \_, \_), L1),*

*solucoes(T, rpf(IdU, D, G, E, T), L2), concatenar(L1,L2,S).*

*cs\_historico\_utente\_f(CS, IdU, S):- CS=4, solucoes((IdU, N), utente(IdU, N, \_, \_, \_), L1),*

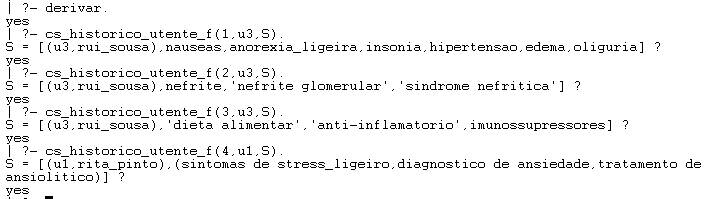
*solucoes(('sintomas' de D1), (facto(IdU,D1)::'tem sintomas', rpf(IdU, D1, \_, \_, \_)), L2),*

*solucoes(('diagnostico' de D2), (facto(IdU,D1)::'tem sintomas', rpf(IdU, D2, \_, \_, \_)), L3), sem\_repetidos(L3,R3), solucoes(('tratamento' de T), (facto(IdU,D1)::' tem sintomas',rpf(IdU,D2,G,E, T)), L4), sem\_repetidos(L4,R4),*

*concatenar(L1,L2,L5), concatenar(L5,R3,L6), concatenar(L6,R4,S).*

***Justificação:*** *As funcionalidades aplicadas através do sistema forward chaining foram construídas de um modo semelhante ao sistema anterior.*

***Simulação Foward Chaining:***

Nota: previamente utilizou-se o “derivar.”.

### 3.4.2. Consultar Utentes que apresentam um determinado Diagnóstico

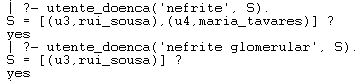
Relativamente à funcionalidade de consultar os utentes que sofrem de uma determinada doença foram definidas as seguintes instruções (*backward chaining*):

*%Extensao do predicado utente\_doenca: D, S -> {V,F}*

*utente\_doenca(D, S):-solucoes((IdU, N), utente(IdU, N, \_, \_, \_), L1),*

*solucoes(D, rp(IdU, D, \_, \_, \_), L2), concatenar(L1,L2,S).*

***Justificação:*** *Nesta funcionalidade pesquisa-se o ID e o nome do utente que, em simultâneo, noutra pesquisa, verifique a existência desse diagnóstico. Cada uma das pesquisas devolve uma lista que é concatenada para dar a solução final. Salienta-se que a opção de se executarem duas pesquisas se deveu a evitar a repetição do ID e do nome do utente.*

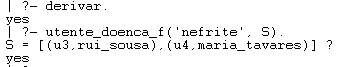
***Simulação:***

No caso do *forward chaining*:

*%Extensao do predicado utente\_doenca\_f: D, S -> {V,F}*

*utente\_doenca\_f(D, S):-solucoes((IdU, N), (utente(IdU, N, \_, \_, \_),rpf(IdU, D, \_, \_, \_)), S).*

***Justificação:*** *Esta funcionalidade é semelhante à anterior mantendo-se, por isso, válida a justificação anterior.*

***Simulação:***

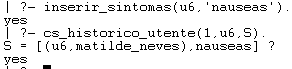
### 3.4.3. Evolução do Conhecimento: Adicionar e Remover Sintomas

*% Extensao do predicado inserir\_sintomas: ID, D -> {V,F}*

*inserir\_sintomas(ID, D):-*

*solucoes(Invariante, +(facto(ID, D) :: 'tem sintomas') ::: Invariante, S), insere(ID, D), teste(S).*

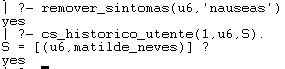
***Justificação:*** *Esta funcionalidade recorre à funcionalidade auxiliar insere, que é uma adaptação da funcionalidade inserir do Trabalho prático 1, adaptando-se para o par (ID, D).*

***Simulação:***

*% Extensao do predicado remove\_sintomas: Facto,Sintoma -> {V,F}*

*remover\_sintomas(ID, D):- (facto(ID, D)) :: 'tem sintomas', remove(ID, D).*

***Justificação:*** *Esta funcionalidade recorre à funcionalidade auxiliar remove que é uma adaptação da funcionalidade remover do Trabalho Prático 1, adaptando-se para o par (ID, D).*

***Simulação:***

## 3.5. Predicados Auxiliares

A construção dos diversos predicados recorre, para a sua implementação, a predicados mais simples e que auxiliam na implementação de predicados mais complexos. Assim, foram criados os seguintes predicados auxiliares:

* **Não,** que devolve o valor oposto de uma afirmação;
* **Comprimento,** que calcula o número total de elementos de uma lista;
* **Concatenar,** junta listas e devolve a lista total;
* **Soluções,** que devolve os valores que respeitam as regras de uma pesquisa efetuada;
* **Pertence,** que verifica se um elemento pertence a uma lista;
* **Sem\_repetidos,** que elimina os elementos repetidos de uma lista;
* **Insere,** que permite inserir novos sintomas associados a um utente;
* **Remove,** que apaga sintomas existente em um utente;
* **Teste,** que define a forma de se efetuarem os testes aos invariantes;

## 3.6. Invariantes

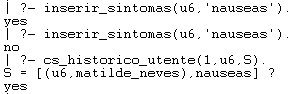
No presente trabalho prático incluiram-se invariantes que auxiliem na manutenção da coerência dos dados tendo em conta os sistemas de *backward* e *forward*. Assim, e como foi permitida a evolução do conhecimento, ao nível da introdução de novos sintomas ou de apagar sintomas existentes, concebeu-se o seguinte invariante, que impede a inserção de sintomas repetidos num mesmo utente:

*+(facto(ID, D) :: C1) ::: (solucoes((ID, D), (facto(ID, D) :: C2), X), comprimento(X, H), H==1).*

***Justificação:*** *No facto(ID, D) considera-se o par (ID, D) e, através do predicados “soluções”, encontram-se os registos que, na lista de todos os factos, tenham o mesmo ID (utente) e D (sintoma), devolvendo-os na lista H. Simultaneamente impõe-se que H tenha comprimento 1, apenas se permitindo, deste modo, a inserção de pares (ID, D) únicos. O motivo de o comprimento ser igual a 1 prende-se com o facto de o novo facto ter sido temporariamente acrescentado à base de conhecimento e, por isso, aparecer na lista S.*

***Simulação:***

**X**



**✓**

**✓**

# **4. Conclusão**

Com a realização deste trabalho foi possível testar sistemas de inferência que, de uma forma automática, lidem com questões associadas a relações do tipo SE … ENTÃO. Foi constatado que é possível desenvolver um programa que considere os dados iniciais e, através de relações de precedência, atinja as conclusões pretendidas. No caso em estudo, dado um conjunto de sintomas, foi possível comprovar-se o diagnóstico e o tratamento aplicáveis. Além disso, comprovou-se que se pode aceder ao histórico de doenças, bastando, para tal, declararem-se os sintomas e as relações de precedência. Dito de outro modo, verificou-se que este sistema não necessita de declarar todo o conhecimento para se aceder a ele. Por exemplo, não foi declarado que o utente u3 padecia de síndrome nefrítica, mas o software deduziu esse facto.

Conclui-se assim, que este tipo de linguagem de programação estabelece relações que poderão diminuir os registos em memória e, contribuir como instrumentos de apoio à decisão.

Para um trabalho futuro poderiam ser analisadas cláusulas mais complexas com a conjugação de mais bases de conhecimento e com a inclusão de condições de conhecimento imperfeito.

Em suma, apurou-se que, com o estabelecimento de relações de precedência e um adequado uso da lógica matemática, é possível implementar uma base de conhecimento que simule, de forma adequada, situações que poderão ser utilizadas na realidade.

# **5. Bibliografia**

[1] Bratko, Ivan

“PROLOG Programming for Artificial Intelligence”

Pearson, Third Impression, 2008

[2] Al-Ajlan, Ajlan

“The Comparison between Forward and Backward Chaining”

International Journal of Machine Learning and Computing, Vol. 5, No. 2, April 2015